Project 2

<Bulwark>

CIS-17A 48983
Branden Hitt
Date: 12/10/2015

Table of Contents

Introduction

Title: Bulwark

This is a small program that lets a user play a game of Bulwark, an RPG horde simulator. To put it into simple terms, you pick a character class, give your character a name, and fight through as many waves of enemies as you can before you die.

At the beginning of the program, the user is greeted by a message describing the conflict at hand:

> "*Quickly soldier, we need your help to defend the gates!*
>
> *Cthulhu has brought a heinous army to destroy all of mankind!!*
>
> *Fight as long as you can and leave none standing!"*

They are then told to create a name for their character and pick a class. For instance, they could name themselves *Mark the Ruthless* and pick the *Beserker* class:

Available classes)

1. Ranger / weapon - bow / special - "HeadShot"
2. Warlock / weapon - staff / special - "Transfusion"
3. Paladin / weapon – sword / special - "Fortify"
4. Beserker / weapon – war axe / special - "Ragnarok"

From there, the player is thrown into the battle and they begin fighting wave after wave of enemies. Every wave contains a set of units that must be fought, one at a time.

Wave)

1) Random weak enemy – (young vampire, dark rider, lunatic, abomination, or cultist.)
   -user gets some health back
2) Random weak enemy – (young vampire, dark rider, lunatic, abomination, or cultist.)
   -user gets some health back

3) Random elite* weak enemy – same thing as above just with better stats or extra modifier

   -user gets health and a random stat bonus

4) Random moderate enemy – (elder vampire, dread horseman, maniac, monstrocity, or fanatic.)

   -user gets some health back

5) Random moderate enemy – (elder vampire, dread horseman, maniac, monstrocity, or fanatic.)

   -user gets some health back

6) Random elite* moderate enemy – same thing as above just with better stats or modifiers.

   -user gets some health back and a random stat bonus

7) Random Boss – (Nosferatu, the Headless Horseman, Dr. Jeckyl, Frankenstein's Monster, or Cthulhu.)

   -user gets some health back, gains max health, and a random stat bonus

Combat)

Each time the player faces a new enemy, there is a round of combat until one of them is dead. By default the enemy attacks first unless the player has the “First Strike” weapon bonus. During the player's attack, they chose from 4 options:

1)Attack

2)Self-Heal

3)Use Special attack

4)Surrender (and die) *this is a way to exit the game early

The enemy will only use a regular attack during their turn but may also have other effects throughout the combat depending on the type of enemy.

*Note that both the player and the enemy has a chance to miss their attacks

As soon as a wave is completed, another wave is thrown at the player. This repeats until the player is dead. From there the player is given their recorded stats which are also sent to a text file of the character's name. Last but not least, the player is prompted for whether or not they want to see the highscores from previous players. This is simply a file that is read and written (binary) that has a list of previous player's and their scores.

Summary

Project size: 1695 lines (with comments)

Number of Classes: 12

Functions in Main: 14

For this project I decided to create some sort of adventure game that would have enemies and players so that I could utilize and get a lot of practice in with classes. What I ended up with was a game where a player must defend against an endless onslaught of random enemies and bosses.

To begin, I had to decide what sort of attributes an "Enemy" could have. This would be my practice with inheritance in general. It seemed that at least with enemies they would all have a name/type , a health value, and a damage value. The EnemyUnit class would end up being a base class for the various types of enemies that are generated.

First, I inherited the EnemyUnit class onto the weakest type of enemy which I reffered to as the CrapUnit class. When one of these are constructed, the game randomly chooses from set values to create the enemy. In addition to the inherrited attributes, my CrapUnit class was given a weakness and a type. The weakness was used to find out if they had less of a chance to hit a certain player type, whereas the "type" attribute was never actually implemented and instead was just given a value to describe the type of enemy. Beyond that, the CrapUnit class was also inherited into the elite class version called CrapUnitElite. The only difference with this class is that the units are called elite versions with improved stats and a bonus. The bonus attribute may have an effect during combat to aid the elite enemy.

Another inheritance from the EnemyUnit class was my ModerateUnit class. This was an enemy type that would be the next step up from the CrapUnit class. These units would also be randomly generated when created and would have a different set of names and values to pick from. The idea up to that point was that there would be different levels of enemies to fight but there would be a random version so that it could be different every game:

**CrapUnit < CrapUnitElite < ModerateUnit < ModerateUnitElite < BossUnit*

I should note that the ModerateUnit class was similar to the CrapUnit class since it also added its own set of weaknesses and types. Beyond that, the ModerateUnit class was also inherited by the elite version of the class.

Last but not least, the EnemyUnit class was inherited by my BossUnit class. This was the class of enemy that I had decided would be at the end of an enemy wave and would give the toughest challenge. The boss is also randomly generated and has its own set of values in addition to a individual bonus effect and a special attack. The strongest of the bosses you can face is Cthulhu himself, whose special (if it is used) will automatically kill the player and end their run.

So now that I had the enemies worked out for how they might be implemented, I had to decide how the Player class would work. Similar to the enemy class, the Player class had a bunch of attributes and functions involved with its attributes. The biggest impact to the player is his initial choice on his archetype. The player is given 4 choices, and each one provides them with a unique playstyle:

Archetypes)

1)Ranger

2)Warlock

3)Paladin

4)Beserker

Through the Player class, I also decided to try out aggregation instead of inheritance. This was good practice for including a separate class inside another one. I achieved this effect by means of a Weapon class since every Player might have a Weapon. Seemed simple enough, yet I still had to understand how to implement it and make it work correctly. Once that was all done, I merely had to get the gameplaye down.

Most of the gameplay takes place within functions in main. The organization of the combat was not too difficult to figure out, however understanding the notation involved with classes and their behaviors was a learning experience. Through a lot of trial and error I was eventually able to get a running game. The format is still not the prettiest, but I am more than happy with what I learned involving classes. Classes had made sense in concept, but I still had to learn what they would look like in practice.

Beyond the gameplay, I also implemented some file input and output as continued practice. At the end of each game, the current player's stats are sent both to a .txt file and to a .dat file (.dat file is sent in binary). The text file can be viewed outside the program, however the .dat file has a record of multiple games and would need to be read in through the game to be viewed (which can be done at the end if the player chooses).

To end my project, I tried to implement some other concepts that we had covered even if they did not have any real practical effect on the game. One instance of this was the vector template that we were learning about in class. You ,Dr. Lehr, had encouraged us to alter it in a way that would allow a push function that could be added. You also taught us how to make it more efficient by doubling the allocated size (when needed) instead of creating a new array every time. I really wanted to make sure I could implement the more efficient version and so I added it to my program. Sure enough, I got it to work and am happy I took the time since it was good practice with other things such as pointer's and destructors.

Overall, I feel that I have learned a great deal this semester and can see my improvement with coding every week. Hopefully, the game is somewhat enjoyable.

Future Improvements

Most of what I would want to improve would be display wise, as usual. Once I have knowledge in GUIs I might revisit this program and make it more visually appealing. Other than that I would maybe work on balancing the game out since the player archetypes are not at all balanced and since the game can be very unforgiving.

Simple Program Function Overview

```
main(
        Player user; ← Player object creation
        newWave();   ← starts the actual game
                *gameplay ends here
        PlayerData records; ← structure for file I/O
        retPD();          ← fill the structure
        writeRec():     ← write stats to txt file
        writeHi();   ← append stats to binary file
        readHi();  ← read stats from binary file
        vecFun();  ← vector template practice
)
newWave(
        duelEn();  ← fight a weak enemy
                .addHel(); ← add health to player
        duelEn();  ← fight a weak enemy
                .addHel(); ← add health to player
        duelEEn(): ← fight elite weak enemy
                .addHel();  ← add health to player
                statBon(); ← give player a bonus to their stats
        duelMEn();  ← fight a moderate enemy
                .addHel(); ← add health to player
        duelMEn();  ← fight a moderate enemy
                .addHel(); ← add health to player
```

```
    duelMEE(); ← fight a moderate elite enemy
        .addHel();  ← add health to player
        statBon(); ← give player a bonus to their stats
    pauseG():  ← pause the game
    duelBos();  ← duel a boss enemy
        .addMaXH();  ← add maximum health to player
        .addHel();  ← add health to player
        statBon(); ← give a player bonus to their stats
    *Player eventually dies after the wave is looped
    user.setWave();  ← sets the value for how many waves they beat
)
```

Psuedocode

```
/*
    * File: main.cpp
    * Author: Branden Hitt
    * Created on November 27th, 2015, 8:28 PM
    * Purpose: Bulwark (project #2, adventure game)
    */

    //System Libraries
     //I/O standard
     //string library
     //rand
     //file i/o
     //strcpy


    //User Libraries
     //player library
     //weapon library
     //enemy library
     //crap unit library
     //crap unit elite library
     //moderate unit library
     //moderate unit elite library
     //boss unit library
```

```
 //my vector library/template
 //vector library
 //unit abstract class
//Global Constants

//Function Prototypes
  //new wave of enemies
  //pick char move
  //duel enemy
  //duel elite enemy
  //duel moderate enemy
  //duel moderate elite enemy
  //duel boss
  //stat bonus for player
  //pause the game
  //write records to a file
  //write record to highscore file
  //read highscores to player
  //returns a structure from data
  //vector function
//Execution Begins Here!

//declare variables
 //character name
 //archetype choice
 //repeat
 //set random seed
//greet the user
//set up player file
//create player object

//start waves

//set player data


//Output Stats to Player
//write out data to txt

//write out data to dat

//read in data for output
//vector fun

//Exit Stage Right


//********************************//
// pause //
//********************************//
  //prompt for enter key
```

```
//*******************************//
// new wave //
//*******************************//

//declare variables
 //wave count
 //continue playing


//enemy1


//enemy2


//elite enemy


//enemy 4


//enemy 5


//enemy 6
//boss
//*******************************//
// pick move //
//*******************************//

//prompt for choice

//*******************************//
// duel enemy //
//*******************************//

//declare variables

//create enemy

//introduce enemy

//find who hits first


//user hits first
  //loop
  //enemy attacks player
  //player attacks enemy
 //continue until one dead
```

```
//check for user death
//*******************************//
// duel elite enemy //
//*******************************//

//declare variables

//create enemy

//introduce enemy

//find who hits first


//user hits first
//enemy attacks player
//player attacks enemy
 //continue until one dead
//check for user death
//*******************************//
// duel moderate enemy //
//*******************************//

//declare variables

//create enemy

//introduce enemy

//find who hits first


//user hits first
//enemy attacks player
//player attacks enemy
 //continue until one dead
//check for user death
//*******************************//
// duel moderate elite enemy //
//*******************************//

//declare variables

//create enemy

//introduce enemy

//find who hits first


//user hits first
```

```
//enemy attacks player
//player attacks enemy
 //continue until one dead
//check for user death
//********************************//
// duel boss //
//********************************//

//declare variables


//create enemy

//introduce enemy

//check for special
//find who hits first

//enemy hits first
//user hits first (or second if the enemy has trample)

//check for miss
//deal damage

//enemy attacks player
 //check for dodge
//check for miss

//deal damage
//player attacks enemy
 //continue until one dead
//check for user death
 //win prompt
//********************************//
// stat bonus //
//********************************//


 //random bonus given

//small health gain

//big health gain

//small dps gain
//large dps gain
//small max health gain
//large max health gain
//small health, max health, and dps gain
//********************************//
// return PlayerData //
```

```
//********************************//

//declare struct
 //fill struct
//return struct
//********************************//
// write Record //
//********************************//

//open file
//write structure to file
//close file
//********************************//
// file i/o highscore //
//********************************//
//create structure
//open file
//write file from structure
//close file
//********************************//
// read Record //
//********************************//
//create structure

//open file



//read file into structure

//output structure

//close file
//********************************//
// vector fun //
//********************************//

//create an array of vectors from template


/*for(int i=0;i<11;i++){
cout<<stuff[i]<<endl;
}*/
}
```

Concepts from the Book (Savitch 9th Ed)

Chapter	Concept	Description	Location
9. Pointers	T *aptr	Pointer variable	Line 20 “SimpleVector.h”
	`aptr = new T [alSize];`	Initializing a dynamic array	Line 67 “SimpleVector.h”
	T *aptr	Pointer to dynamic array	Line 20 “SimpleVector.h”
	delete [] aptr;	De-allocation of array	Line 188 “SimpleVector.h”
10. Characters, C-Strings, string class	char name[SIZE];	c-string	Line 15 “PlayerData.h”
	`strcpy(temp.name,user.getName().c_str());`	strcpy – copy a c-string into another	Line 634 main
	`strcpy(temp.name,user.getName().c_str());`	c_str() - send string to c-string	Line 634 main
	string name;	String variable/object	Line 16 “Player.h”
	getline();	Grabs a string from input	Line 60 main
11.Structured Data	struct PlayerData	Structure initialization	Line 14 “PlayerData.h”
	PlayerData temp	Structure Variable	Line 682 main
	return temp	returning a structure	Line 642 main
12. Advanced File Operations	fstream fStuff	File stream object	Line 649 main
	ios::in	Input mode	Line 685 main
	ios::out	Output mode	Line 671 main
	ios::binary	Binary mode	Line 685 main
	“Highscores.dat”	Binary File	Line 685 main
	fin.read	Reads in binary input from a file	Line 688 main
	fout.write	Writes out binary input to a file	Line 673 main
	reinterpret_cast<char *>(&p),sizeof(p)	Converts a pointer to one of a char *	Line 673 main
13. Intro to Classes	Class EnemyUnit	Class	Line 13 “EnemyUnit.h”
	public:	Access specifiers	Line 19 “EnemyUnit.h”
	getName	Accessor	Line 23 “EnemyUnit.h”
	setName	Mutator	Line 22 “EnemyUnit.h”

	Player user()	Object initialization	Line 76 main
	`int getHlth(){return health;}`	In-line member function	Line 25 “EnemyUnit.h”
	`EnemyUnit(string,int,int);`	constructor	Line 21 “EnemyUnit.h”
	`EnemyUnit(){name="Empty",health=0,dps=0;}`	Defaulted constructor	Line 20 “EnemyUnit.h”
	~SimpleVector()	destructor	Line 37 “SimpleVector.h”
14. More about Classes	static string allegiance	Static member variable	Line 18 “EnemyUnit.h”
	`SimpleVector(const SimpleVector &);`	Copy constructor	Line 34 “SimpleVector.h”
	`T &operator[](const int &);`	Operator overloading	Line 47 “SimpleVector.h”
	Weapon weapon;	aggregation	Line 23 “Player.h”
15. Inheritance, Polymorphism, and Virtual Functions	`class ModerateUnit: public EnemyUnit{`	inheritance	Line 11 “ModerateUnit.h”
	Protected: string type	Protected member	Line 12 “ModerateUnit.h”
	virtual int getHate()=0;	Pure virtual function	Line 17 “Unit.h”
16. Exceptions, Templates, and STL	Try{} catch{	Exceptions	Line 65 “SimpleVector.h”
	`template <class T>`	class template	Line 16 “SimpleVector.h”

References

1.Textbook – Starting out with C++ (Gaddis, 7th Edition)
2. The Call of Cthulhu – H.P. Lovecraft

Flowcharts

Flowcharts can be found in the Flowcharts folder

Program Code

```
/*
    * File: main.cpp
    * Author: Branden Hitt
    * Created on November 27th, 2015, 8:28 PM
    * Purpose: Bulwark (project #2, adventure game)
    */

    //System Libraries
    #include <iostream>//I/O standard
    #include <string>//string library
    #include <cstdlib>//rand
    #include <fstream>//file i/o
    #include <cstring>//strcpy
    using namespace std;
```

```
//User Libraries
#include "Player.h"//player library
#include "Weapon.h"//weapon library
#include "EnemyUnit.h"//enemy library
#include "CrapUnit.h"//crap unit library
#include "CrapUnitElite.h"//crap unit elite library
#include "ModerateUnit.h"//moderate unit library
#include "ModerateUnitElite.h"//moderate unit elite library
#include "BossUnit.h"//boss unit library
#include "SimpleVector.h"//my vector library/template
#include "PlayerData.h"//vector library
#include "Unit.h"//unit abstract class
//Global Constants

//Function Prototypes
void newWave(Player &); //new wave of enemies
short pickMov(Player &); //pick char move
bool duelEn(Player &); //duel enemy
bool duelEEn(Player &); //duel elite enemy
bool duelMEn(Player &); //duel moderate enemy
bool duelMEE(Player &); //duel moderate elite enemy
bool duelBos(Player &); //duel boss
void statBon(Player &); //stat bonus for player
void pauseG(); //pause the game
void writeRec(PlayerData ,string); //write records to a file
void writeHi(PlayerData); //write record to highscore file
void readHi(); //read highscores to player
PlayerData retPD(Player); //returns a structure from data
void vecFun(); //vector function
//Execution Begins Here!
int main(int argc, char** argv) {
//declare variables
string cName; //character name
short archPic=5,hi=0;//archetype choice
bool rep=false; //repeat
srand(static_cast<unsigned int>(time(0))); //set random seed
//greet the user
cout<<"******************************************************************"<<endl;
cout<<"| Quickly soldier, we need your help to defend the gates! |"<<endl;
cout<<"| Cthulhu has brought a heinous army to destroy all of mankind! |"<<endl;
cout<<"| Fight as long as you can and leave none standing! |"<<endl;
cout<<"******************************************************************"<<endl;
//set up player file
cout<<"Before you go, what was your name? (Ex. Alexander the Great)"<<endl;
getline(cin,cName);
cout<<"Finally, what is your archetype."<<endl;
do{
rep=false;
cout<<"Enter in 1 for Ranger: "<<endl;
cout<<" 2 for Warlock:"<<endl;
cout<<" 3 for Paladin:"<<endl;
cout<<" 4 for Beserker:"<<endl;
cin>>archPic;
if(archPic>4||archPic<1){
cout<<"Invalid Entry"<<endl;
rep=true;
}
}while(rep==true);
cin.ignore();
//create player object
Player user(cName,archPic);
//start waves
cout<<"God speed, "<<user.getArch()<<" "<<user.getName()<<endl;
cout<<endl;
cout<<endl;
cout<<endl;
newWave(user);
//set player data
PlayerData records;
records=retPD(user);
//Output Stats to Player
cout<<endl;
cout<<"Character Name : "<<user.getName()<<endl;
cout<<"Archetype : "<<user.getArch()<<endl;
cout<<"Health : "<<user.getHel()<<endl;
cout<<"Damage : "<<user.getDps()<<endl;
cout<<"Maximum Health : "<<user.getMaxH()<<endl;
cout<<"Weapon Name : "<<user.getWepn().getWNam()<<endl;
cout<<"Weapon Bonus : "<<user.getWepn().getWBon()<<endl;
cout<<"Waves Completed: "<<user.getWave()<<endl;
cout<<endl;
cout<<endl;
//write out data to txt
writeRec(records,cName);
//write out data to dat
writeHi(records);
//read in data for output
do{
cout<<"Would you like to view the High Scores?"<<endl;
cout<<"Enter in 1 for yes or 2 for no:"<<endl;
cin>>hi;
}while(hi<1 || hi>2);
if(hi==1) readHi();
//vector fun
vecFun();
//Exit Stage Right
return 0;
}
//*******************************//
// pause //
//*******************************//
void pauseG(){
string trash;
```

```
cout<<"******************************"<<endl;
cout<<"*Enter to Continue...  *"<<endl;
cout<<"******************************"<<endl;
cin.ignore();//prompt for enter key
getline(cin,trash);
}
//*******************************//
// new wave //
//*******************************//
void newWave(Player &user){
//declare variables
unsigned short wCount=0; //wave count
bool cont=true; //continue playing
do{
cout<<"Wave : "<<wCount+1<<endl;
//enemy1
if(cont) cont=duelEn(user);
if(cont) cout<<"(Kill Reward) ", user.addHel(50);
//enemy2
if(cont) cont=duelEn(user);
if(cont) cout<<"(Kill Reward) ", user.addHel(50);
//elite enemy
if(cont) cont=duelEEn(user);
if(cont){
cout<<"(Kill Reward) ";
user.addHel(100);
statBon(user);
}
//enemy 4
if(cont) cont=duelMEn(user);
if(cont) cout<<"(Kill Reward) ", user.addHel(50);
//enemy 5
if(cont) cont=duelMEn(user);
if(cont) cout<<"(Kill Reward) ", user.addHel(50);
//enemy 6
if(cont) cont=duelMEE(user);
if(cont){
cout<<"(Kill Reward) ",
user.addHel(100);
statBon(user);
}
//boss
if(cont) cout<<"PREPARE YOURSELF, A BOSS APPROACHES!!!"<<endl;
if(cont) pauseG();
if(cont) cont=duelBos(user);
if(cont){
cout<<"(Kill Reward) ";
user.addMaxH(25);
cout<<"(Kill Reward) ";
user.addHel(100);
statBon(user);
```

```
}
if(cont) wCount++;
}while(cont);
cout<<"You have fought bravely"<<endl;
user.setWave(wCount);
}
//********************************//
// pick move //
//********************************//
short pickMov(Player &user){
short cho;//prompt for choice
do{
cout<<"Your Maximum Health: "<<user.getMaxH()<<endl;
cout<<"Your Current Health: "<<user.getHel()<<endl;
cout<<"1.Attack : "<<user.getDps()<<" (damage) ("<<user.getWepn().getWBon()<<")"<<endl;
cout<<"2.Self-Heal : "<<user.getSlfH()<<" (health)"<<endl;
cout<<"3.Special : "<<user.getSpec()<<" ("<<user.getSpC()<<" left)"<<endl;
cout<<"4.Surrender : (The coward's way out)"<<endl;
cin>>cho;
if(cho==3 && user.getSpC()<=0) cho=6;
}while(cho<1 || cho>4);
return cho;
}
//********************************//
// duel enemy //
//********************************//
bool duelEn(Player &user){
//declare variables
short hitCh=0, userCho=0;
//create enemy
CrapUnit enemy;
//introduce enemy
cout<<"Before you stands an enemy "<<enemy.getName()<<endl;
//find who hits first
if(user.getWepn().getWBon()=="First Strike"){
cout<<"(First Strike) You get a free hit."<<endl;
//user hits first
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()||enemy.getWeak()=="Any") hitCh+=2;
if(hitCh<=2) cout<<"You missed your attack."<<endl;
else enemy.takeDmg(user.getDps());
pauseG();
}
do{ //loop
//enemy attacks player
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()) hitCh-=1;
if(user.getWepn().getWBon()=="Dodge Chance") hitCh-=2;
if(hitCh<=3) cout<<"The enemy missed."<<endl;
else user.takeDam(enemy.getDps());
```

```
cout<<"********************"<<endl;
}
//player attacks enemy
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
userCho=pickMov(user);
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()||enemy.getWeak()=="Any") hitCh+=2;
if(userCho==1){
if(hitCh<=2) cout<<"Your weapon has missed. No damage done."<<endl;
else{
enemy.takeDmg(user.getDps());
if(user.getWepn().getWBon()=="LifeSteal"){
user.addHel(user.getDps());
}
}
}else if(userCho==2){
user.healS();
}else if(userCho==3){
enemy.setHlth(user.useSpec(enemy.getHlth()));
}else{
user.setHel(0);
}
pauseG();

}
}while((user.getHel()>0) && (enemy.getHlth()>0)); //continue until one dead
//check for user death
if(user.getHel()==0){
cout<<"You have died"<<endl;
pauseG();
return false;
}
else{
cout<<"You have slain the enemy"<<endl;
pauseG();
return true;
}
}
//********************************//
// duel elite enemy //
//********************************//
bool duelEEn(Player &user){
//declare variables
short hitCh=0, userCho=0;
//create enemy
CrapUnitElite enemy;
//introduce enemy
cout<<"Before you stands an enemy "<<enemy.getName()<<endl;
//find who hits first
if(user.getWepn().getWBon()=="First Strike"){
cout<<"(First Strike) You get a free attack"<<endl;
//user hits first
```

```
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()||enemy.getWeak()=="Any") hitCh+=1;
if(enemy.getBon()=="Dodge Chance") hitCh-=1;
if(hitCh<=4) cout<<"You missed your attack."<<endl;
else enemy.takeDmg(user.getDps());
pauseG();
}
do{
//enemy attacks player
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()) hitCh-=1;
if(user.getWepn().getWBon()=="Dodge Chance") hitCh-=2;
if(hitCh<=3) cout<<"The enemy missed."<<endl;
else {
user.takeDam(enemy.getDps());
if(enemy.getBon()=="LifeSteal"){
cout<<"(Lifesteal) The enemy stole life with his attack."<<endl;
enemy.setHlth((enemy.getHlth())+enemy.getDps());
}
}
cout<<"********************"<<endl;
}
//player attacks enemy
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
userCho=pickMov(user);
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()||enemy.getWeak()=="Any") hitCh+=1;
if(enemy.getBon()=="Dodge Chance") hitCh-=1;
if(userCho==1){
if(hitCh<=4) cout<<"You weapon has missed. No damage done."<<endl;
else{
enemy.takeDmg(user.getDps());
if(user.getWepn().getWBon()=="LifeSteal"){
user.addHel(user.getDps());
}
}
}else if(userCho==2){
user.healS();
}else if(userCho==3){
enemy.setHlth(user.useSpec(enemy.getHlth()));
}else{
user.setHel(0);
}
pauseG();
}
}while((user.getHel()>0) && (enemy.getHlth()>0)); //continue until one dead
//check for user death
if(user.getHel()==0){
cout<<"You have died"<<endl;
pauseG();
```

```
return false;
}
else{
cout<<"You have slain the enemy"<<endl;
pauseG();
return true;
}
}
//******************************//
// duel moderate enemy //
//******************************//
bool duelMEn(Player &user){
//declare variables
short hitCh=0, userCho=0;
//create enemy
ModerateUnit enemy;
//introduce enemy
cout<<"Before you stands an enemy "<<enemy.getName()<<endl;
//find who hits first
if(user.getWepn().getWBon()=="First Strike"){
cout<<"(First Strike) You get a free hit."<<endl;
//user hits first
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()||enemy.getWeak()=="Any") hitCh+=2;
if(hitCh<=2) cout<<"You missed your attack."<<endl;
else enemy.takeDmg(user.getDps());
pauseG();
}
do{
//enemy attacks player
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()) hitCh-=1;
if(user.getWepn().getWBon()=="Dodge Chance") hitCh-=2;
if(hitCh<=3) cout<<"The enemy missed."<<endl;
else user.takeDam(enemy.getDps());
cout<<"*******************"<<endl;
}
//player attacks enemy
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
userCho=pickMov(user);
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()||enemy.getWeak()=="Any") hitCh+=2;
if(userCho==1){
if(hitCh<=2) cout<<"You weapon has missed. No damage done."<<endl;
else{
enemy.takeDmg(user.getDps());
if(user.getWepn().getWBon()=="LifeSteal"){
user.addHel(user.getDps());
}
}
```

```
}else if(userCho==2){
user.healS();
}else if(userCho==3){
enemy.setHlth(user.useSpec(enemy.getHlth()));
}else{
user.setHel(0);
}
pauseG();

}
}while((user.getHel()>0) && (enemy.getHlth()>0)); //continue until one dead
//check for user death
if(user.getHel()==0){
cout<<"You have died"<<endl;
pauseG();
return false;
}
else{
cout<<"You have slain the enemy"<<endl;
pauseG();
return true;
}
}
//*******************************//
// duel moderate elite enemy //
//*******************************//
bool duelMEE(Player &user){
//declare variables
short hitCh=0, userCho=0;
//create enemy
ModerateUnitElite enemy;
//introduce enemy
cout<<"Before you stands an enemy "<<enemy.getName()<<endl;
//find who hits first
if(user.getWepn().getWBon()=="First Strike"){
cout<<"(First Strike) You get a free attack"<<endl;
//user hits first
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()||enemy.getWeak()=="Any") hitCh+=1;
if(enemy.getBon()=="Dodge Chance") hitCh-=1;
if(hitCh<=4) cout<<"You missed your attack."<<endl;
else enemy.takeDmg(user.getDps());
pauseG();
}
do{
//enemy attacks player
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()) hitCh-=1;
if(user.getWepn().getWBon()=="Dodge Chance") hitCh-=2;
if(hitCh<=3) cout<<"The enemy missed."<<endl;
else {
```

```
user.takeDam(enemy.getDps());
if(enemy.getBon()=="Rupture"){
cout<<"(Rupture) The enemy has left a permanent wound."<<endl;
cout<<" -20 to your maximum health"<<endl;
user.setMaxH((user.getMaxH())-20);
if(user.getMaxH()>user.getHel()) user.setHel(user.getMaxH());
}
}
cout<<"*******************"<<endl;
}
//player attacks enemy
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
userCho=pickMov(user);
hitCh=rand()%10+1;
if(enemy.getWeak()==user.getArch()||enemy.getWeak()=="Any") hitCh+=1;
if(enemy.getBon()=="Dodge Chance") hitCh-=1;
if(userCho==1){
if(hitCh<=4) cout<<"You weapon has missed. No damage done."<<endl;
else{
enemy.takeDmg(user.getDps());
if(user.getWepn().getWBon()=="LifeSteal"){
user.addHel(user.getDps());
}
}
}else if(userCho==2){
user.healS();
}else if(userCho==3){
enemy.setHlth(user.useSpec(enemy.getHlth()));
}else{
user.setHel(0);
}
pauseG();
}
}while((user.getHel()>0) && (enemy.getHlth()>0)); //continue until one dead
//check for user death
if(user.getHel()==0){
cout<<"You have died"<<endl;
pauseG();
return false;
}
else{
cout<<"You have slain the enemy"<<endl;
pauseG();
return true;
}
}
//*******************************//
// duel boss //
//*******************************//
bool duelBos(Player &user){
//declare variables
```

```
short hitCh=0, userCho=0, spez=0, fear=0,hyde=0,devour=0;
int temp=0;
//create enemy
BossUnit enemy;
//introduce enemy
cout<<"Before you stands the abhorrent "<<enemy.getName()<<endl;
//check for special
if(enemy.getBon()=="Toxic"){
cout<<"(Toxic) A vile cloud has lowered your resilience."<<endl;//check for bonus
cout<<" -100 to your maximum health."<<endl;
user.setMaxH((user.getMaxH())-100);
if(user.getHel()>user.getMaxH()) user.setHel(user.getMaxH());
}
//find who hits first
if(enemy.getSpec()=="Trample"&& user.getHel()>100){
cout<<"(Trample) Hoofs storm across your person."<<endl;//enemy hits first
cout<<" -100 health"<<endl;
user.takeDam(100);
}
if(user.getWepn().getWBon()=="First Strike"){
cout<<"(First Strike) You get a free hit."<<endl;
//user hits first (or second if the enemy has trample)
hitCh=rand()%10+1;
if(hitCh<=3) cout<<"You missed your attack."<<endl;//check for miss
else enemy.takeDmg(user.getDps()),hyde++;//deal damage
pauseG();
}
do{
//enemy attacks player
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
hitCh=rand()%10+1;
if(user.getWepn().getWBon()=="Dodge Chance") hitCh-=2;//check for dodge
if(hitCh<=1) cout<<"The enemy missed."<<endl;//check for miss
else {
user.takeDam(enemy.getDps());//deal damage
if(enemy.getBon()=="LifeSteal") enemy.setHlth((enemy.getHlth())+enemy.getDps());
cout<<"(LifeSteal) "<<enemy.getName()<<" has stolen life equal to the damage dealt"<<endl;
}
cout<<"********************"<<endl;
}
devour=rand()%4+1;
if(devour==1 && enemy.getSpec()=="Devour"){
cout<<"(Devour) The Great Old One Cthulhu has devoured you whole."<<endl;
user.setHel(0);
}
//player attacks enemy
if((enemy.getHlth()!=0) && (user.getHel()!=0)){
userCho=pickMov(user);
hitCh=rand()%10+1;
if(enemy.getBon()=="Fear") fear=rand()%3+1;
if(fear==1){
```

```
hitCh=1;
cout<<"(Fear) "<<enemy.getName()<<" has struck you with fright."<<endl;
}
if(userCho==1){
if(hitCh==1&& enemy.getBon()=="The Call"){
cout<<"(The Call) You lose focus and harm yourself."<<endl;
cout<<"You take "<<user.getDps()<<" damage."<<endl;
user.takeDam(user.getDps());
}
else if(hitCh<=3) cout<<"You weapon has missed. No damage done."<<endl;
else{
enemy.takeDmg(user.getDps());
hyde++;
if(user.getWepn().getWBon()=="LifeSteal"){
user.addHel(user.getDps());
}
if(enemy.getBon()=="Conduit"){
cout<<"(Conduit) Electricity forks off the creature and onto you"<<endl;
cout<<" 20 Damage to your health."<<endl;
user.takeDam(20);
}
}
}else if(userCho==2){
user.healS();
}else if(userCho==3){
enemy.setHlth(user.useSpec(enemy.getHlth()));
}else{
user.setHel(0);
}
pauseG();

}
if((spez==0)&&(enemy.getHlth()<=100)&&(enemy.getSpec()=="LifeSwap")){
cout<<"(LifeSwap) Your life totals have been swapped"<<endl;
temp=enemy.getHlth();
enemy.setHlth(user.getHel());
user.setHel(temp);
spez++;
}
if((spez==0)&&(hyde>=5)&&(enemy.getSpec()=="Anger")){
cout<<"(Anger) You have driven "<<enemy.getName()<<" to desperation."<<endl;
cout<<"After swallowing a nefarious concoction, he has embraced his"<<endl;
cout<<"inner Mr. Hyde."<<endl;
enemy.setDps(200);
spez++;
}
}while((user.getHel()>0) && (enemy.getHlth()>0)); //continue until one dead
//check for user death
if(user.getHel()==0){
cout<<"You have died"<<endl;
pauseG();
return false;
```

```
}
else{
cout<<"You have slain the enemy"<<endl;//win prompt
pauseG();
return true;
}
}
//*******************************//
// stat bonus //
//*******************************//
void statBon(Player &user){
short luck;
luck=rand()%7+1;//random bonus given
if(luck==1){
//small health gain
user.setHel((user.getHel())+20);
cout<<"(Bonus) You have been healed +20 health."<<endl;
}else if(luck==2){
//big health gain
user.setHel((user.getHel())+50);
cout<<"(Bonus) You have been healed +50 health."<<endl;
}else if(luck==3){
//small dps gain
user.setDps((user.getDps())+10);
cout<<"(Bonus) You have gained +10 to your damage."<<endl;
}else if(luck==4){
//large dps gain
user.setDps((user.getDps())+20);
cout<<"(Bonus) You have gained +20 to your damage."<<endl;
}else if(luck==5){
//small max health gain
user.setMaxH((user.getMaxH())+30);
cout<<"(Bonus) You have gained +30 to your maximum health."<<endl;
}else if(luck==6){
//large max health gain
user.setMaxH((user.getMaxH())+60);
cout<<"(Bonus) You have gained +60 to your maximum health."<<endl;
}else{
//small health, max health, and dps gain
user.setMaxH((user.getMaxH())+20);
user.setHel((user.getHel())+20);
user.setDps((user.getDps())+20);
cout<<"(Bonus) You have gained +20 to your max health, +20 to your "<<endl;
cout<<" current health, and +20 to your damage."<<endl;
}
pauseG();
}
//*******************************//
// return PlayerData //
//*******************************//
PlayerData retPD(Player user){
```

```
//declare struct
PlayerData temp;//fill struct
strcpy(temp.name,user.getName().c_str());
temp.maxHlth=user.getMaxH();
temp.dps=user.getDps();
strcpy(temp.arch,user.getArch().c_str());
strcpy(temp.weapon,user.getWepn().getWNam().c_str());
strcpy(temp.special,user.getSpec().c_str());
temp.waveC=user.getWave();
//return struct
return temp;
}
//*******************************//
// write Record //
//*******************************//
void writeRec(PlayerData r, string cName){
//open file
fstream fStuff;
cName+=".txt";
fStuff.open(cName.c_str(), ios::out);
//write structure to file
fStuff<<"Name : "<<r.name<<endl;
fStuff<<"Maximum Health : "<<r.maxHlth<<endl;
fStuff<<"Damage : "<<r.dps<<endl;
fStuff<<"Archetype : "<<r.arch<<endl;
fStuff<<"Weapon : "<<r.weapon<<endl;
fStuff<<"Special : "<<r.special<<endl;
fStuff<<"Waves Completed : "<<r.waveC<<endl;
//close file
fStuff.close();
}
//*******************************//
// file i/o highscore //
//*******************************//
void writeHi(PlayerData r){
//create structure
PlayerData temp;
//open file
fstream fout;
fout.open("Highscores.dat", ios::out | ios::binary | ios::app);
//write file from structure
fout.write(reinterpret_cast<char *>(&r),sizeof(r));
//close file
fout.close();
}
//*******************************//
// read Record //
//*******************************//
void readHi(){
//create structure
PlayerData temp;
```

```
//open file
fstream fin;
fin.open("Highscores.dat", ios::in | ios::binary);
while(fin){
//read file into structure
fin.read(reinterpret_cast<char *>(&temp),sizeof(temp));
//output structure
cout<<"Name : "<<temp.name<<endl;
cout<<"Maximum Health : "<<temp.maxHlth<<endl;
cout<<"Damage : "<<temp.dps<<endl;
cout<<"Archetype : "<<temp.arch<<endl;
cout<<"Weapon : "<<temp.weapon<<endl;
cout<<"Special : "<<temp.special<<endl;
cout<<"Waves Completed : "<<temp.waveC<<endl;
}
//close file
fin.close();
}
//*******************************//
// vector fun //
//*******************************//
void vecFun(){
//create an array of vectors from template
SimpleVector<int> stuff(10);
stuff.pushEle(5);
/*for(int i=0;i<11;i++){
cout<<stuff[i]<<endl;
}*/
}


/*
* File: Unit.h
* Author: Branden
* Purpose: Abstract base class
* Created on December 11, 2015, 8:12 PM
*/
#include <string>
using namespace std;
#ifndef UNIT_H
#define UNIT_H
class Unit{
private:
int hate;
public:
Unit(){ hate=0;} //constructor
virtual int getHate()=0; //pure virtual function
};
#endif /* UNIT_H */
```

```
/*
    * File: EnemyUnit.h
    * Author: Branden Hitt
    * Purpose: Specification of the EnemyUnit Class
    * Created on November 27, 2015, 8:30 PM
    */
    #include <string>
    using namespace std;

    #ifndef ENEMYUNIT_H
    #define ENEMYUNIT_H
    #include "Unit.h"
    class EnemyUnit: public Unit{
    protected:
    string name;
    int health;
    int dps;
    static string allegiance;
    public:
    EnemyUnit(){name="Empty",health=0,dps=0;} //Default Constructor (in-line)
    EnemyUnit(string,int,int); //Constructor
    void setName(string); //Mutator
    string getName(); //Accessor
    void setHlth(int); //Mutator
    int getHlth(){return health;} //Accessor (in-line)
    void setDps(int); //Mutator
    int getDps(); //Accessor
    void takeDmg(int); //take damage function
    int getHate(){ return 0;}
    };

    #endif /* ENEMYUNIT_H */


/*
    * File: EnemyUnit.h
    * Author: Branden Hitt
    * Purpose: Implementation of the EnemyUnit Class
    * Created on November 27, 2015, 8:40 PM
    */

    #include "EnemyUnit.h"
    #include <string>
    #include <iostream>
    using namespace std;

    string EnemyUnit::allegiance="The Dark Lord Cthulhu";

    EnemyUnit::EnemyUnit(string n, int h, int d){
    name=n;
    health=h;
    dps=d;
    }
    void EnemyUnit::setName(string n){
    name=n;
    }
    string EnemyUnit::getName(){
    return name;
    }
    void EnemyUnit::setHlth(int h){
    health=h;
    }
    void EnemyUnit::setDps(int d){
    dps=d;
    }
    int EnemyUnit::getDps(){
    return dps;
    }
    void EnemyUnit::takeDmg(int d){
    cout<<"You hit them for "<<d<<" damage!"<<endl;
    health-=d;
    if(health<0) health=0;
    }
```

```
/*
    * File: CrapUnit.h
    * Author: Branden Hitt
    * Purpose: Specification of the CrapUnit class
    * Created on December 1, 2015, 1:25 PM
    */
    #ifndef CRAPUNIT_H
    #define CRAPUNIT_H
    #include "EnemyUnit.h"
    class CrapUnit : public EnemyUnit{
    protected:
    string type;
    string weakness;
    public:
    CrapUnit(); //Default Constructor
    CrapUnit(string,int,int,string,string); //Constructor
    void setType(string); //Mutator
    void setWeak(string); //Mutator
    string getType(){ return type;} //Accessor (in-line)
    string getWeak(){ return weakness;} //Accessor (in-line)
    };
    #endif /* CRAPUNIT_H */
```

```
/*
    * File: CrapUnit.h
    * Author: Branden Hitt
    * Purpose: Implementation of the CrapUnit class
    * Created on December 1, 2015, 1:35 PM
    */
```

```
#include "CrapUnit.h"
#include <cstdlib>
#include <iostream>

CrapUnit::CrapUnit(){
srand(static_cast<unsigned int>(time(0)));
int pick=(rand()%5+1);
if(pick==1){
name="Young Vampire";
health=160;
dps=25;
type="Vampire";
weakness="Paladin";
}else if(pick==2){
name="Dark Rider";
health=140;
dps=30;
type="Vandal";
weakness="Ranger";
}else if(pick==3){
name="Lunatic";
health=100;
dps=50;
type="Mad Scientist";
weakness="Warlock";
}else if(pick==4){
name="Abomination";
health=200;
dps=20;
type="Frankenstein";
weakness="Beserker";
}else{
name="Cultist";
health=50;
dps=120;
type="Cthulhu Spawn";
weakness="Any";
}
}
CrapUnit::CrapUnit(string na, int he, int d, string ty, string weak){
name=na;
health=he;
dps=d;
type=ty;
weakness=weak;
}
void CrapUnit::setType(string t){
type=t;
}
void CrapUnit::setWeak(string w){
weakness=w;
}
```

```
/*
    * File: CrapUnitElite.h
    * Author: Branden Hitt
    * Purpose: Specification of the CrapUnitElite class
    * Created on December 1, 2015, 1:48 PM
    */

    #ifndef CRAPUNITELITE_H
    #define CRAPUNITELITE_H

    #include "CrapUnit.h"

    class CrapUnitElite : public CrapUnit {
    private:
    string bonus;
    public:
    CrapUnitElite(); //Defaulted Constructor
    CrapUnitElite(string,int,int,string,string,string); //Constructor
    void setBon(string); //Mutator
    string getBon(); //Accessor
    };

    #endif /* CRAPUNITELITE_H */


/*
    * File: CrapUnitElite.h
    * Author: Branden Hitt
    * Purpose: Specification of the CrapUnitElite class
    * Created on December 1, 2015, 1:54 PM
    */
    #include <cstdlib>
    #include "CrapUnitElite.h"

    CrapUnitElite::CrapUnitElite(){
    srand(static_cast<unsigned int>(time(0)));
    int pick=(rand()%5+1);
    if(pick==1){
    name="Young Vampire Elite*";
    health=320;
    dps=25;
    type="Vampire";
    weakness="Paladin";
    bonus="LifeSteal";
    }else if(pick==2){
    name="Dark Rider Elite*";
    health=280;
    dps=30;
    type="Vandal";
    weakness="Ranger";
    bonus="None";
    }else if(pick==3){
    name="Lunatic Elite*";
    health=200;
    dps=50;
    type="Mad Scientist";
    weakness="Warlock";
    bonus="None";
    }else if(pick==4){
    name="Abomination Elite*";
    health=400;
    dps=20;
    type="Frankenstein";
    weakness="Beserker";
    bonus="Dodge Chance";
    }else{
    name="Cultist Elite*";
    health=100;
    dps=120;
    type="Vampire";
    weakness="Any";
    bonus="None";
    }
    }
    CrapUnitElite::CrapUnitElite(string na, int he, int d, string ty, string weak,string bon){
    name=na;
    health=he;
    dps=d;
    type=ty;
    weakness=weak;
    bonus=bon;
    }
    void CrapUnitElite::setBon(string bon){
    bonus=bon;
    }
    string CrapUnitElite::getBon(){
    return bonus;
    }


    /*
    * File: ModerateUnit.h
    * Author: Branden
    * Purpose: Specification for the ModerateUnit class
    * Created on December 3, 2015, 12:42 PM
    */

    #ifndef MODERATEUNIT_H
    #define MODERATEUNIT_H
    #include "EnemyUnit.h"
    class ModerateUnit: public EnemyUnit{
    protected:
    string type;
    string weakness;
    public:
```

```
ModerateUnit(); //Default Constructor
ModerateUnit(string,int,int,string,string); //Constructor
void setType(string); //Mutator
void setWeak(string); //Mutator
string getType(){ return type;} //Accessor (in-line)
string getWeak(){ return weakness;} //Accessor (in-line)
};
#endif /* MODERATEUNIT_H */
```

```
/*
 * File: ModerateUnit.h
 * Author: Branden
 * Purpose: Implementation for the ModerateUnit class
 * Created on December 3, 2015, 12:49 PM
 */
#include "ModerateUnit.h"
#include <cstdlib>
ModerateUnit::ModerateUnit(){
srand(static_cast<unsigned int>(time(0)));
int pick=(rand()%5+1);
if(pick==1){
name="Elder Vampire";
health=260;
dps=45;
type="Vampire";
weakness="Paladin";
}else if(pick==2){
name="Dread Horseman";
health=240;
dps=50;
type="Vandal";
weakness="Ranger";
}else if(pick==3){
name="Maniac";
health=200;
dps=70;
type="Mad Scientist";
weakness="Warlock";
}else if(pick==4){
name="Monstrocity";
health=300;
dps=40;
type="Frankenstein";
weakness="Beserker";
}else{
name="Fanatic";
health=100;
dps=140;
type="Cthulhu Spawn";
```

```
    weakness="Any";
    }
    }
    ModerateUnit::ModerateUnit(string na, int he, int dp, string ty, string we){
    name=na;
    health=he;
    dps=dp;
    type=ty;
    weakness=we;
    }
    void ModerateUnit::setType(string t){
    type=t;
    }
    void ModerateUnit::setWeak(string w){
    weakness=w;
    }
```

```
/*
    * File: ModerateUnitElite.h
    * Author: Branden Hitt
    * Purpose: Specification for the ModerateUnitElite class
    * Created on December 3, 2015, 12:54 PM
    */

    #ifndef MODERATEUNITELITE_H
    #define MODERATEUNITELITE_H
    #include "ModerateUnit.h"
    class ModerateUnitElite : public ModerateUnit{
    private:
    string bonus;
    public:
    ModerateUnitElite(); //Default Constructor
    ModerateUnitElite(string,int,int,string,string); //Constructor
    void setBon(string); //Mutator
    string getBon(){return bonus;} //Accessor (in-line)
    };

    #endif /* MODERATEUNITELITE_H */
```

```
/*
    * File: ModerateUnitElite.h
    * Author: Branden Hitt
    * Purpose: Implementation for the ModerateUnitElite class
    * Created on December 3, 2015, 1:00 PM
    */
    #include "ModerateUnitElite.h"
    #include <cstdlib>

    ModerateUnitElite::ModerateUnitElite(){
```

```
srand(static_cast<unsigned int>(time(0)));
int pick=(rand()%5+1);
if(pick==1){
name="Elder Vampire Elite*";
health=360;
dps=55;
type="Vampire";
weakness="Paladin";
bonus="none";
}else if(pick==2){
name="Dread Horseman Elite*";
health=340;
dps=60;
type="Vandal";
weakness="Ranger";
bonus="none";
}else if(pick==3){
name="Maniac Elite*";
health=300;
dps=80;
type="Mad Scientist";
weakness="Warlock";
bonus="none";
}else if(pick==4){
name="Monstrocity Elite*";
health=400;
dps=50;
type="Frankenstein";
weakness="Beserker";
bonus="Rupture";
}else{
name="Fanatic Elite*";
health=200;
dps=150;
type="Cthulhu Spawn";
weakness="Any";
bonus="none";
}
}
ModerateUnitElite::ModerateUnitElite(string na, int he, int dp, string ty, string bo){
name=na;
health=he;
dps=dp;
type=ty;
weakness="none";
bonus=bo;
}
void ModerateUnitElite::setBon(string bo){
bonus=bo;
}
```

```
/*
    * File: BossUnit.h
    * Author: Branden Hitt
    * Purpose: Specification of the BossUnit class
    * Created on December 3, 2015, 1:06 PM
    */

    #ifndef BOSSUNIT_H
    #define BOSSUNIT_H
    #include "EnemyUnit.h"
    class BossUnit : public EnemyUnit{
    private:
    string bonus;
    string special;
    public:
    BossUnit(); //Default Constructor
    void setBon(string); //Mutator
    void setSpec(string); //Mutator
    string getBon(){return bonus;} //Accessor (in-line)
    string getSpec(){return special;} //Accessor (in-line)
    };

    #endif /* BOSSUNIT_H */


/*
    * File: BossUnit.h
    * Author: Branden Hitt
    * Purpose: Implementation of the BossUnit class
    * Created on December 3, 2015, 1:11 PM
    */
    #include "BossUnit.h"
    #include <cstdlib>
    BossUnit::BossUnit(){
    srand(static_cast<unsigned int>(time(0)));
    int pick=(rand()%5+1);
    if(pick==1){
    name="Nosferatu";
    health=1200;
    dps=70;
    bonus="LifeSteal"; //Steals health from successful attacks
    special="LifeSwap"; //Swaps health with the player at a given percentage
    }else if(pick==2){
    name="Headless Horseman";
    health=1400;
    dps=50;
    bonus="Fear"; //Chance to force player to miss attack
    special="Trample"; //Attacks First
    }else if(pick==3){
    name="Dr. Jeckyl";
    health=1200;
    dps=30;
    bonus="Toxic"; //Chance to lower Max Health permanently
    special="Anger"; //Chance to transform to Mr.Hyde and gain stats
    }else if(pick==4){
    name="Frankenstein's Monster";
    health=1600;
    dps=60;
    bonus="Conduit"; //Chance to shock player for damage when he is hit
    special="none"; //Chance to break the player's weapon, returning it to default stats
    }else{
    name="Cthulhu";
    health=5000;
    dps=300;
    bonus="The Call"; //Chance for the player to go mad and attack themselves
    special="Devour"; //Chance to automatically kill the player
    }
    }
    void BossUnit::setBon(string b){
    bonus=b;
    }
    void BossUnit::setSpec(string s){
    special=s;
    }
```

```
/*
 * File: Player.h
 * Author: Branden Hitt
 * Purpose: Specification of the Player class
 * Created on December 1, 2015, 2:01 PM
 */
#include "Weapon.h"
#include <string>
using namespace std;

#ifndef PLAYER_H
#define PLAYER_H

class Player{
private:
string name;
int health;
int dps;
int selfH;
string special;
int specC;
string arche;
Weapon weapon;
int maxHlth;
int waveCnt;
public:
Player(); //Default Constructor
Player(string,int); //Constructor
Player(string,int,int,string); //Constructor
void setName(string); //Mutator
void setHel(int); //Mutator
void setDps(int); //Mutator
void setSlfH(int); //Mutator
void setSpec(string); //Mutator
void setSpC(int); //Mutator
void setArch(string); //Mutator
void setWepn(Weapon &); //Mutator
void setMaxH(int); //Mutator
void setWave(int); //Mutator
string getName(){ return name;} //Accessor (in-line)
int getHel(){ return health;} //Accessor (in-line)
int getDps(){ return dps;} //Accessor (in-line)
int getSlfH(){ return selfH;} //Accessor (in-line)
string getSpec(){ return special;} //Accessor (in-line)
int getSpC(){ return specC;} //Accessor (in-line)
string getArch(){ return arche;} //Accessor (in-line)
Weapon getWepn(){ return weapon;} //Accessor (in-line)
int getMaxH(){ return maxHlth;} //Accessor (in-line)
int getWave(){ return waveCnt;} //Accessor (in-line)
void takeDam(int); //Take damage function
void addHel(int); //Add health function
void addMaxH(int); //Add max health
int useSpec(int); //Use special attack
void healS(); //Self heal
};

#endif /* PLAYER_H */
```

```
/*
 * File: Player.h
 * Author: Branden Hitt
 * Purpose: Implementation of the Player class
 * Created on December 2, 2015, 11:35 AM
 */
#include <iostream>
#include "Player.h"

Player::Player(){
name="default";
health=0;
dps=0;
arche="default";
weapon=Weapon();
maxHlth=0;
waveCnt=0;
selfH=0;
}
Player::Player(string n,int pick){
if(pick==1){
name=n;
health=500;
dps=70;
arche="Ranger";
weapon=Weapon(1);
special="Headshot";
specC=5;
selfH=60;
}else if(pick==2){
name=n;
health=450;
dps=90;
arche="Warlock";
weapon=Weapon(2);
special="Transfusion";
specC=5;
selfH=50;
}else if(pick==3){
name=n;
health=550;
dps=50;
arche="Paladin";
weapon=Weapon(3);
special="Fortify";
specC=5;
selfH=70;
}else{
name=n;
health=400;
dps=100;
arche="Beserker";
weapon=Weapon(4);
special="Ragnarok";
specC=5;
selfH=40;
}
maxHlth=health;
waveCnt=0;
}
Player::Player(string na, int he, int dp, string ar){
name=na;
health=he;
dps=dp;
arche=ar;
weapon=Weapon();
maxHlth=he;
waveCnt=0;
}
void Player::setName(string na){
name=na;
}
void Player::setHel(int he){
```

```
health=he;
}
void Player::setDps(int d){
dps=d;
}
void Player::setSlfH(int s){
selfH=s;
}
void Player::setSpec(string s){
special=s;
}
void Player::setSpC(int c){
specC=c;
}
void Player::setArch(string a){
arche=a;
}
void Player::setWepn(Weapon &w){
weapon=w;
}
void Player::setMaxH(int m){
maxHlth=m;
}
void Player::setWave(int w){
waveCnt=w;
}
void Player::takeDam(int d){
cout<<"You have been hit for "<<d<<" damage."<<endl;
health-=d;
if(health<0)health=0;
}
void Player::addHel(int add){
health+=add;
if(health>maxHlth) health=maxHlth;
cout<<"You have been healed for "<<add<<endl;
}
void Player::addMaxH(int add){
maxHlth+=add;
cout<<"You have gained +"<<add<<" max health."<<endl;
}
int Player::useSpec(int eneHlth){
if(this->special=="Headshot"){
int halved=eneHlth/2;
eneHlth-=halved;
if(eneHlth<0) eneHlth=0;
cout<<"You have fired a direct hit to the enemy for "<<halved<<" damage!"<<endl;
}else if(this->special=="Transfusion"){
int steal=eneHlth/4;
eneHlth-=steal;
if(eneHlth<0) eneHlth=0;
health+=steal;
if(health>maxHlth) health=maxHlth;
cout<<"You have stolen "<<steal<<" health from the enemy"<<endl;
}else if(this->special=="Fortify"){
int placeH=maxHlth/3;
health+=placeH;
maxHlth+=placeH;
if(health>maxHlth) health=maxHlth;
cout<<"Your max health has increased to "<<maxHlth<<"."<<endl;
cout<<"You have also healed for "<<placeH<<" health."<<endl;
}else{
int half=health/2;
health-=half;
cout<<"You have given up half of your current health"<<endl;
eneHlth=0;
cout<<"The enemy is now dead"<<endl;
}
//decrement special count
specC--;
return eneHlth;
}
void Player::healS(){
health+=selfH;
if(health>maxHlth) health=maxHlth;
}
```

```
/*
    * File: Weapon.h
    * Author: Branden Hitt
    * Purpose: Specification for the Weapon class
    * Created on December 1, 2015, 2:10 PM
    */

    #include <string>
    using namespace std;

    #ifndef WEAPON_H
    #define WEAPON_H

    class Weapon{
    private:
    string wName;
    string wBonus;
    string wType;
    public:
    Weapon(); //Default Constructor
    Weapon(int); //Constructor
    Weapon(string,string,string); //Constructor
    void setWNam(string); //Mutator
    void setWBon(string); //Mutator
    void setWTyp(string); //Mutator
    string getWNam(){return wName;} //Accessor (in-line)
    string getWBon(){return wBonus;}//Accessor (in-line)
    string getWTyp(){return wType;} //Accessor (in-line)
    };
    #endif /* WEAPON_H */
```

```
#include "Weapon.h"
/*
* File: Weapon.h
* Author: Branden Hitt
* Purpose: Implementation for the Weapon class
* Created on December 1, 2015, 2:18 PM
*/
Weapon::Weapon(){
wName="default";
wBonus="none";
wType="default";
}
Weapon::Weapon(int pick){
if(pick==1){
wName="Long Bow";
wType="Ranged";
wBonus="Dodge Chance";
}else if(pick==2){
wName="Staff";
wType="Ranged";
wBonus="LifeSteal";
}else if(pick==3){
wName="Sword";
wType="Melee";
wBonus="First Strike";
}else{
wName="War Axe";
wType="Melee";
wBonus="LifeSteal";
}
}
Weapon::Weapon(string na, string bo, string ty){
wName=na;
wBonus=bo;
wType=ty;
}
void Weapon::setWNam(string na){
wName=na;
}
void Weapon::setWBon(string bo){
wBonus=bo;
}
void Weapon::setWTyp(string ty){
wType=ty;
```

```
    }



/*
    * File: SimpleVector.h
    * Author: Branden Hitt
    * Purpose: specification and implementation of a vector template
    * Created on December 11, 2015, 2:30 PM
    */
    #ifndef SIMPLEVECTOR_H
    #define SIMPLEVECTOR_H
    #include <iostream>
    #include <new> // Needed for bad_alloc exception
    #include <cstdlib> // Needed for the exit function
    using namespace std;
    template <class T>
    class SimpleVector
    {
    private:
    T *aptr; // To point to the allocated array
    int arrSize; // Number of elements in the array
    int alSize; // Size of the allocated array
    void memError(); // Handles memory allocation errors
    void subError(); // Handles subscripts out of range
    public:
    // Default constructor
    SimpleVector()
    { aptr = 0; arrSize = 0; alSize=0;}
    // Constructor declaration
    SimpleVector(int);
    // Copy constructor declaration
    SimpleVector(const SimpleVector &);
    // Destructor declaration
    ~SimpleVector();
    // Accessor to return the array size
    int size() const
    { return arrSize; }
    // Accessor to return a specific element
    T getElementAt(int position);
    // Overloaded [] operator declaration
    T &operator[](const int &);
    //push function
    void pushEle(T);
    };
    //************************************************************
    // Constructor for SimpleVector class. Sets the size of the *
    // array and allocates memory for it. *
    //***********************************************************

    template <class T>
    SimpleVector<T>::SimpleVector(int s)
    {
    arrSize = s;
    alSize= 2*s;
    // Allocate memory for the array.
    try
    {
    aptr = new T [alSize];
    }
    catch (bad_alloc)
    {
    memError();
    }

    // Initialize the array.
    for (int count = 0; count < arrSize; count++)
    *(aptr + count) = 0;
    }

    //*******************************************
    // Copy Constructor for SimpleVector class. *
    //*******************************************

    template <class T>
    SimpleVector<T>::SimpleVector(const SimpleVector &obj)
    {
    // Copy the array size.
    arrSize = obj.arrSize;

    // Allocate memory for the array.
    aptr = new T [arrSize];
    if (aptr == 0)
    memError();

    // Copy the elements of obj's array.
    for(int count = 0; count < arrSize; count++)
    *(aptr + count) = *(obj.aptr + count);
    }

    //**************************************
    // Destructor for SimpleVector class. *
    //**************************************

    template <class T>
    SimpleVector<T>::~SimpleVector()
    {
    if (arrSize > 0)
    delete [] aptr;
    }

    //*********************************************************
    // memError function. Displays an error message and *
    // terminates the program when memory allocation fails. *
```

```
//*******************************************************
template <class T>
void SimpleVector<T>::memError()
{
cout << "ERROR:Cannot allocate memory.\n";
exit(EXIT_FAILURE);
}
//***********************************************************
// subError function. Displays an error message and *
// terminates the program when a subscript is out of range. *
//***********************************************************
template <class T>
void SimpleVector<T>::subError()
{
cout << "ERROR: Subscript out of range.\n";
exit(EXIT_FAILURE);
}
//*******************************************************
// getElementAt function. The argument is a subscript. *
// This function returns the value stored at the sub- *
// cript in the array. *
//*******************************************************
template <class T>
T SimpleVector<T>::getElementAt(int sub)
{
if (sub < 0 || sub >= arrSize)
subError();
return aptr[sub];
}
//*******************************************************
// Overloaded [] operator. The argument is a subscript. *
// This function returns a reference to the element *
// in the array indexed by the subscript. *
//*******************************************************
template <class T>
T &SimpleVector<T>::operator[](const int &sub)
{
if (sub < 0 || sub >= arrSize)
subError();
return aptr[sub];
}
//*******************************************************
// Push function *
// *
// *
//*******************************************************
template <class T>
void SimpleVector<T>::pushEle(T newEle){
```

```
//check the allocated size
if(alSize<=(arrSize+1)){
//double allocated size
alSize*=2;
// Copy the array size.
arrSize+=1;

// Allocate memory for the array.
T *nAptr = new T [alSize];
cout<<"New allocation"<<endl;
// Copy the elements of obj's array.
for(int count = 0; count < arrSize; count++)
*(nAptr + count) = *(aptr + count);

//Add the new element at the end
*(nAptr + (arrSize-1))=newEle;

//delete old array
delete []aptr;

//set pointer to new array
aptr=nAptr;
}else{
//add new element
*(aptr + arrSize)= newEle;
//increment array size
arrSize++;
}

}

#endif /* SIMPLEVECTOR_H */
```