



SIMPLE FLIGHT CONTROLLER

Manual

Version 1.0.0

Simple Flight Controller

© Gojo Entertainment

Please contact me if you have any questions:

diorgo@gmail.com

Contents

Overview.....	3
How to setup the InputManager.....	4
Importing into an empty project.....	4
Importing into an existing project.....	4
How run (or build) the demo.....	5
Ship Hierarchy.....	6
Pivot Points and Axes.....	7
How to setup a ship from scratch.....	8
Tips.....	11

Overview

This manual contains information about Simple Flight Controller. Simple Flight Controller is a Unity Asset that contains player controllers and a follow camera script for flight based games. It can be used to control a spaceship in space and an aeroplane flying over a landscape.

How to setup the InputManager

Importing into an empty project

After importing you will find the file "InputManager.zip" in the SimFlight folder. The zip contains the **InputManager.asset** file. Close the Unity project, then extract the InputManager.asset file and move it to your project's **ProjectSettings** folder.

InputManager.asset contains the input settings for the game. These have been setup via Unity's Input Manager.

Importing into an existing project

If you have setup your own input settings in Unity's Input Manager then you should not use the **InputManager.asset** supplied with the package. Instead you need to change the keys, buttons and axes in the scripts:

SfcControllerBase.cs

SfcDemoPlayer.cs

How run (or build) the demo

After importing the package, execute the menu item:

Tools\Simple Flight\Setup Demo

It will add the following demo scenes to the build settings ("Startup" must be the first scene):

Startup

MainMenu

Ground

Space

You can test the demo in the editor by running the Startup scene.

Ship Hierarchy

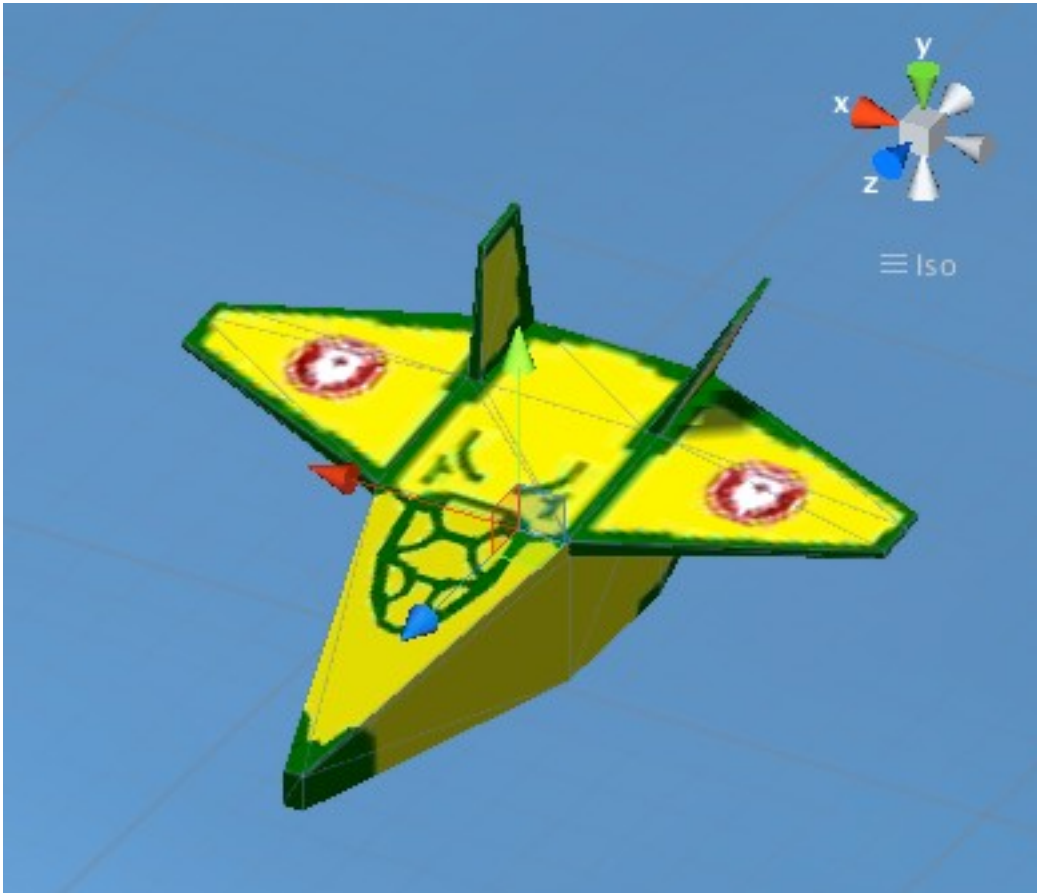
The mesh (e.g. FBX) should be a child object to the game object that has the ship controller, because the mesh is rotated independently (e.g. mesh rolls when player turns left/right).

The controller's property **Mesh Child Or Prefab** must be set to point to the mesh child (or to the mesh/FBX asset to spawn and attach as a child when the game runs).

(See ***How to setup a ship from scratch***)

Pivot Points and Axes

The ship's pivot should be roughly in the centre of the ship, so that it looks good when rotating.



The ship's axes must be aligned so that Z points forward, X to the right and Y up. If your mesh's axes are not aligned this way, then there is an option to spawn and attach the mesh as a child object to the controller (which is the preferred method). See the **Mesh Child Or Prefab** property on the ship's controller component.

(See ***How to setup a ship from scratch***)

How to setup a ship from scratch

Step 1:

Add an empty object to the scene and name it "MyShip". (It can be any name, we are just using MyShip as easy reference in the rest of the steps.)

Step 2:

Attach one of the flight controllers to MyShip:

SfcControllerFreeRoam.cs *(Use for a space based game.)*

SfcControllerGround.cs *(Use for a ground based game.)*

Step 3 (option A):

Drag your ship's mesh/FBX into the scene and attach it as a child to MyShip. Rotate the child mesh so that its front points down the positive Z axis of MyShip, and its top points up the positive Y axis of MyShip.

Set the controller's property **Mesh Child Or Prefab** to reference the child mesh.

OR

Step 3 (option B):

Set MyShip's property **Mesh Child Or Prefab** to reference a mesh/FBX asset. The mesh will be spawned and attached to MyShip when the game runs.

Step 4:

Select the **Input Type** on the ship's controller (e.g. keyboard, gamepad, touch accelerometer).

Step 5:

You can add a **Character Controller** or a **Rigidbody** to MyShip. The Character Controller is easier to tweak to get the desired movement/rotation results, but it is limited to a capsule collider. You must use a Rigidbody if you want your ship to receive collision callbacks, or to use different shaped colliders.

If you add a Rigidbody then:

- Disable the Rigidbody's gravity.
- Add a collider to MyShip (or to one of its children). Ideally, use a simple collider for optimisation (e.g. Sphere Collider, Box Collider).
- You may need to tweak the controller's rotation speeds (e.g. yaw, pitch, roll), and also "Clear Rigidbody Angular Velocity" which is the rate at which rotation is slowed down so it does not rotate too long (or forever).

Step 6:

Add a camera to the scene (if there is not one already).

Step 7:

Add the component **SfcFollowCamera** to the camera.

Step 8:

Set the **Mode** on the camera component to one of the following:

Ground Based	<i>(Use for a ground based game.)</i>
Ground Based Low	<i>(Use for low flying ships in a ground based game.)</i>
Free Roaming	<i>(Use for a space based game, or ground based.)</i>
Cockpit	<i>(Camera will be attached to the inside of the ship.)</i>

Note: Avoid using a ground based mode to follow a ship which has a free roaming controller, because the camera's rotation will be limited.

Step 9:

Run the scene.

Test out the different controllers to see which fits your game, and tweak their properties as needed.

Tips

Tip 1: Many camera prefabs

If you have various player ships that behave differently then the game might feel better if you create a camera for each ship, instead of a single camera for all the ships. Therefore each camera can be tweaked to suit the ship's behaviour.

Tip 2: Camera affects ship

The camera affects how the ship feels. If you tweak the ship and change the camera afterwards, then the ship might feel different.

Tip 3: Rigidbody freeze rotations

This only applies if your ship is moved via a Rigidbody:

If the ship behaves strange (e.g. sticks to surfaces) during collisions then you can try to freeze the Rigidbody's rotations. You may need to increase the rotation speeds on the controller (e.g. yaw, pitch, roll).

The strange behaviour is more likely with the free roaming controller, because it uses torque to rotate the Rigidbody.