Edward Butler 2345

OCR Computer Science Coursework

# 2D Platformer

## Contents

## Analysis

### Features that make the problem solvable by computational methods

In my 2D platformer there will be movement mechanics for the player. Computational methods are required for smooth 2D movement to be coded in Unity as (my software of choice) as void Update is used to repeatedly run the code on every frame in my game, this coupled with Time.deltaTime can ensure the movement is smooth. In the game there will be hazards, all hazards will use computational methods to interact with the player in some sort of way, some could trigger a game over screen etc. Some hazards in the game will be moving from side to side, I need to use computational methods to create a script which will make these hazards move from side to side, iteration can be used to create this repetitive movement. Selection will also be used within the code to take keyboard and mouse inputs from the player. Sequence will also be important throughout the project in relevance to the order that codes are executed.

### Stakeholders

There are three main stakeholders for my project. Each of these three stakeholders will benefit from the project in a different way.

Casual Gamers

The first stakeholder is the casual gamer. Casual gamers are people who just want to play the game without worrying about how skilled they are at it. They usually don’t play games too much however they will every now and again when they have free time. Casual gamers are predominantly younger people who are within the children to young adult age range.

The identified problem for the casual gamer stakeholder is general boredom as most of them are young (children to young adult) who have lots of free time to fill due to reasons such as: not having a job, which may also apply to some casual gamers who fall under older age categories, though not quite as likely. Adult casual gamers may also need something to pass time when they are at home and not working.

The solution that the project provides for casual gamers is this game now exists and is widely accessible as it is available on PC, so most casual gamers won't have trouble getting the game. The game is simple to understand and doesn't require much skill to play at a level where it is enjoyable, however there would still be room for the player to improve at the game which would make it playable for longer.

A drawback of this solution, however, is that casual gamers who don't usually play games on a PC because either they don't prefer it, or they don't own a PC would be inconvenienced as either the game isn't available for them or it isn't available on their preferred platform.

Speed Runners

The second stakeholder is the speed runner. Speed Runners in gaming are people who attempt to complete the game as fast as they possibly can. They record the time it takes them to complete the game or a certain section of the game and then they individually upload the score to a website/forum where then it will be put into a leader board with other players to see who has the fastest time.

The identified problem for speed runners is not having a simple game which is easy to understand to speed run. Often speed runners will change the game that they speed run which can be due to either being burned out of a previous game or they might be looking for a new game which has top scores and times which are realistically beatable without having to put in hundreds of hours of time into. New games that are simple and easily understandable can be few and far between.

The solution which this project provides for speed runners is a new fresh game which will have in game time keeping which is beneficial for speed runners as they will not have to use an external software to keep a track of the times fairly for all players. As the game will be new, there will be no existing time records which would make the earlier days of the game more exciting and therefore attractive for speed runners, this coupled with the easy-to-understand game mechanics could lay the ground for a good community of speed runners on the game.

Content Creators

The third main stakeholder of the game would be content creators. Content creators are people who make content which, in this case is central to a game. Content is usually in the form of video or stream however it could also be an online blog.

The identified problem that contents creators face is a possible lack of platform to make content for, which in this case is a game. Content creators constantly need to produce content as it is their job and their brand. Fans follow their favourite content creators and expect them to produce content, many content creators create content for lots of different games and not introducing any new ones can get stale for their audience.

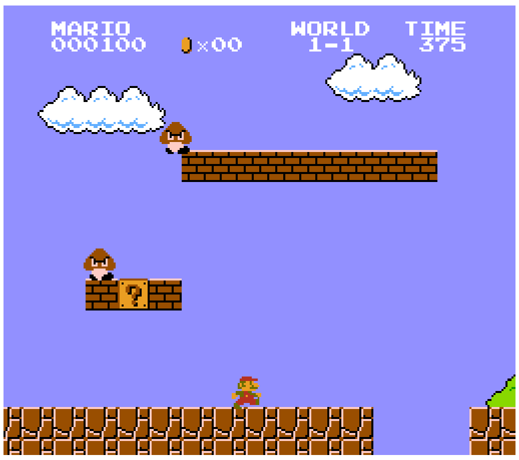
The solution that the project hopes to provide for content creators is to give them a new platform (game) to create content with which will help mix up the content they produce, preventing it from becoming stale and repetitive which benefits their analytics meaning their watch time/views/monetisation increases. This game would be good for content creators as it would be effective at bringing in new content consumers as the game is easy to look at and understand mindlessly.

## Research Existing Solutions

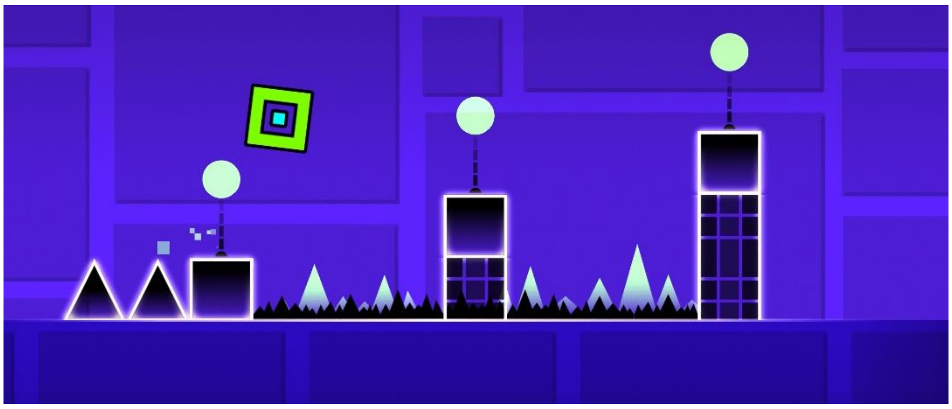
There are two existing solutions, I have selected which are relevant to the game that I am producing. It is important for me to mentally refer to these two games when designing my own.

The first of the two games are Super Mario Bros. which was released in 1985. It is also a 2D based game with a controllable character where the camera is at a side view to the stage. This game is relevant to the one I am producing as it is simple for anyone who is looking at it for the first time to understand however when it comes to the mechanics of the game itself they are easy as they come to control initially however there comes immense depth and preciseness with the controls as the player begins to play more and improve, which helps the game become one that is played very competitively by speed runners despite being so simple and easy to understand to new people. This dynamic of simple game that is also attractive to more competitive gamers like speed runners is what I am hoping to achieve with my project.

Inspiration I would like to take from Super Mario Bros. is that the game has a timer implemented into it. While this timer in Super Mario Bros. is not great for speed running as it doesn’t use real seconds, I would like to use the idea of having a timer in my game to benefit speed runners, even if that wasn’t the intention for the timer in Super Mario Bros. I would also like to add hazards which is something that Super Mario Bros. has; however I would like to implement them differently to Super Mario Bros and have some be stationary and some be moving in a predictable movement pattern.



The second of the two games is Geometry Dash which was released for PC in 2013. I have chosen this game to be relevant to mine as it is also a side view 2D game, however the attraction with geometry dash is that it is extremely simple for even a child to understand from first looking at it, which is an aspect what I want my game to have.



Inspiration I would like to take from Geometry Dash is that the game many different modes however there is only one which is predominantly used which is the "cube" mode where the players only function is to jump, other modes include flying and rolling which I would like to implement something similar into my game if my timeframe allows so.

<u>Essential Features</u>

Horizontal movement

Basic horizontal movement using the keys to navigate side to side (left and right). Where the A key moves the player to the left and the D key moves the player to the right. This is essential for the basic movement mechanics for the player.

Gravity

Basic gravity which enables in game objects to fall freely. This is essential for the basic movement mechanics of the player.

Hazards

There should be at least one (preferably more) form of hazard in the game which would be either stationary or moving and would trigger an event upon contact with the player (for example if the player touches hazard: "spikes" then it would produce a "game over"). This is essential in making the game more complex while still coming across as simple.

Scoring

There should be some sort of score tracker where score could be obtained by obtaining collectibles in each level which is inspiration taken from both existing solutions or could be obtained by the number of hazards avoided per unit of time in an individual level. The score feature is essential to encourage competitive interaction between players.

Timer

There should be a timer feature which tracks the time in which it takes for the player to complete a certain level, the time will then display in large text at the end of the level. A timer is essential to satisfy the needs of specific stakeholders such as speed runners and competitive gamers.

## Limitations

### Platform

The platform that the game will be produced for is just PC. It would be beneficial for the game to be made for other platforms such as mobiles and home game consoles, however due to time constraints, I have decided that limiting the platform to PC only is most beneficial as most households have a computer and I feel as if the controls that a keyboard comes with are most suitable for this genre of game.

### Graphics

I have decided to keep the sprites used in the game at a low resolution, this is because I am the only person working on the project and have little skill in graphic design. Spending too much time trying to create attractive high-resolution graphics for the game could be detrimental to other parts of the project as I am working with a deadline. Using low resolution textures could however, be beneficial in making the game look more simple and would make the game more accessible for people with low end computers.

### Input methods

The method of input for the game will be restricted to keyboard and mouse only. I have decided against implementing other methods of inputs like games controllers as I am working with a deadline, and I don't want the implementation of a new input device to affect the quality of other parts of the project. I think that adding the ability to play the game with other forms of input devices – mainly a games controller – would be beneficial to specifically the casual gamer stakeholder (though it may be beneficial to others too) as they can use their preferred choice of controller device to play the game with.

## Requirements

### Music

Background music should be added into the game. This would add a layer of personality to the game, which is important in enhancing the appearance of the game. Music would also help prevent the game from being too silent.

### Sound effects

Sound effect should be added into the game as they would help the player better understand what's going on in the game using audio – even if they haven't yet noticed something has happened visually, the audio can help guide them.

### Animations

Animations should be added to the player and hazards to make them feel more "alive" and enhance the visuals of the game, ensuring that it doesn't look lifeless.

### Sprites

Sprites are necessary for the player to know what is happening on the screen. If there are no sprites the game is unplayable

Menus

When the game is opened, menus should show up which the user can select to either start game, exit game, or open the games settings.

Logo

The game should also have a logo. A logo would make the game more recognisable; this logo could be put on the main menu screen and in the corner of the “level complete” and “game over” screens.

<u>Exploits</u>

There shouldn’t be any exploits in the game, which I will discover and overcome via testing.

<u>Measurable success criteria</u>

Music

The music added to the game should be quiet and an ambient as it is only there to create an atmosphere for the game and not something too extreme like pop/rock music etc.

Sound effects

The sound effects that are added into the game should be relevant to events which happen in game – they should not just occur randomly – sound effects should also sound according to what they are representing. If the player dies the sound effect should attempt to sound negative.

There must also be a way to alter the volume of the music and sound effects (separately) – should be located within the options menu.

Animations

The animations in game should be relevant to the action that they are trying to represent. If the character starts walking, then the walking animation should be played and the character should face the direction accordingly.

Sprites

Sprites should also be relevant to what they are trying to represent. If there is a hazard in game, it must be easily identifiable as a hazard even by someone who has never played the game before. Ground sprites should also all fit together nicely with no uneven lines and textures between them.

Menus

The menus that are presented in the game should be easily readable and must stand out from the background by using different colours to ensure everyone has a seamless time reading them. Menus such as the “exit game” function can just be represented by a red cross surrounded by a box in the corner of the screen. The menus should all use the same two to three colours and a uniform font.

Logo

Like the menus, the logo should be easily readable and should just be the name of the game. The logo should use two to three colours which match with the colours on the menus for the game.

References

Geometry dash image – https://geometry-dash.fandom.com/wiki/Stereo_Madness

Super Mario Bros image - https://www.mariowiki.com/World_1-1_(Super_Mario_Bros.)

Edward Butler

2D platformer

## Design

### Pseudocode

Pseudocode for horizontal movement

```
-------------------------------------------------------------start of code ---------------------------------------------------------


public float movementSpeed = 1


float horizontalMovement = input getaxis "horizontal"

temp = transform position + horizontalMovement,0 * movementSpeed

transform position = temp


-------------------------------------------------------------end of code-------------------------------------------------------------
```

Description

The public float variable which is declared at the start of the script as "movementSpeed" is set to 1 which is just a placeholder to ensure the code works. The variable is adjustable inside of unity's UI and I will use this to fine tune it to make sure the horizontal movement feels right and then I will go back to the code to edit this variable to what I have found works best within unity's UI.

horizontalMovement is declared as getaxis "horizontal" which sets up an axis which allows a 2 direction scale along the x axis of the 2 dimensional game which in other words means it allows the player to move the character from left to right, it does this by allowing an input to return a float value between values -1 and 1 where -1 is movement to the left and 1 is movement to the right.

The temp variable is a temporary store used to calculate how much the position of the character is going to move after the player has made an input. Horizontal movement is the input made by the player for movement along the x axis, this is multiplied by the float movementSpeed which will allow for the character to actually move and for the speed of the character to be altered later on by either me or in game modifiers which may change movement speed.

Transform position is then set to equal the value stored in the temp variable to actually move the sprite.

-----------------------------------------------------------------------------------------------------------------------------------

Psuedocode procedure for jumping

-----------------------------------------------------------start of code -------------------------------------------------------

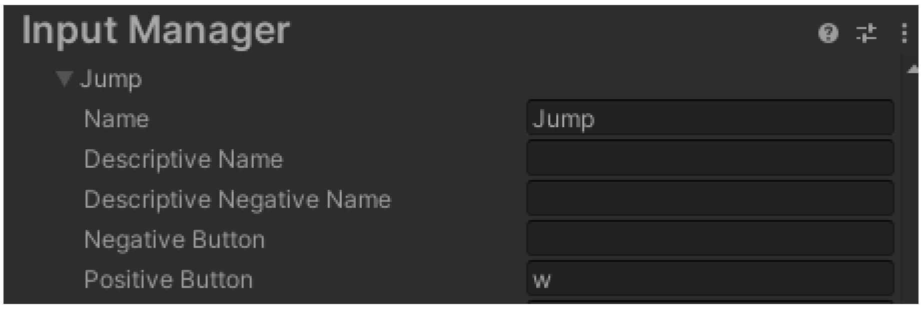
```
public float jumpHeight = 1


if input getbuttondown “jump”
        addforce 0,jumpHeight
```

-----------------------------------------------------------end of code---------------------------------------------------------

<u>Description</u>

The public float variable “jumpHeight” is declared at the start of the script to later determine how high the character will jump, this float is similar to the float variable movementSpeed which I used previously in the script for horizontal movement in the sense that this variable is just set to the value 1 as a placeholder and can be later fine tuned in the UI of unity which I will use to figure out what feels right for the game and once i’ve decided what feels right I will go back into the actual script to change it.

Then a simple if statement is used to execute the force of jumping on the character upon an input that the player has made. getbuttondown is the way I have chosen to take inputs from the player in unity as this allows me to have named inputs as opposed to just using key codes. In this case, I am using the referenced button input “jump” which is built into unity by default and is assigned to the space bar, however in my game I would like the jumping function to be set to the key W, as I believe this would make for a more comfortable control scheme so I will go into unity’s input manager to change the key of the referenced button input from the space bar to the W key. Addforce is a unity function which adds a force in the direction specified, in this case the Y axis and since jumpHeight is a positive value the force will be upwards. So in abstraction this script just means ‘if the W key is pressed then add an upwards force to the character”.


Input Manager

Jump

| | |
|---|---|
| Name | Jump |
| Descriptive Name | |
| Descriptive Negative Name | |
| Negative Button | |
| Positive Button | w |

It's worth mentioning that the jumping procedure is also called at the start of the horizontal movement procedure to make it so jumping and horizontal movement can occur at the same time.

---

Pseudocode for gravity

------------------------------------------------------------start of code --------------------------------------------------------

```
public float strength = 1

velocity = gravity * strength
changeinposition = velocity
```

------------------------------------------------------------end of code-----------------------------------------------------------

<u>Description</u>

The public float variable called strength is declared at the start of the code and is assigned the value 1 which will later be used as a multiplier to decide how strong the global gravity is going to be across the project. This variable is just a placeholder for now and since it is set to 1 it is currently just using the default built in gravity value in unity which is -9.8 however I will change this later when I am fine tuning the movement of the character to make sure it feels great.

The float variable velocity is declared as being equal to the product of gravity * strength. “gravity” in this case is the physics2D gravity which is built into unity as the acceleration due to gravity globally for the project, this value is a float -9.8 by default and at this stage I see myself leaving this value at -9.8 as I do not see a need to change it within the project settings because I can use a multiplier float variable to edit it more easily if needed, which is where the “strength” variable comes into play. The gravity is multiplied by the strength float so I have an easy way into editing the gravity which I may do later on depending on how the default strength feels.

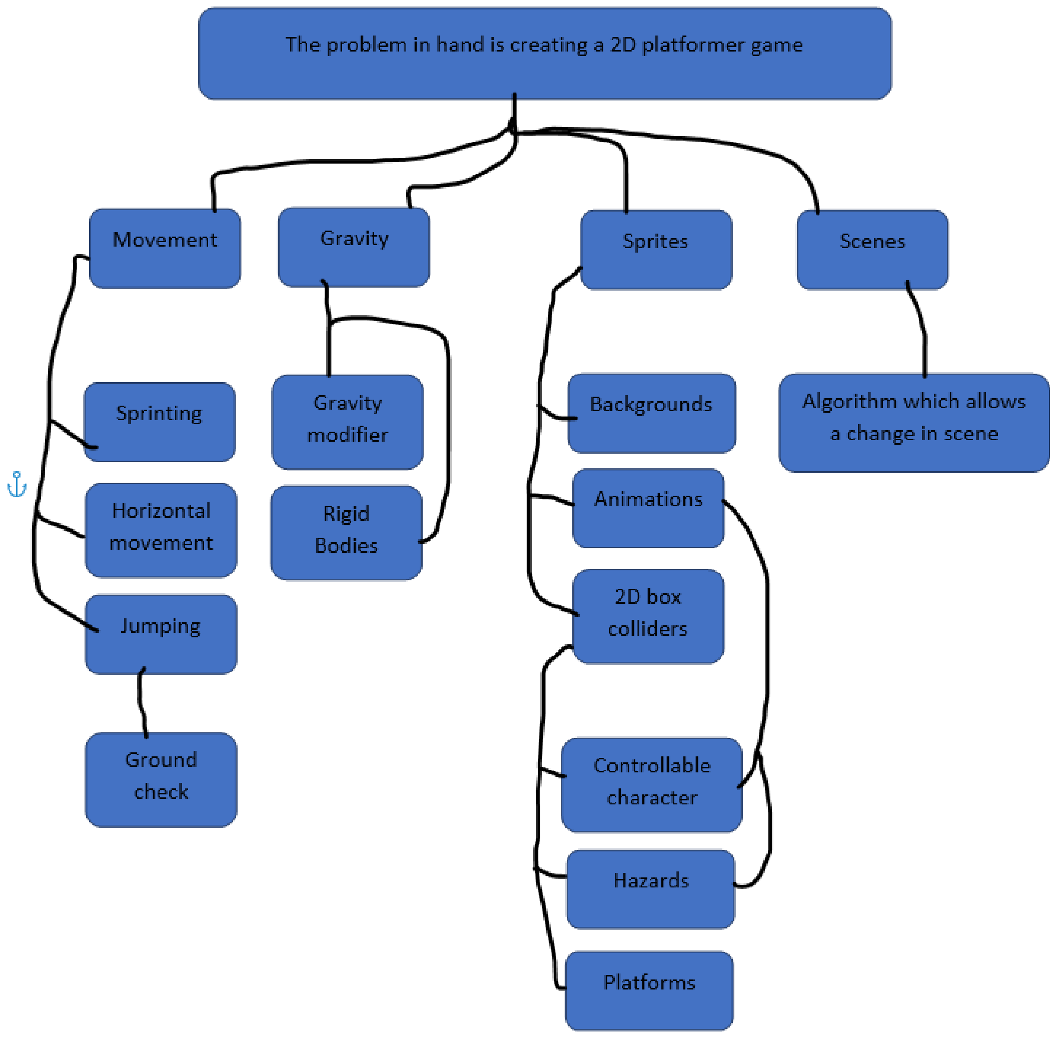
Lastly changeinposition is equal to velocity which means the position will be changed along the y axis by the value which is stored in the velocity variable which will normally be a negative value (and in this case it is) which means the direction is downwards. So for example, if the strength is set to 1 then changeinposition is -9.8 on the y axis.

---------------------------------------------------------------------------------------------------------------------------------

<u>The breakdown of the problem into smaller solutions</u>

It is important to breakdown a larger problem into smaller problems as smaller problems can be easier to handle as opposed to trying to tackle one huge problem at once. It is also important in some instances as you can allocate the different smaller problems to yourself at different times or in some cases (not my instance) they can be allocated to other people to improve the efficiency of finding a solution.

Here is a diagram for the breakdown of my solution which I sketched in Microsoft word:



The greater problem at hand here is creating the 2D platformer game which I have decided to break down into four different major sections, which are as listed: movement, gravity, sprites and scenes. I have done this to ensure designing the solution is easier for me to comprehend and I can allocate myself different time periods to create these four different components. Within all the four major segments I have further broken them down into more bitesize chunks so I can get them done efficiently in my time to ensure I am working well towards the deadline.

Movement

The first of the four major segments is the most important and therefore will be the one I will start with when I am developing the first iteration prototype for my project. The segment is movement which is vital for the game to function. Since this game is going to be a side view camera 2D platformer the basic movement necessary is that of moving from left to right (along the x axis). The smaller sections which I have movement broken down into include: sprinting, horizontal movement, jumping and a ground check.

The first section within movement is sprinting. Sprinting will involve a short function which uses selection to identify if an input is being made by the user in the form of a held key press, if it is a multiplier will be added to the speed of the horizontal movement.

The second section within movement is horizontal movement which is the most important. Horizontal movement will consist of a script with a procedure which is attached to the sprite which is controlled by the player. The scrip will allow the player to move the sprite along the x axis (left and right) by using corresponding keys "a" for left movements and "d" for right movements.

The third section within movement is the function for jumping which, like the sprinting function, will make use of the programming construct selection by utilising an "if statement" where if the user makes a button press – in this case the "w" key – the function returns an upwards force on the sprite which the script is attached to which in this case will be the sprite controlled by the player. This function will also be called at the start of the procedure for horizontal movement so both horizontal movement and jumping can both be used in unison.

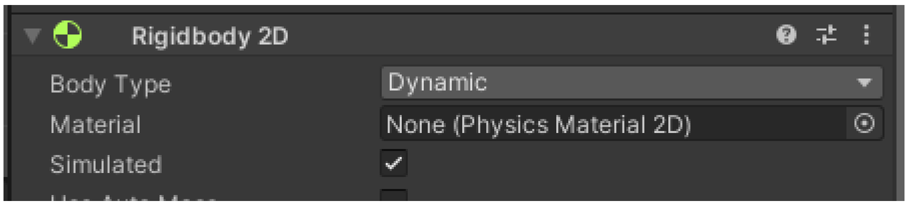
The fourth section of movement is the ground check which will link into the use of gravity and 2D hitbox colliders in other sections of the project. The ground check allows for the player to only jump if they are touching the ground which in this case the ground is a platform sprite with a 2D box collider applied to it. The ground check is necessary as without it the user would be able to make the user to make the controllable sprite jump infinitely without touching the ground.

Gravity

Despite there being not much to it, a section of the project that I have considered to be a major one is the gravity in the game. The gravity I will use is built into unity's "physics2D" which has no effect to the x axis but returns a -9.8 to change in position in the y axis. If the float value -9.8 doesn't feel right for the gravity however, I will edit it with a gravity strength variable. Gravity will be the same globally across the game and will affect every sprite with a rigid body unless specified otherwise within the inspector of the sprite in unity's UI.

A subsection of gravity which I am also going to implement is a variable which can alter the strength of the global gravity which I can use to adjust the gravity if it doesn't feel right or use for some sort of temporary gravity change in the game. The variable will be declared globally as a float and will later be used to multiply the effect of unity's physics2D gravity.

The other subsection of gravity is the use of rigid bodies. Rigid bodies will be used to define whether a sprite in my game will be affected by gravity. Rigid bodies are important for sprites such as platforms/the ground/ the background as I do not want these sprites to be affected by gravity. To change whether they are affected by gravity I will apply a rigid body to them and inside unity's inspector I will change the body type to "kinematic" which means they will stay in place whereas sprites which I do want to be affected by gravity will be set to "dynamic".


Rigidbody 2D

| | |
|---|---|
| Body Type | Dynamic |
| Material | None (Physics Material 2D) |
| Simulated | ✓ |

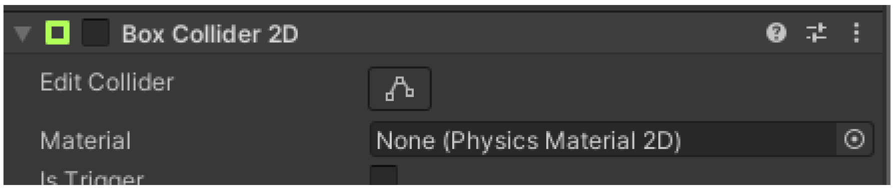
Sprites

Sprites are important for the game to look nice visually and for the user to receive visual feedback on how their inputs are affecting the gameplay. It is also important that sprites look like what they are meant to be which makes the game more understandable. I have broken down sprites into three main sections which are: backgrounds, sprites with box colliders and sprites which have animations. It is also worth noting that I will be using the software "paint.net" for creating all the sprites.

The first section of sprites is the backgrounds which are quite simple. The backgrounds are not going have any script attached to them as their only purpose is to make the game look nicer. All the backgrounds will be set to kinematic in the sprite inspector under rigid body 2D o make it so they aren't affected by global gravity.

The second section of the sprites is sprites which have animations attached. Animations will be created for the sprite by using a frame by frame sprite sheet. I can then cut up this sprite sheet in unity so that the frames are separate.

The third section of the sprites is sprites which have 2 dimensional box colliders and added onto them. These box colliders are necessary so the player will actually stop when they walk into/land on another sprite as without the box colliders the player would pass right through other sprites. I might also consider using edge colliders as they do the same job as box colliders but provide more intricacy while drawing them.


Box Collider 2D

Edit Collider

Material None (Physics Material 2D)

Is Trigger

The first sprite which I will create is the one which the player controls. This sprite will have the script which contains the code for horizontal movement, sprinting and jumping (which are all on the same script) which will allow the control of the sprite by the user. This sprite will also have a 2D box collider on it to allow for the control of the sprite on a solid. A rigid body 2D will also be added onto this sprite with the body type set to "dynamic" to allow gravity to affect the movement of this sprite. This sprite will have animations for the different types of movements listed: left, right, jumping and idle. Unity will decide which animation to play from the flow diagram which I will build in the animator. The animations will looped by iteration until told to stop by selection when an input which triggers a different animation is made.

The second lot of sprites I will create is the platforms and the ground. There will only be code on the platforms which move, otherwise these sprites don't have any scripts attached to them. These sprites will have a 2D box collider to make it so other sprites such as the player or hazards can move on top of them or collide with them without passing through. They will also get a rigid body 2D attached to them which will be set to body type kinematic to stop them from being affected by gravity as it is required for them to stay in situ.

Finally hazards will be a sprite which will via selection, damage the player in a certain way if it collides with them. There will be a script attached to these sprites which will give them a pattern to move which will generally just be repeating left to right movements. They will also have a 2D box collider and a rigid body 2D set to dynamic attached to them as gravity may affect them and collisions are necessary to code damages to the player. Hazards will also have short movement animations which will loop by iteration and change to whether they are moving to the left or the right from a short flow diagram which I will make in the animator.

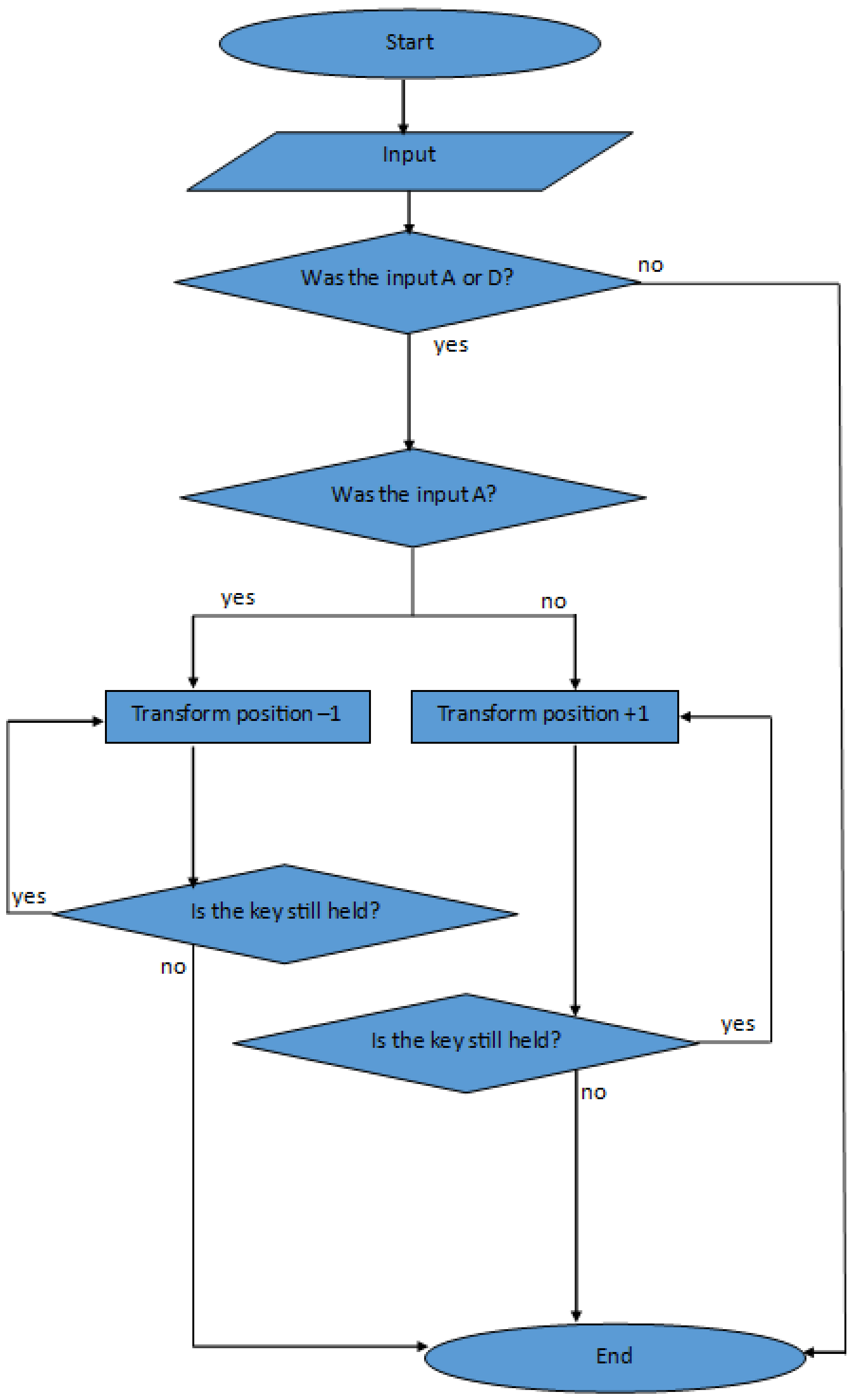
Scenes

Finally the last and the shortest section is scenes. Changing the scene is important when changing level. When a new level needs to be loaded the scene will change with a new background and a new layout of platforms will be present. An algorithm for changing scenes will be used which will use only selection when the player reaches a certain point in one scene then it will be replaced by the next scene.

## Flow charts

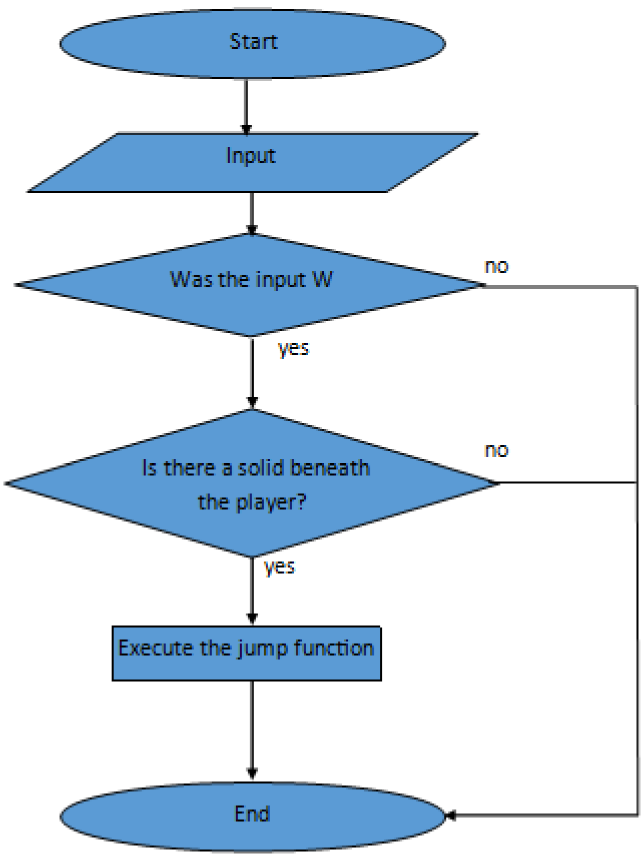
Flow charts are important to present what happens during the execution of different parts of the program. I have put together some flow charts that are relevant to different parts of my project. These flow charts are beneficial as they help better understand how the code is working.

Flow chart for horizontal movement:



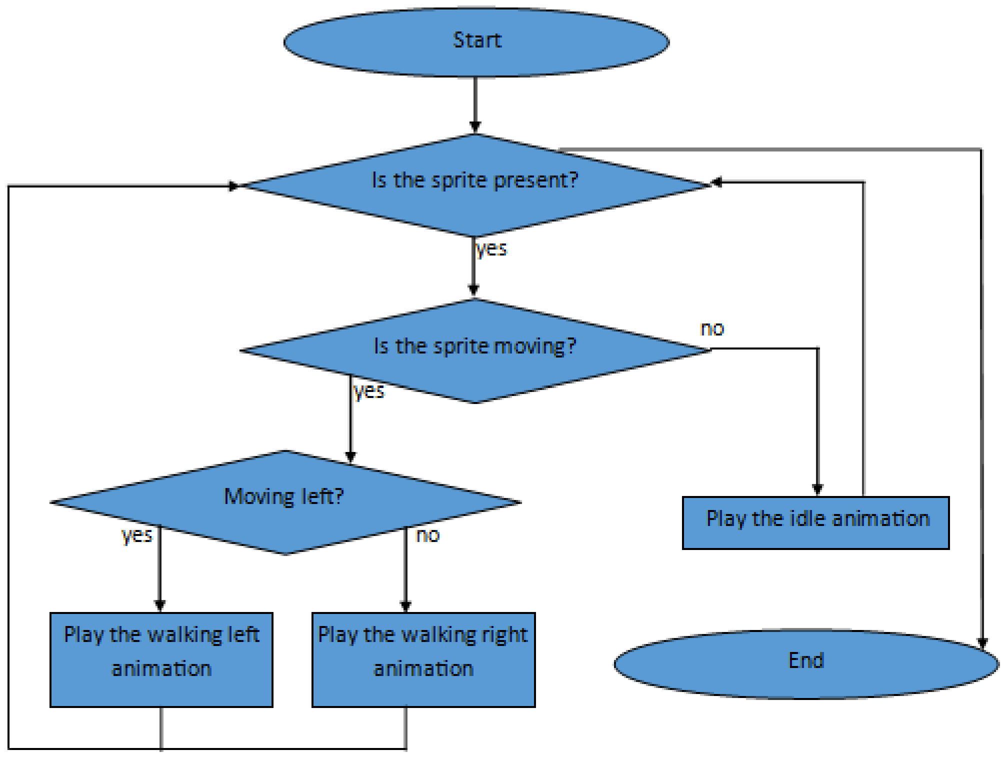
This is a flowchart which I made in Microsoft publisher to display what happens within the script of horizontal movement. Firstly an input is made by the user. Then the computer asks if the input made was either the “a” key or the “d” key, if it is neither then the flow chart ends however if it is one of “a” or “d”, the computer continues and asks if it was the “a” key, if it was the a key then position of the sprite is transformed by -1 which i.e. to the left and if the input wasn’t “a” then it must be “d” so the process “transform position by +1” is executed which means the sprite moves to the right. Then for both of these processes the computer asks if the key is still held, if it is then the process will repeat itself with iteration until the key is no longer held, if the key is no longer held then the flow chart ends.

Flow chart for a ground check:


Start
Input
Was the input W
no
yes
Is there a solid beneath the player?
no
yes
Execute the jump function
End

This is a flow chart I made in Microsoft publisher to show what will happen during a ground check. Firstly an input is made by the user, then the computer asks if the input was the “w” key which is the key which runs the jumping function. If the input wasn’t “w” then the flow chart ends however if the input was “w” the computer then asks if there is a solid beneath the player, if there isn’t solid beneath the player then the jump cannot be executed as the player is already airborne however if there is solid beneath the player then the jump command is executed and the flowchart then comes to an end.

Flow chart for walking animation:



This is a flow chart that I made in Microsoft publisher to demonstrate walking animations along the x axis for a sprite. Firstly the computer asks if the sprite is actually present on screen, if it isn't the flow chart instantly ends while if it is the computer then goes on to ask if the sprite is moving, if it isn't moving the idle animation will be played and the question prior will be infinitely iterated unless something changes such as the sprite disappearing or the sprite begins to move. If the sprite is moving the computer then goes onto ask if the sprite is moving left, if it is it will play the animation for moving left and will constantly iterate this question (like with the idle animation) until something changes. If the sprite is moving but its not moving left then it must be moving right and so the animation for moving right is played and will constantly iterate the flowchart until something happens otherwise.

## Iterative development

### Iteration 1

When I first created the unity document, the only folder present by default was the "Assets" folder. So to keep things organised within the assets folder, I created three new folders:

1) Sprites – where all the 2D sprites for the project would be located.
2) Scripts – where all C# scripts for the project would be located.
3) Animations – where animations for sprite sheets which have their sprite type set to "multiple" would be located.

To begin my project I started out by creating three sprites:

1) A sprite which will be called "character", this sprite will designed as a sprite sheet as is has animations for walking and standing still. When I first dragged this sprite onto the scene, I

had to set sprite type to "multiple" and I had to change the filter mode to point no filter. Since this sprite was set to multiple, I used the sprite editor to split the sprite sheet up into 32 x 32 sprites which is the size of each sprite frame of the sprite sheet

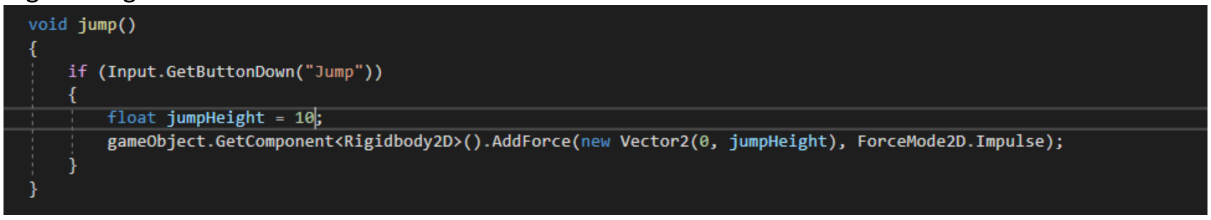
2) The second sprite I added is a "platform" sprite which is what the character would be able to walk along. I added a 2D box collider to this sprite and set it to static and I also added a 2D rigid body to and set it the body type to kinematic to stop the gravity, which I would later implement from affecting it.
3) The third sprite I added wasn't necessary at this point in time however it wouldn't take long for me to add so I decided I would. This sprite was the "background" sprite. For this sprite I just dragged it onto the screen and aligned it properly with the camera.

All the sprites I added were too small at first so I changed their scale from 1 to 10. All the sprites I added were small pixel arts so they came out blurry onto unity so to tackle this, I changed their filter type to "point (no filter).

Movement script

The first script in the script is the void Update() procedure, this procedure is a unity procedure which updates on every frame. I have all other functions and procedures called into the void Update() procedure as I want them to update every frame.

The next part of the script is the jump() function which is used for jumping. This function consists of selection via the use of an if statement where if the button that "jump" is set to within the unity input manager, which by default is the space bar however I would prefer the jump button to be set the W key and so I went into the input manager to change the "jump" button from the space bar to W. Within the if statement a float variable is declared which is called "jumpHeight", jumpHeight is a multiplier on the upward force which is giving a jumping effect. I set jumpHeight to 1 however when I test this I will change it to something which feels better for the game. Then within the if statement is the code: "`gameObject.GetComponent<Rigidbody2D>().AddForce(new Vector2(0, jumpHeight), ForceMode2D.Impulse);`" this code simply means a force is applied to the 2d rigid body in a direction on the y axis where that direction is the value held in the float jumpHeight.
After testing this I discovered the code ran as intended but the jumpHeight being set to 1 was not high enough so for now I amended it to 10



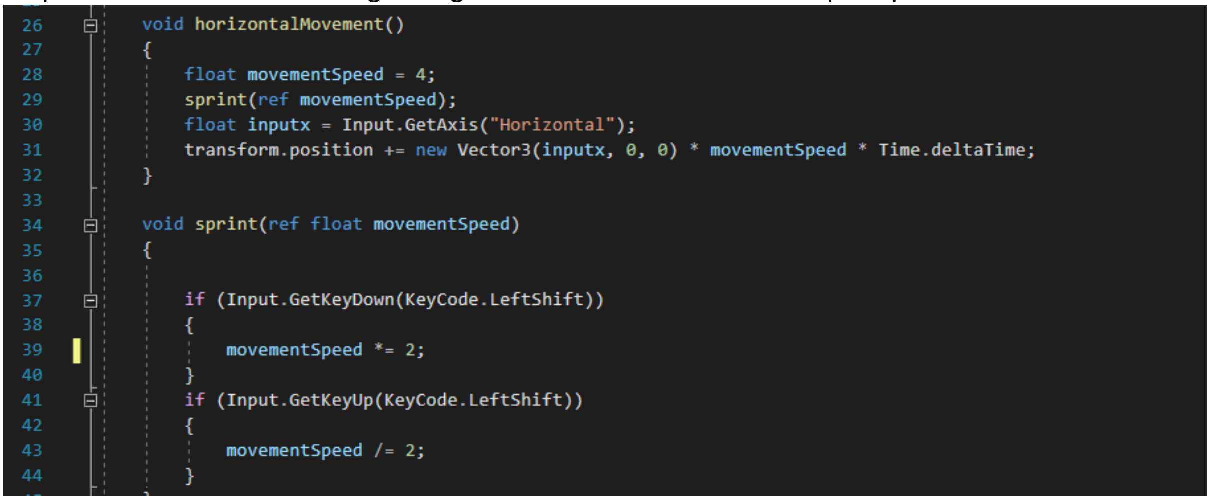
```
void jump()
{
    if (Input.GetButtonDown("Jump"))
    {
        float jumpHeight = 10;
        gameObject.GetComponent<Rigidbody2D>().AddForce(new Vector2(0, jumpHeight), ForceMode2D.Impulse);
    }
}
```

The next section of code is that for moving horizontally, so a new function is created called horizontalMovement. The first line of code within this function is a float is declared called movementSpeed. movementSpeed is a multiplier which will be applied to the amount the subjects position will be transformed by, giving the effect of changing the speed. movementSpeed is set to 1 which is a value which I will later change in testing. The next line of code is where another float is declared as the value which axis "horizontal" returns from an input which should range from -1 to 1, however since the inputs are being made by keys on a keyboard, they are either just -1 for the A key and 1 for the D key. Get axis "horizontal" within this float, sets up an axis along the x coordinate which can return a value between -1 and 1 which depends on the input made by the user. The next line of the function then puts together the two float variables that have previously been created by transforming the position of the subject by declaring a vector3 variable with its value set only on the x axis which is the value given from the variable inputx which was previously declared as either -1 or

1 depending on the users input, this is then multiplied by the movementSpeed variable which is the multiplier of how much the subject's position is going to be changed by, finally this is multiplied by Time.deltaTime which is a unity function and in this case is being used to smoothen out the transform in position of the subject over the frames.

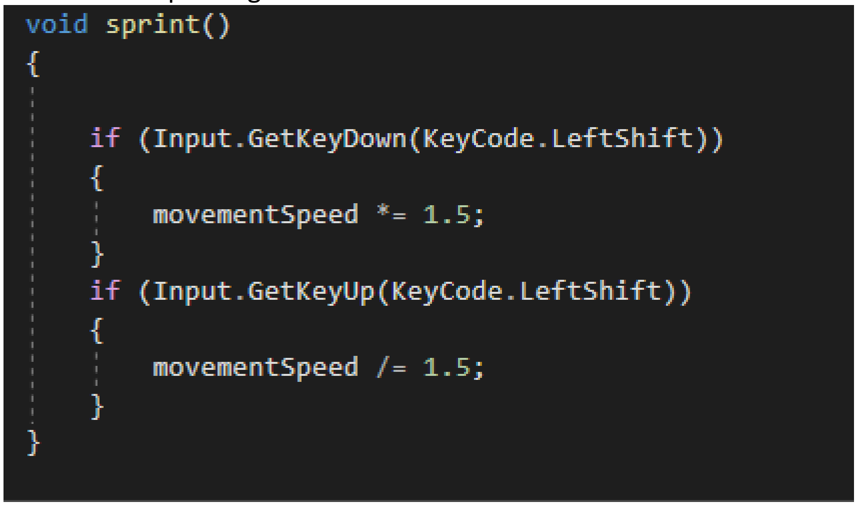
I ran into some errors with this section of the code as at first I was trying to declare the new vector3 as a vector2 as I saw no reason of the z axis being relevant as it is a 2 dimensional game, so I then tried to declare it as a vector3 and it worked fine as intended – left right movements and jumping - so for now it will stay as a vector3 and later I will look into whether it is possible to change that to a vector2.

The next section of the code is for sprinting, originally for the sprinting part of the code I intended to have it in a separate procedure in which I passed the movementSpeed variable over by reference however when testing, I managed to get the code with sprinting as a separate function to run without any compiler errors but the sprinting part of it didn't actually work.
My initial intention with this code was to have the float movementSpeed passed by reference over to a separate procedure for sprinting. This passing of the variable by reference worked though so I suspected there was something wrong with the if statements in the sprint procedure.



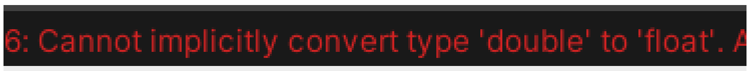
```
void horizontalMovement()
{
    float movementSpeed = 4;
    sprint(ref movementSpeed);
    float inputx = Input.GetAxis("Horizontal");
    transform.position += new Vector3(inputx, 0, 0) * movementSpeed * Time.deltaTime;
}

void sprint(ref float movementSpeed)
{

    if (Input.GetKeyDown(KeyCode.LeftShift))
    {
        movementSpeed *= 2;
    }
    if (Input.GetKeyUp(KeyCode.LeftShift))
    {
        movementSpeed /= 2;
    }
```

Looking at the if statements, they didn't seem like they had anything wrong with them so I instead tried something else…
Instead of having the movementSpeed variable declared locally in horizontalMovement and then passed into sprint, I just declared movementSpeed as a public variable instead, then I made sure the sprint class was called inside void Update to ensure it was updating every frame and upon testing, the code for sprinting worked fine.



```
void sprint()
{

    if (Input.GetKeyDown(KeyCode.LeftShift))
    {
        movementSpeed *= 1.5;
    }
    if (Input.GetKeyUp(KeyCode.LeftShift))
    {
        movementSpeed /= 1.5;
    }
}
```

When I tried to change the sprint modifier to a number which had a decimal in, I got a compiler error.



```
6: Cannot implicitly convert type 'double' to 'float'. A
```

This error was an easy fix however, all I needed to do was add an “f” to then end of the value to show that it is meant to be a float and not a double.

Here is a later video showing that the speed variable now works as a float:

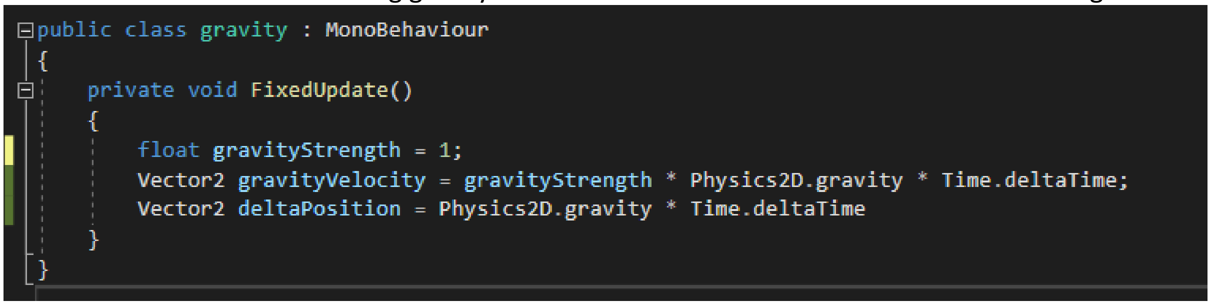
Access youtube video at timestamp 1:25 - https://youtu.be/5qqjR0d2YM8

speed variable working as a float.mkv

Having movementSpeed as a public variable will most likely make a lot of sense later on as many other parts of the code will most likely be needing to use it.
Something which I intend to do with the sprint class later on is to make the increase in speed after holding down the shift key gradual instead of instantaneous as it is now.

Gravity script

The gravity script is quite short and simple. It all takes place in a new procedure I created called void FixedUpdate. I am using FixedUpdate to run the code every physics frame to make the gravity as smooth as possible. Firstly, a float called gravityStrength is declared as 1 which is a value which I will later fine tune to make the games physics feel better for the user. The float gravityStrength is a variable which is going to be used as a multiplier on the physics2d gravity to make it weaker or stronger. Next, a vector2 variable called gravityVelocity is declared, gravityVelocity is equal to the result of gravityStrength * Physics2D.gravity where Physics2D.gravity is unity’s build in vector2 value for acceleration due to gravity which is (0,-9.8) by default. This is also multiplied by Time.deltaTime to ensure the changes in position are smoothed out over the frames.
At this stage the gravity script is all running and working however there is still one thing which is not working as intended, and that is the gravity strength variable as when I attempt to adjust the value held in gravityStrength no effect is given which will be important later on but early on it isn’t that relevant as there is still a working gravity. I can leave it for now and I will come back to fixing it later.



```
public class gravity : MonoBehaviour
{
    private void FixedUpdate()
    {
        float gravityStrength = 1;
        Vector2 gravityVelocity = gravityStrength * Physics2D.gravity * Time.deltaTime;
        Vector2 deltaPosition = Physics2D.gravity * Time.deltaTime
    }
}
```

## Iteration 2

To begin my second iteration of the project I first wanted to test how the moveable character sprite would behave when coming into contact with another body and to test wall collisions.

So I created a new sprite which was just a box and dragged it onto the scene. I then added a rigidbody2D which was set to dynamic, and a 2D box collider to the box sprite.

When I ran the game to test it the box and the character behaved exactly how I wanted them to as the character is able to push the box by walking into it and topple it over by jumping against it. However I did have one issue which I needed to fix. When the player ran into the box the collision was not solid, meaning when the player collided with the box the two sprites went slightly inside each other.

I tested this problem to the extreme by increasing the movement speed of the character to investigate how the speed would affect it. And I suspected, an increase in movement speed allowed the player to pass right through the box.

I tried changing the rigid body collision detection settings in the inspector in a number of ways such as having one of the sprites set to continuous (which is default) and another as discrete or both discrete and both continuous however I was still encountering the same bug.

Here is a later video of the problem:

Access youtube video at timestamp 0:03 - https://youtu.be/5qqjR0d2YM8

collision issue.mkv

In this video, if you look closely you can see the player not colliding with the box as expected.

Solution

- I am still yet to find a solution for this section

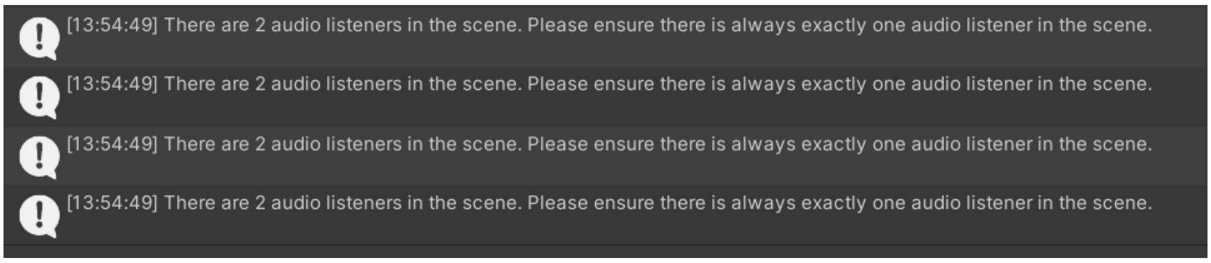
One of the first features I was willing to implement for the second iteration of my project was some menus. Referring to my requirements in my analysis section, There will need to be a menu screen when the game is first started this is so the game is more user friendly as the user will not be immediately put into “gameplay”.

So I created a simplistic new sprite to be the backdrop for the main menu and added this sprite to my sprites folder. I then created a new scene in the hierarchy tab and named this “mainMenu”, within the scene I created a canvas so I can lay out UI elements such as buttons which I would be using. Then within the canvas I created a panel which would be the backdrop for the main menu and named this “main menu” and set the source image to my main menu sprite which I added.

I then took a break to create a font for my project which I efficiently did by pixilating the default windows font “Calibri” and then editing this, this way I didn’t have to fully create a new font from scratch so I was able to create it with great time efficiency and with reference to my requirements in the Analysis section of this document. They state I must have a readable and clear font which is a target I followed when creating the font. They also state the font must be uniform across the project which is why I decided to make it before finalizing the menus.

The creation of the font is important as I needed to create 3 new sprites to be the buttons for the menus: “playGame”, “options” and “exitGame”. I then added these new sprites to the “main menu” panel within the “mainMenu” scene and used the inspector to align these three buttons proportionally. When creating these sprites I kept in mind size and made sure they were all the same size with use of the font. These sprites came out blurry at first so I switched their filter mode to “point (no filter)” to adjust them to normal. They also came out with a different aspect ratio they were made with so to fix this I just checked the “preserve aspect” box in the inspector for all three.

For now I only need one of these buttons to be functional which was the “playGame” button which would be the button which would navigate the scene from the main menu back to the “Sample Scene” scene which is the one which the actual game is currently held in. To achieve this, I needed to create a new c# script and named it “sceneChanger”. Within the new c# script I created a new public class which I named buttonPlay() accordingly and in this class was just one simple line of code which is “`SceneManager.LoadScene("SampleScene");`” This code simply just gets the scene manager to load the scene named “SampleScene” in the hierarchy. Then I saved the code and finally added it to the canvas in the “mainMenu” scene, from there I went into the inspector of the “playGame” sprite and went the the “on click” section where I would set the source to the canvas from the same scene and set the function to the buttonPlay() class. Upon testing the code ran as I expected it to and the scene changed upon using the button.



```
[13:54:49] There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.
[13:54:49] There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.
[13:54:49] There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.
[13:54:49] There are 2 audio listeners in the scene. Please ensure there is always exactly one audio listener in the scene.
```

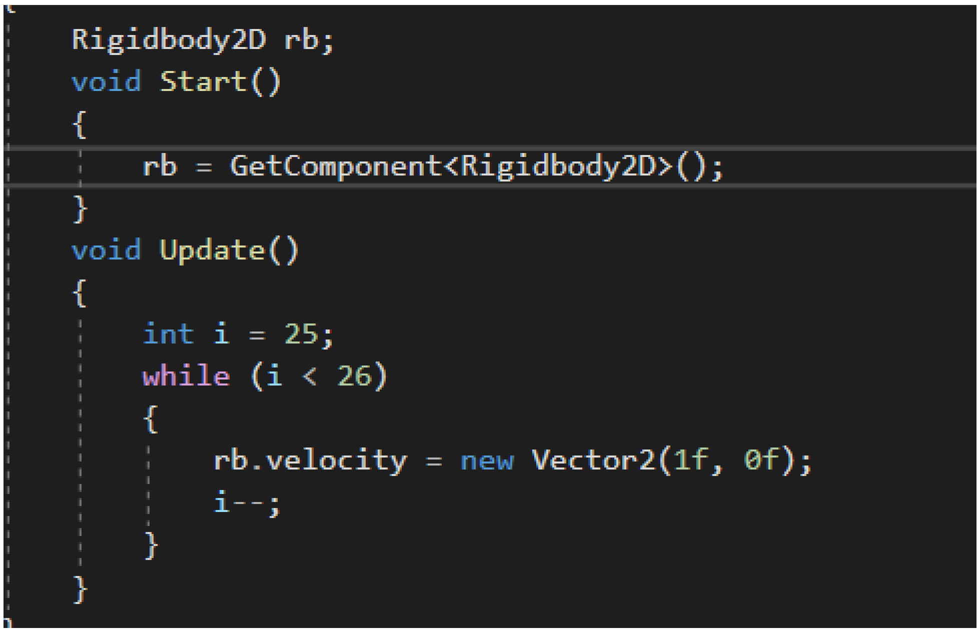
This is the only problem I had, upon running the game unity would repeatedly send this error message, warning me about 2 audio listeners in the scene. Removing this message was a simple fix as all I needed to do was uncheck the “audio listener” box in the inspector of the main camera in the “mainMenu” scence. However I wanted to check if the game would still play audio if I wanted it to as I will be adding audio and sound effects later which is a requirement I set for the project in the analysis section. So I downloaded the audio program audacity and generated a temporary rhythm track which only took a few minutes. I then exported this track as an mp3 file and added it to a new folder which I created in my project called “sounds” and then dragged this onto the main camera which upon testing the audio played as I had hoped.

Something also to note, I gave the button a highlighted colour and a pressed colour in the inspector to make it easier for the user to know the button is usable which is another requirement I set in the analysis section of this document.

## Iteration 3

For my third iteration I wanted to start out by adding basic hazard movement. For now all I needed to do was make a sprite repeatedly move left and right. So I started off by creating a prototype sprite and adding it to the scene. I then created a new c# script which I named “hazardMovement” which I first declared an integer variable as value 25 used two while loops to transform the position of the sprite where at the end of the loop “i” was decreased by 1 and when the value of “i” became negative the position would be transformed in the opposite direction until “i” reached negative 25 where it would then be reassigned back up to 25.

Then I needed to test this code so I attached the script to my temporary hazard sprite, this all worked with no errors up until I ran the program, where on every test the unity editor would stop responding, I hadn't yet encountered this issue with any other scripts and this issue also occurred when attaching the script to a different sprite so I suspected there was something in the code which was causing the program to crash. I started to attempt altering the code which I started out by incorporating the Rigidbody2D component into the code as I thought the issue may have been the code not referencing the rigid body at all. However, after altering the code to incorporate the rigid body, I was still encountering the same issue upon running it despite the IDE – visual studio telling me there is no issue found with the code.



```
Rigidbody2D rb;
void Start()
{
    rb = GetComponent<Rigidbody2D>();
}
void Update()
{
    int i = 25;
    while (i < 26)
    {
        rb.velocity = new Vector2(1f, 0f);
        i--;
    }
}
```

Another requirement I have which is due for me to add for my third iteration is a following camera, so I began by creating a new c# script and named it accordingly – "cameraFollow", I then attached this script to the camera within unity, in this script I created a new public gameobject variable called "player", which I would then assign with the character sprite in the unity editor. Then I went back into the script and went to the update class and set the position of the camera to transform accordingly to the changes of the players location on the x axis and the y axis where the z axis stays constant.

For the most part, camera movement worked as expected – aside from the fact that it can be a bit shaky, however I think incorporating a Time.deltaTime into the script somewhere to smoothen out the movement over the frame would be useful.

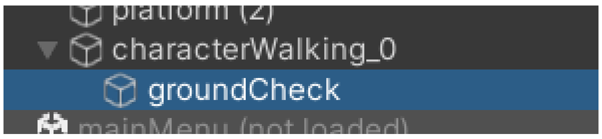
Next I have decided to finally incorporate a ground check into the project – which is a requirement I set for myself during my first iteration - as at this point the player can still continuously jump without having to touch the ground as this relates to my requirement I made earlier, that there shouldn't be any "exploits".

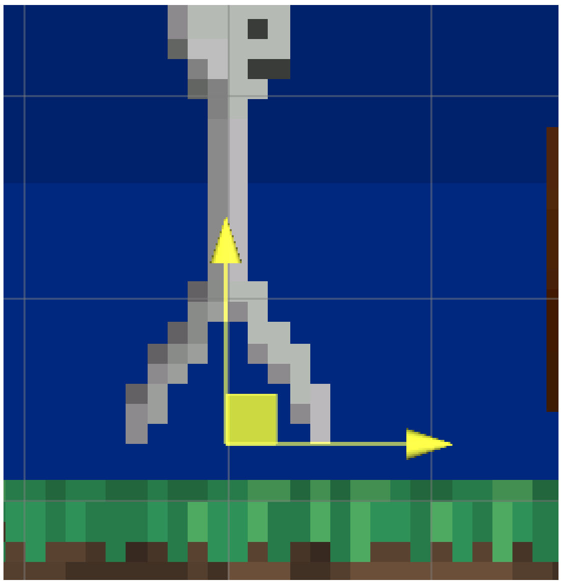
I did not know how to implement a ground check, however it was necessary for me to implement due to it being a fix to an "exploit" where my requirements state there must be no exploits.

I had to look online for a way to implement a ground check, I took to youtube and found a tutorial which demonstrates how to implement a ground check.

// start of tutorial code

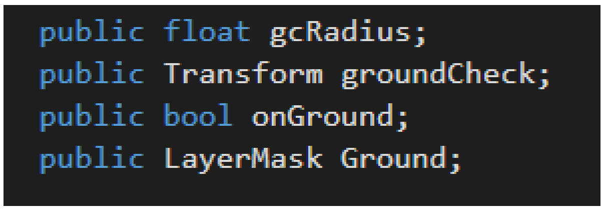
The tutorial guides me into creating an object which is a child of the character object, so I do this in the hierarchy and named the object "groundCheck".





I then used the move tool to move the object towards the bottom of the player as that's where the player will be touching the ground.

I then went into the movement script and declared some variables



```
public float gcRadius;
public Transform groundCheck;
public bool onGround;
public LayerMask Ground;
```

In the inspector I set the groundCheck variable to be equal to the groundCheck child object

I also set the radios of gcRadius

I also set Ground to be a new layer which I have created called "Ground" for layers for platform layers.

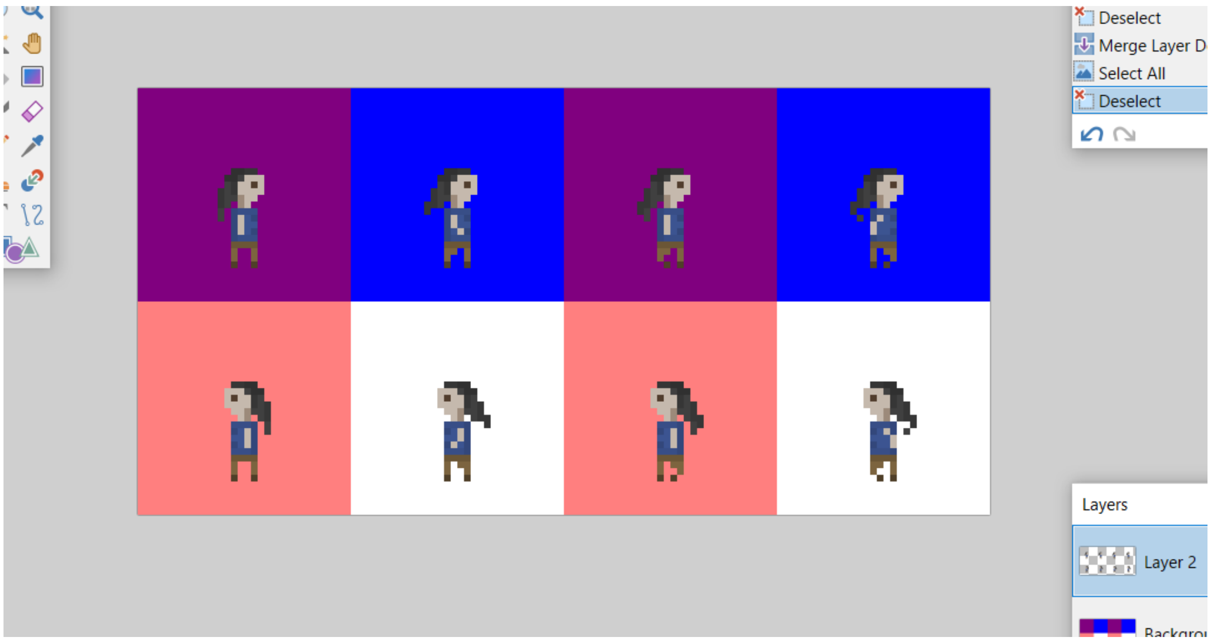
The tutorial showed me a ground check method which creates an invisible circle at the players feet which allows the player to jump if there is ground in the area of the circle

// end of tutorial code

I then used C#s and operator which is "&&" to set the condition for jumping (where both the jump botton and ground check must both be true)

I then needed to add animations for the players movement. To start off with this I created a new sprite sheet for my player using the application paint.net because, as I stated previously the previous one was just a place holder.

Here is the spritesheet



(Note that the coloured squares are just there to distinguish the difference between the different cells – they would normally be transparent)

As a design choice, I decided to make the character in the game female as I think females are underrepresented in gaming – specifically as protagonists.

## Evaluation and final testing

### Final testing

The ground check now works as intended like the tutorial used shows.

Here is the video to exemplify that

access youtube video at timestamp 0:32 - https://youtu.be/5qqjR0d2YM8

ground check working.mkv

The walking left and right and sprint inputs work as they should,

Heres a video to show me applying and unapplying the sprint key.

access youtube video at timestamp 1:47 - https://youtu.be/5qqjR0d2YM8

sprinting on and off.mkv

Gravity works as expected and so does the jumping function for the player.

Here's the video showing jumping and gravity works, I have also used a crate which I have added to the game to test collisions and gravity with the player.

access youtube video at timestamp 1:00 - https://youtu.be/5qqjR0d2YM8

jumping and gravity.mkv

Collisions are a little broken in the game as if the player is moving at a very high speed they can pass through a solid object, however I don't consider this to be "game breaking" as they cannot reach such a high speed within the game.

I spent hours trying to fix this collision issue and couldn't find a way to fix it however it's something I want to look into rectifying in the future.

The camera follows the player perfectly as I wanted it to – most the videos in the documentation show it working but here is another to show it.

access youtube video at timestamp 1:00 - https://youtu.be/5qqjR0d2YM8

camera following working.mkv

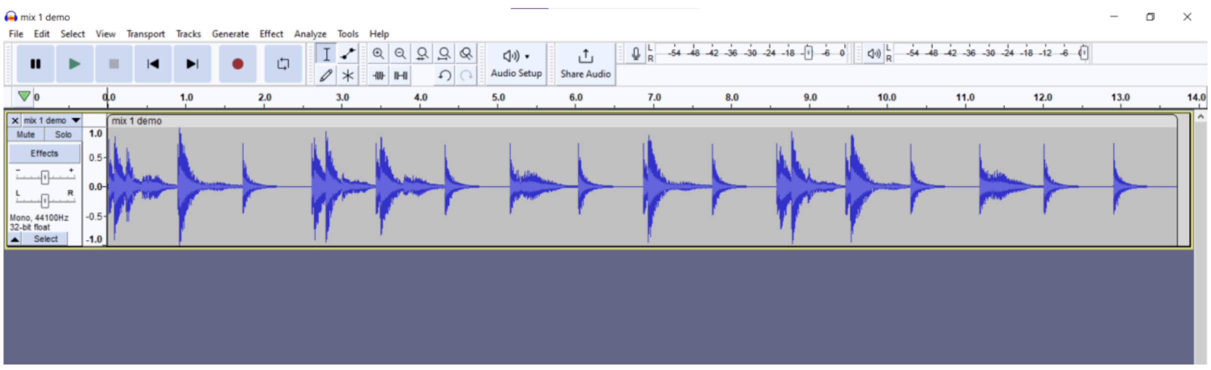
## Evaluation

How well have I met my measureable success criteria?

Music

The music added to the game should be quiet and an ambient as it is only there to create an atmosphere for the game and not something too extreme like pop/rock music etc.

I have used the audio software, audacity to create a short piece of music which loops smoothly and complements the game which isn't too distracting.



I selected loop when adding it to the camera in unity, to make the mix loop.

Sound effects

The sound effects that are added into the game should be relevant to events which happen in game – they should not just occur randomly – sound effects should also sound according to what they are representing. If the player dies the sound effect should attempt to sound negative.

Due to time limits, I have not had the chance to add as many sound effects as I would have liked to, however I did manage to add a sound effect for footsteps when the character is walking.

There must also be a way to alter the volume of the music and sound effects (separately) – should be located within the options menu.

I have implemented an options menu, however the only option on it is to be able to change the volume of the music in the game – not the sound effects.

Animations

The animations in game should be relevant to the action that they are trying to represent. If the character starts walking, then the walking animation should be played and the character should face the direction accordingly.

I have managed to create animations for when the character is walking so this measureable success criteria has been met.

Here's an example of the walking animation in action

access youtube video at timestamp 2:10 - https://youtu.be/5qqjR0d2YM8

walking animation.mkv

Sprites should also be relevant to what they are trying to represent. If there is a hazard in game, it must be easily identifiable as a hazard even by someone who has never played the game before. Ground sprites should also all fit together nicely with no uneven lines and textures between them.

The hazard sprite which I have created is a huge spider – which could potentially be identifiable as a hazard from first glance as it is "unnatural"

Menus

The menus that are presented in the game should be easily readable and must stand out from the background by using different colours to ensure everyone has a seamless time reading them. Menus such as the "exit game" function can just be represented by a red cross surrounded by a box in the corner of the screen. The menus should all use the same two to three colours and a uniform font.

When the game is first loaded up, there is a main menu, which has three buttons (play game, options and exit game)

Heres an example of the main menu working

access youtube video at timestamp 0:00 - https://youtu.be/5qqjR0d2YM8

main menu working.mkv

Another requirement which I also set for myself was, there must be a visual cue that the buttons have been pressed, which the video above shows when I click playGame – it lights up green

Logo

Like the menus, the logo should be easily readable and should just be the name of the game. The logo should use two to three colours which match with the colours on the menus for the game.

I have not yet created a logo for the game, due to putting most of my time into coding and documenting the project, however creating a logo is something which I intend to do in the future.

Exploits

There should not be any exploits which allow the user to be able to "cheat" the game. This is a requirement which I created during the first iteration of my development due to my realisation that the player could infinitely jump – even if they weren't touching the ground.

I rectified this defect during the third iteration of my development where I implemented a "ground check" which created a condition where the player could only jump if the player is "grounded".

The issue was originally like so:

access youtube video at timestamp 0:15 - https://youtu.be/5qqjR0d2YM8

ground check not working.mkv

However this is what the program looked like after I implemented – I cannot jump mid air

access youtube video at timestamp 0:32 - https://youtu.be/5qqjR0d2YM8

ground check working.mkv

The existing solution which I came up with in my project analysis "super mario bros." is the existing solution I picked which closely resembles my project the most, this is because this existing solution shares similar functions as my game such as a held button for sprinting and jumping. However my project differs from this solution due to aspects of the physics and camera being different.

Bibliography

Geometry dash image – https://geometry-dash.fandom.com/wiki/Stereo_Madness

Super Mario Bros image - https://www.mariowiki.com/World_1-1_(Super_Mario_Bros.)

Ground check Youtube tutorial - https://youtu.be/ld5QQm-ADdE

Youtube video of evidence for this project by me - https://youtu.be/5qqjR0d2YM8

## Another iteration

I've decided to further work on the project after the submission for A-Level Computer Science. I will continue the solution with a new iteration

With this iteration of the project I would like to introduce levels. This iteration will involve the completion of the first level within the game.

Requirements

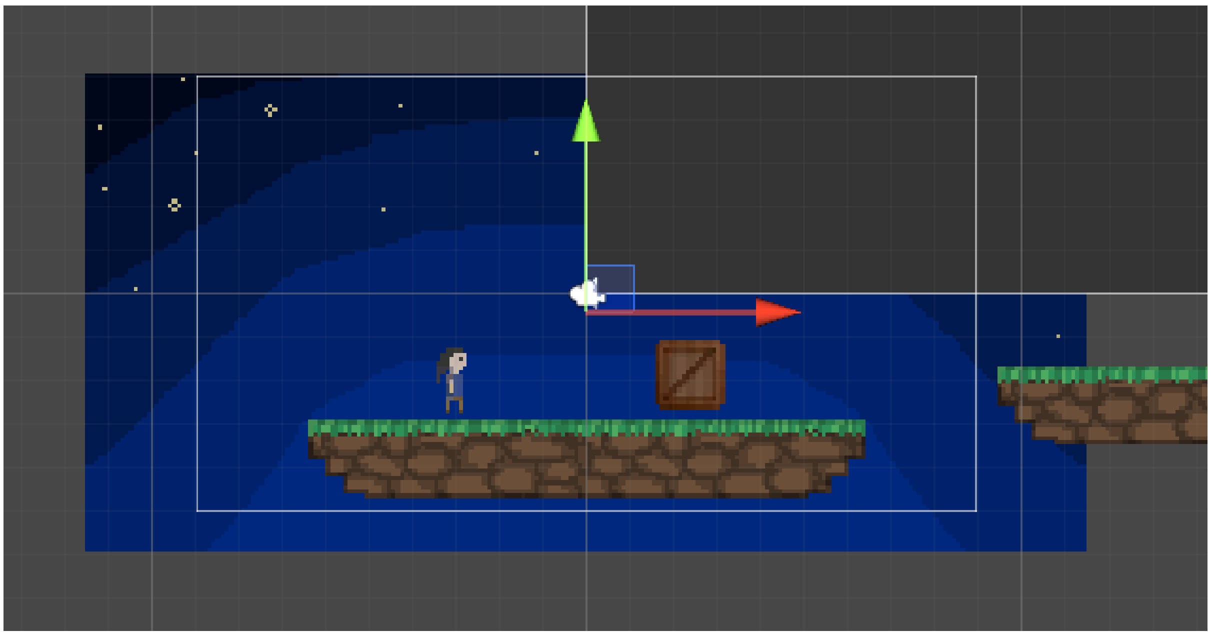
The level will play out from left to right in a "classic platformer" style.

At the end of the level there will be a clear goal which will allow progression onto the next level.

After the goal is reached a menu will pull up asking the player if they want to continue to the next level.

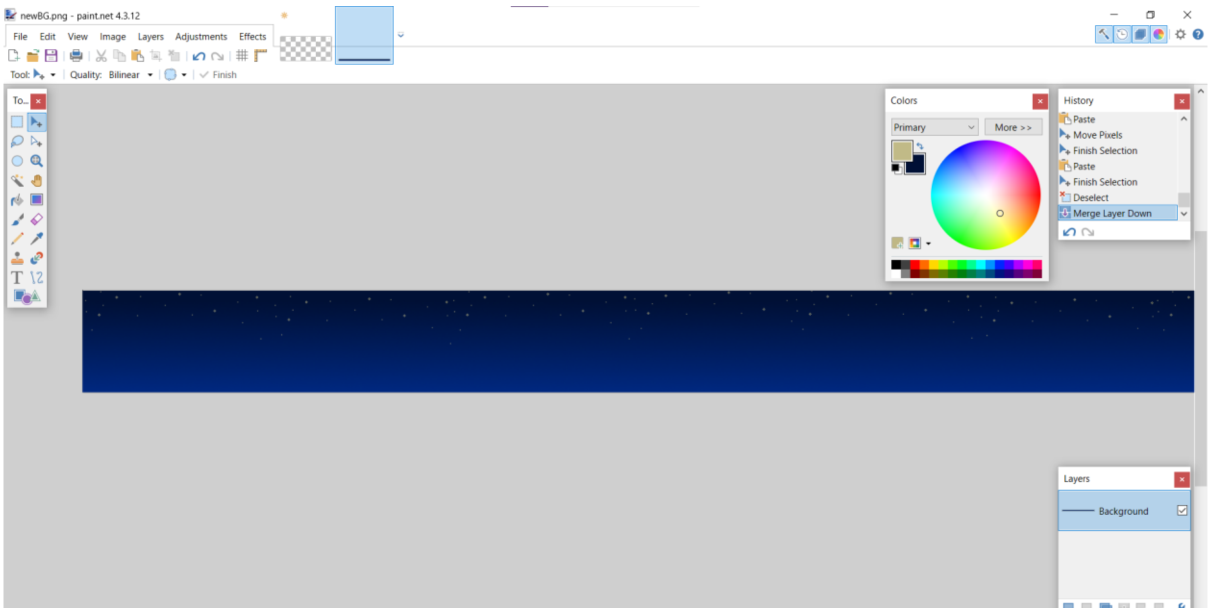
Designing the next iteration

The

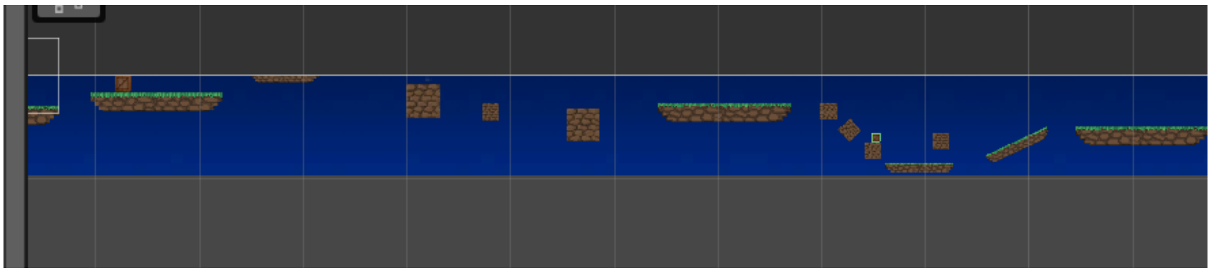


Firstly, I need a new background as this one is outdated and only fits the area of the start of the level.

The new background should be long so that it always fills the camera throughout the whole level.



I quickly made a new background by reusing assets from the old one. This one is much longer and slightly wider, it should work well enough.

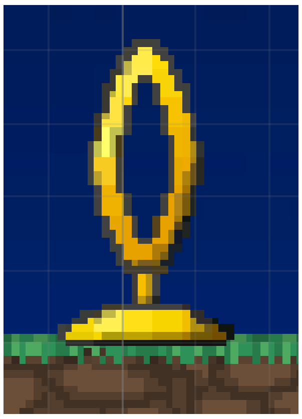


I designed the level within the unity editor with static blocks and platforms and a few dynamic crates, just simple assets I already had which can be changed later on.

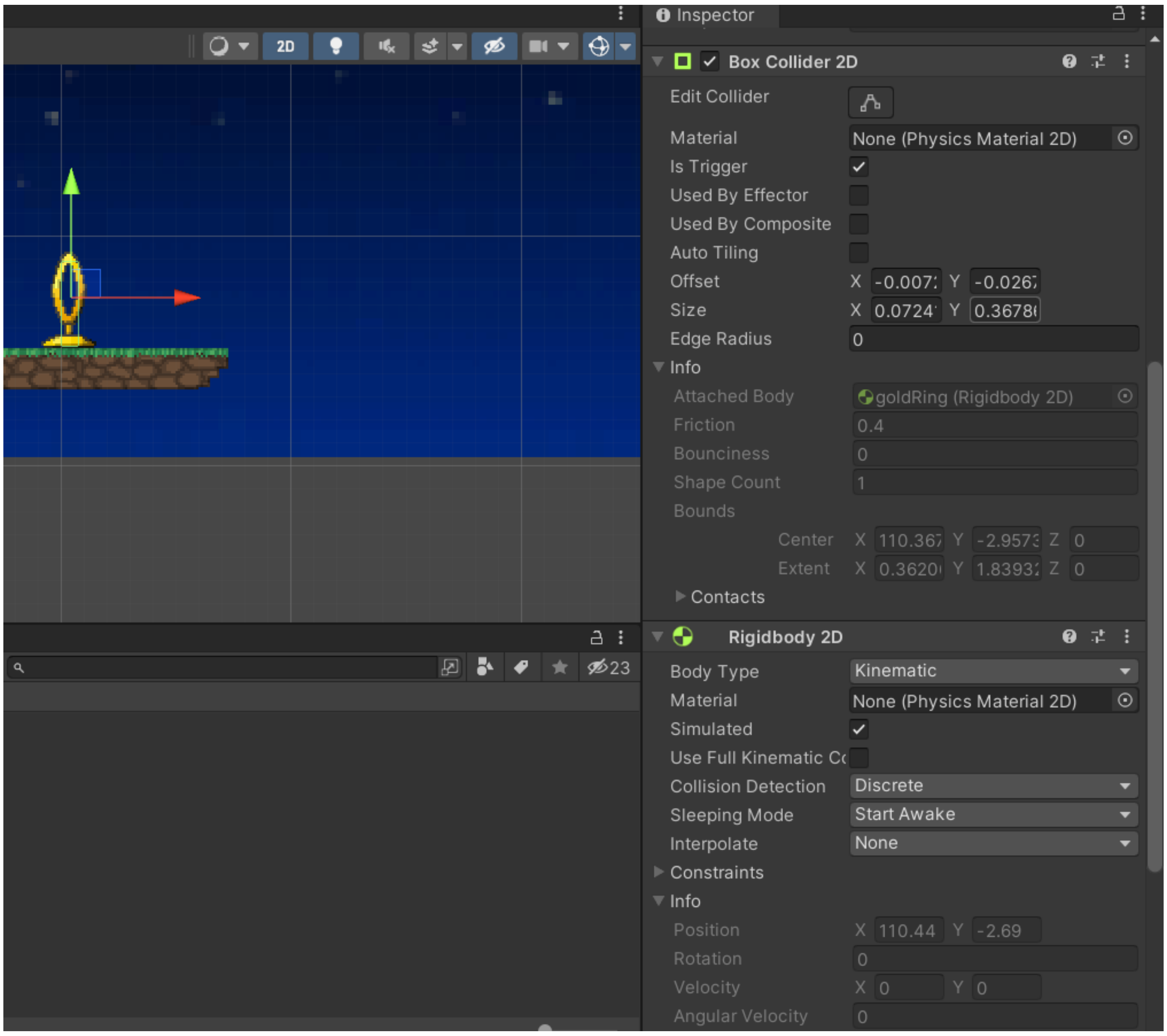
I now needed a level end sprite.

I was thinking to do some sort of gold hoop for the end of the level.

I quite liked this idea and I decided to do it.



So I created that sprite and placed it at the end of the level.

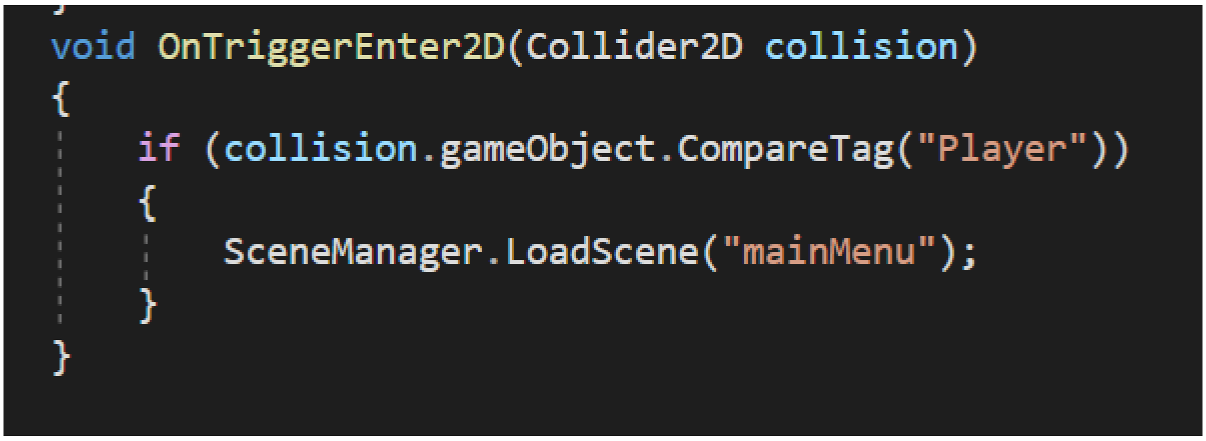


```
Box Collider 2D
Edit Collider
Material                 None (Physics Material 2D)
Is Trigger               ✓
Used By Effector
Used By Composite
Auto Tiling
Offset                   X -0.007:  Y -0.026;
Size                     X 0.0724'  Y 0.3678(
Edge Radius              0
Info
  Attached Body          goldRing (Rigidbody 2D)
  Friction               0.4
  Bounciness             0
  Shape Count            1
  Bounds
         Center          X 110.36;  Y -2.957:  Z 0
         Extent          X 0.3620(  Y 1.8393;  Z 0
  Contacts
Rigidbody 2D
Body Type                Kinematic
Material                 None (Physics Material 2D)
Simulated                ✓
Use Full Kinematic Cc
Collision Detection      Discrete
Sleeping Mode            Start Awake
Interpolate              None
Constraints
Info
  Position               X 110.44   Y -2.69
  Rotation               0
  Velocity               X 0        Y 0
  Angular Velocity       0
```

I dragged the sprite onto the scene of the level and gave it a box collider and rigid body, I later checked the “Is Trigger” box in the box collider section, this is useful for my next part of the code which will allow the sprite to trigger the scene change which ends the level.

I then created a new c# script and attached it to the level ending sprite. The script was called levelEnd.

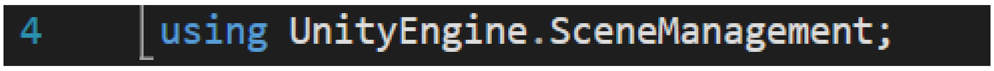
Within this script I created a function which used the OnTriggerEnter event which means the trigger body would cause something to happen. In this case the function contained an if statement which would change the scene upon the trigger.



```
void OnTriggerEnter2D(Collider2D collision)
{
    if (collision.gameObject.CompareTag("Player"))
    {
        SceneManager.LoadScene("mainMenu");
    }
}
```

Here is the code. It currently loads the main menu scene as a placeholder since there is no second level.

For this section of code to work I also had to have the line using UnityEngine.SceneManagement; at the top, just like when I created the scene changer script.



```
using UnityEngine.SceneManagement;
```