Carlos Reyes
9/7/13

SwordRun Analysis

Time Review:

It was estimated that, in total, approximately 50 hours were put into making this game. The project was begun at the end of Tuesday, August 27th. On the 27th, most of the time that I spend working on this project was spent sifting through the JGame website, and looking at sample code and the tutorial, looked for game sprites, and planned the general idea of my game in total I worked about 8 hours on this. On Wednesday the 28th, I continued to plan out the gist of my game, writing the Readme file and the gitignore file (to ignore .DS_Store, an OSX hidden file), and trying out, and experimenting with the sample code that came with the JGame download. This took about 9 hours, the readme and gitignore only took about half an hour however I worked on modifying the default code (to allow the spaceship to move in all four directions, and change the speed) for a significant amount of time, I then moved on to working on getting the sprites that I found passed into the game. On the 29th I worked on getting several sprites to work together to form an animation, from here I moved to attempting to get the object to walk. On the 30th I spent around 10 hours, first I was able to enable the movement animation when I was walking, and get the animation to stop when the movement key was no longer being pressed. Second I was able to put different tiles in the background of the game to give it a grass background. I was also able to make roads, however I later decided to scrap that code, as it was something that I wished to implement later. The next day (the 31st) I spent about 11 hours working and was able to get a machinegun as another weapon implemented into my game. However the machine gun was a only a small part of my day, one of the more extensive parts of the day was reading through the JGame API and figuring out how to go about how to step off the current screen into another one. Additionally I spent an incredible amount of time learning about how java classes interact with each other and how the JGame engine was set up so that I could break the hero and enemy classes out of the Game class where they existed at the time. Moving these classes out of the Game class led to part of the game such as the hero's weapon not working. On September 1st I spent about 10 hours on the game. First I was able to fix the code so that the guns were re-enabled. Next I noticed that whenever I was moving to another screen, the game would be delayed. As it turned out I was creating a new hero object every time I went to a new world. Thus I fixed this issue so that only one hero was created. In addition to this I created a class Block that were objects, such as boulders that could be pushed around the screen. On the 2nd I spent a short period of time resizing the playing field so that gameplay would be more aesthetically enjoyable, I allocated the rest of my time to enable the boulders to "roll" around by enabling animation when hit by the hero, this all took about 8 hours. On September 3rd I spent about 10 hours on my code, first I created a wall class which is simply an item that, when hit, doesn't do anything but also prevents the hero from moving through it. I then spent the rest of the time refining the gameplay: adding limited health to the hero, making a sandboxed world, and adding a graphic when enemies are hit. On September 4th, I spent about 9 hours on the game, although official levels were added with different types of enemies, most of the time spent was refactoring code.

Planning:
The planning stage of the project started as soon as the sample code was observed, upon seeing the possibilities that JGame had to offer, it was planned that a more adventurous top down game such as legend of Zelda would be made. The design plan was to first prepare all of the needed objects such as the hero, the enemies, the blocks, and the boss. Next was setting up the worlds in which the player can move between. Finally the objects would be placed on the worlds as seen necessary to make interesting levels. The planned time was quite high for this project and was approximately the same or a bit less than the actual time it took to make the project.

Status:
Although the code does compile and fully function, it does not truly reflect how the Game was meant to be designed. For example, much of the code requires several changes in multiple classes to simply add a new Enemy with a different graphic.

In the implementation of the machine gun, it was expected that bullets would in a more animated like way because two exist and each bullet is offset by a certain amount.

Several dependencies exist in the source code. One of these dependencies is within the paintFrameInTitle method. Another example of dependencies in the code is illustrated in the hit method of Hero because there is an instanceof in the if-tree.

One of the main things that could have been done is using abstract classes and subclasses to remove several very large if-trees. This would both make the code cleaner and more extensible.

It is fairly clear how to extend the code, if, for example enemies with different graphics are desired, it is quite easy to do this by extending the enemy class.

One problem with the game is that within the hero java class the method to switch weapons has far too many if statement which would be resolved with a class and subclasses used to deal with all of the cases. As a result this is also the method that would require the most work to extend.

This code would most easily be tested by playing the game, some of the easiest aspects of the code to test would be the how the objects interact with each other physically, as all that must be done is play the game.

Conclusion:

Without a doubt the types of defects that were seen the most were issues with access to JGame engine methods due to the JGame engine not being referenced correctly. The most time consuming errors were those that required searching through files in the directory to find a change that may have been refactored in one location but not the other. The most consuming bugs to fix were those that passed objects into several different methods but missed several instances of old objects.

The most refactoring was needed in the Game and Hero classes, as these are the classes that are more frequently seen and worked with. Hero was refactored to move the move function from an if-tree to several subclasses. To reduce the probability of these huge if-trees in the next project it would be good to start out by using abstract classes and subclasses to take over those statements.

Future Work:
If I were to work on one part of the code right now to improve my grade, it would certainly be the Hero class. Although I cleaned up one of the methods, many remain full of if-trees that I would love to get rid of. On project expansion for this that I would have loved to do is to have an infinitely large map with random stuff on it. Thus I would start planning that each world have its own unique walkway which would be a collection of tiles that look like a road, and at the end of each playing field a new one with randomized enemies and challenges appears.