ACTION SCENES AND EQUIPMENT CONCEPTUALIZATION

**Equipment**

-equipment may have

-hit points (each one is like a life, taking all hit points out results in an injury/death

-stat bonuses (augments the protagonist’s current stats, so they are the same ones the protagonist has, I/E, N/S, F/J, P/T)

-moves (to be used in action scenes, see below)

-there are no usable items in the game, only equipment that may be equipped

-regarding the protagonist’s equipment

-armor pieces (head, body, feet)

-cannot be swapped or traded for

-can be upgraded so as to have more hitpoints, different stat bonuses, or different moves

-upgrading may occur as a reward for completing certain arcs

-upgrading may also occur when talking to certain characters

-arm pieces (left arm, right arm)

-weapons are equipped on these body parts and are interchangeable and upgradable

-swapping weapons may occur when

-completing a certain arc where the reward is a weapon

-talking to certain characters who might be willing to give it to you

-during an arc, such as when passing through an armory or something

-upgrading weapons may occur when

-completing a certain arc

-talking to certain characters

**Action Scene Moves**

-moves may come from

-equipped items

-as a reward for completing certain arcs so that they result in a certain ending (example: resolving an arc in favor of a character in it who is a soldier might result in a reward that is a combative move)

-party members (each provides one move at least?)

-general method of a move

1. Calculate difficulty of the move

-uses an algorithm/mathematical function that is unique to the move

-takes into account the current conditions of the action scene

2. Calculate the skill of the protagonist

-uses an algorithm/mathematical function that is unique to the move

-takes into account the current stats of the protagonist including the bonuses given by equipped items

3. Generate random number

4. Subtract skill of protagonist from difficulty of the move and compare number generated from step 3 to the result or add randomly generated number to skill of protagonist and compare to difficulty of move.

5. If the result is greater, then the move was successful. Depending on how much greater one can define different levels of success. If the result was lesser, the move was unsuccessful. Again, depending on how much lesser, one can define different levels of failure.

6. Take appropriate action depending on the level of success or failure of the move. This will involve one, some, or all of the following:

-altering the progress bar

-change conditions of the action scene

-environment conditions

-actor conditions

-change stats of the actor who made the move

-change stats of the opposing actor

-positively modify the success for the next move made by the actor

-this will most likely be implemented by modifying settings for generating the random number in step 3 for the actor’s next move

-negatively modify the success for the next move made by the opposing actor

-as mentioned in the above point, will play with random number settings but in the opposing actor’s next move

-access and display stats of the opposing actor (only for the player)

-access and display conditions of the action scene (only for the player)

7. Output text relating to what happened in step 6. Depending on level of success or failure, the output text will be altered.

-note that any of the results mentioned in step 6 that involve accessing/displaying info is only useful/applicable to the player

-most moves will increase or decrease the progress bar, though how much they increase or decrease the bar is dependent on

-type of action scene (combat or chase) (refer to action scenes below for more info on this)

-level of success or failure

**Action Scenes**

-to start (same for combat or chase)

-set variables

-set conditions

-set environment conditions

-set actor conditions

-prepare player info

-determine hit points and moves

-prepare opponent and opponent info

-create opponent object via baseclass or use an already existing object for frequently used models

-give opponent appropriate items (this will be determined by story)...if using existing object then disregard this

-display blurb about win condition?

-in combat, hitting is primary, movement is secondary...hitting will result in larger changes to the progress bar than movement

-in chase, movement is primary, hitting is secondary...movement will result in larger changes to the progress bar than hitting

**Combat Action Scenes**

-for each turn (player)

-display moves for player to choose from

-wait on input from player (select whichever move)

-run move function (the following indented will be handled by the move function)

-calculate difficulty of the move...difficulty will be uniquely calculated for each move based on the current conditions of the action scene

-check if move is successful or how successful move is...some moves can be more or less successful while others are either successful or failed

-take appropriate action if move is successful...this often means altering the progress bar but it might mean something else like changing stats or for ex, Scan, if done successfully, would display info about the opponent

-output text to describe what happened (we could write this to be a bit random)...think pokemon, “it was not very effective...” but maybe a bit more dynamic to add some flavor by changing the text a bit based on how successful the move was or the current conditions of the scene or both

-output more text if need be, like a summary of what just happened?

-pass turn to opponent

-for each turn (opponent)

-using random number generator, determine move to use

-run move function

-calculate difficulty of the move...difficulty will be uniquely calculated for each move based on the current conditions of the action scene

-check if move is successful or how successful move is...some moves can be more or less successful while others are either successful or failed

-take appropriate action if move is successful...this often means altering the progress bar but might involve temporarily changing stats or something

-output text to describe what happened (we could write this to be a bit random)...think pokemon, “it was not very effective...” but maybe a bit more dynamic to add some flavor by changing the text a bit based on how successful the move was or the current conditions of the scene or both

- output more text if need be, like a summary of what just happened?

-pass turn to player

-win by hitting the opponent until they don’t have anymore hitpoints

-lose if opponent hits you for all your hitpoints

**Chase Action Scenes**

-distance between actors determines the win/lose condition

-moves play out very similarly to combat action scenes

-however, the result of a move will be handled a bit differently...the progress bar will still be altered, but now it will be based more on distance as opposed to hits

-if the actor who is chasing hits the actor who is running, the distance between them is decreased

-if the actor who is running hits the actor who is chasing, the distance between them is increased

-if running away, if distance is greater than certain value, player wins

-if chasing, if distance is less than certain value, player wins

-the value to win is determined by story and hardcoded