Christian Bondalo, Leslye Erazo, Jesson Gannon, Alice Gao, Surya Makkar, Tina Wang

Introduction to Programming 9/10 B

Ms. Gerstein

27 January 2016

Our project draws inspiration from the classic game “Scary Maze” with some personal touches of our own to make it unique. To make the maze more interesting, we created four mini-games inside of the maze that the player must beat in order to win. Each of the games was created by smaller groups within our team, KodetoSuccess.

Surya created the “Light-Catching Game” which, when finished, gives the player a lantern they can use to navigate through the game. This is one of the simpler games to create within the maze because one of our old Raindrop games was used as the base code. The original catcher was swapped out, and a few lines were written to create a light within the lantern.

The second game is the “Brick-Breaking Game” made by Christian and Leslye. This game is modeled after the “Game of Pong” we created earlier in the year. The player fires a cannonball from a cannon which serves as the paddle. However, instead of an AI opponent, there is a wall of bricks that must be broken through in order to find the key to get through the door. Once the cannonball touches the key, the winning end screen pops up and the player is taken back into the maze to continue on.

Next, there is the “Crocodile-Jumping Game” developed by Jess and Tina. This game consisted of arrays of rocks and crocodiles spread out equidistantly in the x direction. We used our previous knowledge about for-loops and arraylists to create the game.The objective of the game, like the title suggests, is for the player to successfully hop over each rock without touching a crocodile or falling off, so that they escape the sewer system under the maze. If the player does touch a crocodile, their lifeline decreases, which is shown at the top of the screen. If the player’s lifeline reaches zero, the game ends and the player must replay the game again. One of the most difficult parts to code for this game was making the player move when specific keys are pressed. Although making the player move itself was not too difficult to code, the glitches that came afterward were difficult to fix, including the player flying randomly out of the screen and the player continuously hopping although stationary. The only outside source the team used was another student in another school who is also a good programmer. While he did help Link stop vibrating on the rocks, the user has to be very precise with Link jumping onto the middle of a rock or else he falls right through in the water. Additional help from the teacher was needed to help code for the up, left, and right keys to make Link jump and move.

The final game, which was the hardest to code, is the sword fighting game, which serves as the boss battle of our “Scary Maze” created by our lead programmer Alice. In this game Link (the player) battles against the final boss, a Shadow. The Shadow and Link’s health decreases depending on where the Shadow is. The Shadow’s movement was coded by mapping the x location of Link. In addition, a small animation was created for Link to swing his sword. Once the Shadow’s health reaches zero, Link returns to the maze. Once Link returns to the maze he continues until he reaches the green portion of the screen. There, Link has finally reached the “end” of the maze and his quest is finished.

As a whole, we are very proud of the variety and complexity of our code. We incorporated almost every skill and knowledge we obtained from tech class in our game, including functions, arraylists, and for-loops from the “Raindrop Game” and the “Pong Game”. However, we also added new characteristics to the games and made the game as intriguing as possible. This project definitely shows our improvement and evolution from the beginning of this class up until now. Although, at times, we faced the unexpected challenges and bugs, we were able to fix them and learn more as a result.