Faculty of Engineering

Department of Electrical Engineering

Advanced Computer Lab (361.1.4693)

OOP C++

Final Project
Final Report

תאריך:
5.9.2020

מגיש:
נתנאל מזוז 203973532

1. Definitions:
    - The classes in the project

      Class: Character and Gameloader

      The Character class is the bade class for all other figures/character that participate in the scenarios. The Gameloader class is used to initiate the saving/loading of the game from the text file or into it.

      Sub-Classes of Character:

      For players: Warrior, Mage and Thief.

      As enemies: Hooded woman, Zombie, Bear, skeletons and villagers.
    - The use of inheritance in the project

      Warrior, Mage and Thief l sub-classes inherit the following fields:
        - NAME
        - INIT (initiative)
        - HP (hit points)
        - MP (mana points)
        - DFF (defense)
        - ATK (attack)
        - EXP (experience)
        - LVL (level)
        - GOLD
        - ITEMS
            - Torso (armor, robe, etc.)
            - hands (Sword, Bow, Dagger, Staff, etc.)
            - Head (Helmet, Cape, etc.)
            - Legs (boots)
    - The use of polymorphism in the project

      The method LEVEL UP upgrades the stats and/or new skills of the character depending on his/her job. (warrior, mage, thief, etc.)
2. Data privileges:
    - Private: methods or stats that only this job (warrior, mage or thief) that only has.
    - Protected: non
    - Public: methods that are required to operate globally, like battle, stats and shop.
3. Redefined construction and the destruction:
    - The basic stats of a new character are defined using a non-basic construction and depends on its job. For example, mage will have mana points while warrior will have increased basic ATK and no MP.
4. Memory allocation:
    - I use memory allocation to create new character (either enemy or player).
5. Global and local variables in the project: non.

6. Game Mechanism:

Main menu: The First thing the player sees is the main menu in which he/she can choose between the following options:

1. Start new game
2. Load game
3. Exit

1. Start new game: The player then has to choose the difficulty, number of players (1 to 3) and then choose NAME and JOB for every character.
2. Load game: The player starts the scenario from the saving point, but the progress of battle is reset (player cannot save the game during a fight) thus, it is important to save the progress between battles. Saving the game overwrites the date from the previously saved game.
3. Exit: Simply closing the program.

NAME of a character:

The NAME of the character will be used in some dialog boxes that address to the player. All (one to three) players start the at LVL (level) 0 and with:

- 0 EXP ()
- 10 GOLD in their pocket (except thief that starts with 5).

A character's job:

Influences the character's max HP (hit points), MP (mana points), ATK (attack), DFF (defense), INIT (initiative) and starting items.

Job methods:

Each job (Warrior, Mage and thief) has a special unique method"

- Warrior – non. He is just strong enough without a special method.
- Mage – "Magic attack" that causes 4 damage and ignoring the enemy's armor.
- Thief – "Back Stab" that cause ATK damage and steals 2 GOLD.

To complete the game all players must win all 5 scenarios. Each scenario has its monsters/boss and rewards. The battle is turn based; the turn order is decided based on the initiative of each character participating the fight.

Store:

Before each scenario (except the first one) the players can buy items from a shop. Players can buy only one item at the shop (once each). The items work as an improvement to the stats of the characters, each item affect different stat (for example, the boots increase one's initiative). Players cannot sell items and cannot buy items that does not fit them. For example, Mage cannot buy sword and thief cannot buy staff.

Battle mechanics:

The battle is divided into rounds in which every character (players and monsters) can attack one (or more, depending on the attack type) other character/s (monsters can only attack players and players can only attack monsters). The character with the highest initiative starts first, and in turns, each character can choose an attack target. The HP lost will be calculated as follows:

HP lost = ATK (of the attacker)- DFF (of the defender)

Monster will always attack the lowest HP player when possible. When the HP of a character reaches to 0 the character dies and cannot participate the battle and cannot continue to the next scenarios. However, when the battle ends, if a character participated the battle it may gain exp (even if defeated) and level up which heals it to max hit points.

Difficulty

The HP, ATK and DFF of the monsters is scaled according to the difficulty:

1. Easy
2. Normal
3. Hard

Here, I changed the formula that I wrote in the preliminary report. Each enemy has a different addition to its stats according to the difficulty and number of players. For example:

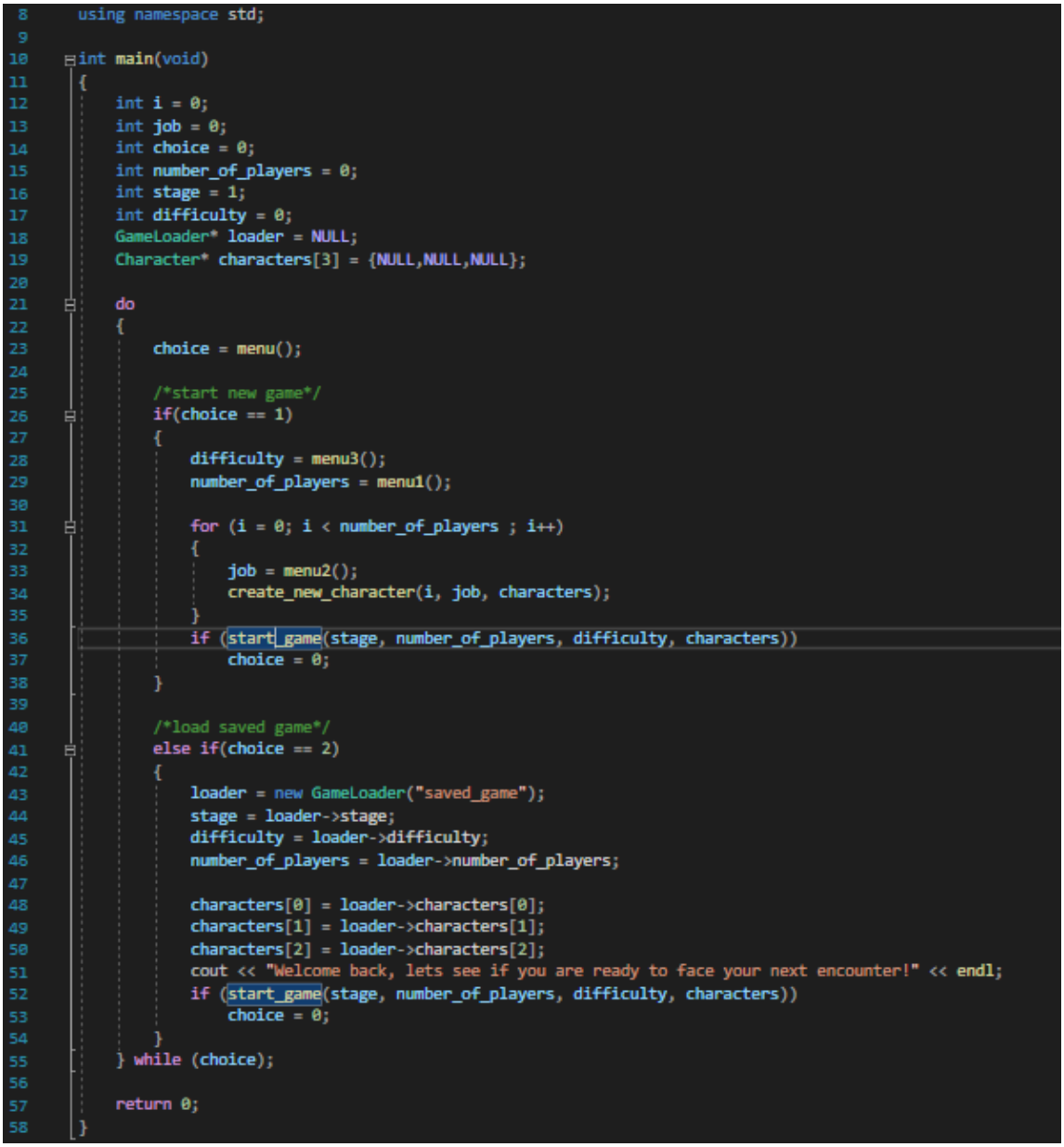
The constructor of the Hooded woman:

```
Hooded_woman::Hooded_woman(string name, int number_of_players, int difficulty) : Character(name)
{
    NAME = name;
    hp = 20 + number_of_players * 2 + difficulty;
    HP = 20 + number_of_players * 2 + difficulty;
    mp = 10 + difficulty * 2;
    MP = 10 + difficulty * 2;
    exp = 0;
    EXP = 10;
    GOLD = 0;
    INIT = 11;
    ATK = 3 + number_of_players;
    DFF = 3 + difficulty;
    LVL = 0;
    Torso = 0;
    Hands = 0;
    Head = 0;
    Legs = 0;
}
```

Code:

The code is divided into 3 major elements (files):

Main – which includes the main function to start the game, load from the previous saved game or exit.



```
using namespace std;

int main(void)
{
    int i = 0;
    int job = 0;
    int choice = 0;
    int number_of_players = 0;
    int stage = 1;
    int difficulty = 0;
    GameLoader* loader = NULL;
    Character* characters[3] = {NULL,NULL,NULL};

    do
    {
        choice = menu();

        /*start new game*/
        if(choice == 1)
        {
            difficulty = menu3();
            number_of_players = menu1();

            for (i = 0; i < number_of_players ; i++)
            {
                job = menu2();
                create_new_character(i, job, characters);
            }
            if (start_game(stage, number_of_players, difficulty, characters))
                choice = 0;
        }

        /*load saved game*/
        else if(choice == 2)
        {
            loader = new GameLoader("saved_game");
            stage = loader->stage;
            difficulty = loader->difficulty;
            number_of_players = loader->number_of_players;

            characters[0] = loader->characters[0];
            characters[1] = loader->characters[1];
            characters[2] = loader->characters[2];
            cout << "Welcome back, lets see if you are ready to face your next encounter!" << endl;
            if (start_game(stage, number_of_players, difficulty, characters))
                choice = 0;
        }
    } while (choice);

    return 0;
}
```

Header (Header.h and Header.cpp) file that includes the class's declaration and definitions and battle function.

```
class Character
{
public:
    string NAME;            // The name of the character
    int hp;                 // The current hit points
    int HP;                 // Maximun hit points of the character
    int mp;                 // The current mana points
    int MP;                 // Maximum mana points of the character
    int exp;                // The current experience of the character
    int EXP;                // Maximum experience to level up for the current level
    int GOLD;               // Current gold amount
    int INIT;               // Initiative of the character to determine turn order
    int ATK;                // Attack of the character
    int DFF;                // Defense of the character
    int LVL;                // Level of the character
    int Torso;              // Item for the torso
    int Hands;              // Item for the hands
    int Head;               // Item for the head
    int Legs;               // Item for the legs

public:
    virtual void Display();
    virtual void Heal(int x);
    virtual void GainEXP(int x);
    virtual void LoseHP(int x);
    virtual void perform_attack(Character* deffender);
    virtual void LevelUP();
    virtual void shop();
    Character(string name);
    Character();
    ~Character();
};
```

Input (Input.h and Input.cpp) files that include the handling communication with the user

```
bool check(char input[]);
void create_new_character(int i, int job, Character* (&character)[3]);
void sortCharacters(Character* (&destenation)[3], Character* (&source)[3]);
int start_game(int stage, int number_of_players, int difficulty, Character* (&character)[3]);
int battle(int number_of_players, int number_of_enemies, int difficulty, Character* (&character)[3], Character* (&enemies)[3]);
int menu();
int menu1();
int menu2();
int menu3();
int stage1_option();
int stage2_option();
int stage4_option();
int shop2_option();
```

.