**Work in Progress Report**

Major developments/breakthroughs(reference specific code please):

One of our first major developments was finally getting our Collision Detection scratch to work after a few weeks of work. We eventually came to the realization that the method we were previously using to detect collisions just wasn’t going to work for us. We were originally reading in a layer of tiles that were going to be solely used for collision detection, then converting them to rectangles in order to check for collisions. We realized that this method wasn’t working because the x and y values of the tiles in the Tiled map were different than the x and y values of the screen, which were the ones we needed to create a rectangle for bounds checking. In order to fix this, we decided to create an Object layer in our Tiled map, and read in those objects, creating an Arraylist of rectangles based on their information. After doing this, collision detection worked perfectly for us, and we are currently in the process of implementing it into our main game maps.

Another one of our major breakthroughs was getting our D-pad to work in our main project. Using the animation 1 project from MrGFHCI as a basis, I was able to create a d-pad that depending on which arrow is clicked the nDir is updated. When nDir is passed to the ActChar’s setDir method the characters act method reacts by change the direction and animation that is played, how ever the animation are not quite working.

Fonts were another thing we worked on, we found a way to import fonts so they are resizeable without the use of having to import separate fonts using the heiro tool. When creating a new libgdx project the “Freetype” extension must be selected, to add to an existing project you have to add lines of code to the build.gradle files.

Also, we were able to achieve success in the battle screen by setting up all the basics in the screen and getting a working health bar for the enemy that decreases when you click the attack button, and then brings you to a “you win” screen” once you successfully destroy the enemy. Through this development, we now have all the basic code set for when we work harder concepts into the battle screen in the future, for example, the enemy attacking back through a turn based system.

Another development was getting the camera to pan with the player. When the player reaches the edge of the map, the camera will translate one screen length over to show the next portion of the map, moving the player accordingly. We accomplished this by

Major Challenges/setbacks( reference specific code please):

One of our major setbacks was getting the Collision Detection to work. As previously stated, it took a while to get our code to work, due to our method of performing collision detection. We were experiencing a lot of challenges with reading in a layer of tiles designated to be our “Collision” layer (e.g. trees, walls, etc), converting these tiles to rectangles, adding them to an Arraylist, and then checking to see if these rectangles intersect a rectangle around the player. This method did not work, and we tried many different things (initially we thought it was an issue with the Orthographic Camera), before coming to the realization that it was a problem with reading in the tiles themselves, and the difference between the x and y values of the tiles and the x and y values required to create the accurate rectangle. All of these challenges were fixed as soon as we used Object Layers for collisions instead of a Tile layers.

Another major challenge of ours was getting stuck on image buttons. We wanted image buttons to use for the using weapons buttons in the Battle Screen instead of text, but whenever we tried to make an image appear on the screen, the code crashed saying the “Table must have a skin set” for the drawable picture, while there was a skin set. We plan to look more into this in the coming week.

Another issue we had was in the Battle Screen we have the different weapon’s damages set to different values, but we are having issues figuring out how to pass the variable from the weapons screen back to the battle screen while also updating the screen. We cannot figure out how to both update the screen and pass the damage variable.

Another recent setback of ours is in our main project switching to the battle screen has been giving some trouble, switching between the menu and game screen works, as well as switching between the battle and continue screen however switch to the between battle and game screens causes the program to crash. Also the animations for the character only seem to work when the character is moving downward in the game screen.

We also encountered another setback within our Camera Panning scratch. Our camera pans just fine, but we’d like to incorporate the concept of allowing the player to stay within the boundaries of the map, using a similar collision detection system to our previous Tiled Collision Detection scratch. However, this isn’t quite working the way we want it to, and we’re not quite sure why. Regardless of whether the map on the screen that the player is moving on has an edge or not, the collision detection will occur. We think it is due to the fact that the map objects and their coordinates are potentially being changed with the changing camera, but we’re not sure. We plan on looking into this in depth next week, and we hope to have the problem fixed soon.

Any modifications to your specifications/release schedule:

We have not integrated updated versions of most of scratches this WIP into our main project yet. We plan to do this in the near future, so that we can start working on more complicated concepts that require all of our earlier scratches together to accomplish.

We have not worked on switching to a new screen when the character reaches certain points on the map yet. This will happen later on in our release schedule than previously planned.

**Description of your scratch/test program:**

Describe the generic concept you needed to test out:

We created a few scratch programs to test out a few generic concepts:

1. Tiled Collision Detection Scratch (GitHub name: ITWCollisionDetectionScratch): Our goal with this scratch was to test collision detection betweens the player sprite and various elements of the map within Tiled.
2. Battle Screen Scratch (GitHub name: ITWBattleScreenScratch) - The section of the game the player will go when they encounter an enemy. We made it so that the user can choose different weapons, attack the enemy, and eventually kill the enemy, meaning you win
3. Font Scratch (GitHub name: FontTest)- Working to see if there was a way to resize a font without having to create a new file. Using the freetype extension we were able to use 1 font file to create many fonts with different attributes within the code use a simple method.
4. Tiled Camera Scroll Scratch (GitHub name: ITWTiledScratch- UpdatedOrthoCam): Our goal with this scratch was to allow the camera to pan around the map as the player moves. We eventually expanded onto the concept of this scratch, as we wanted there to be an element of collision detection between the player and edge of the map, so that the player could not go beyond the edge of the map.

Source any web site/book that helped you with that concept:

1. Tiled Collision Detection Scratch: We used the following sources in this scratch:
   1. <https://www.youtube.com/watch?v=MT5YwZsQnF8>
   2. https://github.com/libgdx/libgdx/wiki/Tile-maps
   3. <https://www.youtube.com/watch?v=xdc_1Pf-jnA>
   4. <http://stackoverflow.com/questions/22152925/libgdx-tiled-map-collision-detection>
   5. http://www.gamefromscratch.com/post/2014/04/16/LibGDX-Tutorial-11-Tiled-Maps-Part-1-Simple-Orthogonal-Maps.aspx
2. Battle Screen Scratch - We got the idea for how to make the health bar from (http://stackoverflow.com/questions/24356672/how-to-create-a-healthbar-in-libgdx) , with a rectangle that shrinks in width as the attack button is pressed
3. Font Scratch - [*https://github.com/libgdx/libgdx/wiki/Gdx-freetype*](https://github.com/libgdx/libgdx/wiki/Gdx-freetype)
4. Tiled Camera Scroll Scratch: We used the following sources in this scratch:
   1. <http://www.gamefromscratch.com/post/2014/04/16/LibGDX-Tutorial-11-Tiled-Maps-Part-1-Simple-Orthogonal-Maps.aspx>
   2. <https://github.com/libgdx/libgdx/wiki/Orthographic-camera>

Describe the code and the lesson that you learned from it:

1. Tiled Collision Detection Scratch: We learned a lot about Tiled maps and Object Layers, and how to read in and manipulate them in our game. We also learned what about how to perform collision detection using rectangles, as well as how to create and set the coordinates of rectangles. In this scratch, the player can move around freely on the screen with the four arrow keys, but a collision detection will occur if the player attempts to make contact with any one of the three plants on the screen,
2. Battle Screen Scratch - We learned more about how to manipulate things to make them do things you generally wouldn’t think of - for example using a shrinking rectangle for the health bar. As well, through trying to make image buttons work, we learned more about buttons and what does and doesn’t work.
3. Font Scratch - Learned a way to control fonts within libgdx
4. Tiled Camera Scroll Scratch - We learned that there are several ways to handle how to control the camera, the one we are currently using is where once the character reaches the edge of the screen, it translates the camera one screen length, however another approach we may implement is having the character remain static in the middle of the screen and the camera scroll around it. This scratch is also teaching us a lot about collision detection and Object Layers in Tiled as well, as we are attempting to do collision detection between the player and the edge of the map. It is not currently working, but we’re learning a lot about the characteristics of Object Layers in Tiled.

Describe any challenges that you enjoyed in integrating this scratch code into your major project:

As of right now, we have not yet integrated any of these scratches into our main project, with the exception of the Battle Screen Scratch. However, this battle screen scratch is an older, out of date version of this scratch.