Modular Player controller Documentation

Here you will find an in depth explanation of the modular player controller I made for my final project.

It will explain each section in detail for non-programmers and how a more experienced person can add in their own variables to suit their own game.

This controller provides a robust base in which you can either add it to your project to test how a certain character will act in your game or, for you to edit any way you see fit for it to better suit your project.

Legal Disclaimer

You are free to distribute, remix, adapt, and build upon the work, even commercially, as long as you credit the original creation by linking to my website and GitHub Repository.

Website: https://www.ethanward.co.uk/

GitHub Repository: https://github.com/E-Ward/FinalProject-Dissertation-

Contents

Movement Keys

Movement variables

Attributes

Damage

Camera variables

Camera positions

Capsule collider variables

Scripts

**Movement Keys**

This section is where all the keys for the player controller are created and stored. It uses Unity’s inbuilt Keycode function to provide maximum compatibility and to mitigate user error as you pick from a drop-down menu of predefined keyboard keys in the inspector.

To add a new key to the controller use the code:

(Enter the name you would like the key to be called after Keycode, not including the brackets)

public KeyCode “Name of key here”

Making it public will allow it to appear in the inspector and be accessed by other scripts.

To make use of the newly created key add it in whenever you are setting a key in the controller e.g.

Instead of:

if (Input.GetKey(KeyCode.W))

{

transform.position += transform.forward \* currentSpeed \* Time.deltaTime;

}

In this example only W will make the code in the if statement work

You should use:

if (Input.GetKey(“Key code name hear”))

{

transform.position += transform.forward \* currentSpeed \* Time.deltaTime;

}

Entering the name of the keycode you made in the brackets, with no quotation marks, will then make the code in the if statement work whenever the user presses the key set in the inspector.

**Movement variables and Attributes**

This section includes all the variables associated with movement. If you want to add in a new variable for your character create a public float in this section. As this is in the movement variables it will most likely be some kind of speed related variable.

The public float will allow the user to ender the number they want for their character in the inspector.

There are also booleans in this section. They are used so when a movement action is happening, the Boolean can be activated to let other parts of the code know the action is happening.

Currently in the character controller Booleans get activated if the character is running or crouching.

Attributes is like movement variables. It is where the health and stamina related variables are stored.

**Damage**

This is where you can store the values for how much damage the character takes for specific objects. Currently in the controller there is a variable for fire damage and explosion damage.

A demo script is in the main scene that applies fire damage to the character to give an example on how you can implement a similar function to your own game.

**Camera variables**

This is where the camera variables are stored. At the moment only the camera sensitivity can be changed in the character controller however, if you need want to add more camera features to the controller this is where to do it.

**Camera positions**

Camera positions is a set of transforms that the user can drag and drop empty game objects to in the inspector to assign them.

The camera positions work by using blank game objects positions. You can place an empty game object in the location you want a camera to be and then once you drag it into the inspector the transform can be used to move the position of the camera to the position of the empty game object.

**Capsule collider variables**

While the collider is attached to the same game object the character controller is, I wanted the controller to be as feature rich as possible. Because of this you can edit the collider variables in the character controller.

If you want to be able to edit a new collider, make sure to make a public collider variable.

Then you can create floats for the attributes of the collider that the user can edit in the Unity inspector.

**Scripts**

This section is where all of the scripts the controller needs to access are stored.

Currently the controller is only accessing one script. This is the mouse script. It is so the mouse look sensitivity can be changed in the character controller.