Computer Science Project

Contents

[Introduction 2](#_Toc60796808)

[Analysis 2](#_Toc60796809)

[Computational Approach 2](#_Toc60796810)

[Pattern Recognition 2](#_Toc60796811)

[Stakeholders 2](#_Toc60796812)

[Existing Solutions 3](#_Toc60796813)

[Celeste 3](#_Toc60796814)

[Limbo 4](#_Toc60796815)

[Cuphead 5](#_Toc60796816)

[Essential Features 6](#_Toc60796817)

[Hardware and Software Requirements 6](#_Toc60796818)

[Measurable Success Criteria 6](#_Toc60796819)

[Design 7](#_Toc60796820)

[Problem Decomposition 7](#_Toc60796821)

[Problem structure 7](#_Toc60796822)

[Player Movement 7](#_Toc60796823)

[Algorithms 7](#_Toc60796824)

[Usability Features 7](#_Toc60796825)

[Data Dictionary 7](#_Toc60796826)

[Test Data to be used 7](#_Toc60796827)

[Post development additional data 7](#_Toc60796828)

[Development 7](#_Toc60796829)

[Development log 7](#_Toc60796830)

[Testing for Evaluation 9](#_Toc60796831)

[The original test plan 9](#_Toc60796832)

[Testing process 9](#_Toc60796833)

[Evaluation 9](#_Toc60796834)

[Test evidence 9](#_Toc60796835)

[How to develop unmet criteria 9](#_Toc60796836)

[Evaluation of usability features 9](#_Toc60796837)

[Maintenance Issues 9](#_Toc60796838)

[Potential improvements 9](#_Toc60796839)

# Introduction

The video game I want to create will be designed to tackle boredom and electrify the player’s brain with mind bending puzzles and challenging enemies. This game aims to keep players thinking about ways to overcome the obstacles before them. It will encourage lateral thinking and improve the players puzzle – solving ability. I believe there is a need for a casual puzzle – solving game, where the player can leave the game at any point. This aims to be the kind of game that non-gamers can enjoy. I want music to be a very important part of this game. It should be purely instrumental and pleasing to the player’s ears. I researched a very popular platforming game and its reviews, and I understood that what most people enjoyed aside from the story and mechanics was the soundtrack. They all gave positive reviews about the sounds and music. Therefore, I would also like my game to include a smooth soundtrack. The music could also indicate a difference in a stage, death, respawn, timed challenges and victory.

# Analysis

## Computational Approach

This game tackles boredom using computers in ways that would be far too difficult or impossible in reality. For instance, tackling boredom without computers would involve things like board games. These would not always be ideal because they lack the fast-paced action that most people long for. However, there are sports that could give a fast-paced feeling, but they require many players to play.

Computers can give the fast-paced action that board games lack with increasing difficulty and different level designs. Computers can never get tired or bored of running the same game for long periods of time unlike other human players who could easily be tired of playing the same board game for long periods of time.

Most board games require other players to play, which means if there are no other players the game cannot be played. However, using a computer would allow the game to be played without any other human player in the sense that the computer would handle the enemies, hazards and so on. Progress cannot be saved in board games or single player card games. Although, they could be carefully packed up to retain the data, but it would not be ideal. Computers can save data and the player can resume to where they stopped with ease. Therefore, a computer would be more ideal to solve this problem.

### Pattern Recognition

My game will make use of pattern recognition as part of the computational approach. The enemies of the game will repeat certain actions in the game and their primary aim is to prevent the progression of the player. This will require the enemies to be coded in a certain way that would be replicated for most of the other enemies and certain hazards, such as falling platforms and retracting spikes. However, this would be very tough in board games because everything would have to be set up manually. As well as certain sports. To be able to create the same situation in a sports game, the players would have to return to the same position repeatedly. This would be tiresome and hence why the use of computers would be better to solve this problem.

## Stakeholders

The stakeholders are the people that are invested into my project. These people would include age 10 and above, male players. As well as females at age 18 and above. The stakeholders cannot be below the age of 10 because I believe the puzzles would prove too difficult for their minds. And for females below the age of 18, I believe they are not interested in video games at all.

The younger ages (10-17) would be more attracted to the platforming style of the game, the cartoon design would also appeal to them greatly, and they will have a greater sense of achievement when they complete a level. And it does not feature any excessive violence. It would also have many levels which would keep them invested for a long time.

The older ages (18 and above) would be attracted to the platforming style of the game as well, but they would also appreciate the soundtrack that would be dynamic and change depending on the situation. They would also like how the puzzles make them think outside the box for different solutions. My research in reviews of games has led me to believe that the soundtrack of a game is a very important part of people's enjoyment. So, I believe the soundtrack will appeal more to the older stakeholders. It would also meet the needs of the 18 and above aged male stakeholders because of its longevity, due to its many levels. It satisfies their need for a non-competitive game that can be played, dropped and returned to at any time without the stress of losing their rank.

The game would appeal to the females aged 18 and above because it gives them a casual game for them to play in their free time when they are not at school or work. They might find the sprite character cute and enjoy the simplicity of the puzzle solving mechanics. As it is not a gory game that portrays women in a bad way, more females would be interested in it. Candy crush is a puzzle-solving game, and it is very popular with women. Therefore, I believe this platforming/ puzzle game will be able to resonate with them as well. It also meets their need for a source of entertainment outside of social media. They also have the need for non-violent games that are not complicated and easy to get immersed into.

## Existing Solutions

There are already some solutions to this problem of boredom that involves platforming games. These games are very well made, and they tackle boredom and bring entertainment.

### Celeste

Firstly, a game called ‘Celeste’. This is a 2D platforming game that released in 2018. This game was credited for its simple mechanics, relatable story, amazing soundtrack, well-conceived level design and hidden secrets. Although it had simple mechanics, a lot of trial and error would still be required to steadily progress the game. The player had to traverse the mountain by completing different levels and would be hindered by an enemy which would occasionally lead to boss battles. A lot of people that played it gave stellar reviews. They mainly mentioned how the soundtrack of the game was a major factor in the players enjoyment. The respawn time is very short, which motivates players to keep trying, even when they are stuck. I would like to include the simple mechanics, engaging audio, good level design and a very short respawn time. I believe having these features in my game would improve the players experience greatly. But I think having a story would prove too difficult for a one-man project. The hidden secrets would be a good touch, however, the level of detain that would go into designing different levels with secret rooms and collectibles would be much too tedious because it would take too much time to complete. This game is also multi-platform, meaning it can be played on different platforms. This is highly beneficial because this allows a vast number of people on different devices and platforms to be able to access the same software.

This is a screenshot of a part of a level in Celeste, where the player must carefully navigate through the level by avoiding the spikes and getting to the other side. Although, it may take many tries to complete this area but the quick respawn time and rich background music will encourage the player to keep trying. This is the kind of design approach I want to take with levels in my game.

This is a screenshot of a level with a special collectible, which in this case is a strawberry. The player would have to take alternative routes or find false walls to find these collectibles. This method rewards the player for exploring more and going the extra mile to find these collectibles.

### Limbo

Secondly, there is also a game called ‘Limbo’. Limbo is a 2D puzzle-platforming game that was released in 2010. The player played as a nameless boy who is in search for his sister. Limbo is a much darker game than Celeste. The game used a black and white aesthetic and did not use any background music, only sounds of footsteps, hazards, interactable objects and deaths. They used this to create an eerie atmosphere usually associated with horror games. This game included a physics system that controls environmental objects and the player character. This would cause the player to think about how physics would affect the obstacle they are trying to overcome. The game was credited for its puzzle design, simplicity of controls and its graphical and audio presentation. The games controls consist of two simple actions, jumping and grabbing objects. Limbo is a game that expects you to fail before finding the correct solution, they used a style of play which the developers called ‘trial and death’. They also use gruesome death animations to make players avoid unworkable solutions. Reviewers praised the games sound effects and how it was critical to the games impact. They also praised the games art direction and how it created an unsettling and eerie environment. I would like to include the physics aspect of this game as well as the grabbing, pushing and pulling mechanics. The game also makes sure that solutions to the puzzles are varied and the player would have to think to solve different puzzles and prevents the player from becoming too familiar to one solution. However, I think the art direction of this game would not appeal to some of my stakeholders as it is dark and gloomy. I also believe the gruesome animations used for the characters death will be bad for some of the stakeholders. I would also like my game to have varied puzzles that prevent the player from becoming too accustomed to the same solution.

This is an image of the grabbing and pushing mechanic of the game. It also shows the black and white aesthetic of the game. It also portrays the characters as silhouettes which adds to its spooky vibe.

This is a screenshot of a player solving a puzzle by stacking boxes to make the jump to avoid the hazard. The crane machines at the top of the screen drop the boxes and one of the boxes would be dropped on the other. The first box would have to be dragged to the correct position for the second box to be dropped on it. This is a puzzle that can only be achieved using the dragging mechanic.

### Cuphead

Lastly, a game called ‘Cuphead’. Cuphead is a run and gun game inspired by the animation style used in cartoons in the 1930s. The game sees the player play as one of two characters, Cuphead or Mugman or both in multiplayer. You play through several levels, fighting multiple bosses and there are some run and gun levels where the player must traverse through each level while protecting themselves from enemies and hazards. The game was credited for its art style and challenging difficulty. Reviewers also praised the jazzy soundtrack. They went on to say that every boss fight had an ideal strategy, and it could be uncovered through trial and error. Cuphead uses jumping, shooting, dashing and parrying mechanics. Cuphead also featured flying levels where the player’s character would be piloting a plane and fight bosses in that manner. I do not want my game to be a shooter, therefore I will be ignoring the shooting aspect. I would rather not have boss battles in my games because I want it to be a puzzle solving game with occasional enemies that are blocking your path. I like the idea that different bosses have different strategies to beat and I would like to link that idea to different puzzles having different solutions, as mentioned when I talked about Limbo. I also think flying levels would be a fun mix for the game’s formula but considering that I am the only person working on this project I believe having flying levels as well as ground levels would be too tasking. And I believe it would be very difficult to create a puzzle level when the player is flying in the sky in the sky. Although, I do not want the boss battle aspect in my game, the run and gun aspect would be a good fit for my game but without the gunning because this puzzle game should have a bit of action to draw in potential action loving players.

This is a screenshot of a run and gun level in Cuphead including enemies. Since in my game there will be no shooting, the ideal way to survive enemies would be to dodge them. Here there are enemies rolling down and a woodpecker bird is creating an obstacle. The player would have to correctly time their movement to pass this obstacle.

## Essential Features

The essential feature of my game is to succeed at being platforming- puzzle game that conquers boredom and keeps players coming back.

The game must have good sound- The background music that are used for different groups of levels must be soothing and unique to each group. The sounds for death and respawning should be polished and fit well with the game's aesthetic.

It must have puzzles that are both challenging and fun to complete- It is important that the game makes the player have a sense of accomplishment after each level they complete and at the same time bring entertainment while they attempt to complete it. This is a very crucial feature of the game that must be achieved for the game to succeed.

It must have low computer hardware requirements-This is important because not every person has high end PCs that are built to run heavy programs. Therefore, having low requirements will allow a wider range of users to be able to access the program.

The game must have simple mechanics- Through my research, I have discovered that simple mechanics has been a recurring item of praise for platforming games. For my game to succeed it must follow suit. Also, having a platforming game with mechanics that are hard to learn would not be ideal for most of my stakeholders. In conclusion, simple mechanics will be the best option for this game.

## Hardware and Software Requirements

* RAM:
* OS:
* FREE DISK SPACE:
* CPU:
* GRAPHICS:

\*Insert reasons for these requirements here\*

## Measurable Success Criteria

I want 80% of the people that play the game to give feedback that it was fun, and they enjoyed it. I will ask an equal number of males and females that fall within the category of my stakeholders. I would also want to receive reports that the game had good sounds, challenging puzzles and simple mechanics. If I can achieve these set criteria, then I believe this project would be a success.

# Design

## Problem Decomposition

Before I can begin development of my project, I need to acquire the necessary skills that will help me produce the project that I aspire to create. To do this, my Computer Science teacher has referred me to an online training course website called “udemy.com”. This site contains many different courses that teach you about many different things, from game development to web development and so on. I have enrolled in a course that teaches about Unity and C#. Since I want to make my game in Unity, this course will help me develop the skills to do so. I enrolled in a course on how to make 2D games and I learned many different things that would aid me in my development process. I was originally going to go for a 2D platforming game but now I have decided to change my perspective slightly to a much more grounded puzzle game in which there is no jumping, and the levels are more of finding the right combination of moves to reach the goal. I was inspired to take this approach from a recent puzzle game that I played called ‘Helltaker’. It has a very enjoyable formula that I would like in my game as well. So, I imagine this game as a top down 2D game where the player can only move one step at a time either up, down, left or right.

## Problem structure

I am going to break up my design process into three parts. Player movement, level design and user interface. This will prove more efficient as the problems are clear and split up rather than it being one big obstacle.

Decomposition Diagram:

### Player Movement

Here, I want my player to start at one side of the screen where they would have to navigate to the opposite side while traversing through a maze. The player should only be able to move one step at a time and they should not be able to move in any diagonal direction. The player is also bound by the walls at the edges of the screen and the blocks that make up the maze. This is so the player will not be able to move through obstacles which would defeat the point of the game. In addition to the moving restrictions, I want the player to have a certain number of moves they can make for each run. This way, the player would not only think about reaching the goal but the most optimal way to reach the goal using the least number of moves. Every time a player exhausts all their moves, they are returned to the starting point automatically while refilling their move counter. For convenience, if the player wishes to retry the level, they will be able to press the ‘R’ key which would have the same effect as exhausting all moves. This will make it easier for the player to give it another go without having to see a game over screen or wait for a loading screen.

### Level Design

I want my levels to comprise of blocks arranged in a particular order that requires the player not to use up all their moves. In this way I can allocate certain grades to players depending on how many moves they have left. For example, level 1 will give the player 20 moves. Realistically, you can beat the level in 15. But the player may not have found the optimal route, so they use 19 moves. The player will be awarded with a low-grade because there are better routes they could have taken. This also creates replay value because the player would be motivated to find the routes to the goal that consume the least number of moves. If the player manages to complete the maze in 15 moves, then they are awarded with the highest grade.

### User Interface

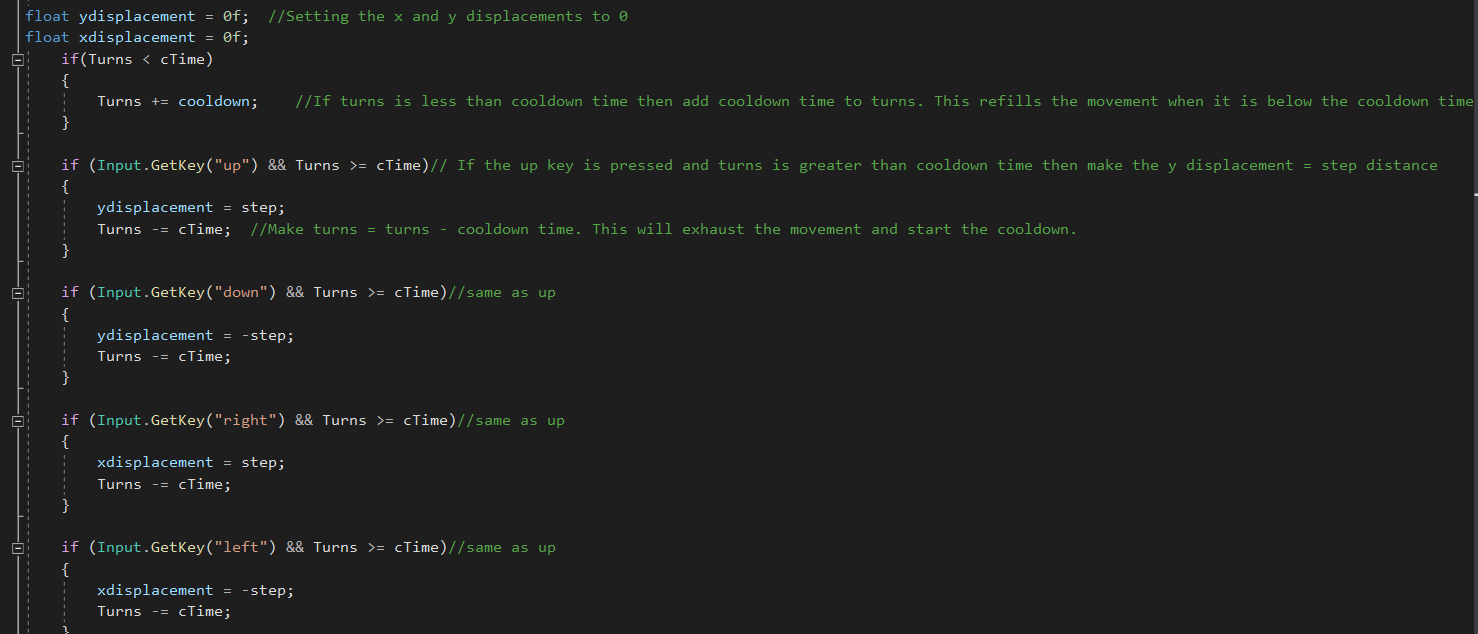
The UI will consist of a simple menu where the user will select the level in which they decide to play which vary in difficulty. Once a level has been completed there will be a victory screen that displays the player’s grade for that level. To overcome the complexity hurdle, I plan to implement a system that displays the highest scores for that specific level that has been completed. So, the highest scores will be saved to a list and displayed as a table when the level has been completed.

## Algorithms

### Player Movement

I have an algorithm for how my player moves one step at a time. I decided to create a cooldown system for the player’s movement. I created 3 new variables:

* private float Turns = 1f
* public float step = 0.5f
* public float ctime = 0.5f
* Float cooldown = Time.deltaTime



This pseudocode defines how the variables stated above will be used in the cooldown system. ‘Turns’ variable is meant to signify when the player presses a movement key. The ‘step’ variable shows the distance that the player moves per step. ‘ctime’ stands for cooldown time and it is the amount of time that must be elapsed before the player is allowed to move again. I will set cooldown time to be in real-time using ‘Time.deltaTime’ as this uses the system clock to calculate the change in time. From there I will use if statements to create the decisions needed for the movement system.

### Level Complete Checking Algorithm

INSERT INFO HERE

## Usability Features

I want my program to be simple to use and easy for anyone to pick up. The only input needed from the player would be for the player movement and to restart.

The movement controls consist of only the arrow keys. These are simple controls that anyone would be able to easily use, learn and remember. These are very important things to consider when creating a program.

There will also be a restart feature which would improve the player’s control while using the program. They will be able to restart their progress from any point on the maze. There will also be a counter displayed on the screen which keeps track of the number of moves the player has left and it reduces by 1 every time the player moves.

This gives them a visual representation of how they are doing on the level and depending on that, they can retry before they complete the level to get a better route.

## Data Dictionary

|  |  |  |
| --- | --- | --- |
| Variable Name | Data Type | Description |
| movespeed | float | This describes how fast the player will move when it is translating |
| ydisplacement | float | I set this to zero at the start because I will add variables to it later on and it serves as the distance moved in the y axis. |
| xdisplacement | float | I also set this to zero because of the same reason. |
| Turns | float | This works in tandem with the cooldown. |
| step | float | This is the distance that the player moves for each step it takes. |
| cTime | float | This stands for cooldown time. It is the amount of time that must be elapsed for the player to move after taking a step |
| Cooldown | float | This is set to ‘Time.deltaTime’ because it is used to refill the movement cooldown |

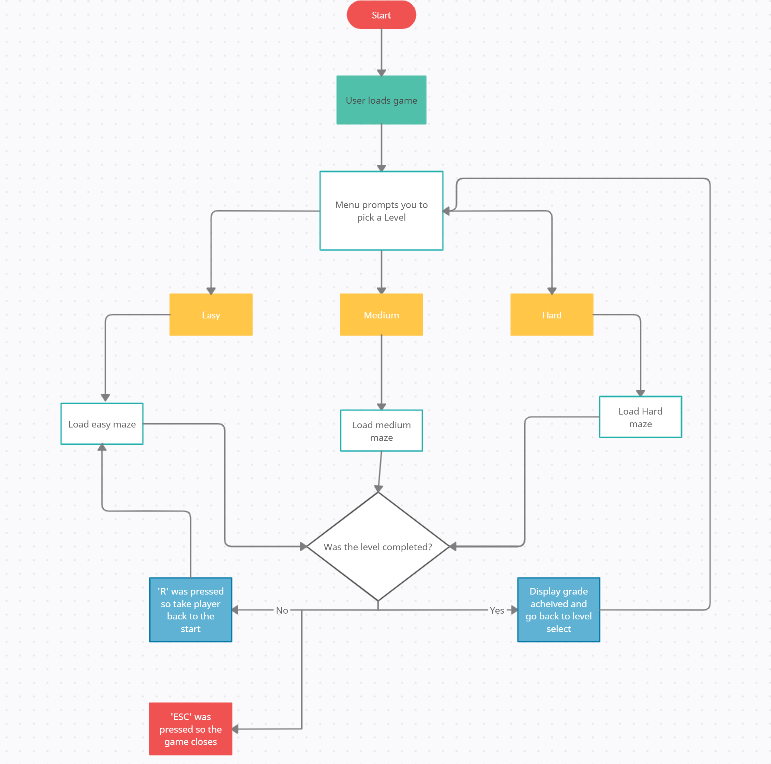
## Test Data to be used

These are some of the things I will be testing during the development process.

|  |  |  |
| --- | --- | --- |
| Test | Expected Result | Test Data |
| Press any one of the arrow keys to check the player’s movement | The game object that represents the player moves by a certain distance in the direction of the key that was pressed |  |
| Hold down any one of the arrow keys to check if the player moves one step at a time as intended | The player only moves one step at a time and has to wait for the set cooldown time before it is able to move again |  |
| Press the arrow keys in the diagonal position. | The player should not be able to move in diagonal direction so the player either moves up or down depending on the keys that are pressed. |  |
| Press the ‘R’ key | This should reset the level and take the player back to its starting position as well as reset the move counter. |  |
| Complete a level | A menu should pop up showing your grade depending on your performance. |  |
| Attempt to move the player through an obstacle | The player should stay at the same place because the obstacles should have physics attached to them. |  |
| Make the player move in any direction | The counter that should be displayed on the screen should reduce by the number of steps the player takes |  |

## Post development additional data

I would like to explore the realm of randomly generated mazes. It sounds like an interesting concept and it would be a great help when trying to come up with level ideas. However, I am going to include a move counter for each level and it would be impossible to create a movement counter for a random maze, therefore, I have discarded the idea. I am aiming to engage the player’s mind so I will add audio to the game consisting of sound effects for when the player moves and general background music for the levels. This is to give the player a relaxed mind while they think of their next move.

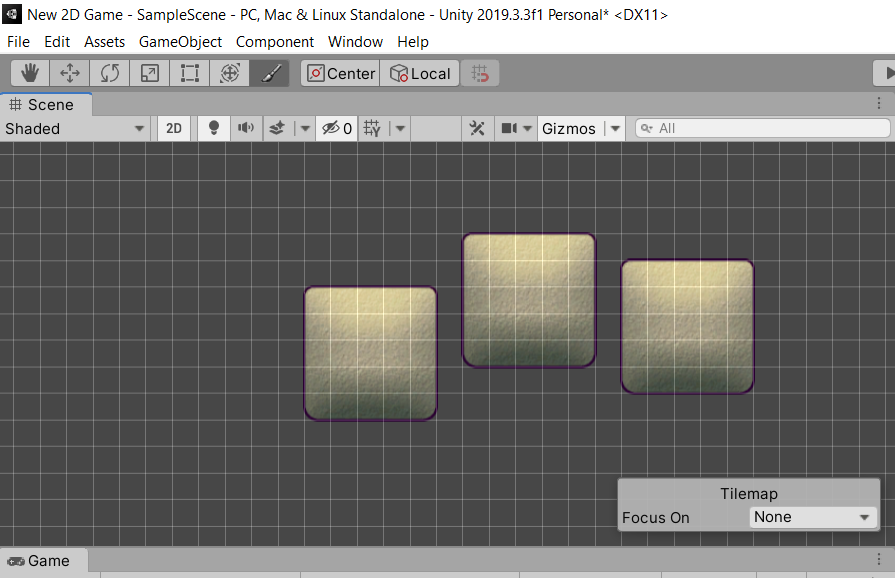


Creating this flowchart helped me break all the bigger problems into smaller ones and wrap my head around what I needed to do and the iterative process that goes into coding.

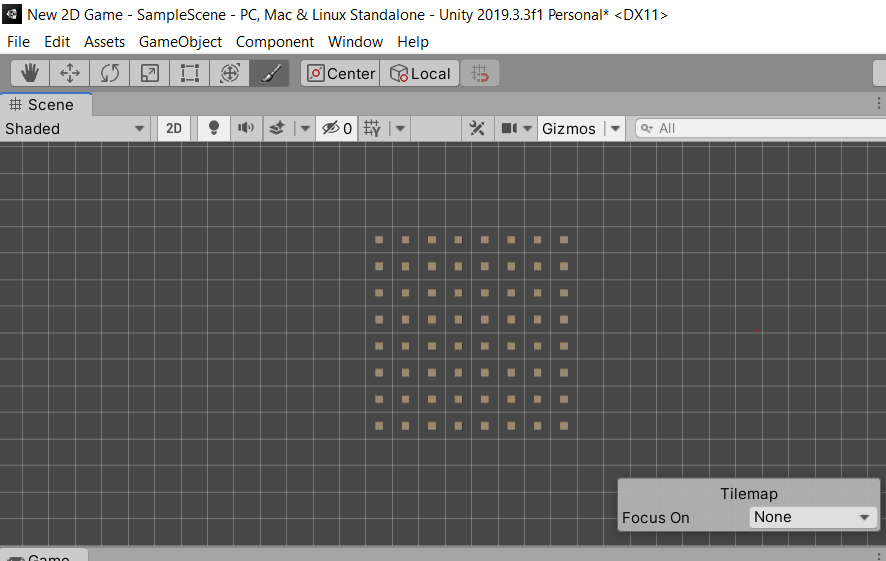
# Development

## Development log

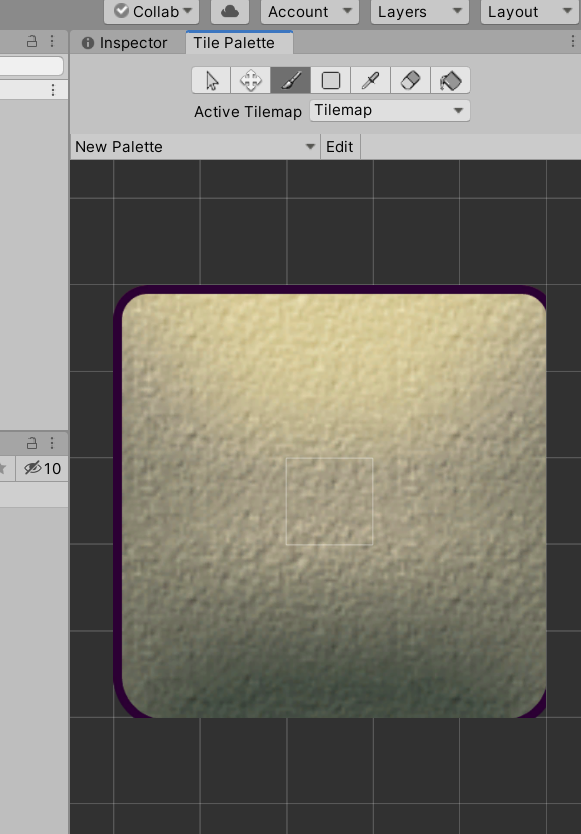
### Level Design

In order to learn how tile maps worked I watched a YouTube video on tile maps in unity. It showed me how to create palettes and a grid. It also explained how to apply your tiles in the game scene. The video was called ‘TILEMAPS in Unity’ by ‘Brackeys’. Next, I needed to acquire assets that I could use in my project. I acquired assets from “assetstore.unity.com”. They consisted of 2D rounded blocks which I would need as the ground and obstacles for the puzzle. I imported them into unity and created a palette called solid. Next, I created a Tile map to easily put the assets on a grid. The first problem I came across was putting my assets on the grid. After I created a new tile palette for the ground, the image was far too big for the grid. As shown below.

Upon encountering this problem, I researched online for a way to resize the images to fit the grid. I found out that it was possible to slice the images. At the time I did not know that the dimensions were important when it came to slicing assets, so I just used the default and got this.

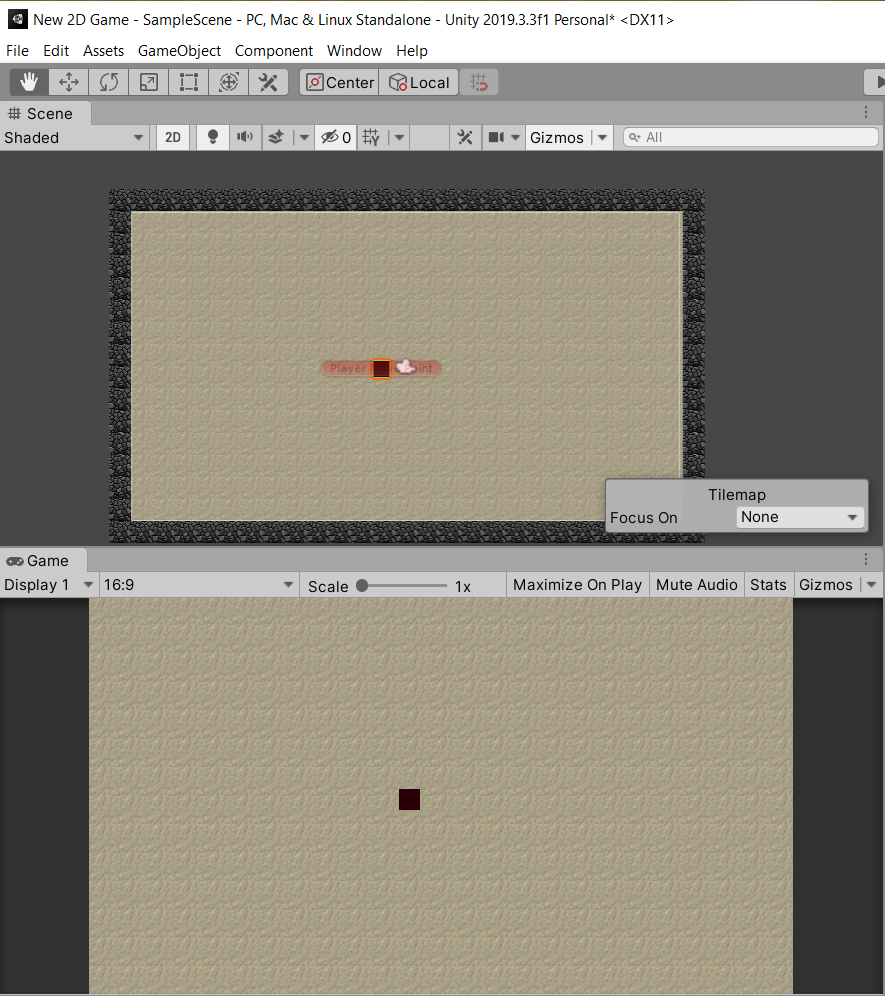


The blocks were too small to fit fully in the grid. At this point it was clear that the slicing and dimensions were to blame. I went to the image files and found their original dimensions and used that to slice them. The blocks that I acquired were rounded and rounded blocks look awkward on a grid. I decided to slice them in a way that I can take a piece of the block and use that as the ground.



This way, I was able to use the centre of the block to create the ground for the maze. I had to make sure that I was consistent with the part of the block that I was using as the ground because other parts of the block are in a different shade.

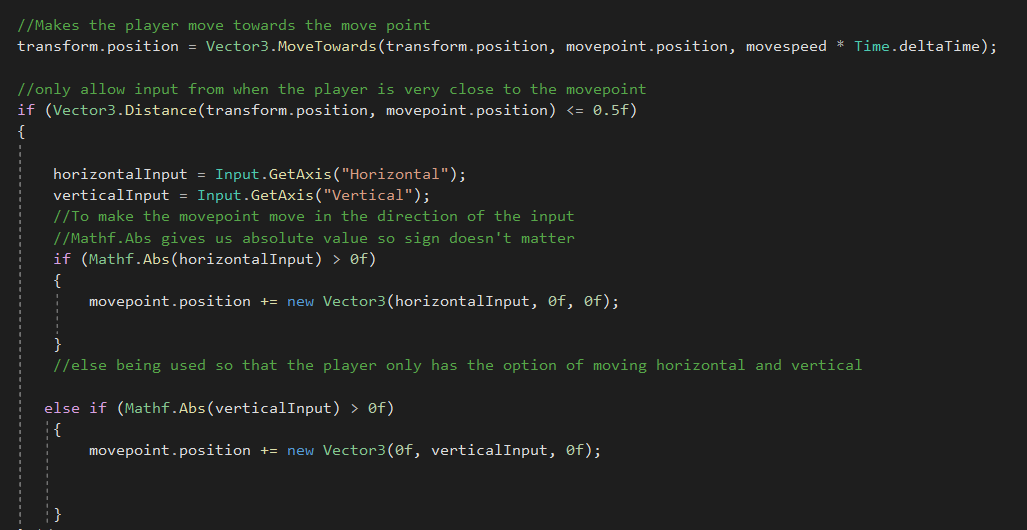
Next, I used a different rounded block to act as the boundaries of the maze. I couldn’t use a sliced version of the block because it did not have a smooth texture like the on above. It was rough and abstract so it would have come out weird if I sliced it. Since I was using this block as a boundary for the maze anyway, I used my default image processor to crop out the edges, so it looks more attractive. As of now I have my scene camera in a way that the boundaries do not show. But I may change this later.



With the base layout of the levels done, the next thing to do was to create the player and write its script. I created a 3D game object and named it ‘Player’. This is the object that can be seen in the middle of the level. This player will serve as the object that the human would control and use to traverse the maze. Since this project is being done in Unity, the script for the player is written in C# using visual studio.

### Player Movement

Regarding the script for the player movement, I went through some challenges deciding on what to do. At first, I searched on YouTube about how to make a player have grid-based movement in Unity and came across a video called ‘Grid based movement in Unity’ by ‘gamesplusjames’. I followed the tutorial in this video, and it consisted of using a move point. It was setup that the move point moves whenever there is input from the player and the player simply follows the move point.



However, my friend who has experience with Unity explained that there could be an easier way of achieving player movement without using a move point. He explained that I could simply create variables for movement in the x and y axis, then create an if statement that moves the player by a certain distance when the corresponding key is pressed.

Void Update()

{

float ydisplacement = 0f;

float xdisplacement = 0f;

if(Input.GetKey(“up”))

{

ydisplacement = 0.1f;

}

if(Input.GetKey(“down”))

{

ydisplacement =-0.1;

}

if(Input.GetKey(“right”))

{

xdisplacement = 0.1f;

}

If(Input.GetKey(“left”))

{

xdisplacement = -0.1f;

}

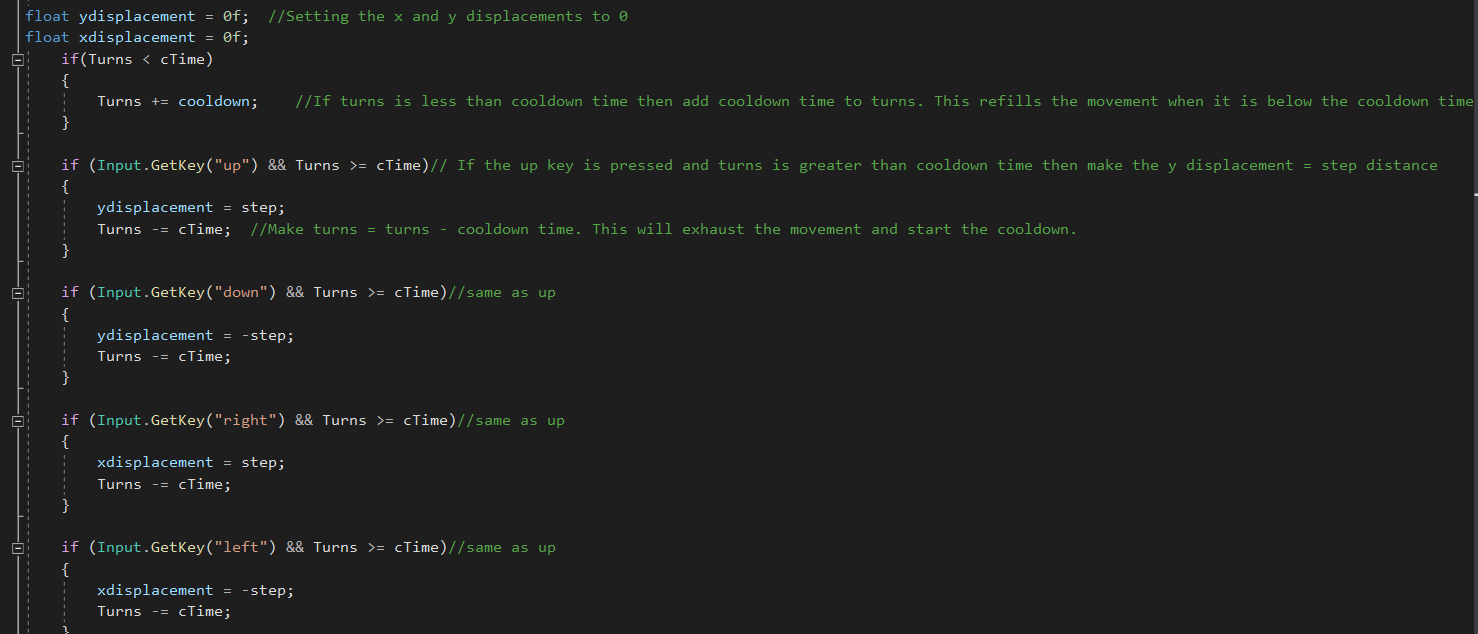
The position of the player is then translated to its appropriate position. This is done by adding the values of the x and y displacements to the original positions of x and y. I created a Vector2 variable called ‘targetposition’

targetposition = new Vector2(transform.position.x + xdisplacement\*movespeed\*Time.delaTime, transform.position.y + ydisplacement\*movespeed\*Time.deltaTime);

transform.position = targetposition;

This was a good solution at the time, but it did not have the grid-based movement that I was looking for. The player just slid along the map, which was not ideal. Then I got the idea to add a cooldown to the movement. Sure, cooldowns are mainly used to prevent players from spamming the same action in games. So, I decided that it would serve the same purpose in this game. I used the example above but with some added variables to include the cooldown system. These are:

* private float Turns = 1f
* public float step = 0.5f
* public float ctime = 0.5f
* Float cooldown = Time.deltaTime



I made the ‘Turns’ variable private because it would not need to be modified in the inspector. I modified the if statements for the movement and made it so that if the up-arrow key is pressed and the turns variable is greater than or equal to ctime, then make ydisplacement equal to the step distance and take away cooldown time from turns to deplete the movement and start the cooldown. To refill ‘Turns’ every time the cooldown is used, I used an if statement where if turns is less than the cooldown time then add cooldown to it (cooldown being Time.deltaTime). This will refill its time whenever it goes below cooldown time. The reason that I used greater than or equal to when creating the if statement for movement is because when the cooldown for ‘Turns’ is refilling it will sometimes go above the cooldown time and if I used equal to, it would not move anymore after it exceeded the cooldown time. Hence, why I used greater than or equal to. I repeated this for the other directions but used negative step for ‘down’ and ‘left’ because they are moving in the opposite direction.

# Testing for Evaluation

## The original test plan

## Testing process

# Evaluation

## Test evidence

## How to develop unmet criteria

## Evaluation of usability features

## Maintenance Issues

## Potential improvements