CSCI 115

Team Project: Maze Game

Team members: Steven Nguyen(ID#109933505), Anurag Uppuluri(ID#110352456), Israel Perez(ID#109666108)

This project is a 2D turn-based game using a graph(matrix), and one of the shortest path algorithms(breadth-first search) to calculate the enemies’ movements.

The goal is for the player to have to reach the treasure chest in order to win the game. The player can use the UP-DOWN-LEFT-RIGHT keys to move, and SPACEBAR to shoot an arrow(lightning bolt) to kill one enemy at a time. The player will run out of arrows after a few times he/she has shot. The player can reach the arrow set(power source) to reload more arrows. There are walls that the player and the enemies cannot walk through. There are bushes where only the player can go in and hide from the enemies. Enemies can not walk through the bushes. Each time the player moves UP/DOWN/LEFT/RIGHT, the enemies also move and find the shortest path to the player. If the enemies touch the player, game over.

There are classes that have been provided like: Maze class, Player class, Enemy class, Time class, Wall class, and Commonthing class. There is also a main.cpp that incorporates all other classes for seamless gameplay experience. We added one more class, which is the bush class. The bush class is used to create bushes. We used breadth-first search(BFS) to find a continuous chain of predecessors of each node of the graph, right from the goal node back upto the start node and thus got action taken by the start node to reach it’s first child. We check if the player coordinates(x,y) is the same as enemy coordinates (x,y), then we determine that the game is over.

The display function is called several times a second. As soon as one of the status flags (gameOn, hasWon or hasLost) is set to true or false the display changes to reflect the appropriate screen. These flags are modified primarily in the key and the Specialkeys function but even before these functions return anything, the display changes, understandably.

The major collision check for the enemies is the one with the player. It is done for each case up/down/left/right which corresponds to each keypress that the player does and is handled by the function Specialkeys. Specialkeys also loops over all the enemies each time the player moves and computes the shortest/shallowest path to the player from each enemy. While the auxiliary function BreadthFirstSearch returns the predecessors/parents of all the nodes (squares in the grid) in the graph, the function BFS that calls it, backtracks over the chain of parents starting from that of the goal state (player position) all the way up to the child of the start state (enemy position). And then it compares the position of the child of the start state to the start state and determines the action needed to reach it from the start state i.e., either up, down, left or right and this is the action that is returned as a string which is then fed to the moveEnemy function, so each enemy can take the first action toward the player.

The way BFS works is that if there is no obvious path to the player (the goal state) i.e., the parent chain is broken, no enemy (each being its own start state) will move as the BFS function will not return any action for each enemy to take.

When the game boots up, it will show the user a menu that will list out different modes ranging in difficulty level. The player can also choose a random level which will give them a random layout to play every time. Depending on the mode they decide the game will automatically give them a layout to fit their choice. Lastly, the user will have an option to quit the game by simply pressing ‘q’.

* Steven Nguyen:
  + Display a maze based on the values from a matrix, and make a layout for the enemies and another for the player. //(load\_level) and (load\_maze)
  + Possibility to load levels from a file. // (key function)
  + Collisions related to the actions of players. // (key function)
  + Determine if the level is completed. // (key function)
  + Collisions related to the actions of the enemies. // (key function)
* Anurag Uppuluri:
  + Shortest path function for the enemies. Breadth-first search.
  + User control: move up/down/left/right.
* Israel Perez
  + A menu to start the game or exit the program (display function)
  + Three modes from the start menu (key function)
  + Arrow collisions with the enemies (key function)
  + A personal touch to improve the project.