Team:

LEADER: San Tran (CWID: 889054417)

Kailie Chang (CWID: 890005309)

Justin Castillo (CWID: 889919924)

Victoria Tran (CWID: 889773024)

CPSC 481 Assignment 3 – Progress checkpoint 2

In our previous progress checkpoint, we stated that we had planned out how we would advance through our progress of the project. Since then, we have completed multiple tasks, and have fully developed our maze grid that the Minotaur AI will use in the program. In addition, we have created the basis for the Minotaur's behavior in its own class. In this report, we will explain our in-depth analysis of how we initiated these tasks and how we plan on proceeding with our overall project specifications.

As discussed in our previous progress report, we used a Depth-First-Search algorithm to create the maze. To achieve this, we first created a blank slate that represented the foundation for the maze with four walls in each direction (N, S, E, W). The DFS program carves a path through the grid by accessing its neighbors and randomly choosing one to be its next cell in its path. If a cell exists next to a pathway, it is considered a wall and cannot be chosen as a neighbor by the algorithm. The algorithm continues making a path until a dead end is hit (where there are no more neighboring cells because it is surrounded by walls, the maze border, or its former path). Once a dead end is hit, the algorithm back-tracks until a suitable neighbor is found for a given cell. This algorithm is run until the entire maze is completed. Once here, each cell wall is rendered as a white space forming the outline of the maze.

After generating the maze, we began developing the Minotaur AI.We created a Minotaur class that spawns the Minotaur at cell 0,0. As for the physical properties of the Minotaur, we have initialized it to start at cell (0,0) in the maze. We have set the minotaur's appearance as a simple pink rectangle that represents the Minotaur's location. For testing purposes, we will keep its appearance this way and will change it when we complete the majority of the project. As for the movement of the Minotaur, we initially based it off a Breadth-First-Search algorithm to find its target. This however, was recognized to be a mistake in our program as it proved to have a slow traversal through the maze. To improve

efficiency, we decided to change the basis of its behavior from a Breadth-First-Search to a Depth-First-Search type algorithm.

To develop this algorithm we added a function to our maze class that will serialize our maze. The serialization entails categorizing each “cell” and creating a list of “neighbors” that the current cell can access. In a sense, after the maze is completed, each cell will have a set of children to which it is adjacent to. In this way were are able to better generalize the graph traversal problems in the terms that we discussed in class. In terms of parents, and children. Of course, it took us a bit to come to that development. You can see in our function dfs_ver2 take the more generalized approach than our dfs and our bfs function, which took in the tuple which we defined as a set of cartesian coordinates. Now we plan to adapt our bfs algorithm to take in the serialized inputs.

So the plan is that the Minotaur AI takes a DFS approach to initially find the player, and once it it close enough to the player it will transition to a bfs algorithm.

And possibly if we can fit it in. We would like to apply mini-max algorithm so that the minotaur will minimize the amount of steps to get to the player, and maximize the number of steps the player must take to get to the goal

By the next checkpoint, we hope to initialize a victim for the Minotaur to hunt for. To accomplish this, we plan on making a simple program that will choose arbitrary points throughout the maze to represent a “lost” target trying to find its way out. If possible, we want to eventually make the victim use a smarter approach to finding the maze’s exit, such as leaving “bread crumbs” or having a map of the maze to give it an advantage over the Minotaur. If the Minotaur is able to compete with this version of the victim, then it will be considered highly efficient in its hunting. However, for now we will focus on creating a simple victim for testing purposes.