BCS Digital Industries Apprenticeship

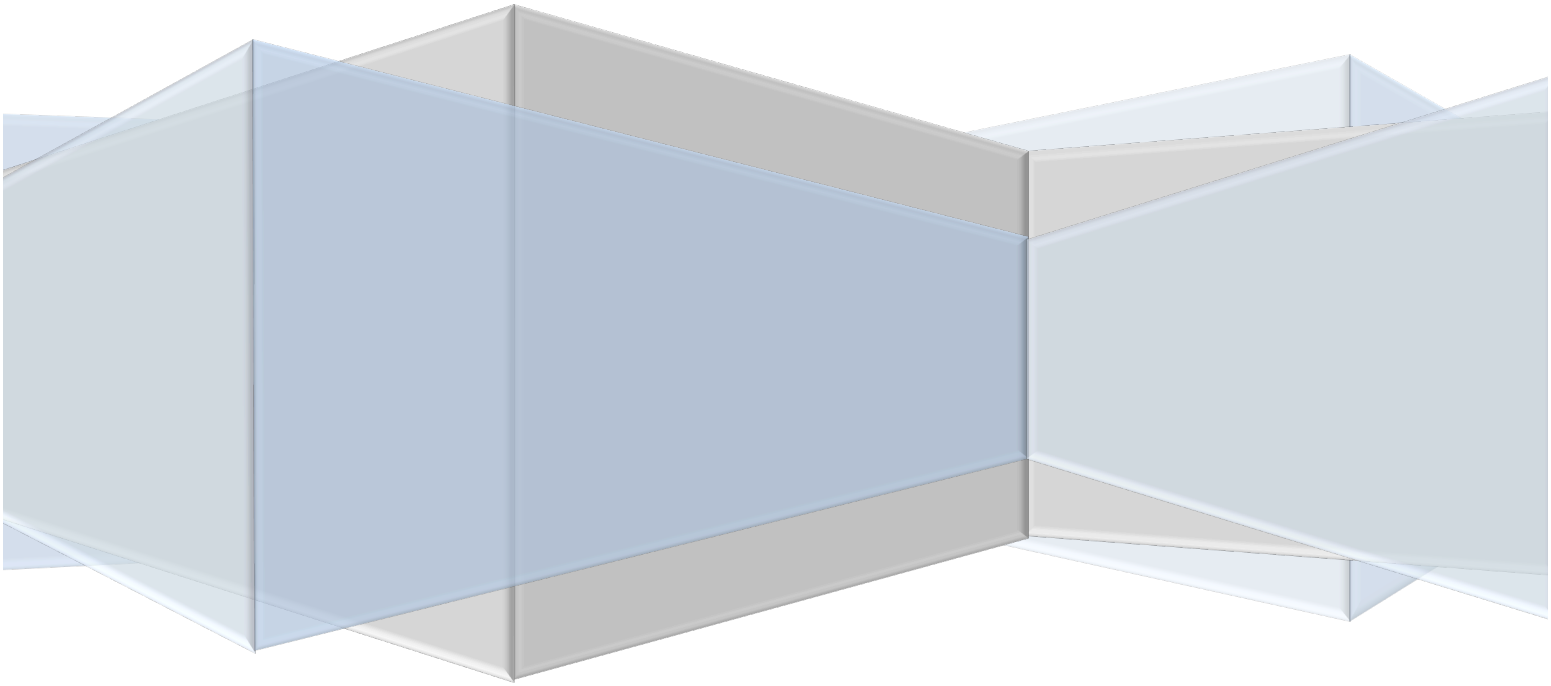
Project B – Maze Game

Software Development Technician

OCTOBER 15, 2018

Student Name : Jason Fernandes

Tutor : Vy Tran



Contents

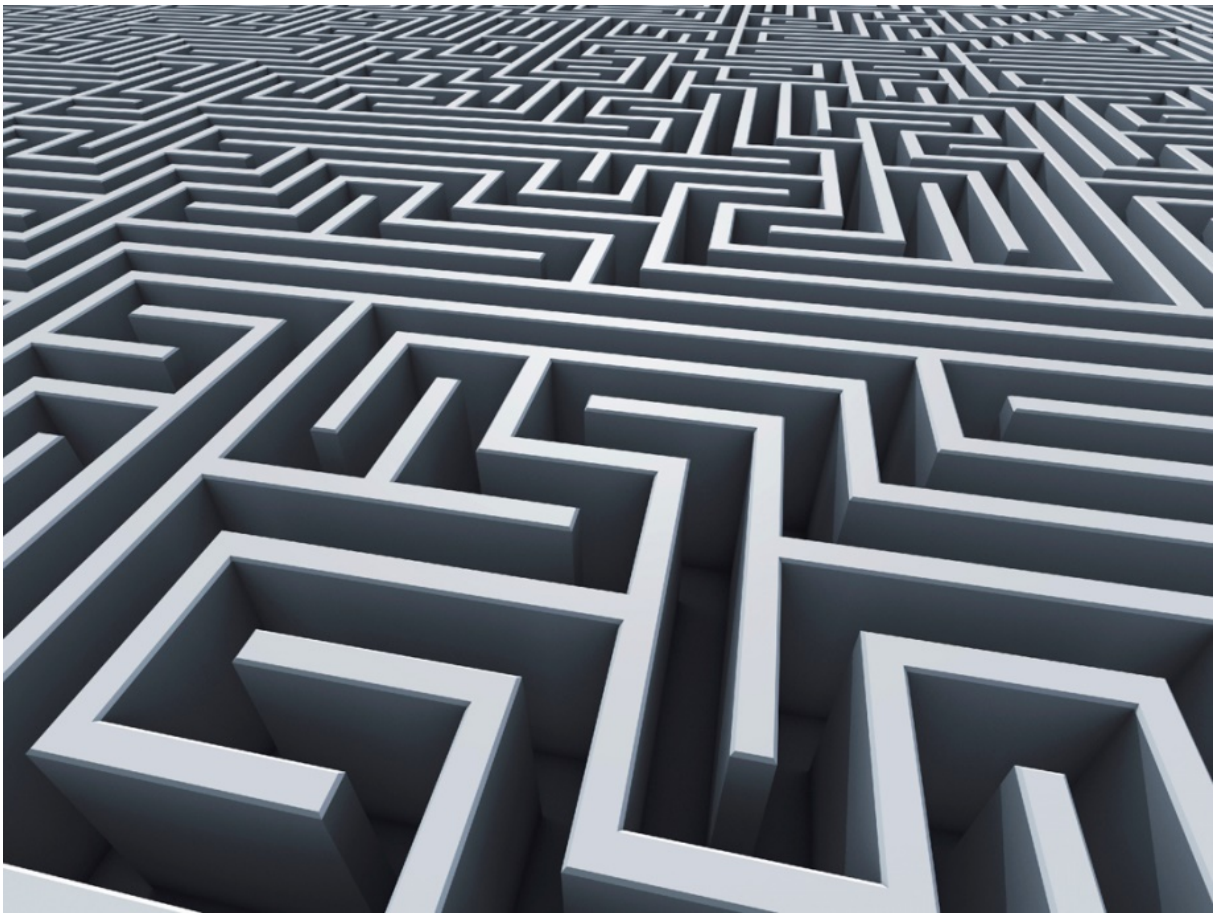
Introduction

Project Brief

In order to complete the apprenticeship, my line manager: Katie Hirschovits assigned me “Project B – Maze Game.” This project would entail planning, designing, implementing and documenting a maze game in order to increase the number of visitors that click on Olde Worlde Phunne website. As said in the initial brief, the steps that should be taken are outlined as follows:

1. Review all the key information and create a design for the maze game
2. Construct the maze game in accordance with the design
3. Test the maze game meets its requirements
4. Document what you built

The reason why Olde Worlde Phunne, a gaming company has suggested to create a maze game is because they want this program to be fun and interactive, in which will entice users to their website.



Purpose of the document

I am creating this document because it will show the evolution of the maze game from start to finish. The reason why Olde Worlde Phunne wanted a Maze game to be created is because it is a fun, interactive way in which will entice users to visit their website and consequently look at the other games in which they provide. I have chosen to develop this game on Python 3.6 because I am most proficient on this software, as well as the client, Olde Worlde Phunne have said that they would like me to use Python, since their current programmers use Python and therefore if editing needs to be done in the future, they will have the skills/be equipped to carry out the necessary tasks.

The reasoning behind the Maze game is because it is a puzzle that requires logical thinking in order to win. Furthermore, it also requires being able to predict the next step as in the Maze game, the user will be required to choose the direction in which they want to travel to, in order to try escaping the maze.

Scope

As Olde Worlde Phunne “want to increase the number of visitors to their website by offering a free maze game”, this consequently indicates that this game will be released to all users whom choose to visit Olde Worlde Phunne website. To get full access to the game, potential customers will need to sign up and create a new login on the website.

Who with?

I will be conducting all of the stages in the software development life cycle alone for this project; however, I will be regularly documenting all of the required information within this document. The reason for this is because, if improvements/edits need to be made, then a developer will be able to follow this documentation in order to edit the program successfully.

Analysis

Functional Requirements

Functional requirements - concerned with system services such as the scope of the system, the necessary business functions and the required data structure.

Req Number	Functional Requirement	Reasoning
1	Specifies the text file that contains the maze configuration data and outputs it on the screen	As stated in the brief, the python program should retrieve the particular data within the text file which will determine the maze shape
2	The game starts only when the maze has been configured properly	If the maze game is not retrieved from the text file then the game should not run, since it would not work properly. Therefore an error message should appear when this occurs, so the user can try running the program again
3	When the game starts, the player is automatically launched in a random room within the maze	It is important that the player starts in a random room because if the user chooses to play the game multiple times, then each time is able to be different
4	When the game starts, the player should begin with no wealth	Player should start with no wealth, as when they make their own way through the maze they will receive points for the treasures found
5	Each room within the maze are connected to each other by pathways	In order to travel from one room to another within the maze, it is essential that pathways are created.
6	The game will report to the user which threats and treasures are to be seen in the current room.	As stated in the brief, the user should be able to see the current treasure or threat within the room they are in until they collect/deactivate the item
7	When Treasure is found/threat is defeated, it should be removed from that specific room for the remainder of the game	The game should update specifically when either the treasure is found/threat is detected, so the user is able to determine what their next move should be
8	When treasure/threat is found/deactivated it gets added/subtracted to the current wealth of the user	A calculation needs to be performed in order for the user to be awarded/deducted for finding the treasure/threat
9	All treasures and threats should be collected/eliminated before leaving the room	As said in the brief, all items should be retrieved before the maze game finishes, since otherwise there would be no point of a score.
10	A threat is eliminated by carrying out the Action that defeats it.	Since there is going to be both treasures and threats within the Maze game, it is essential that all threats are found and deactivated as said in the brief

11	A game may be started at any time during play and will reset the maze template from the text file randomly	The user should have the ability to start the game again if they choose to. By adding this, it will make the game more flexible and less rigid.
12	Each room may have 1 to 4 passages leaving in the directions of North, East, South or West	The number of passages available to take from each of the rooms will differ, thus making it harder to find the exit.
13	If the user chooses a direction in which there is no passage, the program should announce “You may not go that way”	As said in the brief, the program should indicate to the user that there is no passage the way they are choosing to go.
14	When exit pathway has been found, the game finishes	The maze game should end automatically once the user has found the correct pathway leading to the exit.
15	When the game is finished, the user’s final wealth should be calculated and outputted.	When the user has found the exit pathway, the game should automatically finish and output the final wealth of the user

Non-Functional Requirements

Non-functional requirements - Deals with system constraints on the proposed software such as operation, how easy it is to use, performance and security.

Req Number	Non-Functional Requirement	Reasoning
1	The game must be easy to use and play	As the game is aimed at new users, it must be easy to play otherwise the user will lose interest.
2	The game must be available 24 hours per day, 364 days per year	New users should be able to use the game whenever they load the website and on a desktop. Websites are able to be accessed as long as there is a valid internet connection. Therefore, the game should be readily available all the time.
3	The program should look into the text file and copy template of maze within 10 seconds	As the program imports the maze from a text file on to the graphical user interface, it is important that it doesn't take long to load otherwise the user will get bored for waiting
4	The game should not disclose any personal information about the user to the operators of the system when new users create a login due to data confidentiality	This is due to the data protection act rule, must be respected due to legal reasons
6	Two different types of users, - Free trial user who has access to a limited amount of the game. - Full Member – Access to the all parts of the Maze game	As said in the brief, this should be implemented at a later stage but it is very important in order to increase the amount of loyal visitors to Old Worlde Phunne
7	The game must be reliable and if unable to process the request then appropriate error message should show	This is important so the user is able to identify everything that is happening in order to run the maze
8	Yellow circles should be shown on the maze, when treasure/gold needs to be collected	As said in the brief, the maze must contain treasure. I have chosen for that treasure to be illustrated as a yellow circle within the maze
9	Red circles should be shown on the maze, when threats are present	As said in the brief, the maze must contain threats. I have chosen for that treasure to be illustrated as a red circle within the maze
10	A green square should be shown on the maze, which	In order to finish the game, I have chosen for one green square to be

	indicates the finish point that the user has to go to.	illustrated on the maze, so the user is able to identify where they have to go in order to finish the maze
11	There should be multiple threats within the maze, if touched then it should be game over.	In order to make the game challenging, there should be multiple threats, to make the game more interesting
12	A blue square represents the user playing the game. This should be able to move around the maze accordingly.	As the game requires a GUI, I have chosen for the players position to be illustrated as a blue square.

Choosing between platforms

In order to choose the best platform to deploy the maze game for Olde Worlde Phunne, I will conduct my own research into the platforms highlighting the features, advantages and disadvantages of each and ultimately implementing the Maze Game on that platform. The platforms in which I will be looking into further are World Wide Web and Desktop

- ### World Wide Web

Advantages and Disadvantages of the World Wide Web:

Platform	Advantages	Disadvantages
Web	**Worldwide access** Every user is able to access the game through the website **Always up-to-date** Accessing through a web platform would ensure that users are accessing the same URL [Unique Resource Locator], meaning that all employees will be able to see the most up-to-date version of the software. **No need to install/upgrades** Web-based applications do not require any installation/upgrades process. Moreover, they don't occupy space on the hard drive **Access from Mobile Device** A web application is adaptable through a mobile as web apps can be accessed from a mobile device without any issues. This makes employees able to sign in and monitor the software from their phones, since more and more people are using smartphones for their daily routines.	**Internet Reliance** Requires internet connectivity at all times to access the website. **Security** Data is less secure when on the web, as users from all over the world are accessing the same server hosted by a third party. However there are ways to strengthen security such as encryption, etc. **Reduced Speed** Likely to be a bit slower than an application hosted on a company's server, since the web platform is worldwide access. **Performance** Transmitting data via the internet rather than on a desktop changes the performance on how quick the tasks take. As the internet depends on the connection, users can be annoyed and easily irritated by the slower performance.

- **Desktop**

Advantages and Disadvantages of Desktop:

Platform	Advantages	Disadvantages
Desktop	**Enhanced Security** Since the user is keeping their data on their own computer systems, [as the application is running off the user's desktop], this makes it harder for hackers to gain access to the data, meaning that data is more secure when compared to the web platform. **No Internet Reliance Required** Doesn't require internet connectivity to access the application. **Performance** Typically one would find that a well-written desktop software running on a top-notch computer runs faster than a web-based application **Integration with other products** It is much easier to integrate products within a desktop application rather than a browser.	**Lack of Portability** Desktop applications are not portable and restricted to one location, which is the computer. The desktop application will need to be installed **Installation is necessary** Desktop applications requires to be installed on the desktop, as well as some place on the hard drive. The installation process is quick, however can easily irritate a user who is needing to access the application immediately

Recommendation: Desktop

I have chosen that a desktop computer would be the best way to deploy the maze game to new users. The reason for this is because it will be faster for users to access as it will not rely on an internet connection, therefore less glitches will occur. To add to this, as the graphical user interface requires lots of performance from the computer by displaying a 25x25 grid with a variety of colours - it is important that this is displayed correctly so that the user is able to play the game smoothly

Design

High Level Overview

- ### How the game should work

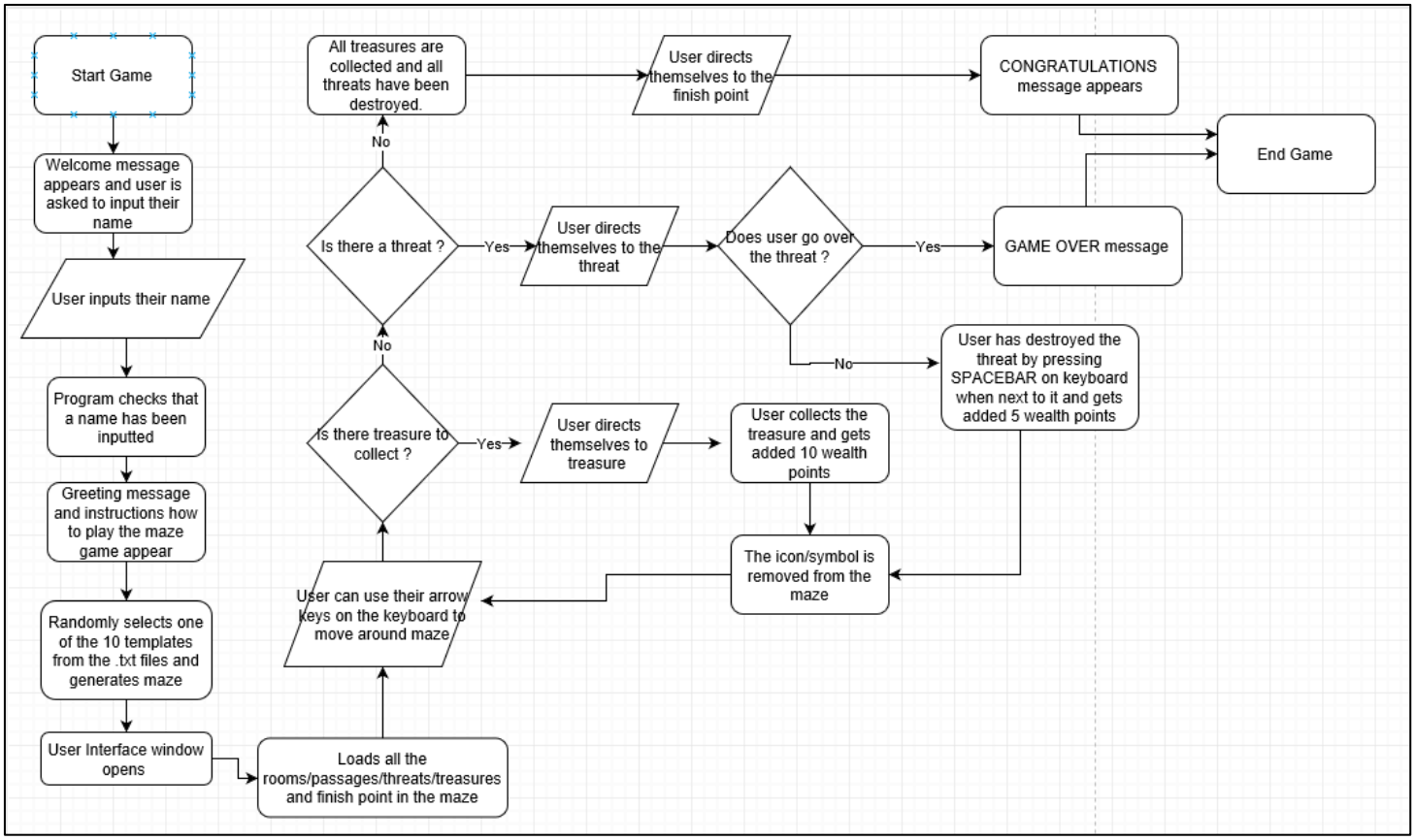
I will be creating a maze game on python that will aim to attract potential gamers to the website of Olde Worlde Phunne and consequently entice them to sign up/create a login to play the full version of the Maze game and play other games that Olde Worlde Phunne provides.

How the maze game should work is that when the program is run, a maze should be automatically configured from a text file and outputted to the user if it is within the correct format. Once the maze is created, it will allow the user to start playing, the default player being set to no wealth and dropped in a random room within the maze.

To make the game more interesting, Olde Worlde Phunne has asked to integrate a level of complexity in which items are found in each room. Both threats and treasures are scattered within the various rooms of the Maze and will enable the user to get either rewards or penalties towards their wealth. Every move that the user makes, they will be able to identify the current threats/treasures in that room, so that they will be able to make a judgement about which passage they should choose for their next move. It is mandatory that all threats are eliminated before finishing the Maze. The end goal is that the user identifies the correct pathway that leads them to the exit of the maze in which they are able to see their total wealth.

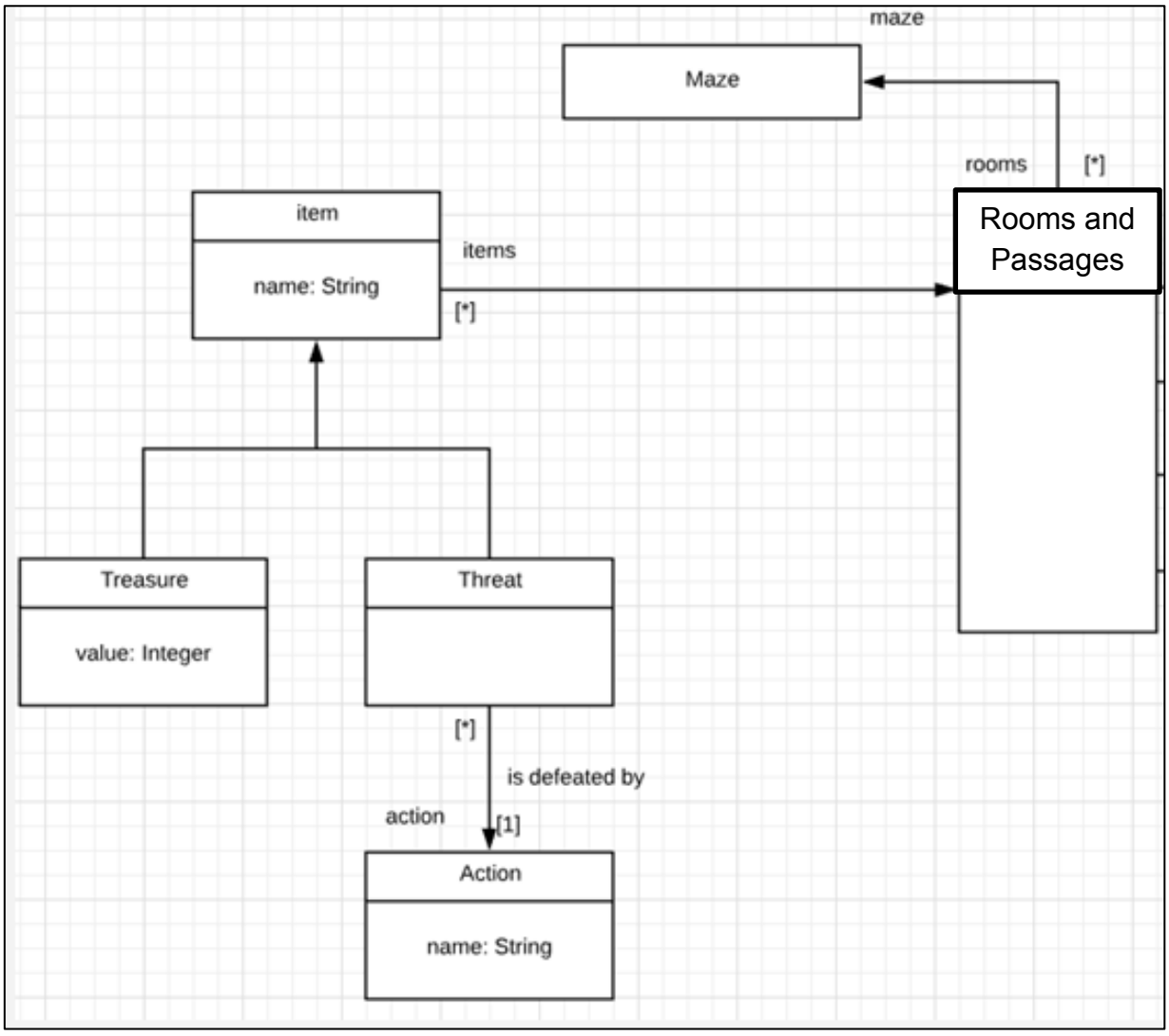
Flow Chart

Below is a screenshot of the flowchart which I have created for the maze game. I will be following these steps while coding my program since it is essential that the game contains specific bits.



Conceptual Model

Below is a screenshot of the conceptual model in which I have created/edited for the maze game. I will be following this model while coding my program since it is essential that the game contains specific bits. I have changed the original one from the brief, because in my game as the maze is represented by a graphical user interface the player is able to see where all the passages and rooms lead to, to ultimately try and escape the maze



Assumptions

There are a few assumptions that I have made in order to roll out the Maze game to customers of Olde Worlde Phunne. These are as follows:

- The game and text files has to be downloaded of Olde Worlde Phunne website
- User must have python 3.6 installed on their Desktop computer
- Target audience to play this maze game is kids [Age: 8-12]

Changes to the requirements

- All treasure and threats have to be collected and destroyed before the player is able to finish the game.
- Players movements should be bound to keyboards directional keys
- The players final time to complete the game should be outputted on the screen.
- The player should be able to see all of the rooms and passages within the maze
- The point system is illustrated in the table below:

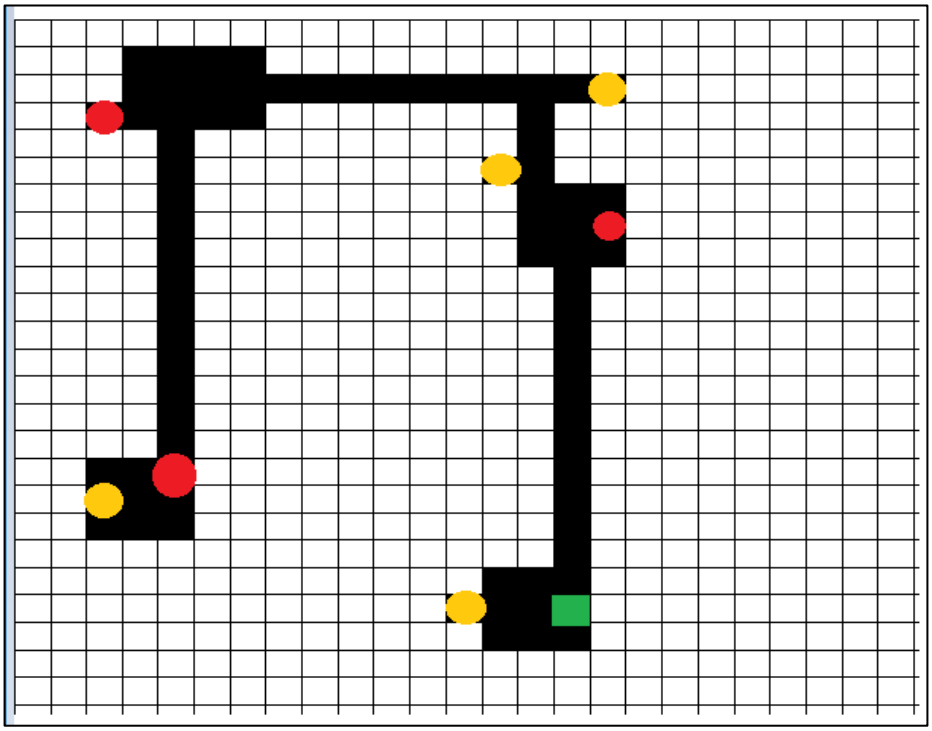
Action	**Result**
Collecting Treasure	Add 10 wealth points
Destroying Threat	Add 5 wealth points
Going into a Threat	End game automatically
Going to Finish point too early	Minus 10 points

I have made the above changes to the original requirements because it will make the maze more fun and consequently will increase the number of visitors that Olde Worlde Phunne gets on their website. As I have assumed previously that the game is aimed at kids [age 8 to 12] , they are able to compete against their friends for the highest score in the least amount of time like other games currently have. They time it has taken a user to complete the maze should be outputted on the screen as well as their wealth. Furthermore, it is proven that kids like to visually see a game, rather than play a text based game because they are attracted to moving objects, therefore I thought it would be suit this task better to create a GUI.

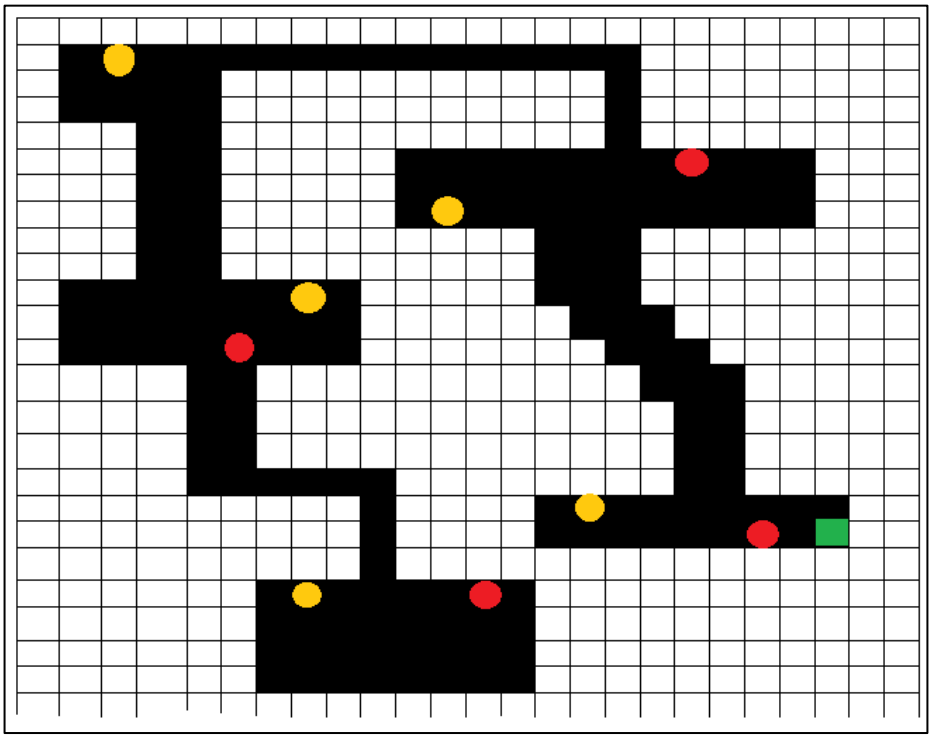
Sketches of User Interface

As I plan for the game to have a Graphical user interface, the program should choose a random text file out of a possible 10 files. Once it has done this, the configured data should be retrieved from the specific text file and outputted on the GUI when the program is run in python. Below are the designs for the 10 different mazes that I plan to create for this project, these have been developed in the application “Paint”.

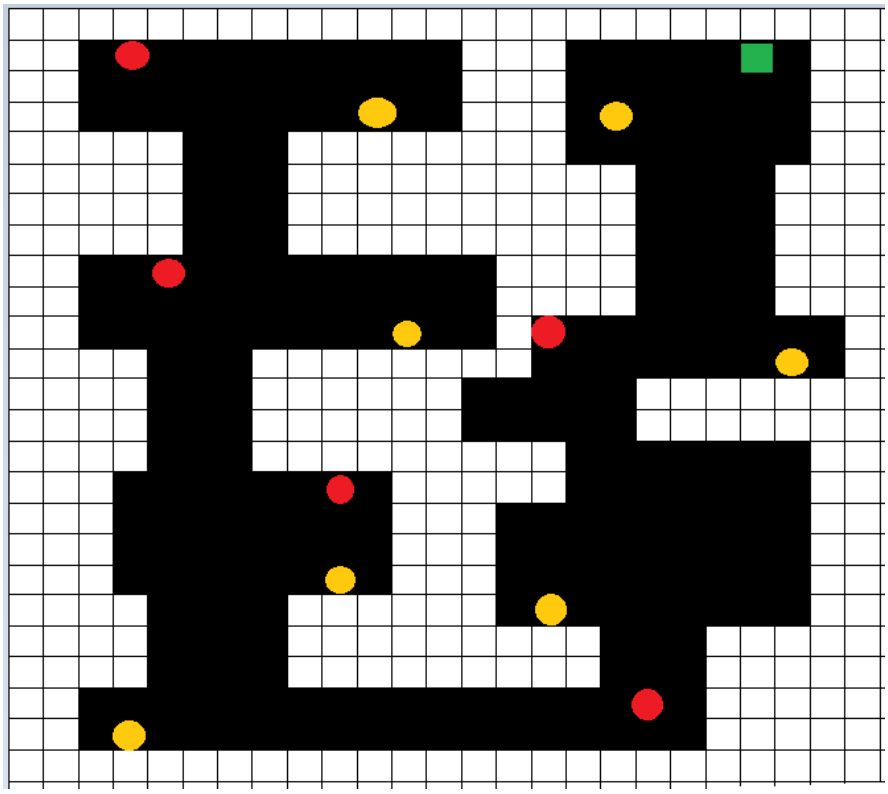
Maze template 1



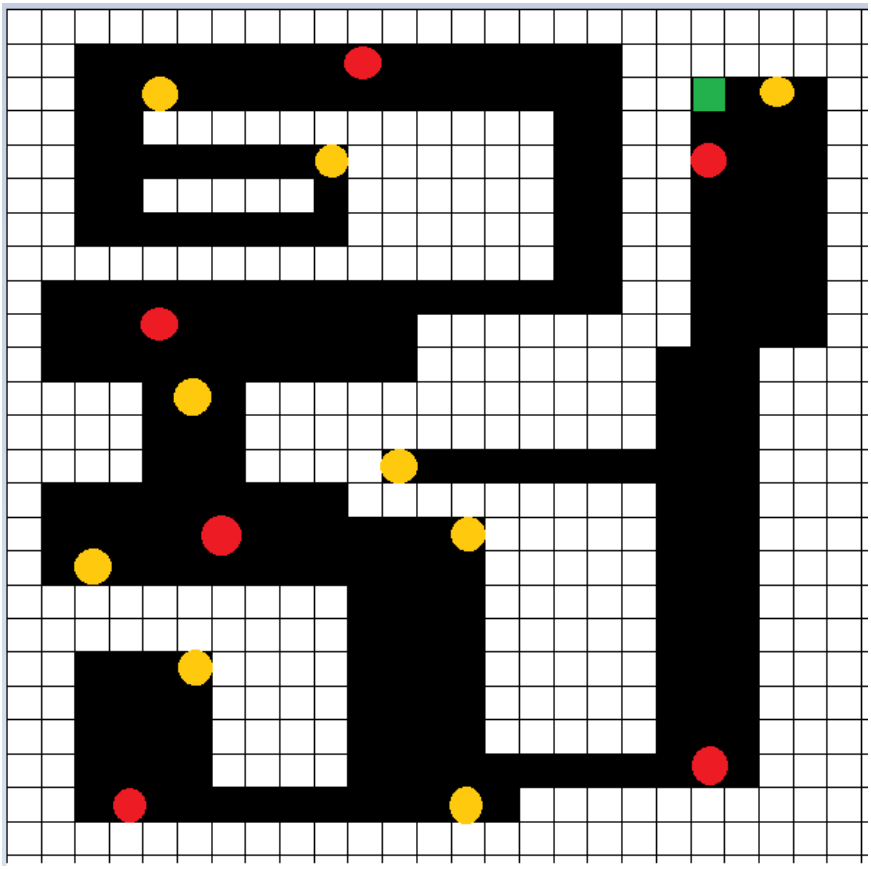
Maze template 2



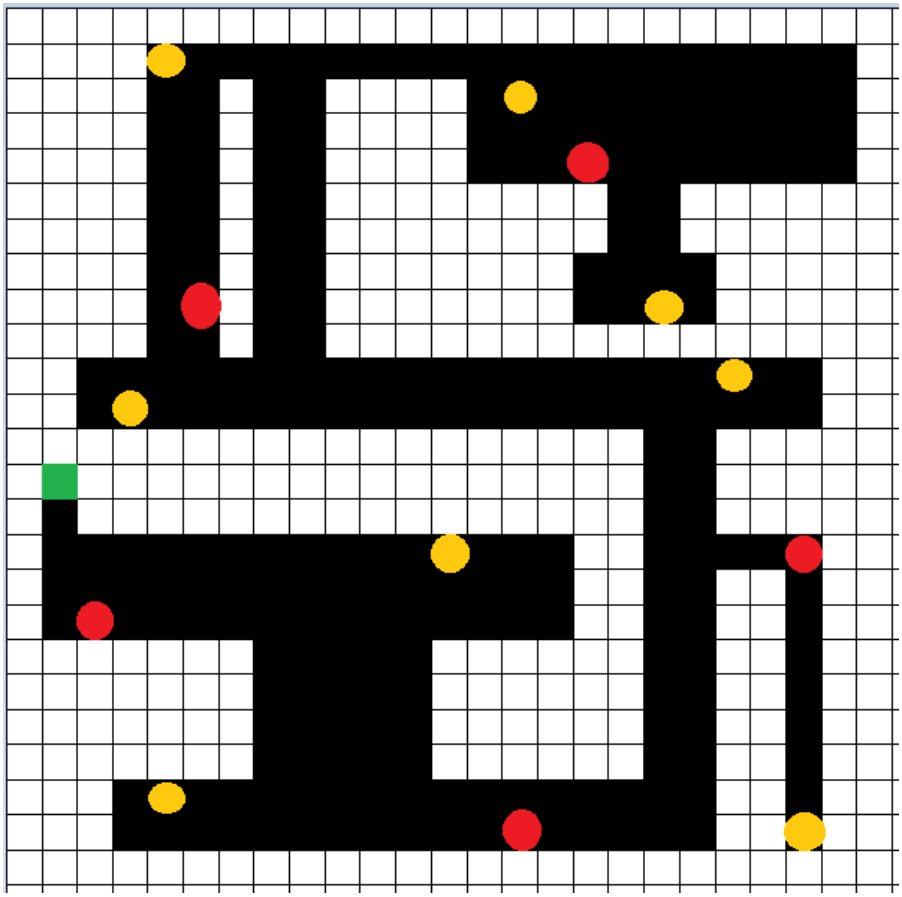
Maze template 3



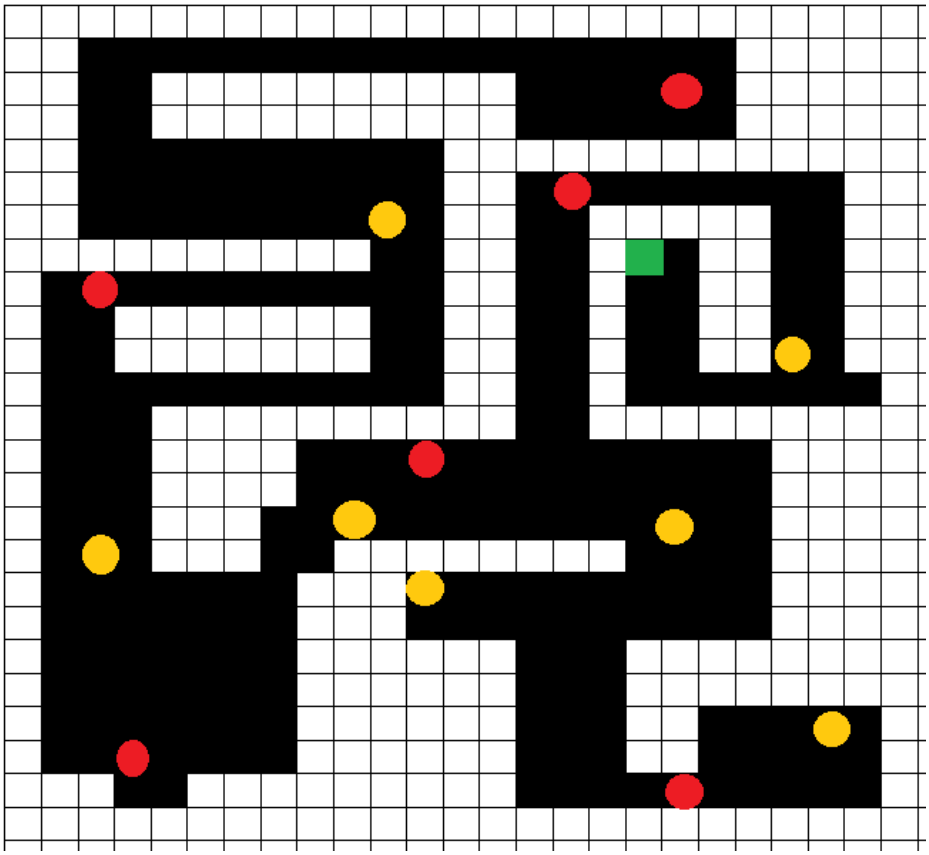
Maze template 4



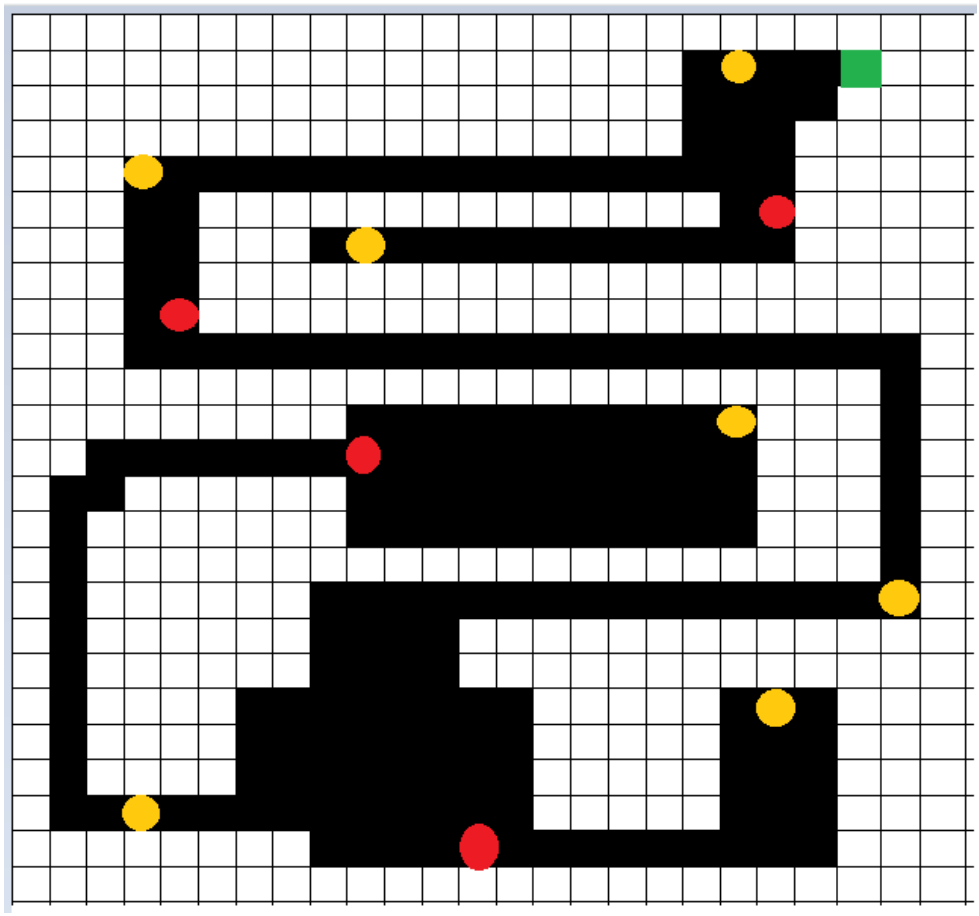
Maze template 5



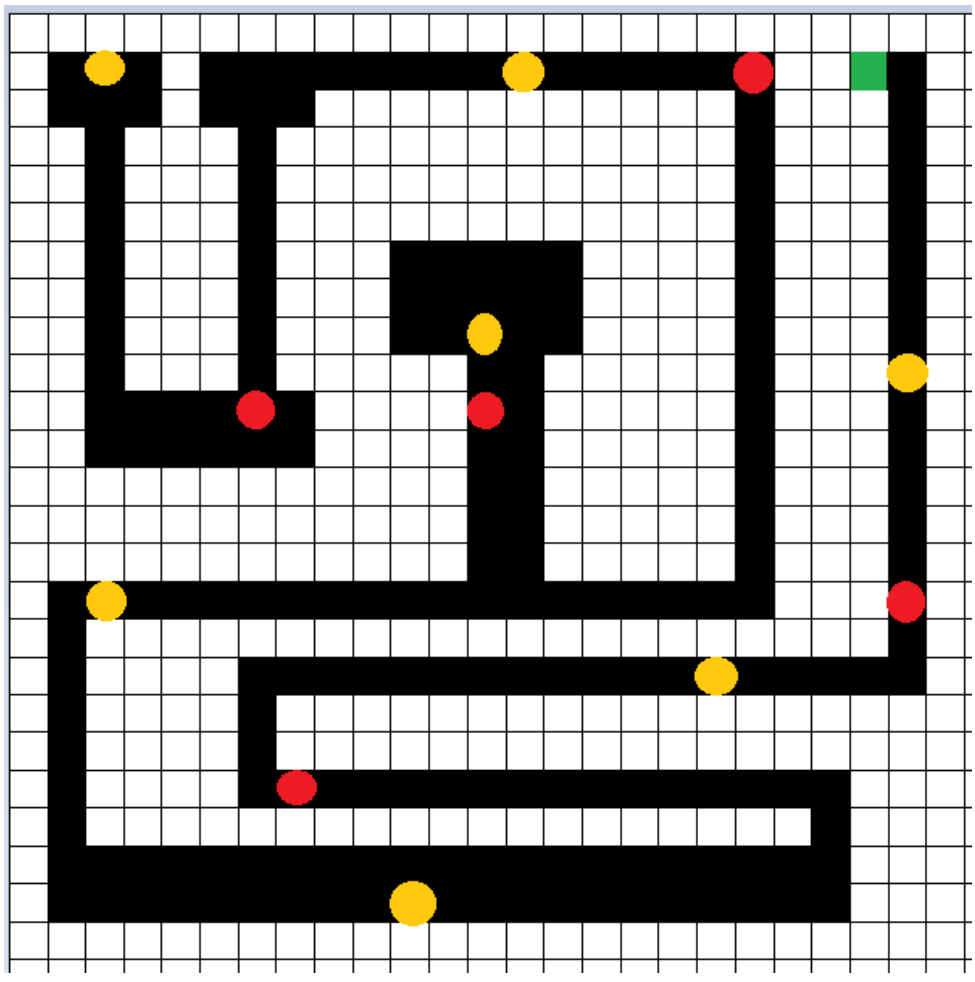
Maze template 6



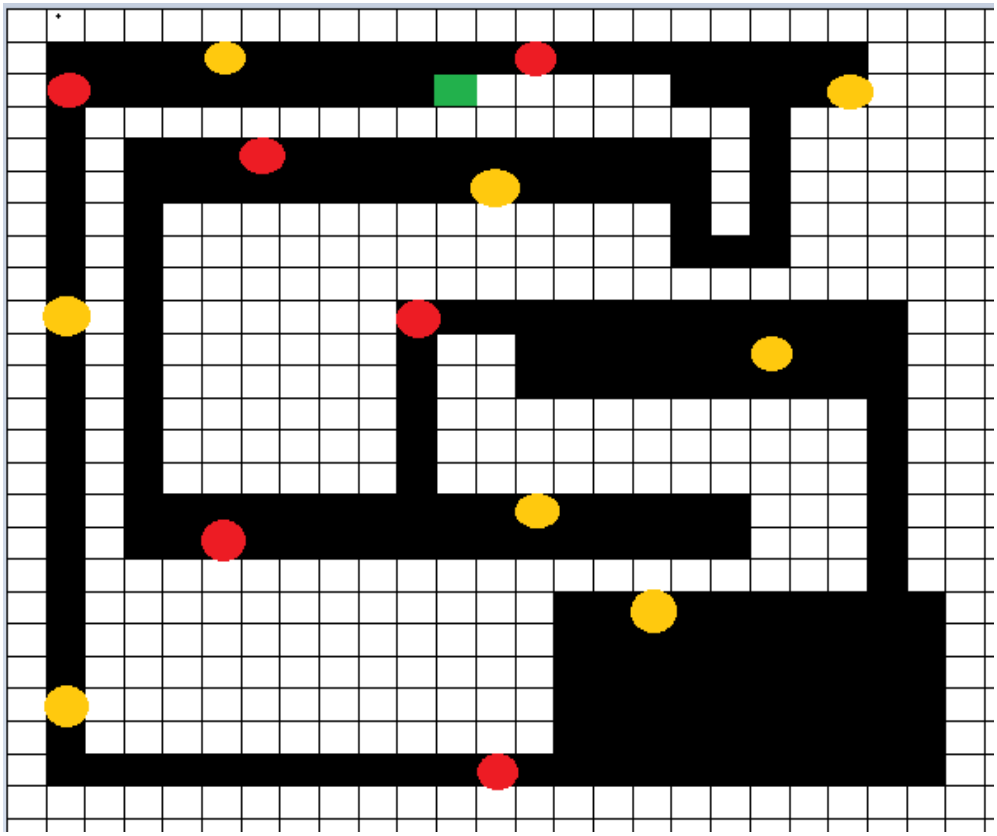
Maze template 7



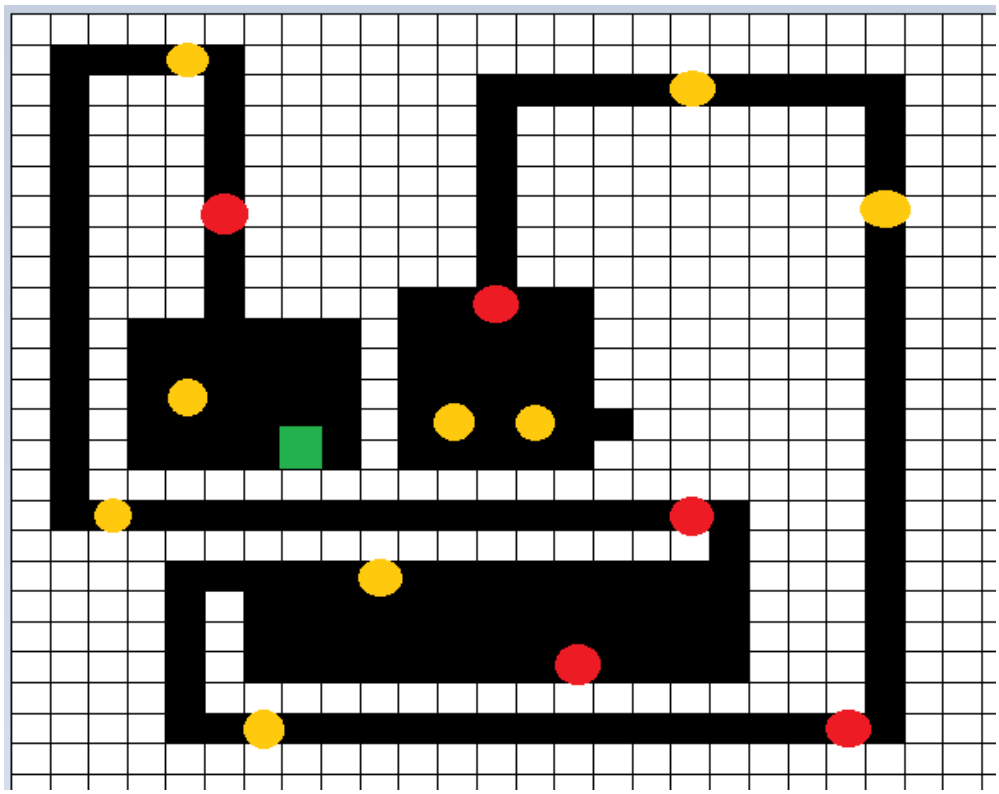
Maze template 8



Maze template 9



Maze template 10



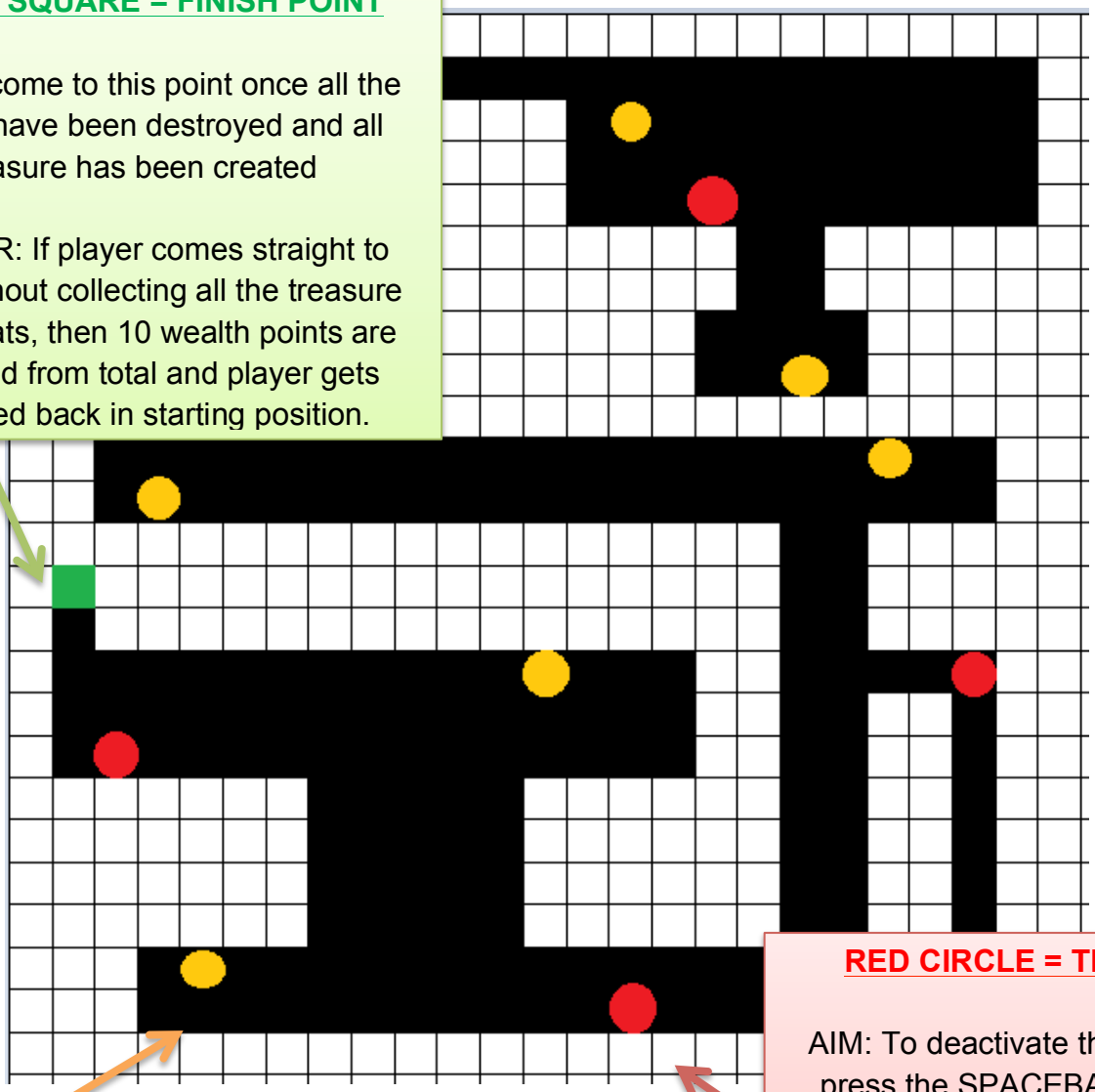
Each element of the User Interface

Below is an example of a maze which I have created, illustrating all the key components of the interface.

GREEN SQUARE = FINISH POINT

AIM: To come to this point once all the threats have been destroyed and all treasure has been created

DANGER: If player comes straight to finish without collecting all the treasure and threats, then 10 wealth points are deducted from total and player gets launched back in starting position.



YELLOW CIRCLE = TREASURE

AIM: To collect the gold by going over it, using the arrow keys on keyboard to navigate. Earn 10 wealth points

RED CIRCLE = THREAT

AIM: To deactivate the threats; press the SPACEBAR on the keyboard when player is next to it. Earn 5 wealth points

DANGER: If player touches the threat, then it is game over and a message is outputted to user

*Each maze will be 25x25 grid

*The player will be able to see their own position on the maze by looking at a blue square. The blue square is not included in any of the maze drawings because the player is randomly placed in any room of the maze

= PLAYERS CURRENT POSITION

Implementation

Explaining the code

Importing the various libraries needed for the code to function.

Turtle – To draw objects on screen

Math – To calculate distances between a player and object

Random – To randomly choose a level

Time – To calculate how long a user has spent on the game

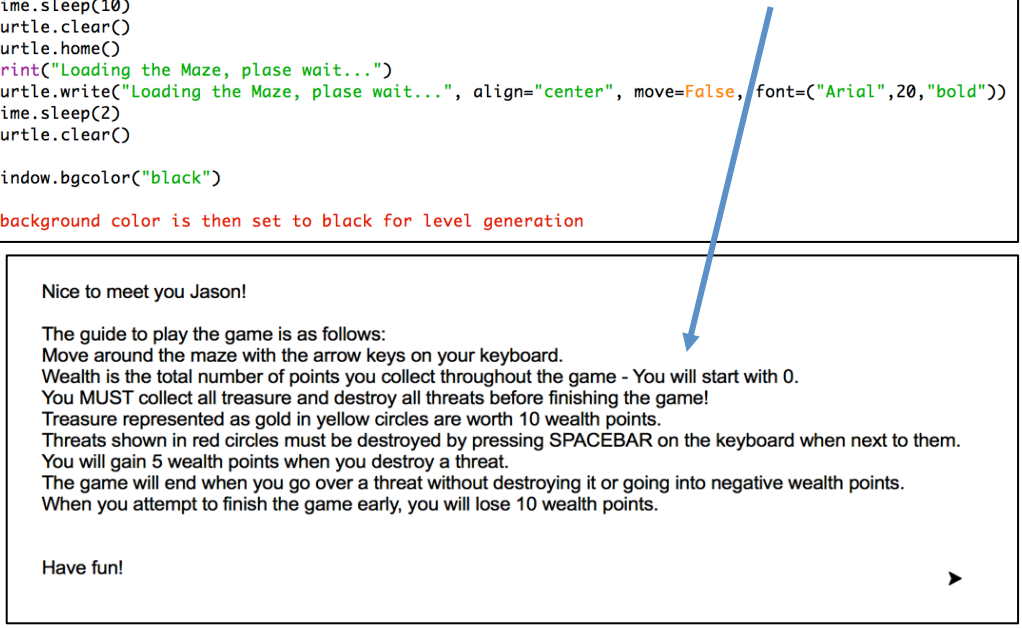
- Section 1

```
#IMPORTING the various libraries needed to make this game run
import turtle
import math
import random
import time


#Make the game loop when you lose or win by setting game_lose_loop to true
game_loop = True

while game_loop:

    #-----------------SECTION 1-----------------#
    #Where the game window and settings are configured

    print("Time for a game! Loading configuration..")
    start_time=time.time() #Start the timer count using the time function


    #creating a new screen
    window = turtle.Screen()

    window.clearscreen() #reset window from other games

    #How big the window should be
    window.setup(1200,700)

    #Name window 'Jason's  maze game'
    window.title("Jason's maze game")


    #set backbground to white

    window.bgcolor("white")
    turtle.color("black")
    turtle.clear()
    turtle.home()
    turtle.penup()
    #Various message to the user is displayed here
    print("Welcome message displayed")
    turtle.write("Welcome to the Maze Game!", align="center", font=("Arial",20,"bold"))
    playername=str(window.textinput("Enter your name", "What is your name?"))
    time.sleep(2)
    turtle.clear()
    turtle.home()
    print("Player greet and instructions displayed")
    turtle.write("Nice to meet you "+playername+"!\n\nThe guide to play the game is as follows: \nMove aro
    time.sleep(10)
    turtle.clear()
    turtle.home()
    print("Loading the Maze, plase wait...")
    turtle.write("Loading the Maze, plase wait...", align="center", move=False, font=("Arial",20,"bold"))
    time.sleep(2)
    turtle.clear()

    window.bgcolor("black")

    #background color is then set to black for level generation
```

This is the first major section of the code. The window settings are defined and titled as “Jason’s maze game”. It is then given a white background for various messages to be displayed to the user. The user is also asked for their name.

Setting a game loop by default to true means the game will always keep playing unless set to False.

This is where the various messages are displayed to the user. The welcome message and other messages are displayed in Arial font size 20.



Once these messages are displayed, all writing is cleared and the background colour of the window goes to black.

- Section 2 – Defining different classes to be used in the game

The wall class code

The wall class will be drawn by a turtle. It is defined to be a square shape and have a colour that is white. When it is being added to the screen, the pen will be up so that a trail will not be left behind.

```
class Walls(turtle.Turtle):
    def __init__(self): #referring to the object that will be called on
        turtle.Turtle.__init__(self) #initialise pen
        self.shape("square") #shape of the person
        self.color("white") #color of the person
        self.penup() #By default, a turtle leaves a trail behind, we don't want this
        self.speed(1000) #Animation speed
```

The wall class output

The output of the wall class is simply seen below

The player class code

```
class Player(turtle.Turtle):
    def __init__(self):
        turtle.Turtle.__init__(self)
        self.shape("square")
        self.color("blue")
        self.penup()
        self.speed(0)
        self.gold = 0 #defining the gold player has
        self.touchedthreat = 0 #if this is 1 then the game will end and loop
        self.win = 0 #If this turns to 1 then the game will end and loop

    #Defining the movement of the player##
    #Going up is a positive y coorindate
    def go_up(self):
        move_x = player.xcor()
        move_y = player.ycor() + 24
        #if where the player will move to somewhere that is not a wall, then it will allow them to move
        if (move_x, move_y) not in wall_coords:
            self.goto(move_x, move_y) #Y cor is vertical so + is up

    #Going down is a negative y coordinate
    def go_down(self):
        move_x = player.xcor()
        move_y = player.ycor() - 24
        if (move_x, move_y) not in wall_coords:
            self.goto(move_x, move_y)

    #Going left is a negative x coordinate
    def go_left(self):
        move_x = player.xcor() - 24
        move_y = player.ycor()
        if (move_x, move_y) not in wall_coords:
            self.goto(move_x, move_y)

    #Going right is positive x coordinate
    def go_right(self):
        move_x = player.xcor() + 24
        move_y = player.ycor()
        if (move_x, move_y) not in wall_coords:
            self.goto(move_x, move_y)

    #Defining what would count as an object being 'touched' by a played
    def touched_object(self, other): #other would be the object concerned such as threat or treasure
        a = self.xcor()-other.xcor()
        b = self.ycor()-other.ycor()
        distance = math.sqrt((a ** 2) + (b ** 2) )

        #if the distance between the two objects is less than 5, then the object has been 'touched' and returns True
        if distance < 5:
            return True
        else:
            return False

    def near(self, other): #other would be the object concerned such as threat or treasure
        a = self.xcor()-other.xcor()
        b = self.ycor()-other.ycor()
        distance = math.sqrt((a ** 2) + (b ** 2) )

        #if the distance between the two objects is less than 5, then the object has been 'touched' and returns True
        if distance < 26:
            return True
        else:
            return False
```

The player object is defined as being a blue square. It also has the attributes gold, touchedthreat and win.

Each go routine has either an x or y coordinate change to the player. These routines will be called upon when the player presses any of the arrow keys which will be seen later.

For both touched_object and near routines they have similar objectives. The maths behind it is that a distance can be calculated when comparing the current player coordinates to the object in question, such as a threat or treasure.

An object is defined as ‘touched_object’ by the code when the player’s distance is less than 5 coordinates away.

An object is defined as ‘near’ when the player’s distance is less than 26 coordinates away.

The treasure class code

The treasure class code defines the object and characteristics that a treasure item on screen would have.

```
#Defining a new class for treasure that can be collected in game - it will be circle and gold
class Treasure(turtle.Turtle):
    def __init__(self, x, y): #referring to the object that will be called on + where we want the treasure to appear
        turtle.Turtle.__init__(self) #initialise pen
        self.shape("circle") #shape of the person
        self.color("gold") #color of the person
        self.penup() #By default, a turtle leaves a trail behind, we don't want this
        self.speed(0) #Animation speed
        self.gold = 10 #set the value of the gold
        self.goto(x, y)

    #Destroying a treasure hides its object and places it out of the screen
    def destroy(self):
        self.goto(2000, 2000)
        self.hideturtle()
```

The treasure object is defined as being a gold circle.

The destroy routine within the treasure class is when the treasure would be removed from the players view

The treasure class output

The output of when the treasure class is called can be identified as a yellow circle on the maze, as shown on the right hand side.

The threat class code

The threat class code defines the object and characteristics that a threat item on screen would have.

```
class Threat(turtle.Turtle):
    def __init__(self, x, y): #referring to the object that will be called on + where we want the treasure to appear
        turtle.Turtle.__init__(self) #initialise pen
        self.shape("circle") #shape of the person
        self.color("red") #color of the person
        self.penup() #By default, a turtle leaves a trail behind, we don't want this
        self.speed(0) #Animation speed
        self.gold = 5 #set the value of the gold
        self.goto(x, y)

    #defining destroying the threat
    def destroy_key(self):
        global threatscaught
        if player.near(self):
            self.goto(2000, 2000)
            self.hideturtle()
            player.gold = player.gold + threat.gold
            threats_coords.remove(self)
            print("BOOM! Threat removed! Player wealth points are now: " +(str(player.gold)))
            threatscaught = threatscaught + 1

        else:
            return
```

The treasure object is defined as being a red circle.

The destroy routine within the threat class is when the player is near the threat, presses the spacebar and therefore triggers this routine to occur

The threat class output

The output of when the threat class is called can be identified as a red circle on the maze, as shown on the right hand side.

The end class code

The end class code defines the object that would be see by the player. The end class also defines the routine that runs when the game ends and when the player attempts to end the game early.

The treasure object is defined as being a green circle.

When the game ends, the screen will be cleared and a congratulations message will be displayed (as shown below)

The player.win status is also updated to 1, which allows the game to end.

When the player tries to end the game early, the screen will turn white and let them know that they can't finish. (as shown below)

```
#Defining a new End point class - it will be circle and green
class End(turtle.Turtle):
    def __init__(self, x, y): #referring to the object that will be called on
        turtle.Turtle.__init__(self) #initialise pen
        self.shape("circle") #shape of the person
        self.color("green") #color of the person
        self.penup() #By default, a turtle leaves a trail behind, we don't want this
        self.speed(0) #Animation speed
        self.goto(x, y)

    #Defining the end routine - It will display a message to the player to congratulate them that they have won
    def ended(self):
        turtle.clear()
        turtle.penup()
        turtle.home()
        window.bgcolor("white")
        turtle.color("black")
        print("Displayed congratulations message")
        end_time_minutes=round((time.time()-start_time)/60,1)
        end_time_seconds=round(time.time()-start_time,1)
        turtle.write("Congratulations "+playername+"! You have reached the end!\n\nTotal player wealth points: " +(str(player.gold))+"\nTime
        time.sleep(10)
        turtle.clear()
        turtle.home()
        player.goto(player_coords)
        turtle.clear
        player.win = player.win + 1 #adding 1 to break the loop and start again

    #Defining the routine when the game has not ended, but when player attempts to end it early - it will inform the player what else they ne
    def not_ended(self):
        turtle.clear()
        turtle.penup()
        turtle.home()
        window.bgcolor("white")
        turtle.color("black")
        print("Displayed player attempted to finish early message")
        turtle.write("You cannot finish yet!\nYou have lost 10 wealth points\n\nPlease ensure that you collect all the treasure and destroy
        player.gold = player.gold - 10
        time.sleep(10)
        turtle.clear()
        turtle.home()
        print("Player wealth points: " +(str(player.gold)))
        turtle.write("Current wealth points: " +(str(player.gold))+"\n\nReturning to start point", align="center", font=("Arial",20,"bold"))
        player.goto(player_coords)
        time.sleep(5)
        turtle.clear()
        window.bgcolor("black")
```

The end class outputs

- **Ending a game completely**

Congratulations Jason! You have reached the end!

Total player wealth points: 70
Time taken to complete: 0.9 minutes and 55.8 seconds

- **Ending a game early**

You cannot finish yet!
You have lost 10 wealth points

Please ensure that you collect all the treasure and destroy all threats before finishing

- Section 3 – Maze building and Configuration

Within this section is where the maze level and definition of objects comes to place. Each level is configured within their own text file and is composed of 25 characters horizontally and vertically. The code then randomly chooses a number between 1 and 10 (which is currently how many levels there are) and it is loaded. Also, each individual character within the text file are defined here.

```
##----------Section 3--------------------##
#Maze building and configuration

#list that will hold the level imported from the text file that will be randomly selected
levels = [""]

#choosing a random number between 1,10 to choose the Maze to import to the list
txtfilenumber=random.randint(1,10)
filename=str(txtfilenumber)+".txt"

with open(filename, "r") as f:
    level0 = [""]
    for line in f:
        level0.append(line) #add each line from the .txt file to the levels list
    print("OPENED FILE: "+filename)

levels.append(level0) #adding the .txt file to the levels list

#The routine that defines the actual drawing of the maze
def draw_maze(level):
    global treasurecount #reference to the treasurecount variable that is created below
    global threatcount #references to the threatcount variable that is created below
    for y in range(len(level)): #for y coords
        for x in range(len(level[y])):
            #Get the characgter at each x,y coord
            #Y goes first as the maze is stored in a LIST, so goes line by line, vertically
            character = level[y][x]
            #Go through the list and note down all coordinates for each character
            x_coords = -288 + (x * 24) #24 is the size for each block accross
            y_coords = 288 - (y * 24)

            finalcoords = (x_coords, y_coords) #Combines the current coordinate to one single variable

            ##Defining what character represents each object in the Maze##
            #Define what character a wall is and adding it to the wall list if found
            if character == "X":
                walls.goto(finalcoords)
                walls.stamp()
                wall_coords.append((finalcoords))

            #Defining what character a piece of gold is and adding it to the treasures list if found
            if character == "G":
                treasures_coords.append(Treasure(x_coords, y_coords)) #add the coordinates to the treasure list and also assign the coordinates to the treasure class
                treasurecount = treasurecount+1

            #Defining what character a threat is and adding it to the threats list if found
            if character == "T":
                threats_coords.append(Threat(x_coords, y_coords)) #add the coordinates to the threats list and also assign the coordinates to the threats class
                threatcount = threatcount+1

            #Defining what character the end point is and adding it to the endpoints list if found
            if character == "E":
                endpoint_coords.append(End(x_coords, y_coords)) #add the coordinates to the enpointcoord list and assign the coordinates to the End class

            #Defining empty spaces and adding it to the emptyspace list if found
            if character == " ":
                emptyspace_coords.append([x_coords, y_coords])
```

For every line within the text file, add this to the level0 list.

Defining the coordinates in the level. Each coordinate ranges from x -288 – 288 and y -288 - 288

This section defines what character within the textfile belongs to what class. The comments explain what each character is

- Section 4 – Defining lists and trackers

```
##------------------SECTION 4------------------##

#Making new variables to call on the classes created earlier
walls = Walls()
player = Player()

#create a list of coordinates for each object defined here: walls, endpoint, treasures, threats, empty spaces
wall_coords = [] #So the player can't walk into walls
endpoint_coords = [] #So the player can go into an endpoint
treasures_coords = [] #So the player can claim treasure
threats_coords = [] #So the player can destroy threats
emptyspace_coords =[] #So the player can randomly spawn in empty spaces

#defining all of the different counts during the game
treasurecount = 0 #Will count how many gold treasures can be collecxted
threatcount = 0 #Will count how many threats there are
treasurecaught = 0 #Will count how much gold a user has taken
threatscaught = 0 #Will count how many threats a user has destroyed


#Setup the level - to draw the maze from the list levels that has been imported from the text file
draw_maze(levels[1])

#Setting and placing the player on the maze once drawn up
player_coords= random.choice(emptyspace_coords) #Choose a random coordinate from the empty space list
player.goto(player_coords) #Then go to the coordinate
print("Player allocated to coords: " +str(player_coords)) #Print the result


#Keyboard controls
window.listen()
window.onkey(player.go_left,"Left") #left arrow
window.onkey(player.go_right,"Right") #right arrow
window.onkey(player.go_up,"Up") #Up key
window.onkey(player.go_down,"Down") #Down key
```

Defining a variable for the walls class and player class so that they can be called on

This section defines all of the different variables that will hold key information about the maze.

Set the player to a random coordinate within the maze that has been recorded as empty

Setting the keyboard controls for the player and calling the routine when the key is pressed

- Section 5 [Part 1] – Checking conditions to win/lose a game

Section 5 contains a continuous loop which checks against certain conditions before deciding to end or continue a game.

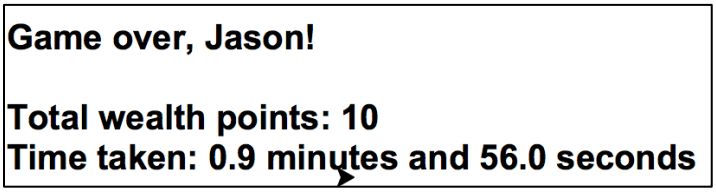
```
##---------SECTION 5-----------------##
#This section will continuously loop when the player moves to check each condition seen below, the game over routine is also here

def game_over():
turtle.clearscreen()
turtle.penup()
turtle.home()
window.bgcolor("white")
turtle.color("black")
print("Displayed game over message")
end_time_minutes=round((time.time()-start_time)/60,1)
end_time_seconds=round(time.time()-start_time,1)
turtle.write("Game over, "+playername+"!\n\nTotal wealth points: " +(str(player.gold))+"\nTime taken: "+str(end_time_minutes)+" minutes and "+str(end_time_seco
time.sleep(10)
turtle.clear()
turtle.home()
print("Loading next game..")
turtle.write("Loading another game...", align="center", font=("Arial",15,"normal"))
time.sleep(5)

#Loop that updates everytime the player moves

while True:

    #If the player gets into a negative balance below 0, then it will say game over and the game will actually loop again
    if player.gold<0:
        time.sleep(5)
        game_over()
        break

    if player.touchedthreat>0:
        turtle.clearscreen()
        turtle.penup()
        turtle.home()
        window.bgcolor("white")
        turtle.write("RIP. You have gone over a threat!", align="center", font=("Arial",20,"bold"))
        time.sleep(5)
        print("Game over with reason message displayed")
        game_over()
        break


    if player.win>1:
        break
```

The game over routine clears the screen and displays a game over message to the user. The output can be seen below

If the player gold goes below 0 to negative or if a player touches a threat the game will end and go to the game over routine. Upon touching a threat, an additional message telling the user that they have touched a threat will be displayed. If the player has won a game, then the code will break out of the loop and start a new game

Outputs from Section 5 (Part 1)

- Game over message



- Going over a threat

- Section 5 [Part 2] – Checking conditions to win/lose a game

```
    #For every treasure in the treasures list - if the player 'touches' a specific treasure item and it matches the
    #coordinates in the treasures_coords list, then it will get removed and claimed by the user
    #They will also get a wealth gold balance of 10

    for treasure in treasures_coords:
        if player.touched_object(treasure):
            player.gold = player.gold + treasure.gold
            print ("Gold picked up. Player wealth points now: " +(str(player.gold)))
            treasures_coords.remove(treasure) #remove that treasure from the treasures list
            treasure.destroy()
            treasurecaught = treasurecaught + 1

    #For every threat in the threats list - if the player goes near or goes onto a specific threat and it matches the
    #coordinates in the threats_coords list, either lose the game straight away or
    #they move away from the threat. If they press 'spacebar' next to a threat then they can destroy the threat anbd gain 5 wealth.

    for threat in threats_coords:#for every threat in the treasures list
        if player.near(threat):#If player is near the threat
            window.onkey(threat.destroy_key, "space")

        if player.touched_object(threat):
            player.touchedthreat = player.touchedthreat + 1



    #Once the player reaches the end point, there are two possibilities
    #Outcome 1: The user hasn't collected all treasures or destroyed threats. They will be told what is left to collect / destroy abnd be brought back
    #to where they first started
    #Outcome 2: The user has collected everything, and the end routine is performed where their final wealth is displayed and they are congratulated
    for end in endpoint_coords:#for the one enpoint in the endpoint list
        if player.touched_object(end): #if player has touched the end point
            if treasurecaught == treasurecount and threatscaught == threatcount:
                player.win = player.win + 1
                end.ended()
                break #supposed to break out of the loop but it doesn't


            else:
                end.not_ended()
                continue



    #update the window with any changes
    window.update()

#end the program
window.update()
window.clearscreen()
print("playing again!!!")
```

This section checks conditions for both treasures and threats. If a treasure is picked up it runs the treasure.destroy routine which removes it away from the screen.

For a threat, if the player is near and presses spacebar, then the threat.destroy routine is run which removes the threat from the screen. If the player touches the screen, then it updates the .touchedthreat tracker to 1 and the game will end.

For a player going to an endpoint there are 2 outcomes as seen in the comments. If the player meets the first condition where everything has been collected, the player.win tracker variable will update to 1 and will end the congratulations message. If not then the attempted to finish early message will be displayed.

At the end, the window is updated to reflect changes and when the loop is broken a message saying playing again is displayed in the background python window.

Outputs from section 5 (part 1)

- Collecting treasure

```
Gold picked up. Player wealth points now: 10
```

- Attempting to finish early

```
You cannot finish yet!
You have lost 10 wealth points

Please ensure that you collect all the treasure and destroy all threats before finishing
```

- Finishing the game completely

```
Congratulations Jason! You have reached the end!

Total player wealth points: 70
Time taken to complete: 0.9 minutes and 55.8 seconds
```

- Playing another game

```
Loading next game..
playing again!!!
```

Classes used in the python game

Walls = Defined the characteristics of a wall in the game
Player = Defined the characteristics of a player in the game
Treasure = Defined the characteristics of a treasure in the game
Threat = Defined the characteristics of a threat in the game
End = Defined the end point in the game and what to do when the game was ended early

Routines used in the python game

Player.go_up = Move the player up
Player.go_down = Move the player down
Player.go_left = Move the player left
Player.go_right = Move the player right
Player.touched_object = Outputting TRUE or FALSE if a player has touched an object
Player.near = Outputting TRUE or FALSE if a player is within distance of another object
Player.gold = Contains how much wealth points a player has
Player.touchedthreat = If this turns to 1, the player game is ended instantly
Player.win = If this turns to 1m the player game is ended and congratulations message shown
Threat.destroy_key = When the player presses SPACEBAR near a threat, the threat will be removed and 10 wealth points added to the player
End.ended = Defining the routine for the end message to the player and breaking the loop that checks if conditions have been met to end a game
End.not_ended = Displays the attempted to finish early message to player and reduces player wealth points by 10
Draw_maze = Defining the routine that draws the maze and what characters defined what objects were where

Variables used in the python game

Window = Defines the turtle window in the game
Levels = The list that contains the level that is loaded into the game
Txtfilenumber = Will hold a random number between 1 – 10 for the text file to be opened
Filename = combines the txtfilenumber to .txt to make it into a valid file
Level0 = Defines a list that will be added to the levels list when loading the game
Wall_coords = The list that contains all coordinates of walls in the game
Endpoint_coords = The list that contains the end point within the game
Treasures_coords = The list that contains the coordinates of all treasure in the game
Emptyspace_coords = The list that contains all coordinates of empty space in the game
Treasurecount = Contains how many gold treasures there are
Threatcount = Contains how many threats there are
Treasurecaught = Contains how many items of gold a player has caught
Threatscaught = Contains how many threats a player has destroyed
Player_coords = Contains the randomly selected coordinates from the emptyspace_coords list

Best practices for code development

Throughout the development of my code, I have followed and aimed to maintain the following practices for the project:

- **Commenting**

Within my code, I have added comments in a consistent manner and I have always defined what each line does and if there are any libraries involved in making the code work

- **Indentation of code**

In the code that I have developed, the indentation of the code is always kept consistent and followed to prevent any errors before running the code

- **Understandable variable names**

I always ensure that any variables that I create can be understood by someone who has not read my code before

- **Version control**

Whenever I make a significant change to parts of my code, I ensure to save them under a new version so that the code can either be compared to the previous version or rolled back in case anything goes wrong

- **Simple code**

I always aim to keep the code to be as simple and minimal as possible, avoiding unnecessary lines if it can be done

- **Test driven development**

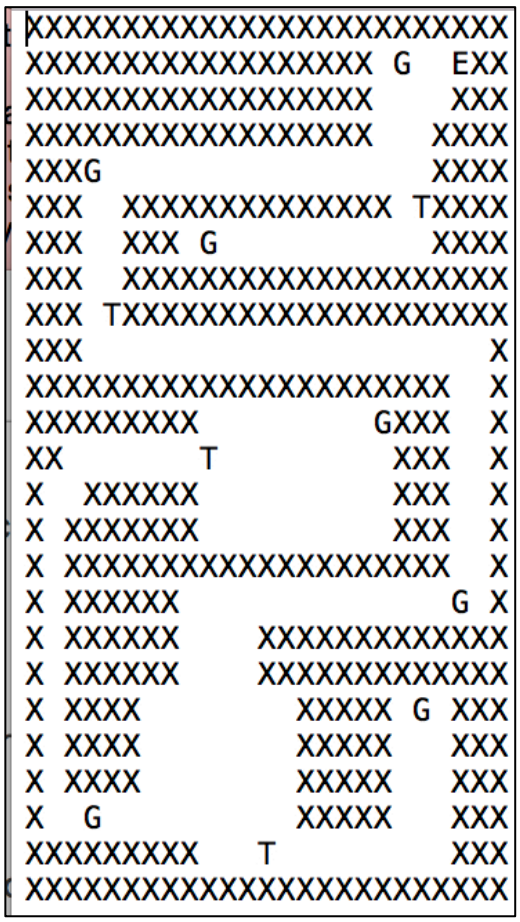
After designing each version, I test what functionality is there and what is not there, and as a result it leads to the next version of the code if something is missing. It also allows me to see how a version of the code would really run for the end user

- **File and folder organisation**

I always keep my files and folders organised to ensure that I can achieve version control and keep track of where everything is

Text files

The text files included within the same directory of the code are all crucial to what maze the user plays. Each text file contains the configuration of each maze and how they would look like on a graphical user interface. You will notice that all the text files look the same to how they actually are loaded when the user plays the game. This is because each text configuration file contains characters seen like the one below.



X represents a square, T represents a threat, G represents an item of gold / treasure which the player can collect, and E represents the end point in which the user goes to in order to complete the game. Each text file is numbered – this is key because the code randomly selects a number between 1 to 10 which then determines what text file to open.

Testing

Test plan (Before carrying out testing)

Before testing the code that I made, I created the following test plan to test my game program.

- **Functional Tests**

Test Number	What to do	Given Input	Expected Output	Actual Output
1	Run the program and wait for the welcome message	N/A	"Welcome to the Maze Game"	
2	Run the program and wait for it to ask the user to input their name	N/A	"What is your name"	
3	Run the program, input a name and wait for a second greeting message and instructions to appear	Jason	"Nice to meet you Jason" ….	
4	Run the program and let it randomly choose a number between 1-10 to open a text file	B/A	"OPENED file x"	
5	Run the program and make sure a game starts only when the maze has been configured properly	N/A	Completed maze	
6	When starting to play the Maze game, the player should be randomly allocated to an empty space	N/A	Player allocated to an empty space	
7	When the game starts, the player should begin with 0 wealth	N/A	Instructions at the beginning	
8	When the Maze is shown, all rooms should be able to be seen and passages identified	N/A	Completed maze	
9	Run program and collect the treasure	User should touch the treasure [yellow circle]	The treasure should disappear when the player goes over it and 10 wealth points get added	

10	Run program and destroy a threat	User should go near the threat [red circle] and press SPACEBAR on the keyboard to destroy it.	The threat should disappear when the player destroys it and 5 wealth points get added	
11	All threats and treasures should be removed once destroyed/collected before finishing	Going to the end point [green square]	Congratulations message should be displayed or if not finished then attempted finish message displayed	
12	When collecting treasure, user should gain 10 wealth points	Going over a treasure	User gets 10 wealth added to their current wealth points	
13	When destroying a threat, user should gain 5 wealth points	Pressing space bar when near a threat	User gets 5 wealth points added to their current wealth points	
14	When user touches a threat, game over	User going over a threat which is a red circle	Game over message should be displayed to user	
15	When finishing the game too early, user should get deducted 10 wealth points from total and launched back in their starting position	User going to endpoint while there are still threats / treasures that have not been collected	Game not finished yet message should be displayed to the user	
16	When the game finishes, the total wealth points and time taken to complete the Maze is calculated and outputted on GUI	User going over the end point once all treasures and threats have been collected / destroyed	Congratulations message along with their score and time taken to complete the game	

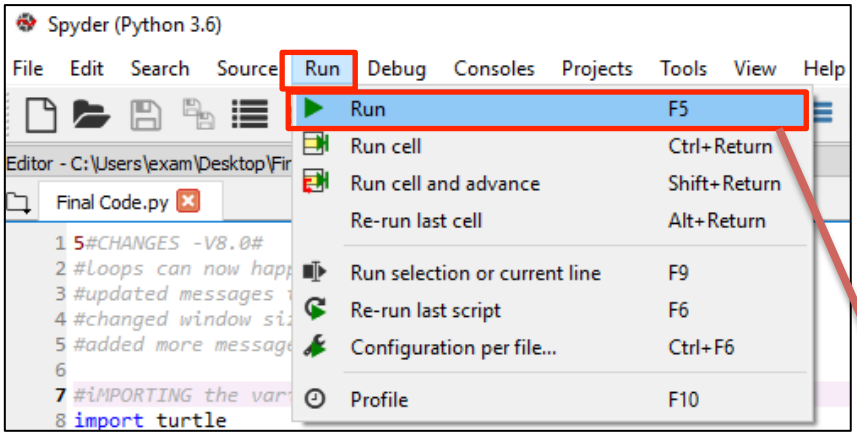
- **Non-Functional Tests**

Test Number	What to do	Given Input	Expected Output	Actual Output
1	The game must be easy to use and play	Running the program	Welcome message with clear instructions and easy controls to move the player	
2	The program should look into the chosen text file and copy template of maze within 10 seconds	Randomly selected file number	Beginning of maze printed	
3	White squares within the maze should show where the walls are and therefore the player shouldn't be allowed in that area	Player attempting to go into a wall (white square)	Player can't go into wall (white square)	
4	Yellow circles should be shown on the maze, when treasure/gold are located	User starting the program and the level containing the configuration of the treasure locations	Yellow circles on the level the is loaded and placed in their configured locations	
5	Red circles should be shown on the maze, when threats are present	User starting the program and the level containing the configuration of the threat locations	Red circles on the level the is loaded and placed in their configured locations	
6	A green circle should be shown on the maze, which indicates the finish point that the user has to go to.	User starting the program and the level containing the configuration of the end point location	Green circles on the level the is loaded and placed in their configured locations	
7	A blue square represents the user playing the game. This should be able to move around the maze accordingly.	User starting the program and the program randomly allocating the player to an empty space	User being placed in a randomly assigned location that is empty	

Test plan (After carrying out testing)

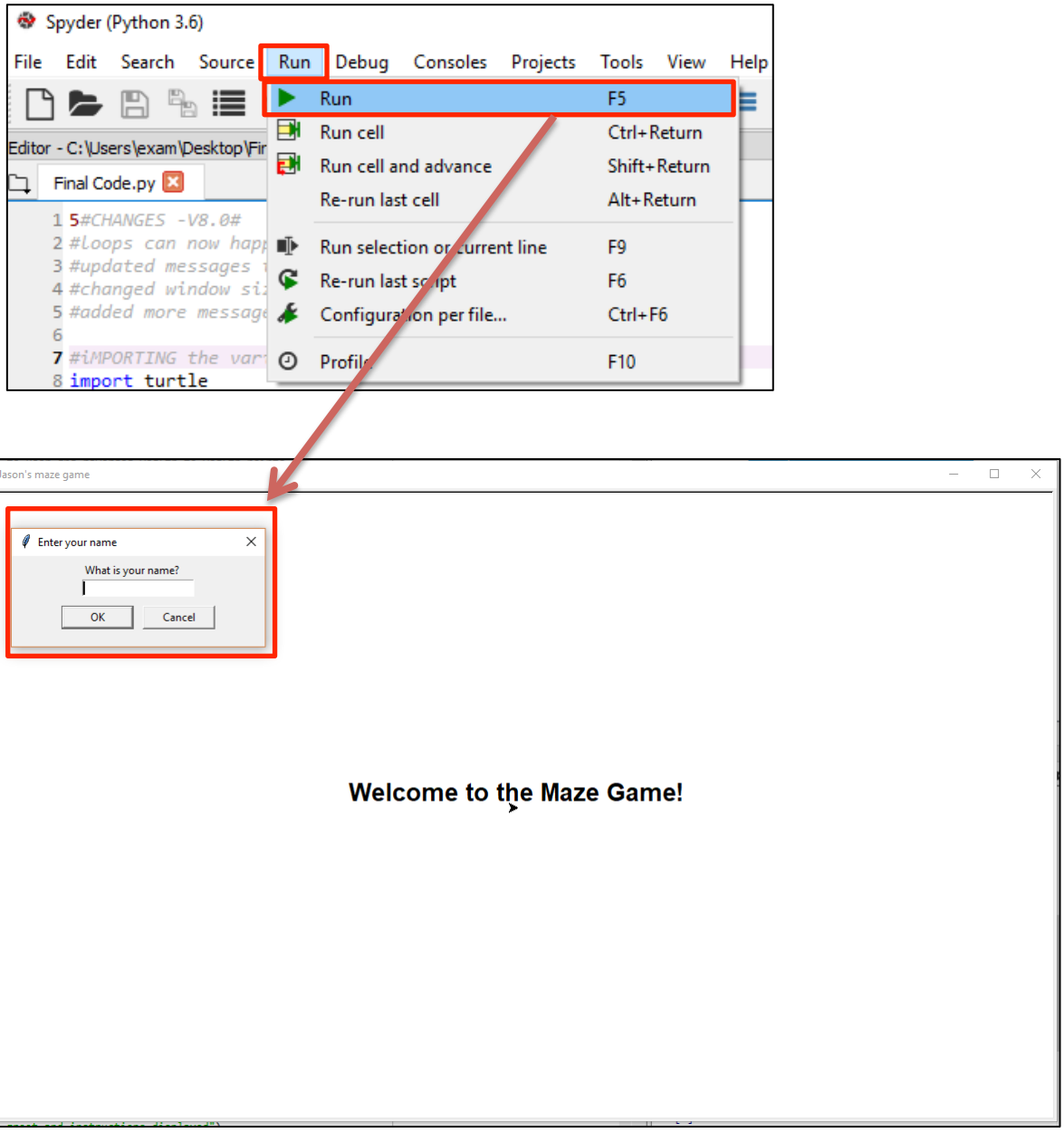
- **Functional Tests**

Test Number	What to do	Given Input	Expected Output	Actual Output
1	Run the program and wait for the welcome message	N/A	“Welcome to the Maze Game”	See below **Test passed**



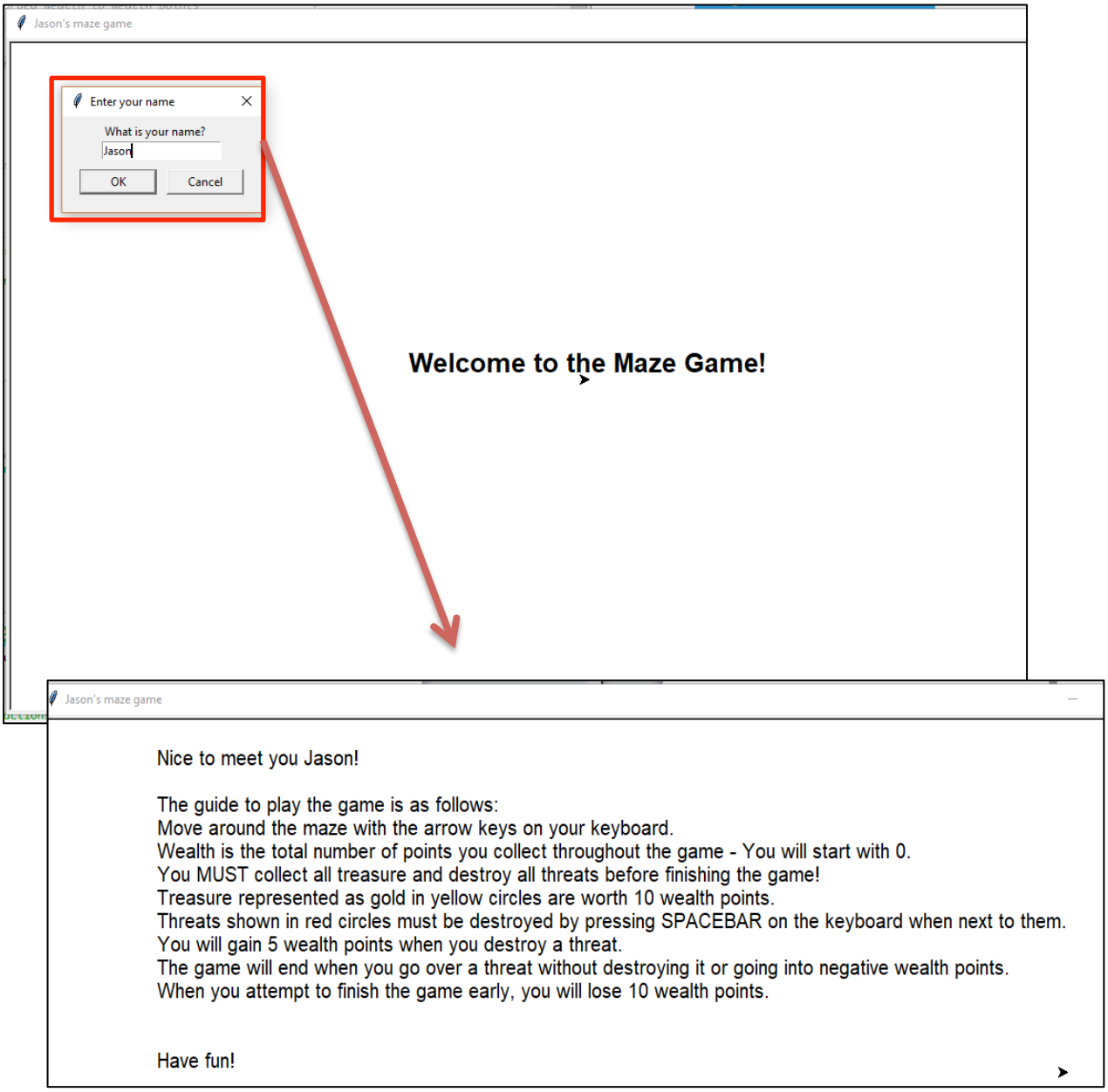
Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
2	Run the program and wait for it to ask the user to input their name	N/A	“What is your name”	See below **Test passed**



Test Passed

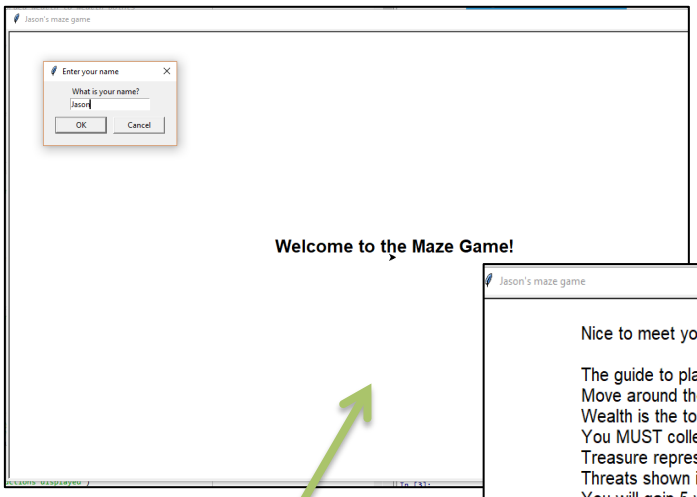
Test Number	What to do	Given Input	Expected Output	Actual Output
3	Run the program, input a name and wait for a second greeting message and instructions to appear	Jason	“Nice to meet you Jason” ….	See below **Test passed**

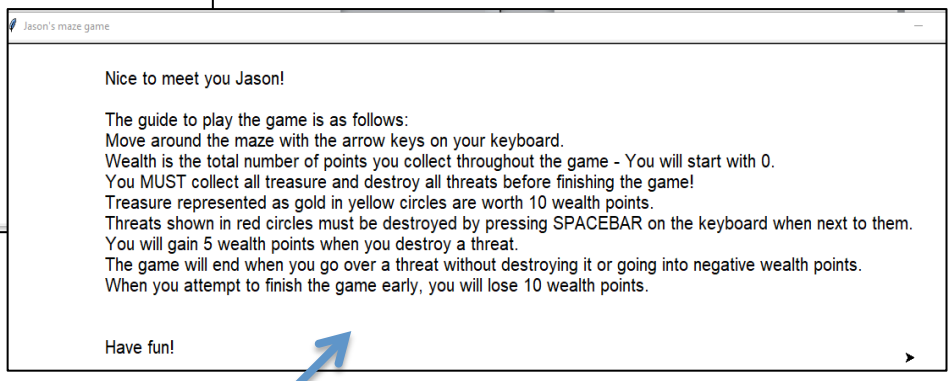


As seen in the screenshots, once the user has inputted their name, the greet message/instructions on how to play the game appear for 15 seconds.

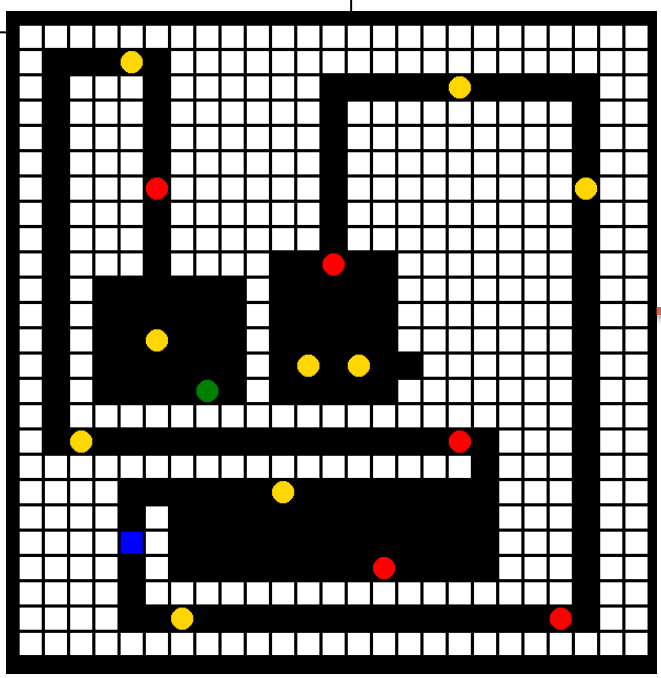
Test Passed

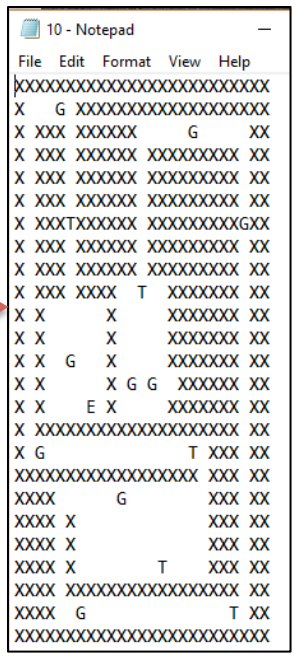
Test Number	What to do	Given Input	Expected Output	Actual Output
4	Run the program and let it randomly choose a number between 1-10 to open a text file	N/A	“OPENED file x”	See below **Test passed**






```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 10.txt
```

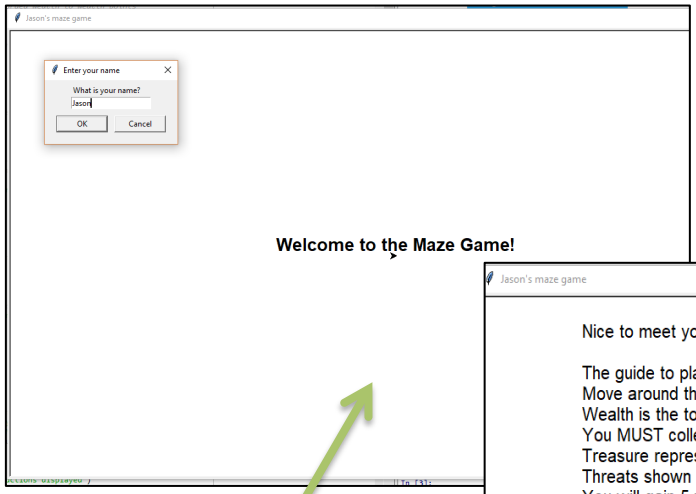


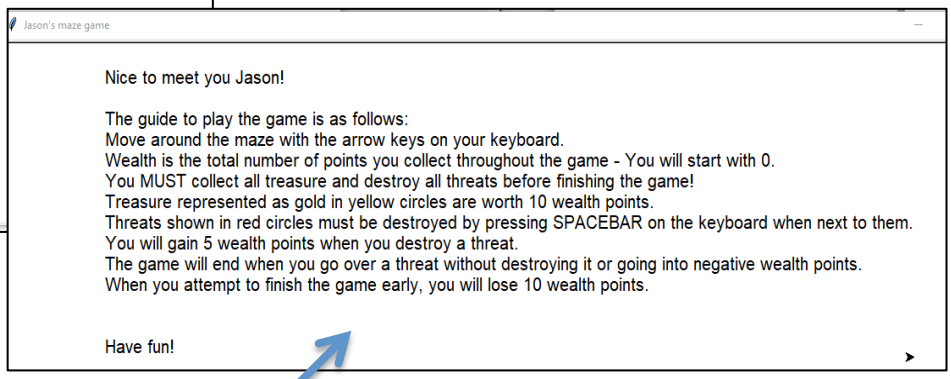


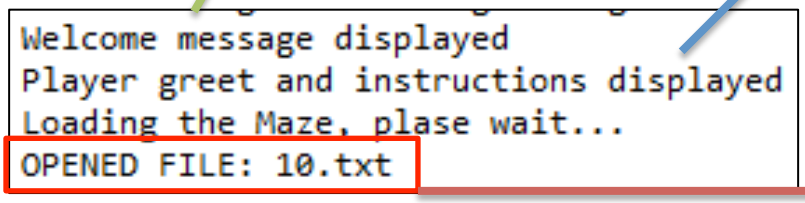
The program in this scenario randomly selected Maze 10, and therefore outputted the same design as in the text file of 10 on to the GUI.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
5	Run the program and make sure a game starts only when the maze has been configured properly	N/A	Completed maze	See below **Test passed**

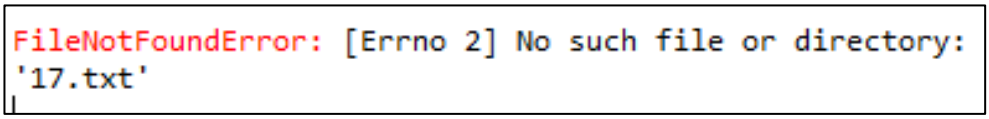






This shows that the program has randomly selected the file 10 and successfully opened the file in order to get the configuration details

The above scenario occurs when both the text files and the python program is located within the same folder on the computer. This is essential in order for the Maze game to work – below shows the error message if the text file cannot be found:

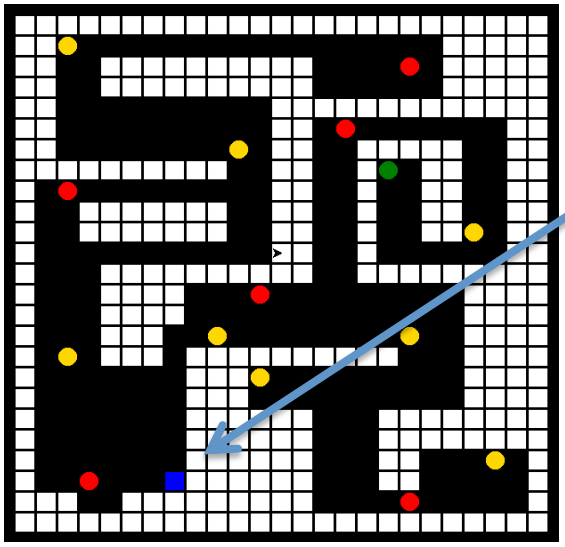


Test Passed

6	When starting to play the Maze game, the player should be randomly allocated to an empty space	N/A	Player allocated to an empty space	See below **Test passed**

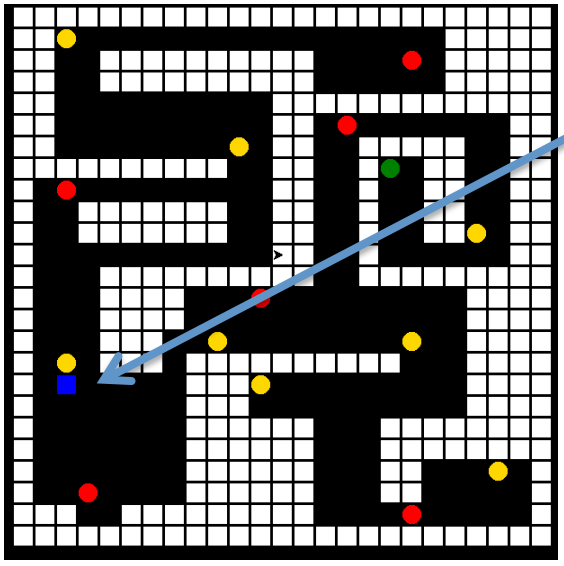
The code I have developed allocates the player randomly at the start of the game as shown in the screenshots below:

```
OPENED FILE: 6.txt
Player allocated to coords: [-120, -264]
```



In this scenario, the player is playing maze 6 and starting from the coordinates [-120, -264] as these were chosen randomly by the program Their start position is represented as a blue square.

```
OPENED FILE: 6.txt
Player allocated to coords: [-240, -144]
```

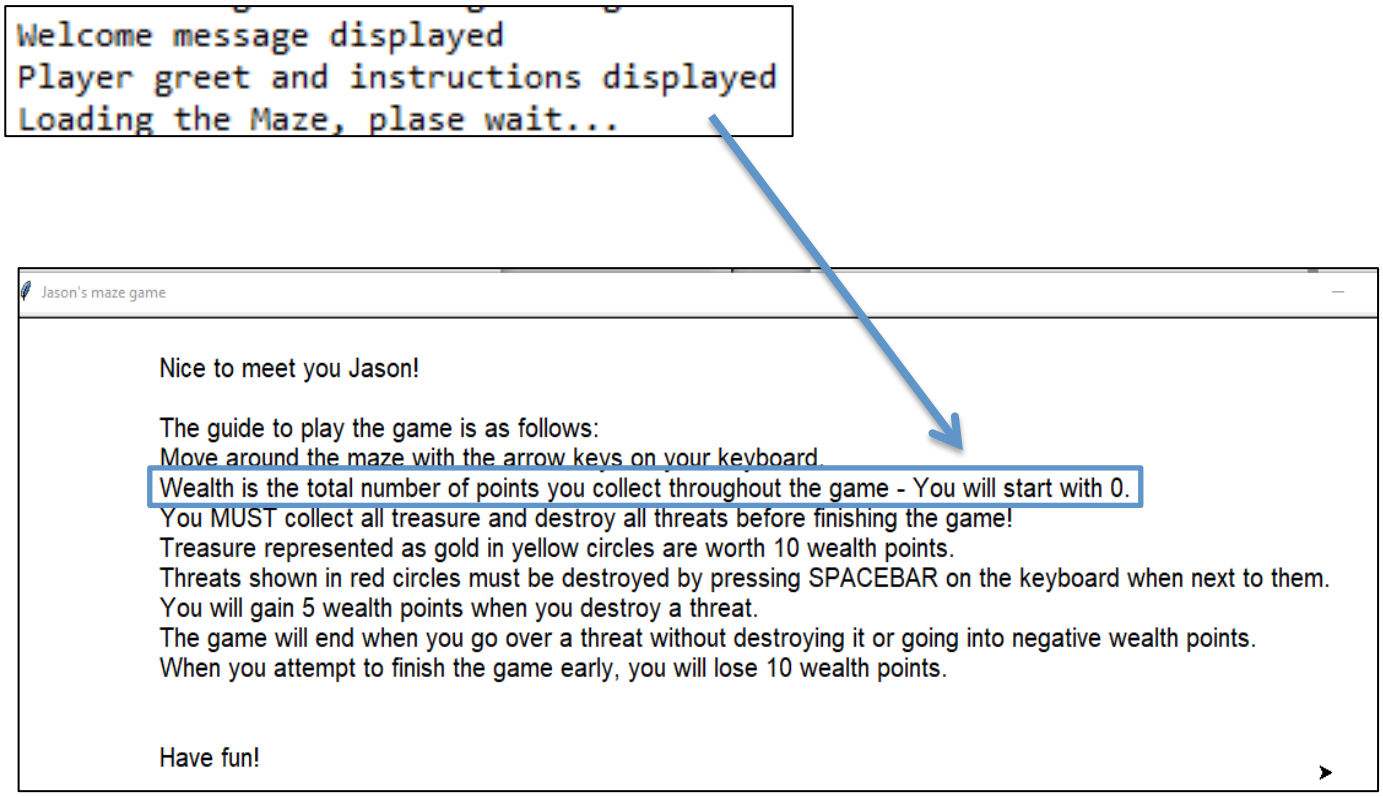


In this scenario, the player is also playing maze 6 and starting from the coordinates [-240, -144] as these were chosen randomly by the program. Their start position is represented as a blue square.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
7	When the game starts, the player should begin with 0 wealth	N/A	Instructions at the beginning	See below **Test passed**

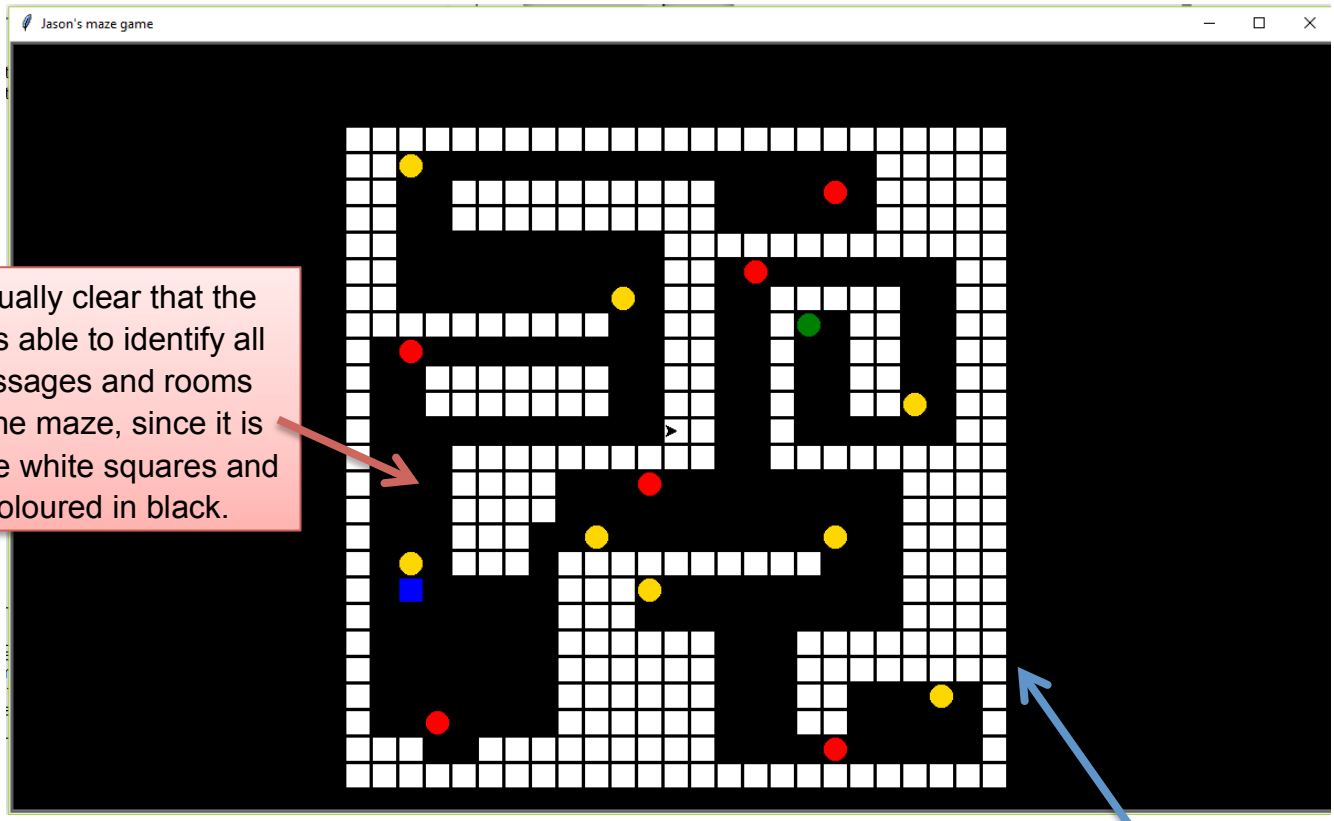
Screenshot below shows running the program



As seen in the screenshot above, any player who plays the game will start on 0 wealth. This is written on the instructions which are shown before playing the maze game.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
8	When the Maze is shown, all rooms should be able to be seen and passages identified	N/A	Completed maze	See below **Test passed**



It is visually clear that the player is able to identify all the passages and rooms within the maze, since it is within the white squares and are coloured in black.

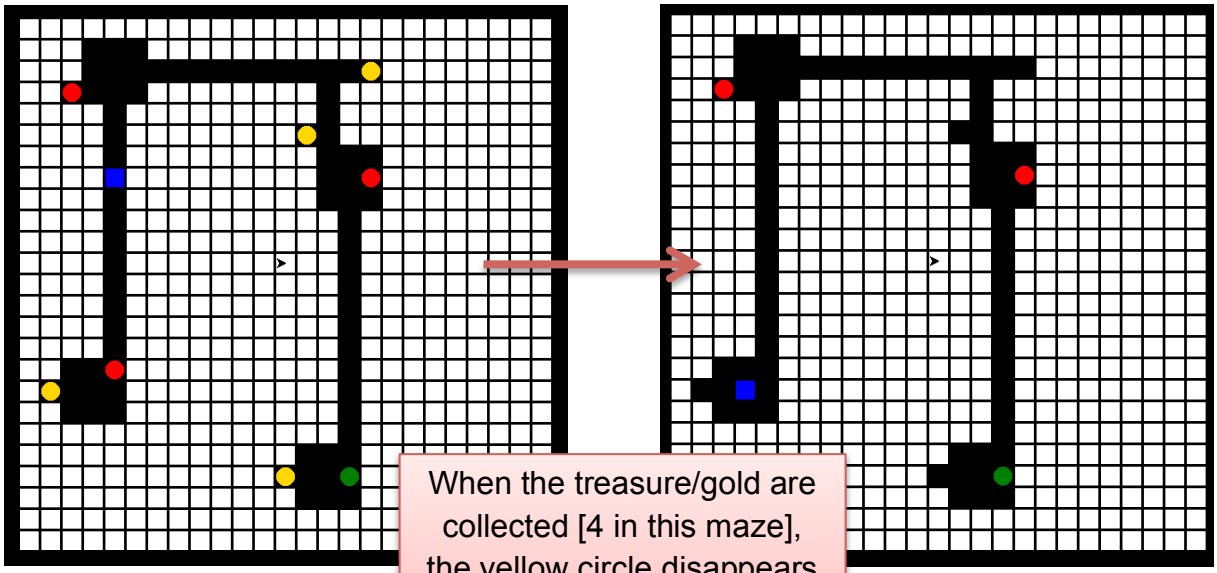
The white squares represent the walls of the maze. This is the area in which the player is not able to go into.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
9	Run program and collect the treasure	User should touch the treasure [yellow circle]	The treasure should disappear when the player goes over it and 10 wealth points get added	See below **Test passed**

Maze at start, when treasure can be seen

Maze when all treasures been collected



When the treasure/gold are collected [4 in this maze], the yellow circle disappears as seen in the screenshots above.

```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 1.txt
Player allocated to coords: [-192, 96]
Gold picked up. Player wealth points now: 10
Gold picked up. Player wealth points now: 20
Gold picked up. Player wealth points now: 30
BOOM! Threat removed! Player wealth points are now: 35
Gold picked up. Player wealth points now: 45
```

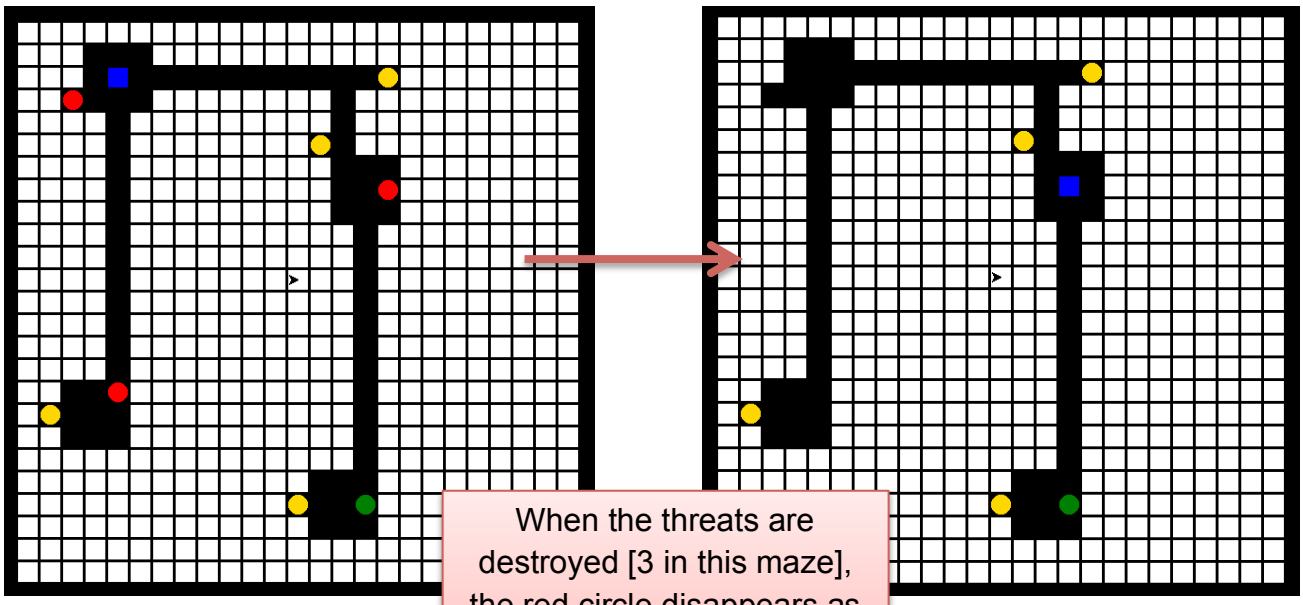
This shows the log as to when the player collects gold. As seen in the screenshot, every time a player collects gold – 10 points are added to their wealth points.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
10	Run program and destroy the threats	User should go near the threat [red circle] and press SPACEBAR on the keyboard to destroy it.	The threat should disappear when the player destroys it and 5 wealth points get added every time	See below **Test passed**

Maze at start, when threats can be seen

Maze when all threats been destroyed



When the threats are destroyed [3 in this maze], the red circle disappears as seen in the screenshots above.

```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 1.txt
Player allocated to coords: [-192, 216]
BOOM! Threat removed! Player wealth points are now: 5
BOOM! Threat removed! Player wealth points are now: 10
BOOM! Threat removed! Player wealth points are now: 15
```

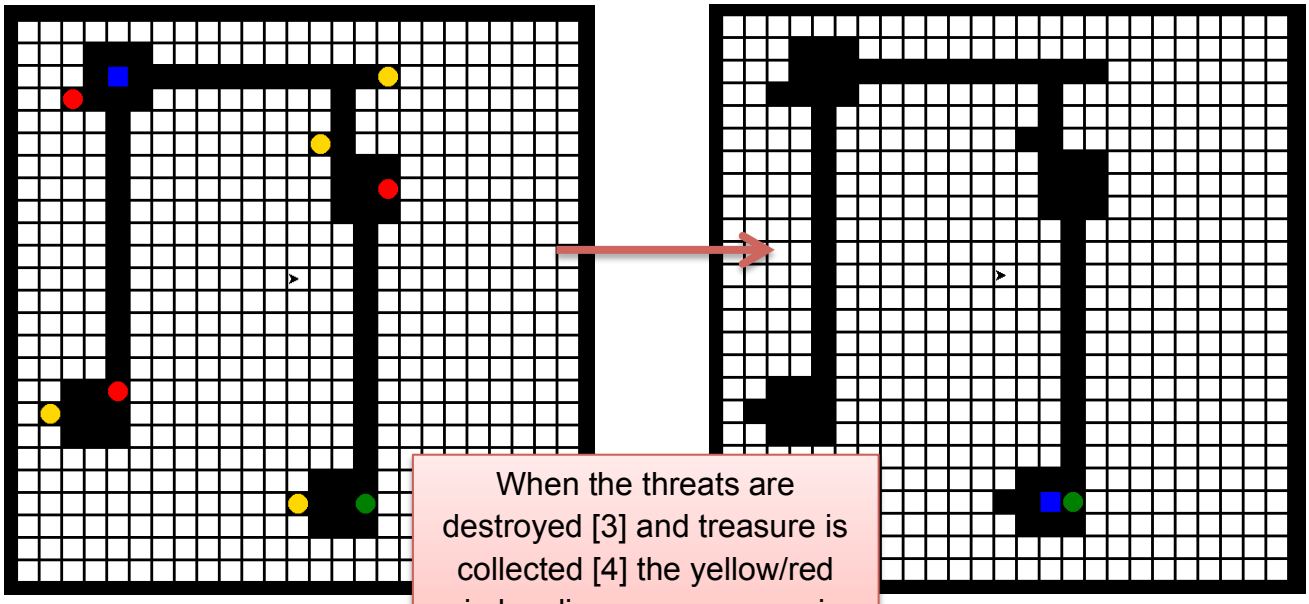
This shows the log as to when the player destroys threats. As seen in the screenshot, every time a player destroys a threat – 5 points are added to their wealth points.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
11	All threats and treasures should be removed once destroyed/collected before finishing	Going to the end point [green square]	Congratulations message should be displayed or if not finished then attempted finish message displayed	See below **Test passed**

Maze at start with all treasure/threats

Maze when all treasures/threats retrieved



When the threats are destroyed [3] and treasure is collected [4] the yellow/red circles disappear as seen in the screenshots above.

```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 1.txt
Player allocated to coords: [-192, 216]
BOOM! Threat removed! Player wealth points are now: 5
BOOM! Threat removed! Player wealth points are now: 10
BOOM! Threat removed! Player wealth points are now: 15
Gold picked up. Player wealth points now: 25
Gold picked up. Player wealth points now: 35
Gold picked up. Player wealth points now: 45
Gold picked up. Player wealth points now: 55
Displayed congratulations message
```

This shows the log as to when the player destroys threats and collects the treasure. After the green circle has been activated, the congratulations message appears as shown below:

Congratulations Jason! You have reached the end!

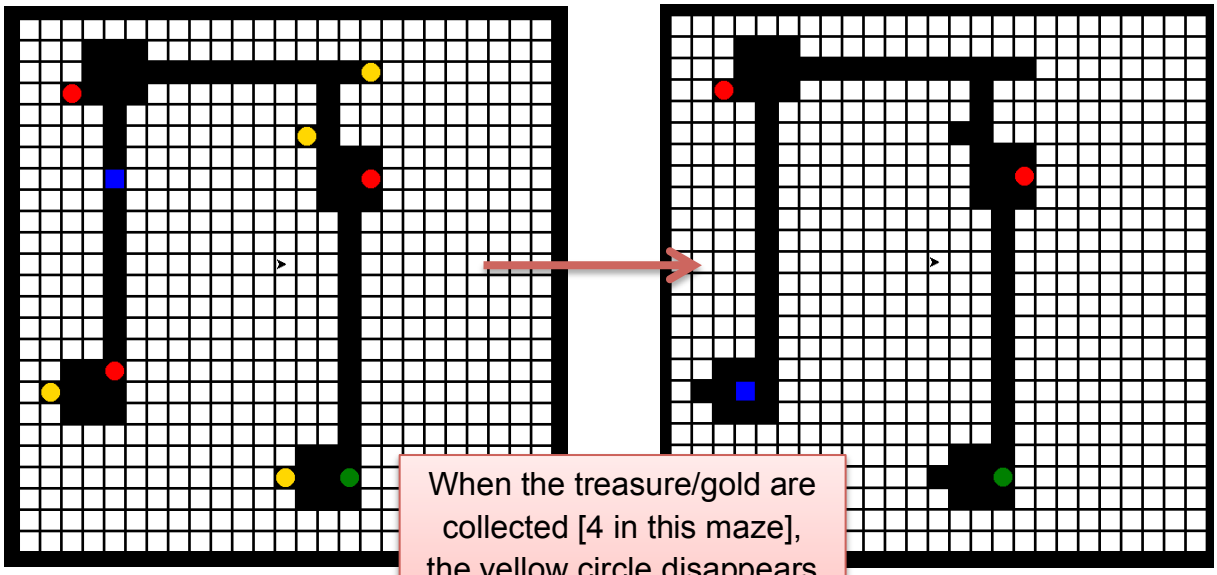
Total player wealth points: 55
Time taken to complete: 1.0 minutes and 61.6 seconds

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
12	When collecting treasure, user should gain 10 wealth points	Going over a treasure	User gets 10 wealth added to their current wealth points	See below **Test passed**

Maze at start, when treasure can be seen

Maze when all treasure been collected



When the treasure/gold are collected [4 in this maze], the yellow circle disappears as seen in the screenshots above.

```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 1.txt
Player allocated to coords: [-192, 96]
Gold picked up. Player wealth points now: 10
Gold picked up. Player wealth points now: 20
Gold picked up. Player wealth points now: 30
BOOM! Threat removed! Player wealth points are now: 35
Gold picked up. Player wealth points now: 45
```

This shows the log as to when the player collects gold. Every gold item is worth 10 points and this log gets updated every time there is a change in their total wealth points.

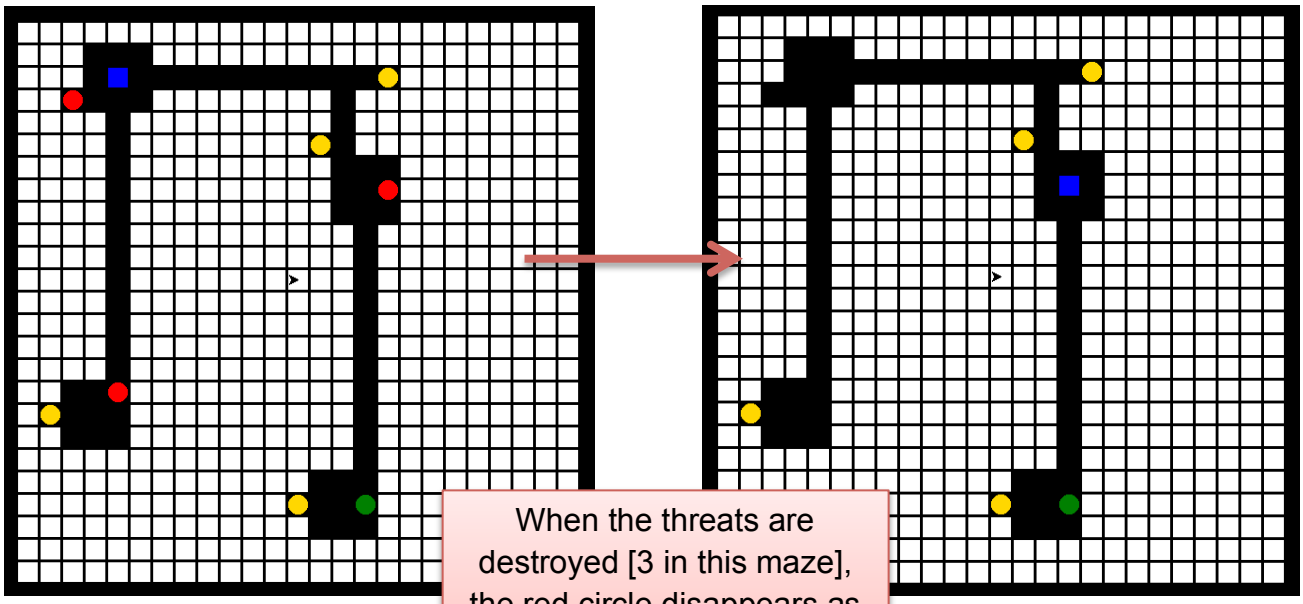
In the example above, there are 4 treasures; however 1 threat needs to be destroyed in order to get the treasure. As the player starts with a default of 0 wealth points, 10 are added every time for treasure meaning 4 x 10 = 40. However as they do need to destroy a threat 5 also gets added, totalling their current wealth points to 45

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
13	When destroying a threat, user should gain 5 wealth points	Pressing space bar when near a threat	User gets 5 wealth points added to their current wealth points	See below **Test passed**

Maze at start, when threats can be seen

Maze when all threats been destroyed



When the threats are destroyed [3 in this maze], the red circle disappears as seen in the screenshots above.

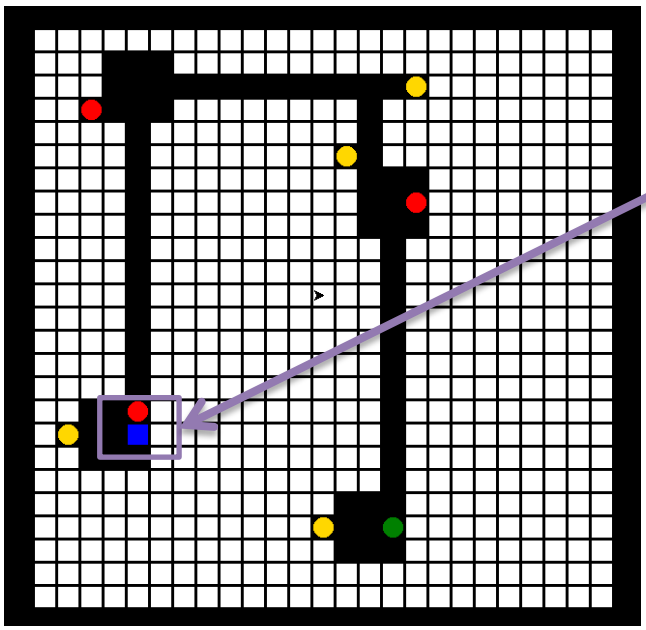
```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 1.txt
Player allocated to coords: [-192, 216]
BOOM! Threat removed! Player wealth points are now: 5
BOOM! Threat removed! Player wealth points are now: 10
BOOM! Threat removed! Player wealth points are now: 15
```

This shows the log as to when the player destroys threats. As seen in the screenshot, every time a player destroys a threat – 5 points are added to their wealth points.

There are 3 threats in the example above and since the players starts with a default of 0 wealth points, 15 wealth points is currently their total.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
14	When user touches a threat, game over	User going over a threat which is a red circle	Game over message should be displayed to user	See below **Test passed**



When the player goes straight on top of/touches the threat, the game automatically ends and outputs the following messages!

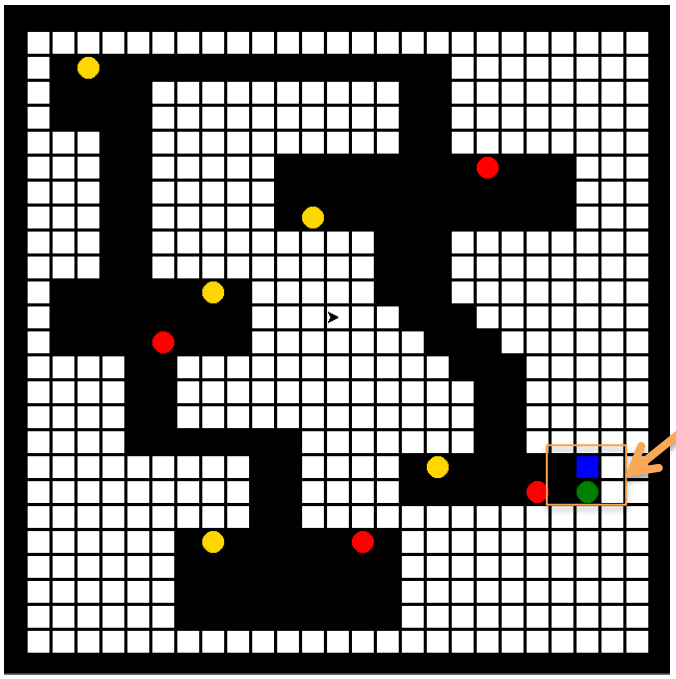
RIP. You have gone over a threat!

Game over, jason!

Total wealth points: 0
Time taken: 0.6 minutes and 36.9 seconds

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
15	When finishing the game too early, user should get deducted 10 wealth points from total and launched back in their starting position	User going to endpoint while there are still threats / treasures that have not been collected	Game not finished yet message should be displayed to the user	See below **Test passed**

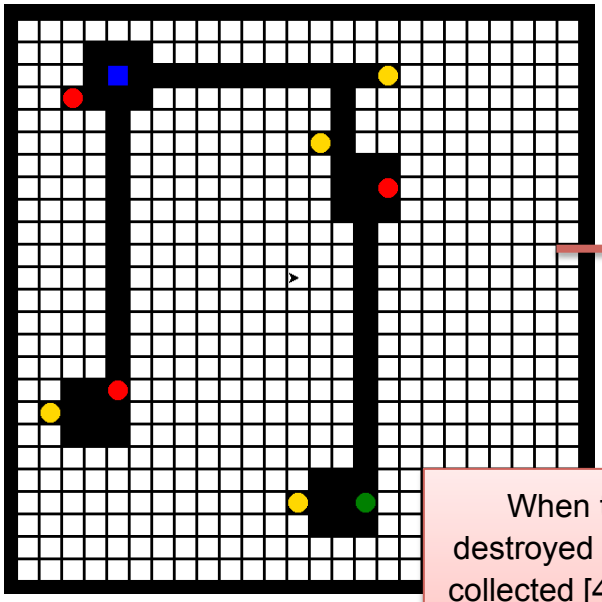


When the player goes straight to the finish point [green circle], the game automatically ends and outputs the following messages!

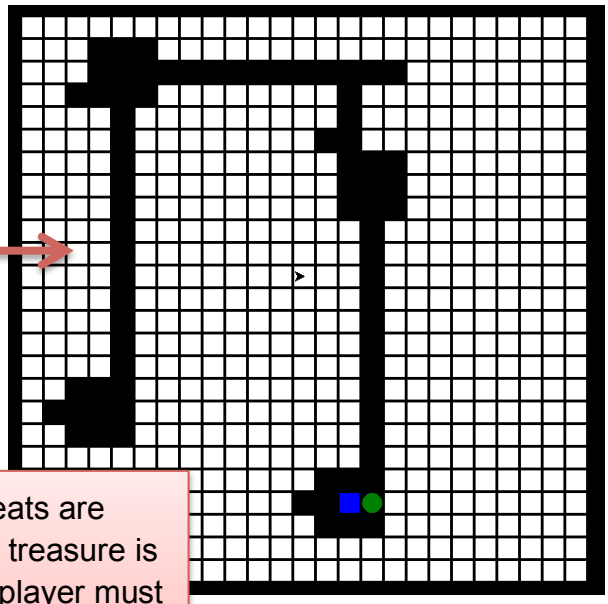
Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
16	When the game finishes, the total wealth points and time taken to complete the Maze is calculated and outputted on GUI	User going over the end point once all treasures and threats have been collected / destroyed	Congratulations message along with their score and time taken to complete the game	See below **Test passed**

Maze at start



Maze when all items are retrieved



When the threats are destroyed [3] and treasure is collected [4] , the player must go to the green circle to finish

```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 1.txt
Player allocated to coords: [-192, 216]
BOOM! Threat removed! Player wealth points are now: 5
BOOM! Threat removed! Player wealth points are now: 10
BOOM! Threat removed! Player wealth points are now: 15
Gold picked up. Player wealth points now: 25
Gold picked up. Player wealth points now: 35
Gold picked up. Player wealth points now: 45
Gold picked up. Player wealth points now: 55
Displayed congratulations message
```

This shows the log as to when the player destroys threats and collects the treasure. After the green circle has been activated, the congratulations message appears, outputting both their total wealth score and time taken to complete the maze, as shown below:

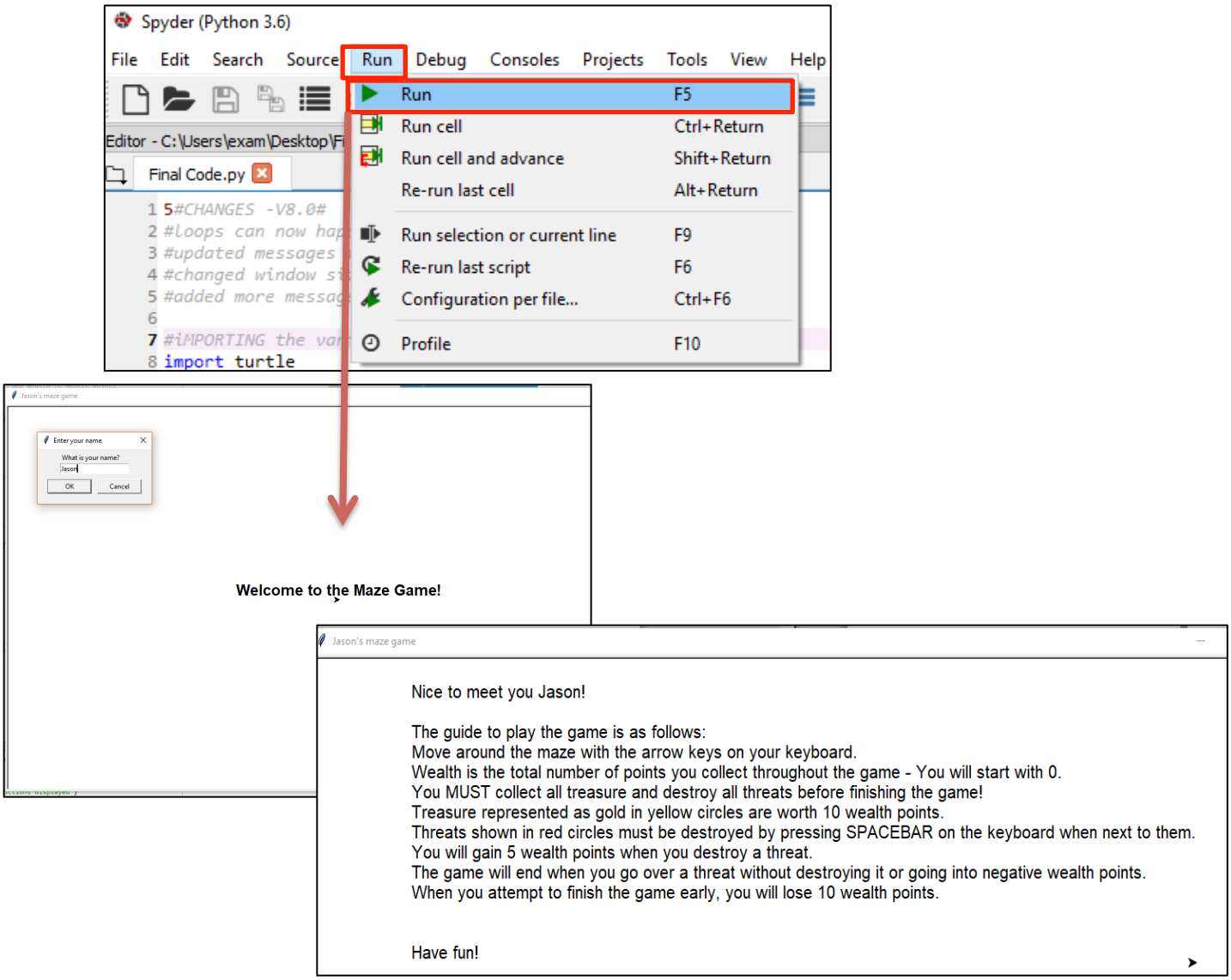
Congratulations Jason! You have reached the end!

Total player wealth points: 55
Time taken to complete: 1.0 minutes and 61.6 seconds

Test Passed

- **Non-Functional Tests**

Test Number	What to do	Given Input	Expected Output	Actual Output
1	The game must be easy to use and play	Running the program	Welcome message with clear instructions and easy controls to move the player	See below **Test passed**



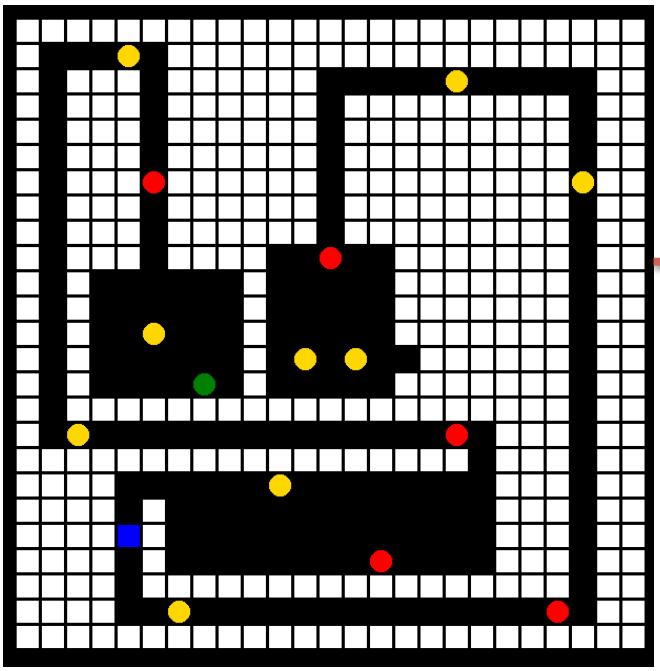
As seen in the screenshots, once the user has inputted their name, the greet message/instructions on how to play the game appear for 15 seconds, making sure that the user knows all the required information to play. This makes the game much easier to use as they do not have to download a separate document like a user guide if they want to find quick / brief instructions.

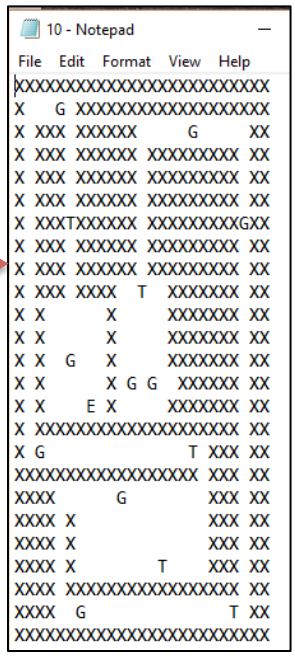
Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
2	The program should look into the chosen text file and copy template of maze within 10 seconds	Randomly selected file number	Beginning of maze printed	See below **Test passed**

When the program is run, it randomly selects a maze and outputs the same design as in the text file onto the GUI. This can be seen below:

```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 10.txt
```





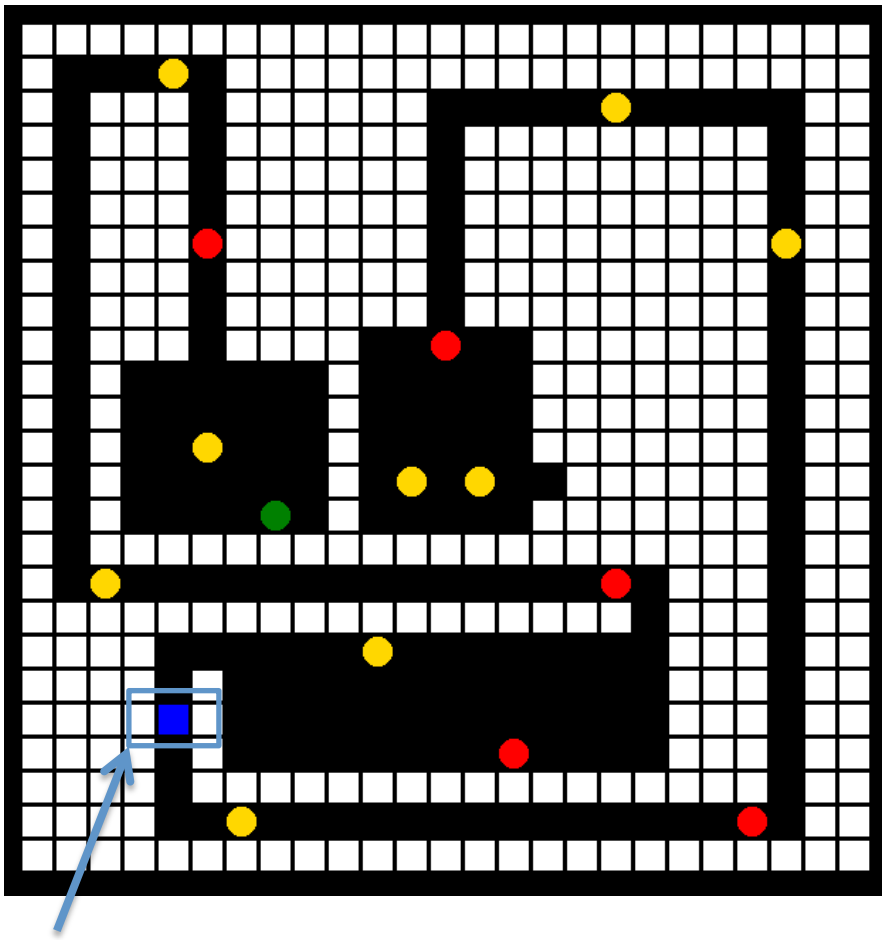
The program in this scenario randomly selected Maze 10, and therefore outputted the same design as in the text file of 10.txt on to the GUI.

I have timed this and due to the GUI creating a 25x25 grid, the loading process usually takes between 8-10 seconds once the instructions have disappeared.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
3	White squares within the maze should show where the walls are and therefore the player shouldn't be allowed in that area	Player attempting to go into a wall (white square)	Player can't go into wall (white square)	See below **Test passed**

Maze 10

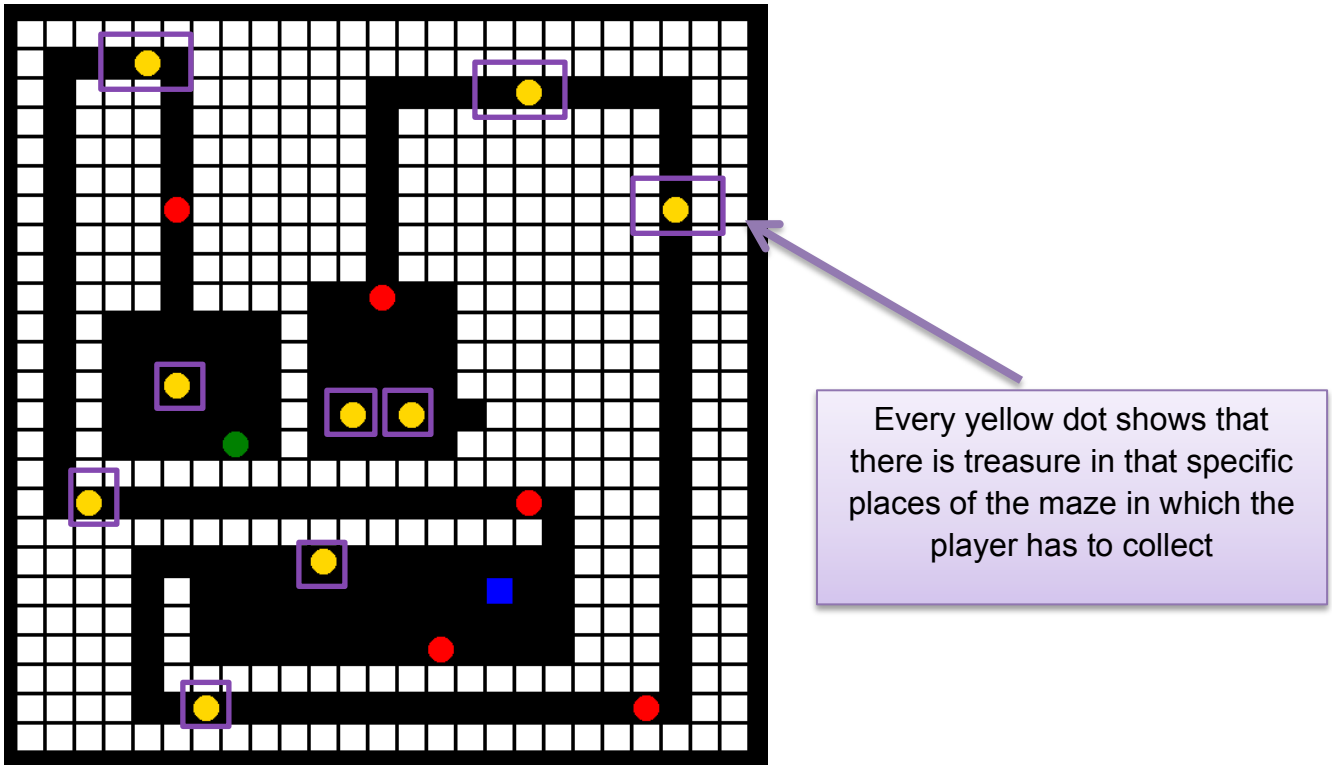


The game only lets the player move in the black areas, and when they try and go into a white wall by pressing the arrows on the keyboard, the player stays in their original position as this action is not recognised by the program.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
4	Yellow circles should be shown on the maze, when treasure/gold are located	User starting the program and the level containing the configuration of the treasure locations	Yellow circles on the level the is loaded and placed in their configured locations	See below **Test passed**

Maze at start, once the program has been run

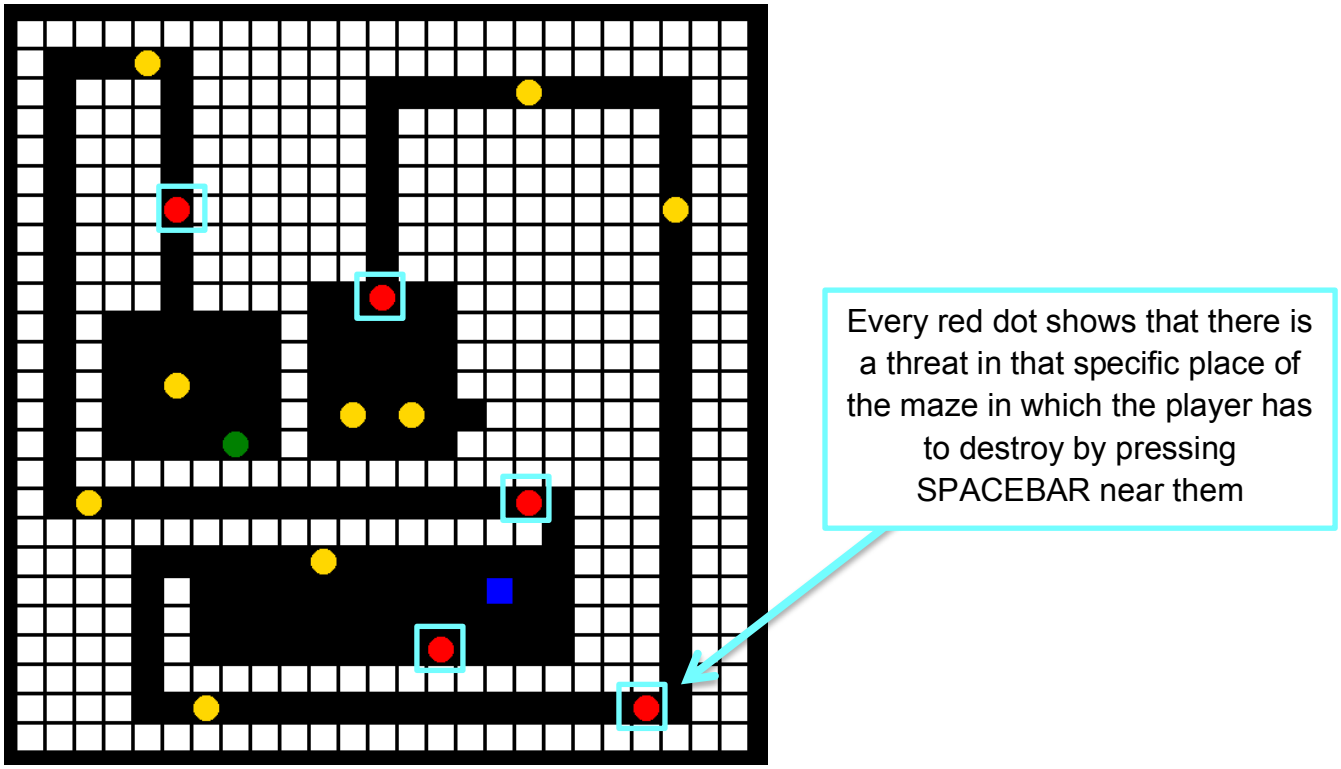


As shown in the above screenshot all the yellow circles are able to be seen in each room when the program is run. The yellow circles indicate to the player that there is gold in that specific spot.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
5	Red circles should be shown on the maze, when threats are present	User starting the program and the level containing the configuration of the threat locations	Red circles on the level the is loaded and placed in their configured locations	See below **Test passed**

Maze at start, once the program has been run

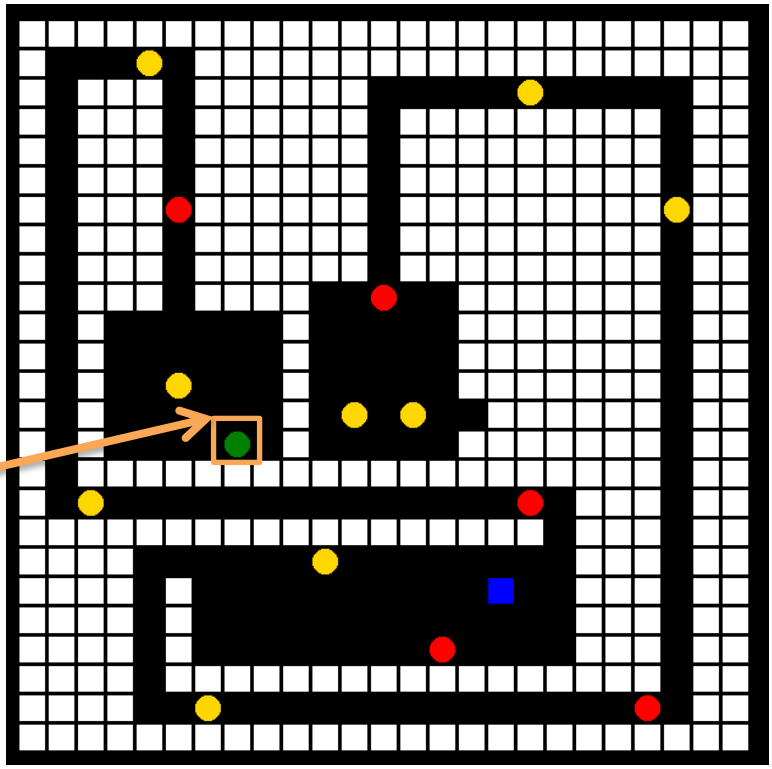


As shown in the above screenshot all the red circles can be seen in each room when the program is run. The red circles indicate to the player that there is a threat present in that specific spot. The player is required to destroy the threat in order to proceed.

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
8	A green circle should be shown on the maze, which indicates the finish point that the user has to go to.	User starting the program and the level containing the configuration of the end point location	Green circles on the level the is loaded and placed in their configured locations	See below **Test passed**

Maze at start, once the program has been run



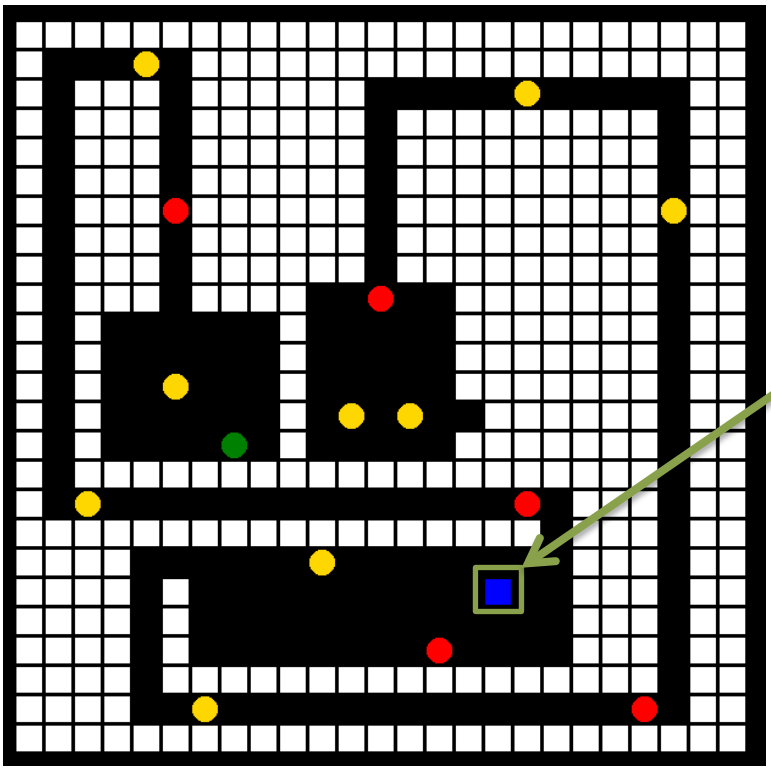
The green circle shows the player where the finish point is to complete the maze

As shown in the above screenshot the green circle can be identified as soon as the GUI has loaded and the program is run. The green circle indicates to the player that this is the place they have to go to finish the Maze after collecting all the current treasures and threats

Test Passed

Test Number	What to do	Given Input	Expected Output	Actual Output
9	A blue square represents the user playing the game. This should be able to move around the maze accordingly.	User starting the program and the program randomly allocating the player to an empty space	User being placed in a randomly assigned location that is empty	See below **Test passed**

Maze at start, once the program has been run



The blue square represents the player's current position. They are able to move anywhere around the maze, but not on the white walls. They move themselves by pressing the arrow keys on their keyboard

Test Passed

Documentation

Improvements that can be made to the program

1. **Reducing the number of lines within the code**

Currently the total number of lines within the code is 434. There are many lines that could be removed or at least cut down. This could be done by removing some lines which may be duplicate and doing the same thing or placing similar lines together.

2. **Increasing efficiency in the code**

Efficiency within the code could be increased by considering adding repeated lines of code to a routine like most of the code. For example, when the timer was ended, it was duplicated when a game ended or when a person had died. This could be added to a routine next time.

3. **Allowing a player to continue a game or not**

At the end of every game, the player has no control of the starting the game again. It would be nice to allow the player to have the choice to be able to start a new game or not.

4. **Adding messages during the game on screen**

Currently messages such as key events when a player picks up a gold item are displayed within the background window of the game. It would be nice to be able to display this somewhere within the game while the player is running so that they aware of their wealth points and if they die, the reason why.

5. **Adding moving threats**

Threats don't currently move, but if they did it would add more challenging conditions for the player to make the game more interesting

6. **Adding more rooms and not allowing the player to view the whole level**

Currently the game runs and when loaded it shows the player the whole maze. In the future, it would be good to make a version of the game where the player is put into a room, where they cannot move away from unless they choose a certain path. When the player moves, more of the maze and pathways will be revealed to them.

Reflection

In conclusion this project has been a success in terms of creating a maze game in python. I have been able to integrate most of the requirements stated into the brief on the Maze, such as the ability to collect treasures and destroy wealth. To add to this, the game automatically removes the item [threat and treasure] from the selective room within the maze when the player had either collected or destroyed it and adds the selective points to the players wealth score.

I chose the target audience of this game to be aimed for kids aged 8-12, as stated in the initial assumptions because I believe that they would be the ones whom of which are most attractive/interested in particular to online games. After doing further research into this age group, I realised that the maze should contain a graphical user interface because statistics have shown children to be more attractive in games that are visual. Also, the most successful and traditional computer games such as Pac-man, Tetris and Snake, whose majority of its users are kids aged 8 to 12 all have a graphical interface in order for the player to interact and play the game.

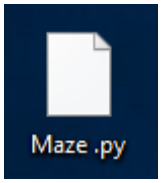
I have chosen to add extra functionality to my maze game, which was not included in the initial brief such as adding the time it has taken the user to complete the maze. By adding this time aspect into the maze game, it will allow kids to compete with each other on both the number of wealth points and time. Another feature I have decided to include is that the player has to input their name and read the instructions when they run the program. The reason why I have decided to add this in, is because the name adds a personal aspect from the player to the game, and also by including the instructions before the game starts, kids will not be able to use the excuse to their friends that they did not know what to do since it is shown in every game.

If I had more time on this project, I would have liked to add a high score table in which shows the 5 highest scores a player has achieved on the game. To add to this, I would have also liked to include wealth point updates on the user interface rather than the python program, since this would have alerted the user straight away and been much clearer to notice. The final thing I would have liked to include if I had more time, would be the ability to see the time when playing the game, not just at the finish - this is so the player is able to keep their eye on it.

To conclude, overall I think this project has been a success because I believe that this is a game in which the younger generation would enjoy and be addicted to. The game has no faults and has been designed with lots of thought as to what children aged 8 to 12 years old would like. I have been able to provide full documentation for this project, and have also created a user guide for whom a user is able to look into if they want a more in depth analysis as to how to run/play the maze game. By undertaking this project, it has not only developed my coding skills, but also considerations I should take when creating a user interface and importing this from text file

User Guide

How to prepare to play the Maze game?



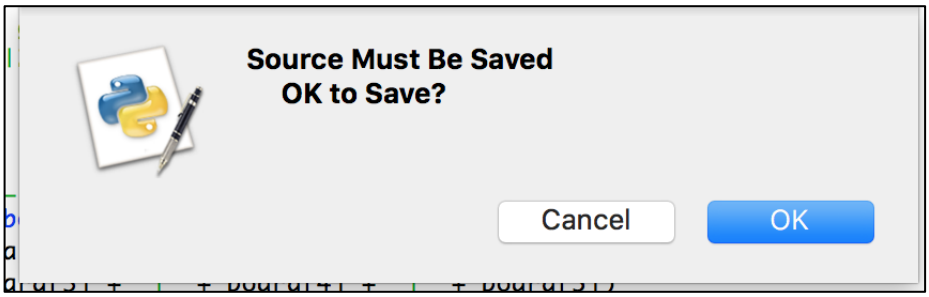
Make sure that you have downloaded the Maze.py from the website to your desktop computer, as well as all of the txt files so that the program is able to retrieve the maze layout.

In order to prepare this game or open this game in Python, follow the steps:

- From desktop or start menu, select python 3.6 (or IDLE Python 3.6 32 bit)
- Click the File menu and Open option and select Maze.py file from your chosen directory/folder. Make sure that you have all the txt documents within the same folder.
- Now you can see source code of the game program.

How to run the Maze game?

When your game program's source code is available on the screen, press F5 from keyboard or select Run Module/F5 from Run menu to run the game program.



Python may ask you that the source must be saved. Click OK and the program will be opened

Minimum software requirements to run the maze

As the Maze game runs on python, anyone whom wishes to edit it is encouraged to make sure that they are able to run python. As stated on the python website, below are the minimum requirements needed to run python.

Processors: Intel Atom® processor or Intel® Core™ i3 processor

Disk space: 1 GB

Operating systems: Windows* 7 or later, macOS, and Linux

Python* versions: 2.7.X, 3.6.X

Included development tools: conda*, conda-env, Jupyter Notebook* (IPython)

Compatible tools: Microsoft Visual Studio*, PyCharm*

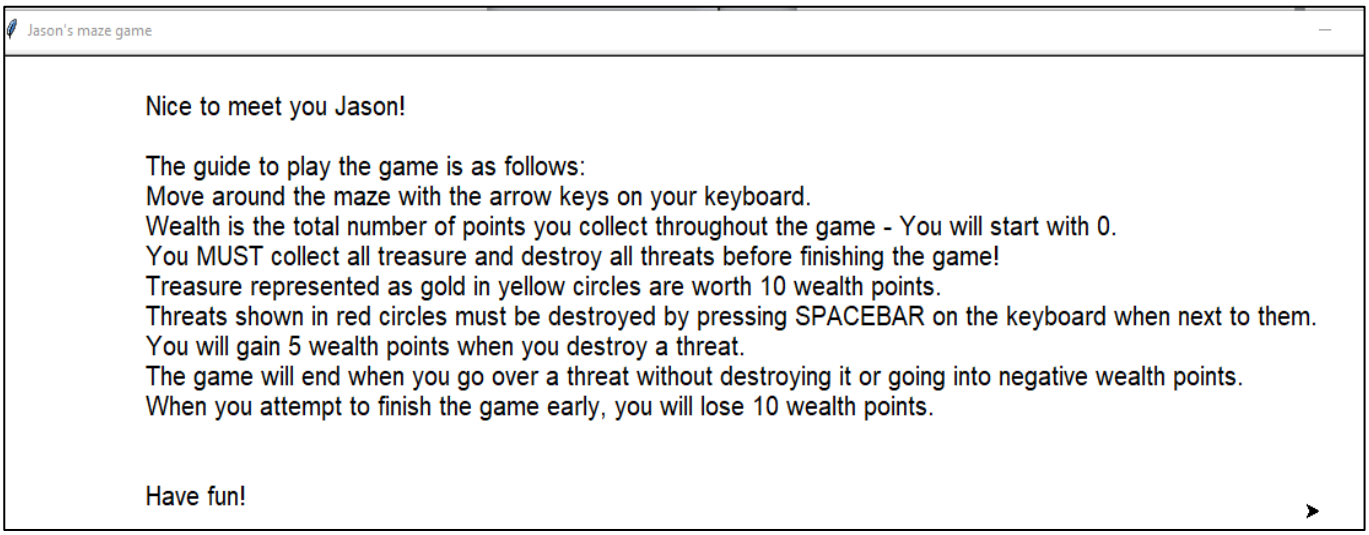
Included Python packages: NumPy, SciPy, scikit-learn*, pandas, Matplotlib, Numba*, Intel® Threading Building Blocks, pyDAAL, Jupyter, mpi4py, PIP*, and other

How to play the Maze game?

1. Once the game is loaded, you will be greeted by the welcome text onscreen like below:

2. You will be required to input your name, in order to proceed to play the Maze game.

3. Then you will see a greeting message and instructions on how to play the game, as seen in the screenshot below:



This will be shown for approximately 15 seconds and will tell you everything you need to know about the game.

4. After 15 seconds, the maze will automatically be populated by a 25x25 area. There are 10 different mazes in which you are able to play, but remember these are chosen randomly. An example of the maze is as follows:

RED CIRCLE = THREAT

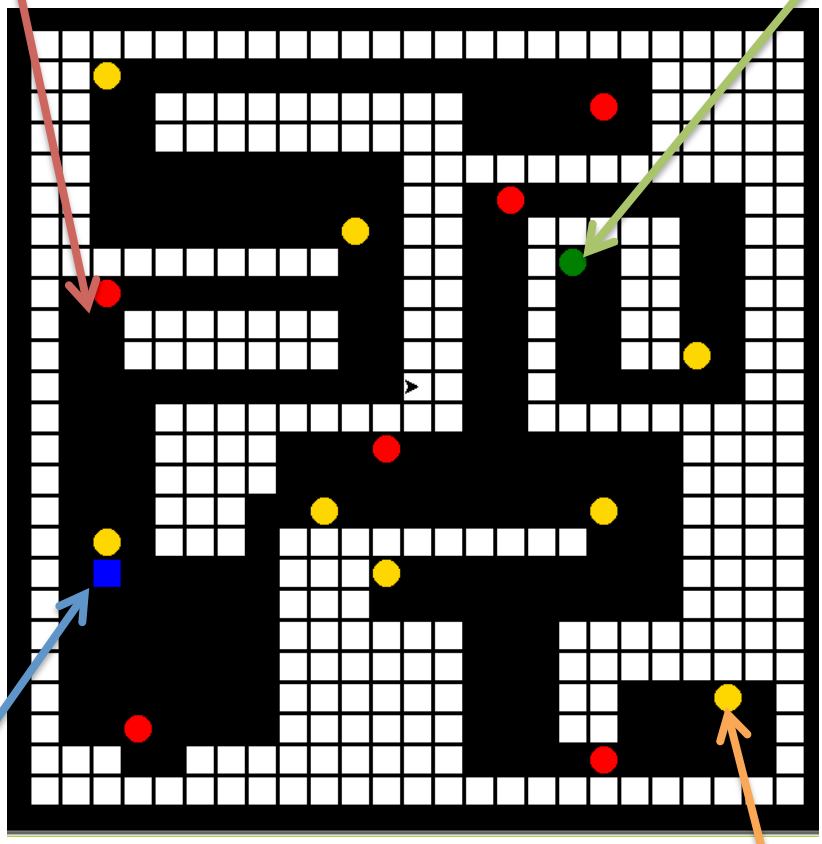
AIM: To deactivate the threats; press the SPACEBAR on the keyboard when player is next to it. Earn 5 wealth points

DANGER: If player touches the threat, then it is game over and a message is outputted to user

GREEN CIRCLE = FINISH POINT

AIM: To come to this point once all the threats have been destroyed and all treasure has been created

DANGER: If player comes straight to finish without collecting all the treasure and threats, then 10 wealth points are deducted from total and player gets launched back in starting position.



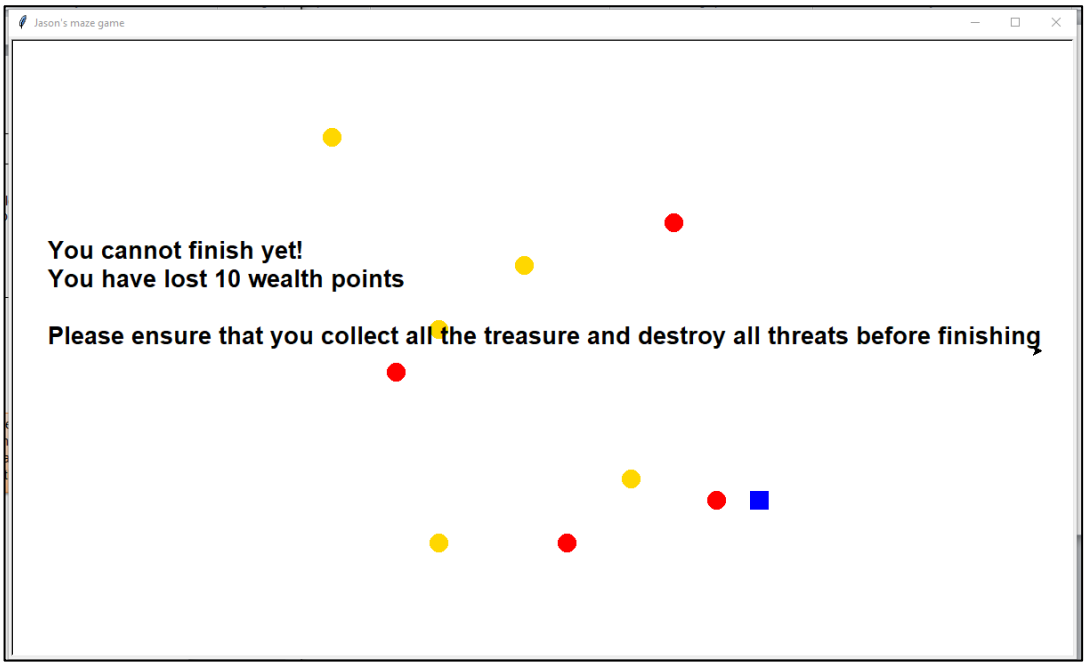
= PLAYERS CURRENT POSITION

AIM: Use the arrow keys on the keyboard in order to move around the maze

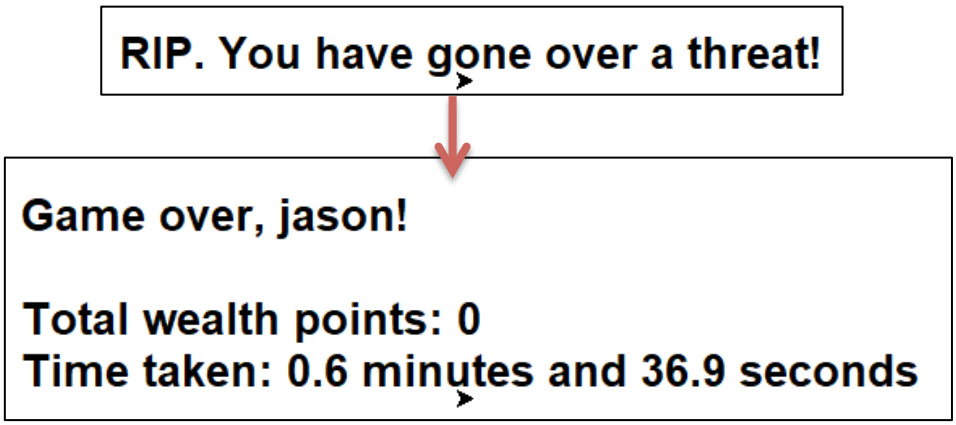
YELLOW CIRCLE = TREASURE

AIM: To collect the gold by going over it, using the arrow keys on keyboard to navigate. Earn 10 wealth points

5. Make sure that you are collect all the treasure and destroy all the threats before finishing the game. You will be deducted wealth points if you go straight to the finish symbol [green dot] – this will consequently return you to your starting position of the same game. The message is shown below:



6. If you go over a threat, rather than destroying it you will automatically lose and the game will be over instantly. The message you will get outputted is shown below:



This will automatically generate another maze which you are able to play the game on.

7. When you have completed the game successfully, the game will output the time it has taken you to complete the maze, as well as your total wealth points as shown below.

Congratulations Jason! You have reached the end!

Total player wealth points: 55
Time taken to complete: 1.0 minutes and 61.6 seconds

8. The wealth point system is rewarded as shown in the table:

Action	Result
Collecting Treasure	Adds 10 wealth points
Destroying Threat	Adds 5 wealth points
Going into a Threat	Ends game automatically
Going to Finish point too early	Minuses 10 points

***TIP**

You are able to keep track your wealth points throughout the game since it will be outputted in the python log, as shown below:

```
Welcome message displayed
Player greet and instructions displayed
Loading the Maze, plase wait...
OPENED FILE: 1.txt
Player allocated to coords: [-192, 216]
BOOM! Threat removed! Player wealth points are now: 5
BOOM! Threat removed! Player wealth points are now: 10
BOOM! Threat removed! Player wealth points are now: 15
Gold picked up. Player wealth points now: 25
Gold picked up. Player wealth points now: 35
Gold picked up. Player wealth points now: 45
Gold picked up. Player wealth points now: 55
Displayed congratulations message
```

From this log, you are able to gather how many wealth points you have throughout the game, either generated by destroying a threat or by collecting treasure.

Troubleshooting

1. If you get an error to play this game, what should you do?

If you use recommended Python 3.6, you should not get any error message to run this game. If you use any other version of Python, you may get error message or may not be able to play this game. If you still use Python 3.6 version and get error to run this game, open and run original backup copy. Assume that you have duplicate copy of this game as backup in your hard drive

2. Do you need to save this game after playing?

After playing this game, you should not save this game. Somehow if this program code is changed or modified, during closing this game, a message would come on the screen to save the program file and click on “No” button to close program file without saving

3. How to update/modify this game program?

You should be familiar with Python program and its syntax. For further information, visit www.python.org and https://www.w3schools.com/python/