Project Presentation – Simple RPG style battle game

Introduction

This project will focus on a simple RPG style battle game. The game has two sies, the player and the computer. The player will act as the brave who adventure deep in the forest to battle strong monster The game allows the player to give all sorts of commands during the battle and after the battle, there might be additional features added, such as the inventory system or an accessory system that allows the player to change weapons or accessories based on their own likings.

Project Design

Assessment Concepts

Memory allocation from stack and the heap

- Arrays: we will use dynamic array to increase the inventory size and static array for displaying the inventory.
- Strings: Strings will be used for the name of the weapons and the name of the monsters.
- Objects: Player, Inventory, Monster.
- Integers: The item id of an item in the inventory.

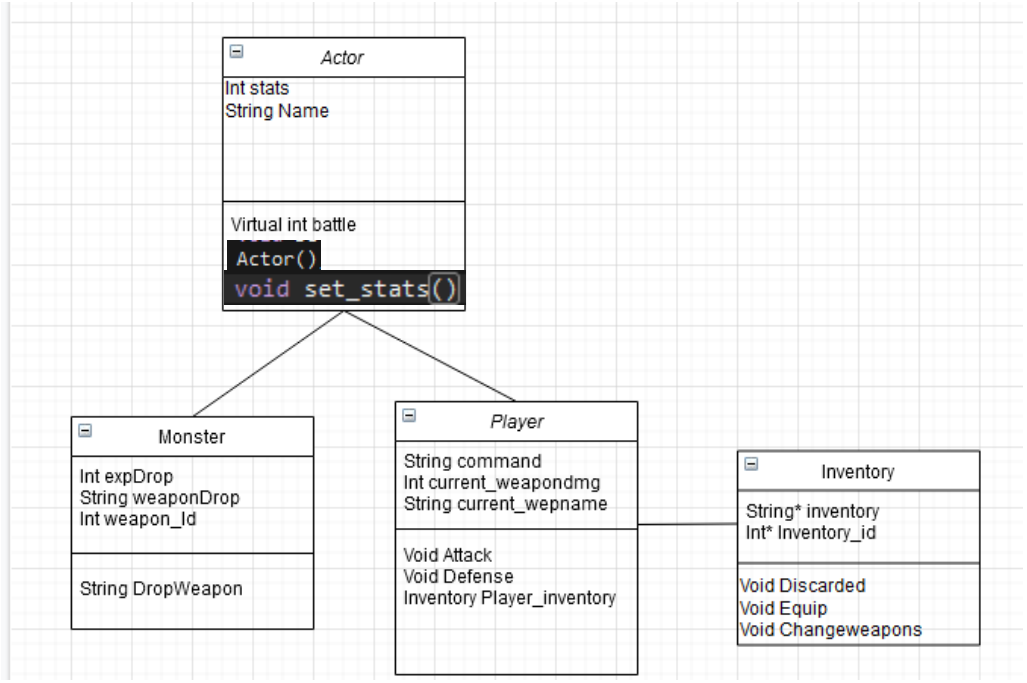
User input and Output

- Mostly string commands such as , Attack, Defense, which performs different functions. This also includes the commands for the inventory, such as replacing a weapon when the inventory is full or changing your current equipment.

Object-oriented programming and design

- Inheritance: The Player and Monster will be children classes from the parent class Actor. So that the player and monster class can access projected integers or strings.
- Abstract Classes: The Actor class will have a pure virtual functions or objects such as battle and damage. The Player and Monster class will use the function and the objects when player input a command.

Class Diagram



Class Descriptions

Actor

Actor is an abstract base class, it contains the stats integer array for the basic stats of the Player and Monster class, which are Hit Points, Defense, Attack etc.. and the string Name for Player and Monster class. The function battle will be pure virtual and returns the stats of the player after a single battle. The function set_stats will set the stats and name for the stats array and the Actor(name) function will create a Actor variable with an empty stats array.

Monster

Monster is the children class of Actor, it holds the exp, weapon name and weapon id that it will drop after it has been defeated.

Player

Player is the children class of Actor, it holds the command that the user inputs and performs various functions, such as attack, defense, etc. The Player class also contains an Inventory* type function, which allows the player to access function from the inventory class.

Inventory

The inventory class allows the player to access the inventory to store weapons, or to change the current equipped weapon from a command input.

User interface

Users of this program will be the Players who battle monster. They will receive command-line inputs to promote the use commands of available commands as inputs and receive command-line outputs.

Code style

All code will be indented with one Tab.

Class will begin with a capital letter.
Additonally, the code will all have “using namespace std;” at the start of the header files and the main files.
Comments will be added at the start of each functions including functions within the classes. The comment will include the input variable and the return variable of the function, and a short description of the function will also be included.

Testing Plan

The tests will be run through makefile with inputs automatically. Additionally, we will test our program when a new function or class has been added.

Unit testing

The unit test will cover all function in the classes, each class will have a main file with hard coded inputs and tests for all the function in that class. The user inputs will be inputted via makefiles so that we can test for all functions for our programs.

Combined testing

In the last stages of development of our program, we will test the classes combined. The Player class which will have functions that access the Inventory class will be tested together first. Then the Player, Monster and Actor class will be tested together. At last, we will run our final test with the Player, Monster, Actor and Inventory class combined.

Schedule Plan

Goals

The goal is to complete the main function from week 9 so that there will be additional time for testing and running in week 11, then in week 10 we will add more features such as an inventory system and a monster drop system.

Break week 2

Forming the plan and the outline of this project

Week 9

Work on the Actor, Player and Monster Classes.

Student id – Name	Allocated work	Work deadline
Week 10		
a1832770 – Yuxin Cao	Finish the Player class and work on a monster data base. Additionally integrate the monster class into other class and add additional functions for testing.	Week 10 Monday Week 10 Thursday
a1802893 – Ruijie Fan	Start working on the monster class. And tests the classes once ready.	Week 10 Monday
a1776727 – Hangyu Chen	Start working on the inventory class. And tests the classes once ready.	Week 10 Friday
Week 11		
a1832770 – Yuxin Cao	Add the inventory class into the classes. Test the inventory class's function when integrated into other classes	Week 11 Monday Week 11 Monday
a1802893 – Ruijie Fan	Test all the class functions, and report for any bugs.	Week 11 Thursday
a1776727 – Hangyu Chen	Add addition feature to the inventory class if required.	Week 11 Thursday