Android (Unity exported)

Generated by Doxygen 1.8.16

Chapter 1

TDF02-145 Tablet

1.1 Introduction

This is the repository for Android (initially exported from Unity) app related matters for robot for Autism Spectrum Disorder therapy development funded through HEC TDF. The documentation folder contains all the code for Android remote division of HEC funded project TDF 02-145. There are multiple files contained in this folder. This code will run on most Android phones and Tablets.

1.2 JAVA

This section details the procedure of exporting and running a Unity program in the Android Studio IDE and running it with the JAVA code.

1.2.1 Exporting to JAVA inside Unity

- Go to File > Build Settings... or press Ctrl + Shift + B
- Select Android and click on Switch Platform
- Enable the Export Project Option. The "Build" option will change to "Export".
- Click on Export.
- Navigate to the TDF02-145 Tablet folder. Click on Select Folder.

1.2.2 Opening in Android Studio

- This folder can now be opening in the Android Studio IDE
- Select "Use Android Studio's SDK"
- MainActivity.java is named UnityPlayerActivity.java

1.2.2.1 Note

Exporting for the first time you may see the following comment
"// GENERATED BY UNITY. REMOVE THIS COMMENT TO PREVENT OVERWRITING WHEN EXPORTING AGAIN".
Delete this line. Otherwise the next time you Export it'll overwrite anything you've written in JAVA.

Exporting after already having exported you'll see a UnityPlayerActivity.NEW in addition to the UnityPlayerActivity.java you already edited.
You may delete this NEW file.

1.2.2.2 Author

Taha Shaheen

1.2.2.3 Version

chotuX

Chapter 2

Namespace Index

2.1 Packages

Here are the packages with brief descriptions (if available):

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 Package com

Packages

- package unity3d

6.2 Package com.unity3d

Packages

- package player

6.3 Package com.unity3d.player

Classes

- class MyAdminReceiver

  This allows the app to turn off the device.
- class MyBluetoothService

  Handles everything Bluetooth.
- class UnityPlayerActivity

  This is where everything important happens.

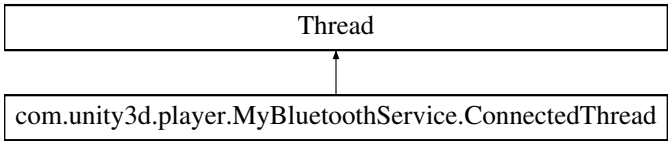
Chapter 7

Class Documentation

7.1 com.unity3d.player.MyBluetoothService.ConnectedThread Class Reference

Thread that manages a Bluetooth connection.

Inheritance diagram for com.unity3d.player.MyBluetoothService.ConnectedThread:



Public Member Functions

- ConnectedThread (BluetoothSocket socket)

  Constructor for the ConnectedThread class.
- void run ()

  Main code that runs in the Thread.
- void write (byte[ ] bytes)

  Sends data to the remote device.
- void cancel ()

  Closes the socket.

Private Attributes

- final BluetoothSocket mmSocket
- final InputStream mmInStream
- final OutputStream mmOutStream

7.1.1 Detailed Description

Thread that manages a Bluetooth connection.

- A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.
- This one manages a BluetoothSocket

Definition at line 235 of file MyBluetoothService.java.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 ConnectedThread()

```
com.unity3d.player.MyBluetoothService.ConnectedThread.ConnectedThread (
            BluetoothSocket socket )
```

Constructor for the ConnectedThread class.

Parameters

socket	RFCOMM Bluetooth Socket object

Manages a RFCOMM Bluetooth Socket

Definition at line 255 of file MyBluetoothService.java.

```
255                                                 {
256             Log.d(TAG, "ConnectedThread: Starting.");
257
258             mmSocket = socket;
259
260             // Temporary storage for  an input stream for reading bytes from this socket//
261             InputStream tmpIn = null;
262
263             // Temporary storage for  an output stream for writing bytes from this socket//
264             OutputStream tmpOut = null;
265
266             // dismiss the progressDialog when connection is established //
267             try {
268                 mProgressDialog.dismiss();
269             } catch (NullPointerException e) {
270                 e.printStackTrace();
271             }
272
273             // Get the InputStream and OutputStream that handle transmissions through the socket using
    getInputStream() and getOutputStream(), respectively. //
274             try {
275                 tmpIn = mmSocket.getInputStream();
276                 tmpOut = mmSocket.getOutputStream();
277             } catch (IOException e) {
278                 e.printStackTrace();
279             }
280
281             mmInStream = tmpIn;
282             mmOutStream = tmpOut;
283         }
```

References com.unity3d.player.MyBluetoothService.ConnectedThread.mmInStream, com.unity3d.player.My↩
BluetoothService.ConnectedThread.mmOutStream, com.unity3d.player.MyBluetoothService.ConnectedThread.↩

mmSocket, com.unity3d.player.MyBluetoothService.mProgressDialog, and com.unity3d.player.MyBluetooth↩
Service.TAG.

7.1.3 Member Function Documentation

7.1.3.1 cancel()

```
void com.unity3d.player.MyBluetoothService.ConnectedThread.cancel ( )
```

Closes the socket.

Closes the client socket and causes the thread to finish. Called from the main activity to shut down the connection.

Definition at line 348 of file MyBluetoothService.java.

```
348                                 {
349             try {
350                 mmSocket.close();
351             } catch (IOException e) { }
352         }
```

References com.unity3d.player.MyBluetoothService.ConnectedThread.mmSocket.

Referenced by com.unity3d.player.MyBluetoothService.cancelThreads().

7.1.3.2 run()

```
void com.unity3d.player.MyBluetoothService.ConnectedThread.run ( )
```

Main code that runs in the Thread.

Try Catch to read data being sent through the connection with the remote device.

Definition at line 289 of file MyBluetoothService.java.

```
289                             {
290             // buffer store for the stream //
291             byte[] buffer = new byte[1024];
292
293             // String buffer to hold incoming data //
294             String concatenatedString = "";
295
296             // bytes returned from read() //
297             int bytes;
298
299             // Keep listening to the InputStream until an exception occurs //
300             while (true) {
301                 // Read from the InputStream //
302                 try {
303                     bytes = mmInStream.read(buffer);
304                     String incomingMessage = new String(buffer, 0, bytes);
305                     Log.d(TAG, "InputStream: " + incomingMessage);
306
307                     concatenatedString = concatenatedString + incomingMessage;
308
309                     if (incomingMessage.contains("#")) {
310                         // send the String till # to be broken up //
311                         String[] messagePieces = breakUpString(concatenatedString.substring(0,
    concatenatedString.indexOf("#")));
312
313                         Log.d(TAG, concatenatedString.substring(0, concatenatedString.indexOf("#")));
314
```

```
315                         // empty the concatenatedString //
316                         concatenatedString = "";
317
318                         // send it to be processed //
319                         Log.d(TAG, messagePieces[0]);
320                         messageProcessorFunction(messagePieces);
321                     }
322                 } catch (IOException e) {
323                     Log.e(TAG, "write: Error reading Input Stream. " + e.getMessage() );
324                     break;
325                 }
326             }
327         }
```

References com.unity3d.player.MyBluetoothService.breakUpString(), com.unity3d.player.MyBluetoothService.↩ messageProcessorFunction(), com.unity3d.player.MyBluetoothService.ConnectedThread.mmInStream, and com.unity3d.player.MyBluetoothService.TAG.

7.1.3.3 write()

```
void com.unity3d.player.MyBluetoothService.ConnectedThread.write (
            byte[] bytes )
```

Sends data to the remote device.

Parameters

bytes	Array of bytes to be sent to the remote Bluetooth device

Called this from the main activity to send data to the remote device

Definition at line 334 of file MyBluetoothService.java.

```
334                                     {
335             String text = new String(bytes, Charset.defaultCharset());
336             Log.d(TAG, "write: Writing to output stream: " + text);
337             try {
338                 mmOutStream.write(bytes);
339             } catch (IOException e) {
340                 Log.e(TAG, "write: Error writing to output stream. " + e.getMessage() );
341             }
342         }
```

References com.unity3d.player.MyBluetoothService.ConnectedThread.mmOutStream, and com.unity3d.player.↩ MyBluetoothService.TAG.

Referenced by com.unity3d.player.MyBluetoothService.write().

7.1.4 Member Data Documentation

7.1.4.1 mmInStream

```
com.unity3d.player.MyBluetoothService.ConnectedThread.mmInStream  [private]
```

Holds a reference to an InputStream object, one of two types of streams. One that you can read data from

Definition at line 247 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.ConnectedThread.ConnectedThread(), and com.unity3d.↩ player.MyBluetoothService.ConnectedThread.run().

7.1.4.2 mmOutStream

```
com.unity3d.player.MyBluetoothService.ConnectedThread.mmOutStream  [private]
```

Holds a reference to an OutputStream object, one of two types of streams. One that you can either write data to

Definition at line 248 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.ConnectedThread.ConnectedThread(), and com.unity3d.↩player.MyBluetoothService.ConnectedThread.write().

7.1.4.3 mmSocket

```
final BluetoothSocket com.unity3d.player.MyBluetoothService.ConnectedThread.mmSocket  [private]
```

Holds the RFCOMM Socket object

Definition at line 239 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.ConnectedThread.cancel(), and com.unity3d.player.My↩BluetoothService.ConnectedThread.ConnectedThread().

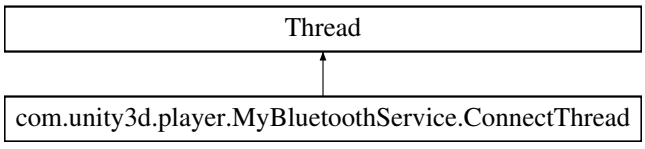
The documentation for this class was generated from the following file:

- MyBluetoothService.java

7.2 com.unity3d.player.MyBluetoothService.ConnectThread Class Reference

Client thread that initiates a Bluetooth connection.

Inheritance diagram for com.unity3d.player.MyBluetoothService.ConnectThread:



Public Member Functions

- ConnectThread (BluetoothDevice device, UUID uuid)

  Public constructor for the ConnectThread class.
- void run ()

  Main code that runs in the Thread.
- void cancel ()

  Closes the socket.

Private Attributes

- final BluetoothSocket mmSocket

  An instance of a Bluetooth socket.

7.2.1 Detailed Description

Client thread that initiates a Bluetooth connection.

- A Bluetooth client sends the connection request and the Bluetooth Server component accepts the request.
- A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.
- This one creates a BluetoothSocket

Definition at line 128 of file MyBluetoothService.java.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 ConnectThread()

```
com.unity3d.player.MyBluetoothService.ConnectThread.ConnectThread (
            BluetoothDevice device,
            UUID uuid )
```

Public constructor for the ConnectThread class.

Parameters

device	BluetoothDevice object
uuid	UUID object

Creates a RFCOMM Bluetooth Socket

Definition at line 144 of file MyBluetoothService.java.

```
144                                                     {
145
146             // Use a temporary object that is later assigned to mmSocket because mmSocket is final //
147             BluetoothSocket tmp = null;
148             mmDevice = device;
149             try {
150                 // Get a BluetoothSocket to connect with the given BluetoothDevice. //
151                 tmp = device.createRfcommSocketToServiceRecord(uuid);
152             } catch (IOException e) {
153                 Log.e(TAG, "Socket's create() method failed", e);
154             }
155             mmSocket = tmp;
156         }
```

References com.unity3d.player.MyBluetoothService.mmDevice, com.unity3d.player.MyBluetoothService.↩ ConnectThread.mmSocket, and com.unity3d.player.MyBluetoothService.TAG.

7.2.3 Member Function Documentation

7.2.3.1 cancel()

```
void com.unity3d.player.MyBluetoothService.ConnectThread.cancel ( )
```

Closes the socket.

Closes the client socket and causes the thread to finish. Called from the main activity to shut down the connection.

Definition at line 204 of file MyBluetoothService.java.

```
204                             {
205             try {
206                 mmSocket.close();
207             } catch (IOException e) {
208                 Log.e(TAG, "Could not close the client socket", e);
209             }
210         }
```

References com.unity3d.player.MyBluetoothService.ConnectThread.mmSocket, and com.unity3d.player.My↩BluetoothService.TAG.

Referenced by com.unity3d.player.MyBluetoothService.cancelThreads().

7.2.3.2 run()

```
void com.unity3d.player.MyBluetoothService.ConnectThread.run ( )
```

Main code that runs in the Thread.

Try Catch to attempt a connection to the remote device.

Definition at line 162 of file MyBluetoothService.java.

```
162                             {
163
164             // Cancel discovery because it otherwise slows down the connection //
165             bluetoothAdapter.cancelDiscovery();
166
167             try {
168                 // Connect to the remote device through the socket. This call blocks until it succeeds
    or throws an exception //
169                 mmSocket.connect();
170             } catch (IOException connectException) {
171                 // Unable to connect; close the socket and return //
172
173                 Log.d(TAG, "R.string.CONNECTION_TO_DEVICE_UNSUCCESSFUL");
174                 showToast(R.string.CONNECTION_TO_DEVICE_UNSUCCESSFUL);
175
176                 try {
177                     mmSocket.close();
178                 } catch (IOException closeException) {
179                     Log.e(TAG, "Could not close the client socket", closeException);
180                 }
181
182                 // dismiss the progressDialog //
183                 try {
184                     mProgressDialog.dismiss();
185                 } catch (NullPointerException e) {
186                     e.printStackTrace();
187                 }
188
189                 return;
190             }
```

```
191
192             // The connection attempt succeeded. Perform work associated with the connection in a
        separate thread. //
193             Log.d(TAG, "R.string.CONNECTION_TO_DEVICE_SUCCESSFUL");
194             showToast(R.string.CONNECTION_TO_DEVICE_SUCCESSFUL);
195
196             // Start a Thread that'll manage the work associated with the connection //
197             connected(mmSocket, mmDevice);
198         }
```

References com.unity3d.player.MyBluetoothService.bluetoothAdapter, com.unity3d.player.MyBluetoothService.↩ connected(), com.unity3d.player.MyBluetoothService.mmDevice, com.unity3d.player.MyBluetoothService.↩ ConnectThread.mmSocket, com.unity3d.player.MyBluetoothService.mProgressDialog, com.unity3d.player.My↩ BluetoothService.showToast(), and com.unity3d.player.MyBluetoothService.TAG.

7.2.4 Member Data Documentation

7.2.4.1 mmSocket

```
final BluetoothSocket com.unity3d.player.MyBluetoothService.ConnectThread.mmSocket  [private]
```

An instance of a Bluetooth socket.

- A socket is one endpoint of a two-way communication link
- The most common type of Bluetooth socket is RFCOMM, which is the type supported by the Android APIs

Definition at line 136 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.ConnectThread.cancel(), com.unity3d.player.My↩ BluetoothService.ConnectThread.ConnectThread(), and com.unity3d.player.MyBluetoothService.Connect↩ Thread.run().

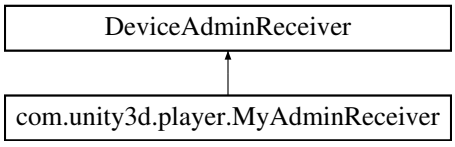
The documentation for this class was generated from the following file:

- MyBluetoothService.java

7.3 com.unity3d.player.MyAdminReceiver Class Reference

This allows the app to turn off the device.

Inheritance diagram for com.unity3d.player.MyAdminReceiver:



7.3.1 Detailed Description

This allows the app to turn off the device.

- DeviceAdminReceiver is the base class for implementing a device administration component
- This class provides a convenience for interpreting the raw intent actions that are sent by the system.
- Definitely read the `Android Developer's page` on this
- The manifest file was also updated when adding this class

Definition at line 23 of file MyAdminReceiver.java.

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

- MyAdminReceiver.java

7.4 com.unity3d.player.MyBluetoothService Class Reference

Handles everything Bluetooth.

Classes

- class ConnectedThread

  Thread that manages a Bluetooth connection.
- class ConnectThread

  Client thread that initiates a Bluetooth connection.

Public Member Functions

- MyBluetoothService (Context context)

  Constructor.
- synchronized void cancelThreads ()

  Cancels any running threads.
- void startClient (BluetoothDevice device, UUID uuid)
- abstract void showToast (int resourceID)

  Displays Toasts.
- abstract void messageProcessorFunction (String[ ] messagePieces)

  Processes the instruction received.
- void write (byte[ ] out)

  Writes to the any Bluetooth device.

Public Attributes

- ConnectedThread mConnectedThread

  Custom class ConnectedThread type object.

Package Attributes

- ProgressDialog mProgressDialog

  A progress dialog.

Static Package Attributes

- static Context mContext

  Reference to the Activity where the MyBluetoothService object instance is created.

Private Member Functions

- void connected (BluetoothSocket mmSocket, BluetoothDevice mmDevice)
- String[ ] breakUpString (String messageStream)

  Breaks up the String instruction received.

Private Attributes

- final BluetoothAdapter bluetoothAdapter

  A BluetoothAdapter object.
- ConnectThread mConnectThread

  Custom class ConnectThread type object.
- BluetoothDevice mmDevice

  Represents a remote Bluetooth device.

Static Private Attributes

- static final String TAG = "DEBUG_BLUETOOTH_SERVICE"

7.4.1 Detailed Description

Handles everything Bluetooth.

- Starts up and maintains Bluetooth connections between devices
- Sends and handles reception of messages from connected devices

Definition at line 32 of file MyBluetoothService.java.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 MyBluetoothService()

```
com.unity3d.player.MyBluetoothService.MyBluetoothService (
            Context context )
```

Constructor.

Parameters

context	Context of the Activity that creates an object of the MyBluetoothService class

The constructor used to create an object of the MyBluetoothService class

Definition at line 80 of file MyBluetoothService.java.

```
80                                                  {
81         mContext = context;
82         bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
83         cancelThreads();
84     }
```

References com.unity3d.player.MyBluetoothService.bluetoothAdapter, com.unity3d.player.MyBluetoothService.↩ cancelThreads(), and com.unity3d.player.MyBluetoothService.mContext.

7.4.3 Member Function Documentation

7.4.3.1 breakUpString()

```
String [] com.unity3d.player.MyBluetoothService.breakUpString (
             String messageStream )  [private]
```

Breaks up the String instruction received.

Uses the separator to break up the String instruction

Parameters

messageStream	the String instruction received

Returns

A String array containing the String instruction received, but in pieces

Definition at line 361 of file MyBluetoothService.java.

```
361                                                     {
362         String[] messagePieces = messageStream.split("_");
363         return messagePieces;
364     }
```

Referenced by com.unity3d.player.MyBluetoothService.ConnectedThread.run().

7.4.3.2 cancelThreads()

```
synchronized void com.unity3d.player.MyBluetoothService.cancelThreads ( )
```

Cancels any running threads.

Cancels any threads attempting to create a BluetoothSocket or any

Definition at line 90 of file MyBluetoothService.java.

```
90                                          {
91         Log.d(TAG, "start");
92
93         // Cancel any thread attempting to make a connection //
94         if (mConnectThread != null) {
95             mConnectThread.cancel();
96             mConnectThread = null;
97         }
98         if (mConnectedThread != null) {
99             mConnectedThread.cancel();
100             mConnectedThread = null;
101         }
102     }
```

References com.unity3d.player.MyBluetoothService.ConnectThread.cancel(), com.unity3d.player.MyBluetooth↩Service.ConnectedThread.cancel(), com.unity3d.player.MyBluetoothService.mConnectedThread, com.unity3d.↩player.MyBluetoothService.mConnectThread, and com.unity3d.player.MyBluetoothService.TAG.

Referenced by com.unity3d.player.MyBluetoothService.MyBluetoothService().

7.4.3.3 connected()

```
void com.unity3d.player.MyBluetoothService.connected (
             BluetoothSocket mmSocket,
             BluetoothDevice mmDevice )  [private]
```

Called to start a Thread to manages the Bluetooth connection

Parameters

mmSocket	RFCOMM Bluetooth Socket object
mmDevice	BluetoothDevice object

Definition at line 219 of file MyBluetoothService.java.

```
219                                                                         {
220         Log.d(TAG, "connected: Starting.");
221
222         // Start the thread to manage the connection and perform transmissions //
223         mConnectedThread = new ConnectedThread(mmSocket);
224
225         // Starts Thread //
226         mConnectedThread.start();
227     }
```

References com.unity3d.player.MyBluetoothService.mConnectedThread, and com.unity3d.player.MyBluetooth↩Service.TAG.

Referenced by com.unity3d.player.MyBluetoothService.ConnectThread.run().

7.4.3.4 messageProcessorFunction()

```
abstract void com.unity3d.player.MyBluetoothService.messageProcessorFunction (
             String[] messagePieces )  [abstract]
```

Processes the instruction received.

- I wanted to keep the message processing in the main Thread and keep the MyBluetoothService.java as free of these app specific "environmental factors" as possible
- So I made the messageProcessorFunction() abstract
- Any function calls from it can now be done in the main Thread

Parameters

messagePieces	A String array containing the String instruction received, but in pieces

Referenced by com.unity3d.player.MyBluetoothService.ConnectedThread.run().

7.4.3.5 showToast()

```
abstract void com.unity3d.player.MyBluetoothService.showToast (
            int resourceID )  [abstract]
```

Displays Toasts.

Parameters

resourceID	The resource ID of the String to be displayed in a Toast

Toasts only run on the main thread. This allows displaying of a Toast from another thread. By being an abstract method, it can be defined in the MainActivity which runs on the main thread. A method there, showToastMethod(), can then display the Toast.

Referenced by com.unity3d.player.MyBluetoothService.ConnectThread.run().

7.4.3.6 startClient()

```
void com.unity3d.player.MyBluetoothService.startClient (
            BluetoothDevice device,
            UUID uuid )
```

Called to start a Bluetooth connection

Parameters

device	Represents a remote Bluetooth device. A BluetoothDevice lets you create a connection with the respective device or query information about it, such as the name, address, class, and bonding state.
uuid	Universally Unique Identifier. UUIDs are not tied to particular devices. They identify software services. You just need both sides to use the same one.

Definition at line 110 of file MyBluetoothService.java.

```
110                                                                 {
111         // progress dialog appears //
112         mProgressDialog = ProgressDialog.show(mContext, "Connecting Bluetooth"
113                 , "Please Wait...", true);
114
115         mConnectThread = new ConnectThread(device, uuid);
116
117         // Starts Thread //
118         mConnectThread.start();
119     }
```

References com.unity3d.player.MyBluetoothService.mConnectThread, com.unity3d.player.MyBluetoothService.↩ mContext, and com.unity3d.player.MyBluetoothService.mProgressDialog.

Referenced by com.unity3d.player.UnityPlayerActivity.beginConnection().

7.4.3.7 write()

```
void com.unity3d.player.MyBluetoothService.write (
             byte[] out )
```

Writes to the any Bluetooth device.

Parameters

out	Array of bytes

- Called from the main activity
- Hands over the byte Array the Thread managing communication with the Bluetooth device
- Here for future use

Definition at line 391 of file MyBluetoothService.java.

```
391                                 {
392         //perform the write
393         mConnectedThread.write(out);
394     }
```

References com.unity3d.player.MyBluetoothService.mConnectedThread, and com.unity3d.player.MyBluetooth↩ Service.ConnectedThread.write().

7.4.4 Member Data Documentation

7.4.4.1 bluetoothAdapter

```
final BluetoothAdapter com.unity3d.player.MyBluetoothService.bluetoothAdapter  [private]
```

A BluetoothAdapter object.

A BluetoothAdapter lets you perform fundamental Bluetooth tasks, such as initiate device discovery, query a list of bonded (paired) devices, instantiate a BluetoothDevice using a known MAC address, and create a BluetoothServerSocket to listen for connection requests from other devices, and start a scan for Bluetooth LE devices.

Definition at line 44 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.MyBluetoothService(), and com.unity3d.player.MyBluetoothService.ConnectThread.run().

7.4.4.2 mConnectedThread

`com.unity3d.player.MyBluetoothService.mConnectedThread`

Custom class ConnectedThread type object.

Extends `Thread`

Definition at line 61 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.cancelThreads(), com.unity3d.player.MyBluetoothService.connected(), and com.unity3d.player.MyBluetoothService.write().

7.4.4.3 mConnectThread

`com.unity3d.player.MyBluetoothService.mConnectThread [private]`

Custom class ConnectThread type object.

Extends `Thread`

Definition at line 60 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.cancelThreads(), and com.unity3d.player.MyBluetoothService.startClient().

7.4.4.4 mContext

`Context com.unity3d.player.MyBluetoothService.mContext [static], [package]`

Reference to the Activity where the MyBluetoothService object instance is created.

Services, such as this one, require a Context from the Activity that creates an object of its type to hook it to that Activity and provide it access to the application specific resources.

Definition at line 38 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.MyBluetoothService(), and com.unity3d.player.MyBluetoothService.startClient().

7.4.4.5 mmDevice

```
BluetoothDevice com.unity3d.player.MyBluetoothService.mmDevice  [private]
```

Represents a remote Bluetooth device.

A BluetoothDevice lets you create a connection with the respective device or query information about it, such as the name, address, class, and bonding state.

Definition at line 67 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.ConnectThread.ConnectThread(), and com.unity3d.player.MyBluetoothService.ConnectThread.run().

7.4.4.6 mProgressDialog

```
ProgressDialog com.unity3d.player.MyBluetoothService.mProgressDialog  [package]
```

A progress dialog.

A dialog showing a progress indicator and an optional text message or view. Only a text message or a view can be used at the same time.

Definition at line 50 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.ConnectedThread.ConnectedThread(), com.unity3d.player.MyBluetoothService.ConnectThread.run(), and com.unity3d.player.MyBluetoothService.startClient().

7.4.4.7 TAG

```
final String com.unity3d.player.MyBluetoothService.TAG = "DEBUG_BLUETOOTH_SERVICE"  [static],
[private]
```

Debugging tool

Definition at line 73 of file MyBluetoothService.java.

Referenced by com.unity3d.player.MyBluetoothService.ConnectThread.cancel(), com.unity3d.player.MyBluetoothService.cancelThreads(), com.unity3d.player.MyBluetoothService.connected(), com.unity3d.player.MyBluetoothService.ConnectedThread.ConnectedThread(), com.unity3d.player.MyBluetoothService.ConnectThread.ConnectThread(), com.unity3d.player.MyBluetoothService.ConnectThread.run(), com.unity3d.player.MyBluetoothService.ConnectedThread.run(), and com.unity3d.player.MyBluetoothService.ConnectedThread.write().

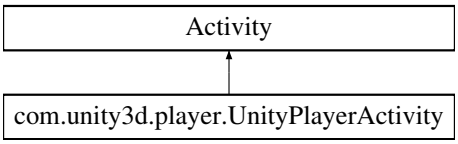
The documentation for this class was generated from the following file:

- MyBluetoothService.java

7.5 com.unity3d.player.UnityPlayerActivity Class Reference

This is where everything important happens.

Inheritance diagram for com.unity3d.player.UnityPlayerActivity:



Public Member Functions

- void onLowMemory ()
- void onTrimMemory (int level)
- void onConfigurationChanged (Configuration newConfig)
- void onWindowFocusChanged (boolean hasFocus)
- void showToastMethod (final String message)

  Toasts for threads other than the main.

Static Public Attributes

- static MyBluetoothService myBluetoothService

  Handles Bluetooth stuff.

Protected Member Functions

- void onCreate (Bundle savedInstanceState)

  fires when the system first creates the activity.
- void onDestroy ()

  The final call you receive before your activity is destroyed.
- void onPause ()

  Pause Unity.
- void onResume ()

  Resume Unity.
- void onActivityResult (int requestCode, int resultCode, @Nullable Intent data)

  Called back after focus is returned from process started by startActivityForResult()
- String updateUnityCommandLineArguments (String cmdLine)

  Created when exporting from Unity.
- void onNewIntent (Intent intent)

  Created when exporting from Unity.

Protected Attributes

- UnityPlayer mUnityPlayer

Package Attributes

- final int SAD = 1
- final int SURPRISED = 2
- final int ANGRY = 3
- final int IDLE = 4
- MediaPlayer voiceYourEmotion

  Media Player for running audio files that "voice emotion".
- String eyebrowsObject

  References Unity object.
- String mouthObject

  References Unity object.
- String tearObject

  References Unity object.
- String ouchZoneObject

  References Unity object.
- String eyelidsObject

  References Unity object.
- String setEmotionFunction

  Represents Unity object.
- String setSpeakingFunction

  Represents Unity object.
- String setEyePokeEnabledStateFunction

  Represents Unity object.
- String goToSleepFunction

  Represents Unity object.
- String stopSpeaking

  Unity parameter.
- String falseString
- final Handler handler = new Handler()
- String backgroundColor = "000000"

Static Package Attributes

- static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")

  Universally Unique Identifier (UUID)

Private Member Functions

- void speakOut (String[ ] messagePieces)

  Text to speech code.
- void setEmotion (String emotionString)

  Emotion control code.
- String emotionBackgroundColor (int emotion)
- void beginConnection ()

  Establishes a Bluetooth serial communication.
- void settingsUpdate (String[ ] messagePieces)

  Poke enable/disable and turn off tab.

Private Attributes

- final boolean TESTING = false

  Debugging tool.
- final int HAPPY = 0
- final BroadcastReceiver mBTStateBroadcastReceiver

  a BroadcastReceiver type object
- String MAC_ADDRESS
- String mainCameraObject

  References Unity object.
- String changeBackgroundColorFunction

  Represents Unity object.
- String startSpeaking

  Unity parameter.
- String trueString

  Unity parameter.
- TextToSpeech textToSpeech

  For text to speech functionality.
- BluetoothAdapter bluetoothAdapter

  A BluetoothAdapter object.
- int REQUEST_ENABLE_BT = 1
- AudioManager audioManager

  Manages the volume when the app is running.
- int savedVolume

  Saves the current volume setting of the device.
- DevicePolicyManager mDevicePolicyManager

  Public interface for managing policies enforced on a device.
- ComponentName mComponentName

Static Private Attributes

- static String TAG = "DEBUG_UNITY_PLAYER"
- static final int ADMIN_INTENT = 15

7.5.1 Detailed Description

This is where everything important happens.

This is the first screen to appear when the user launches the app. This handles everything and calls everything.

Definition at line 42 of file UnityPlayerActivity.java.

7.5.2 Member Function Documentation

7.5.2.1 beginConnection()

```
void com.unity3d.player.UnityPlayerActivity.beginConnection ( )  [private]
```

Establishes a Bluetooth serial communication.

Fetches device info from paired devices.

Definition at line 654 of file UnityPlayerActivity.java.

```
654                                     {
655
656         boolean deviceFound = false;
657
658         Set<BluetoothDevice> pairedDevices = bluetoothAdapter.getBondedDevices();
659         if (pairedDevices.size() > 0) {
660             // There are paired devices. Get the name and address of each paired device. //
661             for (BluetoothDevice device : pairedDevices) {
662                 String deviceHardwareAddress = device.getAddress(); // MAC address
663                 if (deviceHardwareAddress.equals(MAC_ADDRESS)) {
664                     // This bit matches the MAC address of your "face device" to a paired device's //
665                     // Then establishes a connection on a separate thread //
666                     deviceFound = true;
667                     myBluetoothService.startClient(device, MY_UUID);
668                     break;
669                 }
670             }
671             if (!deviceFound) {
672                 Toast.makeText(getBaseContext(), "ERROR: Device with MAC address " + MAC_ADDRESS + " not
    paired", Toast.LENGTH_LONG).show();
673                 this.finishAffinity();
674             }
675
676         } else {
677             Toast.makeText(getBaseContext(), "ERROR:" + getString(R.string.NO_PAIRED_DEVICES),
    Toast.LENGTH_LONG).show();
678             this.finishAffinity();
679         }
680     }
```

References com.unity3d.player.UnityPlayerActivity.bluetoothAdapter, com.unity3d.player.UnityPlayerActivity.MAC_ADDRESS, com.unity3d.player.UnityPlayerActivity.MY_UUID, com.unity3d.player.UnityPlayerActivity.myBluetoothService, and com.unity3d.player.MyBluetoothService.startClient().

Referenced by com.unity3d.player.UnityPlayerActivity.onActivityResult(), and com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.2.2 emotionBackgroundColor()

```
String com.unity3d.player.UnityPlayerActivity.emotionBackgroundColor (
            int emotion )  [private]
```

Returns emotion color

Parameters

emotion	emotion integer

Returns

String containing the hex code of an emotion background color.

Definition at line 628 of file UnityPlayerActivity.java.

```
628                                                     {
629         switch (emotion) {
630             case HAPPY:
631                 backgroundColor = "FFB400";
632                 break;
633             case SAD:
634                 backgroundColor = "058548";
635                 break;
636             case SURPRISED:
637                 backgroundColor = "FFFFFF";
638                 break;
639             case ANGRY:
640                 backgroundColor = "B60000";
641                 break;
642             case IDLE:
643                 backgroundColor = "878787";
644                 break;
645             default:
646         }
647         return (backgroundColor);
648     }
```

References com.unity3d.player.UnityPlayerActivity.ANGRY, com.unity3d.player.UnityPlayerActivity.backgroundColor, com.unity3d.player.UnityPlayerActivity.HAPPY, com.unity3d.player.UnityPlayerActivity.IDLE, com.unity3d.player.UnityPlayerActivity.SAD, and com.unity3d.player.UnityPlayerActivity.SURPRISED.

Referenced by com.unity3d.player.UnityPlayerActivity.setEmotion().

7.5.2.3 onActivityResult()

```
void com.unity3d.player.UnityPlayerActivity.onActivityResult (
             int requestCode,
             int resultCode,
             @Nullable Intent data )  [protected]
```

Called back after focus is returned from process started by startActivityForResult()

Parameters

requestCode	the requestCode passed as the second parameter to startActivityForResult(), here it is REQUEST_ENABLE_BT
resultCode	Possible values - RESULT_OK or RESULT_CANCELED
data	Optional parameter. An Intent, which can return result data to the caller. `@Nullable` denotes that a value can be null.

If enabling Bluetooth succeeds, this activity receives the RESULT_OK result code in the onActivityResult() callback. If Bluetooth was not enabled due to an error (or the user responded "No") then the result code is RESULT_CANCELED.

Definition at line 744 of file UnityPlayerActivity.java.

```
744                                                                                 {
745         if (requestCode == REQUEST_ENABLE_BT && resultCode == RESULT_OK) {
746             beginConnection();
747         }
748         if (resultCode == RESULT_CANCELED)
749             Toast.makeText(this, "Unable to access Bluetooth", Toast.LENGTH_SHORT).show();
750     }
```

References com.unity3d.player.UnityPlayerActivity.beginConnection(), and com.unity3d.player.UnityPlayerActivity.REQUEST_ENABLE_BT.

7.5.2.4 onConfigurationChanged()

```
void com.unity3d.player.UnityPlayerActivity.onConfigurationChanged (
            Configuration newConfig )
```

Notifies Unity of any configuration change This ensures the layout will be correct

Definition at line 484 of file UnityPlayerActivity.java.

```
484                                                         {
485         super.onConfigurationChanged(newConfig);
486         mUnityPlayer.configurationChanged(newConfig);
487     }
```

References com.unity3d.player.UnityPlayerActivity.mUnityPlayer.

7.5.2.5 onCreate()

```
void com.unity3d.player.UnityPlayerActivity.onCreate (
            Bundle savedInstanceState )  [protected]
```

fires when the system first creates the activity.

Parameters

savedInstanceState	A Bundle object containing the activity's previously saved state. If the activity has never existed before, the value of the Bundle object is null. (Bundle is generally used for passing data between various activities of android.)

In the onCreate() method, you perform basic application startup logic that should happen only once for the entire life of the activity.

Definition at line 249 of file UnityPlayerActivity.java.

```
249                                                 {
250
251         // Make this activity, full screen //
252         requestWindowFeature(Window.FEATURE_NO_TITLE);
253
254         // By calling super.onCreate(savedInstanceState);, you tell the Dalvik VM (an android virtual
      machine optimized for mobile devices) to run your code in addition to the existing code in the
      onCreate() of the parent class. If you leave out this line, then only your code is run. The existing
      code is ignored completely. //
255         super.onCreate(savedInstanceState);
256
257         // This bit of code preserves the state of the app. Switching focus back to this app from
      another app will not lead to a long load time. //
258         String cmdLine = updateUnityCommandLineArguments(getIntent().getStringExtra("unity"));
259         getIntent().putExtra("unity", cmdLine);
260
261
262         // Constructor for UnityPlayer View object //
263         mUnityPlayer = new UnityPlayer(this);
264
265         // the Activity class takes care of creating a window for you in which you can place your UI
      with setContentView(View) //
266         setContentView(mUnityPlayer);
267
268         // Call this to try to give focus to a specific view //
269         mUnityPlayer.requestFocus();
270
271
272         audioManager = (AudioManager) getSystemService(Context.AUDIO_SERVICE);
273         // Save volume //
274         savedVolume = audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
275         // Set volume to max //
```

```
276         audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
    audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC), 0);
277
278         // String code //
279         mainCameraObject = getString(R.string.MAIN_CAMERA_OBJECT);
280         eyebrowsObject = getString(R.string.EYEBROWS_OBJECT);
281         mouthObject = getString(R.string.MOUTH_OBJECT);
282         tearObject = getString(R.string.TEAR_OBJECT);
283         ouchZoneObject = getString(R.string.OUCH_ZONE);
284         eyelidsObject = getString(R.string.EYELIDS_OBJECT);
285
286         changeBackgroundColorFunction = getString(R.string.CHANGE_BACKGROUND_COLOR_FUNCTION);
287         setEmotionFunction = getString(R.string.SET_EMOTION_FUNCTION);
288         setSpeakingFunction = getString(R.string.SET_SPEAKING_FUNCTION);
289         setEyePokeEnabledStateFunction = getString(R.string.SET_EYE_POKE_ENABLED_STATE_FUNCTION);
290         goToSleepFunction = getString(R.string.GO_TO_SLEEP_FUNCTION);
291
292         startSpeaking = getString(R.string.START_SPEAKING);
293         stopSpeaking = getString(R.string.STOP_SPEAKING);
294         trueString = getString(R.string.TRUE);
295         falseString = getString(R.string.FALSE);
296
297         //Bluetooth code //
298         if (TESTING)
299             MAC_ADDRESS = getString(R.string.TESTING_MAC_ADDRESS_1); // MAC_ADDRESS IN USE - EASIER TO
    CHANGE HERE THAN ALL OVER THE PLACE //
300         else
301             MAC_ADDRESS = getString(R.string.MAC_ADDRESS); // MAC_ADDRESS IN USE - EASIER TO CHANGE HERE
    THAN ALL OVER THE PLACE //
302
303         myBluetoothService = new MyBluetoothService(UnityPlayerActivity.this) {
304             @Override
305             public void showToast(int resourceID) {
306                 // Will receive a resourceID, convert it into a String, and send it to showToastMethod()
    //
307                 showToastMethod(getString(resourceID));
308             }
309
310             @Override
311             public void messageProcessorFunction(String[] messagePieces) {
312                 // Will receive String array, and call the appropriate method for execution //
313                 Log.d(TAG, messagePieces[0]);
314                 switch (messagePieces[0].trim()) {
315                     case "E":
316                         setEmotion(messagePieces[1]);
317                         break;
318                     case "G":
319                         speakOut(messagePieces);
320                         break;
321                     case "C":
322                         settingsUpdate(messagePieces);
323                         break;
324                 }
325             }
326         };
327
328         // Get a handle/reference to the default local Bluetooth adapter of the device being used //
329         bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
330         if (bluetoothAdapter == null)
331             Toast.makeText(this, "Your device does not support Bluetooth", Toast.LENGTH_LONG).show();
332         else {
333             if (!bluetoothAdapter.isEnabled()) {
334
335                 // Creating an Intent //
336                 Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
337
338                 // A dialog appears requesting user permission to enable Bluetooth. //
339                 // If the user responds "Yes", the system begins to enable Bluetooth //
340                 // Focus returns to your application once the process completes (or fails) //
341                 // onActivityResult() that gets called upon return of focus //
342                 startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
343
344             } else {
345                 beginConnection();
346             }
347             // Informs us when BT condition changes //
348             // Registers a BroadcastReceiver to be run in the main activity thread. //
349             // The receiver will be called with any broadcast Intent that matches filter //
350             // The Broadcast Receiver implementation is outside onCreate() //
351             registerReceiver(mBTStateBroadcastReceiver, new
    IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED));
352
353             //Text to speech code //
354             textToSpeech = new TextToSpeech(getApplicationContext(), new TextToSpeech.OnInitListener() {
355                 @Override
356                 public void onInit(int status) {
357                     if (status != TextToSpeech.ERROR)
```

```
358                         textToSpeech.setLanguage(Locale.ENGLISH);
359
360                     // This bit of code handles mouth animation during TTS //
361                     textToSpeech.setOnUtteranceProgressListener(new UtteranceProgressListener() {
362                         @Override
363                         public void onStart(String utteranceId) {
364                             UnityPlayer.UnitySendMessage(mouthObject, setSpeakingFunction,
    startSpeaking);
365                         }
366
367                         @Override
368                         public void onDone(String utteranceId) {
369                             UnityPlayer.UnitySendMessage(mouthObject, setSpeakingFunction,
    stopSpeaking);
370                         }
371
372                         @Override
373                         public void onError(String utteranceId) {
374                         }
375                     });
376                 }
377             });
378
379         }
380
381         // Screen Control code //
382         mDevicePolicyManager = (DevicePolicyManager)getSystemService(Context.DEVICE_POLICY_SERVICE);
383
384         // Ask the user to add a new device administrator to the system. //
385         mComponentName = new ComponentName(this, MyAdminReceiver.class);
386         Intent intent = new Intent(DevicePolicyManager.ACTION_ADD_DEVICE_ADMIN);
387         intent.putExtra(DevicePolicyManager.EXTRA_DEVICE_ADMIN, mComponentName);
388         startActivityForResult(intent, ADMIN_INTENT);
389     }
```

References com.unity3d.player.UnityPlayerActivity.ADMIN_INTENT, com.unity3d.player.UnityPlayerActivity.audio↩Manager, com.unity3d.player.UnityPlayerActivity.beginConnection(), com.unity3d.player.UnityPlayerActivity.↩bluetoothAdapter, com.unity3d.player.UnityPlayerActivity.changeBackgroundColorFunction, com.unity3d.player.↩UnityPlayerActivity.eyebrowsObject, com.unity3d.player.UnityPlayerActivity.eyelidsObject, com.unity3d.player.↩UnityPlayerActivity.falseString, com.unity3d.player.UnityPlayerActivity.goToSleepFunction, com.unity3d.player.↩UnityPlayerActivity.MAC_ADDRESS, com.unity3d.player.UnityPlayerActivity.mainCameraObject, com.unity3d.↩player.UnityPlayerActivity.mBTStateBroadcastReceiver, com.unity3d.player.UnityPlayerActivity.mComponentName, com.unity3d.player.UnityPlayerActivity.mDevicePolicyManager, com.unity3d.player.UnityPlayerActivity.mouth↩Object, com.unity3d.player.UnityPlayerActivity.mUnityPlayer, com.unity3d.player.UnityPlayerActivity.myBluetooth↩Service, com.unity3d.player.UnityPlayerActivity.ouchZoneObject, com.unity3d.player.UnityPlayerActivity.RE↩QUEST_ENABLE_BT, com.unity3d.player.UnityPlayerActivity.savedVolume, com.unity3d.player.UnityPlayer↩Activity.setEmotion(), com.unity3d.player.UnityPlayerActivity.setEmotionFunction, com.unity3d.player.UnityPlayer↩Activity.setEyePokeEnabledStateFunction, com.unity3d.player.UnityPlayerActivity.setSpeakingFunction, com.↩unity3d.player.UnityPlayerActivity.settingsUpdate(), com.unity3d.player.UnityPlayerActivity.showToastMethod(), com.unity3d.player.UnityPlayerActivity.speakOut(), com.unity3d.player.UnityPlayerActivity.startSpeaking, com.↩unity3d.player.UnityPlayerActivity.stopSpeaking, com.unity3d.player.UnityPlayerActivity.TAG, com.unity3d.player.↩UnityPlayerActivity.tearObject, com.unity3d.player.UnityPlayerActivity.TESTING, com.unity3d.player.UnityPlayer↩Activity.textToSpeech, com.unity3d.player.UnityPlayerActivity.trueString, and com.unity3d.player.UnityPlayer↩Activity.updateUnityCommandLineArguments().

7.5.2.6 onDestroy()

```
void com.unity3d.player.UnityPlayerActivity.onDestroy ( )  [protected]
```

The final call you receive before your activity is destroyed.

This opportunity is used to unregister the BroadcastReceiver, mBTStateBroadcastReceiver

Definition at line 396 of file UnityPlayerActivity.java.

```
396                             {
397         mUnityPlayer.destroy();
398         super.onDestroy();
```

```
399
400         // Unregistering broadcast listener to free up resources //
401         unregisterReceiver(mBTStateBroadcastReceiver);
402         super.onDestroy();
403
404         // Shutting down TextToSpeech //
405         if (textToSpeech != null) {
406             textToSpeech.stop();
407             textToSpeech.shutdown();
408         }
409
410         // Release MediaPlayer object //
411         if (voiceYourEmotion != null) voiceYourEmotion.release();
412
413         // set volume to value it was at before app started //
414         audioManager.setStreamVolume(AudioManager.STREAM_MUSIC, savedVolume, 0);
415
416     }
```

References com.unity3d.player.UnityPlayerActivity.audioManager, com.unity3d.player.UnityPlayerActivity.mBT↩StateBroadcastReceiver, com.unity3d.player.UnityPlayerActivity.mUnityPlayer, com.unity3d.player.UnityPlayer↩Activity.savedVolume, com.unity3d.player.UnityPlayerActivity.textToSpeech, and com.unity3d.player.UnityPlayer↩Activity.voiceYourEmotion.

7.5.2.7 onLowMemory()

```
void com.unity3d.player.UnityPlayerActivity.onLowMemory ( )
```

When the phone's memory is low, the background processes will be killed by framework. Unity will be informed as well

Definition at line 458 of file UnityPlayerActivity.java.

```
458                                 {
459         super.onLowMemory();
460         mUnityPlayer.lowMemory();
461     }
```

References com.unity3d.player.UnityPlayerActivity.mUnityPlayer.

7.5.2.8 onNewIntent()

```
void com.unity3d.player.UnityPlayerActivity.onNewIntent (
             Intent intent )  [protected]
```

Created when exporting from Unity.

Note

Created when exporting from Unity. As far as I can tell, it helps load the app back up quickly when returning from another app.

- To support deep linking, we need to make sure that the client can get access to the last sent intent.
- The clients access this through a JNI api that allows them to get the intent set on launch.
- To update that after launch we have to manually replace the intent with the one caught here.
- When the activity is re-launched while at the top of the activity stack instead of a new instance of the activity being started, onNewIntent() will be called on the existing instance with the Intent that was used to re-launch it.
- More here.

Parameters

intent	The new intent that was started for the activity

Definition at line 778 of file UnityPlayerActivity.java.

```
778                                                 {
779         setIntent(intent);
780         mUnityPlayer.newIntent(intent);
781     }
```

References com.unity3d.player.UnityPlayerActivity.mUnityPlayer.

7.5.2.9 onPause()

```
void com.unity3d.player.UnityPlayerActivity.onPause ( )  [protected]
```

Pause Unity.

Called when another app is run on the device

Definition at line 423 of file UnityPlayerActivity.java.

```
423                                 {
424         super.onPause();
425         mUnityPlayer.pause();
426         if (textToSpeech != null) {
427             textToSpeech.stop();
428         }
429         //set volume to value it was at before app started //
430         audioManager.setStreamVolume(AudioManager.STREAM_MUSIC, savedVolume, 0);
431
432         // Stop MediaPlayer object //
433         if (voiceYourEmotion != null) {
434             UnityPlayer.UnitySendMessage(mouthObject, setSpeakingFunction, stopSpeaking);
435             voiceYourEmotion.stop();
436         }
437     }
```

References com.unity3d.player.UnityPlayerActivity.audioManager, com.unity3d.player.UnityPlayerActivity.mouthObject, com.unity3d.player.UnityPlayerActivity.mUnityPlayer, com.unity3d.player.UnityPlayerActivity.savedVolume, com.unity3d.player.UnityPlayerActivity.setSpeakingFunction, com.unity3d.player.UnityPlayerActivity.stopSpeaking, com.unity3d.player.UnityPlayerActivity.textToSpeech, and com.unity3d.player.UnityPlayerActivity.voiceYourEmotion.

7.5.2.10 onResume()

```
void com.unity3d.player.UnityPlayerActivity.onResume ( )  [protected]
```

Resume Unity.

Called when focus return to this app

Definition at line 444 of file UnityPlayerActivity.java.

```
444                                  {
445         super.onResume();
446         mUnityPlayer.resume();
447
448         // Volume to max //
449         audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,
        audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC), 0);
450
451     }
```

References com.unity3d.player.UnityPlayerActivity.audioManager, and com.unity3d.player.UnityPlayerActivity.mUnityPlayer.

7.5.2.11 onTrimMemory()

```
void com.unity3d.player.UnityPlayerActivity.onTrimMemory (
            int level )
```

Callback for finer-grained memory management

Parameters

level	different types of clues about memory availability

Definition at line 468 of file UnityPlayerActivity.java.

```
468                                     {
469         super.onTrimMemory(level);
470
471         if (level == TRIM_MEMORY_RUNNING_CRITICAL) {
472             // The device is running extremely low on memory. //
473             // App is not yet considered a killable process, but the system will begin killing
      background processes if apps do not release resources. //
474             // Releasing non-critical resources now to prevent performance degradation. //
475             mUnityPlayer.lowMemory();
476         }
477     }
```

References com.unity3d.player.UnityPlayerActivity.mUnityPlayer.

7.5.2.12 onWindowFocusChanged()

```
void com.unity3d.player.UnityPlayerActivity.onWindowFocusChanged (
            boolean hasFocus )
```

Notifies Unity of the focus change

Definition at line 493 of file UnityPlayerActivity.java.

```
493                                                 {
494         super.onWindowFocusChanged(hasFocus);
495         mUnityPlayer.windowFocusChanged(hasFocus);
496     }
```

References com.unity3d.player.UnityPlayerActivity.mUnityPlayer.

7.5.2.13 setEmotion()

```
void com.unity3d.player.UnityPlayerActivity.setEmotion (
            String emotionString )  [private]
```

Emotion control code.

- The setEmotion() function (the one that sends commands to Unity) takes in integers
- The string received gets converted to an int
- Then that integer is used to work out mouth, eyebrow and other dynamic face properties

Parameters

emotionString	String containing information on what emotion is to be displayed

Definition at line 529 of file UnityPlayerActivity.java.

```
529                                                 {
530
531         // Was an audio also requested //
532         boolean voiceYourEmotionInstruction = false;
533         if(emotionString.substring(emotionString.length() - 1).equals("V")) {
534             voiceYourEmotionInstruction = true;
535             emotionString = emotionString.substring(0, emotionString.length() - 1);
536         }
537
538         // If audio already playing, stop it //
539         if(voiceYourEmotion!=null){
540             voiceYourEmotion.stop();
541             voiceYourEmotion.release();
542         }
543         voiceYourEmotion = null;
544
545         // String to int conversion //
546         int emotionInteger;
547         switch (emotionString.toUpperCase()) {
548             case "HAPPY":
549                 emotionInteger = HAPPY;
550                 if(voiceYourEmotionInstruction) voiceYourEmotion =
    MediaPlayer.create(UnityPlayerActivity.this, R.raw.happy);
551                 break;
552             case "SAD":
553                 emotionInteger = SAD;
554                 if(voiceYourEmotionInstruction) voiceYourEmotion =
    MediaPlayer.create(UnityPlayerActivity.this, R.raw.sad);
555                 break;
556             case "ANGRY":
557                 emotionInteger = ANGRY;
558                 if(voiceYourEmotionInstruction) voiceYourEmotion =
    MediaPlayer.create(UnityPlayerActivity.this, R.raw.angry);
559                 break;
560             case "SURPRISED":
561                 emotionInteger = SURPRISED;
562                 break;
563             case "IDLE":
564                 emotionInteger = IDLE;
565                 break;
566             default:
567                 emotionInteger = 10;    //because 10 isn't a registered emotion, nothing happens //
568         }
569
570         // This part talks to UNITY. Using the emotionInteger //
571         // The format is UnityPlayer.UnitySendMessage(unityObjectName, methodName, parameterToPass) //
572         // All must be String values //
573         String backgroundColor = emotionBackgroundColor(emotionInteger);
574         UnityPlayer.UnitySendMessage(mainCameraObject, changeBackgroundColorFunction, backgroundColor);
575         switch (emotionInteger) {
576             case HAPPY:
577                 UnityPlayer.UnitySendMessage(eyebrowsObject, setEmotionFunction, "HAPPY");
578                 UnityPlayer.UnitySendMessage(mouthObject, setEmotionFunction, "HAPPY");
579                 UnityPlayer.UnitySendMessage(tearObject, setEmotionFunction, "HAPPY");
580                 break;
581             case SAD:
582                 UnityPlayer.UnitySendMessage(eyebrowsObject, setEmotionFunction, "SAD");
583                 UnityPlayer.UnitySendMessage(mouthObject, setEmotionFunction, "SAD");
584                 UnityPlayer.UnitySendMessage(tearObject, setEmotionFunction, "SAD");
585                 break;
586             case SURPRISED:
587                 UnityPlayer.UnitySendMessage(eyebrowsObject, setEmotionFunction, "SURPRISED");
588                 UnityPlayer.UnitySendMessage(mouthObject, setEmotionFunction, "SURPRISED");
589                 UnityPlayer.UnitySendMessage(tearObject, setEmotionFunction, "SURPRISED");
590                 break;
591             case ANGRY:
592                 UnityPlayer.UnitySendMessage(eyebrowsObject, setEmotionFunction, "ANGRY");
593                 UnityPlayer.UnitySendMessage(mouthObject, setEmotionFunction, "ANGRY");
594                 UnityPlayer.UnitySendMessage(tearObject, setEmotionFunction, "ANGRY");
595                 break;
596             case IDLE:
597                 UnityPlayer.UnitySendMessage(eyebrowsObject, setEmotionFunction, "IDLE");
598                 UnityPlayer.UnitySendMessage(mouthObject, setEmotionFunction, "IDLE");
599                 UnityPlayer.UnitySendMessage(tearObject, setEmotionFunction, "IDLE");
600                 break;
601             default:
602         }
```

```
603
604         // This part makes the mouth move if speaking is happening //
605         if (voiceYourEmotion != null) {
606             voiceYourEmotion.start();
607             UnityPlayer.UnitySendMessage(mouthObject, setSpeakingFunction, startSpeaking);
608             voiceYourEmotion.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
609                 @Override
610                 public void onCompletion(MediaPlayer mediaPlayer) {
611                     // This bit gets called back when audio is finished and mouth needs to stop moving
    //
612                     UnityPlayer.UnitySendMessage(mouthObject, setSpeakingFunction, stopSpeaking);
613                 }
614             });
615         }
616     }
```

References com.unity3d.player.UnityPlayerActivity.ANGRY, com.unity3d.player.UnityPlayerActivity.backgroundColor, com.unity3d.player.UnityPlayerActivity.changeBackgroundColorFunction, com.unity3d.player.UnityPlayerActivity.emotionBackgroundColor(), com.unity3d.player.UnityPlayerActivity.eyebrowsObject, com.unity3d.player.UnityPlayerActivity.HAPPY, com.unity3d.player.UnityPlayerActivity.IDLE, com.unity3d.player.UnityPlayerActivity.mainCameraObject, com.unity3d.player.UnityPlayerActivity.mouthObject, com.unity3d.player.UnityPlayerActivity.SAD, com.unity3d.player.UnityPlayerActivity.setEmotionFunction, com.unity3d.player.UnityPlayerActivity.setSpeakingFunction, com.unity3d.player.UnityPlayerActivity.startSpeaking, com.unity3d.player.UnityPlayerActivity.stopSpeaking, com.unity3d.player.UnityPlayerActivity.SURPRISED, com.unity3d.player.UnityPlayerActivity.tearObject, and com.unity3d.player.UnityPlayerActivity.voiceYourEmotion.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.2.14 settingsUpdate()

```
void com.unity3d.player.UnityPlayerActivity.settingsUpdate (
            String[] messagePieces )  [private]
```

Poke enable/disable and turn off tab.

- Turns on or off Poking
- Triggers sleepy animation and turns off face device

Parameters

messagePieces	String instruction received, broken into pieces

Definition at line 703 of file UnityPlayerActivity.java.

```
703                                                 {
704
705         // Will change these to different letters later //
706         switch (messagePieces[1].toUpperCase()) {
707             case "POKE":
708                 switch (messagePieces[2]) {
709                     case "DISABLE":
710                         UnityPlayer.UnitySendMessage(ouchZoneObject, setEyePokeEnabledStateFunction,
    falseString);
711                         break;
712                     case "ENABLE":
713                         UnityPlayer.UnitySendMessage(ouchZoneObject, setEyePokeEnabledStateFunction,
    trueString);
714                         break;
715                 }
716                 break;
717             case "SLEEP":
718                 // run the sleepy animation //
```

```
719                 UnityPlayer.UnitySendMessage(eyelidsObject, goToSleepFunction, "");
720
721                 // delay a little bit //
722                 handler.postDelayed(new Runnable() {
723                     @Override
724                     public void run() {
725                         // turn the device off //
726                         mDevicePolicyManager.lockNow();
727                     }
728                 }, 2500);
729                 break;
730             default:
731                 break;
732         }
733
734     }
```

References com.unity3d.player.UnityPlayerActivity.eyelidsObject, com.unity3d.player.UnityPlayerActivity.false↩String, com.unity3d.player.UnityPlayerActivity.goToSleepFunction, com.unity3d.player.UnityPlayerActivity.handler, com.unity3d.player.UnityPlayerActivity.mDevicePolicyManager, com.unity3d.player.UnityPlayerActivity.ouchZone↩Object, com.unity3d.player.UnityPlayerActivity.setEyePokeEnabledStateFunction, and com.unity3d.player.Unity↩PlayerActivity.trueString.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.2.15 showToastMethod()

```
void com.unity3d.player.UnityPlayerActivity.showToastMethod (
             final String message )
```

Toasts for threads other than the main.

Parameters

message	Message to be shown in Toast

Toasts can only be displayed on the main thread. To call Toasts from other threads, a public method in the Activity running on the main thread can be used.

Definition at line 687 of file UnityPlayerActivity.java.

```
687                                                 {
688         //No idea how this works but what it does is run the Toast message called from another thread on
      the main thread
689         runOnUiThread(new Runnable() {
690             public void run() {
691                 Toast.makeText(getBaseContext(), message, Toast.LENGTH_SHORT).show();
692             }
693         });
694     }
```

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.2.16 speakOut()

```
void com.unity3d.player.UnityPlayerActivity.speakOut (
             String[] messagePieces )  [private]
```

Text to speech code.

- Sets the pitch value
- Sets the speed of speech
- Executes Text To Speech

Parameters

messagePieces	String instruction received, broken into pieces

Definition at line 506 of file UnityPlayerActivity.java.

```
506                                                 {
507
508         float pitchValue = Float.parseFloat(messagePieces[2]);
509         textToSpeech.setPitch(pitchValue);
510         textToSpeech.setSpeechRate(0.7f);
511
512         Log.d(TAG + " pitchValue:", String.valueOf(pitchValue));
513
514         HashMap<String, String> map = new HashMap<String, String>();
515         map.put(TextToSpeech.Engine.KEY_PARAM_UTTERANCE_ID, "texToSpeech");
516         textToSpeech.speak(messagePieces[1], TextToSpeech.QUEUE_FLUSH, map);
517
518     }
```

References com.unity3d.player.UnityPlayerActivity.TAG, and com.unity3d.player.UnityPlayerActivity.textToSpeech.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.2.17 updateUnityCommandLineArguments()

```
String com.unity3d.player.UnityPlayerActivity.updateUnityCommandLineArguments (
            String cmdLine )  [protected]
```

Created when exporting from Unity.

Note

Created when exporting from Unity. As far as I can tell, it helps load the app back up quickly when returning from another app.

- Override this in your custom UnityPlayerActivity to tweak the command line arguments passed to the Unity Android Player
- The command line arguments are passed as a string, separated by spaces
- UnityPlayerActivity calls this from 'onCreate'
- Supported: -force-gles20, -force-gles30, -force-gles31, -force-gles31aep, -force-gles32, -force-gles, -force-vulkan
- See https://docs.unity3d.com/Manual/CommandLineArguments.html

Parameters

cmdLine	the current command line arguments, may be null

Returns

the modified command line string or null

Definition at line 763 of file UnityPlayerActivity.java.

```
763                                                                 {
764         return cmdLine;
765     }
```

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.3 Member Data Documentation

7.5.3.1 ADMIN_INTENT

```
final int com.unity3d.player.UnityPlayerActivity.ADMIN_INTENT = 15  [static], [private]
```

An integer passed to startActivityForResult() and received by onActivityResult(). If it is greater >= 0, this code will be returned in onActivityResult() when the activity exits. Does nothing of significance at present.

Definition at line 231 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.3.2 ANGRY

```
final int com.unity3d.player.UnityPlayerActivity.ANGRY = 3  [package]
```

Definition at line 73 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.emotionBackgroundColor(), and com.unity3d.player.Unity↩PlayerActivity.setEmotion().

7.5.3.3 audioManager

```
AudioManager com.unity3d.player.UnityPlayerActivity.audioManager  [private]
```

Manages the volume when the app is running.

An AudioManager object provides access to volume and ringer mode control

Definition at line 213 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), com.unity3d.player.UnityPlayerActivity.on↩Destroy(), com.unity3d.player.UnityPlayerActivity.onPause(), and com.unity3d.player.UnityPlayerActivity.on↩Resume().

7.5.3.4 backgroundColor

```
String com.unity3d.player.UnityPlayerActivity.backgroundColor = "000000"  [package]
```

Holds Background color hex

Definition at line 621 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.emotionBackgroundColor(), and com.unity3d.player.Unity↩ PlayerActivity.setEmotion().

7.5.3.5 bluetoothAdapter

```
BluetoothAdapter com.unity3d.player.UnityPlayerActivity.bluetoothAdapter  [private]
```

A BluetoothAdapter object.

A BluetoothAdapter lets you perform fundamental Bluetooth tasks, such as initiate device discovery, query a list of bonded (paired) devices, instantiate a BluetoothDevice using a known MAC address, and create a Bluetooth↩ ServerSocket to listen for connection requests from other devices, and start a scan for Bluetooth LE devices.

Definition at line 202 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.beginConnection(), and com.unity3d.player.UnityPlayer↩ Activity.onCreate().

7.5.3.6 changeBackgroundColorFunction

```
com.unity3d.player.UnityPlayerActivity.changeBackgroundColorFunction  [private]
```

Represents Unity object.

A String that Represents the Unity function "ChangeBackgroundColor"

Definition at line 186 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.set↩ Emotion().

7.5.3.7 eyebrowsObject

```
com.unity3d.player.UnityPlayerActivity.eyebrowsObject  [package]
```

References Unity object.

A String that represents the Unity object "eyebrows"

Definition at line 185 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.set↩ Emotion().

7.5.3.8 eyelidsObject

`com.unity3d.player.UnityPlayerActivity.eyelidsObject [package]`

References Unity object.

A String that represents the Unity object "eyelids"

Definition at line 185 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.↩ settingsUpdate().

7.5.3.9 falseString

`com.unity3d.player.UnityPlayerActivity.falseString [package]`

A String that holds the parameter "FALSE". This is passed to a Unity function to enable the animation.

Definition at line 188 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.↩ settingsUpdate().

7.5.3.10 goToSleepFunction

`com.unity3d.player.UnityPlayerActivity.goToSleepFunction [package]`

Represents Unity object.

A String that represents the Unity function "GoToSleep"

Definition at line 186 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.↩ settingsUpdate().

7.5.3.11 handler

`final Handler com.unity3d.player.UnityPlayerActivity.handler = new Handler() [package]`

Used to schedule messages and runnables to be executed at some point in the future

Definition at line 241 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.settingsUpdate().

7.5.3.12 HAPPY

```
final int com.unity3d.player.UnityPlayerActivity.HAPPY = 0  [private]
```

Definition at line 73 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.emotionBackgroundColor(), and com.unity3d.player.Unity↩PlayerActivity.setEmotion().

7.5.3.13 IDLE

```
final int com.unity3d.player.UnityPlayerActivity.IDLE = 4  [package]
```

Definition at line 73 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.emotionBackgroundColor(), and com.unity3d.player.Unity↩PlayerActivity.setEmotion().

7.5.3.14 MAC_ADDRESS

```
String com.unity3d.player.UnityPlayerActivity.MAC_ADDRESS  [private]
```

MAC ADDRESS of the Bluetooth device to establish communication with

Definition at line 122 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.beginConnection(), and com.unity3d.player.UnityPlayer↩Activity.onCreate().

7.5.3.15 mainCameraObject

```
com.unity3d.player.UnityPlayerActivity.mainCameraObject  [private]
```

References Unity object.

A String that represents the Unity object "Main Camera"

Definition at line 185 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.set↩Emotion().

7.5.3.16 mBTStateBroadcastReceiver

```
final BroadcastReceiver com.unity3d.player.UnityPlayerActivity.mBTStateBroadcastReceiver  [private]
```

Initial value:

```
= new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {

            final String action = intent.getAction();
            if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED)) {
                final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, BluetoothAdapter.ERROR);
                switch (state) {
                    case BluetoothAdapter.STATE_OFF:
                        Toast.makeText(context, R.string.BT_STATE_OFF_TEXT, Toast.LENGTH_SHORT).show();
                        break;
                    case BluetoothAdapter.STATE_TURNING_OFF:
                        Toast.makeText(context, R.string.BT_STATE_TURNING_OFF_TEXT,
       Toast.LENGTH_SHORT).show();
                        break;
                    case BluetoothAdapter.STATE_ON:
                        Toast.makeText(context, R.string.BT_STATE_ON_TEXT, Toast.LENGTH_SHORT).show();
                        break;
                    case BluetoothAdapter.STATE_TURNING_ON:
                        Toast.makeText(context, R.string.BT_STATE_TURNING_ON_TEXT,
       Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        }
    }
```

a BroadcastReceiver type object

- Receives and handles broadcast intents sent by `Context.sendBroadcast(Intent)`.
- This is an implementation of BroadcastReceiver registered to be run in the main activity thread. Its receiver is called with any broadcast Intent that matches filter, in this case is **BluetoothAdapter.ACTION_STATE_↩ CHANGED** (in other words, when the state of the local Bluetooth adapter has been changed).

Definition at line 81 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.on↩ Destroy().

7.5.3.17 mComponentName

```
ComponentName com.unity3d.player.UnityPlayerActivity.mComponentName  [private]
```

Identifier for a specific application component (Activity, Service, BroadcastReceiver, or ContentProvider) that is available.

Definition at line 236 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.3.18 mDevicePolicyManager

```
DevicePolicyManager com.unity3d.player.UnityPlayerActivity.mDevicePolicyManager  [private]
```

Public interface for managing policies enforced on a device.

Most clients of this class must be registered with the system as a device administrator

Note

I'm not sure how this works

Definition at line 226 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.↩ settingsUpdate().

7.5.3.19 mouthObject

```
com.unity3d.player.UnityPlayerActivity.mouthObject  [package]
```

References Unity object.

A String that represents the Unity object "mouth"

Definition at line 185 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), com.unity3d.player.UnityPlayerActivity.on↩ Pause(), and com.unity3d.player.UnityPlayerActivity.setEmotion().

7.5.3.20 mUnityPlayer

```
UnityPlayer com.unity3d.player.UnityPlayerActivity.mUnityPlayer  [protected]
```

Referenced from native coded Don't change the name of this variable

Definition at line 111 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onConfigurationChanged(), com.unity3d.player.UnityPlayer↩ Activity.onCreate(), com.unity3d.player.UnityPlayerActivity.onDestroy(), com.unity3d.player.UnityPlayerActivity.↩ onLowMemory(), com.unity3d.player.UnityPlayerActivity.onNewIntent(), com.unity3d.player.UnityPlayerActivity.↩ onPause(), com.unity3d.player.UnityPlayerActivity.onResume(), com.unity3d.player.UnityPlayerActivity.onTrim↩ Memory(), and com.unity3d.player.UnityPlayerActivity.onWindowFocusChanged().

7.5.3.21 MY_UUID

```
final UUID com.unity3d.player.UnityPlayerActivity.MY_UUID = UUID.fromString("00001101-0000-1000-8000-00805F9↩
B34FB")  [static], [package]
```

Universally Unique Identifier (UUID)

- Creating a UUID which represents a 128-bit value.
- More information on UUIDs by the Internet Engineering Task Force can be found here.
- UUIDs are not tied to particular devices. They identify software services. You just need both sides to use the same one.
- "00001101-0000-1000-8000-00805F9B34FB" is the one and only UUID for SPP (serial port profile). Check out the Android Developer's page.

Definition at line 60 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.beginConnection().

7.5.3.22 myBluetoothService

```
MyBluetoothService com.unity3d.player.UnityPlayerActivity.myBluetoothService  [static]
```

Handles Bluetooth stuff.

Custom class. Handles Bluetooth stuff.

Definition at line 66 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.beginConnection(), and com.unity3d.player.UnityPlayer↩Activity.onCreate().

7.5.3.23 ouchZoneObject

```
com.unity3d.player.UnityPlayerActivity.ouchZoneObject  [package]
```

References Unity object.

A String that represents the Unity object "ouch_zone"

Definition at line 185 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.↩settingsUpdate().

7.5.3.24 REQUEST_ENABLE_BT

```
int com.unity3d.player.UnityPlayerActivity.REQUEST_ENABLE_BT = 1  [private]
```

An integer passed to startActivityForResult() and received by onActivityResult(). If it is greater >= 0, this code will be returned in onActivityResult() when the activity exits. Does nothing of significance at present.

Definition at line 207 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onActivityResult(), and com.unity3d.player.UnityPlayer↩ Activity.onCreate().

7.5.3.25 SAD

```
final int com.unity3d.player.UnityPlayerActivity.SAD = 1  [package]
```

Definition at line 73 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.emotionBackgroundColor(), and com.unity3d.player.Unity↩ PlayerActivity.setEmotion().

7.5.3.26 savedVolume

```
int com.unity3d.player.UnityPlayerActivity.savedVolume  [private]
```

Saves the current volume setting of the device.

After the app is closed,this allows for the volume to go back to what it was before the app started

Definition at line 219 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), com.unity3d.player.UnityPlayerActivity.on↩ Destroy(), and com.unity3d.player.UnityPlayerActivity.onPause().

7.5.3.27 setEmotionFunction

```
com.unity3d.player.UnityPlayerActivity.setEmotionFunction  [package]
```

Represents Unity object.

A String that represents the Unity function "SetEmotion"

Definition at line 186 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.set↩ Emotion().

7.5.3.28 setEyePokeEnabledStateFunction

`com.unity3d.player.UnityPlayerActivity.setEyePokeEnabledStateFunction [package]`

Represents Unity object.

A String that represents the Unity function "SetEyePokeEnabledState"

Definition at line 186 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.↩ settingsUpdate().

7.5.3.29 setSpeakingFunction

`com.unity3d.player.UnityPlayerActivity.setSpeakingFunction [package]`

Represents Unity object.

A String that represents the Unity function "SetSpeaking"

Definition at line 186 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), com.unity3d.player.UnityPlayerActivity.on↩ Pause(), and com.unity3d.player.UnityPlayerActivity.setEmotion().

7.5.3.30 startSpeaking

`com.unity3d.player.UnityPlayerActivity.startSpeaking [private]`

Unity parameter.

A String that holds the parameter "START". This is passed to a Unity function to start speaking animation.

Definition at line 187 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.set↩ Emotion().

7.5.3.31 stopSpeaking

`com.unity3d.player.UnityPlayerActivity.stopSpeaking [package]`

Unity parameter.

A String that holds the parameter "STOP". This is passed to a Unity function to stop the speaking animation.

Definition at line 187 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), com.unity3d.player.UnityPlayerActivity.on↩ Pause(), and com.unity3d.player.UnityPlayerActivity.setEmotion().

7.5.3.32 SURPRISED

```
final int com.unity3d.player.UnityPlayerActivity.SURPRISED = 2  [package]
```

Definition at line 73 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.emotionBackgroundColor(), and com.unity3d.player.Unity↩ PlayerActivity.setEmotion().

7.5.3.33 TAG

```
String com.unity3d.player.UnityPlayerActivity.TAG = "DEBUG_UNITY_PLAYER"  [static], [private]
```

Debugging tool

Definition at line 71 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.↩ speakOut().

7.5.3.34 tearObject

```
com.unity3d.player.UnityPlayerActivity.tearObject  [package]
```

References Unity object.

A String that represents the Unity object "tear"

Definition at line 185 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.set↩ Emotion().

7.5.3.35 TESTING

```
final boolean com.unity3d.player.UnityPlayerActivity.TESTING = false  [private]
```

Debugging tool.

- Toggle to FALSE when using it with the two HC-05's on a breadboard. Check their MAC Addresses before using.
- FALSE implies controlling the robot body

Definition at line 50 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate().

7.5.3.36 textToSpeech

```
TextToSpeech com.unity3d.player.UnityPlayerActivity.textToSpeech  [private]
```

For text to speech functionality.

- A TextToSpeech object synthesizes speech from text for immediate playback or to create a sound file.
- This allows the robot to speak

Definition at line 196 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), com.unity3d.player.UnityPlayerActivity.on↩Destroy(), com.unity3d.player.UnityPlayerActivity.onPause(), and com.unity3d.player.UnityPlayerActivity.speak↩Out().

7.5.3.37 trueString

```
com.unity3d.player.UnityPlayerActivity.trueString  [private]
```

Unity parameter.

A String that holds the parameter "TRUE". This is passed to a Unity function to disable the poking animation.

Definition at line 188 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onCreate(), and com.unity3d.player.UnityPlayerActivity.↩settingsUpdate().

7.5.3.38 voiceYourEmotion

```
MediaPlayer com.unity3d.player.UnityPlayerActivity.voiceYourEmotion  [package]
```

Media Player for running audio files that "voice emotion".

A MediaPlayer class object can be used to control playback of audio/video files and streams

Definition at line 117 of file UnityPlayerActivity.java.

Referenced by com.unity3d.player.UnityPlayerActivity.onDestroy(), com.unity3d.player.UnityPlayerActivity.on↩Pause(), and com.unity3d.player.UnityPlayerActivity.setEmotion().

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

- UnityPlayerActivity.java

Chapter 8

File Documentation

8.1 MyAdminReceiver.java File Reference

This allows the app to turn off the device.

Classes

- class com.unity3d.player.MyAdminReceiver

  This allows the app to turn off the device.

Packages

- package com.unity3d.player

8.1.1 Detailed Description

This allows the app to turn off the device.

- DeviceAdminReceiver is the base class for implementing a device administration component
- This class provides a convenience for interpreting the raw intent actions that are sent by the system.
- Definitely read the `Android Developer's page` on this
- The manifest file was also updated when adding this class

8.2 MyBluetoothService.java File Reference

Handles everything Bluetooth.

Classes

- class com.unity3d.player.MyBluetoothService

  Handles everything Bluetooth.
- class com.unity3d.player.MyBluetoothService.ConnectThread

  Client thread that initiates a Bluetooth connection.
- class com.unity3d.player.MyBluetoothService.ConnectedThread

  Thread that manages a Bluetooth connection.

Packages

- package com.unity3d.player

8.2.1 Detailed Description

Handles everything Bluetooth.

- Starts up and maintains Bluetooth connections between devices
- Sends and handles reception of messages from connected devices

8.3 README.md File Reference

8.4 UnityPlayerActivity.java File Reference

The main activity.

Classes

- class com.unity3d.player.UnityPlayerActivity

  This is where everything important happens.

Packages

- package com.unity3d.player

8.4.1 Detailed Description

The main activity.

The first screen to appear when the user launches the app

Index