User Activity Recognition

by Piotr Zmudzinski

ptr.zmudzinski@gmail.com

http://www.kokosoft.eu

About

This plugin uses device sensors in order to classify user movement into one of categories:
- walking,
- cycling,
- running,
- stationary,
- automative.

It is based on engines implemented by Apple and Google and exposed as a part of their SDK. Plugin does not modify these results in any way.
Keep in mind that results you are getting depend on device you are using (quality of sensors to be more specific).

Supported versions:
- **Android >= 4.0,**
- **iOS >= 11.0.**

Plugin doesn't work in Editor!
You have to run your app on real iOS or Android device.

Quick Start

Open example scene

Go to *UserActivityRecognition/Example* folder inside Unity and open *ExampleScene.*

It shows basic usage of a plugin, which is:

1. Detecting if activity recognition engine exists on user's device,
2. If it exists, and user clicks on "Start Recognition" it displays recognized activities together with confidence level and recognition time.

Before running it on Android or iOS device you have to…

Setup project

All plugin files are stored under *UserActivityRecognition* subdirectory which should be moved to your root directory:

1. Move *UserActivityRecognition/Editor* to your root directory,
2. Move *UserActivityRecognition/Plugins* to your root directory,
3. Plugin needs some external dependencies on Android platform. In order to avoid conflicts with other plugins it uses unity-jar-resolver which will download all of those and package those into result bundle.
   3.1. If you are already using it in your project it should detect new dependencies declared in *UserActivityRecognitionDependencies.xml,*
   3.2. If you are not using it, move *UserActivityRecognition/PlayServicesResolver* into your root directory. It should start resolving dependencies afterwards.
4. Profit.

Permissions setup

iOS

After generating Xcode project (keep in mind that you have to use Xcode 8 or higher) you have to add one permission description:

NSMotionUsageDescription
explanation from Apple docs:

> *A message that tells the user why the app is requesting access to the device's accelerometer.*

- Automatic (default and recommended)
Open *UserActivityRecognitionPermissionSetter* script from *Editor/ UserActivityRecognition* folder inside your project. Change values of those texts:

```
public static string motionUsageDescription = "Recognizing user activity.";
```

Generate your Xcode project. You are ready.

- Manually
Disable *UserActivityRecognitionPermissionSetter* script by either removing it or setting *shouldRun* variable to false.
After generating your Xcode project open *Info.plist* file and right-click on it and click *"Add row"*. Then enter *NSMotionUsageDescription* and set description manually.

Android

There must be this permission in your AndroidManifest.xml:

```
<uses-permission
android:name="com.google.android.gms.permission.ACTIVITY_RECOGNITION" />
```

It should be automatically added by including AAR library coming from a plugin but if you see permission issues make sure this line is part of your final .apk.

If you don't know how to add permission to your AndroidManifest check that link: http://answers.unity3d.com/questions/525838/help-about-adding-permissions-on-android.html

Now you are ready to play with the app. Press Build&Run and…

Play with activity recognition app

Click on *"Start Recognition"* button and start moving around. You should see live results in text above - e.g. information you are walking or staying in place.

You can start looking at *RecognitionCanvas* script which covers all functionality, but if you need more details take a look at **API Overview** chapter below.

API Overview

Plugin consists of two main classes:

- *UserActivityRecognition* which lets you start/stop activity tracking,
- *UserActivityListener* which sends you callback about new recognition using *UserActivityInfo.*

Both of them are under *KKUserActivityRecognition* namespace so remember to put

```
using KKUserActivityRecognition;
```

in your scripts.

Check *RecognitionCanvas* script for example usage.

UserActivityRecognition

Basic usage:

```
UserActivityRecognition.StartUpdates();

UserActivityRecognition.StopUpdates();
```

static bool IsAvailable()

Returns *true* if activity recognition service exists on a device.

!!! **You should call it before using other *UserActivityRecognition* methods and prepare your UI in case recognition system is missing. !!!**

static void StartUpdates()

Starts recognition process. In order to get results use *UserActivityListener* and its callback: *onUserActivityRecognized*.

static void StopUpdates()

Stops recognition process. Nothing special.

static AuthorizationStatus AuthorizationStatus()

Returns one of these values: *notDetermined*, *restricted*, *denied* and *authorized*.

Meaning can be expressed as:

Status	iOS	Android
notDetermined	User hasn't seen authorization dialog yet	-
restricted	Device does not support motion recognition	-
denied	User rejected authorizaiton dialog	Permission is missing in AndroidManifest.xml
authorized	User accepted authorization dialog	Permission is present in AndroidManifest.xml

UserActivityListener

In order to get callbacks from Activity Recognition you have to drag&drop *UserActivityListener* prefab from *UserActivityRecognition/Prefabs.* **Keep in mind that you cannot change name of that GameObject!**

Later on, you can register yourself to callbacks, as here:

```
GameObject.FindObjectOfType<UserActivityListener>()
    .onUserActivityRecognized
    .AddListener(myMethod);
```

On each detection you will gain a new instance of *UserActivityInfo* which returns boolean value for each of those activity types:

```
public enum ActivityType {
    automative,
    cycling,
    running,
    stationary,
    unknown,
    walking
}
```

You can determine each of those types by using *HasRecognized(type)* method or *recognizedType* property.

In addition *UserActivityInfo* contains confidence level being one of the *low*, *medium* and *high* and t*imestamp* pointing to a time when recognition took place.

Keep in mind that confidence is available only on iOS and on Android it will simply return *medium*.

Future plans

Suggestions appreciated!

Contact

Do you meet issues while using this plugin?

Do you have suggestions how to improve API?

Feel free to use our support forum at: http://www.kokosoft.eu/forums/forum/unity-plugins/