Rigidbody FPS Controller

Quickstart guide

PS: This is a large WIP from one person. Please excuse the very limited quickstart guide

Creating your own character

Creating the player
Creating a new player is a very simple process. Simply create a new empty gameobject in your scene, and attach the Motor.cs script to the object. This should automatically add a collider and rigidbody. You do not need to touch the two components added by the script, as they are handled by the motor. Change the settings as you like to fit your project.

This is technically the only needed script in order to run the controller, however, the motor serves as the brain of the controller and does not have any functionality (except gravity) on its own.

Setting up the camera
In this controller, the camera has to be the child of a camera holder object. The camera holder should be another gameobject with the CameraController.cs script attached. To set this up: Camera in scene → Right Click → Create Empty Parent → Rename new object → Add Component → Camera Controller.

After adding the script, you have a lot of options for how you want to control the camera. Make sure to reference the player object in the designated field. Any other settings are up to how you want your game to play.

Some scripts, including the camera controller, use the CameraUtility.cs script. You should add this component to the gameobject containing your camera component, which should be the child of the newly created gameobject.

Important note: The camera should not be a child of the player, as this will introduce jitter since the camera will move with rigidbody.

Adding modules
By now you can look around and fall down, which is not exactly exciting gameplay. In order to extend the functionality, you will have to attach modules to the motor. These are scripts inheriting from Module.cs that can be attached to either the same gameobject as motor, or in a child in the motor. Since there can be quite a large amount of modules active, I recommend creating an empty gameobject as a child of the player where you add all your modules.

To quickly get up and running (or walking I suppose), add the Movement.cs script to either the child you just created, or onto the gameobject of the Motor.cs. Here you can tweak the values for movement on the ground and in the air.

If you start playing the game again, you should now be able to move along the horizontal plane! I recommend that you try playing around with different values to see how it affects the movement. You can check out the module section in this document in order to learn about the different modules in the package along with a more detailed explanation of what all parameters do.

Modules

This section is a WIP and will be updated later. Currently, you will just find a list of modules and a brief description of what they do. For more information about how they work, you can read the source files themselves, as they are quite well commented.

Module	Description
Movement	This module controls the movement of the motor on the horizontal plane. You can modify ground and air movement freely.
Crouching	Allows the shrinking of the collider and reduction of movement speed.
Jumping	Implement jumps into a sequence of jumps in order to allow for variable amounts of jumps.
Footsteps	Contains various ways to play footsteps for the motor.
Headbob	Moves the camera up and down with movement.
Sprinting	Increases the movement speed of the motor and optional FOV changes.

Additional functionality

Audio bundles
To more easily manage collections of sounds, the controller implements and uses ScriptableObjects called audio bundles. These are just a list of audio clips with extra parameters

you can assign to each clip, like volume, weights, tags etc. You can create a new one via the asset menu.

Camera utilities

Camera utilities is a class responsible for providing multiple methods for camera effects. Controls camera shake, showing/hiding of cursor, setting of FOV.