Федеральное агентство связи

Государственное образовательное учреждение высшего профессионального образования

«Сибирский Государственный Университет Телекоммуникаций и Информатики»

Курсовая работа по ООП

По теме “Битва 2 противников”

Выполнил:

Студенты группы ИП-917

Мухомедзянов Данил

Новосибирск, 2020

**Оглавление:**

О проекте (стр 2)

Принципы ООП (стр 3)

Комментарии к коду (стр 4)

Работа программы (стр 10)

Код (стр 12)

Личный вклад (стр 33)

**Битва 2 противников.**

Данную тему мы выбрали потому, что были заинтересованы в реализации проекта. Мы подготовили ресурсы (в Asset) и приступили к работе.

Мы усложнили задачу и сделали бой нескольких противников (3 на 3) и одной из этих команд управляет пользователь. Также мы добавили повышение характеристик пользователю. Если пользователь выигрывает поединок, то он получает предмет, который повысит одну его характеристику: силу, здоровье, выносливость.

Персонажи могут повысить свои характеристики во время боя. Первый персонаж повышает силу, второй наносит урон и восстанавливает здоровье команде, третий «жертвует» своим здоровьем и получает силу, нанося урон врагу.

Враги же наносят фиксированный урон, что было сделано для баланса в сторону пользователя. Враг с помощью функции из класса выбирает цель и атакует ее, если его здоровье выше 40. Если меньше, то он восстанавливает здоровье.

При разработке мы столкнулись с нехваткой ресурсов в виде текстур.

# ПРИНЦИПЫ ООП

Для проекта мы использовали несколько принципов ООП:

Наследование — [концепция](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%86%D0%B8%D1%8F) [объектно-ориентированного программирования](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), согласно которой [абстрактный тип данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) может наследовать данные и функциональность некоторого существующего типа, способствуя повторному использованию компонентов [программного обеспечения](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).

Полиморфизм – это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта.

Был создан абстрактный класс, в котором описывалось: прорисовка персонажа, если его здоровье больше нуля, и функция, которая возвращала ID персонажа при нажатии. Далее мы наследовали этот класс к team и enemy.

Инкапсуляция – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе и скрыть детали реализации от пользователя.

Также мы использовали конструкторы для всех классов, кроме абстрактного obj.

В private помещен специальный эффект. Когда персонажи в бою повышают характеристики, то его можно увидеть.

## Комментарии к коду

Конструктор по умолчанию для команды выглядит так.

team() {

std::ifstream savedTeam("Save/TeamStat.save");

for (int i = 0; i <= 2; i++) {

animPlay = false;

step = true;

savedTeam >> health[i];

savedTeam >> spirit[i];

savedTeam >> power[i];

savedTeam >> minPower[i];

}

savedTeam.close();

buffTexture.loadFromFile("Asset/Effect.png");

for (int i = 0; i < 3; i++) {

buff[i].setTexture(buffTexture);

buff[i].setScale(2, 2);

}

buff[0].setPosition(150, 392);

buff[1].setPosition(190, 484);

buff[2].setPosition(150, 622);

for (int i = 0; i < 3; i++)

isBuffed[i] = 0;

character.loadFromFile("Asset/Char.png");

teamCharacter[0].setPosition(200, 372);

teamCharacter[1].setPosition(240, 464);

teamCharacter[2].setPosition(200, 600);

for (int i = 0; i < 3; i++) {

teamCharacter[i].setTexture(character);

teamCharacter[i].setTextureRect(sf::IntRect(i \* 100, 0, 100, 100));

}

}

**Он открывает файл и записывает характеристики пользователя в класс**

std::ifstream savedTeam("Save/TeamStat.save");

for (int i = 0; i <= 2; i++) {

animPlay = false;

step = true;

savedTeam >> health[i];

savedTeam >> spirit[i];

savedTeam >> power[i];

savedTeam >> minPower[i];

}

savedTeam.close();

Это сделано для того, чтобы пользователь мог менять характеристики в меню Upgrade

Далее мы ставим текстуры, персонажей и места для интерфейса.

Далее написаны функции для анимации

void playPrepareToAttack(int charNum) {

teamCharacter[charNum].setTextureRect(sf::IntRect(charNum \* 100, 100, 100, 100));

}

Тут мы выбираем персонажа с помощью charNum и «обрезаем» текстуру.

Далее, в классе enemy

int AI(int\* healthPlayer, int enemyToAttack) {

int i, charToAttack = 0;

int minHealth = healthPlayer[0];

for (int j = 0; j < 3; j++) {

if (healthPlayer[j] > minHealth)

minHealth = healthPlayer[j];

}

if (health[enemyToAttack] < 50 && healTime < 3) {

healTime++;

health[enemyToAttack] += 40;

return 5;

}

for (i = 0; i < 3; i++) {

if (healthPlayer[i] <= minHealth && healthPlayer[i] >= 0) {

minHealth = healthPlayer[i];

charToAttack = i;

}

}

return charToAttack;

}

Это выбор противника. По умолчанию он выбирает противника с самым низким здоровьем, но только если его здоровье выше 50. Тогда функция возвращает 5 и противник восстанавливает здоровье.

Далее описан класс Button.

В private помещен шрифт и текст кнопки.

Button(float x, float y, float xend, float yend) {

posX = x;

posY = y;

endX = xend;

endY = yend;

};

Button(float x, float y, float xend, float yend, std::string text) {

posX = x;

posY = y;

endX = xend;

endY = yend;

font.loadFromFile("Asset/FontC.ttf");

buttonText.setString(text);

buttonText.setPosition(x, y);

buttonText.setFont(font);

buttonText.setCharacterSize(20);

};

У него есть 2 конструктора. Один делает полностью функциональную кнопку без текста. Это нужно для инвентаря. А другой с текстом для меню.

И класс inventory

inventory() {

std::ifstream savedInventory("Save/inventory.save");

font.loadFromFile("Asset/FontC.TTF");

for (int i = 0; i < 3; i++) {

savedInventory >> id[i];

item[i].setFont(font);

item[i].setCharacterSize(20);

}

nameId[0] = "Life";

nameId[1] = "Power";

nameId[2] = "Spirit";

x = 520, y = 200;

}

Cтроит инвентарь из файла, записывает называния предметов и дает начальные координаты.

void get(sf::RenderWindow& window, int num) {

id[num] = id[num] + 1;

win.setCharacterSize(30);

win.setFont(font);

win.setPosition(380, 384);

win.setString("You've got Essension of " + nameId[num]);

window.draw(win);

std::ofstream inv("Save/inventory.save");

for (int i = 0; i < 3; i++) {

inv << id[i] << " ";

}

inv.close();

}

С помощью get() можно получить предмет в инвентарь и вывести уведомление об этом.

К самой программе:

Есть несколько функций

main, fight, menu, upgrade.

Из основной функции main можно получить значение из menu, которое отправит в fight или upgrade. Тут инициализируется класс inventory и ожидается победа для получения предмета с помощью get.

while (positionMenu != 2) {

positionMenu = menu(window);

if (positionMenu == 0) {

beneathTheMask.stop();

int idItem = fight(window);

if (idItem != 10) {

beneathTheMask.play();

window.clear();

Inventory.get(window, idItem);

window.display();

Sleep(5000);

}

if (idItem == 10) {

beneathTheMask.play();

window.clear();

window.draw(Escape);

window.display();

Sleep(1000);

}

}

В меню инициализируются кнопки и ожидается нажатие по ним.

В бою вычисляется, чей сейчас ход и производятся операции со здоровьем.

А upgrade повышает уровень персонажей и сохраняет его.

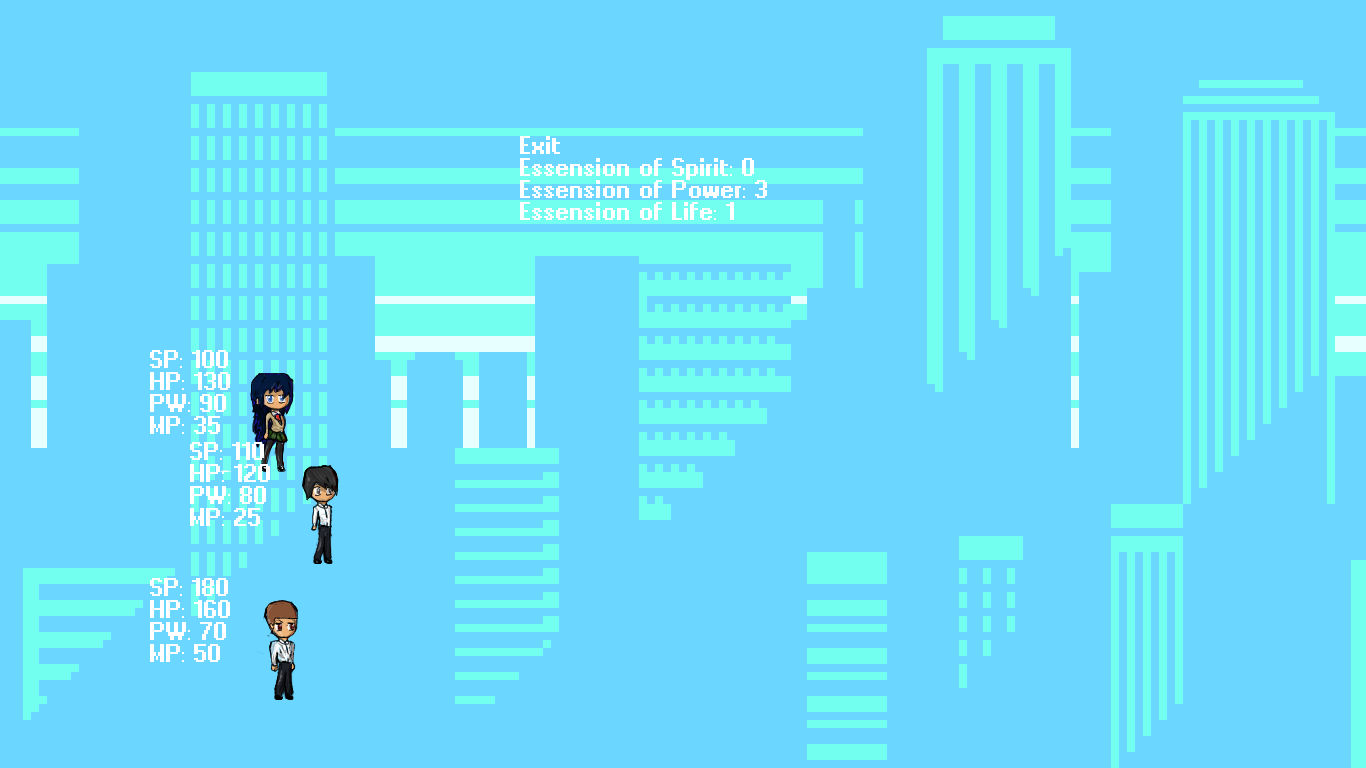
Благодаря принципам ООП функции небольшие и очень простые.

# Работа программы

Так выглядит меню. В нем играет такая же музыка, как и в Upgrade.



В Upgrade предлагается выбрать персонажа и усилить его



Cам Бой



## Код:

#pragma once

#include <SFML/Graphics.hpp>

#include <iostream>

#include <sstream>

#include <fstream>

class obj {

public:

sf::Font font;

int health[3]{};

bool animPlay = NULL;

void draw(sf::RenderWindow& window) {

for (int i = 0; i < 3; i++)

if (health[i] >= 0)

window.draw(teamCharacter[i]);

}

bool alive() {

int j = 0;

for (int i = 0; i < 3; i++) {

if (health[i] < 0)

j++;

}

if (j == 3)

return false;

return true;

}

int onMouse(sf::RenderWindow& window, float X, float forMid) {

if (sf::IntRect(X, 372, 100, 100).contains(sf::Mouse::getPosition(window)) && health[0] > 0) {

if (sf::Mouse::isButtonPressed(sf::Mouse::Left))

return 0; //1100 enem and 200

}

if (sf::IntRect(X - forMid, 464, 100, 100).contains(sf::Mouse::getPosition(window)) && health[1] > 0) {

if (sf::Mouse::isButtonPressed(sf::Mouse::Left))

return 1;

}

if (sf::IntRect(X, 600, 100, 100).contains(sf::Mouse::getPosition(window)) && health[2] > 0) {

if (sf::Mouse::isButtonPressed(sf::Mouse::Left))

return 2;

}

return 3;

}

sf::Sprite teamCharacter[3];

sf::Texture character;

};

class team : public obj{

private:

sf::Sprite buff[3]{};

sf::Texture buffTexture;

int isBuffed[3]{};

public:

int power[3]{};

int minPower[3]{};

int spirit[3]{};

bool step = NULL;

sf::Text hp[3]{}, sp[3]{}, pw[3]{}, minpw[3]{};

team() {

std::ifstream savedTeam("Save/TeamStat.save");

for (int i = 0; i <= 2; i++) {

animPlay = false;

step = true;

savedTeam >> health[i];

savedTeam >> spirit[i];

savedTeam >> power[i];

savedTeam >> minPower[i];

}

savedTeam.close();

buffTexture.loadFromFile("Asset/Effect.png");

for (int i = 0; i < 3; i++) {

buff[i].setTexture(buffTexture);

buff[i].setScale(2, 2);

}

buff[0].setPosition(150, 392);

buff[1].setPosition(190, 484);

buff[2].setPosition(150, 622);

for (int i = 0; i < 3; i++)

isBuffed[i] = 0;

character.loadFromFile("Asset/Char.png");

teamCharacter[0].setPosition(200, 372);

teamCharacter[1].setPosition(240, 464);

teamCharacter[2].setPosition(200, 600);

for (int i = 0; i < 3; i++) {

teamCharacter[i].setTexture(character);

teamCharacter[i].setTextureRect(sf::IntRect(i \* 100, 0, 100, 100));

}

}

void drawBuff(sf::RenderWindow &window) {

for (int i = 0; i < 3; i++) {

if (isBuffed[i] == 1 && health[i] >= 0)

window.draw(buff[i]);

}

}

int skillPower(int characterNum) {

srand(time(NULL));

int skill = rand() % power[characterNum];

if (skill < minPower[characterNum])

skill = minPower[characterNum];

spirit[characterNum] = spirit[characterNum] - 10;

return skill;

}

int chooseSkill(int characterNum, sf::RenderWindow &window) {

int characterEffect = 3;

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Q) && spirit[characterNum] - 10 > 0) {

step = false;

return skillPower(characterNum);

}

if (sf::Keyboard::isKeyPressed(sf::Keyboard::E) && spirit[characterNum] - 5 > 0) {

switch (characterNum) {

case 0:

spirit[characterNum] -= 5;

power[characterNum] += 20;

step = false;

isBuffed[0] = 1;

return 1;

break;

case 1:

spirit[characterNum] -= 5;

while (characterEffect == 3) {

characterEffect = onMouse(window, 200, -40);

}

health[characterEffect] += 45;

power[1] += 5;

step = false;

isBuffed[1] = 1;

return 1;

break;

case 2:

health[characterNum] -= 90;

spirit[characterNum] -= 5;

power[characterNum] += 100;

isBuffed[2] = 1;

step = false;

return 1;

break;

default:

return 0;

}

}

if (sf::Keyboard::isKeyPressed(sf::Keyboard::F))

step = false;

return 0;

}

void setFont() {

font.loadFromFile("Asset/FontC.TTF");

hp[0].setPosition(150, 370);

hp[1].setPosition(190, 462);

hp[2].setPosition(150, 598);

sp[0].setPosition(150, 348);

sp[1].setPosition(190, 440);

sp[2].setPosition(150, 576);

pw[0].setPosition(150, 392);

pw[1].setPosition(190, 484);

pw[2].setPosition(150, 620);

minpw[0].setPosition(150, 414);

minpw[1].setPosition(190, 506);

minpw[2].setPosition(150, 642);

for (int i = 0; i < 3; i++) {

hp[i].setFont(font);

sp[i].setFont(font);

pw[i].setFont(font);

minpw[i].setFont(font);

hp[i].setCharacterSize(20);

sp[i].setCharacterSize(20);

pw[i].setCharacterSize(20);

minpw[i].setCharacterSize(20);

}

}

void playPrepareToAttack(int charNum) {

teamCharacter[charNum].setTextureRect(sf::IntRect(charNum \* 100, 100, 100, 100));

}

void playIdle() {

for (int i = 0; i < 3; i++)

teamCharacter[i].setTextureRect(sf::IntRect(i \* 100, 0, 100, 100));

animPlay = false;

}

void playGetDamage(int charNum) {

animPlay = true;

teamCharacter[charNum].setTextureRect(sf::IntRect(charNum \* 100, 200, 100, 100));

}

void stat(sf::RenderWindow &window) {

std::ostringstream hpString[3], spString[3];

for (int i = 0; i < 3; i++) {

hpString[i] << health[i];

spString[i] << spirit[i];

}

for (int i = 0; i < 3; i++) {

if (health[i] > 0) {

hp[i].setString("HP: " + hpString[i].str());

sp[i].setString("SP: " + spString[i].str());

window.draw(hp[i]);

window.draw(sp[i]);

}

}

}

void fullStat(sf::RenderWindow& window){

stat(window);

std::ostringstream pwString[3], minpwString[3];

for (int i = 0; i < 3; i++) {

pwString[i] << power[i];

minpwString[i] << minPower[i];

}

for (int i = 0; i < 3; i++) {

if (health[i] > 0) {

pw[i].setString("PW: " + pwString[i].str());

minpw[i].setString("MP: " + minpwString[i].str());

window.draw(pw[i]);

window.draw(minpw[i]);

}

}

}

};

class enemy : public obj {

private:

sf::Text hp[3]{};

public:

int healTime;

enemy(int\* healthSet) {

character.loadFromFile("Asset/Shadow.png");

animPlay = false;

healTime = 0;

teamCharacter[0].setPosition(1100, 372);

teamCharacter[1].setPosition(1060, 464);

teamCharacter[2].setPosition(1100, 600);

for (int i = 0; i < 3; i++) {

teamCharacter[i].setTexture(character);

teamCharacter[i].setTextureRect(sf::IntRect(0, 0, 100, 100));

}

for (int i = 0; i < 3; i++) {

health[i] = healthSet[i];

}

}

int AI(int\* healthPlayer, int enemyToAttack) {

int i, charToAttack = 0;

int minHealth = healthPlayer[0];

for (int j = 0; j < 3; j++) {

if (healthPlayer[j] > minHealth)

minHealth = healthPlayer[j];

}

if (health[enemyToAttack] < 50 && healTime < 3) {

healTime++;

health[enemyToAttack] += 40;

return 5;

}

for (i = 0; i < 3; i++) {

if (healthPlayer[i] <= minHealth && healthPlayer[i] >= 0) {

minHealth = healthPlayer[i];

charToAttack = i;

}

}

return charToAttack;

}

void playIdle() {

for (int i = 0; i < 3; i++) {

teamCharacter[i].setTextureRect(sf::IntRect(0, 0, 100, 100));

}

animPlay = false;

}

void playAttack(int enemNum) {

teamCharacter[enemNum].setTextureRect(sf::IntRect(100, 0, 100, 100));

animPlay = true;

}

void playGetDamage(int enemNum) {

teamCharacter[enemNum].setTextureRect(sf::IntRect(200, 0, 100, 100));

animPlay = true;

}

void setFont() {

font.loadFromFile("Asset/FontC.TTF");

hp[0].setPosition(1200, 370);

hp[1].setPosition(1160, 462);

hp[2].setPosition(1200, 598);

for (int i = 0; i < 3; i++) {

hp[i].setFont(font);

hp[i].setCharacterSize(20);

}

}

void stat(sf::RenderWindow& window) {

std::ostringstream hpString[3];

for (int i = 0; i < 3; i++) {

hpString[i] << health[i];

}

for (int i = 0; i < 3; i++) {

if (health[i] > 0) {

hp[i].setString("HP: "+ hpString[i].str());

window.draw(hp[i]);

}

}

}

};

class Button {

private:

sf::Text buttonText;

sf::Font font;

public:

float posX, posY, endX, endY;

Button(float x, float y, float xend, float yend) {

posX = x;

posY = y;

endX = xend;

endY = yend;

};

Button(float x, float y, float xend, float yend, std::string text) {

posX = x;

posY = y;

endX = xend;

endY = yend;

font.loadFromFile("Asset/FontC.ttf");

buttonText.setString(text);

buttonText.setPosition(x, y);

buttonText.setFont(font);

buttonText.setCharacterSize(20);

};

void draw(sf::RenderWindow& window) {

window.draw(buttonText);

}

bool click(sf::RenderWindow& window) {

if (sf::IntRect(posX, posY, endX, endY).contains(sf::Mouse::getPosition(window))) {

buttonText.setFillColor(sf::Color(148, 148, 148));

if (sf::Mouse::isButtonPressed(sf::Mouse::Left))

return true;

}

else

buttonText.setFillColor(sf::Color(255, 255, 255));

return false;

}

};

class inventory {

private:

float x, y;

std::string nameId[3]{};

sf::Text item[3]{};

sf::Font font;

sf::Text win;

public:

int id[3]{};

inventory() {

std::ifstream savedInventory("Save/inventory.save");

font.loadFromFile("Asset/FontC.TTF");

for (int i = 0; i < 3; i++) {

savedInventory >> id[i];

item[i].setFont(font);

item[i].setCharacterSize(20);

}

nameId[0] = "Life";

nameId[1] = "Power";

nameId[2] = "Spirit";

x = 520, y = 200;

}

void get(sf::RenderWindow& window, int num) {

id[num] = id[num] + 1;

win.setCharacterSize(30);

win.setFont(font);

win.setPosition(380, 384);

win.setString("You've got Essension of " + nameId[num]);

window.draw(win);

std::ofstream inv("Save/inventory.save");

for (int i = 0; i < 3; i++) {

inv << id[i] << " ";

}

inv.close();

}

void set() {

for (int i = 0; i < 3; i++) {

std::ostringstream idStr[3];

for (int i = 0; i < 3; i++) {

idStr[i] << id[i];

}

item[i].setString("Essension of " + nameId[i] + ": " + idStr[i].str());

}

}

void setPosition() {

for (int i = 0; i < 3; i++){

item[i].setPosition(x, y);

y -= 22;

}

}

void show(sf::RenderWindow& window) {

for(int i = 0; i < 3; i++)

window.draw(item[i]);

}

};

#include <iostream>

#include <fstream>

#include <SFML/Audio.hpp>

#include <windows.h>

#include "class.h"

int fight(sf::RenderWindow &window) {

int healthEnemy[3];

team Player;

for (int i = 0; i < 3; i++)

healthEnemy[i] = Player.health[i];

enemy Shadow(healthEnemy);

sf::Music battleTheme;

sf::SoundBuffer buffer;

buffer.loadFromFile("Asset/videoplayback.ogg");

sf::Sound hit;

hit.setBuffer(buffer);

battleTheme.openFromFile("Asset/MassDestruction.ogg");

battleTheme.setLoop(true);

battleTheme.play();

sf::Sprite Background;

sf::Texture BackgroundTexture;

BackgroundTexture.loadFromFile("Asset/Background.png");

Background.setTexture(BackgroundTexture);

Player.setFont();

Shadow.setFont();

sf::Text turn("Your turn!", Player.font, 20);

turn.setPosition(200, 200);

int characterNum = 3, enemyChoose = 3, charAttack, enemyToAttack = 0;

int skill = 0, frame = 0, shadowBefore;

while (window.isOpen()) {

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

window.close();

}

window.draw(Background);

if (Player.step == true && Player.animPlay == false && Shadow.animPlay == false) {

if (characterNum == 3) {

characterNum = Player.onMouse(window, 200, -40);

}

if (enemyChoose == 3) {

enemyChoose = Shadow.onMouse(window, 1100, 40);

}

if (enemyChoose != 3 && characterNum != 3) {

Player.playPrepareToAttack(characterNum);

skill = Player.chooseSkill(characterNum, window);

if(skill == 1)

hit.play();

shadowBefore = Shadow.health[enemyChoose];

Shadow.health[enemyChoose] = Shadow.health[enemyChoose] - skill;

}

window.draw(turn);

}

if (Player.step == false && Player.animPlay == false && Shadow.animPlay == false && Shadow.alive() == true) {

if (skill > 0)

Shadow.playGetDamage(enemyChoose);

Player.playIdle();

while (Shadow.health[enemyToAttack] < 0) {

enemyToAttack++;

if (enemyToAttack > 2)

enemyToAttack = 0;

}

Shadow.playAttack(enemyToAttack);

charAttack = Shadow.AI(Player.health, enemyToAttack);

Player.playGetDamage(charAttack);

if (charAttack != 5) {

Player.health[charAttack] = Player.health[charAttack] - 40;

}

enemyToAttack++;

characterNum = 3;

enemyChoose = 3;

Player.step = true;

}

Player.draw(window);

Player.stat(window);

Player.drawBuff(window);

Shadow.draw(window);

Shadow.stat(window);

window.display();

if (Player.animPlay == true || Shadow.animPlay == true) {

Sleep(200);

Player.playIdle();

Shadow.playIdle();

}

if (Player.alive() == false) {

return 10;

}

if (Shadow.alive() == false) {

return rand() % 3;

}

if (sf::Keyboard::isKeyPressed(sf::Keyboard::Escape))

return 10;

}

return 10;

}

#pragma once

#include <SFML/Graphics.hpp>

int fight(sf::RenderWindow& window);

int upgradeMenu(sf::RenderWindow& window);

int menu(sf::RenderWindow& window);

#include "class.h"

#include <fstream>

int upgradeMenu(sf::RenderWindow& window) {

team Player;

inventory PlayersInventory;

sf::Sprite Background;

sf::Texture BackTexture;

Button itemLife(520, 200, 340, 20);

Button itemPower(520, 178, 340, 20);

Button itemSpirit(520, 156, 340, 20);

Button Exit(520, 134, 340, 20, "Exit");

BackTexture.loadFromFile("Asset/Upgrade.png");

Background.setTexture(BackTexture);

int charNum = 3;

PlayersInventory.setPosition();

Player.setFont();

bool delay = false;

while (window.isOpen()) {

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

window.close();

}

PlayersInventory.set();

window.clear();

if (Player.onMouse(window, 200, -40) != 3)

charNum = Player.onMouse(window, 200, -40);

if (itemLife.click(window) == true && charNum != 3 && PlayersInventory.id[0] > 0 && delay == false) {

Player.health[charNum] += 10;

PlayersInventory.id[0] -= 1;

delay = true;

}

if (itemPower.click(window) == true && charNum != 3 && PlayersInventory.id[1] > 0 && delay == false) {

Player.power[charNum] += 10;

Player.minPower[charNum] += 10;

PlayersInventory.id[1] -= 1;

delay = true;

}

if (itemSpirit.click(window) == true && charNum != 3 && PlayersInventory.id[2] > 0 && delay == false) {

Player.spirit[charNum] += 10;

PlayersInventory.id[2] -= 1;

delay = true;

}

if (itemSpirit.click(window) == false && itemPower.click(window) == false && itemLife.click(window) == false) {

delay = false;

}

if (Exit.click(window) == true) {

std::ofstream team("Save/TeamStat.save");

std::ofstream inv("Save/inventory.save");

for (int i = 0; i < 3; i++) {

inv << PlayersInventory.id[i] << " ";

team << Player.health[i] << " " << Player.spirit[i] << " " << Player.power[i] << " " << Player.minPower[i] << " ";

}

inv.close();

team.close();

return 0;

}

window.draw(Background);

Exit.draw(window);

PlayersInventory.show(window);

Player.draw(window);

Player.fullStat(window);

window.display();

}

return 1;

}

#include <SFML/Audio.hpp>

#include <windows.h>

#include "class.h"

#include "function.h"

int main() {

sf::RenderWindow window(sf::VideoMode(1366, 768), "Battle of 2 Opponents", sf::Style::Fullscreen);

window.setVerticalSyncEnabled(true);

inventory Inventory;

sf::Sprite Escape;

Escape.setPosition(227, 300);

sf::Texture escTexture;

sf::Music beneathTheMask;

beneathTheMask.openFromFile("Asset/BeneathTheMask.ogg");

beneathTheMask.setLoop(true);

beneathTheMask.play();

escTexture.loadFromFile("Asset/Escape.png");

Escape.setTexture(escTexture);

int positionMenu = NULL;

while (positionMenu != 2) {

positionMenu = menu(window);

if (positionMenu == 0) {

beneathTheMask.stop();

int idItem = fight(window);

if (idItem != 10) {

beneathTheMask.play();

window.clear();

Inventory.get(window, idItem);

window.display();

Sleep(5000);

}

if (idItem == 10) {

beneathTheMask.play();

window.clear();

window.draw(Escape);

window.display();

Sleep(1000);

}

}

if (positionMenu == 1)

upgradeMenu(window);

}

}

#include "class.h"

int menu(sf::RenderWindow& window) {

Button startGame(1246, 600, 100, 20, "Start");

Button upgrade(1246, 642, 140, 20, "Upgrade");

Button exit(1246, 684, 80, 20, "Exit");

sf::Sprite background, cloud, logo;

cloud.setScale(0.5, 0.5);

sf::Texture backTexture, cloudTexture, logoTexture;

backTexture.loadFromFile("Asset/Menu.png");

cloudTexture.loadFromFile("Asset/cloud.png");

logoTexture.loadFromFile("Asset/logo.png");

logo.setTexture(logoTexture);

logo.setPosition(30, 30);

cloud.setTexture(cloudTexture);

float x = 0;

background.setTexture(backTexture);

while (window.isOpen()) {

sf::Event event;

while (window.pollEvent(event))

{

if (event.type == sf::Event::Closed)

window.close();

}

window.clear();

cloud.setPosition(x, 0);

x += 0.9;

if (x > 2000)

x = -2200;

if (startGame.click(window) == true)

return 0;

if (upgrade.click(window) == true)

return 1;

if (exit.click(window) == true)

return 2;

window.draw(background);

window.draw(cloud);

window.draw(logo);

startGame.draw(window);

upgrade.draw(window);

exit.draw(window);

window.display();

}

return 2;

}

## Личный вклад:

Создание ассетов для проекта.

Написание основ в функции fight и разработка классов team и enemy.

Создание структуры проекта.

Запись характеристик в файл.

Направление команды.