Assignment 5 Process

Required elements to include:

- **You must outline and deconstruct the design of the game you wish to implement. This can be a deconstruction of an existing game, a modified version of an existing game, or a wholly new game.**
    - Make sure to keep your game confined to a single screen! This assignment timeline is short; reduce unnecessary complexity.
    - If you are making a wholly new game or a modified version of an existing game, spend time ideating and defining how your new or modified game works. This is both important for deconstructing the game and as a tool to communicate to your professor what your game is meant to be.
    - Begin thinking about your game in terms of pixels on a grid; pixel grid sketching paper is provided to help with this step. Next, consider what objects are part of your game. Are there multiple enemies, coins, balls, blocks, etc. on screen at once? Anything sufficiently complex, especially if duplicates exist, should be encapsulated in a class.
- **You must plan your 2D game using drawings/sketches.**
    - Begin thinking about your game in terms of shapes on a screen.
        - Many engines use units rather than pixels. You should not use pixel coordinates for this assignment.
    - Next, consider what objects are part of your game. Are there multiple enemies, coins, balls, blocks, etc. on screen at once?
        - Anything object's behaviour, especially if repeated, should be it's own script.

I'm considering making a simple platformer with free assets and 1-2 mechanics being move and jump.

Outline and Deconstruction

- I plan to create a simple 2D platformer with 2 mechanics. (move/jump)

Day 1 (April 10,2024)

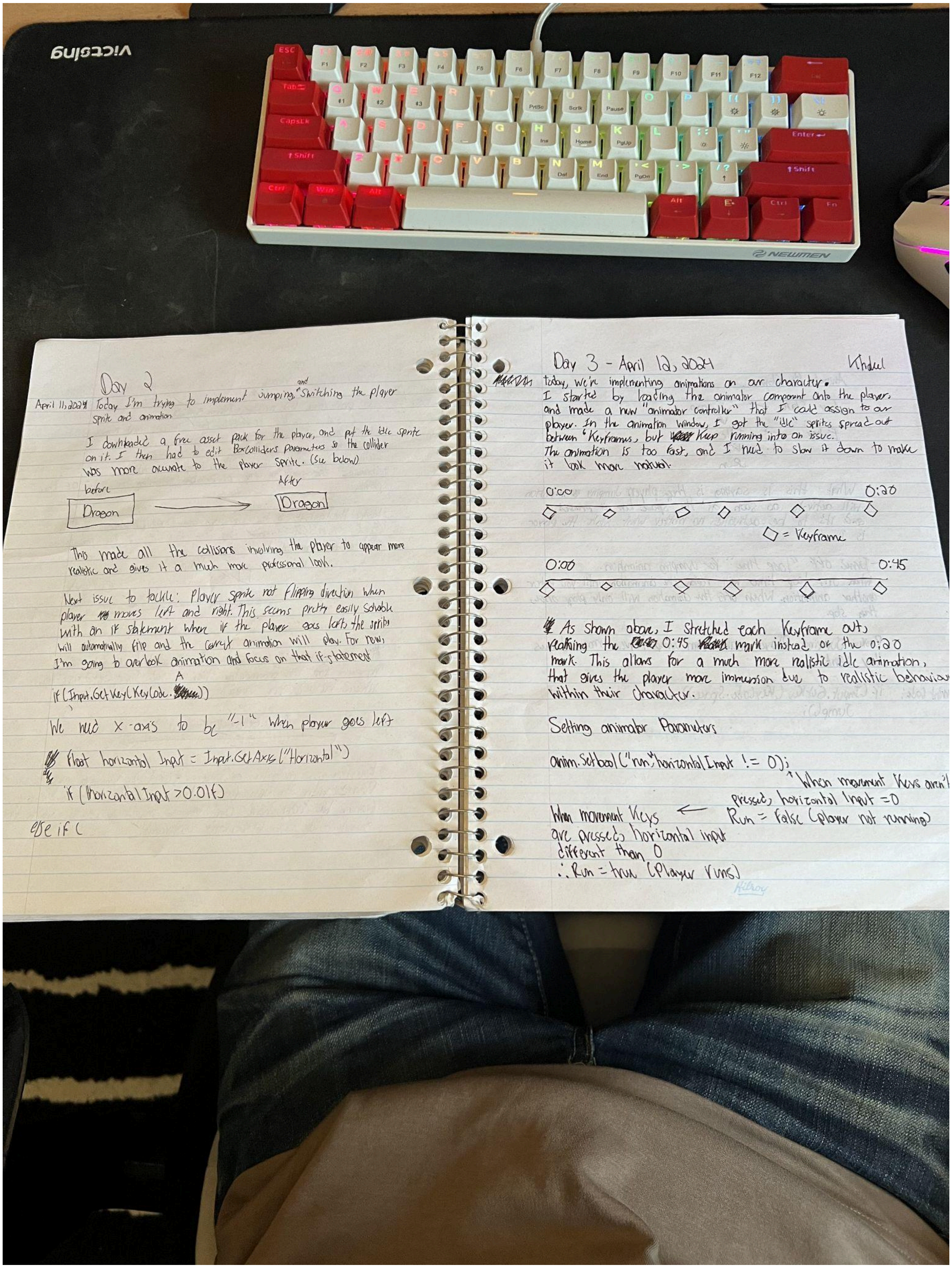
On day 1 I managed to lay some groundwork down. I found some free 2d platformer assets via the Unity Asset Store, imported them into my project.

Day 1 Progress

- Created the player, added sprites, boxcollider, rigidbody 2d and a PlayerMovement script that tracks input on every frame with the 'update' function.
- Created the ground and added boxcollider
- Figured out how to implement Left-Right Player Movement

Day 2 (April 11,2024)

Today I'm trying to implement a few things. Those being jumping, flipping the player sprite when they turn left or right, fixing the collider boundaries on the player, and maybe animation if I can get to it.



Day 2

April 11, 2024 Today I'm trying to implement jumping, and Switching the player sprite and animation

I downloaded a free asset pack for the player, and put the idle sprite on it. I then had to edit Boxcolliders parameters so the collider was more accurate to the player Sprite. (See below)

before

Dragon

After

Dragon

This made all the collisions involving the player to appear more realistic and gives it a much more professional look.

Next issue to tackle: Player sprite not flipping direction when player moves left and right. This seems pretty easily solvable with an if statement where if the player goes left, the sprite will automatically flip and the correct animation will play. For now, I'm going to overlook animation and focus on that if-statement

```
if(Input.GetKey(KeyCode.A))
```

We need x-axis to be "-1" when player goes left

```
float horizontal Input = Input.GetAxis("Horizontal")

  if (horizontalInput >0.01f)

else if (
```

Day 3 - April 12, 2024 Khdul

today, we're implementing animations on our character.
I started by loading the animator component onto the player, and made a new "animator controller" that I could assign to our player. In the animation window, I got the "idle" sprites spread out between 'Keyframes, but Keep running into an issue.
The animation is too fast, and I need to slow it down to make it look more natural.

0:00 0:20

◇ ◇ ◇ ◇ ◇ ◇

◇ = Keyframe

0:00 0:45

◇ ◇ ◇ ◇ ◇ ◇

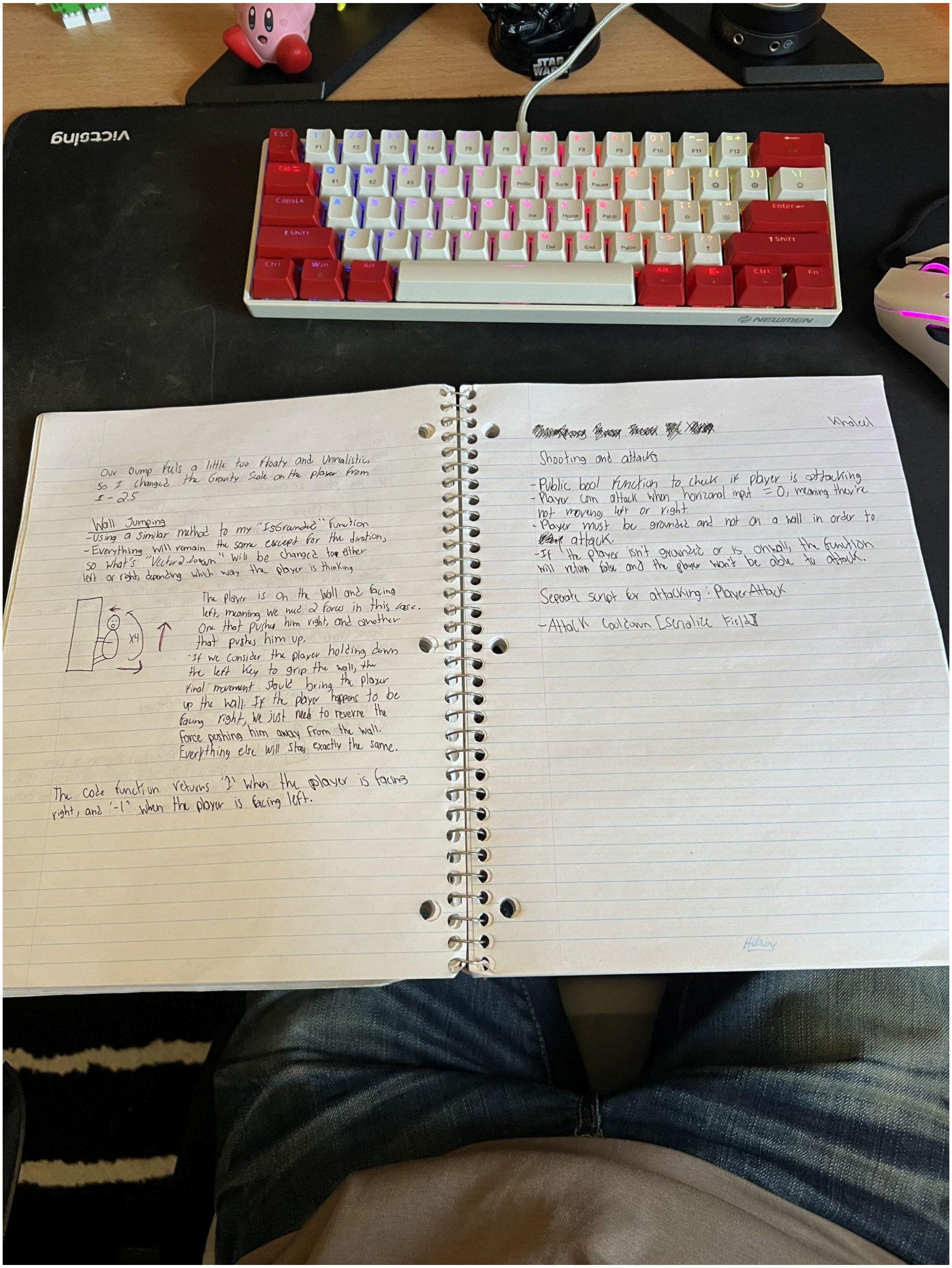
As shown above, I stretched each Keyframe out, reaching the 0:45 mark instead of the 0:20 mark. This allows for a much more realistic idle animation, that gives the player more immersion due to realistic behaviour within their character.

Setting animator Parameters

```
anim.SetBool("run", horizontalInput != 0);
```

When movement Keys aren't pressed, horizontal Input = 0
Run = False (player not running)

When movement Keys are pressed, horizontal input different than 0
∴ Run = true (Player runs)



Our Oump feels a little too floaty and Unrealistic so I changed the Gravity Scale on the player from 1 - 2.5

Wall Jumping
- Using a Similar method to my "IsGrounded" function
- Everything will remain the same except for the direction, so what's "Vector2.down" will be changed to either left or right, depending which way the player is thinking.

The player is on the wall and facing left, meaning we need 2 forces in this case. One that pushes him right, and another that pushes him up.

- If we consider the player holding down the left key to grip the wall, the final movement should bring the player up the wall. If the player happens to be facing right, we just need to reverse the force pushing him away from the wall. Everything else will stay exactly the same.

The code function returns '1' when the player is facing right, and '-1' when the player is facing left.

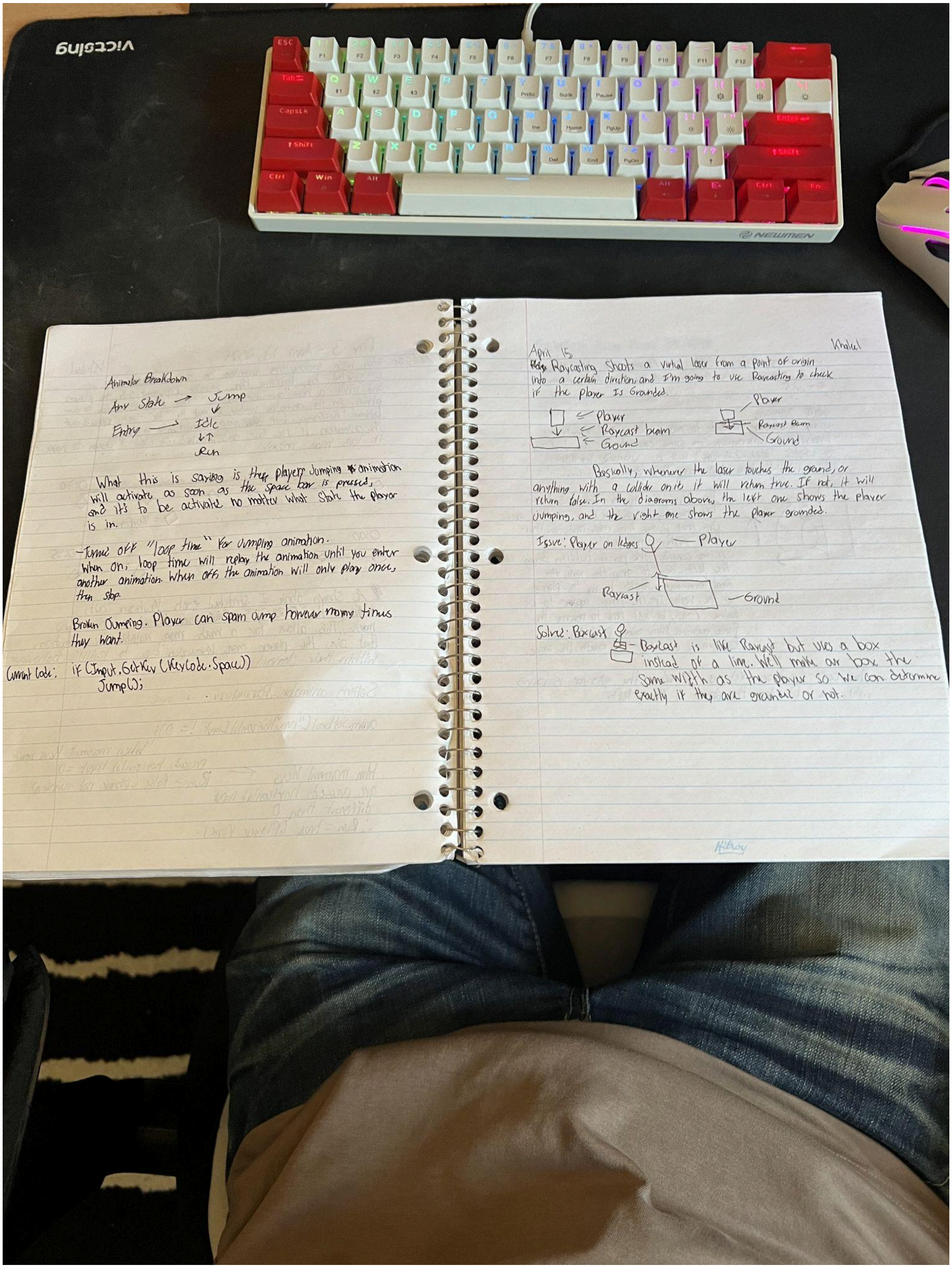
[redacted] Khaled

Shooting and attacks

- Public bool function to check if player is attacking
- Player can attack when horizonal input = 0, meaning they're not moving left or right.
- Player must be grounded and not on a wall in order to attack.
- If the player isn't grounded or is onWall, the function will return false and the player won't be able to attack.

Seperate script for attacking : PlayerAttack

- Attack Cooldown [SerializeField]



Animator Breakdown

Any State → Jump
↓
Entry → Idle
↓↑
Run

What this is saying is the players Jumping animation will activate as soon as the space bar is pressed, and it's to be activate no matter what State the player is in.

-Turned off "loop time" for Jumping animation. When on, loop time will replay the animation until you enter another animation. When off the animation will only play once, then stop.

Broken Jumping. Player can spam Jump however many times they want.

Current Code:

```
if (Input.GetKey(KeyCode.Space))
    Jump();
```

April 15 Khalil

~~Ray~~ Raycasting Shoots a virtual laser from a point of origin into a certain direction, and I'm going to use Raycasting to check if the player Is Grounded.

Basically, whenever the laser touches the ground, or anything with a collider on it it will return true. If not, it will return false. In the diagrams above, the left one shows the player jumping, and the right one shows the player grounded.

Issue: Player on ledges

Solve: Boxcast

- Boxcast is like Raycast but uses a box instead of a line. We'll make our box the same width as the player so we can determine exactly if they are grounded or not.