#include <SFML/Graphics.hpp>

//#include "map.h"

#include "view.h"

#include <iostream>

#include <sstream>

#include "mission.h"

#include "LifeBar.h"

#include "iostream"

#include "level.h"

#include <vector>

#include <list>

#include "TinyXML/tinyxml.h"

using namespace sf;

////////////////////////////////////Общий класс-родитель//////////////////////////

class Entity {

public:

std::vector<Object> obj;//вектор объектов карты

float dx, dy, x, y, speed, moveTimer;

int w, h, health;

bool life, isMove, onGround;

Texture texture;

Sprite sprite;

String name;

bool isShootEnemy;

//bool isShootPlayer;

float currentFrame;//текущий кадр для анимации

Entity(Image& image, String Name, float X, float Y, int W, int H) {

x = X; y = Y; w = W; h = H; name = Name; moveTimer = 0;

speed = 0; health = 100; dx = 0; dy = 0;

life = true; onGround = false; isMove = false; isShootEnemy = true;

texture.loadFromImage(image);

sprite.setTexture(texture);

sprite.setOrigin(w / 2, h / 2);

}

FloatRect getRect() {//ф-ция получения прямоугольника. его коорд,размеры (шир,высот).

return FloatRect(x, y, w, h);//эта ф-ция нужна для проверки столкновений

}

virtual void update(float time) = 0;//Виртуальная функция — это функция, объявленная с ключевым словом virtual в базовом классе и переопределенная в одном или в нескольких производных классах.

};

////////////////////////////////////////////////////КЛАСС ИГРОКА////////////////////////

class Player :public Entity {

public:

enum { left, right, up, down, jump, stay } state;

int playerScore;

bool isShootPlayer;

Player(Image& image, String Name, Level& lev, float X, float Y, int W, int H) :Entity(image, Name, X, Y, W, H) {

playerScore = 0; state = stay; obj = lev.GetAllObjects();//инициализируем.получаем все объекты для взаимодействия персонажа с картой

if (name == "Player1") {

sprite.setTextureRect(IntRect(0, 254, w, h));

}

}

void control(float &time) {

if (Keyboard::isKeyPressed) {

if (Keyboard::isKeyPressed(Keyboard::Space))

{

isShootPlayer = true;

}

if (Keyboard::isKeyPressed(Keyboard::Left)) {

state = left; speed = 0.1;

currentFrame += 0.01 \* (time);

if (currentFrame > 4) currentFrame -= 4;

if (dx < 0) sprite.setTextureRect(IntRect(40 \* int(currentFrame) + 40, 254, -40, 40));

}

if (Keyboard::isKeyPressed(Keyboard::Right)) {

state = right; speed = 0.1;

currentFrame += 0.010 \* (time);

if (currentFrame > 4) currentFrame -= 4;

if (dx > 0) sprite.setTextureRect(IntRect(40 \* int(currentFrame), 254, 40, 40));//будем менять первую координату (координата текстуры сдвигается каждый раз на 40)

}

if ((Keyboard::isKeyPressed(Keyboard::Up)) && (onGround)) {

state = jump; dy = -0.6; onGround = false;

}

if (Keyboard::isKeyPressed(Keyboard::Down)) {

state = down;

}

}

}

void checkCollisionWithMap(float Dx, float Dy)

{

for (int i = 0; i < obj.size(); i++)//проходимся по объектам

if (getRect().intersects(obj[i].rect))//проверяем пересечение игрока с объектом

{

if (obj[i].name == "solid")//если встретили препятствие

{

if (Dy > 0) { y = obj[i].rect.top - h; dy = 0; onGround = true; }

if (Dy < 0) { y = obj[i].rect.top + obj[i].rect.height; dy = 0; }

if (Dx > 0) { x = obj[i].rect.left - w; }

if (Dx < 0) { x = obj[i].rect.left + obj[i].rect.width;}

}

if (obj[i].name == "thorn")//если встретили препятствие

{

dy = -0.5;

health -= 5;

}

}

}

void update(float time)

{

control(time);

switch (state)

{

case right:dx = speed; break;

case left:dx = -speed; break;

case up: break;

case down: dx = 0; break;

case stay: break;

}

x += dx \* time;

checkCollisionWithMap(dx, 0);

y += dy \* time;

checkCollisionWithMap(0, dy);

sprite.setPosition(x + w / 2, y + h / 2);

if (health <= 0) { life = false; speed = 0; }

if (!isMove) { speed = 0; }

setPlayerCoordinateForView(x, y);

if (life) { setPlayerCoordinateForView(x, y); }

dy = dy + 0.0015 \* time;

}

};

class Enemy :public Entity {

public:

//bool isShootEnemy;

Enemy(Image& image, String Name, Level& lvl, float X, float Y, int W, int H) :Entity(image, Name, X, Y, W, H) {

obj = lvl.GetObjects("solid");//инициализируем.получаем нужные объекты для взаимодействия врага с картой

if (name == "easyEnemy")

{

health = 150;

sprite.setTextureRect(IntRect(0, 0, w, h));

dx = 0.1;

//isShootEnemy = false;

}

}

void checkCollisionWithMap(float Dx, float Dy)

{

for (int i = 0; i < obj.size(); i++)//проходимся по объектам

if (getRect().intersects(obj[i].rect))//проверяем пересечение игрока с объектом

{

if (obj[i].name == "solid")

{//если встретили препятствие (объект с именем solid)

if (Dy > 0) { y = obj[i].rect.top - h; dy = 0; onGround = true; }

if (Dy < 0) { y = obj[i].rect.top + obj[i].rect.height; dy = 0; }

if (Dx > 0) { x = obj[i].rect.left - w; dx = -0.1; /\*sprite.scale(-1, 1);\*/ }

if (Dx < 0) { x = obj[i].rect.left + obj[i].rect.width; dx = 0.1; /\*sprite.scale(-1, 1);\*/ }

}

}

}

void update(float time)

{

if (name == "easyEnemy") {

if (life == true)

{

if (health >= 50 )

{

//moveTimer += time;if (moveTimer>3000){ dx \*= -1; moveTimer = 0; }//меняет направление примерно каждые 3 сек

checkCollisionWithMap(dx, 0);

x += dx \* time;

if (dx < 0)

{

currentFrame += 0.01 \* (time);

if (currentFrame > 6) currentFrame -= 6;

sprite.setTextureRect(IntRect(57.5 \* int(currentFrame), 63, 52, 52));

}

if (dx > 0)

{

currentFrame += 0.01 \* (time);

if (currentFrame > 6) currentFrame -= 6;

sprite.setTextureRect(IntRect(57.5 \* int(currentFrame) + 52, 63, -52, 52));

}

if (!isShootEnemy)

{

moveTimer += time;

if (moveTimer >= 1000)

{

isShootEnemy = true;

moveTimer = 0;

}

}

}

if (health <=50) //|| health <50)

{

isShootEnemy = false;

dx = 0;

currentFrame += 0.007 \* (time);

if (currentFrame > 6) currentFrame \*= dx;

sprite.setTextureRect(IntRect(57.5 \* int(currentFrame) + 52, 428, -52, 52));

}

sprite.setPosition(x + w / 2, y + h / 2);

//if (health >=30 || health<=60)// ТУТ ДОБАВЛЮ НОВУЮ АНИМАЦИЮ С РАНЕНЫМ ЧЕЛИКОМ И НОВОЙ СКОРОСТЬ ПЕРЕДВИЖЕНИЯ { life = false; }

if (health <= 0) { life = false; }

}

if (life == false)//ЕЩЕ РАЗ ВЫСТРЕЛИТЬ ЧТОБЫ УДАЛИТЬ ОКОНЧАТЕЛЬНО ВРАГА

{

dx = 0;

currentFrame += 0.007 \* (time);

if (currentFrame > 6) currentFrame \*= dx;

sprite.setTextureRect(IntRect(57.5 \* int(currentFrame) + 52, 428, -52, 52));

}

}

}

};

class MovingPlatform : public Entity {//класс движущейся платформы

public:

MovingPlatform(Image& image, String Name, Level& lvl, float X, float Y, int W, int H) :Entity(image, Name, X, Y, W, H) {

sprite.setTextureRect(IntRect(0, 0, W, H));//прямоугольник

dx = 0.08;//изначальное ускорение по Х

}

void update(float time)//функция обновления платформы.

{

x += dx \* time;//реализация движения по горизонтали

moveTimer += time;//наращиваем таймер

if (moveTimer > 2000) { dx \*= -1; moveTimer = 0; }//если прошло примерно 2 сек, то меняется направление движения платформы,а таймер обнуляется

sprite.setPosition(x + w / 2, y + h / 2);//задаем позицию спрайту

}

};

class Bullet :public Entity {//класс пули

public:

int direction;//направление пули

Bullet(Image& image, String Name, Level& lvl, float X, float Y, int W, int H, int dir) :Entity(image, Name, X, Y, W, H) {//всё так же, только взяли в конце состояние игрока (int dir)

obj = lvl.GetObjects("solid");//инициализируем .получаем нужные объекты для взаимодействия пули с картой

x = X;

y = Y;

direction = dir;

speed = 0.1;

w = h = 36;

life = true;

//выше инициализация в конструкторе

}

void update(float time)

{

if (name == "Bullet")

{

speed = 0.3;

switch (direction)

{

case 0: dx = -speed; dy = 0; break;//интовое значение state = left

case 1: dx = speed; dy = 0; break;//интовое значение state = right

case 2: dx = 0; dy = -speed; break;//интовое значение state = up

case 3: dx = 0; dy = -speed; break;//интовое значение не имеющее отношения к направлению, пока просто стрельнем вверх, нам сейчас это не важно

case 4: dx = 0; dy = -speed; break;//интовое значение не имеющее отношения к направлению, пока просто стрельнем вверх, нам сейчас это не важно

case 5: dx = 0; dy = -speed; break;//интовое значение не имеющее отношения к направлению, пока просто стрельнем вверх, нам сейчас это не важно

}

x += dx \* time;//само движение пули по х

y += dy \* time;//по у

if (x <= 0) x = 1;// задержка пули в левой стене, чтобы при проседании кадров она случайно не вылетела за предел карты и не было ошибки

if (y <= 0) y = 1;

for (int i = 0; i < obj.size(); i++) {//проход по объектам solid

if (getRect().intersects(obj[i].rect)) //если этот объект столкнулся с пулей,

{

life = false;// то пуля умирает

}

}

sprite.setPosition(x + w / 2, y + h / 2);//задается позицию пуле

}

if (name == "BulletEnemy")

{

speed = 0.2;

switch (direction)

{

case 0: dx = -speed; dy = 0; break;//интовое значение state = left

case 1: dx = speed; dy = 0; break;//интовое значение state = right

case 2: dx = 0; dy = -speed; break;//интовое значение state = up

case 3: dx = 0; dy = -speed; break;//интовое значение не имеющее отношения к направлению, пока просто стрельнем вверх, нам сейчас это не важно

case 4: dx = 0; dy = -speed; break;//интовое значение не имеющее отношения к направлению, пока просто стрельнем вверх, нам сейчас это не важно

case 5: dx = 0; dy = -speed; break;//интовое значение не имеющее отношения к направлению, пока просто стрельнем вверх, нам сейчас это не важно

}

x += dx \* time;//само движение пули по х

y += dy \* time;//по у

if (x <= 0) x = 1;// задержка пули в левой стене, чтобы при проседании кадров она случайно не вылетела за предел карты и не было ошибки

if (y <= 0) y = 1;

for (int i = 0; i < obj.size(); i++) {//проход по объектам solid

if (getRect().intersects(obj[i].rect)) //если этот объект столкнулся с пулей,

{

life = false;// то пуля умирает

}

}

sprite.setPosition(x + w / 2, y + h / 2);//задается позицию пуле

}

}

};

int main()

{

RenderWindow window(VideoMode(640, 480), "Lesson 22. kychka-pc.ru");

view.reset(FloatRect(0, 0, 640, 480));

Level lvl;//создали экземпляр класса уровень

lvl.LoadFromFile("map.tmx");//загрузили в него карту, внутри класса с помощью методов он ее обработает.

std::list<Entity\*> entities;//создаю список, сюда буду кидать объекты.например врагов.

std::list<Entity\*>::iterator it;//итератор чтобы проходить по эл-там списка

std::vector<Object> e = lvl.GetObjects("easyEnemy");//все объекты врага на tmx карте хранятся в этом векторе

LifeBar lifeBarPlayer;

Image heroImage;

heroImage.loadFromFile("images/fang1.png");

Image easyEnemyImage;

easyEnemyImage.loadFromFile("images/дементор2.png");

easyEnemyImage.createMaskFromColor(Color(255, 0, 0));\

for (int i = 0; i < e.size(); i++)//проходимся по элементам этого вектора(а именно по врагам)

entities.push\_back(new Enemy(easyEnemyImage, "easyEnemy", lvl, e[i].rect.left, e[i].rect.top, 52, 52));//и закидываем в список всех наших врагов с карты

Image BulletImage;//изображение для пули

BulletImage.loadFromFile("images/син.png");//загрузили картинку в объект изображения

BulletImage.createMaskFromColor(Color(0, 0, 0));//маска для пули по черному цвету

Image BulletEnemyImage;//изображение для пули

BulletEnemyImage.loadFromFile("images/красный.png");//загрузили картинку в объект изображения

BulletEnemyImage.createMaskFromColor(Color(0, 0, 0));//маска для пули по черному цвету

Image movePlatformImage;

movePlatformImage.loadFromFile("images/MovingPlatform.png");

Object player = lvl.GetObject("player");//объект игрока на нашей карте.задаем координаты игроку в начале при помощи него

//Object easyEnemyObject = lvl.GetObject("easyEnemy");//объект легкого врага на нашей карте.задаем координаты игроку в начале при помощи него

Player p(heroImage, "Player1", lvl, player.rect.left, player.rect.top, 32, 40);//передаем координаты прямоугольника player из карты в координаты нашего игрока

//Enemy easyEnemy(easyEnemyImage, "EasyEnemy", lvl, easyEnemyObject.rect.left, easyEnemyObject.rect.top, 200, 97);//передаем координаты прямоугольника easyEnemy из карты в координаты нашего врага

e = lvl.GetObjects("MovingPlatform");//забираем все платформы в вектор

for (int i = 0; i < e.size(); i++) entities.push\_back(new MovingPlatform(movePlatformImage, "MovingPlatform", lvl, e[i].rect.left, e[i].rect.top,180, 50));//закидываем платформу в список.передаем изображение имя уровень координаты появления (взяли из tmx карты), а так же размеры

Clock clock;

while (window.isOpen())

{

float time = clock.getElapsedTime().asMicroseconds();

clock.restart();

time = time / 800;

Event event;

while (window.pollEvent(event))

{

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

window.close();

////////////////пуля событие////////////

if (p.isShootPlayer == true)

{

p.isShootPlayer = false;

entities.push\_back(new Bullet(BulletImage, "Bullet", lvl, p.x, p.y,36,36,p.state));//если выстрелили, то появляется пуля, unam передаем как int

}

/\*for (it = entities.begin(); it != entities.end(); it++) {

if ((\*it)->name == "easyEnemy")

{

if (p.state == p.right)

{

if (FloatRect(p.x, p.y, p.w \* 3, p.h).intersects((\*it)->getRect()))

//if(((\*it)->getRect()).intersects(IntRect(p.x, p.y, p.w \* 3, p.h)))

{

if ((\*it)->isShootEnemy == true) { (\*it)->isShootEnemy = false; entities.push\_back(new Bullet(BulletImage, "Bullet", lvl, (\*it)->x, (\*it)->y, 36, 36, p.left)); }

}

}

else if (p.state == p.left)

{

//if (((\*it)->getRect()).intersects(IntRect(p.x, p.y, p.w \* 3, p.h)))

if (FloatRect(p.x - p.w \* 3, p.y, p.w \* 3, p.h).intersects((\*it)->getRect()))

if ((\*it)->isShootEnemy == true) { (\*it)->isShootEnemy = false; entities.push\_back(new Bullet(BulletImage, "Bullet", lvl, (\*it)->x, (\*it)->y, 36, 36, p.right)); }

}

}

}\*/

///////////////

}

lifeBarPlayer.update(100);//сюда передаем значение, которое надо нарисовать. Можно передать здоровья игрока тогда будет lifeBarPlayer.update(player.getHealth()); так

p.update(time);

for (it = entities.begin(); it != entities.end(); it++) { (\*it)->update(time); }//для всех элементов списка(пока это только враги,но могут быть и пули к примеру) активируем ф-цию update

window.setView(view);

window.clear(Color(77, 83, 140));

lvl.Draw(window);//рисуем новую карту

/////////////////////Если я захочу удалять ВСЕ объекты при их смерти

for (it = entities.begin(); it != entities.end();)//говорим что проходимся от начала до конца

{

Entity\* b = \*it;//для удобства, чтобы не писать (\*it)->

//b->update(time);//вызываем ф-цию update для всех объектов (по сути для тех, кто жив)

if (b->life == false) { it = entities.erase(it); delete b; }// если этот объект мертв, то удаляем его

else it++;//и идем курсором (итератором) к след объекту. так делаем со всеми объектами списка

//////////////////////////////////

}

for (it = entities.begin(); it != entities.end(); it++) {

////////////////////////////////////////какой-то код про убийство врага

//////////////////// ПОКА НЕ РАБОТАЕТ///// РАБОТАЕТ!!!!!

if ((\*it)->name == "easyEnemy")

{

Entity\* enemy = \*it;

for (std::list<Entity\*>::iterator it2 = entities.begin(); it2 != entities.end(); it2++)

{

Entity\* bullet = \*it2;

if (bullet->name == "Bullet")

if (bullet->life == true)

{

if (bullet->getRect().intersects(enemy->getRect()))

{

bullet->life = false;

std::cout << bullet->life << std::endl;

enemy-> /\*life = false;\*/ health -= 50;

}

}

}

}

///////////////////////////////////////////////////////

if ((\*it)->name == "easyEnemy")

{

if (p.state == p.right)// && p.state == p.stay)//Не работает!! все время стреляют вправо!!! и пропал один враг!

{

if (FloatRect(p.x, p.y, p.w \* 5, p.h\*2).intersects((\*it)->getRect()))

//if(((\*it)->getRect()).intersects(IntRect(p.x, p.y, p.w \* 3, p.h)))

{

if ((\*it)->isShootEnemy == true) { (\*it)->isShootEnemy = false; entities.push\_back(new Bullet(BulletEnemyImage, "BulletEnemy", lvl, (\*it)->x, (\*it)->y, 36, 36, p.left)); }

}

}

else if (p.state == p.left)

{

//if (((\*it)->getRect()).intersects(IntRect(p.x, p.y, p.w \* 3, p.h)))

if (FloatRect(p.x - p.w , p.y, p.w \* 5, p.h\*2).intersects((\*it)->getRect()))

if ((\*it)->isShootEnemy == true) { (\*it)->isShootEnemy = false; entities.push\_back(new Bullet(BulletEnemyImage, "BulletEnemy", lvl, (\*it)->x, (\*it)->y, 36, 36, p.right)); }

}

}

if (((\*it)->name == "MovingPlatform") && ((\*it)->getRect().intersects(p.getRect())))//если игрок столкнулся с объектом списка и имя этого объекта movingplatform

{

Entity\* movPlat = \*it;

if ((p.dy > 0) || (p.onGround == false))//при этом игрок находится в состоянии после прыжка, т.е падает вниз

if (p.y + p.h < movPlat->y + movPlat->h)//если игрок находится выше платформы, т.е это его ноги минимум (тк мы уже проверяли что он столкнулся с платформой)

{

p.y = movPlat->y - p.h +1; p.x += movPlat->dx \* time; p.dy = 0; p.onGround = true; // то выталкиваем игрока так, чтобы он как бы стоял на платформе

}

}

window.draw((\*it)->sprite); //рисуем entities объекты (сейчас это только враги)

}

lifeBarPlayer.draw(window);//рисуем полоску здоровья (РАБОТАЕТ)

window.draw(p.sprite);

window.display();

}

return 0;

}