Project 02B

FPS Game Player Controller

# Homework Description

In this Homework Assignment, you will create your own FPS Controller from scratch, inside of your previously built Game Scene. This will teach you some of the fundamentals for putting together a good Character Control system for an FPS-style game. We will be using a written document for our learning resource to mix up our learning styles.

Once you have completed your character controller, you will make sure that your overall FPS project is still navigable from beginning to end. Player should be able to move from Main Menu, to Game with the player controller, back to Menu and finally Quit.

# Required Resources

FIRST PERSON MOVEMENT in Unity – FPS Controller – Brackeys (23:52)

<https://youtu.be/_QajrabyTJc>

Cursor Lock/Hide (Unity Documentation)

<https://docs.unity3d.com/ScriptReference/Cursor-lockState.html>

<https://docs.unity3d.com/ScriptReference/Cursor-visible.html>

# Requirements

*\*Denotes a more challenging requirement. Attempt the others first if you’re having difficulty*

1. **Continue to build off of your previous assignment**, and create your Player Controller inside of your Game Scene
2. **View the Concepts material** (if available) to understand the concepts covered in this homework assignment.
3. **View the Required Resources material** (if available) to understand the specific requirements associated with the assignment
   1. Your player **must move according to the final Player Controller** built in the document, though the exact speed/jump values (etc.) are up to you and the ‘feel’ you’d like to create:
      1. Must **move with A,W,S,D**
      2. Must **rotate camera with mouse**
      3. Must **not be able to look past straight up, or straight down.** Clamp the camera.
      4. Must **jump with Spacebar**
      5. Must **not be able to jump in mid-air**
4. **Implement the following functionality on your FPS Controller:**
   1. **Sprinting** – In script whatever script you’re detecting Input, detect when Left Shift is pressed and when it is released. Increase speed amount when pressed and return to original value when released.
   2. **Create a simple shape to represent the player’s Primary Ability/Weapon/etc.**
      1. Position this so that it shows in the bottom-right portion of the camera
      2. Make your weapon a child of the Camera, so that it follows our view when looking up and down
      3. ***NOTE:*** *Remove colliders on your ability/weapon visual! Otherwise it will give your character weird collision events.*
   3. **Left Mouse Button - “Fire”** – Detect if Left Mouse button is pressed, and if so, activate your primary Ability/Weapon
      1. Triggers a One-Shot **particle Effect** at the end of your representative ‘weapon’. This should simulate a ‘muzzle flash’.
      2. Plays a “Shoot” **sound effect**
      3. *NOTE: make sure you use GetKeyDown, NOT Get Key, or it will rapid fire*
      4. *NOTE: this doesn’t need to do anything functional yet. This week is just triggering it properly*
      5. **HINT:** if you’re stuck, this requirement will have a very similar setup as Jump implementation, except without any of the additional grounded checks.
   4. **“Cursor Lock”** – Make sure that cursor locks during Gameplay, but unlocks during menu navigation
5. **In-Game Toggle Menu** – Pressing Escape should bring up the pause menu and **Unlock/Lock** the cursor appropriately
   1. Cursor should **Hide/Unhide** appropriately
   2. Pause Menu does not need to freeze time! (but it can, if you want for Innovation)
6. **Player Health System** - Create a Health system for the player
   1. **Player must take damage** that subtracts health
   2. If player loses all health, the **player must die**
      1. What happens upon player death is up to you, but you must give an option for the player to continue playing/retry/etc.
7. **Damage Volume** – Create a damage volume that applies damage to the player on enter
   1. Primarily, this is for testing your damage system
   2. Give your damage volume an obvious material, or I will not notice it and you will not receive credit
8. **Player HUD** - Create the Player HUD (UI elements display game information about player state)
   1. **Player Health Bar** – Graphical representation of current health as a Bar/Slider
   2. Must scale properly with final executable
      1. Executable will look different than Editor, so make sure to check!
   3. Health must update properly anytime your health changes in game
9. **Make a Commit and Push to your Remote Repo**
   1. This will allow you to save this week’s progress in case something goes wrong later

# Deliverables

Create a folder inside of the appropriate Turn-in folder with the following naming convention:  
**LastnameFirstName\_P##\_Progress** (ex: ChandlerAdam\_P02\_PlayerController)

In this folder you must include the following:

1. **.zip** that includes **Exe with all related build files** . Double/Triple check your .exe to make sure it runs before submitting your Build folder to box.com! *(…HW##.zip)*
2. A **README file** in the top of the folder that contains the following:
   1. **Link to public Repository** on GitHub
   2. **Control Scheme** and any required information for playing your game
   3. (optional) Any difficulties with the assignment or any other info you’d like to communicate to me

**NOTE: If you turn in your Source files (your Unity Project folder) in any way, I will deduct points!**

If you have questions about this, please ask

# Grading Criteria

**10% - Deliverables** – Executable runs when played, all deliverables turned in and named properly

**10% - Project Repo** – Link to Project Repository is provided with turn-in

**80% - Requirements** – All Assignment Requirements met

# Notes

If you get stuck:

* Ensure that your code matches the provided examples
* Double-check to make sure you have drag/dropped your references into the Inspector
* Make sure that you have attached any new Scripts you have created
* Sometimes you may get temporary errors until you finish the longer example. If the error is looking for a function that does not exist yet, consider continuing to follow along with the example to see if it gets fixed.
* Do independent research to Debug/Troubleshoot your issue. Worst case scenario, get things functional using whatever method you can (even if it’s different than the provided example).

# Optional Resources

*FPS Controller Setup Document*

This is a document-style resource for a different-style FPS Controller with more modularity. If you prefer documents over videos, you can use this.

<https://utdallas.box.com/shared/static/2dmp5qbkxddh3kspvopoyi2ancu7avqk.docx>