Rogelio Romero

CSCI 115

Shortest Path Report

This application creates a grid-based level and allows for the selections of one of four characters. Several terrain elements exist that have different costs associated with them. The characters can move to a desired point in the grid, considering the terrain of the grid spaces to determine their movement path. Characters will avoid moving to the same space as other characters if a character is in their path.

The terrain of the level can be randomly generated using the generateTerrain() function, found in the maze.cpp file. The maze class contains elements associated with the grid, such as grid size, that are needed to be able to generate an appropriately sized level. The terrain generation function uses the Diamond-Square (Midpoint displacement) algorithm and a 2-D vector to create the level. In Diamond-Square Algorithm, the four corners of the grid (e.g., (0,0), (0,bound), (bound,0), (bound,bound)), are assigned a random value between 0 and 120. Next, the diamond step will take the average of the value of the four corners, adding an offset, and assign that value to the center point of the grid. The square step will then use that center value and the corner values to assign values to the halfway points of the grid (e.g., (0,bound/2), (bound/2,0),…). The algorithm will repeat with smaller steps each recursion until the whole grid is filled with elements. Diamond-Square algorithm requires a grid size of 2^n + 1, so the algorithm may create a level that is bigger than the desired grid size. These extra values will be truncated when drawing the grid elements in the main function. The level generated will then be stored in a text file named, “terrain.txt”.

In the main function, a file containing a level is read line by line, and character by character. It assigns values based on the character in the text file to a 2-D character array that will hold the terrain values for the grid. This process begins from the top left corner of the grid and will proceed to the right, and then down. Truncation will occur of the bottom-right most elements of the text file when appropriate. By default, “terrain.txt” is being read, but other text files containing levels could be used to replace this file. For example, “level.txt” could be read by inputting that into the parameters of the string read. This process also initializes and places the graphical data in the appropriate location, using a vector for each of the different tiles to store this data. Vectors were used as they are more efficient when data needs to be reallocated. Graphical data for highlighting tiles was also created in this step and will be drawn after the textures to ensure that the highlighting is in the front of the texture.

Characters will be initially placed in a different corner of the map. Placement will occur if there is a space on the grid that they can traverse or exist on. For example, the fish cannot move outside of the ocean (or the beach), so it will be placed in an ocean tile if one exists. If such a tile does not exist, the fish will not initialize. Characters can also not be placed on a tile that is already occupied by another character. The process is repeated for the other characters, as appropriate.

A character can be selected by left-clicking on the top-right corner of the tile that the character is on. That character’s state changes to selected, allowing us to select the path for that character to travel. The user can then right-click the top-right corner of the tile that they want the character to travel to. Using Dijkstra’s algorithm, the shortest path from the character’s current location to the selected point will be found. This path will be highlighted, and the character will begin moving to the selected tile. The path-finding algorithm will avoid traveling to a tile that another character is already on and will travel around them. Additionally, the characters will try to avoid unpassable terrain but will cross it is the only path that they can take. In the current version of the project, if the destination tile is occupied, the application will crash. This is intentional behavior and will be changed to a more reasonable solution in future developments.