Mobile Sensor Visualization

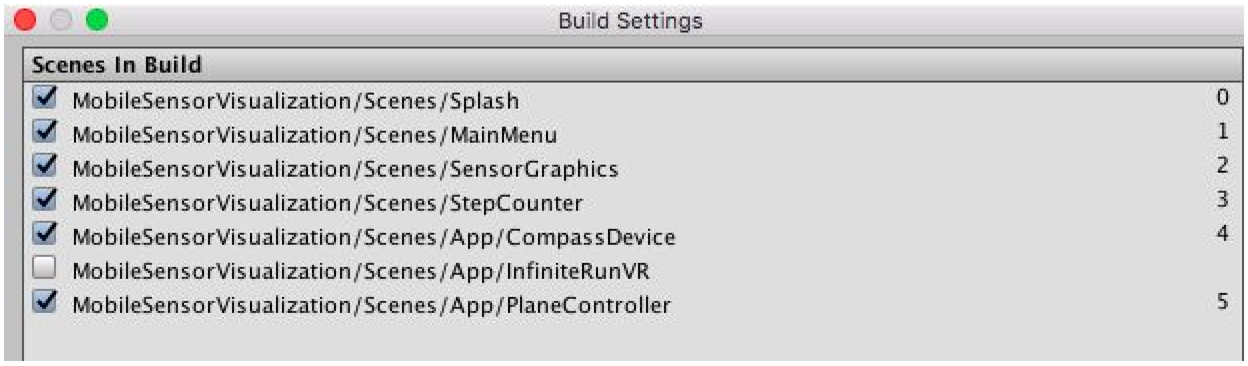
This package aims for sensor data observation via visual/graphical graph overtime including accelerometer, gyroscopes, compass, etc. Based on this visualization, you can add your own algorithm to detect/trigger any customized action. The code was designed to be clean and well inherited structure OOP for different algorithm development.

Apart from the main scene with visualized graphs, we propose three different demo scenes using three types of sensors including accelerometer, gyroscope, compass. Step Counter application is used to demonstrate the capability of algorithm development of accelerometer and gyroscope combination whereas the Compass Device provides the use of compass sensor data.

For more information, please don't hesitate to contact me via email dttngan91@gmail.com or udrawr@gmail.com

Demo:
Trailer 1: https://youtu.be/W0RoNNA16l8
Trailer 2: https://youtu.be/ozQTjmvI-7Y

This package provide 3 API of sensor data observation including Sensor Graphics, Gyroscope Graphics, Compass Graphics and Step Counter. Moreover, it also provide many demo applications such as Infinite Run in Virtual Reality mode or Plane Controller.
For better data observation, except Infinite Run, the other scenes in Scene folder which can be add to main menu to build run and test at the same time.

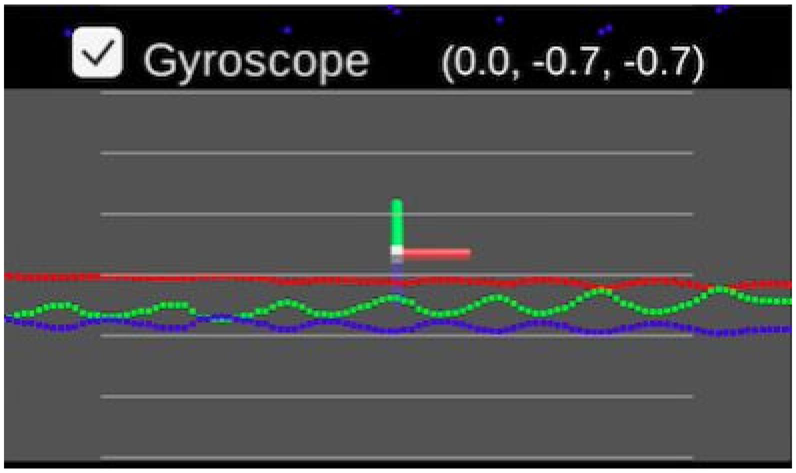


Scenes In Build

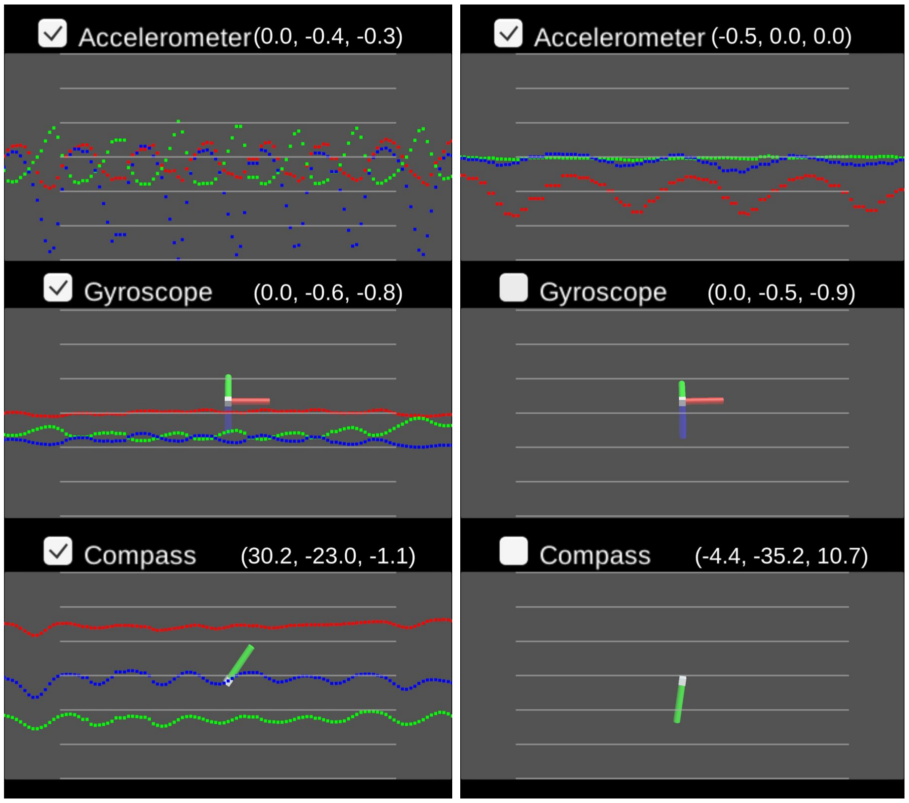
✓	MobileSensorVisualization/Scenes/Splash	0
✓	MobileSensorVisualization/Scenes/MainMenu	1
✓	MobileSensorVisualization/Scenes/SensorGraphics	2
✓	MobileSensorVisualization/Scenes/StepCounter	3
✓	MobileSensorVisualization/Scenes/App/CompassDevice	4
☐	MobileSensorVisualization/Scenes/App/InfiniteRunVR	
✓	MobileSensorVisualization/Scenes/App/PlaneController	5

Sensor Graphics

Sensor graphics is a wrapped GUI to display all data Vector3 into horizontal line (x, y, z coordinates). Based on the color of coordinates, red, green and blue are representative for X, Y, Z. The graph is automatically updated and added point every frame. The above graph label is gyroscope/accelerometer/compass for current selection sensor type and the last frame retrieved data. User can toggle the checkbox to enable/disable data observation from corresponding sensor type.



With SensorGraphics demo scene, we propose the use of three basic sensor on almost smartphones including Accelerometer, Gyroscope and Compass.

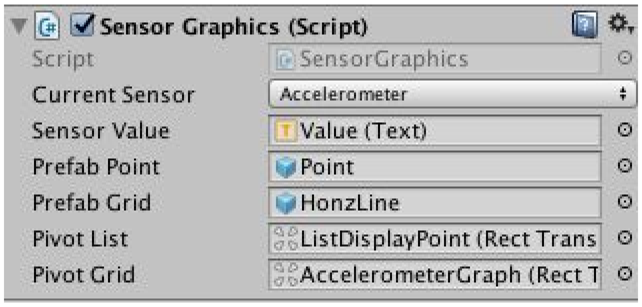

Accelerometer (0.0, -0.4, -0.3)
Accelerometer (-0.5, 0.0, 0.0)
Gyroscope (0.0, -0.6, -0.8)
Gyroscope (0.0, -0.5, -0.9)
Compass (30.2, -23.0, -1.1)
Compass (-4.4, -35.2, 10.7)

The below captured screenshot demonstrates how to use the API. Current sensor provides three selections.

- Accelerometer - Input.acceleration

- Gyroscope - Input.gyro.gravity
- Compass - Input.compass.rawVector

For more information about this component, please see the relevant Unity document.

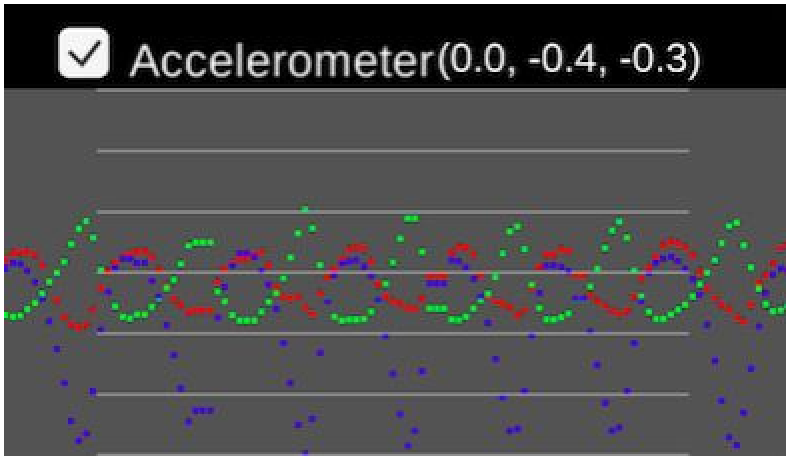


Sensor Graphics (Script)

Script	SensorGraphics
Current Sensor	Accelerometer
Sensor Value	Value (Text)
Prefab Point	Point
Prefab Grid	HonzLine
Pivot List	ListDisplayPoint (Rect Trans
Pivot Grid	AccelerometerGraph (Rect T

Graph drawing has two parts, including grid line and the pivot storing list of points. You can define the number of lines by *SetGraphicsSize()*, depending on data size. For example, accelerometer value is range from -3 to 3 (6 unit) and the drawing graph height is around 500 unit, then we define as following. As it always divide into 6 segments, it means within 500 visual size of the graph, each segments is 1 unit of accelerometer value. Likewise, compass value segment is range from -60 to 60, expands 120 unit, means each segment scores 20 unit.

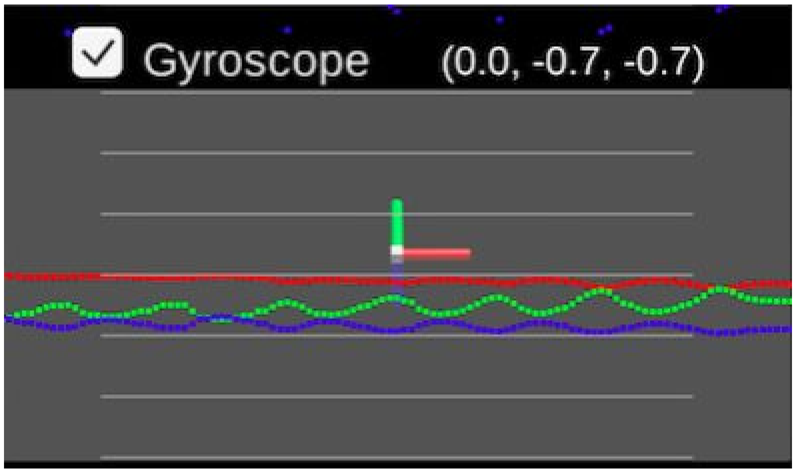
```
accelerometerGraph.SetGraphicSize(new Vector2(10, 3), new Vector2(Screen.width, 500));
accelerometerGraph.Init();
gyroscopeGraph.SetGraphicSize(new Vector2(10, 3), new Vector2(Screen.width, 500));
gyroscopeGraph.Init();
compassGraph.SetGraphicSize(new Vector2(10, 60), new Vector2(Screen.width, 500));
compassGraph.Init();
```

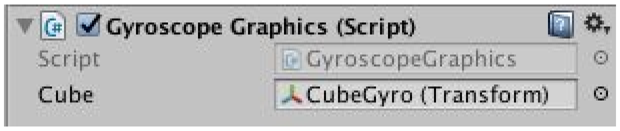


SensorManager is an example of how to use these APIs.

Gyroscope Graphics

Gyroscope data is also retrieved by gravity vector which match to the smartphone's rotation. To be more specific, gyroscope data has a 3D perspective coordinates behind the graph, which matching the rotation of devices based on the real gyroscope.





Gyroscope Graphics (Script)

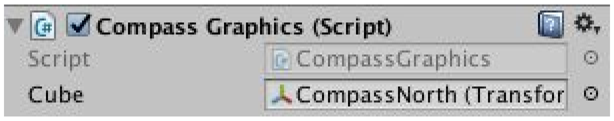
Script	GyroscopeGraphics
Cube	CubeGyro (Transform)

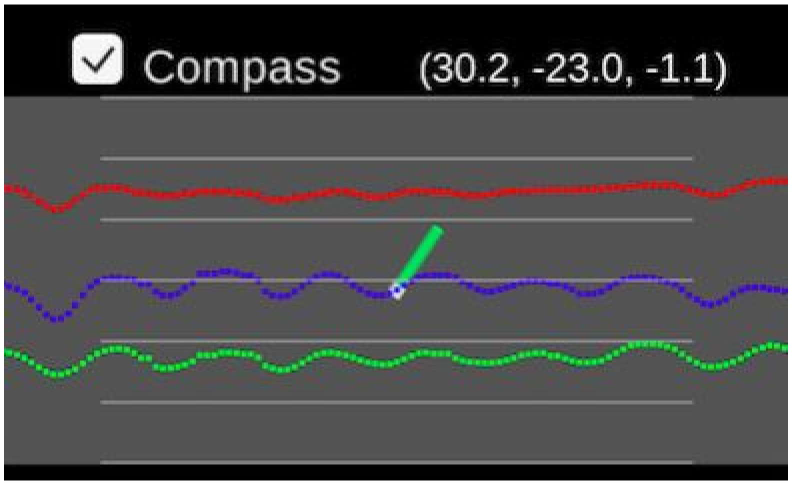
Once you would like to observe mobile rotation based on gyroscope, you can drag and drop the relevant gameobject to Cube parameter of *GyroscopeGraphics.cs.* For example, the game of airplane controlling using gyro.

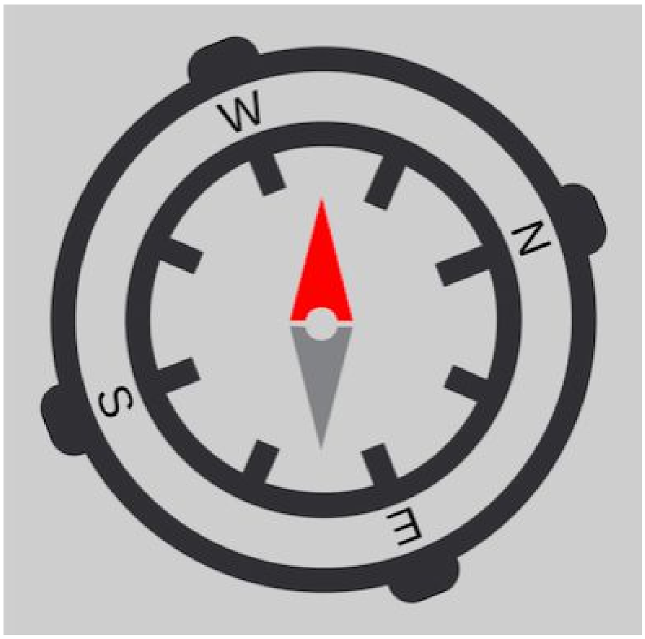
On of the most common algorithm using the accelerometer and gyroscope combination is **Step Counter.** We propose a simple algorithm with high accuracy (true positive) of step counter detection by spike detection and threshold comparison. Please see section 4 for more detail.

Compass Graphics

Likewise, the most concern compass data, which is the degree distance from the North Pole, is described by the green direction vector. *CompassGraphics.cs* is used for easier gameobject reference.


Compass Graphics (Script)
Script CompassGraphics
Cube CompassNorth (Transfor

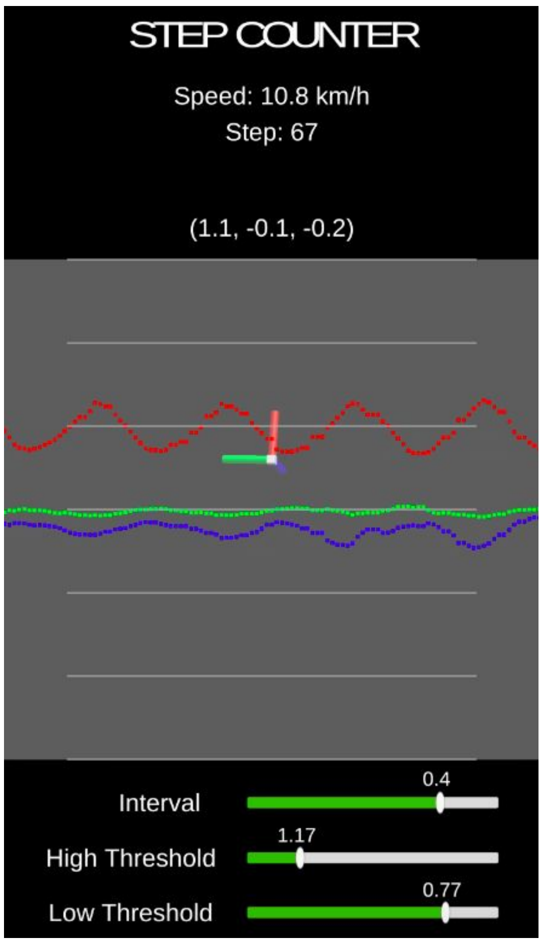




Step Counter

Step Counter simple algorithm allows user to adjust three parameters, including time interval, high threshold and low threshold. Time interval is maximum time stamp between the highest value and the lowest value. Following the flow of data, a spike is detected whether it is greater than the high threshold and a valley is below the low threshold. These threshold are determined by comparison to a center threshold whereas users can change the difference from the center threshold by slider bar. The center threshold may change based on the phone orientation and is retrieved by the horizontal value of gravity of gyroscope. In particular, each frame, we use one horizontal value of accelerometer and one horizontal value of gyroscope, which coordinate value is defined by screen orientation like the above image. **As this reason, this algorithm is valid in terms of smartphone placement at landscape in front of human eye (Google cardboard)**.

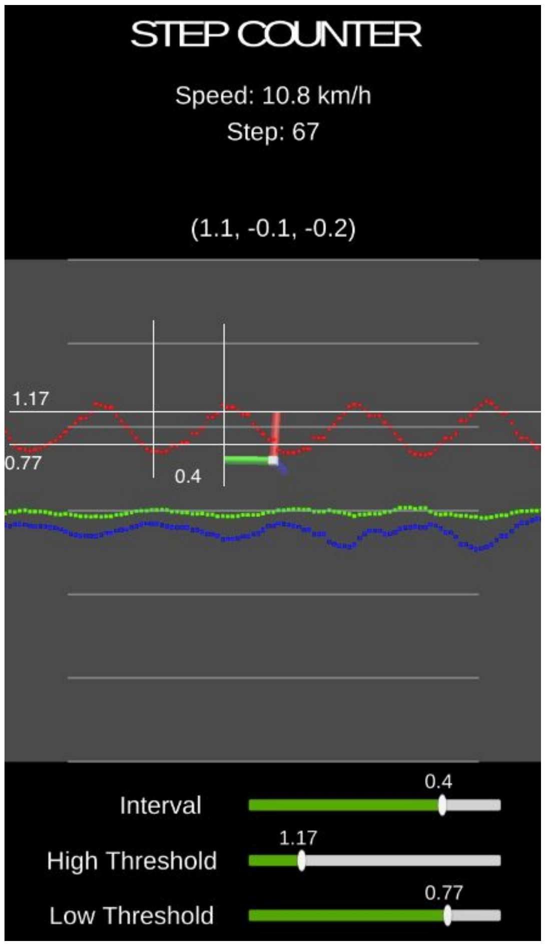
A step is counted whether a spike and a valley are detected and the time between them is smaller than the time interval threshold. Each human step distance is 0.75m in average, which is multiplied by the time span to derive the velocity. Speed, on the other hand, is calculated based on the average of the nearest 5 velocity values.


STEP COUNTER

Speed: 10.8 km/h
Step: 67

(1.1, -0.1, -0.2)

Interval 0.4
High Threshold 1.17
Low Threshold 0.77


STEP COUNTER

Speed: 10.8 km/h
Step: 67

(1.1, -0.1, -0.2)

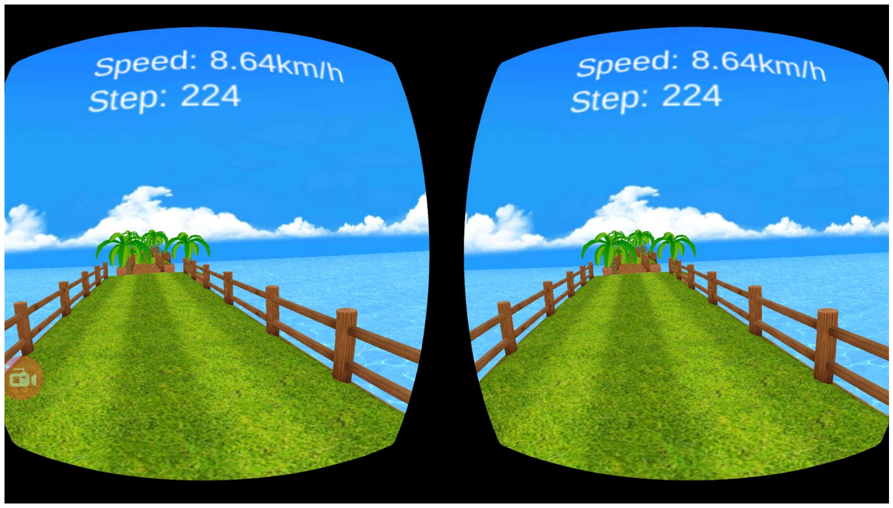
Interval 0.4
High Threshold 1.17
Low Threshold 0.77

Based on our own experiment, the recommendation parameters are 0.2 unit different from the center threshold and 0.4 unit of time interval.

We also provide a simple infinite run using this step counter algorithm.

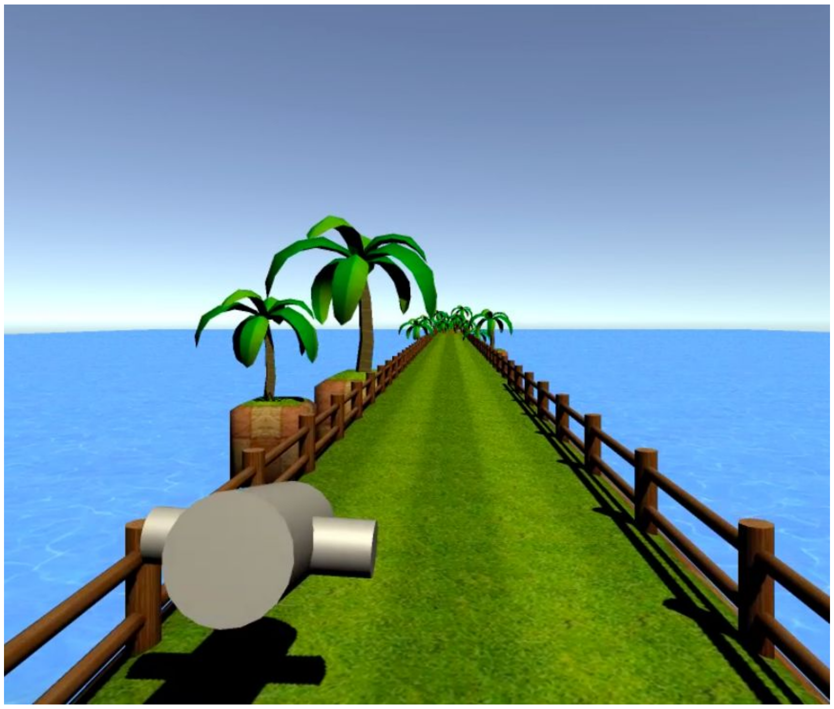
Infinited Run VR

To run this scene demo, please turn on the mode Virtual Reality support in Unity Project Settings.



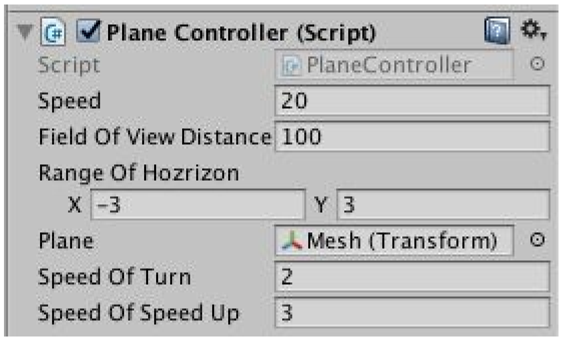
Gyroscope Plane controller

Taking advantage of gyroscope data, we provide a kind of game controller. Gyroscope of mobile device contains the useful information related to gravity vector with three axis. When we map this data with pivot of the plane, we can fully control the lean left/right or speed up of the controller which can be applied into many typical game, for example, plane fighter.



To be more specific, the scene namely PlaneController as well as the same script's name is inherited from the above PlayerController to control the plane to move forward. Otherwise, it provides many functions including lean the plane left/right but limit in the range of the road by tilting the corresponding the mobile device, speed up or hold down the speed of the plane's moving forward by tilting device forwards or backwards.

The script also allows to adjust many related parameters as following.



Plane Controller (Script)

Script	PlaneController
Speed	20
Field Of View Distance	100
Range Of Hozrizon	
X	-3
Y	3
Plane	Mesh (Transform)
Speed Of Turn	2
Speed Of Speed Up	3

It works across screen orientation because its algorithm is based on gravity of gyroscope data.

This was tested on Samsung Galaxy S6 (Android) and iPad air 1 (iOS). In case you used this and doesn't work on your devices, please give me informed your devices details.

If you have any questions, support, or idea contribution, please don’t hesitate to contact me via dttngan91@gmail.com or udrawr@gmail.com