A-LEVEL COURSEWORK: PROJECT THESIUS

Fraaz Kagzi

CANDIDATE NO: CENTRE NO:

Clitheroe Royal Grammar Sixth Form

Contents

ANALYSIS

Client: Douy Singsamaran; Portfolio- https://singsamran.artstation.com/

Background to the problem

Douy is an artist, graduated from Kent university. Throughout the course of his degree he has made many art pieces and animations. He desires to enter the video game scene as an artist, however Douy is unsure whether his art style will be suitable for video games. In order to test the suitability for his art, he asked me to create a video game. If this test is successful, the game would be a great addition to his portfolio as an inspiring game artist. Although he did influence what the main frame of the game should be, I was given freedom to add features which I thought would make the game more interesting as well as challenging when it comes to implementing these ideas to a program.

Pre - Development Interview with Douy Singsamaran

Is there a certain type of game you would like to be made?

DS: I don't want the game to be very difficult. Something not too simple and not too complicated. Something that feels familiar like Pac-Man, but with a twist. I would like it to be endless, so like the game lasts as long as the player's character is alive.

You want this game to be heavily influenced by Pac-Man, but how would you like it to differ?

DS: Pac-Man's main target is just to collect those points and survive. I would like the game to have this as the basis however, in addition to this, if possible it would be good to add some sort of weapon to fight off against enemies and give it some horror aspects.

How will this game be useful for you?

DS: For artists, your portfolio is the most significant thing when applying for a job in art. Specifically for me, as I want to hopefully become a game artist, most companies will look through my portfolio to see if I am a suitable candidate. This game will be able to showcase my art to companies and show how my art, whether it fits or not, looks when situated in a game. This would make my portfolio unique.

You said you want to showcase your art; do you have an idea of how you want this to be done?

DS: I'm going to create all the necessary assets for this game whether that be the characters and enemies or the backgrounds and resources. So, however the game is played, the player would always be able to see some form of my art. The characters and enemies should be the most eye-catching. It'll be good if the game has some sort of story behind it, but apart from that, there isn't a specific way I want the game to showcase my art.

Potential Ideas and Chosen Idea

1. Project Lava

An endless side-scroller platform game, where the objective is to avoid falling. Collecting some sort of items (e.g.: coins) will determine the score. If the character touches the ground the game will end and a table of high scores will be displayed

- This project would have been too simple, and I believe it would have been difficult to think about how to implement high level skills in such a simple program
- This project would limit how much Douy's art is shown, a side scroller would mean that only one side of the art would be shown
- Due to only one side being shown, a mutual agreement between me and the end user was that this project would not be effective enough for Douy in terms of the limited skills displayed whether it comes to programming or art

2. Project Playtime

An application consisting of 5 mini games such as: a simple platformer, an endless runner and others. I got this idea from Douy's previous artwork. For his university project, he created 5 3D character models, which he based on children's games, like: the floor is lava, hide & seek, king of the hill and tag. I had the idea of basing my game around these 5 characters and give each character their own mini game.

- This idea will require a lot of time, since I would be creating 5 separate games with different types of AI.
- Making this game very simple would make it suitable for the deadline, however doing this will decrease and minimize the high-level qualities my program could have.

3. [CHOSEN IDEA] Project Theseus

   A survival maze game, heavily influenced by Pac-Man, where the objective is to find a certain number of keys and escape. This will be done whilst the player is getting tracked and chased down by monsters.

   - This project would be difficult; however, I think it would be more manageable than Project Playtime
   - This project has potential for me to implement many useful A-Level techniques as it has some features that make it fascinating to program

Description of My Game

Project Theseus is a level-based survival game, where the player is trapped in a maze and has to find a certain number of keys, to unlock the door and escape. However, it is not that simple. Whilst the player is trying to find the keys, they are being hunted and chased down by monsters in the maze. To give this more of a horror theme, the maze will be quite dark.

I want this game to last as long as the player survives and not limit it to a certain number of levels. Therefore, I am going to make it so every level, the maze is going to be randomly generated which will make every level different. A personal feature I wanted to add, is to make every square in the maze accessible. Making the maze randomly generated, I am aware that there will be some squares that will not be accessible. I believe adding this feature would make the game more satisfying as well as being an opportunity to add some A-Level programming skills.

Features I could add

Whilst the player is in the maze, he will have a flash light which is slowly deteriorating as the game is played. The battery level will determine the area covered by the light. There will be pickable battery cells which can increase the battery level.

The player would have 3 lives and every time in comes in contact with a monster they will lose a life. After all, lives are lost the game will end, displaying a table of high score consisting of players who made it the highest levels

Users and Limitations

This game will primarily be used by Douy and also by companies viewing Douy's art, who will be the main source of feedback. However, this game is not just limited to a certain number of users, as anyone with an interest for the game will be able to play it and become a secondary source of feedback. I am going to make this game available for alpha testing so I can gather as much feedback as possible, which will enable me to have a list of improvements needed.

Creating this game would not be simple, there are some definite limitations which restrict me for creating this game to be the best quality.

This is the first time I will be attempting to create a game so my knowledge would be a limitation. I will be spending a lot of time researching different algorithms, traversals and C# functions which would allow me to make my game closer to what I actually want it to be. I'll be creating this game using unity which I have also never used before however unity allows me to program in C# which I have prior programming experience in. To boost my knowledge on unity and game developing, I am going to watch and follow many tutorials on making simple games on unity and gain knowledge on specific features such as implementing character movement.

Time will also be a major limitation since I only have till Easter to complete this whole project Which includes researching, documentation and the development of the project

Research

Most suitable software

For the game, I had to decide which software to us to create the project. This was my first time creating a game so I did research on 2 game engines Unity and Unreal Engine. Unreal only allowed games to made with C++ or their own blueprints system which requires no programming. On the other hand, Unity allows games to made with C# which is the language I am most confident with, so I chose unity as the software I will be using to create my game.

Player

For the player, I needed to know how to make the character move. I watched many YouTube tutorials by professionals such as brackeys and learnt different ways this can be done.

I had to make sure my player wouldn't be able to go through walls & objects so as well as player movement I also researched about collisions and rigid bodies

AI and Maze

When researching about how I could make my AI, I came across pathfinding algorithms, such as "Dijkstra's Algorithm" which finds the shortest path from one node to another in a graph. I also researched about 2 traversal algorithms: breadth first and depth first traversals. Which can be used to make the AI decide where to move. Whilst researching I saw that these traversals can become as issue for the CPU if ran multiple times as it can cause programs to crash. So, I found out how to potentially fix this problem with a method called multithreading, where instead of running all these traversals on only one of the CPU's core, multithreading takes advantage of multi core CPU's by spreading these instructions between multiple cores.

References

https://youtu.be/Au8oX5pu5u4 - *Brackeys - "How to make a video game in unity – MOVEMENT"*

https://youtu.be/S2mK6KFdv0I - *Brackeys - "MOVEMENT - Making an RPG in unity"*

https://youtu.be/zc8ac_qUXQY - *Brackeys - "START MENU in Unity"*

https://youtu.be/FSEbPxf0kfs - *Dapper Dino - "How To Create A Scoreboard System For Your Game – Unity Tutorial"*

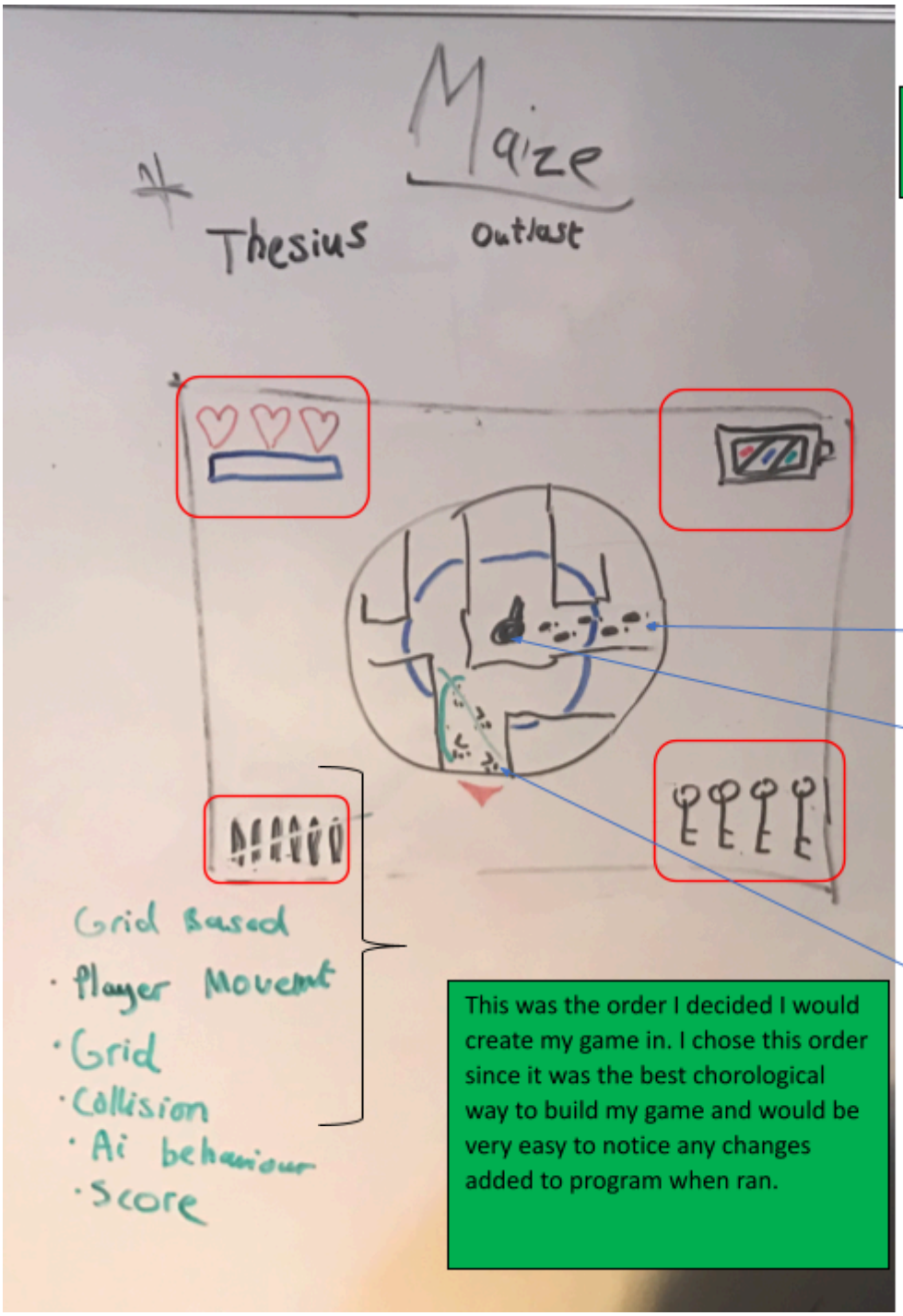
An Existing game

In Pacman, the objective is to collect the yellow circles called Pac dots. Collecting Pac dots determines your score, the more you collect the higher your score. Whilst your collect Pac dots, you'll have 4 Ghosts after you. The 4 ghost each has a different algorithm. One guesses your next move and intercepts it, one follows you, one patrols a certain area and one moves completely randomly. If any of these ghosts catch you lose a life from your starting 3 lives. However there an item called power pellets which turn the table and allow you to eat the ghosts.

How my Game will be different from Pacman

First of all, my game would have a randomly generated maze unlike Pacman's pre-set maze. My game has a different objective. The score system won't be based on points, but will be based on the number of levels survived. Unlike Pacman's 4 different AI's with different algorithms, my game will only have one type of AI following a traversal algorithm.

What I want my game to look like



This is what I would like my game to look like on a screen

Everything circled in red would be part of the HUD. This could include things like the keys needed to escape, your health and stamina, the battery on your flash light and your ammo capacity

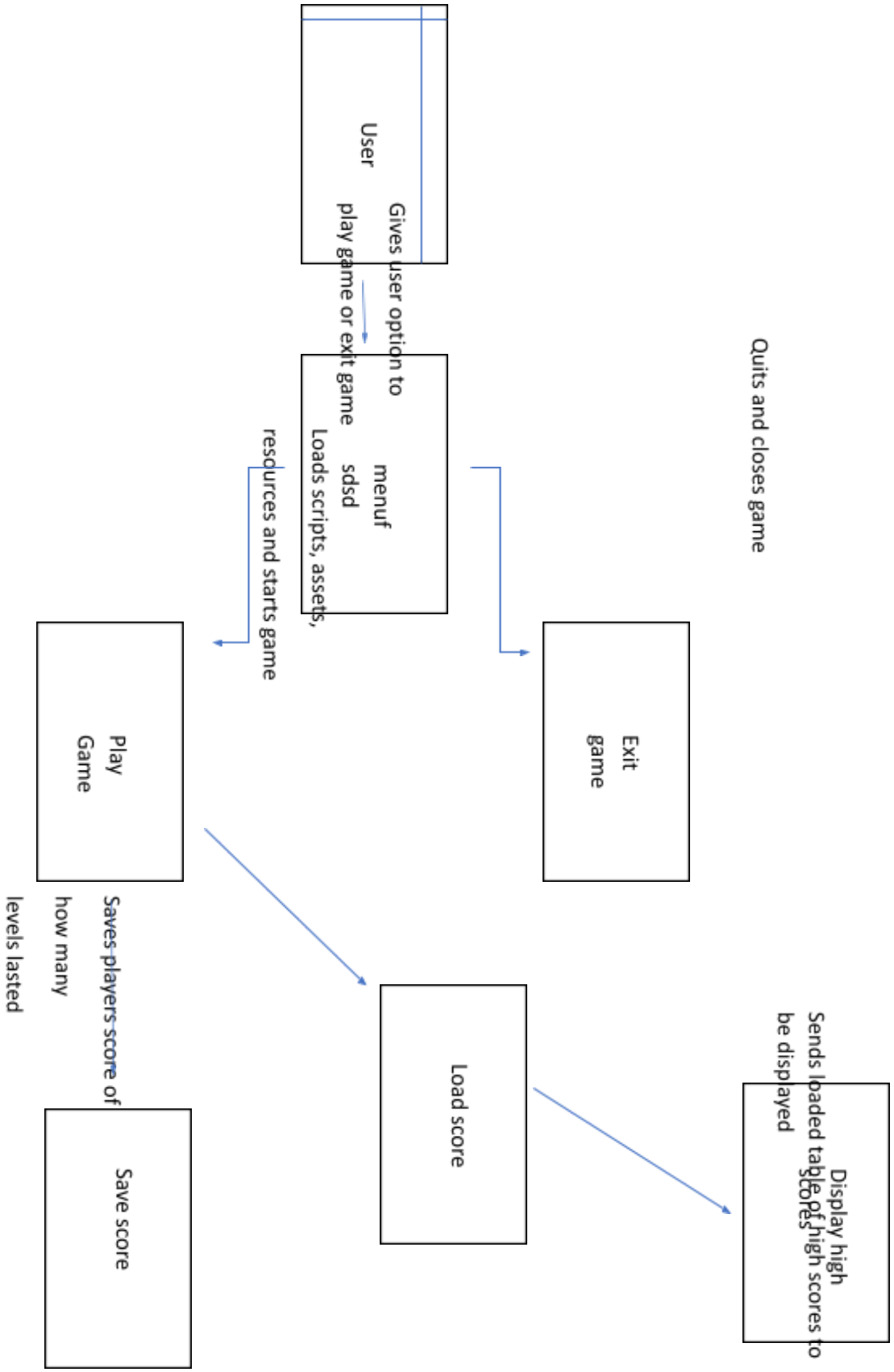
Footprints left by player

player

Footprints left by enemy

This was the order I decided I would create my game in. I chose this order since it was the best chorological way to build my game and would be very easy to notice any changes added to program when ran.

Data Flow Diagram



My Objectives for the Game

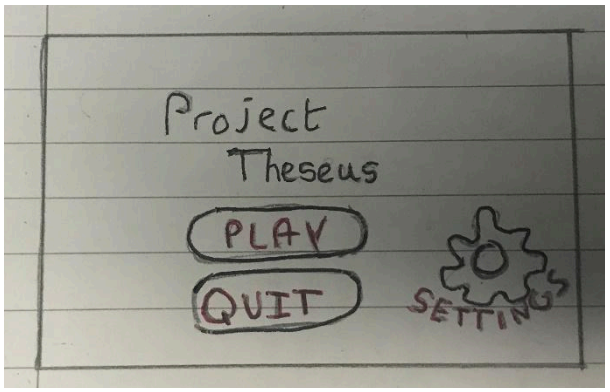
(in priority order) (should be done or could be done)

1. The game should be able to clearly showcase the end user's art.
    - Should be able to rotate around character to show all sides
2. The game should have a simple to understand user interface
    - Should have a menu
    - Could have a in game settings menu
3. The game should be playable with or without an objective.
    - Should have smooth function player movement
    - Should have implemented player collisions
    - Should be able to pick up items
    - Should have a functioning level system
    - Could have extra features like flash light, weapons, stamina bar, hearts(lives)
4. The game should have a maze
    - This maze should be randomly generated, so that every level there will be a completely different maze
    - In this maze, every square should be accessible
    - This maze could be interpreted as a graph so that it can be traversed for the AI
5. The game should have a fully functional enemy AI
    - Shouldn't get stuck between walls
    - Should be able to follow the player (could be done by implementing a traversal algorithm)
6. The game should last as long as the player survives
    - Player has 3 lives
    - Should stop game as soon as player dies (or runs out of lives)
    - After player finds keys and escapes a new maze should be generated.
7. After the games finishes the game should save the score and display a table of high scores
    - Should save only the top 5 high scores
    - If player score is high enough to enter high scores, it should ask them for their names.
    - Should save the score corresponding to the input given by player

DESIGN

Screen transitions

screen 1 - Menu



This is the first screen that users will see.

The user will have 3 options:

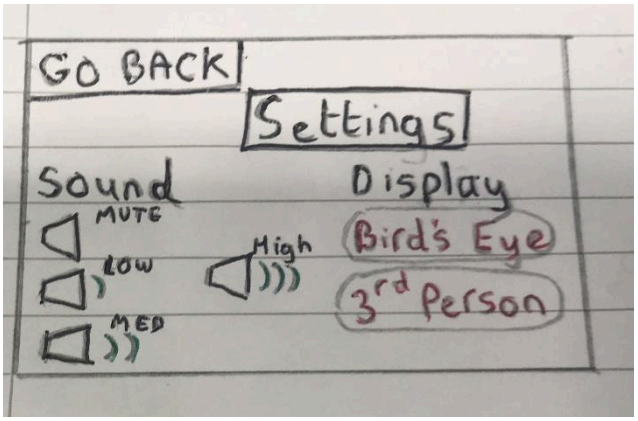
To play the game

To open the settings menu

To quit the game

In the final system, the title screen would be more appealing and would have more pictures.

Screen 2 Settings Menu



In the settings menu, the user will have the option to change the sound settings, change the display settings, or go back to the main menu.

There are 4 different sound settings to choose from:

Mute – 0% volume

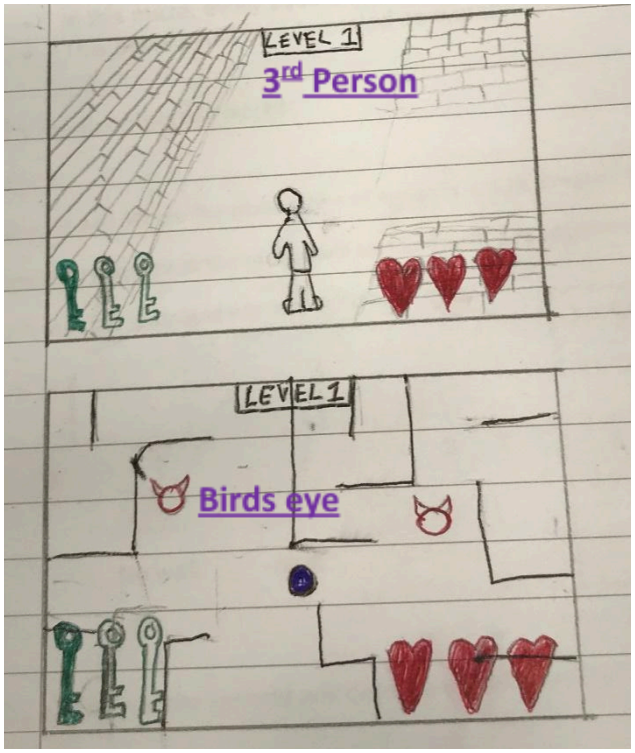
Low – 25% volume

Medium – 50% volume

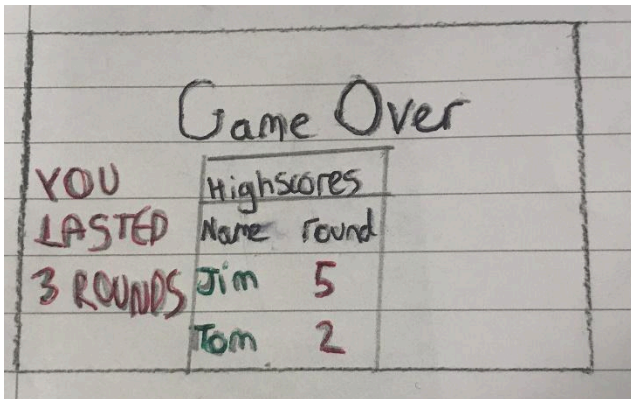
High – 100% volume

In the display settings, the user will have to choose whether to play the game in 3rd person or birds eye view.

Screen 3 – Game Screen



Screen 4 – Game Over/ High score screen



After all, 3 lives have been lost. The game would finish and the game over screen would be displayed. In this screen, the game would display how many rounds the player lasted. As well as this the game would also display table of high scores which would display the names of the top

Maze Generator Plan

Since my game will be revolved around a maze I would need an algorithm, which would randomly generate a maze so that every level in my game is different. so, I created my own algorithm to randomly generate a maze.

Objectives for the Maze Generator

- This maze should be randomly generated, so that every level there will be a completely different maze
- In this maze, every square should be accessible
- This maze could be interpreted as a graph so that it can be traversed for the AI

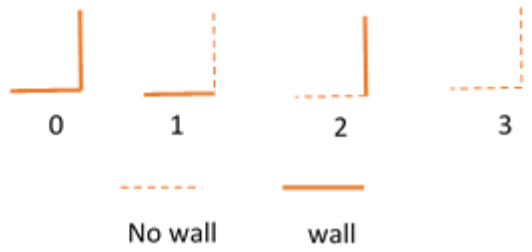
How the algorithm will work?

Stage 1:

The user will decide the maze size and a matrix will be created based on the user's input

Then every square in the matrix will be filled with a random number, either 0,1,2 or 3.

Each number is assigned to a specific type of wall



0 = RIGHT WALL & DOWN WALL

1 = ONLY DOWN WALL

2 = ONLY RIGHT WALL

3 = NO WALL

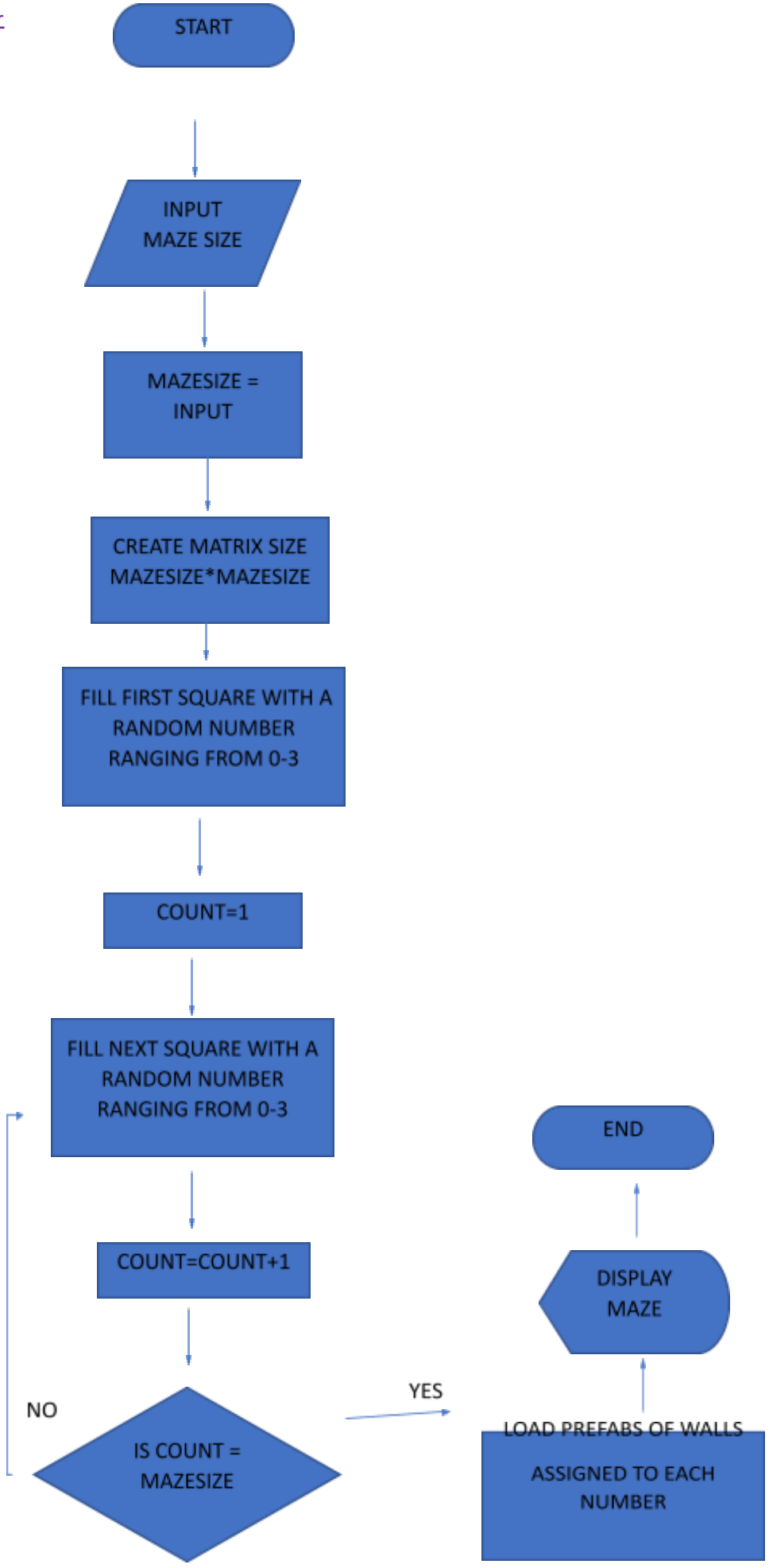
The numbers in the squares will determine which wall will be placed on that square

Stage 2:

After this, a basic maze would've been made. However not all the objectives are met. There is a very high possibility that every square is not accessible. So, to fix this there will be a recursive Depth First traversal algorithm. This will check how many squares are accessible from the top left of the maze. Until the number of squares accessible is not equal to the area of the maze (the users input squared) the algorithm will break walls and repeat the traversal algorithm until so that the number of squares accessible is equal to the area.

Flowchart of Maze Generator

Stage 1:



Example of the algorithm

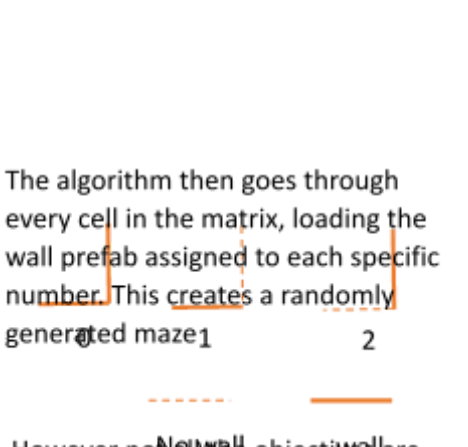
This is an example of how the maze Generator Algorithm works

I decided to set the maze size input as 6, which created a 6*6 matrix.

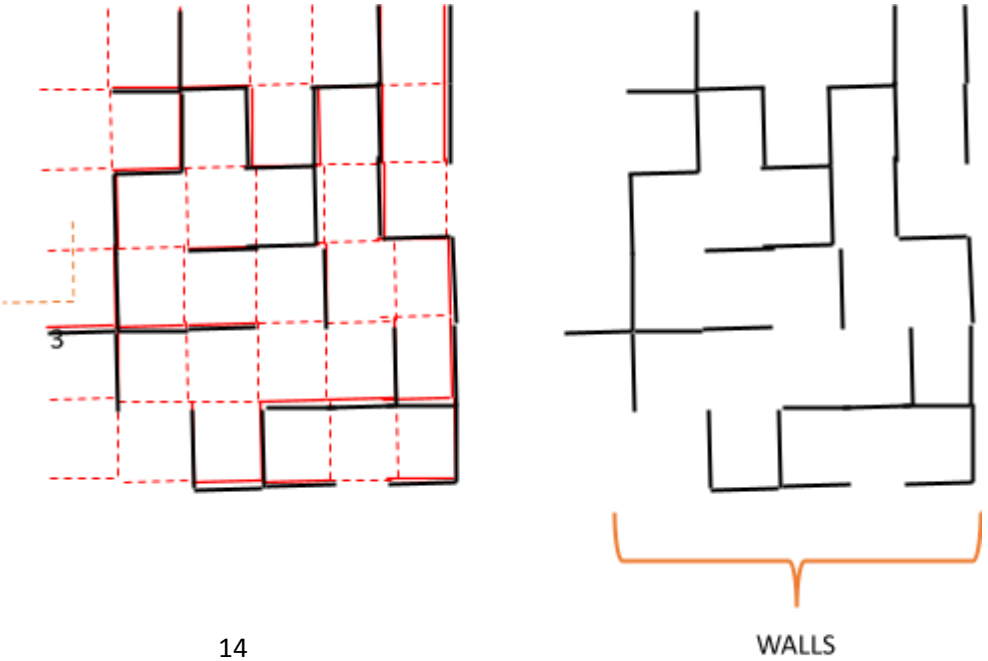
The algorithm then fills every cell in the matrix with a number ranging from 0 to 3, using a random number generator (limiting the numbers generated to either 0,1,2 or 3)

3	0	1	3	0	2
3	0	2	0	2	2
2	3	1	0	2	1
0	1	1	2	3	2
2	3	3	1	0	0
3	2	0	1	3	0

The algorithm then goes through every cell in the matrix, loading the wall prefab assigned to each specific number. This creates a randomly generated maze



However not all the objectives are met yet, since not every square in the maze is still not accessible

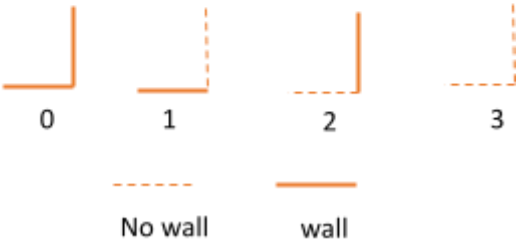


Explaining Maze Generator

```
public class MazeGenerator : MonoBehaviour
{
    public int mazeSize;

    void GenerateMatrix(int[,] mazeMx)
    {
        for (int i = 0; i < mazeSize; i++)
        {
            for (int j = 0; j < mazeSize; j++)
            {
                mazeMx[i, j] = Random.Range(0, 4); //0,1,2,3 each combination of bottom and right walls
                //Debug.Log(i.ToString() + j.ToString() + "Matrix value " + mazeMx[i, j]);
            }
        }
        mazeMx[mazeSize - 2, mazeSize - 1] = 1;//set as 1 to avoid errors
        mazeMx[(mazeSize - 1) / 2, mazeSize - 1] = 3;// forced to 3 in order to allow a clear exit from elevator
    }
```

This fills every single square in the matrix with either a: 0,1,2,3. The number put in the square is completely randomly generated from the choice of 4 numbers. Each number is designated to a specific type of wall

I had to force some squares to have a certain type of wall, in order to allow the player to exit spawn.



Since every square will have a random type of wall, not every square will be accessible. In order to fix this issue, I created several different methods. The 4 methods I created to fix this issue are: FixGraph, CountSet, DepthFirst and AddToSet.

The way did this, is by creating an array, which stores all the coordinates of the squares that linked to an accessible square. I also made a depth first traversal method which starting from the top left of the matrix see's how many squares it can go to. If the Number of squares accessible from by the depth first traversal method is equal to the area of the matrix (mazeSize * mazeSize), this would mean that the issue is fixed and every square is accessible someway.

```
void FixGraph(int[,] mazeMx)
{
    Vector2[] linkSet = new Vector2[mazeSize * mazeSize];  // an array of coordinates that are linked to another accessable coordinate
```

First of all, in the method FixGraph I created an empty vector 2 array called linkset which will store all the coordinates of squares that are linked to another accessible square

The Boolean method AddToSet checks whether the coordinate of a square is already in the linkSet array or if they need to be added. If the coordinates are not in the array AddToSet returns true indicating it needs to be added to the set. If they are in the array it returns false.

```
bool AddToSet(Vector2 vec, Vector2[] linkSet)
{
    for (int i = 0; i < mazeSize * mazeSize; i++)
    {
        if (linkSet[i] != new Vector2(-1, -1) && linkSet[i] == vec)// checks if (x,y) is already in linkset
        {
            return false; // returns false if already in linkset
        }
    }
    for (int i = 0; i < mazeSize * mazeSize; i++)
    {
        if (linkSet[i] == new Vector2(-1, -1)) // already checked if (x,y) is in linkset
        {
            linkSet[i] = vec;
            return true; // returns true if not in linkset, adds vector to link set
        }
    }
    return false;
}
```

Before AddToSet is run, in the method CountSet all the elements in the vector array are set to (-1, -1), since (0, 0) is a used vector in this program

After already checking whether the coordinates are in the array, the program then checks whether the coordinates at index of the array are (-1, -1). If they are, this means that this part of the array has been untouched and the coordinates of the square can replace the (-1, -1)

```
void DepthFirst(int x, int y, int[,] mazeMx, Vector2[] linkSet)
{
    AddToSet(new Vector2(x, y), linkSet); // adds starting square to link set
    // order priority: right, down ,left, up

    if (x < mazeSize - 1) // has to have right border wall if   x = mazeSize -1
    {
        if (mazeMx[x, y] == 1 || mazeMx[x, y] == 3) // if the coordinates have no right wall
        {
            if (AddToSet(new Vector2(x + 1, y), linkSet)) //checks  right
            {
                DepthFirst(x + 1, y, mazeMx, linkSet); // recursive function
            }
        }
    }

    if (y < mazeSize - 1) // has to have down border wall   y = mazeSize -1
    {
        if (mazeMx[x, y] == 2 || mazeMx[x, y] == 3) // if the coordinates have no down wall
        {
            if (AddToSet(new Vector2(x, y + 1), linkSet)) //checks down
            {
                DepthFirst(x, y + 1, mazeMx, linkSet);
            }
        }
    }

    if (x > 0)
    {
        if (mazeMx[x - 1, y] == 1 || mazeMx[x - 1, y] == 3)
        {
            if (AddToSet(new Vector2(x - 1, y), linkSet)) // checks left
            {
                DepthFirst(x - 1, y, mazeMx, linkSet);
            }
        }
    }

    if (y > 0)
    {
        if (mazeMx[x, y - 1] == 2 || mazeMx[x, y - 1] == 3)
        {
            if (AddToSet(new Vector2(x, y - 1), linkSet)) // checks up
            {
                DepthFirst(x, y - 1, mazeMx, linkSet);
            }
        }
    }


}
```

I created a recursive function to implement depth first traversal on the maze. I called this method DepthFirst. Depth first checks all the squares accessible from the origin (0,0).

It starts at the origin (0,0) and first checks the square right of that. If that square is accessible (has no right wall in between) it runs the AddToSet method in order to add that coordinate to linkSet. Then it calls the recursive DepthFirst method with the same parameters except the x value is incremented by 1, causing the square to move right by 1 square each time.

After going through all the rows, while x<mazeSize the program then repeats all this for the columns while y<mazeSize. . After one y value has been traversed it then calls the recursive Depth First method with the same parameters except the y value is incremented by 1, causing the square to move down by 1 square each time.

Then, for all the squares where y>0, it checks whether the square above has a down wall. If there is no down wall the square above can be accessible and the coordinates are added to linkSet. DepthFirst is called again with the same parameter except y is decreased by 1, causing the square to be 1 to the up each time

After, for all squares where x>0, it checks whether the square to the left has a right wall. If there is no wall, the square to the left can be accessible and the coordinates are added to linkSet. DepthFirst is called again with the same parameter except x is decreased by 1, causing the square to be 1 to the left each time

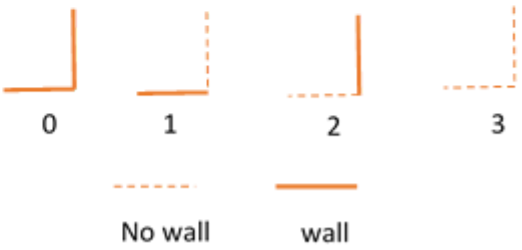
```
int CountSet(int[,] mazeMx, Vector2[] linkSet) // resets linkset and counts
{
    for (int i = 0; i < mazeSize * mazeSize; i++)
    {
        linkSet[i] = new Vector2(-1, -1); // -1 is used here since 0 is a value in this program
    }

    DepthFirst(0, 0, mazeMx, linkSet); // starts depth first at matrix origin 0,0
    int setSum = 0;

    for (int i = 0; i < mazeSize * mazeSize; i++)
    {
        if (linkSet[i] != new Vector2(-1, -1))
        {
            setSum = setSum + 1; // needs to be equal to n*n to be a maze where every square is accessable
        }
    }

    return setSum;
}
```

```
void FixGraph(int[,] mazeMx)
{

    Vector2[] linkSet = new Vector2[mazeSize * mazeSize];  // an array of coordinates that are linked to another accessable coordinate
    while ((CountSet(mazeMx, linkSet) < mazeSize * mazeSize))
    {
        for (int i = 0; i < mazeSize - 1; i++)
        {
            for (int j = 0; j < mazeSize - 1; j++)
            {
                if ((AddToSet(new Vector2(i + 1, j), linkSet))&& ((mazeMx[i,j] == 0)|| (mazeMx[i, j] == 2)))
                {
                    mazeMx[i, j] = mazeMx[i, j] + 1; // breaks right wall
                    CountSet(mazeMx, linkSet); // resets and does depth first again
                }
                else if (AddToSet(new Vector2(i, j + 1), linkSet) && ((mazeMx[i, j] == 0) || (mazeMx[i, j] == 1)))
                {
                    mazeMx[i, j] = mazeMx[i, j] + 2; // breaks down wall
                    CountSet(mazeMx, linkSet);
                }
            }
        }
    }
}
```

The method CountSet is an integer method which returns the integer variable setSum. CountSet resets linkSet so that every element in the vector array is set to (-1, -1). It then runs DepthFirst starting with the x and y coordinates set t

FixGraph is the method whic the fix to make every square accessible. As mentioned bef starts by making an empty ve array called linkSet. Then it g through the entire matrix, wh first it runs the AddToSet me check whether the coordinat linkSet, if not, depending on number it breaks a wall. corresponding to the square 2 it breaks right wall) (0 or 1 down wall) and the runs Cou reset linkSet and refill it with square accessible from origin

```
void MazeVisualizer(int[,] mazeMx)
{
    mazeGrid = new GameObject[mazeSize, mazeSize];
    mazeWall = new GameObject[4 * mazeSize - 2];
    int wallCounter = 0;

    for (int i = 0; i < mazeSize; i++)
    {
        for (int j = 0; j < mazeSize; j++)
        {


            if (mazeMx[i, j] == 0)
            {
                mazeGrid[i, j] = Instantiate(Resources.Load("wall0", typeof(GameObject))) as GameObject;

            }
            if (mazeMx[i, j] == 1)
            {
                mazeGrid[i, j] = Instantiate(Resources.Load("wall1", typeof(GameObject))) as GameObject;

            }
            if (mazeMx[i, j] == 2)
            {
                mazeGrid[i, j] = Instantiate(Resources.Load("wall2", typeof(GameObject))) as GameObject;
                //Debug.Log("Wall 2");
            }
            if (mazeMx[i, j] == 3)
            {
                mazeGrid[i, j] = Instantiate(Resources.Load("wall3", typeof(GameObject))) as GameObject;
                //Debug.Log("Wall 3");
            }

            mazeGrid[i, j].GetComponent<Transform>().transform.position = new Vector3(i * 10, 0, -10 * j);


            if (i == 0)
            {
                mazeWall[wallCounter] = Instantiate(Resources.Load("RightWall", typeof(GameObject))) as GameObject;
                mazeWall[wallCounter].GetComponent<Transform>().transform.position = new Vector3(-5, 7.5f, -10 * j);
                wallCounter++;
            }
            if (i == mazeSize - 1 && mazeMx[i, j] != 0 && mazeMx[i, j] != 2)
            {
                mazeWall[wallCounter] = Instantiate(Resources.Load("RightWall", typeof(GameObject))) as GameObject;
                mazeWall[wallCounter].GetComponent<Transform>().transform.position = new Vector3(10 * mazeSize - 5, 7.5f, -10 * j);
                wallCounter++;
            }
            if (j == 0 && i != (mazeSize - 1) / 2)
            {
                mazeWall[wallCounter] = Instantiate(Resources.Load("DownWall", typeof(GameObject))) as GameObject;
                mazeWall[wallCounter].GetComponent<Transform>().transform.position = new Vector3(i * 10, 7.5f, 5);
                wallCounter++;
            }
            if (j == mazeSize - 1 && i != (mazeSize - 1) / 2 && mazeMx[i, j] != 0 && mazeMx[i, j] != 1)
            {
                mazeWall[wallCounter] = Instantiate(Resources.Load("DownWall", typeof(GameObject))) as GameObject;
                mazeWall[wallCounter].GetComponent<Transform>().transform.position = new Vector3(i * 10, 7.5f, -10 * mazeSize + 5);
                wallCounter++;
            }

        }
    }

}
```



MazeVisualizer is a method which actually creates the playable maze. As mentioned before that every square is assigned to a specific type of wall (either 0,1,2 or 3), MazeVisualizer loads the actual prefab for the specific wall from the Prefabs folder from the games file in unity.

MazeVisualizer also creates all the border walls for the matrix.

```
void NewLevel()
{

    int[,] mazeMx = new int[mazeSize, mazeSize];
    GenerateMatrix(mazeMx);
    FixGraph(mazeMx);
    MazeVisualizer(mazeMx);

}
```

Creates a matrix (table/graph) for the base for the maze. The dimensions are mazeSize by mazeSize which is a variable which value is assigned before program runs

Maze Generator- Table of methods and Variables

Method Name	Variable Name	Data type	What it does?
	mazeSize	int	User inputs a number of how big they want the length and width of the maze to be
NewLevel()		void	When called, creates a new maze for a new level.
	mazeMx	Int, 2D array	Creates a matrix size mazeSize*mizeSize. It is the base of the maze
GenerateMatrix()		null	This fills every single square in the matrix with either a: 0,1,2,3.
	i	int	Used as an index
	j	int	Used as an index
FixGraph()		null	Starts the process of breaking down certain walls so that every square in the maze is accessible
	linkSet	Vector 2 array	Stores the coordinates or squares that are linked to another square
	i	Int	Used as an index
	j	int	Used as an index
CountSet()		int	Resets the linkSet array and fills it with (-1,-1), then runs DepthFirst(). Repeats this until setSum equals mazeSize*mazeSize
	i	int	Used as an index

	setSum	int	Counts how many squares are accessible from the origin (0,0)
DepthFirst()		Void	checks all the squares accessible from the origin (0,0) via a traversal
	x	int	Used as x coordinate in a vector
	y	int	Used as y coordinate in a vector
AddToSet()		bool	checks whether the coordinates of a square are already in the linkSet array or if they need to be added
	i	int	Used as an index
	vec	Vector 2	Used to check if vector coordinate has already been traversed

Maze Generator - table of parameters

Method Name	Parameters Passed	Methods called
NewLevel()		GenerateMatrix() FixGraph()
GenerateMatrix()	mazeMx	
FixGraph()	mazeMx	CountSet() AddToSet()
CountSet()	mazeMx linkSet	DepthFirst()
DepthFirst()	x (declared in CountSet) y (declared in CountSet) mazeMx linkSet	AddToSet() DepthFirst()
AddToSet()	Vec (declared in FixGraph() and DepthFirst()) linkSet	

Enemy AI plan

To make this game more interesting and complex, I decided to add an enemy ai which would follow the player around the maze. To do this I decided to convert the maze matrix into a graph which would enable the enemy to traverse it and move around. If not within a certain area, the enemy would patrol and certain area and when in range of the player, the enemy would do a breadth first traversal every frame to constantly find out where to move next and will move to that position.

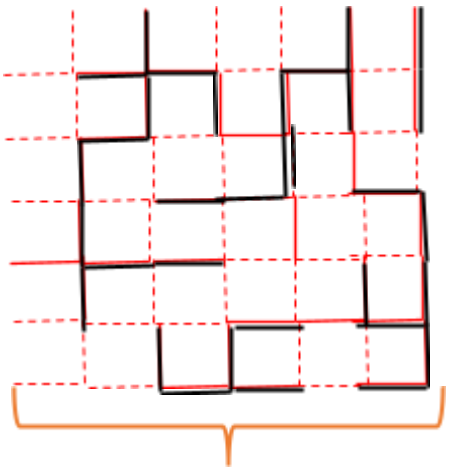
Objectives for the Enemy AI

- Should convert the maze matrix into a graph
- Should have a range where the enemy will not be able to follow the player
- If the enemy is not following the player, it should patrol a certain area
- Enemies should only start moving once the player has left the start lift

How the algorithm will work

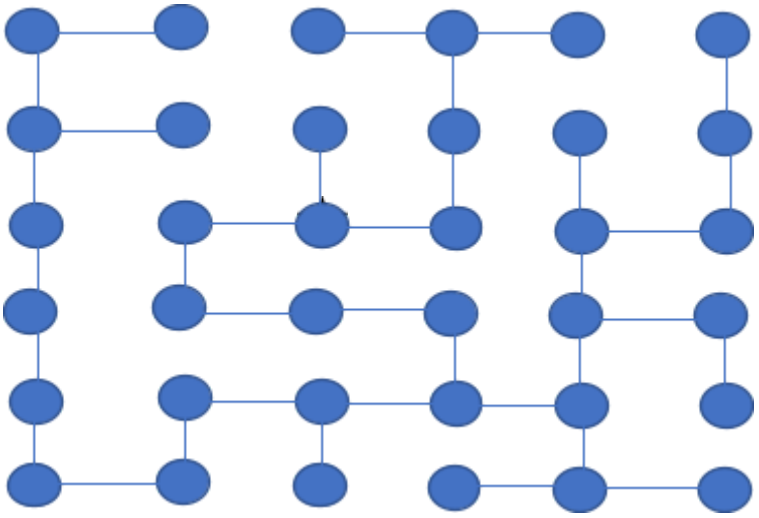
.

using a premade maze as an example, first the algorithm converts the maze matrix into a graph. Where each square is a node and the possible path to each square are represented as edges.



Matrix

WALLS

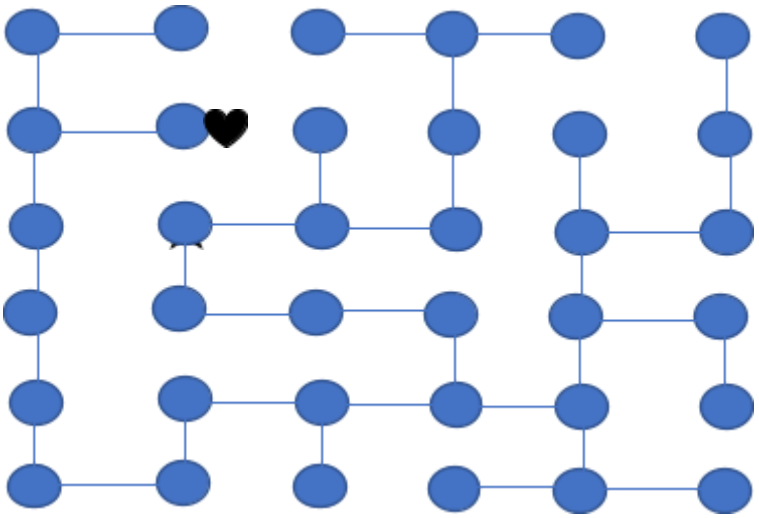


enemy

player

The enemies are spawned in on a random square within the maze. After the maze matrix has been converted to a graph and at least one other square contains the player, the enemy algorithm will start.

Every Frame the program performs breadth first traversal on each enemy, using the player as the targeted location. It is done every frame so that if the player moves to another node, the enemy would be able to change its direction and move toward the players new location. After the breadth first traversal finds out which node the enemy should move to, it converts the node location into a matrix coordinate and tells the enemy to move to that coordinate.



Over Here the enemy moved a node closer to the player. To make sure the enemy doesn't get stuck between walls, I can only travel using the edges between the nodes.

Testing

Public Testing

After the development was finished, before showing my end user the final system, I allowed friends and family to play the game so that they can provide helpful feedback and also inform me of any bugs and issues with the game, which I may not have been aware of. To my advantage the game testers provided me with unbiased and helpful feedback.

Feedback

Shazaib Naeem (friend) – “The game was quite simple to understand, I liked how it got progressively more difficult, it made it more challenging and enjoyable. However, after I completed level 8 I was unable to play the game as it seemed like my character was under the map and couldn’t pick up items. So, I think that is a bug in the game but apart from that the game was good. I was able to move my character freely and its impressive how the game goes on until the player loses all lives.”

Meghan Vora (friend) – “I enjoyed playing project Theseus. I noticed how the game got harder as the levels increased, bigger mazes, more keys to collect and more monsters. I don’t know if you’re already aware but after level 8 the game becomes unplayable, so I wasn’t able to test out any after game features. I think there’s a bug which makes the player go out the maze a fall forever. As well as think I thought that I was too easy to avoid the monsters so maybe the speed of them could be increased to make it more difficult and challenging.

Ben Redfearn (friend) – “Project Theseus was quite a scary game, running into a dead end with an enemy behind gave me quite a panic, the game had elements of a well-developed game I quite enjoyed the art of the game and with the game being 3D I was able to see the arts full potential. However, I found the game to be quite difficult, after level 5 it got a bit too difficult for me. I noticed a bug where I was able to go through walls by sprinting into corners. I noticed that there was a high score table after the game finished and would like this table to be accessed at the start menu as well, I think it’d be quite irritating to only see the top scores after you’ve died so that’s one improvement which would be good.

Results from feedback

- Fix bug where the game is unplayable after level 8
- Change high score table so that it is accessible at the start as well as the end.
- Increase the enemy speed at high levels.
- Fix bug where player is able to go through walls

Menu Testing Table

Test Number	Test Data	Input (if needed)	Expected Outcome	Actual Outcome / evidence	Solution (if needed)
1	Main Menu				
1.1		Click “PLAY”	Scene gets changed to game scene and game starts.		
1.2		Click “QUIT”	Application closes		
1.3		Click “CONTROLS”	Opens controls menu		
2	Controls Menu		Should display the controls of the game		
2.1		Click “BACK”	Should return back to main menu		
3	Pause Menu	Press Esc key whilst in game.	Should open pause menu		
3.1		Click “RETURN TO MENU”	Should stop the game an return to main menu.		

Player Movement Testing Table

Test number	Test Data (description)	Input (if needed)	Expected Outcome	Actual Outcome / evidence	Solution (if needed)
4	Whether player movement corresponds to the right input.				
4.1		W	Player moves forward		
4.2		A	Player moves left		
4.3		S	Player moves down		
4.4		D	Player moves right		
4.5		W & A **simultaneously**	Player moves diagonally		

			forward & left		
4.6		W & D **simultaneously**	Player moves diagonally forward &		
4.7		Left shift **Whilst pressing any of W, A, S, D**	Player movement speed increases		
5	Whether camera moves based on mouse movement				
5.1		Move mouse right	Camera moves to look right		
5.2		Move mouse left	Camera moves to look left		
5.3		Move mouse up	Camera moves to look up		
5.4		Move mouse down	Camera moves to look down		
6	Character collision				
6.1		Walk into wall	Wall stops player from walking through		
6.2		Sprint into wall	Wall stops player from walking through		

Maze Generator Testing Table

Test number	Test Data (description)	Input (if needed)	Expected Outcome	Actual Outcome / evidence	Solution (if needed)
7	Randomly generates a maze (starting with 10)	Level 1	Should generate a 10*10 maze		
7.1	In EVERY generated maze ALL squares must be accessible.				
8	After level 1 is completed the				

	maze size should be increased by 1 for each level after 1 (capping at 21 after level) 12				
8.1		Level 2	Should generate a new 11*11 maze		
8.2		Level 3	Should generate a new 12*12 maze		
8.3		Level 7	Should generate a new 16*16 maze		
8.4		Level 12	Should generate a new 21*21 maze		
8.5		Level 20	Should generate a new 21*21 maze		

Game Testing Table

Test number	Test Data (description)	Input (if needed)	Expected Outcome	Actual Outcome/ Evidence	Solution (if needed)
9	The number of keys that need collecting increases by 1 every 3 levels (capping at 4 keys)				
9.1		Level 1	Should start spawning in 1 key		

9.2		Level 4	Should start spawning in 2 keys		
9.3		Level 7	Should start spawning in 3 keys		
9.4		Level 10	Should start spawning in 4 keys		
9.5		Level 25	Should spawn in 4 keys		
10	The number of enemy AI monsters increases by 2 every 3 levels (caps at 7)				
10.1		Level 1	Should spawn no monsters		
10.2		Level 4	Should start spawning 2 monsters		
10.3		Level 7	Should start spawning 4 monsters		
10.4		Level 10	Should start spawning 6 monsters		
10.5		Level 13	Should start spawning 7 monsters		
10.6		Level 30	Should spawn 7 monsters		
11	The Number of lives should decrease by 1 every time you come in contact with an enemy				
11.1	Comes in contact with enemy 1st time		Should remove 1 life (remaining 2/3 lives)		

11.2	Comes in contact with enemy 2nd time		Should remove 1 life (1/3 lives remaining)		
11.3	Comes in contact with enemy 3rd time		Should remove 1 life (no lives remaining) should change scene to game over scene		

Enemy Testing Table

Test number	Test Data (description)	Input (if needed)	Expected Outcome	Actual Outcome / evidence	Solution (if needed)
12	Enemy AI				
12.1			Should patrol a random location if nit close to player		
12.2			If close enough to player, enemy should track and follow player		
12.3			After contact with player it should respawn at a random location.		

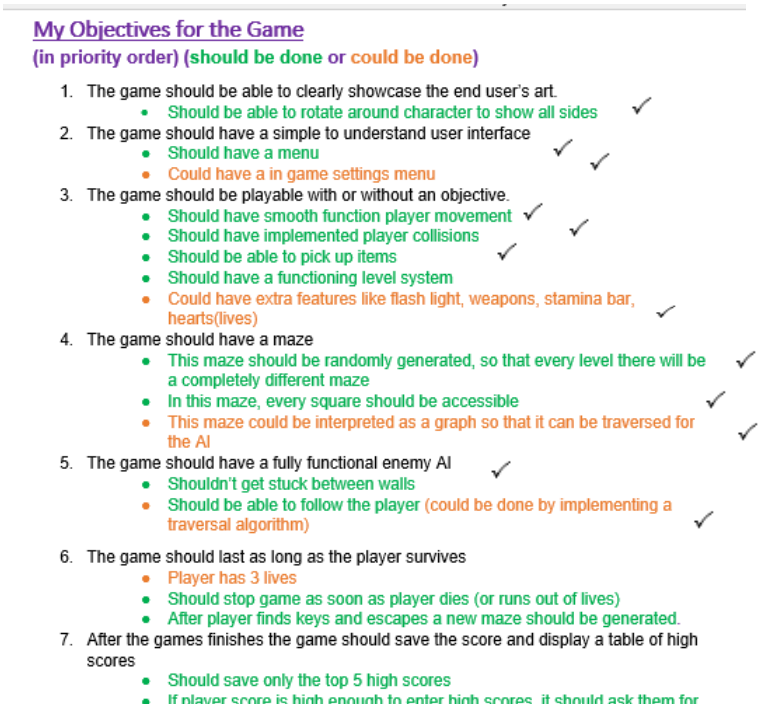
High Score Testing Table

Test number	Test Data (description)	Input (if needed)	Expected Outcome	Actual Outcome / evidence	Solution (if needed)
12	High scores				
12.1	Clicking high scores on game over menu		should show display the table of high scores		
12.2			Table of high scores should be in order of		

			descending scores.		
12.3	Getting Name and score		Should set name as the user input at the start and score to level lasted.		
12.4	Saving to file		Should save the name and score together in a json file		
12.5	Max entries		The table should not exceed 5 entries		
12.6	Replace Entries		After all, 5 entries are full. If the player gets a score higher than one of the high scores, it should replace the lowest entry and resort the table		

Evaluation

Comparison with objectives



My Objectives for the Game

(in priority order) (should be done or could be done)

1. The game should be able to clearly showcase the end user's art.
   - Should be able to rotate around character to show all sides ✓
2. The game should have a simple to understand user interface
   - Should have a menu ✓
   - Could have a in game settings menu ✓
3. The game should be playable with or without an objective.
   - Should have smooth function player movement ✓
   - Should have implemented player collisions ✓
   - Should be able to pick up items ✓
   - Should have a functioning level system
   - Could have extra features like flash light, weapons, stamina bar, hearts(lives) ✓
4. The game should have a maze
   - This maze should be randomly generated, so that every level there will be a completely different maze ✓
   - In this maze, every square should be accessible ✓
   - This maze could be interpreted as a graph so that it can be traversed for the AI ✓
5. The game should have a fully functional enemy AI ✓
   - Shouldn't get stuck between walls
   - Should be able to follow the player (could be done by implementing a traversal algorithm) ✓
6. The game should last as long as the player survives
   - Player has 3 lives
   - Should stop game as soon as player dies (or runs out of lives)
   - After player finds keys and escapes a new maze should be generated.
7. After the games finishes the game should save the score and display a table of high scores
   - Should save only the top 5 high scores
   - If player score is high enough to enter high scores, it should ask them for

1. The game be able to clearly showcase the end user's art (completed 1/1)

* Should be able to rotate around character to show all sides. (COMPLETED) ✓

I attached a camera object to the player object so that whenever the player moves the camera will follow. As well as this I assigned the camera to correspond with mouse input. So that moving the mouse will rotate the camera around the character, showcasing all its artistic features which that end users wanted to show.

2. The game should have a simple to understand interface (completed 2/2)

* Should have a menu COMPLETED) ✓

When the game loads, the player with first be greeted by a menu screen with 3 options (play, controls and quit) after receiving feedback from my public testing I decided to add the option to view the high scores from the starting menu

* Could have an in-game settings menu. COMPLETED) ✓

Pressing esc key whilst playing the game will open the in-game settings menu, where the user will have the option to return to main menu, check the controls and quit the application. Pressing e whilst in the in-game menu will return you to the game screen. Unfortunately, I wasn't able to make the game pause whilst in the in-game menu so pressing esc will still carry the game on meaning the playing can still lose lives even whilst in this menu.

3. The game should be playable with or without an objective (completed 4/5)

* Should have smooth functioning playing movement COMPLETED) ✓

in my game I have added the basic W-A-S-D movement system to move forward – left – down – right, as well as this I added features to allow the player to sprint and have incorporated a stamina bar to go with this. Pressing left shift will make the player sprint, as well as making the stamina bar go down at the same time. Pressing left ctrl will slow but the player down but will increases the rate at which the stamina bar recharges.

*Should have implemented player collisions COMPLETED) ✓

To my objects (such as player model, enemy model and walls) I have added in a Rigid body component built in unity. I also gave the walls box colliders and the player and enemy mesh colliders. I started to create a specific script for the player collision however I found it to be faster and more effective to use unity's built in components.

*Should be able to pick up items COMPLETED) ✓

There are a certain amount keys around the maze that the player needs to pick up. I have made it so that based off the number of keys that need spawning per level, that many keys will get loaded from the resources folder and each key will be transformed to a random square within the maze. I have also added a box collider to the

keys and wrote a program where if the player collides with the key, it will delete the key object and remove one key from the HUD (which shows how many more are needed to collect).

*Should have a functioning level system

Per level there is a set number of keys to collect (starting with 1) and every level that is a multiple of 3 the number of keys needed is incremented by 1 (up to a max of 4 keys). E.g. level 3- 2 keys, level 6 -3 keys, level 9- 4 keys. As well as this there are enemies in the maze which the player needs to avoid. To progress on to the next level the player needs to collect the keys needed to unlock the elevator and step inside the unlocked elevator to progress on to the next level. This game should last as long as the player survives to give a more arcade element to it, however there was a bug at level 9 which dropped the player under the maze. I was unaware to fix this bug as I was unsure about what was causing it and as well as this I was restricted by time. So, because of this bug I am not counting this objective as completed.

*Could have extra features like flash light, weapons, stamina bar, hearts(lives) (PARTIALLY COMPLETED) ✓

I was able to implement a stamina bar which is affected sprinting. Sprinting makes the bar go down, if empty the player won't be able to sprint unless you wait until the bar charges back up or unless you hold left ctrl to charge it faster. I have also implemented lives. The player will have 3 lives, colliding with an enemy reduces one life. Losing all 3 lives will cause the game to end.

4. The game should have a maze (completed 3/3)

*This maze should be randomly generated so that every level there will be a completely different maze. COMPLETED) ✓

As mentioned in the design section, there is a method of randomly generating numbers where each number (between 0-3) is assigned to a specific type of wall. This is used to create a randomly generated maze and since the maze size increases every level, no two level will have the same maze.

*In this maze every square should be accessible COMPLETED) ✓

Also mentioned in the design section, I use a traversal algorithm to make sure that from the top left, every square is accessible in some way. If not, walls are broken to make sure that certain squares are accessible. No matter what level you are on, every square of the mazes will all be accessible.

*This maze could be interpreted as a graph so that it can be traversed for the AI COMPLETED) ✓

I have created a graph class which would help convert the maze matrix into a graph, where each square is a node and the different ways of accessing these squares are represented by edges.

5.The game should have a fully functioning AI (completed 2/2)

*shouldn't get stuck between walls COMPLETED) ✓

One concern I had when planning the game was that since it is a maze, the enemy could get stuck between walls and would remain there until the next level. However, since the edges show which paths the enemy could take, it doesn't get stuck between walls. When the maze is converted into a graph the graph will only show the possible routes to each square

*Should be able to follow a player (could be done by implementing a traversal algorithm). COMPLETED) ✓

To make this system high level I didn't want the enemy to move randomly, but instead wanted it to track and move towards the player. I implemented this by making the enemy perform breadth first traversal every frame to find out where to move next, using the players location as the target.

6.The game should last as long as the player survives (completed 3/3)

*Player has 3 lives COMPLETED) ✓

The Player has 3 lives. Coming in contact with an enemy makes you lose a life.

*Should stop game as soon as player dies (or runs out of lives) COMPLETED) ✓
As soon as all three lives have gone, the game switches over to the game over menu where you can see you high scores.

*After player finds keys and escapes a new maze should be generated. COMPLETED) ✓
Collecting all the necessary keys and entering the end lift, creates a new maze with the maze size 1 bigger than the previous maze. It then randomly spawns in enemies and keys, then teleports the player to the start lift to start the level.

7. After the games finishes the game should save the score and display a table of high scores (completed 3/3)

*Should save only the top 5 high scores COMPLETED) ✓
The high score table is only made for top 5 scores once it reaches it limit of 5 it will start removing lower scores and replacing them with higher scores.

*If player score is high enough to enter high scores, it should ask them for their names. COMPLETED) ✓
In the main menu it asks the player to input their name, the game will not start until the player enters their name.

*Should save the score corresponding to the input given by player COMPLETED) ✓
If the players score is high enough to enter the high score table it uses the name that the player inputted at the start.

Transcript of End-Users Post-Development Review

In addition to giving the executable file to the end user, I was giving a text file where I told him about the different scenes in the game and told him to keep these different scenes in mind when reviewing the game as well as informing my end user about the bug that makes the game unplayable after level 8.

(Douy Singsamaran = DS)

Me: Have you played the game and were there any issues setting the game up?

DS: Yes, I have played the game, the setup wasn't too difficult all I had to do was unzip the file and run the exe file.

Me: What is your complete review of the game?

DS: Firstly, opening the application, I was greeted by a menu screen where I was happy to see some of my art being showcased. It was fairly simple to navigate through the menu. I like that there is an option for people who aren't aware of the controls to open the controls menu and view the controls of the game before they play. I was

asked to enter my name before starting the game. After entering my name and clicking play the game loaded pretty much instantly and the first thing I saw was the player model surrounded by walls. The objective of the game wasn't quite obvious, however eventually I was able to understand what I had to do. The key picks up was fast and there was no delay when picking up the key. The movement of the player was smooth and fluid, no lag or frames issues. I admire that fact that sprinting increases the field of view, I think this helps because the player can see more of their surroundings and can be aware of where and what they're running in to. You said that every level a new maze is randomly generated, it seemed like this was true, however as the player there is no way of seeing if this is definitely true. But based on my experience, no two levels were the same. In the in-game menu I liked how players were able to revisit the control menu mid game and also had option to return to menu. However, whilst in the in-game menu, I expected the game to be paused, but I was quite confused when I returned to the game to see I had lost one of my lives. I think opening the in-game menu should pause the game as it gives players time to do whatever they have to do and return to the game when comfortable without losing any of their progress. I think as the game went on to higher levels, it may have gotten a bit too difficult, with the number of keys increasing as well the number of enemies and the size of the maze, I found myself spending around 10 minutes per level. This is something that could potentially put someone off playing the game. Also, I think there should be a bit of a delay after completing a level, because as soon as you step into that last square, you instantly teleport and start the next level. I was made aware about the game becoming unplayable after level 8, due to a bug so I intentionally gave away the rest of my lives to see how the system would be after the game finished. After losing my final life I was instantly greeted by a game over screen, with a choice of 3 options. "play again", "quit" or "high scores". After clicking high scores, I was able to see my name along with the round that I lasted. I'm assuming that the scores are saved locally and are different for each computer as I was only able to see scores from the games that I played. I also noticed that the high score table was quite bland and would prefer it stand out more and be more eye catching.

Me: You said you want this game to be like Pac-Man but with a twist, do you feel like that has been achieved?

DS: I feel like this has been achieved, although there is still room for improvement. The base of the game is still running away from monsters in a maze, which gives it similarity to Pac-Man, however I would've liked it if the game had more extra features such as weapons and features like a battery powered flash light which could only be powered by collecting cells in the map. Nevertheless, there are still features that differ it from Pac-Man such as collecting keys and escaping to a new level and the score being based on rounds and not points. I also stated that I wanted the game to be endless and even though there was a bug after level 8, I think if there was no deadline for this system, this bug could be fixed and the game can continue to be endless.

Me: Do you feel like your art has been showcased the way you wanted it to be in this system?

DS: Definitely, if not its been done better. In the menu's my art is shown. In the game wherever you look around my art is shown. Also, the game being 3rd person, the

player would always be able to see the character model. In this system, no matter which screen you are on, you'll always be able to see my art. This is exactly the way I wanted it to be and would an excellent way to showcase my art work as well as an great addition to my portfolio.

Me: What improvements do you think can be made to the system?

DS:

Improvements I could make to the system if I had time.