**Entity.cpp**

#include "stdafx.h"

#include "Entity.h"

using namespace sf;

Entity::Entity(Image &image, float X, float Y, int W, int H)

{

x = X; y = Y;

w = W; h = H;

dx = 0.1; dy = 0.1; //изменение координат

speed = 0.1; //скорость

CurrentFrame = 0;

life = false; //герой по умолчанию жив

game = true;

texture.loadFromImage(image); //изображение сущности

sprite.setTexture(texture); //заливка спрайта дял сущности

}

bool Entity::getLife(){

return life;

}

void Entity::setLife(bool l){

life = l;

}

bool Entity::getGame(){

return game;

}

void Entity::setGame(bool l){

game = l;

}

**Player.cpp**

#include "stdafx.h"

#include "Player.h"

using namespace sf;

Player::Player(Image &image, float X, float Y, int W, int H) :Entity(image, X, Y, W, H)

{

playerScore = 0;

dir = 1;

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

}

int Player::getScore()

{

return(playerScore);

}

void Player::setScore(int score)

{

playerScore = score;

}

void Player::control()

{ //При нажатии на одну из клавиш, меняется направление движения по координате

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

{

dx = -0.1;

}

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

{

dx = 0.1;

}

if (Keyboard::isKeyPressed(Keyboard::Up))

{

dy = -0.1;

}

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

{

dy = 0.1;

}

}

void Player::checkCollisionWithMap(float Dx, float Dy)

{

for (int i = y / 32; i < (y + h) / 32; i++)//проходимся по элементам карты

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

{

if (mp.TileMap[i][j] == '0')//если есть стена

{

if ((Dy > 0) && (dir == 2)) { y = i \* 32 - h; dy = 0; }//столкновение снизу

if ((Dy < 0) && (dir == 2)){ y = i \* 32 + 32; dy = 0; }//столкновение сверху

if ((Dx > 0) && (dir == 1)) { x = j \* 32 - w; dx = 0; }//столкновение справа

if ((Dx < 0) && (dir == 1)) { x = j \* 32 + 32; dx = 0; }//столкновение слева

}

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

{

setScore(++playerScore); //Добавлени очки игроку

mp.TileMap[i][j] = ' ';

}

}

}

void Player::update(float time)//обновление объекта класса.

{

if (life)

{//проверяем, жив ли герой

control();

x += dx\*time; //движение по “X”

dir = 1;

checkCollisionWithMap(dx, 0);//обрабатываем столкновение по Х

y += dy\*time; //движение по “Y”

dir = 2;

checkCollisionWithMap(0, dy);//обрабатываем столкновение по Y

if (dx > 0) {//состояние идти вправо

dx = speed;

dy = 0;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

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

}

if (dx < 0)

{//состояние идти влево

dx = -speed;

dy = 0;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

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

}

if (dy < 0)

{//идти вверх

dy = -speed;

dx = 0;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

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

}

if (dy > 0)

{//идти вниз

dy = speed;

dx = 0;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

sprite.setTextureRect(IntRect(32 \* int(CurrentFrame), 64, 32, 32));

}

}

sprite.setPosition(x, y); //спрайт в позиции (x, y).

}

**Enemy.cpp**

#pragma once

#include "stdafx.h"

#include "Enemy.h"

using namespace sf;

Enemy::Enemy(Image &image, float X, float Y, int W, int H) :Entity(image, X, Y, W, H)

{

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

direction = rand() % (3); //случайное направление

speed = 0.1; //скорость

dx = speed;

}

void Enemy::checkCollisionWithMap(float Dx, float Dy)//проверка на препятствие

{

for (int i = y / 32; i < (y + h) / 32; i++)//цикл проверки следующих элементов

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

{

int k = y;

int l = x;

if ((k % 32 == 0) && (l % 32 == 0) && ((float)((int)x) == x) && ((float)((int)y) == y)){

if ((mp.TileMap[i + 1][j] == ' ') || (mp.TileMap[i + 1][j] == 's')) direction = rand() % (4);

if ((mp.TileMap[i - 1][j] == ' ') || (mp.TileMap[i - 1][j] == 's')) direction = rand() % (4);

}

if (mp.TileMap[i][j] == '0')//если впереди препятствие

{//в соответствие с выбранным направлением перемещает на предыдущий элемент

if (Dy > 0) {

y = i \* 32 - h; dy = -0.1;

direction = rand() % (3); //случайное направление

}//по Y

if (Dy < 0) {

y = i \* 32 + 32; dy = 0.1;

direction = rand() % (3);

}

if (Dx > 0) {

x = j \* 32 - w; dx = -0.1;

direction = rand() % (3);

}

if (Dx < 0) {

x = j \* 32 + 32; dx = 0.1;

direction = rand() % (3);

}

}

}

}

void Enemy::update(float time)

{

if (Keyboard::isKeyPressed(Keyboard::Enter)) life = true;

switch (direction) //выбор направления

{

case 0:

{//вправо

dx = speed; //скорость по х

dy = 0; //скорость по у

CurrentFrame += 0.005\*time; //просчтет кадров

if (CurrentFrame > 3) CurrentFrame -= 3;//если достигнет 3, то рисовать спрайт заново

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

break;

}

case 1:

{//влево

dx = -speed;

dy = 0;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

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

break;

}

case 2:

{//вверх

dy = -speed;

dx = 0;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

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

break;

}

case 3:

{//вниз

dy = speed;

dx = 0;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 3) CurrentFrame -= 3;

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

break;

}

}

if (dx > 0){

xx = x;

x += dx\*time; //движение по “X”

if (xx < floor(x)) x = floor(x);

}

if (dx < 0){

xx = x;

x += dx\*time; //движение по “X”

if (x<floor(xx - 0.01)) x = round(x);

}

if (dy < 0){

yy = y;

y += dy\*time; //движение по “Y”

if (yy < floor(y)) y = floor(y);

}

if (dy > 0){

yy = y;

y += dy\*time; //движение по “Y”

if (y<floor(yy - 0.01)) y = round(y);

}

checkCollisionWithMap(dx, 0);//обрабатываем столкновение по Х

checkCollisionWithMap(0, dy);//обрабатываем столкновение по Y

sprite.setPosition(x, y);

}

**Interface.cpp**

#include "stdafx.h"

#include "Interface.h"

#include "Player.h"

#include "Entity.h"

#include "Enemy.h"

using namespace sf;

Interface::Interface(){

font.loadFromFile("fonts/Strenuous.ttf");//установка шрифта

map\_image.loadFromFile("images/map.png");//загружаем файл с текстурой карты

map.loadFromImage(map\_image);//заряжаем текстуру карты из картинки

s\_map.setTexture(map);//устанавливаем текстуру карты

gameTime = 0;//начало игрового времени

heroImage.loadFromFile("images/hero.png"); // загружаем изображение ПакМэна

enemy1.loadFromFile("images/enemy1.png"); // загружаем изображение призрака

enemy2.loadFromFile("images/enemy2.png");

enemy3.loadFromFile("images/enemy3.png");

p = new Player(heroImage, 288, 512, 30, 30);

}

Interface::~Interface()

{

delete p;

while (!enemy.empty())

{

it = enemy.begin();

delete \*it;

enemy.erase(it);

}

};

void Interface::interact(){

sf::VideoMode desktop = sf::VideoMode::getDesktopMode();

sf::RenderWindow window(sf::VideoMode(608, 704, desktop.bitsPerPixel), "PacMan");

Text text("", font, 20), menu("", font, 30);//создаем объект текст

text.setColor(Color::Yellow);//покрасили текст в красный

text.setStyle(Text::Bold);//жирный текст.

menu.setColor(Color::Yellow);//покрасили текст в красный

menu.setStyle(Text::Bold);//жирный текст.

//Player p(heroImage, 288, 512, 30, 30);//объект класса игрока

srand(time(0));

enemy.push\_back(new Enemy(enemy1, 288, 288, 32, 32)); //создаем врагов и помещаем в список

enemy.push\_back(new Enemy(enemy2, 256, 320, 32, 32));

enemy.push\_back(new Enemy(enemy3, 288, 320, 32, 32));

while (window.isOpen()) { //пока открыто

if (p->getGame() == true)

if (Keyboard::isKeyPressed(Keyboard::Enter)){ p->setLife(true); p->setGame(false); }

float time = clock.getElapsedTime().asMicroseconds(); //таймер логики

if (p->getLife()) gameTime = gameTimeClock.getElapsedTime().asSeconds();//игровое время

clock.restart(); //перезапуск таймера

time = time / 800;

while (window.pollEvent(event)) //обработчик событий на закрытие

{

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

{

window.close();

}

if (Keyboard::isKeyPressed(Keyboard::Q)) {

window.close();

}

}

p->update(time); //обновление игрока

if (p->getLife())

for (it = enemy.begin(); it != enemy.end(); it++)

{

(\*it)->update(time); //запускаем метод update()

}

if (p->getLife() == true)

{//если игрок жив

for (it = enemy.begin(); it != enemy.end(); it++)

{//бежим по списку врагов

if ((p->getRect().intersects((\*it)->getRect())))

{

p->setLife(false);

std::cout << "Game over";

}

}

}

window.clear();

for (int i = 0; i < 22; i++)//отрисовка карты по шаблону из map.cpp

for (int j = 0; j < 19; j++)

{

if (inter.TileMap[i][j] == ' ') s\_map.setTextureRect(IntRect(0, 0, 32, 32)); //если пусто, то рисовать блок земли

if (inter.TileMap[i][j] == 's') s\_map.setTextureRect(IntRect(32, 0, 32, 32)); //если точка, то рисовать кружок

if ((inter.TileMap[i][j] == '0')) s\_map.setTextureRect(IntRect(64, 0, 32, 32)); //если препятствие, то рисовать блок стены

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

window.draw(s\_map); //рисовать

}

std::ostringstream playerScoreString, gameTimeString;

playerScoreString << p->getScore(); gameTimeString << gameTime;//Получаем счёт и время в игре

text.setString("Score: " + playerScoreString.str() + " Time: " + gameTimeString.str());//задаем строку тексту

text.setPosition(5, 2);//задаем позицию текста

window.draw(text);//рисуем этот текст

if (p->getScore() == 176) p->setLife(false);

if (p->getGame()){

menu.setString("Press ENTER for start");

menu.setPosition(100, 315);//задаем позицию текста

window.draw(menu);//рисуем этот текст

}

if ((!p->getLife()) && (!p->getGame())){

menu.setString("Press Q for exit");

menu.setPosition(150, 315);//задаем позицию текста

window.draw(menu);//рисуем этот текст

}

window.draw(p->sprite);//рисуем спрайт ПакМэна

for (it = enemy.begin(); it != enemy.end(); it++)

{

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

}

window.display();

}

}

**Map.cpp**

#include "map.h"

std::string Map::TileMap[] = { //массив строк карты

"0000000000000000000",

"0ssssssss0ssssssss0",

"0s00s000s0s000s00s0",

"0s00s000s0s000s00s0",

"0sssssssssssssssss0",

"0s00s0s00000s0s00s0",

"0ssss0sss0sss0ssss0",

"0000s000 0 000s0000",

" 0s0 0s0 ",

"0000s0 00 00 0s0000",

"0ssssss ssssss0",

"0000s0 00000 0s0000",

" 0s0sssssss0s0 ",

"0000s0s00000s0s0000",

"0ssssssss0ssssssss0",

"0s00s000s0s000s00s0",

"0ss0sssssssssss0ss0",

"00s0s0s00000s0s0s00",

"0ssss0sss0sss0ssss0",

"0s000000s0s000000s0",

"0sssssssssssssssss0",

"0000000000000000000",

};