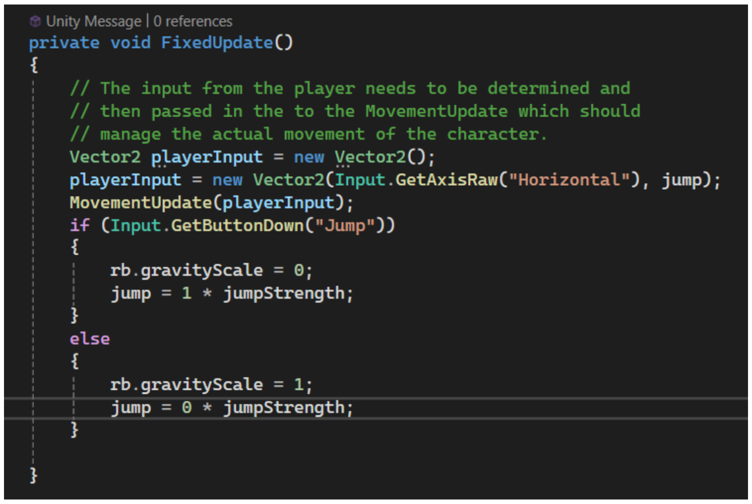
Journal #7 - Jump Mechanics

Task 1

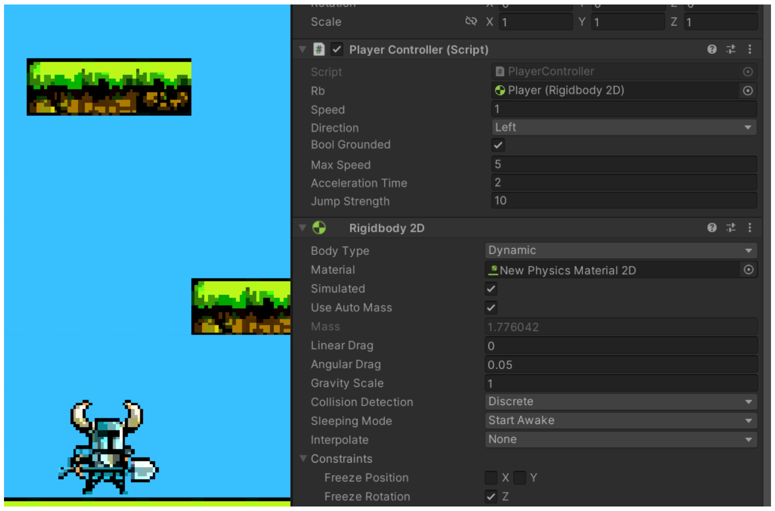
I've started by adding a float jump and public float jump strength. It broke my movement and currently, I cannot move, I've tried reverting my changes and it still breaks.



```
Unity Message | 0 references
private void FixedUpdate()
{
    // The input from the player needs to be determined and
    // then passed in the to the MovementUpdate which should
    // manage the actual movement of the character.
    Vector2 playerInput = new Vector2();
    playerInput = new Vector2(Input.GetAxisRaw("Horizontal"), jump);
    MovementUpdate(playerInput);
    if (Input.GetButtonDown("Jump"))
    {
        rb.gravityScale = 0;
        jump = 1 * jumpStrength;
    }
    else
    {
        rb.gravityScale = 1;
        jump = 0 * jumpStrength;
    }

}
```

[Unity - Scripting API: Input.GetButtonDown](#)

Lots of time to an hour completely confused. It was yet another problem that was in the unity, my acceleration time was set to 2 which made my character accelerate too slow that the friction would overpower any acceleration.



Scale X 1 Y 1 Z 1

Player Controller (Script)

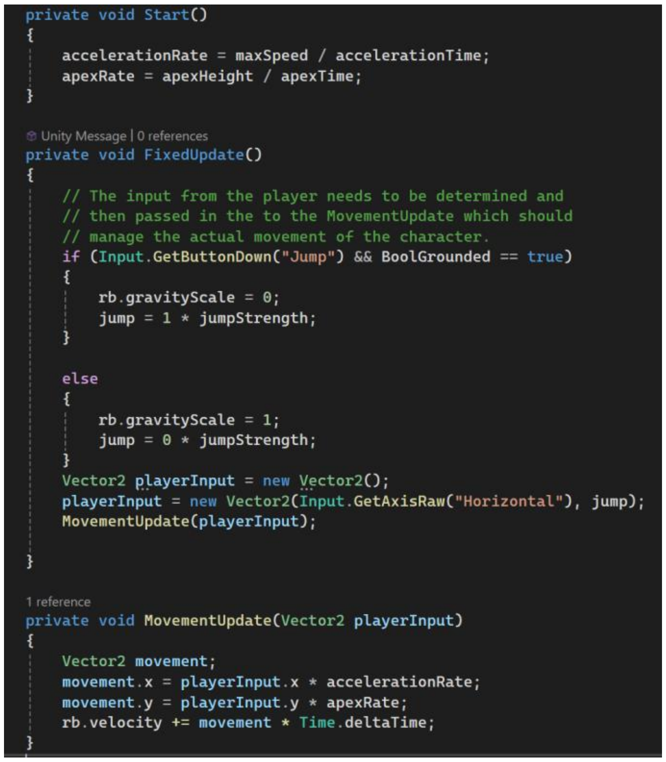
Script	PlayerController
Rb	Player (Rigidbody 2D)
Speed	1
Direction	Left
Bool Grounded	✓
Max Speed	5
Acceleration Time	2
Jump Strength	10

Rigidbody 2D

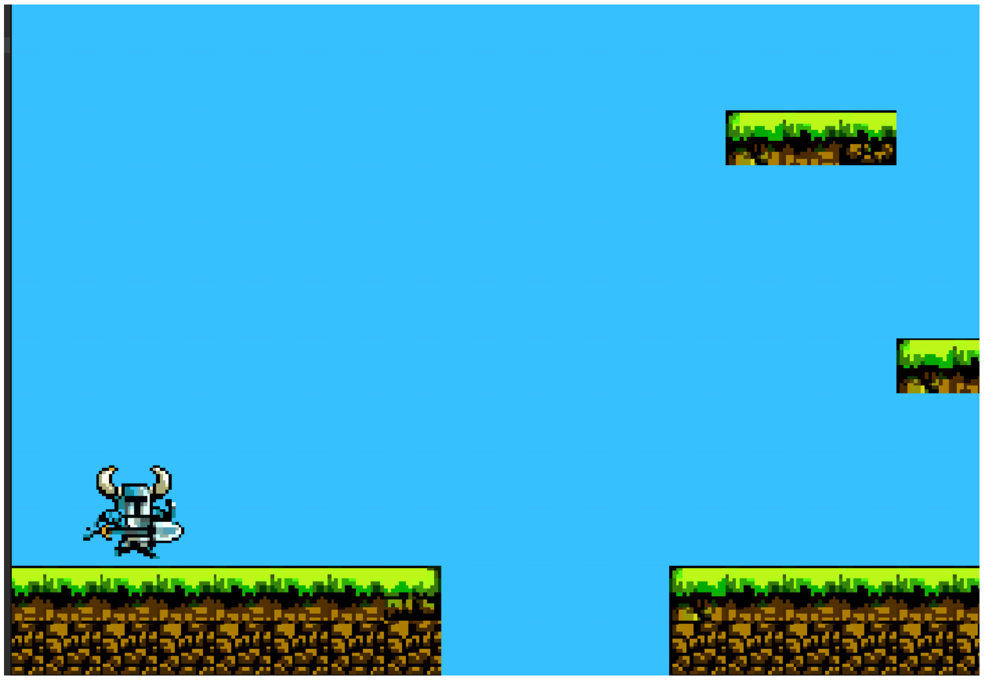
Body Type	Dynamic
Material	New Physics Material 2D
Simulated	✓
Use Auto Mass	✓
Mass	1.776042
Linear Drag	0
Angular Drag	0.05
Gravity Scale	1
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	None
Constraints	
Freeze Position	X Y
Freeze Rotation	✓ Z

I reorganized a similar format from acceleration to add apex time and height to create it.

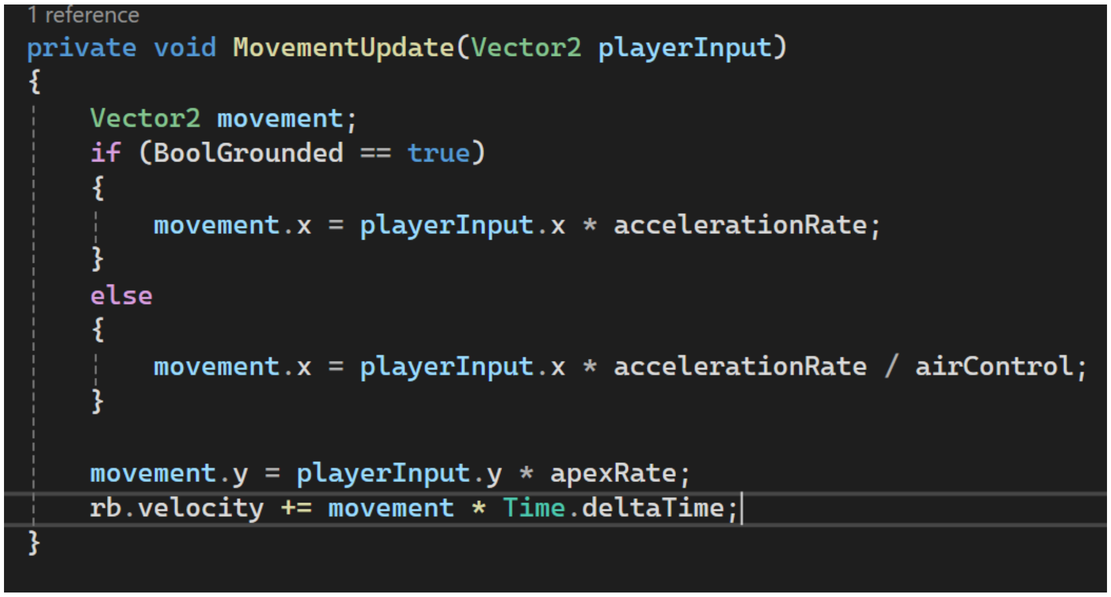
Though the speed of the character in the air would be significantly faster.



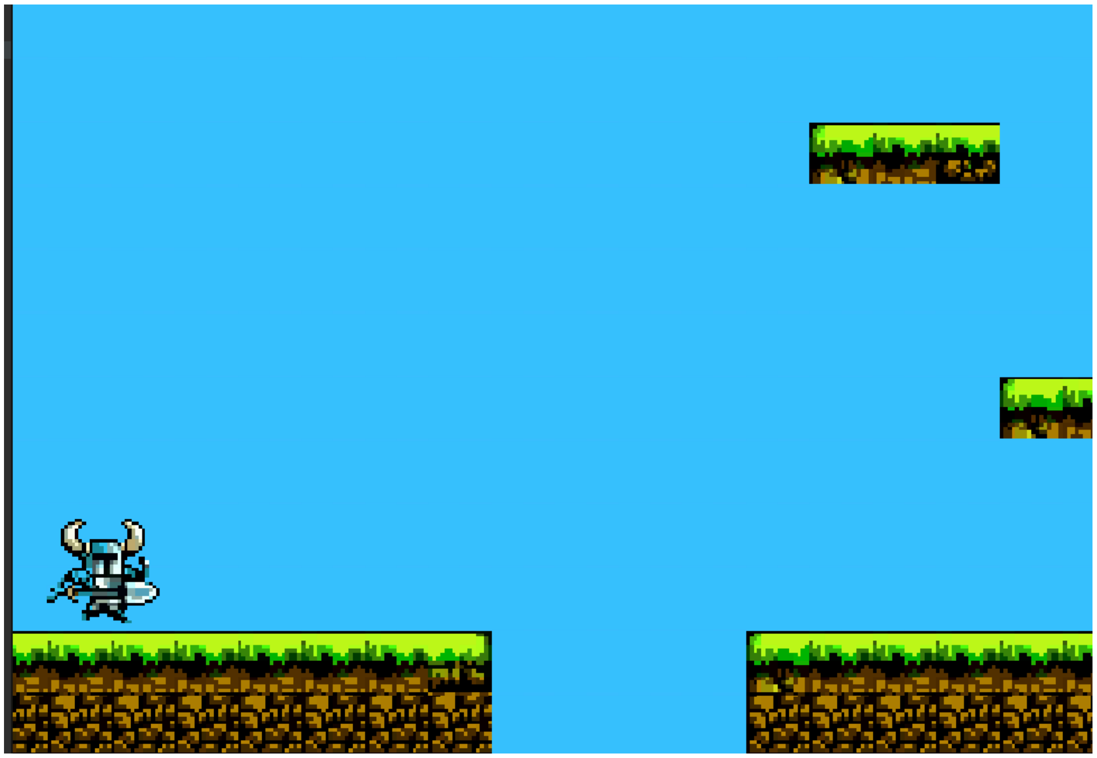
```
private void Start()
{
    accelerationRate = maxSpeed / accelerationTime;
    apexRate = apexHeight / apexTime;
}

Unity Message | 0 references
private void FixedUpdate()
{
    // The input from the player needs to be determined and
    // then passed in the to the MovementUpdate which should
    // manage the actual movement of the character.
    if (Input.GetButtonDown("Jump") && BoolGrounded == true)
    {
        rb.gravityScale = 0;
        jump = 1 * jumpStrength;
    }

    else
    {
        rb.gravityScale = 1;
        jump = 0 * jumpStrength;
    }
    Vector2 playerInput = new Vector2();
    playerInput = new Vector2(Input.GetAxisRaw("Horizontal"), jump);
    MovementUpdate(playerInput);

}

1 reference
private void MovementUpdate(Vector2 playerInput)
{
    Vector2 movement;
    movement.x = playerInput.x * accelerationRate;
    movement.y = playerInput.y * apexRate;
    rb.velocity += movement * Time.deltaTime;
}
```



Added air control float as the movement would be way too fast in the air.

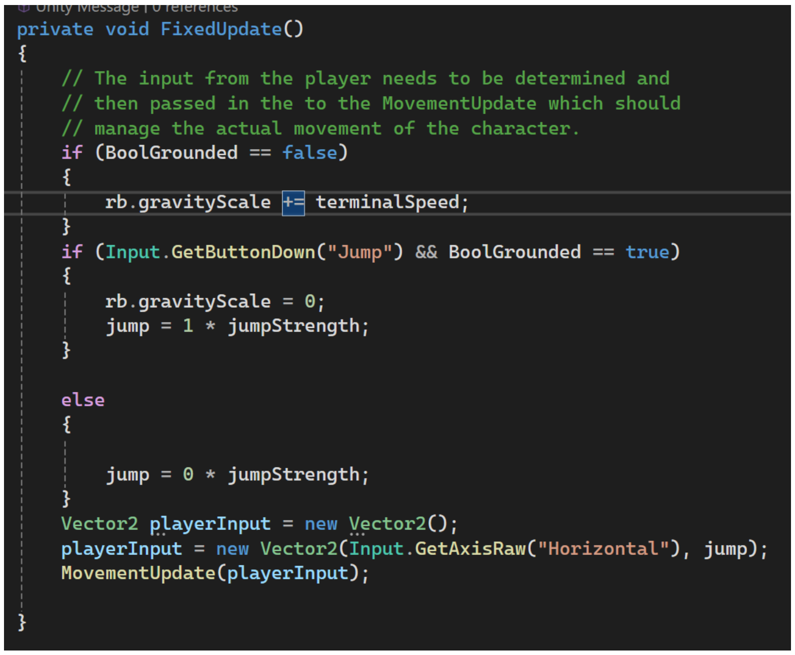


```
1 reference
private void MovementUpdate(Vector2 playerInput)
{
    Vector2 movement;
    if (BoolGrounded == true)
    {
        movement.x = playerInput.x * accelerationRate;
    }
    else
    {
        movement.x = playerInput.x * accelerationRate / airControl;
    }

    movement.y = playerInput.y * apexRate;
    rb.velocity += movement * Time.deltaTime;
}
```



Task 2

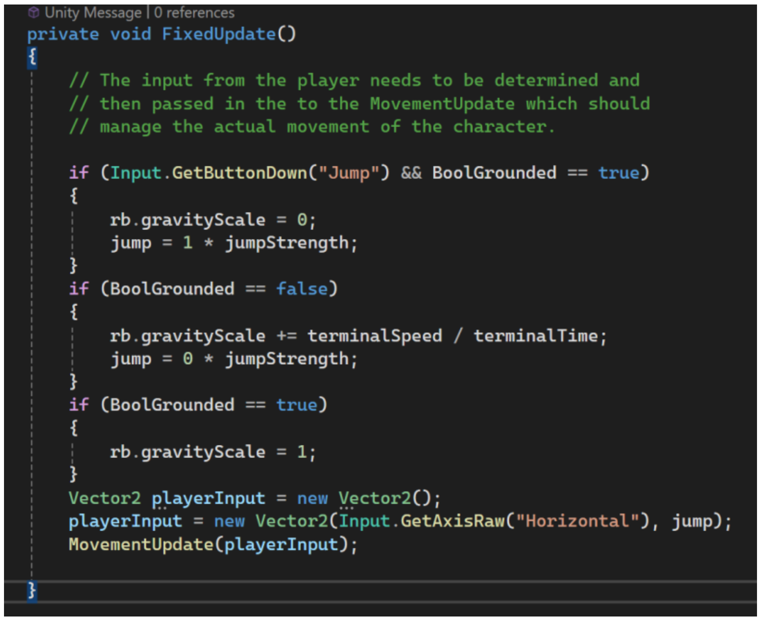
I tried adding values to gravityscale with terminal speed, but it would cause the character to stay with a high gravity value.



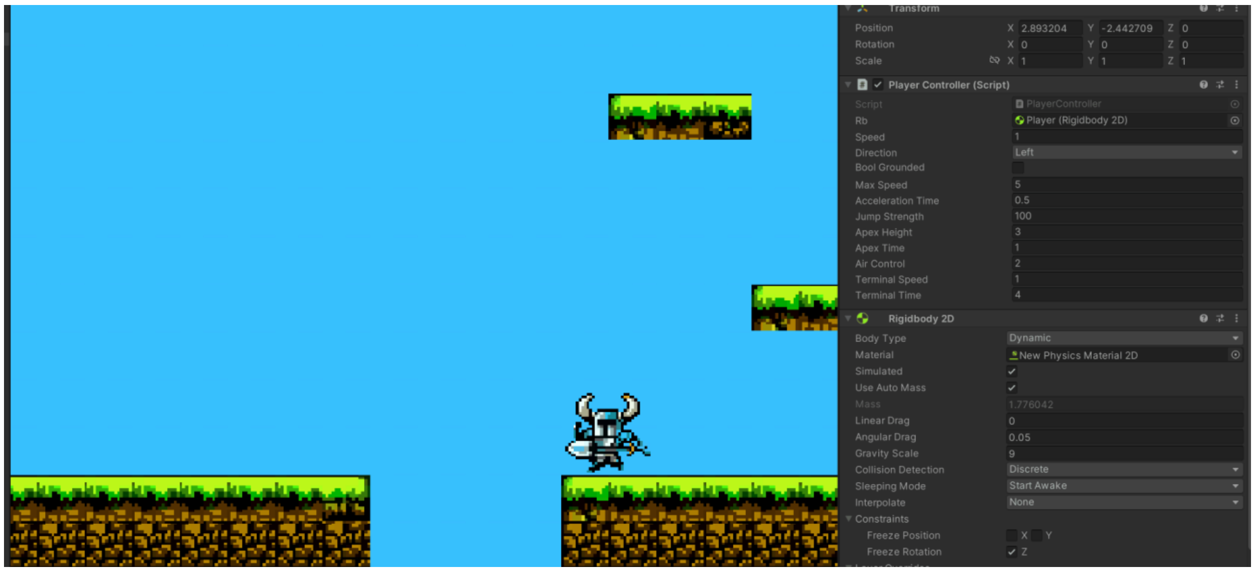
```
Unity Message | 0 references
private void FixedUpdate()
{
    // The input from the player needs to be determined and
    // then passed in the to the MovementUpdate which should
    // manage the actual movement of the character.
    if (BoolGrounded == false)
    {
        rb.gravityScale += terminalSpeed;
    }
    if (Input.GetButtonDown("Jump") && BoolGrounded == true)
    {
        rb.gravityScale = 0;
        jump = 1 * jumpStrength;
    }

    else
    {

        jump = 0 * jumpStrength;
    }
    Vector2 playerInput = new Vector2();
    playerInput = new Vector2(Input.GetAxisRaw("Horizontal"), jump);
    MovementUpdate(playerInput);

}
```

I tried reorganizing as there was an issue of the player going high when colliding with any object beside it, but there wasn't much of a solution as that had to do with collision.

My best solution would be to use a raycast instead of oncollision though I'm running out of time I've been on both tasks for hours.



```
Unity Message | 0 references
private void FixedUpdate()
{
    // The input from the player needs to be determined and
    // then passed in the to the MovementUpdate which should
    // manage the actual movement of the character.

    if (Input.GetButtonDown("Jump") && BoolGrounded == true)
    {
        rb.gravityScale = 0;
        jump = 1 * jumpStrength;
    }
    if (BoolGrounded == false)
    {
        rb.gravityScale += terminalSpeed / terminalTime;
        jump = 0 * jumpStrength;
    }
    if (BoolGrounded == true)
    {
        rb.gravityScale = 1;
    }
    Vector2 playerInput = new Vector2();
    playerInput = new Vector2(Input.GetAxisRaw("Horizontal"), jump);
    MovementUpdate(playerInput);

}
```

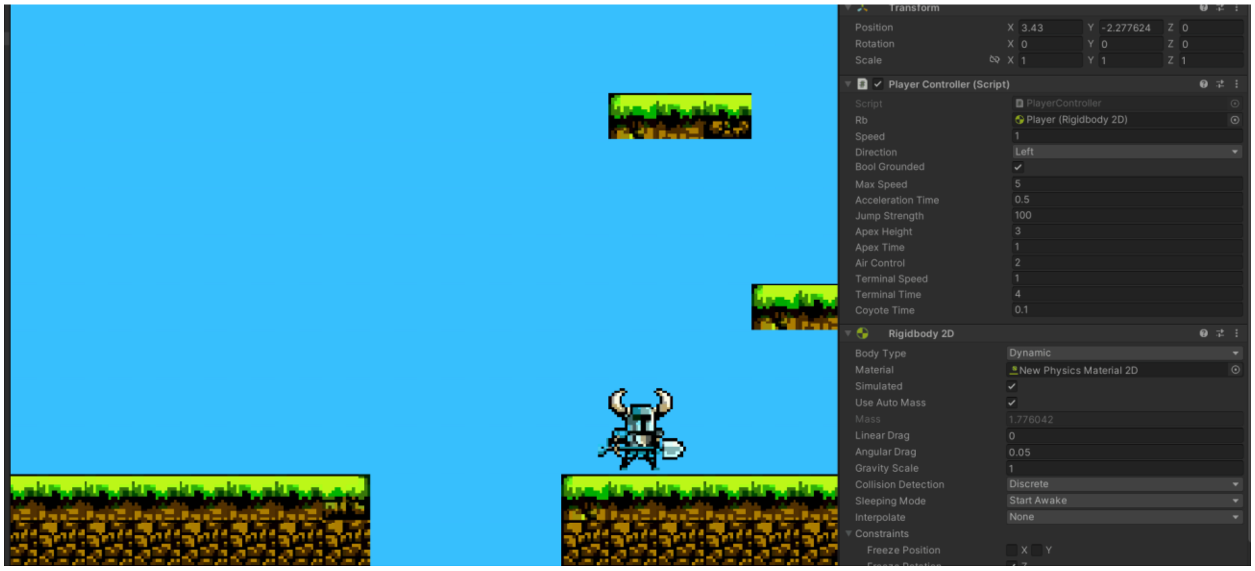


```
Transform
Position            X 2.893204   Y -2.442709   Z 0
Rotation            X 0          Y 0           Z 0
Scale               X 1          Y 1           Z 1

Player Controller (Script)
Script              PlayerController
Rb                  Player (Rigidbody 2D)
Speed               1
Direction           Left
Bool Grounded       ☐
Max Speed           5
Acceleration Time   0.5
Jump Strength       100
Apex Height         3
Apex Time           1
Air Control         2
Terminal Speed      1
Terminal Time       4

Rigidbody 2D
Body Type           Dynamic
Material            New Physics Material 2D
Simulated           ✓
Use Auto Mass       ✓
Mass                1.776042
Linear Drag         0
Angular Drag        0.05
Gravity Scale       9
Collision Detection Discrete
Sleeping Mode       Start Awake
Interpolate         None
Constraints
  Freeze Position   ☐ X ☐ Y
  Freeze Rotation   ✓ Z
```

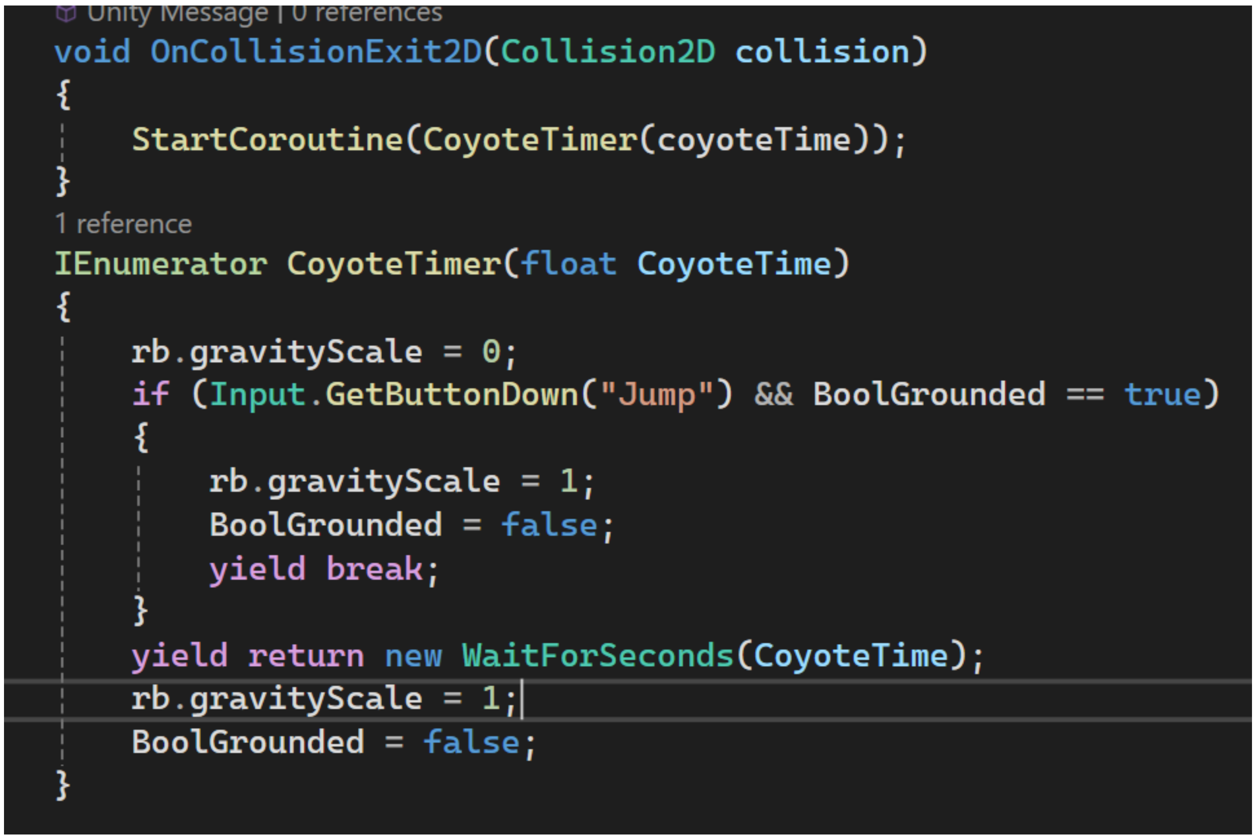
Task 3

For CoyoteTime I will delay the time it takes for the book grounded turns to false and set gravity to 0 temporarily.

This unfortunately would not give the results I was looking for.



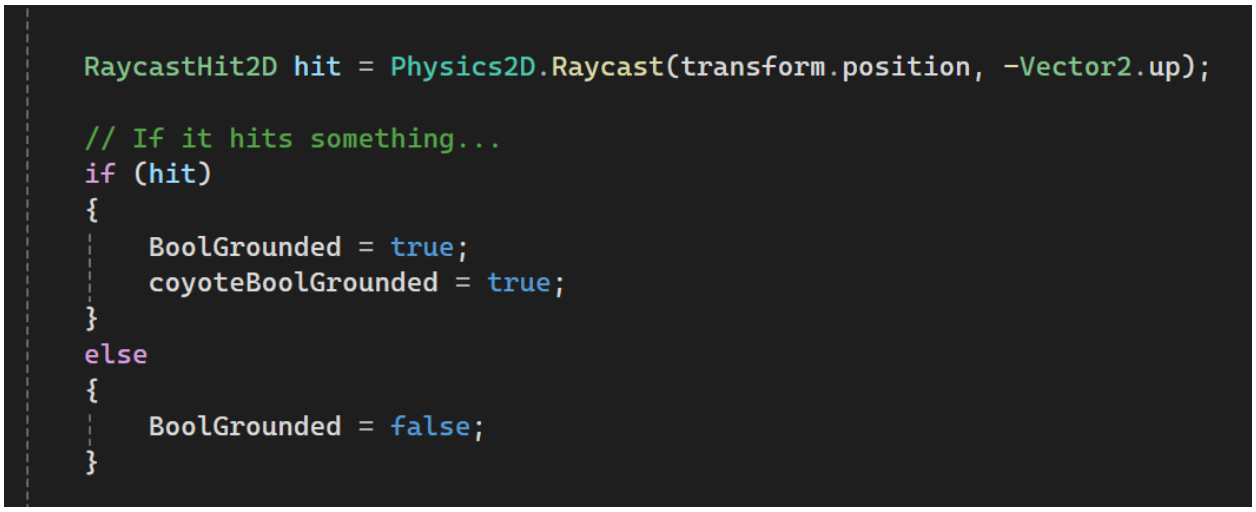
```
Transform
Position            X 3.43       Y -2.277624   Z 0
Rotation            X 0          Y 0           Z 0
Scale               X 1          Y 1           Z 1

Player Controller (Script)
Script              PlayerController
Rb                  Player (Rigidbody 2D)
Speed               1
Direction           Left
Bool Grounded       ✓
Max Speed           5
Acceleration Time   0.5
Jump Strength       100
Apex Height         3
Apex Time           1
Air Control         2
Terminal Speed      1
Terminal Time       4
Coyote Time         0.1

Rigidbody 2D
Body Type           Dynamic
Material            New Physics Material 2D
Simulated           ✓
Use Auto Mass       ✓
Mass                1.776042
Linear Drag         0
Angular Drag        0.05
Gravity Scale       1
Collision Detection Discrete
Sleeping Mode       Start Awake
Interpolate         None
Constraints
  Freeze Position   ☐ X ☐ Y
```


```
Unity Message | 0 references
void OnCollisionExit2D(Collision2D collision)
{
    StartCoroutine(CoyoteTimer(coyoteTime));
}
1 reference
IEnumerator CoyoteTimer(float CoyoteTime)
{
    rb.gravityScale = 0;
    if (Input.GetButtonDown("Jump") && BoolGrounded == true)
    {
        rb.gravityScale = 1;
        BoolGrounded = false;
        yield break;
    }
    yield return new WaitForSeconds(CoyoteTime);
    rb.gravityScale = 1;
    BoolGrounded = false;
}
```

https://discussions.unity.com/t/how-to-use-the-new-stopcoroutine-ienumerator/539750/10

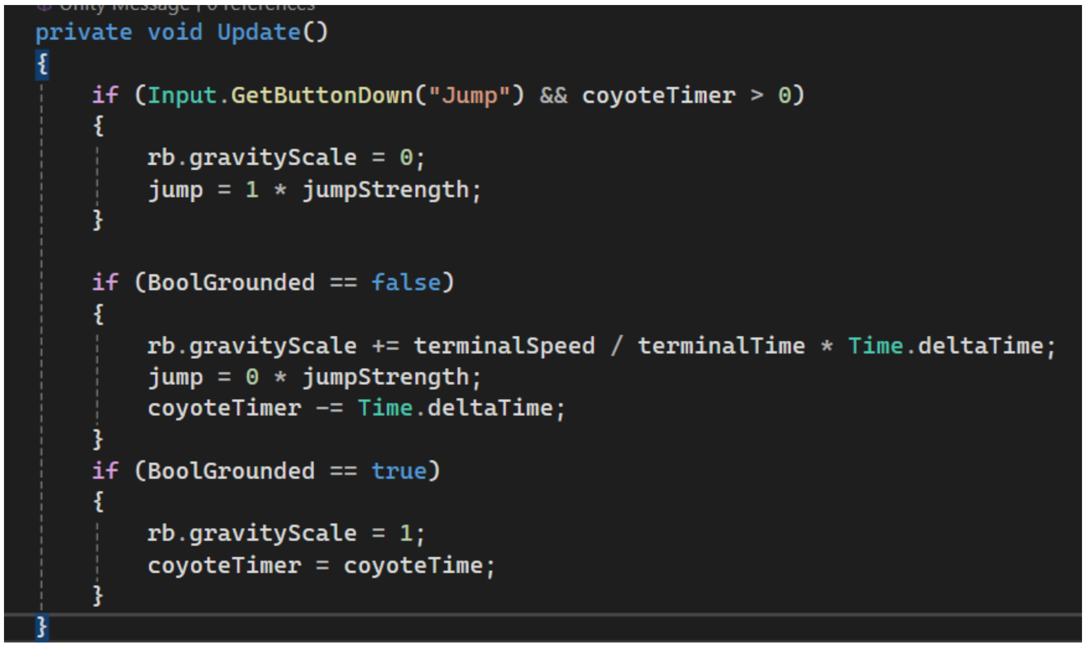
What is the best way to delay a function? - Unity Engine - Unity Discussions

I attempted to use Raycast but the character would fly as it wouldn't detect when it wasn't hitting the ground.


```
RaycastHit2D hit = Physics2D.Raycast(transform.position, -Vector2.up);

// If it hits something...
if (hit)
{
    BoolGrounded = true;
    coyoteBoolGrounded = true;
}
else
{
    BoolGrounded = false;
}
```

Unity - Scripting API: Physics2D.Raycast

I tried this formula with a float it would jump off the ground, but it wouldn't make the avatar jump during the timer.

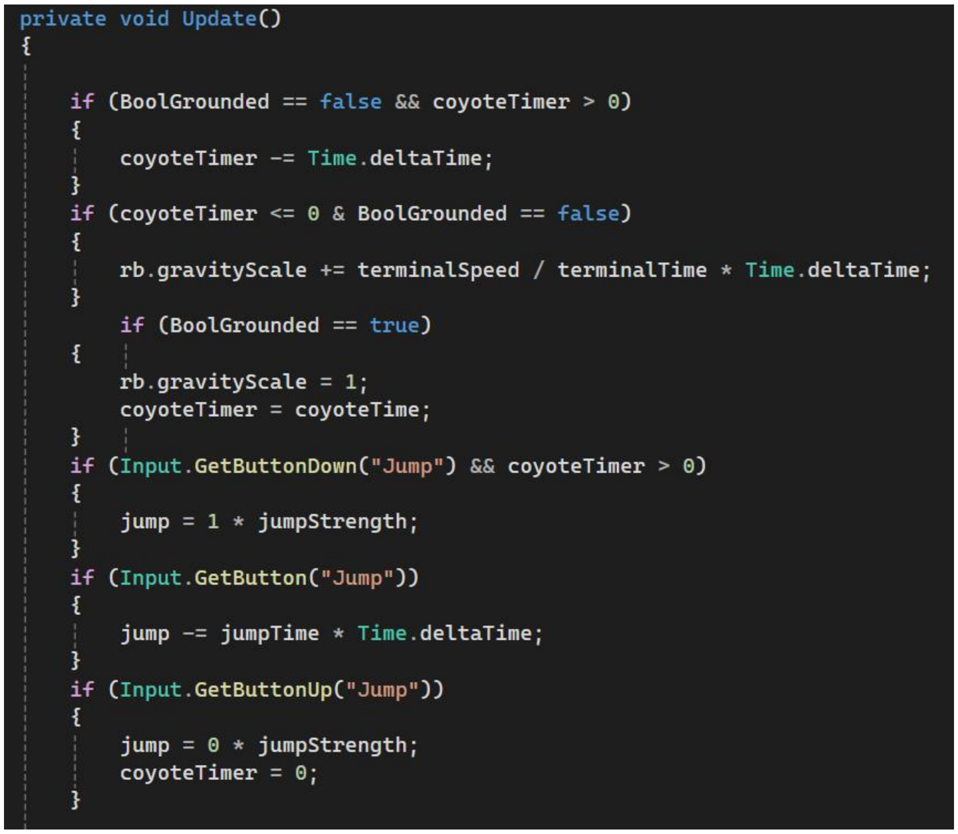


```
private void Update()
{
    if (Input.GetButtonDown("Jump") && coyoteTimer > 0)
    {
        rb.gravityScale = 0;
        jump = 1 * jumpStrength;
    }

    if (BoolGrounded == false)
    {
        rb.gravityScale += terminalSpeed / terminalTime * Time.deltaTime;
        jump = 0 * jumpStrength;
        coyoteTimer -= Time.deltaTime;
    }
    if (BoolGrounded == true)
    {
        rb.gravityScale = 1;
        coyoteTimer = coyoteTime;
    }
}
```

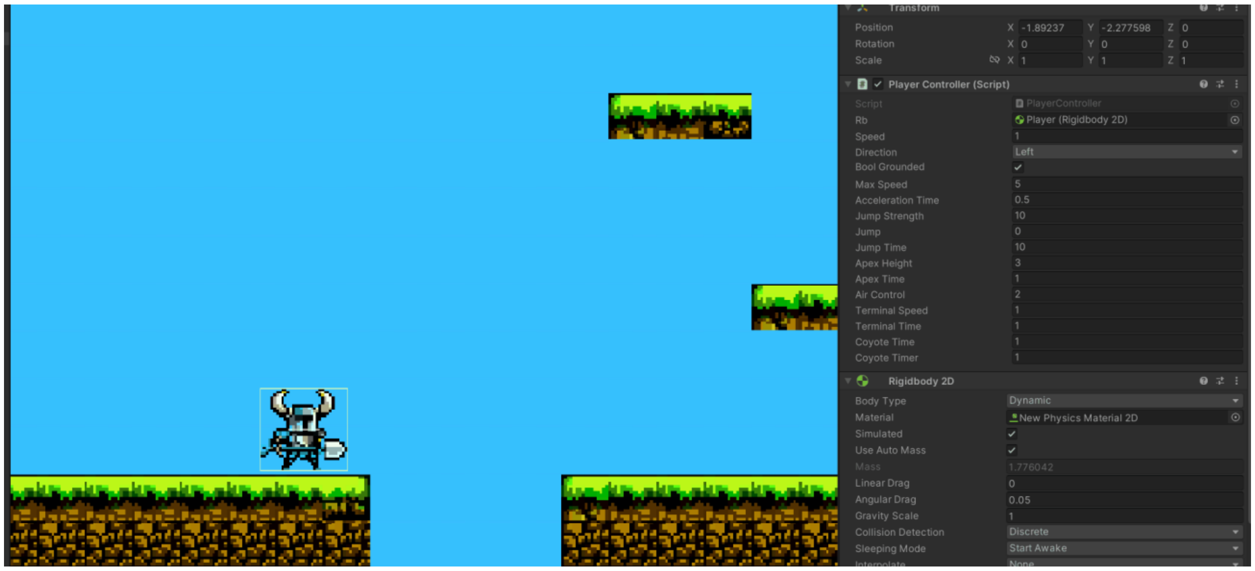
This took me multiple hours around 2, probably the most out of everything I've done. I ran into a lot of issues with double jumping and the jump going insanely high as jump strength would stay at 1 because coyote time did not reset to 0 when the jump button was pressed. For double jumping, I've been saved thanks to getbuttonup as it would reset the coyote time correctly before the next button press.

As for the jump strength going insane when jumping in mid air, instead of using boodgrounded to reset jump strength I went my way first to use GetButtonUp to set jump strength to 0 for temporary fix, this would mean the player would only fly infinitely as long as the player presses the key. I concluded by adding GetButton to reduce the jump overtime with JumpTime amplifying the time it takes to reach 0. Ironically ended up with better dynamic jumping, which is not something I was expecting to achieve.

I was left with movement I was probably the most I've ever been with my programming even though this took me a massive amount of time.



```
private void Update()
{

    if (BoolGrounded == false && coyoteTimer > 0)
    {
        coyoteTimer -= Time.deltaTime;
    }
    if (coyoteTimer <= 0 & BoolGrounded == false)
    {
        rb.gravityScale += terminalSpeed / terminalTime * Time.deltaTime;
    }
        if (BoolGrounded == true)
    {
        rb.gravityScale = 1;
        coyoteTimer = coyoteTime;
    }
    if (Input.GetButtonDown("Jump") && coyoteTimer > 0)
    {
        jump = 1 * jumpStrength;
    }
    if (Input.GetButton("Jump"))
    {
        jump -= jumpTime * Time.deltaTime;
    }
    if (Input.GetButtonUp("Jump"))
    {
        jump = 0 * jumpStrength;
        coyoteTimer = 0;
    }
```



Transform

Position	X -1.89237	Y -2.277598	Z 0
Rotation	X 0	Y 0	Z 0
Scale	X 1	Y 1	Z 1

Player Controller (Script)

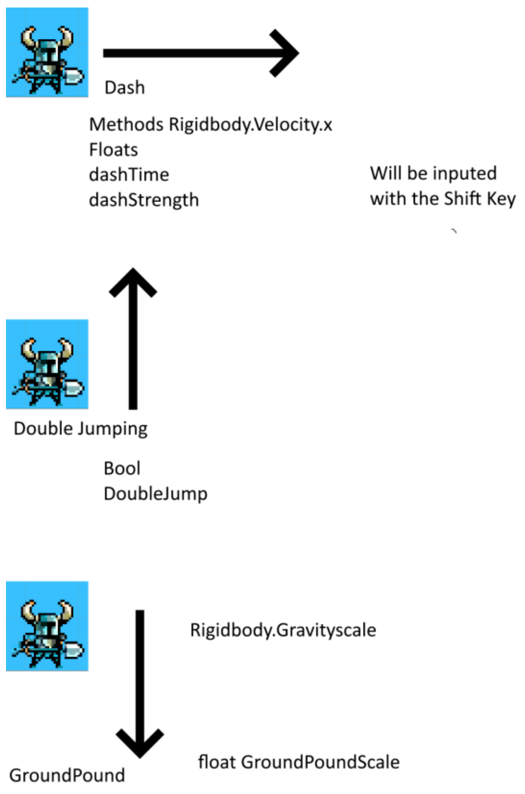
Script	PlayerController
Rb	Player (Rigidbody 2D)
Speed	1
Direction	Left
Bool Grounded	✓
Max Speed	5
Acceleration Time	0.5
Jump Strength	10
Jump	0
Jump Time	10
Apex Height	3
Apex Time	1
Air Control	2
Terminal Speed	1
Terminal Time	1
Coyote Time	1
Coyote Timer	1

Rigidbody 2D

Body Type	Dynamic
Material	New Physics Material 2D
Simulated	✓
Use Auto Mass	✓
Mass	1.776042
Linear Drag	0
Angular Drag	0.05
Gravity Scale	1
Collision Detection	Discrete
Sleeping Mode	Start Awake
Interpolate	None

https://www.youtube.com/watch?v=RFix_Kg2Di0

Part 3 Proposal



Dash will be measured by dashStrength and dashTime

If (Shift Key)

Rigidbody.Velocity.x += dashStrength / dashTime;

Double jumping will be be allowed in an if statement.

If (doubleJump == False && Grounded == false)

JumpStrength = 1;

GroundPound will amplify gravity to go down faster

If(Input.GetKeyDown(“CTRL”)

Rigidbody.gravityscale = GroundPoundScale;