Sensor Camera: Gyroscope & Accelerometer — Unity Asset: FPS Camera with Motion Sensors by makaka.org

Sensor Camera — cross-platform Unity Asset that provides a First Person Camera Controller (just like in FPS Games). It automatically selects a *Motion Sensor (Gyroscope or Accelerometer)* on **Mobile Devices** or *Mouse + Keyboard* on **Desktop Platforms**.

It doesn't use live video input from the Back/Rear Camera of a Mobile Device like in AR Camera Lite (docs) or in AR + VR: MR Camera (docs).

Contents [hide]

Features of Sensor Camera

Bring the enchanting Power of the Universal First-Person View Camera into your amazing Game or App:

- *Motion Sensors* on **Mobile Devices** (iOS, Android, WebGL):
  - Gyroscope or Accelerometer.
- *Spectator (Flythrough) Mode* on **Desktop Platforms** (Windows, macOS, WebGL):
  - Smooth Movement using WASDQE keys.
  - Smooth Camera Rotation using the Mouse.
- Auto Selection of User Input Device: Gyroscope → Accelerometer → Mouse + Keyboard.
- Device Orientations: Portrait, Landscape.
- Gyroscope Mode: 3DoF — it can track rotational motion, but not translational.
- Accelerometer Mode (horizontal rotation is limited): tilting the phone to the left or right rotates the camera around the Y-axis.

Package Contains

- Demo Scene with Cubes.
- Menu Scene with Safety Tutorial.
- Loading Screen to switch scenes seamlessly.

Check the Map of Unity Assets to choose the product that best suits your needs.

Gyro vs. Accelerometer

90% of all mobile devices have an accelerometer and video camera, but only 40% have a gyroscope.

If the user's smartphone has a Gyroscope, then it will be used for camera motion first. Otherwise, an Accelerometer will be used because it has less accuracy & stability than a Gyroscope in the case of the First-Person View.

Limitations

Hardware nuances of the gyroscope & accelerometer (asset code does not affect it):

- Different devices have different sensors, and therefore different deviations and drifts.
- Drift is natural for the mobile sensors.

Known Issues

- **iOS standalone**: *Orientation Changing during Gameplay causes delay in the Game* — Bug in the Unity Engine. *Solution*: don't use Auto Rotation. Limit the Device Orientation as described below.

Pro Gamer Tip

Accelerometer & Gyroscope are used in games and apps to control gameplay like in PUBG MOBILE game. Sometimes these sensors can be set up incorrectly for some reason & break the gameplay. If you guess that your drift of gyro or accelerometer is not normal, then try to calibrate them with system tools provided by your smartphone manufacturer.

Use Cases of Sensor Camera

1. Basketball Game 3D (docs).

---

2. Football Game 3D (docs).

With adding live video input from the Back/Rear Camera of a Mobile Device, Sensor Camera can be used as a "Pseudo AR Camera" to display 2D or 3D objects as though they were in the real world.

This technique was implemented in AR Camera Lite (docs).

---

3. AR Shooter (docs).

Tutorial

> *This tutorial is relevant for Sensor Camera 3.0+.*
> *Tutorial for the previous version can be found only in the asset folder.*

Getting Started with Sensor Camera

Folders & Files in the package by default:

- Makaka Games.

Steps

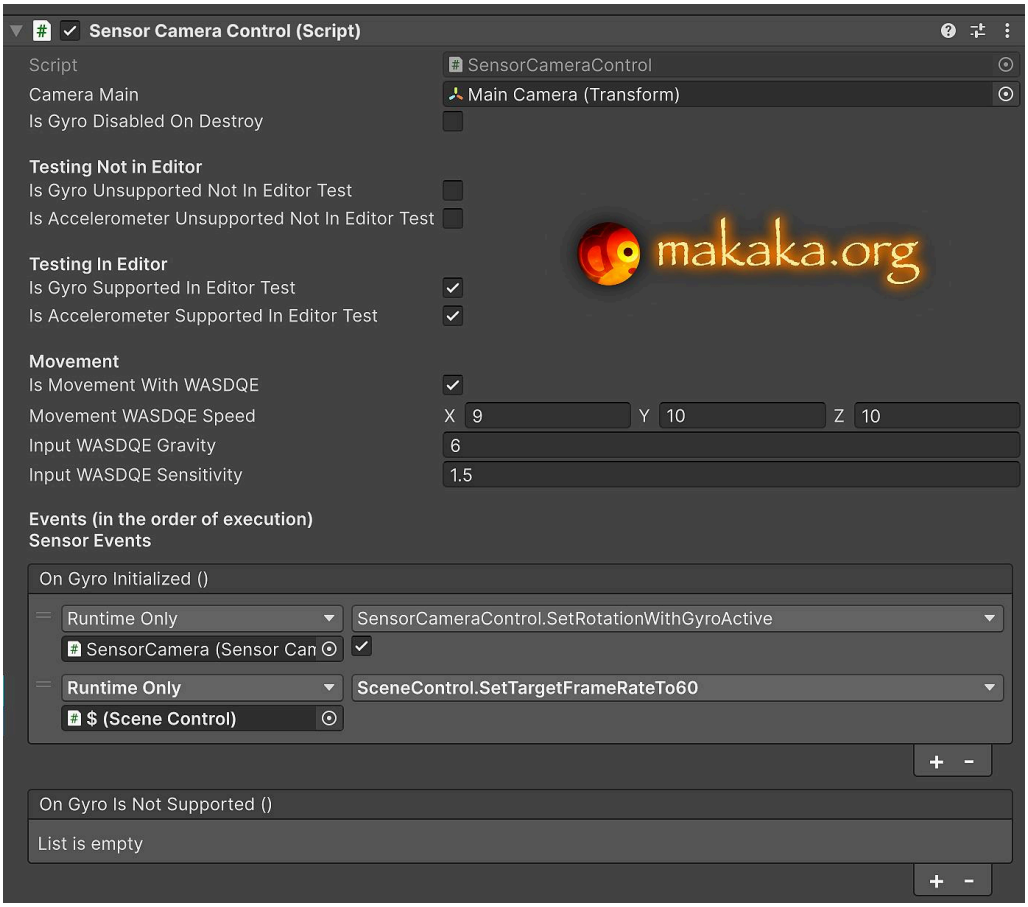
> *If you have any issues with the first launch, then just Reach Support with Invoice Number and Get Help.*
>
> *If you read this tutorial from PDF, first check the latest docs online to get actual information.*

1. Create a New Unity Project with Unity 6000.0.28 & “Universal 3D” Template.
2. File > Build Settings > Windows, Mac, Android, iOS, WebGL > Switch Platform.
3. Download and Import Sensor Camera into Unity.
4. Window > TextMeshPro > Import TMP Essential Resources.
5. Reopen Unity Project.
6. Open Scene: Makaka Games > Camera > Sensor Camera > Scenes > *Demo*.
7. File > Build Settings > Add Scenes to build:
   - ⭐ *Menu_SensorCamera*,
   - ⭐ *LoadScreen* (from Makaka Games > Publisher > Everyday Tools > SceneControl > Scenes),
   - ⭐ *Demo_SensorCamera*.
8. ***Required for iOS standalone:*** Limit the Screen Orientation from Auto (by default) to the Portrait only or Landscape only by indication of appropriate Scene Loading Methods on the corresponding buttons that cause a transition to another scenes. Orientation in the Player Settings is used for the 1st scene in the build.
9. ***Option for macOS standalone:*** Edit > Project Settings > Quality > VSync Count > enable any option to avoid Screen Tearing.
10. Test in the Unity Editor or Build for Mobile.

Useful Article: How to Test iOS App without Developer Account?

Prefab and Script Reference

Asset provides a **Demo Scene** that is **ready to use**: you don’t need to customize anything. You can also use *SensorCamera* **prefab** that has a main camera control script: *SensorCameraControl.cs*.



Parameters

Flag: Is Gyro Disabled On Destroy

If it's *"true"*, then Gyro's "Y" Rotation is reset on Scene Closing or Reloading. Useful if you need to Control the Start Rotation of the Camera when Restart.

Testing Not in Editor (Flags)

Is Gyro Unsupported Not In Editor Test

If it's *"true"*, then the Gyroscope will be unavailable on Gyro supported devices. It is used when you need to test the Accelerometer on Gyro supported devices.

Is Accelerometer Unsupported Not In Editor Test

If it's *"true"*, then the Accelerometer will be unavailable on Accelerometer supported devices.

Testing in Editor (Flags)

Is Gyro Supported In Editor Test

If it's *"true"*, then you can test Gyro-related Events in Unity Editor.

Is Accelerometer Supported In Editor Test

If it's *"true"*, then you can test Accelerometer-related Events in Unity Editor.

Movement

Is Movement With WASDQE

If it's *"true"*, then you can fly in the scene when Play Mode.
It can be used for different testing scenarios or for the Gameplay.

Movement WASDQE Speed

You can set Speed Value for Each Axis separately.
This is the final factor in the smooth movement.

Input WASDQE Gravity

The lower the value, the smoother the movement you get.
Vector3.MoveTowards() is used.

Input WASDQE Sensitivity

It is applied immediately after the key is pressed before setting a Speed.

Events (in the order of execution)

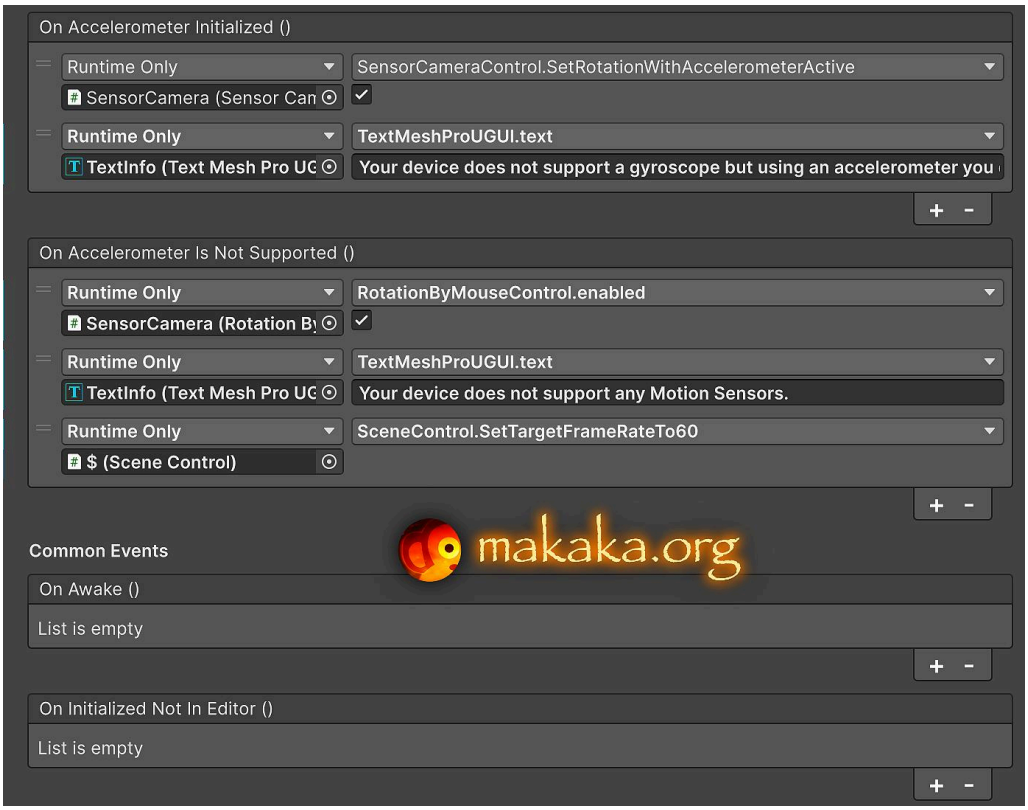
Sensor Events

On Gyro Initialized

The case when the smartphone has the Gyroscope.

On Gyro Is Not Supported

The case when the smartphone has not the Gyroscope.



On Accelerometer Initialized

The case when the smartphone has the Accelerometer and doesn't have a Gyro.

On Accelerometer Is Not Supported

The case when the smartphone has not the Accelerometer and a Gyro.

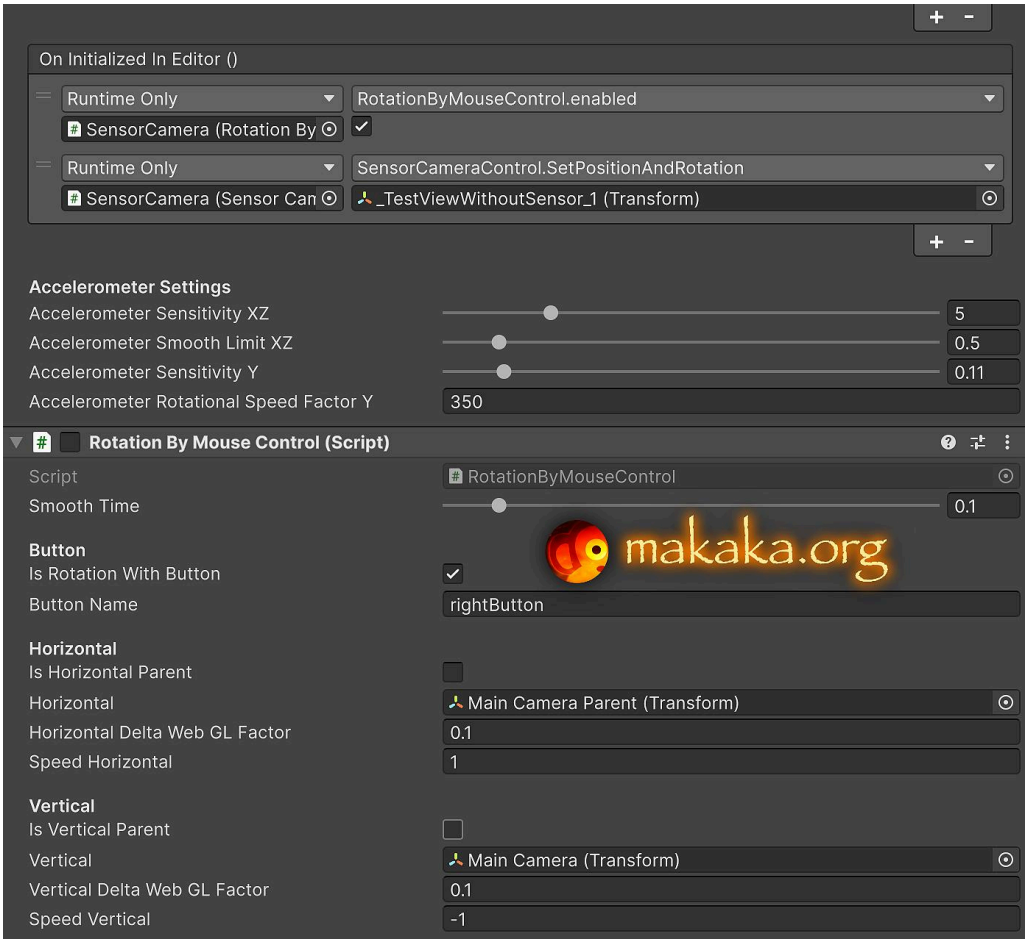
Common Events

On Awake

It's invoked in the default *Awake()* function.

On Initialized Not in Editor

The case when Runtime on Device.



On Initialized in Editor

The case when Play Mode in Unity Editor.

Accelerometer Settings

Since the accelerometer cannot provide the same camera rotation natural experience as a Gyroscope, you can adjust the Accelerometer Sensitivity required for your game.

Testing

There are 2 Ways of Testing Device Experience without building an app:

- Testing with Unity Remote & Smartphone connected to Computer.
- Testing without a Smartphone in Unity Editor.

Testing without the Smartphone in Unity Editor

You can test **Camera Rotation** in Unity Editor without a Smartphone through the Right Mouse Button (or any other mouse button, or without a button pressed).

Also, you can use WASDQE keys to move the camera.

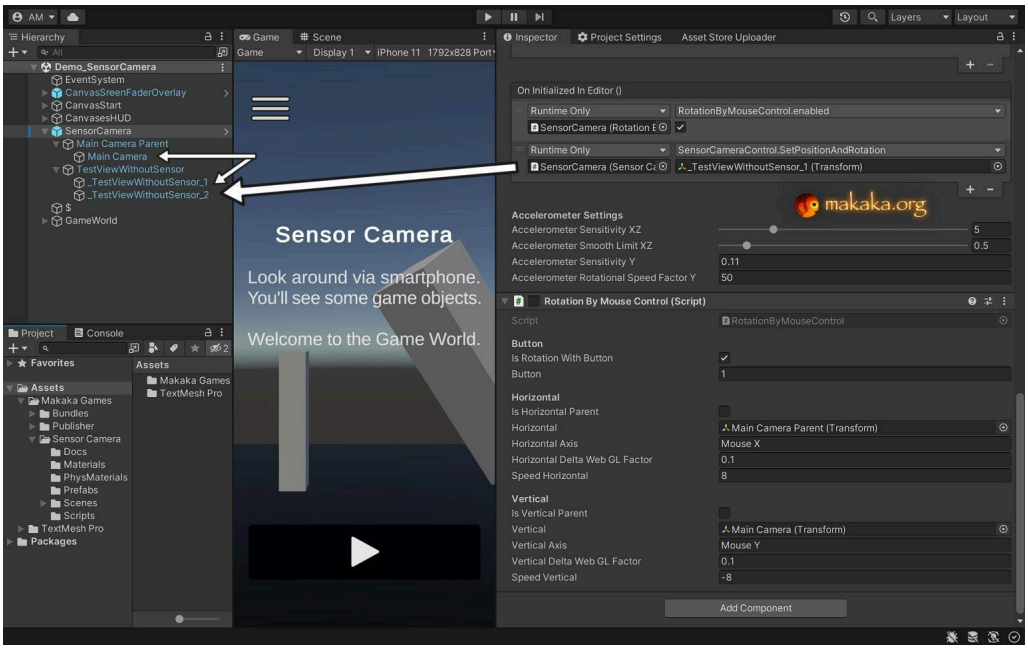
> *Use Fullscreen of Game View in Unity Editor while testing to get a seamless experience.*

Accelerometer Instead of Gyro

You can forcibly test Game Version with **accelerometer** on **gyro** supported devices:
Hierarchy View > *SensorCamera* > *SensorCameraControl* > *Is Gyro Unsupported Not In Editor Test* > *Check*, then Build And Run.

Predefined Data

Also, you can start the scene with Predefined Data of position & rotation. It's a convenient way to frequently test the same positions and rotations of the camera: *"Main Camera"* Game Object.



So, you can save these data with custom *Transform* components on Game Objects as shown on the screenshot: as children of *"TestViewWithoutSensor"* Game Object.

These transform components are parameters for the function called *SetPositionAndRotation (Transform)* which is executed on game start when *"OnInitializedInEditor"* event is called.

Use Case

I used this testing method when developing Basketball (docs). I needed to periodically test Normal Ball (with touching of Ring) & Clear Ball (without touching of Ring). Since throwing "Clear Ball" is a hard task, I saved 2 different camera *Transform* components to change them when needed:

1. Right Above the Ring;
2. A Few Meters from the Ring.

Testing Time was decreased well because I didn't need to take the mobile phone every time in my hands after changes in the game logic and move the phone manually. Instead of it, I had predefined data.

WebGL

Learn the Article called WebGL and Unity about Building and Testing Unity games and apps for WebGL.

Tested with Platforms

1. ***Mobile Platforms*** with *Motion Sensors*:

   - ⭐ **iOS** on iPhone 15;
   - ⭐ **Android** on Samsung Galaxy A71;
   - ⭐ **WebGL** in Google Chrome on:
     - ⭐ *Own Website* using this tutorial for iOS & Android.
     - ⭐ *CrazyGames Website* for Android.

2. ***Desktop Platforms*** with *Right Mouse Button + WASDQE keys*:

   - ⭐ **Windows**;
   - ⭐ **macOS**;
   - ⭐ **WebGL** in Google Chrome on:
     - ⭐ *Own Website* using this tutorial.
     - ⭐ *CrazyGames Website*.

Support

First, read the latest docs online.
If it didn't help, get the support.

Changelog

Check the current version of Sensor Camera on Asset Store.
The latest versions will be added as soon as possible.

3.0:

Improvements:

- ⭐ Unity 6000.0.28.
- ⭐ URP (New Standard in Unity starting with Unity 6): *instead of BRP*.
- ⭐ Input System (New Standard in Unity) for All Camera-related Scripts: *instead of Input Manager*.
- ⭐ User Input on ***Desktop Platforms*** in the Build and in the Unity Editor:
  - ⭐ **Spectator (Flythrough) Mode** with Smooth Movement using **WASDQE keys**.
  - ⭐ **Smooth Camera Rotation** using the **Mouse** with or without the custom mouse button pressed.

2.1:

Features:

- iOS WebGL Support via HTTPS (the bug was fixed by Unity without confirmation).

Improvements:

- Unity 2022.3.18.

2.0 (WebGL Support):

Features:

- WebGL Support via HTTPS: Android.
  - Check information about iOS in the *Limitations* section.

Improvements:

- Unity 2022.3.6.

1.2:

Fixes:

- Camera Y Rotation Sensitivity with Accelerometer for Different FPS.

1.1 (New Settings of Accelerometer – can be set in Unity Editor):

Features:

- Checkboxes for Testing:
  - Is Accelerometer Unsupported Not In Editor Test;
  - Is Accelerometer Supported In Editor Test.
- Events:
  - On Accelerometer Initialized;
  - On Accelerometer Is Not Supported.

Improvements:

- Unity 2021.3.21.