Josiah Gallegos

Prof. Tucker

CS372

25 January 2017

CS372 Final – Write Up

Going into the project I decided I would make a maze game where the player would have to drag the Hero through the maze to a flag without touching the walls nor the spikes in order to win. I began by creating all of my own images on a browser application called Piskel. After making sure I had all the images I wanted and I could access them, I coded the functionality of each. For the Hero, I made it so once the player clicked him, he would then be able to move just by moving the mouse. Clicking again would cause the Hero to stop moving whenever the mouse moved. Having the mouse move outside the frame would also stop the Hero from being moved. I then created separate classes for the vertical walls, horizontal walls, and the spikes. Each class object had their own x and y values in order to be painted onto the frame in their own spot; they also had their own image variables. The SpikeThread class I had implement Runnable. I created SpikeThread objects (spikes) and started their own threads in order for them to move up and down the screen. From here I wanted to be able to repaint the frame every time the spike moved so I ended up passing the whole MazeGame object to the SpikeThread class in order for the spike objects to access the paint method. Another aspect of the spikes was the fact that they needed to have their own unique run methods because they wouldn’t be following the same instructions. To get around this, I gave each spike an ID, that way when the run method started, it would check the spike’s ID and tell it how to move based on that. Moving on to what I found to be the hardest part of the project: collision checking, first I had to find out the dimension of each image and not just their image frame (all being 32x32). Second, I realized that the idea was to check if any point on the Hero had hit any point on the walls/spikes. Instead I just decided to check whether one of the four corners of the Hero hit any point on the walls/spikes which would then end the game. The Hero needed to check collision with all of the vertical walls, horizontal walls, and spikes, so what I did was have each of the corresponding classes implement an interface, CollisionCheck, which contained one method: collision. Since each object had different dimensions, collision would check whether the passed in Hero’s x and y variables were inside each objects unique widths and heights. Now that the hard part was over, I set out to complete the most time consuming and repetitive (yet enjoyable) part of the project which was created the level for the game. I gave the Hero, each wall and each spike their own specific x and y values to be painted on the frame. I ended up making a total of 62 wall objects alone. In the end though, I was satisfied with my game and it reached the goal I set for it.