Text-based RPG

## Vincent Alexander Seliang

## 2001585562

Simple Adventure

**Section 1**

**Description** This application is, pretty much self-explanatory, a text-based RPG. It follows the style of the one of the earliest RPGs (the 1980’s style) that does not use any, or at least use a minimal amount of graphical game objects. Instead, these types of video games heavily depend on text, obviously, and some newer version of those types of RPGs utilizes the ASCII characters to form interface and HUD (Head-Up Display) for visualization purpose. However, this project takes a different angle of RPG. Instead of making a rogue-like style like the one made in the 1980’s (it is a game genre, where the player is required to explore a certain area in order to complete objective, also it is usually implemented as a subgenre of RPG), it took the path of one of the earlier predecessor of RPG, which is the Dungeons and Dragons series, which were first publish in the early 1970’s. Dungeons and Dragons is a non-digital version of RPG that requires three or more players to participate the game, where one of them is the Dungeon Master, which is the player who sets the setting of the game and the one who decides the fate of the other players based on the number that appears on the dices. However, this game does only require only one player and have a mixture of linear and slight branched progression, inspired from the newer, digital version RPGs. However, the aspect of having multiple choices and the random chance of events’ occurrence still exists and it is also the main aspect of the game. Also, the aspect of battle and role –playing also exist.

**Section 2** 2A

Figure 1, Flowchart of how the game flows

Important notes:

* In order to play the game, players have to follow the instructions displayed on the screen. If there are a list of actions can be done, usually, the player has to input the integer that matches to the choice’s number in order to proceed. Failing to do so might cause the game to fail to operate.

However, if there are only two choices, the player can still input an integer such as one and zero, but the player is also allowed to input words, such as “true” and “yes”.

* The first step, opening, handles the choice of player, either the player can choose to play the game or simply exit the program.
* The second step, player creation, instructs the player to choose their avatar’s race and a job that will be used to initializing the avatar’s status and starting equipment.
* The final step, story, handles on the game’s story flow, where it will allow the player to choose their action with a slight element of Dungeons and Dragons. This includes managing the storyline and processing the battles and loots.

Figure 2.1.1, Class diagrams (Player), simplified

Figure 2.1.2, Class diagrams (Player), UML diagram

|  |
| --- |
| **Player** |
| * sPlayerName : string * nHealth : int = 0 * nMana : int = 0 * nCurrency : int = 500 * nJob : int * nRace : int * nLevel : int = 1 * nEXP : int = 0 * nAttack : int = 0 * nDefence : int = 0 * nMagicAttack : int = 0 * nMagicDefence : int = 0 * nAgility : int = 0 * nLuck : int = 0 * nWeapon : Weapon * nArmor : Armor |
| * Player(job : int, race : int, health : int, mana : int, attack : int, defence : int, magic\_attack : int, magic\_defence : int, agility : int, luck : int, weapon : Weapon\*, weaponnum : int, armor : Armor\*, armornum : int) * Player() * SetStatus(health : int, mana : int, attack : int, defence : int, magic\_attack : int, magic\_defence : int, agility : int, luck) : void * JobIncrementer(: int, health : int, mana : int, attack : int, defence : int, magic\_attack : int, magic\_defence : int, agility : int, luck : int, weapon : Weapon\*, weaponnum : int, armor : Armor\*, armornum : int) : void * SetName(name : string) : void * SetRace(race : int) : void * SetJob(job : int) : void * SetWeapon(weapon : int) : void * SetArmor(armor : int) : void * GetName() : string * GetLevel() : int * GetHealth() : int * GetMana() : int * GetAttack() : int * GetDefenec() : int * GetMagicAttack() : int * GetMagicDefence() : int * GetAgility() : int * GetLuck() : int * GetJob() : int * GetData() : void * Leveling(exp : int) : int |

Figure 2.2.1, Class Diagram (Enemy), simplified

Figure 2.2.2, Class Diagram (Enemies), UML diagram

|  |
| --- |
| **Enemy** |
| * sEName : string * nEHealth : int * nEMana : int * nEAttack : int * nEDefence : int * nEMagicAttack : int * nEMagicDefence : int * nEAgility : int * nELuck : int * nEXPGain : int |
| * Enemy(name : string, health : int, mana : int, attack : int, defence : int, magic\_attack : int, magic\_defence : int, agility : int, luck : int) * Enemy() * eGetName() : string * eGetHealth() : int * eGetMana() : int * eGetAttack() : int * eGetDefence() : int * eGetMagicAttack() : int * eGetMagicDefence() : int * eGetAgilty() : int * eGetLuck() : int * eGetEXP() : int |

Figure 2.3.1, Class Diagram (Weapon), simplified

Figure 2.3.2, Class Diagram (Weapon), UML diagram

|  |
| --- |
| **Weapon** |
| * nWName : string * nWHealth : int * nWMana : int * nWAttack : int * nWDefence : int * nWMagicAttack : int * nWMagicDefence : int * nWAgility : int * nWLuck : int |
| * Weapon(name : string, health : int, mana : int, attack : int, defence : int, magic\_attack : int, magic\_defence : int, agility : int, luck : int) * Weapon() * wGetName() : string * wGetAttack() : int * wGetMagicAttack() : int |

Figure 2.4.1, Class Diagram (Armor), simplified

Figure 2.4.2, Class Diagram (Armor), UML diagram

|  |
| --- |
| **Armor** |
| * sAName : string * nAHealth : int * nAMana : int * nAAttack : int * nADefence : int * nAMagicAttack : int * nAMagicDefence : int * nAAgility : int * nALuck : int |
| * Armor(name : string, health : int, mana : int, attack : int, defence : int, magic\_attack : int, magic\_defence : int, agility : int, luck : int) * Armor() * aGetName() : string * aGetDefense() : int * aGetMagicDefense() : int |

Figure 2.5.1, Class Diagram (InvalidRace), simplified

Figure 2.6.1, Class Diagram (InvalidJob), simplified

Note: class of figure 2.5 and figure 2.6 are used for exception handling.

Figure 2.5.2, Class Diagram (InvalidRace), UML diagram

|  |
| --- |
| **InvalidRace** |
| “Empty class”  Only for exception handling |

Figure 2.6.2, Class Diagram (InvalidJob), UML diagram

|  |
| --- |
| **InvalidJob** |
| “Empty class”  Only for exception handling |

2B

**Classes**

* Class Player This class is used to create and initialize the player’s status, such its race and job, which will be used to give the player’s initial status at level 1.
* Class Enemy This class is created for making the enemies that the player will face along the way. They are also the ones who will provide the player with EXP.
* Class Weapon and Class Armor

These classes are used to create weapons and armors that can be used by the player to aid them during battle. Weapon provides extra damage, while the armor dampens the incoming attack from the elements, such as enemies.

* Class InvalidJob and Class InvalidRace

These classes are used for error handling, the exception feature. Those objects are used for throwing and catching part.

**Functions**

* void BattleSystem(Player &, Enemy &, Weapon &, Armor &)

This function is used to handle whatever is happening during encountering with an enemy. This includes attacking (conventional method), talking to it (pacifist method), using a potion to heal the player, and escape if the player does not or not able to fight it. The parameter pointer to Player and Enemy uses to accept any Player and Enemy objects.

* + void PlayerTurn(Player &, Enemy &) and void EnemyTurn(Enemy &, Player &)

This nested function of the battle system function is used to handle the calculations of the amount of damage inflicted on either side and calculate the probability of dodging incoming attacks.

* + void SpecialMove(Player &, Enemy &, Weapon &, int)

This nested function of battle system function is used to control the calculation of damage inflicted based on the player’s weapon.

* string DisplayArt(ifstream&)

This function is use to both store the text file and displaying the content of the file on the console. It is used only at the opening of the game. The parameter ifstream is used to accept any variable that does input.

* void Opening()

This function is used to emulate and opening menu for like most video games, where is display the ASCII art and the option whether to play the game of exit the game.

* string DisplayWeaponName(int) and string DisplayArmorName(int)

These functions are used to display the names of the equipment because class method, for some reason, could not access the arrays of object’s content. The parameter int is used for accessing the array’s index.

* void PlayerCreation()

This function is used to let the player choose their race and job to initialize their status and their starting weapon and armors.

* void Story()

This function is used to handle all the narrative parts of the video game. It also holds several nested functions that always start with the word Choice such as ChoiceOne() and ChoiceTwo().

* + Choice…()

These nested functions are used to hold the result of the player’s action, results of it could slightly vary.

* int main()

This function is essential for all C++ programs to run and, in this case, it is used to run some functions that are essential for the game to play

* void end()

This function is used for playing the ending part of the game.

**Section 3** 3A

The lessons obtained while making this project:

1. Making function’s prototypes are important because, with the usage of prototypes, the programmer does not need to rearrange to functions into the right position due to it is already declared before.
2. Making the game or program into the MCV (Model Control View) form is actually quite efficient compared to combining them into one huge file because it allows easier implementation for reusability. However, my program only has model and view due to having a control tends to cause multiple errors.

Note: this problem has been resolved by moving only the global functions into a separate header file.

1. It is impossible to directly declare a function and its content within another function, which could cause an error when it is done.
2. Making a get-set method for each class can be very useful, which can help in requiring just a single value from each property of the object.
3. Switch conditional is sometimes more efficient than the if-else conditional for receiving and evaluation an equivalence statement, the == statement.
4. A class method, for some reason, is not able to retrieve a part of object’s properties if the object is a part of an array.
5. Error-handling (exceptions) works almost similar to if-else conditional, if it is taken to its most basic level, but only up to a certain extent.
6. Using the fstream library is really useful, especially when it is used for storing a game object that has a high chance to be changed on a regular basis, such as ASCII arts.

3B

The problems encountered and how it was overcome while making this script:

1. It was, somehow, impossible to access an object’s property if it is a part of an array from the class’ method. In order to overcome that problem, using an external function, the function that is not a class method, is the excellent method for accessing the object’s property in an array.
2. Making and ASCII art is difficult and changing would require a lot of time. This can be solved by using the fstream library, which allows easy modification simply by changing the content of the text file.
3. Sometimes, a function won’t execute because it was not declared before the other function that is going to execute it. However, this can be solved by using function prototyping.
4. Instead of using multiple arrays that hold several values for storing the values of the game objects value, using the class to create an object can be a better alternative because the data within the object are connected, which mean accessing an object property would be more efficient than accessing multiple arrays to find each property.
5. The value of the avatar’s property does not increment, but it got change whenever the method JobIncrementer was called. It got fixed after the addition symbol, +, was added before the assignment symbol, =.
6. When the object was created, it shows an error saying it requires some parameters where it is clearly there is no need of using one. This was fixed by overloading another constructor that does not take any parameter.

Section 4

The script in MCV form

**File Name: Final\_project\_view.cpp**

1. #include <iostream>
2. #include <fstream>
3. #include <cstdlib>
4. #include <cstdio>
5. #include <string>
6. #include <cstring>
7. #include <chrono>
8. #include <cmath>
9. #include <sstream>
10. #include <iomanip>
11. #include <algorithm>
12. #include <vector>
13. #include <array>
14. #include "Final\_project\_model.h"
15. #include "Final\_project\_controller.h"
16. using namespace std;
17. int main()
18. {
    1. Opening();
    2. Story();
    3. system("PAUSE");
    4. return 0;
19. }

**File Name: Final\_project\_model.h**

1. #pragma once
2. #include <iostream>
3. #include <fstream>
4. #include <cstdlib>
5. #include <cstdio>
6. #include <string>
7. #include <cstring>
8. #include <chrono>
9. #include <cmath>
10. #include <sstream>
11. #include <iomanip>
12. #include <algorithm>
13. #include <vector>
14. #include <array>
15. using namespace std;
16. //class for exception
17. class InvalidRace
18. {};
19. class InvalidJob
20. {};
21. //Initializing class for Weapon
22. class Weapon
23. {
24. private:
25. string nWName;
26. int nWHealth;
27. int nWMana;
28. int nWAttack;
29. int nWDefence;
30. int nWMagicAttack;
31. int nWMagicDefence;
32. int nWAgility;
33. int nWLuck;
34. public:
35. Weapon(string name, int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck)
36. {
37. nWName = name;
38. nWHealth = health;
39. nWMana = mana;
40. nWAttack = attack;
41. nWDefence = defence;
42. nWMagicAttack = magic\_attack;
43. nWMagicDefence = magic\_defence;
44. nWAgility = agility;
45. nWLuck = luck;
46. }
47. Weapon()
48. {
49. nWName = "";
50. nWHealth = 0;
51. nWMana = 0;
52. nWAttack = 0;
53. nWDefence = 0;
54. nWMagicAttack = 0;
55. nWMagicDefence = 0;
56. nWAgility = 0;
57. nWLuck = 0;
58. }
59. string wGetName()
60. {
61. return nWName;
62. }
63. int wGetAttack()
64. {
65. return nWAttack;
66. }
67. int wGetMagicAttack()
68. {
69. return nWMagicAttack;
70. }
71. };
72. //Weapon(string name, int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck)
73. Weapon aWeapon[15] = {
74. Weapon("Cheap Sword", 0, 0, 4, 1, 0, 0, 0, 0),
75. Weapon("Decent Sword", 0, 0, 5, 1, 0, 0, 0, 0),
76. Weapon("Good Sword", 0, 0, 7, 2, 0, 0, 0, 0),
77. Weapon("Cheap Staff", 0, 5, 1, 0, 2, 1, 0, 0),
78. Weapon("Decent Staff", 0, 6, 1, 0, 3, 1, 0, 0),
79. Weapon("Good Staff", 0, 8, 2, 0, 5, 2, 0, 0),
80. Weapon("Cheap Gun", 0, 0, 3, 0, 0, 0, 2, 0),
81. Weapon("Decent Gun", 0, 0, 4, 0, 0, 0, 2, 0),
82. Weapon("Good Gun", 0, 0, 6, 0, 0, 0, 4, 0),
83. Weapon("Cheap Dagger", 0, 0, 1, 0, 0, 0, 2, 2),
84. Weapon("Decent Dagger", 0, 0, 2, 0, 0, 0, 3, 2),
85. Weapon("Good Dagger", 0, 0, 4, 0, 0, 0, 5, 3),
86. Weapon("Cheap Whip", 0, 0, 3, 0, 1, 0, 1, 3),
87. Weapon("Decent Whip", 0, 0, 4, 0, 2, 0, 2, 3),
88. Weapon("Good Whip", 0, 0, 5, 0, 3, 0, 3, 5),
89. };
90. Weapon weapon;
91. string DisplayWeaponName(int index)
92. {
93. return aWeapon[index].wGetName();
94. }
95. //Initializing class for Armor
96. class Armor
97. {
98. private:
99. string sAName;
100. int nAHealth;
101. int nAMana;
102. int nAAttack;
103. int nADefence;
104. int nAMagicAttack;
105. int nAMagicDefence;
106. int nAAgility;
107. int nALuck;
108. public:
109. Armor(string name, int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck)
110. {
111. sAName = name;
112. nAHealth = health;
113. nAMana = mana;
114. nAAttack = attack;
115. nADefence = defence;
116. nAMagicAttack = magic\_attack;
117. nAMagicDefence = magic\_defence;
118. nAAgility = agility;
119. nALuck = luck;
120. }
121. Armor()
122. {
123. sAName = "";
124. nAHealth = 0;
125. nAMana = 0;
126. nAAttack = 0;
127. nADefence = 0;
128. nAMagicAttack = 0;
129. nAMagicDefence = 0;
130. nAAgility = 0;
131. nALuck = 0;
132. }
133. string aGetName()
134. {
135. return sAName;
136. }
137. int aGetDefense()
138. {
139. return nADefence;
140. }
141. int aGetMagicDefense()
142. {
143. return nAMagicDefence;
144. }
145. };
146. //Armor(string name, int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck)
147. Armor aArmor[15] = {
148. Armor("Cheap Mail", 20, 0, 0, 4, 0, 0, -1, 0),
149. Armor("Decent Mail", 20, 0, 0, 5, 0, 0, -2, 0),
150. Armor("Good Mail", 40, 0, 0, 8, 0, 0, -2, 0),
151. Armor("Cheap Robe", 0, 20, 0, 1, 1, 2, 0, 0),
152. Armor("Decent Robe", 0, 20, 0, 1, 2, 3, 0, 0),
153. Armor("Good Robe", 0, 40, 0, 3, 4, 5, 0, 0),
154. Armor("Cheap Jumpsuit", 0, 0, 0, 2, 0, 2, 1, 1),
155. Armor("Decent Jumpsuit", 0, 0, 0, 3, 0, 2, 2, 2),
156. Armor("Good Jumpsuit", 0, 0, 0, 4, 0, 2, 2, 2),
157. Armor("Cheap Garb", 0, 0, 0, 2, 0, 0, 2, 3),
158. Armor("Decent Garb", 0, 0, 0, 2, 0, 0, 4, 5),
159. Armor("Good Garb", 0, 0, 0, 3, 0, 0, 8, 7),
160. Armor("Cheap Noble Suit", 0, 0, 1, 2, 1, 0, 0, 2),
161. Armor("Decent Noble Suit", 0, 0, 0, 1, 3, 0, 0, 2),
162. Armor("Good Noble Suit", 0, 0, 0, 3, 5, 2, 0, 6)
163. };
164. Armor armor;
165. string DisplayArmorName(int index)
166. {
167. return aArmor[index].aGetName();
168. }
169. //Initializing class for the player
170. class Player
171. {
172. private:
173. string sPlayerName;
174. int nHealth = 0;
175. int nMana = 0;
176. int nCurrency = 500;
177. int nJob;
178. int nRace;
179. int nLevel = 1;
180. int nEXP = 0;
181. int nAttack = 0;
182. int nDefence = 0;
183. int nMagicAttack = 0;
184. int nMagicDefence = 0;
185. int nAgility = 0;
186. int nLuck = 0;
187. Weapon nWeapon;
188. Armor nArmor;
189. public:
190. Player(int job, int race, int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck, Weapon \*weapon, int weaponnum, Armor \*armor, int armornum)
191. {
192. nJob = job;
193. nRace = race;
194. nHealth = health;
195. nMana = mana;
196. nAttack = attack;
197. nDefence = defence;
198. nMagicAttack = magic\_attack;
199. nMagicDefence = magic\_defence;
200. nAgility = agility;
201. nLuck = luck;
202. nWeapon = \*(weapon + weaponnum);
203. nArmor = \*(armor + armornum);
204. }
205. Player()
206. {
207. nJob = 0;
208. nRace = 0;
209. nHealth = 0;
210. nMana = 0;
211. nAttack = 0;
212. nDefence = 0;
213. nMagicAttack = 0;
214. nMagicDefence = 0;
215. nAgility = 0;
216. nLuck = 0;
217. nWeapon = aWeapon[0];
218. nArmor = aArmor[0];
219. }
220. void SetStatus(int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck)
221. {
222. nHealth += health;
223. nMana += mana;
224. nAttack += attack;
225. nDefence += defence;
226. nMagicAttack += magic\_attack;
227. nMagicDefence += magic\_defence;
228. nAgility += agility;
229. nLuck += luck;
230. return;
231. }
232. void JobIncrementer(int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck, Weapon \*weapon, int weaponnum, Armor \*armor, int armornum)
233. {
234. nHealth += health;
235. nMana += mana;
236. nAttack += attack;
237. nDefence += defence;
238. nMagicAttack += magic\_attack;
239. nMagicDefence += magic\_defence;
240. nAgility += agility;
241. nLuck += luck;
242. nWeapon = \*(weapon + weaponnum);
243. nArmor = \*(armor + armornum);
244. return;
245. }
246. void SetName(string name)
247. {
248. sPlayerName = name;
249. return;
250. }
251. void SetRace(int race)
252. {
253. nRace = race;
254. return;
255. }
256. void SetJob(int job)
257. {
258. nJob = job;
259. return;
260. }
261. void SetWeapon(int weapon)
262. {
263. \*(aWeapon + weapon);
264. return;
265. }
266. void SetArmor(int armor)
267. {
268. \*(aArmor + armor);
269. return;
270. }
271. string GetName()
272. {
273. return sPlayerName;
274. }
275. int GetLevel()
276. {
277. return nLevel;
278. }
279. int GetHealth()
280. {
281. return nHealth;
282. }
283. int GetMana()
284. {
285. return nMana;
286. }
287. int GetAttack()
288. {
289. return nAttack;
290. }
291. int GetDefence()
292. {
293. return nDefence;
294. }
295. int GetMagicAttack()
296. {
297. return nMagicAttack;
298. }
299. int GetMagicDefence()
300. {
301. return nMagicDefence;
302. }
303. int GetAgility()
304. {
305. return nAgility;
306. }
307. int GetLuck()
308. {
309. return nLuck;
310. }
311. int GetAJob()
312. {
313. return nJob;
314. }
315. void GetData()
316. {
317. cout << "Health: " << nHealth << endl;
318. cout << "Mana: " << nMana << endl;
319. cout << "Attack: " << nAttack << endl;
320. cout << "Defence: " << nDefence << endl;
321. cout << "Magic Attack: " << nMagicAttack << endl;
322. cout << "Magic Defence: " << nMagicDefence << endl;
323. cout << "Agility: " << nAgility << endl;
324. cout << "Luck: " << nLuck << endl;
325. return;
326. }
327. void Leveling(int exp)
328. {
329. nEXP += exp;
330. int LevelUp = (nLevel \* 2) + 50;
331. if (nEXP >= LevelUp)
332. {
333. cout << "Level up!" << endl;
334. cout << "You leveled up from " << nLevel << " to " << nLevel << "!" << endl;
335. nLevel++;
336. if (nRace == 1)
337. {
338. nHealth += 3;
339. nMana += 2;
340. nAttack += 2;
341. nDefence += 2;
342. nMagicAttack += 2;
343. nMagicDefence += 2;
344. nAgility += 2;
345. nLuck += 2;
346. }
347. else if (nRace == 2)
348. {
349. nHealth += 1;
350. nMana += 4;
351. nAttack += 1;
352. nDefence += 1;
353. nMagicAttack += 3;
354. nMagicDefence += 3;
355. nAgility += 2;
356. nLuck += 3;
357. }
358. else if (nRace == 3)
359. {
360. nHealth += 3;
361. nMana += 1;
362. nAttack += 3;
363. nDefence += 2;
364. nMagicAttack += 1;
365. nMagicDefence += 1;
366. nAgility += 1;
367. nLuck += 2;
368. }
369. else if (nRace == 4)
370. {
371. nHealth += 5;
372. nMana += 1;
373. nAttack += 4;
374. nDefence += 4;
375. nMagicAttack += 1;
376. nMagicDefence += 1;
377. nAgility += 2;
378. nLuck += 1;
379. }
380. else
381. {
382. nHealth += 1;
383. nMana += 3;
384. nAttack += 2;
385. nDefence += 1;
386. nMagicAttack += 2;
387. nMagicDefence += 2;
388. nAgility += 4;
389. nLuck += 4;
390. }
391. nEXP = 0;
392. }
393. return;
394. }
395. };
396. //Reminder about race variable:
397. //1 - Human
398. //2 - Elf
399. //3 - Dwarf
400. //4 - Orc
401. //5 - Halfling
402. //Reminder about job variable:
403. //1 - Knight
404. //2 - Mage
405. //3 - Mechanic
406. //4 - Thief
407. //5 - Noble
408. Player player;
409. //Initializing class for enemies
410. class Enemy
411. {
412. private:
413. string sEName;
414. int nEHealth;
415. int nEMana;
416. int nEAttack;
417. int nEDefence;
418. int nEMagicAttack;
419. int nEMagicDefence;
420. int nEAgility;
421. int nELuck;
422. int nEXPGain;
423. public:
424. Enemy(string name, int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck, int exp)
425. {
426. this->sEName = name;
427. this->nEHealth = health;
428. this->nEMana = mana;
429. this->nEAttack = attack;
430. this->nEDefence = defence;
431. this->nEMagicAttack = magic\_attack;
432. this->nEMagicDefence = magic\_defence;
433. this->nEAgility = agility;
434. this->nELuck = luck;
435. this->nEXPGain = exp;
436. }
437. Enemy()
438. {
439. this->sEName = "";
440. this->nEHealth = 0;
441. this->nEMana = 0;
442. this->nEAttack = 0;
443. this->nEDefence = 0;
444. this->nEMagicAttack = 0;
445. this->nEMagicDefence = 0;
446. this->nEAgility = 0;
447. this->nELuck = 0;
448. this->nEXPGain = 0;
449. }
450. string eGetName()
451. {
452. return sEName;
453. }
454. int eGetHealth()
455. {
456. return nEHealth;
457. }
458. int eGetMana()
459. {
460. return nEMana;
461. }
462. int eGetAttack()
463. {
464. return nEAttack;
465. }
466. int eGetDefence()
467. {
468. return nEDefence;
469. }
470. int eGetMagicAttack()
471. {
472. return nEMagicAttack;
473. }
474. int eGetMagicDefence()
475. {
476. return nEMagicDefence;
477. }
478. int eGetAgility()
479. {
480. return nEAgility;
481. }
482. int eGetLuck()
483. {
484. return nELuck;
485. }
486. int eGetEXP()
487. {
488. return nEXPGain;
489. }
490. };
491. //Enemy(int health, int mana, int attack, int defence, int magic\_attack, int magic\_defence, int agility, int luck, int exp)
492. Enemy enemy[10] = {
493. Enemy("Slime", 50, 10, 6, 6, 6, 4, 4, 2, 6),
494. Enemy("Bat", 40, 20, 8, 6, 3, 3, 6, 2, 6),
495. Enemy("Snake", 60, 10, 8, 8, 2, 2, 6, 4, 10),
496. Enemy("Gazer", 100, 30, 4, 4, 10, 4, 4, 4, 15),
497. Enemy("Bandit", 150, 20, 12, 7, 3, 5, 6, 2, 20),
498. Enemy("Troll", 300, 20, 18, 15, 6, 6, 6, 7, 30),
499. Enemy("Goblin", 100, 30, 6, 6, 6, 6, 10, 6, 10),
500. Enemy("Skeleton", 150, 30, 10, 15, 3, 3, 3, 3, 20),
501. Enemy("Wolf", 145, 15, 8, 4, 4, 2, 12, 3, 20),
502. Enemy("Dragon", 500, 70, 20, 20, 15, 10, 8, 5, 100)
503. };

**File Name: Final\_project\_controller.h**

1. #pragma once
2. #include <iostream>
3. #include <fstream>
4. #include <cstdlib>
5. #include <cstdio>
6. #include <string>
7. #include <cstring>
8. #include <chrono>
9. #include <cmath>
10. #include <sstream>
11. #include <iomanip>
12. #include <algorithm>
13. #include <vector>
14. #include <array>
15. #include "Final\_project\_model.h"
16. using namespace std;
17. //Initializing potions
18. int nPotion = 10;
19. //Function prototypes
20. void Opening();
21. int PlayerTurn(Player& ppTurn, Enemy& enemy, Weapon& weapon);
22. int EnemyTurn(Enemy& peTurn, Player& player, Armor& armor);
23. int SpecialMove(Player& ppTurn, Enemy& enemy, Weapon& weapon, int weapon\_num);
24. void BattleSystem(Enemy& enemy, Player& player, Weapon& weapon, Armor& armor);
25. void Story();
26. void PlayerCreation();
27. void ChoiceOne();
28. void ChoiceOneOne();
29. void ChoiceTwo();
30. void ChoiceTwoOne();
31. void End();
32. string DisplayArt(ifstream& File);
33. //Function for taking in ASCII art
34. string DisplayArt(ifstream& File)
35. {
36. string Lines = "";
37. if (File)
38. {
39. while (File.good())
40. {
41. string TempLine;
42. getline(File, TempLine);
43. TempLine += "\n";
44. Lines += TempLine;
45. }
46. return Lines;
47. }
48. else
49. {
50. return "File does not exist.";
51. }
52. }
53. //Function to create opening scene
54. void Opening()
55. {
56. ifstream inputFile;
57. inputFile.open("opening.txt");
58. string sTitleArt;
59. sTitleArt = DisplayArt(inputFile);
60. inputFile.close();
61. for (int i = 0; i <= 119; i++)
62. {
63. cout << "\*";
64. }
65. cout << endl;
66. cout << sTitleArt << endl;
67. for (int i = 0; i <= 119; i++)
68. {
69. cout << "\*";
70. }
71. cout << endl;
72. int nChoice;
73. cout << " Input your choice in integers" << endl;
74. cout << " 1. Start your adventure" << endl;
75. cout << " 2. Exit the game" << endl;
76. cin >> nChoice;
77. cin.ignore();
78. switch (nChoice)
79. {
80. case 1:
81. cout << "Have fun!" << endl << endl;
82. PlayerCreation();
83. break;
84. case 2:
85. cout << "See you later!" << endl;
86. system("PAUSE");
87. exit(0);
88. break;
89. default:
90. cout << "See you later!" << endl;
91. system("PAUSE");
92. exit(0);
93. break;
94. }
95. return;
96. }
97. //Function to create the player
98. int nEquipmentIndex;
99. void PlayerCreation()
100. {
101. cout << "Welcome to the your Simple Adventure!" << endl;
102. cout << "Before we continue, can you please tell your name?" << endl;
103. string sPlaceName;
104. getline(cin, sPlaceName);
105. player.SetName(sPlaceName);
106. cout << endl << "What are you?" << endl;
107. cout << "1.Human" << setw(10) << "2.Elf" << setw(10) << "3.Dwarf" << setw(10) << "4.Orc" << setw(15) << "5.Halfling" << endl;
108. cout << "Enter 1-5." << endl;
109. int nPlaceRace;
110. cin >> nPlaceRace;
111. player.SetRace(nPlaceRace);
112. if (nPlaceRace < 1 || nPlaceRace > 5)
113. {
114. throw InvalidRace();
115. }
116. try
117. {
118. if (nPlaceRace == 1)
119. {
120. cout << player.GetName() << " the human." << endl;
121. player.SetStatus(100, 50, 5, 5, 5, 5, 5, 5);
122. player.GetData();
123. }
124. else if (nPlaceRace == 2)
125. {
126. cout << player.GetName() << " the elf." << endl;
127. player.SetStatus(70, 120, 2, 3, 8, 8, 8, 7);
128. player.GetData();
129. }
130. else if (nPlaceRace == 3)
131. {
132. cout << player.GetName() << " the dwarf." << endl;
133. player.SetStatus(120, 40, 7, 5, 3, 2, 4, 5);
134. player.GetData();
135. }
136. else if (nPlaceRace == 4)
137. {
138. cout << player.GetName() << " the orc." << endl;
139. player.SetStatus(150, 20, 10, 8, 1, 2, 6, 3);
140. player.GetData();
141. }
142. else
143. {
144. cout << player.GetName() << " the halfling." << endl;
145. player.SetStatus(50, 80, 6, 3, 4, 6, 10, 7);
146. player.GetData();
147. }
148. }
149. catch (InvalidRace())
150. {
151. cout << "Race not found, exiting program..." << endl;
152. system("PAUSE");
153. exit(0);
154. }
155. catch (...)
156. {
157. cerr << "Invalid error, exiting program..." << endl;
158. system("PAUSE");
159. exit(0);
160. }
161. cout << endl << "What is your job?" << endl;
162. cout << "1.Knight" << setw(10) << "2.Mage" << setw(15) << "3.Mechanic" << setw(10) << "4.Thief" << setw(10) << "5.Noble" << endl;
163. cout << "Enter 1-5." << endl;
164. int nPlaceJob;
165. cin >> nPlaceJob;
166. player.SetJob(nPlaceJob);
167. if (nPlaceJob < 1 || nPlaceJob > 5)
168. {
169. throw InvalidJob();
170. }
171. try
172. {
173. if (nPlaceJob == 1)
174. {
175. nEquipmentIndex = 0;
176. cout << player.GetName() << " the knight." << endl;
177. player.JobIncrementer(50, 10, 3, 3, 1, 1, -2, 2, aWeapon, 0, aArmor, 0);
178. weapon = aWeapon[0];
179. armor = aArmor[0];
180. string WeaponName = aWeapon[nEquipmentIndex].wGetName();
181. string ArmorName = aArmor[nEquipmentIndex].aGetName();
182. cout << "Weapon: " << WeaponName << endl;
183. cout << "Armor: " << ArmorName << endl;
184. player.GetData();
185. }
186. else if (nPlaceJob == 2)
187. {
188. nEquipmentIndex = 3;
189. cout << player.GetName() << " the mage." << endl;
190. player.JobIncrementer(10, 50, 2, 2, 3, 3, 0, 2, aWeapon, 3, aArmor, 3);
191. weapon = aWeapon[3];
192. armor = aArmor[3];
193. string WeaponName = aWeapon[nEquipmentIndex].wGetName();
194. string ArmorName = aArmor[nEquipmentIndex].aGetName();
195. cout << "Weapon: " << WeaponName << endl;
196. cout << "Armor: " << ArmorName << endl;
197. player.GetData();
198. }
199. else if (nPlaceJob == 3)
200. {
201. nEquipmentIndex = 6;
202. cout << player.GetName() << " the mechanic." << endl;
203. player.JobIncrementer(30, 30, 1, 2, 1, 2, -3, -1, aWeapon, 6, aArmor, 6);
204. weapon = aWeapon[6];
205. armor = aArmor[6];
206. string WeaponName = aWeapon[nEquipmentIndex].wGetName();
207. string ArmorName = aArmor[nEquipmentIndex].aGetName();
208. cout << "Weapon: " << WeaponName << endl;
209. cout << "Armor: " << ArmorName << endl;
210. player.GetData();
211. }
212. else if (nPlaceJob == 4)
213. {
214. nEquipmentIndex = 9;
215. cout << player.GetName() << " the thief." << endl;
216. player.JobIncrementer(10, 40, 2, -2, 1, -3, 3, 3, aWeapon, 9, aArmor, 9);
217. weapon = aWeapon[9];
218. armor = aArmor[9];
219. string WeaponName = aWeapon[nEquipmentIndex].wGetName();
220. string ArmorName = aArmor[nEquipmentIndex].aGetName();
221. cout << "Weapon: " << WeaponName << endl;
222. cout << "Armor: " << ArmorName << endl;
223. player.GetData();
224. }
225. else
226. {
227. nEquipmentIndex = 12;
228. cout << player.GetName() << " the noble." << endl;
229. player.JobIncrementer(25, 40, 1, 1, 1, 1, -1, 5, aWeapon, 12, aArmor, 12);
230. weapon = aWeapon[12];
231. armor = aArmor[12];
232. string WeaponName = aWeapon[nEquipmentIndex].wGetName();
233. string ArmorName = aArmor[nEquipmentIndex].aGetName();
234. cout << "Weapon: " << WeaponName << endl;
235. cout << "Armor: " << ArmorName << endl;
236. player.GetData();
237. }
238. }
239. catch (InvalidJob())
240. {
241. cout << "Invalid job, exiting program..." << endl;
242. system("PAUSE");
243. exit(0);
244. }
245. catch (...)
246. {
247. cerr << "Invalid error, exiting program..." << endl;
248. system("PAUSE");
249. exit(0);
250. }
251. cout << endl << "Good luck on your adventure, " << player.GetName() << endl << endl;
252. return;
253. }
254. //Battle System Function
255. int PlayerTurn(Player& ppTurn, Enemy& enemy, Weapon& weapon)
256. {
257. srand(time(NULL));
258. int npDamage;
259. int npRandom = rand() % 10 + 1;
260. int npRandomDamage = rand() % 3 + 1;
261. string spName = ppTurn.GetName();
262. string seName = enemy.eGetName();
263. if (npRandom > 3)
264. {
265. npDamage = ((ppTurn.GetAttack() + 3) + (weapon.wGetAttack()) - (enemy.eGetDefence())) \* npRandomDamage;
266. if (npDamage < 0)
267. {
268. npDamage = 1;
269. }
270. cout << seName << " suffers " << npDamage << " damage points." << endl;
271. }
272. else
273. {
274. cout << seName << " successfully dodge the attack!" << endl;
275. npDamage = 0;
276. }
277. return npDamage;
278. }
279. int EnemyTurn(Enemy& peTurn, Player& player, Armor& armor)
280. {
281. srand(time(NULL));
282. int neRandom = rand() % 10 + 1;
283. int neRandomDamage = rand() % 3 + 1;
284. int neDamage;
285. string seName = peTurn.eGetName();
286. string spName = player.GetName();
287. if (neRandom > 3)
288. {
289. neDamage = ((peTurn.eGetAttack() + 3) - ((player.GetDefence()) + (armor.aGetDefense()))) \* neRandomDamage;
290. if (neDamage <= 0)
291. {
292. neDamage = 1;
293. }
294. cout << spName << " suffers " << neDamage << " damage points." << endl;
295. }
296. else
297. {
298. cout << spName << " successfully dodge the attack!" << endl;
299. neDamage = 0;
300. }
301. return neDamage;
302. }
303. int SpecialMove(Player& player, Enemy& enemy, Weapon& weapon, int weapon\_num)
304. {
305. srand(time(NULL));
306. int npRandom = rand() % 10 + 1;
307. int npRandomDamage = rand() % 3 + 1;
308. int npSpecial;
309. int npWeaponIndex = weapon\_num;
310. string seName = enemy.eGetName();
311. string spName = player.GetName();
312. if (weapon\_num >= 0 || weapon\_num <= 2)
313. {
314. cout << "You swing your sword with high power!" << endl;
315. if (npRandom > 3)
316. {
317. npSpecial = ((((player.GetAttack() + 5) + (weapon.wGetAttack())) - ((enemy.eGetDefence()) / 2))) \* npRandomDamage \* 2;
318. cout << seName << " suffers " << npSpecial << " damage points." << endl;
319. }
320. else
321. {
322. cout << seName << " successfully dodge the attack!" << endl;
323. npSpecial = 0;
324. }
325. }
326. else if (weapon\_num >= 3 || weapon\_num <= 5)
327. {
328. cout << "You cast a burning magical spell!" << endl;
329. if (npRandom > 3)
330. {
331. npSpecial = (((player.GetMagicAttack() \* 2) \* (weapon.wGetMagicAttack())) - ((enemy.eGetMagicDefence()) / 3)) \* npRandomDamage;
332. cout << seName << " suffers " << npSpecial << " damage points." << endl;
333. }
334. else
335. {
336. cout << seName << " successfully dodge the attack!" << endl;
337. npSpecial = 0;
338. }
339. }
340. else if (weapon\_num >= 6 || weapon\_num <= 8)
341. {
342. cout << "You fire you gun with high concentration!" << endl;
343. if (npRandom > 3)
344. {
345. npSpecial = (((player.GetAttack() + 5) + (weapon.wGetAttack()) + (player.GetLuck() / 2)) - ((enemy.eGetDefence()) / 4)) \* npRandomDamage;
346. cout << seName << " suffers " << npSpecial << " damage points." << endl;
347. }
348. else
349. {
350. cout << seName << " successfully dodge the attack!" << endl;
351. npSpecial = 0;
352. }
353. }
354. else if (weapon\_num >= 9 || weapon\_num <= 11)
355. {
356. cout << "You charge towards your target with full speed!" << endl;
357. if (npRandom > 3)
358. {
359. npSpecial = (((player.GetAttack()) + (weapon.wGetAttack()) + (player.GetLuck() / 2) + (player.GetAgility())) - ((enemy.eGetDefence()) / 4)) \* npRandomDamage;
360. cout << seName << " suffers " << npSpecial << " damage points." << endl;
361. }
362. else
363. {
364. cout << seName << " successfully dodge the attack!" << endl;
365. npSpecial = 0;
366. }
367. }
368. else
369. {
370. cout << "You swing your whip with full power" << endl;
371. if (npRandom > 3)
372. {
373. npSpecial = (((player.GetAttack() / 2) + (weapon.wGetMagicAttack()) + (weapon.wGetAttack()) + (player.GetLuck() / 2)) - ((enemy.eGetDefence()) / 4)) \* npRandomDamage;
374. cout << seName << " suffers " << npSpecial << " damage points." << endl;
375. }
376. else
377. {
378. cout << seName << " successfully dodge the attack!" << endl;
379. npSpecial = 0;
380. }
381. }
382. return npSpecial;
383. }
384. void BattleSystem(Enemy& enemy, Player& player, Weapon& weapon, Armor& armor)
385. {
386. srand(time(NULL));
387. int nRandomChance;
388. string bpName = player.GetName();
389. int bpHealth = player.GetHealth();
390. int bpMana = player.GetMana();
391. int bpAgility = player.GetAgility();
392. string beName = enemy.eGetName();
393. int beHealth = enemy.eGetHealth();
394. int beMana = enemy.eGetMana();
395. int beAgility = enemy.eGetAgility();
396. cout << enemy.eGetName() << " has appeared!" << endl;
397. while (beHealth > 0)
398. {
399. if (bpHealth <= 0)
400. {
401. cout << "You died!" << endl;
402. exit(0);
403. }
404. if (beHealth <= 0)
405. {
406. cout << beName << " has died!" << endl;
407. cout << "You earned " << enemy.eGetEXP() << "!";
408. player.Leveling(enemy.eGetEXP());
409. break;
410. }
411. cout << "Player HP: " << bpHealth << endl;
412. cout << "Player MN: " << bpMana << endl << endl;
413. cout << beName << " health: " << beHealth << endl << endl;
414. nRandomChance = rand() % 3 + 1;
415. cout << "What are you going to do?" << endl;
416. cout << " 1. Attack" << endl;
417. cout << " 2. Talk" << endl;
418. cout << " 3. Use potion" << endl;
419. cout << " 4. Use skill" << endl;
420. cout << " 5. Escape" << endl;
421. int nChoice;
422. cin >> nChoice;
423. switch (nChoice)
424. {
425. case 1:
426. if (bpAgility >= beAgility)
427. {
428. beHealth -= PlayerTurn(player, enemy, weapon);
429. bpHealth -= EnemyTurn(enemy, player, armor);
430. }
431. else
432. {
433. bpHealth -= EnemyTurn(enemy, player, armor);
434. beHealth -= PlayerTurn(player, enemy, weapon);
435. }
436. break;
437. case 2:
438. cout << "You talk to the " << beName << endl;
439. if (player.GetLuck() >= 20 && nRandomChance < 3)
440. {
441. cout << "You successfully convinced the " << beName << "!" << endl;
442. cout << "You gain " << enemy.eGetEXP() << " EXP!" << endl;
443. player.Leveling(enemy.eGetEXP());
444. break;
445. }
446. else
447. {
448. cout << "You failed convinced the " << beName << "!" << endl;
449. bpHealth -= EnemyTurn(enemy, player, armor);
450. }
451. break;
452. case 3:
453. if (nPotion > 0)
454. {
455. cout << "You use the potion!" << endl;
456. nPotion--;
457. if (bpHealth > player.GetHealth() + 30)
458. {
459. bpHealth += 30;
460. }
461. else
462. {
463. bpHealth = player.GetHealth();
464. }
465. }
466. break;
467. case 4:
468. if (bpMana >= 15)
469. {
470. if (bpAgility >= beAgility)
471. {
472. beHealth -= SpecialMove(player, enemy, aWeapon[nEquipmentIndex], nEquipmentIndex);
473. bpHealth -= EnemyTurn(enemy, player, armor);
474. }
475. else
476. {
477. bpHealth -= EnemyTurn(enemy, player, armor);
478. beHealth -= SpecialMove(player, enemy, aWeapon[nEquipmentIndex], nEquipmentIndex);
479. }
480. bpMana -= 15;
481. }
482. else
483. {
484. cout << "You don't have enough mana to perform special attack" << endl;
485. bpHealth -= EnemyTurn(enemy, player, armor);
486. }
487. break;
488. default:
489. cout << "You tried to escape." << endl;
490. if (nRandomChance < 3)
491. {
492. cout << "You successfully escaped!" << endl;
493. cout << "You earned 10 EXP!" << endl;
494. player.Leveling(10);
495. break;
496. }
497. else
498. {
499. cout << "You failed to escape!" << endl;
500. bpHealth -= EnemyTurn(enemy, player, armor);
501. }
502. break;
503. }
504. }
505. }
506. //Game part
507. void Story()
508. {
509. cout << "Welcome to the Simple World, where you can explore the world " << endl;
510. cout << "and probably be what you want!" << endl;
511. cout << "To start off, let say you were enjoying a goblet of ale at a " << endl;
512. cout << "pub named, 'The Bannered Mare'." << endl;
513. cout << "You saw a bounty that will reward you handsomely." << endl;
514. cout << "What are you going to do?" << endl;
515. cout << "1. Take the bounty." << endl;
516. cout << "2. Leave the bounty." << endl;
517. int nChoiceOne;
518. cin >> nChoiceOne;
519. if (nChoiceOne < 1 || nChoiceOne > 2)
520. {
521. for (int i = 0; i > 1 || i < 2; )
522. {
523. cout << "Choice not found." << endl;
524. cout << "You saw a bounty that will reward you handsomely." << endl;
525. cout << "What are you going to do?" << endl;
526. cout << "1. Take the bounty." << endl;
527. cout << "2. Leave the bounty." << endl;
528. int nChoiceOne;
529. cin >> nChoiceOne;
530. }
531. }
532. else if (nChoiceOne == 1)
533. {
534. ChoiceOne();
535. }
536. else
537. {
538. ChoiceTwo();
539. }
540. }
541. void ChoiceOne()
542. {
543. cout << "You took the bounty poster." << endl;
544. cout << "It says that you will need to venture into a dungeon and\n";
545. cout << "search for the Chalice." << endl;
546. cout << "Would you like to take it? " << endl;
547. cout << "Yes/No?" << endl;
548. string sBountyAcceptance;
549. cin >> sBountyAcceptance;
550. if (sBountyAcceptance == "Yes" || sBountyAcceptance == "yes" || sBountyAcceptance == "1" || sBountyAcceptance == "true" || sBountyAcceptance == "True")
551. {
552. cout << "You took the bounty." << endl;
553. cout << "Your first clue is to head out of the city's wall." << endl;
554. ChoiceOneOne();
555. }
556. else
557. {
558. ChoiceTwo();
559. }
560. }
561. void ChoiceOneOne()
562. {
563. cout << "You exited the city and you saw several things that interest you." << endl;
564. cout << "Which one would you like to inspect?" << endl;
565. cout << "1. The grassy path." << endl;
566. cout << "2. The broken carriage." << endl;
567. int nChoices;
568. cin >> nChoices;
569. string sLooting;
570. switch (nChoices)
571. {
572. case 1:
573. cin.ignore();
574. cout << "You decided to explore the grassy road." << endl;
575. cout << "At the end of the road, you were ambushed by a group of bandits." << endl;
576. cout << "What are you going to do?" << endl;
577. cout << "1. Retaliate" << endl;
578. cout << "2. Talk to your way out" << endl;
579. int nChoiceOne;
580. cin >> nChoiceOne;
581. switch (nChoiceOne)
582. {
583. case 1:
584. BattleSystem(enemy[4], player, weapon, armor);
585. cout << "You leave the scene and decided to head out." << endl;
586. cout << "You found a cave, you decided to enter it." << endl;
587. ChoiceTwoOne();
588. break;
589. case 2:
590. cout << "You talk your way out" << endl;
591. cout << "You told them there is a treasure near the hill not far from the wall" << endl;
592. int LuckPar = player.GetLuck();
593. if (LuckPar > 10)
594. {
595. cout << "You are lucky that those bandits are not the brightest ones." << endl;
596. cout << "They thank you and ran off towards the place you told them." << endl;
597. cout << "You earn 40 EXP." << endl;
598. player.Leveling(40);
599. cout << "You decided to wander off further." << endl;
600. }
601. else
602. {
603. cout << "You failed to convinced the bandit" << endl;
604. cout << "The bandit attacks you!" << endl;
605. BattleSystem(enemy[4], player, weapon, armor);
606. cout << "You leave the scene and decided to head out." << endl;
607. cout << "You found a cave, you decided to enter it." << endl;
608. }
609. ChoiceTwoOne();
610. break;
611. }
612. break;
613. case 2:
614. srand(time(NULL));
615. cout << "You approached the torn down carriage." << endl;
616. cout << "You inspect it closer." << endl;
617. cout << "Among the debris, there is one chest that is intact." << endl;
618. cout << "Would you like to inspect the intact chest?" << endl;
619. cout << "Yes/No" << endl;
620. getline(cin, sLooting);
621. if (sLooting == "Yes" || sLooting == "yes" || sLooting == "1" || sLooting == "true" || sLooting == "True")
622. {
623. int nProbabilityItem = rand() % 35 + 0;
624. cout << "Surprisingly, the chest is not locked." << endl;
625. if (nProbabilityItem <= 14)
626. {
627. cout << "You open the chest and you found the " << aWeapon[nProbabilityItem].wGetName() << "." << endl;
628. cout << "Do you want to change it?" << endl;
629. cout << "Yes/No?" << endl;
630. string sChanging;
631. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
632. {
633. player.SetWeapon(nProbabilityItem);
634. weapon = aWeapon[nProbabilityItem];
635. }
636. else
637. {
638. cout << "You decided to leave the equipment behind." << endl;
639. }
640. }
641. else if (nProbabilityItem >= 15 || nProbabilityItem <= 29)
642. {
643. cout << "You open the chest and you found the " << aArmor[nProbabilityItem].aGetName() << "." << endl;
644. cout << "Do you want to change it?" << endl;
645. cout << "Yes/No?" << endl;
646. string sChanging;
647. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
648. {
649. player.SetArmor(nProbabilityItem - 15);
650. armor = aArmor[nProbabilityItem - 15];
651. }
652. else
653. {
654. cout << "You decided to leave the equipment behind." << endl;
655. }
656. }
657. else
658. {
659. cout << "You open the chest and you found a potion." << endl;
660. nPotion++;
661. }
662. cout << "You earn 20 EXP." << endl;
663. player.Leveling(20);
664. cout << "You also found a laying note near the chest." << endl;
665. cout << "Apperantly the note is a treasure map." << endl;
666. cout << "You decided to follow the map." << endl;
667. ChoiceTwoOne();
668. }
669. else
670. {
671. cout << "You leave the chest alone." << endl;
672. ChoiceTwoOne();
673. }
674. break;
675. default:
676. cout << "You are not sure what to do because you can't input the number 1 or 2." << endl;
677. cout << "You wander off and ended up right front of the entrance of a dungeon." << endl;
678. cout << "Would you go in?" << endl;
679. cout << "Yes/No?" << endl;
680. string sEnteringDungeon;
681. getline(cin, sEnteringDungeon);
682. if (sEnteringDungeon == "Yes" || sEnteringDungeon == "yes" || sEnteringDungeon == "1" || sEnteringDungeon == "true" || sEnteringDungeon == "True")
683. {
684. cout << "You Entered the dungeon." << endl;
685. ChoiceTwoOne();
686. }
687. else
688. {
689. cout << "Nothing around you interests you." << endl;
690. cout << "You decided to enter the dungeon." << endl;
691. ChoiceTwoOne();
692. }
693. }
694. }
695. void ChoiceTwo()
696. {
697. cin.ignore();
698. cout << "You are not interested with the bounty." << endl;
699. cout << "Someone is calling you in the corner of the room." << endl;
700. cout << "Would you answer its calling?" << endl;
701. cout << "Yes/No" << endl;
702. string sCalling;
703. getline(cin, sCalling);
704. if (sCalling == "Yes" || sCalling == "yes" || sCalling == "1" || sCalling == "true" || sCalling == "True")
705. {
706. cout << "You approached the man." << endl;
707. cout << "He seems to be shaddy and a bit suspicious." << endl;
708. cout << "He say that he would pay you to enter steal" << endl;
709. cout << "a certain item from a ruin." << endl;
710. cout << "Would you do it?" << endl;
711. cout << "Yes/No?" << endl;
712. string sYes;
713. getline(cin, sYes);
714. if (sYes == "Yes" || sYes == "yes" || sYes == "1" || sYes == "true" || sYes == "True")
715. {
716. }
717. else
718. {
719. cout << "You decided to leave the pub." << endl;
720. cout << "You left the town." << endl;
721. ChoiceOneOne();
722. }
723. }
724. else
725. {
726. cout << "You left the pub." << endl;
727. cout << "You decided to left the city." << endl;
728. ChoiceOneOne();
729. }
730. return;
731. }
732. void ChoiceTwoOne()
733. {
734. srand(time(NULL));
735. int nArea = 0;
736. int nRoomCheck;
737. int nProbabilityEvent;
738. int nProbabilityItem;
739. int nProbabilityEnemy;
740. for (nRoomCheck = 0; nRoomCheck <= 50; nRoomCheck++)
741. {
742. nProbabilityEvent = rand() % 10 + 1;
743. int nDirection = 0;
744. cout << "Area: " << nArea << endl;
745. cout << "Where are you going?" << endl;
746. cout << "1. Forward" << endl;
747. cout << "2. Right" << endl;
748. cout << "3. Left" << endl;
749. cout << "4. Backward" << endl;
750. cin >> nDirection;
751. switch (nDirection)
752. {
753. case 1:
754. cout << "An enemy appeared!" << endl;
755. if (nProbabilityEvent >= 1 || nProbabilityEvent <= 4)
756. {
757. nProbabilityEnemy = rand() % 20 + 1;
758. if (nProbabilityEnemy > 0 || nProbabilityEnemy < 4)
759. {
760. BattleSystem(enemy[0], player, weapon, armor);
761. }
762. if (nProbabilityEnemy >= 4 || nProbabilityEnemy <= 6)
763. {
764. BattleSystem(enemy[1], player, weapon, armor);
765. }
766. if (nProbabilityEnemy >= 7 || nProbabilityEnemy <= 9)
767. {
768. BattleSystem(enemy[2], player, weapon, armor);
769. }
770. if (nProbabilityEnemy >= 10 || nProbabilityEnemy <= 12)
771. {
772. BattleSystem(enemy[3], player, weapon, armor);
773. }
774. if (nProbabilityEnemy >= 13 || nProbabilityEnemy <= 15)
775. {
776. BattleSystem(enemy[5], player, weapon, armor);
777. }
778. if (nProbabilityEnemy >= 16 || nProbabilityEnemy <= 18)
779. {
780. BattleSystem(enemy[6], player, weapon, armor);
781. }
782. if (nProbabilityEnemy >= 19 || nProbabilityEnemy <= 20)
783. {
784. BattleSystem(enemy[7], player, weapon, armor);
785. }
786. }
787. else if (nProbabilityEvent >= 5 || nProbabilityEvent <= 7)
788. {
789. nProbabilityItem = rand() % 35 + 0;
790. cout << "You found a chest!" << endl;
791. if (nProbabilityItem <= 14)
792. {
793. cout << "You open the chest and you found the " << aWeapon[nProbabilityItem].wGetName() << "." << endl;
794. cout << "Do you want to change it?" << endl;
795. cout << "Yes/No?" << endl;
796. string sChanging;
797. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
798. {
799. player.SetWeapon(nProbabilityItem);
800. weapon = aWeapon[nProbabilityItem];
801. }
802. else
803. {
804. cout << "You decided to leave the equipment behind." << endl;
805. }
806. }
807. else if (nProbabilityItem >= 15 || nProbabilityItem <= 29)
808. {
809. cout << "You open the chest and you found the " << aArmor[nProbabilityItem].aGetName() << "." << endl;
810. cout << "Do you want to change it?" << endl;
811. cout << "Yes/No?" << endl;
812. string sChanging;
813. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
814. {
815. player.SetArmor(nProbabilityItem - 15);
816. armor = aArmor[nProbabilityItem - 15];
817. }
818. else
819. {
820. cout << "You decided to leave the equipment behind." << endl;
821. }
822. }
823. else
824. {
825. cout << "You open the chest and you found a potion." << endl;
826. nPotion++;
827. }
828. }
829. else
830. {
831. cout << "There is nothing here..." << endl;
832. }
833. break;
834. case 2:
835. cout << "An enemy appeared!" << endl;
836. if (nProbabilityEvent >= 1 || nProbabilityEvent <= 4)
837. {
838. nProbabilityEnemy = rand() % 20 + 1;
839. if (nProbabilityEnemy > 0 || nProbabilityEnemy < 4)
840. {
841. BattleSystem(enemy[0], player, weapon, armor);
842. }
843. if (nProbabilityEnemy >= 4 || nProbabilityEnemy <= 6)
844. {
845. BattleSystem(enemy[1], player, weapon, armor);
846. }
847. if (nProbabilityEnemy >= 7 || nProbabilityEnemy <= 9)
848. {
849. BattleSystem(enemy[2], player, weapon, armor);
850. }
851. if (nProbabilityEnemy >= 10 || nProbabilityEnemy <= 12)
852. {
853. BattleSystem(enemy[3], player, weapon, armor);
854. }
855. if (nProbabilityEnemy >= 13 || nProbabilityEnemy <= 15)
856. {
857. BattleSystem(enemy[5], player, weapon, armor);
858. }
859. if (nProbabilityEnemy >= 16 || nProbabilityEnemy <= 18)
860. {
861. BattleSystem(enemy[6], player, weapon, armor);
862. }
863. if (nProbabilityEnemy >= 19 || nProbabilityEnemy <= 20)
864. {
865. BattleSystem(enemy[7], player, weapon, armor);
866. }
867. }
868. else if (nProbabilityEvent >= 5 || nProbabilityEvent <= 7)
869. {
870. nProbabilityItem = rand() % 35 + 0;
871. cout << "You found a chest!" << endl;
872. if (nProbabilityItem <= 14)
873. {
874. cout << "You open the chest and you found the " << aWeapon[nProbabilityItem].wGetName() << "." << endl;
875. cout << "Do you want to change it?" << endl;
876. cout << "Yes/No?" << endl;
877. string sChanging;
878. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
879. {
880. player.SetWeapon(nProbabilityItem);
881. weapon = aWeapon[nProbabilityItem];
882. }
883. else
884. {
885. cout << "You decided to leave the equipment behind." << endl;
886. }
887. }
888. else if (nProbabilityItem >= 15 || nProbabilityItem <= 29)
889. {
890. cout << "You open the chest and you found the " << aArmor[nProbabilityItem].aGetName() << "." << endl;
891. cout << "Do you want to change it?" << endl;
892. cout << "Yes/No?" << endl;
893. string sChanging;
894. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
895. {
896. player.SetArmor(nProbabilityItem - 15);
897. armor = aArmor[nProbabilityItem - 15];
898. }
899. else
900. {
901. cout << "You decided to leave the equipment behind." << endl;
902. }
903. }
904. else
905. {
906. cout << "You open the chest and you found a potion." << endl;
907. nPotion++;
908. }
909. }
910. else
911. {
912. cout << "There is nothing here..." << endl;
913. }
914. break;
915. case 3:
916. cout << "An enemy appeared!" << endl;
917. if (nProbabilityEvent >= 1 || nProbabilityEvent <= 4)
918. {
919. nProbabilityEnemy = rand() % 20 + 1;
920. if (nProbabilityEnemy > 0 || nProbabilityEnemy < 4)
921. {
922. BattleSystem(enemy[0], player, weapon, armor);
923. }
924. if (nProbabilityEnemy >= 4 || nProbabilityEnemy <= 6)
925. {
926. BattleSystem(enemy[1], player, weapon, armor);
927. }
928. if (nProbabilityEnemy >= 7 || nProbabilityEnemy <= 9)
929. {
930. BattleSystem(enemy[2], player, weapon, armor);
931. }
932. if (nProbabilityEnemy >= 10 || nProbabilityEnemy <= 12)
933. {
934. BattleSystem(enemy[3], player, weapon, armor);
935. }
936. if (nProbabilityEnemy >= 13 || nProbabilityEnemy <= 15)
937. {
938. BattleSystem(enemy[5], player, weapon, armor);
939. }
940. if (nProbabilityEnemy >= 16 || nProbabilityEnemy <= 18)
941. {
942. BattleSystem(enemy[6], player, weapon, armor);
943. }
944. if (nProbabilityEnemy >= 19 || nProbabilityEnemy <= 20)
945. {
946. BattleSystem(enemy[7], player, weapon, armor);
947. }
948. }
949. else if (nProbabilityEvent >= 5 || nProbabilityEvent <= 7)
950. {
951. nProbabilityItem = rand() % 35 + 0;
952. cout << "You found a chest!" << endl;
953. if (nProbabilityItem <= 14)
954. {
955. cout << "You open the chest and you found the " << aWeapon[nProbabilityItem].wGetName() << "." << endl;
956. cout << "Do you want to change it?" << endl;
957. cout << "Yes/No?" << endl;
958. string sChanging;
959. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
960. {
961. player.SetWeapon(nProbabilityItem);
962. weapon = aWeapon[nProbabilityItem];
963. }
964. else
965. {
966. cout << "You decided to leave the equipment behind." << endl;
967. }
968. }
969. else if (nProbabilityItem >= 15 || nProbabilityItem <= 29)
970. {
971. cout << "You open the chest and you found the " << aArmor[nProbabilityItem].aGetName() << "." << endl;
972. cout << "Do you want to change it?" << endl;
973. cout << "Yes/No?" << endl;
974. string sChanging;
975. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
976. {
977. player.SetArmor(nProbabilityItem - 15);
978. armor = aArmor[nProbabilityItem - 15];
979. }
980. else
981. {
982. cout << "You decided to leave the equipment behind." << endl;
983. }
984. }
985. else
986. {
987. cout << "You open the chest and you found a potion." << endl;
988. nPotion++;
989. }
990. }
991. else
992. {
993. cout << "There is nothing here..." << endl;
994. }
995. break;
996. default:
997. cout << "An enemy appeared!" << endl;
998. if (nProbabilityEvent >= 1 || nProbabilityEvent <= 4)
999. {
1000. nProbabilityEnemy = rand() % 20 + 1;
1001. if (nProbabilityEnemy > 0 || nProbabilityEnemy < 4)
1002. {
1003. BattleSystem(enemy[0], player, weapon, armor);
1004. }
1005. if (nProbabilityEnemy >= 4 || nProbabilityEnemy <= 6)
1006. {
1007. BattleSystem(enemy[1], player, weapon, armor);
1008. }
1009. if (nProbabilityEnemy >= 7 || nProbabilityEnemy <= 9)
1010. {
1011. BattleSystem(enemy[2], player, weapon, armor);
1012. }
1013. if (nProbabilityEnemy >= 10 || nProbabilityEnemy <= 12)
1014. {
1015. BattleSystem(enemy[3], player, weapon, armor);
1016. }
1017. if (nProbabilityEnemy >= 13 || nProbabilityEnemy <= 15)
1018. {
1019. BattleSystem(enemy[5], player, weapon, armor);
1020. }
1021. if (nProbabilityEnemy >= 16 || nProbabilityEnemy <= 18)
1022. {
1023. BattleSystem(enemy[6], player, weapon, armor);
1024. }
1025. if (nProbabilityEnemy >= 19 || nProbabilityEnemy <= 20)
1026. {
1027. BattleSystem(enemy[7], player, weapon, armor);
1028. }
1029. }
1030. else if (nProbabilityEvent >= 5 || nProbabilityEvent <= 7)
1031. {
1032. nProbabilityItem = rand() % 35 + 0;
1033. cout << "You found a chest!" << endl;
1034. if (nProbabilityItem <= 14)
1035. {
1036. cout << "You open the chest and you found the " << aWeapon[nProbabilityItem].wGetName() << "." << endl;
1037. cout << "Do you want to change it?" << endl;
1038. cout << "Yes/No?" << endl;
1039. string sChanging;
1040. if (nProbabilityItem <= 14)
1041. {
1042. cout << "You open the chest and you found the " << aWeapon[nProbabilityItem].wGetName() << "." << endl;
1043. cout << "Do you want to change it?" << endl;
1044. cout << "Yes/No?" << endl;
1045. string sChanging;
1046. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
1047. {
1048. player.SetWeapon(nProbabilityItem);
1049. weapon = aWeapon[nProbabilityItem];
1050. }
1051. else
1052. {
1053. cout << "You decided to leave the equipment behind." << endl;
1054. }
1055. }
1056. else if (nProbabilityItem >= 15 || nProbabilityItem <= 29)
1057. {
1058. cout << "You open the chest and you found the " << aArmor[nProbabilityItem].aGetName() << "." << endl;
1059. cout << "Do you want to change it?" << endl;
1060. cout << "Yes/No?" << endl;
1061. string sChanging;
1062. if (sChanging == "Yes" || sChanging == "yes" || sChanging == "1" || sChanging == "true" || sChanging == "True")
1063. player.SetArmor(nProbabilityItem - 15);
1064. armor = aArmor[nProbabilityItem - 15];
1065. }
1066. else
1067. {
1068. cout << "You decided to leave the equipment behind." << endl;
1069. }
1070. }
1071. else
1072. {
1073. cout << "You open the chest and you found a potion." << endl;
1074. nPotion++;
1075. }
1076. }
1077. else
1078. {
1079. cout << "There is nothing here..." << endl;
1080. }
1081. break;
1082. }
1083. nArea++;
1084. }
1085. }
1086. void End()
1087. {
1088. cout << "You have reached the end of the room." << endl;
1089. cout << "You saw a pedestal at the end of the room." << endl;
1090. cout << "There is something on top of the pedestal, something precious." << endl;
1091. cout << "However, something large descends from the sky." << endl;
1092. cout << "It was a dragon!" << endl;
1093. cout << "It starting to spew fire out of his snout." << endl;
1094. BattleSystem(enemy[9], player, weapon, armor);
1095. cout << "As you passed by the dead beast, you approached the object." << endl;
1096. cout << "You touch the object and it glows intensely." << endl;
1097. cout << "It was the chalice." << endl;
1098. cout << "You have succeed the game!" << endl;
1099. system("PAUSE");
1100. exit(0);
1101. }