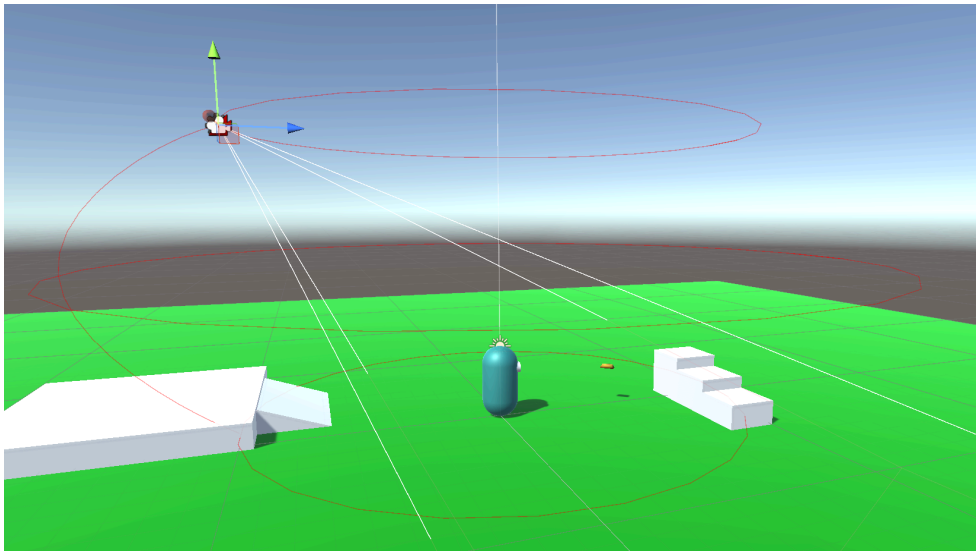
Jacob Yankee
COSC 457
Spantidi

Enhancements:

- The first change I wanted to make was the camera as it genuinely made me nauseous. From what I could tell, the easiest way to do this was to use the Cinemachine package. Doing this was relatively simple, mainly because I was following a tutorial, but it used zero code at all. I did end up tinkering with the sensitivity to get something I liked, but overall it was very straightforward. Attached below is how the Cinemachine camera looks in the scene view with its three orbits: top rig, middle rig, and bottom rig.



- Next up, I decided to add a simple sprint mechanic for the speed change requirement. I had written code before to make a slow movement button, so I just reused the code there. However, my initial implementation either allowed the player to fly or do a very small jump, so I had to add an extra conditional to let you jump normally out of a sprint. I wanted to include a proper sprint jump that could cover longer distances, but I couldn't really figure it out in a reasonable amount of time.

```
if(Input.GetButton("Fire3"))
{
    moveDirection = moveDirection.normalized * movementSpeed * sprintSpeed;
    isSprinting = true;
}

if(Input.GetButtonDown("Jump") && isSprinting)
{
    moveDirection.y = jumpForce;
}
```

- For my jump requirement, I decided to add a simple double jump. It was as simple as creating a boolean and a conditional, but implementing this made me aware of an issue with the CharacterController's isGrounded boolean as it would be rather inconsistent about being grounded or not, so I had to rework the ground check. To do this, I simply

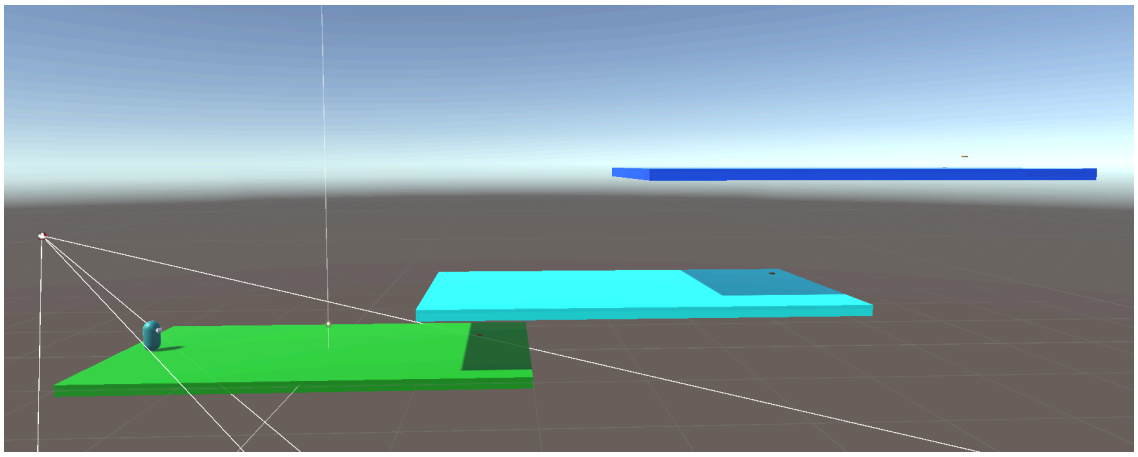
created a new boolean altGrounded and checked when moveDirection.y less than one and greater than negative one. But none of this seemed to work, so I just reused an idea from a previous project which involved adding an empty object at the bottom of the player to determine whether it's grounded or not. This also didn't want to work, so I decided to finally look online and just replace the grounded check with a raycast instead. Getting the grounded check to work took way longer than I'd like to admit. I didn't even mean to make my features the examples in the assignment.

```
1 reference
bool checkGrounded()
{
    return Physics.Raycast(transform.position, Vector3.down, 1.51f, 1 << LayerMask.NameToLayer("Ground"));
}
```

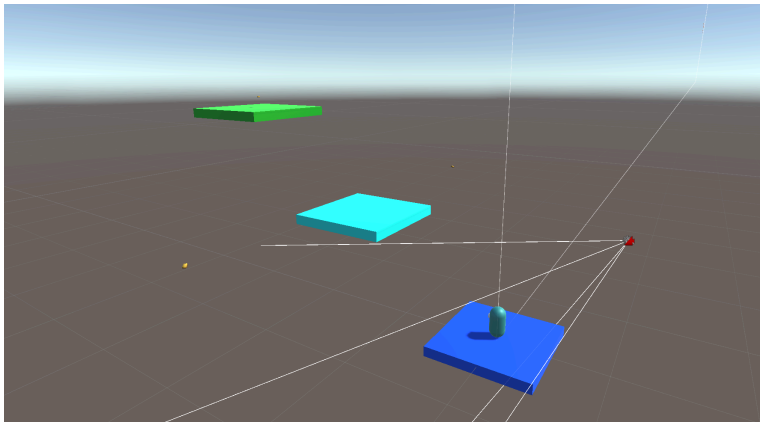
```
if(!grounded && canDJump && Input.GetButtonDown("Jump"))
{
    moveDirection.y = jumpForce;
    canDJump = false;
}
```

Levels

- For the first level, I wanted to make something super basic to introduce the mechanics. It's a very straightforward level with a set of three platforms and three gold bars, but the platforms are spaced in a way to showcase the single and double jump while being long enough to encourage the player to sprint to the bars.



- For the second level, I also made it rather simple as I was running out of time at this point. I made sure this one had the moving platform requirement, but other than that it was fairly simple.



Menu and UI

- For the menu, I attempted to steal some code from a previous project. I have a knack for doing that, but there really isn't a reason to reinvent the wheel. Luckily, cinemachine doesn't lock the mouse so I can reuse some of my UI buttons. Except, that didn't really want to work, so I reverted all of my changes and made the bare minimum for a menu.