**Status Report**

David, Matt

**List of programs. Clearly describe the problem that you are solving. Please put the date that you worked on it:**

<https://github.com/DavidPeet8/Terarriag12>

David: Code/Releases/Release4

Matt: <https://github.com/MattM7/Discrete-Detection-in-Processing>

David:

Mining and placing (March 24 - now)

* Date on this one is kind of arbitrary as I have been working on different aspects of this for a long time, but this latest aspects only recently
* Get data from json as to hitpoints of specific block you are breaking
* Add blocks to players inventory when they are broken intelligently. Add first to any index that already contains that type of item if it is below stack limit. If that is not possible add to next null index

Viewports keeping aspect ratio(March 28 - April 2)

* Use extend viewport to scale the game to the window on resize so that no black bars are visible
* Keep aspect ratio

Implement hit detection scratch(March 23 - March 29)

* Use Matt’s discrete hit detection scratch to do hit testing for our player
* Speed up said hit detection by modifying the array we feed to it

Clean code (April 1 - April 2)

* Json values prefixed with jv, json reader with jr

Inventory set up (March 24 - now)

* Create all objects needed and general structure of inventory
* Give player items to begin the game

Change to polling hit detection instead of event driven (not begun)

Matthew:

Bullet through paper problem.

* Improves discrete detection by checking all the possible spots it will go. Doesn’t actually work yet.

Porting Discrete Detection to Processing.

* The same sort of hit detection I was doing in LibGDX now in Processing.

**Major Challenges/setbacks( reference specific code please):**

David:

* Viewport was not scaling properly as it was supposed to, did not realize that I needed to pass in the camera I was using as well. It was creating a new camera when it initialized. Thus as annoying as it is, each screen must have its own viewport (Release4: ScrPlay)
* Implementing hit detection was a bit of a pain, but overall went pretty well. Matt programmed the hit detection very modularly so I only had to fix a few bugs, and then the main annoyance, making it more efficient. I fixed a time bomb issue with matt, did not take very long, we just needed to reset acceleration on contact so you did not just plummet through the map after a short time period. Main time suck here was efficiency. Took a while to conceptualize and program how to get a subset of a 2D array so that I only had to hit test against blocks in the players immediate vicinity, instead of the whole map (was lagging like crazy) (Release 4: SpriteDiscrete, ScrPlay)
* Pulling json durabilities was also a bit of a pain. I was not sure how exactly to handel where the information should be stored and accessed before I finally decided to have each tile keep its own json data including its durability, and decrement it inside each tile on hits. (Release 4: Tile)
* Trying to set up an inventory and hotbar system intelligently and at least relatively modularly is difficult as it all more or less ties into other pieces it is very difficult to break down into smaller steps (Release 4: InventoryObj, Item)
* Giving the player items to begin the game is a bit hard, no idea where I would initially get the json values for the initial items (Release 4: Item)

Matthew:

Bullet Through Paper / Speculative Contacts (Code/Scratches/DiscreteHitBTP)

* I thought the bullet through paper problem was fixed but the numbers I gave it just made it look like it worked.
* I dislike the way the blog post I read about Speculative Contacts from handles it. They move the Sprite forward one unit at a time and hit detect every step along the way. I think I can move by the Sprites width/height before hit detecting. The way the blog does it is definitely easier so I might end up using it.

Porting to Processing

* The y-coordinates flipping was a pain to fix.
* Extra lines that set the dNewX and dNewY were needed in the checkMoveX() and checkMoveY() functions and I’m still not sure why but it works.

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

David:

* <https://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/utils/viewport/ExtendViewport.html> learned that I needed to pass camera to viewport

Matthew:

* I learned about the idea of discrete detection and the bullet through paper problem from <https://katyscode.wordpress.com/2013/01/18/2d-platform-games-collision-detection-for-dummies/>

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

David:

The new parts of my program include, a viewport that is both resizing on screen resize, filling the screen as I would like, and keeping aspect ratio. I gained knowledge about how exactly the viewport functions by reading the libgdx API for the viewport. As well my program now contains a hit detecting player with his surroundings. The player can also mine blocks in his direct vicinity. The blocks even include durability values so some are more difficult to break than others. This taught me how to integrate a partner's program into the main program, and how to program scratch programs by figuring out what things I liked and found annoying when implementing the code. I also learned a significant amount about structure and efficiency. I got this from improving the efficiency of Matt’s hit detection by modifying the size of the array I was passing to it, and from making many architectural decisions with respect to the inventory and how much I can store in subclasses.

**Side note:** I found possibly my favourite funcion ever in intellij; the ability to collapse whatever code you would like and label it. You are no longer limited to collapsing things like functions

Matthew:

* I learned how to use the Git Tool in Processing to push my code.