Anima Anwar

Ms. Gerstein

TECH III

13 June 2013

*Hansel and Gretel Game*

In the group, I was the graphic designer. Although I assisted in the picking of graphics for the game, that was mostly a team effort and, if anything, we focused on our own games separately. The dynamic of our team was different, since our game was unique. We had a fairytale theme, with a four mini-games in each. Because I started my game without a clue how to go about it, I was honestly not able to tend to my team as well as I should have. This is evident in the fact that our time management was not the best. Still, I think we were really on the same page as far as the idea of the game itself.

When I began writing the game, I used a simple background and then put in place the image of the house, the mouse (as an ellipse) and the start and end points of my class Path. My game is supposed to be the mouse following a path until it reaches in the end. I tried to use the line tool in order to create a spline; however, creating a bunch of individual line segments was a less than efficient way to create a curved path. Instead I got help and decided to use the curveVertex() option in order to create a thick spline. With the help of the instructor, I then created a function where if the mouse was pressed, the spline (black, with a white background) would save as an image.

Using this image, I used a pixel function where the mouse could react according to the color of the pixels. This was a new process to me, so I needed the help of the instructor in order to follow the syntax and construction of this function. After the writing up of this code, I realized that the path in the saved image could be thick, in order to make the game easy enough. In the game, the black and white path is not displayed, but still in process. In place is the background picture and opaque spline of the path (for graphic purposes, to make it look as though the path blends with the background.)

Then I set limits for the screen for winning and losing. First of all, if mouseX and mouseY does not equal the black pixels, the gameStatus changes to gameOver. If the mouseX is evident where x>460, then gameStatus changed to run gameWin. This means that reaching the end of the path is a win.

The game was still considered particularly easy because there was no time limit. Therefore, instead of setting a time limit, I wanted to make the game itself more difficult. I decided to insert something similar to the rain-catcher activity we did, where a timer would set off random falling objects that when intersecting with the mouse, would run a different function. The images of candy were the falling objects (the theme being Hansel and Gretel) and the mouse was the image of Hansel and Gretel themselves. If the candy fell on them, they lost. This is where I found my bug that I had trouble fixing. It turns out that the ellipses that the void check had falling where offset way to far from the image. These ellipses did not have a stroke or fill. So basically, there were random ellipses falling that caused my game to lose, but I did not see them when I moved the mouse over the path. I put the image of the ellipses back, and my partner Mike Batkiw explained to me that the ellipses were offset. That is when I changed it so that the location of the ellipses that were being checked matched the candy images. The game now runs the right way.

During the team working, although we were all working assiduously, the amount of work we put on ourselves made it hard not to stress over time constraints. We ended up sharing some of our coding in order to get everything done efficiently for the mini- games. I also revised the original flow chart in order to try and explain the pre-defined processes better.