**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«ПЕТРОЗАВОДСКИЙ государственный университет»**

**Отчет по практической работе**

**«Создание игры», в рамках курса «Технология Программирования»**

Выполнили: студенты 3 курса

Физико-Технического

факультета, гр.21312:

Куземичев Виктор,

Бухманова Дарья,

Бараева Екатерина

Преподаватель:

Бульба Артем Владимирович

Петрозаводск 2017

Цели работы:

* создание игры на определенную тематику с использование методического пособия.
* знакомство с библиотекой SFML для С++
* моделирование реально ежедневной работы программистов, т.е. удаленная работа в кооперативе с использование сервиса GitHub (<https://github.com>), в качестве файлообменника между участниками кооператива.
* закрепление полученных знаний работы в среде VS C++ (VS 2015) и в сервисе GitHub.

Сюжет игры:

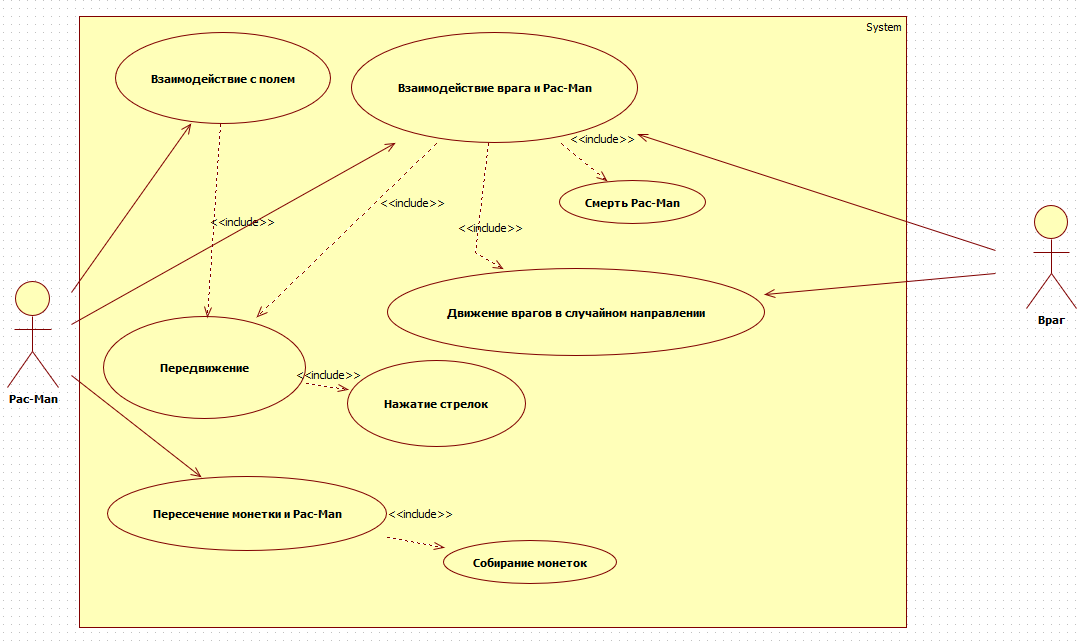
* Имеется игровое поле, на котором реализован лабиринт. Пользователь ПК выступает в роли PacMan’а (игрока), который может свободно перемещаться в рамках лабиринта.
* На игровом поле реализуются появление персонажей-врагов (враг), данные персонажи могут свободно перемещаться по лабиринту (направление выбирается случайно).
* При взаимодействии врага и игрока происходит «смерть» последнего.
* Цель игрока - не дать врагам «поймать» себя и собрать 1000 монет для прохождения игры (монеты в начале игры занимают все свободные клетки поля, после они случайно появляются на карте с определенным интервалом времени).

Программная реализация:

* В качестве среды разработки выбрана среда Visual Studio 2015, как наиболее знакомая среда (версия 2017 года не подошла из-за отсутствия работоспособной версии библиотеки SFML для С++).
* В качестве языка реализации проекта был выбран язык С++.
* Элементы проекта:
  + Enemy.h, Enemy.cpp – реализация врага
  + Player.h, PlayerClass.cpp – реализация игрока
  + Map.h, Map.cpp – реализация игровой карты
  + Entity.h, Entity.cpp – реализация класс предка для игрока и врага
  + Main.cpp – основная часть проекта (главный файл)
  + Global.h – для реализации движения игрока и врага на карте
* Полное описание главного файла проекта:
  + В главном файле связываются все ранее перечисленный элементы проекта
  + Осуществляется настройка игрового поля (размер и глубина цвета)
  + Выставляется шрифт для всех надписей
  + Загружаются текстуры игроков и врагов, так же и стен поля/лабиринта
  + Реализуется игровое время, для функционирования элементов игры.
  + Создается «вечный цикл» - «пока открыто окно», в котором происходит реализация игроков и врагов на карте, респаунт монеток на поле, проверка на завершение игры (игрок «умер» или собрал 100 монеток), проверка пересечения игрока и врага на игровом поле, респаунт на игровом поле препятствий/свободных клеток, вывод на экран: время игры, кол-во собранных монет, кол-во очков жизни игрока, освобождение занимаемой динамической памяти по завершению игры.

Заключение:

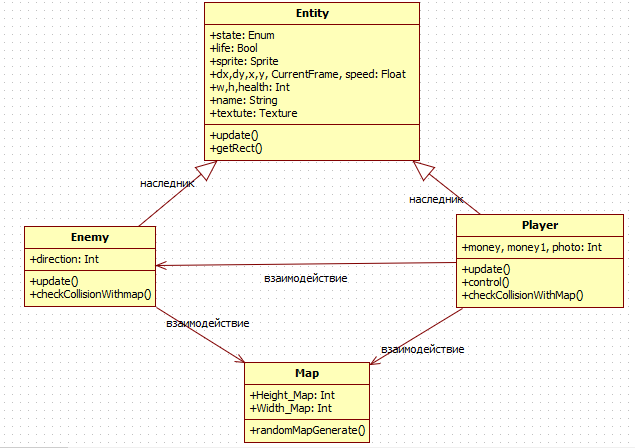
* Все прецеденты, согласно диаграмме прецедентов, реализованы.



* Игра тестировалась на топовом пк для 2017 года, поэтому никаких подтормаживаний и зависаний со стороны «железа» не наблюдалось.
* В данном проекте реализована раздельная компиляция входящих в проект элементов (см. выше).
* В ходе работы проект потребует выделение динамической памяти, которая при завершении игры освобождается от ранее записанной игровой информации.
* Все переменный в данном проекте используются в той или иной мере.
* Есть незначительная избыточность алгоритмов вывода надписей во время игры (такая реализация выбрана с целью неподвижности надписей).
* Цели были достигнуты.
* Приложены диаграммы прецедентов и классов для данного проекта.

Приложение:

* Диаграмма прецедентов находится выше.
* Диаграмма классов:



* Код элементов программы:

**Global.h**

#ifndef global\_h

#define global\_h

extern std::string TileMap[HEIGHT\_MAP];//внешняя переменная для отображения на карте элементов

//она необходима, т.к. используется в классе игрока и главной функции, т.е. проиходит конфлик двух переменных и их переопределение.

#endif

* **EnemyClass.cpp**

#include <iostream>

#include <sstream>

#include <SFML/Graphics.hpp>

#include "map.h"

#include "Global.h"

#include "Entity.h"

#include "Enemy.h"

using namespace std;

using namespace sf;

Enemy::Enemy(Image &image, float X, float Y, int W, int H, std::string Name) :Entity(image, X, Y, W, H, Name) {

if (name == "Enemy1") {

//Задаем спрайту один прямоугольник для

//вывода одного игрока. IntRect – для приведения типов

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

direction = rand() % (3); //Направление движения врага задаём случайным образом

//через генератор случайных чисел

speed = 0.1;//даем скорость.этот объект всегда двигается

dx = speed;

}

}

void Enemy::checkCollisionWithMap(float Dx, float Dy)//ф-ция проверки столкновений с картой

{

for (int i = y / 37; i < (y + h) / 37; i++)//проходимся по элементам карты

for (int j = x / 37; j<(x + w) / 37; j++)

{

if (TileMap[i][j] == '0')//если элемент - тайлик земли

{

if (Dy > 0) {

y = i \* 37 - h; dy = -0.1;

direction = rand() % (4); //Направление движения врага

}//по Y

if (Dy < 0) {

y = i \* 37 + 37; dy = 0.1;

direction = rand() % (4);//Направление движения врага

}//столкновение с верхними краями

if (Dx > 0) {

x = j \* 37 - w; dx = -0.1;

direction = rand() % (4);//Направление движения врага

}//с правым краем карты

if (Dx < 0) {

x = j \* 37 + 37; dx = 0.1;

direction = rand() % (4); //Направление движения врага

}// с левым краем карты

}

}

}

void Enemy::update(float time)

{

if (name == "Enemy1") {//для персонажа с таким именем логика будет такой

if (life) {//проверяем, жив ли герой

switch (direction)//делаются различные действия в зависимости от состояния

{

case 0: {//состояние идти вправо

dx = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 2) CurrentFrame -= 2;

sprite.setTextureRect(IntRect(0, 0, 35, 35));

break;

}

case 1: {//состояние идти влево

dx = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 2) CurrentFrame -= 2;

sprite.setTextureRect(IntRect(33, 0, 35, 35));

break;

}

case 2: {//идти вверх

dy = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 2) CurrentFrame -= 2;

sprite.setTextureRect(IntRect(65, 0, 35, 35));

break;

}

case 3: {//идти вниз

dy = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > 2) CurrentFrame -= 2;

sprite.setTextureRect(IntRect(100, 0, 35, 35));

break;

}

}

x += dx\*time; //движение по “X”

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

y += dy\*time; //движение по “Y”

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

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

if (health <= 0) { life = false; }//если жизней меньше 0, либо равно 0, то умираем

}

}

}

* **EntityClass.cpp**

#include <SFML/Graphics.hpp>

#include "entity.h"

#include "map.h"

#include <iostream>

#include "global.h"

#include <sstream>

using namespace std;

using namespace sf;

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

{

x = X; y = Y; //координата появления спрайта

w = W; h = H; //размер спрайта

name = Name; // имя персонажа

dx = 0; dy = 0;

speed = 0;

CurrentFrame = 0;

health = 100;

life = true; //инициализировали логическую переменную жизни, герой жив

image.createMaskFromColor(Color(0, 0, 0)); //убираем ненужный black color, в предке, т.к. у всех техтур надо убрать черный цвет.

texture.loadFromImage(image); //заносим наше изображение в текстуру

sprite.setTexture(texture); //заливаем спрайт текстурой

}

* **Map.cpp**

#include <SFML\Graphics.hpp>

#include "map.h"

std::string TileMap[HEIGHT\_MAP] = {

"00000000000000000000",

"0hhhhh00hhhhh0hhhhh0",

"0hhhhhhhhh0hhhh0hhh0",

"00hh0000000000000hh0",

"0hhhhhhhhhh0hhhhhhh0",

"0hhhhhhhhhh0h0h00hh0",

"0hhh0hhhhhh0h0hhhhh0",

"0hh00hhh0hhhh0000hh0",

"0hh0hhh00hhhhhhh0hh0",

"0hh0hhhh0hhhhhhh0hh0",

"0hh0hhhh00hhhhhhhhh0",

"0hh0hh0000000hhhhhh0",

"0hhhhhhhhhhhhhhhhh00",

"0hhhhhhhhhhhhhh0hhh0",

"00000000000000000000",

};

void randomMapGenerate() {//рандомно расставляем камни

int randomElementX = 0;//переменная для хранения случайного элемента по горизонтали

int randomElementY = 0;//переменная для хранения случайного элемента по вертикали

srand(time(0));//Инициализация генератора случайных чисел

int countStone = 10;//количество камней

if (countStone > 0) {

randomElementX = 1 + rand() % (WIDTH\_MAP - 1);//псевдослучайное значение по “x” от 1 до

//ширина карты-1. Ограничение введено чтобы не получать числа бордюра карты

randomElementY = 1 + rand() % (HEIGHT\_MAP - 1);//по “y”

if (TileMap[randomElementY][randomElementX] == ' ') {//если встретили символ пробел,

TileMap[randomElementY][randomElementX] = 'h'; //то ставим туда money.

countStone--;

if (TileMap[randomElementY][randomElementX] != ' ');

}

}

};

* **Map.h**

#ifndef map\_h

#define map\_h

#include <SFML\Graphics.hpp>

const int HEIGHT\_MAP = 15;//размер карты по высоте

const int WIDTH\_MAP = 20;//размер карты по ширине

void randomMapGenerate();

#endif

* **Main.cpp**

#include <iostream>

#include <sstream>

#include <SFML/Graphics.hpp> // подключаем СФМЛ библиотеку

#include "map.h" //подключили код с картой

#include <list>

#include "Global.h"

#include "Entity.h"

#include "Player.h"

#include "Enemy.h"

using namespace sf;

int main()

{

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

sf::RenderWindow window(sf::VideoMode(742, 555, desktop.bitsPerPixel), "PACMAN"); // создаем переменную window, в скобках конструктор с параметрами (размер окна и глубина пикселей)

Font font;//шрифт

font.loadFromFile("mysor/CyrilicOld.ttf");//передаем нашему шрифту файл шрифта

Text text("", font, 20);//создаем объект текст. закидываем в объект текст строку, шрифт, размер шрифта(в пикселях);//сам объект текст (не строка)

text.setFillColor(Color::White);//покрасили текст в красный. если убрать эту строку, то по умолчанию он белый

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

Image map\_image;//объект изображения для карты

map\_image.loadFromFile("images/mappac.png");//загружаем файл для карты

Texture map;//текстура карты

map.loadFromImage(map\_image);//заряжаем текстуру картинкой

Sprite s\_map;//создаём спрайт для карты

s\_map.setTexture(map);//заливаем текстуру спрайтом

Clock clock;

Clock gameTimeClock;//хранит игровое время

int gameTime = 0; //игровое время

Image EnemyImage[3];//массив перменных для врагов

EnemyImage[2].loadFromFile("images/enemypac.png"); // загружаем изображение врага

EnemyImage[1].loadFromFile("images/enemypac1.png"); // загружаем изображение врага

EnemyImage[0].loadFromFile("images/enemypac2.png"); // загружаем изображение врага

Image image\_pac;// переменная для изображения игрока

image\_pac.loadFromFile("images/pacman.png");// загрузка изображения игрока

Player p(image\_pac, 250, 250, 35, 35, "Pacman"); // переменная игрока

std::list<Entity\*> enemies; //список врагов, указателей на объекты врагов

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

const int ENEMY\_COUNT = 3; //максимальное количество врагов в игре

int enemiesCount = 0; //текущее количество врагов в игре

//Заполняем список объектами врагами

for (int i = 0; i < ENEMY\_COUNT; i++)

{

float xr = 150 + rand() % 500; //случайная координата врага на поле игры по оси “x”

float yr = 150 + rand() % 350; //случайная координата врага на поле игры по оси “y”

//создаем врагов и помещаем в список

enemies.push\_back(new Enemy(EnemyImage[i], xr, yr, 35, 35, "Enemy1"));

enemiesCount += 1; //увеличили счётчик врагов

}

int createObjectForMapTimer = 0;//Переменная под время для генерирования камней

while (window.isOpen())

{

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

if (p.life) gameTime = gameTimeClock.getElapsedTime().asSeconds();

//игровое время в секундах идёт вперед, пока жив игрок.

//Перезагружать как time его не надо. оно не обновляет логику игры

clock.restart();

time = time / 800;

createObjectForMapTimer += time;//наращиваем таймер

if (createObjectForMapTimer > 60) { //60000 = 1 min

randomMapGenerate();//генерация money

createObjectForMapTimer = 1;//обнуляем timer of money object on map

}

sf::Event event;

while (window.pollEvent(event))

{

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

window.close();

}

p.update(time); //оживляем объект “p” класса “Player” с помощью времени sfml,

// передавая время в качестве параметра функции update.

//оживляем врагов

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

{

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

}

//Проверка пересечения игрока с врагами

//Если пересечение произошло, то "health = 0", игрок обездвижевается и

//выводится сообщение "you are lose"

if (p.life == true) {//если игрок жив

for (it = enemies.begin(); it != enemies.end(); it++) {//бежим по списку врагов

if ((p.getRect().intersects((\*it)->getRect())) && ((\*it)->name == "Enemy1"))

{

p.health = 0;

std::cout << "you are lose";

}

}

}

window.clear();// чистим экран

/////////////////////////////Рисуем карту/////////////////////

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

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

{

if (TileMap[i][j] == ' ') s\_map.setTextureRect(IntRect(0, 0, 37, 37)); //если встретили символ пробел, то рисуем 1-й квадратик - свободную клетку поля

if (TileMap[i][j] == 'h') s\_map.setTextureRect(IntRect(37, 0, 37, 37));//если встретили символ "h", то рисуем 2й квадратик - money

if ((TileMap[i][j] == '0')) s\_map.setTextureRect(IntRect(74, 0, 37, 37));//если "0" ... - stena

s\_map.setPosition(j \* 37, i \* 37);//раскладываем квадратики в карту.

window.draw(s\_map);//рисуем квадратики на экран

}

std::ostringstream playerMoneyString, gameTimeString, playerHealthString;//объявили переменную здоровья и времени

playerMoneyString << p.money; gameTimeString << gameTime, playerHealthString << p.health;;//формируем строки

text.setString("Время игры: " + gameTimeString.str());

text.setPosition(500, 10);//задаем позицию текста

window.draw(text);//рисуем этот текст

text.setString("Money: " + playerMoneyString.str());

text.setPosition(50, 10);//задаем позицию текста

window.draw(text);//рисуем этот текст

text.setString("Health: " + playerHealthString.str());

text.setPosition(250, 10);//задаем позицию текста

window.draw(text);//рисуем этот текст

if (p.health <= 1) {// ну тип игру прошли, если 1000 рублей собрали...

window.clear();

text.setString("YOU LOSE");

text.setPosition(300, 200);//задаем позицию текста

window.draw(text);//рисуем этот текст

}

if (p.money > p.money1) {// ну тип игру прошли, если 1000 рублей собрали...

window.clear();

text.setString("YOU WIN");

text.setPosition(300, 200);//задаем позицию текста

window.draw(text);//рисуем этот текст

}

if ((p.money < p.money1) && (p.health > 0)) {

window.draw(p.sprite);//рисуем спрайт объекта “p” класса “Player”

//рисуем врагов

}

if ((p.money < p.money1) && (p.health > 0)) {

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

{

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

}

}

window.display();

}

if (p.life == false) { // освобождение динамической памяти

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

{

delete \*it;

}

}

return 0;

}

* **Entity.cpp**

#ifndef Entity\_h

#define Entity\_h

using namespace sf;

class Entity {

public:

enum { left, right, up, down, stay } state;// тип перечисления - состояние объекта

float dx, dy, x, y, speed;

int w, h, health; //размер спрайта,переменная “health”, хранящая жизни игрока

bool life; //переменная “life” жизнь, логическая

Texture texture;//сфмл текстура

Sprite sprite;//сфмл спрайт

float CurrentFrame;//хранит текущий кадр

std::string name;//враги могут быть разные, врагов можно различать по именам

//каждому можно дать свое действие в update() в зависимости от имени

Entity(Image &image, float X, float Y, int W, int H, std::string Name);

virtual void update(float time) = 0;

FloatRect getRect() {//метод получения прямоугольника. его коорд, размеры (шир,высот).

FloatRect FR(x, y, w, h); // переменная FR типа FloatRect

return FR;

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

//Тип данных (класс) "sf::FloatRect" позволяет хранить четыре координаты прямоугольника

//в нашей игре это координаты текущего расположения тайла на карте

//далее это позволит спросить, есть ли ещё какой-либо тайл на этом месте

//эта ф-ция нужна для проверки пересечений

}

};

#endif

* **Enemy.h**

#ifndef Enemy\_h

#define Enemy\_h

using namespace sf;

class Enemy :public Entity

{

public:

int direction;//направление движения врага

Enemy(Image &image, float X, float Y, int W, int H, std::string Name);

void checkCollisionWithMap(float Dx, float Dy);

void update(float time);

};

#endif

* **PlayerClass.cpp**

#include <iostream>

#include <sstream>

#include <SFML/Graphics.hpp>

#include "map.h"

#include "Global.h"

#include "Entity.h"

#include "Player.h"

using namespace std;

using namespace sf;

Player::Player(Image &image, float X, float Y, float W, float H, std::string Name) :Entity(image, X, Y, W, H, Name)

{

money = -4; photo = 2; money1 = 1000;

state = stay; // объект стоит на месте

if (name == "Pacman") {

sprite.setTextureRect(IntRect(0, 0, w, h)); //Задаем спрайту один прямоугольник для вывода одного игрока. IntRect – для приведения типов

}

}

void Player::control() { // управление персом

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)) {

state = up;

speed = 0.1;

}

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

state = down;

speed = 0.1;

}

}

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

{

for (int i = y / 37; i < (y + h) / 37; i++)//проходимся по элементам карты

for (int j = x / 37; j < (x + w) / 37; j++)

{

if (TileMap[i][j] == '0')//если элемент - стена

{

if (Dy > 0) { y = i \* 37 - h; dy = 0; }//по Y

if (Dy < 0) { y = i \* 37 + 37; dy = 0; }//столкновение с верхними краями

if (Dx > 0) { x = j \* 37 - w; dx = 0; }//с правым краем карты

if (Dx < 0) { x = j \* 37 + 37; dx = 0; }// с

левым краем карты

}

if (TileMap[i][j] == 'h') { //если символ равен 'h' (money)

money += 1;// +1 money

TileMap[i][j] = ' ';//убираем money, ставим пустой/свободный элемент поля

}

}

}

void Player::update(float time) //функция "оживления/обновления" объекта класса. Принимает в себя

//время SFML, вследствие чего работает бесконечно, давая персонажу движение.

{

if (life) { // а герой жив?

control(); // управление персом, оО перс жив!!!

switch (state)//реализуем поведение в зависимости от направления. Каждая цифра соответствует направлению. ну или не цифра, а кнопка))

{

case right: {//состояние идти вправо

dx = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > photo) CurrentFrame -= photo;

sprite.setTextureRect(IntRect(35 \* int(CurrentFrame), 0, 35, 35));

break;

}

case left: {//состояние идти влево

dx = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > photo) CurrentFrame -= photo;

sprite.setTextureRect(IntRect(35 \* int(CurrentFrame), 35, 35, 35));

break;

}

case up: {//идти вверх

dy = -speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > photo) CurrentFrame -= photo;

sprite.setTextureRect(IntRect(70, 35 \* int(CurrentFrame), 35, 35));

break;

}

case down: {//идти вниз

dy = speed;

CurrentFrame += 0.005\*time;

if (CurrentFrame > photo) CurrentFrame -= photo;

sprite.setTextureRect(IntRect(105, 35 \* int(CurrentFrame), 35, 35));

break;

}

case stay: {//стоим, отдыхаем

dy = speed;

dx = speed;

break;

}

}

x += dx\*time; //движение по “X”

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

y += dy\*time; //движение по “Y”

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

speed = 0; //обнуляем скорость, чтобы персонаж остановился.

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

if (health <= 0) { life = false; }//тип игрок умер

}

}

* **Player.h**

#ifndef Player\_h

#define Player\_h

using namespace sf;

class Player :public Entity {

public:// класс Bгрока

int money, photo, money1; // игровые очки, кол-во кадров для анимации движения,переменная с "жизнью", кол-во очков до победы

//Конструктор с параметрами для класса Player

Player(Image &image, float X, float Y, float W, float H, std::string Name) ;

void control();

//Метод проверки столкновений с элементами карты

void checkCollisionWithMap(float Dx, float Dy);

void update(float time);

};

#endif

P.S. данная реализация игры «PACMAN» текстурно сильно отличается от первоначального оригинала, но весь смысл игры сохранен.