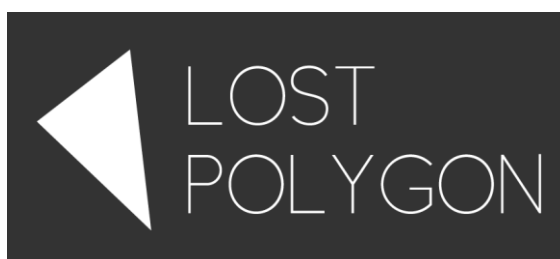


Bluetooth Multiplayer for Android v1.21



by



Contents

1. General information
2. BluetoothMultiplayerAndroid.cs methods overview
3. BluetoothMultiplayerAndroid.cs methods
4. BluetoothMultiplayerAndroidManager.cs events
5. Configuring AndroidManifest.xml and extending Activities
6. Contact
7. Changelog

General information

This plugin gives you an ability to add Bluetooth multiplayer on Android using a simple interface, similar to that of Unity Network component. It also fully compatible with Unity built-in networking. This means you can easily reuse your networking code for Internet and local gaming with minimal changes, or use any of existing tutorials about Unity built-in networking.

Plugin overrides built-in Unity activity and adds new permissions to AndroidManifest.xml in order to function. AndroidManifest.xml is generated automatically in case it is not present, however, you can do that manually. To do that, in case your project doesn't uses any plugins that modify AndroidManifest.xml, use

Component → Lost Polygon → Bluetooth Mutiplayer Android → Generate AndroidManifest.xml

menu item, otherwise try

Component → Lost Polygon → Bluetooth Mutiplayer Android → Patch existing AndroidManifest.xml

That will work for most cases. In case of any problems, refer to “Configuring AndroidManifest.xml and extending Activities” section of this file.

BluetoothMultiplayerAndroid.cs file wraps all interactions between Unity and Java. All methods are declared static, so no object instantiation is required.

BluetoothMultiplayerAndroidManager.cs script, attached to a GameObject, is required to successfully receive callbacks from Java. It will be created automatically when you use any of its events. Do not forget to unregister the delegates upon destruction of objects that used the events (on OnDestroy(), for example), as that may lead to undefined behavior.

Do not rename the instantiated object or *BluetoothMultiplayerAndroidManager.cs* class, as that will break the Java callback reception.

It is important to call `Init(string uuid)` before any other interaction with the plugin. UUID is an identifier that must be unique for every Bluetooth application.

Bluetooth connection can only be established if the connecting and host device have the same UUID. You can generate random unique UUID for your game using

Component → Lost Polygon → Bluetooth Mutiplayer Android → UUID generator

The package has two demo scenes included (see “Plugins/BluetoothMultiplayerAndroid/Demo/”), one is for a very simple Bluetooth multiplayer interaction, and second is for Bluetooth device discovery. It is highly recommended that you run the demos on your device to see if plugin is working okay. The demo directory and its contents can be safely deleted for production, or just if you don’t need it anymore.

Note: as Android Bluetooth implementation guarantees data delivery, it is highly discouraged to use reliable state synchronization for NetworkView’s, as that may lead to unnecessary delays and stuttering.

Note: Device discovery is a heavyweight procedure. New connections to remote Bluetooth devices should not be attempted while discovery is in progress, and existing connections will experience limited bandwidth and high latency. Because of that, `StopDiscovery()` will always be called automatically when connecting to server.

Use the logcat while testing the plugin and debugging your game — some debug information is available only in the log, and most of it requires calling `setVerboseLog(true)`.

Plugin is tested in Unity 3.5.2 - 4.2.0. Both Free and Pro versions are supported.

BluetoothMultiplayerAndroid.cs methods overview

<code>bool</code>	<code>Init(string uuid)</code> Initialize the plugin and set the Bluetooth service UUID.
<code>bool</code>	<code>InitializeServer(ushort port)</code> Start server, listening for incoming Bluetooth connections.
<code>bool</code>	<code>Connect(string address, ushort port)</code> Connect to a Bluetooth device.
<code>bool</code>	<code>Disconnect()</code> Stop all Bluetooth connectivity.
<code>bool</code>	<code>StopListen()</code> Stop listening for new incoming connections.
<code>bool</code>	<code>ShowDeviceList()</code> Show the Bluetooth device picker dialog.
<code>bool</code>	<code>RequestBluetoothEnable()</code> Open a dialog asking user to enable Bluetooth.
<code>bool</code>	<code>BluetoothEnable()</code> Enable the Bluetooth adapter, if possible.
<code>bool</code>	<code>BluetoothDisable()</code> Disable the Bluetooth adapter, if possible.
<code>bool</code>	<code>RequestDiscoverable(int discoverableDuration)</code> Open a dialog asking user to make device discoverable on Bluetooth.
<code>BluetoothMultiplayerMode</code>	<code>CurrentMode()</code> Returns the current plugin mode.
<code>bool</code>	<code>IsBluetoothEnabled()</code> Return true if Bluetooth is currently enabled and ready for use.
<code>bool</code>	<code>IsBluetoothAvailable()</code> Return true if Bluetooth is available on the device.

BluetoothDevice	GetCurrentDevice() Return current Bluetooth device.
bool	StartDiscovery() Start discovery of nearby discoverable Bluetooth devices.
bool	StopDiscovery() Stop discovery of nearby discoverable Bluetooth devices.
bool	IsDiscovering() Return true if Bluetooth device discovery is going on.
BluetoothDevice[]	GetBondedDevices() Return array of bonded (paired) Bluetooth devices.
BluetoothDevice[]	GetNewDiscoveredDevices() Return array of Bluetooth devices discovered during current discovery session.
BluetoothDevice[]	GetDiscoveredDevices() Return array of bonded (paired) Bluetooth devices and Bluetooth devices discovered during current discovery session.
bool	SetRawPackets(bool doEnable) Enable or disable raw packets. Use only if you know what you are doing.
void	SetVerboseLog(bool doEnable) Enables or disables verbose logging.

BluetoothMultiplayerAndroid.cs methods

```
public static bool Init(string uuid)
```

Initializes the plugin and sets the Bluetooth service UUID.

Parameters

uuid Bluetooth service UUID. Must be different for each game.

Returns

true on success, false if UUID format is incorrect.

```
public static bool InitializeServer(ushort port)
```

Start server, listening for incoming Bluetooth connections. Must be called before [Network.InitializeServer](#).

Throws [BluetoothNotEnabledException](#) if called when Bluetooth was not enabled.

Parameters

port Server port number. Must be the same as passed to [Network.InitializeServer](#).

Returns

true on success, false on error.

```
public static bool Connect(string address, ushort port)
```

Connect to a Bluetooth device. Must be called before [Network.Connect](#).

Throws [BluetoothNotEnabledException](#) if called when Bluetooth was not enabled.

Parameters

address Address of host Bluetooth device to connect to.

port Server port number. Must be the same as passed to [Network.Connect](#).

Returns

true on success, false on error.

```
public static bool Disconnect()
```

Stop all Bluetooth connectivity.

Returns

true on success, false on error.

```
public static bool StopListen()
```

Stop listening for new incoming connections. For example, you should listen for connections while in game lobby, and stop listening when the actual game has started to make sure no new device could connect.

Returns

true on success, false on error.

```
public static bool ShowDeviceList()
```

Show the Bluetooth device picker dialog.

Throws `BluetoothNotEnabledException` if called when Bluetooth was not enabled.

Returns

true on success, false on error.

```
public static bool RequestDiscoverable(int discoverableDuration = 120)
```

Open a dialog asking user to make device discoverable on Bluetooth for `discoverableDuration` seconds. This will also request the user to turn on Bluetooth if it is not currently enabled.

On Android 4.0 and higher, setting the parameter to 0 allows making device discoverable “forever” (until discoverability is disabled manually or Bluetooth is disabled).

Parameters

<i>discoverableDuration</i>	The desired duration of discoverability (in seconds). Default value 120 seconds.
-----------------------------	---

Returns

true on success, false on error.

```
public static BluetoothMultiplayerMode CurrentMode()
```

Returns the current plugin mode (None, Server, or Client).

Returns

Current plugin mode on success, `BluetoothMultiplayerMode.None` on error.

```
public static bool RequestBluetoothEnable()
```

Open a dialog asking user to enable Bluetooth. It is recommended to use this method instead of `BluetoothEnable()` for more native experience.

Returns

true on success, false on error.

```
public static bool BluetoothEnable()
```

Enable the Bluetooth adapter, if possible.

Do not use this method unless you have implemented a custom GUI, allowing user to confirm the action. Otherwise use `RequestBluetoothEnable()`.

Returns

true on success, false on error.

```
public static bool BluetoothDisable()
```

Disable the Bluetooth adapter, if possible.

Returns

true on success, false on error.

```
public static bool IsBluetoothEnabled()
```

Return true if Bluetooth is currently enabled and ready for use.

Returns

true if Bluetooth connectivity is available and enabled, false otherwise.


```
public static bool IsBluetoothAvailable()
```

Return the Bluetooth availability. Bluetooth can be unavailable if no Bluetooth adapter is present, or if some error occurred.

Returns

true if Bluetooth connectivity is available, false otherwise.

```
public static BluetoothDevice GetCurrentDevice()
```

Return BluetoothDevice of current device the application runs on.

Returns

BluetoothDevice if Bluetooth connectivity is available and enabled, null otherwise or on error.

```
public static bool StartDiscovery()
```

Start the process of discovering nearby discoverable Bluetooth devices.

The process is asynchronous and is usually held for 10-30 seconds in time. Note that performing device discovery is a heavy procedure for the Bluetooth adapter and will consume a lot of its resources and drain battery power.

Returns

true if Bluetooth connectivity is available and enabled , false otherwise.

```
public static bool StopDiscovery()
```

Stop the process of discovering nearby discoverable Bluetooth devices.

Because discovery is a heavyweight procedure for the Bluetooth adapter, this method is called automatically when connecting to the server.

Returns

true if Bluetooth connectivity is available and enabled , false otherwise.

```
public static bool IsDiscovering()
```

Return true if the local Bluetooth adapter is currently in process of device discovery.

Returns

true if Bluetooth connectivity is available and enabled and device discovery is currently going on, false otherwise.

```
public static BluetoothDevice[] GetBondedDevices()
```

Return `BluetoothDevice[]` array of bonded (paired) devices. This method is available even without starting the discovery process.

Returns

`BluetoothDevice[]` if Bluetooth connectivity is available and enabled, `null` otherwise or on error.

```
public static BluetoothDevice[] GetNewDiscoveredDevices()
```

Return `BluetoothDevice[]` array of devices discovered during the ongoing discovery session. This list is not cleared after the discovery ends.

Returns

`BluetoothDevice[]` if Bluetooth connectivity is available and enabled and device discovery is currently going on, `null` otherwise or on error.

```
public static BluetoothDevice[] GetDiscoveredDevices()
```

Return `BluetoothDevice[]` array of bonded (paired) devices *and* devices discovered during the ongoing discovery session. This list is not cleared after the discovery ends.

Returns

`BluetoothDevice[]` if Bluetooth connectivity is available and enabled and device discovery is currently going on, `null` otherwise or on error.

```
public static bool SetRawPackets(bool doEnable)
```

Enable or disable raw packets. Can only be called when no Bluetooth networking is going on. This option can be used if you want to exchange raw data with a generic Bluetooth device (like Arduino). Use this only if you know what you are doing.

Parameters

doEnable The new state of raw packets mode.

Returns

true if no Bluetooth networking is going on, false otherwise.

```
public static void SetVerboseLog(bool doEnable)
```

Enables or disables verbose logging. Useful for testing and debugging.

Parameters

doEnable The new state of verbose logging.

BluetoothMultiplayerAndroidManager.cs events

```
// Fired when server is started and waiting for incoming connections
public static event Action onBluetoothListeningStartedEvent;

// Fired when listening for incoming connections
// was stopped by BluetoothMultiplayerAndroid.StopListening()
public static event Action onBluetoothListeningCanceledEvent;

// Fired when Bluetooth was enabled
public static event Action onBluetoothAdapterEnabledEvent;

// Fired when request to enabled Bluetooth failed for some reason
// (user did not authorized to enable Bluetooth or an error occurred)
public static event Action onBluetoothAdapterEnableFailedEvent;

// Fired when Bluetooth was disabled
public static event Action onBluetoothAdapterDisabledEvent;

// Fired when Bluetooth client successfully connected to the Bluetooth server
// Provides BluetoothDevice of server device
public static event Action<BluetoothDevice> onBluetoothConnectedToServerEvent;

// Fired when Bluetooth client failed to connect to the Bluetooth server
// Provides BluetoothDevice of server device
public static event Action<BluetoothDevice> onBluetoothConnectToServerFailedEvent;

// Fired when Bluetooth client disconnected from the Bluetooth server
// Provides BluetoothDevice of server device disconnected from
public static event Action<BluetoothDevice> onBluetoothDisconnectedFromServerEvent;

// Fired on Bluetooth server when an incoming Bluetooth
// client connection was accepted
// Provides BluetoothDevice of connected client device
public static event Action<BluetoothDevice> onBluetoothClientConnectedEvent;

// Fired on Bluetooth server when a Bluetooth client had disconnected
// Provides BluetoothDevice of disconnected client device
public static event Action<BluetoothDevice> onBluetoothClientDisconnectedEvent;

// Fired when user selects a device in the device picker dialog
// Provides BluetoothDevice of picked device
public static event Action<BluetoothDevice> onBluetoothDevicePickedEvent;

// Fired when Bluetooth discovery is started
public static event Action onBluetoothDiscoveryStartedEvent;

// Fired when Bluetooth discovery is finished
public static event Action onBluetoothDiscoveryFinishedEvent;

// Fired when a new device was found during Bluetooth discovery procedure
// Provides BluetoothDevice of found device
public static event Action<BluetoothDevice> onBluetoothDiscoveryDeviceFoundEvent;
```

Configuring AndroidManifest.xml and extending Activities

Plugin overrides built-in Unity activity and adds new permissions to AndroidManifest.xml in order to function.

Added permissions are:

```
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>  
<uses-permission android:name="android.permission.BLUETOOTH"/>
```

Activity classes for this plugin, overriding the built-in Unity activities, are:

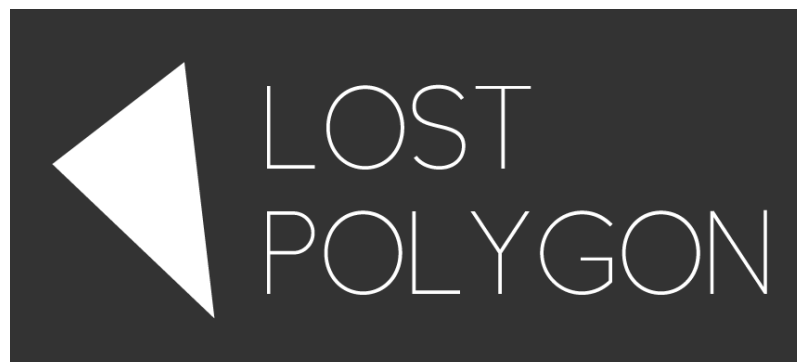
```
biz.zimm.unity.bluetoothmediator.player.BluetoothUnityPlayerProxyActivity  
biz.zimm.unity.bluetoothmediator.player.BluetoothUnityPlayerActivity  
biz.zimm.unity.bluetoothmediator.player.BluetoothUnityPlayerNativeActivity
```

You must override BluetoothUnityPlayerNativeActivity and BluetoothUnityPlayerActivity to implement your custom functionality (for example, adding some other plugin). Refer to

<http://docs.unity3d.com/Documentation/Manual/PluginsForAndroid.html>

for more details.

Contact



e-mail: contact@zimm.biz

Skype: serhij.yolkin

Changelog

1.0:

- initial release

1.03:

- minor fixes

1.1:

- add device discovery API (with demo usage example)
- fix manifest generation on Mac OS X
- minor fixes

1.2:

- new `GetCurrentDevice()` method to get current Bluetooth device information
- `BluetoothMultiplayerAndroidManager` is now instantiated automatically, no need to add prefab
- new `SetRawPackets()` method to transmit data as raw (for advanced usage)

1.21:

- minor fixes