Project2

<The Gladiator>

Name: Shisheng Liu

Professor: Mark Lehr

Class: CIS 17A-48096

Contents

1. Introduction…………………………………………...3
2. Project Summary...........................................................3
3. Variable types include..................................................4
4. Problems during coding...............................................4
5. Pseudo Code ...............................................................5
6. System Libraries.........................................................5
7. Function List...............................................................6
8. Sample I/O and Game running details......................7,8
9. Flowchart......................................................................9,11
10. Doxygen........................................................................10
11. Code..............................................................................12

**1.Introduction**

1.1 About the Game

I originally intended to do a game like "Tower of the Sorcerer", but because of limited time, and technical problems, can only do a relatively simple 1V1 battle game. At the start of the game you will be able to customize your gladiator by choosing the weapon, armor and fighting style that fits your preferred play type. Your opponents, as heroes of the empire are all pre-generated. Combat is dealt in simultaneous action rounds until one combatant is reduced below five hit points. After combat is resolved, the player (if alive) will be allowed to advance or quit. There is no save feature at this time.

1.2 How to Play

As previously mentioned, the game opens with character creation. Naming your

Character is just the start. Choosing your weapons and armor, and selecting a fighting style have real consequences to how combat is played out. At the end of character creation, you will be given the option to start over, or enter the arena. Once combat begins, the gladiators are announced, salute the emperor and chose their actions. Turns are taken simultaneously allowing for damage mitigation from blocks and dodges, and counter attacks through parries. There are also spots of inaction when both players chose a not attacking action. Combat is decided when one of the gladiators is taken below five health. At five health, the player is presented with the option to surrender, accepting the emperor’s mercy. The opponent simply loses. There are five total opponents of increasing difficulty. After each opponent is defeated, the player is given a choice to advance to the next or quit the game. At the end of combat, victory is declared for the winning side.

**2.Project Summary:**

Coded Lines: 1378

Functions in Main: 18

Supporting CPP files: 3

Class files: 4

Templates: 4

**3.Variable types include:**

Void, Class, fstream, Template and basic int, short, float, char and string data types. The project also includes text files for reading about the game and how to play.

**4.Problems during coding：**

At the beginning, I was considered to update my previous project (from Project 1), However, I think if I try to add some stuff in it will more difficult than re do a new one. Like a said in "Introduce", I was plan create a childhood game call "Tower of the Sorcerer", but I just realize it would be a huge work, so I decided building a text based RPG for the final project. Deciding that a text based game was going to be too text heavy, I opted for a combination RPG combat simulator. During the initial design phase, I had created a system that became very cluttered and bloated with too many variables and options to work through in combat. During the development cycle, some initial options I had wanted to include. One of these was to pull the information for the game’s “How To” section in to a structure and read it out. Similar to Project 1, I was unable to successfully store the file input into a structure for output and reverted back to using text files. Another value I had hoped to include a “favor” system that provided different outcomes depending on how well a player did in combat. The combat simulation was also going to originally be more robust, with a system for counters, special attacks and health recovery. While I did not experience significant difficulty with the assignment, there were some specific implementations that were more difficult than others. Overwriting the base variables proved problematic until correctly using polymorphism. Even though it was one of my favorite parts, balancing the game play and getting the variables to adjust properly with the chosen player variables was also a bit time consuming. The end game is functional, but there are still some “easy mode” discrepancies due to it not being fully balanced.

**5. Pseudo Code**

Introduce the game which is a simple1V1 RGP game, player act as Gladiator

Enter the game, pick up a weapon, armor, and choose your fighting style.

A battle begin

When in player’s round, selected a way to attack or choose defense.

Attack CpuPlayer

CpuPlayer will get hurt or It decided defense your attack.

If CpuPlay did not died, turn to its round and it will choose a way to attack Player

Player will accept attack or defense the following attack

If Player did jot died, turn to Player’s round and Player choose its attack way

Else choose a way to defense

If no any player died, go on the game

Else end the game and declared the winner

Output the battle stats

**6. System Libraries**

**#include <iostream>**

**#include <fstream>**

**#include <cstdlib>**

**#include <string>**

**7. Function List**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Type** | **Name** | **Argument** | **Function** | **Location** |
| **String** | **To running** | **int** | **Get the game running successful** | **156** |
| **Void** | **introduce** | **None** | **Introduce the game** | **46** |
| **Void** | **ask** | **Char&,int&** | **Prompt player for information** | **247** |
| **char** | **check** | **Char,const,char[],int** | **check whether stats and infor are**  **correct** | **285** |
| **bool** | **index** | **Char,const,char[],int** | **return whether the game is in the end** | **316** |
| **void** | **replace** | **String&,char,int** | **replace of the correct option** | **124** |
| **bool** | **inside** | **Const vector<int>,int** | **return whether this game is finished** | **224** |
| **void** | **sample** | **None** | **display the sample of game** | **870** |

**8.Sample I/O and Game running details.**

The Game

At load, the user is presented with the title screen and prompted to enter to continue.

After entering, the player is provided a menu to play, read about the game and read the how to.

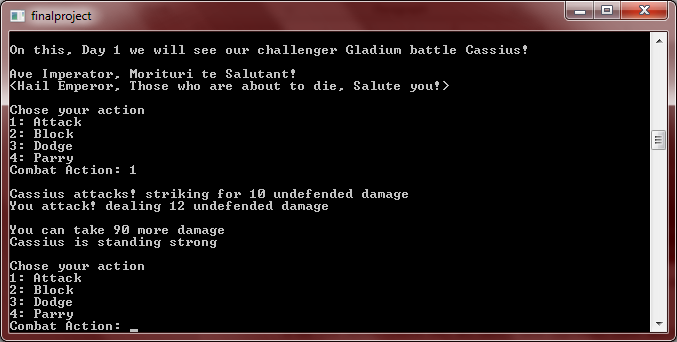
Playing the game begins with entering the chosen name of the gladiator, followed by your preferred weapon, armor and fighting style. After choices are made, the player is presented with their character and given the option to remake them or enter the coliseum.



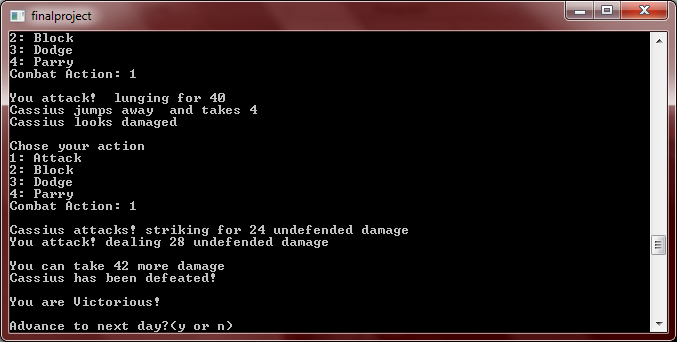
The gladiator and combat stats are derived from the player choices and displayed as a check for the user to gauge how they should play the character they created.

If the user enters combat they are shown their opponents name and enter combat after a brief salute to the emperor. Stats for the opponent are not shown.

Users chose their combat action at the same time that the computer opponent does, providing simultaneous combat.

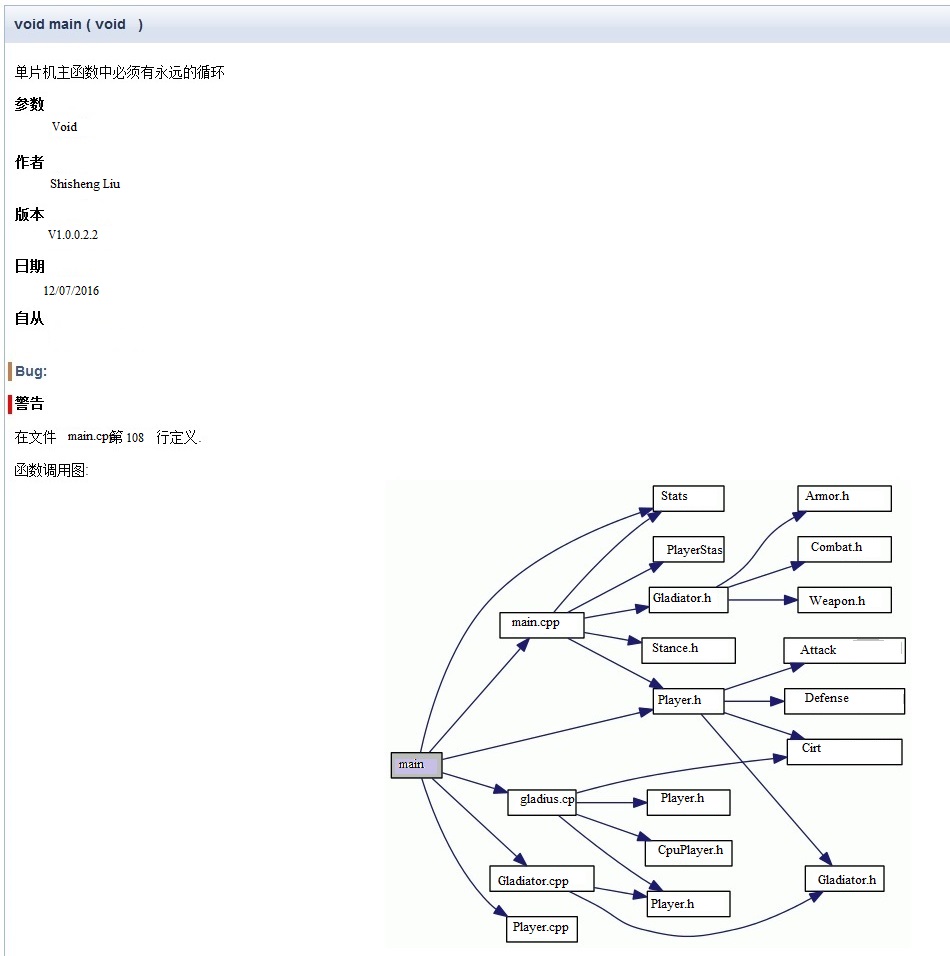


Action resolution is shown for both combatants is shown, with damage being dealt. The opponent’s health is not shown, but his condition is displayed when he takes damage. At the end of combat, the user is prompted to advance to the next day, or quit the game.



**9.Flowchart**

**10.Doxygen**

****

Flowchart(2)



**11.Code**

/\*

\* File: main.cpp

\* Author: Shisheng Liu

\* Created on 12/01/2016, 16:46 PM

\* Purpose: Project2---The Gladiator

\*/

//\_\_\_\_\_\_\_\_\_\_\_.\_\_ \_\_\_\_\_\_\_\_.\_\_ .\_\_\_.\_\_ \_\_

//\\_\_ \_\_\_/| |\_\_ \_\_\_\_ / \_\_\_\_\_/| | \_\_\_\_\_ \_\_| \_/|\_\_|\_\_\_\_ \_/ |\_ \_\_\_\_\_\_\_\_\_\_\_

// | | | | \\_/ \_\_ \ / \ \_\_\_| | \\_\_ \ / \_\_ | | \\_\_ \\ \_\_\/ \_ \\_ \_\_ \

// | | | Y \ \_\_\_/ \ \\_\ \ |\_\_/ \_\_ \\_/ /\_/ | | |/ \_\_ \| | ( <\_> ) | \/

// |\_\_\_\_| |\_\_\_| /\\_\_\_ > \\_\_\_\_\_\_ /\_\_\_\_(\_\_\_\_ /\\_\_\_\_ | |\_\_(\_\_\_\_ /\_\_| \\_\_\_\_/|\_\_|

// \/ \/ \/ \/ \/ \/

//Functional Includes

#include <iostream>

#include <fstream>

#include <cstdlib>

//Class Includes

#include "Player.h"

#include "HowToPlay.h"

#include "Combat.h"

using namespace std;

//Global Constants

//Function Prototypes

void header();

void mainMenu();

void playGame();

void about();

void howToPlay();

Player mkStats();

int getWepChoice();

int \*getWeaponStats(short);

int getArmrChoice();

int \*getArmrStats(short);

int getStyleChoice();

int \*getStyleStats(short);

int \*getOpp(int);

string getOppName(int);

char combat(string, int\*, string, int \*, int);

int getCpuAction();

int \*getCmbtActs(int, int);

//Start the Game!

int main() {

//Welcome Menu

header();

cout << endl << "Welcome to The wild Gladiator";

cout << endl << endl << "Press Enter to continue the game.";

cin.ignore();

cout << endl;

mainMenu();

return 0;

}

void header() {

string title; //Show The title mainMenu

fstream titleFile; //Gets tile mainMenu

// Open the file in input mode.

titleFile.open("title.txt", ios::in);

// If the file was successfully opened, continue.

if (titleFile) {

// Read an item from the file.

getline(titleFile, title);

// Read in file.

while (titleFile) {

// Display the last item read.

cout << title << endl;

// Read the next item.

getline(titleFile, title);

}

// Close the file.

titleFile.close();

} else {

cout << "ERROR: Cannot open file.\n";

}

}

void mainMenu() {

short main;

cout << endl << "Main Menu\n"

<< "----------------------\n"

<< "1: Play The Gladius\n"

<< "2: About The Gladius\n"

<< "3: How To Play The Gladius\n"

<< "4: Exit The Gladius\n"

<< "----------------------";

cout << endl << "Menu Selection: ";

cin >> main;

while (main < 1 || main > 4) {

cout << endl << "ERROR: please re-enter";

}

if (main == 1) {

playGame();

} else if (main == 2) {

about();

} else if (main == 3) {

howToPlay();

}

}

void playGame() {

char begin;

int round = 1; //Sets game's combat rounds

int MAXROUNDS = 5; //Total number of rounds

string name;

int statsSize = 4;

int \*plyrStats = new int[statsSize];

Player stats;

//Input your choices

stats = mkStats();

//Display your choices

stats.prntPlyrStrgs();

stats.prntPlyrStats();

name = stats.getPlryName();

plyrStats[0] = stats.getHealth();

plyrStats[1] = stats.getPlryAtkP();

plyrStats[3] = stats.getPlryDdge();

plyrStats[4] = stats.getPlryPrry();

cout << endl;

cout << "Would you like enjoy the fighting with this Gladiator?(y or n)";

cin >> begin;

if (begin == 'y') {

cout << endl << "This is your destiny!Your combat stats are: "

<< "Total Health: " << plyrStats[0] << "\t"

<< "Total Attack Power: " << plyrStats[1] << "\t"

<< "Total Block Rating: " << plyrStats[2] << "\t"

<< "Total Dodge Rating: " << plyrStats[3] << "\t"

<< "Total Parry Rating: " << plyrStats[4] << endl;

} else {

mkStats();

}

cin.ignore();

char play;

for (int i = 0; i < MAXROUNDS; i++) {

int \*opponent;

string oppName;

opponent = getOpp(round);

oppName = getOppName(round);

play = combat(name, plyrStats, oppName, opponent, round);

cout << endl << "Your answer is: " << play << endl;

if (play == 'y') {

round++;

}

if (play == 'n') {

cout << endl << "Thank you for playing The Gladiator!";

cout << endl;

exit(1);

}

}

}

void about() {

cout << endl;

string title; //Shows in the title mainMenu

fstream titleFile; //Gets the title mainMenu

// Open the file in input mode.

titleFile.open("about.txt", ios::in);

// If the file was successfully opened, continue.

if (titleFile) {

// Read an item from the file.

getline(titleFile, title);

// Read in file.

while (titleFile) {

// Display the last item read.

cout << title << endl;

// Read the next item.

getline(titleFile, title);

}

// Close the file.

titleFile.close();

} else {

cout << "ERROR: The file cannot be open.\n";

}

mainMenu();

}

void howToPlay() {

short menu;

HowToPlay read; //Read in to structure

const short NUM\_FILES = 4;

fstream play; //Stream Reader

cout << endl << "Opening file to read" << endl;

// Open a file for input in binary mode.

play.open("howtoplay.dat", ios::in | ios::binary);

//test for errors

if (!play) {

cout << "Error in opening file. Program aborting.\n";

}

//Read in first record

play.read(reinterpret\_cast<char \*> (&read),

sizeof (read));

//Loop for end of file

while (!play.eof()) {

//Load all other records

play.read(reinterpret\_cast<char \*> (&read),

sizeof (read));

}

//Close file.

play.close();

//HowTo Menu

cout << endl << "How To Menu\n"

<< "--------------------\n"

<< "1: Playing The Gladiator\n"

<< "2: Gladiator's Stats\n"

<< "3: Go Combat\n"

<< "4: Weapons and Armor list\n"

<< "5: Return to Main Menu\n"

<< "--------------------";

cout << endl << "How To: ";

cin >> menu;

//Bounds check

while (menu < 1 || menu > 5) {

cout << endl << cout << endl << "ERROR: please re-enter";

}

if (menu == 1) {

cout << read.gamePlay;

}

if (menu == 2) {

cout << read.playerStats;

}

if (menu == 3) {

cout << read.combat;

}

if (menu == 4) {

cout << read.wpnsArmr;

}

if (menu == 5) {

mainMenu();

}

howToPlay();

}

Player mkStats() {

//Getter

short getPlyrWep;

int \*wepStats;

short getPlyrArmr;

int \*armrStats;

short getPlyrStyl;

int \*styleStats;

//Variables

string name; //get player name

string style;

string weapon; //get player weapon

string armor; //get player armor

//Adjustments

short plyrHlth; //adjusted health

short plyrFati; //adjusted fatigue

short plyrStr; //adjusted strength

short plyrAgi; //adjusted agility

short plyrDef; //adjusted defense

short plyrMvSp; //adjusted movement

//Get player name

cout << "What is your name, My brave Gladiator! ";

cin >> name;

cin.ignore();

cout << endl;

//Get player weapon

getPlyrWep = getWepChoice();

wepStats = getWeaponStats(getPlyrWep);

if (getPlyrWep == 1) {

weapon = "Common Short Sword";

} else if (getPlyrWep == 2) {

weapon = "Common One-Handed Mace";

} else if (getPlyrWep == 3) {

weapon = "Common One-Handed Hammer";

} else if (getPlyrWep == 4) {

weapon = "Common One-Handed Flail";

} else if (getPlyrWep == 5) {

weapon = "Common Two-Handed Long Sword";

} else if (getPlyrWep == 6) {

weapon = "Common Two-Handed Hammer";

} else if (getPlyrWep == 7) {

weapon = "Common Two Handed Trident";

} else if (getPlyrWep == 8) {

weapon = "Common Two-Hand Short Swords";

}

cout << endl;

//Get player armor

getPlyrArmr = getArmrChoice();

armrStats = getArmrStats(getPlyrArmr);

if (getPlyrArmr == 1) {

armor = "Common Half Leather";

} else if (getPlyrArmr == 2) {

armor = "Common Full Leather";

} else if (getPlyrArmr == 3) {

armor = "Common Broze Armor";

} else if (getPlyrArmr == 4) {

armor = "Common Silver Armor";

} else if (getPlyrArmr == 5) {

armor = "Common Golden Armor";

} else if (getPlyrArmr == 6) {

armor = "Common Legency Armor";

}

cout << endl;

//Style menu

getPlyrStyl = getStyleChoice();

styleStats = getStyleStats(getPlyrStyl);

//Style, Health, Strength, Fatigue, Agility, Defense, Movement

if (getPlyrStyl == 1) {

style = "Agile";

} else if (getPlyrStyl == 2) {

style = "Tactful";

} else if (getPlyrStyl == 3) {

style = "Brute";

} else if (getPlyrStyl == 4) {

style = "Tank";

}

}

int getWepChoice() {

int plyrWep; //holds weapon choice

cout << "Pick a dream of the arms of it " << endl;

cout << "1: Comman One-Handed Short Sword\n"

<< "2: Comman One-Handed Mace\n"

<< "3: Comman One-Handed Hammer\n"

<< "4: Comman One-Handed Flail\n"

<< "5: Comman Two-Handed Long Sword\n"

<< "6: Comman Two-Handed Hammer\n"

<< "7: Comman Two Handed Trident\n"

<< "8: Comman Two Short Swords\n"

<< "Weapon Choice Menu: ";

cin >> plyrWep;

//Return weapon

return plyrWep;

}

int \*getWeaponStats(short choice) {

int size = 6;

int \*wepStats = new int[size];

short wpDmg; //Weapon' Damage

short wpSpd; //Weapon' Attack Speed

short wpRch; //Weapon'S Reach

short wpDef; //Defense Adjustment from Weapon

short wpAgi; //Agility Adjustment from Weapon

short wpFti; //Fatigue Adjustment from Weapon

//type, damage, speed, reach, defense adj, agility adj

if (choice == 1) {

wpDmg = 5;

wpSpd = 4;

wpRch = 2;

wpDef = 0;

wpAgi = 0;

wpFti = 0;

} else if (choice == 2) {

wpDmg = 7;

wpSpd = 3;

wpRch = 1;

wpDef = 0;

wpAgi = 0;

wpFti = 0;

} else if (choice == 3) {

wpDmg = 6;

wpSpd = 3;

wpRch = 1;

wpDef = 0;

wpAgi = 0;

wpFti = 0;

} else if (choice == 4) {

wpDmg = 8;

wpSpd = 2;

wpRch = 2;

wpDef = 0;

wpAgi = 0;

wpFti = 1;

} else if (choice == 5) {

wpDmg = 3;

wpSpd = 5;

wpRch = 3;

wpDef = 2;

wpAgi = 2;

wpFti = 1;

} else if (choice == 6) {

wpDmg = 12;

wpSpd = 2;

wpRch = 3;

wpDef = 0;

wpAgi = 3;

wpFti = 2;

} else if (choice == 7) {

wpDmg = 10;

wpSpd = 3;

wpRch = 1;

wpDef = 2;

wpAgi = 3;

wpFti = 1;

} else if (choice == 8) {

wpDmg = 5;

wpSpd = 8;

wpRch = 1;

wpDef = 1;

wpAgi = 1;

wpFti = 1;

}

//set values

wepStats[0] = wpDmg;

wepStats[1] = wpSpd;

wepStats[2] = wpRch;

wepStats[3] = wpDef;

wepStats[4] = wpAgi;

wepStats[5] = wpFti;

//Return

return wepStats;

}

int getArmrChoice() {

int plyrArmr;

cout << "You gonna get a fit Aromor, Right? " << endl;

cout << "1: Conman Half Leather\n"

<< "2: Coman Leather\n"

<< "3: Common Broze Armor\n"

<< "4: Common Sliver Armor\n"

<< "5: Common Golden Armor\n"

<< "6: Common Legency Armor\n"

<< "Armor Choice: ";

cin >> plyrArmr;

//Return armor choice

return plyrArmr;

}

int \*getArmrStats(short choice) {

int size = 4;

int \*armrStats = new int[size];

short arDef; //Defense Adjustment from Armor

short arAgi; //Agility Adjustment from Armor

short arFti; //Fatigue Adjustment from Armor

short arMv; //Movement Adjustment from Armor

//type, defense adj, agility adj

if (choice == 1) {

arDef = 3;

arAgi = 2;

arFti = 1;

arMv = 1;

} else if (choice == 2) {

arDef = 5;

arAgi = 2;

arFti = 1;

arMv = 1;

} else if (choice == 3) {

arDef = 6;

arAgi = 2;

arFti = 0;

arMv = 1;

} else if (choice == 4) {

arDef = 3;

arAgi = 4;

arFti = 1;

arMv = 4;

} else if (choice == 5) {

arDef = 4;

arAgi = 3;

arFti = 2;

arMv = 4;

} else if (choice == 6) {

arDef = 6;

arAgi = 5;

arFti = 3;

arMv = 3;

}

//Set values

armrStats[0] = arDef;

armrStats[1] = arAgi;

armrStats[2] = arFti;

armrStats[3] = arMv;

//Return

return armrStats;

}

int getStyleChoice() {

int style;

cout << "What Kind of Gladiator(Fighting style) you are? Hahahahahaha!!!: " << endl;

cout << "1: Assasin\n"

<< "2: Hercules\n"

<< "3: Knight\n"

<< "4: Barbarian\n"

<< "Fighting Style: ";

cin >> style;

//Return style

return style;

}

int \*getStyleStats(short choice) {

int size = 6;

int \*styleStats = new int[size];

short hlthAdj; //Value to Adjust Base Health

short fatiAdj; //Value to Adjust Base Fatigue

short strAdj; //Value to Adjust Base Strength

short agiAdj; //Value to Adjust Base Agility

short defAdj; //Value to Adjust Base Defense

short moveAdj; //Value to Adjust base movement

//style, health, strength, fatigue, agility, defense, movement

if (choice == 1) {

hlthAdj = 0;

fatiAdj = 17;

strAdj = 10;

agiAdj = 14;

defAdj = 11;

moveAdj = 10;

} else if (choice == 2) {

hlthAdj = 0;

fatiAdj = 13;

strAdj = 10;

agiAdj = 14;

defAdj = 12;

moveAdj = 11;

} else if (choice == 3) {

hlthAdj = 11;

fatiAdj = 11;

strAdj = 10;

agiAdj = 15;

defAdj = 10;

moveAdj = 10;

} else if (choice == 4) {

hlthAdj = 20;

fatiAdj = 10;

strAdj = 8;

agiAdj = 11;

defAdj = 18;

moveAdj = 10;

}

//set values

styleStats[0] = hlthAdj;

styleStats[1] = fatiAdj;

styleStats[2] = strAdj;

styleStats[3] = agiAdj;

styleStats[4] = defAdj;

styleStats[5] = moveAdj;

//return

return styleStats;

}

int \*getOpp(int round) {

int size = 5;

int \*opponent = new int[size];

string oppName;

//opponent[0] = health, opponent[1] = attack power, opponent[2] = block rating, opponent[3] = dodge rating, opponent[4] = parry rating

if (round == 1) {

opponent[0] = 100;

opponent[1] = 45;

opponent[2] = 30;

opponent[3] = 30;

opponent[4] = 35;

} else if (round == 2) {

opponent[0] = 90;

opponent[1] = 30;

opponent[2] = 40;

opponent[3] = 30;

opponent[4] = 35;

} else if (round == 3) {

opponent[0] = 100;

opponent[1] = 30;

opponent[2] = 30;

opponent[3] = 55;

opponent[4] = 30;

} else if (round == 4) {

opponent[0] = 95;

opponent[1] = 35;

opponent[2] = 35;

opponent[3] = 50;

opponent[4] = 30;

} else if (round == 5) {

opponent[0] = 125;

opponent[1] = 30;

opponent[2] = 25;

opponent[3] = 35;

opponent[4] = 30;

}

//return opponent

return opponent;

}

string getOppName(int round) {

string name;

if (round == 1) {

name = "Assasin";

} else if (round == 2) {

name = "Hercules";

} else if (round == 3) {

name = "Knight";

} else if (round == 4) {

name = "Barbarian";

} else if (round == 5) {

name = "Conquistador";

}

//Return

return name;

}

char combat(string plyrName, int \*plyrStats, string oppName, int \*oppStats, int round) {

char advance;

char wounded;

int cmbtAct;

int cpuCombat;

int plyrHealth = plyrStats[0];

int oppHealth = oppStats[0];

cout << endl << "Welcome to the Hell of the Gladiator!" << endl;

cout << endl << "On Day " << round << " Let us see ourrrrrrr Challenger!!! " << plyrName <<

" battle " << oppName << "!";

cout << endl << "The Highest Gladiator!" << endl;

//[0] = health, [1] = attack power, [2] = block rating, [3] = dodge rating, [4] = parry rating

cout << endl << "Player Stats: " << endl

<< "Total Health: " << plyrHealth << "\t"

<< "Total Attack Power: " << plyrStats[1] << "\t"

<< "Total Block Rating: " << plyrStats[2] << "\t"

<< "Total Dodge Rating: " << plyrStats[3] << "\t"

<< "Total Parry Rating: " << plyrStats[4] << endl;

cout << endl;

cout << endl << "Opponent Stats: " << endl

<< "Total Health: " << oppHealth << "\t"

<< "Total Attack Power: " << oppStats[1] << "\t"

<< "Total Block Rating: " << oppStats[2] << "\t"

<< "Total Dodge Rating: " << oppStats[3] << "\t"

<< "Total Parry Rating: " << oppStats[4] << endl;

cout << endl << "Press enter to continue";

cin.ignore();

//Show the Combat Loop

while (oppHealth >= 5) {

//Reset all variables fir the start of combat rounds

int plyrAtk = 0;

int plyrDmg = 0;

int plyrBlk = 0;

int plyrDdge = 0;

int plyrPrry = 0;

int oppAtk = 0;

int oppDmg = 0;

int oppBlk = 0;

int oppDdge = 0;

int oppPrry = 0;

//Get CPU Action

cpuCombat = getCpuAction();

//Player combat menu

cout << endl << "Choose your combat action\n"

<< "1: Attack\n"

<< "2: Block\n"

<< "3: Dodge\n"

<< "4: Crit\n"

<< "Combat Action: ";

cin >> cmbtAct;

//Both attack

if (cpuCombat == 1 && cmbtAct == 1) {

cout << endl;

//Opponent attack

cout << oppName << " attacks ";

oppAtk = attack(oppStats[1], plyrStats[2]);

if (oppAtk <= 0) {

cout << "but misses";

oppDmg = 0;

} else if (oppAtk >= 1) {

cout << oppName << " Unmiss ";

oppDmg = getDamage(oppStats[1], plyrStats[2], oppAtk);

cout << oppDmg << " No Damage";

}

//Player attack

plyrAtk = attack(plyrStats[1], oppStats[2]);

if (plyrAtk <= 0) {

cout << "You attack, but miss";

plyrDmg = 0;

} else if (plyrAtk >= 1) {

plyrDmg = getDamage(plyrStats[1], oppStats[2], plyrAtk);

cout << "You attack, Unmiss " << plyrDmg << " No Damage";

}

//Show Damage

plyrHealth =plyrHealth - oppDmg;

oppHealth = oppHealth - plyrDmg;

//Show current health

cout << endl << "Player Health: " << plyrHealth;

cout << endl << "Opp Health: " << oppHealth;

}

//Both block

if (cpuCombat == 2 && cmbtAct == 2) {

//No combat result

cout << endl;

cout << oppName << " Takes a Defensive position";

cout << endl;

cout << "Attack Coming !";

}

//Both dodge

if (cpuCombat == 3 && cmbtAct == 3) {

//No combat result

cout << endl;

cout << oppName << " Slides away from you";

cout << endl;

cout << "Good job! You slide away from an attack";

}

//Both crit

if (cpuCombat == 4 && cmbtAct == 4) {

//No combat result

cout << endl;

cout << oppName << " Looks for an heavey attack";

cout << endl;

cout << "You are trying give a heavey attack!";

}

//Opponent attack - player block

if (cpuCombat == 1 && cmbtAct == 2) {

cout << endl;

//Opponent attack

cout << oppName << " attacks " << endl;

oppAtk = attack(oppStats[1], plyrStats[2]);

if (oppAtk = 0) {

cout << "but misses";

oppDmg = 0;

} else if (oppAtk >= 1) {

cout << oppName << " strikes for ";

oppDmg = getDamage(oppStats[1], plyrStats[2], oppAtk);

cout << oppDmg << " No damage";

}

//Player block versus attack

plyrBlk = getBlock(oppDmg, plyrStats[2]);

cout << "You blocked ";

if (plyrBlk > oppDmg) {

cout << "all ";

oppDmg = 0;

} else {

cout << plyrBlk;

oppDmg = oppDmg - plyrBlk;

}

cout << " Out of the damage, taking " << oppDmg;

plyrHealth = plyrHealth - oppDmg;

cout << endl << "Player Health: " << plyrHealth;

cout << endl << "Opp Health: " << oppHealth;

}

//Opponent attack - player dodge

if (cpuCombat == 1 && cmbtAct == 3) {

cout << endl;

//Opponent attack

cout << oppName << " attacks " << endl;

oppAtk = attack(oppStats[1], plyrStats[2]);

if (oppAtk = 0) {

cout << "but misses";

oppDmg = 0;

} else if (oppAtk >= 1) {

cout << oppName << " strikes for ";

oppDmg = getDamage(oppStats[1], plyrStats[2], oppAtk);

cout << oppDmg << " No Damage";

}

//Player dodge against attack

plyrDdge = getDodge(oppDmg, plyrStats[3]);

cout << endl;

cout << "You dodged ";

if (plyrDdge > oppAtk) {

cout << "all ";

oppDmg = 0;

} else {

cout << plyrDdge;

oppDmg = oppDmg - plyrDdge;

}

cout << " Out of the damage, taking" << oppDmg;

plyrHealth = plyrHealth - oppDmg;

cout << endl << "Player Health: " << plyrHealth;

cout << endl << "Opp Health: " << oppHealth;

}

//Opponent attacks - player paries

if (cpuCombat == 1 && cmbtAct == 4) {

cout << endl;

//Opponent attack

cout << oppName << " attacks " << endl;

oppAtk = attack(oppStats[1], plyrStats[2]);

if (oppAtk = 0) {

cout << "but misses";

oppDmg = 0;

} else if (oppAtk >= 1) {

cout << oppName << " strikes for ";

oppDmg = getDamage(oppStats[1], plyrStats[2], oppAtk);

cout << oppDmg << " No damage";

}

oppDmg = attack(oppStats[1], plyrStats[2]);

//Player crit against attack

plyrPrry = getCirt(oppDmg, plyrStats[4]);

if (plyrPrry > oppDmg) {

cout << oppName << " attacks " << endl;

cout << "You crit and counter for ";

int counter = 7 / 100;

plyrDmg = getDamage(plyrStats[1], oppStats[2], counter);

oppHealth = oppHealth - plyrDmg;

cout << plyrDmg;

} else {

cout << oppName << " attacks " << endl;

cout << "You crit, taking no damage but are unable to counter";

}

cout << endl << "Player Health: " << plyrHealth;

cout << endl << "Opp Health: " << oppHealth;

}

//Opponent blocks - player attacks

if (cpuCombat == 2 && cmbtAct == 1) {

cout << endl;

//Player attack

cout << "You attack " << endl;

plyrAtk = attack(plyrStats[1], oppStats[2]);

if (plyrAtk = 0) {

cout << "but miss";

plyrDmg = 0;

} else if (plyrAtk >= 1) {

cout << "striking for ";

plyrDmg = getDamage(plyrStats[1], oppStats[2], plyrAtk);

cout << plyrDmg << " No damage";

}

//opponent dodge against attack

oppBlk = getBlock(plyrAtk, oppStats[2]);

cout << oppName << " blocked ";

if (oppBlk > plyrAtk) {

cout << "all ";

plyrDmg = 0;

} else {

cout << oppBlk;

plyrDmg = plyrDmg - oppBlk;

}

cout << " Out of the damage, taking" << plyrDmg;

oppHealth = oppHealth - plyrDmg;

cout << endl << "Player Health: " << plyrHealth;

cout << endl << "Opp Health: " << oppHealth;

}

//Opponent blocks - player dodges

if (cpuCombat == 2 && cmbtAct == 3) {

//No combat action

cout << endl;

cout << oppName << " takes a defensive position";

cout << endl;

cout << "You have steped back from your opponent";

}

//Opponent blocks - player paries

if (cpuCombat == 2 && cmbtAct == 4) {

//No combat action

cout << endl;

cout << oppName << " takes a defensive position";

cout << endl;

cout << "You look to deflect an attack that never comes";

}

//Opponent dodges - player attacks

if (cpuCombat == 3 && cmbtAct == 1) {

cout << endl;

//Player attacks

cout << "You attack " << endl;

plyrAtk = attack(plyrStats[1], oppStats[2]);

if (plyrAtk = 0) {

cout << "but miss";

plyrDmg = 0;

} else if (plyrAtk >= 1) {

plyrDmg = getDamage(plyrStats[1], oppStats[2], plyrAtk);

cout << "and strike for " << plyrDmg << " damage";

}

cout << endl;

//Opponent dodges versus attack

oppDdge = getDodge(plyrAtk, oppStats[3]);

cout << oppName << " dodged ";

if (oppDdge > plyrDmg) {

cout << "all ";

} else {

cout << oppDdge;

}

cout << " Out of the damage";

oppHealth = oppHealth - plyrDmg;

cout << endl << "Player Health: " << plyrHealth;

cout << endl << "Opp Health: " << oppHealth;

}

//Opponent dodges - player blocks

if (cpuCombat == 3 && cmbtAct == 2) {

//No combat action

cout << endl;

cout << oppName << " slides away from you";

cout << endl;

cout << "Attack coming!";

}

//Opponent dodges - player paries

if (cpuCombat == 3 && cmbtAct == 4) {

//No combat action

cout << endl;

cout << oppName << " Hides back, miss attack";

cout << endl;

cout << "You predicted an attack, but it never comes";

}

//opponent paries - player attacks

if (cpuCombat == 4 && cmbtAct == 1) {

cout << endl;

plyrDmg = attack(plyrStats[1], oppStats[2]);

oppPrry = getCirt(plyrDmg, oppStats[4]);

if (oppPrry > plyrDmg) {

cout << "You attack" << endl;

cout << oppName << "A heavey attack coming ";

int counter = 7 / 100;

oppDmg = getDamage(oppStats[1], plyrStats[2], counter);

plyrHealth = plyrHealth - oppDmg;

cout << oppDmg;

} else {

cout << "You attack " << endl;

cout << oppName << " Heavy Attack, such painful! taking extra damage";

}

cout << endl << "Player Health: " << plyrHealth;

cout << endl << "Opp Health: " << oppHealth;

}

//opponent paries - player blocks

if (cpuCombat == 4 && cmbtAct == 2) {

//non combat action

cout << endl;

cout << oppName << " Attack, but it never comes";

cout << endl;

cout << "You accept a defensive stance, preparing to block";

}

//opponent paries - player dodges

if (cpuCombat == 4 && cmbtAct == 3) {

//non combat action

cout << endl;

cout << oppName << " slide away to deflect an attack";

cout << endl;

cout << "You slide away from " << oppName;

}

if (plyrHealth <= 5 && plyrHealth >=0)

{

cout << "You are bleeding wounded, will you close your eyes, enjoy the peace\n"

"or becoming the Legency?";

cin >> wounded;

if(wounded == 'y')

{

cout << endl << "(......)R.I.R.";

}

}

if (plyrHealth < 0)

{

cout << endl << "You have been died in combat" << endl;

exit(2);

}

}

if (round < 5) {

cout << endl << endl << "Trying to challenge next day?(y or n)";

cin >> advance;

while (advance != 'y' && advance != 'n') {

cout << endl << "You have entered an invalid entry, please re-enter: ";

cin >> advance;

}

return advance;

} else {

cout << endl << "Thank you for playing The Gladiator!";

cout << endl;

exit(1);

}

}

int getCpuAction() {

short oppCh = 0; //Cpu choice

//Set random 1 through 5 for the cpu

oppCh = rand() % 4 + 1;

return oppCh;

}