**Work in Progress Report 1**

Major developments/breakthroughs(reference specific code please):

We were able to load a tiled map, and grab objects from it to add into the box2D world. The user can move a player sprite, and interact with the world around him (just hitting other boxes). We were also able to use the Box2DLights library to create a “flashlight” and another “lamp” type light that interacts with the world and has shadows and everything

Major Challenges/setbacks( reference specific code please):

Before using Box2D, we had started off attempting to manually hit detect. We got this working in a scratch, but adding gravity to it caused a lot of problems. At certain locations, the player sprite would phase through the obstacle, or suddenly have unlimited jumping abilities. We fixed this, but it didn’t work well when implemented with a Tiled map. The code for this can be seen in the TiledMap scratch as the old code is still there in its own file.

Another challenge was creating a scrolling camera. For some reason, world-units never seemed to work, and it was always rendering wrong. We found out that it has something to do with SpriteBatch.draw(sprite) vs. Sprite.draw(SpriteBatch).

Any modifications to your specifications/release schedule:

We updated features of the first few versions of our game as we ended up using box2d instead of the original manual hit detection that we had planned.

|  |  |  |
| --- | --- | --- |
| **Version** | **Release Name** | **New incremental features of this release** |
| 1.0 | [Alpha Hydranoid](http://bakugan.wikia.com/wiki/Alpha_Hydranoid) | Create and import a map/level (using Tiled) with solid tiles. Have hit detection with Box2D on the hero |
| 2.0 | [Apollonir](http://bakugan.wikia.com/wiki/Apollonir) | Create ambient lighting using Box2DLights |
| 2.1 | [Blade Tigrerra](http://bakugan.wikia.com/wiki/Blade_Tigrerra) | Hero becomes animated |
| 2.2 | [Clayf](http://bakugan.wikia.com/wiki/Clayf) | Non-infinite jumping using contact listeners in Box2D, and the hero can run instead of walk |
| 2.3 | [Dragonoid](http://bakugan.wikia.com/wiki/Dragonoid) | Spikes |
| 3.0 | [Dual Hydranoid](http://bakugan.wikia.com/wiki/Dual_Hydranoid) | The hero is able to shoot |

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

Describe the generic concept you needed to test out:

Scratch: Box2DLights

We needed to test loading a tiled map, and using box2d for collisions (we did this in the TiledMap scratch). After that, we even added box2dLights to make some cool lighting effects (Box2DLights).

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

We used Connar Anderson’s youtube videos to use libgdx and box2d. He had some great videos on loading a tiled map, creating chainshapes with objects from the tiled map, and adding lighting.

[Link here](https://www.youtube.com/watch?v=_y1RvNWoRFU&list=PLD_bW3UTVsElsuvyKcYXHLnWb8bD0EQNI)

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

We load a tiled map, and then getObjects() from it. Using that, we look for PolyLineMapObjects, and create chainshapes to add to the Box2D world. Box2D lights uses *rayHandler* to manage all the different cone, point, and chain lights, much like Box2D uses *world.*

We learned that Box2D as a physics engine is a lot more well documented to be used with libgdx than it was for processing. Many more people have messed with box2d besides just the creator of the library. However, Box2DLights is very poorly documented, as there isn’t even an API for it!

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

One of the main challenges we had was reorganizing the scratch to make it release compatible. We had to reorganize everything so that it the screen class could be implemented. This caused stuff to break everywhere. A very frustrating issue was has the render() function never seemed to work, and that’s where all of our rendering code was. It turned out that the game class was always calling render(float delta) in ScrPlay, so that took us longer than it should have to figure it out.

**Peer Assessment:**

We both agree that we did equal work, so the mark should be weighted equally. 100:100