Team G: *The Fighting Mongooses*

Game: *Bel Nix: Tactics*

Mentor: *Lee Sparks*

Mentor Meeting Notes

The main point that we gathered from our meeting with Lee Sparks is that we should put an absolute focus on making the game playable and play it every opportunity we have, because a game is made through iteration, not documentation.

The following is the rest of the notes from the meeting and are more or less ordered in terms of importance and relevance to our game project:

* Find fun in playing a working version of the game, not in the docs you write for it
  + Get the game on screen and play and play and play
    - Take notes on improvements from the playable version and implement
      * The game you write down is not going to be the game you make
* The docs are good listing key values and pillars of the game and how you want the game and the player to feel
  + Refer back to these docs while you play your game
    - The game is made through iteration, not documentation
* Stability of the playable game is HUGE
  + The game needs to always be playable by the team so it can be iterated on
* Organize yourselves and your tasks around iteration
  + Keep in mind when your role/task is required for someone else (this makes it a much higher priority)
* Focus on what excites you to make the game better and work on that
  + As a player, how does the game feel?
    - As a player, what do I want to be able to do?
      * Implement that
* Use the playable game as a guide of what to do next, not future tasks that you planned before
* Make two lists:
  + Complex (what is completely new about our game)
    - May not have been in a game already because others tried it and realized it didn’t work, so be careful of wasting time going through the same ordeal
  + Simple (what is in our game that has proven successful in other games)
    - Proven to work and are safe
  + Decide to focus on complex first (because you know simple is a given), or simple first (because you don’t want to possibly waste time on a complex idea that doesn’t work out)
* Make sure to consider things that are easy on the table-top version but would be difficult to emulate on the computer
  + And vice-versa to be able to implement features not possible in the table-top version
* Make two lists:
  + Everything necessary to make the core game playable and fun
  + Linear development (tech and fluff for extra cool stuff)
* Take a look at the menu of Super Meat Boy for an example of a simple mission select menu that still gives a representation of a cohesive world
* Document the process of making the game
  + If a new developer joins in, they can look at these documents to know who to implement their work into the game
* Docs for UI should focus on values of the UI, not the full flow and layout
  + Iterate the UI in the game, not on the ppt
* Think about how you want information displayed to the character
  + Example: on a hit, do you want a number shown, a flashy effect, a word or phrase for the hit, a hit reaction, sound effect, etc.
    - Find out what is most important for your game out of the given options and implement that
* For saving the game:
  + Being able to save in the hub between missions only makes it seem a bit more like a puzzle game, while being able to save mid combat may make it feel more like a tactics game?
  + Also a question of complexity of persistence
    - Lot less variables we have to worry about saving to file if we don’t allow the player to save during combat on mission maps
      * Even a temporary checkpoint save system (Zelda: Majora’s Mask owl statues) would require a complete overhaul of what we are deciding to save
        + We have to make sure every single variable is being properly saved and loaded
* One method of task organization is to translate tasks into complexity, not time
  + This is easier to understand and quantify