Report GTA game : Minion GTA – Mile Stone 2

Farida Sarhank

Menna Emam

Yasmeen Mohamed

Before attempting milestone two we knew we needed to fix the small bugs in our milestone one such as the timer issue where the minion would not use the allocated 10 seconds it had to be invincible and the reset issue where the enemy would reset incorrectly. The timer issue turned out to be one that was due to using the same slot twice and was fixed by implementing a new function and adding a unique connection to the slot. The reset issue was a simple incorrectly placed position.

This milestone, although shorter, was much more difficult and mentally strenuous. This was due to the fact that implementing a graph far beyond a conceptual application seemed impossible. We first set out to choose which graph we wanted , the choice was between A\* and Dijkstra. After many YouTube videos and recommendations, we finally decided Dijkstra was the way to go. We looked at its code on blackboard and quickly figured it required an adjacency matrix as well as a mathematical formula for the movement of the enemies. Finally, after days of working attempting to understand and adapt the code as well as input our board into an adjacency matrix we finally had a working algorithm. However, we stumbled into an abundance of errors.

Our initial errors appeared due to the size of the matrix which we had chosen to be 15 ( the size of our board). We soon realized it was too tight of a space and figured it needed o operate in a range of 15\*15. Our second obstacle was that we had no idea how to actually work Dijkstra to move the enemies after intensely studying the algorithm we figured it would return a value, and that this value would be in a 1d array , that needed to be converted in a 2D array in order to work as coordinate. We also realized Dijkstra needed something to calculate the shortest path to and hence implemented the values px and py that mapped where the minion was live time and sent it to enemy to preform Dijkstra on. Finally, our last error was Dijkstra v reset. Our algorithm would work perfectly if the player did not die but upon dying it would exhibit weird behavior such as the enemy jumping around the board and the player dying for what seemed to be no absolute reason. This took the longest to figure out but ended up being a quick fix of adding the columns and rows to set the enemy at rather than only the position in our reset function, and just like that mile stone 2 was completed.

After we got the requirement out of the way, we knew it was time to start working on some flare. We wanted to added music , but after watching tons of tutorials nothing seemed to work , up until we realized we were attempting to down the multimedia of QT 6.4.1 when we had 6.4.0. Hence. We downloaded the newer version and music finally played.

We then decided to make coins. Those were simple and gave us no trouble as we had used the same setting procedure millions of time through the code.

The login feature of our game was our second bonus. It didn’t take much effort except when we were learning how to properly read and write from finals and the one error we had that it only remembered the last person who registered. This was fixed by simply adding a break after the first condition. It was also fun designing it to fit our needs.

No game is complete without the option to play again. This took the most time to brainstorm as all are options seemed complex. But after skimming our code, we realized the answer lied in key press event. So, we built a rest all function that was extremely easy to build and now have a thankfully functioning play again and exit feature.

We attempted a select character option but failed to implement it due to our inability to understand how to pass the reset all function to the menu.cpp file all whilst calling its object in minion as you can’t recursively call. We all attempted the shop but felt too constrained on time, but feel as though it could have been implemented had we had more time. Similarly, with the levels.

In conclusion, this project took us more hours than we can count and more effort than we ever imagined, but seeing as did everything together it also taught us more than we can comprehend. We now understand true patience as well as the importance of OOP, pointers, breaks, properly mapping out code , functions , graphs , and so much more .