Приложение Б

**Enemies.cpp**

#include "SFML/Graphics.hpp" /// подключаем заголовочный файл, который отвечает за работу с графикой

#include <iostream>

#include "Map.h"

#include "globals.h"

#include "Enemies.h"

#include "Player.h"

Boy::Boy(int x, int y) {

t.loadFromFile("images/boy.png");

sprite.setTexture(t);

sprite.setTextureRect(IntRect(0, 0, 23, 23));

rect = FloatRect(x \* 32, y \* 32, 0, 0);

dy = -0.08;

}

Girl::Girl(int x, int y)

{

t.loadFromFile("images/girl.png");

sprite.setTexture(t);

sprite.setTextureRect(IntRect(0, 0, 23, 23));

rect = FloatRect(x \* 32, y \* 32, 0, 0);

dx = -0.05;

}

void Child\_Enemies::checkCollisionWithMap()

{

for (int i = (rect.top) / 32; i < (rect.top + 32) / 32; i++)

for (int j = (rect.left) / 32; j < (rect.left + 32) / 32; j++)

{

if (TileMap[i][j] == '0' || TileMap[i][j] == '1' || TileMap[i][j] == 'e' || TileMap[i][j] == 'y' || TileMap[i][j] == 'w' || TileMap[i][j] == 'm')

{

if (dx > 0) { rect.left = j \* 32 - 32; }

if (dx < 0) { rect.left = j \* 32 + 32; }

dx = dx \* -1;

if (dy > 0) { rect.top = i \* 32 - 32; }

if (dy < 0) { rect.top = i \* 32 + 32; }

dy = dy \* -1;

}

}

}

void Child\_Enemies::drawing(float pX, float pY, Player & playerhp)

{

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

clock.restart();

time = time / 800;

sprite.setPosition(rect.left, rect.top);

rect.left = rect.left + dx \* time;

rect.top = rect.top + dy \* time;

checkCollisionWithMap();

CurrentFrame += 0.005 \* time;

if (CurrentFrame > 3) CurrentFrame -= 3;

if (dx < 0) sprite.setTextureRect(IntRect(32 \* int(CurrentFrame) + 0, 32, 32, 32));

if (dx > 0) sprite.setTextureRect(IntRect(32 \* int(CurrentFrame) + 0, 64, 32, 32));

if (dy < 0) sprite.setTextureRect(IntRect(32 \* int(CurrentFrame), 64, 32, 32));

if (dy > 0) sprite.setTextureRect(IntRect(32 \* int(CurrentFrame) + 32, 32, -32, 32));

window.draw(sprite);

if ((pY >= (rect.top - 15)) && (pY <= (rect.top + 32 + 15)) && (pX >= (rect.left - 15) && (pX <= (rect.left + 32 + 15)))) {

playerhp.Damage();

}

}

**Main.cpp**

#include <iostream>

#include <SFML/Graphics.hpp> /// подключаем заголовочный файл, который отвечает за работу с графикой

#include "map.h"

#include "globals.h"

#include "player.h"

#include "Enemies.h"

#include <list>

#include "Shooting.h"

using namespace std;

using namespace sf;

void winner(RenderWindow & window) // пояление окна победителя

{

Texture winning\_screen;

winning\_screen.loadFromFile("images/win.png"); // загружаем изображение

Sprite winscreen(winning\_screen);

winscreen.setPosition(32, 32);

window.draw(winscreen);

window.display(); // выводим на экран

}

void lose(RenderWindow & window) // появление окна проигравшего

{

Texture losee;

losee.loadFromFile("images/lose.jpg");

Sprite lose(losee);

lose.setPosition(32, 32);

window.draw(lose);

window.display();

}

int main()

{

Map drawing;

Player z("santa.png", 48, 48, 23.0, 23.0);

Map m;

Clock clock;

//КОНТЕЙНЕР С ИСПОЛЬЗОВАНИЕМ СПИСКА LIST ( ДЕТИ, которые могут нас заметить)

std::list<Child\_Enemies\*> child; //чтобы каждый раз не создавать врага, то они типа хранятся в контейнере, и их можно сколько угодно раз

std::list<Child\_Enemies\*>::iterator cc;

//КОНТЕЙНЕР С ИСПОЛЬЗОВАНИЕМ СПИСКА LIST ( СТРЕЛКИ, которые могут в нас выстрелить)

std::list<Shooting\*> shoott;

std::list<Shooting\*>::iterator shh;

//КОНТЕЙНЕР С ИСПОЛЬЗОВАНИЕМ СПИСКА LIST (снаряды стреляющих врагов)

std::list<Shooting::Bullet\*> bullets;

std::list<Shooting::Bullet\*>::iterator bul;

//НА КАРТЕ РАССТАВЛЯЕМ НАШИХ "ВРАГОВ"

for (int i = 0; i < HEIGHT\_MAP; i++) {

for (int j = 0; j < WIDTH\_MAP; j++) {

//МАЛЬЧИКИ

if (TileMap[i][j] == 'L') { //Если на карте в Maps поставлена "L", то появляется новый враг (мальчик)

child.push\_back(new Boy(j, i)); // возвращаем из контейнера ребенка - мальчика

}

//ДЕВОЧКИ

if (TileMap[i][j] == 'S') {

child.push\_back(new Girl(j, i));// возвращаем из контейнера ребенка - девочка

}

//СТРЕЛКИ

if (TileMap[i][j] == 'P') {

shoott.push\_back(new Shoot(j, i)); // возвращаем из контейнера пулю

}

}

}

Font font; // шрифт

font.loadFromFile("Ben\_Krush.ttf");

Text text("", font, 20); // размер текста

text.setStyle(sf::Text::Bold); // стиль текста

while (window.isOpen()) // пока окно отрыто

{

if (z.win >= 1) //в случае победы появляется окно победы

{

window.close();

RenderWindow window(sf::VideoMode(789, 514), "WIN");

winner(window);

while (!Mouse::isButtonPressed(Mouse::Right));

}

if (z.hp <= 0) // в случае проигрыша появляется окно проигрыша

{

window.close();

RenderWindow window(sf::VideoMode(1000, 667), "LOSE");

lose(window);

while (!Mouse::isButtonPressed(Mouse::Right));

}

sf::Event event;

while (window.pollEvent(event))

{

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

window.close();

}

window.clear(Color(33, 30, 30));

m.drawing(); //рисуется карта (пока окно открыто)

z.draw\_p(); //рисуется игрок

for (cc = child.begin(); cc != child.end(); cc++) {

(\*cc)->drawing(z.GetPlayerCoordinateX(), z.GetPlayerCoordinateY(), z);

}

for (shh = shoott.begin(); shh != shoott.end(); shh++) {

(\*shh)->draw();

if ((\*shh)->shoot > 300) {

(\*shh)->shoot = 0;

FloatRect polozh = (\*shh)->position();

float uskor = (\*shh)->acceleration();

bullets.push\_back(new Shooting::Bullet(polozh, uskor));

}

(\*shh)->shoot++; //новые пули вылетают

}

for (bul = bullets.begin(); bul != bullets.end(); bul++) {

if ((\*bul)->life) {

(\*bul)->WithPlayer(z); //взаимодействие пуль с игроком

(\*bul)->drawing();

}

else {

delete((\*bul)); //пуля удаляется ударяясь об стену, взамен выходит новая

bullets.remove(\*bul);

break;

}

}

text.setString("ЗДОРОВЬЕ:");

text.setPosition(5, 0);

window.draw(text);

window.draw(z.hpbar);

text.setString("КЛЮЧИ:");

text.setPosition(800, 0);

window.draw(text);

window.draw(z.keybar);

window.display();

}

return 0;

}

**Map.cpp**

#include <SFML/Graphics.hpp> /// подключаем заголовочный файл, который отвечает за работу с графикой

#include "globals.h"

#include "Map.h"

#include <fstream>

#include <string>

RenderWindow window(sf::VideoMode(1380, 770), "Happy New Year"); // создаём функцию, которая будет отвечать за появление окна игры

/////// рисуем карту

Map::Map()//конструктор

{

image1.loadFromFile("images/map2.png"); // подгружаем изображение для нашей карты

map.loadFromImage(image1); // в заголовочном файле Map.h у был создан объект map. Передаем в него объект image 1

s\_map.setTexture(map); // в объект s\_map передаем текстуру

Clock clock; // создаем переменную времени

}

Map::~Map() // создаём деструктор

{

}

void Map::drawing() {

for (int i = 0; i < HEIGHT\_MAP; i++) // рисуем карту по высоте и по ширине

for (int j = 0; j < WIDTH\_MAP; j++)

{

// ниже рисуется карта по символам, поэтому когда тот символ равен тому-то, то рисуется определённый элемент

if (TileMap[i][j] == ' ') s\_map.setTextureRect(IntRect(0, 32, 32, 32));//дорога

if (TileMap[i][j] == '2') s\_map.setTextureRect(IntRect(32, 32, 32, 32)); //деревянный пол

if ((TileMap[i][j] == '0')) s\_map.setTextureRect(IntRect(0, 0, 32, 32));//если встретили символ 0, то рисуем 3й квадратик

if ((TileMap[i][j] == 'q')) s\_map.setTextureRect(IntRect(0, 96, 32, 32));//ковер

if ((TileMap[i][j] == 'r')) s\_map.setTextureRect(IntRect(0, 128, 32, 32));//ковер

//дверь

if ((TileMap[i][j] == '!')) s\_map.setTextureRect(IntRect(32, 32, 32, 32)); //дверь

if ((TileMap[i][j] == '@')) s\_map.setTextureRect(IntRect(32, 32, 32, 32)); //дверь

if ((TileMap[i][j] == '$')) s\_map.setTextureRect(IntRect(32, 32, 32, 32)); //дверь

if ((TileMap[i][j] == 'k')) s\_map.setTextureRect(IntRect(128, 64, 32, 32)); //дверь

if ((TileMap[i][j] == '8')) s\_map.setTextureRect(IntRect(128, 0, 32, 32)); //дверь

if ((TileMap[i][j] == '9')) s\_map.setTextureRect(IntRect(128, 32, 32, 32)); //дверь

if ((TileMap[i][j] == '1')) s\_map.setTextureRect(IntRect(128, 96, 32, 32)); //крыша

if ((TileMap[i][j] == 'w')) s\_map.setTextureRect(IntRect(96, 32, 32, 32)); //стена ->

if ((TileMap[i][j] == 'e')) s\_map.setTextureRect(IntRect(0, 64, 32, 32)); //стена |

if ((TileMap[i][j] == 'y')) s\_map.setTextureRect(IntRect(32, 64, 32, 32)); //стена |

if ((TileMap[i][j] == 't')) s\_map.setTextureRect(IntRect(96, 96, 32, 32)); //стена |

if ((TileMap[i][j] == 'r')) s\_map.setTextureRect(IntRect(0, 128, 32, 32)); //стена L

//камин

if ((TileMap[i][j] == '^')) s\_map.setTextureRect(IntRect(128, 128, 32, 32));//вверх

if ((TileMap[i][j] == '>')) s\_map.setTextureRect(IntRect(128, 160, 32, 32));//вниз

if ((TileMap[i][j] == 'x')) s\_map.setTextureRect(IntRect(96, 160, 32, 32));//вправо

//ель

if ((TileMap[i][j] == 'n')) s\_map.setTextureRect(IntRect(32, 96, 32, 32));//вверх

if ((TileMap[i][j] == 'm')) s\_map.setTextureRect(IntRect(32, 128, 32, 32));//вниз

if ((TileMap[i][j] == '/')) s\_map.setTextureRect(IntRect(64, 128, 32, 32)); //стена ->

if ((TileMap[i][j] == 'Z')) s\_map.setTextureRect(IntRect(64, 192, 32, 32));

if ((TileMap[i][j] == 'Y')) s\_map.setTextureRect(IntRect(96, 192, 32, 32));

if ((TileMap[i][j] == '&')) s\_map.setTextureRect(IntRect(128, 192, 32, 32));

if ((TileMap[i][j] == '%')) s\_map.setTextureRect(IntRect(128, 224, 32, 32));

if ((TileMap[i][j] == '(')) s\_map.setTextureRect(IntRect(0, 229, 32, 32));

//труба

if ((TileMap[i][j] == 'j')) s\_map.setTextureRect(IntRect(64, 160, 32, 32));//вверх

if ((TileMap[i][j] == '3')) s\_map.setTextureRect(IntRect(64, 160, 32, 32));//вверх

if ((TileMap[i][j] == '4')) s\_map.setTextureRect(IntRect(64, 160, 32, 32));//вверх

if ((TileMap[i][j] == '5')) s\_map.setTextureRect(IntRect(64, 160, 32, 32));//вверх

//лестница

if ((TileMap[i][j] == 'a')) s\_map.setTextureRect(IntRect(32, 160, 32, 32));//вверх

if ((TileMap[i][j] == '6')) s\_map.setTextureRect(IntRect(64, 0, 32, 32));//вверх

if ((TileMap[i][j] == '7')) s\_map.setTextureRect(IntRect(32, 0, 32, 32));//вверх

//выход

if ((TileMap[i][j] == '.')) s\_map.setTextureRect(IntRect(0, 32, 32, 32));//вверх

if ((TileMap[i][j] == '|')) s\_map.setTextureRect(IntRect(32, 196, 32, 32));//вверх

if ((TileMap[i][j] == '-')) s\_map.setTextureRect(IntRect(0, 260, 32, 32));//вверх

if ((TileMap[i][j] == ',')) s\_map.setTextureRect(IntRect(0, 196, 32, 32));//вверх

//большая ель

if ((TileMap[i][j] == 'A')) s\_map.setTextureRect(IntRect(160, 0, 32, 32));

if ((TileMap[i][j] == 'B')) s\_map.setTextureRect(IntRect(192, 0, 32, 32));

if ((TileMap[i][j] == 'C')) s\_map.setTextureRect(IntRect(224, 0, 32, 32));

if ((TileMap[i][j] == 'D')) s\_map.setTextureRect(IntRect(256, 0, 32, 32));

if ((TileMap[i][j] == 'E')) s\_map.setTextureRect(IntRect(298, 0, 32, 32));

if ((TileMap[i][j] == 'F')) s\_map.setTextureRect(IntRect(160, 32, 32, 32));

if ((TileMap[i][j] == 'G')) s\_map.setTextureRect(IntRect(192, 32, 32, 32));

if ((TileMap[i][j] == 'H')) s\_map.setTextureRect(IntRect(224, 32, 32, 32));

if ((TileMap[i][j] == 'I')) s\_map.setTextureRect(IntRect(256, 32, 32, 32));

if ((TileMap[i][j] == 'J')) s\_map.setTextureRect(IntRect(298, 32, 32, 32));

if ((TileMap[i][j] == 'K')) s\_map.setTextureRect(IntRect(160, 64, 32, 32));

if ((TileMap[i][j] == 'M')) s\_map.setTextureRect(IntRect(192, 64, 32, 32));

if ((TileMap[i][j] == 'N')) s\_map.setTextureRect(IntRect(224, 64, 32, 32));

if ((TileMap[i][j] == 'O')) s\_map.setTextureRect(IntRect(256, 64, 32, 32));

if ((TileMap[i][j] == 'R')) s\_map.setTextureRect(IntRect(298, 64, 32, 32));

if ((TileMap[i][j] == 'T')) s\_map.setTextureRect(IntRect(160, 96, 32, 32));

if ((TileMap[i][j] == 'U')) s\_map.setTextureRect(IntRect(192, 96, 32, 32));

if ((TileMap[i][j] == 'V')) s\_map.setTextureRect(IntRect(224, 96, 32, 32));

if ((TileMap[i][j] == 'W')) s\_map.setTextureRect(IntRect(256, 96, 32, 32));

if ((TileMap[i][j] == 'X')) s\_map.setTextureRect(IntRect(298, 96, 32, 32));

s\_map.setPosition(j \* 32, i \* 32);

window.draw(s\_map);

}

}

// рисовка карты символами

String TileMap[HEIGHT\_MAP] = {

"0000000000000000000000000000000000000000000",

"0( 111%%22222n22e ABCDE y2n222%2222Y220",

"0 1112Z22222m22e FGHIJ y2m22222S222220",

"0 1112222222222e KMNOR 9$22222222222Z0",

"0 ajxq222222222e TUVWX y222L2222222220",

"0 1112222222L22e y22222222222220",

"0 11122S2222222e y2&22q2222S2220",

"0 111222222222Ye P 11111^111111110",

"01111112&22!22222e 111114111111110",

"0111111wwwwkwwwwwr 111116111111110",

"0 11 1110",

"0 P 11 P .|-0",

"0 11 1110",

"0111111711111111111111111 111111111711110",

"0111111311111111111111111 111111111511110",

"0111111>11111111111111111 111111111>11110",

"0y22222q22e2n222222222Z2e y22222222q222&0",

"0y22S22222e2m22222222222e y2S222222222220",

"0y222e2wwwr222222e22222#8 9$222222L222220",

"0y222e2S222222222e22L222e y2222222222S220",

"0y222e222222L2222e222222e y2222n222222220",

"0y2n2e22222222222e222n22e y2222m222222220",

"0y2m2e&22Y2222222eZ22m22e y22Y2S222222220",

"0000000000000000000000000000000000000000000",

};

**Player.cpp**

#pragma once

#include <SFML/Graphics.hpp>

#include <SFML/Audio.hpp>

#include "Player.h"

#include "globals.h"

#include <iostream>

#include "Map.h"

using namespace sf;

Player::Player(String File, float X, float Y, float W, float H)

{

File;

w = W; h = H;

imagee.loadFromFile("images/" + File);

texturee.loadFromImage(imagee);

sprite.setTexture(texturee);

x = X; y = Y;

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

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

dx = 0;

dy = 0;

speed = 0;

dir = 0;

key = 0,00001;

hp = 200;

win = 0;

hpbar.setFillColor(Color::Red);

hpbar.setPosition(128, 10.f);

hpbar.setSize(Vector2f((float)hp \* 1.f, 10.f));

keybar.setFillColor(Color::Yellow);

keybar.setPosition(900, 10.f);

keybar.setSize(Vector2f((float)key \* 10.f, 10.f));

CurrentFrame = 0;

Clock clock;

}

Player::~Player()

{

}

Player::Player()

{

}

void Player::update(float time)

{

switch (dir)

{

case 0: dx = speed; dy = 0; break;

case 1: dx = -speed; dy = 0; break;

case 2: dx = 0; dy = speed; break;

case 3: dx = 0; dy = -speed; break;

}

x += dx \* time;

y += dy \* time;

speed = 0;

sprite.setPosition(x, y);

InteractionWithMap();

hpbar.setSize(Vector2f((float)hp \* 1.f, 10.f));

keybar.setSize(Vector2f((float)key \* 70.f, 10.f));

}

void Player::draw\_p()

{

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

clock.restart();

time = time / 650;

float coordinatePlayerX, coordinatePlayerY = 0;

coordinatePlayerX = GetPlayerCoordinateX();

coordinatePlayerY = GetPlayerCoordinateY();

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

dir = 1; speed = 0.1;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(32 \* int(CurrentFrame), 32, 32, 32));

}

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

dir = 0; speed = 0.1;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(32 \* int(CurrentFrame), 65, 32, 32));

}

if ((Keyboard::isKeyPressed(Keyboard::Up) || (Keyboard::isKeyPressed(Keyboard::W)))) {

dir = 3; speed = 0.1;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(32 \* int(CurrentFrame), 96, 32, 32));

}

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

dir = 2; speed = 0.1;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(32 \* int(CurrentFrame), 0, 32, 32));

}

update(time);

window.draw(sprite);

}

void Player::InteractionWithMap()

{

for (int i = y / 32; i < ((y + h) / 32); i++)

for (int j = x / 32; j < ((x + w) / 32); j++)

{

////////////////////////////////////СТОЛКНОВЕНИЕ С КАРТОЙ//////////////////////////////////////////////////////////

if (TileMap[i][j] == '0' || TileMap[i][j] == '1' || TileMap[i][j] == 'e' || TileMap[i][j] == 'y' || TileMap[i][j] == 'w' || TileMap[i][j] == 'n' || TileMap[i][j] == 'P')

{

if (dy > 0)

{

y = i \* 32 - h;

}

if (dy < 0)

{

y = i \* 32 + 32;

}

if (dx > 0)

{

x = j \* 32 - w;

}

if (dx < 0)

{

x = j \* 32 + 32;

}

}

/////////////////////////////ПОДАРКИ//////////////////////////////////////

if (TileMap[i][j] == 'm') {

key += 1;

hp += 50;

}

if (TileMap[i][j] == 'm')

{

{

TileMap[i][j] = '/';

}

}

//////////////////////////////ХП/////////////////////////////

/////////////////////////////////////ТЕЛЕПОРТАЦИЯ/////////////////////////////////////////////

if (TileMap[i][j] == 'j') { //происходит телепортация (появление персонажа в новых координатах)

x = 225; y = 125; //падение в камин и пояавление в доме

}

if (TileMap[i][j] == '3') {

x = 225; y = 500;

}

if (TileMap[i][j] == '4') {

x = 1050; y = 175;

}

if (TileMap[i][j] == '5') {

x = 1175; y = 525;

}

////////////////////////////////////////////ВЫХОД/////////////////////////////////////////////////////////////

if (TileMap[i][j] == '.')

{

if (key >= 6) //если все подарки разложены ворота откроются

{

TileMap[i][j + 1] = ',';

}

}

if (TileMap[i][j] == '|') {

if (dx > 0)

{

x = j \* 32 - w;

}

}

if (TileMap[i][j] == '-')

{

win += 1;

}

////////////////////////////////////////////////////ДВЕРИ///////////////////////////////////////////////////

if (TileMap[i][j] == '!')

{

if (key >= 1) //если все подарки разложены ворота откроются

{

TileMap[i + 1][j] = '@';

}

}

if (TileMap[i][j] == 'k') {

if (dy > 0)

{

y = i \* 32 - h;

}

}

if (TileMap[i][j] == '#')

{

if (key >= 4) //если все подарки разложены ворота откроются

{

TileMap[i][j + 1] = '@';

}

}

if (TileMap[i][j] == '8') {

if (dx > 0)

{

x = j \* 32 - w;

}

}

if (TileMap[i][j] == '$')

{

if (key >= 5) //если все подарки разложены ворота откроются

{

TileMap[i][j - 1] = '@';

}

}

if (TileMap[i][j] == '9') {

if (dx > 0)

{

x = j \* 32 - w;

}

}

}

}

float Player::GetPlayerCoordinateX()

{

return x;

}

float Player::GetPlayerCoordinateY() {

return y;

}

void Player::Damage()

{

hp--;

return;

}

void Player::bDamage()

{

hp -= 20;

return;

}

void Player::Keys()

{

key++;

return;

}

**Shooting.cpp**

#include "SFML/Graphics.hpp"

#include <iostream>

#include "map.h"

#include "globals.h"

#include "Enemies.h"

#include "player.h"

#include "Shooting.h"

// конструктор стрелка

Shoot::Shoot(int x, int y) {

t.loadFromFile("images/evil.png");

sprite.setTexture(t);

sprite.setTextureRect(IntRect(0, 0, 38, 38));

rect = FloatRect(x \* 32, y \* 32, 0, 0);

dx = 0;

}

void Shooting::draw()

{

sprite.setPosition(rect.left, rect.top); //получение местоположения стрелка

window.draw(sprite); //отрисовка стрелка

}

//получение местоположения стрелка

FloatRect Shooting::position() {

return rect;

}

//получение сдвига ля пули

float Shooting::acceleration() {

return dx;

}

//конструктор пули

Shooting::Bullet::Bullet(FloatRect polozh, float uskor) {

bullet\_rect = polozh;

life = true;

im.loadFromFile("images/pula.png");

sprite.setTextureRect(IntRect(0, 0, 23, 23));

t.loadFromImage(im);

sprite.setTexture(t);

dx = -0.1;

}

//движение пули

void Shooting::Bullet::drawing() {

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

clock.restart();

time = time / 400;

sprite.setPosition(bullet\_rect.left, bullet\_rect.top); // получение местоположения пули

bullet\_rect.left = bullet\_rect.left + dx \* time; //сдвиг пули

WithMap(); //проверка на попадание в стену

window.draw(sprite); //отрисовка пули

}

//пуля попала в стену

void Shooting::Bullet::WithMap() {

// сравнение положения пули и стены

for (int i = (bullet\_rect.top) / 32; i < (bullet\_rect.top + 32) / 32; i++)

for (int j = (bullet\_rect.left) / 32; j < (bullet\_rect.left + 32) / 32; j++)

{

if (TileMap[i][j] == '0' || TileMap[i][j] == 'e' || TileMap[i][j] == '1')

{

life = false; //уничтожение пули

}

}

}

// деструктор пули

Shooting::Bullet::~Bullet() {

}

// пуля попадает в игрока

void Shooting::Bullet::WithPlayer(Player &p) {

// сравниваем положение игрока и пули

if ((((p.GetPlayerCoordinateX()) >= ((bullet\_rect.left))) && ((p.GetPlayerCoordinateX()) <= ((bullet\_rect.left) + 10))) && (((p.GetPlayerCoordinateY()) >= (bullet\_rect.top)) && ((p.GetPlayerCoordinateY()) <= (bullet\_rect.top) + 32))) {

life = false; //уничтожение пули

p.bDamage();//уменьшение здоровья

return;

}

}