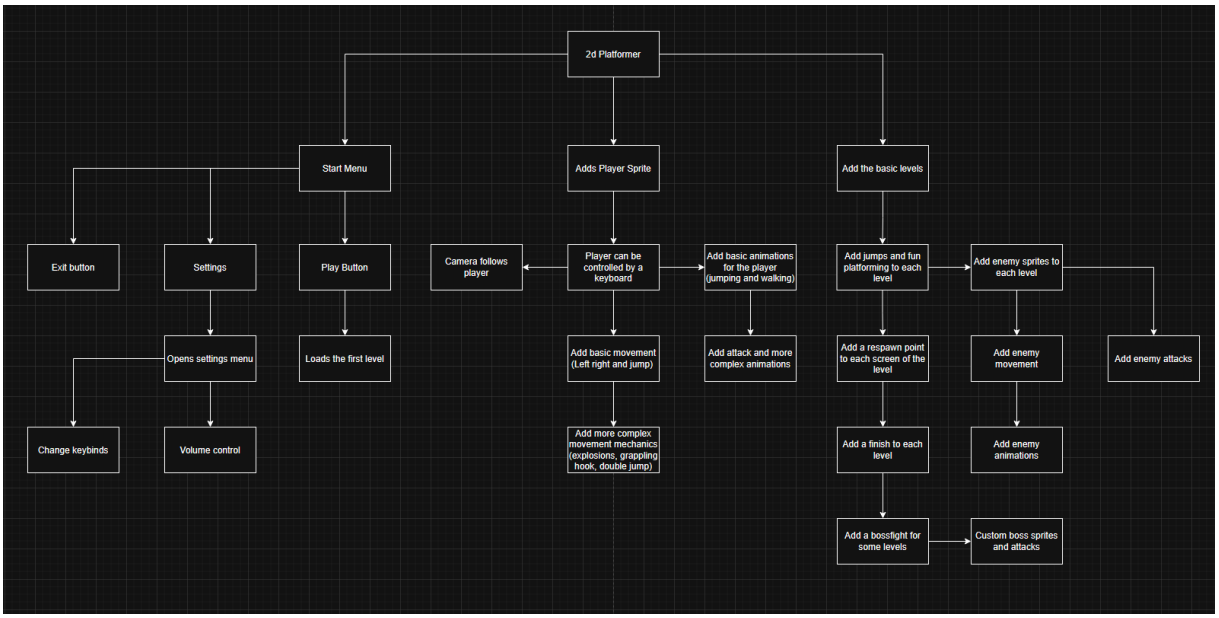
2d Platformer Design:

Structure Diagram:



This structure diagram is split into 3 parts,

Stages of Development Plan:
Stage 1:

Before I go in depth for any feature of the project, like mechanics, level design and user interface, I should make a simple version of the game. This will include simple movement and a few jumps to test everything works properly.

The simple version will have the following features:

The ability to move left, right and jump, make sure that the movement is enjoyable and responsive from the start.

Add some objects to the scene to see how they interact as they should, e.g being able to stand on top/ not walk through.

Add a respawn platform for the player to go to when they die/fall off the map.

Test No.	Description	Type of test	Test data	Expected Result	Actual Result
1	Can the player move?		Input movement button	Player moves	Yes
2	Does the camera follow the player?		Input movement button	Camera moves with the player	Yes

3	Can the player interact with objects?		Place player on top of a platform	Player stands on top	yes
4	Does the player respawn?		Move the player off the map	Player is placed back on its respawn point	Yes
5	Can the player jump?		Input the jump button	Player jumps	Not working

Stage 2:

To make the game more complex, I will have to add more features to the game, this includes more movement mechanics and. The first of these will be a double jump, which hopefully is the simplest, and a rope that can be used to swing. I will also add a start menu and a pause menu during the game.

The 2nd version will have the following features:

A menu that appears when the game starts that lets you start the game or enter settings.

A pause menu that will let you return to the menu during the game.

A double jump that allows the player to jump again while already in the air, potentially playing an animation, letting the player jump further.

A rope/grappling hook that can be thrown/shot to attach to the ceiling, to let the player swing across a gap that would otherwise be too large to travel across

Test No.	Description	Type of test	Test data	Expected Result	Actual Result
1	Does the game start?		Press start	Scene, and sprites appear	
2	Does a pause menu appear?		Press the pause button	Pause menu appears	
3	Can the player double jump?		Press space twice	Player can jump	

4	Can the rope attach to the ceiling		Touch the two objects	Rope attaches to the object	

5					

Stage 3:

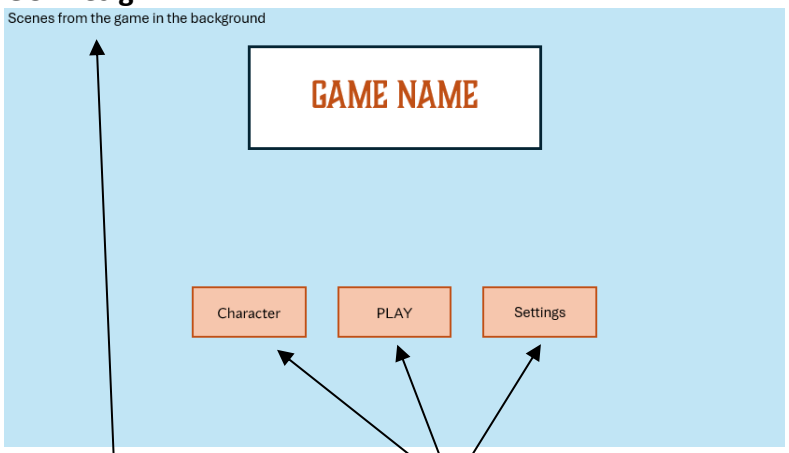
After adding more complex mechanics to the game, I would like to start making it more playable and fun. This means adding things like enemies and fighting, plus changing features to be higher quality.

The 3rd version will have the following features:

Enemies that can move towards or fire objects at the player. The player should also be able to take damage from these attacks or by touching an enemy. But, they can fight back using the tools they have been given (bomb/grappling hook) to damage the enemies.

Test No.	Description	Type of test	Test data	Expected Result	Actual Result
1	Do the enemies try to attack the player?		Move player towards enemy.	Enemies aggress towards the player, once they get close enough	
2	Does an animation play when the enemies move?		Move the enemy	Movement animation plays	
3	When hit by an attack, does the player take damage/get reset?		Player touches component that damages them	Teleported to the beginning of the screen	
4	Do the enemies take damage from the player's attacks?		Enemy touches player attack.	Enemy takes damage/ disappears.	
5	Is the player able to attack?		Press attack key	Player attacks	
6	Does an animation play while attacking?		Press attack key	Attack animation plays	

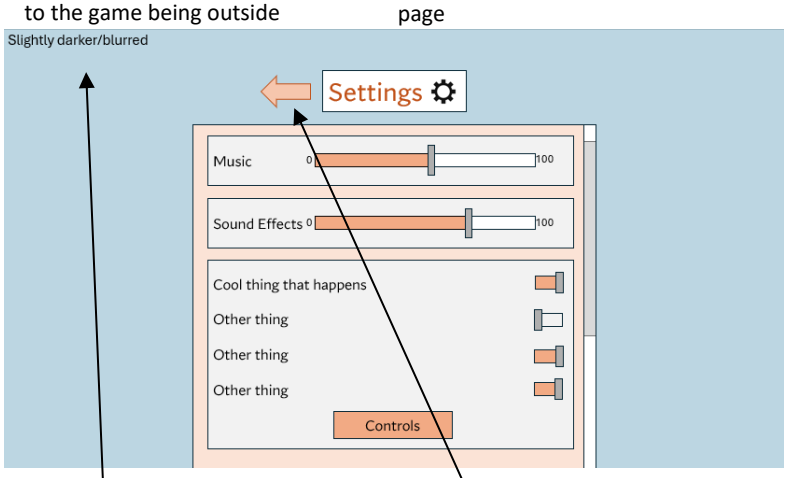
GUI Design:



Background contains scenes from the game, so the placeholder colour is blue, as it is abundant due to the game being outside

Each button takes the user to its respective page

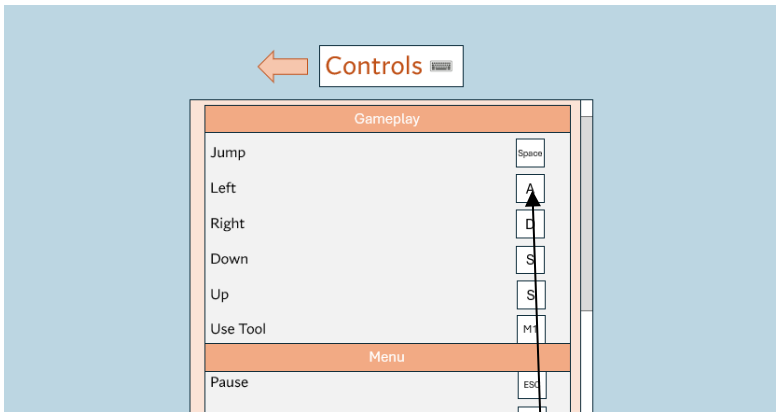
The game will open to this page which contains the title of the game, and multiple buttons below. I have done it in this way as it makes the name more prominent, letting the player know what game it is instantly, as opposed to something else being more noticeable, making the game name less obvious. The buttons are just below the title, in an easy-to-read font that stands out compared to the orange background. I have decided to use the colour orange as it stands out against the blue background, which is blue as it will contain backgrounds from the game, which is mainly set outside, with a blue sky.



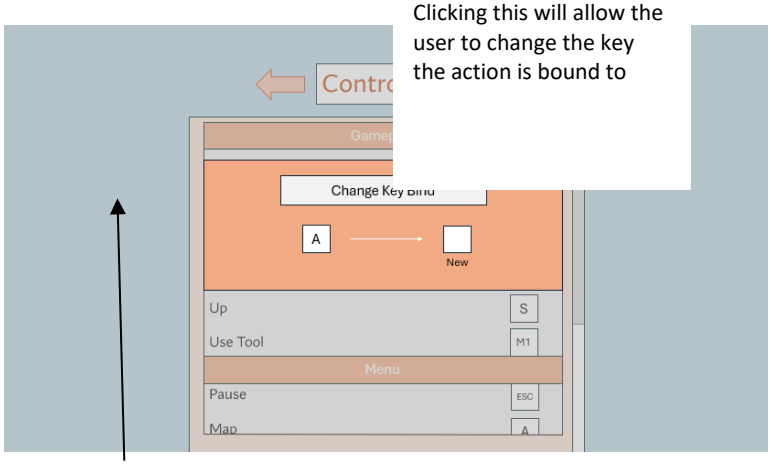
The settings background is slightly darker compared to the main menu, the background will also become blurred; I think this helps to indicate that it is a sub-page and take the attention away from the background and focus on the main point of the page

Back arrow indicates that it will bring the user back to the main menu. This is used instead of 'back' as it became too cluttered with words and I preferred it to be more simple

When the settings button of the previous layout is pressed, this page will appear. There is a title at the top of the page telling the user it is the settings, plus an icon, just in case the user isn't able to read English. This tells them they are in the settings. Below the title, there are different sliders and buttons you can press. For the sliders, there are contrasting colours: orange and white, which is a good visual representation to show how far along it is. Similarly, the buttons have that feature, where it is obvious if they are on or not. These are helpful as they make navigating settings as easy as possible.



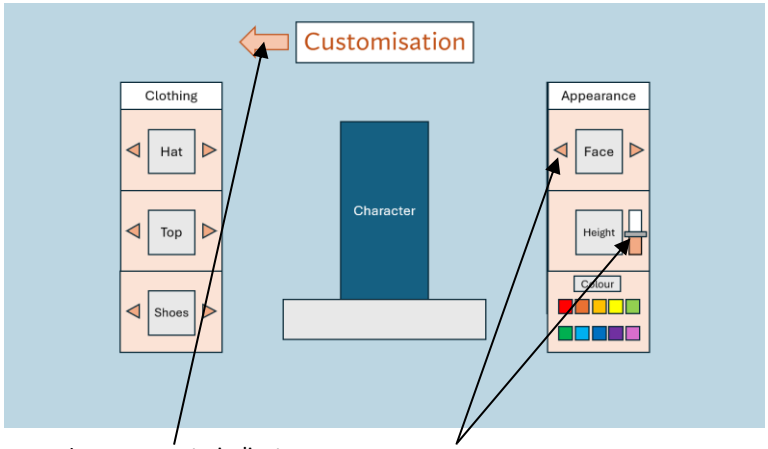
After the Controls button has been pressed, this page appears, it shows the user what each key does in game, this is helpful in case they do not know. The background is a different colour to the key binds' background, this makes them stand out more and easier to see. I have also used a keyboard icon next to the title to help indicate the usage of the page, like the settings icon. The user can also change the keys by clicking on them. There is also a



Clicking this will allow the user to change the key the action is bound to

The background is again darker, to indicate what page the game is focused on.

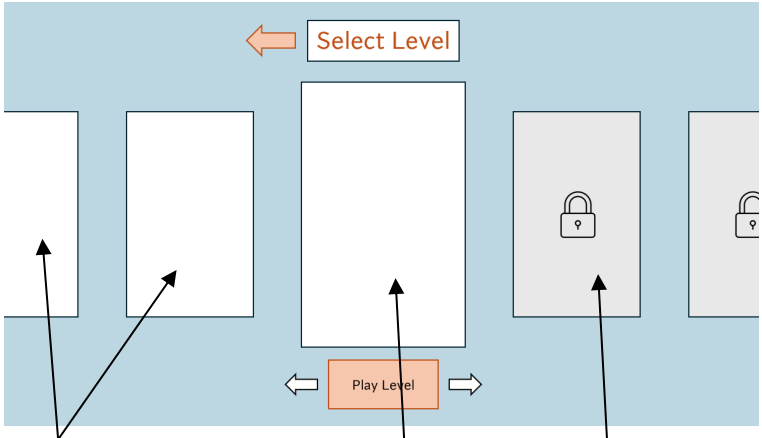
When the user tries to change their key binds, this window will appear, where they can press any button to change the key. The background becomes darker as it takes attention away and places it on the new window. It is orange as it matches with the colour scheme of the other components of the GUI.



Large arrow to indicate that it will take the user back to the main menu.

Allows the user to cycle along the different options/ change the appearance

The Customisation page is clearly labelled at the top, I would like to add an icon, like an item of clothing or something similar. The character is located in the middle of the screen, surrounded by windows that can be used to change the design. On the left, it is clearly labelled 'clothing' and there are arrows on each component that can be used to cycle along to the next one. This Is the same for the 'appearance' window, but the height has a slider. I chose this over something else like a set height/ a few set heights, as it allows the user to fully customize the character. And there are a few select colours to choose from.



Large arrow to indicate that it will take the user back to the main menu.

Larger level image to show it's the one selected.

Locked levels have a lock icon covering the image.

The level select screen contains a list of the different levels the user has unlocked, in the form of multiple different images in a row. Levels that haven't been unlocked are hidden behind a lock icon and potentially a blurry image of the level to be unlocked. There are arrows to the left and right of the 'Play Level' button, which can be used to scroll along the row. When the level is the one that can be selected, it is larger, indicating it's the one the user is on.

Development:

Player:

```
public class PlayerMovement : MonoBehaviour
{
    float speed = 15f;

    Vector2 pos  =  transform.position;

    pos.x += h * speed * Time.deltaTime;
    pos.y += v * speed * Time.deltaTime;

    transform.position = pos;
}
```

he first thing I wanted to do was make the player move. I started off by creating a variable named speed, which will decide how fast the player moves, I later changed this to moveForce as I think it better describes how it affects the player. At first, I also used the transform function to move the player, which changes the player's position, but later changed it to directly affect the rigidbody of the player, which is better for physics-based movement which I would like. I also declared variables movementX and movementY, these are the directions that the player will move.

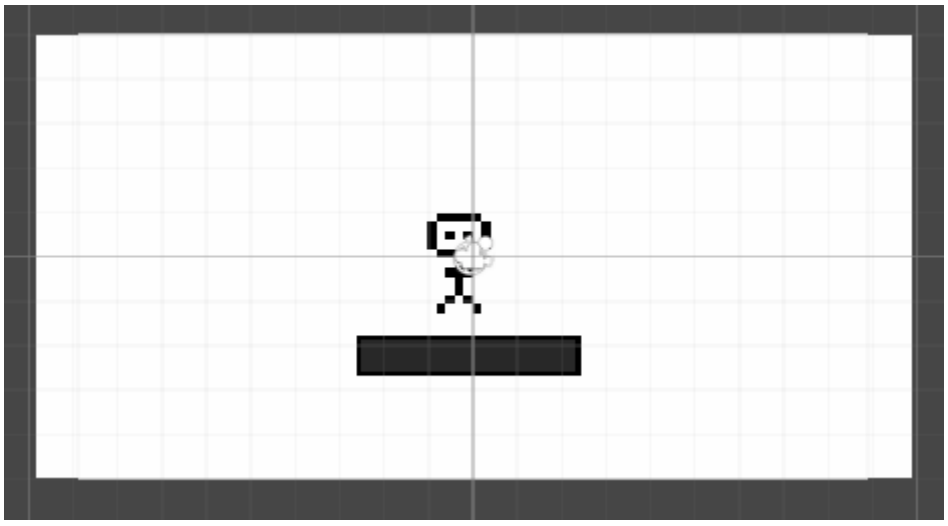
```
public class PlayerMovement : MonoBehaviour
{
    public float moveForce = 10f;  Unchanged
    private float movementX;
    private float movementY;
```

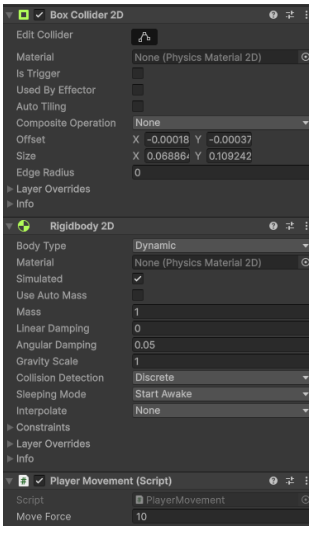
```
void Update()
{
    PlayerMoveKeyboard();

}

Frequently called  1 usage
void PlayerMoveKeyboard()
{
    movementX = Input.GetAxisRaw("Horizontal");
    transform.position += new Vector3(movementX, 0f, 0f) * moveForce * Time.deltaTime;
}
```

I then made a function which checks what button is pressed on the keyboard in 'GetAxisRaw' and adds it to the transform position equation. This multiplies the x position, moveForce and Time.deltaTime to move the player.

```
[10:58:01] InvalidOperationException: You are trying to read Input using the UnityEngine.Input class, but you have switched active Input handling to Input System package in Player Settings.
UnityEngine.Internal.InputUnsafeUtility.GetAxisRaw (System.String axisName) (at <ba054975f46641288f52f3ccb50da3ee>:0)
```

Program was not running. Fixed by changing 'Active Input Handling' to 'both





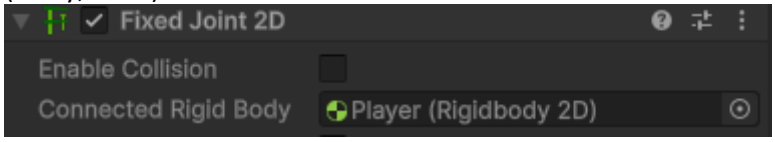
Camera:

```
public class CameraFollowPlayer : MonoBehaviour
{

    private Transform player;
    private Vector3 tempPos;

    // Start is called before the first frame update
    Event function
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player").transform;
    }

    // Update is called once per frame
    Event function
    void Update()
    {
        tempPos = transform.position;
        tempPos.x = player.position.x;

        transform.position = tempPos;
    }
}
```

Does not follow player
This is because the character is not called ‘player’ in unity.
So, I renamed it to player for it to work
Camera only followed player horizontally

```
void Update()
{
    tempPos = transform.position;
    tempPos.x = player.position.x;
    tempPos.y = player.position.y;

    transform.position = tempPos;
}
```

Tools:
Added a placeholder grappling hook that the player could hold. At first, I thought I could attach the player movement script to it and have them move together, but that didn’t work.
(Unity, 2025)

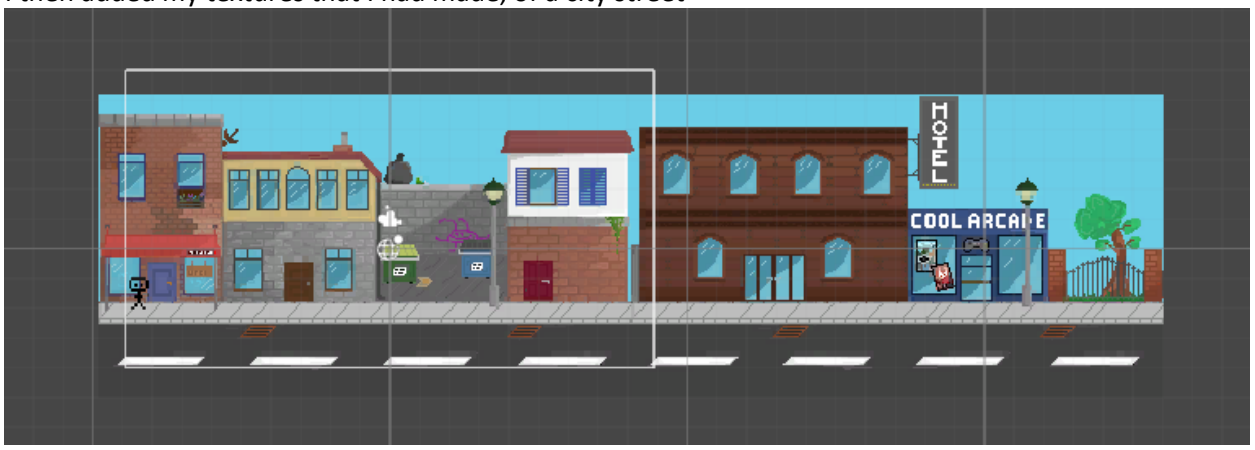


I used this to connect the player and the grappling hook

I wanted to have the tool point towards the player’s mouse cursor.
(Brackeys, 2025) – this video on top-down shooting provides code that makes the player look at the mouse position. I watched that section of the video and used the in-built functions I had learned to add it to my program.

```
void FixedUpdate()
{
    rb.MovePosition(rb.position + movement * moveSpeed * Time.fixedDeltaTime);

    Vector2 lookDir = mousePos - rb.position;
    float angle = Mathf.Atan2(lookDir.y, lookDir.x) * Mathf.Rad2Deg - 90f;
    rb.rotation = angle;
}
```

I then added my textures that I had made, of a city street



But the camera wasn't focused on the street, and you could see outside, which I didn't want

```
void Update()
{
    tempPos = transform.position;
    tempPos.x = player.position.x;
    tempPos.y = player.position.y + 5;
```

I added 5 to the position of the camera to see how that would affect it. I then settled on 5/2 being a good number to use as you could see the whole of the scene.

```
using UnityEngine;

No asset usages
public class SetBounds : MonoBehaviour Apply Rename refactoring
{
    Event function
    private void Awake()
    {
        var bounds = GetComponent<SpriteRenderer>().bounds;
        Globals.WorldBounds = bounds;
    }
}
```

I then wanted the camera the stop when the player reaches the end of the street, I couldn't figure out how to do this, so I watched a tutorial that (chonk, 2025) showed me how, and then I used what I had learned to code it into my game.

```
private Bounds = _cameraBounds;
```

```
private Bounds _cameraBounds;
private Vector3 _targetPos;
```

```
cam  = Camera.main;
//follow player
player = GameObject.FindGameObjectWit
//camera boundaries
var height = cam.size;
var width = height * cam.aspect;
```

```
public class PlayerTeleport : MonoBehaviour
{
    private Collider2D Teleporter;

    Event function  Bertie *
    void Start()
    {
        Teleporter = GetComponent<Collider2D>();
        Teleporter.isTrigger = true;
    }
    Event function  Bertie *
    private void OnTriggerEnter(Collider playercollider)
    {
        ;
    }
```

To move the player between levels, I have decided I want to have them teleport in game to change the scene, to test this, I made 2 portals and had the player teleport between them. I also want the screen to momentarily turn black to make the transition more seamless, as I will use this often, it will be a public function so it can be used all the time. (Unity, 2025) helped me to use the isTrigger function and onTriggerEnter to check if the player was touching the portal.

I haven't added anything to the function yet as I need to add a collider for the player.

```
private Collider2D playerCollider;

Event function  new *
void Start()
{
    playerCollider = GetComponent<Collider2D>();
    playerCollider.isTrigger = true;
}
```

```
private void OnTriggerEnter(Collider playerCollider)
{
    transform.position = (-20.184, 10.408, 0);
}
```

I tried just changing the position of the player when they entered the portal, but that didn't work. I then added an f after each number to specify a float, with a vector3 object. This came back with no errors

```
transform.position = new Vector3(-20.184f, 10.408f, 0f);
```

When I started the program, the player kept falling through the floor, I didn't know if it was a problem with the player or the ground, so I made a new object of a 2d box, the box fell and landed on the ground so I knew it was a problem with the player object.

```
private void OnTriggerEnter(Collider playerCollider){
    //transform.position = new Vector3(-20.184f, 10.408f, 0f);
}
```

I temporarily removed the code I had just added and it worked, because the isTrigger box being selected was removing the collision.

After this, the player wasn't being teleported, so I created a new object called PlayerCollider which moved with the player and moved the collider code from the player to a new script attached to that object.

```
public class PlayerCollider : MonoBehaviour
{
    private Collider2D playerCollider;

    Event function
    void Start()
    {
        playerCollider = GetComponent<Collider2D>();
        playerCollider.isTrigger = false;
    }
    Event function
    private void OnTriggerEnter(Collider2D playerCollider){
        transform.position = new Vector3(-20.184f, 10.408f, 0f);
    }
}
```

```
public class PlayerCollider : MonoBehaviour
{
    private GameObject player;
    private Collider2D playerCollider;

    Event function
    void Start()
    {
        player = GameObject.FindGameObjectWithTag("Player");
        playerCollider = GetComponent<Collider2D>();
        playerCollider.isTrigger = false;
```

This code ran, but it still didn't work, so I decided to teleport the player using a button that the player clicks when they get close to the teleporter. I made the previous code a comment.

```
/*private void OnTriggerEnter(Collider Teleporter){
    targetPosition = new Vector3(-20.184f, 10.408f, 0f);
    player.transform.position = targetPosition;
}*/
```

```
void PlayerTeleportButton()
{
    if (Input.GetKey( name: "f"))
    {
        transform.position = new Vector3(-20.184f, 10.408f, 0f);
    }
}
```

I added this function that just checks for when the 'F' key is pressed, I chose F as it is close to the WASD and E controls which are used in the game. And this moved the player to the correct position

Then I added a barrier to stop the player falling off the map, but the player could walk through it, so I changed the movement from transform.position to Rigidbody2d.MovePosition. (Unity, 2025)

```
void PlayerMoveKeyboard()
{
    movementX = Input.GetAxisRaw("Horizontal");
    rb.MovePosition( transform.position + movementX * Time.fixedDeltaTime * moveForce);
}
```

```
void PlayerMoveKeyboard()
{
    Vector3 movementX = new Vector3(Input.GetAxisRaw("Horizontal"), 0);
    rb.MovePosition( transform.position + movementX * Time.fixedDeltaTime * moveForce);
}
```

```
[14:56:37] NullReferenceException: Object reference not set to an instance of an object
PlayerMovement.PlayerMoveKeyboard () (at Assets/Scripts/Player/PlayerMovement.cs:18)
```

This didn't work at first, but then I changed movementX to Vector3 and removed the line initializing it as a float. And there were no errors, until I ran the program and there was a continuous error, I didn't know what was wrong, so I used this site (plbm, 2025) to help me figure out why.

```
public float moveForce = 6f;
private float movementY;
private Rigidbody2D rb;

Event function
void Start()
{
    rb = GetComponent<Rigidbody2D>();
}
```

I then added this line inside of the start function which referenced the rigidbody, which applies a force to the rigid body instead of just changing the position with transform. This worked, the player moves and cannot move through the barrier I placed.

I now wanted to make the player jump, as I had sorted out the movement.

```
void PlayerJump()
{
    if (Input.GetButtonDown("Jump"))
    {
        rb.AddForce(new Vector2(0f, jumpForce), ForceMode2D.Impulse);
    }
}
```

```
void Update()
{
    PlayerMoveKeyboard();
    PlayerJump();
    PlayerTeleportButton();
}
```

```
if (Input.GetKey( name: "f"))
{
    rb.AddForce(new Vector2(0f, jumpForce), ForceMode2D.Impulse);
}
```

```
void PlayerJump()
{
    if (Input.GetKey( name: "e"))
    {
        Jump = new Vector3(0, jumpForce, 0);
        rb.AddForce( Jump * jumpForce, ForceMode2D.Impulse);
    }
}
```

I added a function named PlayerJump to void Update, as that is called every frame, so it will always be checking for a button to be pressed. I added the impulse mode so the force is applied instantly.
This didn't work at first, as the player didn't jump when the space bar was pressed. I then changed the input type from GetButtonDown to GetKey and it still didn't work, as the player only jumped occasionally, and with a lot of force.
I still didn't know what was wrong so I slightly altered the code again but nothing changed

```
void PlayerJump()
{
    if (Input.GetKey( name: "e"))
    {
        Jump = new Vector3(0, jumpForce, 0);
        rb.AddForce( Jump * jumpForce);
    }
}
```

```
private void FixedUpdate()
{
    PlayerJump();
}
```

```
private void FixedUpdate()
{
    if (Input.GetKey( name: "e"))
    {
        PlayerJump();
    }

}
```

```
void PlayerJump()
{
        Jump = new Vector3(0, jumpForce, 0);
        rb.AddForce( Jump * jumpForce, ForceMode2D.Impulse);
}
```

I then removed the Impulse from the line, and the player now jumped but it was quite simple and didn't work very well, I wanted to add it back in to make it seem more like a proper jump. The player also fell slowly so I increased the gravity scale to about 30, as that seemed to mimic real life gravity more accurately.
I then moved the player jump to FixedUpdate, which is better for physics-based functions than just the Update function. But the player still didn't jump when the button was pressed.
I then decided to take the inputs for the movement functions in Update rather than in their specific functions, as I thought it may change something.

```
private bool isGrounded;
Event function  Bertie *
void OnCollisionEnter2D(Collision2D collision) {
    if (collision.gameObject.CompareTag("Ground")) {
        isGrounded = true;
    }
}

Event function  new *
void OnCollisionExit2D(Collision2D collision)
{
    if (collision.gameObject.CompareTag("Ground"))
    {
        isGrounded = false;
    }
}
```

I decided to move on as it still wasn't jumping, I added 2 functions to see if the player was on the ground and to return true or false, I then edited the jump function to require