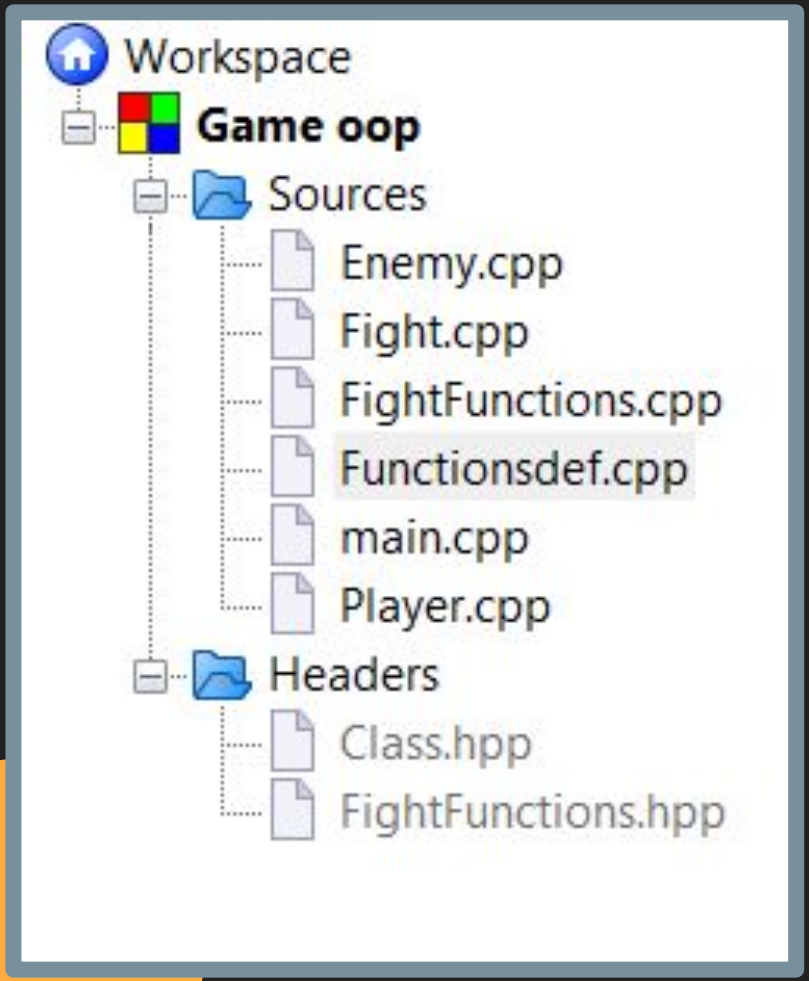
Simple RPG game

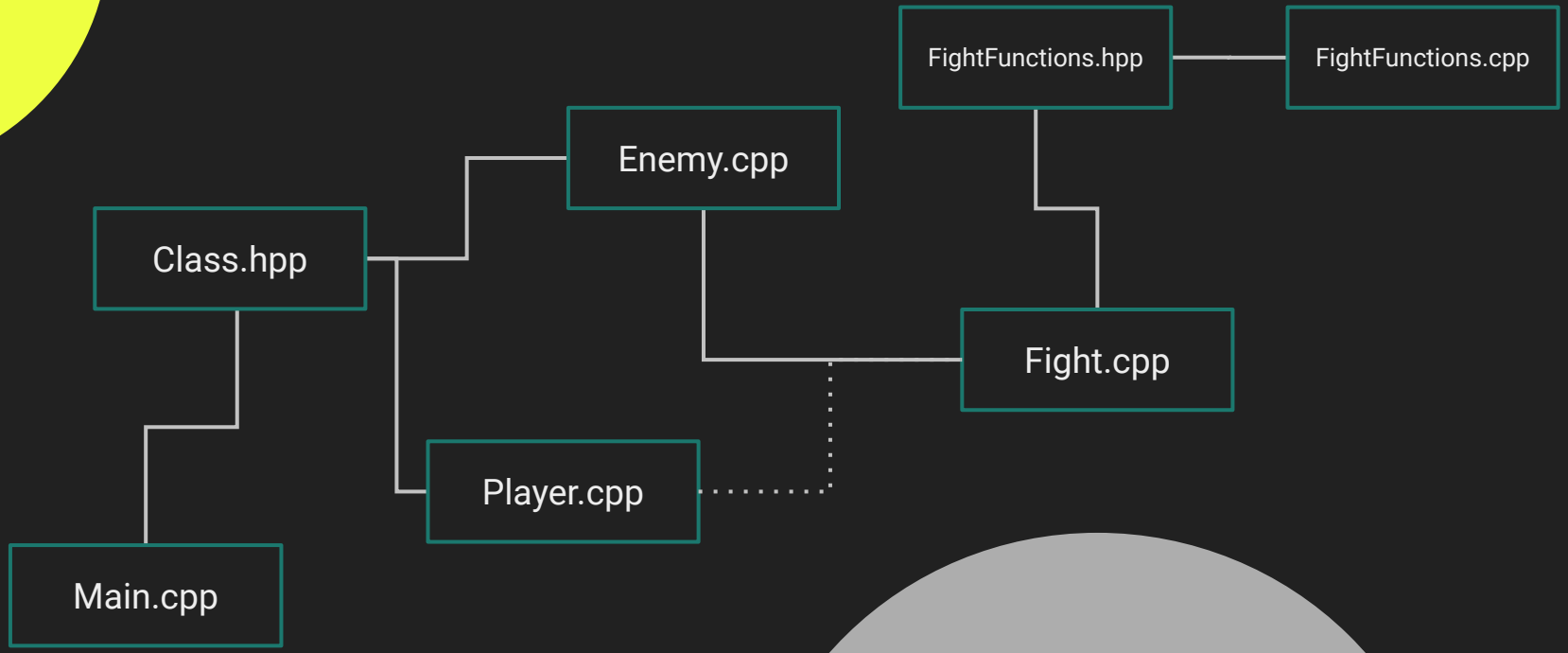
Bacik Emilia 155634

Main idea of the game

- A simple system of obtaining experience and levelling.
- Two different types of enemies – one of them possible to tame.
- Duel system with the enemy.
- A functional menu of choices available to the player.

File hierarchy




FightFunctions.hpp
FightFunctions.cpp
Enemy.cpp
Class.hpp
Fight.cpp
Player.cpp
Main.cpp

Player Class

```cpp
Player::Player()
{
    cout << "Enter your login: ";
    cin >> login;
    stats[0]=1;
    stats[1]=1;
    stats[2]=1;
    hp=65;
    max_hp=hp;
    exp=0;
    pet=false;
    cout << endl;

}
```

```cpp
class Player
{
    friend void Enemy::Fight(Player &);
    friend void FireEnemy::Fight(Player &);
    friend void WaterEnemy::Fight(Player &);
private:
    string login;
    int stats[3]; // 0-strength 1-speed 2-agility
    int hp;
    int max_hp;
    int exp;
    static int level;
    bool pet;
    void ImproveStats();
public:
    Player();
    ~Player() = default;
    void CheckStats();
    int CheckHp();
    string CheckName();
    void LevelUp();
    void HealYourself();
    void TrainYourself();
    bool EndGame();
};
```

Some of the interesting methods

```
void Player::ImproveStats()
{
    cout << "Enter 1 to improve strength, 2 to improve speed or 3 to improve agility." << endl;
    int tmp;
    while(1)
    {
        try
        {
            cin >> tmp;
            if (tmp==1 || tmp==2 || tmp==3)
                break;
            else
              throw invalid_argument("error");
        }
        catch(invalid_argument)
        {
```

```
void Player::TrainYourself()
{
    cout << endl << "You spend the whole day training!" << endl;
    exp++;
    ImproveStats();
}
```

```
void Player::LevelUp()
{
    if(exp>=3)
    {
        level++;
        exp=0;
        max_hp+=15;
        cout << endl <<"Congratulation, you have a level up! Now you have " << level << " level." << endl;
        cout << "You got two stat points. You can choose which two stats you want to improve." << endl;
        ImproveStats();
        cout << "And second point?" << endl;
        ImproveStats();
        CheckStats();
    }

}
```

Enemy Class and its descendants

```cpp
class Enemy
{
protected:
    int strength;
    int speed;
    int agility;
    int hp;
public:
    Enemy() = default;
    virtual ~Enemy() = default;
    virtual void Fight(Player &)=0;
    virtual void DrawStats(int strength_p, int speed_p, int agility_p, int hp_p);
};
```

```cpp
class FireEnemy :public Enemy
{
private:
    string name;
    static int magic;
public:
    FireEnemy(string name_p, int strength_p, int speed_p, int agility_p, int hp_p);
    ~FireEnemy() = default;
    void CheckStats();
    virtual void Fight(Player &) override;
};
```

```
int ChooseAction()
{
    cout << endl << "Your turn. Enter 1 if you want to attack, 2 if you want to defense and 3 if you want to run away: ";
    int tmp;
    while(1)
    {
        try
        {
            cin >> tmp;
            if(tmp==1 || tmp==2 || tmp==3)
                break;
            else
              throw invalid_argument("error");
        }
        catch(invalid_argument)
        {
            cout << "Try again!" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
    }
    return tmp;
}
```

```
int Attack(int attacker_stats[3], int defence_stats[3], bool defence)
{
    srand(time(NULL));
    int hp = 0;
    if(defence)
        hp=(3*attacker_stats[0]*2*attacker_stats[1]) - (3*defence_stats[2]*2*defence_stats[1]);
    else
        hp=(3*attacker_stats[0]*2*attacker_stats[1]) - (1*defence_stats[2]*1*defence_stats[1]);
    hp=hp*2;
    hp = hp +(rand()%6)-3;
    if(hp<=0)
      {
          hp=0;
      }
    return hp;
}
```

Some of the methods which are used in the fight method

Fight methods

```cpp
void FireEnemy::Fight(Player &p)
{
    int stats[3]={strength, speed, agility};
    //cout << endl << "Enemystat's: Strenght - " << stats[0] << " Speed - " << stats[1] << " Agility - " << stats[2];
    int x = 0, y=0;
    int enemy_max_hp=hp;
    bool enemy_defence=false, user_defence=false;
    cout << endl << endl << "The fight between you and " << name << " has begun!";
    cout << endl << "Be careful, fire's enemies attack a lot!";

    int turn = WhoStarts();
    while(p.hp>0 && hp>0)
    {
    cout << endl << "Your HP: " << p.hp << "    Enemy's HP: " << hp << endl;
        if (turn==0)
        {
            x = ChooseAction();
            switch (x)
            {
            case 1:{
                int power_of_attack = Attack(p.stats, stats, enemy_defence);
                hp = hp - power_of_attack;
                cout << p.login << " deals " << power_of_attack << " damage!";
                user_defence=false;
                break;}
            case 2: {
                user_defence=true;
                if(p.hp<=p.max_hp-2)
                    p.hp=p.hp+2;
                cout << p.login << " defends himself and regenerate 2 points of hp.";
                break;}
```

```cpp
            case 3:{
                if(p.hp<=p.max_hp-2)
                    p.hp=p.hp+2;
                if(RunAway(p.stats[1], stats[1])==true)
                {
                    p.hp=p.max_hp;
                    if(p.hp<0)
                        p.hp=0;
                    return;
                }
                break;}
            }
            turn=1;
        }
        else if(turn==1)
        {
            if(hp<20)
                y=2;
            else if(rand()%5==0)
                y=2;
            else
                y=1;
            switch (y)
            {
            case 1:{
                int power_of_attack2 = Attack(stats, p.stats, user_defence);
                p.hp= p.hp - power_of_attack2;
                cout << endl << name << " deals " << power_of_attack2 << " damage!";
                enemy_defence=false;
                break;}
            case 2:{
                enemy_defence=true;
                cout << endl << name << " defends himself.";
                break;}
            }
            turn=0;
        }
    }
```

```cpp
    if(p.hp<=0)
    {
        cout << endl << endl << "You lost! You have to wait until you can fight again..." << endl;
        hp=enemy_max_hp;
        if(p.hp<0)
            p.hp=0;
        return;
    }
    else
    {
        cout << endl << endl << "Victory! Congratulation, now you can increase your stats by point." << endl << "You got an experience point. ";
        p.exp++;
        p.ImproveStats();
        cout << endl;
        hp=enemy_max_hp;
        if(p.hp<0)
            p.hp=0;
        return;
    }
}
```

```cpp
    cout << endl << endl << "This is a good moment to try to tame it!" << endl << "Enter 1 if you want to try or 2 if you want to win this duel: ";
    int tmp;
    while(1)
    {
        try
        {
            cin >> tmp;
            if(tmp==1 || tmp==2)
                break;
            else
            throw invalid_argument("error");
        }
        catch(invalid_argument)
        {
            cout << "Try again!" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
    }
    if(tmp==1)
    {
      bool have_pet = Tame(name);
      if(have_pet==true)
      {
          cout << endl << "Success! You have a pet now! It goes home to rest a little... You will be able to play with it in a few hours." << endl;
          p.pet=true;
          p.exp++;
          p.exp++;
```

```cpp
bool Tame(string name)
{
    cout << endl << name << " asks you a riddle... " << endl << "A box without hinges, key, or lid," << endl << "Yet golden treasure inside is hid." << endl;
    cout << endl << "If you think it's geode enter 1, if you think ,,maybe eggs'' - enter 2, if you want to guess it's the core of the Earth - enter 3: ";
    int tmp;
    while(1)
    {
        try
        {
            cin >> tmp;
            if(tmp==1 || tmp==2 || tmp==3)
                break;
            else
            throw invalid_argument("error");
        }
        catch(invalid_argument)
        {
            cout << "Try again!" << endl;
            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            continue;
        }
    }
    if(tmp==2)
        return true;
    else
    {
        cout << endl << "Unfortunately, it's not the correct answer. " << name << " runs away, and you stay alone.";
        return false;
    }
}
```

Did you learn anything new?

What could be improved?

Did you have any problems?

Thanks for your attention!