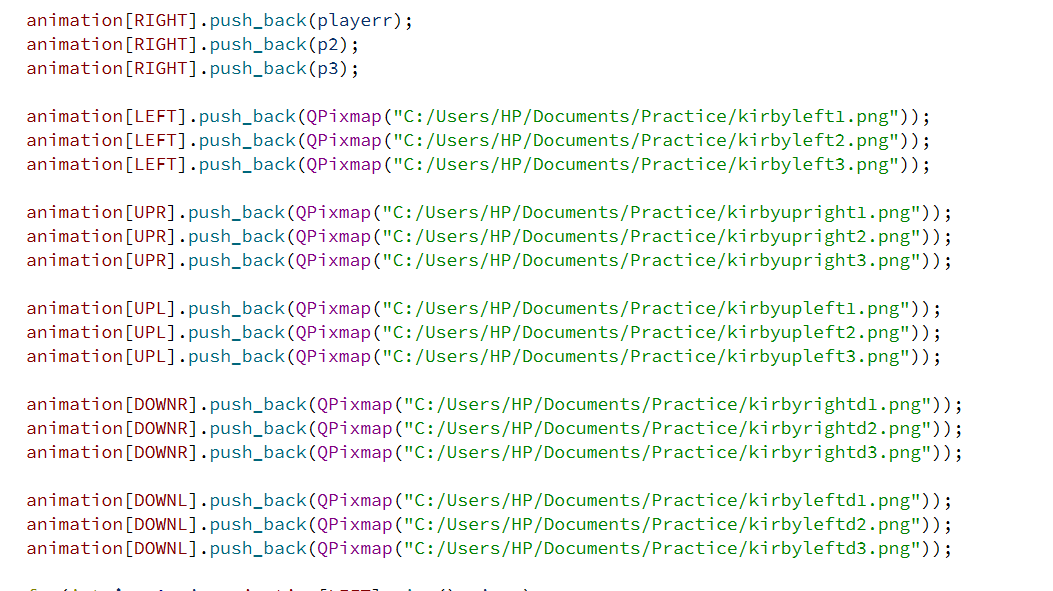
Report Milestone 1 Project

//End of the report there is who did what

Class Player:

Constructor:



In the constructor of the player we have a vector for the animation of the player. Each direction we consider it as an index we pushback the each animation of the player in that direction.

Right represents the normal right

Left represents the normal left

UPR represents up right which is basically when the player is going to turn into the upwards direction then they will turn into up right if the previous move was down right or normal right.

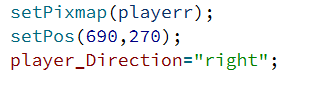
The same thing is with UPL. UPL represents the up left direction. Which is basically when the previous direction of the player is either down left or left normally. Thus the next movement or direction is up left.

DOWNR represents the down right direction. This is basically when the previous direction of the player is right or upright. Thus if the next is down and their previous is right or up right, then the down direction will down right. With its animations.

DOWNL represents the down left direction. This is basically when the previous direction of the player is left or upleft. Thus if the next direction is down and their previous is left or upleft then the next direction of the player (if down) will be down left.



This is to set the width and the height of each picture in each index.



We will set the starting point of the player right with the first picture of the right index in the animation vector.



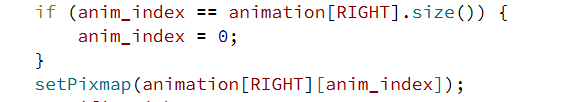
This is the reset function. When the player loses a life, we will reset the picture, the position, the row and the colomn. And also the player Direction which determines the current player direction

**Key event**

****

We have the specifier of the remove house which is initialized to zero. Once the user presses any key then this house will be disappeared from the scene.

The second if statement is when the queue of the hearts of the player is empty then we will stop the game which will stop all the moving items in the scene (the player, the enemy, the second enemy)



This will be found in the each of the key pressed events. This basically goes over, the animation photos and is the main driver for the animation (legs movement) of the player. Going over which index is specified by the direction of the player.

*The specification of the player direction.*

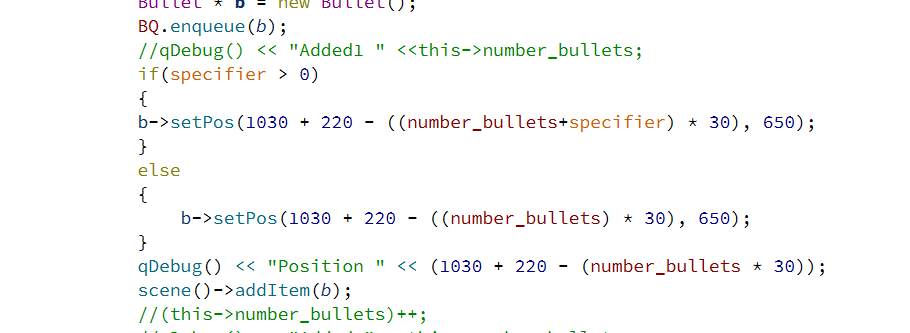
In each key we specify the player direction. There are 2 player directions. One variable which is (player\_direction) and the second (player\_direction 2). The first one includes the upright, upleft, downright, downleft, right and left. Player\_direction is the general one which only includes the normal right, left, up and down.

The general player direction will be used in the bullet shot animation. Since the bullet is being shot depending on the direction of the player.

Both the down and the up key pressed we have to specify the rightup, rightdown and the non-generalized directions.

This screenshot only shows the specification of the direction of the player in the key up pressed.

**Collision of the bullet**

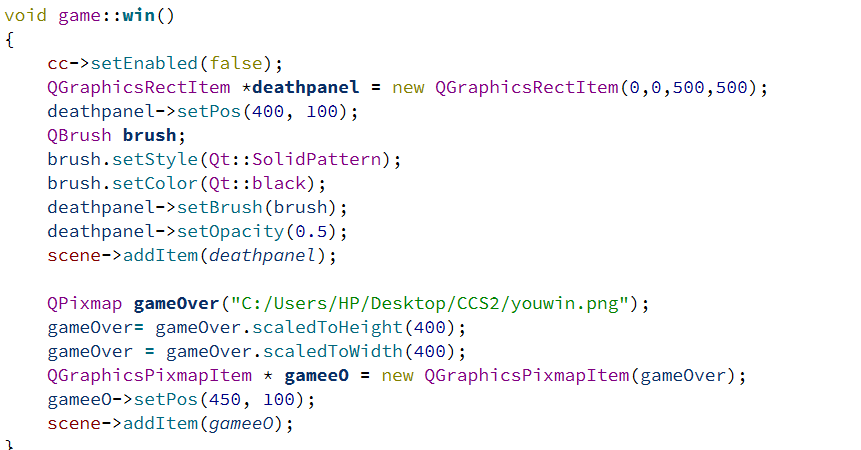


Once the player collides with the bullet it will show in the bullet age.

**Game**

****

Game stop that resets everything and will display the game over window. Which is Display death

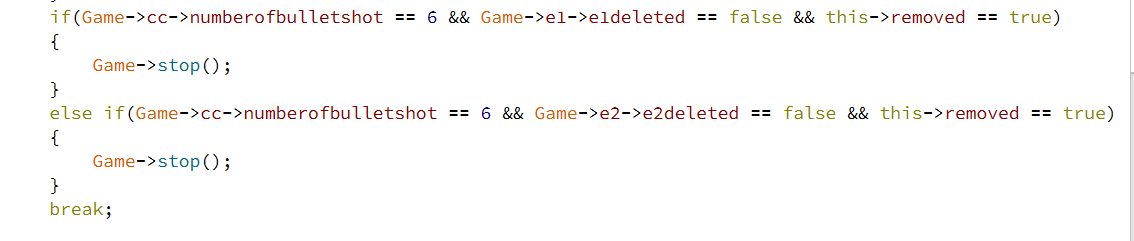


This will show the “you won” and will disable the movements of the player.

// General note.

Game over is determined by 2 things. If the all the bullets are shot and the enemies are still alive or all the hearts of the player are removed.

This is determined in:



This will be found in the bullet shot class.

**Enemy class:**

****This is a method used to make the first enemy to follow the player. NOT GRAPH

Power pallet:

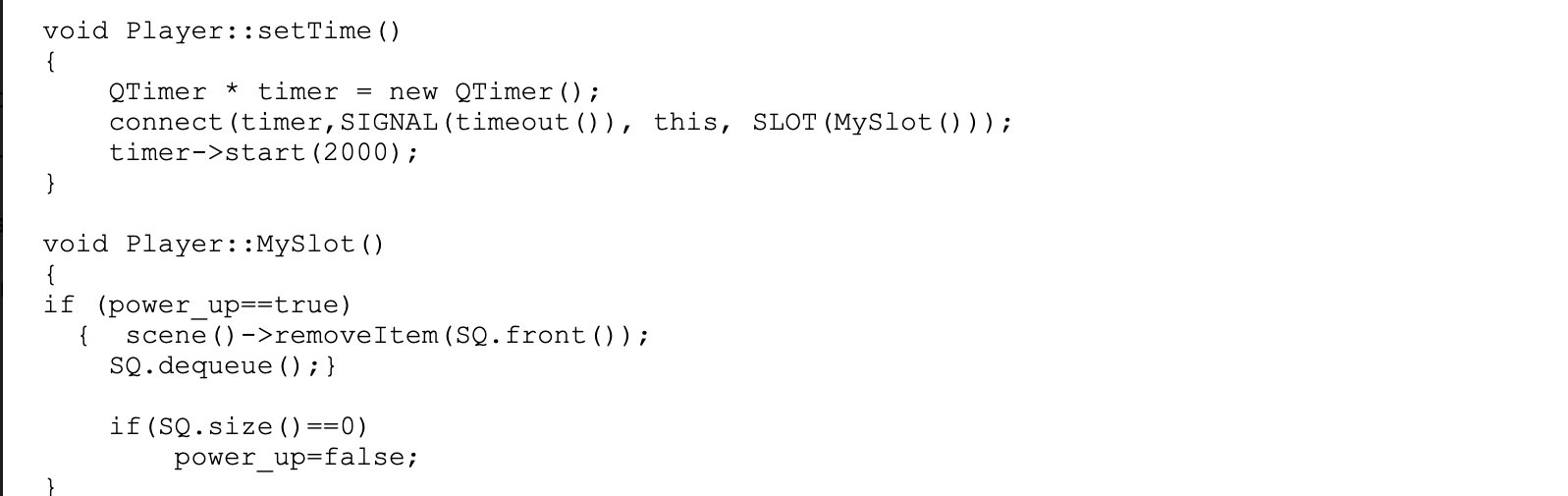
In the game class we created 2 objects of class power to be added on the maze, and we set it in specific positions

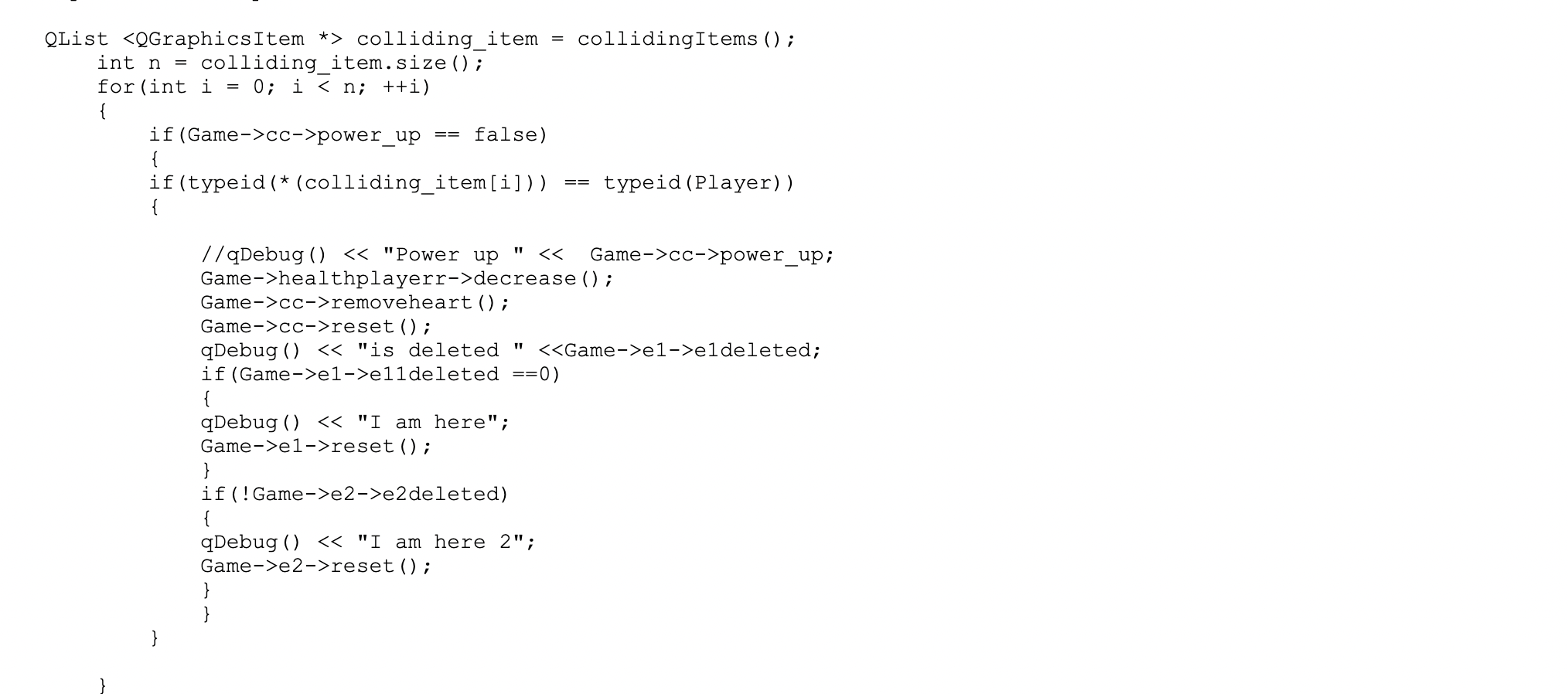
In player class, there is a function that is called: (QList<QGraphicsItem\*>power\_1=collidingItems();) this function works when the player collides with the power, it disappears from the maze. In the for loop of the collide power, there is a queue that has five objects of shape green squares that appear below the maze when the player eats the power, there is a queue of type power that has the five power squares. Also, in the for loop there is a call for set Time function that is created in the player cpp, and this function is used to dequeue each square in the queue every 2 seconds using another function called Myslot that removes the green square from the screen. In the collide loop, there is a flag that has a relation with the enemy to prevent the enemy from attacking the player during the whole 10 sec.



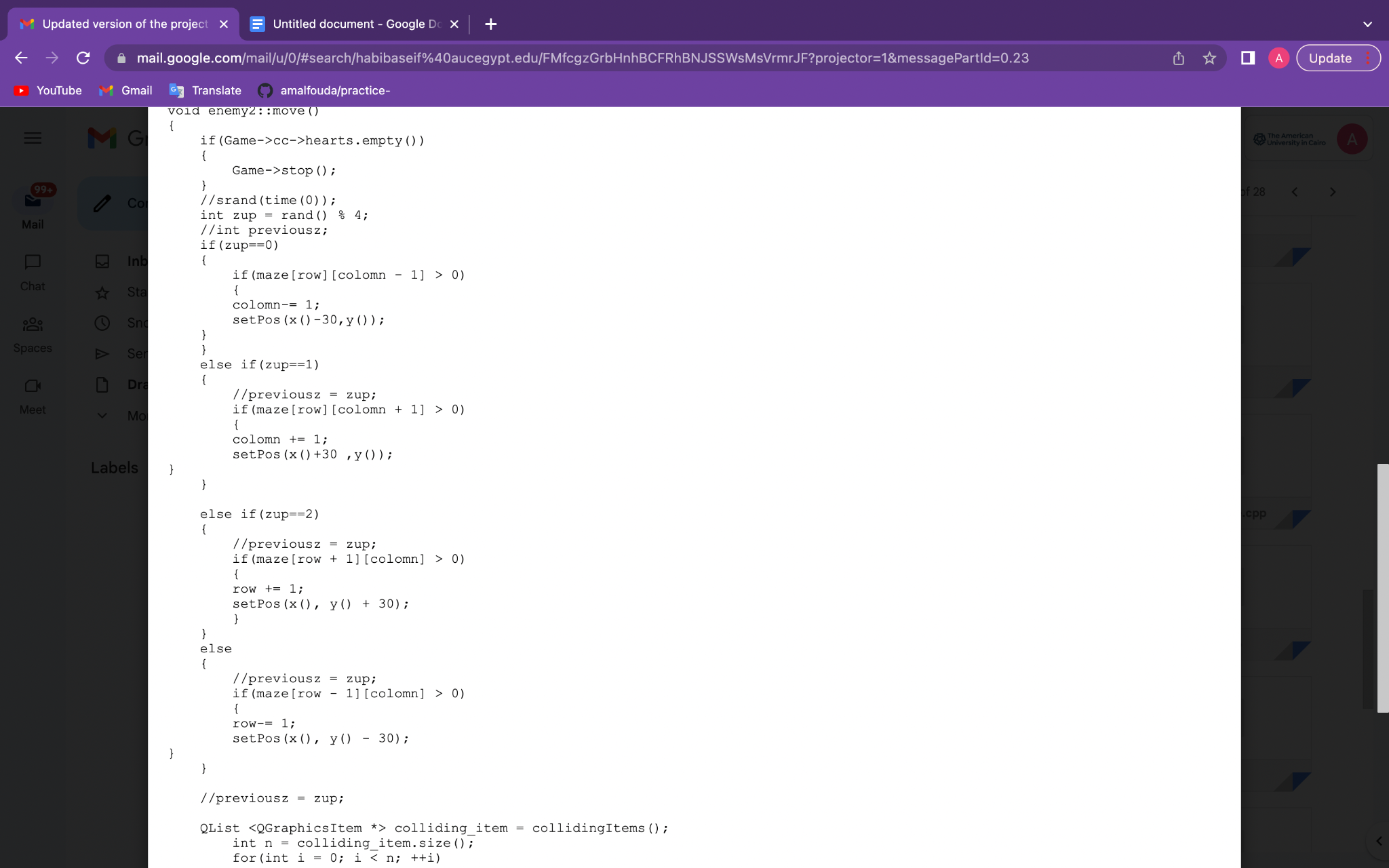
Health:

In health class, there is a decrease function that decrease the health count, and it is used in the player, in the player there is a function that set three hearts in the screen then add them in queue, and another function called remove heart that remove it from the queue, and this reflected on the

screen and removed. 



Random enemy:

In player, there is a function when the player collides with the enemy it health decreases, as for the enemy when it is shooted by the player it is health decreases. And the random enemy moves randomly in the maze according to some calculation depending on the size of the maze. 

|  |  |  |
| --- | --- | --- |
| Habiba | Amal | Both |
| Player animation  Player reset  Enemies reset  Game over  Game win  Game stop  Bulletshot  Final version of Maze  Enemy health  Scene set  Trees and houses addition  Player health(Habiba changed by doing a show function in the class of the player and a remove function for dequeing to make it more modular) | Power(including power refill)  Enemy 2 random  GUI (still not shown)  Player health (First version shows hearts in the main and dequeue in the class) | First version of maze |