

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 = 0.8;

w = h = 16;

life = true;

//выше инициализация в конструкторе

}

void update(float time)

{

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

}

};

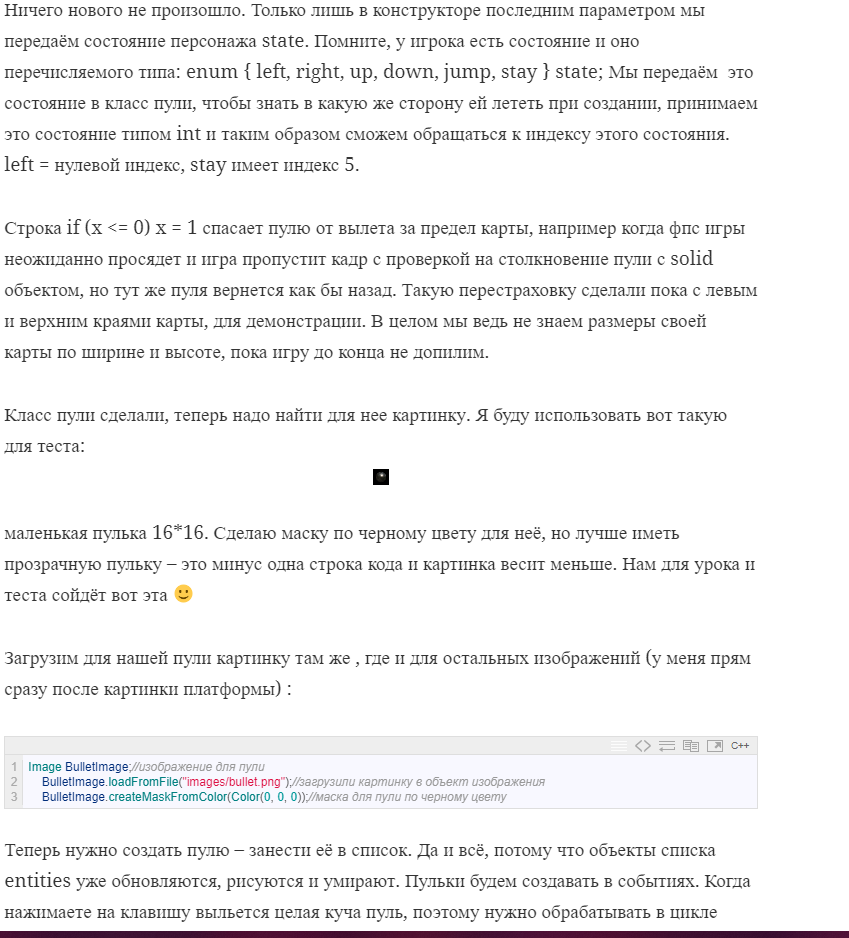
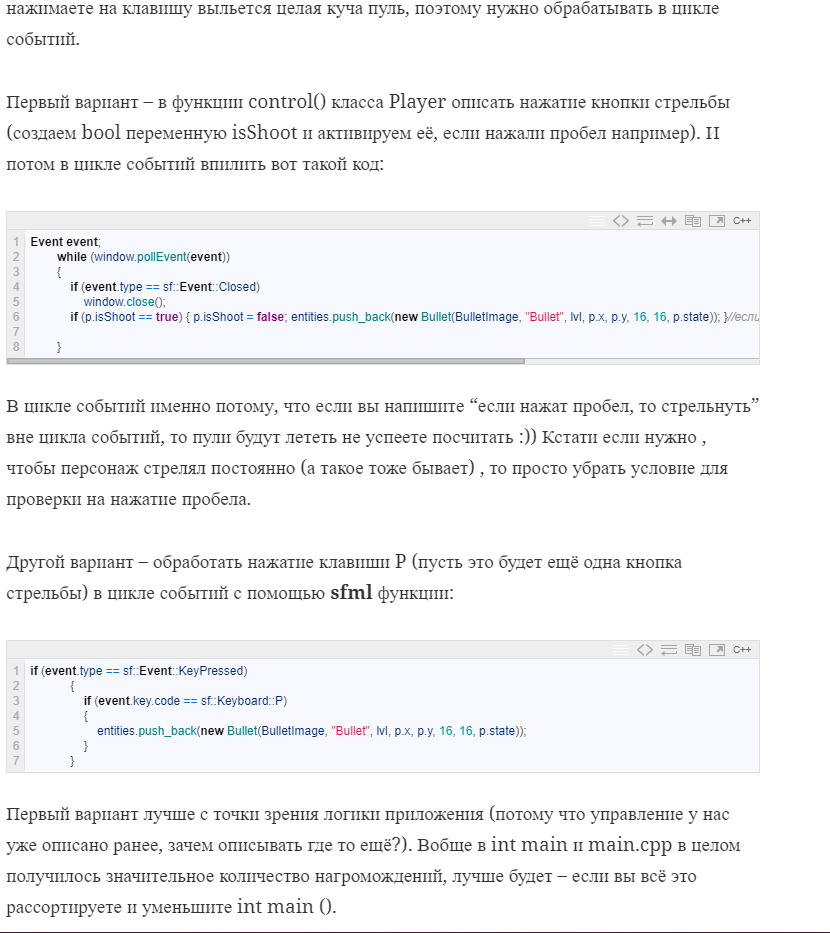


Image BulletImage;//изображение для пули

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

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



Event event;

while (window.pollEvent(event))

{

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

window.close();

if (p.isShoot == true) { p.isShoot = false; entities.push\_back(new Bullet(BulletImage, "Bullet", lvl, p.x, p.y, 16, 16, p.state)); }//если выстрелили, то появляется пуля. enum передаем как int

}

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

{

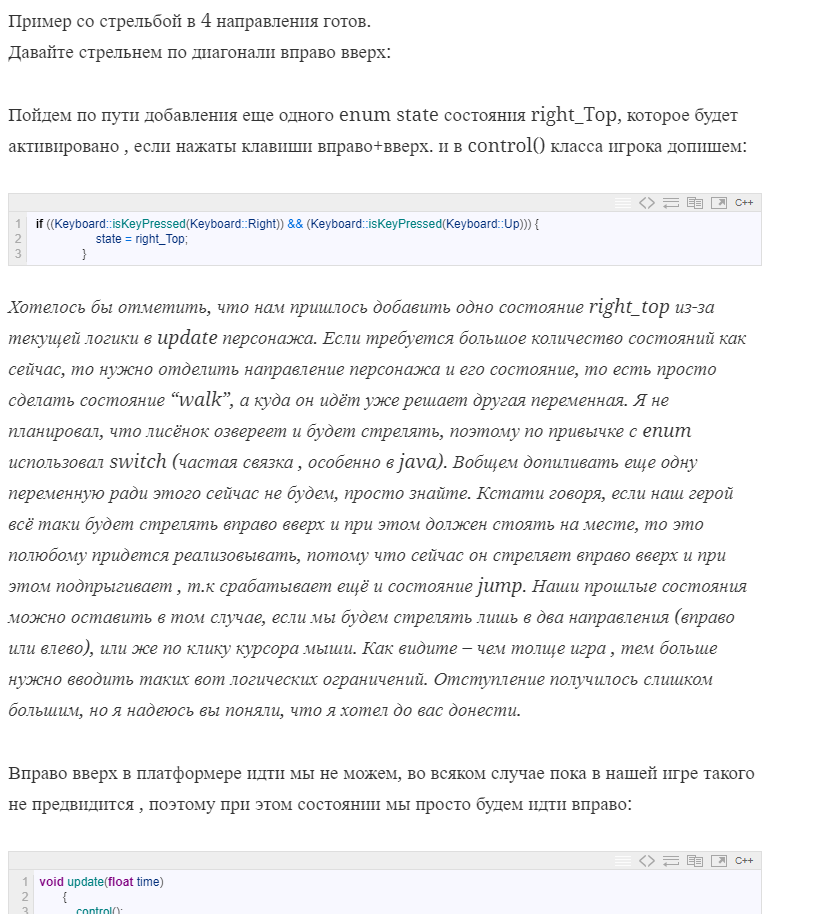
if (event.key.code == sf::Keyboard::P)

{

entities.push\_back(new Bullet(BulletImage, "Bullet", lvl, p.x, p.y, 16, 16, p.state));

}

}



void update(float time)

   {

   control();

   switch (state)

   {

   case right:dx = speed; break;

   case left:dx = -speed; break;

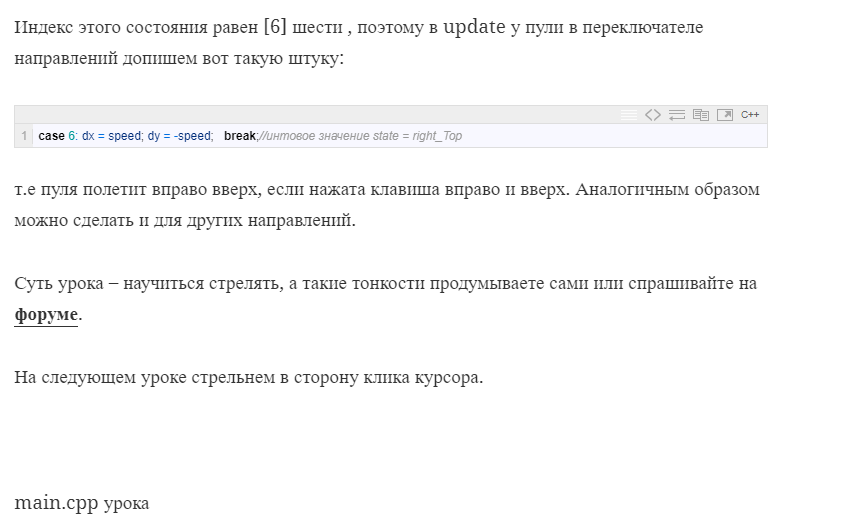
   case up: break;

   case down: dx = 0; break;

   case stay: break;

   case right\_Top:dx = speed; break;//состояние вправо вверх, просто продолжаем идти вправо

   }



case 6: dx = speed; dy = -speed;   break;//интовое значение state = right\_Top

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

#include <SFML/Graphics.hpp>

#include <SFML/Audio.hpp>

#include "view.h"

#include <iostream>

//#include <sstream>//сейчас не нужно. раньше выводили текст ( в 13 ом уроке )

#include "mission.h"

#include "iostream"

#include "level.h"

#include <vector>

#include <list>

using namespace sf;

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

class Entity {

public:

std::vector<Object> obj;

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

int w,h,health;

bool life, isMove, onGround;

Texture texture;

Sprite sprite;

String name;

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;

texture.loadFromImage(image);

sprite.setTexture(texture);

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

}

FloatRect getRect(){

return FloatRect(x, y, w, h);

}

virtual void update(float time) = 0;//все потомки переопределяют эту ф-цию

};

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

class Player :public Entity {

public:

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

int playerScore;

bool isShoot;

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

   playerScore = isShoot = 0; state = stay; obj = lev.GetAllObjects();

   if (name == "Player1"){

   sprite.setTextureRect(IntRect(4, 19, w, h));

   }

   }

   void control(){

   if (Keyboard::isKeyPressed){

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

   state = left; speed = 0.1;

   }

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

   state = right; speed = 0.1;

   }

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

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

   }

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

   state = down;

   }

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

   state = right\_Top;

   }

   /////выстрел

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

   isShoot = true;

   }

   }

   }

   void checkCollisionWithMap(float Dx, float Dy)

   {

   for (int i = 0; i<obj.size(); i++)

   if (getRect().intersects(obj[i].rect))

   {

   if (obj[i].name == "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; }

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

   }

  // else { onGround = false; }

   }

   }

   void update(float time)

   {

   control();

   switch (state)

   {

   case right:dx = speed; break;

   case left:dx = -speed; break;

   case up: break;

   case down: dx = 0; break;

   case stay: break;

   case right\_Top:dx = speed; break;//состояние вправо вверх, просто продолжаем идти вправо

   }

   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; }

   if (!isMove){ speed = 0; }

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

   dy = dy + 0.0015\*time;

   }

};

class Enemy :public Entity{

public:

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"){

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

dx = 0.1;

}

}

void checkCollisionWithMap(float Dx, float Dy)

{

for (int i = 0; i<obj.size(); i++)

if (getRect().intersects(obj[i].rect))

{

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;  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"){

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

checkCollisionWithMap(dx, 0);

x += dx\*time;

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

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

}

}

};

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 = 0.8;

w = h = 16;

life = true;

//выше инициализация в конструкторе

}

void update(float time)

{

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

case 6: dx = speed; dy = -speed;   break;//интовое значение state = right\_Top

}

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

}

};

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

}

};

int main()

{

RenderWindow window(VideoMode(640, 480), "Lesson 28. kychka-pc.ru");

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

Level lvl;

lvl.LoadFromFile("map.tmx");

Image heroImage;

heroImage.loadFromFile("images/MilesTailsPrower.gif");

Image easyEnemyImage;

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

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

Image movePlatformImage;

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

Image BulletImage;//изображение для пули

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

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

std::list<Entity\*>  entities;

std::list<Entity\*>::iterator it;

std::list<Entity\*>::iterator it2;//второй итератор.для взаимодействия между объектами списка

std::vector<Object> e = lvl.GetObjects("EasyEnemy");

for (int i = 0; i < e.size(); i++) entities.push\_back(new Enemy(easyEnemyImage, "EasyEnemy", lvl, e[i].rect.left, e[i].rect.top, 200, 97));

Object player=lvl.GetObject("player");

Player p(heroImage, "Player1", lvl, player.rect.left, player.rect.top, 40, 30);

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, 95, 22));//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();

if (p.isShoot == true) { p.isShoot = false; entities.push\_back(new Bullet(BulletImage, "Bullet", lvl, p.x, p.y, 16, 16, p.state)); }//если выстрелили, то появляется пуля. enum передаем как int

//второй варант стрельнуть

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

{

if (event.key.code == sf::Keyboard::P)

{

entities.push\_back(new Bullet(BulletImage, "Bullet", lvl, p.x, p.y, 16, 16, p.state));

}

}

}

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++)//проходимся по эл-там списка

{

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 + 3; p.x += movPlat->dx\*time; p.dy = 0; p.onGround = true; // то выталкиваем игрока так, чтобы он как бы стоял на платформе

}

}

if (((\*it)->name == "EasyEnemy") && ((\*it)->getRect().intersects(p.getRect())))

{

////////выталкивание врага

if ((\*it)->dx>0)//если враг идет вправо

{

std::cout << "(\*it)->x" << (\*it)->x << "\n";//коорд игрока

std::cout << "p.x" << p.x << "\n\n";//коорд врага

(\*it)->x = p.x - (\*it)->w; //отталкиваем его от игрока влево (впритык)

(\*it)->dx = 0;//останавливаем

std::cout << "new (\*it)->x" << (\*it)->x << "\n";//новая коорд врага

std::cout << "new p.x" << p.x << "\n\n";//новая коорд игрока (останется прежней)

}

if ((\*it)->dx < 0)//если враг идет влево

{

(\*it)->x = p.x + p.w; //аналогично - отталкиваем вправо

(\*it)->dx = 0;//останавливаем

}

///////выталкивание игрока

if (p.dx < 0) { p.x = (\*it)->x + (\*it)->w;}//если столкнулись с врагом и игрок идет влево то выталкиваем игрока

if (p.dx > 0) { p.x = (\*it)->x - p.w;}//если столкнулись с врагом и игрок идет вправо то выталкиваем игрока

}

for (it2 = entities.begin(); it2 != entities.end(); it2++)

{

if ((\*it)->getRect() != (\*it2)->getRect())//при этом это должны быть разные прямоугольники

if (((\*it)->getRect().intersects((\*it2)->getRect())) && ((\*it)->name == "EasyEnemy") && ((\*it2)->name == "EasyEnemy"))//если столкнулись два объекта и они враги

{

(\*it)->dx \*= -1;//меняем направление движения врага

(\*it)->sprite.scale(-1, 1);//отражаем спрайт по горизонтали

}

}

}

p.update(time);//перенесли сюда update игрока

window.setView(view);

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

lvl.Draw(window);

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

window.draw((\*it)->sprite);

}

window.draw(p.sprite);

window.display();

}

return 0;

}

//////////////////////////////////Движение по клику мыши/////////////////////////

