Getting Started

Open unity hub and we are going to start a new project. Click 3D core and thus we will create a new project. Name it: Dodgeball

This creates our space to work on our game.

Notice:
The scene view and the game view. Its empty. The only thing in the scene view is a camera and a sunshine. We can deal with that later but for now it is important to know that it is the **main camera** for the game view and the **light source** of the game.

Main Camera- This is just the simple view of either the player or for the game.

Light Source- Gives directional light for your game

Lets Get Started

Right Click on the Hierarchy side bar and go to **3d object**. We want a **plane.** Also gather a **Cylinder** and a **sphere.** The cylinder will the "player" and the sphere will be the ball going in for the attack.

Feel free to move around the objects as these will appear in the middle of the plane. Since this is a 3D space, you have a green arrow going up, and a blue and right going in different directions. These represent the X,Y,Z plane in our space. If you just click an object, you can just move it.

You can select an object and hit Y to move, scale and rotate the object. To scale something use the little box of the item to something bigger. **Try it out with the plane**. Feel free to mess around a bit to understand how exactly it works.

The key thing we want to do is make the plane bigger, this is our court. Extend it to a decent size, then move the sphere and cylinder in a line.

Object Movement- Rigidbody

When you hit play at the top of the screen notice how nothing moves. Everything is floating and just there. Lets give these objects some life by adding rigidbody!

Click on the object you want to add first. Notice the inspector window to the right. Scroll down to **add component** and click it. In the search bar type **rigidbody**. Click it and then it will add the component to the item for you so now your objects act with the laws of gravity. Yay!

Asset Creation- Color

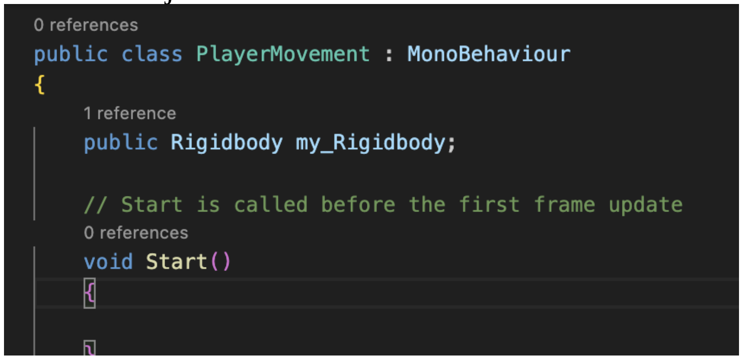
Right now we are looking at colorless items. Lets spice it up. Right click the assets area and go to **create, then find material**. This will create a new material for us. Looking at the new material, look at the inspector view then notice the line that says **albedo.** You can double click on the color and a color wheel with values should appear and then we can mess with that to make a color that you want. When you have a color you want, drag that asset to the object you want to have that color. Rinse and repeat 2 more times so all objects have color and look less boring.

Script Components- Making the game playable

Click on your cylinder. Go to add component and then type **PlayerMovement**. There should be an option to **add a new script**. Click that then hit create and add. There will now be a file in our assets. Double click and this will bring us to VS Code. There will be code there already. Methods that are there to help you implement your code.

Start is called once this is where the game is initialized. Update is called every frame per game. **Rename update to FixedUpdate()** this runs at 50 fps.

To move our player around we will add the following code:
public Rigidbody my_Ridgidbody; This will contain a bunch of premade methods to help us move our object around. Should look like this



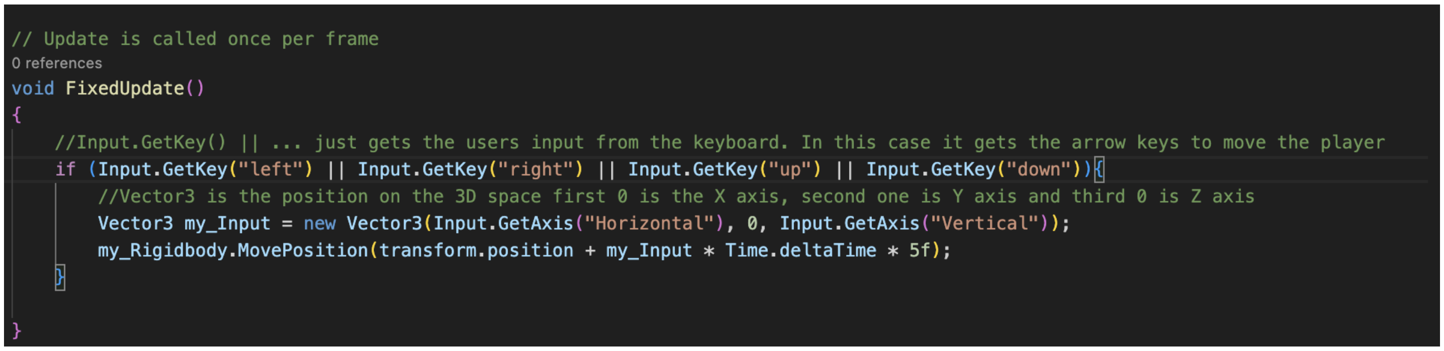
```
0 references
public class PlayerMovement : MonoBehaviour
{
    1 reference
    public Rigidbody my_Rigidbody;

    // Start is called before the first frame update
    0 references
    void Start()
    {

    }
```

To assign the property to an object, we can bring it to the game by going back to the unity editor and assign the rigidbody to the cylinder. If it doesn't show when you check out the ridgidbody component, make sure that VS code saved. Everything should update naturally. (it will be near player movement script My_RidgidBody if you did it right. Rember to click cylinder)

In FixedUpdate we can use player input to move around our player using the arrow keys. For now we will make it so the player can just move left to right. We will also add something for speed like so :



```
// Update is called once per frame
0 references
void FixedUpdate()
{
    //Input.GetKey() || ... just gets the users input from the keyboard. In this case it gets the arrow keys to move the player
    if (Input.GetKey("left") || Input.GetKey("right") || Input.GetKey("up") || Input.GetKey("down")){
        //Vector3 is the position on the 3D space first 0 is the X axis, second one is Y axis and third 0 is Z axis
        Vector3 my_Input = new Vector3(Input.GetAxis("Horizontal"), 0, Input.GetAxis("Vertical"));
        my_Rigidbody.MovePosition(transform.position + my_Input * Time.deltaTime * 5f);
    }

}
```

After implementing go ahead and move around the cylinder. We can now move it around how we please. Try it out

Next is the ball and its movement.
Same as before create a new script we will call it **ball**. When we go in we see that there are some different methods quite similar to that of the PlayerMovement we created.

In a similar fashion to the player movement we need to give the sphere rigidbody so that it can move. We will input the line of code that is
public RigidBody ball_RigidBody;

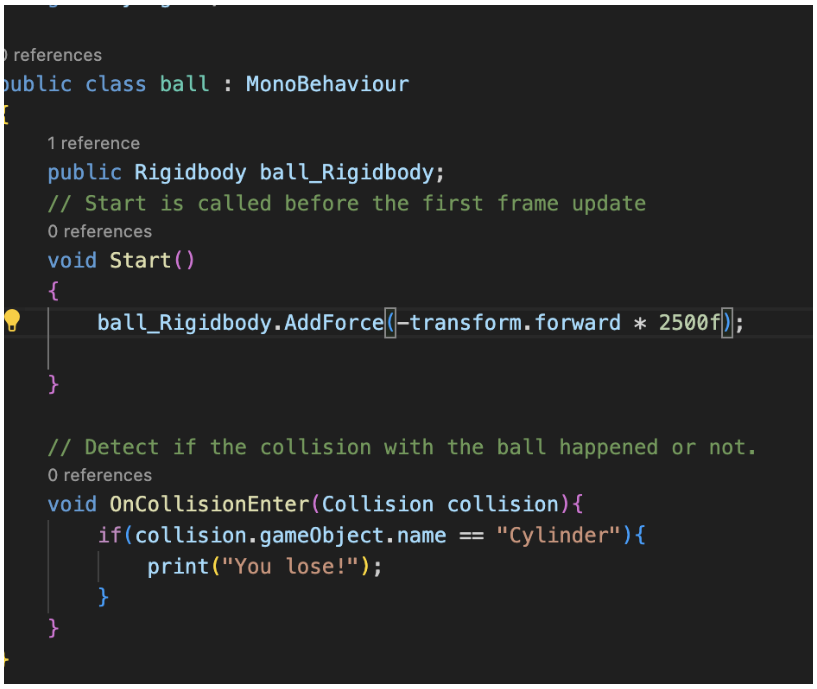
In start we will add the line
ball_Rigidbody.AddForce(-transform.foward *2500f);
Remember that this is called imediidately, so it makes sense to move it fowarward and we give it 2500f

We can get rid of Update() and instead we will put in

```
void OnCollisionEnter(Collision collision){
        if(collision.gameObject.name == "Cylinder"){
        print("You lose!");
        }
}
```

This will detect if there is a collision. The print statement happens in the left hand corner of the editor. It is small and will say what we tell it to say.

At the end it should look like this:



```
 references
ublic class ball : MonoBehaviour

    1 reference
    public Rigidbody ball_Rigidbody;
    // Start is called before the first frame update
    0 references
    void Start()
    {
        ball_Rigidbody.AddForce(-transform.forward * 2500f);

    }

    // Detect if the collision with the ball happened or not.
    0 references
    void OnCollisionEnter(Collision collision){
        if(collision.gameObject.name == "Cylinder"){
            print("You lose!");
        }
    }
```

Remember to save this code and make sure t update the rigid body movement for the sphere! It is important update .

If everything is working right then the game should be working!. You can mess around with the speed the ball is traveling, how fast your character moves.

Now that you have a simple game you can even duplicate objects like the ball to try and make the game more challenging too.

(Feel free to ask Ashley about anything)