**Work in Progress Report 4**

Major developments/breakthroughs(reference specific code please):

-player death screen

-player collisions with obstacles

-better inheritence

-raycasting

-damage system

Major Challenges/setbacks( reference specific code please):

-rearranging code from each other

Any modifications to your specifications/release schedule:

-no modifications

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

**1. 893StomperScratch (Rueban)**

Describe the generic concept you needed to test out:

* Death screen and game reset(Still working on it… not pushed yet)
* Player collisions with “harmful objects”(spikes, etc.) and temporary immunity
* Player jumping on enemies and killing them
* ”HUD”(used for debugging so it’s not pretty)

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

* IntotheWoods team: https://github.com/spidermanchild/IntoTheWoodsMultScreens
* Don and Kevin’s code for hit detection
* Mr. Grondin’s ButtonScratch: https://github.com/Mrgfhci/LibGDX1

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

* In the main game class, enum is used to set game states. An updateScreen function then sets the screen based on the current game state
* When the player body collides with any part of the enemy, an immunity function is called which has a 5 second timer task that sets a boolean bImmune to true which doesn’t allow any collision detection between enemies and players
* If the player’s foot sensor collides with the enemy body, the enemy is destroyed(same method bullets get destroyed)
* When player health reaches 0, the game state is set to DEAD and the screen is set to the death screen that just has an image and a button to return to the game screen
* When returned the player’s position is reset and all bullets and enemies are destroyed(Still working on this, not pushed yet)

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

* none since I made it in the final program then separated it into a scratch file

**2. 893InheritanceScratch (Kevin)**

Describe the generic concept you needed to test out:

* Simplifying classes using interfaces to reduce the amount of code and improve functionality
* Abstracting players, enemies, enemy spawners, etc. into a giant array of entities which can all be updated and drawn instead of individually like with player, enemy list, etc. (Still working on this)

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

<http://docs.oracle.com/javase/tutorial/java/concepts/inheritance.html>

My own ideas about how OOP should work

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

It was a lot of work to remove all the code I was saving by doing inheritance right </self-promotion>

**3. SpawnerScratch and Raycasting**

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

Spawner Scratch:

* Simplify enemy spawning system
* Rewrite Rueban’s code to allow for manual spawning (debug purposes) and application of damage
* Implement rudimentary AI (enemy stops after certain distance, this is all it will be until I find out how to implement behaviour trees)
* Added a very low level inheritance to the enemies (FastEnemy derives from Entity)
* Changed EnemySpawner so that all it contains is a position rather than a full body setup
* Changed render methods to Kevin’s method

HOW DAMAGE WORKS: THE NOVEL

In the same area where the bullets are being flagged for removal, I check the opposing fixture for an instance of an Integer in the userData. If there’s an Integer, that means that it’s health, and that the opposing fixture is an Enemy. I subtract health based on the bullet type and damage value (default 1), then in the same place where I render the Enemy sprites, I check the health and remove it if the health is less than or equal to zero. The reason I do this is because I don’t have access to a variable within the FastEnemy or Enemy class, but I do have global access to the userData that I assign to it.

Raycasting:

* Draws a raycast line from a controllable body and draws a circle at first intersection with another body
* Pressing A moves the body towards the end of the raycast line
* If the controllable body moves to a certain distance to another body, it will stop and begin firing using the raycast line as a trajectory

Sources:

* Distance checking: <http://badlogicgames.com/forum/viewtopic.php?f=11&t=12813>
* Raycast: <http://www.iforce2d.net/b2dtut/raycasting> (converted from C++ to Java)

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

Rearranging Rueban’s code to allow for manual spawning was pretty difficult. I had to change the arrays to arraylists to allow for infinite enemies spawned, and I had to write a bulk of the code to let myself do manual spawns of enemies. I also had to rewrite a bulk of the code to let enemies render their sprites and handle their movement.

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:**

Don 100

Kevin 100

Rueban 100