PROJECT REPORT

Smart Input Remote Keyboard and Mouse Control from Android Device

Presented By

Name	Roll No.	Registration No.
Sayantan Majumdar	MCS-15-09	A01-1112-0755-12
Abhisek Maiti	MCS-15-41	A01-1112-0762-12
Sudipto Bhattacharjee	MCS-15-61	A01-1112-0774-12

Under Guidance Of

Dr. Asoke Nath
Associate Professor
Department of Computer Science
St. Xavier's College (Autonomous), Kolkata

**Submitted to the Department of Computer Science,
ST. XAVIER'S COLLEGE (AUTONOMOUS), KOLKATA,
in partial fulfilment of the requirements for the award of the degree,
M.Sc. Computer Science, Under University of Calcutta.**

CERTIFICATE OF ORIGINALITY

This is to certify that the project report entitled *Smart Input* is submitted to **St. Xavier's College (Autonomous), Kolkata** in partial fulfilment of the requirement for the award of the degree of M.Sc. Computer Science, is an authentic and original work carried out under my guidance by:

Sl. no.	Name	Roll No.	Registration No.
1.	Sayantan Majumdar	MCS-15-09	A01-1112-0755-12
2.	Abhisek Maiti	MCS-15-41	A01-1112-0762-12
3.	Sudipto Bhattacharjee	MCS-15-61	A01-1112-0774-12

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this College or to any other University/Institute for the fulfilment of the requirements of any course of study.

Dr. Asoke Nath (Project Guide)

Date: _____

Sayantan Majumdar (Roll : MCS-15-09)

Abhisek Maiti (Roll : MCS-15-41)

Sudipto Bhattacharjee (Roll : MCS-15-61)

Date: _____

ACKNOWLEDGEMENT

We would like to express our gratitude to our project supervisor Dr. Asoke Nath from St. Xavier's College (Autonomous), Kolkata, for his support, inspiration, guidance and continuous help. We are immensely grateful to him for giving his valuable time and constant advice in completing this project report.

Thank You,

Sayantana Majumdar (Roll : MCS-15-09)

Abhisek Maiti (Roll : MCS-15-41)

Sudipto Bhattacharjee (Roll : MCS-15-61)

Contents

1. Introduction	5
1.1. Background	
1.2. Objectives	
1.3. Purpose and Scope	
1.3.1. Purpose	
1.3.2. Scope	

2. System Analysis	7
2.1. Identification of Need	
2.2. Preliminary Investigation	
2.3. Feasibility Study	
2.4. Project Planning and Scheduling	
2.4.1. PERT Chart	
2.4.2. Gantt Chart	
2.5. Software Requirement Specifications	
2.6. Software Engineering Paradigm Applied	
2.7. Data Models	
2.7.1. Data Flow Diagram	

3. System Design	13
3.1. Project Structure	
3.2. User Interface Design	
3.3. Test Cases	

4. Coding	21
4.1. Android	
4.1.1. Complete Code	
4.1.2. Class Documentation	
4.2. PC	
4.2.1. Complete Code	
4.2.2. Class Documentation	
4.3. Standardization of Coding	

5. Testing	364
5.1. Testing Techniques and Strategies	
5.2. Test Results	
5.2.1. Data Packet Analysis	

6. Project Cost Estimation	375
-----------------------------------	------------

7. Future Scope and Conclusion	376
---------------------------------------	------------

8. References	377
----------------------	------------

1. Introduction

Smart Input aims to be a utility which enables its users to imitate any Android smartphone device as keyboard and mouse of the PC. In order to establish this feature the devices must be connected through any LAN- wired or wireless. The app in the mobile device communicates with an application in the PC.

1.1. Background

Although there are several mobile apps for remotely controlling PC I/O devices, all of these apps are proprietary. There is no single app that can be used to control more than one input devices and provide end-to-end security at the same time. Smart Input is being developed to fill up this void. It is targeted to serve as a one-app solution for securely controlling mouse and keyboard from a mobile device. This app will cut off users' expenditure of purchasing wireless I/O devices for PCs and also ensure that the users' data will not be compromised over the network.

1.2. Objectives

1.2.1. Primary Objectives

- Provide control for mouse and keyboard from Android device.
- Provide parallelism in order to connect more than one Android device to simultaneously control mouse and keyboard.
- Secure communication between mobile device and PC.
- Combining the data reading from accelerometer and gyroscope of Android device and projecting it in 2D.
- Reducing the noise in the data reading to smoothen the mouse movement.

1.2.2. Secondary Objectives

- Provide feature for controlling PC speakers.
- Remotely rendering the screen of PC on mobile device and ultimately use it as a Remote Desktop Client (RDC).
- Extend the app for iOS and Windows.

1.3. Purpose and Scope

1.3.1. Purpose

The use of android device as an I/O device is not a new concept. In almost all the previous approaches the touchscreen of the android device has been used in order to achieve remote mouse control. However, newer smart devices are more than just a small hand held computer with a touchscreen. Almost all of these devices have numerous sensors which actually make them smart. Accelerometer and Gyroscope are the two most common sensors found in these devices. Our approach is to take advantage of these two inertial sensors in order

to sense the movement in 3D space. These 3D data is then transmitted as packets to the PC application. The task of the PC application is to project the received 3D data to the corresponding 2D data and perform mouse movement via the native PC mouse controller.

As for the remote keyboard control, the keystrokes read from the inbuilt android smartphone keyboard is sent asynchronously to the PC application which then uses the native PC keyboard controller to handle the respective key functions.

The entire packet transmission operation (for both remote mouse and keyboard control) from the Android smartphone to the PC is carried out over a secured channel so as to thwart man-in-the-middle (MITM^[1]) attacks.

1.3.2.Scope

It is trivial that we are able to remotely control the mouse by fusing accelerometer and gyroscope data in real-time. It should be noted that, 2D projection of the 3D points is sufficient for mouse control. But an application can take advantage of the 3D movement data in order to provide more immersive experience than typical mouse to the user.

A typical example would be gaming and virtual reality applications. Currently gaming controllers with inbuilt IMU (Inertial Measurement Unit^[2]) and complicated hardware are commercially available to perform the similar task but they are very expensive and are often found to be miscalibrated. With our system, a user can experience the same using his/her smartphone only. Moreover users can use their android device as a full fledged multi-input device performing all the tasks of both mouse and keyboard.

Setting up our system is meant to be easy and hasslefree. A user only needs to connect the PC and the smartphone to the same WLAN (wireless LAN). The application takes advantage of the Zeroconf (Zero Configuration Networking^[3]) where a user does not need to manually detect a specified server. Moreover, knowledge of the IP address of the server (PC) is not required in order to connect. The Android application is able to detect the available servers (in this case PCs) running this application in the local network.

2. System Analysis

2.1. Identification of Need

There are several proprietary Android mobile apps that can be used for remote keyboard and mouse control. However, these apps cannot be trusted since there is no assurance of end-to-end security. Furthermore, there are very few 3D remote mouse controlling applications available in the market and all of these are again proprietary. A person using such “untrusted” applications may be vulnerable to snooping attacks or in the worst case may compromise all of his/her sensitive data. Our project aims to provide remote keyboard control and mouse control (both 3D and 2D), and also ensure the safety of the users’ data.

2.2. Preliminary Investigation

Android mobile OS is the world’s most popular mobile operating system. It essentially uses the Java programming language which is also quite programmer- friendly when it comes to network programming. The following technologies were found to be suitable for the project during the preliminary investigation:

- **Android OS:** The app is built for Android OS, which is currently the most popular mobile operating system.
- **Java:** The Java programming language is required to build android applications. Also the PC application would be built using Java to make it portable. Furthermore, the Java programming language allows extensive use of APIs and provides well documented language constructs suitable for network programming.
- **Zeroconf:** Zero Configuration Networking ^[3] is used to detect the available servers in the network from the Android device and is implemented using *UDP (User Datagram Protocol)* sockets. In order to declare itself available, the server(s) continuously make UDP broadcasts. On receiving this broadcast, an Android device can detect the IP address of the server.
- **TCP:** *Transmission Control Protocol (TCP)* is used for the sensor and keyboard data packet transmissions. Since it is a connection-oriented protocol, it lends reliability to the communication. It does away with the problem of out-of-order delivery of the data packets which may drastically lower the efficiency and accuracy of the application.
- **Inertial Sensors:** *Accelerometer* and *Gyroscope* of the android device are used to track the movement and trace the path. Accelerometer and

gyroscope provides acceleration of the device and orientation of the device respectively in real-time.

- **Real-time Bayesian Filter:** A real-time recursive bayesian filter ^[4] (such as *Kalman filter* ^[5]) is used to filter the noisy signal provided by the sensors and fuse the different sensor data ^[6]. It also helps to estimate and remove the sensor bias and errors.
- **End to end encryption:** The entire communication channel is secured using end to end encryption that is compliant with TLS v1.2 ^[7], so that the data do not leak.
- **Key-Store and Certificate Generation:** A key-store is maintained in order to keep track of the paired devices and other authentication purposes.
- **GNU Octave:** It is used for simulation, visualization and testing purposes of the real-time 3D and 2D data.
- **Wireshark:** It is used for network traffic analysis. TCP packets are monitored for verifying the effectiveness of the end-to-end encryption algorithm.

2.3. Feasibility Study

The major challenges in the project include:

- **Android application development:** Google provides useful documentation for Android APIs. Hence the development work of the Android app is carried out following the documentation.
- **Implementing the Network Service :** The Java network programming APIs provide support for TCP/IP and UDP. The Zeroconf technology (uses UDP) and mouse, keyboard packet transmissions (using TCP/IP) is implemented using these APIs.
- **Implementing End-to-End Security :** The Java security APIs along with trusted third-party cryptographic APIs such as BouncyCastle (for PC) and SpongyCastle (for Android) is used to implement the end-to-end security scheme.
- **Real-time Data Filtering:** Although the exact working of the Kalman filter is difficult to understand, the algorithm itself is quite easy to implement. The challenge lies in feeding the filtering function with proper input data. The inertial sensors of any smartphone are inherently noisy. This has been illustrated in Fig-1.

Fig-1: Time (X) vs Acceleration (Y) Plot

2.4. Project Planning and Scheduling

2.4.1. PERT Chart

Activity	Predecessor	Time Estimates			Expected Time
		Optimistic	Normal	Pessimistic	
A) Start	---	0	0	0	0
B) S/W Requirements and Analysis	A	15	25	32	23.34
C) Theoretical Study	B	50	55	60	55.00
D) Design and Coding	C	80	95	120	98.34
E) Testing	D	30	35	40	35.00
F) Documentation	C, D	35	50	60	48.34
G) End	F	0	0	0	0

Table-1: PERT Chart

2.4.2. Gantt Chart

Fig-2: Gantt Chart

2.5. Software Requirement Specifications

2.5.1. Functional Requirements

- **Remote Mouse Control:** This includes the feature of acquiring the accelerometer and gyroscope data from the Android device at regular intervals and sending it to PC. Consequently, the mouse is controlled with respect to the received data using the native methods of the OS.
- **Remote Keyboard Control:** This includes the feature of acquiring the data from the Android keyboard service and sending it to PC. Consequently, the keyboard input is controlled using the native methods of the OS.
- **Zero Configuration Networking (Zeroconf):** This feature is used to detect devices which are available for connection by continuously sending UDP broadcasts within the network.
- **Real-time Data Filtering:** The data acquired from the accelerometer and gyroscope is inherently noisy. If these data are fused and integrated to calculate the displacement, the errors will get squared up. This feature includes minimization of this error by using filters and that too in real time to avoid delay in mouse movement.

2.5.2. Non-functional Requirements

- **Parallelism:** This feature includes the ability of the PC software to handle simultaneous data from more than one smartphones. For example, connecting two devices, one as mouse and other as keyboard.
- **Secured transmission of sensor data:** This includes the feature of encrypting the sensor data which is then transferred.
- **Key-Store and Certificate Generation:** This feature is used for entity authentication.
- **Providing user friendly UI/UX.**

2.5.3. Software Requirements

- Java Development Kit 1.8 or above.
- Android API 16 or above.

2.5.4. Hardware Requirements

- Inbuilt Android smartphone Accelerometer.
- Inbuilt Android smartphone Gyroscope.
Note: Required for only 3D mouse operations.
- Recommended PC H/W requirements for JDK 1.8 or above.
- Recommended smartphone H/W requirements for Android API 16 or above.

2.6. Software Engineering Paradigm Applied

The evolutionary prototyping model is used as the software engineering paradigm. Initially a minimalistic prototype is developed that provides mouse click and scroll functions, basic text input via Android keyboard. It also implements the network service module. In the next prototype, end-to-end encryption is implemented. Function keys like Ctrl, Alt, etc is also supported by this prototype. In the final prototype the filter module is implemented. This prototype provides all the functional and non-functional requirements.

Fig-3: Evolutionary Prototyping

2.7. Data Model

2.7.1. Data Flow Diagram

Fig-4: Data Flow Diagram

3. System Design

3.1. Project Structure

3.1.1. Modularization details

There are five primary modules in the project :

- a. Network module:* Initially, a UDP broadcast is made by the PC (server) where its public key would be stored in the data part of the datagram. The Android app (client) receives this broadcast, obtain the IP address of the server and attempt to make a connection to that server. Here, a pairing key is used to authenticate the android app user. The data packets for the keyboard and mouse operations are sent to the PC using TCP.
- b. Security module:* The server generates a random pairing key per client request. The SHA-256 ^[8] hash of this pairing key acts as the initialization vector (IV) for the 256-bit AES ^[9] encryption module. Here ECDH ^[10] is used as the key agreement algorithm. The server has its ECC ^[11] public and private keys stored in its key store. Similarly the android app has its ECC public and private keys stored in its key store. Brainpoolp256r1 ^[12] is used as the elliptic curve for the key pair generation. Each time packets are sent from the server, they get encrypted using a symmetric key (generated from the server's private key and the client's public key). The client decrypts these data packets using the same secret key (generated from the client's private key and the server's public key).
- c. Filter module:* The accelerometer and gyroscope data from the android sensors are fused to minimize the error occurred due to the sensor noise. The Kalman Filter probabilistically estimates the errors and removes it from the data.
 - *State Prediction:* The current state is predicted from the previous state using a prediction model. The prediction model uses simple kinematics formulas. The Kalman filter model assumes the true state at time k is evolved from the state at $(k - 1)$ according to,

$$\rightarrow x_k = F_k x_{k-1} + B_k u_k + w_k \quad (1)$$

where,

- F_k is the state transition model which is applied to the previous state x_{k-1} .
- B_k is the control-input model which is applied to the control vector u_k
- w_k is the process noise which is assumed to be drawn from a zero mean multivariate normal distribution with covariance Q_k . R_k is the noise covariance matrix. At time k

an observation (or measurement) z_k of the true state x_k is made according to

$$\rightarrow z_k = H_k x_k + v_k \quad (2)$$

where,

- H_k = observation matrix, and
- v_k = observation noise.

$$\rightarrow P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \quad (3)$$

where,

- $P_{k|k-1}$ = predicted (*a priori*) state estimate

$$\rightarrow y_k = z_k - H_k x_{k|k-1} \quad (4)$$

where,

- y_k = measurement residual

$$\rightarrow S_k = H_k P_{k|k-1} H_k^T + R_k \quad (5)$$

where,

- S_k = residual covariance

$$\rightarrow K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (6)$$

where,

- K_k = optimal Kalman Gain

$$\rightarrow x_{k|k} = x_{k|k-1} + K_k y_k \quad (7)$$

where,

- $x_{k|k}$ = *a posteriori* state estimate at time k given observations up to and including at time k

$$\rightarrow P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (8)$$

where,

- $P_{k|k}$ = *a posteriori* error covariance matrix

- **Measurement:** The measured state consists of the current values obtained from the sensor.
- **Fusion:** The measured state is corrected using the values obtained from another sensor. The corrected state is known as the fused state.
- **Feedback:** The predicted state and fused state are probabilistically compared and the current feedback is generated. The fused state is added with the feedback from previous state to obtain the actual filtered state.
- **Quaternions:** Quaternion ^[13] is a convenient way of representing 3D rotation. The definition of an orientation by a quaternion is coordinate system independent and isotropic. Quaternions are generally represented in the form:

$$q = a + b\hat{i} + c\hat{j} + d\hat{k} \quad (9)$$

Compared to other 3D rotation representations, quaternions have better efficiency and smaller memory footprint. As a consequence unit normalisation of floating point quaternions suffers from fewer rounding defects than other representations. Also it's extremely easy to interpolate between two quaternions using SLERP (Spherical Linear Interpolation) ^[13], which is useful for smoothing the movement. Converting a quaternion to other representation is quite easy. For these reasons, in this project, quaternions are used for representing 3D rotation.

- **Advantages over Moving Average Filter:** For the simulation purpose, random noise is used in place of both process noise and observation noise. Here, process noise coefficient is taken as 0.5 and observation noise coefficient taken as 10. The simulation result is shown in Fig-5.

Fig-5: Curve Smoothing using Kalman Filter

For contrast, the simulation result of simple moving average filter with window size five is given below.

Fig- 6: Curve Smoothing using Moving Average Filter

It is clear that Kalman filter is way more accurate than the moving average filter and the accuracy of prediction improves with time. This model is very simple and the time (t) vs position ($\vec{p}$) graph is basically generic second degree polynomial curve.

Fig-7: Filtering noisy Sine Curve using Kalman Filter

In the second simulation a more complex model was simulated. This model depicts the sine curve. The simulation result is given in Fig-7.

d. Mouse module: This module includes the following features :

- **Movement:** The 3D mouse movement is implemented using the data obtained from the inertial sensors (accelerometer and gyroscope). The filter module removes the inherent noise of these sensors and feeds the corrected data to the encryption module. The 2D mouse movement is implemented using the relative touch positions on the touchpad. The relative x and y coordinates are likewise encrypted. The encrypted data packets are then sent to the

PC which decrypts these data and feeds these to the native mouse control methods of the OS.

- **Click and Scroll Functions:** There are three buttons that emulate left, right and middle click functions of a mouse. Also there are two more buttons that emulate the up-scroll and down-scroll functions. These button click action data are encrypted and sent to the server which decrypts these and feeds these data to the native mouse methods of the OS.
- e. **Keyboard module:** The Android keyboard service is used to receive key press data which are then encrypted and sent to the server. The server decrypts these data and feeds these to the native keyboard methods of the OS. The keyboard module provides the following features:
- **Predictive Text Input:** Android provides prediction for future words on a panel above the keyboard. Users can tap on these words and the same gets typed on the PC.
 - **Auto-correct:** Similar to word prediction, Android provides feature to automatically correct and complete words on pressing the spacebar key. This feature also gets emulated to the PC.
 - **Support for special keys:** This app provides a layout of special keys (e.g. Alt, Ctrl, F-keys etc) which are not provided in the layout of the Android Keyboard Service.
 - **Hiding Visible Text:** This app also allows the characters that are visible on the app while typing to be hidden. This prohibits any bystander to view sensitive text while it is typed.

3.1.2. Conceptual Model

Fig-8: Conceptual Model

3.2. User Interface Design

Fig-9: Connection Layout

Fig-10: Pairing Key Layout

Fig-11: 2D Mouse Layout

Fig-12: 3D Mouse Layout

Fig-13: Keyboard Layout 1

Fig-14: Keyboard Layout 2

Fig-15: Navigation UI

Fig-16: Settings Layout

Fig-17: PC GUI Prototype

Fig-18: PC Pairing Key Dialog

Fig-19: Cube Demo

4. Coding

4.1. Android

4.1.1 Complete Coding

Directory Structure

[detail level 1 2 3 4 5 6 7 8 9]

```
▼ app
  ▼ src
    ▼ main
      ▼ assets
        about.html
      ▼ java
        ▼ project
          ▼ android
            ▼ net
              BroadcastReceiverThread.java
              Client.java
              ClientInfo.java
              ClientIOThread.java
              ConnectionTask.java
              DataWrapper.java
              KeyboardThread.java
              NetworkService.java
              ServerInfo.java
            ▼ security
              EKEProvider.java
              KeyStoreManager.java
            ▼ sensor
              ▼ orientation
                KalmanFilterProvider.java
                OrientationProvider.java
              ▼ representation
                Matrix.java
                MatrixF4x4.java
                Quaternion.java
                Vector3f.java
```

- Vector4f.java**
 - AboutFragment.java**
 - ConnectionFragment.java**
 - CustomAdapter.java**
 - KeyboardFragment.java**
 - MainActivity.java**
 - MouseFragment.java**
 - SettingsActivity.java**
 - SettingsFragment.java**
 - TouchpadFragment.java**
- ▼ **res**
 - ▼ **layout**
 - activity_main.xml**
 - activity_settings.xml**
 - app_bar_main.xml**
 - content_main.xml**
 - content_settings.xml**
 - fragment_about.xml**
 - fragment_connect.xml**
 - fragment_keyboard.xml**
 - fragment_mouse.xml**
 - fragment_touchpad.xml**
 - local_devices.xml**
 - nav_header_main.xml**
 - ▼ **menu**
 - activity_main_drawer.xml**
 - main.xml**
 - ▼ **values**
 - array.xml**
 - colors.xml**
 - dimens.xml**
 - drawables.xml**
 - strings.xml**
 - styles.xml**
 - ▼ **xml**
 - preferences.xml**
- AndroidManifest.xml**

about.html

```
1 <!DOCTYPE HTML>
2 <html>
3 <head>
4   <title>About : Remouse</title>
5   <meta charset="utf-8" />
6   <meta name="viewport" content="width=device-width, initial-scale=1" />
7   <link rel="stylesheet" href="assets/css/main.css" />
8 </head>
9 <body>
10
11 <!-- Header -->
12 <div id="header">
13   
14   <h1>Remouse</h1>
15   <p>A secured remote input app based on android sensors.<br/>
16     Currently mouse and keyboard works.<br/>
17 </p>
18 </div>
19
20 <!-- Main -->
21 <div id="main">
22
23   <header class="major container 75%">
24     <b>Meet the Developers</b>
25   </header>
26
27   <div class="box alt container">
28     <section class="feature left">
29       <div class="content" style="text-align: center;">
30         <br/>Xtocat<br
31 /></img>
32         <h3>Sayantan</h3>
33         <h1>MCS-15-09</h1>
34         <p>
35           <br/> Sayantan is currently completing his masters at St. Xavier's
36           College Kolkata. His main areas of interest are Network Security and Mobile computing.
37         </p>
38         <ul class="icons">
39           <li><a href="https://www.linkedin.com/in/sayantanmajumdar" class="icon
40 fa-linkedin"><span class="label">LinkedIn</span></a></li>
41         </ul>
42       </div>
43     </section>
44     <section class="feature right">
45       <div class="content" style="text-align: center;">
46         <br/>Ironcat<br
47 /></img>
48         <h3>Abhisek</h3>
49         <h1>MCS-15-41</h1>
50         <p>
51           Abhisek is currently completing his masters at St. Xavier's College
52           Kolkata. His main areas of interest are Wireless Sensor Network and Signal Processing.
53         </p>
54         <ul class="icons">
55           <li><a href="https://www.linkedin.com/in/abhisek-maiti" class="icon
56 fa-linkedin"><span class="label">LinkedIn</span></a></li>
57         </ul>
58       </div>
59     </section>
60     <section class="feature left">
61       <div class="content" style="text-align: center;">
62         
63         <br/>Spidertocat<br/></img>
64         <h3>Sudipto</h3>
65         <h1>MCS-15-61</h1>
66         <p>Sudipto is currently completing his masters at St. Xavier's College
67           Kolkata. His main areas of interest are Network Security and Mobile Computing.
68         </p>
69         <ul class="icons">
70           <li><a href="#" class="icon fa-linkedin"><span
71 class="label">LinkedIn</span></a></li>
72         </ul>
73       </div>
74     </section>
75   </div>
76   <br/><br/><br/>
77 </div>
78
79 <!-- Footer -->
80 <div id="footer">
81   <div class="container 75%">
82     <header class="major last">
83       <h2>Found a Bug?</h2>
84     </header>
85     <p>
86       Email us. We will try to resolve issues as soon as possible.
87     </p>
88   </div>
89 </div>
```

```
81         </p>
82         <ul>
83             <li><a href="mailto:monti.majumdar@gmail.com?Subject=[Remouse]%20Bug%20Report"
class="icon fa-envelope"><span class="label">Issues</span></a></li>
84         </ul>
85         <ul class="copyright">
86             <li>Design: Abhisek, Sayantan, Sudipto</li>
87         </ul>
88     </div>
89 </div>
90 </div>
91 </body>
92 </html>
```

BroadcastReceiverThread.java

```
1 package project.android.net;
2
3 import android.app.Activity;
4 import android.net.wifi.WifiManager;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Toast;
8
9 import com.google.gson.Gson;
10
11 import java.io.IOException;
12 import java.net.DatagramPacket;
13 import java.net.DatagramSocket;
14 import java.util.ArrayList;;
15
16 import project.android.ConnectionFragment;
17 import project.android.MainActivity;
18
19 import static project.android.ConnectionFragment.sConnectionAlive;
20
21 /**
22  * Implementation of the <code>java.lang.Runnable</code> for the thread
23  * responsible for receiving broadcast from the server.
24  *
25  * <p>
26  * This class receives the UDP packets broadcasted by the server and updates
27  * the <code>ListView</code> in the {@link project.android.ConnectionFragment}
28  * accordingly. This thread is started by the
29  * {@link ConnectionFragment#onViewCreated(View, Bundle)} method and it runs
30  * until the app is closed.
31  * </p>
32  *
33  * @see project.android.ConnectionFragment
34  * @see project.android.net.ServerInfo
35  * @see project.android.ConnectionFragment#onViewCreated(View, Bundle)
36  */
37 public class BroadcastReceiverThread implements Runnable {
38
39     private Activity mActivity;
40     private WifiManager.MulticastLock mLock;
41     private ArrayList<String> mServerAddress;
42     private DatagramSocket mDatagramSocket;
43     private ConnectionFragment mConnectionFragment;
44
45     private static final int UDP_PORT = 1235;
46
47     /**
48      * Constructor.
49      *
50      * Initializes this <code>BroadcastReceiverThread</code>.
51      *
52      * @param activity The current <code>android.app.Activity</code> object.
53      * @param lock an object of <code>WifiManager.MulticastLock</code> class
54      * of the Android API.
55      */
56     public BroadcastReceiverThread(Activity activity, WifiManager.MulticastLock lock) {
57         mActivity = activity;
58         mLock = lock;
59         mDatagramSocket = null;
60         mServerAddress = new ArrayList<>();
61         mConnectionFragment = (ConnectionFragment) MainActivity.getConnectionFragment();
62     }
63
64     /**
65      * Performs the broadcast receiving operation.
66      *
67      * <ul>
68      * <li>
69      * Checks whether a stop signal has been broadcast by the server. If so,
70      * removes the {@link ServerInfo} object from the <code>ListView</code> in
71      * the {@link project.android.ConnectionFragment}.
72      * </li>
73      * <li>
74      * Constructs a {@link ServerInfo} object from the data obtained from the
75      * <code>DatagramPacket</code> using <code>com.google.gson</code> API and
76      * updates the <code>ListView</code> in the
77      * {@link project.android.ConnectionFragment} accordingly.
78      * </li>
79      * </ul>
80      *
81      * @see project.android.ConnectionFragment
82      */
83     @Override
84     public void run() {
85         try {
86             mLock.acquire();
87             if (mDatagramSocket == null) {
88                 mDatagramSocket = new DatagramSocket(UDP_PORT);
89                 mDatagramSocket.setBroadcast(true);
```

```

90     }
91     } catch (IOException e) { e.printStackTrace(); }
92
93     while (mDatagramSocket != null) {
94         DatagramPacket datagramPacket = null;
95         try {
96             if (!mLock.isHeld()) mLock.acquire();
97             datagramPacket = new DatagramPacket(new byte[ServerInfo.SERVER_INFO_LENGTH],
98                 ServerInfo.SERVER_INFO_LENGTH);
99             mDatagramSocket.receive(datagramPacket);
100             mLock.release();
101         } catch (IOException ignored) {}
102         String gsonString = new String(datagramPacket.getData());
103         gsonString = gsonString.substring(0, gsonString.indexOf("}") + 1);
104         ServerInfo serverInfo = new Gson().fromJson(gsonString, ServerInfo.class);
105         String serverAddress = datagramPacket.getAddress().toString().substring(1);
106         serverInfo.setServerAddress(serverAddress);
107
108         checkAvailableDevices(serverInfo, serverAddress);
109     }
110 }
111 private void checkAvailableDevices(final ServerInfo serverInfo, final String
serverAddress) {
112
113     if (serverInfo.getStopFlag()) {
114         mServerAddress.remove(serverAddress);
115         mActivity.runOnUiThread(new Runnable() {
116             @Override
117             public void run() {
118                 Toast.makeText(mActivity, "Server " + serverInfo.getAddress() + "
119                 stopped!",
120                     Toast.LENGTH_SHORT).show();
121                 mConnectionFragment.removeItem(serverInfo);
122                 mConnectionFragment.dismissAlertDialog();
123             }
124         });
125     } else {
126         if (mServerAddress.isEmpty() || !mServerAddress.contains(serverAddress)) {
127             mServerAddress.add(serverAddress);
128             mActivity.runOnUiThread(new Runnable() {
129                 @Override
130                 public void run() {
131                     mConnectionFragment.addItem(serverInfo);
132                     sConnectionAlive.put(serverAddress, false);
133                 }
134             });
135         }
136     }
137 }

```

Client.java

```
1 package project.android.net;
2
3 import com.google.gson.Gson;
4
5 import java.io.BufferedReader;
6 import java.io.IOException;
7 import java.io.InputStreamReader;
8 import java.io.PrintWriter;
9 import java.net.Socket;
10
11 import project.android.security.EKEProvider;
12 import project.android.sensor.representation.Quaternion;
13
14 import static project.android.MainActivity.DEVICE_NAME;
15 import static project.android.MainActivity.sPublicKey;
16 import static project.android.SettingsActivity.sMouseSensitivity2d;
17 import static project.android.SettingsActivity.sMouseSensitivity3d;
18
19 /**
20  * Class representing the client.
21  *
22  * <p>
23  * The constructor {@link #Client(String)} is used to set up a TCP connection
24  * with the respective server. The sensor data obtained through
25  * {@link project.android.KeyboardFragment}, {@link project.android.MouseFragment}
26  * and {@link project.android.TouchpadFragment} are wrapped using {@link DataWrapper}
27  * which internally uses the <code>com.google.gson.Gson</code> API. The wrapped <i>JSON
28  * string</i> is encrypted using the
29  * {@link project.android.security.EKEProvider#encryptString(String)} method and
30  * transmitted over the network.
31  * </p>
32  *
33  * @see project.android.net.DataWrapper
34  * @see project.android.security.EKEProvider
35  */
36 public class Client {
37     private DataWrapper mDataWrapper;
38
39     private static Socket sSocket;
40     private static PrintWriter sOut;
41     private static final int TCP_PORT = 1234;
42
43     private EKEProvider mEKEProvider;
44
45     /**
46      * Constructor.
47      *
48      * Initializes this <code>Client</code>.
49      *
50      * @param address IPv4 address of the server.
51      * @throws IOException
52      */
53     public Client(String address) throws IOException {
54         sSocket = new Socket(address, TCP_PORT);
55         sOut = new PrintWriter(sSocket.getOutputStream(), true);
56         String gsonStr = new Gson().toJson(new ClientInfo(DEVICE_NAME, sPublicKey));
57         sOut.println(gsonStr);
58     }
59
60     /**
61      * Sets the <code>EKEProvider</code> object.
62      *
63      * @param ekeProvider the {@link project.android.security.EKEProvider} object.
64      * @see project.android.security.EKEProvider
65      */
66     void setEKEProvider(EKEProvider ekeProvider) { mEKEProvider = ekeProvider; }
67
68     /**
69      * Sends the pairing key provided by the app user to the server.
70      *
71      * @param pairingKey the pairing key provided by the app user.
72      */
73     void sendPairingKey(String pairingKey) { sOut.println(pairingKey); }
74
75     /**
76      * Returns <code>BufferedReader</code> of the client <code>Socket</code>.
77      *
78      * @return the <code>BufferedReader</code> object.
79      * @throws IOException
80      */
81     BufferedReader getSocketReader() throws IOException {
82         return new BufferedReader(new InputStreamReader(sSocket.getInputStream()));
83     }
84
85     /**
86      * Sends mouse button or keyboard data.
87      *
88      * @param operationType type of the operation.
89      */
90 }
```

```

90     * @param data data to be sent.
91     * @see project.android.net.DataWrapper#DataWrapper(String, String)
92     */
93     public void sendData(String operationType, String data) {
94         mDataWrapper = new DataWrapper(operationType, data);
95         data = DataWrapper.getGsonString(mDataWrapper);
96         data = mEKEProvider.encryptString(data);
97         sOut.println(data);
98     }
99
100     /**
101     * Sends special key data.
102     *
103     * @param data special key data.
104     * @see project.android.net.DataWrapper#DataWrapper(String)
105     */
106     public void sendData(String data) {
107         mDataWrapper = new DataWrapper(data);
108         data = DataWrapper.getGsonString(mDataWrapper);
109         data = mEKEProvider.encryptString(data);
110         sOut.println(data);
111     }
112
113     /**
114     * Sends 2D mouse movement data.
115     *
116     * @param x relative x coordinate.
117     * @param y relative y coordinate.
118     * @see project.android.net.DataWrapper#DataWrapper(int, int, float)
119     */
120     public void sendData(int x, int y) {
121         mDataWrapper = new DataWrapper(x, y, sMouseSensitivity2d);
122         String data = new Gson().toJson(mDataWrapper);
123         sOut.println(mEKEProvider.encryptString(data));
124     }
125
126     /**
127     * Sends 3D mouse movement data.
128     *
129     * @param quaternion a {@link project.android.sensor.representation.Quaternion}
130     *                    object.
131     * @param isInitQuat <code>true</code>, if initial <code>Quaternion</code>,<br/>
132     *                    <code>false</code>, otherwise.
133     * @see project.android.sensor.representation.Quaternion
134     * @see project.android.net.DataWrapper#DataWrapper(Quaternion, boolean, float)
135     */
136     public void sendData(Quaternion quaternion, boolean isInitQuat) {
137         mDataWrapper = new DataWrapper(quaternion, isInitQuat, sMouseSensitivity3d);
138         String data = new Gson().toJson(mDataWrapper);
139         sOut.println(mEKEProvider.encryptString(data));
140     }
141
142     /**
143     * Sends a stop signal if the user disconnects the server.
144     * @param makeSecured <code>true</code>, for valid pairing key, <br/>
145     *                    <code>false</code>, if pairing key alert dialog was cancelled.
146     */
147     public void sendStopSignal(boolean makeSecured) {
148         if(makeSecured && mEKEProvider != null) {
149             sendData("Stop","");
150         } else sOut.println("Cancelled");
151     }
152
153     /**
154     * Closes the client socket.
155     * @throws IOException
156     */
157     public void close() throws IOException {
158         if(sSocket != null) sSocket.close();
159     }
160 }

```


ClientInfo.java

```
1 package project.android.net;
2
3 import java.net.Socket;
4
5 /**
6  * Encapsulation of the data sent to the server at the time of
7  * establishing TCP connection.
8  *
9  * <p>
10 * The data contains :
11 * <ul>
12 * <li>Public key of the client</li>
13 * <li>Canonical hostname of the client</li>
14 * <li>Client <code>Socket</code></li>
15 * </ul>
16 * </p>
17 *
18 * @see java.net.Socket
19 * @see project.android.net.Client
20 */
21
22 public class ClientInfo {
23     private String mClientInfo;
24     private byte[] mPublicKey;
25     private Socket mClientSocket;
26
27     /**
28      * Constructor.
29      *
30      * <p>
31      * Initializes this <code>ClientInfo</code> object. The stored
32      * object of {@link java.net.Socket} (client <code>Socket</code>)
33      * is not initialized. It can be set by the {@link #setSocket(Socket)}
34      * method.
35      * </p>
36      *
37      * @param clientInfo canonical hostname of the client.
38      * @param publicKey public key of the client.
39      * @see java.net.Socket
40      * @see #setSocket(Socket)
41      */
42     public ClientInfo(String clientInfo, byte[] publicKey) {
43         mClientInfo = clientInfo;
44         mPublicKey = publicKey;
45     }
46
47     /**
48      * Adds {@link java.net.Socket} of the client.
49      *
50      * @param socket the client {@link java.net.Socket} object to be
51      * stored.
52      */
53     public void setSocket(Socket socket) { mClientSocket = socket; }
54
55     /**
56      * Returns the client <code>Socket</code>.
57      *
58      * @return the client {@link java.net.Socket}.
59      */
60     public Socket getSocket() { return mClientSocket; }
61
62     /**
63      * Returns the canonical hostname of the client.
64      *
65      * @return canonical hostname of the client.
66      */
67     public String getClientInfo() { return mClientInfo; }
68
69     /**
70      * Returns the public key of the client.
71      *
72      * @return public key of the client.
73      */
74     public byte[] getPublicKey() { return mPublicKey; }
75 }
```

ClientIOThread.java

```
1 package project.android.net;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.widget.Toast;
6
7 import java.io.BufferedReader;
8 import java.io.IOException;
9
10 import project.android.ConnectionFragment;
11 import project.android.MainActivity;
12 import project.android.security.EKEProvider;
13
14 import static project.android.ConnectionFragment.sConnectionAlive;
15 import static project.android.ConnectionFragment.sSelectedServer;
16 import static project.android.ConnectionFragment.sSecuredClient;
17
18 /**
19  * Implementation of the <code>java.lang.Runnable</code> for the thread
20  * responsible for client-server interaction.
21  *
22  * <p>
23  * This thread is started by the {@link ConnectionTask#onPostExecute(EKEProvider)}
24  * method.
25  * </p>
26  *
27  * @see project.android.net.ConnectionTask
28  * @see project.android.net.Client
29  * @see project.android.net.NetworkService
30  */
31 class ClientIOThread implements Runnable {
32
33     private Activity mActivity;
34     private EKEProvider mEKEProvider;
35     private ServerInfo mServerInfo;
36     private String mAddress;
37     private boolean mStopFlag;
38
39     /**
40      * Constructor.
41      * Initializes this <code>ClientIOThread</code>.
42      *
43      * @param activity the current <code>Activity</code>.
44      * @param serverInfo {@link ServerInfo} object.
45      * @param ekeProvider {@link project.android.security.EKEProvider} object.
46      * @throws IOException
47      */
48     ClientIOThread(Activity activity, final ServerInfo serverInfo, EKEProvider ekeProvider)
49     throws IOException {
50         mActivity = activity;
51         mServerInfo = serverInfo;
52         mAddress = serverInfo.getAddress();
53         mEKEProvider = ekeProvider;
54         sSecuredClient.setEKEProvider(ekeProvider);
55         new Thread(new Runnable() {
56             @Override
57             public void run() {
58                 sSecuredClient.sendPairingKey(mEKEProvider.encryptString(serverInfo.getPairingKey()));
59             }
60         }).start();
61
62         /**
63          * Performs operations for reading data from a server.
64          * <ul>
65          * <li>
66          * If it reads the stop signal sent from server, it disables the
67          * respective server
68          * </li>
69          * <li>Starts a {@link NetworkService} upon successful server connection</li>
70          * </ul>
71          * @see project.android.ConnectionFragment
72          */
73         @Override
74         public void run() {
75             try {
76                 while (!mStopFlag) {
77                     recieveData();
78                 }
79                 mActivity.runOnUiThread(new Runnable() {
80                     @Override
81                     public void run() {
82                         sSelectedServer.remove(mServerInfo);
83                         ConnectionFragment connectionFragment = (ConnectionFragment)
84                         MainActivity.getConnectionFragment();
85                         try {
86                             connectionFragment.resetIcon(mServerInfo);
87                         } catch (RuntimeException ignored) {}
88                     }
89                 });
90             }
91         }
92     }
93 }
```

```

87 |         Toast.makeText(mActivity, "Server " + mAddress + " disconnected! ",
88 | Toast.LENGTH_SHORT).show();
89 |     });
90 |     sSecuredClient.close();
91 | } catch (IOException ignored) {}
92 | }
93 |
94 | private void recieveData() throws IOException {
95 |     BufferedReader in = sSecuredClient.getSocketReader();
96 |     if(!in.ready()) return;
97 |     String s = mEKEProvider.decryptString(in.readLine());
98 |     if(s != null) {
99 |         if (s.equals("Stop")) {
100 |             sConnectionAlive.put(mAddress, false);
101 |             mStopFlag = true;
102 |             return;
103 |         }
104 |         sConnectionAlive.put(mAddress, s.equals("1"));
105 |     }
106 |     if (s == null || !sConnectionAlive.get(mAddress)) {
107 |         displayError(mActivity);
108 |     } else {
109 |         Intent intent = new Intent(mActivity, NetworkService.class);
110 |         intent.putExtra("Server", mAddress);
111 |         mActivity.startService(intent);
112 |     }
113 | }
114 |
115 | private void displayError(final Activity activity) throws IOException {
116 |     activity.runOnUiThread(new Runnable() {
117 |         @Override
118 |         public void run() {
119 |             Toast.makeText(activity, "Incorrect pin! Try connecting again",
120 | Toast.LENGTH_LONG).show();
121 |             sSelectedServer.remove(mServerInfo);
122 |         }
123 |     });
124 |     sSecuredClient.close();
125 | }
126 | }

```

ConnectionTask.java

```
1 package project.android.net;
2
3 import android.app.ProgressDialog;
4 import android.os.AsyncTask;
5
6 import java.io.IOException;
7
8 import project.android.ConnectionFragment;
9 import project.android.MainActivity;
10 import project.android.security.EKEProvider;
11
12 /**
13  * Extends the AsyncTask class of the Android API.
14  *
15  * <p>
16  * It performs the following operations :
17  * <ul>
18  * <li>
19  * Displaying a ProgressDialog.
20  * </li>
21  * <li>
22  * Constructing a {@link project.android.security.EKEProvider} object
23  * that will be used for encryption and decryption of TCP packets.
24  * </li>
25  * <li>
26  * Starting a {@link ClientIOThread}.
27  * </li>
28  * </ul>
29  * <code>AsyncTask</code> enables these operations to be performed asynchronously
30  * by defining them in the {@link #onPreExecute()},
31  * {@link #doInBackground(ServerInfo...)} and {@link #onPostExecute(EKEProvider)}
32  * methods respectively. The starting of {@link ClientIOThread} waits for the
33  * construction of the {@link project.android.security.EKEProvider} object.
34  * A ProgressDialog is displayed for this duration.
35  * </p>
36  *
37  * @see project.android.ConnectionFragment
38  * @see project.android.net.ClientIOThread
39  * @see project.android.net.ServerInfo
40  * @see project.android.security.EKEProvider
41  */
42
43 public class ConnectionTask extends AsyncTask<ServerInfo, Void, EKEProvider> {
44
45     private ProgressDialog mProgressDialog;
46     private ConnectionFragment mConnectionFragment;
47
48     /**
49      * Overrides the AsyncTask.onPreExecute() method of the Android API.
50      *
51      * Runs on the UI thread before the
52      * {@link ConnectionTask#doInBackground(ServerInfo...)} method.
53      *
54      * @see project.android.net.ConnectionTask#doInBackground(ServerInfo...)
55      * @see project.android.net.ConnectionTask#onPostExecute(EKEProvider)
56      */
57     @Override
58     protected void onPreExecute() {
59         mConnectionFragment = (ConnectionFragment) MainActivity.getConnectionFragment();
60         mProgressDialog = ProgressDialog.show(mConnectionFragment.getContext(),
61 "Connecting...", "Please wait!", true, false);
62     }
63
64     /**
65      * Overrides the AsyncTask.doInBackground(Params...) method of the
66      * Android API.
67      *
68      * Performs a computation on a background thread.
69      *
70      * @param serverInfos {@link ServerInfo} object(s).
71      * @return {@link project.android.security.EKEProvider} object.
72      * @see project.android.net.ConnectionTask#onPreExecute()
73      * @see project.android.net.ConnectionTask#onPostExecute(EKEProvider)
74      * @see project.android.net.ServerInfo
75      */
76     @Override
77     protected EKEProvider doInBackground(ServerInfo... serverInfos) {
78         return new EKEProvider(serverInfos[0].getPairingKey().getBytes(),
79 serverInfos[0].getServerPubKey());
80     }
81
82     /**
83      * Overrides the AsyncTask.onPostExecute(Result) method of the Android API.
84      *
85      * This method won't be invoked if the task was cancelled.
86      *
87      * @param ekeProvider {@link project.android.security.EKEProvider} object returned by
88      * the {@link ConnectionTask#doInBackground(ServerInfo...)} method.
89      * @see project.android.net.ConnectionTask#doInBackground(ServerInfo...)
```

```

88     * @see project.android.net.ConnectionTask#onPostExecute()
89     * @see project.android.net.ClientIOThread
90     * @see project.android.security.EKEProvider
91     */
92     @Override
93     protected void onPostExecute(EKEProvider ekeProvider) {
94         try {
95             mProgressDialog.dismiss();
96             ClientIOThread clientIOThread = new
ClientIOThread(mConnectionFragment.getActivity(), mConnectionFragment.mServerInfo,
ekeProvider);
97             new Thread(clientIOThread).start();
98         } catch (IOException ignored) {}
99     }
100 }

```

DataWrapper.java

```
1 package project.android.net;
2
3 import com.google.gson.Gson;
4
5 import project.android.sensor.representation.Quaternion;
6
7 /**
8  * Encapsulates the data sent to the server during TCP communication.
9  *
10  * <p>
11  * This data contains all the information acquired from the
12  * {@link project.android.MouseFragment}, the
13  * {@link project.android.TouchpadFragment} and the
14  * {@link project.android.KeyboardFragment}. It contains a
15  * <i>type of operation</i> field which helps the server determine
16  * the operation requested by the client and a <i>data</i> field
17  * which acts as the operand for the input operations.
18  * </p>
19  * <p>
20  * For keyboard input, the <i>type of operation</i> field contains
21  * <code>Key</code> and the <i>data</i> field contains the sequence
22  * of characters and keyboard strokes for input. In case of special
23  * keys, the <code>data</code> field contains the special key type.
24  * </p>
25  * <p>
26  * For 3-D mouse movement operation, the <i>type of operation</i>
27  * field contains <code>Mouse_Move</code> and the <i>data</i> field
28  * contains a {@link project.android.sensor.representation.Quaternion}
29  * object.
30  * </p>
31  * <p>
32  * For 2-D mouse movement operation, the <i>type of operation</i>
33  * field contains <code>Mouse_Touch</code> and the <i>data</i> field
34  * contains a pair of integers representing the displacement along the
35  * x-axis and y-axis relative to the current mouse pointer location.
36  * </p>
37  * <p>
38  * For mouse button click operation, the <i>type of operation</i>
39  * field contains <code>Mouse_Button</code> and the <i>data</i> field
40  * contains any of the following :
41  * <ul>
42  * <li><code>left</code> - representing left click.</li>
43  * <li><code>right</code> - representing right click.</li>
44  * <li><code>middle</code> - representing middle button click.</li>
45  * <li><code>upscroll</code> - representing scroll up.</li>
46  * <li><code>downscroll</code> - representing scroll down.</li>
47  * </ul>
48  * </p>
49  *
50  * @see project.android.sensor.representation.Quaternion
51  * @see project.android.MouseFragment
52  * @see project.android.TouchpadFragment
53  * @see project.android.KeyboardFragment
54  */
55 class DataWrapper {
56     private String mOperationType;
57     private String mData;
58     private Quaternion mQuaternion;
59
60     private boolean mIsInitQuat;
61     private int mMoveX;
62     private int mMoveY;
63     private float mSensitivity;
64
65     /**
66      * Constructor.
67      *
68      * Used to create a <code>DataWrapper</code> for keyboard and mouse
69      * button click operations.
70      *
71      * @param operationType the <i>type of operation</i> field.
72      * @param data the <i>data</i> field.
73      */
74     DataWrapper(String operationType, String data) {
75         mOperationType = operationType;
76         mData = data;
77     }
78
79     /**
80      * Constructor.
81      *
82      * Used to create a <code>DataWrapper</code> for special keys like Ctrl,
83      * Alt etc.
84      *
85      * @param data the <i>data</i> field.
86      */
87     DataWrapper(String data) {
88         mOperationType = "Special_Key";
89     }
```

```

90     mData = data;
91 }
92
93 /**
94  * Constructor.
95  *
96  * Used to create a <code>DataWrapper</code> for 2-D mouse movement
97  * operation. The <i>type of operation</i> field is implicit.
98  *
99  * @param x displacement data along x-axis.
100  * @param y displacement data along y-axis.
101  * @param sensitivity mouse movement sensitivity.
102  */
103 DataWrapper(int x, int y, float sensitivity) {
104     mMoveX = x;
105     mMoveY = y;
106     mSensitivity = sensitivity;
107     mOperationType = "Mouse_Touch";
108 }
109
110 /**
111  * Constructor.
112  *
113  * Used to create a <code>DataWrapper</code> for 3-D mouse movement
114  * operation. The <i>type of operation</i> field is implicit.
115  *
116  * @param quaternionObject a {@link project.android.sensor.representation.Quaternion}
117  *                           object.
118  * @param isInitQuat <code>true</code>, if <code>quaternionObject</code>
119  *                   is the initial, <br/>
120  *                   <code>false</code>, otherwise.
121  * @param sensitivity mouse movement sensitivity.
122  * @see project.android.sensor.representation.Quaternion
123  */
124 DataWrapper(Quaternion quaternionObject, boolean isInitQuat, float sensitivity) {
125     mOperationType = "Mouse_Move";
126     mSensitivity = sensitivity;
127     mIsInitQuat = isInitQuat;
128     mQuaternion = quaternionObject;
129 }
130
131 /**
132  * Returns a JSon-string of an object.
133  *
134  * @param object the input {@link java.lang.Object}.
135  * @return a JSon-string.
136  * @see com.google.gson.Gson
137  */
138 static String getGsonString(Object object) {
139     return new Gson().toJson(object);
140 }
141
142 /**
143  * Returns the <i>type of operation</i> field of this
144  * <code>DataWrapper</code>.
145  *
146  * @return the <i>type of operation</i> field.
147  */
148 String getOperationType() { return mOperationType; }
149
150 /**
151  * Returns the <i>data</i> field of this <code>DataWrapper</code>
152  * for keyboard and mouse button click operations.
153  *
154  * @return the <i>data</i> field.
155  */
156 String getData() { return mData; }
157
158 /**
159  * Checks if the stored {@link project.android.sensor.representation.Quaternion}
160  * object is the initial one.
161  *
162  * @return <code>true</code>, if it is initial, <br/>
163  *         <code>false</code>, otherwise.
164  */
165 boolean isInitQuat() { return mIsInitQuat; }
166
167 /**
168  * Returns the {@link project.android.sensor.representation.Quaternion} object
169  * stored in this <code>DataWrapper</code>.
170  *
171  * @return the {@link project.android.sensor.representation.Quaternion} object.
172  */
173 Quaternion getQuaternionObject() { return mQuaternion; }
174
175 /**
176  * Returns the displacement value along x-axis stored in this
177  * <code>DataWrapper</code>.
178  *
179  * @return the displacement along x-axis.
180  */
181 int getX() { return mMoveX; }
182
183 /**
184  * Returns the displacement value along y-axis stored in this

```

```
185     * <code>DataWrapper</code>.
186     *
187     * @return the displacement along y-axis.
188     */
189     int getY() { return mMoveY; }
190
191     /**
192     * Returns the mouse movement sensitivity value stored in this
193     * <code>DataWrapper</code>.
194     *
195     * @return the mouse movement sensitivity.
196     */
197     float getSensitivity() { return mSensitivity; }
198 }
```


KeyboardThread.java

```
1 package project.android.net;
2
3 import android.os.Bundle;
4 import android.util.Pair;
5 import android.view.LayoutInflater;
6 import android.view.ViewGroup;
7
8 import java.util.concurrent.LinkedBlockingQueue;
9 import java.util.concurrent.TimeUnit;
10
11 import static project.android.ConnectionFragment.sSecuredClient;
12
13 /**
14  * Implementation of the java.lang.Runnable for the thread responsible
15  * for sending keyboard data to the server.
16  *
17  * <p>
18  * This thread maintains a buffer to store the data to be sent to the server. It
19  * is implemented as a java.util.concurrent.LinkedBlockingQueue. The
20  * data contains two fields :
21  * <ul>
22  * <li>
23  * A data field which represents the data to be sent, and
24  * </li>
25  * <li>
26  * A boolean flag which determines if the data represents a special key.
27  * </li>
28  * </ul>
29  * </p>
30  * <p>
31  * The data is inserted to the buffer by the listener methods defined in the
32  * {@link project.android.KeyboardFragment} class. This thread continuously
33  * monitors the buffer to check if data is available. If available, it sends
34  * the data.
35  * </p>
36  * <p>
37  * This thread is started when the keyboard UI is loaded and the
38  * {@link project.android.KeyboardFragment#onCreateView(LayoutInflater, ViewGroup,
39  * Bundle)} method is called. It is stopped by invoking the {@link #setStopFlag()}
40  * method when the keyboard UI is destroyed and the
41  * {@link project.android.KeyboardFragment#onDestroyView()} method is called.
42  * </p>
43  *
44  * @see java.lang.Runnable
45  * @see java.util.concurrent.LinkedBlockingQueue
46  * @see project.android.KeyboardFragment
47  * @see project.android.KeyboardFragment#onCreateView(LayoutInflater, ViewGroup, Bundle)
48  * @see project.android.KeyboardFragment#onDestroyView()
49  * @see #setStopFlag()
50  */
51 public class KeyboardThread implements Runnable {
52     private boolean mStopFlag;
53     private LinkedBlockingQueue<Pair<String, Boolean>> mBuffer;
54
55     private static final long POLL_TIMEOUT = 100;
56     private static final TimeUnit TIME_UNIT = TimeUnit.MILLISECONDS;
57
58     /**
59      * Constructor.
60      *
61      * Initializes this thread.
62      */
63     public KeyboardThread() {
64         mStopFlag = false;
65         mBuffer = new LinkedBlockingQueue<>();
66     }
67
68     /**
69      * Inserts keyboard data to the buffer.
70      *
71      * This data is sent to the server by this KeyboardThread.
72      *
73      * @param data the data to be sent.
74      * @param isSpecialKey true, if data represents
75      * special key, false, otherwise.
76      */
77     public void addToBuffer(String data, boolean isSpecialKey) {
78         try {
79             mBuffer.put(Pair.create(data, isSpecialKey));
80         } catch (InterruptedException e) { e.printStackTrace(); }
81     }
82
83     /**
84      * Sets stop flag.
85      *
86      * Used to exit from the {@link #run()} method.
87      */
88 }
```

```

89     */
90     public void setStopFlag() { mStopFlag = true; }
91
92     /**
93     * Performs the sending of keyboard data in the buffer to the server.
94     *
95     * <p>
96     *     It checks if data is available in the buffer.
97     *     <ul>
98     *         <li>If available, it sends the data.</li>
99     *         <li>Otherwise, it waits for the data to be available.</li>
100    *     </ul>
101    * </p>
102    */
103    @Override
104    public void run() {
105        while(!mStopFlag) {
106            try {
107                Pair<String, Boolean> data = mBuffer.poll(POLL_TIMEOUT, TIME_UNIT);
108                if(data != null) {
109                    if (data.second) {
110                        sSecuredClient.sendData(data.first);
111                    } else sSecuredClient.sendData("Key", data.first);
112                    // Log.d("Keyboard sent", data.first + "___" + data.first.length());
113                }
114            } catch (InterruptedException e) { e.printStackTrace(); }
115        }
116    }
117 }

```

NetworkService.java

```
1 package project.android.net;
2
3 import android.app.Service;
4 import android.content.Intent;
5 import android.os.IBinder;
6 import android.widget.Toast;
7
8 import java.io.IOException;
9
10 import project.android.ConnectionFragment;
11 import project.android.MainActivity;
12 import project.android.security.EKEProvider;
13
14 import static project.android.ConnectionFragment.sSecuredClient;
15 import static project.android.ConnectionFragment.sConnectionAlive;
16 import static project.android.MouseFragment.sMouseAlive;
17
18 /**
19  * Class representing the <code>Service</code> that runs upon successful client-server
20  * connection.
21  *
22  * <p>
23  *     {@link ClientIOThread} starts this <code>NetworkService</code> when a successful
24  *     client-server connection has been established. This <code>NetworkService</code>
25  *     is destroyed when the {@link #onDestroy()} method is invoked upon app closing.
26  * </p>
27  *
28  * @see project.android.net.ConnectionTask#onPostExecute(EKEProvider)
29  * @see project.android.net.ClientIOThread#recieveData()
30  * @see project.android.MainActivity#onDestroy()
31  */
32 public class NetworkService extends Service {
33
34     private static String sAddress;
35
36     /**
37      * Overrides the <code>Service.onBind(Intent)</code> method of the Android API.
38      *
39      * Return the communication channel to the service. May return <code>null</code>,
40      * if clients can not bind to the service.
41      *
42      * @param intent The <code>Intent</code> that was used to bind to this service,
43      *               as given to <code>Context.bindService</code>. Note that any extras
44      *               that were included with the Intent at that point will not be seen
45      *               here.
46      * @return <code>IBinder</code> through which clients can call on to the service.
47      */
48     @Override
49     public IBinder onBind(Intent intent) { return null; }
50
51     /**
52      * Overrides the <code>Service.onStartCommand(Intent, int, int)</code> method of
53      * the Android API.
54      *
55      * Called by the system every time a client explicitly starts the service by calling
56      * <code>startService(Intent)</code>, providing the arguments it supplied and a unique
57      * integer token representing the start request.
58      *
59      * @param intent the <code>Intent</code> supplied to <code>startService(Intent)</code>,
60      *               as given. This may be <code>null</code> if the service is being
61      *               restarted after its process has gone away, and it had previously
62      *               returned anything except <code>START_STICKY_COMPATIBILITY</code>.
63      * @param flags the additional data about this start request. Currently either
64      *               <code>0</code>, <code>START_FLAG_REDELIVERY</code> or
65      *               <code>START_FLAG_RETRY</code>.
66      * @param startId a unique Integer representing this specific request to start.
67      * @return an integer indicating the semantics the system should use for the service's
68      *         current started state.
69      */
70     @Override
71     public int onStartCommand(Intent intent, int flags, int startId) {
72         ConnectionFragment connectionFragment = (ConnectionFragment)
73         MainActivity.getConnectionFragment();
74         connectionFragment.setIcon();
75         sAddress = intent.getStringExtra("Server");
76         Toast.makeText(this, "Connected to " + sAddress + "\nOpen either Mouse " +
77             "or Keyboard Tabs from the navigation bar",
78             Toast.LENGTH_SHORT).show();
79         return START_NOT_STICKY;
80     }
81
82     /**
83      * Overrides the <code>Service.onDestroy()</code> method of the Android API.
84      *
85      * <p>
86      *     Called by the system to notify a <code>Service</code> that it is no longer
87      *     used and is being removed. The service should clean up any resources it holds
88      *     (threads, registered receivers, etc) at this point. Upon return, there will
89      *     be no more calls in to this <code>android.app.Service</code> object and it is
90      *     effectively dead.
91      */
92 }
```

```

89      * </p>
90      */
91      @Override
92      public void onDestroy() {
93          super.onDestroy();
94          if(sConnectionAlive.get(sAddress)) {
95              new Thread(new Runnable() {
96                  @Override
97                  public void run() {
98                      sSecuredClient.sendStopSignal(true);
99                      sMouseAlive = false;
100                      try {
101                          sSecuredClient.close();
102                          sConnectionAlive.put(sAddress, false);
103                      } catch (IOException ignored) {}
104                  }
105              }).start();
106              Toast.makeText(this, sAddress + " disconnected successfully",
107                  Toast.LENGTH_SHORT).show();
108          }
109      }

```

ServerInfo.java

```
1 package project.android.net;
2
3 /**
4  * Encapsulation of the data sent during UDP broadcast by the server.
5  *
6  * <p>
7  *   This data is received by the {@link BroadcastReceiverThread}. This
8  *   contains :
9  *   <ul>
10  *     <li>the public key of the server, </li>
11  *     <li>status of the broadcast, and</li>
12  *     <li>address of the server</li>
13  *   </ul>
14  * </p>
15  * <p>
16  *   This message helps a client to discover a server.
17  * </p>
18  *
19  * @see project.android.net.BroadcastReceiverThread
20  */
21 public class ServerInfo {
22     private byte[] mServerPubKey;
23     private String mServerInfo;
24     private String mAddress;
25     private String mPairingKey;
26     private boolean mStopFlag;
27
28     static final int SERVER_INFO_LENGTH = 600;
29
30     /**
31      * Constructor.
32      * Initializes this <code>ServerInfo</code>.
33      *
34      * @param publicKey public key of the server.
35      * @param serverInfo canonical hostname of the server.
36      */
37     public ServerInfo(byte[] publicKey, String serverInfo) {
38         mServerPubKey = publicKey;
39         mServerInfo = serverInfo;
40         mStopFlag = false;
41     }
42
43     /**
44      * Compares two <code>ServerInfo</code> objects.
45      *
46      * @param obj the object to be compared.
47      * @return <code>true</code>, if the two objects are equal, <br/>
48      *         <code>false</code>, otherwise.
49      */
50     @Override
51     public boolean equals(Object obj) {
52         if (this == obj) return true;
53         if (obj == null) return false;
54         if (getClass() != obj.getClass()) return false;
55         ServerInfo other = (ServerInfo) obj;
56         return mAddress.equals(other.mAddress);
57     }
58
59     /**
60      * Sets the IPv4 server address.
61      * @param address IPv4 server address.
62      */
63     void setServerAddress(String address) { mAddress = address; }
64
65     /**
66      * Sets the pairing key used during client-server authentication.
67      * @param pairingKey the pairing key.
68      */
69     public void setPairingKey(String pairingKey) { mPairingKey = pairingKey; }
70
71     /**
72      * Returns the status of the server.
73      * @return server status.
74      */
75     boolean getStopFlag() { return mStopFlag; }
76
77     /**
78      * Returns the server public key.
79      * @return server public key.
80      */
81     byte[] getServerPubKey() { return mServerPubKey; }
82
83     /**
84      * Returns the canonical host name of the server.
85      * @return the canonical host name.
86      */
87     public String getServerInfo() { return mServerInfo; }
88
89     /**
```

```
90     * Returns the IPv4 address of the server.
91     * @return the IPv4 server address.
92     */
93     public String getAddress() { return mAddress; }
94
95     /**
96     * Returns the pairing key used during client-server authentication.
97     * @return the pairing key.
98     */
99     String getPairingKey() { return mPairingKey; }
100 }
```

EKEProvider.java

```
1 package project.android.security;
2
3 import org.apache.commons.codec.binary.Base64;
4 import org.spongycastle.cert.X509CertificateHolder;
5 import org.spongycastle.cert.X509v3CertificateBuilder;
6 import org.spongycastle.cert.jcajce.JcaX509CertificateConverter;
7 import org.spongycastle.cert.jcajce.JcaX509v3CertificateBuilder;
8 import org.spongycastle.jce.ECNamedCurveTable;
9 import org.spongycastle.jce.provider.BouncyCastleProvider;
10 import org.spongycastle.jce.spec.ECNamedCurveParameterSpec;
11 import org.spongycastle.operator.OperatorCreationException;
12 import org.spongycastle.operator.jcajce.JcaContentSignerBuilder;
13
14 import javax.crypto.BadPaddingException;
15 import javax.crypto.Cipher;
16 import javax.crypto.IllegalBlockSizeException;
17 import javax.crypto.KeyAgreement;
18 import javax.crypto.NoSuchPaddingException;
19 import javax.crypto.SecretKey;
20 import javax.crypto.spec.IvParameterSpec;
21 import javax.crypto.spec.SecretKeySpec;
22 import javax.security.auth.x500.X500Principal;
23
24 import java.io.IOException;
25 import java.io.UnsupportedEncodingException;
26 import java.math.BigInteger;
27 import java.security.InvalidAlgorithmParameterException;
28 import java.security.InvalidKeyException;
29 import java.security.Key;
30 import java.security.KeyFactory;
31 import java.security.KeyPair;
32 import java.security.KeyPairGenerator;
33 import java.security.KeyStoreException;
34 import java.security.MessageDigest;
35 import java.security.NoSuchAlgorithmException;
36 import java.security.NoSuchProviderException;
37 import java.security.PrivateKey;
38 import java.security.PublicKey;
39 import java.security.Security;
40 import java.security.UnrecoverableEntryException;
41 import java.security.cert.Certificate;
42 import java.security.cert.CertificateException;
43 import java.security.spec.InvalidKeySpecException;
44 import java.security.spec.X509EncodedKeySpec;
45 import java.util.Date;
46
47 /**
48  * Utility for end-to-end security module.
49  *
50  * <p>
51  *   This class provides the following features:
52  *   <ul>
53  *     <li>
54  *       <i>256-bit Elliptic Curve</i> private and public key generation.
55  *     </li>
56  *     <li><i>AES 256-bit</i> plaintext encryption.</li>
57  *     <li><i>AES 256-bit</i> ciphertext decryption.</li>
58  *     <li>Self-signed certificate generation.</li>
59  *   </ul>
60  * </p>
61  *
62  */
63 public class EKEProvider {
64
65     private SecretKey mSecretKey;
66     private PublicKey mPublicKey;
67     private byte[] mIV;
68
69     private static final String SIGNATURE_ALGORITHM = "SHA256withECDSA";
70     private static final String MESSAGE_DIGEST = "SHA-256";
71     private static final String KEY_GENERATION_ALGORITHM = "EC";
72     private static final String KEY_AGREEMENT_ALGORITHM = "ECDH";
73     private static final String SECURITY_PROVIDER = "SC";
74     private static final String ECC_CURVE_NAME = "brainpoolp256r1";
75     private static final String ENCRYPTION_ALGORITHM = "AES";
76     private static final String CIPHER_MODE = "AES/GCM/NoPadding";
77
78     private static final Date NOT_BEFORE = new Date(System.currentTimeMillis() - 24 * 60 *
60 * 1000);
79     private static final Date NOT_AFTER = new Date(System.currentTimeMillis() +
2592000000L);
80
81     static {
82         Security.insertProviderAt(new BouncyCastleProvider(), 1);
83     }
84
85     /**
86      * Constructor.
87      */
88 }
```

```

88     * Initializes this <code>EKEProvider</code>. If keystore exists, private and
89     * public <code>java.security.KeyPair</code> is loaded, otherwise the
90     * <code>KeyPair</code> is generated.
91     *
92     * @see project.android.security.KeyStoreManager
93     */
94     public EKEProvider() {
95         try {
96             KeyPair keyPair = null;
97             if (KeyStoreManager.keyStoreExists()) {
98                 keyPair = KeyStoreManager.getKSKeyPair();
99                 if (keyPair != null) mPublicKey = keyPair.getPublic();
100             }
101             if (keyPair == null) {
102                 keyPair = generateEKeys();
103                 if (keyPair != null) {
104                     mPublicKey = keyPair.getPublic();
105                     PrivateKey privateKey = keyPair.getPrivate();
106                     Certificate certificate = genSelfSignedCert(mPublicKey, privateKey);
107                     new KeyStoreManager().setMasterKey(privateKey, certificate);
108                 }
109             }
110         } catch (CertificateException | NoSuchAlgorithmException
111         | KeyStoreException | IOException | OperatorCreationException |
112         UnrecoverableEntryException e) {
113             e.printStackTrace();
114         }
115     }
116     /**
117     * Constructor.
118     *
119     * <p>
120     *     Initializes this <code>EKEProvider</code>. Generates a secret key using
121     *     the <i>SHA-256 message digest</i> of the pairing key used for client-server
122     *     authentication and the server <code>PublicKey</code>.
123     * </p>
124     * @param pairingKey the pairing key.
125     * @param serverPublicKey public key of the server.
126     */
127     public EKEProvider(byte[] pairingKey, byte[] serverPublicKey) {
128         MessageDigest msgDigest;
129         try {
130             msgDigest = MessageDigest.getInstance(MESSAGE_DIGEST, SECURITY_PROVIDER);
131             mIV = msgDigest.digest(pairingKey);
132             KeyPair keyPair = KeyStoreManager.getKSKeyPair();
133             if (keyPair != null) {
134                 X509EncodedKeySpec pubKeySpec = new
135                 X509EncodedKeySpec(Base64.decodeBase64(serverPublicKey));
136                 KeyFactory keyFactory = KeyFactory.getInstance(KEY_GENERATION_ALGORITHM,
137                 SECURITY_PROVIDER);
138                 mSecretKey = generateSharedSecret(keyPair.getPrivate(),
139                 keyFactory.generatePublic(pubKeySpec));
140             } catch (NoSuchAlgorithmException | NoSuchProviderException
141             | CertificateException | KeyStoreException | UnrecoverableEntryException
142             | IOException | InvalidKeySpecException e) {
143                 e.printStackTrace();
144             }
145         }
146     }
147     /**
148     * Returns the <code>Base64</code> encoded client <code>PublicKey</code>.
149     * @return the <code>Base64</code> encoded server public key. <br/>
150     *     <b><i>Note:</i></b> Returns <code>null</code>
151     *     if no public has been generated during
152     *     instantiation.
153     */
154     public byte[] getBase64EncodedPubKey() {
155         if (mPublicKey != null) {
156             return Base64.encodeBase64(mPublicKey.getEncoded());
157         }
158         return null;
159     }
160     private KeyPair generateEKeys() {
161         try {
162             ECNamedCurveParameterSpec parameterSpec =
163             ECNamedCurveTable.getParameterSpec(ECC_CURVE_NAME);
164             KeyPairGenerator keyPairGenerator =
165             KeyPairGenerator.getInstance(KEY_GENERATION_ALGORITHM, SECURITY_PROVIDER);
166             keyPairGenerator.initialize(parameterSpec);
167             return keyPairGenerator.generateKeyPair();
168         } catch (NoSuchAlgorithmException | InvalidAlgorithmParameterException |
169         NoSuchProviderException e) {
170             e.printStackTrace();
171             return null;
172         }
173     }
174     private SecretKey generateSharedSecret(PrivateKey privateKey, PublicKey publicKey) {
175         try {
176             KeyAgreement keyAgreement = KeyAgreement.getInstance(KEY_AGREEMENT_ALGORITHM,
177             SECURITY_PROVIDER);
178             keyAgreement.init(privateKey);

```



```

175         keyAgreement.doPhase(publicKey, true);
176         return keyAgreement.generateSecret(ENCRYPTION_ALGORITHM);
177     } catch (InvalidKeyException | NoSuchAlgorithmException | NoSuchProviderException e)
178     {
179         e.printStackTrace();
180         return null;
181     }
182
183     /**
184     * Encrypts plaintext using <i>AES 256-bit</i> encryption algorithm.
185     * @param plainText the plaintext to be encrypted.
186     * @return the ciphertext.
187     */
188     public String encryptString(String plainText) {
189         try {
190             Cipher cipher = Cipher.getInstance(CIPHER_MODE, SECURITY_PROVIDER);
191             cipher.init(Cipher.ENCRYPT_MODE, mSecretKey, new IvParameterSpec(mIV));
192             byte[] cipherText = cipher.doFinal(plainText.getBytes());
193             return new String(Base64.encodeBase64(cipherText));
194         } catch (NoSuchAlgorithmException | NoSuchProviderException
195                 | NoSuchPaddingException | InvalidKeyException
196                 | InvalidAlgorithmParameterException
197                 | IllegalBlockSizeException | BadPaddingException | NullPointerException e) {
198             e.printStackTrace();
199             return null;
200         }
201     }
202
203     /**
204     * Decrypts ciphertext using <i>AES 256-bit</i> bit decryption algorithm.
205     * @param cipherText the ciphertext to be decrypted.
206     * @return the plaintext.
207     */
208     public String decryptString(String cipherText) {
209         try {
210             Key decryptionKey = new SecretKeySpec(mSecretKey.getEncoded(),
211                 mSecretKey.getAlgorithm());
212             IvParameterSpec ivSpec = new IvParameterSpec(mIV);
213             Cipher cipher = Cipher.getInstance(CIPHER_MODE, SECURITY_PROVIDER);
214             byte[] cipherTextBytes = Base64.decodeBase64(cipherText.getBytes());
215             cipher.init(Cipher.DECRYPT_MODE, decryptionKey, ivSpec);
216             return new String(cipher.doFinal(cipherTextBytes), "UTF-8");
217         } catch (NoSuchAlgorithmException | NoSuchProviderException
218                 | NoSuchPaddingException | InvalidKeyException
219                 | InvalidAlgorithmParameterException
220                 | IllegalBlockSizeException | BadPaddingException
221                 | UnsupportedEncodingException | NullPointerException e) {
222             e.printStackTrace();
223             return null;
224         }
225     }
226
227     private Certificate genSelfSignedCert(PublicKey publicKey, PrivateKey privateKey)
228         throws OperatorCreationException, CertificateException {
229         X500Principal issuer = new X500Principal("CN=127.0.0.1");
230         BigInteger serial = BigInteger.valueOf(System.currentTimeMillis());
231         X509v3CertificateBuilder certGen =
232             new JcaX509v3CertificateBuilder(issuer, serial, NOT_BEFORE, NOT_AFTER,
233                 issuer, publicKey);
234         X509CertificateHolder certHolder = certGen.build(new
235             JcaContentSignerBuilder(SIGNATURE_ALGORITHM)
236                 .setProvider(SECURITY_PROVIDER)
237                 .build(privateKey));
238         return new
239             JcaX509CertificateConverter().setProvider(SECURITY_PROVIDER).getCertificate(certHolder);
240     }

```

KeyStoreManager.java

```
1 package project.android.security;
2
3 import android.content.SharedPreferences;
4
5 import java.io.File;
6 import java.io.FileInputStream;
7 import java.io.FileOutputStream;
8 import java.io.IOException;
9 import java.math.BigInteger;
10 import java.security.Key;
11 import java.security.KeyPair;
12 import java.security.KeyStore;
13 import java.security.KeyStoreException;
14 import java.security.NoSuchAlgorithmException;
15 import java.security.PrivateKey;
16 import java.security.PublicKey;
17 import java.security.SecureRandom;
18 import java.security.UnrecoverableEntryException;
19 import java.security.cert.Certificate;
20 import java.security.cert.CertificateException;
21
22 import static project.android.MainActivity.sRemouseDir;
23 import static project.android.MainActivity.sSharedPrefs;
24
25 /**
26  * Manages the operations on the <code>KeyStore</code>.
27  *
28  * <p>
29  *   This class provides a {@link java.security.KeyStore} which stores :
30  *   <ul>
31  *     <li> The private key ({@link java.security.PrivateKey}).</li>
32  *     <li>
33  *       A self-signed certificate ({@link java.security.cert.Certificate})
34  *       that contains the public key and the digital signature.
35  *     </li>
36  *   </ul>
37  *   The <code>KeyStore</code> is protected by a randomly generated
38  *   password.
39  * </p>
40  *
41  * @see project.android.security.EKEProvider
42  * @see java.security.KeyStore
43  * @see java.security.KeyPair
44  * @see java.security.cert.Certificate
45  */
46 class KeyStoreManager {
47     private KeyStore mKeyStore;
48
49     private static final String KEY_STORE_TYPE = "pkcs12";
50     private static final String KEY_STORE_ALIAS = "Remouse KeyStore";
51     private static final String KEY_STORE_PASSWORD = generateKSPassword();
52     private static final String KEY_STORE_NAME = sRemouseDir + "/remouse_keystore";
53
54     /**
55      * Constructor.
56      * Initializes this <code>KeyStoreManager</code>.
57      *
58      * @throws CertificateException
59      * @throws NoSuchAlgorithmException
60      * @throws KeyStoreException
61      * @throws IOException
62      */
63     KeyStoreManager() throws CertificateException, NoSuchAlgorithmException,
64     KeyStoreException, IOException {
65         mKeyStore = KeyStore.getInstance(KEY_STORE_TYPE);
66         mKeyStore.load(null, KEY_STORE_PASSWORD.toCharArray());
67     }
68
69     /**
70      * Checks if a <code>KeyStore</code> exists.
71      *
72      * @return <code>true</code>, if the <code>KeyStore</code> exists, <br/>
73      *         <code>false</code>, otherwise.
74      */
75     static boolean keyStoreExists() { return new File(KEY_STORE_NAME).exists(); }
76
77     /**
78      * Method to store the private key and the certificate in the
79      * <code>KeyStore</code>.
80      *
81      * @param privateKey the {@link java.security.PrivateKey} object.
82      * @param cert the {@link java.security.cert.Certificate} object(s).
83      * @throws KeyStoreException
84      * @throws IOException
85      * @throws CertificateException
86      * @throws NoSuchAlgorithmException
87      */
88     void setMasterKey(PrivateKey privateKey, Certificate ...cert) throws KeyStoreException,
```

```

89 | IOException, CertificateException, NoSuchAlgorithmException {
    |     KeyStore.PrivateKeyEntry privateKeyEntry = new KeyStore.PrivateKeyEntry(privateKey,
90 |     cert);
    |     mKeyStore.setEntry(KEY_STORE_ALIAS, privateKeyEntry, new
    |     KeyStore.PasswordProtection(KEY_STORE_PASSWORD.toCharArray()));
91 |     mKeyStore.store(new FileOutputStream(KEY_STORE_NAME),
    |     KEY_STORE_PASSWORD.toCharArray());
92 | }
93 |
94 | /**
95 |  * Returns the private-public key pair.
96 |  *
97 |  * @return {@link java.security.KeyPair} object.
98 |  * @throws IOException
99 |  * @throws KeyStoreException
100 |  * @throws CertificateException
101 |  * @throws NoSuchAlgorithmException
102 |  * @throws UnrecoverableEntryException
103 |  */
104 | static KeyPair getKSKeyPair() throws IOException, KeyStoreException,
    | CertificateException, NoSuchAlgorithmException, UnrecoverableEntryException {
105 |     FileInputStream fis = new FileInputStream(KEY_STORE_NAME);
106 |
107 |     KeyStore keyStore = KeyStore.getInstance(KEY_STORE_TYPE);
108 |     keyStore.load(fis, KEY_STORE_PASSWORD.toCharArray());
109 |
110 |     Key key = keyStore.getKey(KEY_STORE_ALIAS, KEY_STORE_PASSWORD.toCharArray());
111 |     if (key instanceof PrivateKey) {
112 |         Certificate certificate = keyStore.getCertificate(KEY_STORE_ALIAS);
113 |         PublicKey publicKey = certificate.getPublicKey();
114 |         return new KeyPair(publicKey, (PrivateKey) key);
115 |     }
116 |     return null;
117 | }
118 |
119 | private static String generateKSPassword() {
120 |     String password = sSharedPrefs.getString("Password", null);
121 |     if (password == null) {
122 |         SecureRandom random = new SecureRandom();
123 |         StringBuilder stringBuilder = new StringBuilder(new BigInteger(36, 0,
124 |         random).toString(Character.MAX_RADIX));
125 |         while (stringBuilder.length() > 6) {
126 |             stringBuilder.deleteCharAt(random.nextInt(stringBuilder.length()));
127 |         }
128 |         for (int i = 0; i < stringBuilder.length(); i++) {
129 |             char ch = stringBuilder.charAt(i);
130 |             if (Character.isLetter(ch) && Character.isLowerCase(ch) &&
131 |             random.nextFloat() < 0.5) {
132 |                 stringBuilder.setCharAt(i, Character.toUpperCase(ch));
133 |             }
134 |             SharedPreferences.Editor editor = sSharedPrefs.edit();
135 |             password = stringBuilder.toString();
136 |             editor.putString("Password", password);
137 |             editor.apply();
138 |         }
139 |         return password;
140 |     }
    | }

```

KalmanFilterProvider.java

```
1 package project.android.sensor.orientation;
2
3 import project.android.sensor.representation.Quaternion;
4
5 import android.hardware.Sensor;
6 import android.hardware.SensorEvent;
7 import android.hardware.SensorManager;
8
9 import static project.android.ConnectionFragment.sConnectionAlive;
10 import static project.android.ConnectionFragment.sSecuredClient;
11 import static project.android.MouseFragment.sMouseAlive;
12
13 /**
14  * Utility for providing the filter module.
15  *
16  * <p>
17  * This class provides methods that deliver the relative orientation from the
18  * gyroscope present in the Android device. This sensor does not deliver an
19  * absolute orientation (with respect to magnetic north and gravity) but only
20  * a relative measurement starting from the point where it started.
21  * </p>
22  * <p>
23  * It extends the {@link OrientationProvider} class which is responsible for
24  * accessing the sensor hardware.
25  * </p>
26  *
27  * @see project.android.sensor.orientation.OrientationProvider
28  */
29 public class KalmanFilterProvider extends OrientationProvider {
30     private final Quaternion deltaQuaternion = new Quaternion();
31     private Quaternion mCorrectedQuaternion = new Quaternion();
32
33     private long mTimestamp;
34     private boolean mIsInitQuat;
35
36     private static final float NS2S = 1.0f / 1000000000.0f;
37
38     private static final double EPSILON = 0.1f; //[Experimental value] Filter-threshold
39
40     /**
41      * Constructor.
42      *
43      * Initializes this <code>KalmanFilterProvider</code>.
44      *
45      * @param sensorManager a <code>SensorManager</code> object.
46      */
47     public KalmanFilterProvider(SensorManager sensorManager) {
48         super(sensorManager);
49         mSensorList.add(sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE));
50         mIsInitQuat = true;
51         sensorStart();
52     }
53
54     /**
55      * Overrides the <code>SensorEventListener.onSensorChanged()</code>
56      * method of the Android API.
57      *
58      * <p>
59      * Called when there is a new sensor event. It is also called if there
60      * is a reading available from a sensor with the exact same sensor
61      * values (but a newer timestamp).
62      * </p>
63      *
64      * @param event the <code>SensorEvent</code> object containing full
65      * information about the event.
66      */
67     @Override
68     public void onSensorChanged(SensorEvent event) {
69         // it is a good practice to check that we received the proper event
70         if (event.sensor.getType() == Sensor.TYPE_GYROSCOPE) {
71             // This timestamps delta rotation to be multiplied by the current rotation
72             if (mTimestamp != 0) {
73                 final float dT = (event.timestamp - mTimestamp) * NS2S;
74                 // Axis of the rotation sample, not normalized yet.
75                 float axisX = event.values[0];
76                 float axisY = event.values[1];
77                 float axisZ = event.values[2];
78
79                 // Calculate the angular speed of the sample
80                 double gyroscopeRotationVelocity = Math.sqrt(axisX * axisX + axisY * axisY +
81 axisZ * axisZ);
82
83                 if (gyroscopeRotationVelocity > EPSILON) {
84                     axisX /= gyroscopeRotationVelocity;
85                     axisY /= gyroscopeRotationVelocity;
86                     axisZ /= gyroscopeRotationVelocity;
87                 }
88             }
89         }
90     }
91 }
```

```

89         axisZ /= gyroscopeRotationVelocity;
90     }
91
92     double thetaOverTwo = gyroscopeRotationVelocity * dT / 2.0f;
93     double sinThetaOverTwo = Math.sin(thetaOverTwo);
94     double cosThetaOverTwo = Math.cos(thetaOverTwo);
95     deltaQuaternion.setX((float) (sinThetaOverTwo * axisX));
96     deltaQuaternion.setY((float) (sinThetaOverTwo * axisY));
97     deltaQuaternion.setZ((float) (sinThetaOverTwo * axisZ));
98     deltaQuaternion.setW(-(float) cosThetaOverTwo);
99
100     synchronized (synchronizationToken) {
101         deltaQuaternion.multiplyByQuat(currentOrientationQuaternion,
currentOrientationQuaternion);
102     }
103
104     mCorrectedQuaternion.set(currentOrientationQuaternion);
105
106     mCorrectedQuaternion.setW(-mCorrectedQuaternion.getW());
107
108     synchronized (synchronizationToken) {
109         // Set the rotation matrix as well to have both representations
110         SensorManager.getRotationMatrixFromVector(currentOrientationRotationMatrix.matrix,
mCorrectedQuaternion.array());
111     }
112
113     if(sConnectionAlive.containsValue(true) && sMouseAlive) {
114         new Thread(new Runnable() {
115             @Override
116             public void run() {
117                 sSecuredClient.sendData(mCorrectedQuaternion, mIsInitQuat);
118                 mIsInitQuat = false;
119             }
120         }).start();
121     } else sensorStop();
122 }
123 mTimestamp = event.timestamp;
124 }
125 }
126
127 @Override
128 public void onAccuracyChanged(Sensor sensor, int i) {}
129 }

```

OrientationProvider.java

```
1 package project.android.sensor.orientation;
2
3 import android.app.Activity;
4 import android.content.Context;
5 import android.hardware.Sensor;
6 import android.hardware.SensorEventListener;
7 import android.hardware.SensorManager;
8 import project.android.sensor.representation.MatrixF4x4;
9 import project.android.sensor.representation.Quaternion;
10
11 import java.util.ArrayList;
12 import java.util.List;
13
14 /**
15  * Accesses the sensor hardware to read sensor data.
16  *
17  * <p>
18  *   Classes implementing this interface provide an orientation of the device
19  *   either by directly accessing hardware, using Android sensor fusion or
20  *   fusing sensors itself. The orientation can be provided as a rotation matrix
21  *   ({@link project.android.sensor.representation.MatrixF4x4}) or a quaternion
22  *   ({@link project.android.sensor.representation.Quaternion}).
23  * </p>
24  *
25  * @see project.android.sensor.representation.Quaternion
26  * @see project.android.sensor.representation.MatrixF4x4
27  * @see project.android.sensor.orientation.KalmanFilterProvider
28  */
29 public abstract class OrientationProvider implements SensorEventListener {
30     final Object synchronizationToken = new Object();
31
32     final MatrixF4x4 currentOrientationRotationMatrix;
33
34     final Quaternion currentOrientationQuaternion;
35
36     List<Sensor> mSensorList = new ArrayList<>();
37
38     private SensorManager sensorManager;
39
40     /**
41      * Constructor.
42      *
43      * Initializes this <code>OrientationProvider</code>.
44      *
45      * @param sensorManager a <code>SensorManager</code> object.
46      */
47     OrientationProvider(SensorManager sensorManager) {
48         this.sensorManager = sensorManager;
49
50         // Initialise with identity
51         currentOrientationRotationMatrix = new MatrixF4x4();
52
53         // Initialise with identity
54         currentOrientationQuaternion = new Quaternion();
55     }
56
57     /**
58      * Checks if gyroscope is available in the device.
59      *
60      * @param activity the current <code>Activity</code>.
61      * @return <code>true</code>, if gyroscope is available, <br/>
62      *         <code>false</code>, otherwise.
63      */
64     public static boolean checkGyro(Activity activity) {
65         SensorManager sensorManager = (SensorManager)
66             activity.getSystemService(Context.SENSOR_SERVICE);
67         return sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE) != null;
68     }
69
70     /**
71      * Starts the sensor fusion (e.g., when resuming the activity).
72      */
73     void sensorStart() {
74         for (Sensor sensor : mSensorList) {
75             sensorManager.registerListener(this, sensor,
76                 SensorManager.SENSOR_DELAY_UI);
77         }
78     }
79
80     /**
81      * Stops the sensor fusion (e.g., when pausing/suspending the activity).
82      */
83     public void sensorStop() {
84         for (Sensor sensor : mSensorList) {
85             sensorManager.unregisterListener(this, sensor);
86         }
87     }
88 }
```

```

89     /**
90     * Gets the current rotation of the device.
91     *
92     * @param matrix a {@link project.android.sensor.representation.MatrixF4x4}
93     *               object representing the rotation.
94     * @see project.android.sensor.representation.MatrixF4x4
95     */
96     public void getRotationMatrix(MatrixF4x4 matrix) {
97         synchronized (synchronizationToken) {
98             matrix.set(currentOrientationRotationMatrix);
99         }
100     }
101
102     /**
103     * Gets the current rotation of the device.
104     *
105     * @param quaternion a {@link project.android.sensor.representation.Quaternion}
106     *                   object representing the rotation.
107     * @see project.android.sensor.representation.Quaternion
108     */
109     public void getQuaternion(Quaternion quaternion) {
110         synchronized (synchronizationToken) {
111             quaternion.set(currentOrientationQuaternion);
112         }
113     }
114
115     /**
116     * Gets the current rotation of the device.
117     *
118     * @param angles a float array representing the rotation.
119     */
120     public void getEulerAngles(float angles[]) {
121         synchronized (synchronizationToken) {
122             SensorManager.getOrientation(currentOrientationRotationMatrix.matrix, angles);
123         }
124     }
125 }

```

MatrixF4x4.java

[illegible]


```

87     boolean isColumnMajor() {
88         return mColMaj;
89     }
90
91     /**
92     * Multiplies the given vector by this <code>MatrixF4x4</code>.
93     *
94     * This should only be used if the matrix is of size 16.
95     *
96     * @param vector a {@link Vector4f}.
97     * @throws InvalidParameterSpecException
98     */
99     public void multiplyVector4fByMatrix(Vector4f vector) throws
InvalidParameterSpecException {
100
101         if (matrix.length == 16) {
102             float x = 0;
103             float y = 0;
104             float z = 0;
105             float w = 0;
106
107             float[] vectorArray = vector.array();
108
109             if (mColMaj) {
110                 for (int i = 0; i < 4; i++) {
111
112                     int k = i * 4;
113
114                     x += this.matrix[k] * vectorArray[i];
115                     y += this.matrix[k + 1] * vectorArray[i];
116                     z += this.matrix[k + 2] * vectorArray[i];
117                     w += this.matrix[k + 3] * vectorArray[i];
118                 }
119             } else {
120                 for (int i = 0; i < 4; i++) {
121
122                     x += this.matrix[i] * vectorArray[i];
123                     y += this.matrix[4 + i] * vectorArray[i];
124                     z += this.matrix[8 + i] * vectorArray[i];
125                     w += this.matrix[12 + i] * vectorArray[i];
126                 }
127             }
128
129             vector.setX(x);
130             vector.setY(y);
131             vector.setZ(z);
132             vector.setW(w);
133         } else {
134             throw new InvalidParameterSpecException("Expects the matrix to be of size 16");
135         }
136     }
137
138     /**
139     * Multiplies the given vector by this <code>MatrixF4x4</code>.
140     *
141     * This should only be used if the matrix is of size 9.
142     *
143     * @param vector a {@link Vector3f}.
144     * @throws InvalidParameterSpecException
145     */
146     public void multiplyVector3fByMatrix(Vector3f vector) throws
InvalidParameterSpecException {
147
148         if (matrix.length == 9) {
149             float x = 0;
150             float y = 0;
151             float z = 0;
152
153             float[] vectorArray = vector.toArray();
154
155             if (!mColMaj) {
156                 for (int i = 0; i < 3; i++) {
157
158                     int k = i * 3;
159
160                     x += this.matrix[k] * vectorArray[i];
161                     y += this.matrix[k + 1] * vectorArray[i];
162                     z += this.matrix[k + 2] * vectorArray[i];
163                 }
164             } else {
165                 for (int i = 0; i < 3; i++) {
166
167                     x += this.matrix[i] * vectorArray[i];
168                     y += this.matrix[3 + i] * vectorArray[i];
169                     z += this.matrix[6 + i] * vectorArray[i];
170                 }
171             }
172
173             vector.setX(x);
174             vector.setY(y);
175             vector.setZ(z);
176         } else {
177             throw new InvalidParameterSpecException("Expects the matrix to be of size 9");
178         }
179     }

```

```

180
181 /**
182  * Multiplies a <code>MatrixF4x4</code> with this.
183  *
184  * @param matrixf a <code>MatrixF4x4</code>.
185  */
186 public void multiplyMatrix4x4ByMatrix(MatrixF4x4 matrixf) {
187     float[] bufferMatrix = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
188     float[] matrix = matrixf.getMatrix();
189     multiplyMatrix(matrix, 0, bufferMatrix, 0);
190     matrixf.setMatrix(bufferMatrix);
191 }
192
193
194 private void multiplyMatrix(float[] input, int inputOffset, float[] output, int
outputOffset) {
195     for (int i = 0; i < 4; i++) {
196         for (int j = 0; j < 4; j++) {
197             int k = i * 4;
198             output[outputOffset + j] += this.matrix[k + j] * input[inputOffset + i];
199             output[outputOffset + 4 + j] += this.matrix[k + j] * input[inputOffset + 4 +
200 i];
201             output[outputOffset + 2 * 4 + j] += this.matrix[k + j] * input[inputOffset +
202 2 * 4 + i];
203             output[outputOffset + 3 * 4 + j] += this.matrix[k + j] * input[inputOffset +
204 3 * 4 + i];
205         }
206     }
207 }
208
209 public void transpose() {
210     if (this.matrix.length == 16) {
211         float[] newMatrix = new float[16];
212         for (int i = 0; i < 4; i++) {
213             int k = i * 4;
214             newMatrix[k] = matrix[i];
215             newMatrix[k + 1] = matrix[4 + i];
216             newMatrix[k + 2] = matrix[8 + i];
217             newMatrix[k + 3] = matrix[12 + i];
218         }
219         matrix = newMatrix;
220     } else {
221         float[] newMatrix = new float[9];
222         for (int i = 0; i < 3; i++) {
223             int k = i * 3;
224             newMatrix[k] = matrix[i];
225             newMatrix[k + 1] = matrix[3 + i];
226             newMatrix[k + 2] = matrix[6 + i];
227         }
228         matrix = newMatrix;
229     }
230 }
231
232 /**
233  * Sets the value of X0.
234  *
235  * @param value value to set.
236  */
237 void setX0(float value) {
238     if (matrix.length == 16) {
239         if (mColMaj) {
240             matrix[MAT_IND_COL16_3X3[0]] = value;
241         } else {
242             matrix[MAT_IND_ROW16_3X3[0]] = value;
243         }
244     } else {
245         if (mColMaj) {
246             matrix[MAT_IND_COL9_3X3[0]] = value;
247         } else {
248             matrix[MAT_IND_ROW9_3X3[0]] = value;
249         }
250     }
251 }
252
253
254 /**
255  * Sets the value of X1.
256  *
257  * @param value value to set.
258  */
259 void setX1(float value) {
260     if (matrix.length == 16) {
261         if (mColMaj) {
262             matrix[MAT_IND_COL16_3X3[1]] = value;
263         } else {
264             matrix[MAT_IND_ROW16_3X3[1]] = value;
265         }
266     } else {
267         if (mColMaj) {
268             matrix[MAT_IND_COL9_3X3[1]] = value;
269         } else {
270

```

```

271         matrix[MAT_IND_ROW9_3X3[1]] = value;
272     }
273 }
274
275 /**
276  * Sets the value of X2.
277  * @param value value to set.
278  */
279 void setX2(float value) {
280
281     if (matrix.length == 16) {
282         if (mColMaj) {
283             matrix[MAT_IND_COL16_3X3[2]] = value;
284         } else {
285             matrix[MAT_IND_ROW16_3X3[2]] = value;
286         }
287     } else {
288         if (mColMaj) {
289             matrix[MAT_IND_COL9_3X3[2]] = value;
290         } else {
291             matrix[MAT_IND_ROW9_3X3[2]] = value;
292         }
293     }
294 }
295
296 /**
297  * Sets the value of Y0.
298  * @param value value to set.
299  */
300 void setY0(float value) {
301
302     if (matrix.length == 16) {
303         if (mColMaj) {
304             matrix[MAT_IND_COL16_3X3[3]] = value;
305         } else {
306             matrix[MAT_IND_ROW16_3X3[3]] = value;
307         }
308     } else {
309         if (mColMaj) {
310             matrix[MAT_IND_COL9_3X3[3]] = value;
311         } else {
312             matrix[MAT_IND_ROW9_3X3[3]] = value;
313         }
314     }
315 }
316
317 /**
318  * Sets the value of Y1.
319  * @param value value to set.
320  */
321 void setY1(float value) {
322
323     if (matrix.length == 16) {
324         if (mColMaj) {
325             matrix[MAT_IND_COL16_3X3[4]] = value;
326         } else {
327             matrix[MAT_IND_ROW16_3X3[4]] = value;
328         }
329     } else {
330         if (mColMaj) {
331             matrix[MAT_IND_COL9_3X3[4]] = value;
332         } else {
333             matrix[MAT_IND_ROW9_3X3[4]] = value;
334         }
335     }
336 }
337
338 /**
339  * Sets the value of Y2.
340  * @param value value to set.
341  */
342 void setY2(float value) {
343
344     if (matrix.length == 16) {
345         if (mColMaj) {
346             matrix[MAT_IND_COL16_3X3[5]] = value;
347         } else {
348             matrix[MAT_IND_ROW16_3X3[5]] = value;
349         }
350     } else {
351         if (mColMaj) {
352             matrix[MAT_IND_COL9_3X3[5]] = value;
353         } else {
354             matrix[MAT_IND_ROW9_3X3[5]] = value;
355         }
356     }
357 }
358
359 /**
360  * Sets the value of Z0.

```

```

366     *
367     * @param value value to set.
368     */
369 void setZ0(float value) {
370     if (matrix.length == 16) {
371         if (mColMaj) {
372             matrix[MAT_IND_COL16_3X3[6]] = value;
373         } else {
374             matrix[MAT_IND_ROW16_3X3[6]] = value;
375         }
376     } else {
377         if (mColMaj) {
378             matrix[MAT_IND_COL9_3X3[6]] = value;
379         } else {
380             matrix[MAT_IND_ROW9_3X3[6]] = value;
381         }
382     }
383 }
384 }
385
386 /**
387  * Sets the value of Z1.
388  *
389  * @param value value to set.
390  */
391 void setZ1(float value) {
392     if (matrix.length == 16) {
393         if (mColMaj) {
394             matrix[MAT_IND_COL16_3X3[7]] = value;
395         } else {
396             matrix[MAT_IND_ROW16_3X3[7]] = value;
397         }
398     } else {
399         if (mColMaj) {
400             matrix[MAT_IND_COL9_3X3[7]] = value;
401         } else {
402             matrix[MAT_IND_ROW9_3X3[7]] = value;
403         }
404     }
405 }
406 }
407
408 /**
409  * Sets the value of Z2.
410  *
411  * @param value value to set.
412  */
413 void setZ2(float value) {
414     if (matrix.length == 16) {
415         if (mColMaj) {
416             matrix[MAT_IND_COL16_3X3[8]] = value;
417         } else {
418             matrix[MAT_IND_ROW16_3X3[8]] = value;
419         }
420     } else {
421         if (mColMaj) {
422             matrix[MAT_IND_COL9_3X3[8]] = value;
423         } else {
424             matrix[MAT_IND_ROW9_3X3[8]] = value;
425         }
426     }
427 }
428 }
429
430 /**
431  * Sets the value of X3.
432  *
433  * @param value value to set.
434  */
435 void setX3(float value) {
436     if (matrix.length == 16) {
437         if (mColMaj) {
438             matrix[MAT_IND_COL16_4X4[3]] = value;
439         } else {
440             matrix[MAT_IND_ROW16_4X4[3]] = value;
441         }
442     } else {
443         throw new IllegalStateException("length of matrix should be 16");
444     }
445 }
446 }
447
448 /**
449  * Sets the value of Y3.
450  *
451  * @param value value to set.
452  */
453 void setY3(float value) {
454     if (matrix.length == 16) {
455         if (mColMaj) {
456             matrix[MAT_IND_COL16_4X4[7]] = value;
457         } else {
458             matrix[MAT_IND_ROW16_4X4[7]] = value;
459         }
460     }

```

```

461     } else {
462         throw new IllegalStateException("length of matrix should be 16");
463     }
464 }
465
466 /**
467  * Sets the value of Z3.
468  *
469  * @param value value to set.
470  */
471 void setZ3(float value) {
472     if (matrix.length == 16) {
473         if (mColMaj) {
474             matrix[MAT_IND_COL16_4X4[11]] = value;
475         } else {
476             matrix[MAT_IND_ROW16_4X4[11]] = value;
477         }
478     } else {
479         throw new IllegalStateException("length of matrix should be 16");
480     }
481 }
482
483 /**
484  * Sets the value of W0.
485  *
486  * @param value value to set.
487  */
488 void setW0(float value) {
489     if (matrix.length == 16) {
490         if (mColMaj) {
491             matrix[MAT_IND_COL16_4X4[12]] = value;
492         } else {
493             matrix[MAT_IND_ROW16_4X4[12]] = value;
494         }
495     } else {
496         throw new IllegalStateException("length of matrix should be 16");
497     }
498 }
499
500 /**
501  * Sets the value of W1.
502  *
503  * @param value value to set.
504  */
505 void setW1(float value) {
506     if (matrix.length == 16) {
507         if (mColMaj) {
508             matrix[MAT_IND_COL16_4X4[13]] = value;
509         } else {
510             matrix[MAT_IND_ROW16_4X4[13]] = value;
511         }
512     } else {
513         throw new IllegalStateException("length of matrix should be 16");
514     }
515 }
516
517 /**
518  * Sets the value of W2.
519  *
520  * @param value value to set.
521  */
522 void setW2(float value) {
523     if (matrix.length == 16) {
524         if (mColMaj) {
525             matrix[MAT_IND_COL16_4X4[14]] = value;
526         } else {
527             matrix[MAT_IND_ROW16_4X4[14]] = value;
528         }
529     } else {
530         throw new IllegalStateException("length of matrix should be 16");
531     }
532 }
533
534 /**
535  * Sets the value of W3.
536  *
537  * @param value value to set.
538  */
539 void setW3(float value) {
540     if (matrix.length == 16) {
541         if (mColMaj) {
542             matrix[MAT_IND_COL16_4X4[15]] = value;
543         } else {
544             matrix[MAT_IND_ROW16_4X4[15]] = value;
545         }
546     } else {
547         throw new IllegalStateException("length of matrix should be 16");
548     }
549 }
550 }
551 }
552 }
553 }
554 }
555 }

```

```

1 package project.android.sensor.representation;
2
3 /**
4  * Class to represent a quaternion.
5  *
6  * <p>
7  * A quaternion  $H$  may be identified with  $R^4$ ,
8  * a four-dimensional vector space over the real numbers. The
9  * elements of this basis are customarily denoted as  $1$ ,
10 *  $i$ ,  $j$ , and  $k$ . Every element of  $H$ 
11 * can be uniquely written as a linear combination of these
12 * basis elements, that is, as  $a1 + bi + cj + dk$ , where
13 *  $a$ ,  $b$ ,  $c$ , and  $d$  are real
14 * numbers.
15 * </p>
16 * <p>
17 * Quaternions allow for elegant descriptions of 3D rotations,
18 * interpolations as well as extrapolations and compared to
19 * Euler angles, they don't suffer from gimbal lock. I
20 * nterpolations between two Quaternions are called SLERP
21 * (Spherical Linear Interpolation).
22 * </p>
23 * <p>
24 * If  $a + bi + cj + dk$  is any quaternion, then  $a$ 
25 * is called its scalar part and  $bi + cj + dk$  is called
26 * its vector part. The scalar part of a quaternion is always
27 * real, and the vector part is always pure imaginary.
28 * </p>
29 * <p>
30 * Using the basis  $1, i, j, k$  of  $H$  it is possible to
31 * write  $H$  as a set of quadruples:
32 * </p>
33 * <p>
34 *  $H = \{(a,b,c,d) \mid a,b,c,d \in R\}$ .
35 * </p>
36 * <p>
37 * Then the basis elements are:
38 *  $1 = (1,0,0,0)$ ,
39 *  $i = (0,1,0,0)$ ,
40 *  $j = (0,0,1,0)$ ,
41 *  $k = (0,0,0,1)$ .
42 * </p>
43 * <p>
44 * <b>Addition of two quaternions :</b>
45 *  $(a_1, b_1, c_1, d_1) + (a_2, b_2, c_2, d_2) =$ 
46 *  $(a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2)$ 
47 * </p>
48 * <b>Multiplications of two quaternions :</b>
49 *  $(a_1, b_1, c_1, d_1) \cdot (a_2, b_2, c_2, d_2) =$ 
50 *  $(a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2,$ 
51 *  $a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2,$ 
52 *  $a_1c_2 - c_1a_2 + b_1d_2 + d_1b_2,$ 
53 *  $a_1d_2 + d_1a_2 - b_1c_2 + c_1b_2)$ 
54 * </p>
55 * <b>Normalization of quaternion</b>  $(a,b,c,d)$ 
56 *  $(\frac{a}{mag}, \frac{b}{mag}, \frac{c}{mag}, \frac{d}{mag})$ 
57 * where,  $mag = \sqrt{a^2 + b^2 + c^2 + d^2}$ .
58 * </p>
59 * <p>
60 * The sensor data for 3-D mouse movement is sent from the
61 * mobile device is represented in the form of a
62 * Quaternion.
63 * </p>
64 * @see project.android.sensor.representation.Vector4f
65 */
66 public class Quaternion extends Vector4f {
67
68 }

```

```

90 //For better performance update it only when it is accessed, not on every change
91
92 private MatrixF4x4 mMatrix;
93 private boolean mDirty = false;
94 private Vector4f mTmpVector = new Vector4f();
95 private Quaternion mTmpQuaternion;
96
97 /**
98  * Constructor.
99  *
100  * Initializes this <code>Quaternion</code>.
101  */
102 public Quaternion() {
103     super();
104     mMatrix = new MatrixF4x4();
105     loadIdentityQuat();
106 }
107
108 /**
109  * Normalizes this <code>Quaternion</code>.
110  */
111 public void normalize() {
112     this.mDirty = true;
113     float mag = (float) Math.sqrt(mPoints[3] * mPoints[3] + mPoints[0] * mPoints[0] +
114 mPoints[1] * mPoints[1] +
115 mPoints[2] * mPoints[2]);
116     mPoints[3] = mPoints[3] / mag;
117     mPoints[0] = mPoints[0] / mag;
118     mPoints[1] = mPoints[1] / mag;
119     mPoints[2] = mPoints[2] / mag;
120 }
121
122 /**
123  * Copies the values from the given <code>Quaternion</code>
124  * to this one.
125  * @param quat the <code>Quaternion</code> to copy from.
126  */
127 public void set(Quaternion quat) {
128     this.mDirty = true;
129     copyVec4(quat);
130 }
131
132 /**
133  * Multiplies this <code>Quaternion</code> with another.
134  *
135  * @param input <code>Quaternion</code> with which
136  * <code>this</code> is multiplied.
137  * @param output result of <code>(this * input)</code>.
138  */
139 public void multiplyByQuat(Quaternion input, Quaternion output) {
140
141     if (input != output) {
142         output.mPoints[3] = (mPoints[3] * input.mPoints[3] - mPoints[0] *
143 input.mPoints[0] - mPoints[1] * input.mPoints[1] - mPoints[2] * input.mPoints[2]);
144         output.mPoints[0] = (mPoints[3] * input.mPoints[0] + mPoints[0] *
145 input.mPoints[3] + mPoints[1] * input.mPoints[2] - mPoints[2] * input.mPoints[1]);
146         output.mPoints[1] = (mPoints[3] * input.mPoints[1] + mPoints[1] *
147 input.mPoints[3] + mPoints[2] * input.mPoints[0] - mPoints[0] * input.mPoints[2]);
148         output.mPoints[2] = (mPoints[3] * input.mPoints[2] + mPoints[2] *
149 input.mPoints[3] + mPoints[0] * input.mPoints[1] - mPoints[1] * input.mPoints[0]);
150     } else {
151         mTmpVector.mPoints[0] = input.mPoints[0];
152         mTmpVector.mPoints[1] = input.mPoints[1];
153         mTmpVector.mPoints[2] = input.mPoints[2];
154         mTmpVector.mPoints[3] = input.mPoints[3];
155
156         output.mPoints[3] = (mPoints[3] * mTmpVector.mPoints[3] - mPoints[0] *
157 mTmpVector.mPoints[0] - mPoints[1] * mTmpVector.mPoints[1] - mPoints[2] *
158 mTmpVector.mPoints[2]);
159         output.mPoints[0] = (mPoints[3] * mTmpVector.mPoints[0] + mPoints[0] *
160 mTmpVector.mPoints[3] + mPoints[1] * mTmpVector.mPoints[2] - mPoints[2] *
161 mTmpVector.mPoints[1]);
162         output.mPoints[1] = (mPoints[3] * mTmpVector.mPoints[1] + mPoints[1] *
163 mTmpVector.mPoints[3] + mPoints[2] * mTmpVector.mPoints[0] - mPoints[0] *
164 mTmpVector.mPoints[2]);
165         output.mPoints[2] = (mPoints[3] * mTmpVector.mPoints[2] + mPoints[2] *
166 mTmpVector.mPoints[3] + mPoints[0] * mTmpVector.mPoints[1] - mPoints[1] *
167 mTmpVector.mPoints[0]);
168     }
169 }
170
171 /**
172  * Multiplies this <code>Quaternion</code> by the input one.
173  *
174  * @param input <code>Quaternion</code> to be multiplied with.
175  */
176 public void multiplyByQuat(Quaternion input) {
177     this.mDirty = true;
178     if (mTmpQuaternion == null) mTmpQuaternion = new Quaternion();
179     mTmpQuaternion.copyVec4(this);
180     multiplyByQuat(input, mTmpQuaternion);
181     this.copyVec4(mTmpQuaternion);
182 }

```

```

172  /**
173  * Multiplies this <code>Quaternion</code> with a scalar. <br/>
174  *
175  * This overrides the {@link Vector4f#multiplyByScalar(float)}
176  * method.
177  *
178  * @param scalar the scalar.
179  * @see project.android.sensor.representation.Vector4f#multiplyByScalar(float)
180  */
181  public void multiplyByScalar(float scalar) {
182      this.mDirty = true;
183      super.multiplyByScalar(scalar);
184  }
185
186  public void addQuat(Quaternion input) {
187      this.mDirty = true;
188      addQuat(input, this);
189  }
190
191  private void addQuat(Quaternion input, Quaternion output) {
192      output.setX(getX() + input.getX());
193      output.setY(getY() + input.getY());
194      output.setZ(getZ() + input.getZ());
195      output.setW(getW() + input.getW());
196  }
197
198  public void subQuat(Quaternion input) {
199      this.mDirty = true;
200      subQuat(input, this);
201  }
202
203  private void subQuat(Quaternion input, Quaternion output) {
204      output.setX(getX() - input.getX());
205      output.setY(getY() - input.getY());
206      output.setZ(getZ() - input.getZ());
207      output.setW(getW() - input.getW());
208  }
209
210  private void convertQuatToMatrix() {
211      float x = mPoints[0];
212      float y = mPoints[1];
213      float z = mPoints[2];
214      float w = mPoints[3];
215
216      mMatrix.setX0(1 - 2 * (y * y) - 2 * (z * z));
217      mMatrix.setX1(2 * (x * y) + 2 * (w * z));
218      mMatrix.setX2(2 * (x * z) - 2 * (w * y));
219      mMatrix.setX3(0);
220      mMatrix.setY0(2 * (x * y) - 2 * (w * z));
221      mMatrix.setY1(1 - 2 * (x * x) - 2 * (z * z));
222      mMatrix.setY2(2 * (y * z) + 2 * (w * x));
223      mMatrix.setY3(0);
224      mMatrix.setZ0(2 * (x * z) + 2 * (w * y));
225      mMatrix.setZ1(2 * (y * z) - 2 * (w * x));
226      mMatrix.setZ2(1 - 2 * (x * x) - 2 * (y * y));
227      mMatrix.setZ3(0);
228      mMatrix.setW0(0);
229      mMatrix.setW1(0);
230      mMatrix.setW2(0);
231      mMatrix.setW3(1);
232  }
233
234  /**
235  * Gets an axis angle representation of this <code>Quaternion</code>.
236  *
237  * @param output <code>Vector4f</code> axis angle.
238  */
239  public void toAxisAngle(Vector4f output) {
240      if (getW() > 1) {
241          normalize();
242      }
243      float angle = 2 * (float) Math.toDegrees(Math.acos(getW()));
244      float x;
245      float y;
246      float z;
247
248      float s = (float) Math.sqrt(1 - getW() * getW());
249      if (s < 0.001) {
250          x = mPoints[0]; // To get normalised axis replace with x=1 y=z=0
251          y = mPoints[1];
252          z = mPoints[2];
253      } else {
254          // normalise axis
255          x = mPoints[0] / s;
256          y = mPoints[1] / s;
257          z = mPoints[2] / s;
258      }
259
260      output.mPoints[0] = x;
261      output.mPoints[1] = y;
262      output.mPoints[2] = z;
263      output.mPoints[3] = angle;
264  }
265
266  /**

```



```

267     * Returns the heading, attitude and bank of this <code>Quaternion</code>
268     * as euler angles in the double array respectively.
269     *
270     * @return an array of size 3 containing the euler angles for this
271     *         <code>Quaternion</code>.
272     */
273     public double[] toEulerAngles() {
274         double[] ret = new double[3];
275
276         ret[0] = Math.atan2(2 * mPoints[1] * getW() - 2 * mPoints[0] * mPoints[2], 1 - 2 *
(mPoints[1] * mPoints[1]) - 2 * (mPoints[2] * mPoints[2]));
277         ret[1] = Math.asin(2 * mPoints[0] * mPoints[1] + 2 * mPoints[2] * getW());
278         ret[2] = Math.atan2(2 * mPoints[0] * getW() - 2 * mPoints[1] * mPoints[2], 1 - 2 *
(mPoints[0] * mPoints[0]) - 2 * (mPoints[2] * mPoints[2]));
279         return ret;
280     }
281
282     private void loadIdentityQuat() {
283         this.mDirty = true;
284         setX(0);
285         setY(0);
286         setZ(0);
287         setW(1);
288     }
289
290     /**
291     * Returns a <code>String</code> representing this
292     * <code>Quaternion</code>. This overrides the
293     * {@link Vector4f#toString()} method.
294     *
295     * @return <code>String</code> representing this
296     *         <code>Quaternion</code>.
297     */
298     @Override
299     public String toString() {
300         return "{X: " + getX() + ", Y:" + getY() + ", Z:" + getZ() + ", W:" + getW() + "}";
301     }
302
303     private void generateQuaternionFromMatrix() {
304
305         float qx;
306         float qy;
307         float qz;
308         float qw;
309
310         float[] mat = mMatrix.getMatrix();
311         int[] indices;
312
313         if (this.mMatrix.size() == 16) {
314             if (this.mMatrix.isColumnMajor()) {
315                 indices = MatrixF4x4.MAT_IND_COL16_3X3;
316             } else {
317                 indices = MatrixF4x4.MAT_IND_ROW16_3X3;
318             }
319         } else {
320             if (this.mMatrix.isColumnMajor()) {
321                 indices = MatrixF4x4.MAT_IND_COL9_3X3;
322             } else {
323                 indices = MatrixF4x4.MAT_IND_ROW9_3X3;
324             }
325         }
326
327         int m00 = indices[0];
328         int m01 = indices[1];
329         int m02 = indices[2];
330
331         int m10 = indices[3];
332         int m11 = indices[4];
333         int m12 = indices[5];
334
335         int m20 = indices[6];
336         int m21 = indices[7];
337         int m22 = indices[8];
338
339         float tr = mat[m00] + mat[m11] + mat[m22];
340         if (tr > 0) {
341             float s = (float) Math.sqrt(tr + 1.0) * 2;
342             qw = 0.25f * s;
343             qx = (mat[m21] - mat[m12]) / s;
344             qy = (mat[m02] - mat[m20]) / s;
345             qz = (mat[m10] - mat[m01]) / s;
346         } else if ((mat[m00] > mat[m11]) & (mat[m00] > mat[m22])) {
347             float s = (float) Math.sqrt(1.0 + mat[m00] - mat[m11] - mat[m22]) * 2;
348             qw = (mat[m21] - mat[m12]) / s;
349             qx = 0.25f * s;
350             qy = (mat[m01] + mat[m10]) / s;
351             qz = (mat[m02] + mat[m20]) / s;
352         } else if (mat[m11] > mat[m22]) {
353             float s = (float) Math.sqrt(1.0 + mat[m11] - mat[m00] - mat[m22]) * 2;
354             qw = (mat[m02] - mat[m20]) / s;
355             qx = (mat[m01] + mat[m10]) / s;
356             qy = 0.25f * s;
357             qz = (mat[m12] + mat[m21]) / s;
358         } else {
359             float s = (float) Math.sqrt(1.0 + mat[m22] - mat[m00] - mat[m11]) * 2;

```

```

360         qw = (mat[m10] - mat[m01]) / s;
361         qx = (mat[m02] + mat[m20]) / s;
362         qy = (mat[m12] + mat[m21]) / s;
363         qz = 0.25f * s;
364     }
365
366     setX(qx);
367     setY(qy);
368     setZ(qz);
369     setW(qw);
370 }
371
372 public void setColumnMajor(float[] mMatrix) {
373     this.mMatrix.setMatrix(mMatrix);
374     this.mMatrix.setColumnMajor(true);
375
376     generateQuaternionFromMatrix();
377 }
378
379 public void setRowMajor(float[] mMatrix) {
380     this.mMatrix.setMatrix(mMatrix);
381     this.mMatrix.setColumnMajor(false);
382
383     generateQuaternionFromMatrix();
384 }
385
386 public void setEulerAngle(float azimuth, float pitch, float roll) {
387     double heading = Math.toRadians(roll);
388     double attitude = Math.toRadians(pitch);
389     double bank = Math.toRadians(azimuth);
390
391     double c1 = Math.cos(heading / 2);
392     double s1 = Math.sin(heading / 2);
393     double c2 = Math.cos(attitude / 2);
394     double s2 = Math.sin(attitude / 2);
395     double c3 = Math.cos(bank / 2);
396     double s3 = Math.sin(bank / 2);
397     double clc2 = c1 * c2;
398     double sls2 = s1 * s2;
399     setW((float) (clc2 * c3 - sls2 * s3));
400     setX((float) (clc2 * s3 + sls2 * c3));
401     setY((float) (s1 * c2 * c3 + c1 * s2 * s3));
402     setZ((float) (c1 * s2 * c3 - s1 * c2 * s3));
403
404     mDirty = true;
405 }
406
407 public void setAxisAngle(Vector3f vec, float rot) {
408     double s = Math.sin(Math.toRadians(rot / 2));
409     setX(vec.getX() * (float) s);
410     setY(vec.getY() * (float) s);
411     setZ(vec.getZ() * (float) s);
412     setW((float) Math.cos(Math.toRadians(rot / 2)));
413
414     mDirty = true;
415 }
416
417 public void setAxisAngleRad(Vector3f vec, double rot) {
418     double s = rot / 2;
419     setX(vec.getX() * (float) s);
420     setY(vec.getY() * (float) s);
421     setZ(vec.getZ() * (float) s);
422     setW((float) rot / 2);
423
424     mDirty = true;
425 }
426
427 public MatrixF4x4 getMatrix4x4() {
428     if (mDirty) {
429         convertQuatToMatrix();
430         mDirty = false;
431     }
432     return this.mMatrix;
433 }
434
435 public void copyFromVec3(Vector3f vec, float w) {
436     copyFromV3f(vec, w);
437 }
438
439 /**
440  * Computes a linear interpolation between this Quaternion and the
441  * input Quaternion.
442  *
443  * @param input the Quaternion to be slerped with this.
444  * @param output the Quaternion to store the result.
445  * @param t the ratio between the two quaternions where 0 <= t <= 1.0 .
446  * Note: Increase value of t will bring
447  * rotation closer to the input Quaternion.
448  */
449 public void slerp(Quaternion input, Quaternion output, float t) {
450     // Calculate angle between them
451     Quaternion bufferQuat;

```

```

455         float cosHalftheta = this.dotProduct(input);
456
457         if (cosHalftheta < 0) {
458             if(mTmpQuaternion == null) mTmpQuaternion = new Quaternion();
459             bufferQuat = mTmpQuaternion;
460             cosHalftheta = -cosHalftheta;
461             bufferQuat.mPoints[0] = (-input.mPoints[0]);
462             bufferQuat.mPoints[1] = (-input.mPoints[1]);
463             bufferQuat.mPoints[2] = (-input.mPoints[2]);
464             bufferQuat.mPoints[3] = (-input.mPoints[3]);
465         } else {
466             bufferQuat = input;
467         }
468
469         if (Math.abs(cosHalftheta) >= 1.0) {
470             output.mPoints[0] = (this.mPoints[0]);
471             output.mPoints[1] = (this.mPoints[1]);
472             output.mPoints[2] = (this.mPoints[2]);
473             output.mPoints[3] = (this.mPoints[3]);
474         } else {
475             double sinHalfTheta = Math.sqrt(1.0 - cosHalftheta * cosHalftheta);
476             double halfTheta = Math.acos(cosHalftheta);
477
478             double ratioA = Math.sin((1 - t) * halfTheta) / sinHalfTheta;
479             double ratioB = Math.sin(t * halfTheta) / sinHalfTheta;
480
481             //Calculate Quaternion
482             output.mPoints[3] = ((float) (mPoints[3] * ratioA + bufferQuat.mPoints[3] *
ratioB));
483             output.mPoints[0] = ((float) (this.mPoints[0] * ratioA + bufferQuat.mPoints[0] *
ratioB));
484             output.mPoints[1] = ((float) (this.mPoints[1] * ratioA + bufferQuat.mPoints[1] *
ratioB));
485             output.mPoints[2] = ((float) (this.mPoints[2] * ratioA + bufferQuat.mPoints[2] *
ratioB));
486         }
487     }
488
489     /**
490     * Rotates a 3-D vector by this <code>Quaternion</code>.
491     *
492     * @param v the {@link Vector3f} to be rotated.
493     * @return the rotated {@link Vector3f}.
494     */
495     public Vector3f rotateVector(Vector3f v) {
496         float q0 = this.mPoints[3];
497         float q1 = this.mPoints[0];
498         float q2 = this.mPoints[1];
499         float q3 = this.mPoints[2];
500         float v1 = v.getX();
501         float v2 = v.getY();
502         float v3 = v.getZ();
503         float x = (1-2*q2*q2-2*q3*q3)*v1 + 2*(q1*q2+q0*q3)*v2 + 2*(q1*q3-q0*q2)*v3;
504         float y = 2*(q1*q2-q0*q3)*v1 + (1-2*q1*q1-2*q3*q3)*v2 + 2*(q2*q3+q0*q1)*v3;
505         float z = 2*(q1*q3-q0*q2) + (q2*q3-q0*q1)*v2 + (1-2*q1*q1-2*q2*q2)*v3;
506         return new Vector3f(x,y,z);
507     }
508 }

```

Vector3f.java

```

package project.android.sensor.representation;

/**
 * Class to represent a three dimensional vector.
 *
 * <p>
 * It is defined as a point <i>(x,y,z)</i> in three dimensional
 * vector space. It is stored as a array of <code>float</code>s.
 * The three dimensions are known as X,Y and Z.
 * </p>
 * <p>
 * <b>Dot product of two vectors : </b> <br/>
 * <i>
 * ( $a_1$ , $b_1$ , $c_1$ ) .
 * ( $a_2$ , $b_2$ , $c_2$ ) =
 *  $a_1a_2 + b_1b_2 + c_1c_2$ 
 * </i>.
 * </p>
 * <p>
 * <b>Product of a vector with a scalar : </b> <br/>
 * <i>(a, b, c) * s = (a * s, b * s, c * s)</i>.
 * </p>
 * <p>
 * <b>Normalization of a vector</b> <i>(a,b,c)</i> <br/>
 * <math>\frac{1}{\sqrt{a^2+b^2+c^2}}(a,b,c)</math>
 * where, <br/>
 * <math>\text{mag} = \sqrt{a^2+b^2+c^2}</math>
 * </p>
 * @see project.android.sensor.representation.Vector3f
 * @see project.android.sensor.representation.Quaternion
 */

public class Vector3f {

    private float[] mPoints = new float[3];

    /**
     * Constructor.
     *
     * Creates the <code>Vector3f (x,y,z)</code>.
     *
     * @param x the X-dimension.
     * @param y the Y-dimension.
     * @param z the Z-dimension.
     */
    public Vector3f(float x, float y, float z) {
        this.mPoints[0] = x;
        this.mPoints[1] = y;
        this.mPoints[2] = z;
    }

    public Vector3f(float value) {
        this.mPoints[0] = value;
        this.mPoints[1] = value;
        this.mPoints[2] = value;
    }

    /**
     * Constructor.
     *
     * Default constructor.
     */
    public Vector3f() {
    }

    /**
     * Constructor.
     *
     * Copy constructor. Copies another <code>Vector3f</code> to
     * this one.
     *
     * @param vector the <code>Vector3f</code> to be copied.
     */
    public Vector3f(Vector3f vector) {
        this.mPoints[0] = vector.mPoints[0];
        this.mPoints[1] = vector.mPoints[1];
        this.mPoints[2] = vector.mPoints[2];
    }

    public Vector3f(Vector4f vector) {
        if (vector.getW() != 0) {
            this.mPoints[0] = vector.getX() / vector.getW();
            this.mPoints[1] = vector.getY() / vector.getW();
            this.mPoints[2] = vector.getZ() / vector.getW();
        } else {

```

```

90         this.mPoints[0] = vector.getX();
91         this.mPoints[1] = vector.getY();
92         this.mPoints[2] = vector.getZ();
93     }
94 }
95
96 /**
97  * Returns the array representation of this <code>Vector3f</code>.
98  *
99  * @return the array <code>[ X, Y, Z ]</code>.
100 */
101 float[] toArray() {
102     return this.mPoints;
103 }
104
105 public void add(Vector3f summand) {
106     this.mPoints[0] += summand.mPoints[0];
107     this.mPoints[1] += summand.mPoints[1];
108     this.mPoints[2] += summand.mPoints[2];
109 }
110
111 public void add(float summand) {
112     this.mPoints[0] += summand;
113     this.mPoints[1] += summand;
114     this.mPoints[2] += summand;
115 }
116
117 public void subtract(Vector3f subtrahend) {
118     this.mPoints[0] -= subtrahend.mPoints[0];
119     this.mPoints[1] -= subtrahend.mPoints[1];
120     this.mPoints[2] -= subtrahend.mPoints[2];
121 }
122
123 /**
124  * Multiplies this <code>Vector3f</code> with a scalar.
125  *
126  * @param scalar the scalar.
127 */
128 public void multiplyByScalar(float scalar) {
129     this.mPoints[0] *= scalar;
130     this.mPoints[1] *= scalar;
131     this.mPoints[2] *= scalar;
132 }
133
134 /**
135  * Normalizes this <code>Vector3f</code>.
136 */
137 public void normalize() {
138     double a = Math.sqrt(mPoints[0] * mPoints[0] + mPoints[1] * mPoints[1] + mPoints[2]
139 * mPoints[2]);
140     this.mPoints[0] = (float) (this.mPoints[0] / a);
141     this.mPoints[1] = (float) (this.mPoints[1] / a);
142     this.mPoints[2] = (float) (this.mPoints[2] / a);
143 }
144
145 /**
146  * Returns the X-dimension of this <code>Vector3f</code>.
147  *
148  * @return the X-dimension.
149 */
150 public float getX() {
151     return mPoints[0];
152 }
153
154 /**
155  * Returns the Y-dimension of this <code>Vector3f</code>.
156  *
157  * @return the Y-dimension.
158 */
159 public float getY() {
160     return mPoints[1];
161 }
162
163 /**
164  * Returns the Z-dimension of this <code>Vector3f</code>.
165  *
166  * @return the Z-dimension.
167 */
168 public float getZ() {
169     return mPoints[2];
170 }
171
172 /**
173  * Sets the X-dimension of this <code>Vector3f</code>.
174  *
175  * @param x the X-dimension.
176 */
177 void setX(float x) {
178     this.mPoints[0] = x;
179 }
180
181 /**
182  * Sets the Y-dimension of this <code>Vector3f</code>.
183

```

```

184     *
185     * @param y the Y-dimension.
186     */
187     void setY(float y) {
188         this.mPoints[1] = y;
189     }
190
191     /**
192     * Sets the Z-dimension of this <code>Vector3f</code>.
193     *
194     * @param z the Z-dimension.
195     */
196     void setZ(float z) {
197         this.mPoints[2] = z;
198     }
199
200     /**
201     * Sets this <code>Vector3f</code> as <code>(x,y,z)</code>.
202     *
203     * @param x the X-dimension.
204     * @param y the Y-dimension.
205     * @param z the Z-dimension.
206     */
207     public void setXYZ(float x, float y, float z) {
208         this.mPoints[0] = x;
209         this.mPoints[1] = y;
210         this.mPoints[2] = z;
211     }
212
213     /**
214     * Performs the dot product of two vectors.
215     *
216     * @param inputVec the <code>Vector3f</code> to be multiplied.
217     * @return result of <code>(this . inputVec)</code>.
218     */
219     public float dotProduct(Vector3f inputVec) {
220         return mPoints[0] * inputVec.mPoints[0] + mPoints[1] * inputVec.mPoints[1] +
221             mPoints[2] * inputVec.mPoints[2];
222     }
223
224     public void crossProduct(Vector3f inputVec, Vector3f outputVec) {
225         outputVec.setX(mPoints[1] * inputVec.mPoints[2] - mPoints[2] * inputVec.mPoints[1]);
226         outputVec.setY(mPoints[2] * inputVec.mPoints[0] - mPoints[0] * inputVec.mPoints[2]);
227         outputVec.setZ(mPoints[0] * inputVec.mPoints[1] - mPoints[1] * inputVec.mPoints[0]);
228     }
229
230     public float getLength() {
231         return (float) Math.sqrt(mPoints[0] * mPoints[0] + mPoints[1] * mPoints[1] +
232             mPoints[2] * mPoints[2]);
233     }
234
235     /**
236     * Returns the <code>String</code> representation of this <code>
237     * Vector3f</code>.
238     *
239     * It overrides the {@link java.lang.Object#toString()} method.
240     *
241     * @return <code>String</code> representing this
242     *         <code>Vector3f</code>.
243     */
244     @Override
245     public String toString() {
246         return "X:" + mPoints[0] + " Y:" + mPoints[1] + " Z:" + mPoints[2];
247     }
248
249     public void set(Vector3f source) {
250         set(source.mPoints);
251     }
252
253     private void set(float[] source) {
254         System.arraycopy(source, 0, mPoints, 0, 3);
255     }

```

Vector4f.java

```

1 package project.android.sensor.representation;
2
3 /**
4  * Class to represent a four dimensional vector.
5  *
6  * <p>
7  * It is defined as a point <i>(x,y,z,w)</i> in four dimensional
8  * vector space. It is stored as a array of <code>float</code>s.
9  * The four dimensions are known as X,Y,Z and W.
10 * </p>
11 * <p>
12 * <b>Dot product of two vectors : </b><br/>
13 * <i>
14 *  $(a_{<sub>1</sub>}, b_{<sub>1</sub>}, c_{<sub>1</sub>}, d_{<sub>1</sub>}) \cdot$ 
15 *  $(a_{<sub>2</sub>}, b_{<sub>2</sub>}, c_{<sub>2</sub>}, d_{<sub>2</sub>}) =$ 
16 *  $a_{<sub>1</sub>}a_{<sub>2</sub>} + b_{<sub>1</sub>}b_{<sub>2</sub>} +$ 
17 *  $c_{<sub>1</sub>}c_{<sub>2</sub>} + d_{<sub>1</sub>}d_{<sub>2</sub>}$ 
18 * </i>.
19 * </p>
20 * <p>
21 * <b>Product of a vector with a scalar : </b><br/>
22 * <i> $(a, b, c, d) * s = (a * s, b * s, c * s, d * s)$ </i>.
23 * </p>
24 * <p>
25 * <b>Normalization of a vector</b> <i>(a,b,c,d)</i> <br/>
26 *  $\frac{(a,b,c,d)}{\sqrt{a^2+b^2+c^2+d^2}}$ .
27 * = <i> $(\frac{a}{mag}, \frac{b}{mag}, \frac{c}{mag}, \frac{d}{mag})$ </i>, <br/>
28 * where, <br/> mag =
29 *  $\sqrt{a^2+b^2+c^2+d^2}$ .
30 * </p>
31 */
32 @see project.android.sensor.representation.Vector3f
33 @see project.android.sensor.representation.Quaternion
34 */
35 public class Vector4f {
36     float mPoints[] = { 0, 0, 0, 0 };
37
38     /**
39      * Constructor.
40      * Creates the <code>Vector4f (x,y,z,w)</code>.
41      * @param x the X-dimension.
42      * @param y the Y-dimension.
43      * @param z the Z-dimension.
44      * @param w the W-dimension.
45      */
46     public Vector4f(float x, float y, float z, float w) {
47         this.mPoints[0] = x;
48         this.mPoints[1] = y;
49         this.mPoints[2] = z;
50         this.mPoints[3] = w;
51     }
52
53     /**
54      * Constructor.
55      * Default constructor. Creates the <code>Vector4f (0,0,0,0)</code>.
56      */
57     public Vector4f() {
58         this.mPoints[0] = 0;
59         this.mPoints[1] = 0;
60         this.mPoints[2] = 0;
61         this.mPoints[3] = 0;
62     }
63
64     /**
65      * Constructor.
66      * Creates the <code>Vector4f (vector3f.getX(), vector3f.getY(),
67      * vector3f.getZ(), w)</code>.
68      * @param vector3f a {@link Vector3f}.
69      * @param w the w-dimension.
70      */
71     public Vector4f(Vector3f vector3f, float w) {
72         this.mPoints[0] = vector3f.getX();
73         this.mPoints[1] = vector3f.getY();
74         this.mPoints[2] = vector3f.getZ();
75         this.mPoints[3] = w;
76     }
77
78     /**
79      * Returns the array representation of this <code>Vector4f</code>.
80      * @return the array <code>[ X, Y, Z, W ]</code>.
81      */
82 
```

```

90     public float[] array() {
91         return mPoints;
92     }
93
94     /**
95     * Copies another Vector4f to this one.
96     * @param vec the Vector4f to be copied.
97     */
98     void copyVec4(Vector4f vec) {
99         this.mPoints[0] = vec.mPoints[0];
100        this.mPoints[1] = vec.mPoints[1];
101        this.mPoints[2] = vec.mPoints[2];
102        this.mPoints[3] = vec.mPoints[3];
103    }
104
105    public void add(Vector4f vector) {
106        this.mPoints[0] += vector.mPoints[0];
107        this.mPoints[1] += vector.mPoints[1];
108        this.mPoints[2] += vector.mPoints[2];
109        this.mPoints[3] += vector.mPoints[3];
110    }
111
112    public void add(Vector3f vector, float w) {
113        this.mPoints[0] += vector.getX();
114        this.mPoints[1] += vector.getY();
115        this.mPoints[2] += vector.getZ();
116        this.mPoints[3] += w;
117    }
118
119    public void subtract(Vector4f vector) {
120        this.mPoints[0] -= vector.mPoints[0];
121        this.mPoints[1] -= vector.mPoints[1];
122        this.mPoints[2] -= vector.mPoints[2];
123        this.mPoints[3] -= vector.mPoints[3];
124    }
125
126    public void subtract(Vector4f vector, Vector4f output) {
127        output.setXYZW(this.mPoints[0] - vector.mPoints[0], this.mPoints[1] -
128        vector.mPoints[1], this.mPoints[2] - vector.mPoints[2], this.mPoints[3] - vector.mPoints[3]);
129    }
130
131    public void subdivide(Vector4f vector) {
132        this.mPoints[0] /= vector.mPoints[0];
133        this.mPoints[1] /= vector.mPoints[1];
134        this.mPoints[2] /= vector.mPoints[2];
135        this.mPoints[3] /= vector.mPoints[3];
136    }
137
138    /**
139    * Multiplies this Vector4f with a scalar.
140    * @param scalar the scalar.
141    */
142    public void multiplyByScalar(float scalar) {
143        this.mPoints[0] *= scalar;
144        this.mPoints[1] *= scalar;
145        this.mPoints[2] *= scalar;
146        this.mPoints[3] *= scalar;
147    }
148
149    /**
150    * Performs the dot product of two vectors.
151    * @param input the Vector4f to be multiplied.
152    * @return result of (this . input).
153    */
154    float dotProduct(Vector4f input) {
155        return this.mPoints[0] * input.mPoints[0] + this.mPoints[1] * input.mPoints[1] +
156        this.mPoints[2] * input.mPoints[2] + this.mPoints[3] * input.mPoints[3];
157    }
158
159    public void lerp(Vector4f input, Vector4f output, float t) {
160        output.mPoints[0] = (mPoints[0] * (1.0f * t) + input.mPoints[0] * t);
161        output.mPoints[1] = (mPoints[1] * (1.0f * t) + input.mPoints[1] * t);
162        output.mPoints[2] = (mPoints[2] * (1.0f * t) + input.mPoints[2] * t);
163        output.mPoints[3] = (mPoints[3] * (1.0f * t) + input.mPoints[3] * t);
164    }
165
166    /**
167    * Normalizes this Vector4f.
168    */
169    public void normalize() {
170        if (mPoints[3] == 0)
171            return;
172
173        mPoints[0] /= mPoints[3];
174        mPoints[1] /= mPoints[3];
175        mPoints[2] /= mPoints[3];
176
177        double a = Math.sqrt(this.mPoints[0] * this.mPoints[0] + this.mPoints[1] *
178        this.mPoints[1] + this.mPoints[2]

```



```

182         * this.mPoints[2]);
183         mPoints[0] = (float) (this.mPoints[0] / a);
184         mPoints[1] = (float) (this.mPoints[1] / a);
185         mPoints[2] = (float) (this.mPoints[2] / a);
186     }
187
188     /**
189     * Returns the X-dimension of this <code>Vector4f</code>.
190     *
191     * @return the X-dimension.
192     */
193     public float getX() {
194         return this.mPoints[0];
195     }
196
197     /**
198     * Returns the Y-dimension of this <code>Vector4f</code>.
199     *
200     * @return the Y-dimension.
201     */
202     public float getY() {
203         return this.mPoints[1];
204     }
205
206     /**
207     * Returns the Z-dimension of this <code>Vector4f</code>.
208     *
209     * @return the Z-dimension.
210     */
211     public float getZ() {
212         return this.mPoints[2];
213     }
214
215     /**
216     * Returns the W-dimension of this <code>Vector4f</code>.
217     *
218     * @return the W-dimension.
219     */
220     public float getW() {
221         return this.mPoints[3];
222     }
223
224     /**
225     * Sets the X-dimension of this <code>Vector4f</code>.
226     *
227     * @param x the X-dimension.
228     */
229     public void setX(float x) {
230         this.mPoints[0] = x;
231     }
232
233     /**
234     * Sets the Y-dimension of this <code>Vector4f</code>.
235     *
236     * @param y the Y-dimension.
237     */
238     public void setY(float y) {
239         this.mPoints[1] = y;
240     }
241
242     /**
243     * Sets the Z-dimension of this <code>Vector4f</code>.
244     *
245     * @param z the Z-dimension.
246     */
247     public void setZ(float z) {
248         this.mPoints[2] = z;
249     }
250
251     /**
252     * Sets the W-dimension of this <code>Vector4f</code>.
253     *
254     * @param w the X-dimension.
255     */
256     public void setW(float w) {
257         this.mPoints[3] = w;
258     }
259
260     /**
261     * Sets all four dimensions of this <code>Vector4f</code>.
262     */
263     private void setXYZW(float x, float y, float z, float w) {
264         this.mPoints[0] = x;
265         this.mPoints[1] = y;
266         this.mPoints[2] = z;
267         this.mPoints[3] = w;
268     }
269
270     public boolean compareTo(Vector4f rhs) {
271         boolean ret = false;
272         if (this.mPoints[0] == rhs.mPoints[0] && this.mPoints[1] == rhs.mPoints[1] &&
this.mPoints[2] == rhs.mPoints[2]
&& this.mPoints[3] == rhs.mPoints[3])
273             ret = true;
274         return ret;
275     }

```

```

276     }
277
278
279     /**
280     * Copies a {@link Vector3f} to this <code>Vector4f</code> using
281     * a separate value for W-dimension.
282     *
283     * @param input the {@link Vector3f}.
284     * @param w <code>float</code> representing the W-dimension.
285     */
286     void copyFromV3f(Vector3f input, float w) {
287         mPoints[0] = (input.getX());
288         mPoints[1] = (input.getY());
289         mPoints[2] = (input.getZ());
290         mPoints[3] = (w);
291     }
292
293     /**
294     * Returns the <code>String</code> representation of this <code>
295     * Vector4f</code>.
296     *
297     * It overrides the {@link java.lang.Object#toString()} method.
298     *
299     * @return <code>String</code> representing this
300     *         <code>Vector4f</code>.
301     */
302     @Override
303     public String toString() {
304         return "X:" + mPoints[0] + " Y:" + mPoints[1] + " Z:" + mPoints[2] + " W:" +
305             mPoints[3];
306     }

```

AboutFragment.java

```
1 package project.android;
2
3 import android.content.pm.ActivityInfo;
4 import android.os.Bundle;
5 import android.support.v4.app.Fragment;
6 import android.view.LayoutInflater;
7 import android.view.View;
8 import android.view.ViewGroup;
9 import android.webkit.WebSettings;
10 import android.webkit.WebView;
11
12 /**
13  * Class representing the <code>Fragment</code> for displaying app information.
14  *
15  */
16 public class AboutFragment extends Fragment {
17
18     /**
19      * Overrides the
20      * <code>Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)</code>
21      * method of the Android API.
22      *
23      * Called to have the fragment instantiate its user interface view.
24      *
25      * @param inflater The <code>LayoutInflater</code> object that can be used to
26      *                 inflate any views in the fragment.
27      * @param container If non-null, this is the parent view that the fragment's
28      *                 UI should be attached to. The fragment should not add the
29      *                 view itself, but this can be used to generate the
30      *                 <code>LayoutParams</code> of the view.
31      * @param savedInstanceState If non-null, this fragment is being re-constructed
32      *                 from a previous saved state as given here.
33      * @return the <code>View</code> for the fragment's UI, or <code>null</code>.
34      */
35     @Override
36     public View onCreateView(LayoutInflater inflater, ViewGroup container,
37                             Bundle savedInstanceState) {
38         View view = inflater.inflate(R.layout.fragment_about, container, false);
39         WebView mWebView = (WebView) view.findViewById(R.id.webView);
40         mWebView.loadUrl("file:///android_asset/about.html");
41         WebSettings webSettings = mWebView.getSettings();
42         webSettings.setJavaScriptEnabled(false);
43         return view;
44     }
45
46     /**
47      * Overrides the <code>Fragment.onResume()</code> method
48      * of the Android API.
49      *
50      * Called when the activity will start interacting with the user.
51      */
52     @Override
53     public void onResume() {
54         super.onResume();
55         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
56     }
57
58     /**
59      * Overrides the <code>Fragment.onPause()</code> method
60      * of the Android API.
61      *
62      * Called when the system is about to start resuming a previous
63      * activity.
64      */
65     @Override
66     public void onPause() {
67         super.onPause();
68         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
69     }
70 }
```

ConnectionFragment.java

```
1 package project.android;
2
3 import android.Manifest;
4 import android.content.Context;
5 import android.content.Intent;
6 import android.content.pm.ActivityInfo;
7 import android.content.pm.PackageManager;
8 import android.net.wifi.WifiManager;
9 import android.os.Build;
10 import android.os.Bundle;
11 import android.support.annotation.NonNull;
12 import android.support.v4.app.ActivityCompat;
13 import android.support.v4.app.Fragment;
14 import android.support.v4.content.ContextCompat;
15 import android.support.v7.app.AlertDialog;
16 import android.text.InputFilter;
17 import android.text.InputType;
18 import android.text.method.PasswordTransformationMethod;
19 import android.view.LayoutInflater;
20 import android.view.View;
21 import android.view.ViewGroup;
22 import android.widget.AdapterView;
23 import android.widget.EditText;
24 import android.widget.ImageView;
25 import android.widget.ListView;
26 import android.widget.TextView;
27 import android.widget.Toast;
28
29 import java.io.IOException;
30 import java.util.ArrayList;
31 import java.util.HashMap;
32
33 import project.android.net.Client;
34 import project.android.net.BroadcastReceiverThread;
35 import project.android.net.ConnectionTask;
36 import project.android.net.NetworkService;
37 import project.android.net.ServerInfo;
38
39 /**
40  * Class representing the <code>Fragment</code> that provides GUI frontend for
41  * the network module.
42  *
43  * <p>
44  * It performs the following operations:
45  * <ul>
46  * <li>
47  * Displays the list of available servers with appropriate icons. It
48  * maintains a list of servers available.
49  * </li>
50  * <li>
51  * Displays an alert dialog upon selecting a server from the list to
52  * input the pairing key as shown in the PC.
53  * </li>
54  * <li>
55  * Connects with the respective server upon successful authentication
56  * and starts the communication by creating a TCP socket. It also changes
57  * the icon of the selected server in the list.
58  * </li>
59  * </ul>
60  * </p>
61  *
62  * @see project.android.net.ServerInfo
63  * @see project.android.net.Client
64  * @see BroadcastReceiverThread
65  * @see project.android.net.ConnectionTask
66  */
67 public class ConnectionFragment extends Fragment {
68
69     private View mView;
70     private ListView mListView;
71     private TextView mTextView;
72     private CustomAdapter mCustomAdapter;
73     private ArrayList<ServerInfo> mNetworkList;
74     private AlertDialog mAlertDialog;
75     private boolean mInitDiscover;
76     private int mSelectedServerPos;
77
78     public ServerInfo mServerInfo;
79
80     private static final int REQUEST_INTERNET_ACCESS = 1001;
81     private static final int PAIRING_KEY_LENGTH = 6;
82
83     public static Client sSecuredClient;
84     public static ArrayList<ServerInfo> sSelectedServer = new ArrayList<>();
85     public static HashMap<String, Boolean> sConnectionAlive = new HashMap<>();
86
87     /**
88      * Overrides the
89      * <code>Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)</code>
```

```

90     * method of the Android API.
91     *
92     * Called to have the fragment instantiate its user interface view.
93     *
94     * @param inflater The <code>LayoutInflater</code> object that can be used
95     *               to inflate any views in the fragment.
96     * @param container If non-null, this is the parent view that the fragment's UI
97     *               should be attached to. The fragment should not add the view
98     *               itself, but this can be used to generate the
99     *               <code>LayoutParams</code> of the view.
100    * @param savedInstanceState If non-null, this fragment is being re-constructed
101    *               from a previous saved state as given here.
102    * @return the <code>View</code> for the fragment's UI, or <code>null</code>.
103    */
104    @Override
105    public View onCreateView(LayoutInflater inflater, ViewGroup container,
106                           Bundle savedInstanceState) {
107        // Inflate the layout for this fragment
108        if (mView == null) {
109            mView = inflater.inflate(R.layout.fragment_connect, container, false);
110            mNetworkList = new ArrayList<>();
111            mCustomAdapter = new CustomAdapter(getActivity(), R.layout.local_devices,
112            mNetworkList);
113            mListView = (ListView) mView.findViewById(R.id.listView);
114            mListView.setAdapter(mCustomAdapter);
115            mTextView = (TextView) getActivity().findViewById(R.id.device_conn);
116        }
117        return mView;
118    }
119    /**
120     * Overrides the <code>Fragment.onViewCreated(View, Bundle)</code>
121     * method of the Android API.
122     *
123     * Called immediately after {@link #onCreateView(LayoutInflater, ViewGroup, Bundle)}
124     * has returned, but before any saved state has been restored in to the view.
125     *
126     * @param view The <code>View</code> returned by
127     *               {@link #onCreateView(LayoutInflater, ViewGroup, Bundle)}.
128     * @param savedInstanceState If non-null, this fragment is being re-constructed
129     *               from a previous saved state as given here.
130     */
131    @Override
132    public void onViewCreated(View view, Bundle savedInstanceState) {
133        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
134            getInternetPermission();
135        }
136        if (!mInitDiscover) {
137            discoverLocalDevices();
138            mInitDiscover = true;
139        }
140        mListView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
141            @Override
142            public void onItemClick(AdapterView<?> adapterView, View view, int position,
143            long id) {
144                ServerInfo serverInfo = (ServerInfo)
145                adapterView.getAdapter().getItem(position);
146                if (serverInfo != null) {
147                    if (!sSelectedServer.contains(serverInfo)) {
148                        sSelectedServer.add(serverInfo);
149                        if (sSelectedServer.size() > 1) {
150                            Toast.makeText(getContext(), "Device already connected! " +
151                            "Disconnect to connect another device",
152                            Toast.LENGTH_SHORT).show();
153                            sSelectedServer.remove(serverInfo);
154                        } else {
155                            mSelectedServerPos = position;
156                            startCommunication(serverInfo);
157                        }
158                    } else disconnectDevice(position);
159                }
160            }
161        });
162    }
163    /**
164     * Overrides the <code>Fragment.onResume()</code> method
165     * of the Android API.
166     *
167     * Called when the activity will start interacting with
168     * the user.
169     */
170    @Override
171    public void onResume() {
172        super.onResume();
173        getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
174    }
175    /**
176     * Overrides the <code>Fragment.onPause()</code> method
177     * of the Android API.
178     *
179     * Called when the system is about to start resuming a
180     * previous activity.
181     */

```

```

181     @Override
182     public void onPause() {
183         super.onPause();
184         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
185     }
186
187     /**
188     * Override the
189     * <code>Fragment.onRequestPermissionsResult(int, String[], int[])</code>
190     * method of the Android API.
191     *
192     * @param requestCode the request code.
193     * @param permissions the requested permissions. Never <code>null</code>.
194     * @param grantResults the grant results for the corresponding permissions
195     *       which is either <code>PERMISSION_GRANTED</code> or
196     *       <code>PERMISSION_DENIED</code>. Never <code>null</code>.
197     */
198     @Override
199     public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
200     @NonNull int[] grantResults) {
201         super.onRequestPermissionsResult(requestCode, permissions, grantResults);
202         switch (requestCode) {
203             case REQUEST_INTERNET_ACCESS: {
204                 if (grantResults.length == 0 || grantResults[0] !=
205                 PackageManager.PERMISSION_GRANTED) {
206                     Toast.makeText(this.getActivity(), "Permission denied!",
207                     Toast.LENGTH_LONG).show();
208                 }
209             }
210         }
211
212     /**
213     * Returns the <code>View</code> of the clicked list item.
214     *
215     * @param pos position of the item.
216     * @return the <code>View</code> of the clicked list item.
217     */
218     public View getViewByPosition(int pos) {
219         final int firstListItemPosition = mListView.getFirstVisiblePosition();
220         final int lastListItemPosition = firstListItemPosition + mListView.getChildCount() -
221         1;
222         if (pos < firstListItemPosition || pos > lastListItemPosition) {
223             return mCustomAdapter.getView(pos, null, mListView);
224         } else {
225             final int childIndex = pos - firstListItemPosition;
226             return mListView.getChildAt(childIndex);
227         }
228     }
229
230     /**
231     * Returns a list of <code>ServerInfo</code> objects.
232     *
233     * @return a list of {@link project.android.net.ServerInfo}
234     *       objects.
235     * @see project.android.net.ServerInfo
236     */
237     ArrayList<ServerInfo> getNetworkList() { return mNetworkList; }
238
239     private void disconnectDevice(int position) {
240         getActivity().stopService(new Intent(getContext(), NetworkService.class));
241         resetIcon(position);
242         sSelectedServer.remove(0);
243     }
244
245     private void getInternetPermission() {
246         boolean hasPermission1 = (ContextCompat.checkSelfPermission(this.getContext(),
247         Manifest.permission.INTERNET) == PackageManager.PERMISSION_GRANTED);
248         boolean hasPermission2 = (ContextCompat.checkSelfPermission(this.getContext(),
249         Manifest.permission.CHANGE_WIFI_MULTICAST_STATE) ==
250         PackageManager.PERMISSION_GRANTED);
251         boolean hasPermission3 = (ContextCompat.checkSelfPermission(this.getContext(),
252         Manifest.permission.ACCESS_NETWORK_STATE) ==
253         PackageManager.PERMISSION_GRANTED);
254         if (!hasPermission1 || !hasPermission2 || !hasPermission3) {
255             ActivityCompat.requestPermissions(this.getActivity(),
256             new String[]{Manifest.permission.INTERNET,
257             Manifest.permission.CHANGE_WIFI_MULTICAST_STATE,
258             Manifest.permission.ACCESS_NETWORK_STATE},
259             REQUEST_INTERNET_ACCESS);
260         }
261     }
262
263     private void startCommunication (final ServerInfo serverInfo) {
264         new Thread(new Runnable() {
265             @Override
266             public void run() {
267                 try {
268                     sSecuredClient = new Client(serverInfo.getAddress());
269                 } catch (IOException ignored) {}
270             }
271         }).start();
272     }

```

```

270         EditText editText = new EditText(getActivity());
271         editText.setInputType(InputType.TYPE_TEXT_VARIATION_PASSWORD);
272         editText.setTransformationMethod(PassWordTransformationMethod.getInstance());
273         editText.setSelection(editText.getText().length());
274         editText.setHint("Password");
275         editText.setTextSize(18);
276         InputFilter[] FilterArray = new InputFilter[1];
277         FilterArray[0] = new InputFilter.LengthFilter(PAIRING_KEY_LENGTH);
278         editText.setFilters(FilterArray);
279
280         displayAlertDialog(editText, serverInfo);
281     }
282
283     private void displayAlertDialog(final EditText editText, final ServerInfo serverInfo) {
284         mAlertDialog = new AlertDialog.Builder(getContext())
285             .setView(editText)
286             .setTitle("Connect to PC")
287             .setMessage("Enter pairing key as shown in PC")
288             .setPositiveButton(R.string.Send, null)
289             .setNegativeButton(android.R.string.cancel, null)
290             .setCancelable(false)
291             .create();
292
293         mAlertDialog.show();
294         mAlertDialog.getButton(AlertDialog.BUTTON_POSITIVE).setOnClickListener(new
View.OnClickListener() {
295             @Override
296             public void onClick(View view) {
297                 String pairingKey = editText.getText().toString();
298                 serverInfo.setPairingKey(pairingKey);
299                 if(sSecuredClient != null) {
300                     mServerInfo = serverInfo;
301                     new ConnectionTask().execute(serverInfo);
302                 }
303                 mAlertDialog.dismiss();
304             }
305         });
306
307         mAlertDialog.getButton(AlertDialog.BUTTON_NEGATIVE).setOnClickListener(new
View.OnClickListener() {
308             @Override
309             public void onClick(View view) {
310                 if(sSecuredClient != null) {
311                     try {
312                         sSecuredClient.sendStopSignal(false);
313                         sSecuredClient.close();
314                         sSelectedServer.remove(serverInfo);
315                     } catch (IOException ignored) {}
316                     mAlertDialog.dismiss();
317                 }
318             }
319         });
320     }
321
322     private void discoverLocalDevices() {
323         WifiManager wifi = (WifiManager)
getContext().getApplicationContext().getSystemService(Context.WIFI_SERVICE);
324         WifiManager.MulticastLock lock = wifi.createMulticastLock("remouseMulticastLock");
325         BroadcastReceiverThread broadcastReceiverThread = new
BroadcastReceiverThread(getActivity(), lock);
326         new Thread(broadcastReceiverThread).start();
327     }
328
329     /**
330      * Adds the <code>ServerInfo</code> object of an active server
331      * to the list of servers.
332      *
333      * @param serverInfo the active {@link project.android.net.ServerInfo}
334      * object.
335      * @see project.android.net.ServerInfo
336      */
337     public void addItem(ServerInfo serverInfo) {
338         mNetworkList.add(serverInfo);
339         mCustomAdapter.notifyDataSetChanged();
340     }
341
342     /**
343      * Removes the <code>ServerInfo</code> object of an inactive server
344      * from the list of servers.
345      *
346      * @param serverInfo the inactive {@link project.android.net.ServerInfo}
347      * object.
348      * @see project.android.net.ServerInfo
349      */
350     public void removeItem(ServerInfo serverInfo) {
351         sSelectedServer.remove(serverInfo);
352         sConnectionAlive.put(serverInfo.getAddress(), false);
353         mNetworkList.remove(serverInfo);
354         mCustomAdapter.notifyDataSetChanged();
355     }
356
357     /**
358      * Sets an icon for an inactive server.
359      *
360      * @param serverInfo the {@link project.android.net.ServerInfo}

```

```

361         *           of the inactive server.
362         * @see project.android.net.ServerInfo
363         */
364     public void resetIcon(ServerInfo serverInfo) {
365         int position = mCustomAdapter.getPosition(serverInfo);
366         ImageView img = (ImageView)
367         getViewByPosition(position).findViewById(R.id.connectIcon);
368         img.setImageResource(R.mipmap.laptop_icon);
369         mTextView.setText(R.string.no_device_connected);
370     }
371     /**
372     * Sets an icon for an inactive server.
373     *
374     * @param position position of the inactive server in the list.
375     */
376     public void resetIcon(int position) {
377         ImageView img = (ImageView)
378         getViewByPosition(position).findViewById(R.id.connectIcon);
379         img.setImageResource(R.mipmap.laptop_icon);
380         mTextView.setText(R.string.no_device_connected);
381     }
382     /**
383     * Sets an icon for an active server.
384     */
385     public void setIcon() {
386         ImageView img = (ImageView)
387         getViewByPosition(mSelectedServerPos).findViewById(R.id.connectIcon);
388         img.setImageResource(R.mipmap.connect_icon);
389         ServerInfo serverInfo = mCustomAdapter.getItem(mSelectedServerPos);
390         if(serverInfo != null) {
391             String s = "\nConnected to:\n" + serverInfo.getServerInfo();
392             s += "\nIP: " + serverInfo.getAddress();
393             mTextView.setText(s);
394         }
395     }
396     /**
397     * Method to dismiss the alert dialog prompting the pairing key input.
398     */
399     public void dismissAlertDialog() { if(mAlertDialog != null) mAlertDialog.dismiss(); }
400 }

```


CustomAdapter.java

```
1 package project.android;
2
3 import android.app.Activity;
4 import android.support.annotation.NonNull;
5 import android.view.LayoutInflater;
6 import android.view.View;
7 import android.view.ViewGroup;
8 import android.widget.ArrayAdapter;
9 import android.widget.ImageView;
10 import android.widget.TextView;
11
12 import java.util.ArrayList;
13
14 import project.android.net.ServerInfo;
15
16 import static project.android.ConnectionFragment.sConnectionAlive;
17
18 /**
19  * Displays the list of available servers in a <code>ListView</code>
20  * format.
21  *
22  * <p>
23  * This <code>CustomAdapter</code> populates the <code>ListView</code>
24  * in the {@link ConnectionFragment}. List items consist of
25  * {@link project.android.net.ServerInfo} objects which are added/removed
26  * dynamically.
27  * </p>
28  *
29  * @see project.android.ConnectionFragment
30  * @see project.android.net.ServerInfo
31  */
32 class CustomAdapter extends ArrayAdapter<ServerInfo> {
33     private Activity mActivity;
34     private ConnectionFragment mConnectionFragment;
35
36     /**
37      * Constructor.
38      *
39      * Initializes this <code>CustomAdapter</code>.
40      *
41      * @param activity the current <code>android.app.Activity</code> object.
42      * @param textViewResourceId the resource ID for a layout file containing a
43      * <code>TextView</code> to use when instantiating views.
44      * @param arrayList the {@link project.android.net.ServerInfo} objects to be
45      * displayed in the <code>ListView</code>.
46      * @see project.android.net.ServerInfo
47      */
48     CustomAdapter(Activity activity, int textViewResourceId, ArrayList<ServerInfo>
49         arrayList) {
50         super(activity, textViewResourceId, arrayList);
51         mActivity = activity;
52         mConnectionFragment = (ConnectionFragment) MainActivity.getConnectionFragment();
53     }
54
55     private class ViewHolder {
56         private TextView mTextView;
57         private ImageView mImageView;
58     }
59
60     /**
61      * Overrides the <code>ArrayAdapter.getCount()</code>
62      * method.
63      *
64      * Returns the number of items in the data set represented by this adapter.
65      *
66      * @return the count of items.
67      */
68     @Override
69     public int getCount() {
70         super.getCount();
71         return mConnectionFragment.getNetworkList().size();
72     }
73
74     /**
75      * Overrides <code>ArrayAdapter.getView(int, View, ViewGroup)</code>
76      * method.
77      *
78      * Gets a <code>View</code> that displays the data at the specified
79      * position in the data set.
80      *
81      * @param position the position of the item within the adapter's data
82      * set of the item whose view is required.
83      * @param convertView The old view to reuse, if possible.
84      * @param parent The parent that this view will eventually be attached to.
85      * @return A <code>View</code> corresponding to the data at the specified
86      * position.
87      */
88     @NonNull
```

```

89 | @Override
90 | public View getView(int position, View convertView, @NonNull ViewGroup parent) {
91 |
92 |     ViewHolder viewHolder;
93 |     if (convertView == null) {
94 |         LayoutInflater inflater = LayoutInflater.from(mActivity);
95 |         convertView = inflater.inflate(R.layout.local_devices, parent, false);
96 |         viewHolder = new ViewHolder();
97 |         viewHolder.mTextView = (TextView) convertView.findViewById(R.id.code);
98 |         viewHolder.mImageView = (ImageView) convertView.findViewById(R.id.connectIcon);
99 |         convertView.setTag(viewHolder);
100 |     }
101 |     else {
102 |         viewHolder = (ViewHolder) convertView.getTag();
103 |     }
104 |
105 |     ArrayList<ServerInfo> serverList = mConnectionFragment.getNetworkList();
106 |     if(!serverList.isEmpty()) {
107 |         ServerInfo serverInfo = serverList.get(position);
108 |         viewHolder.mTextView.setText(serverInfo.getServerInfo() + " (" +
serverInfo.getAddress() + ")");
109 |         if(sConnectionAlive.get(serverInfo.getAddress())) {
110 |             viewHolder.mImageView.setImageResource(R.mipmap.connect_icon);
111 |         } else {
112 |             viewHolder.mImageView.setImageResource(R.mipmap.laptop_icon);
113 |         }
114 |     }
115 |     return convertView;
116 | }
117 | }

```

KeyboardFragment.java

```
1 package project.android;
2
3 import android.content.Context;
4 import android.content.pm.ActivityInfo;
5 import android.os.Bundle;
6 import android.support.v4.app.Fragment;
7 import android.text.Editable;
8 import android.text.TextWatcher;
9 import android.text.method.PasswordTransformationMethod;
10 import android.view.KeyEvent;
11 import android.view.LayoutInflater;
12 import android.view.View;
13 import android.view.ViewGroup;
14 import android.view.inputmethod.InputMethodManager;
15 import android.widget.Button;
16 import android.widget.CheckBox;
17 import android.widget.CompoundButton;
18 import android.widget.EditText;
19 import android.widget.ImageButton;
20
21 import project.android.net.KeyboardThread;
22
23 import static project.android.ConnectionFragment.sConnectionAlive;
24
25 /**
26  * Class representing the <code>Fragment</code> for providing the
27  * GUI frontend for the keyboard module.
28  *
29  * <p>
30  * The following keyboard features are provided:
31  * <ul>
32  * <li>Predictive text input.</li>
33  * <li>Auto-correct feature.</li>
34  * <li>Hide sensitive text.</li>
35  * <li>Support for special keys like Ctrl, Alt etc. </li>
36  * </ul>
37  * </p>
38  *
39  * @see project.android.net.KeyboardThread
40  */
41 public class KeyboardFragment extends Fragment implements View.OnKeyListener,
42     View.OnClickListener,
43     CompoundButton.OnCheckedChangeListener, TextWatcher {
44
45     private String mLastInput;
46     private InputMethodManager mInput;
47     private String mLastWord;
48     private KeyboardThread mKeyboardThread;
49     private EditText mKeyboardInput;
50
51     private static final int[] SPECIAL_KEY_ID = {
52         R.id.Ctrl, //0
53         R.id.Alt, //1
54         R.id.Shift, //2
55         R.id.Show_Hide, //3
56         R.id.Home, //4
57         R.id.End, //5
58         R.id.Page_Up, //6
59         R.id.Page_Down, //7
60         R.id.Del, //8
61         R.id.Esc, //9
62         R.id.Tab, //10
63         R.id.f1, //11
64         R.id.f2, //12
65         R.id.f3, //13
66         R.id.f4, //14
67         R.id.f5, //15
68         R.id.f6, //16
69         R.id.f7, //17
70         R.id.f8, //18
71         R.id.f9, //19
72         R.id.f10, //20
73         R.id.f11, //21
74         R.id.f12, //22
75         R.id.Insert, //23
76         R.id.upArrow, //24
77         R.id.downArrow, //25
78         R.id.leftArrow, //26
79         R.id.rightArrow //27
80     };
81
82     /**
83      * Overrides the
84      * <code>Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)</code>
85      * of the Android API.
86      *
87      * Called to have the fragment instantiate its user interface view.
88      *
89      * @param inflater The <code>LayoutInflater</code> object that can be
```

```

89      *          used to inflate any views in the fragment.
90      * @param container If non-null, this is the parent view that the
91      *          fragment's UI is attached to. The fragment should
92      *          not add the view itself, but this can be used to
93      *          generate the <code>LayoutParams</code> of the view.
94      * @param savedInstanceState If non-null, this fragment is being
95      *          re-constructed from a previous saved
96      *          state as given here.
97      * @return the <code>View</code> for the fragment's UI, or <code>null</code>.
98      */
99      @Override
100     public View onCreateView(LayoutInflater inflater, ViewGroup container,
101                             Bundle savedInstanceState) {
102         View view = inflater.inflate(R.layout.fragment_keyboard, container, false);
103         mInput = null;
104         mLastInput = null;
105         if(sConnectionAlive.containsValue(true)) {
106             mInput = (InputMethodManager)
107             getContext().getSystemService(Context.INPUT_METHOD_SERVICE);
108             mInput.toggleSoftInput (InputMethodManager.SHOW_FORCED,
109             InputMethodManager.HIDE_IMPLICIT_ONLY);
110             mKeyboardInput = (EditText) view.findViewById(R.id.keyboard);
111             mKeyboardInput.setCursorVisible(false);
112             mKeyboardInput.setLongClickable(false);
113             mKeyboardInput.setOnClickListener(this);
114             mKeyboardInput.setText("");
115             mKeyboardInput.setTextSize(18);
116             mKeyboardInput.setOnKeyListener(this);
117             mKeyboardInput.addTextChangedListener(this);
118             mKeyboardThread = new KeyboardThread();
119             new Thread(mKeyboardThread).start();
120             mLastWord = "";
121             setListeners(view);
122         }
123         return view;
124     }
125     /**
126     * Overrides the <code>Fragment.onDestroyView()</code> method
127     * of the Android API.
128     * Called when the view previously created by
129     * {@link #onCreateView(LayoutInflater, ViewGroup, Bundle)}
130     * has been detached from the fragment.
131     */
132     @Override
133     public void onDestroyView() {
134         super.onDestroyView();
135         View view = getView();
136         if(mInput != null && view != null) {
137             mInput.hideSoftInputFromWindow(view.getWindowToken(), 0);
138         }
139         if(mKeyboardThread != null) mKeyboardThread.setStopFlag();
140     }
141     /**
142     * Overrides the <code>Fragment.onResume()</code> method
143     * of the Android API.
144     * Called when the activity will start interacting with
145     * the user.
146     */
147     @Override
148     public void onResume() {
149         super.onResume();
150         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
151     }
152     /**
153     * Overrides the <code>Fragment.onPause()</code> method
154     * of the Android API.
155     * Called when the system is about to start resuming a
156     * previous activity.
157     */
158     @Override
159     public void onPause() {
160         super.onPause();
161         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
162     }
163     /**
164     * Overrides the
165     * <code>TextWatcher.beforeTextChanged(CharSequence, int, int, int)</code>
166     * method of the Android API.
167     *
168     * <p>
169     * Called to notify that, within <code>s</code>, the <code>count</code>
170     * characters beginning at <code>start</code> are about to be replaced
171     * by new text with length <code>after</code>. It is an error to attempt
172     * to make changes to <code>s</code> from this callback.
173     * </p>
174     * @param s the <code>CharSequence</code>.
175     * @param start start index of the current word.

```

```

182     * @param count number of characters to be replaced.
183     * @param after new length.
184     */
185     @Override
186     public void beforeTextChanged(CharSequence s, int start, int count, int after) {
187         // Log.d("Keyboard before", s.length()+" "+start+" "+count+" "+after);
188         mLastWord = s.subSequence(start, start+count).toString();
189         if(count>after) { //backspace ... less chars in future
190             Log.d("Keyboard before", count-after+" backspaces :
191             "+s.subSequence(start+after,start+count));
192             String data = "";
193             for (int i = 0; i < count - after; ++i) // send (count-after) backspaces.
194                 data = data + "\b";
195             mKeyboardThread.addToBuffer(data, false);
196         }
197     }
198 }
199
200 /**
201  * Overrides the
202  * <code>onTextChanged(CharSequence, int, int, int)</code>
203  * method of the Android API.
204  *
205  * <p>
206  * Called to notify that, within <code>s</code>, the <code>count</code>
207  * characters beginning at <code>start</code> have just replaced old
208  * text that had length <code>before</code>. It is an error to attempt
209  * to make changes to <code>s</code> from this callback.
210  * </p>
211  *
212  * @param s the <code>CharSequence</code>.
213  * @param start start index of the current word.
214  * @param before number of characters that are replaced.
215  * @param count new length.
216  */
217     @Override
218     public void onTextChanged(CharSequence s, int start, int before, int count) {
219         // Log.d("Keyboard on", s.length()+" "+start+" "+before+" "+count);
220         int diff = count - before;
221         if(diff>0) { // 's' contains more chars after text change
222             String currWord = s.subSequence(start, start+count).toString();
223             // insertion of chars
224             if(mLastWord.equals(s.subSequence(start, start+before).toString())) {
225                 String input = s.subSequence(start + before, start + count).toString();
226                 Log.d("Keyboard on", input);
227                 mKeyboardThread.addToBuffer(input, false);
228             }
229             // replace old with new text
230             else {
231                 String data = "";
232                 for(int i=0;i<before;++i)
233                     data = data + "\b";
234                 Log.d("Keyboard on", data+currWord);
235                 mKeyboardThread.addToBuffer((data + currWord), false);
236             }
237             mLastInput = (s.length() == 0) ? null : s.toString();
238         }
239     }
240 }
241
242     @Override
243     public void afterTextChanged(Editable s) {}
244
245     /**
246     * Overrides the
247     * <code>View.OnKeyListener.onKey(View, int, KeyEvent)</code>
248     * method of the Android API.
249     *
250     * Called when a hardware key is dispatched to a view. This allows
251     * listeners to get a chance to respond before the target view.
252     * This has been used for detecting back key down event.
253     *
254     * @param v the view to which the key has been dispatched.
255     * @param keyCode the code for the physical key that was pressed.
256     * @param event the <code>KeyEvent</code> object containing full
257     * information about the event.
258     * @return <code>true</code>, if the listener has consumed the event,<br/>
259     * <code>false</code>, otherwise.
260     */
261     @Override
262     public boolean onKey(View v, int keyCode, KeyEvent event) {
263         if(keyCode == KeyEvent.KEYCODE_DEL && mLastInput == null && event.getAction() ==
264         KeyEvent.ACTION_DOWN) {
265             mKeyboardThread.addToBuffer("\b", false);
266             return true;
267         }
268         return false;
269     }
270 }
271
272 /**
273  * Overrides the
274  * <code>CompoundButton.onCheckedChangeListener(CompoundButton, boolean)</code>

```

```

275     * method of the Android API.
276     *
277     * Called when the checked state of a compound button has changed.
278     *
279     * @param buttonView the <code>CompoundButton</code> view whose state
280     *                   has changed.
281     * @param isChecked the new checked state of <code>buttonView</code>.
282     */
283     @Override
284     public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
285         String data = "";
286         switch (buttonView.getId()) {
287             case R.id.Ctrl:
288                 if (isChecked) data = "Ctrl_On";
289                 else data = "Ctrl_Off";
290                 break;
291
292             case R.id.Alt:
293                 if (isChecked) data = "Alt_On";
294                 else data = "Alt_Off";
295                 break;
296
297             case R.id.Shift:
298                 if (isChecked) data = "Shift_On";
299                 else data = "Shift_Off";
300                 break;
301
302             case R.id.Show_Hide:
303                 if (isChecked) {
304                     mKeyboardInput.setTransformationMethod(PasswordTransformationMethod.getInstance());
305                 } else mKeyboardInput.setTransformationMethod(null);
306                 mKeyboardInput.setSelection(mKeyboardInput.getText().length());
307                 return;
308             }
309         mKeyboardThread.addToBuffer(data, true);
310     }
311
312     /**
313     * Overrides the <code>View.OnClickListener.onClick(View)</code>
314     * method of the Android API.
315     *
316     * Called when a view has been clicked.
317     *
318     * @param v the view that was clicked.
319     */
320     @Override
321     public void onClick(View v) {
322         String data = "";
323         switch (v.getId()) {
324             case R.id.Home:
325                 data = "Home";
326                 break;
327
328             case R.id.End:
329                 data = "End";
330                 break;
331
332             case R.id.Page_Up:
333                 data = "Page_Up";
334                 break;
335
336             case R.id.Page_Down:
337                 data = "Page_Down";
338                 break;
339
340             case R.id.Del:
341                 data = "Del";
342                 break;
343
344             case R.id.Esc:
345                 data = "Esc";
346                 break;
347
348             case R.id.Tab:
349                 data = "Tab";
350                 break;
351
352             case R.id.f1:
353                 data = "F1";
354                 break;
355
356             case R.id.f2:
357                 data = "F2";
358                 break;
359
360             case R.id.f3:
361                 data = "F3";
362                 break;
363
364             case R.id.f4:
365                 data = "F4";
366                 break;
367
368             case R.id.f5:

```

```

369         data = "F5";
370         break;
371
372     case R.id.f6:
373         data = "F6";
374         break;
375
376     case R.id.f7:
377         data = "F7";
378         break;
379
380     case R.id.f8:
381         data = "F8";
382         break;
383
384     case R.id.f9:
385         data = "F9";
386         break;
387
388     case R.id.f10:
389         data = "F10";
390         break;
391
392     case R.id.f11:
393         data = "F11";
394         break;
395
396     case R.id.f12:
397         data = "F12";
398         break;
399
400     case R.id.Insert:
401         data = "Insert";
402         break;
403
404     case R.id.upArrow:
405         data = "Up";
406         break;
407
408     case R.id.downArrow:
409         data = "Down";
410         break;
411
412     case R.id.leftArrow:
413         data = "Left";
414         break;
415
416     case R.id.rightArrow:
417         data = "Right";
418         break;
419
420     case R.id.keyboard:
421         mKeyboardInput.setSelection(mKeyboardInput.getText().length());
422         return;
423     }
424     mKeyboardThread.addToBuffer(data, true);
425 }
426
427 private void setListeners(View view) {
428     for(int numSpecialButtons = 0; numSpecialButtons < SPECIAL_KEY_ID.length;
429 ++numSpecialButtons) {
430         if(numSpecialButtons >=0 && numSpecialButtons <=3) {
431             CheckBox checkBox = (CheckBox)
432             view.findViewById(SPECIAL_KEY_ID[numSpecialButtons]);
433             checkBox.setOnCheckedChangeListener(this);
434         } else if(numSpecialButtons >=4 && numSpecialButtons <=23) {
435             Button button = (Button)
436             view.findViewById(SPECIAL_KEY_ID[numSpecialButtons]);
437             button.setOnClickListener(this);
438         } else {
439             ImageButton imageButton = (ImageButton)
440             view.findViewById(SPECIAL_KEY_ID[numSpecialButtons]);
441             imageButton.setOnClickListener(this);
442         }
443     }
444 }
445 }

```

MainActivity.java

```
1 package project.android;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.content.SharedPreferences;
6 import android.content.pm.ActivityInfo;
7 import android.os.Build;
8 import android.os.Bundle;
9 import android.support.annotation.NonNull;
10 import android.support.design.widget.NavigationView;
11 import android.support.v4.app.Fragment;
12 import android.support.v4.view.GravityCompat;
13 import android.support.v4.widget.DrawerLayout;
14 import android.support.v7.app.ActionBarDrawerToggle;
15 import android.support.v7.app.AppCompatActivity;
16 import android.support.v7.widget.Toolbar;
17 import android.view.Menu;
18 import android.view.MenuItem;
19 import android.view.View;
20 import android.view.inputmethod.InputMethodManager;
21 import android.widget.Toast;
22
23 import java.io.File;
24 import java.util.ArrayList;
25
26 import project.android.net.NetworkService;
27 import project.android.security.EKEProvider;
28 import project.android.sensor.orientation.OrientationProvider;
29
30 /**
31  * Class representing the <code>Activity</code> that is launched
32  * at app startup.
33  *
34  * <p>
35  * It provides an interface for navigation bar which contains
36  * the following navigation items:
37  * <ul>
38  * <li> 3D Mouse </li>
39  * <li> 2D Mouse </li>
40  * <li> Keyboard </li>
41  * <li> Connect </li>
42  * <li> About </li>
43  * </ul>
44  * </p>
45  */
46 public class MainActivity extends AppCompatActivity
47     implements NavigationView.OnNavigationItemSelectedListener {
48
49     private static ArrayList<Fragment> sFragmentManager = new ArrayList<>();
50
51     public static File sRemouseDir = null;
52     public static byte[] sPublicKey;
53     public static SharedPreferences sSharedPrefs;
54     public static final String DEVICE_NAME = Build.MANUFACTURER + " " + Build.MODEL;
55
56     /**
57      * Overrides the <code>AppCompatActivity.onCreate(Bundle)</code>
58      * method of the Android API.
59      *
60      * Called when the activity is first created.
61      *
62      * @param savedInstanceState If the activity is being re-initialized after
63      * previously being shut down then this <code>
64      * Bundle</code> contains the data it most
65      * recently supplied. <br/>
66      * <b><i>Note: Otherwise it is null.</i></b>
67      */
68     @Override
69     protected void onCreate(Bundle savedInstanceState) {
70         super.onCreate(savedInstanceState);
71         setContentView(R.layout.activity_main);
72         sSharedPrefs = getSharedPreferences("REMOUSE", Context.MODE_PRIVATE);
73         makeRemouseDirectory();
74
75         new Thread(new Runnable() {
76             @Override
77             public void run() {
78                 sPublicKey = new EKEProvider().getBase64EncodedPubKey();
79             }
80         }).start();
81
82         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
83         setSupportActionBar(toolbar);
84         final DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
85         ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
86             this, drawer, toolbar, R.string.navigation_drawer_open,
87             R.string.navigation_drawer_close) {
88             @Override
89             public void onDrawerOpened(View view) {
90                 super.onDrawerOpened(view);
91                 view = getCurrentFocus();
92                 if (view != null) {
93                     InputMethodManager imm = (InputMethodManager)
```



```

93     getSystemService(Context.INPUT_METHOD_SERVICE);
94         imm.hideSoftInputFromWindow(view.getWindowToken(), 0);
95     }
96 };
97 drawer.addDrawerListener(toggle);
98 toggle.syncState();
99
100 sFragmentManager.add(new ConnectionFragment());
101 sFragmentManager.add(new AboutFragment());
102
103 NavigationView navigationView = (NavigationView) findViewById(R.id.nav_view);
104 navigationView.setNavigationItemSelectedListener(this);
105 }
106
107 /**
108  * Overrides the <code>FragmentActivity.onResume()</code>
109  * method of the Android API.
110  *
111  * Called when the activity will start interacting with
112  * the user.
113  */
114 @Override
115 public void onResume() {
116     super.onResume();
117     setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
118 }
119
120 /**
121  * Overrides the <code>FragmentActivity.onPause()</code>
122  * method of the Android API.
123  *
124  * Called when the system is about to start resuming a
125  * previous activity.
126  */
127 @Override
128 public void onPause() {
129     super.onPause();
130     setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
131 }
132
133 /**
134  * Overrides the <code>FragmentActivity.onBackPressed()</code>
135  * method of the Android API.
136  *
137  * Called when the activity has detected the user's press of
138  * the back key.
139  */
140 @Override
141 public void onBackPressed() {
142     DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
143     if (drawer.isDrawerOpen(GravityCompat.START)) {
144         drawer.closeDrawer(GravityCompat.START);
145     }
146 }
147
148 /**
149  * Overrides the <code>AppCompatActivity.onDestroy()</code>
150  * method of the Android API.
151  *
152  * The final call received before an activity is destroyed.
153  */
154 @Override
155 public void onDestroy() {
156     super.onDestroy();
157     this.stopService(new Intent(this, NetworkService.class));
158 }
159
160 /**
161  * Overrides the <code>Activity.onCreateOptionsMenu(Menu)</code>
162  * method of the Android API.
163  *
164  * Initializes the contents of the Activity's standard options
165  * menu.
166  *
167  * @param menu The options menu for placing items.
168  * @return <code>true</code>, if the menu is to be displayed,<br/>
169  *         <code>false</code>, otherwise.
170  */
171 @Override
172 public boolean onCreateOptionsMenu(Menu menu) {
173     // Inflate the menu; this adds items to the action bar if it is present.
174     getMenuInflater().inflate(R.menu.main, menu);
175     return true;
176 }
177
178 /**
179  * Overrides the
180  * <code>Activity.onOptionsItemSelected(MenuItem)</code>
181  * method of the Android API.
182  *
183  * Handles action bar item clicks. The action bar will
184  * automatically handle clicks on the Home/Up button, so
185  * long as a parent activity is specified in
186  * <code>AndroidManifest.xml</code>.
187  *
188  * @param item The menu item that was selected.
189  * @return <code>false</code>, if normal menu processing is
190  *         allowed to proceed, <br/>

```

```

191         * <code>true</code>, if it is to be consumed here.
192     */
193 @Override
194 public boolean onOptionsItemSelected(MenuItem item) {
195     int id = item.getItemId();
196
197     //noinspection SimplifiableIfStatement
198     if (id == R.id.action_settings) {
199         startActivity(new Intent(this, SettingsActivity.class));
200         return true;
201     }
202
203     return super.onOptionsItemSelected(item);
204 }
205
206 /**
207  * Implements the
208  * <code>
209  *     NavigationView.OnNavigationItemSelectedListener.onNavigationItemSelectedListener(MenuItem)
210  * </code>
211  * method of the Android API.
212  *
213  * Called when an item in the navigation menu is selected.
214  *
215  * @param item The selected item.
216  * @return @return <code>true</code>, if the selected item is
217  *         to be displayed, <br/>
218  *         <code>false</code>, otherwise.
219  */
220 @SuppressWarnings("StatementWithEmptyBody")
221 @Override
222 public boolean onNavigationItemSelected(@NonNull MenuItem item) {
223     // Handle navigation view item clicks here.
224     int id = item.getItemId();
225     Fragment fragment = null;
226     String title = getString(R.string.app_name);
227     if (id == R.id.nav_mouse) {
228         // Handle the 3D mouse action
229         if(OrientationProvider.checkGyro(this)) {
230             fragment = new MouseFragment();
231             title = "Remote Mouse";
232         } else {
233             Toast.makeText(this, "Gyroscope not present!", Toast.LENGTH_LONG).show();
234         }
235
236     } else if (id == R.id.nav_touchpad) {
237         // 2D Mouse
238         fragment = new TouchpadFragment();
239         title = "Remote Mouse";
240
241     } else if (id == R.id.nav_keyboard) {
242         // Handle the keyboard action
243         fragment = new KeyboardFragment();
244         title="Remote Keyboard";
245
246     } else if (id == R.id.nav_connect) {
247         // connection module
248         fragment = sFragmentManager.get(0);
249         title="Connect to PC";
250
251     } else if (id == R.id.nav_about) {
252         // App info
253         fragment = sFragmentManager.get(1);
254         title="About app";
255     }
256     if(fragment != null) displayFragment(fragment, title);
257
258     DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
259     drawer.closeDrawer(GravityCompat.START);
260     return true;
261 }
262
263 /**
264  * Returns the {@link project.android.ConnectionFragment} object.
265  *
266  * @return the {@link project.android.ConnectionFragment} object.
267  * @see project.android.ConnectionFragment
268  */
269 public static Fragment getConnectionFragment() { return sFragmentManager.get(0); }
270
271 private void displayFragment(Fragment fragment, String title) {
272     if (fragment != null) {
273         getSupportFragmentManager().beginTransaction()
274             .replace(R.id.content_frame, fragment)
275             .commit();
276     }
277     // set the toolbar title
278     if (getSupportActionBar() != null) getSupportActionBar().setTitle(title);
279 }
280
281 private void makeRemouseDirectory() {
282     sRemouseDir = getDir("Remouse", Context.MODE_PRIVATE);
283     if(!sRemouseDir.exists()) sRemouseDir.mkdir();
284 }
285 }

```

MouseFragment.java

```
1 package project.android;
2
3 import android.app.Activity;
4 import android.content.pm.ActivityInfo;
5 import android.hardware.SensorManager;
6 import android.os.Bundle;
7 import android.support.v4.app.Fragment;
8 import android.view.LayoutInflater;
9 import android.view.MotionEvent;
10 import android.view.View;
11 import android.view.ViewGroup;
12 import android.widget.Button;
13 import android.widget.ImageButton;
14
15 import project.android.sensor.orientation.KalmanFilterProvider;
16 import project.android.sensor.orientation.OrientationProvider;
17
18 import static project.android.ConnectionFragment.sConnectionAlive;
19 import static project.android.ConnectionFragment.sSecuredClient;
20
21 /**
22  * Class representing the <code>Fragment</code> for providing the GUI
23  * frontend for the 3D mouse module.
24  *
25  * <p>
26  *     The following features have been provided:
27  *     <ul>
28  *         <li>
29  *             3D mouse movement.
30  *         </li>
31  *         <li>
32  *             Single left click and double left click.
33  *         </li>
34  *         <li>Right click.</li>
35  *         <li>Middle click.</li>
36  *         <li>Scrolling</li>
37  *     </ul>
38  * </p>
39  * <p>
40  *     The 3D mouse movement feature uses the gyroscope and accelerometer
41  *     sensors. The {@link project.android.sensor.orientation.OrientationProvider}
42  *     and {@link project.android.sensor.orientation.KalmanFilterProvider}
43  *     classes are responsible for processing the sensor data. The clicks
44  *     and scrolling feature uses buttons.
45  * </p>
46  * <p>
47  *     <i>
48  *         <b>Note:</b> This 3D mouse module does not work with Android
49  *         devices where the mentioned sensors are not present.
50  *     </i>
51  * </p>
52  *
53  * @see project.android.sensor.orientation.OrientationProvider
54  * @see project.android.sensor.orientation.KalmanFilterProvider
55  */
56 public class MouseFragment extends Fragment implements View.OnClickListener,
57     View.OnTouchListener {
58     private OrientationProvider mOrientationProvider;
59
60     private long mTouchTime;
61     private boolean mFirstTouch;
62
63     public static boolean sMouseAlive;
64
65     /**
66      * Overrides the
67      * <code>Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)</code>
68      * method of the Android API.
69      *
70      * Called to have the fragment instantiate its user interface view.
71      *
72      * @param inflater The <code>LayoutInflater</code> object that can be used to inflate any
73      * views in the fragment.
74      * @param container If non-null, this is the parent view that the fragment's UI should be
75      * attached to.
76      *
77      * The fragment should not add the view itself, but this can be used to
78      * generate
79      * the <code>LayoutParams</code> of the view.
80      * @param savedInstanceState If non-null, this fragment is being re-constructed from a
81      * previous
82      * saved state as given here.
83      * @return the <code>View</code> for the fragment's UI, or <code>null</code>.
84      */
85     @Override
86     public View onCreateView(LayoutInflater inflater, ViewGroup container,
87         Bundle savedInstanceState) {
88         View view = inflater.inflate(R.layout.fragment_mouse, container, false);
89         Button left = (Button) view.findViewById(R.id.button_left);
90         Button right = (Button) view.findViewById(R.id.button_right);
91         Button middle = (Button) view.findViewById(R.id.button_middle);
92         Button moveButton = (Button) view.findViewById(R.id.moveButton);
93         ImageButton upScroll = (ImageButton) view.findViewById(R.id.upscroll);
```

```

89     ImageButton downScroll = (ImageButton) view.findViewById(R.id.downscroll);
90
91     moveButton.setLongClickable(true);
92     moveButton.setOnLongClickListener(new View.OnLongClickListener() {
93         @Override
94         public boolean onLongClick(View view) {
95             if(sConnectionAlive.containsValue(true)) {
96                 sMouseAlive = true;
97                 sendMouseMoveData();
98             }
99             return true;
100         }
101     });
102     moveButton.setOnTouchListener(this);
103     left.setOnTouchListener(this);
104
105     right.setOnClickListeners(this);
106     middle.setOnClickListeners(this);
107     upScroll.setOnClickListeners(this);
108     downScroll.setOnClickListeners(this);
109
110     return view;
111 }
112
113 /**
114  * Overrides the <code>Fragment.onResume()</code> method of
115  * the Android API.
116  *
117  * Called when the activity will start interacting with the user.
118  */
119 @Override
120 public void onResume() {
121     super.onResume();
122     getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
123 }
124
125 /**
126  * Overrides the <code>Fragment.onPause()</code> method of
127  * the Android API.
128  *
129  * Called when the system is about to start resuming a previous
130  * activity.
131  */
132 @Override
133 public void onPause() {
134     super.onPause();
135     getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
136 }
137
138 /**
139  * Overrides the <code>Fragment.onDestroyView()</code> method
140  * of the Android API.
141  *
142  * Called when the view previously created by
143  * {@link #onCreateView(LayoutInflater, ViewGroup, Bundle)}
144  * has been detached from the fragment.
145  */
146 @Override
147 public void onDestroyView() {
148     super.onDestroyView();
149     sMouseAlive = false;
150     if(mOrientationProvider != null)    mOrientationProvider.sensorStop();
151 }
152
153 /**
154  * Overrides the
155  * <code>View.onTouchListener.onTouch(View, MotionEvent)</code>
156  * method of the Android API.
157  *
158  * Called when a touch event is dispatched to a <code>View</code>.
159  * This allows listeners to get a chance to respond before the
160  * target <code>View</code>.
161  *
162  * @param view the view the touch event has been dispatched to.
163  * @param motionEvent the <code>MotionEvent</code> object containing
164  *         full information about the event.
165  * @return <code>true</code>, if the listener has consumed the event,<br/>
166  *         <code>false</code>, otherwise.
167  */
168 @Override
169 public boolean onTouch(View view, MotionEvent motionEvent) {
170     if(view.getId() == R.id.moveButton) {
171         if(sConnectionAlive.containsValue(true) && sMouseAlive &&
172 motionEvent.getActionMasked() == MotionEvent.ACTION_UP) {
173             sMouseAlive = false;
174             if(mOrientationProvider != null)    mOrientationProvider.sensorStop();
175             return false;
176         }
177         if(view.getId() == R.id.button_left) {
178             if (sConnectionAlive.containsValue(true) && motionEvent.getActionMasked() ==
179 MotionEvent.ACTION_DOWN) {
180                 if (mFirstTouch && (System.currentTimeMillis() - mTouchTime) <= 300) {
181                     mFirstTouch = false;
182                     sendMouseButtonData("left");
183                 } else {
184                     mFirstTouch = true;
185                     mTouchTime = System.currentTimeMillis();
186                     sendMouseButtonData("left");
187                 }
188             }
189         }
190     }
191 }

```

```

186         }
187     }
188 }
189     return true;
190 }
191
192 /**
193  * Overrides the <code>View.OnClickListener.onClick(View)</code>
194  * method of the Android API.
195  *
196  * Called when a view has been clicked.
197  *
198  * @param view the view that was clicked.
199  */
200 @Override
201 public void onClick(View view) {
202     String data = "";
203     switch(view.getId()) {
204
205         case R.id.button_right:
206             data = "right";
207             break;
208
209         case R.id.button_middle:
210             data = "middle";
211             break;
212
213         case R.id.upscroll:
214             data = "upscroll";
215             break;
216
217         case R.id.downscroll:
218             data = "downscroll";
219     }
220     sendMouseButtonData(data);
221 }
222
223 private void sendMouseButtonData(final String data) {
224     new Thread(new Runnable() {
225         @Override
226         public void run() {
227             if (sConnectionAlive.containsValue(true)) {
228                 sSecuredClient.sendData("Mouse_Button", data);
229             }
230         }
231     }).start();
232 }
233
234 private void sendMouseMoveData() {
235     new Thread(new Runnable() {
236         @Override
237         public void run() {
238             mOrientationProvider = new KalmanFilterProvider((SensorManager)
239                 getActivity().getSystemService(Activity.SENSOR_SERVICE));
240             if (!sMouseAlive || !sConnectionAlive.containsValue(true))
241                 mOrientationProvider.sensorStop();
242         }
243     }).start();
244 }

```

SettingsActivity.java

```
1 package project.android;
2
3 import android.content.SharedPreferences;
4 import android.content.pm.ActivityInfo;
5 import android.os.Bundle;
6 import android.preference.PreferenceManager;
7 import android.support.v7.app.AppCompatActivity;
8 import android.support.v7.widget.Toolbar;
9
10 import static project.android.MainActivity.sSharedPrefs;
11
12 /**
13  * Class representing the <code>Activity</code> for app settings.
14  *
15  * <p>
16  * It displays the app settings and enables customization
17  * for the users. There are two customizations that are
18  * present :
19  * <ol>
20  * <li>3D Mouse Sensitivity.</li>
21  * <li>2D Mouse Sensitivity.</li>
22  * </ol>
23  * Each of these can have three possible values - : <i>"Low"</i>,
24  * <i>"Medium"</i> and <i>"High"</i>. The default value is
25  * <i>"Medium"</i>.
26  * </p>
27  * <p>
28  * This class is made separate from the {@link MainActivity}
29  * to enable the <i>"Settings"</i> to be opened from any
30  * fragment of the <code>MainActivity</code>. This avoids
31  * overlapping of other fragments - that are already rendered -
32  * by the {@link SettingsFragment}.
33  * </p>
34  * <p>
35  * It also implements the
36  * <code>SharedPreferences.OnSharedPreferenceChangeListener</code>
37  * of the Android API. The
38  * {@link #onSharedPreferenceChanged(SharedPreferences, String)}
39  * method listens to any changes made in the app settings.
40  * </p>
41  *
42  * @see project.android.MainActivity
43  * @see project.android.SettingsFragment
44  * @see #onSharedPreferenceChanged(SharedPreferences, String)
45  */
46 public class SettingsActivity extends AppCompatActivity implements
47     SharedPreferences.OnSharedPreferenceChangeListener {
48
49     public static float sMouseSensitivity3d = sSharedPrefs.getFloat("3d", 2000.0f);
50     public static float sMouseSensitivity2d = sSharedPrefs.getFloat("2d", 1.5f);
51
52     /**
53      * Overrides the <code>AppCompatActivity.onCreate(Bundle)</code>
54      * of the Android API.
55      *
56      * Called when the activity is first created.
57      *
58      * @param savedInstanceState If the activity is being re-initialized
59      * after previously being shut down then
60      * this <code>Bundle</code> contains the
61      * data it most recently supplied.<br/>
62      * <b><i>Note: Otherwise it is null.</i></b>
63      */
64     @Override
65     protected void onCreate(Bundle savedInstanceState) {
66         super.onCreate(savedInstanceState);
67         setContentView(R.layout.activity_settings);
68         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar1);
69         setSupportActionBar(toolbar);
70         getSupportFragmentManager().beginTransaction()
71             .replace(R.id.content_frame_settings, new SettingsFragment())
72             .commit();
73         if (getSupportActionBar() != null) getSupportActionBar().setTitle("Settings");
74     }
75
76     /**
77      * Overrides the
78      * <code>
79      *     OnSharedPreferenceChangeListener.onSharedPreferenceChanged
80      *     (SharedPreferences, String)
81      * </code>
82      * method of the Android API.
83      *
84      * @param sharedPreferences the <code>SharedPreferences</code>
85      * that received the change.
86      * @param key the key of the preference that was changed, added,
87      * or removed.
88      */
89     @Override
```

```

89 |     public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
90 |         if(key.equals("sensitivity_3D")) {
91 |             sMouseSensitivity3d = Float.parseFloat(sharedPreferences.getString(key,
92 | "2000.0"));
93 |         } else {
94 |             sMouseSensitivity2d = Float.parseFloat(sharedPreferences.getString(key, "1.5"));
95 |             SharedPreferences.Editor editor = sSharedPrefs.edit();
96 |             editor.putFloat("2d", sMouseSensitivity2d);
97 |             editor.putFloat("3d", sMouseSensitivity3d);
98 |             editor.apply();
99 |         }
100 |
101 |         /**
102 |          * Overrides the <code>FragmentActivity.onResume()</code>
103 |          * method of the Android API.
104 |          *
105 |          * Called when the activity will start interacting with
106 |          * the user.
107 |          */
108 |         @Override
109 |         public void onResume() {
110 |             super.onResume();
111 |
112 |             PreferenceManager.getDefaultSharedPreferences(this).registerOnSharedPreferenceChangeListener
113 |             (this);
114 |             setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
115 |
116 |             /**
117 |              * Overrides the <code>FragmentActivity.onPause()</code>
118 |              * method of the Android API.
119 |              *
120 |              * Called when the system is about to start resuming a
121 |              * previous activity.
122 |              */
123 |             @Override
124 |             public void onPause() {
125 |                 super.onPause();
126 |
127 |                 PreferenceManager.getDefaultSharedPreferences(this).unregisterOnSharedPreferenceChangeListen
128 |                 er(this);
129 |                 setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
130 |             }
131 |         }

```

SettingsFragment.java

```
1 package project.android;
2
3 import android.content.pm.ActivityInfo;
4 import android.os.Bundle;
5 import android.preference.PreferenceFragment;
6
7 /**
8  * Class representing the <code>Fragment</code> for changing app
9  * settings.
10
11  * This <code>SettingsFragment</code> is created by the
12  * {@link SettingsActivity#onCreate(Bundle)} method.
13
14  * @see project.android.SettingsActivity
15  * @see project.android.SettingsActivity#onCreate(Bundle)
16  */
17 public class SettingsFragment extends PreferenceFragment {
18
19     /**
20      * Overrides the <code>PreferenceFragment.onCreate(Bundle)</code>
21      * method of the Android API.
22      *
23      * Called when this fragment is created.
24      *
25      * @param savedInstanceState If the fragment is being re-created
26      * from a previous saved state, this is
27      * the state.
28      */
29     @Override
30     public void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         addPreferencesFromResource(R.xml.preferences);
33     }
34
35     /**
36      * Overrides the <code>PreferenceFragment.onResume()</code> method
37      * of the Android API.
38      *
39      * Called when this fragment will start interacting with the user.
40      */
41     @Override
42     public void onResume() {
43         super.onResume();
44         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
45     }
46
47     /**
48      * Overrides the <code>PreferenceFragment.onPause()</code> method
49      * of the Android API.
50      *
51      * Called when the system is about to start resuming a previous activity.
52      */
53     @Override
54     public void onPause() {
55         super.onPause();
56         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
57     }
58 }
```


TouchpadFragment.java

```
1 package project.android;
2
3 import android.content.pm.ActivityInfo;
4 import android.os.Bundle;
5 import android.support.v4.app.Fragment;
6 import android.view.LayoutInflater;
7 import android.view.MotionEvent;
8 import android.view.View;
9 import android.view.ViewGroup;
10 import android.widget.Button;
11 import android.widget.ImageButton;
12
13 import static project.android.ConnectionFragment.sConnectionAlive;
14 import static project.android.ConnectionFragment.sSecuredClient;
15
16 /**
17  * Class representing the <code>Fragment</code> for providing the GUI
18  * frontend for the 2D mouse module.
19  *
20  * <p>
21  * The following features have been provided:
22  * <ul>
23  * <li>
24  * 2D mouse movement.
25  * </li>
26  * <li>
27  * Single left click and double left click.
28  * </li>
29  * <li>Right click</li>
30  * <li>Middle click.</li>
31  * <li>Scrolling</li>
32  * </ul>
33  * <p>
34  * The 2D mouse movement feature is provided by implementing
35  * the <code>View.OnTouchListener</code> interface.
36  * </p>
37  */
38 public class TouchpadFragment extends Fragment implements View.OnClickListener,
39 View.OnTouchListener {
40     private long mTouchTime;
41     private boolean mFirstTouch;
42     private float mLastMoveX = Float.MAX_VALUE;
43     private float mLastMoveY = Float.MAX_VALUE;
44
45     /**
46      * Overrides the
47      * <code>Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)</code>
48      * method of the Android API.
49      *
50      * Called to have the fragment instantiate its user interface view.
51      *
52      * @param inflater The <code>LayoutInflater</code> object that can be used
53      * to inflate any views in the fragment.
54      * @param container If non-null, this is the parent view that the fragment's
55      * UI should be attached to. The fragment should not add
56      * the view itself, but this can be used to generate the
57      * <code>LayoutParams</code> of the view.
58      * @param savedInstanceState If non-null, this fragment is being re-constructed
59      * from a previous saved state as given here.
60      * @return the <code>View</code> for the fragment's UI, or <code>null</code>.
61      */
62     @Override
63     public View onCreateView(LayoutInflater inflater, ViewGroup container,
64                             Bundle savedInstanceState) {
65         View view = inflater.inflate(R.layout.fragment_touchpad, container, false);
66         Button left = (Button) view.findViewById(R.id.button_touch_left);
67         Button right = (Button) view.findViewById(R.id.button_touch_right);
68         Button middle = (Button) view.findViewById(R.id.button_touch_middle);
69         Button touchButton = (Button) view.findViewById(R.id.button_touch);
70         ImageButton upScroll = (ImageButton) view.findViewById(R.id.touch_upscroll);
71         ImageButton downScroll = (ImageButton) view.findViewById(R.id.touch_downscroll);
72
73         touchButton.setOnTouchListener(this);
74         left.setOnTouchListener(this);
75
76         right.setOnClickListener(this);
77         middle.setOnClickListener(this);
78         upScroll.setOnClickListener(this);
79         downScroll.setOnClickListener(this);
80
81         return view;
82     }
83
84     /**
85      * Overrides the <code>Fragment.onResume()</code>
86      * method of the Android API.
87      *
88      * Called when the activity will start interacting with the user.
```

```

89     */
90     @Override
91     public void onResume() {
92         super.onResume();
93         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
94     }
95
96     /**
97      * Overrides the <code>Fragment.onPause()</code>
98      * method of the Android API.
99      *
100     * Called when the system is about to start resuming a previous activity.
101     */
102     @Override
103     public void onPause() {
104         super.onPause();
105         getActivity().setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_FULL_SENSOR);
106     }
107
108     /**
109      * Overrides the
110      * <code>View.onTouchListener.onTouch(View, MotionEvent)</code>
111      * method of the Android API.
112      *
113      * Called when a touch event is dispatched to a <code>View</code>.
114      * This allows listeners to get a chance to respond before the
115      * target <code>View</code>.
116      *
117      * @param view the view the touch event has been dispatched to.
118      * @param motionEvent the <code>MotionEvent</code> object containing
119      *         full information about the event.
120      * @return <code>true</code>, if the listener has consumed the event,<br/>
121      *         <code>false</code>, otherwise.
122      */
123     @Override
124     public boolean onTouch(View view, MotionEvent motionEvent) {
125         if (view.getId() == R.id.button_touch) {
126             if (sConnectionAlive.containsValue(true)) processTouch(motionEvent);
127         }
128         if (view.getId() == R.id.button_touch_left) {
129             if (sConnectionAlive.containsValue(true) && motionEvent.getActionMasked() ==
MotionEvent.ACTION_DOWN) {
130                 if (mFirstTouch && (System.currentTimeMillis() - mTouchTime) <= 300) {
131                     mFirstTouch = false;
132                     sendMouseButtonData("left");
133                 } else {
134                     mFirstTouch = true;
135                     mTouchTime = System.currentTimeMillis();
136                     sendMouseButtonData("left");
137                 }
138             }
139         }
140         return true;
141     }
142
143     /**
144      * Overrides the
145      * <code>View.OnClickListener.onClick(View)</code>
146      * method of the Android API.
147      *
148      * Called when a view has been clicked.
149      *
150      * @param view the view that was clicked.
151      */
152     @Override
153     public void onClick(View view) {
154         String data = "";
155         switch (view.getId()) {
156
157             case R.id.button_touch_right:
158                 data = "right";
159                 break;
160
161             case R.id.button_touch_middle:
162                 data = "middle";
163                 break;
164
165             case R.id.touch_upscroll:
166                 data = "upscroll";
167                 break;
168
169             case R.id.touch_downscroll:
170                 data = "downscroll";
171         }
172         sendMouseButtonData(data);
173     }
174
175     private void sendMouseButtonData(final String data) {
176         new Thread(new Runnable() {
177             @Override
178             public void run() {
179                 if (sConnectionAlive.containsValue(true)) {
180                     sSecuredClient.sendData("Mouse_Button", data);
181                 }
182             }
183         })

```

```

183     }).start();
184 }
185
186 private void processTouch(MotionEvent event) {
187     switch (event.getActionMasked()) {
188         case MotionEvent.ACTION_DOWN:
189             onSingleClick(event, true);
190             break;
191         case MotionEvent.ACTION_UP:
192             onSingleClick(event, false);
193             break;
194         case MotionEvent.ACTION_MOVE:
195             onMove(event);
196     }
197 }
198
199 private void sendMouseMovementData(final int x, final int y) {
200     new Thread(new Runnable() {
201         @Override
202         public void run() {
203             if (sConnectionAlive.containsValue(true)) {
204                 sSecuredClient.sendData(x, y);
205             }
206         }
207     }).start();
208 }
209
210 private void onMove(MotionEvent event) {
211     float curMoveX = event.getX();
212     float curMoveY = event.getY();
213     int distanceX = 0, distanceY = 0;
214
215     if (mLastMoveX != Float.MAX_VALUE && mLastMoveY != Float.MAX_VALUE) {
216         distanceX = (int) (curMoveX - mLastMoveX);
217         distanceY = (int) (curMoveY - mLastMoveY);
218     }
219
220     sendMouseMovementData(distanceX, distanceY);
221     mLastMoveX = curMoveX;
222     mLastMoveY = curMoveY;
223 }
224
225 private void onSingleClick(MotionEvent event, boolean down) {
226     if (down) {
227         mLastMoveX = event.getX();
228         mLastMoveY = event.getY();
229     }
230 }
231 }

```

activity_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.v4.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res
  /android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:id="@+id/drawer_layout"
6   android:layout_width="match_parent"
7   android:layout_height="match_parent"
8   android:fitsSystemWindows="true"
9   tools:openDrawer="start">
10
11   <include
12     layout="@layout/app_bar_main"
13     android:layout_width="match_parent"
14     android:layout_height="match_parent" />
15
16   <android.support.design.widget.NavigationView
17     android:id="@+id/nav_view"
18     android:layout_width="wrap_content"
19     android:layout_height="match_parent"
20     android:layout_gravity="start"
21     android:fitsSystemWindows="true"
22     app:headerLayout="@layout/nav_header_main"
23     app:menu="@menu/activity_main_drawer" />
24
25 </android.support.v4.widget.DrawerLayout>
```

activity_settings.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com
  /apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   android:fitsSystemWindows="true"
8   tools:context="project.android.SettingsActivity">
9
10   <android.support.design.widget.AppBarLayout
11     android:layout_width="match_parent"
12     android:layout_height="wrap_content"
13     android:theme="@style/AppTheme.AppBarOverlay">
14
15     <android.support.v7.widget.Toolbar
16       android:id="@+id/toolbar1"
17       android:layout_width="match_parent"
18       android:layout_height="?attr/actionBarSize"
19       android:background="?attr/colorPrimary"
20       app:popupTheme="@style/AppTheme.PopupOverlay" />
21
22   </android.support.design.widget.AppBarLayout>
23
24   <include layout="@layout/content_settings" />
25
26 </android.support.design.widget.CoordinatorLayout>
```

app_bar_main.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com
  /apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="match_parent"
7   android:fitsSystemWindows="true"
8   tools:context="project.android.MainActivity">
9
10   <android.support.design.widget.AppBarLayout
11     android:layout_width="match_parent"
12     android:layout_height="wrap_content"
13     android:theme="@style/AppTheme.AppBarOverlay">
14
15     <android.support.v7.widget.Toolbar
16       android:id="@+id/toolbar"
17       android:layout_width="match_parent"
18       android:layout_height="?attr/actionBarSize"
19       android:background="?attr/colorPrimary"
20       app:popupTheme="@style/AppTheme.PopupOverlay" />
21
22   </android.support.design.widget.AppBarLayout>
```

```

23     <include layout="@layout/content_main" />
24
25 </android.support.design.widget.CoordinatorLayout>

```

content_main.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     app:layout_behavior="@string/appbar_scrolling_view_behavior"
8     tools:context="project.android.MainActivity"
9     tools:showIn="@layout/app_bar_main"
10    android:id="@+id/content_frame">
11    <ImageView
12        android:layout_width="match_parent"
13        android:layout_height="match_parent"
14        android:layout_gravity="center"
15        android:src="@mipmap/wireless" />
16 </FrameLayout>

```

content_settings.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     app:layout_behavior="@string/appbar_scrolling_view_behavior"
8     tools:context="project.android.SettingsActivity"
9     tools:showIn="@layout/activity_settings"
10    android:id="@+id/content_frame_settings">
11    <ImageView
12        android:layout_width="wrap_content"
13        android:layout_height="wrap_content"
14        android:layout_gravity="center"
15        android:src="@mipmap/settings"
16        tools:ignore="ContentDescription" />
17 </FrameLayout>

```

fragment_about.xml

```

1 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:background="#ffffff"
6     tools:context="project.android.AboutFragment">
7
8
9    <WebView
10        android:id="@+id/webView"
11        android:layout_width="match_parent"
12        android:layout_height="match_parent"/>
13
14 </FrameLayout>

```

fragment_connect.xml

```

1 <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:background="#ffffff"
6     tools:context="project.android.ConnectionFragment">
7    <ImageView
8        android:layout_width="match_parent"
9        android:layout_height="match_parent"
10        android:id="@+id/imageView2"
11        android:layout_gravity="center"
12        android:src="@mipmap/connection" />
13
14    <ListView
15        android:layout_width="match_parent"
16        android:layout_height="match_parent"
17        android:id="@+id/listView"
18        android:layout_gravity="center_horizontal|top"

```

```

19         android:background="#abcd"
20         android:divider="@color/colorPrimaryDark"
21         android:dividerHeight="1px"
22         android:listSelector="@android:drawable/list_selector_background" />
23     </FrameLayout>

```

fragment_keyboard.xml

```

1  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:app="http://schemas.android.com/apk/res-auto"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="#ffffff"
7      android:weightSum="1"
8      android:orientation="vertical"
9      tools:context="project.android.KeyboardFragment" >
10
11     <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
12         xmlns:tools="http://schemas.android.com/tools"
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:layout_weight="0.9"
16         android:background="@color/keyColor"
17         tools:context="project.android.KeyboardFragment">
18
19         <EditText
20             android:id="@+id/keyboard"
21             style="@style/Widget.AppCompat.EditText"
22             android:layout_width="match_parent"
23             android:layout_height="wrap_content"
24             android:inputType="text|textImeMultiLine|textAutoComplete|textMultiLine"
25             android:hint="@string/type_here"
26             android:maxLines="10" />
27
28     </FrameLayout>
29
30     <TableLayout
31         android:layout_width="match_parent"
32         android:layout_height="wrap_content"
33         android:layout_weight="0.1"
34         android:background="@color/keyboardColor"
35         android:stretchColumns="0, 1, 2, 3"
36         tools:ignore="ExtraText">
37
38         <TableRow>
39
40             <CheckBox
41                 android:id="@+id/Ctrl"
42                 android:layout_width="wrap_content"
43                 android:layout_height="wrap_content"
44                 android:text="@string/ctrl" />
45
46             <CheckBox
47                 android:id="@+id/Alt"
48                 android:layout_width="wrap_content"
49                 android:layout_height="wrap_content"
50                 android:text="@string/alt" />
51
52             <CheckBox
53                 android:id="@+id/Shift"
54                 android:layout_width="wrap_content"
55                 android:layout_height="wrap_content"
56                 android:text="@string/shift" />
57
58             <CheckBox
59                 android:id="@+id/Show_Hide"
60                 android:layout_width="wrap_content"
61                 android:layout_height="wrap_content"
62                 android:text="@string/show_hide_text" />
63
64         </TableRow>
65
66         <TableRow>
67
68             <Button
69                 android:id="@+id/Home"
70                 android:layout_width="wrap_content"
71                 android:layout_height="wrap_content"
72                 android:text="@string/home" />
73
74             <Button
75                 android:id="@+id/End"
76                 android:layout_width="wrap_content"
77                 android:layout_height="wrap_content"
78                 android:text="@string/end" />
79
80             <Button
81                 android:id="@+id/Page_Up"
82                 android:layout_width="wrap_content"
83

```

```

84         android:layout_height="wrap_content"
85         android:text="@string/page_up" />
86
87     <Button
88         android:id="@+id/Page_Down"
89         android:layout_width="wrap_content"
90         android:layout_height="wrap_content"
91         android:text="@string/page_down" />
92
93
94 </TableRow>
95 <TableRow>
96
97     <Button
98         android:id="@+id/Del"
99         android:layout_width="wrap_content"
100        android:layout_height="wrap_content"
101        android:text="@string/del" />
102
103    <Button
104        android:id="@+id/Esc"
105        android:layout_width="wrap_content"
106        android:layout_height="wrap_content"
107        android:text="@string/esc" />
108
109    <Button
110        android:id="@+id/Tab"
111        android:layout_width="wrap_content"
112        android:layout_height="wrap_content"
113        android:text="@string/tab" />
114
115    <Button
116        android:id="@+id/f1"
117        android:layout_width="wrap_content"
118        android:layout_height="wrap_content"
119        android:text="@string/f1" />
120
121 </TableRow>
122 <TableRow>
123
124    <Button
125        android:id="@+id/f2"
126        android:layout_width="wrap_content"
127        android:layout_height="wrap_content"
128        android:text="@string/f2" />
129
130    <Button
131        android:id="@+id/f3"
132        android:layout_width="wrap_content"
133        android:layout_height="wrap_content"
134        android:text="@string/f3" />
135
136    <Button
137        android:id="@+id/f4"
138        android:layout_width="wrap_content"
139        android:layout_height="wrap_content"
140        android:text="@string/f4" />
141
142    <Button
143        android:id="@+id/f5"
144        android:layout_width="wrap_content"
145        android:layout_height="wrap_content"
146        android:text="@string/f5" />
147
148 </TableRow>
149 <TableRow>
150
151    <Button
152        android:id="@+id/f6"
153        android:layout_width="wrap_content"
154        android:layout_height="wrap_content"
155        android:text="@string/f6" />
156
157    <Button
158        android:id="@+id/f7"
159        android:layout_width="wrap_content"
160        android:layout_height="wrap_content"
161        android:text="@string/f7" />
162
163    <Button
164        android:id="@+id/f8"
165        android:layout_width="wrap_content"
166        android:layout_height="wrap_content"
167        android:text="@string/f8" />
168
169    <Button
170        android:id="@+id/f9"
171        android:layout_width="wrap_content"
172        android:layout_height="wrap_content"
173        android:text="@string/f9" />
174
175 </TableRow>
176 <TableRow>
177
178

```



```

179         <Button
180             android:id="@+id/f10"
181             android:layout_width="wrap_content"
182             android:layout_height="wrap_content"
183             android:text="@string/f10" />
184
185         <ImageButton
186             android:id="@+id/upArrow"
187             android:layout_width="wrap_content"
188             android:layout_height="wrap_content"
189             app:srcCompat="@drawable/ic_arrow_upward_black_24dp" />
190
191         <Button
192             android:id="@+id/f11"
193             android:layout_width="wrap_content"
194             android:layout_height="wrap_content"
195             android:text="@string/f11" />
196
197
198         <Button
199             android:id="@+id/f12"
200             android:layout_width="wrap_content"
201             android:layout_height="wrap_content"
202             android:text="@string/f12" />
203     </TableRow>
204 <TableRow>
205
206     <ImageButton
207         android:id="@+id/leftArrow"
208         android:layout_width="wrap_content"
209         android:layout_height="wrap_content"
210         app:srcCompat="@drawable/ic_arrow_back_black_24dp" />
211
212     <ImageButton
213         android:id="@+id/downArrow"
214         android:layout_width="wrap_content"
215         android:layout_height="wrap_content"
216         app:srcCompat="@drawable/ic_arrow_downward_black_24dp" />
217
218     <ImageButton
219         android:id="@+id/rightArrow"
220         android:layout_width="wrap_content"
221         android:layout_height="wrap_content"
222         app:srcCompat="@drawable/ic_arrow_forward_black_24dp" />
223
224     <Button
225         android:id="@+id/Insert"
226         android:layout_width="wrap_content"
227         android:layout_height="wrap_content"
228         android:text="@string/insert" />
229
230 </TableRow>
231
232 </TableLayout>
233 </LinearLayout>
234
235

```

fragment_mouse.xml

```

1  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:app="http://schemas.android.com/apk/res-auto"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="#000000"
7      tools:context="project.android.MouseFragment">
8
9      <Button
10         android:text="@string/left"
11         android:layout_width="130dp"
12         android:layout_height="wrap_content"
13         android:id="@+id/button_left"
14         android:layout_alignParentBottom="true"
15         android:layout_alignParentLeft="true"
16         android:layout_alignParentStart="true"
17         android:layout_below="@+id/moveButton"
18         style="@style/Widget.AppCompat.Button" />
19
20      <Button
21         android:text="@string/right"
22         android:layout_height="wrap_content"
23         android:id="@+id/button_right"
24         android:layout_alignParentBottom="true"
25         android:layout_alignParentRight="true"
26         android:layout_alignParentEnd="true"
27         android:layout_width="130dp"
28         android:layout_below="@+id/moveButton"
29         style="@style/Widget.AppCompat.Button" />
30
31      <Button

```



```

32         android:text="@string/press_move"
33         android:layout_width="match_parent"
34         android:layout_height="200dp"
35         android:id="@+id/moveButton"
36         android:background="#7fa87f"
37         android:layout_alignParentRight="true"
38         android:layout_alignParentEnd="true"
39         android:layout_alignParentLeft="true"
40         android:layout_alignParentStart="true"
41         android:layout_alignParentTop="true"
42         style="@style/Widget.AppCompat.Button" />
43
44     <ImageButton
45         app:srcCompat="@drawable/ic_expand_less_black_24dp"
46         android:id="@+id/upscroll"
47         android:layout_height="100dp"
48         android:layout_width="90dp"
49         android:layout_below="@+id/moveButton"
50         android:layout_alignLeft="@+id/button_middle"
51         android:layout_alignStart="@+id/button_middle"
52         style="@style/Widget.AppCompat.ImageButton" />
53
54     <Button
55         android:text="@string/middle"
56         android:layout_width="wrap_content"
57         android:id="@+id/button_middle"
58         android:layout_height="wrap_content"
59         android:layout_alignBaseline="@+id/button_left"
60         android:layout_alignBottom="@+id/button_left"
61         android:layout_alignLeft="@+id/downscroll"
62         android:layout_alignStart="@+id/downscroll"
63         style="@style/Widget.AppCompat.Button" />
64
65     <ImageButton
66         android:layout_width="90dp"
67         android:layout_height="100dp"
68         app:srcCompat="@drawable/ic_expand_more_black_24dp"
69         android:id="@+id/downscroll"
70         android:layout_alignParentBottom="true"
71         android:layout_centerHorizontal="true"
72         style="@style/Widget.AppCompat.ImageButton" />
73
74 </RelativeLayout>

```

fragment_touchpad.xml

```

1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:app="http://schemas.android.com/apk/res-auto"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     android:background="#000000"
7     tools:context="project.android.TouchpadFragment">
8
9     <Button
10         android:text="@string/left"
11         android:layout_width="wrap_content"
12         android:layout_height="100dp"
13         android:id="@+id/button_touch_left"
14         style="@style/Widget.AppCompat.Button"
15         android:layout_alignParentBottom="true"
16         android:layout_alignParentLeft="true"
17         android:layout_alignParentStart="true"
18         android:layout_toLeftOf="@+id/touch_downscroll"
19         android:layout_toStartOf="@+id/touch_downscroll" />
20
21     <Button
22         android:text="@string/touchpad"
23         android:layout_width="wrap_content"
24         android:layout_height="400dp"
25         android:id="@+id/button_touch"
26         android:background="#7fa87f"
27         android:layout_alignParentLeft="true"
28         android:layout_alignParentStart="true"
29         android:layout_alignParentTop="true"
30         android:layout_alignParentRight="true"
31         android:layout_alignParentEnd="true"
32         style="@style/Widget.AppCompat.Button"
33         android:layout_above="@+id/button_touch_left" />
34
35     <Button
36         android:text="@string/middle"
37         android:layout_width="80dp"
38         android:id="@+id/button_touch_middle"
39         android:layout_height="100dp"
40         style="@style/Widget.AppCompat.Button"
41         android:layout_alignParentBottom="true"
42         android:layout_toLeftOf="@+id/button_touch_right"
43         android:layout_toStartOf="@+id/button_touch_right" />
44
45     <ImageButton

```

```

46         android:layout_width="50dp"
47         android:layout_height="50dp"
48         app:srcCompat="@drawable/ic_expand_more_black_24dp"
49         android:id="@+id/touch_downScroll"
50         style="@style/Widget.AppCompat.ImageButton"
51         android:layout_alignParentBottom="true"
52         android:layout_toLeftOf="@+id/button_touch_middle"
53         android:layout_toStartOf="@+id/button_touch_middle" />
54
55     <ImageButton
56         app:srcCompat="@drawable/ic_expand_less_black_24dp"
57         android:id="@+id/touch_upscroll"
58         android:layout_height="50dp"
59         android:layout_width="50dp"
60         style="@style/Widget.AppCompat.ImageButton"
61         android:layout_below="@+id/button_touch"
62         android:layout_toLeftOf="@+id/button_touch_middle"
63         android:layout_toStartOf="@+id/button_touch_middle" />
64
65     <Button
66         style="@style/Widget.AppCompat.Button"
67         android:text="@string/right"
68         android:layout_height="100dp"
69         android:id="@+id/button_touch_right"
70         android:layout_width="80dp"
71         android:layout_alignParentBottom="true"
72         android:layout_alignParentRight="true"
73         android:layout_alignParentEnd="true" />
74
75 </RelativeLayout>

```

local_devices.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="fill_parent"
4      android:layout_height="wrap_content"
5      android:orientation="vertical"
6      android:padding="6dip">
7
8      <TextView
9          android:id="@+id/code"
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:textColorLink="@color/colorPrimary"
13         android:textSize="18sp"
14         android:layout_marginLeft="13dp"
15         android:layout_marginStart="13dp"
16         android:layout_centerVertical="true"
17         android:layout_toRightOf="@+id/connectIcon"
18         android:layout_toEndOf="@+id/connectIcon" />
19
20     <ImageView
21         android:id="@+id/connectIcon"
22         android:layout_width="@android:dimen/app_icon_size"
23         android:layout_height="@android:dimen/app_icon_size"
24         android:layout_alignParentLeft="true"
25         android:layout_alignParentStart="true" />
26
27 </RelativeLayout>

```

nav_header_main.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="@dimen/nav_header_height"
5      android:background="@mipmap/welcome"
6      android:gravity="bottom"
7      android:orientation="vertical"
8      android:paddingBottom="@dimen/activity_vertical_margin"
9      android:paddingLeft="@dimen/activity_horizontal_margin"
10     android:paddingRight="@dimen/activity_horizontal_margin"
11     android:paddingTop="@dimen/activity_vertical_margin"
12     android:weightSum="1">
13
14     <TextView
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:id="@+id/tv"
18         android:textColor="#F7F9F9"
19         android:textStyle="italic"
20         android:textSize="24sp"
21         android:text="@string/smart_i_o"/>
22
23     <TextView
24         android:id="@+id/device_conn"
25         android:layout_width="match_parent"

```

```

26         android:layout_height="match_parent"
27         android:layout_below="@+id/tv"
28         android:text="@string/no_device_connected"
29         android:textColor="#F7F9F9"
30         android:textSize="18sp" />
31
32     </RelativeLayout>

```

activity_main_drawer.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android">
3
4      <group android:checkableBehavior="single">
5          <item
6              android:id="@+id/nav_mouse"
7              android:icon="@drawable/ic_mouse_black_24dp"
8              android:title="@string/_3d_mouse" />
9
10         <item
11             android:id="@+id/nav_touchpad"
12             android:icon="@drawable/ic_mouse_black_24dp"
13             android:title="@string/_2d_mouse" />
14
15         <item
16             android:id="@+id/nav_keyboard"
17             android:icon="@drawable/ic_keyboard_black_24dp"
18             android:title="@string/keyboard" />
19         <item
20             android:id="@+id/nav_connect"
21             android:icon="@drawable/ic_wifi_tethering_black_24dp"
22             android:title="@string/connect" />
23     </group>
24
25     <item android:title="@string/app_info">
26         <menu>
27             <item
28                 android:id="@+id/nav_about"
29                 android:icon="@drawable/ic_announcement_black_24dp"
30                 android:title="@string/about" />
31         </menu>
32     </item>
33
34 </menu>

```

main.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android"
3        xmlns:app="http://schemas.android.com/apk/res-auto">
4      <item
5          android:id="@+id/action_settings"
6          android:orderInCategory="100"
7          android:title="@string/action_settings"
8          app:showAsAction="never" />
9  </menu>

```

array.xml

```

1  <resources>
2      <string-array name="listArray3D">
3          <item>Low</item>
4          <item>Medium</item>
5          <item>High</item>
6      </string-array>
7
8      <string-array name="listValues3D">
9          <item>1000</item>
10         <item>2000</item>
11         <item>3000</item>
12     </string-array>
13
14     <string-array name="listArray2D">
15         <item>Low</item>
16         <item>Medium</item>
17         <item>High</item>
18     </string-array>
19
20     <string-array name="listValues2D">
21         <item>1</item>
22         <item>1.5</item>
23         <item>3</item>
24     </string-array>
25 </resources>

```

colors.xml

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3   <color name="colorPrimary">#3f51b5</color>
4   <color name="colorPrimaryDark">#303f9f</color>
5   <color name="colorAccent">#FF98816F</color>
6   <color name="keyboardColor">#846479</color>
7   <color name="keyColor">#efefb2</color>
8 </resources>
```

dimens.xml

```
1 <resources>
2   <!-- Default screen margins, per the Android Design guidelines. -->
3   <dimen name="nav_header_vertical_spacing">16dp</dimen>
4   <dimen name="nav_header_height">160dp</dimen>
5   <!-- Default screen margins, per the Android Design guidelines. -->
6   <dimen name="activity_horizontal_margin">16dp</dimen>
7   <dimen name="activity_vertical_margin">16dp</dimen>
8   <dimen name="fab_margin">16dp</dimen>
9   <dimen name="text_margin">16dp</dimen>
10 </resources>
```

drawables.xml

```
1 <resources xmlns:android="http://schemas.android.com/apk/res/android">
2   <item name="ic_menu_camera" type="drawable">@android:drawable/ic_menu_camera</item>
3   <item name="ic_menu_gallery" type="drawable">@android:drawable/ic_menu_gallery</item>
4   <item name="ic_menu_slideshow"
5     type="drawable">@android:drawable/ic_menu_slideshow</item>
6   <item name="ic_menu_manage" type="drawable">@android:drawable/ic_menu_manage</item>
7   <item name="ic_menu_share" type="drawable">@android:drawable/ic_menu_share</item>
8   <item name="ic_menu_send" type="drawable">@android:drawable/ic_menu_send</item>
9 </resources>
```

strings.xml

```
1 <resources>
2   <string name="app_name">Remouse</string>
3
4   <string name="navigation_drawer_open">Open navigation drawer</string>
5   <string name="navigation_drawer_close">Close navigation drawer</string>
6
7   <string name="action_settings">Settings</string>
8   <string name="content">Select a device</string>
9   <string name="switchtext">Enable/Disable Network Connection</string>
10  <string name="mouseswitch">Turn Mouse Movement On/Off</string>
11
12  <string name="left">Left</string>
13  <string name="right">Right</string>
14  <string name="middle">Middle</string>
15  <string name="smart_i_o">Smart I/O</string>
16  <string name="keyboard">Keyboard</string>
17  <string name="connect">Connect</string>
18  <string name="app_info">App info</string>
19  <string name="about">About</string>
20  <string name="3d_mouse">3D mouse</string>
21  <string name="2d_mouse">2D Mouse</string>
22  <string name="touchpad">Touchpad</string>
23  <string name="press_move">Press hold and Move</string>
24  <string name="textView">TextView</string>
25  <string name="no_device_connected">No device connected</string>
26  <string name="Send">Send</string>
27  <string name="change_master_password">Change Master Password</string>
28  <string name="change_3d_mouse_sensitivity">Change 3D Mouse Sensitivity</string>
29  <string name="change_2d_mouse_sensitivity">Change 2D Mouse Sensitivity</string>
30  <string name="sensitivity2D" format="float" type="dimen">1.5</string>
31  <string name="title_activity_settings">Settings</string>
32  <string name="ctrl">Ctrl</string>
33  <string name="alt">Alt</string>
34  <string name="shift">Shift</string>
35  <string name="f1">F1</string>
36  <string name="f2">F2</string>
37  <string name="f3">F3</string>
38  <string name="f4">F4</string>
39  <string name="f5">F5</string>
40  <string name="f6">F6</string>
41  <string name="f7">F7</string>
42  <string name="f8">F8</string>
43  <string name="f9">F9</string>
```

```

44 <string name="f10">F10</string>
45 <string name="f11">F11</string>
46 <string name="f12">F12</string>
47 <string name="del">Del</string>
48 <string name="esc">Esc</string>
49 <string name="tab">Tab</string>
50 <string name="show_hide_text">Hide Text</string>
51 <string name="type_here">Type here</string>
52 <string name="home">Home</string>
53 <string name="end">End</string>
54 <string name="page_up">Page Up</string>
55 <string name="page_down">Page Down</string>
56 <string name="insert">Insert</string>
57 </resources>

```

styles.xml

```

1 <resources>
2
3 <!-- Base application theme. -->
4 <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
5 <!-- Customize your theme here. -->
6 <item name="colorPrimary">@color/colorPrimary</item>
7 <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
8 <item name="colorAccent">@color/colorAccent</item>
9 </style>
10
11 <style name="AppTheme.NoActionBar">
12 <item name="windowActionBar">false</item>
13 <item name="windowNoTitle">true</item>
14 </style>
15
16 <style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />
17
18 <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Light" />
19
20 <style name="CustomSwitch" parent="Theme.AppCompat.Light">
21 <!-- active thumb & track color (30% transparency) -->
22 <item name="colorControlActivated">#e6e600</item>
23 <item name="colorSwitchThumbNormal">#cc0000</item>
24 </style>
25
26
27 </resources>

```

preferences.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android"
3 <!-- android:layout_height="match_parent"
4 <!-- android:layout_width="match_parent"
5 <!--
6 <!-- android:title="@string/change_3d_mouse_sensitivity"
7 <!-- android:summary="Select the type of sensitivity"
8 <!-- android:key="sensitivity_3d"
9 <!-- android:defaultValue="2000"
10 <!-- android:entries="@array/listArray3D"
11 <!-- android:entryValues="@array/listValues3D"
12 <!-- android:layout_width="wrap_content"
13 <!-- android:layout_height="wrap_content" />
14 <!--
15 <!-- android:title="@string/change_2d_mouse_sensitivity"
16 <!-- android:summary="Select the type of sensitivity"
17 <!-- android:key="sensitivity_2d"
18 <!-- android:defaultValue="@string/sensitivity2D"
19 <!-- android:entries="@array/listArray2D"
20 <!-- android:entryValues="@array/listValues2D"
21 <!-- android:layout_width="wrap_content"
22 <!-- android:layout_height="wrap_content" />
23 </PreferenceScreen>

```

AndroidManifest.xml

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3 <!-- package="project.android">
4 <!--
5 <!-- <uses-permission android:name="android.permission.INTERNET" />
6 <!-- <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
7 <!-- <uses-permission android:name="android.permission.CHANGE_WIFI_MULTICAST_STATE" />
8 <!--
9 <!-- <uses-feature android:name="android.hardware.sensor.accelerometer" />
10 <!-- <uses-feature android:name="android.hardware.sensor.gyroscope" />
11 <!--
12 <!-- <application

```

```
13     android:allowBackup="true"
14     android:icon="@mipmap/ic_launcher"
15     android:label="@string/app_name"
16     android:supportRtl="true"
17     android:theme="@style/AppTheme">
18     <activity
19         android:name=".MainActivity"
20         android:label="@string/app_name"
21         android:theme="@style/AppTheme.NoActionBar">
22         <intent-filter>
23             <action android:name="android.intent.action.MAIN" />
24             <category android:name="android.intent.category.LAUNCHER" />
25         </intent-filter>
26     </activity>
27
28     <service
29         android:name=".net.NetworkService"
30         android:enabled="true"
31         android:exported="true" />
32
33     <activity
34         android:name=".SettingsActivity"
35         android:label="@string/title_activity_settings"
36         android:theme="@style/AppTheme.NoActionBar" />
37 </application>
38
39 </manifest>
```

4.1.2 Class Documentation

List of Classes

[detail level 1 2 3 4 5]

▼ N project

▼ N android

▼ N net

- c **BroadcastReceiverThread** Implementation of the `java.lang.Runnable` for the thread responsible for receiving broadcast from the server
- c **Client** Class representing the client
- c **ClientInfo** Encapsulation of the data sent to the server at the time of establishing TCP connection
- c **ClientIOThread** Implementation of the `java.lang.Runnable` for the thread responsible for client-server interaction
- c **ConnectionTask** Extends the `AsyncTask` class of the Android API
- c **DataWrapper** Encapsulates the data sent to the server during TCP communication
- c **KeyboardThread** Implementation of the `java.lang.Runnable` for the thread responsible for sending keyboard data to the server
- c **NetworkService** Class representing the `Service` that runs upon successful client-server connection
- c **ServerInfo** Encapsulation of the data sent during UDP broadcast by the server

▼ N security

- c **EKEProvider** Utility for end-to-end security module
- c **KeyStoreManager** Manages the operations on the `KeyStore`

▼ N sensor

▼ N orientation

- c **KalmanFilterProvider** Utility for providing the filter module
- c **OrientationProvider** Accesses the sensor hardware to read sensor data

▼ N representation

- c **Matrix** Matrix math utilities
- c **MatrixF4x4** Class to represent a 4x4 matrix (float)
- c **Quaternion** Class to represent a quaternion
- c **Vector3f** Class to represent a three dimensional vector
- c **Vector4f** Class to represent a four dimensional vector
- c **AboutFragment** Class representing the `Fragment` for displaying app information
- c **ConnectionFragment** Class representing the `Fragment` that provides GUI frontend for the network module
- c **CustomAdapter** Displays the list of available servers in a `ListView` format
- c **KeyboardFragment** Class representing the `Fragment` for providing the GUI frontend for the keyboard module

c MainActivity	Class representing the Activity that is launched at app startup
c MouseFragment	Class representing the Fragment for providing the GUI frontend for the 3D mouse module
c SettingsActivity	Class representing the Activity for app settings
c SettingsFragment	Class representing the Fragment for changing app settings
c TouchpadFragment	Class representing the Fragment for providing the GUI frontend for the 2D mouse module

project.android.net.BroadcastReceiverThread Class Reference

Implementation of the `java.lang.Runnable` for the thread responsible for receiving broadcast from the server. [More...](#)

Collaboration diagram for `project.android.net.BroadcastReceiverThread`:

Public Member Functions

BroadcastReceiverThread (Activity activity, WifiManager.MulticastLock lock)
Constructor. [More...](#)

void **run** ()
Performs the broadcast receiving operation. [More...](#)

Detailed Description

Implementation of the `java.lang.Runnable` for the thread responsible for receiving broadcast from the server.

This class receives the UDP packets broadcasted by the server and updates the `ListView` in the [project.android.ConnectionFragment](#) accordingly. This thread is started by the [ConnectionFragment#onViewCreated\(View, Bundle\)](#) method and it runs until the app is closed.

See also

- `project.android.ConnectionFragment`
- `project.android.net.ServerInfo`
- `project.android.ConnectionFragment::onViewCreated(View, Bundle)`

Definition at line **37** of file **BroadcastReceiverThread.java**.

Constructor Documentation

project.android.net.BroadcastReceiverThread.BroadcastReceiverThread (Activity activity, WifiManager.MulticastLock lock)

Constructor.

Initializes this `BroadcastReceiverThread`.

Parameters

activity The current `android.app.Activity` object.
lock an object of `WifiManager.MulticastLock` class of the Android API.

Definition at line **56** of file **BroadcastReceiverThread.java**.

Member Function Documentation

void project.android.net.BroadcastReceiverThread.run ()

Performs the broadcast receiving operation.

- Checks whether a stop signal has been broadcast by the server. If so, removes the **ServerInfo** object from the ListView in the **project.android.ConnectionFragment**.
- Constructs a **ServerInfo** object from the data obtained from the DatagramPacket using com.google.Gson API and updates the ListView in the **project.android.ConnectionFragment** accordingly.

See also

project.android.ConnectionFragment

Definition at line **84** of file **BroadcastReceiverThread.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/net/**BroadcastReceiverThread.java**

project.android.net.Client Class Reference

Class representing the client. [More...](#)

Collaboration diagram for project.android.net.Client:

Public Member Functions

Client (String address) throws IOException
Constructor. [More...](#)

void **close** () throws IOException
Closes the client socket. [More...](#)

void **sendData** (String operationType, String data)
Sends mouse button or keyboard data. [More...](#)

void **sendData** (String data)
Sends special key data. [More...](#)

void **sendData** (int x, int y)
Sends 2D mouse movement data. [More...](#)

void **sendData** (**Quaternion** quaternion, boolean isInitQuat)
Sends 3D mouse movement data. [More...](#)

void **sendStopSignal** (boolean makeSecured)
Sends a stop signal if the user disconnects the server. [More...](#)

Package Functions

BufferedReader **getSocketReader** () throws IOException
Returns BufferedReader of the client Socket. [More...](#)

void **sendPairingKey** (String pairingKey)
Sends the pairing key provided by the app user to the server. [More...](#)

void **setEKEProvider** (**EKEProvider** ekeProvider)
Sets the EKEProvider object. [More...](#)

Detailed Description

Class representing the client.

The constructor **Client(String)** is used to set up a TCP connection with the respective server. The sensor data obtained through **project.android.KeyboardFragment**, **project.android.MouseFragment** and **project.android.TouchpadFragment** are wrapped using **DataWrapper** which internally uses the `com.google.Gson` API. The wrapped *JSON string* is encrypted using the **project.android.security.EKEProvider#encryptString(String)** method and transmitted over the network.

See also

`project.android.net.DataWrapper`

`project.android.security.EKEProvider`

Definition at line **36** of file **Client.java**.

Constructor Documentation

project.android.net.Client.Client (String address) throws IOException

Constructor.

Initializes this Client.

Parameters

address IPv4 address of the server.

Exceptions

`IOException`

Definition at line **54** of file **Client.java**.

Member Function Documentation

void project.android.net.Client.close () throws IOException

Closes the client socket.

Exceptions

`IOException`

Definition at line **157** of file **Client.java**.

BufferedReader
project.android.net.Client.getSocketReader

throws
() IOException

package

Returns BufferedReader of the client Socket.

Returns
the BufferedReader object.

Exceptions
IOException

Definition at line **82** of file **Client.java**.

**void project.android.net.Client.sendData (String operationType,
String data
)**

Sends mouse button or keyboard data.

Parameters
operationType type of the operation.
data data to be sent.

See also
project.android.net.DataWrapper::DataWrapper(String, String)

Definition at line **93** of file **Client.java**.

void project.android.net.Client.sendData (String data)

Sends special key data.

Parameters
data special key data.

See also
project.android.net.DataWrapper::DataWrapper(String)

Definition at line **106** of file **Client.java**.

```
void project.android.net.Client.sendData ( int x,  
                                           int y  
                                           )
```

Sends 2D mouse movement data.

Parameters

x relative x coordinate.

y relative y coordinate.

See also

project.android.net.DataWrapper::DataWrapper(int, int, float)

Definition at line **120** of file **Client.java**.

```
void project.android.net.Client.sendData ( Quaternion quaternion,  
                                           boolean isInitQuat  
                                           )
```

Sends 3D mouse movement data.

Parameters

quaternion a **project.android.sensor.representation.Quaternion** object.

isInitQuat true, if initial Quaternion,
false, otherwise.

See also

project.android.sensor.representation.Quaternion

project.android.net.DataWrapper::DataWrapper(Quaternion, boolean, float)

Definition at line **136** of file **Client.java**.

```
void project.android.net.Client.sendPairingKey ( String pairingKey )
```

package

Sends the pairing key provided by the app user to the server.

Parameters

pairingKey the pairing key provided by the app user.

Definition at line **74** of file **Client.java**.

void project.android.net.Client.sendStopSignal (boolean **makeSecured)**

Sends a stop signal if the user disconnects the server.

Parameters

makeSecured true, for valid pairing key,
false, if pairing key alert dialog was cancelled.

Definition at line **147** of file **Client.java**.

void project.android.net.Client.setEKEProvider (EKEProvider **ekeProvider)**

package

Sets the EKEProvider object.

Parameters

ekeProvider the **project.android.security.EKEProvider** object.

See also

project.android.security.EKEProvider

Definition at line **67** of file **Client.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/net/**Client.java**

project.android.net.ClientInfo Class Reference

Encapsulation of the data sent to the server at the time of establishing TCP connection. [More...](#)

Collaboration diagram for project.android.net.ClientInfo:

Public Member Functions

ClientInfo (String clientInfo, byte[] publicKey)
Constructor. [More...](#)

String **getClientInfo** ()
Returns the canonical hostname of the client. [More...](#)

byte[] **getPublicKey** ()
Returns the public key of the client. [More...](#)

Socket **getSocket** ()
Returns the client Socket. [More...](#)

void **setSocket** (Socket socket)
Adds **java.net.Socket** of the client. [More...](#)

Detailed Description

Encapsulation of the data sent to the server at the time of establishing TCP connection.

The data contains :

- Public key of the client
- Canonical hostname of the client
- Client Socket

See also

`java.net.Socket`

`project.android.net.Client`

Definition at line **22** of file **ClientInfo.java**.

Constructor Documentation

```
project.android.net.ClientInfo.ClientInfo ( String clientInfo,  
                                             byte[] publicKey  
                                             )
```

Constructor.

Initializes this `ClientInfo` object. The stored object of `java.net.Socket` (client `Socket`) is not initialized. It can be set by the `setSocket(Socket)` method.

Parameters

clientInfo canonical hostname of the client.

publicKey public key of the client.

See also

`java.net.Socket`

`setSocket(Socket)`

Definition at line **42** of file **`ClientInfo.java`**.

Member Function Documentation

```
String project.android.net.ClientInfo.getClientInfo ( )
```

Returns the canonical hostname of the client.

Returns

canonical hostname of the client.

Definition at line **67** of file **`ClientInfo.java`**.

```
byte [] project.android.net.ClientInfo.getPublicKey ( )
```

Returns the public key of the client.

Returns

public key of the client.

Definition at line **74** of file **`ClientInfo.java`**.

Socket project.android.net.ClientInfo.getSocket ()

Returns the client Socket.

Returns

the client **java.net.Socket**.

Definition at line **60** of file **ClientInfo.java**.

void project.android.net.ClientInfo.setSocket (Socket **socket)**

Adds **java.net.Socket** of the client.

Parameters

socket the client **java.net.Socket** object to be stored.

Definition at line **53** of file **ClientInfo.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/net/**ClientInfo.java**

project.android.net.ClientIOThread Class Reference

Implementation of the `java.lang.Runnable` for the thread responsible for client-server interaction. [More...](#)

Collaboration diagram for `project.android.net.ClientIOThread`:

Public Member Functions

void **run** ()

Performs operations for reading data from a server. [More...](#)

Package Functions

ClientIOThread (Activity activity, final **ServerInfo** serverInfo, **EKEProvider** ekeProvider) throws IOException
 Constructor. [More...](#)

Detailed Description

Implementation of the `java.lang.Runnable` for the thread responsible for client-server interaction.

This thread is started by the **ConnectionTask#onPostExecute(EKEProvider)** method.

See also

```
project.android.net.ConnectionTask
project.android.net.Client
project.android.net.NetworkService
```

Definition at line 31 of file ClientIOThread.java.

Constructor Documentation

[illegible]

Constructor.

Initializes this ClientIOThread.

Parameters

- activity** the current Activity.
- serverInfo** **ServerInfo** object.
- ekeProvider** **project.android.security.EKEProvider** object.

Exceptions

IOException

Definition at line 48 of file ClientIOThread.java.

Member Function Documentation

void project.android.net.ClientIOThread.run ()

Performs operations for reading data from a server.

- If it reads the stop signal sent from server, it disables the respective server
- Starts a **NetworkService** upon successful server connection

See also

project.android.ConnectionFragment

Definition at line **74** of file **ClientIOThread.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/net/**ClientIOThread.java**

project.android.net.ConnectionTask Class Reference

Extends the AsyncTask class of the Android API. [More...](#)

Collaboration diagram for project.android.net.ConnectionTask:

Protected Member Functions

EKEProvider	doInBackground (ServerInfo...serverInfos) Overrides the AsyncTask.doInBackground(Params...) method of the Android API. More...
void	onPostExecute (EKEProvider ekeProvider) Overrides the AsyncTask.onPostExecute(Result) method of the Android API. More...
void	onPreExecute () Overrides the AsyncTask.onPreExecute() method of the Android API. More...

Detailed Description

Extends the AsyncTask class of the Android API.

It performs the following operations :

- Displaying a ProgressDialog.
- Constructing a **project.android.security.EKEProvider** object that will be used for encryption and decryption of TCP packets.
- Starting a **ClientIOThread**.

AsyncTask enables these operations to performed asynchronously by defining them in the **onPreExecute()**, **doInBackground(ServerInfo...)** and **onPostExecute(EKEProvider)** methods respectively. The starting of **ClientIOThread** waits for the construction of the **project.android.security.EKEProvider** object. A ProgressDialog is displayed for this duration.

See also

project.android.ConnectionFragment
project.android.net.ClientIOThread
project.android.net.ServerInfo
project.android.security.EKEProvider

Definition at line **43** of file **ConnectionTask.java**.

Member Function Documentation

EKEProvider

project.android.net.ConnectionTask.doInBackground (ServerInfo... **serverInfos)** protected

Overrides the AsyncTask.doInBackground(Params...) method of the Android API.

Performs a computation on a background thread.

Parameters

serverInfos **ServerInfo** object(s).

Returns

project.android.security.EKEProvider object.

See also

project.android.net.ConnectionTask::onPreExecute()

project.android.net.ConnectionTask::onPostExecute(EKEProvider)

project.android.net.ServerInfo

Definition at line **76** of file **ConnectionTask.java**.

void

project.android.net.ConnectionTask.onPostExecute (EKEProvider **ekeProvider)** protected

Overrides the AsyncTask.onPostExecute(Result) method of the Android API.

This method won't be invoked if the task was cancelled.

Parameters

ekeProvider **project.android.security.EKEProvider** object returned by the **ConnectionTask#doInBackground(ServerInfo...)** method.

See also

project.android.net.ConnectionTask::doInBackground(ServerInfo...)

project.android.net.ConnectionTask::onPreExecute()

project.android.net.ClientIOThread

project.android.security.EKEProvider

Definition at line **93** of file **ConnectionTask.java**.

void project.android.net.ConnectionTask.onPreExecute ()

protected

Overrides the AsyncTask.onPreExecute() method of the Android API.

Runs on the UI thread before the **ConnectionTask#doInBackground(ServerInfo...)** method.

See also

project.android.net.ConnectionTask::doInBackground(ServerInfo...)

project.android.net.ConnectionTask::onPostExecute(EKEProvider)

Definition at line **58** of file **ConnectionTask.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/net/**ConnectionTask.java**

project.android.net.DataWrapper Class Reference

Encapsulates the data sent to the server during TCP communication. [More...](#)

Collaboration diagram for project.android.net.DataWrapper:

Package Functions

DataWrapper (String operationType, String data)

Constructor. [More...](#)

DataWrapper (String data)

Constructor. [More...](#)

DataWrapper (int x, int y, float sensitivity)

Constructor. [More...](#)

DataWrapper (**Quaternion** quaternionObject, boolean **isInitQuat**, float sensitivity)

Constructor. [More...](#)

String **getData** ()

Returns the *data* field of this DataWrapper for keyboard and mouse button click operations. [More...](#)

String **getOperationType** ()

Returns the *type of operation* field of this DataWrapper. [More...](#)

Quaternion **getQuaternionObject** ()

Returns the **project.android.sensor.representation.Quaternion** object stored in this DataWrapper. [More...](#)

float **getSensitivity** ()

Returns the mouse movement sensitivity value stored in this DataWrapper. [More...](#)

int **getX** ()

Returns the displacement value along x-axis stored in this DataWrapper. [More...](#)

int **getY** ()

Returns the displacement value along y-axis stored in this DataWrapper. [More...](#)

boolean **isInitQuat** ()

Checks if the stored **project.android.sensor.representation.Quaternion** object is the initial one. [More...](#)

Static Package Functions

static String **getGsonString** (Object object)

Returns a JSon-string of an object. [More...](#)

Detailed Description

Encapsulates the data sent to the server during TCP communication.

This data contains all the information acquired from the `project.android.MouseFragment`, the `project.android.TouchpadFragment` and the `project.android.KeyboardFragment`. It contains a *type of operation* field which helps the server determine the operation requested by the client and a *data* field which acts as the operand for the input operations.

For keyboard input, the *type of operation* field contains `Key` and the *data* field contains the sequence of characters and keyboard strokes for input. In case of special keys, the data field contains the special key type.

For 3-D mouse movement operation, the *type of operation* field contains `Mouse_Move` and the *data* field contains a `project.android.sensor.representation.Quaternion` object.

For 2-D mouse movement operation, the *type of operation* field contains `Mouse_Touch` and the *data* field contains a pair of integers representing the displacement along the x-axis and y-axis relative to the current mouse pointer location.

For mouse button click operation, the *type of operation* field contains `Mouse_Button` and the *data* field contains any of the following :

- `left` - representing left click.
- `right` - representing right click.
- `middle` - representing middle button click.
- `upscroll` - representing scroll up.
- `downscroll` - representing scroll down.

See also

`project.android.sensor.representation.Quaternion`

`project.android.MouseFragment`

`project.android.TouchpadFragment`

`project.android.KeyboardFragment`

Definition at line **55** of file **DataWrapper.java**.

Constructor Documentation

```
project.android.net.DataWrapper.DataWrapper ( String operationType,  
                                              String data  
                                              )
```

package

Constructor.

Used to create a `DataWrapper` for keyboard and mouse button click operations.

Parameters

operationType the *type of operation* field.

data the *data* field.

Definition at line **75** of file **DataWrapper.java**.

project.android.net.DataWrapper.DataWrapper (String **data)**

package

Constructor.

Used to create a DataWrapper for special keys like Ctrl, Alt etc.

Parameters

data the *data* field.

Definition at line **88** of file **DataWrapper.java**.

```
project.android.net.DataWrapper.DataWrapper ( int    x,  
                                              int    y,  
                                              float  sensitivity  
                                              )
```

package

Constructor.

Used to create a DataWrapper for 2-D mouse movement operation. The *type of operation* field is implicit.

Parameters

x displacement data along x-axis.

y displacement data along y-axis.

sensitivity mouse movement sensitivity.

Definition at line **103** of file **DataWrapper.java**.

```
project.android.net.DataWrapper.DataWrapper ( Quaternion  quaternionObject,  
                                              boolean    isInitQuat,  
                                              float      sensitivity  
                                              )
```

package

Constructor.

Used to create a DataWrapper for 3-D mouse movement operation. The *type of operation* field is implicit.

Parameters

quaternionObject a **project.android.sensor.representation.Quaternion** object.

isInitQuat true, if quaternionObject is the initial,
false, otherwise.

sensitivity mouse movement sensitivity.

See also

project.android.sensor.representation.Quaternion

Definition at line **124** of file **DataWrapper.java**.

Member Function Documentation

String `project.android.net.DataWrapper.getData ( )`

package

Returns the *data* field of this DataWrapper for keyboard and mouse button click operations.

Returns

the *data* field.

Definition at line **156** of file **DataWrapper.java**.

static String `project.android.net.DataWrapper.getGsonString ( Object object )`

staticpackage

Returns a JSon-string of an object.

Parameters

object the input `java.lang.Object`.

Returns

a JSon-string.

See also

`com.google.gson.Gson`

Definition at line **138** of file **DataWrapper.java**.

String `project.android.net.DataWrapper.getOperationType ( )`

package

Returns the *type of operation* field of this DataWrapper.

Returns

the *type of operation* field.

Definition at line **148** of file **DataWrapper.java**.

Quaternion `project.android.net.DataWrapper.getQuaternionObject ( )`

package

Returns the `project.android.sensor.representation.Quaternion` object stored in this DataWrapper.

Returns

the `project.android.sensor.representation.Quaternion` object.

Definition at line **173** of file **DataWrapper.java**.

float `project.android.net.DataWrapper.getSensitivity ( )`

package

Returns the mouse movement sensitivity value stored in this `DataWrapper`.

Returns

the mouse movement sensitivity.

Definition at line **197** of file **`DataWrapper.java`**.

int `project.android.net.DataWrapper.getX ( )`

package

Returns the displacement value along x-axis stored in this `DataWrapper`.

Returns

the displacement along x-axis.

Definition at line **181** of file **`DataWrapper.java`**.

int `project.android.net.DataWrapper.getY ( )`

package

Returns the displacement value along y-axis stored in this `DataWrapper`.

Returns

the displacement along y-axis.

Definition at line **189** of file **`DataWrapper.java`**.

boolean `project.android.net.DataWrapper.isInitQuat ( )`

package

Checks if the stored **`project.android.sensor.representation.Quaternion`** object is the initial one.

Returns

true, if it is initial,
false, otherwise.

Definition at line **165** of file **`DataWrapper.java`**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/net/DataWrapper.java`

project.android.net.KeyboardThread Class Reference

Implementation of the `java.lang.Runnable` for the thread responsible for sending keyboard data to the server. [More...](#)

Collaboration diagram for `project.android.net.KeyboardThread`:

Public Member Functions

KeyboardThread ()

Constructor. [More...](#)

void **addToBuffer** (String data, boolean isSpecialKey)
Inserts keyboard data to the buffer. [More...](#)

void **run** ()
Performs the sending of keyboard data in the buffer to the server. [More...](#)

void **setStopFlag** ()
Sets stop flag. [More...](#)

Detailed Description

Implementation of the `java.lang.Runnable` for the thread responsible for sending keyboard data to the server.

This thread maintains a buffer to store the data to be sent to the server. It is implemented as a `java.util.concurrent.LinkedBlockingQueue`. The data contains two fields :

- A data field which represents the data to be sent, and
- A boolean flag which determines if the data represents a special key.

The data is inserted to the buffer by the listener methods defined in the **project.android.KeyboardFragment** class. This thread continuously monitors the buffer to check if data is available. If available, it sends the data.

This thread is started when the keyboard UI is loaded and the

project.android.KeyboardFragment#onCreateView(LayoutInflater, ViewGroup, Bundle) method is called. It is stopped by invoking the **setStopFlag()** method when the keyboard UI is destroyed and the **project.android.KeyboardFragment#onDestroyView()** method is called.

See also

java.lang.Runnable

java.util.concurrent.LinkedBlockingQueue

project.android.KeyboardFragment

project.android.KeyboardFragment::onCreateView(LayoutInflater, ViewGroup, Bundle)

project.android.KeyboardFragment::onDestroyView()

setStopFlag()

Definition at line **51** of file **KeyboardThread.java**.

Constructor Documentation

project.android.net.KeyboardThread.KeyboardThread ()

Constructor.

Initializes this thread.

Definition at line **64** of file **KeyboardThread.java**.

Member Function Documentation

void project.android.net.KeyboardThread.addToBuffer (String data, boolean isSpecialKey)

Inserts keyboard data to the buffer.

This data is sent to the server by this KeyboardThread.

Parameters

data the data to be sent.

isSpecialKey true, if data represents special key,
false, otherwise.

Definition at line **79** of file **KeyboardThread.java**.

void project.android.net.KeyboardThread.run ()

Performs the sending of keyboard data in the buffer to the server.

It checks if data is available in the buffer.

- If available, it sends the data.
- Otherwise, it waits for the data to be available.

Definition at line **104** of file **KeyboardThread.java**.

void project.android.net.KeyboardThread.setStopFlag ()

Sets stop flag.

Used to exit from the **run()** method.

Definition at line **90** of file **KeyboardThread.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/net/**KeyboardThread.java**

project.android.net.NetworkService Class Reference

Class representing the Service that runs upon successful client-server connection.
[More...](#)

Collaboration diagram for project.android.net.NetworkService:

Public Member Functions

- `IBinder` **onBind** (Intent intent)
Overrides the `Service.onBind(Intent)` method of the Android API. [More...](#)
-
- `void` **onDestroy** ()
Overrides the `Service.onDestroy()` method of the Android API. [More...](#)
-
- `int` **onStartCommand** (Intent intent, int flags, int startId)
Overrides the `Service.onStartCommand(Intent, int, int)` method of the Android API. [More...](#)
-

Detailed Description

Class representing the Service that runs upon successful client-server connection.

ClientIOThread starts this `NetworkService` when a successful client-server connection has been established. This `NetworkService` is destroyed when the **onDestroy()** method is invoked upon app closing.

See also

- `project.android.net.ConnectionTask::onPostExecute(EKEProvider)`
- `project.android.net.ClientIOThread::recieveData()`
- `project.android.MainActivity::onDestroy()`

Definition at line **32** of file **NetworkService.java**.

Member Function Documentation

IBinder project.android.net.NetworkService.onBind (Intent intent)

Overrides the `Service.onBind(Intent)` method of the Android API.

Return the communication channel to the service. May return `null`, if clients can not bind to the service.

Parameters

intent The Intent that was used to bind to this service, as given to `Context.bindService`. Note that any extras that were included with the Intent at that point will not be seen here.

Returns

IBinder through which clients can call on to the service.

Definition at line **49** of file **NetworkService.java**.

void project.android.net.NetworkService.onDestroy ()

Overrides the `Service.onDestroy()` method of the Android API.

Called by the system to notify a Service that it is no longer used and is being removed. The service should clean up any resources it holds (threads, registered receivers, etc) at this point. Upon return, there will be no more calls in to this `android.app.Service` object and it is effectively dead.

Definition at line **92** of file **NetworkService.java**.

```
int project.android.net.NetworkService.onStartCommand ( Intent intent,  
                                                         int flags,  
                                                         int startId  
                                                         )
```

Overrides the `Service.onStartCommand(Intent, int, int)` method of the Android API.

Called by the system every time a client explicitly starts the service by calling `startService(Intent)`, providing the arguments it supplied and a unique integer token representing the start request.

Parameters

intent the Intent supplied to `startService(Intent)`, as given. This may be null if the service is being restarted after its process has gone away, and it had previously returned anything except `START_STICKY_COMPATIBILITY`.

flags the additional data about this start request. Currently either 0, `START_FLAG_REDELIVERY` or `START_FLAG_RETRY`.

startId a unique integer representing this specific request to start.

Returns

an integer indicating the semantics the system should use for the service's current started state.

Definition at line **71** of file **NetworkService.java**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/net/NetworkService.java`

project.android.net.ServerInfo Class Reference

Encapsulation of the data sent during UDP broadcast by the server. [More...](#)

Collaboration diagram for project.android.net.ServerInfo:

Public Member Functions

ServerInfo (byte[] publicKey, String serverInfo)
Constructor. [More...](#)

boolean **equals** (Object obj)
Compares two ServerInfo objects. [More...](#)

String **getAddress** ()
Returns the IPv4 address of the server. [More...](#)

String **getServerInfo** ()
Returns the canonical host name of the server. [More...](#)

void **setPairingKey** (String pairingKey)
Sets the pairing key used during client-server authentication. [More...](#)

Package Functions

String **getPairingKey** ()
Returns the pairing key used during client-server authentication. [More...](#)

byte[] **getServerPubKey** ()
Returns the server public key. [More...](#)

boolean **getStopFlag** ()
Returns the status of the server. [More...](#)

void **setServerAddress** (String address)
Sets the IPv4 server address. [More...](#)

Detailed Description

Encapsulation of the data sent during UDP broadcast by the server.

This data is received by the **BroadcastReceiverThread**. This contains :

- the public key of the server,
- status of the broadcast, and
- address of the server

This message helps a client to discover a server.

See also

`project.android.net.BroadcastReceiverThread`

Definition at line **21** of file **ServerInfo.java**.

Constructor Documentation

```
project.android.net.ServerInfo.ServerInfo ( byte[] publicKey,  
                                             String serverInfo  
                                             )
```

Constructor.

Initializes this `ServerInfo`.

Parameters

publicKey public key of the server.

serverInfo canonical hostname of the server.

Definition at line **37** of file **ServerInfo.java**.

Member Function Documentation

```
boolean project.android.net.ServerInfo.equals ( Object obj )
```

Compares two `ServerInfo` objects.

Parameters

obj the object to be compared.

Returns

true, if the two objects are equal,
false, otherwise.

Definition at line **51** of file **ServerInfo.java**.

String project.android.net.ServerInfo.getAddress ()

Returns the IPv4 address of the server.

Returns

the IPv4 server address.

Definition at line **93** of file **ServerInfo.java**.

String project.android.net.ServerInfo.getPairingKey ()

package

Returns the pairing key used during client-server authentication.

Returns

the pairing key.

Definition at line **99** of file **ServerInfo.java**.

String project.android.net.ServerInfo.getServerInfo ()

Returns the canonical host name of the server.

Returns

the canonical host name.

Definition at line **87** of file **ServerInfo.java**.

byte [] project.android.net.ServerInfo.getServerPubKey ()

package

Returns the server public key.

Returns

server public key.

Definition at line **81** of file **ServerInfo.java**.

boolean project.android.net.ServerInfo.getStopFlag ()

package

Returns the status of the server.

Returns

server status.

Definition at line **75** of file **ServerInfo.java**.

void project.android.net.ServerInfo.setPairingKey (String **pairingKey)**

Sets the pairing key used during client-server authentication.

Parameters

pairingKey the pairing key.

Definition at line **69** of file **ServerInfo.java**.

void project.android.net.ServerInfo.setServerAddress (String **address)**

package

Sets the IPv4 server address.

Parameters

address IPv4 server address.

Definition at line **63** of file **ServerInfo.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/net/**ServerInfo.java**

project.android.security.EKEProvider Class Reference

Utility for end-to-end security module. [More...](#)

Collaboration diagram for project.android.security.EKEProvider:

Public Member Functions

EKEProvider ()

Constructor. [More...](#)

EKEProvider (byte[] pairingKey, byte[] serverPublicKey)

Constructor. [More...](#)

String **decryptString** (String cipherText)

Decrypts ciphertext using *AES 256-bit* bit decryption algorithm. [More...](#)

String **encryptString** (String plainText)

Encrypts plaintext using *AES 256-bit* encryption algorithm. [More...](#)

byte[] **getBase64EncodedPubKey** ()

Returns the Base64 encoded client PublicKey. [More...](#)

Detailed Description

Utility for end-to-end security module.

This class provides the following features:

- *256-bit Elliptic Curve* private and public key generation.
- *AES 256-bit* plaintext encryption.
- *AES 256-bit* ciphertext decryption.
- Self-signed certificate generation.

Definition at line **63** of file **EKEProvider.java**.

Constructor Documentation

project.android.security.EKEProvider.EKEProvider ()

Constructor.

Initializes this EKEProvider. If keystore exists, private and public `java.security.KeyPair` is loaded, otherwise the `KeyPair` is generated.

Definition at line **94** of file **EKEProvider.java**.

project.android.security.EKEProvider.EKEProvider (byte[] pairingKey, byte[] serverPublicKey)

Constructor.

Initializes this EKEProvider. Generates a secret key using the *SHA-256 message digest* of the pairing key used for client-server authentication and the server `PublicKey`.

Parameters

pairingKey the pairing key.
serverPublicKey public key of the server.

Definition at line **127** of file **EKEProvider.java**.

Member Function Documentation

String project.android.security.EKEProvider.decryptString (String cipherText)

Decrypts ciphertext using *AES 256-bit* decryption algorithm.

Parameters

cipherText the ciphertext to be decrypted.

Returns

the plaintext.

Definition at line **208** of file **EKEProvider.java**.

String `project.android.security.EKEProvider.encryptString ( String plainText )`

Encrypts plaintext using *AES 256-bit* encryption algorithm.

Parameters

plainText the plaintext to be encrypted.

Returns

the ciphertext.

Definition at line **188** of file **EKEProvider.java**.

byte [] `project.android.security.EKEProvider.getBase64EncodedPubKey ( )`

Returns the Base64 encoded client `PublicKey`.

Returns

the Base64 encoded server public key.

Note: Returns null if no public has been generated during instantiation.

Definition at line **152** of file **EKEProvider.java**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/security/EKEProvider.java`

project.android.security.KeyStoreManager Class Reference

Manages the operations on the KeyStore. [More...](#)

Collaboration diagram for project.android.security.KeyStoreManager:

Package Functions

KeyStoreManager () throws `CertificateException`, `NoSuchAlgorithmException`, `KeyStoreException`, `IOException`
Constructor. [More...](#)

void **setMasterKey** (`PrivateKey` privateKey, `Certificate...`cert) throws `KeyStoreException`, `IOException`, `CertificateException`, `NoSuchAlgorithmException`
Method to store the private key and the certificate in the KeyStore. [More...](#)

Static Package Functions

static `KeyPair` **getKSKeyPair** () throws `IOException`, `KeyStoreException`, `CertificateException`, `NoSuchAlgorithmException`, `UnrecoverableEntryException`
Returns the private-public key pair. [More...](#)

static boolean **keyStoreExists** ()
Checks if a KeyStore exists. [More...](#)

Detailed Description

Manages the operations on the KeyStore.

This class provides a [java.security.KeyStore](#) which stores :

- The private key ([java.security.PrivateKey](#)).
- A self-signed certificate ([java.security.cert.Certificate](#)) that contains the public key and the digital signature.

The KeyStore is protected by a randomly generated password.

See also

[project.android.security.EKEProvider](#)
[java.security.KeyStore](#)
[java.security.KeyPair](#)
[java.security.cert.Certificate](#)

Definition at line [46](#) of file [KeyStoreManager.java](#).

Constructor Documentation

throws `CertificateException`,
`NoSuchAlgorithmException`,
`KeyStoreException`, `IOException`

project.android.security.KeyStoreManager.KeyStoreManager () package

Constructor.

Initializes this `KeyStoreManager`.

Exceptions

- `CertificateException`
- `NoSuchAlgorithmException`
- `KeyStoreException`
- `IOException`

Definition at line **64** of file `KeyStoreManager.java`.

Member Function Documentation

throws `IOException`,
`KeyStoreException`,
`CertificateException`,
`NoSuchAlgorithmException`,
`UnrecoverableEntryException`

static KeyPair
project.android.security.KeyStoreManager.getKSKeyPair () static package

Returns the private-public key pair.

Returns

`java.security.KeyPair` object.

Exceptions

- `IOException`
- `KeyStoreException`
- `CertificateException`
- `NoSuchAlgorithmException`
- `UnrecoverableEntryException`

Definition at line **104** of file `KeyStoreManager.java`.

static boolean **project.android.security.KeyStoreManager.keyStoreExists ()** static package

Checks if a `KeyStore` exists.

Returns

- `true`, if the `KeyStore` exists,
- `false`, otherwise.

Definition at line **75** of file `KeyStoreManager.java`.

```

void
project.android.security.KeyStoreManager.setMasterKey ( PrivateKey  privateKey,
                                                         Certificate... cert
                                                         throws KeyStoreException,
                                                         IOException,
                                                         CertificateException,
                                                         NoSuchAlgorithmException
                                                         )

```

package

Method to store the private key and the certificate in the KeyStore.

Parameters

privateKey the [java.security.PrivateKey](#) object.
cert the [java.security.cert.Certificate](#) object(s).

Exceptions

[KeyStoreException](#)
[IOException](#)
[CertificateException](#)
[NoSuchAlgorithmException](#)

Definition at line **88** of file [KeyStoreManager.java](#).

The documentation for this class was generated from the following file:

- [app/src/main/java/project/android/security/KeyStoreManager.java](#)

project.android.sensor.orientation.KalmanFilterProvider Class Reference

Utility for providing the filter module. [More...](#)

Collaboration diagram for project.android.sensor.orientation.KalmanFilterProvider:

Public Member Functions

KalmanFilterProvider (SensorManager sensorManager)
Constructor. [More...](#)

void **onSensorChanged** (SensorEvent event)
Overrides the SensorEventListener.onSensorChanged() method of the Android API. [More...](#)

▼ **Public Member Functions inherited from project.android.sensor.orientation.OrientationProvider**

void **getEulerAngles** (float angles[])
Gets the current rotation of the device. [More...](#)

void **getQuaternion** (Quaternion quaternion)
Gets the current rotation of the device. [More...](#)

void **getRotationMatrix** (MatrixF4x4 matrix)

Gets the current rotation of the device. [More...](#)

void **sensorStop** ()

Stops the sensor fusion (e.g., when pausing/suspending the activity). [More...](#)

Additional Inherited Members

▼ Static Public Member Functions inherited from `project.android.sensor.orientation.OrientationProvider`

static boolean **checkGyro** (Activity activity)

Checks if gyroscope is available in the device. [More...](#)

▼ Package Functions inherited from `project.android.sensor.orientation.OrientationProvider`

OrientationProvider (SensorManager sensorManager)

Constructor. [More...](#)

void **sensorStart** ()

Starts the sensor fusion (e.g., when resuming the activity). [More...](#)

Detailed Description

Utility for providing the filter module.

This class provides methods that deliver the relative orientation from the gyroscope present in the Android device. This sensor does not deliver an absolute orientation (with respect to magnetic north and gravity) but only a relative measurement starting from the point where it started.

It extends the **OrientationProvider** class which is responsible for accessing the sensor hardware.

See also

`project.android.sensor.orientation.OrientationProvider`

Definition at line **29** of file **KalmanFilterProvider.java**.

Constructor Documentation

project.android.sensor.orientation.KalmanFilterProvider.KalmanFilterProvider (SensorManager **sensorManager)**

Constructor.

Initializes this KalmanFilterProvider.

Parameters

sensorManager a SensorManager object.

Definition at line **48** of file **KalmanFilterProvider.java**.

Member Function Documentation

void project.android.sensor.orientation.KalmanFilterProvider.onSensorChanged (SensorEvent **event)**

Overrides the `SensorEventListener.onSensorChanged()` method of the Android API.

Called when there is a new sensor event. It is also called if there is a reading available from a sensor with the exact same sensor values (but a newer timestamp).

Parameters

event the SensorEvent object containing full information about the event.

Definition at line **69** of file **KalmanFilterProvider.java**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/sensor/orientation/KalmanFilterProvider.java`

project.android.sensor.orientation.OrientationProvider Class Reference abstract

Accesses the sensor hardware to read sensor data. [More...](#)

Collaboration diagram for project.android.sensor.orientation.OrientationProvider:

Public Member Functions

- void **getEulerAngles** (float angles[])
Gets the current rotation of the device. [More...](#)
- void **getQuaternion** (**Quaternion** quaternion)
Gets the current rotation of the device. [More...](#)
- void **getRotationMatrix** (**MatrixF4x4** matrix)
Gets the current rotation of the device. [More...](#)
- void **sensorStop** ()
Stops the sensor fusion (e.g., when pausing/suspending the activity). [More...](#)

Static Public Member Functions

- static boolean **checkGyro** (Activity activity)
Checks if gyroscope is available in the device. [More...](#)

Package Functions

- OrientationProvider** (SensorManager sensorManager)
Constructor. [More...](#)
- void **sensorStart** ()
Starts the sensor fusion (e.g., when resuming the activity). [More...](#)

Detailed Description

Accesses the sensor hardware to read sensor data.

Classes implementing this interface provide an orientation of the device either by directly accessing hardware, using Android sensor fusion or fusing sensors itself. The orientation can be provided as a rotation matrix ([project.android.sensor.representation.MatrixF4x4](#)) or a quaternion ([project.android.sensor.representation.Quaternion](#)).

See also

- [project.android.sensor.representation.Quaternion](#)
- [project.android.sensor.representation.MatrixF4x4](#)
- [project.android.sensor.orientation.KalmanFilterProvider](#)

Definition at line [29](#) of file [OrientationProvider.java](#).

Constructor Documentation

project.android.sensor.orientation.OrientationProvider.OrientationProvider ([SensorManager](#) [sensorManager](#)) package

Constructor.

Initializes this OrientationProvider.

Parameters

[sensorManager](#) a [SensorManager](#) object.

Definition at line [48](#) of file [OrientationProvider.java](#).

Member Function Documentation

static boolean project.android.sensor.orientation.OrientationProvider.checkGyro ([Activity](#) [activity](#)) static

Checks if gyroscope is available in the device.

Parameters

[activity](#) the current Activity.

Returns

true, if gyroscope is available,
false, otherwise.

Definition at line [65](#) of file [OrientationProvider.java](#).

void project.android.sensor.orientation.OrientationProvider.getEulerAngles (float [angles](#)[])

Gets the current rotation of the device.

Parameters

[angles](#) a float array representing the rotation.

Definition at line [120](#) of file [OrientationProvider.java](#).

void project.android.sensor.orientation.OrientationProvider.getQuaternion ([Quaternion](#) [quaternion](#))

Gets the current rotation of the device.

Parameters

[quaternion](#) a [project.android.sensor.representation.Quaternion](#) object representing the rotation.

See also

- [project.android.sensor.representation.Quaternion](#)

Definition at line [109](#) of file [OrientationProvider.java](#).

void project.android.sensor.orientation.OrientationProvider.getRotationMatrix ([MatrixF4x4](#) **matrix)**

Gets the current rotation of the device.

Parameters

matrix a [project.android.sensor.representation.MatrixF4x4](#) object representing the rotation.

See also

[project.android.sensor.representation.MatrixF4x4](#)

Definition at line **96** of file [OrientationProvider.java](#).

void project.android.sensor.orientation.OrientationProvider.sensorStart ()

package

Starts the sensor fusion (e.g., when resuming the activity).

Definition at line **73** of file [OrientationProvider.java](#).

void project.android.sensor.orientation.OrientationProvider.sensorStop ()

Stops the sensor fusion (e.g., when pausing/suspending the activity).

Definition at line **83** of file [OrientationProvider.java](#).

The documentation for this class was generated from the following file:

- [app/src/main/java/project/android/sensor/orientation/OrientationProvider.java](#)

project.android.sensor.representation.Matrix Class Reference

Matrix math utilities. [More...](#)

Collaboration diagram for project.android.sensor.representation.Matrix:

Static Public Member Functions

static void **frustumM** (float[] m, int offset, float left, float right, float bottom, float top, float near, float far)
Defines a projection matrix in terms of six clip planes. [More...](#)

static boolean **invertM** (float[] mInv, int mInvOffset, float[] m, int mOffset)
Inverts a 4 x 4 matrix. [More...](#)

static void **multiplyMM** (float[] output, float[] lhs, float[] rhs)
Multiplies two 4x4 matrices. [More...](#)

static void **multiplyMV** (float[] output, int outputOffset, float[] lhs, int lhsOffset, float[] rhs, int rhsOffset)
Multiplies a 4 element vector by a 4x4 matrix. [More...](#)

static void **orthoM** (float[] m, int mOffset, float left, float right, float bottom, float top, float near, float far)
Computes an orthographic projection matrix. [More...](#)

static void **perspectiveM** (float[] m, int offset, float fovy, float aspect, float zNear, float zFar)
Defines a projection matrix in terms of a field of view angle, an aspect ratio, and z clip planes. [More...](#)

static void **rotateM** (float[] rm, int rmOffset, float[] m, int mOffset, float a, float x, float y, float z)
Rotates matrix m in place by angle a (in degrees) around the axis (x, y, z). [More...](#)

static void **scaleM** (float[] sm, int smOffset, float[] m, int mOffset, float x, float y, float z)
Scales matrix m by x, y, and z. [More...](#)

static void **setRotateEulerM** (float[] rm, int rmOffset, float x, float y, float z)
Converts Euler angles to a rotation matrix. [More...](#)

static void **translateM** (float[] tm, int tmOffset, float[] m, int mOffset, float x, float y, float z)
Translates matrix m by x, y, and z. [More...](#)

static void **transposeM** (float[] mTrans, int mTransOffset, float[] m, int mOffset)
Transposes a 4 x 4 matrix. [More...](#)

Static Package Functions

static void **setIdentityM** (float[] sm, int smOffset)
Sets matrix to the identity matrix. [More...](#)

Detailed Description

Matrix math utilities.

These methods operate on OpenGL ES format matrices and vectors stored in float arrays. Matrices are 4 x 4 column-vector(float) matrices stored in column-major order:

m[offset + 0]	m[offset + 4]	m[offset + 8]	m[offset + 12]
m[offset + 1]	m[offset + 5]	m[offset + 9]	m[offset + 13]
m[offset + 2]	m[offset + 6]	m[offset + 10]	m[offset + 14]
m[offset + 3]	m[offset + 7]	m[offset + 11]	m[offset + 15]

Vectors are 4 row x 1 column column-vectors stored in order:

```
v[offset + 0]
v[offset + 1]
v[offset + 2]
v[offset + 3]
```

See also

`project.android.sensor.representation.MatrixF4x4`

Definition at line **31** of file **Matrix.java**.

Member Function Documentation

```
static void project.android.sensor.representation.Matrix.frustumM ( float[] m,
                                                                    int    offset,
                                                                    float  left,
                                                                    float  right,
                                                                    float  bottom,
                                                                    float  top,
                                                                    float  near,
                                                                    float  far
                                                                    )
```

static

Defines a projection matrix in terms of six clip planes.

Parameters

- m** the float array that holds the perspective matrix.
- offset** the offset into m where the perspective matrix data is written.
- left** left clipping value.
- right** right clipping value.
- bottom** bottom clipping value.
- top** top clipping value.
- near** nearer depth clipping value.
- far** farther depth clipping value.

Definition at line **334** of file **Matrix.java**.

```
static boolean project.android.sensor.representation.Matrix.invertM ( float[] mInv,
                                                                    int    mInvOffset,
                                                                    float[] m,
                                                                    int    mOffset
                                                                    )
```

static

Inverts a 4 x 4 matrix.

Parameters

- mInv** the array that holds the output inverted matrix.
- mInvOffset** an offset into mInv where the inverted matrix is stored.
- m** the input array.
- mOffset** an offset into m where the matrix is stored.

Returns

- true, if the matrix could be inverted,
- false, otherwise.

Definition at line **155** of file **Matrix.java**.

```
static void project.android.sensor.representation.Matrix.multiplyMM ( float[] output,
                                                                    float[] lhs,
                                                                    float[] rhs
                                                                    )
```

static

Multiplies two 4x4 matrices.

Parameters

- output** the float array that holds the result.
- lhs** the float array that holds the left-hand-side matrix.
- rhs** the float array that holds the right-hand-side matrix.

Exceptions

- IllegalArgumentException

Definition at line **66** of file **Matrix.java**.

```
static void project.android.sensor.representation.Matrix.multiplyMV ( float[] output,
                                                                    int    outputOffset,
                                                                    float[] lhs,
                                                                    int    lhsOffset,
                                                                    float[] rhs,
                                                                    int    rhsOffset
                                                                    )
```

static

Multiplies a 4 element vector by a 4x4 matrix.

It stores the result in a 4 element column vector. In matrix notation:

$$output = lhs \times rhs$$

Parameters

- output** the float array that holds the result vector.
- outputOffset** the offset into the result array where the result vector is stored.
- lhs** the float array that holds the left-hand-side matrix.
- lhsOffset** the offset into the lhs array.
- rhs** the float array that holds the right-hand-side vector.
- rhsOffset** the offset into the rhs array.

Exceptions

`IllegalArgumentException`

Definition at line **105** of file **Matrix.java**.

```
static void project.android.sensor.representation.Matrix.orthoM ( float[] m,
                                                                    int    mOffset,
                                                                    float  left,
                                                                    float  right,
                                                                    float  bottom,
                                                                    float  top,
                                                                    float  near,
                                                                    float  far
                                                                    )
```

static

Computes an orthographic projection matrix.

Parameters

- m** returns the result.
- mOffset** offset value.
- left** left clipping value.
- right** right clipping value.
- bottom** bottom clipping value.
- top** top clipping value.
- near** nearer depth clipping value.
- far** farther depth clipping value.

Definition at line **283** of file **Matrix.java**.

```
static void project.android.sensor.representation.Matrix.perspectiveM ( float[] m,
                                                                    int    offset,
                                                                    float  fovy,
                                                                    float  aspect,
                                                                    float  zNear,
                                                                    float  zFar
                                                                    )
```

static

Defines a projection matrix in terms of a field of view angle, an aspect ratio, and z clip planes.

Parameters

- m** the float array that holds the perspective matrix.
- offset** the offset into float array m where the perspective matrix data is written.
- fovy** field of view in y direction, in degrees.
- aspect** width to height aspect ratio of the viewport.
- zNear** nearer depth clipping value on Z-plane.
- zFar** farther depth clipping value on Z-plane.

Definition at line **390** of file **Matrix.java**.

```
static void project.android.sensor.representation.Matrix.rotateM ( float[] rm,
                                                                    int    rmOffset,
                                                                    float[] m,
                                                                    int    mOffset,
                                                                    float  a,
                                                                    float  x,
                                                                    float  y,
                                                                    float  z
                                                                    )
```

static

Rotates matrix m in place by angle a (in degrees) around the axis (x, y, z).

Parameters

- m** source matrix.
- mOffset** index into m where the matrix starts.
- a** angle to rotate in degrees.
- x** scale factor x.
- y** scale factor y.
- z** scale factor z.

Definition at line **504** of file **Matrix.java**.

```
static void project.android.sensor.representation.Matrix.scaleM ( float[] sm,
                                                                    int    smOffset,
                                                                    float[] m,
                                                                    int    mOffset,
                                                                    float  x,
                                                                    float  y,
                                                                    float  z
                                                                    )
```

static

Scales matrix m by x, y, and z.

Parameters

sm the result.
smOffset index into sm where the result matrix starts.
m source matrix.
mOffset index into m where the source matrix starts.
x scale factor x.
y scale factor y.
z scale factor z.

Definition at line **445** of file **Matrix.java**.

```
static void
project.android.sensor.representation.Matrix.setIdentityM ( float[] sm,
                                                            int    smOffset
                                                            )
```

static

package

Sets matrix to the identity matrix.

Parameters

sm the result.
smOffset index into sm where the result matrix starts.

Definition at line **425** of file **Matrix.java**.

```
static void
project.android.sensor.representation.Matrix.setRotateEulerM      ( float[]  rm,
                                                                    int      rmOffset,
                                                                    float    x,
                                                                    float    y,
                                                                    float    z
                                                                    )
```

static

Converts Euler angles to a rotation matrix.

Parameters

- rm** the result.
- rmOffset** index into rm where the result matrix starts.
- x** angle of rotation, in degrees.
- y** angle of rotation, in degrees.
- z** angle of rotation, in degrees.

Definition at line **596** of file **Matrix.java**.

```
static void project.android.sensor.representation.Matrix.translateM ( float[]  tm,
                                                                    int      tmOffset,
                                                                    float[]  m,
                                                                    int      mOffset,
                                                                    float    x,
                                                                    float    y,
                                                                    float    z
                                                                    )
```

static

Translates matrix m by x, y, and z.

Parameters

- tm** the result.
- tmOffset** index into sm where the result matrix starts.
- m** source matrix.
- mOffset** index into m where the source matrix starts.
- x** translation factor x.
- y** translation factor y.
- z** translation factor z.

Definition at line **476** of file **Matrix.java**.


```
static void  
project.android.sensor.representation.Matrix.transposeM
```

```
( float[] mTrans,  
  int    mTransOffset,  
  float[] m,  
  int    mOffset  
)
```

static

Transposes a 4 x 4 matrix.

Parameters

- mTrans** the array that holds the output inverted matrix.
- mTransOffset** an offset where the inverted matrix is stored.
- m** the input array.
- mOffset** an offset into m where the matrix is stored.

Definition at line **134** of file **Matrix.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/sensor/representation/**Matrix.java**

project.android.sensor.representation.MatrixF4x4 Class Reference

Class to represent a 4x4 matrix (float). [More...](#)

Collaboration diagram for project.android.sensor.representation.MatrixF4x4:

Public Member Functions

- MatrixF4x4** ()
Constructor. [More...](#)
- void **multiplyMatrix4x4ByMatrix** (**MatrixF4x4** matrixf)
Multiplies a MatrixF4x4 with this. [More...](#)
- void **multiplyVector3fByMatrix** (**Vector3f** vector) throws InvalidParameterSpecException
Multiplies the given vector by this MatrixF4x4. [More...](#)
- void **multiplyVector4fByMatrix** (**Vector4f** vector) throws InvalidParameterSpecException
Multiplies the given vector by this MatrixF4x4. [More...](#)
- void **set** (**MatrixF4x4** source)
Sets the matrix from a float[16] array. [More...](#)

Package Functions

- float[] **getMatrix** ()
Returns the matrix. [More...](#)
- void **setW0** (float value)
Sets the value of W0. [More...](#)
- void **setW1** (float value)
Sets the value of W1. [More...](#)
- void **setW2** (float value)
Sets the value of W2. [More...](#)
- void **setW3** (float value)
Sets the value of W3. [More...](#)
- void **setX0** (float value)
Sets the value of X0. [More...](#)
- void **setX1** (float value)
Sets the value of X1. [More...](#)
- void **setX2** (float value)
Sets the value of X2. [More...](#)
- void **setX3** (float value)
Sets the value of X3. [More...](#)
- void **setY0** (float value)
Sets the value of Y0. [More...](#)
- void **setY1** (float value)
Sets the value of Y1. [More...](#)
- void **setY2** (float value)
Sets the value of Y2. [More...](#)
- void **setY3** (float value)
Sets the value of Y3. [More...](#)
- void **setZ0** (float value)
Sets the value of Z0. [More...](#)
- void **setZ1** (float value)
Sets the value of Z1. [More...](#)

void	setZ2 (float value)	Sets the value of Z2. More...
void	setZ3 (float value)	Sets the value of Z3. More...
int	size ()	Returns size of the matrix. More...

Detailed Description

Class to represent a 4x4 matrix (float).

Internal the matrix is structured as :

```
[ x0 , y0 , z0 , w0 ]
[ x1 , y1 , z1 , w1 ]
[ x2 , y2 , z2 , w2 ]
[ x3 , y3 , z3 , w3 ]
```

See also

[project.android.sensor.representation.Matrix](#)

Definition at line **21** of file [MatrixF4x4.java](#).

Constructor Documentation

project.android.sensor.representation.MatrixF4x4.MatrixF4x4 ()

Constructor.

Initializes this `MatrixF4x4`.

Definition at line **40** of file [MatrixF4x4.java](#).

Member Function Documentation

float [] project.android.sensor.representation.MatrixF4x4.getMatrix ()

package

Returns the matrix.

Returns

the matrix,
null, if the matrix is invalid.

Definition at line **51** of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.multiplyMatrix4x4ByMatrix ([MatrixF4x4](#) matrixf)

Multiplies a `MatrixF4x4` with this.

Parameters

matrixf a `MatrixF4x4`.

Definition at line **186** of file [MatrixF4x4.java](#).

void **throws**
project.android.sensor.representation.MatrixF4x4.multiplyVector3fByMatrix ([Vector3f](#) **vector) InvalidParameterSpecException**

Multiplies the given vector by this MatrixF4x4.

This should only be used if the matrix is of size 9.

Parameters

vector a [Vector3f](#).

Exceptions

[InvalidParameterSpecException](#)

Definition at line **146** of file [MatrixF4x4.java](#).

void **throws**
project.android.sensor.representation.MatrixF4x4.multiplyVector4fByMatrix ([Vector4f](#) **vector) InvalidParameterSpecException**

Multiplies the given vector by this MatrixF4x4.

This should only be used if the matrix is of size 16.

Parameters

vector a [Vector4f](#).

Exceptions

[InvalidParameterSpecException](#)

Definition at line **99** of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.set ([MatrixF4x4](#) **source)**

Sets the matrix from a float[16] array.

If the matrix isn't 16 long then the matrix will be invalid.

Parameters

source the source matrixF4x4.

Definition at line **79** of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setW0 (float **value)**

package

Sets the value of W0.

Parameters

value value to set.

Definition at line **489** of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setW1 (float **value)**

package

Sets the value of W1.

Parameters

value value to set.

Definition at line **507** of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setW2 (float **value)**

package

Sets the value of W2.

Parameters

value value to set.

Definition at line **525** of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setW3 (float value)

package

Sets the value of W3.

Parameters

value value to set.

Definition at line 543 of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setX0 (float value)

package

Sets the value of X0.

Parameters

value value to set.

Definition at line 237 of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setX1 (float value)

package

Sets the value of X1.

Parameters

value value to set.

Definition at line 259 of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setX2 (float value)

package

Sets the value of X2.

Parameters

value value to set.

Definition at line 281 of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setX3 (float value)

package

Sets the value of X3.

Parameters

value value to set.

Definition at line 435 of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setY0 (float value)

package

Sets the value of Y0.

Parameters

value value to set.

Definition at line 303 of file [MatrixF4x4.java](#).

void project.android.sensor.representation.MatrixF4x4.setY1 (float value)

package

Sets the value of Y1.

Parameters

value value to set.

Definition at line 325 of file [MatrixF4x4.java](#).

void `project.android.sensor.representation.MatrixF4x4.setY2 ( float value )`

package

Sets the value of Y2.

Parameters

value value to set.

Definition at line 347 of file `MatrixF4x4.java`.

void `project.android.sensor.representation.MatrixF4x4.setY3 ( float value )`

package

Sets the value of Y3.

Parameters

value value to set.

Definition at line 453 of file `MatrixF4x4.java`.

void `project.android.sensor.representation.MatrixF4x4.setZ0 ( float value )`

package

Sets the value of Z0.

Parameters

value value to set.

Definition at line 369 of file `MatrixF4x4.java`.

void `project.android.sensor.representation.MatrixF4x4.setZ1 ( float value )`

package

Sets the value of Z1.

Parameters

value value to set.

Definition at line 391 of file `MatrixF4x4.java`.

void `project.android.sensor.representation.MatrixF4x4.setZ2 ( float value )`

package

Sets the value of Z2.

Parameters

value value to set.

Definition at line 413 of file `MatrixF4x4.java`.

void `project.android.sensor.representation.MatrixF4x4.setZ3 ( float value )`

package

Sets the value of Z3.

Parameters

value value to set.

Definition at line 471 of file `MatrixF4x4.java`.

int `project.android.sensor.representation.MatrixF4x4.size ( )`

package

Returns size of the matrix.

Returns

the size of the matrix.

Definition at line 60 of file `MatrixF4x4.java`.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/sensor/representation/MatrixF4x4.java`

project.android.sensor.representation.Quaternion Class Reference

Class to represent a quaternion. [More...](#)

Collaboration diagram for project.android.sensor.representation.Quaternion:

Public Member Functions

Quaternion ()

Constructor. [More...](#)

void **multiplyByQuat** (**Quaternion** input, **Quaternion** output)
Multiplies this Quaternion with another. [More...](#)

void **multiplyByQuat** (**Quaternion** input)
Multiplies this Quaternion by the input one. [More...](#)

void **multiplyByScalar** (float scalar)
Multiplies this Quaternion with a scalar. [More...](#)

void **normalize** ()
Normalizes this Quaternion. [More...](#)

Vector3f **rotateVector** (**Vector3f** v)

Rotates a 3-D vector by this Quaternion. [More...](#)

void **set** (**Quaternion** quat)
Copies the values from the given Quaternion to this one. [More...](#)

void **slerp** (**Quaternion** input, **Quaternion** output, float t)

Computes a linear interpolation between this Quaternion and the input Quaternion.
[More...](#)

void **toAxisAngle** (**Vector4f** output)
Gets an axis angle representation of this Quaternion. [More...](#)

double[] **toEulerAngles** ()
Returns the heading, attitude and bank of this Quaternion as euler angles in the double array respectively. [More...](#)

String **toString** ()
Returns a String representing this Quaternion. [More...](#)

▼ Public Member Functions inherited from **project.android.sensor.representation.Vector4f**

Vector4f (float x, float y, float z, float w)
Constructor. [More...](#)

Vector4f ()
Constructor. [More...](#)

Vector4f (**Vector3f** vector3f, float w)
Constructor. [More...](#)

float[] **array** ()
Returns the array representation of this Vector4f. [More...](#)

float **getW** ()
Returns the W-dimension of this Vector4f. [More...](#)

float **getX** ()
Returns the X-dimension of this Vector4f. [More...](#)

float **getY** ()
Returns the Y-dimension of this Vector4f. [More...](#)

float **getZ** ()
Returns the Z-dimension of this Vector4f. [More...](#)

void **multiplyByScalar** (float scalar)
Multiplies this Vector4f with a scalar. [More...](#)

void **normalize** ()
Normalizes this Vector4f. [More...](#)

void **setW** (float w)
Sets the W-dimension of this Vector4f. [More...](#)

void **setX** (float x)
Sets the X-dimension of this Vector4f. [More...](#)

void **setY** (float y)
Sets the Y-dimension of this Vector4f. [More...](#)

void **setZ** (float z)
Sets the Z-dimension of this Vector4f. [More...](#)

String **toString** ()
Returns the String representation of this Vector4f. [More...](#)

Additional Inherited Members

▼ Package Functions inherited from `project.android.sensor.representation.Vector4f`

void **copyFromV3f** (**Vector3f** input, float w)

Copies a **Vector3f** to this Vector4f using a separate value for W-dimension. [More...](#)

void **copyVec4** (**Vector4f** vec)

Copies another Vector4f to this one. [More...](#)

float **dotProduct** (**Vector4f** input)

Performs the dot product of two vectors. [More...](#)

Detailed Description

Class to represent a quaternion.

A quaternion H may be identified with R^4 , a four-dimensional vector space over the real numbers. The elements of this basis are customarily denoted as $1, i, j$, and k . Every element of H can be uniquely written as a linear combination of these basis elements, that is, as $a1 + bi + cj + dk$, where a, b, c , and d are real numbers.

Quaternions allow for elegant descriptions of 3D rotations, interpolations as well as extrapolations and compared to Euler angles, they don't suffer from gimbal lock. Interpolations between two Quaternions are called SLERP (Spherical Linear Interpolation).

If $a + bi + cj + dk$ is any quaternion, then a is called its scalar part and $bi + cj + dk$ is called its vector part. The scalar part of a quaternion is always real, and the vector part is always pure imaginary.

Using the basis $1, i, j, k$ of H it is possible to write H as a set of quadruples:

$$H = \{(a,b,c,d) \mid a,b,c,d \in R\}.$$

Then the basis elements are:

$$1 = (1,0,0,0),$$

$$i = (0,1,0,0),$$

$$j = (0,0,1,0),$$

$$k = (0,0,0,1).$$

Addition of two quaternions :

$$(a_1,b_1,c_1,d_1) + (a_2,b_2,c_2,d_2) = (a_1+a_2, b_1+b_2, c_1+c_2, d_1+d_2) .$$

Multiplications of two quaternions :

$$\begin{aligned} (a_1,b_1,c_1,d_1) (a_2,b_2,c_2,d_2) = \\ (a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2, \\ a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2, \\ a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2, \\ a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2) . \end{aligned}$$

Normalization of quaternion (a,b,c,d)

$$= (a/mag, b/mag, c/mag, d/mag),$$

where,

$$mag = \sqrt{a^2+b^2+c^2+d^2}.$$

The sensor data for 3-D mouse movement is sent from the mobile device is represented in the form of a Quaternion.

See also

`project.android.sensor.representation.Vector4f`

Definition at line **88** of file **Quaternion.java**.

Constructor Documentation

project.android.sensor.representation.Quaternion.Quaternion ()

Constructor.

Initializes this Quaternion.

Definition at line **102** of file **Quaternion.java**.

Member Function Documentation

void

project.android.sensor.representation.Quaternion.multiplyByQuat (Quaternion input, Quaternion output)

Multiplies this Quaternion with another.

Parameters

input Quaternion with which this is multiplied.

output result of (this * input).

Definition at line **139** of file **Quaternion.java**.

void

project.android.sensor.representation.Quaternion.multiplyByQuat (Quaternion input)

Multiplies this Quaternion by the input one.

Parameters

input Quaternion to be multiplied with.

Definition at line **164** of file **Quaternion.java**.

void project.android.sensor.representation.Quaternion.multiplyByScalar (float **scalar)**

Multiplies this Quaternion with a scalar.

This overrides the **Vector4f#multiplyByScalar(float)** method.

Parameters

scalar the scalar.

See also

project.android.sensor.representation.Vector4f::multiplyByScalar(float)

Definition at line **181** of file **Quaternion.java**.

void project.android.sensor.representation.Quaternion.normalize ()

Normalizes this Quaternion.

Definition at line **111** of file **Quaternion.java**.

Vector3f project.android.sensor.representation.Quaternion.rotateVector (Vector3f **v)**

Rotates a 3-D vector by this Quaternion.

Parameters

v the **Vector3f** to be rotated.

Returns

the rotated **Vector3f**.

Definition at line **495** of file **Quaternion.java**.

void project.android.sensor.representation.Quaternion.set (Quaternion **quat)**

Copies the values from the given Quaternion to this one.

Parameters

quat the Quaternion to copy from.

Definition at line **127** of file **Quaternion.java**.

```
void project.android.sensor.representation.Quaternion.slerp ( Quaternion input,
                                                             Quaternion output,
                                                             float t
                                                             )
```

Computes a linear interpolation between this Quaternion and the input Quaternion.

Parameters

input the Quaternion to be slerped with this.

output the Quaternion to store the result.

t the ratio between the two quaternions where $0 \leq t \leq 1.0$.

Note: Increase value of t will bring rotation closer to the input Quaternion.

Definition at line 452 of file [Quaternion.java](#).

```
void project.android.sensor.representation.Quaternion.toAxisAngle ( Vector4f output )
```

Gets an axis angle representation of this Quaternion.

Parameters

output Vector4f axis angle.

Definition at line 239 of file [Quaternion.java](#).

```
double [] project.android.sensor.representation.Quaternion.toEulerAngles ( )
```

Returns the heading, attitude and bank of this Quaternion as euler angles in the double array respectively.

Returns

an array of size 3 containing the euler angles for this Quaternion.

Definition at line 273 of file [Quaternion.java](#).

```
String project.android.sensor.representation.Quaternion.toString ( )
```

Returns a String representing this Quaternion.

This overrides the [Vector4f#toString\(\)](#) method.

Returns

String representing this Quaternion.

Definition at line 299 of file [Quaternion.java](#).

The documentation for this class was generated from the following file:

- [app/src/main/java/project/android/sensor/representation/Quaternion.java](#)

project.android.sensor.representation.Vector3f Class Reference

Class to represent a three dimensional vector. [More...](#)

Collaboration diagram for project.android.sensor.representation.Vector3f:

Public Member Functions

Vector3f (float x, float y, float z)
Constructor. [More...](#)

Vector3f ()
Constructor. [More...](#)

Vector3f (**Vector3f** vector)
Constructor. [More...](#)

float **dotProduct** (**Vector3f** inputVec)
Performs the dot product of two vectors. [More...](#)

float **getX** ()
Returns the X-dimension of this Vector3f. [More...](#)

float **getY** ()
Returns the Y-dimension of this Vector3f. [More...](#)

float **getZ** ()
Returns the Z-dimension of this Vector3f. [More...](#)

void **multiplyByScalar** (float scalar)
Multiplies this Vector3f with a scalar. [More...](#)

void **normalize** ()
Normalizes this Vector3f. [More...](#)

void **setXYZ** (float x, float y, float z)
Sets this Vector3f as (x, y, z). [More...](#)

String **toString** ()
Returns the String representation of this Vector3f. [More...](#)

Package Functions

void **setX** (float x)
Sets the X-dimension of this Vector3f. [More...](#)

void **setY** (float y)
Sets the Y-dimension of this Vector3f. [More...](#)

void **setZ** (float z)
Sets the Z-dimension of this Vector3f. [More...](#)

float[] **toArray** ()
Returns the array representation of this Vector3f. [More...](#)

Detailed Description

Class to represent a three dimensional vector.

It is defined as a point (x,y,z) in three dimensional vector space. It is stored as a array of floats. The three dimensions are known as X,Y and Z.

Dot product of two vectors :

$$(a_1, b_1, c_1) \cdot (a_2, b_2, c_2) = a_1a_2 + b_1b_2 + c_1c_2 .$$

Product of a vector with a scalar :

$$(a, b, c) * s = (a * s, b * s, c * s).$$

Normalization of a vector (a,b,c)

$$= (a/mag, b/mag, c/mag),$$

where,

$$mag = \sqrt{a^2+b^2+c^2}.$$

See also

project.android.sensor.representation.Vector3f
project.android.sensor.representation.Quaternion

Definition at line **37** of file **Vector3f.java**.

Constructor Documentation

```
project.android.sensor.representation.Vector3f.Vector3f ( float x,
                                                         float y,
                                                         float z
                                                         )
```

Constructor.

Creates the Vector3f (x,y,z).

Parameters

- x** the X-dimension.
- y** the Y-dimension.
- z** the Z-dimension.

Definition at line **50** of file **Vector3f.java**.

project.android.sensor.representation.Vector3f.Vector3f ()

Constructor.

Default constructor.

Definition at line **67** of file **Vector3f.java**.

project.android.sensor.representation.Vector3f.Vector3f (Vector3f vector)

Constructor.

Copy constructor. Copies another Vector3f to this one.

Parameters

vector the Vector3f to be copied.

Definition at line **78** of file **Vector3f.java**.

Member Function Documentation

float project.android.sensor.representation.Vector3f.dotProduct (Vector3f inputVec)

Performs the dot product of two vectors.

Parameters

inputVec the Vector3f to be multiplied.

Returns

result of (this . inputVec).

Definition at line **219** of file **Vector3f.java**.

float project.android.sensor.representation.Vector3f.getX ()

Returns the X-dimension of this Vector3f.

Returns

the X-dimension.

Definition at line **151** of file **Vector3f.java**.

float project.android.sensor.representation.Vector3f.getY ()

Returns the Y-dimension of this Vector3f.

Returns

the Y-dimension.

Definition at line **160** of file **Vector3f.java**.

float project.android.sensor.representation.Vector3f.getZ ()

Returns the Z-dimension of this Vector3f.

Returns

the Z-dimension.

Definition at line **169** of file **Vector3f.java**.

void project.android.sensor.representation.Vector3f.multiplyByScalar (float **scalar)**

Multiplies this Vector3f with a scalar.

Parameters

scalar the scalar.

Definition at line **128** of file **Vector3f.java**.

void project.android.sensor.representation.Vector3f.normalize ()

Normalizes this Vector3f.

Definition at line **137** of file **Vector3f.java**.

void project.android.sensor.representation.Vector3f.setX (float **x)**

package

Sets the X-dimension of this Vector3f.

Parameters

x the X-dimension.

Definition at line **178** of file **Vector3f.java**.


```
void project.android.sensor.representation.Vector3f.setXYZ ( float x,  
                                                             float y,  
                                                             float z  
                                                             )
```

Sets this Vector3f as (x,y,z).

Parameters

- x** the X-dimension.
- y** the Y-dimension.
- z** the Z-dimension.

Definition at line **207** of file **Vector3f.java**.

```
void project.android.sensor.representation.Vector3f.setY ( float y )
```

package

Sets the Y-dimension of this Vector3f.

Parameters

- y** the Y-dimension.

Definition at line **187** of file **Vector3f.java**.

```
void project.android.sensor.representation.Vector3f.setZ ( float z )
```

package

Sets the Z-dimension of this Vector3f.

Parameters

- z** the Z-dimension.

Definition at line **196** of file **Vector3f.java**.

```
float [] project.android.sensor.representation.Vector3f.toArray ( )
```

package

Returns the array representation of this Vector3f.

Returns

- the array [X, Y, Z].

Definition at line **101** of file **Vector3f.java**.

String project.android.sensor.representation.Vector3f.toString ()

Returns the String representation of this Vector3f.

It overrides the **java.lang.Object#toString()** method.

Returns

String representing this Vector3f.

Definition at line **244** of file **Vector3f.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/sensor/representation/**Vector3f.java**

project.android.sensor.representation.Vector4f Class Reference

Class to represent a four dimensional vector. [More...](#)

Collaboration diagram for project.android.sensor.representation.Vector4f:

Public Member Functions

Vector4f (float x, float y, float z, float w)

Constructor. [More...](#)

Vector4f ()

Constructor. [More...](#)

Vector4f (**Vector3f** vector3f, float w)

Constructor. [More...](#)

float[] **array** ()

Returns the array representation of this Vector4f. [More...](#)

float **getW** ()

Returns the W-dimension of this Vector4f. [More...](#)

float **getX** ()

Returns the X-dimension of this Vector4f. [More...](#)

float **getY** ()

Returns the Y-dimension of this Vector4f. [More...](#)

float **getZ** ()

Returns the Z-dimension of this Vector4f. [More...](#)

void **multiplyByScalar** (float scalar)

Multiplies this Vector4f with a scalar. [More...](#)

void **normalize** ()

Normalizes this Vector4f. [More...](#)

void **setW** (float w)

Sets the W-dimension of this Vector4f. [More...](#)

void **setX** (float x)

Sets the X-dimension of this Vector4f. [More...](#)

void **setY** (float y)

Sets the Y-dimension of this Vector4f. [More...](#)

void **setZ** (float z)

Sets the Z-dimension of this Vector4f. [More...](#)

String **toString** ()

Returns the String representation of this Vector4f. [More...](#)

Package Functions

- void **copyFromV3f** (**Vector3f** input, float w)
Copies a **Vector3f** to this Vector4f using a separate value for W-dimension. [More...](#)
- void **copyVec4** (**Vector4f** vec)
Copies another Vector4f to this one. [More...](#)
- float **dotProduct** (**Vector4f** input)
Performs the dot product of two vectors. [More...](#)
-

Detailed Description

Class to represent a four dimensional vector.

It is defined as a point (x,y,z,w) in four dimensional vector space. It is stored as a array of floats. The four dimensions are known as X,Y,Z and W.

Dot product of two vectors :

$$(a_1, b_1, c_1, d_1) \cdot (a_2, b_2, c_2, d_2) = a_1a_2 + b_1b_2 + c_1c_2 + d_1d_2.$$

Product of a vector with a scalar :

$$(a, b, c, d) * s = (a * s, b * s, c * s, d * s).$$

Normalization of a vector (a,b,c,d)

$$= (a/mag, b/mag, c/mag, d/mag),$$

where,

$$mag = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

See also

- project.android.sensor.representation.Vector3f
- project.android.sensor.representation.Quaternion

Definition at line **36** of file **Vector4f.java**.

Constructor Documentation

```
project.android.sensor.representation.Vector4f.Vector4f ( float x,  
                                                         float y,  
                                                         float z,  
                                                         float w  
                                                         )
```

Constructor.

Creates the Vector4f (x,y,z,w) .

Parameters

- x** the X-dimension.
- y** the Y-dimension.
- z** the Z-dimension.
- w** the W-dimension.

Definition at line **50** of file **Vector4f.java**.

project.android.sensor.representation.Vector4f.Vector4f ()

Constructor.

Default constructor. Creates the Vector4f (0,0,0,0).

Definition at line **62** of file **Vector4f.java**.

```
project.android.sensor.representation.Vector4f.Vector4f ( Vector3f vector3f,  
                                float w  
                                )
```

Constructor.

Creates the Vector4f (vector3f.getX(), vector3f.getY(), vector3f.getZ(), w).

Parameters

vector3f a **Vector3f**.

w the w-dimension.

Definition at line **78** of file **Vector4f.java**.

Member Function Documentation

float [] project.android.sensor.representation.Vector4f.array ()

Returns the array representation of this Vector4f.

Returns

the array [X, Y, Z, W].

Definition at line **90** of file **Vector4f.java**.

```
void project.android.sensor.representation.Vector4f.copyFromV3f ( Vector3f input,  
                                float w  
                                )
```

package

Copies a **Vector3f** to this Vector4f using a separate value for W-dimension.

Parameters

input the **Vector3f**.

w float representing the W-dimension.

Definition at line **286** of file **Vector4f.java**.

void `project.android.sensor.representation.Vector4f.copyVec4 ( Vector4f vec )`

package

Copies another Vector4f to this one.

Parameters

vec the Vector4f to be copied.

Definition at line **99** of file **Vector4f.java**.

float `project.android.sensor.representation.Vector4f.dotProduct ( Vector4f input )`

package

Performs the dot product of two vectors.

Parameters

input the Vector4f to be multiplied.

Returns

result of (this . input).

Definition at line **157** of file **Vector4f.java**.

float `project.android.sensor.representation.Vector4f.getW ( )`

Returns the W-dimension of this Vector4f.

Returns

the W-dimension.

Definition at line **220** of file **Vector4f.java**.

float `project.android.sensor.representation.Vector4f.getX ( )`

Returns the X-dimension of this Vector4f.

Returns

the X-dimension.

Definition at line **193** of file **Vector4f.java**.

float `project.android.sensor.representation.Vector4f.getY ( )`

Returns the Y-dimension of this Vector4f.

Returns

the Y-dimension.

Definition at line **202** of file **Vector4f.java**.

float project.android.sensor.representation.Vector4f.getZ ()

Returns the Z-dimension of this Vector4f.

Returns

the Z-dimension.

Definition at line **211** of file **Vector4f.java**.

void project.android.sensor.representation.Vector4f.multiplyByScalar (float scalar)

Multiplies this Vector4f with a scalar.

Parameters

scalar the scalar.

Definition at line **144** of file **Vector4f.java**.

void project.android.sensor.representation.Vector4f.normalize ()

Normalizes this Vector4f.

Definition at line **173** of file **Vector4f.java**.

void project.android.sensor.representation.Vector4f.setW (float w)

Sets the W-dimension of this Vector4f.

Parameters

w the X-dimension.

Definition at line **256** of file **Vector4f.java**.

void project.android.sensor.representation.Vector4f.setX (float x)

Sets the X-dimension of this Vector4f.

Parameters

x the X-dimension.

Definition at line **229** of file **Vector4f.java**.

void project.android.sensor.representation.Vector4f.setY (float y)

Sets the Y-dimension of this Vector4f.

Parameters

y the Y-dimension.

Definition at line **238** of file **Vector4f.java**.

void project.android.sensor.representation.Vector4f.setZ (float **z)**

Sets the Z-dimension of this Vector4f.

Parameters

z the Z-dimension.

Definition at line **247** of file **Vector4f.java**.

String project.android.sensor.representation.Vector4f.toString ()

Returns the String representation of this Vector4f.

It overrides the **java.lang.Object#toString()** method.

Returns

String representing this Vector4f.

Definition at line **303** of file **Vector4f.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/sensor/representation/**Vector4f.java**

project.android.AboutFragment Class Reference

Class representing the Fragment for displaying app information. [More...](#)

Collaboration diagram for project.android.AboutFragment:

Public Member Functions

- View

onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
Overrides the `Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)` method of the Android API. [More...](#)
- void

onPause ()
Overrides the `Fragment.onPause()` method of the Android API. [More...](#)
- void

onResume ()
Overrides the `Fragment.onResume()` method of the Android API. [More...](#)

Detailed Description

Class representing the Fragment for displaying app information.

Definition at line 16 of file `AboutFragment.java`.

Member Function Documentation

View `project.android.AboutFragment.onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)`

Overrides the `Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)` method of the Android API.
Called to have the fragment instantiate its user interface view.

Parameters

inflater

The `LayoutInflater` object that can be used to inflate any views in the fragment.

container

If non-null, this is the parent view that the fragment's UI should be attached to. The fragment should not add the view itself, but this can be used to generate the `LayoutParams` of the view.

savedInstanceState

If non-null, this fragment is being re-constructed from a previous saved state as given here.

Returns

the `View` for the fragment's UI, or `null`.

Definition at line 36 of file `AboutFragment.java`.

void project.android.AboutFragment.onPause ()

Overrides the `Fragment.onPause()` method of the Android API.

Called when the system is about to start resuming a previous activity.

Definition at line **66** of file **AboutFragment.java**.

void project.android.AboutFragment.onResume ()

Overrides the `Fragment.onResume()` method of the Android API.

Called when the activity will start interacting with the user.

Definition at line **53** of file **AboutFragment.java**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/AboutFragment.java`

project.android.ConnectionFragment Class Reference

Class representing the Fragment that provides GUI frontend for the network module. [More...](#)

Collaboration diagram for project.android.ConnectionFragment:

Public Member Functions

- void **addItem** (**ServerInfo** serverInfo)
Adds the `ServerInfo` object of an active server to the list of servers. [More...](#)
- void **dismissAlertDialog** ()
Method to dismiss the alert dialog prompting the pairing key input. [More...](#)
- View **getViewByPosition** (int pos)
Returns the View of the clicked list item. [More...](#)
- View **onCreateView** (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
Overrides the `Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)` method of the Android API. [More...](#)
- void **onPause** ()
Overrides the `Fragment.onPause()` method of the Android API. [More...](#)
- void **onRequestPermissionsResult** (int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults)
Override the `Fragment.onRequestPermissionsResult(int, String[], int[])` method of the Android API. [More...](#)
- void **onResume** ()
Overrides the `Fragment.onResume()` method of the Android API. [More...](#)
- void **onViewCreated** (View view, Bundle savedInstanceState)
Overrides the `Fragment.onViewCreated(View, Bundle)` method of the Android API. [More...](#)
- void **removeItem** (**ServerInfo** serverInfo)
Removes the `ServerInfo` object of an inactive server from the list of servers. [More...](#)
- void **resetIcon** (**ServerInfo** serverInfo)
Sets an icon for an inactive server. [More...](#)
- void **resetIcon** (int position)

Sets an icon for an inactive server. [More...](#)

void **setIcon** ()

Sets an icon for an active server. [More...](#)

Package Functions

ArrayList< **ServerInfo** > **getNetworkList** ()

Returns a list of ServerInfo objects. [More...](#)

Detailed Description

Class representing the Fragment that provides GUI frontend for the network module.

It performs the following operations:

- Displays the list of available servers with appropriate icons. It maintains a list of servers available.
- Displays an alert dialog upon selecting a server from the list to input the pairing key as shown in the PC.
- Connects with the respective server upon successful authentication and starts the communication by creating a TCP socket. It also changes the icon of the selected server in the list.

See also

project.android.net.ServerInfo

project.android.net.Client

BroadcastReceiverThread

project.android.net.ConnectionTask

Definition at line **67** of file **ConnectionFragment.java**.

Member Function Documentation

void project.android.ConnectionFragment.addItem (**ServerInfo **serverInfo**)**

Adds the ServerInfo object of an active server to the list of servers.

Parameters

serverInfo the active **project.android.net.ServerInfo** object.

See also

project.android.net.ServerInfo

Definition at line **337** of file **ConnectionFragment.java**.

void project.android.ConnectionFragment.dismissAlertDialog ()

Method to dismiss the alert dialog prompting the pairing key input.

Definition at line **399** of file **ConnectionFragment.java**.

ArrayList<ServerInfo> project.android.ConnectionFragment.getNetworkList ()

package

Returns a list of ServerInfo objects.

Returns

a list of **project.android.net.ServerInfo** objects.

See also

project.android.net.ServerInfo

Definition at line **235** of file **ConnectionFragment.java**.

View project.android.ConnectionFragment.getViewByPosition (int pos)

Returns the View of the clicked list item.

Parameters

pos position of the item.

Returns

the View of the clicked list item.

Definition at line **216** of file **ConnectionFragment.java**.

View project.android.ConnectionFragment.onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)

Overrides the `Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)` method of the Android API.

Called to have the fragment instantiate its user interface view.

Parameters

inflater	The LayoutInflater object that can be used to inflate any views in the fragment.
container	If non-null, this is the parent view that the fragment's UI should be attached to. The fragment should not add the view itself, but this can be used to generate the LayoutParams of the view.
savedInstanceState	If non-null, this fragment is being re-constructed from a previous saved state as given here.

Returns

the View for the fragment's UI, or null.

Definition at line **105** of file **ConnectionFragment.java**.

void project.android.ConnectionFragment.onPause ()

Overrides the `Fragment.onPause()` method of the Android API.

Called when the system is about to start resuming a previous activity.

Definition at line **182** of file **ConnectionFragment.java**.

```

void
project.android.ConnectionFragment.onRequestPermissionsResult ( int                requestCode,
                                                                @NonNull String[] permissions,
                                                                @NonNull int[]    grantResults
                                                                )

```

Override the `Fragment.onRequestPermissionsResult(int, String[], int[])` method of the Android API.

Parameters

requestCode the request code.

permissions the requested permissions. Never null.

grantResults the grant results for the corresponding permissions which is either `PERMISSION_GRANTED` or `PERMISSION_DENIED`. Never null.

Definition at line **199** of file **ConnectionFragment.java**.

```

void project.android.ConnectionFragment.onResume ( )

```

Overrides the `Fragment.onResume()` method of the Android API.

Called when the activity will start interacting with the user.

Definition at line **169** of file **ConnectionFragment.java**.

```

void project.android.ConnectionFragment.onViewCreated ( View    view,
                                                         Bundle savedInstanceState
                                                         )

```

Overrides the `Fragment.onViewCreated(View, Bundle)` method of the Android API.

Called immediately after `onCreateView(LayoutInflater, ViewGroup, Bundle)` has returned, but before any saved state has been restored in to the view.

Parameters

view The View returned by `onCreateView(LayoutInflater, ViewGroup, Bundle)`.

savedInstanceState If non-null, this fragment is being re-constructed from a previous saved state as given here.

Definition at line **132** of file **ConnectionFragment.java**.

```

void project.android.ConnectionFragment.removeItem ( ServerInfo serverInfo )

```

Removes the `ServerInfo` object of an inactive server from the list of servers.

Parameters

serverInfo the inactive `project.android.net.ServerInfo` object.

See also

`project.android.net.ServerInfo`

Definition at line **350** of file **ConnectionFragment.java**.

void `project.android.ConnectionFragment.resetIcon ( ServerInfo serverInfo )`

Sets an icon for an inactive server.

Parameters

`serverInfo` the `project.android.net.ServerInfo` of the inactive server.

See also

`project.android.net.ServerInfo`

Definition at line **364** of file `ConnectionFragment.java`.

void `project.android.ConnectionFragment.resetIcon ( int position )`

Sets an icon for an inactive server.

Parameters

`position` position of the inactive server in the list.

Definition at line **376** of file `ConnectionFragment.java`.

void `project.android.ConnectionFragment.setIcon ( )`

Sets an icon for an active server.

Definition at line **385** of file `ConnectionFragment.java`.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/ConnectionFragment.java`

project.android.CustomAdapter Class Reference

Displays the list of available servers in a ListView format. [More...](#)

Collaboration diagram for project.android.CustomAdapter:

Public Member Functions

int **getCount** ()
Overrides the `ArrayAdapter.getCount()` method. [More...](#)

View **getView** (int position, View convertView,@NonNull ViewGroup parent)
Overrides `ArrayAdapter.getView(int, View, ViewGroup)` method. [More...](#)

Package Functions

CustomAdapter (Activity activity, int textViewResourceId, ArrayList< **ServerInfo** > arrayList)
Constructor. [More...](#)

Detailed Description

Displays the list of available servers in a ListView format.

This CustomAdapter populates the ListView in the **ConnectionFragment**. List items consist of **project.android.net.ServerInfo** objects which are added/removed dynamically.

See also
project.android.ConnectionFragment

Definition at line **32** of file **CustomAdapter.java**.

Constructor Documentation

```
project.android.CustomAdapter.CustomAdapter ( Activity activity,
                                              int textViewResourceId,
                                              ArrayList< ServerInfo > arrayList
                                              )
```

package

Constructor.
Initializes this CustomAdapter.

Parameters
activity the current android.app.Activity object.
textViewResourceId the resource ID for a layout file containing a TextView to use when instantiating views.
arrayList the **project.android.net.ServerInfo** objects to be displayed in the ListView.

See also
project.android.net.ServerInfo

Definition at line **49** of file **CustomAdapter.java**.

Member Function Documentation

int project.android.CustomAdapter.getCount ()

Overrides the `ArrayAdapter.getCount()` method.

Returns the number of items in the data set represented by this adapter.

Returns

the count of items.

Definition at line **69** of file **CustomAdapter.java**.

View project.android.CustomAdapter.getView (int position, View convertView, @NonNull ViewGroup parent)

Overrides `ArrayAdapter.getView(int, View, ViewGroup)` method.

Gets a `View` that displays the data at the specified position in the data set.

Parameters

position the position of the item within the adapter's data set of the item whose view is required.

convertView The old view to reuse, if possible.

parent The parent that this view will eventually be attached to.

Returns

A `View` corresponding to the data at the specified position.

Definition at line **90** of file **CustomAdapter.java**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/CustomAdapter.java`

project.android.KeyboardFragment Class Reference

Class representing the Fragment for providing the GUI frontend for the keyboard module. [More...](#)

Collaboration diagram for project.android.KeyboardFragment:

Public Member Functions

void **beforeTextChanged** (CharSequence s, int start, int count, int after)
Overrides the TextWatcher.beforeTextChanged(CharSequence, int, int, int) method of the Android API. [More...](#)

void **onCheckedChanged** (CompoundButton buttonView, boolean isChecked)
Overrides the CompoundButton.onCheckChanged(CompoundButton, boolean) method of the Android API. [More...](#)

void **onClick** (View v)
Overrides the View.OnClickListener.onClick(View) method of the Android API. [More...](#)

View **onCreateView** (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
Overrides the Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle) of the Android API. [More...](#)

void **onDestroyView** ()
Overrides the Fragment.onDestroyView() method of the Android API. [More...](#)

boolean **onKey** (View v, int keyCode, KeyEvent event)
Overrides the View.OnKeyListener.onKey(View, int, KeyEvent) method of the Android API. [More...](#)

void **onPause** ()
Overrides the Fragment.onPause() method of the Android API. [More...](#)

void **onResume** ()
Overrides the Fragment.onResume() method of the Android API. [More...](#)

void **onTextChanged** (CharSequence s, int start, int before, int count)
Overrides the onTextChanged(CharSequence, int, int, int) method of the Android API. [More...](#)

Detailed Description

Class representing the Fragment for providing the GUI frontend for the keyboard module.

The following keyboard features are provided:

- Predictive text input.
- Auto-correct feature.

- Hide sensitive text.
- Support for special keys like Ctrl, Alt etc.

See also

[project.android.net.KeyboardThread](#)

Definition at line [42](#) of file [KeyboardFragment.java](#).

Member Function Documentation

```
void project.android.KeyboardFragment.beforeTextChanged ( CharSequence s,  
int start,  
int count,  
int after  
)
```

Overrides the `TextWatcher.beforeTextChanged(CharSequence, int, int, int)` method of the Android API.

Called to notify that, within `s`, the `count` characters beginning at `start` are about to be replaced by new text with length `after`. It is an error to attempt to make changes to `s` from this callback.

Parameters

- s** the `CharSequence`.
- start** start index of the current word.
- count** number of characters to be replaced.
- after** new length.

Definition at line [187](#) of file [KeyboardFragment.java](#).

```
void project.android.KeyboardFragment.onCheckedChanged ( CompoundButton buttonView,  
boolean isChecked  
)
```

Overrides the `CompoundButton.onCheckedChangeListener(CompoundButton, boolean)` method of the Android API.

Called when the checked state of a compound button has changed.

Parameters

- buttonView** the `CompoundButton` view whose state has changed.
- isChecked** the new checked state of `buttonView`.

Definition at line [285](#) of file [KeyboardFragment.java](#).

```
void project.android.KeyboardFragment.onClick ( View v )
```

Overrides the `View.OnClickListener.onClick(View)` method of the Android API.

Called when a view has been clicked.

Parameters

- v** the view that was clicked.

Definition at line [322](#) of file [KeyboardFragment.java](#).

```
View project.android.KeyboardFragment.onCreateView ( LayoutInflater inflater,
                                                    ViewGroup container,
                                                    Bundle savedInstanceState
                                                    )
```

Overrides the `Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)` of the Android API.

Called to have the fragment instantiate its user interface view.

Parameters

- inflater** The `LayoutInflater` object that can be used to inflate any views in the fragment.
- container** If non-null, this is the parent view that the fragment's UI is attached to. The fragment should not add the view itself, but this can be used to generate the `LayoutParams` of the view.
- savedInstanceState** If non-null, this fragment is being re-constructed from a previous saved state as given here.

Returns

the `View` for the fragment's UI, or null.

Definition at line **101** of file **KeyboardFragment.java**.

```
void project.android.KeyboardFragment.onDestroyView ( )
```

Overrides the `Fragment.onDestroyView()` method of the Android API.

Called when the view previously created by **`onCreateView(LayoutInflater, ViewGroup, Bundle)`** has been detached from the fragment.

Definition at line **134** of file **KeyboardFragment.java**.

```
boolean project.android.KeyboardFragment.onKey ( View v,
                                                  int keyCode,
                                                  KeyEvent event
                                                  )
```

Overrides the `View.onKeyListener.onKey(View, int, KeyEvent)` method of the Android API.

Called when a hardware key is dispatched to a view. This allows listeners to get a chance to respond before the target view. This has been used for detecting back key down event.

Parameters

- v** the view to which the key has been dispatched.
- keyCode** the code for the physical key that was pressed.
- event** the `KeyEvent` object containing full information about the event.

Returns

true, if the listener has consumed the event,
false, otherwise.

Definition at line **265** of file **KeyboardFragment.java**.

void project.android.KeyboardFragment.onPause ()

Overrides the `Fragment.onPause()` method of the Android API.

Called when the system is about to start resuming a previous activity.

Definition at line **164** of file **KeyboardFragment.java**.

void project.android.KeyboardFragment.onResume ()

Overrides the `Fragment.onResume()` method of the Android API.

Called when the activity will start interacting with the user.

Definition at line **151** of file **KeyboardFragment.java**.

```
void project.android.KeyboardFragment.onTextChanged ( CharSequence s,  
int start,  
int before,  
int count  
)
```

Overrides the `onTextChanged(CharSequence, int, int, int)` method of the Android API.

Called to notify that, within `s`, the `count` characters beginning at `start` have just replaced old text that had length `before`. It is an error to attempt to make changes to `s` from this callback.

Parameters

s the `CharSequence`.

start start index of the current word.

before number of characters that are replaced.

count new length.

Definition at line **219** of file **KeyboardFragment.java**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/KeyboardFragment.java`

project.android.MainActivity Class Reference

Class representing the Activity that is launched at app startup. [More...](#)

Collaboration diagram for project.android.MainActivity:

Public Member Functions

void **onBackPressed** ()
Overrides the `FragmentActivity.onBackPressed()` method of the Android API. [More...](#)

boolean **onCreateOptionsMenu** (Menu menu)
Overrides the `Activity.onCreateOptionsMenu(Menu)` method of the Android API. [More...](#)

void **onDestroy** ()
Overrides the `AppCompatActivity.onDestroy()` method of the Android API. [More...](#)

boolean **onNavigationItemSelectedListener** (@NonNull MenuItem item)
Implements the
`NavigationView.OnNavigationItemSelectedListener.onNavigationItemSelectedListener(MenuItem)`
method of the Android API. [More...](#)

boolean **onOptionsItemSelected** (MenuItem item)
Overrides the `Activity.onOptionsItemSelected(MenuItem)` method of the Android API. [More...](#)

void **onPause** ()
Overrides the `FragmentActivity.onPause()` method of the Android API. [More...](#)

void **onResume** ()
Overrides the `FragmentActivity.onResume()` method of the Android API. [More...](#)

Static Public Member Functions

static Fragment **getConnectionFragment** ()
Returns the **project.android.ConnectionFragment** object. [More...](#)

Protected Member Functions

void **onCreate** (Bundle savedInstanceState)
Overrides the `AppCompatActivity.onCreate(Bundle)` method of the Android API. [More...](#)

Detailed Description

Class representing the Activity that is launched at app startup.

It provides an interface for navigation bar which contains the following navigation items:

- 3D Mouse

- 2D Mouse
- Keyboard
- Connect
- About

Definition at line **46** of file **MainActivity.java**.

Member Function Documentation

static **Fragment** **project.android.MainActivity.getConnectionFragment ()**

static

Returns the **project.android.ConnectionFragment** object.

Returns

the **project.android.ConnectionFragment** object.

See also

project.android.ConnectionFragment

Definition at line **269** of file **MainActivity.java**.

void **project.android.MainActivity.onBackPressed ()**

Overrides the **FragmentManager.onBackPressed()** method of the Android API.

Called when the activity has detected the user's press of the back key.

Definition at line **141** of file **MainActivity.java**.

void **project.android.MainActivity.onCreate (Bundle savedInstanceState)**

protected

Overrides the **AppCompatActivity.onCreate(Bundle)** method of the Android API.

Called when the activity is first created.

Parameters

savedInstanceState If the activity is being re-initialized after previously being shut down then this **Bundle** contains the data it most recently supplied.

Note: Otherwise it is null.

Definition at line **69** of file **MainActivity.java**.

boolean **project.android.MainActivity.onCreateOptionsMenu (Menu menu)**

Overrides the **Activity.onCreateOptionsMenu(Menu)** method of the Android API.

Initializes the contents of the Activity's standard options menu.

Parameters

menu The options menu for placing items.

Returns

true, if the menu is to be displayed,
false, otherwise.

Definition at line **172** of file **MainActivity.java**.

void project.android.MainActivity.onDestroy ()

Overrides the AppCompatActivity.onDestroy() method of the Android API.

The final call received before an activity is destroyed.

Definition at line 155 of file MainActivity.java.

boolean project.android.MainActivity.onNavigationItemSelectedListener (@NonNull MenuItem item)

Implements the NavigationView.OnNavigationItemSelectedListener.onNavigationItemSelectedListener(MenuItem) method of the Android API.

Called when an item in the navigation menu is selected.

Parameters

item The selected item.

Returns

true, if the selected item is to be displayed,
false, otherwise.

Definition at line 222 of file MainActivity.java.

boolean project.android.MainActivity.onOptionsItemSelected (MenuItem item)

Overrides the Activity.onOptionsItemSelected(MenuItem) method of the Android API.

Handles action bar item clicks. The action bar will automatically handle clicks on the Home/Up button, so long as a parent activity is specified in AndroidManifest.xml.

Parameters

item The menu item that was selected.

Returns

false, if normal menu processing is allowed to proceed,
true, if it is to be consumed here.

Definition at line 194 of file MainActivity.java.

void project.android.MainActivity.onPause ()

Overrides the FragmentActivity.onPause() method of the Android API.

Called when the system is about to start resuming a previous activity.

Definition at line 128 of file MainActivity.java.

void project.android.MainActivity.onResume ()

Overrides the FragmentActivity.onResume() method of the Android API.

Called when the activity will start interacting with the user.

Definition at line 115 of file MainActivity.java.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/MainActivity.java

project.android.MouseFragment Class Reference

Class representing the Fragment for providing the GUI frontend for the 3D mouse module. [More...](#)

Collaboration diagram for project.android.MouseFragment:

Public Member Functions

void **onClick** (View view)

Overrides the `View.OnClickListener.onClick(View)` method of the Android API. [More...](#)

View **onCreateView** (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
Overrides the `Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)` method of the Android API. [More...](#)

void **onDestroyView** ()

Overrides the `Fragment.onDestroyView()` method of the Android API. [More...](#)

void **onPause** ()

Overrides the `Fragment.onPause()` method of the Android API. [More...](#)

void **onResume** ()

Overrides the `Fragment.onResume()` method of the Android API. [More...](#)

boolean **onTouch** (View view, MotionEvent motionEvent)

Overrides the `View.OnTouchListener.onTouch(View, MotionEvent)` method of the Android API. [More...](#)

Detailed Description

Class representing the Fragment for providing the GUI frontend for the 3D mouse module.

The following features have been provided:

- 3D mouse movement.
- Single left click and double left click.
- Right click.
- Middle click.
- Scrolling

The 3D mouse movement feature uses the gyroscope and accelerometer sensors. The [project.android.sensor.orientation.OrientationProvider](#) and

project.android.sensor.orientation.KalmanFilterProvider classes are responsible for processing the sensor data. The clicks and scrolling feature uses buttons.

Note: This 3D mouse module does not work with Android devices where the mentioned sensors are not present.

See also

project.android.sensor.orientation.OrientationProvider

project.android.sensor.orientation.KalmanFilterProvider

Definition at line **56** of file **MouseFragment.java**.

Member Function Documentation

void project.android.MouseFragment.onClick (View view)

Overrides the View.OnClickListener.onClick(View) method of the Android API.

Called when a view has been clicked.

Parameters

view the view that was clicked.

Definition at line **201** of file **MouseFragment.java**.

View

project.android.MouseFragment.onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)

Overrides the Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle) method of the Android API.

Called to have the fragment instantiate its user interface view.

Parameters

inflater	The LayoutInflater object that can be used to inflate any views in the fragment.
container	If non-null, this is the parent view that the fragment's UI should be attached to. The fragment should not add the view itself, but this can be used to generate the LayoutParams of the view.
savedInstanceState	If non-null, this fragment is being re-constructed from a previous saved state as given here.

Returns

the View for the fragment's UI, or null.

Definition at line **81** of file **MouseFragment.java**.

void project.android.MouseFragment.onDestroyView ()

Overrides the `Fragment.onDestroyView()` method of the Android API.

Called when the view previously created by `onCreateView(LayoutInflater, ViewGroup, Bundle)` has been detached from the fragment.

Definition at line **147** of file **MouseFragment.java**.

void project.android.MouseFragment.onPause ()

Overrides the `Fragment.onPause()` method of the Android API.

Called when the system is about to start resuming a previous activity.

Definition at line **133** of file **MouseFragment.java**.

void project.android.MouseFragment.onResume ()

Overrides the `Fragment.onResume()` method of the Android API.

Called when the activity will start interacting with the user.

Definition at line **120** of file **MouseFragment.java**.

boolean project.android.MouseFragment.onTouch (View **view, MotionEvent **motionEvent**)**

Overrides the `View.TouchListener.onTouch(View, MotionEvent)` method of the Android API.

Called when a touch event is dispatched to a View. This allows listeners to get a chance to respond before the target View.

Parameters

view the view the touch event has been dispatched to.

motionEvent the MotionEvent object containing full information about the event.

Returns

true, if the listener has consumed the event,
false, otherwise.

Definition at line **169** of file **MouseFragment.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/**MouseFragment.java**

project.android.SettingsActivity Class Reference

Class representing the Activity for app settings. [More...](#)

Collaboration diagram for project.android.SettingsActivity:

Public Member Functions

- void **onPause** ()
Overrides the `FragmentActivity.onPause()` method of the Android API. [More...](#)
- void **onResume** ()
Overrides the `FragmentActivity.onResume()` method of the Android API. [More...](#)
- void **onSharedPreferenceChanged** (SharedPreferences sharedPreferences, String key)
Overrides the `OnSharedPreferenceChangeListener.onSharedPreferenceChanged` (SharedPreferences, String) method of the Android API. [More...](#)

Protected Member Functions

- void **onCreate** (Bundle savedInstanceState)
Overrides the `AppCompatActivity.onCreate(Bundle)` of the Android API. [More...](#)

Detailed Description

Class representing the Activity for app settings.

It displays the app settings and enables customization for the users. There are two customizations that are present :

1. 3D Mouse Sensitivity.
2. 2D Mouse Sensitivity.

Each of these can have three possible values - : "*Low*", "*Medium*" and "*High*". The default value is "*Medium*".

This class is made separate from the **MainActivity** to enable the "*Settings*" to be opened from any fragment of the MainActivity. This avoids overlapping of other fragments - that are already rendered - by the **SettingsFragment**.

It also implements the `SharedPreferences.OnSharedPreferenceListener` of the Android API. The **onSharedPreferenceChanged(SharedPreferences, String)** method listens to any changes made in the app settings.

See also

- project.android.MainActivity
- project.android.SettingsFragment
- onSharedPreferenceChanged(SharedPreferences, String)

Definition at line **46** of file **SettingsActivity.java**.

Member Function Documentation

void project.android.SettingsActivity.onCreate (Bundle savedInstanceState)

protected

Overrides the AppCompatActivity.onCreate(Bundle) of the Android API.

Called when the activity is first created.

Parameters

savedInstanceState If the activity is being re-initialized after previously being shut down then this Bundle contains the data it most recently supplied.

Note: Otherwise it is null.

Definition at line **64** of file **SettingsActivity.java**.

void project.android.SettingsActivity.onPause ()

Overrides the FragmentActivity.onPause() method of the Android API.

Called when the system is about to start resuming a previous activity.

Definition at line **123** of file **SettingsActivity.java**.

void project.android.SettingsActivity.onResume ()

Overrides the FragmentActivity.onResume() method of the Android API.

Called when the activity will start interacting with the user.

Definition at line **109** of file **SettingsActivity.java**.

void
project.android.SettingsActivity.onSharedPreferenceChanged (SharedPreferences savedInstanceState, String key)

Overrides the OnSharedPreferenceChangeListener.onSharedPreferenceChanged (SharedPreferences, String) method of the Android API.

Parameters

sharedPreferences the SharedPreferences that received the change.

key the key of the preference that was changed, added, or removed.

Definition at line **89** of file **SettingsActivity.java**.

The documentation for this class was generated from the following file:

- app/src/main/java/project/android/**SettingsActivity.java**

project.android.SettingsFragment Class Reference

Class representing the Fragment for changing app settings. [More...](#)

Collaboration diagram for project.android.SettingsFragment:

Public Member Functions

void **onCreate** (Bundle savedInstanceState)
Overrides the `PreferenceFragment.onCreate(Bundle)` method of the Android API. [More...](#)

void **onPause** ()
Overrides the `PreferenceFragment.onPause()` method of the Android API. [More...](#)

void **onResume** ()
Overrides the `PreferenceFragment.onResume()` method of the Android API. [More...](#)

Detailed Description

Class representing the Fragment for changing app settings.

This `SettingsFragment` is created by the **`SettingsActivity#onCreate(Bundle)`** method.

See also

`project.android.SettingsActivity`

`project.android.SettingsActivity::onCreate(Bundle)`

Definition at line **17** of file **`SettingsFragment.java`**.

Member Function Documentation

void

project.android.SettingsFragment.onCreate (**Bundle savedInstanceState**)

Overrides the `PreferenceFragment.onCreate(Bundle)` method of the Android API.

Called when this fragment is created.

Parameters

savedInstanceState If the fragment is being re-created from a previous saved state, this is the state.

Definition at line **30** of file **SettingsFragment.java**.

void project.android.SettingsFragment.onPause ()

Overrides the `PreferenceFragment.onPause()` method of the Android API.

Called when the system is about to start resuming a previous activity.

Definition at line **54** of file **SettingsFragment.java**.

void project.android.SettingsFragment.onResume ()

Overrides the `PreferenceFragment.onResume()` method of the Android API.

Called when this fragment will start interacting with the user.

Definition at line **42** of file **SettingsFragment.java**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/SettingsFragment.java`

project.android.TouchpadFragment Class Reference

Class representing the Fragment for providing the GUI frontend for the 2D mouse module. [More...](#)

Collaboration diagram for project.android.TouchpadFragment:

Public Member Functions

void **onClick** (View view)
Overrides the `View.OnClickListener.onClick(View)` method of the Android API. [More...](#)

View **onCreateView** (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
Overrides the `Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)` method of the Android API. [More...](#)

void **onPause** ()
Overrides the `Fragment.onPause()` method of the Android API. [More...](#)

void **onResume** ()
Overrides the `Fragment.onResume()` method of the Android API. [More...](#)

boolean **onTouch** (View view, MotionEvent motionEvent)
Overrides the `View.OnTouchListener.onTouch(View, MotionEvent)` method of the Android API. [More...](#)

Detailed Description

Class representing the Fragment for providing the GUI frontend for the 2D mouse module.

The following features have been provided:

- 2D mouse movement.
- Single left click and double left click.
- Right click
- Middle click.
- Scrolling

The 2D mouse movement feature is provided by implementing the `View.OnTouchListener`

interface.

Definition at line **38** of file **TouchpadFragment.java**.

Member Function Documentation

void project.android.TouchpadFragment.onClick (View view)

Overrides the View.OnClickListener.onClick(View) method of the Android API.

Called when a view has been clicked.

Parameters

view the view that was clicked.

Definition at line **153** of file **TouchpadFragment.java**.

View

**project.android.TouchpadFragment.onCreateView (LayoutInflater inflater,
ViewGroup container,
Bundle savedInstanceState
)**

Overrides the Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle) method of the Android API.

Called to have the fragment instantiate its user interface view.

Parameters

inflater	The LayoutInflater object that can be used to inflate any views in the fragment.
container	If non-null, this is the parent view that the fragment's UI should be attached to. The fragment should not add the view itself, but this can be used to generate the LayoutParams of the view.
savedInstanceState	If non-null, this fragment is being re-constructed from a previous saved state as given here.

Returns

the View for the fragment's UI, or null.

Definition at line **63** of file **TouchpadFragment.java**.

void project.android.TouchpadFragment.onPause ()

Overrides the `Fragment.onPause()` method of the Android API.

Called when the system is about to start resuming a previous activity.

Definition at line **103** of file **TouchpadFragment.java**.

void project.android.TouchpadFragment.onResume ()

Overrides the `Fragment.onResume()` method of the Android API.

Called when the activity will start interacting with the user.

Definition at line **91** of file **TouchpadFragment.java**.

boolean project.android.TouchpadFragment.onTouch (View **view, MotionEvent **motionEvent**)**

Overrides the `View.OnTouchListener.onTouch(View, MotionEvent)` method of the Android API.

Called when a touch event is dispatched to a `View`. This allows listeners to get a chance to respond before the target `View`.

Parameters

view the view the touch event has been dispatched to.

motionEvent the `MotionEvent` object containing full information about the event.

Returns

true, if the listener has consumed the event,
false, otherwise.

Definition at line **124** of file **TouchpadFragment.java**.

The documentation for this class was generated from the following file:

- `app/src/main/java/project/android/TouchpadFragment.java`

4.2. PC

4.2.1. Complete Coding

Directory Structure

[detail level 1 2 3 4 5 6]

```
▼ src
  ▼ project
    ▼ pc
      ▼ device
        KeyboardController.java
        KeyCode.java
        MouseController.java
        MouseMove.java
      ▼ gui
        Cube.java
        MainWindow.java
      ▼ net
        BroadcastThread.java
        ClientInfo.java
        DataWrapper.java
        NetworkManager.java
        NetworkState.java
        NetworkThread.java
        Server.java
        ServerInfo.java
        ServerThread.java
      ▼ security
        EKEProvider.java
        KeyStoreManager.java
      ▼ sensor
        ▼ representation
          Cartesian2D.java
          Matrix.java
          MatrixF4x4.java
          Quaternion.java
          Vector3f.java
          Vector4f.java
          SensorDataHandler.java
          Main.java
```

KeyboardController.java

```
1 package project.pc.device;
2
3 import java.awt.AWTException;
4 import java.awt.Robot;
5 import java.awt.event.KeyEvent;
6
7 /**
8  * Utility for performing keyboard input operations.
9  *
10  * <p>
11  * It uses a {@link java.awt.Robot} object to generate native
12  * system input events. This differs from posting events to
13  * the AWT event queue or AWT components in that the events
14  * are generated in the platform's native input queue.
15  * </p>
16  *
17  * @see java.awt.Robot
18  */
19 public class KeyboardController {
20     private Robot mRobot;
21
22     /**
23      * Constructor.
24      *
25      * Initializes this <code>KeyboardController</code> by
26      * instantiating a {@link java.awt.Robot} object.
27      *
28      * @throws AWTException
29      */
30     public KeyboardController() throws AWTException {
31         mRobot = new Robot();
32     }
33
34     /**
35      * Method to perform the keyboard input operations.
36      *
37      * @param s <code>String</code> representing the sequence of
38      * characters to be typed.
39      */
40     public void doKeyOperation(String s) {
41
42         if(s.length() == 0) {
43             performAction('\n');
44         } else {
45             char[] chars = s.toCharArray();
46             for (char c : chars) performAction(c);
47         }
48     }
49
50     /**
51      * Method to perform keyboard input operations
52      * for different function keys.
53      *
54      * @param s <code>String</code> representing the
55      * key to be pressed.
56      */
57     public void doSpecialKeyOperation(String s) {
58         switch (s) {
59             case "Ctrl_On":
60                 mRobot.keyPress(KeyEvent.VK_CONTROL);
61                 break;
62
63             case "Ctrl_Off":
64                 mRobot.keyRelease(KeyEvent.VK_CONTROL);
65                 break;
66
67             case "Alt_On":
68                 mRobot.keyPress(KeyEvent.VK_ALT);
69                 break;
70
71             case "Alt_Off":
72                 mRobot.keyRelease(KeyEvent.VK_ALT);
73                 break;
74
75             case "Shift_On":
76                 mRobot.keyPress(KeyEvent.VK_SHIFT);
77                 break;
78
79             case "Shift_Off":
80                 mRobot.keyRelease(KeyEvent.VK_SHIFT);
81                 break;
82
83             case "Home":
84                 performSpecialAction(KeyEvent.VK_HOME);
85                 break;
86
87             case "End":
88                 performSpecialAction(KeyEvent.VK_END);
89                 break;
```

```

90
91     case "Page_Up":
92         performSpecialAction(KeyEvent.VK_PAGE_UP);
93         break;
94
95     case "Page_Down":
96         performSpecialAction(KeyEvent.VK_PAGE_DOWN);
97         break;
98
99     case "Insert":
100         performSpecialAction(KeyEvent.VK_INSERT);
101         break;
102
103     case "Del":
104         performSpecialAction(KeyEvent.VK_DELETE);
105         break;
106
107     case "Esc":
108         performSpecialAction(KeyEvent.VK_ESCAPE);
109         break;
110
111     case "Tab":
112         performSpecialAction(KeyEvent.VK_TAB);
113         break;
114
115     case "F1":
116         performSpecialAction(KeyEvent.VK_F1);
117         break;
118
119     case "F2":
120         performSpecialAction(KeyEvent.VK_F2);
121         break;
122
123     case "F3":
124         performSpecialAction(KeyEvent.VK_F3);
125         break;
126
127     case "F4":
128         performSpecialAction(KeyEvent.VK_F4);
129         break;
130
131     case "F5":
132         performSpecialAction(KeyEvent.VK_F5);
133         break;
134
135     case "F6":
136         performSpecialAction(KeyEvent.VK_F6);
137         break;
138
139     case "F7":
140         performSpecialAction(KeyEvent.VK_F7);
141         break;
142
143     case "F8":
144         performSpecialAction(KeyEvent.VK_F8);
145         break;
146
147     case "F9":
148         performSpecialAction(KeyEvent.VK_F9);
149         break;
150
151     case "F10":
152         performSpecialAction(KeyEvent.VK_F10);
153         break;
154
155     case "F11":
156         performSpecialAction(KeyEvent.VK_F11);
157         break;
158
159     case "F12":
160         performSpecialAction(KeyEvent.VK_F12);
161         break;
162
163     case "Up":
164         performSpecialAction(KeyEvent.VK_UP);
165         break;
166
167     case "Down":
168         performSpecialAction(KeyEvent.VK_DOWN);
169         break;
170
171     case "Left":
172         performSpecialAction(KeyEvent.VK_LEFT);
173         break;
174
175     case "Right":
176         performSpecialAction(KeyEvent.VK_RIGHT);
177     }
178 }
179 private void performAction(char key) {
180
181     if(KeyCode.isShiftKey(key)) {
182         int keyCode = KeyCode.getShiftCode(key);
183         if(keyCode == -1) System.out.println("Invalid char! " + key);
184         else doShift(keyCode);

```

```

185         return;
186     }
187     int keyCode = KeyCode.getByChar(key);
188     if (keyCode == -1) System.out.println("Invalid char! " + key);
189     else {
190         if (Character.isUpperCase(key)) doShift(keyCode);
191         else {
192             mRobot.keyPress(keyCode);
193             mRobot.keyRelease(keyCode);
194         }
195     }
196 }
197 private void doShift(int keyCode) {
198     mRobot.keyPress(KeyEvent.VK_SHIFT);
199     mRobot.keyPress(keyCode);
200     mRobot.keyRelease(keyCode);
201     mRobot.keyRelease(KeyEvent.VK_SHIFT);
202 }
203
204 private void performSpecialAction(int keyCode) {
205     mRobot.keyPress(keyCode);
206     mRobot.keyRelease(keyCode);
207 }
208 }

```

KeyCode.java

```
1 package project.pc.device;
2
3 import java.awt.event.KeyEvent;
4 import java.util.HashMap;
5 import java.util.Map;
6
7 /**
8  * Maps the characters to type to the corresponding key codes.
9  *
10  * <p>
11  * These key codes represent the key(s) to be pressed. The
12  * mapping is done with the help of two <code>HashMap</code>s :
13  * <ol>
14  * <li>
15  * <i>Key Map</i> - Maps characters to key codes
16  * where pressing shift key is not required.
17  * </li>
18  * <li>
19  * <i>Shift Key Map</i> - Maps characters to key
20  * codes where pressing shift key is required. For
21  * example, the character <code>%</code> is mapped
22  * to <code>5</code> since <code>Shift+5</code> is
23  * pressed for typing <code>%</code>.
24  * </li>
25  * </ol>
26  * </p>
27  */
28 class KeyCode {
29
30     private static Map<Character, Integer> sKeyMap = new HashMap<>();
31     private static Map<Character, Integer> sShiftKeyMap = new HashMap<>();
32
33     static {
34         for (int i = 'a', j = KeyEvent.VK_A; i <= 'z'; ++i, ++j) {
35             sKeyMap.put((char)i, j);
36         }
37
38         for (int i = 'A', j = KeyEvent.VK_A; i <= 'Z'; i++, j++) {
39             sKeyMap.put((char)i, j);
40         }
41
42         for (int i = '0', j = KeyEvent.VK_0; i <= '9'; i++, j++) {
43             sKeyMap.put((char)i, j);
44         }
45
46         // special char
47         sKeyMap.put('"', KeyEvent.VK_BACK_QUOTE);
48         sKeyMap.put('-', KeyEvent.VK_MINUS);
49         sKeyMap.put('=', KeyEvent.VK_EQUALS);
50         sKeyMap.put('_', KeyEvent.VK_UNDERSCORE);
51         sKeyMap.put('+', KeyEvent.VK_PLUS);
52         sKeyMap.put('[', KeyEvent.VK_OPEN_BRACKET);
53         sKeyMap.put(']', KeyEvent.VK_OPEN_BRACKET);
54         sKeyMap.put('\\', KeyEvent.VK_BACK_SLASH);
55         sKeyMap.put(':', KeyEvent.VK_SEMICOLON);
56         sKeyMap.put('\'', KeyEvent.VK_QUOTE);
57         sKeyMap.put('.', KeyEvent.VK_COMMA);
58         sKeyMap.put('.', KeyEvent.VK_PERIOD);
59         sKeyMap.put('/', KeyEvent.VK_SLASH);
60         sKeyMap.put(' ', KeyEvent.VK_SPACE);
61         sKeyMap.put('\n', KeyEvent.VK_ENTER);
62         sKeyMap.put('\b', KeyEvent.VK_BACK_SPACE);
63
64
65         // set of character which need to be typed with shift key
66         sShiftKeyMap.put('%', KeyEvent.VK_5);
67         sShiftKeyMap.put('!', KeyEvent.VK_1);
68         sShiftKeyMap.put('@', KeyEvent.VK_2);
69         sShiftKeyMap.put('#', KeyEvent.VK_3);
70         sShiftKeyMap.put('$', KeyEvent.VK_4);
71         sShiftKeyMap.put('^', KeyEvent.VK_6);
72         sShiftKeyMap.put('&', KeyEvent.VK_7);
73         sShiftKeyMap.put('*', KeyEvent.VK_8);
74         sShiftKeyMap.put('(', KeyEvent.VK_9);
75         sShiftKeyMap.put(')', KeyEvent.VK_0);
76         sShiftKeyMap.put(':', KeyEvent.VK_SEMICOLON);
77         sShiftKeyMap.put('"', KeyEvent.VK_QUOTE);
78         sShiftKeyMap.put('<', KeyEvent.VK_COMMA);
79         sShiftKeyMap.put('>', KeyEvent.VK_PERIOD);
80         sShiftKeyMap.put('?', KeyEvent.VK_SLASH);
81         sShiftKeyMap.put('_', KeyEvent.VK_MINUS);
82         sShiftKeyMap.put('+', KeyEvent.VK_EQUALS);
83         sShiftKeyMap.put('{', KeyEvent.VK_OPEN_BRACKET);
84         sShiftKeyMap.put('}', KeyEvent.VK_CLOSE_BRACKET);
85         sShiftKeyMap.put('|', KeyEvent.VK_BACK_SLASH);
86         sShiftKeyMap.put('~', KeyEvent.VK_BACK_QUOTE);
87     }
88
89     /**
```

```

90      * Checks if <code>Shift</code> key needs to be pressed for
91      * typing a character.
92      *
93      * @param c the character to be checked.
94      * @return <code>true</code>, if <code>Shift</code> key
95      *         needs to be pressed, <br/>
96      *         <code>false</code>, otherwise.
97      */
98      static boolean isShiftKey(char c) {
99          return sShiftKeyMap.containsKey(c);
100     }
101
102     /**
103     * Checks the <i>Shift Key Map</i> and obtains the shift key code
104     * for a character.
105     *
106     * @param c the character to be checked.
107     * @return the shift key code for <code>c</code>, <br/>
108     *         <code>-1</code>, if <code>Shift</code> key press
109     *         is not required.
110     */
111     static int getShiftCode(char c) {
112         if (!sShiftKeyMap.containsKey(c)) return -1;
113         return sShiftKeyMap.get(c);
114     }
115
116     /**
117     * Checks the <i>Key Map</i> and obtains the key code for a character
118     * (if <code>Shift</code> key press is not required).
119     *
120     * @param c the character to be checked.
121     * @return the key code for <code>c</code>, <br/>
122     *         <code>-1</code>, if <code>Shift</code> key press is
123     *         required.
124     */
125     static int getCodeByChar(char c) {
126         if (!sKeyMap.containsKey(c)) return -1;
127         return sKeyMap.get(c);
128     }
129 }

```


MouseController.java

```
1 package project.pc.device;
2
3 import project.pc.sensor.SensorDataHandler;
4 import project.pc.sensor.representation.Cartesian2D;
5 import project.pc.sensor.representation.Quaternion;
6
7 import java.awt.Robot;
8 import java.awt.AWTException;
9 import java.awt.MouseInfo;
10 import java.awt.event.InputEvent;
11
12 /**
13  * Utility for performing mouse input operations.
14  *
15  * <p>
16  * It uses a {@link java.awt.Robot} object to generate native
17  * system input events. This differs from posting events to
18  * the AWT event queue or AWT components in that the events are
19  * generated in the platform's native input queue.
20  * </p>
21  * <p>
22  * The {@link #move(Quaternion, boolean, float)} and
23  * {@link #move(int, int, float)} methods are invoked for moving the
24  * mouse pointer on the basis of 3-D and 2-D inputs from the
25  * mobile device respectively. The {@link #move(Quaternion, boolean, float)}
26  * method, in turn, calls the
27  * {@link MouseMove#moveRelatively(Robot, int, int, float)} method.
28  * The mouse button inputs are performed by invoking the
29  * {@link #doOperation(String)} method.
30  * </p>
31  *
32  * @see java.awt.Robot
33  * @see project.pc.sensor.representation.Quaternion
34  * @see project.pc.device.MouseMove#moveRelatively(Robot, int, int, float)
35  */
36 public class MouseController {
37     private Robot mRobot;
38     private SensorDataHandler mSensorDataHandler;
39
40     private static Quaternion sQuaternion = new Quaternion();
41
42     /**
43      * Constructor.
44      *
45      * Initializes this <code>MouseController</code> by
46      * instantiating a {@link java.awt.Robot} object.
47      *
48      * @throws AWTException
49      */
50     public MouseController() throws AWTException {
51         mRobot = new Robot();
52     }
53
54     /**
55      * Method to perform mouse move on the basis of 3-D input from
56      * the mobile device.
57      *
58      * @param quaternion a {@link project.pc.sensor.representation.Quaternion}
59      * object.
60      * @param isInitQuat <code>true</code>, if <code>quaternion</code> is
61      * initial, <br/>
62      * <code>false</code>, otherwise.
63      */
64     public void move(Quaternion quaternion, boolean isInitQuat, float sensitivity) {
65         sQuaternion = quaternion;
66         if(isInitQuat) {
67             mSensorDataHandler = new SensorDataHandler(quaternion,sensitivity);
68         }
69         Cartesian2D cartesian2D = mSensorDataHandler.pointerUpdate(quaternion);
70         int x = MouseInfo.getPointerInfo().getLocation().x;
71         int y = MouseInfo.getPointerInfo().getLocation().y;
72         x += (int) cartesian2D.getX();
73         y += (int) cartesian2D.getY();
74         mRobot.mouseMove(x, y);
75     }
76
77     /**
78      * Method to perform mouse move on the basis of 2-D input from
79      * the mobile device.
80      *
81      * @param relativeX displacement along x-axis.
82      * @param relativeY displacement along y-axis.
83      */
84     public void move(int relativeX, int relativeY, float sensitivity) {
85         MouseMove.moveRelatively(mRobot, relativeX, relativeY, sensitivity);
86     }
87
88     /**
```

```

89      * Method to perform mouse button click operations.
90      * <p>
91      *     <i>viz.</i>
92      *     <ul>
93      *         <li>Left click.</li>
94      *         <li>Right click.</li>
95      *         <li>Middle click.</li>
96      *         <li>Scrolling</li>
97      *     </ul>
98      * </p>
99      *
100     * @param operation <code>String</code> representing the type
101     *               of operation.
102     */
103     public void doOperation(String operation) {
104         switch (operation) {
105             case "left":
106                 mRobot.mousePress(InputEvent.BUTTON1_DOWN_MASK);
107                 mRobot.mouseRelease(InputEvent.BUTTON1_DOWN_MASK);
108                 break;
109
110             case "right":
111                 mRobot.mousePress(InputEvent.BUTTON3_DOWN_MASK);
112                 mRobot.mouseRelease(InputEvent.BUTTON3_DOWN_MASK);
113                 break;
114
115             case "middle":
116                 mRobot.mousePress(InputEvent.BUTTON2_DOWN_MASK);
117                 mRobot.mouseRelease(InputEvent.BUTTON2_DOWN_MASK);
118                 break;
119
120             case "upscroll":
121                 mRobot.mouseWheel(-2);
122                 break;
123
124             case "downscroll":
125                 mRobot.mouseWheel(2);
126         }
127     }
128     public void wait(int m) { mRobot.delay(1000); }
129     public static Quaternion getQuaternion() {
130         return sQuaternion;
131     }
132 }

```

MouseMove.java

```
1 package project.pc.device;
2
3 import java.awt.Dimension;
4 import java.awt.MouseInfo;
5 import java.awt.Robot;
6 import java.awt.Toolkit;
7
8 /**
9  * Provides static methods used by {@link MouseController} .
10  *
11  * <p>
12  *   The {@link #moveRelatively(Robot, int, int, float)} method takes
13  *   into account the dimension of the screen and handles the
14  *   input displacement in such a manner that the pointer
15  *   movement is proportional to the screen size.
16  * </p>
17  *
18  * @see project.pc.device.MouseController
19  */
20
21 class MouseMove {
22     private static int sScreenWidth;
23     private static int sScreenHeight;
24
25     static {
26         Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
27         sScreenWidth = (int) screenSize.getWidth();
28         sScreenHeight = (int) screenSize.getHeight();
29     }
30
31     /**
32      * Method for performing mouse movement proportional to the
33      * screen size. This method is invoked by
34      * {@link MouseController#move(int, int, float)}.
35      *
36      * @param robot the {@link java.awt.Robot} object.
37      * @param relativeX displacement along x-axis.
38      * @param relativeY displacement along y-axis.
39      * @param sensitivity mouse movement sensitivity.
40      */
41     static void moveRelatively(Robot robot, int relativeX, int relativeY, float sensitivity)
42     {
43         int curX = MouseInfo.getPointerInfo().getLocation().x;
44         int curY = MouseInfo.getPointerInfo().getLocation().y;
45         int moveX = (int) (curX + (/*ratio*/sensitivity * relativeX));
46         int moveY = (int) (curY + (/*ratio*/sensitivity * relativeY));
47         moveX = moveX < 0 ? 0 : moveX;
48         moveY = moveY < 0 ? 0 : moveY;
49         moveX = moveX >= sScreenWidth ? sScreenWidth - 1 : moveX;
50         moveY = moveY >= sScreenHeight ? sScreenHeight - 1 : moveY;
51         robot.mouseMove(moveX, moveY);
52     }
53 }
```

Cube.java

```
1 package project.pc.gui;
2
3 import com.jogamp.opengl.GLEventListener;
4 import com.jogamp.opengl.GLAutoDrawable;
5 import com.jogamp.opengl.GL2;
6 import com.jogamp.opengl.GL;
7 import com.jogamp.opengl.GLProfile;
8 import com.jogamp.opengl.GLCapabilities;
9 import com.jogamp.opengl.awt.GLCanvas;
10 import com.jogamp.opengl.glu.GLU;
11 import com.jogamp.opengl.util.FPSAnimator;
12 import project.pc.sensor.representation.Vector4f;
13
14 import javax.swing.JFrame;
15 import java.awt.event.WindowAdapter;
16 import java.awt.event.WindowEvent;
17
18 import static com.jogamp.opengl.GL.GL_ONE_MINUS_SRC_ALPHA;
19 import static com.jogamp.opengl.GL.GL_SRC_ALPHA;
20 import static project.pc.device.MouseController.getQuaternion;
21
22 /**
23  * Class to render multi-colored 3D cube.
24  *
25  * It is used for demonstration purposes.
26  */
27 public class Cube implements GLEventListener {
28
29     private GLU mGlu = new GLU();
30     private JFrame mJFrame;
31     private boolean mStarted;
32
33     /**
34      * Constructor.
35      *
36      * Initializes this <code>Cube</code> object.
37      */
38     Cube() { mStarted = false; }
39
40     /**
41      * Displays the 3D cube.
42      *
43      * @param drawable a <code>com.jogamp.opengl.GLAutoDrawable</code> object
44      *                used as the canvas.
45      */
46     @Override
47     public void display(GLAutoDrawable drawable) {
48         final GL2 gl = drawable.getGL().getGL2();
49         gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);
50         gl.glLoadIdentity();
51         gl.glTranslatef( 0f, 0f, -7f );
52
53         Vector4f mAxisAngle = new Vector4f();
54         getQuaternion().toAxisAngle(mAxisAngle);
55
56         // Rotate The Cube On X, Y & Z i.e. pass quaternion as argument
57         gl.glRotatef(mAxisAngle.getW(), mAxisAngle.getX(), mAxisAngle.getZ(),
58             (-1)*mAxisAngle.getY());
59
60         //giving different colors to different sides
61         gl.glBegin(GL2.GL_QUADS); // Start Drawing The Cube
62         gl.glColor3f(1f,0f,0f); //red color
63         gl.glVertex3f(1.0f, 1.0f, -1.0f); // Top Right Of The Quad (Top)
64         gl.glVertex3f(-1.0f, 1.0f, -1.0f); // Top Left Of The Quad (Top)
65         gl.glVertex3f(-1.0f, 1.0f, 1.0f); // Bottom Left Of The Quad (Top)
66         gl.glVertex3f( 1.0f, 1.0f, 1.0f); // Bottom Right Of The Quad (Top)
67
68         gl.glColor3f( 0f,1f,0f ); //green color
69         gl.glVertex3f( 1.0f, -1.0f, 1.0f ); // Top Right Of The Quad
70         gl.glVertex3f(-1.0f, -1.0f, 1.0f ); // Top Left Of The Quad
71         gl.glVertex3f(-1.0f, -1.0f, -1.0f ); // Bottom Left Of The Quad
72         gl.glVertex3f( 1.0f, -1.0f, -1.0f ); // Bottom Right Of The Quad
73
74         gl.glColor3f( 0f,0f,1f ); //blue color
75         gl.glVertex3f( 1.0f, 1.0f, 1.0f ); // Top Right Of The Quad (Front)
76         gl.glVertex3f(-1.0f, 1.0f, 1.0f ); // Top Left Of The Quad (Front)
77         gl.glVertex3f(-1.0f, -1.0f, 1.0f ); // Bottom Left Of The Quad
78         gl.glVertex3f( 1.0f, -1.0f, 1.0f ); // Bottom Right Of The Quad
79
80         gl.glColor3f( 1f,1f,0f ); //yellow (red + green)
81         gl.glVertex3f( 1.0f, -1.0f, -1.0f ); // Bottom Left Of The Quad
82         gl.glVertex3f(-1.0f, -1.0f, -1.0f ); // Bottom Right Of The Quad
83         gl.glVertex3f(-1.0f, 1.0f, -1.0f ); // Top Right Of The Quad (Back)
84         gl.glVertex3f( 1.0f, 1.0f, -1.0f ); // Top Left Of The Quad (Back)
85
86         gl.glColor3f( 1f,0f,1f ); //purple (red + green)
87         gl.glVertex3f(-1.0f, 1.0f, 1.0f ); // Top Right Of The Quad (Left)
88         gl.glVertex3f(-1.0f, 1.0f, -1.0f ); // Top Left Of The Quad (Left)
89         gl.glVertex3f(-1.0f, -1.0f, -1.0f ); // Bottom Left Of The Quad
90         gl.glVertex3f(-1.0f, -1.0f, 1.0f ); // Bottom Right Of The Quad
91
92         gl.glColor3f( 0f,1f, 1f ); //sky blue (blue +green)
```

```

92         gl.glVertex3f( 1.0f, 1.0f, -1.0f ); // Top Right Of The Quad (Right)
93         gl.glVertex3f( 1.0f, 1.0f, 1.0f ); // Top Left Of The Quad
94         gl.glVertex3f( 1.0f, -1.0f, 1.0f ); // Bottom Left Of The Quad
95         gl.glVertex3f( 1.0f, -1.0f, -1.0f ); // Bottom Right Of The Quad
96         gl.glEnd(); // Done Drawing The Quad
97         gl.glFlush();
98     }
99
100     @Override
101     public void dispose(GLAutoDrawable drawable) {}
102
103     /**
104     * Initializes the canvas.
105     *
106     * @param drawable {@link com.jogamp.opengl.GLAutoDrawable} object used as canvas.
107     */
108     @Override
109     public void init( GLAutoDrawable drawable ) {
110         final GL2 gl = drawable.getGL().getGL2();
111         gl.glShadeModel( GL2.GL_SMOOTH );
112         gl.glClearColor( 0f, 0f, 0f, 0f );
113         gl.glClearDepth( 1.0f );
114         gl.glEnable( GL2.GL_DEPTH_TEST );
115         gl.glDepthFunc( GL2.GL_LEQUAL );
116         gl.glEnable(GL.GL_LINE_SMOOTH);
117         gl.glHint( GL2.GL_PERSPECTIVE_CORRECTION_HINT, GL2.GL_NICEST );
118         gl.glHint( GL2.GL_POLYGON_SMOOTH_HINT, GL2.GL_NICEST );
119         gl.glEnable(GL.GL_BLEND);
120         gl.glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
121     }
122
123     /**
124     * Resizes the canvas.
125     *
126     * @param drawable the canvas to be resized.
127     * @param x x-coordinate of the lower left corner of the canvas.
128     * @param y y-coordinate of the lower left corner of the canvas.
129     * @param width width of the canvas.
130     * @param height height of the canvas.
131     */
132     @Override
133     public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
134         final GL2 gl = drawable.getGL().getGL2();
135         if( height <= 0 )
136             height = 1;
137
138         final float h = ( float ) width / ( float ) height;
139         gl.glViewport( 0, 0, width, height );
140         gl.glMatrixMode( GL2.GL_PROJECTION );
141         gl.glLoadIdentity();
142
143         mGlu.gluPerspective( 45.0f, h, 1.0, 30.0 );
144         gl.glMatrixMode( GL2.GL_MODELVIEW );
145         gl.glLoadIdentity();
146     }
147
148     /**
149     * Renders the cube on canvas.
150     */
151     void showCube() {
152         final GLProfile profile = GLProfile.get(GLProfile.GL2);
153         GLCapabilities capabilities = new GLCapabilities(profile);
154
155         // The canvas
156         final GLCanvas glcanvas = new GLCanvas( capabilities );
157         Cube cube = new Cube();
158
159         glcanvas.addGLEventListener(cube);
160         glcanvas.setSize(512, 512);
161
162         MJFrame = new JFrame ( " Multicolored Cube");
163         MJFrame.getContentPane().add(glcanvas);
164         MJFrame.setSize(MJFrame.getContentPane().getPreferredSize());
165         MJFrame.setVisible(true);
166         final FPSAnimator animator = new FPSAnimator(glcanvas, 300,true);
167         MJFrame.addWindowListener(new WindowAdapter() {
168             @Override
169             public void windowClosing(WindowEvent e) {
170                 animator.stop();
171                 MJFrame.dispose();
172                 mStarted = false;
173             }
174         });
175         animator.start();
176         mStarted = true;
177     }
178
179     /**
180     * Checks if rendering has been started.
181     *
182     * @return <code>true</code>, if currently rendering is in progress, <br/>
183     *         <code>false</code>, otherwise.
184     */
185     boolean isStarted() { return mStarted; }
186 }
187

```

MainWindow.java

```
1 package project.pc.gui;
2
3 import project.pc.net.NetworkManager;
4 import project.pc.net.NetworkState;
5 import project.pc.net.NetworkThread;
6 import project.pc.net.ServerThread;
7
8 import javax.swing.JFrame;
9 import javax.swing.JList;
10 import javax.swing.UIManager;
11 import javax.swing.JButton;
12 import javax.swing.JRadioButton;
13 import javax.swing.ButtonGroup;
14 import javax.swing.JPanel;
15 import javax.swing.JOptionPane;
16 import javax.swing.JDialog;
17
18 import java.awt.Dimension;
19 import java.awt.BorderLayout;
20 import java.awt.Insets;
21 import java.awt.GridLayout;
22 import java.awt.event.ActionEvent;
23 import java.awt.event.ActionListener;
24 import java.awt.event.WindowEvent;
25 import java.awt.event.WindowListener;
26
27 import java.io.IOException;
28 import java.net.InetAddress;
29 import java.util.List;
30
31 /**
32  * Defines the main window of the application.
33  *
34  * <p>
35  *   The contents of the main window are :
36  *   <ul>
37  *     <li>Couple of radio buttons for server start and stop.</li>
38  *     <li>A list displaying the clients currently connected.</li>
39  *     <li>
40  *       A disconnect button for disconnecting the selected client.
41  *     </li>
42  *     <li>A button form displaying a 3D demo.</li>
43  *   </ul>
44  *   It also implements <code>ActionListener</code> for these elements
45  *   and <code>WindowListener</code> for graceful termination of the
46  *   application upon closing of this <code>MainWindow</code>. The
47  *   window is initialized and rendered at the time of instantiation
48  *   by the constructor {@link #MainWindow(NetworkManager)}.
49  * </p>
50  *
51  * @see java.awt.event.ActionListener
52  * @see java.awt.event.WindowListener
53  * @see #MainWindow(NetworkManager)
54  */
55 public class MainWindow extends JFrame implements ActionListener, WindowListener {
56     private JList mList;
57     private boolean mLastSelectedOption;
58
59     private NetworkManager mManager;
60     private NetworkState mState;
61     private NetworkThread mNetworkThread;
62     private Thread mThread;
63     private final Cube mCube = new Cube();
64
65     private static final int WINDOW_WIDTH = 640;
66     private static final int WINDOW_HEIGHT = 480;
67     private static final long DIALOG_TIMEOUT = 2000;
68
69     /**
70      * Constructor.
71      *
72      * Initializes the <code>MainWindow</code> and displays it.
73      *
74      * @param manager the {@link project.pc.net.NetworkManager} for
75      *               this network.
76      */
77     public MainWindow(NetworkManager manager) {
78         this.mManager = manager;
79         mState = manager.getNetworkState();
80         mNetworkThread = new NetworkThread(manager);
81
82         try {
83             UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");
84         } catch (Exception e) {System.out.println("Error in loading look-and-feel ...");}
85
86         this.setTitle("SmartIO");
87         this.setMinimumSize(new Dimension(WINDOW_WIDTH, WINDOW_HEIGHT));
88         this.setDefaultCloseOperation(EXIT_ON_CLOSE);
89         this.addWindowListener(this);
90
91         this.setLayout(new BorderLayout());
92
93         JButton disconnectButton = new JButton("Disconnect");
94         disconnectButton.setMargin(new Insets(5,5,5,5));
95         disconnectButton.setActionCommand("disconnect_clicked");
96     }
```

```

97     disconnectButton.addActionListener(this);
98     JButton cubeButton = new JButton("3D Cube Demo");
99     cubeButton.setActionCommand("3d_clicked");
100    cubeButton.setMargin(new Insets(5,5,5,5));
101    cubeButton.addActionListener(this);
102    JPanel buttonPanel = new JPanel(new GridLayout(1,0));
103    buttonPanel.add(disconnectButton);
104    buttonPanel.add(cubeButton);
105
106    JRadioButton onButton = new JRadioButton();
107    onButton.setText("Server On");
108    onButton.setActionCommand("on");
109    JRadioButton offButton = new JRadioButton();
110    offButton.setActionCommand("off");
111    offButton.setText("Server Off");
112    offButton.setSelected(true);
113    mLastSelectedOption = false;
114    ButtonGroup bg = new ButtonGroup();
115    bg.add(offButton);
116    bg.add(onButton);
117    onButton.addActionListener(this);
118    offButton.addActionListener(this);
119    JPanel radioPanel = new JPanel(new GridLayout(1, 0));
120    radioPanel.add(onButton);
121    radioPanel.add(offButton);
122
123    mList = new JList<>(mState.getListModel());
124    mList.setLayoutOrientation(JList.VERTICAL);
125    mList.setFixedCellHeight(50);
126
127    this.add(buttonPanel, BorderLayout.SOUTH);
128    this.add(radioPanel, BorderLayout.NORTH);
129    this.add(mList, BorderLayout.CENTER);
130    this.setVisible(true);
131 }
132
133 /**
134  * Shows a dialog window displaying the pairing key. It gets closed
135  * when the pairing key is sent by the client for validation.
136  *
137  * @param title title of the dialog window - canonical hostname of
138  *             the client for which the pairing key is meant.
139  * @param pairingKey the pairing key.
140  * @param st the {@link project.pc.net.ServerThread} which handles
141  *           this client.
142  * @see project.pc.net.ServerThread
143  */
144 public static void showPairingKeyDialog(String title, String pairingKey, ServerThread st) {
145     String message = "<html>Type the following pairing key to connect your phone: <b><font
146     face=\"Monospaced\" \" +
147         \" size=\"5\" \" + \"color=\"red\" >\" + pairingKey + \"</font></b></html>\";
148     JOptionPane option = new JOptionPane(message, JOptionPane.INFORMATION_MESSAGE);
149     JDialog dialog = st.getDialog();
150     try {
151         dialog.dispose();
152     } catch (NullPointerException ignored) {}
153     dialog = option.createDialog(title);
154     dialog.setModal(false);
155     st.setDialog(dialog);
156     dialog.setVisible(true);
157 }
158
159 /**
160  * Shows a dialog window displaying the error that an incorrect pairing
161  * key is sent by the client.
162  *
163  * @param title title of the window - the canonical hostname of the
164  *             client.
165  * @param st the {@link project.pc.net.ServerThread} which handles this
166  *             client.
167  * @see project.pc.net.ServerThread
168  */
169 public static void showIncorrectPKeyDialog(String title, ServerThread st) {
170     JOptionPane option = new JOptionPane("Incorrect Pairing Key!", JOptionPane.ERROR_MESSAGE);
171     JDialog dialog = st.getDialog();
172     try {
173         dialog.dispose();
174     } catch (NullPointerException ignored) {}
175     dialog = option.createDialog(title);
176     dialog.setModal(false);
177     st.setDialog(dialog);
178     dialog.setVisible(true);
179     try {
180         Thread.sleep(DIALOG_TIMEOUT);
181         dialog.dispose();
182     } catch (InterruptedException | NullPointerException ignored) {}
183 }
184
185 /**
186  * Shows a dialog window displaying the information that a client is
187  * successfully connected.
188  *
189  * @param title title of the window - the canonical hostname of the
190  *             client.
191  * @param st the {@link project.pc.net.ServerThread} which handles
192  *             this client.
193  * @see project.pc.net.ServerThread
194  */
195 public static void showConnectionConfirmationDialog(String title, ServerThread st) {
196     String message = "Connected !!";
197     JOptionPane option = new JOptionPane(message, JOptionPane.INFORMATION_MESSAGE);

```



```

197     JDialog dialog = st.getDialog();
198     try {
199         dialog.dispose();
200     } catch (NullPointerException ignored) {}
201     dialog = option.createDialog(title);
202     dialog.setModal(false);
203     st.setDialog(dialog);
204     dialog.setVisible(true);
205     try {
206         Thread.sleep(DIALOG_TIMEOUT);
207         dialog.dispose();
208     } catch (InterruptedException | NullPointerException ignored) {}
209 }
210
211 /**
212  * Invoked when an action occurs.
213  *
214  * @param event a {@link java.awt.event.ActionEvent} object.
215  * @see java.awt.event.ActionEvent
216  */
217 @Override
218 public void actionPerformed(ActionEvent event) {
219     if (event.getActionCommand().equals("disconnect_clicked")) {
220         List<String> selectedValues = mList.getSelectedValuesList();
221         try {
222             for (String values: selectedValues) {
223                 values = values.substring(values.indexOf("(") + 1, values.length() - 1);
224                 mManager.disconnect(InetAddress.getByName(values));
225             }
226         } catch (IOException | InterruptedException e) {
227             e.printStackTrace();
228         }
229         System.out.println("disconnect clicked");
230     }
231
232     else if (event.getActionCommand().equals("on") && !mLastSelectedOption) {
233         mLastSelectedOption = true;
234         System.out.println("on clicked");
235         mThread = new Thread(mNetworkThread);
236         mThread.start();
237     }
238
239     else if (event.getActionCommand().equals("off") && mLastSelectedOption) {
240         mLastSelectedOption = false;
241         System.out.println("off clicked");
242         try {
243             mManager.stopServer();
244             mThread = null;
245         } catch (Exception e) { e.printStackTrace(); }
246     }
247     else if (event.getActionCommand().equals("3d_clicked")) {
248         if (!mState.isNoClientConnected() && !mCube.isStarted())
249             mCube.showCube();
250     }
251 }
252
253 /**
254  * Gracefully terminates the application.
255  *
256  * <p>
257  * Termination is done by stopping the broadcast disconnecting
258  * every client and finally stopping the server. It is called
259  * during the closing of this <code>MainWindow</code>.
260  * </p>
261  *
262  * @param windowEvent a {@link java.awt.event.WindowEvent} object.
263  * @see java.awt.event.WindowEvent
264  */
265 @Override
266 public void windowClosing(WindowEvent windowEvent) {
267     try {
268         mManager.stopServer();
269         mThread = null;
270     } catch (IOException | InterruptedException e) {
271         e.printStackTrace();
272     }
273 }
274
275 @Override
276 public void windowOpened(WindowEvent windowEvent) {}
277
278 @Override
279 public void windowClosed(WindowEvent windowEvent) {}
280
281 @Override
282 public void windowIconified(WindowEvent windowEvent) {}
283
284 @Override
285 public void windowDeiconified(WindowEvent windowEvent) {}
286
287 @Override
288 public void windowActivated(WindowEvent windowEvent) {}
289
290 @Override
291 public void windowDeactivated(WindowEvent windowEvent) {}

```


BroadcastThread.java

```
1 package project.pc.net;
2
3 import com.google.gson.Gson;
4
5 import java.io.IOException;
6 import java.net.NetworkInterface;
7 import java.net.InterfaceAddress;
8 import java.net.InetAddress;
9 import java.net.DatagramSocket;
10 import java.net.DatagramPacket;
11 import java.util.Enumeration;
12 import java.util.List;
13
14 /**
15  * Implementation of the {@link java.lang.Runnable} for the thread
16  * responsible for making UDP broadcasts.
17  *
18  * <p>
19  * This thread continuously sends UDP broadcasts through all
20  * available network interfaces of the PC except point-to-point
21  * and loopback interfaces. The content of the broadcast data is
22  * a JSon-string of a {@link ServerInfo} object.
23  * </p>
24  * <p>
25  * It is started during server start by the method
26  * {@link NetworkManager#startServer()}. It is stopped during
27  * server stop by the method {@link NetworkManager#stopServer()}.
28  * The interval between two successive broadcast is
29  * <code>1000</code> ms.
30  * </p>
31  * <p>
32  * This UDP broadcast is used for device discovery from the Android
33  * devices.
34  * </p>
35  *
36  * @see java.lang.Runnable
37  * @see project.pc.net.ServerInfo
38  * @see com.google.gson.Gson
39  * @see project.pc.net.NetworkManager#startServer()
40  * @see project.pc.net.NetworkManager#stopServer()
41  */
42
43 class BroadcastThread implements Runnable {
44
45     private ServerInfo mServerInfo;
46     private boolean mStopFlag;
47     private static final int BROADCAST_PORT = 1235;
48     private static final int TIMEOUT = 1000;
49
50     /**
51      * Stops the broadcast.
52      *
53      * Used to exit from the {@link #run()} method.
54      */
55     void stopBroadcast() {
56         mStopFlag = true;
57         mServerInfo.setStopFlag();
58         byte[] data = new Gson().toJson(mServerInfo).getBytes();
59         new Thread() -> {
60             try {
61                 broadcastData(data);
62             } catch (IOException ignored) {}
63         }.start();
64     }
65
66     /**
67      * Returns the status of the broadcast.
68      *
69      * @return <code>true</code>, if broadcast is active, <br/>
70      *         <code>false</code>, otherwise.
71      */
72     boolean getBroadcastFlag() { return mStopFlag; }
73
74     /**
75      * Returns the interval between two successive broadcasts.
76      *
77      * @return the interval between two successive broadcasts in milliseconds.
78      */
79     long getTimeout() { return TIMEOUT; }
80
81     /**
82      * Constructor.
83      *
84      * Initializes this thread.
85      *
86      * @param serverInfo the {@link ServerInfo} object containing the data to
87      *                   be broadcast.
88      * @see project.pc.net.ServerInfo
89      */
90 }
```

```

90 BroadcastThread(ServerInfo serverInfo) { mServerInfo = serverInfo; }
91
92 /**
93  * Operations performed by this thread.
94  *
95  * <p>
96  * It continuously sends UDP broadcasts through all
97  * available network interfaces of the PC except point-to-point
98  * and loopback interface until {@link #stopBroadcast()} is
99  * invoked.
100  * </p>
101  */
102 @Override
103 public void run() {
104     mStopFlag = false;
105     try {
106         byte[] data;
107         while(!mStopFlag) {
108             mServerInfo.clearStopFlag();
109             data = new Gson().toJson(mServerInfo).getBytes();
110             broadcastData(data);
111             Thread.sleep(TIMEOUT);
112         }
113     } catch (InterruptedException | IOException e) {
114         e.printStackTrace();
115     }
116 }
117
118 private void broadcastData(byte[] data) throws IOException {
119     Enumeration<NetworkInterface> nis = NetworkInterface.getNetworkInterfaces();
120     while (nis.hasMoreElements()) {
121         NetworkInterface ni = nis.nextElement();
122         if (!ni.isLoopback()) {
123             List<InterfaceAddress> ias = ni.getInterfaceAddresses();
124             for (InterfaceAddress addr: ias) {
125                 InetAddress broadcastIA = addr.getBroadcast();
126                 if (broadcastIA != null) {
127                     DatagramSocket datagramSocket = new DatagramSocket();
128                     datagramSocket.setBroadcast(true);
129                     DatagramPacket datagramPacket = new DatagramPacket(data,
130 data.length,
131 broadcastIA, BROADCAST_PORT);
132 datagramSocket.send(datagramPacket);
133 datagramSocket.close();
134                 }
135             }
136         }
137     }
138 }

```

ClientInfo.java

```
1 package project.pc.net;
2
3 import java.net.Socket;
4
5 /**
6  * Encapsulation of the data received from the client at the time of
7  * establishing TCP connection.
8  *
9  * <p>
10 * The data contains :
11 * <ul>
12 * <li>Public key of the client</li>
13 * <li>Canonical hostname of the client</li>
14 * <li>Client <code>Socket</code></li>
15 * </ul>
16 * </p>
17 *
18 * @see java.net.Socket
19 * @see project.pc.net.ServerThread
20 */
21
22 public class ClientInfo {
23     private String mClientInfo;
24     private byte[] mPublicKey;
25     private Socket mClientSocket;
26
27     /**
28      * Constructor.
29      *
30      * <p>
31      * Initializes this <code>ClientInfo</code> object. The stored
32      * object of {@link java.net.Socket} (client <code>Socket</code>)
33      * is not initialized. It can be set by the {@link #setSocket(Socket)}
34      * method.
35      * </p>
36      *
37      * @param clientInfo canonical hostname of the client.
38      * @param publicKey public key of the client.
39      * @see java.net.Socket
40      * @see #setSocket(Socket)
41      */
42     public ClientInfo(String clientInfo, byte[] publicKey) {
43         mClientInfo = clientInfo;
44         mPublicKey = publicKey;
45     }
46
47     /**
48      * Adds {@link java.net.Socket} of the client.
49      *
50      * @param socket the client {@link java.net.Socket} object to be
51      * stored.
52      */
53     public void setSocket(Socket socket) { mClientSocket = socket; }
54
55     /**
56      * Returns the client <code>Socket</code>.
57      *
58      * @return the client {@link java.net.Socket}.
59      */
60     public Socket getSocket() { return mClientSocket; }
61
62     /**
63      * Returns the canonical hostname of the client.
64      *
65      * @return canonical hostname of the client
66      */
67     public String getClientInfo() { return mClientInfo; }
68
69     /**
70      * Returns the public key of the client.
71      *
72      * @return public key of the client.
73      */
74     public byte[] getPublicKey() { return mPublicKey; }
75 }
```

DataWrapper.java

```
1 package project.pc.net;
2
3
4 import com.google.gson.Gson;
5 import project.pc.sensor.representation.Quaternion;
6
7 /**
8  * Encapsulates the data received from client during TCP communication.
9  *
10  * <p>
11  *   This data contains all the information required by the
12  *   {@link project.pc.device.MouseController} and the
13  *   {@link project.pc.device.KeyboardController} for doing the different
14  *   input operations. It contains a <i>type of operation</i> field which
15  *   helps to determine the operation requested by the client and a
16  *   <i>data</i> field which acts as the operand for the input operations.
17  * </p>
18  * <p>
19  *   For keyboard input, the <i>type of operation</i> field contains
20  *   <code>Key</code> or <code>Special_Key</code>. The <i>data</i> field
21  *   contains the sequence of characters and keyboard strokes
22  *   for input. In case of special keys, the <i>data</i> field
23  *   contains the special key identifier.
24  * </p>
25  * <p>
26  *   For 3-D mouse movement operation, the <i>type of operation</i>
27  *   field contains <code>Mouse_Move</code> and the <i>data</i> field
28  *   contains a {@link project.pc.sensor.representation.Quaternion}
29  *   object.
30  * </p>
31  * <p>
32  *   For 2-D mouse movement operation, the <i>type of operation</i>
33  *   field contains <code>Mouse_Touch</code> and the <i>data</i> field
34  *   contains a pair of integers representing the displacement along the
35  *   x-axis and y-axis relative to the current mouse pointer location.
36  * </p>
37  * <p>
38  *   For mouse button click operation, the <i>type of operation</i>
39  *   field contains <code>Mouse_Button</code> and the <i>data</i> field
40  *   contains any of the following :
41  *   <ul>
42  *     <li><code>left</code> - representing left click.</li>
43  *     <li><code>right</code> - representing right click.</li>
44  *     <li><code>middle</code> - representing middle button click.</li>
45  *     <li><code>upscroll</code> - representing scroll up.</li>
46  *     <li><code>downscroll</code> - representing scroll down.</li>
47  *   </ul>
48  * </p>
49  *
50  * @see project.pc.sensor.representation.Quaternion
51  * @see project.pc.device.MouseController
52  * @see project.pc.device.KeyboardController
53  */
54
55 public class DataWrapper {
56
57     private String mOperationType;
58     private String mData;
59     private Quaternion mQuaternion;
60
61     private boolean mIsInitQuat;
62     private int mMoveX;
63     private int mMoveY;
64     private float mSensitivity;
65
66     /**
67      * Constructor.
68      *
69      * Used to create a <code>DataWrapper</code> for keyboard and mouse
70      * button click operations.
71      *
72      * @param operationType the <i>type of operation field</i>.
73      * @param data the <i>data</i> field.
74      */
75     DataWrapper(String operationType, String data) {
76         mOperationType = operationType;
77         mData = data;
78     }
79
80     /**
81      * Constructor.
82      *
83      * Used to create a <code>DataWrapper</code> for special keys like Ctrl, Alt etc.
84      */
85 }
```

```

85     * @param data the <i>data</i> field.
86     */
87     DataWrapper(String data) {
88         mOperationType = "Special_Key";
89         mData = data;
90     }
91
92     /**
93     * Constructor.
94     *
95     * Used to create a <code>DataWrapper</code> for 2-D mouse movement
96     * operation. The <i>type of operation</i> field is implicit.
97     *
98     * @param x displacement data along x-axis.
99     * @param y displacement data along y-axis.
100    * @param sensitivity mouse movement sensitivity.
101    */
102    DataWrapper(int x, int y, float sensitivity) {
103        mMoveX = x;
104        mMoveY = y;
105        mSensitivity = sensitivity;
106        mOperationType = "Mouse_Touch";
107    }
108
109    /**
110    * Constructor.
111    *
112    * Used to create a <code>DataWrapper</code> for 3-D mouse movement
113    * operation. The <i>type of operation</i> field is implicit.
114    *
115    * @param quaternionObject a {@link project.pc.sensor.representation.Quaternion}
116    * object.
117    * @param isInitQuat <code>true</code>, if <code>quaternionObject</code>
118    * is the initial, <br/>
119    * <code>false</code>, otherwise.
120    * @param sensitivity mouse movement sensitivity.
121    * @see project.pc.sensor.representation.Quaternion
122    */
123    DataWrapper(Quaternion quaternionObject, boolean isInitQuat, float sensitivity) {
124        mOperationType = "Mouse_Move";
125        mIsInitQuat = isInitQuat;
126        mSensitivity = sensitivity;
127        mQuaternion = quaternionObject;
128    }
129
130    /**
131    * Returns a JSon-string of an object.
132    *
133    * @param object the input {@link java.lang.Object}.
134    * @return a JSon-string
135    * @see com.google.gson.Gson
136    */
137    static String getGsonString(Object object) {
138        return new Gson().toJson(object);
139    }
140
141    /**
142    * Returns the <i>type of operation</i> field of this
143    * <code>DataWrapper</code>.
144    *
145    * @return the <i>type of operation</i> field.
146    */
147    String getOperationType() { return mOperationType; }
148
149    /**
150    * Returns the <i>data</i> field of this <code>DataWrapper</code>
151    * for keyboard and mouse button click operations.
152    *
153    * @return the <i>data</i> field.
154    */
155    String getData() { return mData; }
156
157    /**
158    * Checks if the stored {@link project.pc.sensor.representation.Quaternion}
159    * object is the initial one.
160    *
161    * @return <code>true</code>, if it is initial, <br/>
162    * <code>false</code>, otherwise.
163    */
164    boolean isInitQuat() { return mIsInitQuat; }
165
166    /**
167    * Returns the {@link project.pc.sensor.representation.Quaternion}
168    * object stored in this <code>DataWrapper</code>.
169    *
170    * @return the {@link project.pc.sensor.representation.Quaternion}
171    * object.
172    */
173    Quaternion getQuaternionObject() { return mQuaternion; }

```

```

174
175
176     /**
177     * Returns the displacement value along x-axis stored in this
178     * <code>DataWrapper</code>.
179     * @return the displacement along x-axis.
180     */
181     int getX() { return mMoveX; }
182
183     /**
184     * Returns the displacement value along y-axis stored in this
185     * <code>DataWrapper</code>.
186     * @return the displacement along y-axis.
187     */
188     int getY() { return mMoveY; }
189
190     /**
191     * Returns the mouse movement sensitivity value stored in this
192     * <code>DataWrapper</code>.
193     * @return the mouse movement sensitivity.
194     */
195     float getSensitivity() { return mSensitivity; }
196
197 }
198

```

NetworkManager.java

```
1 package project.pc.net;
2
3 import java.awt.AWTException;
4 import java.io.IOException;
5 import java.net.InetAddress;
6 import java.net.Socket;
7 import java.util.HashMap;
8 import java.util.Iterator;
9 import java.util.Map;
10
11 /**
12  * Manages the different operations on the network.
13  *
14  * <p>
15  * This class creates a {@link NetworkState} which stores the information
16  * regarding the network. It provides methods for start and stop of the
17  * server. It also provides method for disconnecting a client. These
18  * methods modify the <code>NetworkState</code> accordingly.
19  * </p>
20  * <p>
21  * This class provides back-end of the network operations initiated by the
22  * GUI of SmartIO.
23  * </p>
24  *
25  * @see project.pc.net.NetworkState
26  */
27 public class NetworkManager {
28
29     private Server mServer;
30     private BroadcastThread mBroadcastThread;
31     private Thread mThread;
32     private NetworkState mState;
33
34     /**
35      * Constructor.
36      * <p>
37      * This instantiates {@link NetworkState} and {@link BroadcastThread}.
38      * The <code>BroadcastThread</code> is not started in the constructor.
39      * It can be started using {@link #startServer()}.
40      * </p>
41      *
42      * @param serverInfo a {@link ServerInfo} object.
43      * @see project.pc.net.ServerInfo
44      * @see project.pc.net.BroadcastThread
45      * @see #startServer()
46      */
47     public NetworkManager(ServerInfo serverInfo) {
48         mState = new NetworkState();
49         mBroadcastThread = new BroadcastThread(serverInfo);
50     }
51
52     /**
53      * Returns the <code>NetworkState</code> of this network.
54      *
55      * @return The {@link NetworkState} of this network
56      */
57     public NetworkState getNetworkState() { return mState; }
58
59     /**
60      * Method to start server.
61      * <p>
62      * This is achieved by executing two primary operations :
63      * <ul>
64      * <li>Starting the <code>BroadcastThread</code>.</li>
65      * <li>Calling {@link Server#listen()}.</li>
66      * </ul>
67      * </p>
68      *
69      * @throws IOException
70      * @throws InterruptedException
71      * @throws AWTException
72      * @see project.pc.net.BroadcastThread
73      * @see Server#listen()
74      */
75     void startServer() throws IOException, InterruptedException, AWTException {
76         mThread = new Thread(mBroadcastThread);
77         mServer = new Server(mState, mBroadcastThread);
78         mThread.start();
79         System.out.println("Broadcast started ...");
80         System.out.println("Server started ...");
81         mServer.listen();
82     }
83
84     /**
```

```

85      * Method to stop server.
86      * <p>
87      *     This is achieved by executing three primary operations :
88      *     <ul>
89      *         <li>
90      *             Stopping <code>BroadcastThread</code> by calling
91      *             {@link BroadcastThread#stopBroadcast()}.
92      *         </li>
93      *         <li>Calling {@link Server#setStopFlag()}.</li>
94      *         <li>
95      *             Disconnecting all clients. This is done by calling
96      *             {@link #disconnect(InetAddress)} for every connection
97      *             stored in the network state.
98      *         </li>
99      *     </ul>
100     * </p>
101     *
102     * @throws IOException
103     * @throws InterruptedException
104     * @see project.pc.net.BroadcastThread
105     * @see BroadcastThread#stopBroadcast()
106     * @see Server#setStopFlag()
107     * @see #disconnect(InetAddress)
108     */
109
110     @SuppressWarnings("unchecked")
111     public void stopServer() throws IOException, InterruptedException {
112         mBroadcastThread.stopBroadcast();
113         Thread.sleep(mBroadcastThread.getTimeout());
114         mThread = null;
115         System.out.println("Broadcast stopped ...");
116
117         if(mServer != null) {
118             mServer.setStopFlag();
119             Thread.sleep(mServer.getTimeout());
120             mServer.close();
121             mServer = null;
122
123             HashMap<InetAddress, Socket> aMap = mState.getsAddressMap();
124             Iterator it = aMap.entrySet().iterator();
125             while(it.hasNext()) {
126                 Map.Entry<InetAddress, Socket> entry = (Map.Entry<InetAddress,
127                 Socket>) it.next();
128                 disconnect(entry.getKey());
129             }
130             System.out.println("Server stopped ...");
131         }
132     }
133
134     /**
135     * Method to disconnect a client.
136     *
137     * It gets the {@link ServerThread} of the connection of the client
138     * to be disconnected from the network state by calling
139     * {@link NetworkState#getServerThread(InetAddress)}.
140     *
141     * @param address {@link java.net.InetAddress} of the client to be
142     *                 disconnected.
143     * @throws IOException
144     * @throws InterruptedException
145     * @see project.pc.net.ServerThread
146     * @see NetworkState#getServerThread(InetAddress)
147     */
148     public void disconnect(InetAddress address) throws IOException,
149     InterruptedException {
150         ServerThread st = mState.getServerThread(address);
151         try {
152             st.setStopFlag();
153             Thread.sleep(st.getTimeout());
154             System.out.println(address.getHostAddress() + " disconnected");
155         } catch (NullPointerException e) {System.out.println("No connected devices.
156         Skipping ...");}
157     }

```


NetworkState.java

```
1 package project.pc.net;
2
3 import javax.swing.*;
4 import java.net.InetAddress;
5 import java.net.Socket;
6 import java.util.HashMap;
7
8 /**
9  * Stores the state information about the network.
10  *
11  * <p>
12  *   It stores the information about the client connections with the help of
13  *   two data structures :
14  *   <ol>
15  *     <li>
16  *       A <i>Connection map</i> which is a <code>HashMap</code> from a
17  *       client <code>Socket</code> to the {@link ServerThread} handling
18  *       that client.
19  *     </li>
20  *     <li>
21  *       An <i>Address map</i> which is a <code>HashMap</code> from the
22  *       <code>InetAddress</code> of a client to the <code>Socket</code>.
23  *     </li>
24  *   </ol>
25  *   The methods to manipulate these data structures are provided in the class.
26  *   The {@link #add(ClientInfo, ServerThread)} and {@link #remove(ClientInfo)}
27  *   methods are invoked by {@link ServerThread} once it successfully
28  *   establishes and closes a client connection respectively.
29  * </p>
30  * <p>
31  *   It also stores a {@link javax.swing.DefaultListModel} which is used by
32  *   {@link project.pc.gui.MainWindow} to populate a list displaying the client
33  *   connections
34  *   in the GUI.
35  * </p>
36  * @see java.util.HashMap
37  * @see project.pc.net.ServerThread
38  * @see project.pc.gui.MainWindow
39  * @see javax.swing.DefaultListModel
40  */
41 public class NetworkState {
42
43     private static HashMap<Socket, ServerThread> sConnectionMap = new HashMap<>();
44     private static HashMap<InetAddress, Socket> sAddressMap = new HashMap<>();
45     private DefaultListModel<String> mListModel;
46
47     /**
48      * Constructor.
49      *
50      * Initializes the state of the network.
51      */
52     NetworkState() { mListModel = new DefaultListModel<>(); }
53
54     /**
55      * Returns the {@link javax.swing.ListModel}.
56      *
57      * It is used by the GUI to populate a list displaying the client
58      * connections.
59      *
60      * @return a {@link javax.swing.DefaultListModel}.
61      * @see project.pc.gui.MainWindow
62      */
63     public DefaultListModel getListModel() { return mListModel; }
64
65     /**
66      * Returns the <i>Connection Map</i>.
67      *
68      * @return <code>HashMap<Socket, ServerThread></code>.
69      * @see java.util.HashMap
70      */
71     HashMap<Socket, ServerThread> getConnectionMap() { return sConnectionMap; }
72
73     /**
74      * Returns the <i>Address Map</i>.
75      *
76      * @return <code>HashMap<InetAddress, Socket></code>.
77      * @see java.util.HashMap
78      */
79     HashMap<InetAddress, Socket> getAddressMap() { return sAddressMap; }
80
81     /**
82      * Returns the {@link ServerThread} for a client connection.
83      *

```

```

84     * @param ia {@link java.net.InetAddress} of the client.
85     * @return <code>ServerThread</code> of the client connection.
86     */
87     ServerThread getServerThread(InetAddress ia) { return
sConnectionMap.get(sAddressMap.get(ia)); }
88
89     /**
90     * Checks if any client is connected.
91     *
92     * @return <code>true</code>, if no client is connected, <br/>
93     *         <code>false</code>, otherwise.
94     */
95     public boolean isNoClientConnected() { return mListModel.isEmpty(); }
96
97
98     /**
99     * Adds a client connection to this <code>NetworkState</code>.
100    *
101    * @param clientInfo the {@link ClientInfo}.
102    * @param serverThread <code>ServerThread</code> of the client connection.
103    */
104    void add(ClientInfo clientInfo, ServerThread serverThread) {
105        Socket socket = clientInfo.getSocket();
106        sConnectionMap.put(socket, serverThread);
107        sAddressMap.put(socket.getInetAddress(), socket);
108        String client = clientInfo.getClientInfo() + " (" +
socket.getInetAddress().getHostAddress() + ")";
109        if(!mListModel.contains(client)) mListModel.addElement(client);
110    }
111
112    /**
113    * Removes a client connection from this <code>NetworkState</code>.
114    *
115    * @param clientInfo the {@link ClientInfo}.
116    */
117    void remove(ClientInfo clientInfo) {
118        Socket socket = clientInfo.getSocket();
119        if(sConnectionMap.containsKey(socket)) sConnectionMap.remove(socket);
120        if(sAddressMap.containsKey(socket.getInetAddress()))
sAddressMap.remove(socket.getInetAddress());
121        String client = clientInfo.getClientInfo() + " (" +
socket.getInetAddress().getHostAddress() + ")";
122        if(mListModel.contains(client)) mListModel.removeElement(client);
123    }
124 }

```

NetworkThread.java

```
1 package project.pc.net;
2
3 import java.awt.AWTException;
4 import java.awt.event.ActionEvent;
5 import java.io.IOException;
6
7 /**
8  * Implementation of the {@link java.lang.Runnable} for the thread that
9  * executes the network operations.
10  *
11  * <p>
12  * This keeps away the network operations from the <i>Swing Event
13  * Dispatch Thread</i> and prevents the GUI from becoming unresponsive.
14  * This thread gets started by {@link project.pc.gui.MainWindow} while
15  * performing the action of starting the server - the first possible
16  * network operation.
17  * </p>
18  *
19  * @see java.lang.Runnable
20  * @see project.pc.gui.MainWindow#actionPerformed(ActionEvent)
21  */
22 public class NetworkThread implements Runnable {
23     private NetworkManager manager;
24
25     /**
26      * Constructor.
27      *
28      * Initializes the thread.
29      *
30      * @param manager the {@link NetworkManager} for this network.
31      * @see project.pc.net.NetworkManager
32      */
33     public NetworkThread(NetworkManager manager) { this.manager = manager; }
34
35     /**
36      * Operations performed by this <code>NetworkThread</code>.
37      */
38     @Override
39     public void run() {
40         try {
41             manager.startServer();
42         } catch (IOException | AWTException | InterruptedException e) {
43             e.printStackTrace();
44         }
45     }
46 }
```

Server.java

```
1 package project.pc.net;
2
3 import java.net.ServerSocket;
4 import java.net.Socket;
5 import java.net.SocketException;
6 import java.net.SocketTimeoutException;
7 import java.io.IOException;
8 import java.awt.AWTException;
9
10 import project.pc.device.KeyboardController;
11 import project.pc.device.MouseController;
12
13 /**
14  * Server data structure.
15  *
16  * <p>
17  * It encapsulates the {@link java.net.ServerSocket} used for
18  * establishing TCP connection with the client.
19  * </p>
20  * <p>
21  * When the <code>Server</code> starts, the {@link #listen()}
22  * method is called which listens on the port in order to accept
23  * incoming client connections. Upon accepting connection it
24  * instantiates and starts a {@link ServerThread} for each
25  * connection.
26  * </p>
27  * <p>
28  * It also instantiates the {@link project.pc.device.MouseController}
29  * and {@link project.pc.device.KeyboardController} which is used for
30  * mouse and keyboard control respectively.
31  * </p>
32  *
33  * @see java.net.ServerSocket
34  * @see project.pc.net.ServerThread
35  * @see #listen()
36  * @see project.pc.device.MouseController
37  * @see project.pc.device.KeyboardController
38  */
39
40 class Server {
41
42     private ServerSocket mServerSocket;
43     private BroadcastThread mBroadcastThread;
44     private MouseController mMouseController;
45     private KeyboardController mKeyboardController;
46     private NetworkState mState;
47
48     private boolean mStopFlag;
49     private static final int TCP_PORT = 1234;
50
51     /**
52      * Constructor.
53      *
54      * Initializes the server.
55      *
56      * @param state {@link NetworkState} of this network.
57      * @param broadcastThread the {@link BroadcastThread}.
58      * @throws IOException
59      * @throws AWTException
60      */
61     Server(NetworkState state, BroadcastThread broadcastThread) throws IOException,
62     AWTException {
63         mState = state;
64         mBroadcastThread = broadcastThread;
65         mMouseController = new MouseController();
66         mKeyboardController = new KeyboardController();
67         mServerSocket = new ServerSocket(TCP_PORT);
68         mServerSocket.setSoTimeout(200);
69         mStopFlag = false;
70     }
71
72     /**
73      * Sets the stop flag.
74      *
75      * Used to exit from the {@link #listen()} method.
76      *
77      * @see #listen()
78      */
79     void setStopFlag() { mStopFlag = true; }
80
81     /**
82      * Returns the timeout of the {@link java.net.ServerSocket} used
83      * by this <code>Server</code>.
84      */
85 }
```

```

84     * @return the timeout in milliseconds of this <code>Server</code>.
85     * @throws IOException
86     */
87     int getTimeout() throws IOException { return mServerSocket.getSoTimeout(); }
88
89     /**
90     * Listens continuously on the <i>TCP PORT</i>.
91     *
92     * <p>
93     *     Once the server accepts a connection, a separate
94     *     {@link ServerThread} is started for each connection. The
95     *     <i>listening</i> can be stopped by invoking the
96     *     {@link #setStopFlag()} method.
97     * </p>
98     *
99     * @throws IOException
100    * @throws InterruptedException
101    * @see project.pc.net.ServerThread
102    * @see #setStopFlag()
103    */
104    void listen() throws IOException, InterruptedException {
105        while(!mStopFlag) {
106            try {
107                Socket clientSocket = mServerSocket.accept();
108
109                ServerThread st = new ServerThread(mState, clientSocket,
mMouseController,
110                    mKeyboardController, mBroadcastThread);
111                Thread t = new Thread(st);
112                t.start();
113            } catch (SocketTimeoutException | SocketException ignored) {}
114        }
115    }
116
117    /**
118    * Closes the <code>ServerSocket</code> used by this
119    * <code>Server</code>.
120    *
121    * @throws IOException
122    */
123    void close() throws IOException {
124        mServerSocket.close();
125    }
126 }

```

ServerInfo.java

```
1 package project.pc.net;
2
3 import project.pc.security.EKEProvider;
4
5 /**
6  * Encapsulation of the data sent during UDP broadcast.
7  *
8  * <p>
9  *   This data is broadcast by the {@link BroadcastThread}. This
10  *   contains :
11  *   <ul>
12  *     <li>the public key of the server, </li>
13  *     <li>status of the broadcast, and</li>
14  *     <li>address of the server</li>
15  *   </ul>
16  * </p>
17  * <p>
18  *   This message helps a client to discover a server.
19  * </p>
20  *
21  * @see project.pc.net.BroadcastThread
22  */
23
24 public class ServerInfo {
25     private byte[] mServerPubKey;
26     private String mServerInfo;
27     private String mAddress;
28     private boolean mStopFlag;
29
30     /**
31      * Constructor.
32      *
33      * Initializes the <code>ServerInfo</code>.
34      *
35      * @param publicKey public key of the server generated by
36      *   {@link
37      *   project.pc.security.EKEProvider#getBase64EncodedPubKey()}.
38      *   @param serverInfo canonical hostname of the server.
39      *   @see project.pc.security.EKEProvider#getBase64EncodedPubKey()
40      */
41     public ServerInfo(byte[] publicKey, String serverInfo) {
42         mServerPubKey = publicKey;
43         mServerInfo = serverInfo;
44         mStopFlag = false;
45     }
46
47     /**
48      * Compares two <code>ServerInfo</code> objects.
49      *
50      * @param obj the object to be compared.
51      * @return <code>true</code>, if the two objects are equal, <br/>
52      *   <code>false</code>, otherwise.
53      */
54     @Override
55     public boolean equals(Object obj) {
56         if (this == obj) return true;
57         if (obj == null) return false;
58         if (getClass() != obj.getClass()) return false;
59         ServerInfo other = (ServerInfo) obj;
60         return mAddress.equals(other.mAddress);
61     }
62
63     void setServerAddress(String address) { mAddress = address; }
64
65     /**
66      * Sets the status of the <i>broadcast</i> as <i>inactive</i>.
67      */
68     void setStopFlag() { mStopFlag = true; }
69
70     /**
71      * Sets the status of the <i>broadcast</i> as <i>active</i>.
72      */
73     void clearStopFlag() { mStopFlag = false; }
74
75     public boolean getStopFlag() { return mStopFlag; }
76     public byte[] getServerPubKey() { return mServerPubKey; }
77     public String getServerInfo() { return mServerInfo; }
78     public String getAddress() { return mAddress; }
```

ServerThread.java

```
1 package project.pc.net;
2
3 import java.io.PrintWriter;
4 import java.net.InetAddress;
5 import java.net.Socket;
6 import java.io.BufferedReader;
7 import java.io.InputStreamReader;
8 import java.io.IOException;
9
10 import com.google.gson.Gson;
11 import project.pc.device.KeyboardController;
12 import project.pc.device.MouseController;
13 import project.pc.gui.MainWindow;
14 import project.pc.security.EKEProvider;
15
16 import javax.swing.JDialog;
17
18 /**
19  * Implementation of the {@link java.lang.Runnable} for the thread that is
20  * responsible for communication with each client.
21  *
22  * <p>
23  * The actions performed by this thread are :
24  * <ul>
25  * <li>Authenticate the client.</li>
26  * <li>According to the result
27  * <ul>
28  * <li>
29  * If the client is not authentic, close the connection
30  * and terminate.
31  * </li>
32  * <li>
33  * Otherwise, confirm the connection and start
34  * communication with the client.
35  * </li>
36  * </ul>
37  * </li>
38  * <li>
39  * Examine the data received from client and transfer it to
40  * a <i>controller</i> ({@link project.pc.device.MouseController}
41  * or {@link project.pc.device.KeyboardController}) to execute the
42  * input operations.
43  * </li>
44  * </ul>
45  * </p>
46  * <p>
47  * Client authentication is done by comparing the pairing key received
48  * from the client with the one displayed on the pairing key dialog
49  * window.
50  * </p>
51  * <p>
52  * It receives data from client and wraps it in a {@link DataWrapper}
53  * object. It, then, invokes the {@link DataWrapper#getOperationType()}
54  * method to examine the type of operation requested by the client and
55  * accordingly transfers the control to either
56  * {@link project.pc.device.MouseController} or
57  * {@link project.pc.device.KeyboardController}.
58  * </p>
59  * <p>
60  * This thread is started by the {@link Server#listen()} upon accepting
61  * every client connection. It gets terminated implicitly after
62  * unsuccessful authentication of the client. Explicitly, it is terminated
63  * by the {@link NetworkManager} ({@link NetworkManager#stopServer()} and
64  * {@link NetworkManager#disconnect(InetAddress)}) by invoking the
65  * {@link #setStopFlag()} method at the time of server stop or
66  * disconnection of this client. A reference of this thread gets stored
67  * in the {@link NetworkState} after successful authentication of the
68  * client.
69  * </p>
70  *
71  * @see java.lang.Runnable
72  * @see project.pc.net.DataWrapper
73  * @see project.pc.net.DataWrapper#getOperationType()
74  * @see project.pc.device.MouseController
75  * @see project.pc.device.KeyboardController
76  * @see project.pc.net.NetworkManager#stopServer()
77  * @see project.pc.net.NetworkManager#disconnect(InetAddress)
78  */
79
80 public class ServerThread implements Runnable {
81
82     private Socket mClientSocket;
83     private MouseController mMouseController;
84     private KeyboardController mKeyboardController;
```

```

85     private BroadcastThread mBroadcastThread;
86     private Thread mDialogThread;
87     private JDialog mDialog;
88     private NetworkState mState;
89     private EKEProvider mEKEProvider;
90     private ClientInfo mClientInfo;
91
92     private boolean mStopFlag;
93     private String mPairingKey;
94
95     /**
96     * Constructor.
97     *
98     * Initializes the thread.
99     *
100    * @param state {@link NetworkState} of this network.
101    * @param skt client <code>Socket</code>.
102    * @param mc the {@link project.pc.device.MouseController}.
103    * @param kc the {@link project.pc.device.KeyboardController}.
104    * @param bt the {@link BroadcastThread}.
105    * @throws IOException
106    */
107    ServerThread(NetworkState state, Socket skt, MouseController mc,
KeyboardController kc, BroadcastThread bt)
108        throws IOException {
109        mClientSocket = skt;
110        mMouseController = mc;
111        mKeyboardController = kc;
112        mBroadcastThread = bt;
113        mStopFlag = false;
114        mState = state;
115        mDialogThread = null;
116        mDialog = null;
117    }
118
119    private boolean isValidPairingKey() throws IOException {
120        BufferedReader in = new BufferedReader(new
InputStreamReader(mClientSocket.getInputStream()));
121        String receivedKey = mEKEProvider.decryptString(in.readLine());
122        return mPairingKey.equals(receivedKey);
123    }
124
125    private boolean isValidClient() {
126        try {
127            BufferedReader in = new BufferedReader(new
InputStreamReader(mClientSocket.getInputStream()));
128            mClientInfo = new Gson().fromJson(in.readLine(), ClientInfo.class);
129            System.out.println(mClientInfo.getClientInfo() + " wants to connect.");
130
131            mPairingKey = EKEProvider.getPairingKey();
132            mEKEProvider = new EKEProvider(mPairingKey, mClientInfo.getPublicKey());
133
134            System.out.println("Type the following pairing key to connect your phone:
" + mPairingKey);
135            mDialogThread = null;
136            mDialogThread = new Thread(
137                () -> MainWindow.showPairingKeyDialog
138                (
139                    mClientInfo.getClientInfo(),
140                    mPairingKey,
141                    this
142                )
143            );
144            mDialogThread.start();
145            return isValidPairingKey();
146        } catch (IOException e) {
147            System.out.println("Exception in ServerThread:validateClient()");
148            e.printStackTrace();
149            return false;
150        }
151    }
152
153    private void closeConnection() {
154        try {
155            mState.remove(mClientInfo);
156            System.out.println(mClientSocket.getInetAddress() + " is now
disconnected!");
157            if (!mClientSocket.isClosed()) {
158                PrintWriter printWriter = new
PrintWriter(mClientSocket.getOutputStream(), true);
159                if (mEKEProvider != null) {
160                    printWriter.println(mEKEProvider.encryptString("Stop"));
161                }
162                mClientSocket.close();
163            }
164        } catch (IOException e) {System.out.println("Exception in
ServerThread:closeConnection()");}
165    }
166

```



```

167 private void doOperation() throws InterruptedException {
168     while(!mStopFlag) {
169         receiveData();
170     }
171 }
172
173 /**
174  * Performs operations for communication with a client.
175  * <ul>
176  * <li>Validates the client,</li>
177  * <li>
178  *     Receives data from client and encapsulates it in a
179  *     {@link DataWrapper} object, and
180  * </li>
181  * <li>
182  *     Examines the type of operation requested by the client and
183  *     accordingly transfers the control to a <i>controller</i>
184  *     ({@link project.pc.device.MouseController} or
185  *     {@link project.pc.device.KeyboardController}).
186  * </li>
187  * </ul>
188  *
189  * @see project.pc.net.DataWrapper
190  * @see project.pc.device.MouseController
191  * @see project.pc.device.KeyboardController
192  */
193 @Override
194 public void run() {
195     try {
196         PrintWriter out = new PrintWriter(mClientSocket.getOutputStream(), true);
197         if(!mBroadcastThread.getBroadcastFlag()) {
198             if (isValidClient()) {
199                 mClientInfo.setSocket(mClientSocket);
200                 out.println(mEKEProvider.encryptString("1"));
201                 mDialogThread = null;
202                 mDialogThread = new Thread(
203                     () -> MainWindow.showConnectionConfirmationDialog
204                         (
205                             mClientInfo.getClientInfo(),
206                             this
207                         )
208                 );
209                 mDialogThread.start();
210                 mClientSocket.setSoTimeout(100);
211                 mState.add(mClientInfo, this);
212                 doOperation();
213                 closeConnection();
214             }
215             else {
216                 mDialogThread = null;
217                 mDialogThread = new Thread(
218                     () -> MainWindow.showIncorrectPKeyDialog
219                         (
220                             mClientInfo.getClientInfo(),
221                             this
222                         )
223                 );
224                 mDialogThread.start();
225                 out.println(mEKEProvider.encryptString("0"));
226                 mClientSocket.close();
227             }
228         }
229         else {
230             if(!mClientSocket.isClosed()) mClientSocket.close();
231         }
232     }
233     catch (IOException|InterruptedException e) {
234         e.printStackTrace();
235     }
236 }
237
238 private void receiveData() throws InterruptedException {
239     try {
240         if(mClientSocket.isClosed()) return;
241         BufferedReader in = new BufferedReader(new
242             InputStreamReader(mClientSocket.getInputStream()));
243         if(!in.ready()) return;
244         String readData = mEKEProvider.decryptString(in.readLine());
245         DataWrapper dataWrapper = new Gson().fromJson(readData,
246             DataWrapper.class);
247         if(dataWrapper != null) {
248             switch (dataWrapper.getOperationType()) {
249                 case "Stop":
250                     setStopFlag();
251                     break;

```

```

254
255         case "Mouse_Move":
256             mMouseController.move(dataWrapper.getQuaternionObject(),
dataWrapper.isInitQuat(), dataWrapper.getSensitivity());
257             break;
258
259         case "Mouse_Touch":
260             mMouseController.move(dataWrapper.getX(), dataWrapper.getY(),
dataWrapper.getSensitivity());
261             break;
262
263         case "Mouse_Button":
264             mMouseController.doOperation(dataWrapper.getData());
265             break;
266
267         case "Key":
268             mKeyboardController.doKeyOperation(dataWrapper.getData());
269             break;
270
271         case "Special_Key":
272             mKeyboardController.doSpecialKeyOperation(dataWrapper.getData());
273     }
274 }
275 } catch (IOException e) {
276     e.printStackTrace();
277 }
278 }
279
280 /**
281  * Sets stop flag.
282  *
283  * Used to exit from the {@link #run()} method.
284  */
285 void setStopFlag() { mStopFlag = true; }
286
287 /**
288  * Sets a dialog window.
289  *
290  * @param d a {@link javax.swing.JDialog} object.
291  */
292 public void setDialog(JDialog d) { mDialog = d; }
293
294 /**
295  * Returns the {@link javax.swing.JDialog} object used for
296  * displaying notifications.
297  *
298  * @return a {@link javax.swing.JDialog} object.
299  */
300 public JDialog getDialog() {return mDialog;}
301
302 /**
303  * Returns the timeout of the client <code>Socket</code>.
304  *
305  * @return timeout of client <code>Socket</code> in milliseconds.
306  * @throws IOException
307  */
308 int getTimeout() throws IOException {
309     if(!mClientSocket.isClosed()) return mClientSocket.getSoTimeout();
310     return 0;
311 }
312 }

```

EKEProvider.java

```
1 package project.pc.security;
2
3 import org.apache.commons.codec.binary.Base64;
4 import org.bouncycastle.cert.X509CertificateHolder;
5 import org.bouncycastle.cert.X509v3CertificateBuilder;
6 import org.bouncycastle.cert.jcajce.JcaX509CertificateConverter;
7 import org.bouncycastle.cert.jcajce.JcaX509v3CertificateBuilder;
8 import org.bouncycastle.jce.ECNamedCurveTable;
9 import org.bouncycastle.jce.provider.BouncyCastleProvider;
10 import org.bouncycastle.jce.spec.ECNamedCurveParameterSpec;
11 import org.bouncycastle.operator.OperatorCreationException;
12 import org.bouncycastle.operator.jcajce.JcaContentSignerBuilder;
13
14 import javax.crypto.BadPaddingException;
15 import javax.crypto.Cipher;
16 import javax.crypto.IllegalBlockSizeException;
17 import javax.crypto.KeyAgreement;
18 import javax.crypto.NoSuchPaddingException;
19 import javax.crypto.SecretKey;
20 import javax.crypto.spec.IvParameterSpec;
21 import javax.crypto.spec.SecretKeySpec;
22 import javax.security.auth.x500.X500Principal;
23
24 import java.io.IOException;
25 import java.io.UnsupportedEncodingException;
26 import java.math.BigInteger;
27
28 import java.security.InvalidAlgorithmParameterException;
29 import java.security.InvalidKeyException;
30 import java.security.Key;
31 import java.security.KeyFactory;
32 import java.security.KeyPair;
33 import java.security.KeyPairGenerator;
34 import java.security.KeyStoreException;
35 import java.security.MessageDigest;
36 import java.security.NoSuchAlgorithmException;
37 import java.security.NoSuchProviderException;
38 import java.security.PrivateKey;
39 import java.security.PublicKey;
40 import java.security.SecureRandom;
41 import java.security.Security;
42 import java.security.UnrecoverableEntryException;
43
44 import java.security.cert.Certificate;
45 import java.security.cert.CertificateException;
46 import java.security.spec.InvalidKeySpecException;
47 import java.security.spec.X509EncodedKeySpec;
48
49 import java.util.Date;
50
51 /**
52  * Utility for the end-to-end security module.
53  *
54  * <p>
55  * This class provides the following features:
56  * <ul>
57  * <li>
58  * <i>256-bit Elliptic Curve</i> private and
59  * public key generation.
60  * </li>
61  * <li><i>AES 256-bit</i> plaintext encryption.</li>
62  * <li><i>AES 256-bit</i> cipher text decryption.</li>
63  * <li>Self-signed certificate generation.</li>
64  * </ul>
65  *
66  * The private-public key pair and the self-signed
67  * certificate is generated by the constructor
68  * during instantiation and is stored in a keystore
69  * ({@link java.security.KeyStore}) maintained by the
70  * {@link KeyStoreManager}. If a <code>KeyStore</code>
71  * already exists, the key pair and the certificate
72  * gets loaded directly. The encryption and
73  * decryption mechanism is provided by the
74  * {@link #encryptString(String)} and
75  * {@link #decryptString(String)} methods respectively.
76  * </p>
77  *
78  * @see project.pc.security.KeyStoreManager
79  * @see java.security.KeyStore
80  * @see java.security.KeyPair
81  * @see project.pc.security.EKEProvider#encryptString(String)
82  * @see project.pc.security.EKEProvider#decryptString(String)
83  */
84 public class EKEProvider {
```

```

85 |
86 |     private SecretKey mSecretKey;
87 |     private PublicKey mPublicKey;
88 |     private byte[] mIV;
89 |
90 |     private static final String SIGNATURE_ALGORITHM = "SHA256withECDSA";
91 |     private static final String MESSAGE_DIGEST = "SHA-256";
92 |     private static final String KEY_GENERATION_ALGORITHM = "EC";
93 |     private static final String KEY_AGREEMENT_ALGORITHM = "ECDH";
94 |     private static final String SECURITY_PROVIDER =
    BouncyCastleProvider.PROVIDER_NAME;
95 |     private static final String ECC_CURVE_NAME = "brainpoolp256r1";
96 |     private static final String ENCRYPTION_ALGORITHM = "AES";
97 |     private static final String CIPHER_MODE = "AES/GCM/NoPadding";
98 |
99 |     private static final Date NOT_BEFORE = new Date(System.currentTimeMillis() - 24 *
    60 * 60 * 1000);
100 |     private static final Date NOT_AFTER = new Date(System.currentTimeMillis() +
    63072000000L);
101 |
102 |     static {
103 |         Security.insertProviderAt(new BouncyCastleProvider(), 1);
104 |     }
105 |
106 |     /**
107 |     * Constructor.
108 |     *
109 |     * <p>
110 |     *     Initializes this <code>EKEProvider</code>. If keystore
111 |     *     exists, private and public {@link java.security.KeyPair}
112 |     *     is loaded, otherwise <code>KeyPair</code> is generated.
113 |     * </p>
114 |     *
115 |     * @see project.pc.security.KeyStoreManager
116 |     */
117 |     public EKEProvider() {
118 |         try {
119 |             KeyPair keyPair = null;
120 |             if (KeyStoreManager.keyStoreExists()) {
121 |                 keyPair = KeyStoreManager.getKSKeyPair();
122 |                 if (keyPair != null) mPublicKey = keyPair.getPublic();
123 |             }
124 |             if (keyPair == null) {
125 |                 keyPair = generateECKeys();
126 |                 if (keyPair != null) {
127 |                     mPublicKey = keyPair.getPublic();
128 |                     PrivateKey privateKey = keyPair.getPrivate();
129 |                     Certificate certificate = genSelfSignedCert(mPublicKey,
    privateKey);
130 |                     new KeyStoreManager().setMasterKey(privateKey, certificate);
131 |                 }
132 |             }
133 |         } catch (CertificateException | NoSuchAlgorithmException
134 |             | KeyStoreException | IOException | OperatorCreationException |
    UnrecoverableEntryException e) {
135 |             e.printStackTrace();
136 |         }
137 |     }
138 |
139 |     /**
140 |     * Constructor.
141 |     *
142 |     * <p>
143 |     *     Initializes this <code>EKEProvider</code>.
144 |     *     Generates a secret key using the <i>SHA-256
145 |     *     message digest</i> of the pairing key generated
146 |     *     by {@link EKEProvider#getPairingKey()} and the
147 |     *     public key of the client.
148 |     * </p>
149 |     *
150 |     * @param pairingKey Pairing key used for client-server
151 |     *     authentication.
152 |     * @param clientPublicKey Public key of the client.
153 |     * @see project.pc.security.KeyStoreManager
154 |     */
155 |     public EKEProvider(String pairingKey, byte[] clientPublicKey) {
156 |         MessageDigest msgDigest;
157 |         try {
158 |             msgDigest = MessageDigest.getInstance(MESSAGE_DIGEST, SECURITY_PROVIDER);
159 |             mIV = msgDigest.digest(pairingKey.getBytes());
160 |             KeyPair keyPair = KeyStoreManager.getKSKeyPair();
161 |             if (keyPair != null) {
162 |                 X509EncodedKeySpec pubKeySpec = new
    X509EncodedKeySpec(Base64.decodeBase64(clientPublicKey));
163 |                 KeyFactory keyFactory =
    KeyFactory.getInstance(KEY_GENERATION_ALGORITHM, SECURITY_PROVIDER);
164 |                 mSecretKey = generateSharedSecret(keyPair.getPrivate(),
    keyFactory.generatePublic(pubKeySpec));
165 |             }

```

```

166         } catch (NoSuchAlgorithmException | NoSuchProviderException
167         | CertificateException | KeyStoreException |
UnrecoverableEntryException
168         | IOException | InvalidKeySpecException e) {
169             e.printStackTrace();
170         }
171     }
172
173     private static String generatePairingKey() {
174         SecureRandom random = new SecureRandom();
175         StringBuilder stringBuilder = new StringBuilder(new BigInteger(36, 0,
random).toString(Character.MAX_RADIX));
176         while(stringBuilder.length() > 6) {
177             stringBuilder.deleteCharAt(random.nextInt(stringBuilder.length()));
178         }
179         for (int i = 0; i < stringBuilder.length(); i++) {
180             char ch = stringBuilder.charAt(i);
181             if (Character.isLetter(ch) && Character.isLowerCase(ch) &&
random.nextFloat() < 0.5) {
182                 stringBuilder.setCharAt(i, Character.toUpperCase(ch));
183             }
184         }
185         return stringBuilder.toString();
186     }
187
188     /**
189     * Generates and returns a randomly generated
190     * pairing key.
191     *
192     * @return the pairing key.
193     */
194     public static String getPairingKey() { return generatePairingKey(); }
195
196     /**
197     * Returns the <code>Base64</code> encoded server
198     * public key.
199     *
200     * @return the <code>Base64</code> encoded server
201     *         public key. <br/>
202     *         <b><i>Note:</i></b> Returns <code>null</code>
203     *         if no public has been generated during
204     *         instantiation.
205     */
206     public byte[] getBase64EncodedPubKey() {
207         if (mPublicKey != null) {
208             return Base64.encodeBase64(mPublicKey.getEncoded());
209         }
210         return null;
211     }
212
213     private KeyPair generateECKeys() {
214         try {
215             ECNamedCurveParameterSpec parameterSpec =
ECNamedCurveTable.getParameterSpec(ECC_CURVE_NAME);
216             KeyPairGenerator keyPairGenerator =
KeyPairGenerator.getInstance(KEY_GENERATION_ALGORITHM, SECURITY_PROVIDER);
217             keyPairGenerator.initialize(parameterSpec);
218             return keyPairGenerator.generateKeyPair();
219         } catch (NoSuchAlgorithmException | InvalidAlgorithmParameterException |
NoSuchProviderException e) {
220             e.printStackTrace();
221             return null;
222         }
223     }
224
225     private SecretKey generateSharedSecret/PrivateKey privateKey, PublicKey
publicKey) {
226         try {
227             KeyAgreement keyAgreement =
KeyAgreement.getInstance(KEY AGREEMENT_ALGORITHM, SECURITY_PROVIDER);
228             keyAgreement.init(privateKey);
229             keyAgreement.doPhase(publicKey, true);
230             return keyAgreement.generateSecret(ENCRYPTION_ALGORITHM);
231         } catch (InvalidKeyException | NoSuchAlgorithmException |
NoSuchProviderException e) {
232             e.printStackTrace();
233             return null;
234         }
235     }
236
237     /**
238     * Method to encrypt plaintext using <i>AES 256-bit</i>
239     * encryption algorithm.
240     *
241     * @param plainText plaintext to be encrypted.
242     * @return the cipher text.
243     * @see project.pc.security.EKEProvider#decryptString(String)
244     */
245     public String encryptString(String plainText) {

```

```

246         try {
247             Cipher cipher = Cipher.getInstance(CIPHER_MODE, SECURITY_PROVIDER);
248             cipher.init(Cipher.ENCRYPT_MODE, mSecretKey, new IvParameterSpec(mIV));
249             byte[] cipherText = cipher.doFinal(plainText.getBytes());
250             return new String(Base64.encodeBase64(cipherText));
251         } catch (NoSuchAlgorithmException | NoSuchProviderException
252                 | NoSuchPaddingException | InvalidKeyException
253                 | InvalidAlgorithmParameterException
254                 | IllegalBlockSizeException | BadPaddingException |
NullPointerException e) {
255             e.printStackTrace();
256             return null;
257         }
258     }
259
260     /**
261      * Method to decrypt cipher text using <i>AES-256 bit</i>
262      * decryption algorithm.
263      *
264      * @param cipherText cipher text to be decrypted.
265      * @return the plaintext.
266      * @see project.pc.security.EKEProvider#encryptString(String)
267      */
268     public String decryptString(String cipherText) {
269         try {
270             Key decryptionKey = new SecretKeySpec(mSecretKey.getEncoded(),
271                 mSecretKey.getAlgorithm());
272             IvParameterSpec ivSpec = new IvParameterSpec(mIV);
273             Cipher cipher = Cipher.getInstance(CIPHER_MODE, SECURITY_PROVIDER);
274             byte[] cipherTextBytes = Base64.decodeBase64(cipherText.getBytes());
275             cipher.init(Cipher.DECRYPT_MODE, decryptionKey, ivSpec);
276             return new String(cipher.doFinal(cipherTextBytes), "UTF-8");
277         } catch (NoSuchAlgorithmException | NoSuchProviderException
278                 | NoSuchPaddingException | InvalidKeyException
279                 | InvalidAlgorithmParameterException
280                 | IllegalBlockSizeException | BadPaddingException
281                 | UnsupportedEncodingException | NullPointerException e) {
282             return null;
283         }
284     }
285
286     private Certificate genSelfSignedCert(PublicKey publicKey, PrivateKey privateKey)
287         throws OperatorCreationException, CertificateException {
288         X500Principal issuer = new X500Principal("CN=127.0.0.1");
289         BigInteger serial = BigInteger.valueOf(System.currentTimeMillis());
290         X509v3CertificateBuilder certGen =
291             new JcaX509v3CertificateBuilder(issuer, serial, NOT_BEFORE,
NOT_AFTER,
292                 issuer, publicKey);
293         X509CertificateHolder certHolder = certGen.build(new
JcaContentSignerBuilder(SIGNATURE_ALGORITHM)
294             .setProvider(SECURITY_PROVIDER)
295             .build(privateKey));
296         return new JcaX509CertificateConverter()
297             .setProvider(SECURITY_PROVIDER)
298             .getCertificate(certHolder);
299     }
300 }

```


KeyStoreManager.java

```
1 package project.pc.security;
2
3 import java.io.File;
4 import java.io.FileInputStream;
5 import java.io.FileOutputStream;
6 import java.io.IOException;
7
8 import java.math.BigInteger;
9 import java.security.Key;
10 import java.security.KeyPair;
11 import java.security.KeyStore;
12 import java.security.KeyStoreException;
13 import java.security.NoSuchAlgorithmException;
14 import java.security.PrivateKey;
15 import java.security.PublicKey;
16 import java.security.SecureRandom;
17 import java.security.UnrecoverableEntryException;
18 import java.security.cert.Certificate;
19 import java.security.cert.CertificateException;
20 import java.util.prefs.Preferences;
21
22 /**
23  * Manages the operations on the <code>KeyStore</code>.
24  *
25  * <p>
26  * This class provides a {@link java.security.KeyStore} which stores :
27  * <ul>
28  * <li> The private key ({@link java.security.PrivateKey}).</li>
29  * <li>
30  * A self-signed certificate ({@link java.security.cert.Certificate})
31  * that contains the public key and the digital signature.
32  * </li>
33  * </ul>
34  * The <code>KeyStore</code> is protected by a randomly generated
35  * password.
36  * </p>
37  *
38  * @see project.pc.security.EKEProvider
39  * @see java.security.KeyStore
40  * @see java.security.KeyPair
41  * @see java.security.cert.Certificate
42  */
43 class KeyStoreManager {
44     private KeyStore mKeyStore;
45
46     private static final String KEY_STORE_TYPE = "pkcs12";
47     private static final String KEY_STORE_ALIAS = "Remouse KeyStore";
48     private static final String KEY_STORE_PASSWORD = generateKSPassword();
49     private static final String KEY_STORE_NAME = "remouse_keystore";
50
51     /**
52      * Constructor.
53      *
54      * Initializes this <code>KeyStoreManager</code>.
55      *
56      * @throws CertificateException
57      * @throws NoSuchAlgorithmException
58      * @throws KeyStoreException
59      * @throws IOException
60      */
61     KeyStoreManager() throws CertificateException, NoSuchAlgorithmException,
62     KeyStoreException, IOException {
63         mKeyStore = KeyStore.getInstance(KEY_STORE_TYPE);
64         mKeyStore.load(null, KEY_STORE_PASSWORD.toCharArray());
65     }
66
67     /**
68      * Method to store the private key and the certificate in the
69      * <code>KeyStore</code>.
70      *
71      * @param privateKey the {@link java.security.PrivateKey} object.
72      * @param cert the {@link java.security.cert.Certificate} object(s).
73      * @throws KeyStoreException
74      * @throws IOException
75      * @throws CertificateException
76      * @throws NoSuchAlgorithmException
77      */
78     void setMasterKey(PrivateKey privateKey, Certificate ...cert) throws
79     KeyStoreException, IOException, CertificateException, NoSuchAlgorithmException {
80         KeyStore.PrivateKeyEntry privateKeyEntry = new
81         KeyStore.PrivateKeyEntry(privateKey, cert);
82         mKeyStore.setEntry(KEY_STORE_ALIAS, privateKeyEntry, new
83         KeyStore.PasswordProtection(KEY_STORE_PASSWORD.toCharArray()));
```

```

81 |         mKeyStore.store(new FileOutputStream(KEY_STORE_NAME),
KEY_STORE_PASSWORD.toCharArray());
82 |     }
83 |
84 |     /**
85 |      * Returns the private-public key pair.
86 |      *
87 |      * @return {@link java.security.KeyPair} object.
88 |      * @throws IOException
89 |      * @throws KeyStoreException
90 |      * @throws CertificateException
91 |      * @throws NoSuchAlgorithmException
92 |      * @throws UnrecoverableEntryException
93 |      */
94 |     static KeyPair getKSKeyPair() throws IOException, KeyStoreException,
CertificateException, NoSuchAlgorithmException, UnrecoverableEntryException {
95 |         FileInputStream fis = new FileInputStream(KEY_STORE_NAME);
96 |
97 |         KeyStore keyStore = KeyStore.getInstance(KEY_STORE_TYPE);
98 |         keyStore.load(fis, KEY_STORE_PASSWORD.toCharArray());
99 |
100 |         Key key = keyStore.getKey(KEY_STORE_ALIAS, KEY_STORE_PASSWORD.toCharArray());
101 |         if (key instanceof PrivateKey) {
102 |             PublicKey publicKey =
keyStore.getCertificate(KEY_STORE_ALIAS).getPublicKey();
103 |             return new KeyPair(publicKey, (PrivateKey) key);
104 |         }
105 |         return null;
106 |     }
107 |
108 |     /**
109 |      * Checks if a <code>KeyStore</code> exists.
110 |      *
111 |      * @return <code>true</code>, if the <code>KeyStore</code> exists, <br/>
112 |      *         <code>false</code>, otherwise.
113 |      */
114 |     static boolean keyStoreExists() { return new File(KEY_STORE_NAME).exists(); }
115 |
116 |     private static String generateKSPassword() {
117 |         Preferences preferences =
Preferences.userNodeForPackage(KeyStoreManager.class);
118 |         String password = preferences.get("Password", null);
119 |         if (password == null) {
120 |             SecureRandom random = new SecureRandom();
121 |             StringBuilder stringBuilder = new StringBuilder(new BigInteger(36, 0,
random).toString(Character.MAX_RADIX));
122 |             while (stringBuilder.length() > 6) {
123 |                 stringBuilder.deleteCharAt(random.nextInt(stringBuilder.length()));
124 |             }
125 |             for (int i = 0; i < stringBuilder.length(); i++) {
126 |                 char ch = stringBuilder.charAt(i);
127 |                 if (Character.isLetter(ch) && Character.isLowerCase(ch) &&
random.nextFloat() < 0.5) {
128 |                     stringBuilder.setCharAt(i, Character.toUpperCase(ch));
129 |                 }
130 |             }
131 |             password = stringBuilder.toString();
132 |             preferences.put("Password", password);
133 |         }
134 |         return password;
135 |     }
136 | }

```


Cartesian2D.java

```
1 package project.pc.sensor.representation;
2
3 /**
4  * Class to represent a point in 2-D Cartesian coordinate system.
5  */
6 public class Cartesian2D {
7     private double x;
8     private double y;
9
10    /**
11     * Constructor.
12     *
13     * Initializes this <code>Cartesian2D</code> point.
14     * @param x the x-coordinate.
15     * @param y the y-coordinate.
16     */
17    public Cartesian2D(double x, double y) {
18        this.x = x;
19        this.y = y;
20    }
21
22    /**
23     * Returns the x-coordinate of this point.
24     *
25     * @return the x-coordinate.
26     */
27    public double getX() { return this.x; }
28
29    /**
30     * Returns the y-coordinate of this point.
31     *
32     * @return the y-coordinate.
33     */
34    public double getY() { return this.y; }
35
36    /**
37     * Sets the x-coordinate of this point.
38     *
39     * @param x the x-coordinate.
40     */
41    public void setX(double x) { this.x = x; }
42
43    /**
44     * Sets the y-coordinate of this point.
45     *
46     * @param y the y-coordinate.
47     */
48    public void setY(double y) { this.y = y; }
49 }
```

Matrix.java

```
1 package project.pc.sensor.representation;
2
3 /**
4  * Matrix math utilities.
5  *
6  * <p>
7  *     These methods operate on OpenGL ES format matrices and vectors
8  *     stored in float arrays. Matrices are 4 x 4 column-vector(float)
9  *     matrices stored in column-major order:
10  *
11  *     <pre>
12  *         m[offset + 0]    m[offset + 4]    m[offset + 8]    m[offset + 12]
13  *         m[offset + 1]    m[offset + 5]    m[offset + 9]    m[offset + 13]
14  *         m[offset + 2]    m[offset + 6]    m[offset + 10]   m[offset + 14]
15  *         m[offset + 3]    m[offset + 7]    m[offset + 11]   m[offset + 15]
16  *     </pre>
17  * </p>
18  * <p>
19  *     Vectors are 4 row x 1 column column-vectors stored in order:
20  *
21  *     <pre>
22  *         v[offset + 0]
23  *         v[offset + 1]
24  *         v[offset + 2]
25  *         v[offset + 3]
26  *     </pre>
27  * </p>
28  *
29  * @see project.pc.sensor.representation.MatrixF4x4
30  */
31 class Matrix {
32
33     private static final float[] TEMP_MATRIX_ARRAY = new float[32];
34
35     private static void multiplyMM(float[] output, int outputOffset, float[] lhs, int
lhsOffset, float[] rhs, int rhsOffset) {
36         output[outputOffset] = lhs[lhsOffset] * rhs[rhsOffset] + lhs[lhsOffset + 4] *
rhs[rhsOffset + 1] + lhs[lhsOffset + 8] * rhs[rhsOffset + 2] + lhs[lhsOffset + 12] *
rhs[rhsOffset + 3];
37         output[outputOffset + 1] = lhs[lhsOffset + 1] * rhs[rhsOffset] +
lhs[lhsOffset + 5] * rhs[rhsOffset + 1] + lhs[lhsOffset + 9] * rhs[rhsOffset + 2] +
lhs[lhsOffset + 13] * rhs[rhsOffset + 3];
38         output[outputOffset + 2] = lhs[lhsOffset + 2] * rhs[rhsOffset] +
lhs[lhsOffset + 6] * rhs[rhsOffset + 1] + lhs[lhsOffset + 10] * rhs[rhsOffset + 2] +
lhs[lhsOffset + 14] * rhs[rhsOffset + 3];
39         output[outputOffset + 3] = lhs[lhsOffset + 3] * rhs[rhsOffset] +
lhs[lhsOffset + 7] * rhs[rhsOffset + 1] + lhs[lhsOffset + 11] * rhs[rhsOffset + 2] +
lhs[lhsOffset + 15] * rhs[rhsOffset + 3];
40
41         output[outputOffset + 4] = lhs[lhsOffset] * rhs[rhsOffset + 4] +
lhs[lhsOffset + 4] * rhs[rhsOffset + 5] + lhs[lhsOffset + 8] * rhs[rhsOffset + 6] +
lhs[lhsOffset + 12] * rhs[rhsOffset + 7];
42         output[outputOffset + 5] = lhs[lhsOffset + 1] * rhs[rhsOffset + 4] +
lhs[lhsOffset + 5] * rhs[rhsOffset + 5] + lhs[lhsOffset + 9] * rhs[rhsOffset + 6] +
lhs[lhsOffset + 13] * rhs[rhsOffset + 7];
43         output[outputOffset + 6] = lhs[lhsOffset + 2] * rhs[rhsOffset + 4] +
lhs[lhsOffset + 6] * rhs[rhsOffset + 5] + lhs[lhsOffset + 10] * rhs[rhsOffset + 6] +
lhs[lhsOffset + 14] * rhs[rhsOffset + 7];
44         output[outputOffset + 7] = lhs[lhsOffset + 3] * rhs[rhsOffset + 4] +
lhs[lhsOffset + 7] * rhs[rhsOffset + 5] + lhs[lhsOffset + 11] * rhs[rhsOffset + 6] +
lhs[lhsOffset + 15] * rhs[rhsOffset + 7];
45
46         output[outputOffset + 8] = lhs[lhsOffset] * rhs[rhsOffset + 8] +
lhs[lhsOffset + 4] * rhs[rhsOffset + 9] + lhs[lhsOffset + 8] * rhs[rhsOffset + 10] +
lhs[lhsOffset + 12] * rhs[rhsOffset + 11];
47         output[outputOffset + 9] = lhs[lhsOffset + 1] * rhs[rhsOffset + 8] +
lhs[lhsOffset + 5] * rhs[rhsOffset + 9] + lhs[lhsOffset + 9] * rhs[rhsOffset + 10] +
lhs[lhsOffset + 13] * rhs[rhsOffset + 11];
48         output[outputOffset + 10] = lhs[lhsOffset + 2] * rhs[rhsOffset + 8] +
lhs[lhsOffset + 6] * rhs[rhsOffset + 9] + lhs[lhsOffset + 10] * rhs[rhsOffset + 10] +
lhs[lhsOffset + 14] * rhs[rhsOffset + 11];
49         output[outputOffset + 11] = lhs[lhsOffset + 3] * rhs[rhsOffset + 8] +
lhs[lhsOffset + 7] * rhs[rhsOffset + 9] + lhs[lhsOffset + 11] * rhs[rhsOffset + 10] +
lhs[lhsOffset + 15] * rhs[rhsOffset + 11];
50
51         output[outputOffset + 12] = lhs[lhsOffset] * rhs[rhsOffset + 12] +
lhs[lhsOffset + 4] * rhs[rhsOffset + 13] + lhs[lhsOffset + 8] * rhs[rhsOffset + 14] +
lhs[lhsOffset + 12] * rhs[rhsOffset + 15];
52         output[outputOffset + 13] = lhs[lhsOffset + 1] * rhs[rhsOffset + 12] +
lhs[lhsOffset + 5] * rhs[rhsOffset + 13] + lhs[lhsOffset + 9] * rhs[rhsOffset + 14] +
lhs[lhsOffset + 13] * rhs[rhsOffset + 15];
53         output[outputOffset + 14] = lhs[lhsOffset + 2] * rhs[rhsOffset + 12] +
lhs[lhsOffset + 6] * rhs[rhsOffset + 13] + lhs[lhsOffset + 10] * rhs[rhsOffset + 14] +
lhs[lhsOffset + 14] * rhs[rhsOffset + 15];
54     }
```



```

115 | inputV[2] + inputM[12] * inputV[3];
116 | outputV[1] = inputM[1] * inputV[0] + inputM[5] * inputV[1] + inputM[9] *
117 | inputV[2] + inputM[13] * inputV[3];
118 | outputV[2] = inputM[2] * inputV[0] + inputM[6] * inputV[1] + inputM[10] *
119 | inputV[2] + inputM[14] * inputV[3];
120 | outputV[3] = inputM[3] * inputV[0] + inputM[7] * inputV[1] + inputM[11] *
121 | inputV[2] + inputM[15] * inputV[3];
122 | }
123 |
124 | public static void multiplyMV3(float[] outputV, float[] inputM, float[] inputV,
125 | float w) {
126 |     outputV[0] = inputM[0] * inputV[0] + inputM[4] * inputV[1] + inputM[8] *
127 |     inputV[2] + inputM[12] * w;
128 |     outputV[1] = inputM[1] * inputV[0] + inputM[5] * inputV[1] + inputM[9] *
129 |     inputV[2] + inputM[13] * w;
130 |     outputV[2] = inputM[2] * inputV[0] + inputM[6] * inputV[1] + inputM[10] *
131 |     inputV[2] + inputM[14] * w;
132 | }
133 |
134 | /**
135 |  * Transposes a 4 x 4 matrix.
136 |  * @param mTrans the array that holds the output inverted matrix.
137 |  * @param mTransOffset an offset where the inverted matrix is stored.
138 |  * @param m the input array.
139 |  * @param mOffset an offset into <code>m</code> where the matrix is stored.
140 |  */
141 | public static void transposeM(float[] mTrans, int mTransOffset, float[] m, int
142 | mOffset) {
143 |     for (int i = 0; i < 4; i++) {
144 |         int mBase = i * 4 + mOffset;
145 |         mTrans[i + mTransOffset] = m[mBase];
146 |         mTrans[i + 4 + mTransOffset] = m[mBase + 1];
147 |         mTrans[i + 8 + mTransOffset] = m[mBase + 2];
148 |         mTrans[i + 12 + mTransOffset] = m[mBase + 3];
149 |     }
150 | }
151 |
152 | /**
153 |  * Inverts a 4 x 4 matrix.
154 |  * @param mInv the array that holds the output inverted matrix.
155 |  * @param mInvOffset an offset into <code>mInv</code> where the inverted
156 |  * matrix is stored.
157 |  * @param m the input array.
158 |  * @param mOffset an offset into <code>m</code> where the matrix is stored.
159 |  * @return <code>true</code>, if the matrix could be inverted, <br/>
160 |  * <code>false</code>, otherwise.
161 |  */
162 | public static boolean invertM(float[] mInv, int mInvOffset, float[] m, int
163 | mOffset) {
164 |     // Invert a 4 x 4 matrix using Cramer's Rule
165 |
166 |     // transpose matrix
167 |     final float src0 = m[mOffset];
168 |     final float src4 = m[mOffset + 1];
169 |     final float src8 = m[mOffset + 2];
170 |     final float src12 = m[mOffset + 3];
171 |
172 |     final float src1 = m[mOffset + 4];
173 |     final float src5 = m[mOffset + 5];
174 |     final float src9 = m[mOffset + 6];
175 |     final float src13 = m[mOffset + 7];
176 |
177 |     final float src2 = m[mOffset + 8];
178 |     final float src6 = m[mOffset + 9];
179 |     final float src10 = m[mOffset + 10];
180 |     final float src14 = m[mOffset + 11];
181 |
182 |     final float src3 = m[mOffset + 12];
183 |     final float src7 = m[mOffset + 13];
184 |     final float src11 = m[mOffset + 14];
185 |     final float src15 = m[mOffset + 15];
186 |
187 |     // calculate pairs for first 8 elements (cofactors)
188 |     final float atmp0 = src10 * src15;
189 |     final float atmp1 = src11 * src14;
190 |     final float atmp2 = src9 * src15;
191 |     final float atmp3 = src11 * src13;
192 |     final float atmp4 = src9 * src14;
193 |     final float atmp5 = src10 * src13;
194 |     final float atmp6 = src8 * src15;
195 |     final float atmp7 = src11 * src12;
196 |     final float atmp8 = src8 * src14;
197 |     final float atmp9 = src10 * src12;
198 |     final float atmp10 = src8 * src13;
199 |     final float atmp11 = src9 * src12;
200 |
201 |     // calculate first 8 elements (cofactors)

```

```

194 |     final float dst0 = (atmp0 * src5 + atmp3 * src6 + atmp4 * src7) - (atmp1 *
src5 + atmp2 * src6 + atmp5 * src7);
195 |     final float dst1 = (atmp1 * src4 + atmp6 * src6 + atmp9 * src7) - (atmp0 *
src4 + atmp7 * src6 + atmp8 * src7);
196 |     final float dst2 = (atmp2 * src4 + atmp7 * src5 + atmp10 * src7)
- (atmp3 * src4 + atmp6 * src5 + atmp11 * src7);
197 |     final float dst3 = (atmp5 * src4 + atmp8 * src5 + atmp11 * src6)
- (atmp4 * src4 + atmp9 * src5 + atmp10 * src6);
198 |     final float dst4 = (atmp1 * src1 + atmp2 * src2 + atmp5 * src3) - (atmp0 *
src1 + atmp3 * src2 + atmp4 * src3);
199 |     final float dst5 = (atmp0 * src0 + atmp7 * src2 + atmp8 * src3) - (atmp1 *
src0 + atmp6 * src2 + atmp9 * src3);
200 |     final float dst6 = (atmp3 * src0 + atmp6 * src1 + atmp11 * src3)
- (atmp2 * src0 + atmp7 * src1 + atmp10 * src3);
201 |     final float dst7 = (atmp4 * src0 + atmp9 * src1 + atmp10 * src2)
- (atmp5 * src0 + atmp8 * src1 + atmp11 * src2);
202 |
203 | // calculate pairs for second 8 elements (cofactors)
204 | final float btmp0 = src2 * src7;
205 | final float btmp1 = src3 * src6;
206 | final float btmp2 = src1 * src7;
207 | final float btmp3 = src3 * src5;
208 | final float btmp4 = src1 * src6;
209 | final float btmp5 = src2 * src5;
210 | final float btmp6 = src0 * src7;
211 | final float btmp7 = src3 * src4;
212 | final float btmp8 = src0 * src6;
213 | final float btmp9 = src2 * src4;
214 | final float btmp10 = src0 * src5;
215 | final float btmp11 = src1 * src4;
216 |
217 | // calculate second 8 elements (cofactors)
218 | final float dst8 = (btmp0 * src13 + btmp3 * src14 + btmp4 * src15)
- (btmp1 * src13 + btmp2 * src14 + btmp5 * src15);
219 | final float dst9 = (btmp1 * src12 + btmp6 * src14 + btmp9 * src15)
- (btmp0 * src12 + btmp7 * src14 + btmp8 * src15);
220 | final float dst10 = (btmp2 * src12 + btmp7 * src13 + btmp10 * src15)
- (btmp3 * src12 + btmp6 * src13 + btmp11 * src15);
221 | final float dst11 = (btmp5 * src12 + btmp8 * src13 + btmp11 * src14)
- (btmp4 * src12 + btmp9 * src13 + btmp10 * src14);
222 | final float dst12 = (btmp2 * src10 + btmp5 * src11 + btmp1 * src9)
- (btmp4 * src11 + btmp0 * src9 + btmp3 * src10);
223 | final float dst13 = (btmp8 * src11 + btmp0 * src8 + btmp7 * src10)
- (btmp6 * src10 + btmp9 * src11 + btmp1 * src8);
224 | final float dst14 = (btmp6 * src9 + btmp11 * src11 + btmp3 * src8)
- (btmp10 * src11 + btmp2 * src8 + btmp7 * src9);
225 | final float dst15 = (btmp10 * src10 + btmp4 * src8 + btmp9 * src9)
- (btmp8 * src9 + btmp11 * src10 + btmp5 * src8);
226 |
227 | // calculate determinant
228 | final float det = src0 * dst0 + src1 * dst1 + src2 * dst2 + src3 * dst3;
229 |
230 | if (det == 0.0f) {
231 |     return false;
232 | }
233 |
234 | // calculate matrix inverse
235 | final float invdet = 1.0f / det;
236 | mInv[mInvOffset] = dst0 * invdet;
237 | mInv[1 + mInvOffset] = dst1 * invdet;
238 | mInv[2 + mInvOffset] = dst2 * invdet;
239 | mInv[3 + mInvOffset] = dst3 * invdet;
240 |
241 | mInv[4 + mInvOffset] = dst4 * invdet;
242 | mInv[5 + mInvOffset] = dst5 * invdet;
243 | mInv[6 + mInvOffset] = dst6 * invdet;
244 | mInv[7 + mInvOffset] = dst7 * invdet;
245 |
246 | mInv[8 + mInvOffset] = dst8 * invdet;
247 | mInv[9 + mInvOffset] = dst9 * invdet;
248 | mInv[10 + mInvOffset] = dst10 * invdet;
249 | mInv[11 + mInvOffset] = dst11 * invdet;
250 |
251 | mInv[12 + mInvOffset] = dst12 * invdet;
252 | mInv[13 + mInvOffset] = dst13 * invdet;
253 | mInv[14 + mInvOffset] = dst14 * invdet;
254 | mInv[15 + mInvOffset] = dst15 * invdet;
255 |
256 | return true;
257 | }
258 |
259 | /**
260 |  * Computes an orthographic projection matrix.
261 |  *
262 |  * @param m returns the result.
263 |  * @param mOffset offset value.
264 |  * @param left left clipping value.
265 |  * @param right right clipping value.
266 |  * @param bottom bottom clipping value.
267 |  */

```

```

279     * @param top top clipping value.
280     * @param near nearer depth clipping value.
281     * @param far farther depth clipping value.
282     */
283     public static void orthoM(float[] m, int mOffset, float left, float right, float
bottom, float top, float near, float far) {
284         if (left == right) {
285             throw new IllegalArgumentException("left == right");
286         }
287         if (bottom == top) {
288             throw new IllegalArgumentException("bottom == top");
289         }
290         if (near == far) {
291             throw new IllegalArgumentException("near == far");
292         }
293
294         final float r_width = 1.0f / (right - left);
295         final float r_height = 1.0f / (top - bottom);
296         final float r_depth = 1.0f / (far - near);
297         final float x = 2.0f * (r_width);
298         final float y = 2.0f * (r_height);
299         final float z = -2.0f * (r_depth);
300         final float tx = -(right + left) * r_width;
301         final float ty = -(top + bottom) * r_height;
302         final float tz = -(far + near) * r_depth;
303         m[mOffset] = x;
304         m[mOffset + 5] = y;
305         m[mOffset + 10] = z;
306         m[mOffset + 12] = tx;
307         m[mOffset + 13] = ty;
308         m[mOffset + 14] = tz;
309         m[mOffset + 15] = 1.0f;
310         m[mOffset + 1] = 0.0f;
311         m[mOffset + 2] = 0.0f;
312         m[mOffset + 3] = 0.0f;
313         m[mOffset + 4] = 0.0f;
314         m[mOffset + 6] = 0.0f;
315         m[mOffset + 7] = 0.0f;
316         m[mOffset + 8] = 0.0f;
317         m[mOffset + 9] = 0.0f;
318         m[mOffset + 11] = 0.0f;
319     }
320
321     /**
322     * Defines a projection matrix in terms of six clip planes.
323     *
324     * @param m the float array that holds the perspective matrix.
325     * @param offset the offset into <code>m</code> where the perspective
326     * matrix data is written.
327     * @param left left clipping value.
328     * @param right right clipping value.
329     * @param bottom bottom clipping value.
330     * @param top top clipping value.
331     * @param near nearer depth clipping value.
332     * @param far farther depth clipping value.
333     */
334     public static void frustumM(float[] m, int offset, float left, float right, float
bottom, float top, float near, float far) {
335         if (left == right) {
336             throw new IllegalArgumentException("left == right");
337         }
338         if (top == bottom) {
339             throw new IllegalArgumentException("top == bottom");
340         }
341         if (near == far) {
342             throw new IllegalArgumentException("near == far");
343         }
344         if (near <= 0.0f) {
345             throw new IllegalArgumentException("near <= 0.0f");
346         }
347         if (far <= 0.0f) {
348             throw new IllegalArgumentException("far <= 0.0f");
349         }
350
351         final float r_width = 1.0f / (right - left);
352         final float r_height = 1.0f / (top - bottom);
353         final float r_depth = 1.0f / (near - far);
354         final float x = 2.0f * (near * r_width);
355         final float y = 2.0f * (near * r_height);
356         final float A = 2.0f * ((right + left) * r_width);
357         final float B = (top + bottom) * r_height;
358         final float C = (far + near) * r_depth;
359         final float D = 2.0f * (far * near * r_depth);
360         m[offset] = x;
361         m[offset + 5] = y;
362         m[offset + 8] = A;
363         m[offset + 9] = B;
364         m[offset + 10] = C;
365         m[offset + 14] = D;

```

```

366         m[offset + 11] = -1.0f;
367         m[offset + 1] = 0.0f;
368         m[offset + 2] = 0.0f;
369         m[offset + 3] = 0.0f;
370         m[offset + 4] = 0.0f;
371         m[offset + 6] = 0.0f;
372         m[offset + 7] = 0.0f;
373         m[offset + 12] = 0.0f;
374         m[offset + 13] = 0.0f;
375         m[offset + 15] = 0.0f;
376     }
377
378     /**
379     * Defines a projection matrix in terms of a field of view angle, an
380     * aspect ratio, and Z clip planes.
381     *
382     * @param m the float array that holds the perspective matrix.
383     * @param offset the offset into <code>m</code> where the perspective matrix
384     *         data is written.
385     * @param fovy field of view in y direction, in degrees.
386     * @param aspect width to height aspect ratio of the viewport.
387     * @param zNear nearer depth clipping value on Z-plane.
388     * @param zFar farther depth clipping value on Z-plane.
389     */
390     /*
391     public static void perspectiveM(float[] m, int offset, float fovy, float aspect,
float zNear, float zFar) {
392         float f = 1.0f / (float) Math.tan(fovy * (Math.PI / 360.0));
393         float rangeReciprocal = 1.0f / (zNear - zFar);
394
395         m[offset] = f / aspect;
396         m[offset + 1] = 0.0f;
397         m[offset + 2] = 0.0f;
398         m[offset + 3] = 0.0f;
399
400         m[offset + 4] = 0.0f;
401         m[offset + 5] = f;
402         m[offset + 6] = 0.0f;
403         m[offset + 7] = 0.0f;
404
405         m[offset + 8] = 0.0f;
406         m[offset + 9] = 0.0f;
407         m[offset + 10] = (zFar + zNear) * rangeReciprocal;
408         m[offset + 11] = -1.0f;
409
410         m[offset + 12] = 0.0f;
411         m[offset + 13] = 0.0f;
412         m[offset + 14] = 2.0f * zFar * zNear * rangeReciprocal;
413         m[offset + 15] = 0.0f;
414     }
415
416     private static float length(float x, float y, float z) {
417         return (float) Math.sqrt(x * x + y * y + z * z);
418     }
419
420     /**
421     * Sets matrix to the identity matrix.
422     *
423     * @param sm the result.
424     * @param smOffset index into <code>sm</code> where the result matrix starts.
425     */
426     /*
427     static void setIdentityM(float[] sm, int smOffset) {
428         for (int i = 0; i < 16; i++) {
429             sm[smOffset + i] = 0;
430         }
431         for (int i = 0; i < 16; i += 5) {
432             sm[smOffset + i] = 1.0f;
433         }
434     }
435
436     /**
437     * Scales matrix <code>m</code> by <i>x</i>, <i>y</i>, and <i>z</i>.
438     *
439     * @param sm the result.
440     * @param smOffset index into <code>sm</code> where the result matrix starts.
441     * @param m source matrix.
442     * @param mOffset index into <code>m</code> where the source matrix starts.
443     * @param x scale factor x.
444     * @param y scale factor y.
445     * @param z scale factor z.
446     */
447     /*
448     public static void scaleM(float[] sm, int smOffset, float[] m, int mOffset, float
x, float y, float z) {
449         for (int i = 0; i < 4; i++) {
450             int smi = smOffset + i;
451             int mi = mOffset + i;
452             sm[smi] = m[mi] * x;
453             sm[4 + smi] = m[4 + mi] * y;
454             sm[8 + smi] = m[8 + mi] * z;

```



```

453         sm[12 + smi] = m[12 + mi];
454     }
455 }
456
457 public static void scaleM(float[] m, int mOffset, float x, float y, float z) {
458     for (int i = 0; i < 4; i++) {
459         int mi = mOffset + i;
460         m[mi] *= x;
461         m[4 + mi] *= y;
462         m[8 + mi] *= z;
463     }
464 }
465
466 /**
467  * Translates matrix <code>m</code> by <i>x</i>, <i>y</i>, and <i>z</i>.
468  *
469  * @param tm the result.
470  * @param tmOffset index into <code>sm</code> where the result matrix starts.
471  * @param m source matrix.
472  * @param mOffset index into <code>m</code> where the source matrix starts.
473  * @param x translation factor x.
474  * @param y translation factor y.
475  * @param z translation factor z.
476  */
477 public static void translateM(float[] tm, int tmOffset, float[] m, int mOffset,
float x, float y, float z) {
478     System.arraycopy(m, mOffset, tm, tmOffset, 12);
479     for (int i = 0; i < 4; i++) {
480         int tmi = tmOffset + i;
481         int mi = mOffset + i;
482         tm[12 + tmi] = m[mi] * x + m[4 + mi] * y + m[8 + mi] * z + m[12 + mi];
483     }
484 }
485
486 private static void translateM(float[] m, int mOffset, float x, float y, float z)
{
487     for (int i = 0; i < 4; i++) {
488         int mi = mOffset + i;
489         m[12 + mi] += m[mi] * x + m[4 + mi] * y + m[8 + mi] * z;
490     }
491 }
492
493 /**
494  * Rotates matrix m by angle a (in degrees) around the axis (x, y, z).
495  *
496  * @param rm returns the result.
497  * @param rmOffset index into rm where the result matrix starts.
498  * @param m source matrix.
499  * @param mOffset index into m where the source matrix starts.
500  * @param a angle to rotate in degrees.
501  * @param x scale factor x.
502  * @param y scale factor y.
503  * @param z scale factor z.
504  */
505 public static void rotateM(float[] rm, int rmOffset, float[] m, int mOffset,
float a, float x, float y, float z) {
506     synchronized (TEMP_MATRIX_ARRAY) {
507         setRotateM(TEMP_MATRIX_ARRAY, 0, a, x, y, z);
508         multiplyMM(rm, rmOffset, m, mOffset, TEMP_MATRIX_ARRAY, 0);
509     }
510 }
511
512 /**
513  * Rotates matrix m in place by angle a (in degrees)
514  * around the axis (x, y, z).
515  *
516  * @param m source matrix.
517  * @param mOffset index into m where the matrix starts.
518  * @param a angle to rotate in degrees.
519  * @param x scale factor x.
520  * @param y scale factor y.
521  * @param z scale factor z.
522  */
523 public static void rotateM(float[] m, int mOffset, float a, float x, float y,
float z) {
524     synchronized (TEMP_MATRIX_ARRAY) {
525         setRotateM(TEMP_MATRIX_ARRAY, 0, a, x, y, z);
526         multiplyMM(TEMP_MATRIX_ARRAY, 16, m, mOffset, TEMP_MATRIX_ARRAY, 0);
527         System.arraycopy(TEMP_MATRIX_ARRAY, 16, m, mOffset, 16);
528     }
529 }
530
531 private static void setRotateM(float[] rm, int rmOffset, float a, float x, float
y, float z) {
532     rm[rmOffset + 3] = 0;
533     rm[rmOffset + 7] = 0;
534     rm[rmOffset + 11] = 0;
535     rm[rmOffset + 12] = 0;
536     rm[rmOffset + 13] = 0;

```



```

537     rm[rmOffset + 14] = 0;
538     rm[rmOffset + 15] = 1;
539     a *= (float) (Math.PI / 180.0f);
540     float s = (float) Math.sin(a);
541     float c = (float) Math.cos(a);
542     if (1.0f == x && 0.0f == y && 0.0f == z) {
543         rm[rmOffset + 5] = c;
544         rm[rmOffset + 10] = c;
545         rm[rmOffset + 6] = s;
546         rm[rmOffset + 9] = -s;
547         rm[rmOffset + 1] = 0;
548         rm[rmOffset + 2] = 0;
549         rm[rmOffset + 4] = 0;
550         rm[rmOffset + 8] = 0;
551         rm[rmOffset] = 1;
552     } else if (0.0f == x && 1.0f == y && 0.0f == z) {
553         rm[rmOffset] = c;
554         rm[rmOffset + 10] = c;
555         rm[rmOffset + 8] = s;
556         rm[rmOffset + 2] = -s;
557         rm[rmOffset + 1] = 0;
558         rm[rmOffset + 4] = 0;
559         rm[rmOffset + 6] = 0;
560         rm[rmOffset + 9] = 0;
561         rm[rmOffset + 5] = 1;
562     } else if (0.0f == x && 0.0f == y && 1.0f == z) {
563         rm[rmOffset] = c;
564         rm[rmOffset + 5] = c;
565         rm[rmOffset + 1] = s;
566         rm[rmOffset + 4] = -s;
567         rm[rmOffset + 2] = 0;
568         rm[rmOffset + 6] = 0;
569         rm[rmOffset + 8] = 0;
570         rm[rmOffset + 9] = 0;
571         rm[rmOffset + 10] = 1;
572     } else {
573         float len = length(x, y, z);
574         if (1.0f != len) {
575             float recipLen = 1.0f / len;
576             x *= recipLen;
577             y *= recipLen;
578             z *= recipLen;
579         }
580         float nc = 1.0f - c;
581         float xy = x * y;
582         float yz = y * z;
583         float zx = z * x;
584         float xs = x * s;
585         float ys = y * s;
586         float zs = z * s;
587         rm[rmOffset] = x * x * nc + c;
588         rm[rmOffset + 4] = xy * nc - zs;
589         rm[rmOffset + 8] = zx * nc + ys;
590         rm[rmOffset + 1] = xy * nc + zs;
591         rm[rmOffset + 5] = y * y * nc + c;
592         rm[rmOffset + 9] = yz * nc - xs;
593         rm[rmOffset + 2] = zx * nc - ys;
594         rm[rmOffset + 6] = yz * nc + xs;
595         rm[rmOffset + 10] = z * z * nc + c;
596     }
597 }
598
599 /**
600  * Converts Euler angles to a rotation matrix.
601  *
602  * @param rm the result.
603  * @param rmOffset index into <code>rm</code> where the result matrix starts.
604  * @param x angle of rotation, in degrees.
605  * @param y angle of rotation, in degrees.
606  * @param z angle of rotation, in degrees.
607  */
608 public static void setRotateEulerM(float[] rm, int rmOffset, float x, float y,
float z) {
609     x *= (float) (Math.PI / 180.0f);
610     y *= (float) (Math.PI / 180.0f);
611     z *= (float) (Math.PI / 180.0f);
612     float cx = (float) Math.cos(x);
613     float sx = (float) Math.sin(x);
614     float cy = (float) Math.cos(y);
615     float sy = (float) Math.sin(y);
616     float cz = (float) Math.cos(z);
617     float sz = (float) Math.sin(z);
618     float cxsy = cx * sy;
619     float xsy = sx * sy;
620
621     rm[rmOffset] = cy * cz;
622     rm[rmOffset + 1] = -cy * sz;
623     rm[rmOffset + 2] = sy;
624     rm[rmOffset + 3] = 0.0f;

```

```

625         rm[rmOffset + 4] = cxsy * cz + cx * sz;
626         rm[rmOffset + 5] = -cxsy * sz + cx * cz;
627         rm[rmOffset + 6] = -sx * cy;
628         rm[rmOffset + 7] = 0.0f;
629
630         rm[rmOffset + 8] = -sxsy * cz + sx * sz;
631         rm[rmOffset + 9] = sxsy * sz + sx * cz;
632         rm[rmOffset + 10] = cx * cy;
633         rm[rmOffset + 11] = 0.0f;
634
635         rm[rmOffset + 12] = 0.0f;
636         rm[rmOffset + 13] = 0.0f;
637         rm[rmOffset + 14] = 0.0f;
638         rm[rmOffset + 15] = 1.0f;
639     }
640 }
641
642     public static void setLookAtM(float[] rm, int rmOffset, float eyeX, float eyeY,
643     float eyeZ, float centerX,
644     float centerY, float centerZ, float upX, float upY,
645     float upZ) {
646         float fx = centerX - eyeX;
647         float fy = centerY - eyeY;
648         float fz = centerZ - eyeZ;
649
650         // Normalize f
651         float rlf = 1.0f / Matrix.length(fx, fy, fz);
652         fx *= rlf;
653         fy *= rlf;
654         fz *= rlf;
655
656         // compute s = f x up (x means "cross product")
657         float sx = fy * upZ - fz * upY;
658         float sy = fz * upX - fx * upZ;
659         float sz = fx * upY - fy * upX;
660
661         // and normalize s
662         float rls = 1.0f / Matrix.length(sx, sy, sz);
663         sx *= rls;
664         sy *= rls;
665         sz *= rls;
666
667         // compute u = s x f
668         float ux = sy * fz - sz * fy;
669         float uy = sz * fx - sx * fz;
670         float uz = sx * fy - sy * fx;
671
672         rm[rmOffset] = sx;
673         rm[rmOffset + 1] = ux;
674         rm[rmOffset + 2] = -fx;
675         rm[rmOffset + 3] = 0.0f;
676
677         rm[rmOffset + 4] = sy;
678         rm[rmOffset + 5] = uy;
679         rm[rmOffset + 6] = -fy;
680         rm[rmOffset + 7] = 0.0f;
681
682         rm[rmOffset + 8] = sz;
683         rm[rmOffset + 9] = uz;
684         rm[rmOffset + 10] = -fz;
685         rm[rmOffset + 11] = 0.0f;
686
687         rm[rmOffset + 12] = 0.0f;
688         rm[rmOffset + 13] = 0.0f;
689         rm[rmOffset + 14] = 0.0f;
690         rm[rmOffset + 15] = 1.0f;
691
692         translateM(rm, rmOffset, -eyeX, -eyeY, -eyeZ);
693     }

```

MatrixF4x4.java

[illegible]

```

87     boolean isColumnMajor() {
88         return mColMaj;
89     }
90
91     /**
92     * Multiplies the given vector by this <code>MatrixF4x4</code>.
93     *
94     * This should only be used if the matrix is of size 16.
95     *
96     * @param vector a {@link Vector4f}.
97     * @throws InvalidParameterSpecException
98     */
99     public void multiplyVector4fByMatrix(Vector4f vector) throws
InvalidParameterSpecException {
100
101         if (matrix.length == 16) {
102             float x = 0;
103             float y = 0;
104             float z = 0;
105             float w = 0;
106
107             float[] vectorArray = vector.array();
108
109             if (mColMaj) {
110                 for (int i = 0; i < 4; i++) {
111
112                     int k = i * 4;
113
114                     x += this.matrix[k] * vectorArray[i];
115                     y += this.matrix[k + 1] * vectorArray[i];
116                     z += this.matrix[k + 2] * vectorArray[i];
117                     w += this.matrix[k + 3] * vectorArray[i];
118                 }
119             } else {
120                 for (int i = 0; i < 4; i++) {
121
122                     x += this.matrix[i] * vectorArray[i];
123                     y += this.matrix[4 + i] * vectorArray[i];
124                     z += this.matrix[8 + i] * vectorArray[i];
125                     w += this.matrix[12 + i] * vectorArray[i];
126                 }
127             }
128
129             vector.setX(x);
130             vector.setY(y);
131             vector.setZ(z);
132             vector.setW(w);
133         } else {
134             throw new InvalidParameterSpecException("Expects the matrix to be of size 16");
135         }
136     }
137
138     /**
139     * Multiplies the given vector by this <code>MatrixF4x4</code>.
140     *
141     * This should only be used if the matrix is of size 9.
142     *
143     * @param vector a {@link Vector3f}.
144     * @throws InvalidParameterSpecException
145     */
146     public void multiplyVector3fByMatrix(Vector3f vector) throws
InvalidParameterSpecException {
147
148         if (matrix.length == 9) {
149             float x = 0;
150             float y = 0;
151             float z = 0;
152
153             float[] vectorArray = vector.toArray();
154
155             if (!mColMaj) {
156                 for (int i = 0; i < 3; i++) {
157
158                     int k = i * 3;
159
160                     x += this.matrix[k] * vectorArray[i];
161                     y += this.matrix[k + 1] * vectorArray[i];
162                     z += this.matrix[k + 2] * vectorArray[i];
163                 }
164             } else {
165                 for (int i = 0; i < 3; i++) {
166
167                     x += this.matrix[i] * vectorArray[i];
168                     y += this.matrix[3 + i] * vectorArray[i];
169                     z += this.matrix[6 + i] * vectorArray[i];
170                 }
171             }
172
173             vector.setX(x);
174             vector.setY(y);
175             vector.setZ(z);
176         } else {
177             throw new InvalidParameterSpecException("Expects the matrix to be of size 9");
178         }
179     }

```

```

180
181 /**
182  * Multiplies a <code>MatrixF4x4</code> with this.
183  *
184  * @param matrixf a <code>MatrixF4x4</code>.
185  */
186 public void multiplyMatrix4x4ByMatrix(MatrixF4x4 matrixf) {
187     float[] bufferMatrix = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
188     float[] matrix = matrixf.getMatrix();
189     multiplyMatrix(matrix, 0, bufferMatrix, 0);
190     matrixf.setMatrix(bufferMatrix);
191 }
192
193
194 private void multiplyMatrix(float[] input, int inputOffset, float[] output, int
outputOffset) {
195     for (int i = 0; i < 4; i++) {
196         for (int j = 0; j < 4; j++) {
197             int k = i * 4;
198             output[outputOffset + j] += this.matrix[k + j] * input[inputOffset + i];
199             output[outputOffset + 4 + j] += this.matrix[k + j] * input[inputOffset + 4 +
200 i];
201             output[outputOffset + 2 * 4 + j] += this.matrix[k + j] * input[inputOffset +
202 2 * 4 + i];
203             output[outputOffset + 3 * 4 + j] += this.matrix[k + j] * input[inputOffset +
204 3 * 4 + i];
205         }
206     }
207 }
208
209 public void transpose() {
210     if (this.matrix.length == 16) {
211         float[] newMatrix = new float[16];
212         for (int i = 0; i < 4; i++) {
213             int k = i * 4;
214             newMatrix[k] = matrix[i];
215             newMatrix[k + 1] = matrix[4 + i];
216             newMatrix[k + 2] = matrix[8 + i];
217             newMatrix[k + 3] = matrix[12 + i];
218         }
219         matrix = newMatrix;
220     } else {
221         float[] newMatrix = new float[9];
222         for (int i = 0; i < 3; i++) {
223             int k = i * 3;
224             newMatrix[k] = matrix[i];
225             newMatrix[k + 1] = matrix[3 + i];
226             newMatrix[k + 2] = matrix[6 + i];
227         }
228         matrix = newMatrix;
229     }
230 }
231
232 /**
233  * Sets the value of X0.
234  *
235  * @param value value to set.
236  */
237 void setX0(float value) {
238     if (matrix.length == 16) {
239         if (mColMaj) {
240             matrix[MAT_IND_COL16_3X3[0]] = value;
241         } else {
242             matrix[MAT_IND_ROW16_3X3[0]] = value;
243         }
244     } else {
245         if (mColMaj) {
246             matrix[MAT_IND_COL9_3X3[0]] = value;
247         } else {
248             matrix[MAT_IND_ROW9_3X3[0]] = value;
249         }
250     }
251 }
252
253
254 /**
255  * Sets the value of X1.
256  *
257  * @param value value to set.
258  */
259 void setX1(float value) {
260     if (matrix.length == 16) {
261         if (mColMaj) {
262             matrix[MAT_IND_COL16_3X3[1]] = value;
263         } else {
264             matrix[MAT_IND_ROW16_3X3[1]] = value;
265         }
266     } else {
267         if (mColMaj) {
268             matrix[MAT_IND_COL9_3X3[1]] = value;
269         } else {
270

```

```

271         matrix[MAT_IND_ROW9_3X3[1]] = value;
272     }
273 }
274
275 /**
276  * Sets the value of X2.
277  * @param value value to set.
278  */
279 void setX2(float value) {
280
281     if (matrix.length == 16) {
282         if (mColMaj) {
283             matrix[MAT_IND_COL16_3X3[2]] = value;
284         } else {
285             matrix[MAT_IND_ROW16_3X3[2]] = value;
286         }
287     } else {
288         if (mColMaj) {
289             matrix[MAT_IND_COL9_3X3[2]] = value;
290         } else {
291             matrix[MAT_IND_ROW9_3X3[2]] = value;
292         }
293     }
294 }
295
296 /**
297  * Sets the value of Y0.
298  * @param value value to set.
299  */
300 void setY0(float value) {
301
302     if (matrix.length == 16) {
303         if (mColMaj) {
304             matrix[MAT_IND_COL16_3X3[3]] = value;
305         } else {
306             matrix[MAT_IND_ROW16_3X3[3]] = value;
307         }
308     } else {
309         if (mColMaj) {
310             matrix[MAT_IND_COL9_3X3[3]] = value;
311         } else {
312             matrix[MAT_IND_ROW9_3X3[3]] = value;
313         }
314     }
315 }
316
317 /**
318  * Sets the value of Y1.
319  * @param value value to set.
320  */
321 void setY1(float value) {
322
323     if (matrix.length == 16) {
324         if (mColMaj) {
325             matrix[MAT_IND_COL16_3X3[4]] = value;
326         } else {
327             matrix[MAT_IND_ROW16_3X3[4]] = value;
328         }
329     } else {
330         if (mColMaj) {
331             matrix[MAT_IND_COL9_3X3[4]] = value;
332         } else {
333             matrix[MAT_IND_ROW9_3X3[4]] = value;
334         }
335     }
336 }
337
338 /**
339  * Sets the value of Y2.
340  * @param value value to set.
341  */
342 void setY2(float value) {
343
344     if (matrix.length == 16) {
345         if (mColMaj) {
346             matrix[MAT_IND_COL16_3X3[5]] = value;
347         } else {
348             matrix[MAT_IND_ROW16_3X3[5]] = value;
349         }
350     } else {
351         if (mColMaj) {
352             matrix[MAT_IND_COL9_3X3[5]] = value;
353         } else {
354             matrix[MAT_IND_ROW9_3X3[5]] = value;
355         }
356     }
357 }
358
359 /**
360  * Sets the value of Z0.

```

```

366     *
367     * @param value value to set.
368     */
369 void setZ0(float value) {
370     if (matrix.length == 16) {
371         if (mColMaj) {
372             matrix[MAT_IND_COL16_3X3[6]] = value;
373         } else {
374             matrix[MAT_IND_ROW16_3X3[6]] = value;
375         }
376     } else {
377         if (mColMaj) {
378             matrix[MAT_IND_COL9_3X3[6]] = value;
379         } else {
380             matrix[MAT_IND_ROW9_3X3[6]] = value;
381         }
382     }
383 }
384 }
385
386 /**
387  * Sets the value of Z1.
388  *
389  * @param value value to set.
390  */
391 void setZ1(float value) {
392     if (matrix.length == 16) {
393         if (mColMaj) {
394             matrix[MAT_IND_COL16_3X3[7]] = value;
395         } else {
396             matrix[MAT_IND_ROW16_3X3[7]] = value;
397         }
398     } else {
399         if (mColMaj) {
400             matrix[MAT_IND_COL9_3X3[7]] = value;
401         } else {
402             matrix[MAT_IND_ROW9_3X3[7]] = value;
403         }
404     }
405 }
406 }
407
408 /**
409  * Sets the value of Z2.
410  *
411  * @param value value to set.
412  */
413 void setZ2(float value) {
414     if (matrix.length == 16) {
415         if (mColMaj) {
416             matrix[MAT_IND_COL16_3X3[8]] = value;
417         } else {
418             matrix[MAT_IND_ROW16_3X3[8]] = value;
419         }
420     } else {
421         if (mColMaj) {
422             matrix[MAT_IND_COL9_3X3[8]] = value;
423         } else {
424             matrix[MAT_IND_ROW9_3X3[8]] = value;
425         }
426     }
427 }
428 }
429
430 /**
431  * Sets the value of X3.
432  *
433  * @param value value to set.
434  */
435 void setX3(float value) {
436     if (matrix.length == 16) {
437         if (mColMaj) {
438             matrix[MAT_IND_COL16_4X4[3]] = value;
439         } else {
440             matrix[MAT_IND_ROW16_4X4[3]] = value;
441         }
442     } else {
443         throw new IllegalStateException("length of matrix should be 16");
444     }
445 }
446 }
447
448 /**
449  * Sets the value of Y3.
450  *
451  * @param value value to set.
452  */
453 void setY3(float value) {
454     if (matrix.length == 16) {
455         if (mColMaj) {
456             matrix[MAT_IND_COL16_4X4[7]] = value;
457         } else {
458             matrix[MAT_IND_ROW16_4X4[7]] = value;
459         }
460     }

```

```

461     } else {
462         throw new IllegalStateException("length of matrix should be 16");
463     }
464 }
465
466 /**
467  * Sets the value of Z3.
468  *
469  * @param value value to set.
470  */
471 void setZ3(float value) {
472     if (matrix.length == 16) {
473         if (mColMaj) {
474             matrix[MAT_IND_COL16_4X4[11]] = value;
475         } else {
476             matrix[MAT_IND_ROW16_4X4[11]] = value;
477         }
478     } else {
479         throw new IllegalStateException("length of matrix should be 16");
480     }
481 }
482
483 /**
484  * Sets the value of W0.
485  *
486  * @param value value to set.
487  */
488 void setW0(float value) {
489     if (matrix.length == 16) {
490         if (mColMaj) {
491             matrix[MAT_IND_COL16_4X4[12]] = value;
492         } else {
493             matrix[MAT_IND_ROW16_4X4[12]] = value;
494         }
495     } else {
496         throw new IllegalStateException("length of matrix should be 16");
497     }
498 }
499
500 /**
501  * Sets the value of W1.
502  *
503  * @param value value to set.
504  */
505 void setW1(float value) {
506     if (matrix.length == 16) {
507         if (mColMaj) {
508             matrix[MAT_IND_COL16_4X4[13]] = value;
509         } else {
510             matrix[MAT_IND_ROW16_4X4[13]] = value;
511         }
512     } else {
513         throw new IllegalStateException("length of matrix should be 16");
514     }
515 }
516
517 /**
518  * Sets the value of W2.
519  *
520  * @param value value to set.
521  */
522 void setW2(float value) {
523     if (matrix.length == 16) {
524         if (mColMaj) {
525             matrix[MAT_IND_COL16_4X4[14]] = value;
526         } else {
527             matrix[MAT_IND_ROW16_4X4[14]] = value;
528         }
529     } else {
530         throw new IllegalStateException("length of matrix should be 16");
531     }
532 }
533
534 /**
535  * Sets the value of W3.
536  *
537  * @param value value to set.
538  */
539 void setW3(float value) {
540     if (matrix.length == 16) {
541         if (mColMaj) {
542             matrix[MAT_IND_COL16_4X4[15]] = value;
543         } else {
544             matrix[MAT_IND_ROW16_4X4[15]] = value;
545         }
546     } else {
547         throw new IllegalStateException("length of matrix should be 16");
548     }
549 }
550 }
551 }
552 }
553 }
554 }
555 }

```



```

85  *
86  * @see project.pc.sensor.representation.Vector4f
87  */
88
89  public class Quaternion extends Vector4f {
90
91      //For better performance update it only when it is accessed, not on every change
92
93      private MatrixF4x4 mMatrix;
94      private boolean mDirty = false;
95      private Vector4f mTmpVector = new Vector4f();
96      private Quaternion mTmpQuaternion;
97
98      /**
99       * Constructor.
100      *
101      * Initializes this <code>Quaternion</code>.
102      */
103      public Quaternion() {
104          super();
105          mMatrix = new MatrixF4x4();
106          loadIdentityQuat();
107      }
108
109      /**
110       * Normalizes this <code>Quaternion</code>.
111      */
112      public void normalize() {
113          this.mDirty = true;
114          float mag = (float) Math.sqrt(mPoints[3] * mPoints[3] + mPoints[0] *
mPoints[0] + mPoints[1] * mPoints[1] +
115              mPoints[2] * mPoints[2]);
116          mPoints[3] = mPoints[3] / mag;
117          mPoints[0] = mPoints[0] / mag;
118          mPoints[1] = mPoints[1] / mag;
119          mPoints[2] = mPoints[2] / mag;
120      }
121
122      /**
123       * Copies the values from the given <code>Quaternion</code>
124       * to this one.
125       *
126       * @param quat the <code>Quaternion</code> to copy from.
127       */
128      public void set(Quaternion quat) {
129          this.mDirty = true;
130          copyVec4(quat);
131      }
132
133      /**
134       * Multiplies this <code>Quaternion</code> with another.
135       *
136       * @param input <code>Quaternion</code> with which
137       * <code>this</code> is multiplied.
138       * @param output result of <code>(this * input)</code>.
139       */
140      public void multiplyByQuat(Quaternion input, Quaternion output) {
141
142          if (input != output) {
143              output.mPoints[3] = (mPoints[3] * input.mPoints[3] - mPoints[0] *
input.mPoints[0] - mPoints[1] * input.mPoints[1] - mPoints[2] * input.mPoints[2]);
144              output.mPoints[0] = (mPoints[3] * input.mPoints[0] + mPoints[0] *
input.mPoints[3] + mPoints[1] * input.mPoints[2] - mPoints[2] * input.mPoints[1]);
145              output.mPoints[1] = (mPoints[3] * input.mPoints[1] + mPoints[1] *
input.mPoints[3] + mPoints[2] * input.mPoints[0] - mPoints[0] * input.mPoints[2]);
146              output.mPoints[2] = (mPoints[3] * input.mPoints[2] + mPoints[2] *
input.mPoints[3] + mPoints[0] * input.mPoints[1] - mPoints[1] * input.mPoints[0]);
147          } else {
148              mTmpVector.mPoints[0] = input.mPoints[0];
149              mTmpVector.mPoints[1] = input.mPoints[1];
150              mTmpVector.mPoints[2] = input.mPoints[2];
151              mTmpVector.mPoints[3] = input.mPoints[3];
152
153              output.mPoints[3] = (mPoints[3] * mTmpVector.mPoints[3] - mPoints[0] *
mTmpVector.mPoints[0] - mPoints[1] * mTmpVector.mPoints[1] - mPoints[2] *
mTmpVector.mPoints[2]);
154              output.mPoints[0] = (mPoints[3] * mTmpVector.mPoints[0] + mPoints[0] *
mTmpVector.mPoints[3] + mPoints[1] * mTmpVector.mPoints[2] - mPoints[2] *
mTmpVector.mPoints[1]);
155              output.mPoints[1] = (mPoints[3] * mTmpVector.mPoints[1] + mPoints[1] *
mTmpVector.mPoints[3] + mPoints[2] * mTmpVector.mPoints[0] - mPoints[0] *
mTmpVector.mPoints[2]);
156              output.mPoints[2] = (mPoints[3] * mTmpVector.mPoints[2] + mPoints[2] *
mTmpVector.mPoints[3] + mPoints[0] * mTmpVector.mPoints[1] - mPoints[1] *
mTmpVector.mPoints[0]);
157          }
158      }
159
160      /**

```

```

161     * Multiplies this <code>Quaternion</code> by the input one.
162     *
163     * @param input <code>Quaternion</code> to be multiplied with.
164     */
165     public void multiplyByQuat(Quaternion input) {
166         this.mDirty = true;
167         if (mTmpQuaternion == null) mTmpQuaternion = new Quaternion();
168         mTmpQuaternion.copyVec4(this);
169         multiplyByQuat(input, mTmpQuaternion);
170         this.copyVec4(mTmpQuaternion);
171     }
172
173     /**
174     * Multiplies this <code>Quaternion</code> with a scalar.
175     *
176     * This overrides the {@link Vector4f#multiplyByScalar(float)}
177     * method.
178     *
179     * @param scalar the scalar.
180     * @see project.pc.sensor.representation.Vector4f#multiplyByScalar(float)
181     */
182     public void multiplyByScalar(float scalar) {
183         this.mDirty = true;
184         super.multiplyByScalar(scalar);
185     }
186
187     public void addQuat(Quaternion input) {
188         this.mDirty = true;
189         addQuat(input, this);
190     }
191
192     private void addQuat(Quaternion input, Quaternion output) {
193         output.setX(getX() + input.getX());
194         output.setY(getY() + input.getY());
195         output.setZ(getZ() + input.getZ());
196         output.setW(getW() + input.getW());
197     }
198
199     public void subQuat(Quaternion input) {
200         this.mDirty = true;
201         subQuat(input, this);
202     }
203
204     private void subQuat(Quaternion input, Quaternion output) {
205         output.setX(getX() - input.getX());
206         output.setY(getY() - input.getY());
207         output.setZ(getZ() - input.getZ());
208         output.setW(getW() - input.getW());
209     }
210
211     private void convertQuatToMatrix() {
212         float x = mPoints[0];
213         float y = mPoints[1];
214         float z = mPoints[2];
215         float w = mPoints[3];
216
217         mMatrix.setX0(1 - 2 * (y * y) - 2 * (z * z));
218         mMatrix.setX1(2 * (x * y) + 2 * (w * z));
219         mMatrix.setX2(2 * (x * z) - 2 * (w * y));
220         mMatrix.setX3(0);
221         mMatrix.setY0(2 * (x * y) - 2 * (w * z));
222         mMatrix.setY1(1 - 2 * (x * x) - 2 * (z * z));
223         mMatrix.setY2(2 * (y * z) + 2 * (w * x));
224         mMatrix.setY3(0);
225         mMatrix.setZ0(2 * (x * z) + 2 * (w * y));
226         mMatrix.setZ1(2 * (y * z) - 2 * (w * x));
227         mMatrix.setZ2(1 - 2 * (x * x) - 2 * (y * y));
228         mMatrix.setZ3(0);
229         mMatrix.setW0(0);
230         mMatrix.setW1(0);
231         mMatrix.setW2(0);
232         mMatrix.setW3(1);
233     }
234
235     /**
236     * Gets an axis angle representation of this <code>Quaternion</code>.
237     *
238     * @param output <code>Vector4f</code> axis angle.
239     */
240     public void toAxisAngle(Vector4f output) {
241         if (getW() > 1) {
242             normalize();
243         }
244         float angle = 2 * (float) Math.toDegrees(Math.acos(getW()));
245         float x;
246         float y;
247         float z;
248
249         float s = (float) Math.sqrt(1 - getW() * getW());

```

```

250         if (s < 0.001) {
251             x = mPoints[0]; // To get normalised axis replace with x=1 y=z=0
252             y = mPoints[1];
253             z = mPoints[2];
254         } else {
255             // normalise axis
256             x = mPoints[0] / s;
257             y = mPoints[1] / s;
258             z = mPoints[2] / s;
259         }
260
261         output.mPoints[0] = x;
262         output.mPoints[1] = y;
263         output.mPoints[2] = z;
264         output.mPoints[3] = angle;
265     }
266
267     /**
268     * Returns the heading, attitude and bank of this <code>Quaternion</code>
269     * as euler angles in the double array respectively.
270     *
271     * @return an array of size 3 containing the euler angles for this
272     *         <code>Quaternion</code>.
273     */
274     public double[] toEulerAngles() {
275         double[] ret = new double[3];
276
277         ret[0] = Math.atan2(2 * mPoints[1] * getW() - 2 * mPoints[0] * mPoints[2], 1
- 2 * (mPoints[1] * mPoints[1]) - 2 * (mPoints[2] * mPoints[2]));
278         ret[1] = Math.asin(2 * mPoints[0] * mPoints[1] + 2 * mPoints[2] * getW());
279         ret[2] = Math.atan2(2 * mPoints[0] * getW() - 2 * mPoints[1] * mPoints[2], 1
- 2 * (mPoints[0] * mPoints[0]) - 2 * (mPoints[2] * mPoints[2]));
280         return ret;
281     }
282
283     private void loadIdentityQuat() {
284         this.mDirty = true;
285         setX(0);
286         setY(0);
287         setZ(0);
288         setW(1);
289     }
290
291     /**
292     * Returns a <code>String</code> representing this
293     * <code>Quaternion</code>. This overrides the
294     * {@link Vector4f#toString()} method.
295     *
296     * @return <code>String</code> representing this
297     *         <code>Quaternion</code>.
298     */
299     @Override
300     public String toString() {
301         return "X: " + getX() + ", Y: " + getY() + ", Z: " + getZ() + ", W: " + getW()
+ " ";
302     }
303
304     private void generateQuaternionFromMatrix() {
305
306         float qx;
307         float qy;
308         float qz;
309         float qw;
310
311         float[] mat = mMatrix.getMatrix();
312         int[] indices;
313
314         if (this.mMatrix.size() == 16) {
315             if (this.mMatrix.isColumnMajor()) {
316                 indices = MatrixF4x4.MAT_IND_COL16_3X3;
317             } else {
318                 indices = MatrixF4x4.MAT_IND_ROW16_3X3;
319             }
320         } else {
321             if (this.mMatrix.isColumnMajor()) {
322                 indices = MatrixF4x4.MAT_IND_COL9_3X3;
323             } else {
324                 indices = MatrixF4x4.MAT_IND_ROW9_3X3;
325             }
326         }
327
328         int m00 = indices[0];
329         int m01 = indices[1];
330         int m02 = indices[2];
331
332         int m10 = indices[3];
333         int m11 = indices[4];
334         int m12 = indices[5];
335

```

```

336     int m20 = indices[6];
337     int m21 = indices[7];
338     int m22 = indices[8];
339
340     float tr = mat[m00] + mat[m11] + mat[m22];
341     if (tr > 0) {
342         float s = (float) Math.sqrt(tr + 1.0) * 2;
343         qw = 0.25f * s;
344         qx = (mat[m21] - mat[m12]) / s;
345         qy = (mat[m02] - mat[m20]) / s;
346         qz = (mat[m10] - mat[m01]) / s;
347     } else if ((mat[m00] > mat[m11]) & (mat[m00] > mat[m22])) {
348         float s = (float) Math.sqrt(1.0 + mat[m00] - mat[m11] - mat[m22]) * 2;
349         qw = (mat[m21] - mat[m12]) / s;
350         qx = 0.25f * s;
351         qy = (mat[m01] + mat[m10]) / s;
352         qz = (mat[m02] + mat[m20]) / s;
353     } else if (mat[m11] > mat[m22]) {
354         float s = (float) Math.sqrt(1.0 + mat[m11] - mat[m00] - mat[m22]) * 2;
355         qw = (mat[m02] - mat[m20]) / s;
356         qx = (mat[m01] + mat[m10]) / s;
357         qy = 0.25f * s;
358         qz = (mat[m12] + mat[m21]) / s;
359     } else {
360         float s = (float) Math.sqrt(1.0 + mat[m22] - mat[m00] - mat[m11]) * 2;
361         qw = (mat[m10] - mat[m01]) / s;
362         qx = (mat[m02] + mat[m20]) / s;
363         qy = (mat[m12] + mat[m21]) / s;
364         qz = 0.25f * s;
365     }
366
367     setX(qx);
368     setY(qy);
369     setZ(qz);
370     setW(qw);
371 }
372
373 public void setColumnMajor(float[] mMatrix) {
374
375     this.mMatrix.setMatrix(mMatrix);
376     this.mMatrix.setColumnMajor(true);
377
378     generateQuaternionFromMatrix();
379 }
380
381 public void setRowMajor(float[] mMatrix) {
382
383     this.mMatrix.setMatrix(mMatrix);
384     this.mMatrix.setColumnMajor(false);
385
386     generateQuaternionFromMatrix();
387 }
388
389 public void setEulerAngle(float azimuth, float pitch, float roll) {
390
391     double heading = Math.toRadians(roll);
392     double attitude = Math.toRadians(pitch);
393     double bank = Math.toRadians(azimuth);
394
395     double c1 = Math.cos(heading / 2);
396     double s1 = Math.sin(heading / 2);
397     double c2 = Math.cos(attitude / 2);
398     double s2 = Math.sin(attitude / 2);
399     double c3 = Math.cos(bank / 2);
400     double s3 = Math.sin(bank / 2);
401     double clc2 = c1 * c2;
402     double sls2 = s1 * s2;
403     setW((float) (clc2 * c3 - sls2 * s3));
404     setX((float) (clc2 * s3 + sls2 * c3));
405     setY((float) (s1 * c2 * c3 + c1 * s2 * s3));
406     setZ((float) (c1 * s2 * c3 - s1 * c2 * s3));
407
408     mDirty = true;
409 }
410
411 public void setAxisAngle(Vector3f vec, float rot) {
412     double s = Math.sin(Math.toRadians(rot / 2));
413     setX(vec.getX() * (float) s);
414     setY(vec.getY() * (float) s);
415     setZ(vec.getZ() * (float) s);
416     setW((float) Math.cos(Math.toRadians(rot / 2)));
417
418     mDirty = true;
419 }
420
421 public void setAxisAngleRad(Vector3f vec, double rot) {
422     double s = rot / 2;
423     setX(vec.getX() * (float) s);
424     setY(vec.getY() * (float) s);

```

```

425         setZ(vec.getZ() * (float) s);
426         setW((float) rot / 2);
427
428     mDirty = true;
429 }
430
431 public MatrixF4x4 getMatrix4x4() {
432     if (mDirty) {
433         convertQuatToMatrix();
434         mDirty = false;
435     }
436     return this.mMatrix;
437 }
438
439 public void copyFromVec3(Vector3f vec, float w) {
440     copyFromV3f(vec, w);
441 }
442
443 /**
444  * Computes a linear interpolation between this Quaternion and the
445  * input Quaternion.
446  *
447  * @param input the Quaternion to be slerp'd with this.
448  * @param output the Quaternion to store the result.
449  * @param t the ratio between the two quaternions where 0 <= t <= 1.0 . <br/>
450  *          <b><i>Note:</i></b> Increase value of t will bring
451  *          rotation closer to the input Quaternion.
452  */
453 public void slerp(Quaternion input, Quaternion output, float t) {
454     // Calculate angle between them
455     Quaternion bufferQuat;
456     float cosHalftheta = this.dotProduct(input);
457
458     if (cosHalftheta < 0) {
459         if (mTmpQuaternion == null) mTmpQuaternion = new Quaternion();
460         bufferQuat = mTmpQuaternion;
461         cosHalftheta = -cosHalftheta;
462         bufferQuat.mPoints[0] = (-input.mPoints[0]);
463         bufferQuat.mPoints[1] = (-input.mPoints[1]);
464         bufferQuat.mPoints[2] = (-input.mPoints[2]);
465         bufferQuat.mPoints[3] = (-input.mPoints[3]);
466     } else {
467         bufferQuat = input;
468     }
469
470     if (Math.abs(cosHalftheta) >= 1.0) {
471         output.mPoints[0] = (this.mPoints[0]);
472         output.mPoints[1] = (this.mPoints[1]);
473         output.mPoints[2] = (this.mPoints[2]);
474         output.mPoints[3] = (this.mPoints[3]);
475     } else {
476         double sinHalfTheta = Math.sqrt(1.0 - cosHalftheta * cosHalftheta);
477         double halfTheta = Math.acos(cosHalftheta);
478
479         double ratioA = Math.sin((1 - t) * halfTheta) / sinHalfTheta;
480         double ratioB = Math.sin(t * halfTheta) / sinHalfTheta;
481
482         //Calculate Quaternion
483         output.mPoints[3] = ((float) (mPoints[3] * ratioA + bufferQuat.mPoints[3]
484 * ratioB));
485         output.mPoints[0] = ((float) (this.mPoints[0] * ratioA +
486 bufferQuat.mPoints[0] * ratioB));
487         output.mPoints[1] = ((float) (this.mPoints[1] * ratioA +
488 bufferQuat.mPoints[1] * ratioB));
489         output.mPoints[2] = ((float) (this.mPoints[2] * ratioA +
490 bufferQuat.mPoints[2] * ratioB));
491     }
492 }
493
494 /**
495  * Rotates a 3-D vector by this Quaternion.
496  *
497  * @param v the {@link Vector3f} to be rotated.
498  * @return the rotated {@link Vector3f}.
499  */
500 public Vector3f rotateVector(Vector3f v) {
501     float q0 = this.mPoints[3];
502     float q1 = this.mPoints[0];
503     float q2 = this.mPoints[1];
504     float q3 = this.mPoints[2];
505     float v1 = v.getX();
506     float v2 = v.getY();
507     float v3 = v.getZ();
508     float x = (1-2*q2*q2-2*q3*q3)*v1 + 2*(q1*q2+q0*q3)*v2 + 2*(q1*q3-q0*q2)*v3;
509     float y = 2*(q1*q2-q0*q3)*v1 + (1-2*q1*q1-2*q3*q3)*v2 + 2*(q2*q3+q0*q1)*v3;
510     float z = 2*(q1*q3-q0*q2) + (q2*q3-q0*q1)*v2 + (1-2*q1*q1-2*q2*q2)*v3;
511     return new Vector3f(x,y,z);
512 }
513 }

```

Vector3f.java

```

1 package project.pc.sensor.representation;
2
3 /**
4  * Class to represent a three dimensional vector.
5  *
6  * <p>
7  * It is defined as a point <i>(x,y,z)</i> in three dimensional
8  * vector space. It is stored as a array of <code>float</code>s.
9  * The three dimensions are known as X,Y and Z.
10 * </p>
11 * <p>
12 * <b>Dot product of two vectors : </b> <br/>
13 * <i>
14 * ( $a_1, b_1, c_1$ ) .
15 * ( $a_2, b_2, c_2$ ) =
16 *  $a_1a_2 + b_1b_2 + c_1c_2$ 
17 * </i>.
18 * </p>
19 * <p>
20 * <b>Product of a vector with a scalar : </b> <br/>
21 * <i>(a, b, c) * s = (a * s, b * s, c * s)</i>.
22 * </p>
23 * <p>
24 * <b>Normalization of a vector</b> <i>(a,b,c)</i> <br/>
25 * <math>\frac{1}{\sqrt{a^2+b^2+c^2}}(a, b, c)</math>
26 * = <i>(a/mag, b/mag, c/mag)</i>, <br/>
27 * where, mag =
28 * <math>\sqrt{a^2+b^2+c^2}</math>
29 * </p>
30 * @see project.pc.sensor.representation.Vector3f
31 * @see project.pc.sensor.representation.Quaternion
32 */
33
34 public class Vector3f {
35     private float[] mPoints = new float[3];
36
37     /**
38      * Constructor.
39      * Creates the <code>Vector3f (x,y,z)</code>.
40      *
41      * @param x the X-dimension.
42      * @param y the Y-dimension.
43      * @param z the Z-dimension.
44      */
45     public Vector3f(float x, float y, float z) {
46         this.mPoints[0] = x;
47         this.mPoints[1] = y;
48         this.mPoints[2] = z;
49     }
50
51     public Vector3f(float value) {
52         this.mPoints[0] = value;
53         this.mPoints[1] = value;
54         this.mPoints[2] = value;
55     }
56
57     /**
58      * Constructor.
59      * Default constructor.
60      */
61     public Vector3f() {
62     }
63
64     /**
65      * Constructor.
66      * Copy constructor. Copies another <code>Vector3f</code> to
67      * this one.
68      *
69      * @param vector the <code>Vector3f</code> to be copied.
70      */
71     public Vector3f(Vector3f vector) {
72         this.mPoints[0] = vector.mPoints[0];
73         this.mPoints[1] = vector.mPoints[1];
74         this.mPoints[2] = vector.mPoints[2];
75     }
76
77     public Vector3f(Vector4f vector) {

```



```

85         if (vector.getW() != 0) {
86             this.mPoints[0] = vector.getX() / vector.getW();
87             this.mPoints[1] = vector.getY() / vector.getW();
88             this.mPoints[2] = vector.getZ() / vector.getW();
89         } else {
90             this.mPoints[0] = vector.getX();
91             this.mPoints[1] = vector.getY();
92             this.mPoints[2] = vector.getZ();
93         }
94     }
95
96     /**
97     * Returns the array representation of this <code>Vector3f</code>.
98     * @return the array <code>[ X, Y, Z ]</code>.
99     */
100    float[] toArray() {
101        return this.mPoints;
102    }
103
104    public void add(Vector3f summand) {
105        this.mPoints[0] += summand.mPoints[0];
106        this.mPoints[1] += summand.mPoints[1];
107        this.mPoints[2] += summand.mPoints[2];
108    }
109
110    public void add(float summand) {
111        this.mPoints[0] += summand;
112        this.mPoints[1] += summand;
113        this.mPoints[2] += summand;
114    }
115
116    public void subtract(Vector3f subtrahend) {
117        this.mPoints[0] -= subtrahend.mPoints[0];
118        this.mPoints[1] -= subtrahend.mPoints[1];
119        this.mPoints[2] -= subtrahend.mPoints[2];
120    }
121
122    /**
123    * Multiplies this <code>Vector3f</code> with a scalar.
124    * .
125    * @param scalar the scalar.
126    */
127    public void multiplyByScalar(float scalar) {
128        this.mPoints[0] *= scalar;
129        this.mPoints[1] *= scalar;
130        this.mPoints[2] *= scalar;
131    }
132
133    /**
134    * Normalizes this <code>Vector3f</code>.
135    */
136    public void normalize() {
137        double a = Math.sqrt(mPoints[0] * mPoints[0] + mPoints[1] * mPoints[1] +
138        mPoints[2] * mPoints[2]);
139        this.mPoints[0] = (float) (this.mPoints[0] / a);
140        this.mPoints[1] = (float) (this.mPoints[1] / a);
141        this.mPoints[2] = (float) (this.mPoints[2] / a);
142    }
143
144    /**
145    * Returns the X-dimension of this <code>Vector3f</code>.
146    * @return the X-dimension.
147    */
148    public float getX() {
149        return mPoints[0];
150    }
151
152    /**
153    * Returns the Y-dimension of this <code>Vector3f</code>.
154    * @return the Y-dimension.
155    */
156    public float getY() {
157        return mPoints[1];
158    }
159
160    /**
161    * Returns the Z-dimension of this <code>Vector3f</code>.
162    * @return the Z-dimension.
163    */
164    public float getZ() {
165        return mPoints[2];
166    }
167
168    }
169
170
171
172

```



```

173  /**
174   * Sets the X-dimension of this Vector3f.
175   *
176   * @param x the X-dimension.
177   */
178  void setX(float x) {
179      this.mPoints[0] = x;
180  }
181
182  /**
183   * Sets the Y-dimension of this Vector3f.
184   *
185   * @param y the Y-dimension.
186   */
187  void setY(float y) {
188      this.mPoints[1] = y;
189  }
190
191  /**
192   * Sets the Z-dimension of this Vector3f.
193   *
194   * @param z the Z-dimension.
195   */
196  void setZ(float z) {
197      this.mPoints[2] = z;
198  }
199
200  /**
201   * Sets this Vector3f as (x,y,z).
202   *
203   * @param x the X-dimension.
204   * @param y the Y-dimension.
205   * @param z the Z-dimension.
206   */
207  public void setXYZ(float x, float y, float z) {
208      this.mPoints[0] = x;
209      this.mPoints[1] = y;
210      this.mPoints[2] = z;
211  }
212
213  /**
214   * Performs the dot product of two vectors.
215   *
216   * @param inputVec the Vector3f to be multiplied.
217   * @return result of (this . inputVec).
218   */
219  public float dotProduct(Vector3f inputVec) {
220      return mPoints[0] * inputVec.mPoints[0] + mPoints[1] * inputVec.mPoints[1] +
221      mPoints[2] * inputVec.mPoints[2];
222  }
223
224  public void crossProduct(Vector3f inputVec, Vector3f outputVec) {
225      outputVec.setX(mPoints[1] * inputVec.mPoints[2] - mPoints[2] *
226      inputVec.mPoints[1]);
227      outputVec.setY(mPoints[2] * inputVec.mPoints[0] - mPoints[0] *
228      inputVec.mPoints[2]);
229      outputVec.setZ(mPoints[0] * inputVec.mPoints[1] - mPoints[1] *
230      inputVec.mPoints[0]);
231  }
232
233  public float getLength() {
234      return (float) Math.sqrt(mPoints[0] * mPoints[0] + mPoints[1] * mPoints[1] +
235      mPoints[2] * mPoints[2]);
236  }
237
238  /**
239   * Returns the String representation of this 
240   * Vector3f.
241   *
242   * It overrides the {@link java.lang.Object#toString()} method.
243   *
244   * @return String representing this
245   *         Vector3f.
246   */
247  @Override
248  public String toString() {
249      return "X:" + mPoints[0] + " Y:" + mPoints[1] + " Z:" + mPoints[2];
250  }
251
252  public void set(Vector3f source) {
253      set(source.mPoints);
254  }
255
256  private void set(float[] source) {
257      System.arraycopy(source, 0, mPoints, 0, 3);
258  }

```

Vector4f.java

[illegible]

```

85  /**
86  * Returns the array representation of this <code>Vector4f</code>.
87  *
88  * @return the array <code>[ X, Y, Z, W ]</code>.
89  */
90  public float[] array() {
91      return mPoints;
92  }
93
94  /**
95  * Copies another <code>Vector4f</code> to this one.
96  *
97  * @param vec the <code>Vector4f</code> to be copied.
98  */
99  void copyVec4(Vector4f vec) {
100     this.mPoints[0] = vec.mPoints[0];
101     this.mPoints[1] = vec.mPoints[1];
102     this.mPoints[2] = vec.mPoints[2];
103     this.mPoints[3] = vec.mPoints[3];
104 }
105
106 public void add(Vector4f vector) {
107     this.mPoints[0] += vector.mPoints[0];
108     this.mPoints[1] += vector.mPoints[1];
109     this.mPoints[2] += vector.mPoints[2];
110     this.mPoints[3] += vector.mPoints[3];
111 }
112
113 public void add(Vector3f vector, float w) {
114     this.mPoints[0] += vector.getX();
115     this.mPoints[1] += vector.getY();
116     this.mPoints[2] += vector.getZ();
117     this.mPoints[3] += w;
118 }
119
120 public void subtract(Vector4f vector) {
121     this.mPoints[0] -= vector.mPoints[0];
122     this.mPoints[1] -= vector.mPoints[1];
123     this.mPoints[2] -= vector.mPoints[2];
124     this.mPoints[3] -= vector.mPoints[3];
125 }
126
127 public void subtract(Vector4f vector, Vector4f output) {
128     output.setXYZW(this.mPoints[0] - vector.mPoints[0], this.mPoints[1] -
vector.mPoints[1], this.mPoints[2]
- vector.mPoints[2], this.mPoints[3] - vector.mPoints[3]);
129 }
130
131 public void subdivide(Vector4f vector) {
132     this.mPoints[0] /= vector.mPoints[0];
133     this.mPoints[1] /= vector.mPoints[1];
134     this.mPoints[2] /= vector.mPoints[2];
135     this.mPoints[3] /= vector.mPoints[3];
136 }
137
138 /**
139 * Multiplies this <code>Vector4f</code> with a scalar.
140 *
141 * @param scalar the scalar.
142 */
143 public void multiplyByScalar(float scalar) {
144     this.mPoints[0] *= scalar;
145     this.mPoints[1] *= scalar;
146     this.mPoints[2] *= scalar;
147     this.mPoints[3] *= scalar;
148 }
149
150 /**
151 * Performs the dot product of two vectors.
152 *
153 * @param input the <code>Vector4f</code> to be multiplied.
154 * @return result of <code>(this . input)</code>.
155 */
156 float dotProduct(Vector4f input) {
157     return this.mPoints[0] * input.mPoints[0] + this.mPoints[1] *
input.mPoints[1] + this.mPoints[2] * input.mPoints[2]
+ this.mPoints[3] * input.mPoints[3];
158 }
159
160 public void lerp(Vector4f input, Vector4f output, float t) {
161     output.mPoints[0] = (mPoints[0] * (1.0f * t) + input.mPoints[0] * t);
162     output.mPoints[1] = (mPoints[1] * (1.0f * t) + input.mPoints[1] * t);
163     output.mPoints[2] = (mPoints[2] * (1.0f * t) + input.mPoints[2] * t);
164     output.mPoints[3] = (mPoints[3] * (1.0f * t) + input.mPoints[3] * t);
165 }
166
167 /**
168 * Normalizes this <code>Vector4f</code>.
169 */
170
171

```

```

172     */
173     public void normalize() {
174         if (mPoints[3] == 0)
175             return;
176
177         mPoints[0] /= mPoints[3];
178         mPoints[1] /= mPoints[3];
179         mPoints[2] /= mPoints[3];
180
181         double a = Math.sqrt(this.mPoints[0] * this.mPoints[0] + this.mPoints[1] *
this.mPoints[1] + this.mPoints[2]
182             * this.mPoints[2]);
183         mPoints[0] = (float) (this.mPoints[0] / a);
184         mPoints[1] = (float) (this.mPoints[1] / a);
185         mPoints[2] = (float) (this.mPoints[2] / a);
186     }
187
188     /**
189     * Returns the X-dimension of this <code>Vector4f</code>.
190     *
191     * @return the X-dimension.
192     */
193     public float getX() {
194         return this.mPoints[0];
195     }
196
197     /**
198     * Returns the Y-dimension of this <code>Vector4f</code>.
199     *
200     * @return the Y-dimension.
201     */
202     public float getY() {
203         return this.mPoints[1];
204     }
205
206     /**
207     * Returns the Z-dimension of this <code>Vector4f</code>.
208     *
209     * @return the Z-dimension.
210     */
211     public float getZ() {
212         return this.mPoints[2];
213     }
214
215     /**
216     * Returns the W-dimension of this <code>Vector4f</code>.
217     *
218     * @return the W-dimension.
219     */
220     public float getW() {
221         return this.mPoints[3];
222     }
223
224     /**
225     * Sets the X-dimension of this <code>Vector4f</code>.
226     *
227     * @param x the X-dimension.
228     */
229     public void setX(float x) {
230         this.mPoints[0] = x;
231     }
232
233     /**
234     * Sets the Y-dimension of this <code>Vector4f</code>.
235     *
236     * @param y the Y-dimension.
237     */
238     public void setY(float y) {
239         this.mPoints[1] = y;
240     }
241
242     /**
243     * Sets the Z-dimension of this <code>Vector4f</code>.
244     *
245     * @param z the Z-dimension.
246     */
247     public void setZ(float z) {
248         this.mPoints[2] = z;
249     }
250
251     /**
252     * Sets the W-dimension of this <code>Vector4f</code>.
253     *
254     * @param w the W-dimension.
255     */
256     public void setW(float w) {
257         this.mPoints[3] = w;
258     }
259

```

```

260     /**
261     * Sets all four dimensions of this <code>Vector4f</code>.
262     */
263     private void setXYZW(float x, float y, float z, float w) {
264         this.mPoints[0] = x;
265         this.mPoints[1] = y;
266         this.mPoints[2] = z;
267         this.mPoints[3] = w;
268     }
269
270     public boolean compareTo(Vector4f rhs) {
271         boolean ret = false;
272         if (this.mPoints[0] == rhs.mPoints[0] && this.mPoints[1] == rhs.mPoints[1] &&
this.mPoints[2] == rhs.mPoints[2]
&& this.mPoints[3] == rhs.mPoints[3])
273             ret = true;
274         return ret;
275     }
276
277
278
279     /**
280     * Copies a {@link Vector3f} to this <code>Vector4f</code> using
281     * a separate value for W-dimension.
282     *
283     * @param input the {@link Vector3f}.
284     * @param w <code>float</code> representing the W-dimension.
285     */
286     void copyFromV3f(Vector3f input, float w) {
287         mPoints[0] = (input.getX());
288         mPoints[1] = (input.getY());
289         mPoints[2] = (input.getZ());
290         mPoints[3] = (w);
291     }
292
293     /**
294     * Returns the <code>String</code> representation of this <code>
295     * Vector4f</code>.
296     *
297     * It overrides the {@link java.lang.Object#toString()} method.
298     *
299     * @return <code>String</code> representing this
300     *         <code>Vector4f</code>.
301     */
302     @Override
303     public String toString() {
304         return "X:" + mPoints[0] + " Y:" + mPoints[1] + " Z:" + mPoints[2] + " W:" +
mPoints[3];
305     }
306 }

```

SensorDataHandler.java

```
1 package project.pc.sensor;
2
3 import project.pc.sensor.representation.Cartesian2D;
4 import project.pc.sensor.representation.Quaternion;
5 import project.pc.sensor.representation.Vector3f;
6
7 /**
8  * Class to transform filtered sensor data into mouse pointer movement
9  * data.
10  */
11 public class SensorDataHandler {
12     private float mSensitivity;
13     private double mHcos;
14     private double mHsin;
15     private double mBli;
16
17     /**
18      * Constructor.
19      *
20      * Initializes this <code>SensorDataHandler</code>.
21      *
22      * @param q initial <code>Quaternion</code> object.
23      * @param sensitivity sensitivity co-efficient of the mouse movement.
24      */
25     public SensorDataHandler(Quaternion q, float sensitivity) {
26         mSensitivity = sensitivity;
27         initFix(q);
28     }
29
30     //Baseline Initialization
31     private void initFix(Quaternion quaternion) {
32         Vector3f xvec = quaternion.rotateVector(new Vector3f(1.0f,0.0f,0.0f));
33         Vector3f zvec = quaternion.rotateVector(new Vector3f(0.0f,0.0f,1.0f));
34         double blh = -1.0f * Math.atan2(xvec.getY(),xvec.getX());
35         mHcos = Math.cos(blh);
36         mHsin = Math.sin(blh);
37         mBli = Math.asin(zvec.getY());
38     }
39
40     /**
41      *
42      * Determines pointer location from current <code>Quaternion</code>
43      * object.
44      *
45      * @param currentQuaternion current <code>Quaternion</code> object acquired
46      * after filtering.
47      * @return an object of {@link project.pc.sensor.representation.Cartesian2D}
48      * containing current pointer location.
49      */
50     public Cartesian2D pointerUpdate(Quaternion currentQuaternion) {
51         Vector3f cxv = currentQuaternion.rotateVector(new Vector3f(1.0f, 0.0f,
52 0.0f));
53         Vector3f czv = currentQuaternion.rotateVector(new Vector3f(0.0f, 0.0f,
54 1.0f));
55         double xeff = (mHcos * cxv.getX()) - (mHsin * cxv.getY());
56         double yeff = (mHsin * cxv.getX()) + (mHcos * cxv.getY());
57         double hdel = Math.atan2(yeff, xeff);
58         double cli = Math.asin(czv.getY());
59         double yEst = mSensitivity * Math.tan(mBli - cli);
60         double xEst = mSensitivity * Math.tan(hdel);
61         initFix(currentQuaternion);
62         return new Cartesian2D(xEst,yEst);
63     }
64 }
```

Main.java

```
1 package project.pc;
2
3 import project.pc.gui.MainWindow;
4 import project.pc.net.NetworkManager;
5 import project.pc.net.ServerInfo;
6 import project.pc.security.EKEProvider;
7
8 import java.awt.AWTException;
9 import java.io.IOException;
10 import java.net.InetAddress;
11
12 /**
13  * Class containing the <code>main()</code> method.
14  *
15  * <p>
16  *   The {@link #main(String[])} method :
17  *   <ul>
18  *     <li>Generates the public key of the server.</li>
19  *     <li>
20  *       Instantiates the {@link project.pc.net.NetworkManager}
21  *       for handling the network operations.
22  *     </li>
23  *     <li>
24  *       Displays the main window of the GUI.
25  *     </li>
26  *   </ul>
27  * </p>
28  *
29  * @see #main(String[])
30  * @see project.pc.net.NetworkManager
31  * @see project.pc.net.ServerInfo
32  */
33
34 class Main {
35     /**
36      * The <code>main</code> method.
37      *
38      * @param args the command-line arguments.
39      * @throws IOException
40      * @throws AWTException
41      * @throws InterruptedException
42      */
43     public static void main(String args[]) throws
44     IOException, AWTException, InterruptedException {
45         byte[] publicKey = new EKEProvider().getBase64EncodedPubKey();
46         String systemInfo = System.getProperty("user.name") + "@" +
47         InetAddress.getLocalHost().getCanonicalHostName();
48         ServerInfo serverInfo = new ServerInfo(publicKey, systemInfo);
49         NetworkManager manager = new NetworkManager(serverInfo);
50         System.setProperty("awt.useSystemAAFontSettings", "on");
51         System.setProperty("swing.aatext", "true");
52         new MainWindow(manager);
53     }
54 }
```

4.2.2 Class Documentation

List of Classes

[detail level [1](#) [2](#) [3](#) [4](#) [5](#)]

▼ N project	
▼ N pc	
▼ N device	
c KeyboardController	Utility for performing keyboard input operations
c KeyCode	Maps the characters to type to the corresponding key codes
c MouseController	Utility for performing mouse input operations
c MouseMove	Provides static methods used by MouseController
▼ N gui	
c Cube	Class to render multi-colored 3D cube
c MainWindow	Defines the main window of the application
▼ N net	
c BroadcastThread	Implementation of the java.lang Runnable for the thread responsible for making UDP broadcasts
c ClientInfo	Encapsulation of the data received from the client at the time of establishing TCP connection
c DataWrapper	Encapsulates the data received from client during TCP communication
c NetworkManager	Manages the different operations on the network
c NetworkState	Stores the state information about the network
c NetworkThread	Implementation of the java.lang Runnable for the thread that executes the network operations
c Server	Server data structure
c ServerInfo	Encapsulation of the data sent during UDP broadcast
c ServerThread	Implementation of the java.lang Runnable for the thread that is responsible for communication with each client
▼ N security	
c EKEProvider	Utility for the end-to-end security module
c KeyStoreManager	Manages the operations on the KeyStore
▼ N sensor	
▼ N representation	
c Cartesian2D	Class to represent a point in 2-D Cartesian coordinate system
c Matrix	Matrix math utilities
c MatrixF4x4	Class to represent a 4x4 matrix (float)
c Quaternion	Class to represent a quaternion
c Vector3f	Class to represent a three dimensional vector
c Vector4f	Class to represent a four dimensional vector
c SensorDataHandler	Class to transform filtered sensor data into mouse pointer movement data
c Main	Class containing the <code>main()</code> method

project.pc.device.KeyboardController Class Reference

Utility for performing keyboard input operations. [More...](#)

Collaboration diagram for project.pc.device.KeyboardController:

Public Member Functions

KeyboardController () throws AWTException
Constructor. [More...](#)

void **doKeyOperation** (String s)
Method to perform the keyboard input operations. [More...](#)

void **doSpecialKeyOperation** (String s)
Method to perform keyboard input operations for different function keys. [More...](#)

Detailed Description

Utility for performing keyboard input operations.

It uses a **java.awt.Robot** object to generate native system input events. This differs from posting events to the AWT event queue or AWT components in that the events are generated in the platform's native input queue.

See also

[java.awt.Robot](#)

Definition at line **19** of file **KeyboardController.java**.

Constructor Documentation

project.pc.device.KeyboardController.KeyboardController () throws AWTException

Constructor.

Initializes this KeyboardController by instantiating a **java.awt.Robot** object.

Exceptions

AWTException

Definition at line **30** of file **KeyboardController.java**.

Member Function Documentation

void project.pc.device.KeyboardController.doKeyOperation (String s)

Method to perform the keyboard input operations.

Parameters

s String representing the sequence of characters to be typed.

Definition at line **40** of file **KeyboardController.java**.

void project.pc.device.KeyboardController.doSpecialKeyOperation (String s)

Method to perform keyboard input operations for different function keys.

Parameters

s String representing the key to be pressed.

Definition at line **57** of file **KeyboardController.java**.

The documentation for this class was generated from the following file:

- src/project/pc/device/**KeyboardController.java**

project.pc.device.KeyCode Class Reference

Maps the characters to type to the corresponding key codes. [More...](#)

Collaboration diagram for project.pc.device.KeyCode:

Static Package Functions

static int **getCodeByChar** (char c)
Checks the *Key Map* and obtains the key code for a character (if Shift key press is not required). [More...](#)

static int **getShiftCode** (char c)
Checks the *Shift Key Map* and obtains the shift key code for a character. [More...](#)

static boolean **isShiftKey** (char c)
Checks if Shift key needs to be pressed for typing a character. [More...](#)

Detailed Description

Maps the characters to type to the corresponding key codes.

These key codes represent the key(s) to be pressed. The mapping is done with the help of two HashMaps :

1. *Key Map* - Maps characters to key codes where pressing shift key is not required.
2. *Shift Key Map* - Maps characters to key codes where pressing shift key is required. For example, the character % is mapped to 5 since Shift+5 is pressed for typing %.

Definition at line **28** of file **KeyCode.java**.

Member Function Documentation

static int project.pc.device.KeyCode.getCodeByChar (char c)

static package

Checks the *Key Map* and obtains the key code for a character (if Shift key press is not required).

Parameters

c the character to be checked.

Returns

the key code for c,
-1, if Shift key press is required.

Definition at line **125** of file **KeyCode.java**.

static int project.pc.device.KeyCode.getShiftCode (char *c*)

static

package

Checks the *Shift Key Map* and obtains the shift key code for a character.

Parameters

c the character to be checked.

Returns

the shift key code for *c*,
-1, if Shift key press is not required.

Definition at line **111** of file **[KeyCode.java](#)**.

static boolean project.pc.device.KeyCode.isShiftKey (char *c*)

static

package

Checks if Shift key needs to be pressed for typing a character.

Parameters

c the character to be checked.

Returns

true, if Shift key needs to be pressed,
false, otherwise.

Definition at line **98** of file **[KeyCode.java](#)**.

The documentation for this class was generated from the following file:

- [src/project/pc/device/KeyCode.java](#)

project.pc.device.MouseController Class Reference

Utility for performing mouse input operations. [More...](#)

Collaboration diagram for project.pc.device.MouseController:

Public Member Functions

MouseController () throws `AWTException`
Constructor. [More...](#)

void **doOperation** (String operation)
Method to perform mouse button click operations. [More...](#)

void **move** (**Quaternion** quaternion, boolean isInitQuat, float sensitivity)
Method to perform mouse move on the basis of 3-D input from the mobile device. [More...](#)

void **move** (int relativeX, int relativeY, float sensitivity)
Method to perform mouse move on the basis of 2-D input from the mobile device. [More...](#)

Detailed Description

Utility for performing mouse input operations.

It uses a `java.awt.Robot` object to generate native system input events. This differs from posting events to the AWT event queue or AWT components in that the events are generated in the platform's native input queue.

The **move(Quaternion, boolean, float)** and **move(int, int, float)** methods are invoked for moving the mouse pointer on the basis of 3-D and 2-D inputs from the mobile device respectively. The **move(Quaternion, boolean, float)** method, in turn, calls the **MouseMove#moveRelatively(Robot, int, int, float)** method. The mouse button inputs are performed by invoking the **doOperation(String)** method.

See also

`java.awt.Robot`

`project.pc.sensor.representation.Quaternion`

`project.pc.device.MouseMove::moveRelatively(Robot, int, int, float)`

Definition at line **37** of file **MouseController.java**.

Constructor Documentation

project.pc.device.MouseController.MouseController () throws `AWTException`

Constructor.

Initializes this `MouseController` by instantiating a `java.awt.Robot` object.

Exceptions

`AWTException`

Definition at line **52** of file **MouseController.java**.

Member Function Documentation

```
void project.pc.device.MouseController.doOperation ( String operation )
```

Method to perform mouse button click operations.

viz.

- Left click.
- Right click.
- Middle click.
- Scrolling

Parameters

operation String representing the type of operation.

Definition at line **103** of file **MouseController.java**.

```
void project.pc.device.MouseController.move ( Quaternion quaternion,  
boolean isInitQuat,  
float sensitivity  
)
```

Method to perform mouse move on the basis of 3-D input from the mobile device.

Parameters

quaternion a **project.pc.sensor.representation.Quaternion** object.

isInitQuat true, if quaternion is initial,
false, otherwise.

Definition at line **66** of file **MouseController.java**.

```
void project.pc.device.MouseController.move ( int relativeX,  
int relativeY,  
float sensitivity  
)
```

Method to perform mouse move on the basis of 2-D input from the mobile device.

Parameters

relativeX displacement along x-axis.

relativeY displacement along y-axis.

Definition at line **86** of file **MouseController.java**.

The documentation for this class was generated from the following file:

- src/project/pc/device/**MouseController.java**

project.pc.device.MouseMove Class Reference

Provides static methods used by [MouseController](#) . [More...](#)

Collaboration diagram for project.pc.device.MouseMove:

Static Package Functions

static void **moveRelatively** (Robot robot, int relativeX, int relativeY, float sensitivity)
Method for performing mouse movement proportional to the screen size. [More...](#)

Detailed Description

Provides static methods used by [MouseController](#) .

The **moveRelatively(Robot, int, int, float)** method takes into account the dimension of the screen and handles the input displacement in such a manner that the pointer movement is proportional to the screen size.

See also

[project.pc.device.MouseController](#)

Definition at line **21** of file [MouseMove.java](#).

Member Function Documentation

```
static void project.pc.device.MouseMove.moveRelatively ( Robot robot,
                                                         int   relativeX,
                                                         int   relativeY,
                                                         float  sensitivity
                                                         )
```

[static](#) [package](#)

Method for performing mouse movement proportional to the screen size.

This method is invoked by [MouseController#move\(int, int, float\)](#).

Parameters

- robot** the [java.awt.Robot](#) object.
- relativeX** displacement along x-axis.
- relativeY** displacement along y-axis.
- sensitivity** mouse movement sensitivity.

Definition at line **41** of file [MouseMove.java](#).

The documentation for this class was generated from the following file:

- src/project/pc/device/[MouseMove.java](#)

project.pc.gui.Cube Class Reference

Class to render multi-colored 3D cube. [More...](#)

Collaboration diagram for project.pc.gui.Cube:

Public Member Functions

void **display** (GLAutoDrawable drawable)
Displays the 3D cube. [More...](#)

void **init** (GLAutoDrawable drawable)
Initializes the canvas. [More...](#)

void **reshape** (GLAutoDrawable drawable, int x, int y, int width, int height)
Resizes the canvas. [More...](#)

Package Functions

Cube ()
Constructor. [More...](#)

boolean **isStarted** ()
Checks if rendering has been started. [More...](#)

void **showCube** ()
Renders the cube on canvas. [More...](#)

Detailed Description

Class to render multi-colored 3D cube.

It is used for demonstration purposes.

Definition at line **27** of file **Cube.java**.

Constructor Documentation

project.pc.gui.Cube.Cube ()

package

Constructor.

Initializes this Cube object.

Definition at line **38** of file **Cube.java**.

Member Function Documentation

void project.pc.gui.Cube.display (GLAutoDrawable **drawable)**

Displays the 3D cube.

Parameters

drawable a com.jogamp.opengl.GLAutoDrawable object used as the canvas.

Definition at line **47** of file **Cube.java**.

void project.pc.gui.Cube.init (GLAutoDrawable **drawable)**

Initializes the canvas.

Parameters

drawable com.jogamp.opengl.GLAutoDrawable object used as canvas.

Definition at line **109** of file **Cube.java**.

boolean project.pc.gui.Cube.isStarted ()

package

Checks if rendering has been started.

Returns

true, if currently rendering is in progress,
false, otherwise.

Definition at line **186** of file **Cube.java**.

```
void project.pc.gui.Cube.reshape ( GLAutoDrawable drawable,  
                                   int x,  
                                   int y,  
                                   int width,  
                                   int height  
                                   )
```

Resizes the canvas.

Parameters

drawable the canvas to be resized.

x x-coordinate of the lower left corner of the canvas.

y y-coordinate of the lower left corner of the canvas.

width width of the canvas.

height Height of the canvas.

Definition at line **134** of file **Cube.java**.

```
void project.pc.gui.Cube.showCube ( )
```

package

Renders the cube on canvas.

Definition at line **152** of file **Cube.java**.

The documentation for this class was generated from the following file:

- src/project/pc/gui/**Cube.java**

project.pc.gui.MainWindow Class Reference

Defines the main window of the application. [More...](#)

Collaboration diagram for project.pc.gui.MainWindow:

Public Member Functions

MainWindow (**NetworkManager** manager)

Constructor. [More...](#)

void **actionPerformed** (ActionEvent event)

Invoked when an action occurs. [More...](#)

void **windowClosing** (WindowEvent windowEvent)

Gracefully terminates the application. [More...](#)

Static Public Member Functions

static void **showConnectionConfirmationDialog** (String title, **ServerThread** st)

Shows a dialog window displaying the information that a client is successfully connected. [More...](#)

static void **showIncorrectPKeyDialog** (String title, **ServerThread** st)

Shows a dialog window displaying the error that an incorrect pairing key is sent by the client.
[More...](#)

static void **showPairingKeyDialog** (String title, String pairingKey, **ServerThread** st)

Shows a dialog window displaying the pairing key. [More...](#)

Detailed Description

Defines the main window of the application.

The contents of the main window are :

- Couple of radio buttons for server start and stop.
- A list displaying the clients currently connected.
- A disconnect button for disconnecting the selected client.
- A button form displaying a 3D demo.

It also implements `ActionListener` for these elements and `WindowListener` for graceful termination of the application upon closing of this `MainWindow`. The window is initialized and rendered at the time of instantiation by the constructor **MainWindow(NetworkManager)**.

See also

java.awt.event.ActionListener
java.awt.event.WindowListener
MainWindow(NetworkManager)

Definition at line 55 of file **MainWindow.java**.

Constructor Documentation

project.pc.gui.MainWindow.MainWindow (NetworkManager manager)

Constructor.

Initializes the MainWindow and displays it.

Parameters

manager the **project.pc.net.NetworkManager** for this network.

Definition at line 78 of file **MainWindow.java**.

Member Function Documentation

void project.pc.gui.MainWindow.actionPerformed (ActionEvent event)

Invoked when an action occurs.

Parameters

event a **java.awt.event.ActionEvent** object.

See also

java.awt.event.ActionEvent

Definition at line 218 of file **MainWindow.java**.

static void
project.pc.gui.MainWindow.showConnectionConfirmationDialog (String **title**,
ServerThread **st**
) static

Shows a dialog window displaying the information that a client is successfully connected.

Parameters

title title of the window - the canonical hostname of the client.
st the **project.pc.net.ServerThread** which handles this client.

See also

project.pc.net.ServerThread

Definition at line 194 of file **MainWindow.java**.

```
static void project.pc.gui.MainWindow.showIncorrectPKeyDialog ( String      title,
                                                             ServerThread st
                                                             )
```

static

Shows a dialog window displaying the error that an incorrect pairing key is sent by the client.

Parameters

title title of the window - the canonical hostname of the client.

st the `project.pc.net.ServerThread` which handles this client.

See also

`project.pc.net.ServerThread`

Definition at line **168** of file `MainWindow.java`.

```
static void project.pc.gui.MainWindow.showPairingKeyDialog ( String      title,
                                                             String      pairingKey,
                                                             ServerThread st
                                                             )
```

static

Shows a dialog window displaying the pairing key.

It gets closed when the pairing key is sent by the client for validation.

Parameters

title title of the dialog window - canonical hostname of the client for which the pairing key is meant.

pairingKey the pairing key.

st the `project.pc.net.ServerThread` which handles this client.

See also

`project.pc.net.ServerThread`

Definition at line **144** of file `MainWindow.java`.

```
void project.pc.gui.MainWindow.windowClosing ( WindowEvent windowEvent )
```

Gracefully terminates the application.

Termination is done by stopping the broadcast disconnecting every client and finally stopping the server. It is called during the closing of this `MainWindow`.

Parameters

windowEvent a `java.awt.event.WindowEvent` object.

See also

`java.awt.event.WindowEvent`

Definition at line **265** of file `MainWindow.java`.

The documentation for this class was generated from the following file:

- `src/project/pc/gui/MainWindow.java`

project.pc.net.BroadcastThread Class Reference

Implementation of the **java.lang Runnable** for the thread responsible for making UDP broadcasts. [More...](#)

Collaboration diagram for project.pc.net.BroadcastThread:

Public Member Functions

void **run** ()
Operations performed by this thread. [More...](#)

Package Functions

BroadcastThread (**ServerInfo** serverInfo)
Constructor. [More...](#)

boolean **getBroadcastFlag** ()
Returns the status of the broadcast. [More...](#)

long **getTimeout** ()
Returns the interval between two successive broadcasts. [More...](#)

void **stopBroadcast** ()
Stops the broadcast. [More...](#)

Detailed Description

Implementation of the **java.lang Runnable** for the thread responsible for making UDP broadcasts.

This thread continuously sends UDP broadcasts through all available network interfaces of the PC except point-to-point and loopback interfaces. The content of the broadcast data is a JSON-string of a **ServerInfo** object.

It is started during server start by the method **NetworkManager#startServer()**. It is stopped during server stop by the method **NetworkManager#stopServer()**. The interval between two successive broadcast is 1000 ms.

This UDP broadcast is used for device discovery from the Android devices.

See also

java.lang.Runnable
project.pc.net.ServerInfo
com.google.gson.Gson
project.pc.net.NetworkManager::startServer()
project.pc.net.NetworkManager::stopServer()

Definition at line **43** of file **BroadcastThread.java**.

Constructor Documentation

project.pc.net.BroadcastThread.BroadcastThread ([ServerInfo](#) **serverInfo)**

package

Constructor.

Initializes this thread.

Parameters

serverInfo the [ServerInfo](#) object containing the data to be broadcast.

See also

project.pc.net.ServerInfo

Definition at line **90** of file **BroadcastThread.java**.

Member Function Documentation

boolean project.pc.net.BroadcastThread.getBroadcastFlag ()

package

Returns the status of the broadcast.

Returns

true, if broadcast is active,
false, otherwise.

Definition at line **72** of file **BroadcastThread.java**.

long project.pc.net.BroadcastThread.getTimeout ()

package

Returns the interval between two successive broadcasts.

Returns

the interval between two successive broadcasts in milliseconds.

Definition at line **79** of file **BroadcastThread.java**.

void project.pc.net.BroadcastThread.run ()

Operations performed by this thread.

It continuously sends UDP broadcasts through all available network interfaces of the PC except point-to-point and loopback interface until **stopBroadcast()** is invoked.

Definition at line **103** of file **BroadcastThread.java**.

void project.pc.net.BroadcastThread.stopBroadcast ()

package

Stops the broadcast.

Used to exit from the **run()** method.

Definition at line **55** of file **BroadcastThread.java**.

The documentation for this class was generated from the following file:

- src/project/pc/net/**BroadcastThread.java**

project.pc.net.ClientInfo Class Reference

Encapsulation of the data received from the client at the time of establishing TCP connection. [More...](#)

Collaboration diagram for project.pc.net.ClientInfo:

Public Member Functions

ClientInfo (String clientInfo, byte[] publicKey)
Constructor. [More...](#)

String **getClientInfo** ()
Returns the canonical hostname of the client. [More...](#)

byte[] **getPublicKey** ()
Returns the public key of the client. [More...](#)

Socket **getSocket** ()
Returns the client Socket. [More...](#)

void **setSocket** (Socket socket)
Adds **java.net.Socket** of the client. [More...](#)

Detailed Description

Encapsulation of the data received from the client at the time of establishing TCP connection.

The data contains :

- Public key of the client
- Canonical hostname of the client
- Client Socket

See also

`java.net.Socket`

`project.pc.net.ServerThread`

Definition at line **22** of file **ClientInfo.java**.

Constructor Documentation

```
project.pc.net.ClientInfo.ClientInfo ( String clientInfo,  
                                         byte[] publicKey  
                                         )
```

Constructor.

Initializes this `ClientInfo` object. The stored object of `java.net.Socket` (client `Socket`) is not initialized. It can be set by the `setSocket(Socket)` method.

Parameters

clientInfo canonical hostname of the client.

publicKey public key of the client.

See also

`java.net.Socket`

`setSocket(Socket)`

Definition at line **42** of file `ClientInfo.java`.

Member Function Documentation

```
String project.pc.net.ClientInfo.getClientInfo ( )
```

Returns the canonical hostname of the client.

Returns

canonical hostname of the client

Definition at line **67** of file `ClientInfo.java`.

```
byte [] project.pc.net.ClientInfo.getPublicKey ( )
```

Returns the public key of the client.

Returns

public key of the client.

Definition at line **74** of file `ClientInfo.java`.

Socket project.pc.net.ClientInfo.getSocket ()

Returns the client Socket.

Returns

the client **java.net.Socket**.

Definition at line **60** of file **ClientInfo.java**.

void project.pc.net.ClientInfo.setSocket (Socket **socket)**

Adds **java.net.Socket** of the client.

Parameters

socket the client **java.net.Socket** object to be stored.

Definition at line **53** of file **ClientInfo.java**.

The documentation for this class was generated from the following file:

- src/project/pc/net/**ClientInfo.java**

project.pc.net.DataWrapper Class Reference

Encapsulates the data received from client during TCP communication. [More...](#)

Collaboration diagram for project.pc.net.DataWrapper:

Package Functions

DataWrapper (String operationType, String data)

Constructor. [More...](#)

DataWrapper (String data)

Constructor. [More...](#)

DataWrapper (int x, int y, float sensitivity)

Constructor. [More...](#)

DataWrapper (**Quaternion** quaternionObject, boolean **isInitQuat**, float sensitivity)

Constructor. [More...](#)

String **getData** ()

Returns the *data* field of this DataWrapper for keyboard and mouse button click operations. [More...](#)

String **getOperationType** ()

Returns the *type of operation* field of this DataWrapper. [More...](#)

Quaternion **getQuaternionObject** ()

Returns the **project.pc.sensor.representation.Quaternion** object stored in this DataWrapper. [More...](#)

float **getSensitivity** ()

Returns the mouse movement sensitivity value stored in this DataWrapper. [More...](#)

int **getX** ()

Returns the displacement value along x-axis stored in this DataWrapper. [More...](#)

int **getY** ()

Returns the displacement value along y-axis stored in this DataWrapper. [More...](#)

boolean **isInitQuat** ()

Checks if the stored **project.pc.sensor.representation.Quaternion** object is the initial one. [More...](#)

Static Package Functions

static String **getGsonString** (Object object)

Returns a JSon-string of an object. [More...](#)

Detailed Description

Encapsulates the data received from client during TCP communication.

This data contains all the information required by the **project.pc.device.MouseController** and the **project.pc.device.KeyboardController** for doing the different input operations. It contains a *type of operation* field which helps to determine the operation requested by the client and a *data* field which acts as the operand for the input operations.

For keyboard input, the *type of operation* field contains `Key` or `Special_Key`. The *data* field contains the sequence of characters and keyboard strokes for input. In case of special keys, the *data* field contains the special key identifier.

For 3-D mouse movement operation, the *type of operation* field contains `Mouse_Move` and the *data* field contains a **project.pc.sensor.representation.Quaternion** object.

For 2-D mouse movement operation, the *type of operation* field contains `Mouse_Touch` and the *data* field contains a pair of integers representing the displacement along the x-axis and y-axis relative to the current mouse pointer location.

For mouse button click operation, the *type of operation* field contains `Mouse_Button` and the *data* field contains any of the following :

- `left` - representing left click.
- `right` - representing right click.
- `middle` - representing middle button click.
- `upscroll` - representing scroll up.
- `downscroll` - representing scroll down.

See also

`project.pc.sensor.representation.Quaternion`

`project.pc.device.MouseController`

`project.pc.device.KeyboardController`

Definition at line **55** of file **DataWrapper.java**.

Constructor Documentation

```
project.pc.net.DataWrapper.DataWrapper ( String operationType,  
                                           String data  
                                           )
```

package

Constructor.

Used to create a `DataWrapper` for keyboard and mouse button click operations.

Parameters

operationType the *type of operation field*.

data the *data field*.

Definition at line **75** of file **DataWrapper.java**.

project.pc.net.DataWrapper.DataWrapper (String **data)**

package

Constructor.

Used to create a DataWrapper for special keys like Ctrl, Alt etc.

Parameters

data the *data* field.

Definition at line **87** of file **DataWrapper.java**.

```
project.pc.net.DataWrapper.DataWrapper ( int    x,  
                                     int    y,  
                                     float  sensitivity  
                                     )
```

package

Constructor.

Used to create a DataWrapper for 2-D mouse movement operation. The *type of operation* field is implicit.

Parameters

x displacement data along x-axis.

y displacement data along y-axis.

sensitivity mouse movement sensitivity.

Definition at line **102** of file **DataWrapper.java**.

```
project.pc.net.DataWrapper.DataWrapper ( Quaternion quaternionObject,  
                                     boolean   isInitQuat,  
                                     float    sensitivity  
                                     )
```

package

Constructor.

Used to create a DataWrapper for 3-D mouse movement operation. The *type of operation* field is implicit.

Parameters

quaternionObject a **project.pc.sensor.representation.Quaternion** object.

isInitQuat true, if quaternionObject is the initial,
false, otherwise.

sensitivity mouse movement sensitivity.

See also

project.pc.sensor.representation.Quaternion

Definition at line **123** of file **DataWrapper.java**.

Member Function Documentation

String `project.pc.net.DataWrapper.getData ( )`

package

Returns the *data* field of this DataWrapper for keyboard and mouse button click operations.

Returns

the *data* field.

Definition at line **155** of file **DataWrapper.java**.

static String `project.pc.net.DataWrapper.getGsonString ( Object object )`

staticpackage

Returns a JSon-string of an object.

Parameters

object the input `java.lang.Object`.

Returns

a JSon-string

See also

`com.google.gson.Gson`

Definition at line **137** of file **DataWrapper.java**.

String `project.pc.net.DataWrapper.getOperationType ( )`

package

Returns the *type of operation* field of this DataWrapper.

Returns

the *type of operation* field.

Definition at line **147** of file **DataWrapper.java**.

Quaternion `project.pc.net.DataWrapper.getQuaternionObject ( )`

package

Returns the `project.pc.sensor.representation.Quaternion` object stored in this DataWrapper.

Returns

the `project.pc.sensor.representation.Quaternion` object.

Definition at line **173** of file **DataWrapper.java**.

float `project.pc.net.DataWrapper.getSensitivity ( )`

package

Returns the mouse movement sensitivity value stored in this `DataWrapper`.

Returns

the mouse movement sensitivity.

Definition at line **197** of file **`DataWrapper.java`**.

int `project.pc.net.DataWrapper.getX ( )`

package

Returns the displacement value along x-axis stored in this `DataWrapper`.

Returns

the displacement along x-axis.

Definition at line **181** of file **`DataWrapper.java`**.

int `project.pc.net.DataWrapper.getY ( )`

package

Returns the displacement value along y-axis stored in this `DataWrapper`.

Returns

the displacement along y-axis.

Definition at line **189** of file **`DataWrapper.java`**.

boolean `project.pc.net.DataWrapper.isInitQuat ( )`

package

Checks if the stored **`project.pc.sensor.representation.Quaternion`** object is the initial one.

Returns

true, if it is initial,
false, otherwise.

Definition at line **164** of file **`DataWrapper.java`**.

The documentation for this class was generated from the following file:

- `src/project/pc/net/DataWrapper.java`

project.pc.net.NetworkManager Class Reference

Manages the different operations on the network. [More...](#)

Collaboration diagram for project.pc.net.NetworkManager:

Public Member Functions

NetworkManager (**ServerInfo** serverInfo)

Constructor. [More...](#)

void **disconnect** (InetAddress address) throws IOException, InterruptedException
Method to disconnect a client. [More...](#)

NetworkState **getNetworkState** ()

Returns the NetworkState of this network. [More...](#)

void **stopServer** () throws IOException, InterruptedException
Method to stop server. [More...](#)

Package Functions

void **startServer** () throws IOException, InterruptedException, AWTException
Method to start server. [More...](#)

Detailed Description

Manages the different operations on the network.

This class creates a **NetworkState** which stores the information regarding the network. It provides methods for start and stop of the server. It also provides method for disconnecting a client. These methods modify the NetworkState accordingly.

This class provides back-end of the network operations initiated by the GUI.

Definition at line **27** of file **NetworkManager.java**.

Constructor Documentation

project.pc.net.NetworkManager.NetworkManager (**ServerInfo serverInfo)**

Constructor.

This instantiates **NetworkState** and **BroadcastThread**. The BroadcastThread is not started in the constructor. It can be started using **startServer()**.

Parameters

serverInfo a **ServerInfo** object.

See also

project.pc.net.ServerInfo
project.pc.net.BroadcastThread
startServer()

Definition at line **47** of file **NetworkManager.java**.

Member Function Documentation

void **project.pc.net.NetworkManager.disconnect** (**InetAddress** **address**) **throws IOException, InterruptedException**

Method to disconnect a client.

It gets the **ServerThread** of the connection of the client to be disconnected from the network state by calling **NetworkState#getServerThread(InetAddress)**.

Parameters

address **java.net.InetAddress** of the client to be disconnected.

Exceptions

IOException

InterruptedException

See also

project.pc.net.ServerThread

NetworkState::getServerThread(InetAddress)

Definition at line **148** of file **NetworkManager.java**.

NetworkState **project.pc.net.NetworkManager.getNetworkState** ()

Returns the **NetworkState** of this network.

Returns

The **NetworkState** of this network

Definition at line **57** of file **NetworkManager.java**.

void **project.pc.net.NetworkManager.startServer** () **throws IOException, InterruptedException, AWTEException**

package

Method to start server.

This is achieved by executing two primary operations :

- Starting the **BroadcastThread**.
- Calling **Server#listen()**.

Exceptions

IOException

InterruptedException

AWTEException

See also

project.pc.net.BroadcastThread

Server::listen()

Definition at line **75** of file **NetworkManager.java**.

void project.pc.net.NetworkManager.stopServer () throws IOException, InterruptedException

Method to stop server.

This is achieved by executing three primary operations :

- Stopping BroadcastThread by calling **BroadcastThread#stopBroadcast()**.
- Calling **Server#setStopFlag()**.
- Disconnecting all clients. This is done by calling **disconnect(InetAddress)** for every connection stored in the network state.

Exceptions

IOException

InterruptedException

See also

project.pc.net.BroadcastThread

BroadcastThread::stopBroadcast()

Server::setStopFlag()

disconnect(InetAddress)

Definition at line **111** of file **NetworkManager.java**.

The documentation for this class was generated from the following file:

- src/project/pc/net/**NetworkManager.java**

project.pc.net.NetworkState Class Reference

Stores the state information about the network. [More...](#)

Collaboration diagram for project.pc.net.NetworkState:

Public Member Functions

DefaultListModel **getListModel** ()
Returns the **javax.swing.ListModel**. [More...](#)

boolean **isNoClientConnected** ()
Checks if any client is connected. [More...](#)

Package Functions

NetworkState ()
Constructor. [More...](#)

void **add** (**ClientInfo** clientInfo, **ServerThread** serverThread)
Adds a client connection to this NetworkState. [More...](#)

HashMap< Socket, **ServerThread** > **getConnectionMap** ()
Returns the *Connection Map*. [More...](#)

HashMap< InetAddress, Socket > **getAddressMap** ()
Returns the *Address Map*. [More...](#)

ServerThread **getServerThread** (InetAddress ia)
Returns the **ServerThread** for a client connection. [More...](#)

void **remove** (**ClientInfo** clientInfo)
Removes a client connection from this NetworkState.
[More...](#)

Detailed Description

Stores the state information about the network.

It stores the information about the client connections with the help of two data structures :

1. A *Connection map* which is a HashMap from a client Socket to the **ServerThread** handling that client.
2. An *Address map* which is a HashMap from the InetAddress of a client to the Socket.

The methods to manipulate these data structures are provided in the class. The **add(ClientInfo, ServerThread)** and **remove(ClientInfo)** methods are invoked by **ServerThread** once it

successfully establishes and closes a client connection respectively.

It also stores a **javax.swing.DefaultListModel** which is used by **project.pc.gui.MainWindow** to populate a list displaying the client connections in the GUI.

See also

`java.util.HashMap`

`project.pc.net.ServerThread`

`project.pc.gui.MainWindow`

`javax.swing.DefaultListModel`

Definition at line **41** of file **NetworkState.java**.

Constructor Documentation

project.pc.net.NetworkState.NetworkState ()

package

Constructor.

Initializes the state of the network.

Definition at line **52** of file **NetworkState.java**.

Member Function Documentation

void project.pc.net.NetworkState.add (ClientInfo **clientInfo,
 ServerThread **serverThread**
)**

package

Adds a client connection to this NetworkState.

Parameters

clientInfo the **ClientInfo**.

serverThread ServerThread of the client connection.

Definition at line **104** of file **NetworkState.java**.

HashMap<Socket, ServerThread> project.pc.net.NetworkState.getConnectionMap

() package

Returns the *Connection Map*.

Returns

HashMap<Socket, ServerThread>.

See also

java.util.HashMap

Definition at line **71** of file **NetworkState.java**.

DefaultListModel project.pc.net.NetworkState.getListModel ()

Returns the **javax.swing.ListModel**.

It is used by the GUI to populate a list displaying the client connections.

Returns

a **javax.swing.DefaultListModel**.

See also

project.pc.gui.MainWindow

Definition at line **63** of file **NetworkState.java**.

HashMap<InetAddress, Socket> project.pc.net.NetworkState.getAddressMap

() package

Returns the *Address Map*.

Returns

HashMap<InetAddress, Socket>.

See also

java.util.HashMap

Definition at line **79** of file **NetworkState.java**.

ServerThread project.pc.net.NetworkState.getServerThread (InetAddress ia)

package

Returns the **ServerThread** for a client connection.

Parameters

ia **java.net.InetAddress** of the client.

Returns

ServerThread of the client connection.

Definition at line **87** of file **NetworkState.java**.

boolean project.pc.net.NetworkState.isNoClientConnected ()

Checks if any client is connected.

Returns

true, if no client is connected,
false, otherwise.

Definition at line **95** of file **NetworkState.java**.

void project.pc.net.NetworkState.remove (ClientInfo clientInfo)

package

Removes a client connection from this NetworkState.

Parameters

clientInfo the **ClientInfo**.

Definition at line **117** of file **NetworkState.java**.

The documentation for this class was generated from the following file:

- src/project/pc/net/**NetworkState.java**

project.pc.net.NetworkThread Class Reference

Implementation of the [java.lang.Runnable](#) for the thread that executes the network operations. [More...](#)

Collaboration diagram for project.pc.net.NetworkThread:

Public Member Functions

NetworkThread (**NetworkManager** manager)
Constructor. [More...](#)

void **run** ()
Operations performed by this NetworkThread. [More...](#)

Detailed Description

Implementation of the [java.lang.Runnable](#) for the thread that executes the network operations.

This keeps away the network operations from the *Swing Event Dispatch Thread* and prevents the GUI from becoming unresponsive. This thread gets started by [project.pc.gui.MainWindow](#) while performing the action of starting the server - the first possible network operation.

See also

[java.lang.Runnable](#)
[project.pc.gui.MainWindow::actionPerformed\(ActionEvent\)](#)

Definition at line [22](#) of file [NetworkThread.java](#).

Constructor Documentation

project.pc.net.NetworkThread.NetworkThread ([NetworkManager](#) manager)

Constructor.

Initializes the thread.

Parameters

manager the [NetworkManager](#) for this network.

See also

[project.pc.net.NetworkManager](#)

Definition at line [34](#) of file [NetworkThread.java](#).

Member Function Documentation

void project.pc.net.NetworkThread.run ()

Operations performed by this NetworkThread.

Definition at line **40** of file **NetworkThread.java**.

The documentation for this class was generated from the following file:

- src/project/pc/net/**NetworkThread.java**

project.pc.net.Server Class Reference

Server data structure. [More...](#)

Collaboration diagram for project.pc.net.Server:

Package Functions

Server (**NetworkState** state, **BroadcastThread** broadcastThread) throws IOException, AWTException
Constructor. [More...](#)

void **close** () throws IOException
Closes the **ServerSocket** used by this **Server**. [More...](#)

int **getTimeout** () throws IOException
Returns the timeout of the **java.net.ServerSocket** used by this **Server**. [More...](#)

void **listen** () throws IOException, InterruptedException
Listens continuously on the *TCP PORT*. [More...](#)

void **setStopFlag** ()
Sets the stop flag. [More...](#)

Detailed Description

Server data structure.

It encapsulates the **java.net.ServerSocket** used for establishing TCP connection with the client.

When the **Server** starts, the **listen()** method is called which listens on the port in order to accept incoming client connections. Upon accepting connection it instantiates and starts a **ServerThread** for each connection.

It also instantiates the **project.pc.device.MouseController** and **project.pc.device.KeyboardController** which is used for mouse and keyboard control respectively.

See also

- java.net.ServerSocket
- project.pc.net.ServerThread
- listen()
- project.pc.device.MouseController
- project.pc.device.KeyboardController

Definition at line **40** of file **Server.java**.

Constructor Documentation

```
project.pc.net.Server.Server ( NetworkState state,  
                             BroadcastThread broadcastThread  
                             ) throws IOException,  
                             AWTException
```

package

Constructor.

Initializes the server.

Parameters

state **NetworkState** of this network.
broadcastThread the **BroadcastThread**.

Exceptions

IOException
AWTException

Definition at line **61** of file **Server.java**.

Member Function Documentation

```
void project.pc.net.Server.close ( ) throws IOException
```

package

Closes the ServerSocket used by this Server.

Exceptions

IOException

Definition at line **123** of file **Server.java**.

```
int project.pc.net.Server.getTimeout ( ) throws IOException
```

package

Returns the timeout of the **java.net.ServerSocket** used by this Server.

Returns

the timeout in milliseconds of this Server.

Exceptions

IOException

Definition at line **87** of file **Server.java**.

void project.pc.net.Server.listen () throws IOException,InterruptedException

package

Listens continuously on the *TCP PORT*.

Once the server accepts a connection, a separate **ServerThread** is started for each connection. The *listening* can be stopped by invoking the **setStopFlag()** method.

Exceptions

IOException

InterruptedException

See also

project.pc.net.ServerThread

setStopFlag()

Definition at line **104** of file **Server.java**.

void project.pc.net.Server.setStopFlag ()

package

Sets the stop flag.

Used to exit from the **listen()** method.

See also

listen()

Definition at line **78** of file **Server.java**.

The documentation for this class was generated from the following file:

- src/project/pc/net/**Server.java**

project.pc.net.ServerInfo Class Reference

Encapsulation of the data sent during UDP broadcast. [More...](#)

Collaboration diagram for project.pc.net.ServerInfo:

Public Member Functions

ServerInfo (byte[] publicKey, String serverInfo)
Constructor. [More...](#)

boolean **equals** (Object obj)
Compares two ServerInfo objects. [More...](#)

Package Functions

void **clearStopFlag** ()
Sets the status of the *broadcast* as *active*. [More...](#)

void **setStopFlag** ()
Sets the status of the *broadcast* as *inactive*. [More...](#)

Detailed Description

Encapsulation of the data sent during UDP broadcast.

This data is broadcast by the **BroadcastThread**. This contains :

- the public key of the server,
- status of the broadcast, and
- address of the server

This message helps a client to discover a server.

See also

project.pc.net.BroadcastThread

Definition at line **24** of file **ServerInfo.java**.

Constructor Documentation

```
project.pc.net.ServerInfo.ServerInfo ( byte[] publicKey,  
                                         String serverInfo  
                                         )
```

Constructor.

Initializes the ServerInfo.

Parameters

publicKey public key of the server generated by
`project.pc.security.EKEProvider#getBase64EncodedPubKey()`.
serverInfo canonical hostname of the server.

See also

`project.pc.security.EKEProvider::getBase64EncodedPubKey()`

Definition at line **40** of file [ServerInfo.java](#).

Member Function Documentation

```
void project.pc.net.ServerInfo.clearStopFlag ( )
```

package

Sets the status of the *broadcast* as *active*.

Definition at line **72** of file [ServerInfo.java](#).

```
boolean project.pc.net.ServerInfo.equals ( Object obj )
```

Compares two ServerInfo objects.

Parameters

obj the object to be compared.

Returns

true, if the two objects are equal,
false, otherwise.

Definition at line **54** of file [ServerInfo.java](#).

void project.pc.net.ServerInfo.setStopFlag ()

package

Sets the status of the *broadcast* as *inactive*.

Definition at line **67** of file **ServerInfo.java**.

The documentation for this class was generated from the following file:

- src/project/pc/net/**ServerInfo.java**

project.pc.net.ServerThread Class Reference

Implementation of the **java.lang.Runnable** for the thread that is responsible for communication with each client. [More...](#)

Collaboration diagram for project.pc.net.ServerThread:

Public Member Functions

- `JDialog` **getDialog** ()
Returns the **javax.swing.JDialog** object used for displaying notifications. [More...](#)
- `void` **run** ()
Performs operations for communication with a client. [More...](#)
- `void` **setDialog** (JDialog d)
Sets a dialog window. [More...](#)
-

Package Functions

- ServerThread** (**NetworkState** state, Socket skt, **MouseController** mc, **KeyboardController** kc, **BroadcastThread** bt) throws IOException
Constructor. [More...](#)
- `int` **getTimeout** () throws IOException
Returns the timeout of the client Socket. [More...](#)
- `void` **setStopFlag** ()
Sets stop flag. [More...](#)
-

Detailed Description

Implementation of the **java.lang.Runnable** for the thread that is responsible for communication with each client.

The actions performed by this thread are :

- Authenticate the client.
- According to the result
 - If the client is not authentic, close the connection and terminate.
 - Otherwise, confirm the connection and start communication with the client.
- Examine the data received from client and transfer it to a *controller* (**project.pc.device.MouseController** or **project.pc.device.KeyboardController**) to execute the input operations.

Client authentication is done by comparing the pairing key received from the client with the one displayed on the pairing key dialog window.

It receives data from client and wraps it in a **DataWrapper** object. It, then, invokes the **DataWrapper#getOperationType()** method to examine the type of operation requested by the client and accordingly transfers the control to either **project.pc.device.MouseController** or **project.pc.device.KeyboardController**.

This thread is started by the **Server#listen()** upon accepting every client connection. It gets terminated implicitly after unsuccessful authentication of the client. Explicitly, it is terminated by the **NetworkManager (NetworkManager#stopServer())** and **NetworkManager#disconnect(InetAddress))** by invoking the **setStopFlag()** method at the time of server stop or disconnection of this client A reference of this thread gets stored in the **NetworkState** after successful authentication of the client.

See also

java.lang.Runnable
project.pc.net.DataWrapper
project.pc.net.DataWrapper::getOperationType()
project.pc.device.MouseController
project.pc.device.KeyboardController
project.pc.net.NetworkManager::stopServer()
project.pc.net.NetworkManager::disconnect(InetAddress)

Definition at line **80** of file **ServerThread.java**.

Constructor Documentation

```
project.pc.net.ServerThread.ServerThread ( NetworkState      state,  
                                           Socket          skt,  
                                           MouseController mc,  
                                           KeyboardController kc,  
                                           BroadcastThread bt  
                                           ) throws IOException
```

package

Constructor.

Initializes the thread.

Parameters

state **NetworkState** of this network.
skt client **Socket**.
mc the **project.pc.device.MouseController**.
kc the **project.pc.device.KeyboardController**.
bt the **BroadcastThread**.

Exceptions

IOException

Definition at line **107** of file **ServerThread.java**.

Member Function Documentation

JDialog `project.pc.net.ServerThread.getDialog ( )`

Returns the `javax.swing.JDialog` object used for displaying notifications.

Returns

a `javax.swing.JDialog` object.

Definition at line **300** of file `ServerThread.java`.

int `project.pc.net.ServerThread.getTimeout ( )` throws `IOException`

package

Returns the timeout of the client Socket.

Returns

timeout of client Socket in milliseconds.

Exceptions

`IOException`

Definition at line **308** of file `ServerThread.java`.

void `project.pc.net.ServerThread.run ( )`

Performs operations for communication with a client.

- Validates the client,
- Receives data from client and encapsulates it in a `DataWrapper` object, and
- Examines the type of operation requested by the client and accordingly transfers the control to a *controller* (`project.pc.device.MouseController` or `project.pc.device.KeyboardController`).

See also

`project.pc.net.DataWrapper`

`project.pc.device.MouseController`

`project.pc.device.KeyboardController`

Definition at line **194** of file `ServerThread.java`.

void `project.pc.net.ServerThread.setDialog ( JDialog d )`

Sets a dialog window.

Parameters

d a `javax.swing.JDialog` object.

Definition at line **292** of file `ServerThread.java`.

void project.pc.net.ServerThread.setStopFlag ()

package

Sets stop flag.

Used to exit from the **run()** method.

Definition at line **285** of file **ServerThread.java**.

The documentation for this class was generated from the following file:

- src/project/pc/net/**ServerThread.java**

project.pc.security.EKEProvider Class Reference

Utility for the end-to-end security module. [More...](#)

Collaboration diagram for project.pc.security.EKEProvider:

Public Member Functions

EKEProvider ()

Constructor. [More...](#)

EKEProvider (String pairingKey, byte[] clientPublicKey)

Constructor. [More...](#)

String **decryptString** (String cipherText)

Method to decrypt cipher text using *AES-256 bit* decryption algorithm. [More...](#)

String **encryptString** (String plainText)

Method to encrypt plaintext using *AES 256-bit* encryption algorithm. [More...](#)

byte[] **getBase64EncodedPubKey** ()

Returns the Base64 encoded server public key. [More...](#)

Static Public Member Functions

static String **getPairingKey** ()

Generates and returns a randomly generated pairing key. [More...](#)

Detailed Description

Utility for the end-to-end security module.

This class provides the following features:

- *256-bit Elliptic Curve* private and public key generation.
- *AES 256-bit* plaintext encryption.
- *AES 256-bit* cipher text decryption.
- Self-signed certificate generation.

The private-public key pair and the self-signed certificate is generated by the constructor during instantiation and is stored in a keystore (**java.security.KeyStore**) maintained by the **KeyStoreManager**. If a KeyStore already exists, the key pair and the certificate gets loaded directly. The encryption and decryption mechanism is provided by the **encryptString(String)** and **decryptString(String)** methods respectively.

See also

project.pc.security.KeyStoreManager

java.security.KeyStore

java.security.KeyPair

project.pc.security.EKEProvider::encryptString(String)

project.pc.security.EKEProvider::decryptString(String)

Definition at line **84** of file **EKEProvider.java**.

Constructor Documentation

project.pc.security.EKEProvider.EKEProvider ()

Constructor.

Initializes this EKEProvider. If keystore exists, private and public **java.security.KeyPair** is loaded, otherwise KeyPair is generated.

See also

project.pc.security.KeyStoreManager

Definition at line **117** of file **EKEProvider.java**.

project.pc.security.EKEProvider.EKEProvider (String pairingKey, byte[] clientPublicKey)

Constructor.

Initializes this EKEProvider. Generates a secret key using the *SHA-256 message digest* of the pairing key generated by **EKEProvider#getPairingKey()** and the public key of the client.

Parameters

pairingKey Pairing key used for client-server authentication.

clientPublicKey Public key of the client.

See also

project.pc.security.KeyStoreManager

Definition at line **155** of file **EKEProvider.java**.

Member Function Documentation

String project.pc.security.EKEProvider.decryptString (String cipherText)

Method to decrypt cipher text using *AES-256 bit* decryption algorithm.

Parameters

cipherText cipher text to be decrypted.

Returns

the plaintext.

See also

project.pc.security.EKEProvider::encryptString(String)

Definition at line **268** of file **EKEProvider.java**.

String project.pc.security.EKEProvider.encryptString (String *plainText*)

Method to encrypt plaintext using *AES 256-bit* encryption algorithm.

Parameters

plainText plaintext to be encrypted.

Returns

the cipher text.

See also

project.pc.security.EKEProvider::decryptString(String)

Definition at line **245** of file **EKEProvider.java**.

byte [] project.pc.security.EKEProvider.getBase64EncodedPubKey ()

Returns the Base64 encoded server public key.

Returns

the Base64 encoded server public key.

Note: Returns null if no public has been generated during instantiation.

Definition at line **206** of file **EKEProvider.java**.

static String project.pc.security.EKEProvider.getPairingKey ()

static

Generates and returns a randomly generated pairing key.

Returns

the pairing key.

Definition at line **194** of file **EKEProvider.java**.

The documentation for this class was generated from the following file:

- src/project/pc/security/**EKEProvider.java**

project.pc.security.KeyStoreManager Class Reference

Manages the operations on the KeyStore. [More...](#)

Collaboration diagram for project.pc.security.KeyStoreManager:

Package Functions

KeyStoreManager () throws `CertificateException`, `NoSuchAlgorithmException`, `KeyStoreException`, `IOException`
Constructor. [More...](#)

void **setMasterKey** (`PrivateKey` privateKey, `Certificate...`cert) throws `KeyStoreException`, `IOException`, `CertificateException`, `NoSuchAlgorithmException`
Method to store the private key and the certificate in the KeyStore. [More...](#)

Static Package Functions

static `KeyPair` **getKSKeyPair** () throws `IOException`, `KeyStoreException`, `CertificateException`, `NoSuchAlgorithmException`, `UnrecoverableEntryException`
Returns the private-public key pair. [More...](#)

static boolean **keyStoreExists** ()
Checks if a KeyStore exists. [More...](#)

Detailed Description

Manages the operations on the KeyStore.

This class provides a **java.security.KeyStore** which stores :

- The private key (**java.security.PrivateKey**).
- A self-signed certificate (**java.security.cert.Certificate**) that contains the public key and the digital signature.

The KeyStore is protected by a randomly generated password.

See also

`project.pc.security.EKEProvider`
`java.security.KeyStore`
`java.security.KeyPair`
`java.security.cert.Certificate`

Definition at line **43** of file **KeyStoreManager.java**.

Constructor Documentation

throws `CertificateException`,
`NoSuchAlgorithmException`,
`KeyStoreException`,
`IOException`

project.pc.security.KeyStoreManager.KeyStoreManager () package

Constructor.

Initializes this `KeyStoreManager`.

Exceptions

- `CertificateException`
- `NoSuchAlgorithmException`
- `KeyStoreException`
- `IOException`

Definition at line **62** of file **KeyStoreManager.java**.

Member Function Documentation

throws `IOException`,
`KeyStoreException`,
`CertificateException`,
`NoSuchAlgorithmException`,
`UnrecoverableEntryException`

static KeyPair
project.pc.security.KeyStoreManager.getKSKeyPair () static package

Returns the private-public key pair.

Returns

- `java.security.KeyPair` object.

Exceptions

- `IOException`
- `KeyStoreException`
- `CertificateException`
- `NoSuchAlgorithmException`
- `UnrecoverableEntryException`

Definition at line **94** of file **KeyStoreManager.java**.

static boolean **project.pc.security.KeyStoreManager.keyStoreExists ()** static package

Checks if a `KeyStore` exists.

Returns

- `true`, if the `KeyStore` exists,
- `false`, otherwise.

Definition at line **114** of file **KeyStoreManager.java**.


```
void
project.pc.security.KeyStoreManager.setMasterKey ( PrivateKey  privateKey,
                                                    Certificate... cert
                                                    throws KeyStoreException,
                                                    IOException,
                                                    CertificateException,
                                                    NoSuchAlgorithmException )
```

Method to store the private key and the certificate in the KeyStore.

Parameters

privateKey the `java.security.PrivateKey` object.
cert the `java.security.cert.Certificate` object(s).

Exceptions

`KeyStoreException`
`IOException`
`CertificateException`
`NoSuchAlgorithmException`

Definition at line **78** of file **KeyStoreManager.java**.

The documentation for this class was generated from the following file:

- `src/project/pc/security/KeyStoreManager.java`

project.pc.sensor.representation.Cartesian2D Class Reference

Class to represent a point in 2-D Cartesian coordinate system. [More...](#)

Collaboration diagram for project.pc.sensor.representation.Cartesian2D:

Public Member Functions

Cartesian2D (double x, double y)
Constructor. [More...](#)

double **getX** ()
Returns the x-coordinate of this point. [More...](#)

double **getY** ()
Returns the y-coordinate of this point. [More...](#)

void **setX** (double x)
Sets the x-coordinate of this point. [More...](#)

void **setY** (double y)
Sets the x-coordinate of this point. [More...](#)

Detailed Description

Class to represent a point in 2-D Cartesian coordinate system.

Definition at line **6** of file **Cartesian2D.java**.

Constructor Documentation

```
project.pc.sensor.representation.Cartesian2D.Cartesian2D ( double x,  
                                                         double y  
                                                         )
```

Constructor.

Initializes this Cartesian2D point.

Parameters

x the x-coordinate.

y the y-coordinate.

Definition at line **17** of file **Cartesian2D.java**.

Member Function Documentation

double project.pc.sensor.representation.Cartesian2D.getX ()

Returns the x-coordinate of this point.

Returns

thw x-coordinate.

Definition at line **27** of file **Cartesian2D.java**.

double project.pc.sensor.representation.Cartesian2D.getY ()

Returns the y-coordinate of this point.

Returns

the y-coordinate.

Definition at line **34** of file **Cartesian2D.java**.

void project.pc.sensor.representation.Cartesian2D.setX (double **x)**

Sets the x-coordinate of this point.

Parameters

x the x-coordinate.

Definition at line **41** of file **Cartesian2D.java**.

void project.pc.sensor.representation.Cartesian2D.setY (double **y)**

Sets the x-coordinate of this point.

Parameters

y the x-coordinate.

Definition at line **48** of file **Cartesian2D.java**.

The documentation for this class was generated from the following file:

- src/project/pc/sensor/representation/**Cartesian2D.java**

project.pc.sensor.representation.Matrix Class Reference

Matrix math utilities. [More...](#)

Collaboration diagram for project.pc.sensor.representation.Matrix:

Static Public Member Functions

static void **frustumM** (float[] m, int offset, float left, float right, float bottom, float top, float near, float far)
Defines a projection matrix in terms of six clip planes. [More...](#)

static boolean **invertM** (float[] mInv, int mInvOffset, float[] m, int mOffset)
Inverts a 4 x 4 matrix. [More...](#)

static void **multiplyMM** (float[] output, float[] lhs, float[] rhs)
Multiplies two 4x4 matrices. [More...](#)

static void **multiplyMV** (float[] output, int outputOffset, float[] lhs, int lhsOffset, float[] rhs, int rhsOffset)
Multiplies a 4 element vector by a 4x4 matrix. [More...](#)

static void **orthoM** (float[] m, int mOffset, float left, float right, float bottom, float top, float near, float far)
Computes an orthographic projection matrix. [More...](#)

static void **perspectiveM** (float[] m, int offset, float fovy, float aspect, float zNear, float zFar)
Defines a projection matrix in terms of a field of view angle, an aspect ratio, and Z clip planes.
[More...](#)

static void **rotateM** (float[] rm, int rmOffset, float[] m, int mOffset, float a, float x, float y, float z)
Rotates matrix m by angle a (in degrees) around the axis (x, y, z). [More...](#)

static void **rotateM** (float[] m, int mOffset, float a, float x, float y, float z)
Rotates matrix m in place by angle a (in degrees) around the axis (x, y, z). [More...](#)

static void **scaleM** (float[] sm, int smOffset, float[] m, int mOffset, float x, float y, float z)
Scales matrix m by x, y, and z. [More...](#)

static void **setRotateEulerM** (float[] rm, int rmOffset, float x, float y, float z)
Converts Euler angles to a rotation matrix. [More...](#)

static void **translateM** (float[] tm, int tmOffset, float[] m, int mOffset, float x, float y, float z)
Translates matrix m by x, y, and z. [More...](#)

static void **transposeM** (float[] mTrans, int mTransOffset, float[] m, int mOffset)
Transposes a 4 x 4 matrix. [More...](#)

Static Package Functions

static void **setIdentityM** (float[] sm, int smOffset)
Sets matrix to the identity matrix. [More...](#)

Detailed Description

Matrix math utilities.

These methods operate on OpenGL ES format matrices and vectors stored in float arrays. Matrices are 4 x 4 column-vector(float) matrices stored in column-major order:

```
m[offset + 0]    m[offset + 4]    m[offset + 8]    m[offset + 12]
m[offset + 1]    m[offset + 5]    m[offset + 9]    m[offset + 13]
m[offset + 2]    m[offset + 6]    m[offset + 10]   m[offset + 14]
m[offset + 3]    m[offset + 7]    m[offset + 11]   m[offset + 15]
```

Vectors are 4 row x 1 column column-vectors stored in order:

```
v[offset + 0]
v[offset + 1]
v[offset + 2]
v[offset + 3]
```

See also

`project.pc.sensor.representation.MatrixF4x4`

Definition at line [31](#) of file [Matrix.java](#).

Member Function Documentation

```
static void project.pc.sensor.representation.Matrix.frustumM ( float[] m,
                                                             int    offset,
                                                             float  left,
                                                             float  right,
                                                             float  bottom,
                                                             float  top,
                                                             float  near,
                                                             float  far
                                                             )
```

static

Defines a projection matrix in terms of six clip planes.

Parameters

- m** the float array that holds the perspective matrix.
- offset** the offset into m where the perspective matrix data is written.
- left** left clipping value.
- right** right clipping value.
- bottom** bottom clipping value.
- top** top clipping value.
- near** nearer depth clipping value.
- far** farther depth clipping value.

Definition at line [334](#) of file [Matrix.java](#).

```
static boolean project.pc.sensor.representation.Matrix.invertM ( float[] mInv,
                                                                int    mInvOffset,
                                                                float[] m,
                                                                int    mOffset
                                                                )
```

static

Inverts a 4 x 4 matrix.

Parameters

- mInv** the array that holds the output inverted matrix.
- mInvOffset** an offset into mInv where the inverted matrix is stored.
- m** the input array.
- mOffset** an offset into m where the matrix is stored.

Returns

- true, if the matrix could be inverted,
- false, otherwise.

Definition at line **155** of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.multiplyMM ( float[] output,
                                                                float[] lhs,
                                                                float[] rhs
                                                                )
```

static

Multiplies two 4x4 matrices.

Parameters

- output** the float array that holds the result.
- lhs** the float array that holds the left-hand-side matrix.
- rhs** the float array that holds the right-hand-side matrix.

Exceptions

- IllegalArgumentException

Definition at line **66** of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.multiplyMV ( float[] output,
                                                                int    outputOffset,
                                                                float[] lhs,
                                                                int    lhsOffset,
                                                                float[] rhs,
                                                                int    rhsOffset
                                                                )
```

static

Multiplies a 4 element vector by a 4x4 matrix.

It stores the result in a 4 element column vector. In matrix notation:

$output = lhs \times rhs$

Parameters

- output** the float array that holds the result vector.
- outputOffset** the offset into the result array where the result vector is stored.
- lhs** the float array that holds the left-hand-side matrix.
- lhsOffset** the offset into the lhs array.
- rhs** the float array that holds the right-hand-side vector.
- rhsOffset** the offset into the rhs array.

Exceptions

`IllegalArgumentException`

Definition at line **105** of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.orthoM ( float[] m,
                                                            int    mOffset,
                                                            float   left,
                                                            float   right,
                                                            float   bottom,
                                                            float   top,
                                                            float   near,
                                                            float   far
                                                            )
```

static

Computes an orthographic projection matrix.

Parameters

- m** returns the result.
- mOffset** offset value.
- left** left clipping value.
- right** right clipping value.
- bottom** bottom clipping value.
- top** top clipping value.
- near** nearer depth clipping value.
- far** farther depth clipping value.

Definition at line **283** of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.perspectiveM ( float[] m,
                                                                    int    offset,
                                                                    float  fovy,
                                                                    float  aspect,
                                                                    float  zNear,
                                                                    float  zFar
                                                                    )
```

static

Defines a projection matrix in terms of a field of view angle, an aspect ratio, and Z clip planes.

Parameters

- m** the float array that holds the perspective matrix.
- offset** the offset into m where the perspective matrix data is written.
- fovy** field of view in y direction, in degrees.
- aspect** width to height aspect ratio of the viewport.
- zNear** nearer depth clipping value on Z-plane.
- zFar** farther depth clipping value on Z-plane.

Definition at line **391** of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.rotateM ( float[] rm,
                                                                int    rmOffset,
                                                                float[] m,
                                                                int    mOffset,
                                                                float  a,
                                                                float  x,
                                                                float  y,
                                                                float  z
                                                                )
```

static

Rotates matrix m by angle a (in degrees) around the axis (x, y, z).

Parameters

- rm** returns the result.
- rmOffset** index into rm where the result matrix starts.
- m** source matrix.
- mOffset** index into m where the source matrix starts.
- a** angle to rotate in degrees.
- x** scale factor x.
- y** scale factor y.
- z** scale factor z.

Definition at line **505** of file **Matrix.java**.


```
static void project.pc.sensor.representation.Matrix.rotateM ( float[] m,
                                                             int    mOffset,
                                                             float  a,
                                                             float  x,
                                                             float  y,
                                                             float  z
                                                             )
```

static

Rotates matrix **m** in place by angle **a** (in degrees) around the axis (**x**, **y**, **z**).

Parameters

- m** source matrix.
- mOffset** index into **m** where the matrix starts.
- a** angle to rotate in degrees.
- x** scale factor **x**.
- y** scale factor **y**.
- z** scale factor **z**.

Definition at line **523** of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.scaleM ( float[] sm,
                                                             int    smOffset,
                                                             float[] m,
                                                             int    mOffset,
                                                             float  x,
                                                             float  y,
                                                             float  z
                                                             )
```

static

Scales matrix **m** by **x**, **y**, and **z**.

Parameters

- sm** the result.
- smOffset** index into **sm** where the result matrix starts.
- m** source matrix.
- mOffset** index into **m** where the source matrix starts.
- x** scale factor **x**.
- y** scale factor **y**.
- z** scale factor **z**.

Definition at line **446** of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.setIdentityM ( float[] sm,
                                                                int    smOffset
                                                                )
```

static package

Sets matrix to the identity matrix.

Parameters

sm the result.
smOffset index into sm where the result matrix starts.

Definition at line 426 of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.setRotateEulerM ( float[] rm,
                                                                    int    rmOffset,
                                                                    float  x,
                                                                    float  y,
                                                                    float  z
                                                                    )
```

static

Converts Euler angles to a rotation matrix.

Parameters

rm the result.
rmOffset index into rm where the result matrix starts.
x angle of rotation, in degrees.
y angle of rotation, in degrees.
z angle of rotation, in degrees.

Definition at line 608 of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.translateM ( float[] tm,
                                                                int    tmOffset,
                                                                float[] m,
                                                                int    mOffset,
                                                                float  x,
                                                                float  y,
                                                                float  z
                                                                )
```

static

Translates matrix m by x, y, and z.

Parameters

tm the result.
tmOffset index into sm where the result matrix starts.
m source matrix.
mOffset index into m where the source matrix starts.
x translation factor x.
y translation factor y.
z translation factor z.

Definition at line 477 of file **Matrix.java**.

```
static void project.pc.sensor.representation.Matrix.transposeM ( float[] mTrans,  
                                                                int    mTransOffset,  
                                                                float[] m,  
                                                                int    mOffset  
                                                                )
```

static

Transposes a 4 x 4 matrix.

Parameters

mTrans the array that holds the output inverted matrix.
mTransOffset an offset where the inverted matrix is stored.
m the input array.
mOffset an offset into m where the matrix is stored.

Definition at line [134](#) of file [Matrix.java](#).

The documentation for this class was generated from the following file:

- [src/project/pc/sensor/representation/Matrix.java](#)

project.pc.sensor.representation.MatrixF4x4 Class Reference

Class to represent a 4x4 matrix (float). [More...](#)

Collaboration diagram for project.pc.sensor.representation.MatrixF4x4:

Public Member Functions

MatrixF4x4 ()

Constructor. [More...](#)

void **multiplyMatrix4x4ByMatrix** (**MatrixF4x4** matrixf)

Multiplies a MatrixF4x4 with this. [More...](#)

void **multiplyVector3fByMatrix** (**Vector3f** vector) throws InvalidParameterSpecException

Multiplies the given vector by this MatrixF4x4. [More...](#)

void **multiplyVector4fByMatrix** (**Vector4f** vector) throws InvalidParameterSpecException

Multiplies the given vector by this MatrixF4x4. [More...](#)

void **set** (**MatrixF4x4** source)

Sets the matrix from a float[16] array. [More...](#)

Package Functions

float[] **getMatrix** ()

Returns the matrix. [More...](#)

void **setW0** (float value)

Sets the value of W0. [More...](#)

void **setW1** (float value)

Sets the value of W1. [More...](#)

void **setW2** (float value)

Sets the value of W2. [More...](#)

void **setW3** (float value)

Sets the value of W3. [More...](#)

void **setX0** (float value)

Sets the value of X0. [More...](#)

void **setX1** (float value)

Sets the value of X1. [More...](#)

void **setX2** (float value)

Sets the value of X2. [More...](#)

void **setX3** (float value)

Sets the value of X3. [More...](#)

void **setY0** (float value)

Sets the value of Y0. [More...](#)

void **setY1** (float value)

Sets the value of Y1. [More...](#)

void **setY2** (float value)

Sets the value of Y2. [More...](#)

void **setY3** (float value)

Sets the value of Y3. [More...](#)

void **setZ0** (float value)

Sets the value of Z0. [More...](#)

void	setZ1 (float value)	Sets the value of Z1. More...
void	setZ2 (float value)	Sets the value of Z2. More...
void	setZ3 (float value)	Sets the value of Z3. More...
int	size ()	Returns size of the matrix. More...

Detailed Description

Class to represent a 4x4 matrix (float).

Internal the matrix is structured as :

```
[ x0 , y0 , z0 , w0 ]
[ x1 , y1 , z1 , w1 ]
[ x2 , y2 , z2 , w2 ]
[ x3 , y3 , z3 , w3 ]
```

See also

[project.pc.sensor.representation.Matrix](#)

Definition at line **21** of file [MatrixF4x4.java](#).

Constructor Documentation

project.pc.sensor.representation.MatrixF4x4.MatrixF4x4 ()

Constructor.

Initializes this MatrixF4x4.

Definition at line **40** of file [MatrixF4x4.java](#).

Member Function Documentation

float [] project.pc.sensor.representation.MatrixF4x4.getMatrix ()

package

Returns the matrix.

Returns

the matrix,
null, if the matrix is invalid.

Definition at line **51** of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.multiplyMatrix4x4ByMatrix (MatrixF4x4 matrixf)

Multiples a MatrixF4x4 with this.

Parameters

matrixf a MatrixF4x4.

Definition at line **186** of file [MatrixF4x4.java](#).

void **throws**
project.pc.sensor.representation.MatrixF4x4.multiplyVector3fByMatrix (Vector3f vector) InvalidParameterSpecException

Multiplies the given vector by this MatrixF4x4.

This should only be used if the matrix is of size 9.

Parameters

vector a Vector3f.

Exceptions

InvalidParameterSpecException

Definition at line 146 of file MatrixF4x4.java.

void **throws**
project.pc.sensor.representation.MatrixF4x4.multiplyVector4fByMatrix (Vector4f vector) InvalidParameterSpecException

Multiplies the given vector by this MatrixF4x4.

This should only be used if the matrix is of size 16.

Parameters

vector a Vector4f.

Exceptions

InvalidParameterSpecException

Definition at line 99 of file MatrixF4x4.java.

void project.pc.sensor.representation.MatrixF4x4.set (MatrixF4x4 source)

Sets the matrix from a float[16] array.

If the matrix isn't 16 long then the matrix will be invalid.

Parameters

source the source matrixF4x4.

Definition at line 79 of file MatrixF4x4.java.

void project.pc.sensor.representation.MatrixF4x4.setW0 (float value)

package

Sets the value of W0.

Parameters

value value to set.

Definition at line 489 of file MatrixF4x4.java.

void project.pc.sensor.representation.MatrixF4x4.setW1 (float value)

package

Sets the value of W1.

Parameters

value value to set.

Definition at line 507 of file MatrixF4x4.java.

void project.pc.sensor.representation.MatrixF4x4.setW2 (float value)

package

Sets the value of W2.

Parameters

value value to set.

Definition at line 525 of file MatrixF4x4.java.

void project.pc.sensor.representation.MatrixF4x4.setW3 (float value)

package

Sets the value of W3.

Parameters

value value to set.

Definition at line 543 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setX0 (float value)

package

Sets the value of X0.

Parameters

value value to set.

Definition at line 237 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setX1 (float value)

package

Sets the value of X1.

Parameters

value value to set.

Definition at line 259 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setX2 (float value)

package

Sets the value of X2.

Parameters

value value to set.

Definition at line 281 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setX3 (float value)

package

Sets the value of X3.

Parameters

value value to set.

Definition at line 435 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setY0 (float value)

package

Sets the value of Y0.

Parameters

value value to set.

Definition at line 303 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setY1 (float value)

package

Sets the value of Y1.

Parameters

value value to set.

Definition at line 325 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setY2 (float value)

package

Sets the value of Y2.

Parameters

value value to set.

Definition at line 347 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setY3 (float value)

package

Sets the value of Y3.

Parameters

value value to set.

Definition at line 453 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setZ0 (float value)

package

Sets the value of Z0.

Parameters

value value to set.

Definition at line 369 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setZ1 (float value)

package

Sets the value of Z1.

Parameters

value value to set.

Definition at line 391 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setZ2 (float value)

package

Sets the value of Z2.

Parameters

value value to set.

Definition at line 413 of file [MatrixF4x4.java](#).

void project.pc.sensor.representation.MatrixF4x4.setZ3 (float value)

package

Sets the value of Z3.

Parameters

value value to set.

Definition at line 471 of file [MatrixF4x4.java](#).

int project.pc.sensor.representation.MatrixF4x4.size ()

package

Returns size of the matrix.

Returns

the size of the matrix.

Definition at line 60 of file [MatrixF4x4.java](#).

The documentation for this class was generated from the following file:

- [src/project/pc/sensor/representation/MatrixF4x4.java](#)

project.pc.sensor.representation.Quaternion Class Reference

Class to represent a quaternion. [More...](#)

Collaboration diagram for project.pc.sensor.representation.Quaternion:

Public Member Functions

Quaternion ()

Constructor. [More...](#)

void **multiplyByQuat** (**Quaternion** input, **Quaternion** output)

Multiplies this Quaternion with another. [More...](#)

void **multiplyByQuat** (**Quaternion** input)

Multiplies this Quaternion by the input one. [More...](#)

void **multiplyByScalar** (float scalar)

Multiplies this Quaternion with a scalar. [More...](#)

void **normalize** ()

Normalizes this Quaternion. [More...](#)

Vector3f **rotateVector** (**Vector3f** v)

Rotates a 3-D vector by this Quaternion. [More...](#)

void **set** (**Quaternion** quat)

Copies the values from the given Quaternion to this one. [More...](#)

void **slerp** (**Quaternion** input, **Quaternion** output, float t)

Computes a linear interpolation between this Quaternion and the input Quaternion.
[More...](#)

void **toAxisAngle** (**Vector4f** output)

Gets an axis angle representation of this Quaternion. [More...](#)

double[] **toEulerAngles** ()

Returns the heading, attitude and bank of this Quaternion as euler angles in the double array respectively. [More...](#)

String **toString** ()

Returns a String representing this Quaternion. [More...](#)

▼ Public Member Functions inherited from **project.pc.sensor.representation.Vector4f**

Vector4f (float x, float y, float z, float w)

Constructor. [More...](#)

Vector4f ()

Constructor. [More...](#)

Vector4f (**Vector3f** vector3f, float w)

Constructor. [More...](#)

float[] **array** ()

Returns the array representation of this Vector4f. [More...](#)

float **getW** ()

Returns the W-dimension of this Vector4f. [More...](#)

float **getX** ()

Returns the X-dimension of this Vector4f. [More...](#)

float **getY** ()

Returns the Y-dimension of this Vector4f. [More...](#)

float **getZ** ()

Returns the Z-dimension of this Vector4f. [More...](#)

void **multiplyByScalar** (float scalar)

Multiplies this Vector4f with a scalar. [More...](#)

void **normalize** ()

Normalizes this Vector4f. [More...](#)

void **setW** (float w)

Sets the W-dimension of this Vector4f. [More...](#)

void **setX** (float x)

Sets the X-dimension of this Vector4f. [More...](#)

void **setY** (float y)

Sets the Y-dimension of this Vector4f. [More...](#)

void **setZ** (float z)

Sets the Z-dimension of this Vector4f. [More...](#)

String **toString** ()

Returns the String representation of this Vector4f. [More...](#)

Additional Inherited Members

▼ Package Functions inherited from **project.pc.sensor.representation.Vector4f**

void **copyFromV3f** (**Vector3f** input, float w)

Copies a **Vector3f** to this Vector4f using a separate value for W-dimension. [More...](#)

void **copyVec4** (**Vector4f** vec)

Copies another Vector4f to this one. [More...](#)

float **dotProduct** (**Vector4f** input)

Performs the dot product of two vectors. [More...](#)

Detailed Description

Class to represent a quaternion.

A quaternion H may be identified with $\mathbb{R}^4$, a four-dimensional vector space over the real numbers. The elements of this basis are customarily denoted as $1, i, j$, and k . Every element of H can be uniquely written as a linear combination of these basis elements, that is, as $a1 + bi + cj + dk$, where a, b, c , and d are real numbers.

Quaternions allow for elegant descriptions of 3D rotations, interpolations as well as extrapolations and compared to Euler angles, they don't suffer from gimbal lock. Interpolations between two Quaternions are called SLERP (Spherical Linear Interpolation).

If $a + bi + cj + dk$ is any quaternion, then a is called its scalar part and $bi + cj + dk$ is called its vector part. The scalar part of a quaternion is always real, and the vector part is always pure imaginary.

Using the basis $1, i, j, k$ of H it is possible to write H as a set of quadruples:

$$H = \{(a,b,c,d) \mid a,b,c,d \in \mathbb{R}\}.$$

Then the basis elements are:

$$1 = (1,0,0,0),$$

$$i = (0,1,0,0),$$

$$j = (0,0,1,0),$$

$$k = (0,0,0,1).$$

Addition of two quaternions :

$$(a_1, b_1, c_1, d_1) + (a_2, b_2, c_2, d_2) = (a_1 + a_2, b_1 + b_2, c_1 + c_2, d_1 + d_2).$$

Multiplications of two quaternions :

$$(a_1, b_1, c_1, d_1) (a_2, b_2, c_2, d_2) =$$

$$(a_1 a_2 - b_1 b_2 - c_1 c_2 - d_1 d_2,$$

$$a_1 b_2 + b_1 a_2 + c_1 d_2 - d_1 c_2,$$

$$a_1 c_2 - b_1 d_2 + c_1 a_2 + d_1 b_2,$$

$$a_1 d_2 + b_1 c_2 - c_1 b_2 + d_1 a_2).$$

Normalization of quaternion (a,b,c,d)

$$= (a/mag, b/mag, c/mag, d/mag),$$

where,

$$mag = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

The sensor data for 3-D mouse movement is sent from the mobile device is represented in the form of a Quaternion.

See also

project.pc.sensor.representation.Vector4f

Definition at line **89** of file **Quaternion.java**.

Constructor Documentation

project.pc.sensor.representation.Quaternion.Quaternion ()

Constructor.

Initializes this Quaternion.

Definition at line **103** of file **Quaternion.java**.

Member Function Documentation

void

project.pc.sensor.representation.Quaternion.multiplyByQuat (Quaternion **input**,
Quaternion **output**
)

Multiplies this Quaternion with another.

Parameters

input Quaternion with which this is multiplied.

output result of (this * input).

Definition at line **140** of file **Quaternion.java**.

void

project.pc.sensor.representation.Quaternion.multiplyByQuat (Quaternion **input**)

Multiplies this Quaternion by the input one.

Parameters

input Quaternion to be multiplied with.

Definition at line **165** of file **Quaternion.java**.

void project.pc.sensor.representation.Quaternion.multiplyByScalar (float **scalar)**

Multiplies this Quaternion with a scalar.

This overrides the **Vector4f#multiplyByScalar(float)** method.

Parameters

scalar the scalar.

See also

project.pc.sensor.representation.Vector4f::multiplyByScalar(float)

Definition at line **182** of file **Quaternion.java**.

void project.pc.sensor.representation.Quaternion.normalize ()

Normalizes this Quaternion.

Definition at line **112** of file **Quaternion.java**.

Vector3f project.pc.sensor.representation.Quaternion.rotateVector (Vector3f **v)**

Rotates a 3-D vector by this Quaternion.

Parameters

v the **Vector3f** to be rotated.

Returns

the rotated **Vector3f**.

Definition at line **496** of file **Quaternion.java**.

void project.pc.sensor.representation.Quaternion.set (Quaternion **quat)**

Copies the values from the given Quaternion to this one.

Parameters

quat the Quaternion to copy from.

Definition at line **128** of file **Quaternion.java**.

```
void project.pc.sensor.representation.Quaternion.slerp ( Quaternion input,
                                                         Quaternion output,
                                                         float      t
                                                         )
```

Computes a linear interpolation between this Quaternion and the input Quaternion.

Parameters

input the Quaternion to be slerped with this.

output the Quaternion to store the result.

t the ratio between the two quaternions where $0 \leq t \leq 1.0$.

Note: Increase value of t will bring rotation closer to the input Quaternion.

Definition at line 453 of file [Quaternion.java](#).

```
void project.pc.sensor.representation.Quaternion.toAxisAngle ( Vector4f output )
```

Gets an axis angle representation of this Quaternion.

Parameters

output Vector4f axis angle.

Definition at line 240 of file [Quaternion.java](#).

```
double [] project.pc.sensor.representation.Quaternion.toEulerAngles ( )
```

Returns the heading, attitude and bank of this Quaternion as euler angles in the double array respectively.

Returns

an array of size 3 containing the euler angles for this Quaternion.

Definition at line 274 of file [Quaternion.java](#).

```
String project.pc.sensor.representation.Quaternion.toString ( )
```

Returns a String representing this Quaternion.

This overrides the [Vector4f#toString\(\)](#) method.

Returns

String representing this Quaternion.

Definition at line 300 of file [Quaternion.java](#).

The documentation for this class was generated from the following file:

- [src/project/pc/sensor/representation/Quaternion.java](#)

project.pc.sensor.representation.Vector3f Class Reference

Class to represent a three dimensional vector. [More...](#)

Collaboration diagram for project.pc.sensor.representation.Vector3f:

Public Member Functions

Vector3f (float x, float y, float z)

Constructor. [More...](#)

Vector3f ()

Constructor. [More...](#)

Vector3f (**Vector3f** vector)

Constructor. [More...](#)

float **dotProduct** (**Vector3f** inputVec)

Performs the dot product of two vectors. [More...](#)

float **getX** ()

Returns the X-dimension of this Vector3f. [More...](#)

float **getY** ()

Returns the Y-dimension of this Vector3f. [More...](#)

float **getZ** ()

Returns the Z-dimension of this Vector3f. [More...](#)

void **multiplyByScalar** (float scalar)

Multiplies this Vector3f with a scalar. [More...](#)

void **normalize** ()

Normalizes this Vector3f. [More...](#)

void **setXYZ** (float x, float y, float z)

Sets this Vector3f as (x,y,z). [More...](#)

String **toString** ()

Returns the String representation of this Vector3f. [More...](#)

Package Functions

void **setX** (float x)

Sets the X-dimension of this Vector3f. [More...](#)

void **setY** (float y)
Sets the Y-dimension of this Vector3f. [More...](#)

void **setZ** (float z)
Sets the Z-dimension of this Vector3f. [More...](#)

float[] **toArray** ()
Returns the array representation of this Vector3f. [More...](#)

Detailed Description

Class to represent a three dimensional vector.

It is defined as a point (x,y,z) in three dimensional vector space. It is stored as a array of floats. The three dimensions are known as X,Y and Z.

Dot product of two vectors :

$$(a_1, b_1, c_1) \cdot (a_2, b_2, c_2) = a_1 a_2 + b_1 b_2 + c_1 c_2 .$$

Product of a vector with a scalar :

$$(a, b, c) * s = (a * s, b * s, c * s).$$

Normalization of a vector (a,b,c)

$$= (a/mag, b/mag, c/mag),$$

where,

$$mag = \sqrt{a^2 + b^2 + c^2}.$$

See also

project.pc.sensor.representation.Vector3f
project.pc.sensor.representation.Quaternion

Definition at line **37** of file **Vector3f.java**.

Constructor Documentation

```
project.pc.sensor.representation.Vector3f.Vector3f ( float x,  
                                                    float y,  
                                                    float z  
                                                    )
```

Constructor.

Creates the Vector3f (x,y,z).

Parameters

- x** the X-dimension.
- y** the Y-dimension.
- z** the Z-dimension.

Definition at line **50** of file **Vector3f.java**.

project.pc.sensor.representation.Vector3f.Vector3f ()

Constructor.

Default constructor.

Definition at line **67** of file **Vector3f.java**.

project.pc.sensor.representation.Vector3f.Vector3f (Vector3f vector)

Constructor.

Copy constructor. Copies another Vector3f to this one.

Parameters

vector the Vector3f to be copied.

Definition at line **78** of file **Vector3f.java**.

Member Function Documentation

float project.pc.sensor.representation.Vector3f.dotProduct (Vector3f inputVec)

Performs the dot product of two vectors.

Parameters

inputVec the Vector3f to be multiplied.

Returns

result of (this . inputVec).

Definition at line **219** of file **Vector3f.java**.

float project.pc.sensor.representation.Vector3f.getX ()

Returns the X-dimension of this Vector3f.

Returns

the X-dimension.

Definition at line **151** of file **Vector3f.java**.

float project.pc.sensor.representation.Vector3f.getY ()

Returns the Y-dimension of this Vector3f.

Returns

the Y-dimension.

Definition at line **160** of file **Vector3f.java**.

float project.pc.sensor.representation.Vector3f.getZ ()

Returns the Z-dimension of this Vector3f.

Returns

the Z-dimension.

Definition at line **169** of file **Vector3f.java**.

void project.pc.sensor.representation.Vector3f.multiplyByScalar (float scalar)

Multiplies this Vector3f with a scalar.

Parameters

scalar the scalar.

Definition at line **128** of file **Vector3f.java**.

void project.pc.sensor.representation.Vector3f.normalize ()

Normalizes this Vector3f.

Definition at line **137** of file **Vector3f.java**.

void project.pc.sensor.representation.Vector3f.setX (float x)

package

Sets the X-dimension of this Vector3f.

Parameters

x the X-dimension.

Definition at line **178** of file **Vector3f.java**.

```
void project.pc.sensor.representation.Vector3f.setXYZ ( float x,  
                                                    float y,  
                                                    float z  
                                                    )
```

Sets this Vector3f as (x,y,z).

Parameters

x the X-dimension.

y the Y-dimension.

z the Z-dimension.

Definition at line **207** of file **Vector3f.java**.

```
void project.pc.sensor.representation.Vector3f.setY ( float y )
```

package

Sets the Y-dimension of this Vector3f.

Parameters

y the Y-dimension.

Definition at line **187** of file **Vector3f.java**.

```
void project.pc.sensor.representation.Vector3f.setZ ( float z )
```

package

Sets the Z-dimension of this Vector3f.

Parameters

z the Z-dimension.

Definition at line **196** of file **Vector3f.java**.

```
float [] project.pc.sensor.representation.Vector3f.toArray ( )
```

package

Returns the array representation of this Vector3f.

Returns

the array [X, Y, Z].

Definition at line **101** of file **Vector3f.java**.

String project.pc.sensor.representation.Vector3f.toString ()

Returns the String representation of this Vector3f.

It overrides the **java.lang.Object#toString()** method.

Returns

String representing this Vector3f.

Definition at line **244** of file **Vector3f.java**.

The documentation for this class was generated from the following file:

- src/project/pc/sensor/representation/**Vector3f.java**

project.pc.sensor.representation.Vector4f Class Reference

Class to represent a four dimensional vector. [More...](#)

Collaboration diagram for project.pc.sensor.representation.Vector4f:

Public Member Functions

Vector4f (float x, float y, float z, float w)

Constructor. [More...](#)

Vector4f ()

Constructor. [More...](#)

Vector4f (**Vector3f** vector3f, float w)

Constructor. [More...](#)

float[] **array** ()

Returns the array representation of this Vector4f. [More...](#)

float **getW** ()

Returns the W-dimension of this Vector4f. [More...](#)

float **getX** ()

Returns the X-dimension of this Vector4f. [More...](#)

float **getY** ()

Returns the Y-dimension of this Vector4f. [More...](#)

float **getZ** ()

Returns the Z-dimension of this Vector4f. [More...](#)

void **multiplyByScalar** (float scalar)

Multiplies this Vector4f with a scalar. [More...](#)

void **normalize** ()

Normalizes this Vector4f. [More...](#)

void **setW** (float w)

Sets the W-dimension of this Vector4f. [More...](#)

void **setX** (float x)

Sets the X-dimension of this Vector4f. [More...](#)

void **setY** (float y)
Sets the Y-dimension of this Vector4f. [More...](#)

void **setZ** (float z)
Sets the Z-dimension of this Vector4f. [More...](#)

String **toString** ()
Returns the String representation of this Vector4f. [More...](#)

Package Functions

void **copyFromV3f** (**Vector3f** input, float w)
Copies a **Vector3f** to this Vector4f using a separate value for W-dimension. [More...](#)

void **copyVec4** (**Vector4f** vec)
Copies another Vector4f to this one. [More...](#)

float **dotProduct** (**Vector4f** input)
Performs the dot product of two vectors. [More...](#)

Detailed Description

Class to represent a four dimensional vector.

It is defined as a point (x,y,z,w) in four dimensional vector space. It is stored as a array of floats. The four dimensions are known as X,Y,Z and W.

Dot product of two vectors :

$$(a_1,b_1,c_1,d_1) \cdot (a_2,b_2,c_2,d_2) = a_1a_2 + b_1b_2 + c_1c_2 + d_1d_2 .$$

Product of a vector with a scalar :

$$(a, b, c, d) * s = (a * s, b * s, c * s, d * s).$$

Normalization of a vector (a,b,c,d)

$$= (a/mag, b/mag, c/mag, d/mag),$$

where,

$$mag = \sqrt{a^2+b^2+c^2+d^2}.$$

See also

project.pc.sensor.representation.Vector3f

project.pc.sensor.representation.Quaternion

Definition at line **36** of file **Vector4f.java**.

Constructor Documentation

```
project.pc.sensor.representation.Vector4f.Vector4f ( float x,  
                                                    float y,  
                                                    float z,  
                                                    float w  
                                                    )
```

Constructor.

Creates the Vector4f (x,y,z,w).

Parameters

- x** the X-dimension.
- y** the Y-dimension.
- z** the Z-dimension.
- w** the W-dimension.

Definition at line **50** of file **Vector4f.java**.

```
project.pc.sensor.representation.Vector4f.Vector4f ( )
```

Constructor.

Default constructor. Creates the Vector4f (0,0,0,0).

Definition at line **62** of file **Vector4f.java**.

```
project.pc.sensor.representation.Vector4f.Vector4f ( Vector3f vector3f,  
                                                    float w  
                                                    )
```

Constructor.

Creates the Vector4f (vector3f.getX(), vector3f.getY(), vector3f.getZ(), w).

Parameters

- vector3f** a **Vector3f**.
- w** the w-dimension.

Definition at line **78** of file **Vector4f.java**.

Member Function Documentation

float [] project.pc.sensor.representation.Vector4f.array ()

Returns the array representation of this Vector4f.

Returns

the array [X, Y, Z, W].

Definition at line **90** of file **Vector4f.java**.

void
project.pc.sensor.representation.Vector4f.copyFromV3f (**Vector3f** **input**,
float **w**
) package

Copies a **Vector3f** to this Vector4f using a separate value for W-dimension.

Parameters

input the **Vector3f**.

w float representing the W-dimension.

Definition at line **286** of file **Vector4f.java**.

void project.pc.sensor.representation.Vector4f.copyVec4 (Vector4f vec) package

Copies another Vector4f to this one.

Parameters

vec the Vector4f to be copied.

Definition at line **99** of file **Vector4f.java**.

float
project.pc.sensor.representation.Vector4f.dotProduct (**Vector4f** **input**) package

Performs the dot product of two vectors.

Parameters

input the Vector4f to be multiplied.

Returns

result of (this . input).

Definition at line **157** of file **Vector4f.java**.

float project.pc.sensor.representation.Vector4f.getW ()

Returns the W-dimension of this Vector4f.

Returns

the W-dimension.

Definition at line **220** of file **Vector4f.java**.

float project.pc.sensor.representation.Vector4f.getX ()

Returns the X-dimension of this Vector4f.

Returns

the X-dimension.

Definition at line **193** of file **Vector4f.java**.

float project.pc.sensor.representation.Vector4f.getY ()

Returns the Y-dimension of this Vector4f.

Returns

the Y-dimension.

Definition at line **202** of file **Vector4f.java**.

float project.pc.sensor.representation.Vector4f.getZ ()

Returns the Z-dimension of this Vector4f.

Returns

the Z-dimension.

Definition at line **211** of file **Vector4f.java**.

void project.pc.sensor.representation.Vector4f.multiplyByScalar (float **scalar)**

Multiplies this Vector4f with a scalar.

Parameters

scalar the scalar.

Definition at line **144** of file **Vector4f.java**.

void project.pc.sensor.representation.Vector4f.normalize ()

Normalizes this Vector4f.

Definition at line **173** of file **Vector4f.java**.

void project.pc.sensor.representation.Vector4f.setW (float **w)**

Sets the W-dimension of this Vector4f.

Parameters

w the W-dimension.

Definition at line **256** of file **Vector4f.java**.

void project.pc.sensor.representation.Vector4f.setX (float **x)**

Sets the X-dimension of this Vector4f.

Parameters

x the X-dimension.

Definition at line **229** of file **Vector4f.java**.

void project.pc.sensor.representation.Vector4f.setY (float **y)**

Sets the Y-dimension of this Vector4f.

Parameters

y the Y-dimension.

Definition at line **238** of file **Vector4f.java**.

void project.pc.sensor.representation.Vector4f.setZ (float **z)**

Sets the Z-dimension of this Vector4f.

Parameters

z the Z-dimension.

Definition at line **247** of file **Vector4f.java**.

String project.pc.sensor.representation.Vector4f.toString ()

Returns the String representation of this Vector4f.

It overrides the **java.lang.Object#toString()** method.

Returns

String representing this Vector4f.

Definition at line **303** of file **Vector4f.java**.

The documentation for this class was generated from the following file:

- src/project/pc/sensor/representation/**Vector4f.java**

project.pc.sensor.SensorDataHandler Class Reference

Class to transform filtered sensor data into mouse pointer movement data. [More...](#)

Collaboration diagram for project.pc.sensor.SensorDataHandler:

Public Member Functions

SensorDataHandler (**Quaternion** q, float sensitivity)
Constructor. [More...](#)

Cartesian2D **pointerUpdate** (**Quaternion** currentQuaternion)
Determines pointer location from current Quaternion object. [More...](#)

Detailed Description

Class to transform filtered sensor data into mouse pointer movement data.

Definition at line **11** of file **SensorDataHandler.java**.

Constructor Documentation

```
project.pc.sensor.SensorDataHandler.SensorDataHandler ( Quaternion q,  
                                                         float      sensitivity  
                                                         )
```

Constructor.

Initializes this SensorDataHandler.

Parameters

q initial Quaternion object.
sensitivity sensitivity co-efficient of the mouse movement.

Definition at line **26** of file **SensorDataHandler.java**.

Member Function Documentation

Cartesian2D

project.pc.sensor.SensorDataHandler.pointerUpdate (Quaternion currentQuaternion)

Determines pointer location from current Quaternion object.

Parameters

currentQuaternion current Quaternion object acquired after filtering.

Returns

an object of **project.pc.sensor.representation.Cartesian2D** containing current pointer location.

Definition at line **51** of file **SensorDataHandler.java**.

The documentation for this class was generated from the following file:

- src/project/pc/sensor/**SensorDataHandler.java**

project.pc.Main Class Reference

Class containing the `main()` method. [More...](#)

Collaboration diagram for project.pc.Main:

Static Public Member Functions

static void **main** (String args[]) throws IOException,AWTException,InterruptedException
The main method. [More...](#)

Detailed Description

Class containing the `main()` method.

The **main(String[])** method :

- Generates the public key of the server.
- Instantiates the **project.pc.net.NetworkManager** for handling the network operations.
- Displays the main window of the GUI.

See also

`main(String[])`

`project.pc.net.NetworkManager`

`project.pc.net.ServerInfo`

Definition at line **34** of file **Main.java**.

Member Function Documentation

static void **project.pc.Main.main** (**String args[]**) **throws** **IOException,AWTException,InterruptedException** static

The main method.

Parameters

args the command-line arguments.

Exceptions

IOException

AWTException

InterruptedException

Definition at line **44** of file **Main.java**.

The documentation for this class was generated from the following file:

- `src/project/pc/Main.java`

4.3. Standardization of Coding

Java is the primary language used for both PC and Android applications. Other languages include XML, HTML and CSS. XML is used to implement the GUI configurations in Android. HTML and CSS are used for documentation and designing the app information view. The Java 1.8 and Java 1.7 are used for PC and Android respectively. The minimum Android API used is 16 and the targeted API is 25. Gradle Build Tool is used to build the app in Android Studio v2.3.1. The version of Gradle used is 3.3. IntelliJ IDEA version 2017.1.2 is used as the Integrated Development Environment (IDE) for PC application development. The indentation style used is the Java variant of K&R.

5. Testing

5.1. Testing Techniques and Strategies

For testing the project, grey box testing methods have been employed throughout unit testing. For system testing and integration testing, black box testing methods have been employed. In grey box testing, the test cases were designed for edge cases and general cases as well, for which abstract knowledge of project code was required. For system and integration testing, the non-functional requirements were tested without taking into account the project codes. Testing has been a significant process during the project development, which can be judged from the Gantt and PERT charts.

5.2. Test Results

5.2.1. Data Packet Analysis

The network data packet analysis results have been obtained using WireShark. The TCP packet data are encrypted and hence are not susceptible to MITM attacks. Hence the system provides good security measures.

Server Broadcast Packet Dump

No.	Time	Source	Destination	Protocol	Length	Info
7959	482.574104535	192.168.43.239	192.168.43.255	UDP	511	34608 → 1235 Len=469

Frame 7959: 511 bytes on wire (4088 bits), 511 bytes captured (4088 bits) on interface 0
Interface id: 0 (wlp6s0)
Encapsulation type: Ethernet (1)
Arrival Time: May 8, 2017 19:45:09.451753495 IST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1494252909.451753495 seconds
[Time delta from previous captured frame: 1.449363502 seconds]
[Time delta from previous displayed frame: 0.000000000 seconds]
[Time since reference or first frame: 482.574104535 seconds]
Frame Number: 7959
Frame Length: 511 bytes (4088 bits)
Capture Length: 511 bytes (4088 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:udp:data]
[Coloring Rule Name: UDP]
[Coloring Rule String: udp]
Ethernet II, Src: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
Destination: Broadcast (ff:ff:ff:ff:ff:ff)
Address: Broadcast (ff:ff:ff:ff:ff:ff)
.... ..1. = LG bit: Locally administered address (this is NOT the factory default)
.... ..1 = IG bit: Group address (multicast/broadcast)
Source: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
Address: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
.... ..0. = LG bit: Globally unique address (factory default)


```

.... ..0 .... = IG bit: Individual address (unicast)
Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.43.239, Dst: 192.168.43.255
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  0000 00.. = Differentiated Services Codepoint: Default (0)
  .... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
Total Length: 497
Identification: 0x7d98 (32152)
Flags: 0x02 (Don't Fragment)
  0... .... = Reserved bit: Not set
  .1.. .... = Don't fragment: Set
  ..0. .... = More fragments: Not set
Fragment offset: 0
Time to live: 64
Protocol: UDP (17)
Header checksum: 0xe224 [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.43.239
Destination: 192.168.43.255
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]
User Datagram Protocol, Src Port: 34608, Dst Port: 1235
Source Port: 34608
Destination Port: 1235
Length: 477
Checksum: 0x0b41 [unverified]
[Checksum Status: Unverified]
[Stream index: 109]
Data (469 bytes)

0000 7b 22 6d 53 65 72 76 65 72 50 75 62 4b 65 79 22 {"mServerPubKey"
0010 3a 5b 37 37 2c 37 30 2c 31 31 31 2c 31 31 39 2c :[77,70,111,119,
0020 37 30 2c 36 35 2c 38 39 2c 37 32 2c 37 35 2c 31 70,65,89,72,75,1
0030 31 31 2c 39 30 2c 37 33 2c 31 32 32 2c 31 30 36 11,90,73,122,106
0040 2c 34 38 2c 36 37 2c 36 35 2c 38 31 2c 38 39 2c ,48,67,65,81,89,
0050 37 34 2c 37 35 2c 31 32 31 2c 38 31 2c 36 38 2c 74,75,121,81,68,
0060 36 35 2c 31 31 39 2c 37 33 2c 37 33 2c 36 35 2c 65,119,73,73,65,
0070 38 31 2c 36 39 2c 37 32 2c 36 35 2c 34 38 2c 37 81,69,72,65,48,7
0080 33 2c 36 35 2c 36 36 2c 36 38 2c 31 31 38 2c 39 3,65,66,68,118,9
0090 30 2c 38 30 2c 31 30 32 2c 36 39 2c 31 31 33 2c 0,80,102,69,113,
00a0 38 31 2c 39 30 2c 38 31 2c 31 30 35 2c 37 35 2c 81,90,81,105,75,
00b0 37 34 2c 38 34 2c 31 30 32 2c 39 39 2c 31 31 32 74,84,102,99,112
00c0 2c 35 37 2c 38 36 2c 36 39 2c 31 30 38 2c 36 38 ,57,86,69,108,68
00d0 2c 38 30 2c 35 37 2c 37 36 2c 31 30 38 2c 31 30 ,80,57,76,108,10
00e0 34 2c 38 36 2c 36 37 2c 37 37 2c 37 35 2c 38 37 4,86,67,77,75,87
00f0 2c 34 33 2c 39 30 2c 35 35 2c 34 33 2c 37 32 2c ,43,90,55,43,72,
0100 37 31 2c 31 30 39 2c 34 33 2c 34 39 2c 37 32 2c 71,109,43,49,72,
0110 35 31 2c 36 35 2c 34 39 2c 35 34 2c 39 37 2c 31 51,65,49,54,97,1
0120 31 37 2c 34 39 2c 38 33 2c 31 30 31 2c 37 30 2c 17,49,83,101,70,
0130 38 31 2c 31 31 39 2c 37 36 2c 35 36 2c 36 39 2c 81,119,76,56,69,
0140 34 38 2c 37 35 2c 31 30 33 2c 38 35 2c 31 30 38 48,75,103,85,108
0150 2c 31 31 38 2c 39 37 2c 36 36 2c 37 37 2c 38 35 ,118,97,66,77,85
0160 2c 31 31 34 2c 31 31 37 2c 38 31 2c 31 31 33 2c ,114,117,81,113,
0170 36 39 2c 38 39 2c 36 35 2c 39 37 2c 34 38 2c 37 69,89,65,97,48,7
0180 38 2c 36 38 2c 31 32 30 2c 35 33 2c 39 30 2c 36 8,68,120,53,90,6
0190 35 2c 31 31 32 2c 38 33 2c 38 35 2c 31 30 33 2c 5,112,83,85,103,
01a0 36 31 5d 2c 22 6d 53 65 72 76 65 72 49 6e 66 6f 61], "mServerInfo
01b0 22 3a 22 6d 6f 6e 74 69 40 53 41 59 41 4e 54 41 ": "monti@SAYANTA
01c0 4e 22 2c 22 6d 53 74 6f 70 46 6c 61 67 22 3a 66 N", "mStopFlag":f
01d0 61 6c 73 65 7d else}
Data: 7b226d5365727665725075624b6579223a5b37372c37302c...
[Length: 469]

```

Packet Dump during Pairing Key Exchange

No.	Time	Source	Destination	Protocol	Length	Info
11751	1240.677044961	192.168.43.1	192.168.43.239	TCP	527	44215 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=87808 Len=461 TSval=2752217 TSecr=6310170

Frame 11751: 527 bytes on wire (4216 bits), 527 bytes captured (4216 bits) on interface 0

Interface id: 0 (wlp6s0)
Encapsulation type: Ethernet (1)
Arrival Time: May 8, 2017 19:57:47.554693921 IST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1494253667.554693921 seconds
[Time delta from previous captured frame: 0.031775947 seconds]
[Time delta from previous displayed frame: 0.031775947 seconds]
[Time since reference or first frame: 1240.677044961 seconds]

Frame Number: 11751
Frame Length: 527 bytes (4216 bits)
Capture Length: 527 bytes (4216 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:tcp:data]
[Coloring Rule Name: TCP]
[Coloring Rule String: tcp]

Ethernet II, Src: Motorola_07:ec:72 (14:30:c6:07:ec:72), Dst: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)

Destination: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
Address: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
.... 0. = LG bit: Globally unique address (factory default)
.... 0. = IG bit: Individual address (unicast)

Source: Motorola_07:ec:72 (14:30:c6:07:ec:72)
Address: Motorola_07:ec:72 (14:30:c6:07:ec:72)
.... 0. = LG bit: Globally unique address (factory default)
.... 0. = IG bit: Individual address (unicast)

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.239

0100 = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
0000 00.. = Differentiated Services Codepoint: Default (0)
.... ..00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)

Total Length: 513
Identification: 0xe5c6 (58822)
Flags: 0x02 (Don't Fragment)
0... = Reserved bit: Not set
.1.. = Don't fragment: Set
..0. = More fragments: Not set

Fragment offset: 0
Time to live: 64
Protocol: TCP (6)
Header checksum: 0x7aef [validation disabled]
[Header checksum status: Unverified]
Source: 192.168.43.1
Destination: 192.168.43.239
[Source GeoIP: Unknown]
[Destination GeoIP: Unknown]

Transmission Control Protocol, Src Port: 44215, Dst Port: 1234, Seq: 1, Ack: 1, Len: 461

Source Port: 44215
Destination Port: 1234
[Stream index: 47]
[TCP Segment Len: 461]
Sequence number: 1 (relative sequence number)
[Next sequence number: 462 (relative sequence number)]
Acknowledgment number: 1 (relative ack number)
Header Length: 32 bytes
Flags: 0x018 (PSH, ACK)
000. = Reserved: Not set
...0 = Nonce: Not set

```

.... 0... .... = Congestion Window Reduced (CWR): Not set
.... .0... .... = ECN-Echo: Not set
.... ..0. .... = Urgent: Not set
.... ...1 .... = Acknowledgment: Set
.... .... 1... = Push: Set
.... .... .0.. = Reset: Not set
.... .... ..0. = Syn: Not set
.... .... ...0 = Fin: Not set
[TCP Flags: .....AP...]
Window size value: 343
[Calculated window size: 87808]
[Window size scaling factor: 256]
Checksum: 0x1306 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  No-Operation (NOP)
    Type: 1
      0... .... = Copy on fragmentation: No
      .00. .... = Class: Control (0)
      ...0 0001 = Number: No-Operation (NOP) (1)
  No-Operation (NOP)
    Type: 1
      0... .... = Copy on fragmentation: No
      .00. .... = Class: Control (0)
      ...0 0001 = Number: No-Operation (NOP) (1)
  Timestamps: TSval 2752217, TSecr 6310170
    Kind: Time Stamp Option (8)
    Length: 10
    Timestamp value: 2752217
    Timestamp echo reply: 6310170
[SEQ/ACK analysis]
[iRTT: 0.000442249 seconds]
[Bytes in flight: 461]
[Bytes sent since last PSH flag: 461]
Data (461 bytes)

0000 7b 22 6d 43 6c 69 65 6e 74 49 6e 66 6f 22 3a 22 {"mClientInfo":
0010 6d 6f 74 6f 72 6f 6c 61 20 58 54 31 30 35 32 22  "motorola XT1052"
0020 2c 22 6d 50 75 62 6c 69 63 4b 65 79 22 3a 5b 37  , "mPublicKey": [7
0030 37 2c 37 30 2c 31 31 31 2c 31 31 39 2c 37 30 2c  7, 70, 111, 119, 70,
0040 36 35 2c 38 39 2c 37 32 2c 37 35 2c 31 31 31 2c  65, 89, 72, 75, 111,
0050 39 30 2c 37 33 2c 31 32 32 2c 31 30 36 2c 34 38  90, 73, 122, 106, 48
0060 2c 36 37 2c 36 35 2c 38 31 2c 38 39 2c 37 34 2c  , 67, 65, 81, 89, 74,
0070 37 35 2c 31 32 31 2c 38 31 2c 36 38 2c 36 35 2c  75, 121, 81, 68, 65,
0080 31 31 39 2c 37 33 2c 37 33 2c 36 35 2c 38 31 2c  119, 73, 73, 65, 81,
0090 36 39 2c 37 32 2c 36 35 2c 34 38 2c 37 33 2c 36  69, 72, 65, 48, 73, 6
00a0 35 2c 36 36 2c 36 38 2c 31 30 30 2c 31 30 37 2c  5, 66, 68, 100, 107,
00b0 31 31 33 2c 31 31 34 2c 31 32 31 2c 37 38 2c 31  113, 114, 121, 78, 1
00c0 31 38 2c 31 31 32 2c 38 38 2c 31 30 36 2c 35 31  18, 112, 88, 106, 51
00d0 2c 37 31 2c 34 37 2c 36 36 2c 37 33 2c 31 32 32  , 71, 47, 66, 73, 122
00e0 2c 36 35 2c 31 30 32 2c 31 32 30 2c 31 31 33 2c  , 65, 102, 120, 113,
00f0 31 31 38 2c 37 32 2c 31 31 38 2c 35 36 2c 31 30  118, 72, 118, 56, 10
0100 30 2c 36 39 2c 38 36 2c 34 33 2c 36 36 2c 31 32  0, 69, 86, 43, 66, 12
0110 31 2c 35 33 2c 36 39 2c 31 31 30 2c 38 34 2c 31  1, 53, 69, 110, 84, 1
0120 30 35 2c 31 31 39 2c 36 37 2c 37 31 2c 34 37 2c  05, 119, 67, 71, 47,
0130 37 32 2c 34 38 2c 31 30 34 2c 31 30 33 2c 38 37  72, 48, 104, 103, 87
0140 2c 38 31 2c 31 30 34 2c 37 37 2c 36 39 2c 37 32  , 81, 104, 77, 69, 72
0150 2c 37 37 2c 31 31 30 2c 35 37 2c 39 37 2c 35 37  , 77, 110, 57, 97, 57
0160 2c 37 30 2c 34 33 2c 37 35 2c 34 33 2c 31 31 39  , 70, 43, 75, 43, 119
0170 2c 31 30 34 2c 31 30 34 2c 38 30 2c 36 39 2c 35  , 104, 104, 80, 69, 5
0180 33 2c 31 31 35 2c 38 31 2c 35 34 2c 31 31 35 2c  3, 115, 81, 54, 115,
0190 35 31 2c 37 37 2c 38 37 2c 35 30 2c 39 30 2c 38  51, 77, 87, 50, 90, 8
01a0 36 2c 36 35 2c 38 36 2c 35 34 2c 31 31 36 2c 37  6, 65, 86, 54, 116, 7
01b0 34 2c 31 31 31 2c 31 30 33 2c 35 32 2c 31 30 37  4, 111, 103, 52, 107
01c0 2c 31 32 31 2c 35 32 2c 36 31 5d 7d 0a         , 121, 52, 61]}.
Data: 7b226d436c69656e74496e666f223a226d6f746f726f6c61...
[Length: 461]

```

2D Mouse Movement Packet Dump

No.	Time	Source	Destination	Protocol	Length	Info
16646	2148.158154693	192.168.43.1	192.168.43.239	TCP	215	44215 → 1234 [PSH, ACK] Seq=311671 Ack=26 Win=87808 Len=149 TSval=2834220 TSecr=6582421

Frame 16646: 215 bytes on wire (1720 bits), 215 bytes captured (1720 bits) on interface 0

Interface id: 0 (wlp6s0)

Encapsulation type: Ethernet (1)

Arrival Time: May 8, 2017 20:12:55.035803653 IST

[Time shift for this packet: 0.000000000 seconds]

Epoch Time: 1494254575.035803653 seconds

[Time delta from previous captured frame: 0.000542778 seconds]

[Time delta from previous displayed frame: 0.009189801 seconds]

[Time since reference or first frame: 2148.158154693 seconds]

Frame Number: 16646

Frame Length: 215 bytes (1720 bits)

Capture Length: 215 bytes (1720 bits)

[Frame is marked: False]

[Frame is ignored: False]

[Protocols in frame: eth:ethertype:ip:tcp:data]

[Coloring Rule Name: TCP]

[Coloring Rule String: tcp]

Ethernet II, Src: Motorola_07:ec:72 (14:30:c6:07:ec:72), Dst: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)

Destination: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)

Address: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)

.... 0. = LG bit: Globally unique address (factory default)

.... 0. = IG bit: Individual address (unicast)

Source: Motorola_07:ec:72 (14:30:c6:07:ec:72)

Address: Motorola_07:ec:72 (14:30:c6:07:ec:72)

.... 0. = LG bit: Globally unique address (factory default)

.... 0. = IG bit: Individual address (unicast)

Type: IPv4 (0x0800)

Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.239

0100 = Version: 4

.... 0101 = Header Length: 20 bytes (5)

Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)

0000 00.. = Differentiated Services Codepoint: Default (0)

.... 00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)

Total Length: 201

Identification: 0xea8d (60045)

Flags: 0x02 (Don't Fragment)

0... = Reserved bit: Not set

.1... = Don't fragment: Set

..0. = More fragments: Not set

Fragment offset: 0

Time to live: 64

Protocol: TCP (6)

Header checksum: 0x7760 [validation disabled]

[Header checksum status: Unverified]

Source: 192.168.43.1

Destination: 192.168.43.239

[Source GeoIP: Unknown]

[Destination GeoIP: Unknown]

Transmission Control Protocol, Src Port: 44215, Dst Port: 1234, Seq: 311671, Ack: 26, Len: 149

Source Port: 44215

Destination Port: 1234

[Stream index: 47]

[TCP Segment Len: 149]

Sequence number: 311671 (relative sequence number)

[Next sequence number: 311820 (relative sequence number)]

Acknowledgment number: 26 (relative ack number)

```

Header Length: 32 bytes
Flags: 0x018 (PSH, ACK)
  000. .... = Reserved: Not set
  ...0 .... = Nonce: Not set
  .... 0... = Congestion Window Reduced (CWR): Not set
  .... .0.. = ECN-Echo: Not set
  .... ..0. = Urgent: Not set
  .... ...1 = Acknowledgment: Set
  .... .... 1... = Push: Set
  .... .... .0.. = Reset: Not set
  .... .... ..0. = Syn: Not set
  .... .... ...0 = Fin: Not set
[TCP Flags: .....AP...]
Window size value: 343
[Calculated window size: 87808]
[Window size scaling factor: 256]
Checksum: 0x694e [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  No-Operation (NOP)
    Type: 1
      0... .... = Copy on fragmentation: No
      .00. .... = Class: Control (0)
      ...0 0001 = Number: No-Operation (NOP) (1)
  No-Operation (NOP)
    Type: 1
      0... .... = Copy on fragmentation: No
      .00. .... = Class: Control (0)
      ...0 0001 = Number: No-Operation (NOP) (1)
  Timestamps: TSval 2834220, TSecr 6582421
    Kind: Time Stamp Option (8)
    Length: 10
    Timestamp value: 2834220
    Timestamp echo reply: 6582421
[SEQ/ACK analysis]
  [iRTT: 0.000442249 seconds]
  [Bytes in flight: 149]
  [Bytes sent since last PSH flag: 149]
Data (149 bytes)

0000 46 41 41 62 68 68 58 4d 59 52 43 74 57 45 35 6a  FAAbhhXMYRCtWE5j
0010 51 42 70 4e 2b 46 38 46 4e 64 52 4d 42 42 42 42  QBpN+F8FNdRMBBBB
0020 56 46 4d 30 49 33 66 42 41 7a 52 79 62 54 47 52  VFM0I3fBAzRybTGR
0030 47 71 78 31 37 4d 37 32 55 77 57 6b 63 54 46 6e  Gqx17M72UwWkcTFn
0040 57 77 58 66 4e 75 63 47 61 38 49 65 46 47 72 49  WwXfNucGa8IeFGrI
0050 47 33 35 65 4a 38 74 67 56 33 75 49 44 78 7a 2f  G35eJ8tgV3uIDxz/
0060 6a 56 52 37 4d 52 68 47 39 39 4a 70 4c 73 61 61  jVR7MRhG99JpLsaa
0070 42 71 6b 69 66 64 77 72 54 4f 43 69 75 77 4a 45  BqkifdwrTOciuwJE
0080 63 69 78 66 4b 68 48 6b 78 2f 73 77 38 56 75 43  cixfKhHkx/sw8VuC
0090 50 77 3d 3d 0a                                     Pw==.
Data: 464141626868584d595243745745356a5142704e2b463846...
[Length: 149]

```

3D Mouse Movement Packet Dump

No.	Time	Source	Destination	Protocol	Length	Info
13306	1794.335664195	192.168.43.1	192.168.43.239	TCP	531	44215 → 1234 [PSH, ACK] Seq=5318 Ack=26 Win=87808 Len=465 TSval=2803082 TSecr=6476273

```

Frame 13306: 531 bytes on wire (4248 bits), 531 bytes captured (4248 bits) on interface 0
Interface id: 0 (wlp6s0)
Encapsulation type: Ethernet (1)
Arrival Time: May 8, 2017 20:07:01.213313155 IST

```

```

[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1494254221.213313155 seconds
[Time delta from previous captured frame: 0.015512622 seconds]
[Time delta from previous displayed frame: 0.015512622 seconds]
[Time since reference or first frame: 1794.335664195 seconds]
Frame Number: 13306
Frame Length: 531 bytes (4248 bits)
Capture Length: 531 bytes (4248 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:tcp:data]
[Coloring Rule Name: TCP]
[Coloring Rule String: tcp]
Ethernet II, Src: Motorola_07:ec:72 (14:30:c6:07:ec:72), Dst: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
  Destination: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
    Address: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Source: Motorola_07:ec:72 (14:30:c6:07:ec:72)
    Address: Motorola_07:ec:72 (14:30:c6:07:ec:72)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.239
  0100 .... = Version: 4
  ....0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    0000 00.. = Differentiated Services Codepoint: Default (0)
    ....00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
  Total Length: 517
  Identification: 0xe5d4 (58836)
  Flags: 0x02 (Don't Fragment)
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  Header checksum: 0x7add [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.43.1
  Destination: 192.168.43.239
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: 44215, Dst Port: 1234, Seq: 5318, Ack: 26, Len: 465
  Source Port: 44215
  Destination Port: 1234
  [Stream index: 47]
  [TCP Segment Len: 465]
  Sequence number: 5318 (relative sequence number)
  [Next sequence number: 5783 (relative sequence number)]
  Acknowledgment number: 26 (relative ack number)
  Header Length: 32 bytes
  Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ...0. .... = Nonce: Not set
    ....0... = Congestion Window Reduced (CWR): Not set
    ....0... = ECN-Echo: Not set
    ....0... = Urgent: Not set
    ....0... = Acknowledgment: Set
    ....1... = Push: Set
    ....0... = Reset: Not set
    ....0... = Syn: Not set
    ....0... = Fin: Not set
  [TCP Flags: .....AP...]
  Window size value: 343
  [Calculated window size: 87808]
  [Window size scaling factor: 256]
  Checksum: 0x2bd3 [unverified]
  [Checksum Status: Unverified]

```

```

Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  No-Operation (NOP)
    Type: 1
    0... .... = Copy on fragmentation: No
    .00. .... = Class: Control (0)
    ...0 0001 = Number: No-Operation (NOP) (1)
  No-Operation (NOP)
    Type: 1
    0... .... = Copy on fragmentation: No
    .00. .... = Class: Control (0)
    ...0 0001 = Number: No-Operation (NOP) (1)
  Timestamps: TSval 2803082, TSecr 6476273
    Kind: Time Stamp Option (8)
    Length: 10
    Timestamp value: 2803082
    Timestamp echo reply: 6476273
[SEQ/ACK analysis]
  [iRTT: 0.000442249 seconds]
  [Bytes in flight: 465]
  [Bytes sent since last PSH flag: 465]
Data (465 bytes)

```

```

0000 46 41 41 62 68 68 58 4d 59 52 43 74 57 45 35 6a FAAbhhXMYRCtWE5j
0010 51 42 70 4e 2b 46 38 46 4e 64 52 4d 42 42 42 42 QBpN+F8FNdRMBBBB
0020 56 46 4d 30 49 33 66 42 41 7a 52 79 62 54 47 52 VFM0I3fBAzRybTGR
0030 47 71 78 31 37 4d 37 32 55 77 57 6b 63 54 46 6e Gqx17M72UwWkcTFn
0040 57 77 58 66 4e 75 63 47 61 38 49 65 46 47 72 49 WwXfNucGa8IeFGrI
0050 47 33 35 65 4a 38 74 67 56 32 4b 49 44 42 71 31 G35eJ8tgV2KIDBq1
0060 67 31 6f 30 44 54 35 43 37 63 52 79 4e 4d 61 44 g1o0DT5C7cRyNMaD
0070 41 66 39 68 4a 4d 52 33 4a 72 79 74 30 75 63 47 Af9hJMR3Jryt0ucG
0080 4c 41 4c 43 7a 6b 30 6f 70 71 73 30 4a 38 32 6d LALCzk0opqs0J82m
0090 5a 47 43 7a 66 57 4a 5a 4d 52 4e 48 32 71 53 72 ZGCzfWJZMRNH2qSr
00a0 2b 43 61 47 46 4a 6b 66 39 55 54 4e 71 48 67 31 +CaGFJkf9UTNqHg1
00b0 34 67 50 61 36 4a 31 31 2b 49 2f 64 71 7a 61 44 4gPa6J11+I/dqzaD
00c0 74 72 33 76 6d 58 6f 4e 4f 61 2f 61 55 34 75 46 tr3vmXoNoa/aU4uF
00d0 4d 58 72 34 38 42 6e 37 78 70 46 78 49 48 63 65 MXr48Bn7xpFxIHce
00e0 48 2f 71 4b 69 53 36 46 64 64 36 52 79 6f 38 42 H/qKiS6Fdd6Ryo8B
00f0 31 6c 63 61 75 78 6c 65 57 6c 33 30 6a 47 6c 37 1lcauxleWl30jG17
0100 41 5a 62 6c 58 2f 7a 32 61 78 71 74 42 68 4f 69 AZb1X/z2axqtBh0i
0110 70 39 44 43 38 2b 50 75 67 41 35 5a 56 6a 4d 35 p9DC8+PugA5ZVjM5
0120 4d 36 78 37 50 74 46 39 53 69 50 37 78 62 43 78 M6x7PtF9SiP7xbCx
0130 54 46 6c 65 4c 6d 68 37 34 42 66 6f 52 73 63 77 TF1eLmh74BfoRscw
0140 69 58 6a 32 62 72 56 42 45 52 6d 55 6f 54 61 43 iXj2brVBERmUoTaC
0150 6b 6b 4b 31 2b 52 74 30 56 36 6c 36 34 44 66 61 kkK1+Rt0V6l64Dfa
0160 63 37 6e 59 51 49 51 46 62 2f 4d 4f 74 51 53 66 c7nYQIQFb/M0tQSf
0170 46 5a 65 77 37 52 64 75 38 50 66 7a 37 63 4f 69 FZew7Rdu8Ppz7c0i
0180 77 4c 62 67 5a 44 6b 6e 54 54 33 61 6d 6a 77 63 wLbgZDknTT3amjwc
0190 4c 32 67 41 75 2b 67 34 4e 4a 6c 49 46 31 34 2b L2gAu+g4NJlIF14+
01a0 62 57 78 7a 7a 5a 37 78 5a 4f 47 76 73 53 56 4e bWxzzZ7xZ0GvsSVN
01b0 34 47 33 32 61 61 70 45 51 73 78 53 76 37 45 4e 4G32aapEQsxSv7EN
01c0 46 30 70 44 62 46 38 30 65 42 4b 70 7a 32 73 3d F0pDbF80eBKpz2s=
01d0 0a .

```

```

Data: 464141626868584d595243745745356a5142704e2b463846...
[Length: 465]

```

Keyboard Data Packet Dump

No.	Time	Source	Destination	Protocol	Length	Info
16995	2276.295553440	192.168.43.1	192.168.43.239	TCP	219	44215 → 1234 [PSH, ACK] Seq=313881 Ack=26 Win=87808 Len=153 TSval=2846221 TSecr=6620840

Frame 16995: 219 bytes on wire (1752 bits), 219 bytes captured (1752 bits) on interface 0

```

Interface id: 0 (wlp6s0)
Encapsulation type: Ethernet (1)
Arrival Time: May 8, 2017 20:15:03.173202400 IST
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1494254703.173202400 seconds
[Time delta from previous captured frame: 0.083580830 seconds]
[Time delta from previous displayed frame: 0.083580830 seconds]
[Time since reference or first frame: 2276.295553440 seconds]
Frame Number: 16995
Frame Length: 219 bytes (1752 bits)
Capture Length: 219 bytes (1752 bits)
[Frame is marked: False]
[Frame is ignored: False]
[Protocols in frame: eth:ethertype:ip:tcp:data]
[Coloring Rule Name: TCP]
[Coloring Rule String: tcp]
Ethernet II, Src: Motorola_07:ec:72 (14:30:c6:07:ec:72), Dst: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
  Destination: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
    Address: ChiconyE_4c:ab:2b (4c:bb:58:4c:ab:2b)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Source: Motorola_07:ec:72 (14:30:c6:07:ec:72)
    Address: Motorola_07:ec:72 (14:30:c6:07:ec:72)
      ....0. .... = LG bit: Globally unique address (factory default)
      ....0. .... = IG bit: Individual address (unicast)
  Type: IPv4 (0x0800)
Internet Protocol Version 4, Src: 192.168.43.1, Dst: 192.168.43.239
  0100 .... = Version: 4
  ....0101 = Header Length: 20 bytes (5)
  Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    0000 00.. = Differentiated Services Codepoint: Default (0)
    ....00 = Explicit Congestion Notification: Not ECN-Capable Transport (0)
  Total Length: 205
  Identification: 0xea9b (60059)
  Flags: 0x02 (Don't Fragment)
    0... .... = Reserved bit: Not set
    .1.. .... = Don't fragment: Set
    ..0. .... = More fragments: Not set
  Fragment offset: 0
  Time to live: 64
  Protocol: TCP (6)
  Header checksum: 0x774e [validation disabled]
  [Header checksum status: Unverified]
  Source: 192.168.43.1
  Destination: 192.168.43.239
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
Transmission Control Protocol, Src Port: 44215, Dst Port: 1234, Seq: 313881, Ack: 26, Len: 153
  Source Port: 44215
  Destination Port: 1234
  [Stream index: 47]
  [TCP Segment Len: 153]
  Sequence number: 313881 (relative sequence number)
  [Next sequence number: 314034 (relative sequence number)]
  Acknowledgment number: 26 (relative ack number)
  Header Length: 32 bytes
  Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ...0 .... = Nonce: Not set
    ....0... = Congestion Window Reduced (CWR): Not set
    ....0... = ECN-Echo: Not set
    ....0. .... = Urgent: Not set
    ....1. .... = Acknowledgment: Set
    ....1... = Push: Set
    ....0... = Reset: Not set
    ....0... = Syn: Not set
    ....0... = Fin: Not set
  [TCP Flags: .....AP...]
  Window size value: 343
  [Calculated window size: 87808]

```



```

[Window size scaling factor: 256]
Checksum: 0xcdc2 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  No-Operation (NOP)
    Type: 1
    0... .... = Copy on fragmentation: No
    .00. .... = Class: Control (0)
    ...0 0001 = Number: No-Operation (NOP) (1)
  No-Operation (NOP)
    Type: 1
    0... .... = Copy on fragmentation: No
    .00. .... = Class: Control (0)
    ...0 0001 = Number: No-Operation (NOP) (1)
  Timestamps: TSval 2846221, TSecr 6620840
    Kind: Time Stamp Option (8)
    Length: 10
    Timestamp value: 2846221
    Timestamp echo reply: 6620840
[SEQ/ACK analysis]
  [iRTT: 0.000442249 seconds]
  [Bytes in flight: 153]
  [Bytes sent since last PSH flag: 153]
Data (153 bytes)

0000 46 41 41 62 69 77 66 78 62 6c 76 6a 4b 78 74 6a  FAAbiwfxblvjKxtj
0010 46 68 52 56 38 33 63 61 44 39 38 4a 55 69 78 35  FhRV83caD98JUix5
0020 57 6c 46 7a 51 54 4f 61 58 32 73 31 4c 46 36 54  WlFzQT0aX2s1LF6T
0030 49 61 5a 61 71 36 7a 6b 52 52 66 6c 48 43 78 50  IaZaq6zkRRflHCxP
0040 52 68 4c 4f 42 71 70 53 44 35 64 4d 48 41 65 43  RhLOBqpSD5dMHAeC
0050 58 45 46 51 4a 74 46 71 5a 6e 75 65 43 68 71 31  XEFQJtFqZnueChq1
0060 6c 56 6f 53 4f 54 49 42 74 59 4e 74 43 63 71 43  lVoS0TIBtYNtCcqC
0070 48 4c 51 76 4e 70 42 7a 46 71 7a 39 6e 4b 34 4b  HLQvNpBzFqz9nK4K
0080 4a 67 76 52 4a 73 6f 58 66 71 4c 58 4b 63 65 58  JgvRJsoXfqLXKceX
0090 53 76 62 69 73 78 66 64 0a                      Svbisx fd.
Data: 4641416269776678626c766a4b78746a4668525638336361...
[Length: 153]

```

5.2.2. App performance analysis

The app has been tested using several Android smartphones with varying specifications. Moto X (XT1052), Moto E, Nexus 7 are some of the devices that have been used for testing purposes. The app did not have any performance issues and worked as per expectation. The following results have been obtained for Moto X (XT1052) using Android monitoring tools for memory, CPU and network traffic usages.

Fig-20: 2D Mouse Movement Memory, CPU, Network Traffic Usages

Fig-21: 3D Mouse Movement Memory, CPU, Network Traffic Usages

Fig-22: Keyboard Memory, CPU, Network Traffic Usages

6. Project Cost Estimation

The cost estimation for the entire project has been computed as follows:

SLOC	Application	SLOC-by-Language (Sorted)
4044	Android	java=3166, xml=878
2688	PC	java=2688
Totals grouped by language (dominant language first):		
java:	5854	(86.96%)
xml:	878	(13.04%)
Software Project Class: Organic		
Total Physical Source Lines of Code (SLOC)		= 6,732
Development Effort Estimate, Person-Years (Person-Months)		= 1.48 (17.77)
(Basic COCOMO model, Person-Months = $2.4 * (KSLOC^{1.05})$)		
Schedule Estimate, Years (Months)		= 0.62 (7.46)
(Basic COCOMO model, Months = $2.5 * (person-months^{0.38})$)		
Estimated Average Number of Developers (Effort/Schedule)		= 2.38

Here SLOC refers to Source Lines of Code. The basic COCOMO model has been used since it is good for quick estimate of software costs. However it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.

■

7. Future Scope and Conclusion

In this project it is been clearly demonstrated that the usefulness of sensors in modern smartphones are not restricted to the host device itself. The availability of numerous connectivity options makes the process more easy and convenient. Data from wide range of different sensors can reliably converge to completely different meaningful information when used together appropriately. This is nearly impossible to accomplish when the sensors are used individually. Hence the concept of Sensor Fusion plays an important role for these kind of applications. Although Kalman Filter is effective in this scope, the initial predictions are not very accurate. Accuracy of prediction increases with time.

It can be observed that the 3D mouse movement is slightly unstable when nearing the corners of the screen. This is due to the physical limitations of the sensors. This instability might be decreased with the use of an additional low pass filter. But this low pass filter should only be applied when the mouse is nearing the screen corners to reduce the unnecessary computations.

Security is a widely known issue in the domain of Wireless Sensor Network. In this project an effort is made to resolve this problem. Widely acknowledged end-to-end encryption with secure client server handshaking and key agreement is result of the careful design choice.

Automatic Service Discovery makes the whole communication process easy and hassle free. The app user does not need to bother about setting IP address of the server (PC). Although this is a convenient process, the user needs to ensure that the router does not block UDP broadcast. Otherwise the connection module would not be able to detect the broadcast packets sent from the PC.

The keyboard is completely different in PC and modern smartphones. Both have their advantages and disadvantages. PC keyboard is robust, customisable and offers large numbers of keys as compared to smartphone keyboards. But PC keyboards are fixed hardware oriented and less flexible. Smartphone keyboards excels in the weak points of PC keyboards. These smartphone keyboards are software based virtual input devices. They are designed for faster typing with the help of virtual dictionary and autocorrect feature. Although they are very flexible they lack when it comes to the number of keys. In the proposed method advantages of both type of keyboards are combined while eliminating the disadvantages. Typing in PC can be done remotely with auto correct and dictionary support while preserving the access to Function keys, Ctrl key, Alt key, Shift key etc, which are essential for working with PC. File transfer, speaker control and iOS portability features are left for future work.

8. References

- [1] Mauro Conti, Nicola Dragoni, Viktor Lesyk, “*A Survey of Man In The Middle Attacks*”, IEEE Communications Surveys & Tutorials, vol. 18, issue 3, pp. 2027-2051, 29 March 2016.
- [2] D.Hazry, M.Sofian, A. Zul Azfar, “*Study of Inertial Measurement Unit Sensor*”, Proceedings of the International Conference on Man-Machine Systems (ICoMMS), 11 – 13 October 2009, Batu Ferringhi, Penang, Malaysia.
- [3] Se Gi Hong, Suman Srinivasan, Henning Schulzrinne, “*Accelerating Service Discovery in Ad-Hoc Zero Configuration Networking*”, Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE.
- [4] Michael Rubinstein, “*Introduction to recursive Bayesian filtering*”, MIT Computer Science and Artificial Intelligence Lab (CSAIL).
- [5] R. E. Kalman, “*A New Approach to Linear Filtering and Prediction Problems*”, Research Institute for Advanced Study,2 Baltimore, Md, 1960.
- [6] Ubejd Shala, Angel Rodriguez, Fredrik Frisk, Kamilla Klonowska, “*Indoor Positioning using Sensor-fusion in Android Devices*”, School of Health and Society, Department Computer Science, Kristianstad University, Sweden.
- [7] T. Dierks; E. Rescorla (August 2008). "The Transport Layer Security (TLS) Protocol, Version 1.2".
- [8] “*Announcing the Secure Hash Standard*”, Federal Information Processing Standards Publication 180-2, National Institute of Standards and Technology (NIST), August 1, 2002.
- [9] J. Daemen, V. Rijmen, “*AES proposal: Rijndael*”, National Institute of Standards and Technology (NIST), March 9, 2003.
- [10] D.J. Bernstein and et.al, “*Curve25519: new Diffie-Hellman speed records*”, Public Key Cryptography Lecture Notes in Computer Science. 3958 New York, Springer, pp. 207–228.
- [11] V. Miller, “*Use of elliptic curves in cryptography*”, “*CRYPTO. Lecture Notes in Computer Science*”, pp. 417–426, 1985, doi:10.1007/3-540-39799-X_31, ISBN 978-3-540-16463-0.
- [12] M.Lochter, J.Merkle, “*Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation*”, March 2010.
- [13] Erik B. Dam, Martin Koch, Martin Lillholm, “*Quaternions, Interpolation, Animation*”, Technical Report DIKU-TR-98/5, Department of Computer Science, University of Copenhagen, 1998.