**Work in Progress Report**

[WRITTEN BY KESTY]

Major developments/breakthroughs(reference specific code please):

* Created a generic player class that would deal with everything involved with user input to a character
* Created a generic map class that would load tiled maps (.tmx files) and create Box2D bodies accordingly
* Dynamic music decided by the tiled map class rather than hard coded
* Managed to add a foot fixture to the main body to deal with jumping
* Figured out the difference between beginContact and preSolve (world’s contact listener)
* Animation based on state of the player (idle, direction) [incomplete, no animation for airborne states]
* Figured out that body.getLocalCenter is a lot better than .getWorldCenter (gives you the true center as opposed to what it thinks is the center)

Major Challenges/setbacks( reference specific code please):

* Jumping isn’t consistent (jump height lower when moving as opposed to standing still)
* Figuring out how to prevent the player body from sliding down a slope (current approach: using raycasting in order to manually increase the player’s y position rather than letting the physics handle it)
* Modifying the map class to account for a 16x16 tile size (only compatible with 32x32)
* TiledMapTileObject not supported (thanks libGDX, nothing i can do to solve this)

Any modifications to your specifications/release schedule:

* Adding a couple other steps of finalizing the base engine before adding anything else.

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

1. **Game Engine**

Describe the generic concept you needed to test out:

I needed to create a base engine for our game. We’re going to be building on top of it and adding features as necessary, but it’s an engine that is functional and can be used as a sort of template for any game - all it would need is a bit of modification to suit their needs.

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

Main Source: Don

https://github.com/captainkesty/vidyagams/

Screens: <https://code.google.com/p/libgdx-users/wiki/ScreenAndGameClasses>

Fixture filtering: <http://www.aurelienribon.com/blog/2011/07/box2d-tutorial-collision-filtering/>

Generic Map Class: <https://bitbucket.org/dermetfan/libgdx-utils/wiki/net.dermetfan.gdx.physics.box2d.Box2DMapObjectParser>

Original method of collision between a body created from the tiled map and a coded one:

<http://badlogicgames.com/forum/viewtopic.php?f=11&t=12470>

Current method of collision:

None, I figured it out myself. You can turn the bodies in the object layer into fixtures and assign them a maskbit, then assign the fixture to a body.

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

In this WIP, I wrote a default engine for use in our game. In my attempt to extend this WIP by modifying the properties of the physics to suit my needs, I found the limitations of Box2D and TiledMapObjects. I also found out how much of a benefit preparing my code for scaling to a larger project is, because it made my life infinitely easier and the code more legible as I went further into the project.

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

None, this is literally the base of the major project.

**2. Isolate Map Scratch(ScratchTestMap)**

Describe the generic concept you needed to test out:

-isolating the generic map class from the game engine program in order to understand the code at work and how it integrates itself into the base program

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

Don: https://github.com/captainkesty/vidyagams/

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

The properties for the map are made in Tiled and then imported into Android Studio using the Box2DObjectParser. This gets the fixtures and objects from the tmx file where they are defined. You then create a camera using the libGDX class OrthographicCamera to display the map on the screen.

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

Did the opposite of integrating scratch. Isolated concepts from the base game. I had to figure out which sections of code were important to the generation of the map and which ones were unnecessary. It wasn’t difficult since the variables and methods were named appropriately.

With each WIP, you will be submitting EVERYTHING. Organization is key. When I go to the groupwork folder**, I should see your project submitted in the following format:**

YourLastName: Under this folder will be the following folders:

**Asana Specs**: Your Asana calendar will have a task that contains a github link to your project and scratches. Please add any comments within this task that can give me a better understanding, like : “It does not work.”

Even if you provided the link to the same project in a previous task from a previous month – go big – add it again.

**Documents**: It will hold all of your documents: journal, WIP, Specs, Release schedule, list of sources, and all the other documents that will be submitted in your final project.

**Releases**: There will be a folder for each release, with one folder CLEARLY telling me that it is the latest, stable release.

**Scratch**: There will be a folder for each scratch concept that you tested before you integrated it into your final project.

**Peer Assessment:**