iBeacon Unity plugin Documentation

iBeacon

Functionality

The iBeacon acts as the server component for an iBeacon network. An app running the server will advertise an iBeacon UUID, region, major, minor and range to iBeacon receivers in an area.

How to use

Attach the IBeacon script to a gameobject in your scene or drag the “IBeaconServer” Prefab into the scene. You will find it under “Resources/Prefabs/”. The IBeacon script exposes 5 variables which you can configure according to your needs.

- Generate: clicking this will generate a valid UUID in the UUID field
- UUID: you can enter any valid UUID here or generate a random one by clicking on Generate
- Identifier: this is the region you will broadcast, e.g. com.example.myregion
- Major: the major number could acts as a hint for the location of the beacon. A company with more than one structure may for example identify each building by a different Major
- Minor: the minor number will act as a further hint where exactly you are. Said company may for example identify each room inside said buildings by a different Minor

An example would be a company that has 2 branches, one in New York and one in Chicago. The storage room in the second building in Chicago may look like this:

UUID: 9346e6ca-93c4-495d-81d5-efa64a005a8d
Identifier: com.company.chicago
Major: 2
Minor: 3

While the storage room in the first building in New York could be:

UUID: 9346e6ca-93c4-495d-81d5-efa64a005a8d
Identifier: com.company.newyork
Major: 1
Minor: 3

I hope you get the idea, you can edit these values to your needs, just make sure the receiver app you build recognizes them.

IBeacon exposes the following methods:

public static void Init() → Inits the plugin, call this before anything else
public static void Transmit() → Starts advertising
public static void StopTransmit() → Stops advertising

IBeaconReceiver

Functionality

The IBeaconReceiver acts as a receiver for the IBeacon you lernt to set up above.

How to use

Create a gameobject in your scene, call it “IBeaconReceiver” and attach the IBeaconReceiver script. You can also drag the IBeaconReceiver Prefab into your scene that you will find under “Resources/Prefabs/”. The IBeaconReceiver exposes 2 variables and a switch:

The regions variable will hold the ibeaconregion objects that you want to scan for, every region has a region name and a UUID that you have to adjust according to your ibeacons uuid. You can choose the name yourself and it doesn't really do anything.

The NSLocationUsageDescription variable will hold a string that iOS will use to display a dialog to your user that will tell him why you want to monitor his location. Every time you change this string, please press the checkbox below at Update Description so it get's properly saved to your info.plist file when building.

Initialisation:

Although you can just call IBeaconReceiver.Init() to initialise the plugin, it is highly advised that you subscribe to the BluetoothStateChangedEvent for iBeaconReceiver and call iBeaconReceiver.CheckBluetoothLEStatus() first. iBeaconReceiver will now notify you what the current Bluetooth State is and will notify you in the future when it changes. iBeaconReceiver.Init() should only be called when the Event comes back with BluetoothLowEnergyState.POWERED_ON.

If Bluetooth is powered off you can call iBeaconReceiver.EnableBluetooth() to display a dialog to the user asking for permission to turn it on.

iBeaconReceiver fires an event with a List of Beacons that you can attach to. For an example have a look into the IBeaconExampleScene.

iBeaconReceiver exposes the following methods:

public static void Init → Inits the plugin, call this before everything else
public static void Stop → Stops scanning for beacons
public static void Scan → Starts Scanning for beacons, you don’t have to call this if you didn’t stop the scan before, init will call this for you

public static void CheckBluetoothLEStatus → this will check the current Bluetooth status for you, results will come back to every subscriber to BluetoothStateChangedEvent.
Public static void EnableBluetooth() → Will display a dialog to the user asking for permission to turn on Bluetooth (Note: on Ios this will actually display a dialog with a button to go to the Bluetooth settings where the user can switch it on)

IBeaconReceiver fires the following events:

BeaconRangeChangedEvent → delivers a list of all beacons in range

BluetoothStateChangedEvent → fires when the state of the BluetoothAdapter changes.

Note: The old version featured 2 events (BeaconArrivedEvent, BeaconOutOfRangeEvent) that are now deprecated due to the new design which features multiple regions. You can however get the same behaviour by looking into the iBeaconReceiverExample script where it is described how to possibly do this in line 51 ff.

Android Usage

The android usage is essentially the same as the IOs one except that you will need to adjust your AndroidManifest.xml to have access to Bluetooth. An ExampleManifest.xml comes with this plugin and is located under Assets > Plugins > Android. If you don't have any other Android plugins that you use you can safely rename it to AndroidManifest.xml, if you do just copy these 2 lines to your own AndroidManifest.xml

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

You can also add the following Line to the Manifest to declare that your app is using Bluetooth Low Energy and even make it available only to devices that have BLE enabled Hardware by setting the android:required value to true;

<uses-feature android:name="android.hardware.bluetooth_le" android:required="false"/>

Beacon class

The beacon class exposes the following variables:

string UUID = the uuid of the beacon
int major = the major of the beacon
int minor = the minor of the beacon
BeaconRange range = the range of the beacon, this can be IMMEDIATE for 0-50cm, NEAR for 50-200cm and FAR for 200-2000cm. UNKOWN should never happen and is a hint for an error with the beacon
int strength = the strength which with the beacon is sending
double accuracy = the estimated range in meters
DateTime lastSeen = the last time this beacon has been seen

The beacon class also features different interfaces for beacon usages in lists and comparisons, featured methods are:

Equals, ==, != to check if two beacons have the same UUID, major and minor and for IndexOf and Contains methods of lists

CompareTo to check if a beacon is more far aways than another beacon, this is also used by the typed list Sort method to automatically sort beacons by range.