**Data Specification**

The maze is defined in the maze.py file.

**Window:**

The window contains the height, length and width, which contains the map. It also defines the block sizes and the background colour.

1. win = turtle.Screen()
2. win.bgcolor("#808080")
3. win.title("Maze game")
4. win.setup(width=700, height=700)
5. win.tracer(0)
6. grid\_block\_size = 24
7. screen = turtle.Screen()

**Creating maze:**

The maze is setup up in a two-dimensional array. The walls of the maze are created in a class using the shape method to create squares and the colour method to make them white.

1. class Wall(turtle.Turtle):
2. def \_\_init\_\_(self):
3. turtle.Turtle.\_\_init\_\_(self)
4. self.shape("square")
5. self.color("white")
6. self.penup()
7. self.speed(0)
8. self.name = 'Wall'

A loop is created through the different lines in the y coordinate and the length of the x coordinate. This loop sets up the maze in a function called “setup\_maze”.

1. for y in range(len(level)):
2. for x in range(len(level[y])):
3. character = level[y][x]
4. x\_coordinate = -288 + (x \* 24)
5. y\_coordinate = 288 - (y \* 24)

**Exit:**

The exit is a class which takes the value of its position on the map so it can be appended and added to the screen. The exit is appended to an array with their x and y position from the map.

1. class Exit(turtle.Turtle):
2. def \_\_init\_\_(self):
3. turtle.Turtle.\_\_init\_\_(self)
4. self.shape("Sprites/door.gif")
5. self.penup()
6. self.speed(0)

**Treasure:**

The treasure class takes the value of its position on the map and appends it, then adds it to the screen. The treasure is appended to an array with their x and y position from the map.

1. class Treasure(turtle.Turtle):
2. def \_\_init\_\_(self, x, y):
3. turtle.Turtle.\_\_init\_\_(self)
4. self.shape("Sprites/ClosedTreasure.gif")
5. self.penup()
6. self.speed(0)
7. self.goto(x, y)

**Enemies:**

The enemies is a class which takes the value of its position on the map so it can be appended and added to the screen. The enemies is appended to an array with their x and y position from the map.

1. class Enemy(turtle.Turtle):
2. def \_\_init\_\_(self, x, y):
3. turtle.Turtle.\_\_init\_\_(self)
4. self.shape("Sprites/wizz\_left.gif")
5. self.penup()
6. self.speed(0)
7. self.points = 20
8. self.goto(x, y)

Enemies movies in random direction around the map. Whichever direction enemies are facing, enemies will look towards the direction.

1. self.direction = random.choice(("L", "R", "U", "D"))
2. if self.direction == "L":
3. self.shape("Sprites/wizz\_left.gif")
4. elif self.direction == "R":
5. self.shape("Sprites/wizz\_right.gif")

**Player:**

The player is a class which takes the value of its position on the map so it can be appended and added to the screen. The player is appended to an array with their x and y position from the map.

1. class Player(turtle.Turtle):
2. def \_\_init\_\_(self):
3. turtle.Turtle.\_\_init\_\_(self)
4. self.shape("Sprites/player.gif")
5. self.penup()
6. self.speed(0)

Player movement is defined in functions with names depending on their direction. A collision check is made to check when a player meets with a wall, which takes the player new x and y position and the walls as its parameter. Therefore, if the following x and y position is not in walls, it will return true else false. If true, this would start the collision check. If false, then no movement would happen.

1. def move\_up(self):
2. self.take\_next\_step(self.xcor(), self.ycor() + 24)
3. # self.shape(Player\_Up)
4. def move\_down(self):
5. self.take\_next\_step(self.xcor(), self.ycor() - 24)
6. # self.shape(Player\_Down)
7. def move\_right(self):
8. self.take\_next\_step(self.xcor() + 24, self.ycor())
9. # self.shape(Player\_Right)
10. def move\_left(self):
11. self.take\_next\_step(self.xcor() - 24, self.ycor())
12. # self.shape(Player\_Left)
13. def is\_collision(self, other):
14. a = self.xcor() - other.xcor()
15. b = self.ycor() - other.ycor()
16. distance = math.sqrt((a \*\* 2) + (b \*\* 2))
17. if distance < 5:
18. return True
19. else:
20. return False