Ryan Mann
CS162
Project 3
10 November 2019

DESIGN

- Main
    - Welcome message/menu
    - Prompt character selection
    - Create characters
    - Fight loop
        - P1 atk
        - P2 atk
    - Display victory
    - Delete characters
    - Reprompt menu
- Character
    - Abstract, make everything 0
- Vampire
    - Charm
        - Simple check for charm
- Barbarian
- Blue Men
    - Mob
        - Check for strength levels after each hit
        - Scale down defense dice
- Medusa
    - Glare
        - Damage = 100
        - Kill anything if damage is set to 100
- Harry Potter
    - Hogwarts
        - Upon death check if a charge of hogwarts remains
- Menu

CLASS HIERARCHY

CHARACTER ->

Vampire
Barbarian
Blue Men
Medusa
Harry Potter

REFLECTION

Character.hpp

The first file that I set out to build after designing the basic function ideas in my head (I tend to go on a walk/shower and think about how I will build something before coding). It was a simple abstract class that the rest of the classes would inherit from.

Vampire.cpp/hpp

I actually tackled the vampire first as it was the first one listed and it really is not different than the barbarian. The first major design decision I had to make was what armor meant. I was not sure if it followed like halo rules (shielding) or paper mario rules (flat defense mod). I thought shielding was spicier so I picked it. The rest of the logic is pretty straightforward. The charm functionality was implemented with rand [1,2].

Barbarian.cpp/hpp

I copypasted my vampire code and changed the names/flavour text for this one honestly. It is the same without the charm modifier and then changing the specific values for the barbarian implementation.

Bluemen.cpp/hpp

The blue men was the third tackled. I created a member variable in the class to hold a counter for how many dice are lost and then scale down their defense after each attack lands with a strength range check.

Medusa.cpp/hpp

This one gave me trouble for a while until I realized I forgot to return her damage value in her attack function and that was why the values were off. Otherwise, I implemented glare as an override for her damage function, setting her damage to an instakill if procced.

Harrypotter.cpp/hpp

The hogwarts functionality was implemented with several addendums to check the currently number of hogwarts charges held. If one is found it is consumed and harry is returned to life. This applies any time he would die, so to implement the checks would just be added there.

Main

The main method is similar to my other ones. It is separated into several parts. Initially, the game is initialized and the input is prompted, creating the requisite objects and setting their data. Then a combat loop is played, repeatedly cycling attacks until one party is killed, upon which a victory message is displayed, with input to continue reprompted. I did not have any issues so far with this project, with my main one a simple error of forgetting to return a damage value, causing anything with medusa to behave weirdly. I think my background with game/character design allowed me to already be familiar with simple implementations for combat; I have previously worked on simulation logic for several mmorpg class/job communities.

TEST PLAN

vampire.cpp/hpp

TEST	RESULT
Initializes	TRUE
Attacks	TRUE
Defends	TRUE
Parries	TRUE
Hurts	TRUE
Dies	TRUE
Gets glared	TRUE
Charm	TRUE

barbarian.cpp/hpp

TEST	RESULT
Initializes	TRUE
Attacks	TRUE
Defends	TRUE
Parries	TRUE
Hurts	TRUE
Dies	TRUE
Gets glared	TRUE

bluemen.cpp/hpp

TEST	RESULT
Initializes	TRUE
Attacks	TRUE
Defends	TRUE
Parries	TRUE
Hurts	TRUE
Dies	TRUE
Gets glared	TRUE
Mob	TRUE

medusa.cpp/hpp

TEST	RESULT
Initializes	TRUE
Attacks	TRUE
Defends	TRUE
Parries	TRUE
Hurts	TRUE
Dies	TRUE
Gets glared	TRUE
Glare	TRUE

harrypotter.cpp/hpp

TEST	RESULT
Initializes	TRUE
Attacks	TRUE
Defends	TRUE
Parries	TRUE
Hurts	TRUE
Dies	TRUE
Gets glared	TRUE
Hogwarts glare	TRUE
Hogwarts standard rez	TRUE

inputValidation.cpp/hpp / validate.cpp/hpp

TEST	RESULT
Validates and returns	TRUE

combatMenu.cpp/hpp

TEST	RESULT
Displays	TRUE
Gets choice	TRUE

Main

TEST	**RESULT**
Random seed initializes	TRUE
Game plays	TRUE
Character select displays	TRUE
P1 character gets	TRUE
P1 character creates	TRUE
P2 character gets	TRUE
P2 character creates	TRUE
Combat prep messages display	TRUE
Combat runs	TRUE
P1 victory displays	TRUE
P2 victory displays	TRUE
Characters delete	TRUE
Menu reprompted	TRUE
Exits properly	TRUE