Экзаменационный билет

№ 19

«Разработка программных модулей программного обеспечения   
для компьютерных систем»

по специальности 09.02.03 «Программирование в компьютерных системах»

Тема: «Разработка приложения «2D-платформер: Управление игроком»

Выполнил студент группы П2-18

Щепкин Михаил Викторович

Королев, 2021

**Содержание**

[**1.** **ПК 1.6 Текстовая описание задачи.** 3](#_Toc76160257)

[**2.** **ПК 1.6 Диаграмма прецедентов.** 4](#_Toc76160258)

[**3.** **ПК.1.1 Диаграмма классов.** 5](#_Toc76160259)

[**4.** **ПК 1.2 Определение классов по диаграмме.** 6](#_Toc76160260)

[**5.** **ПК 1.2 Проект с классами для выполнения задания.** 9](#_Toc76160261)

[**6.** **ПК 1.3 Отладка одного из модулей при разработке.** 10](#_Toc76160262)

[**7.** **ПК 1.4 Подготовить тестовые наборы и провести тестирование одного из модулей.** 11](#_Toc76160263)

[**8.** **ПК 1.5 Значение временной сложности O для одного из методов. Значение O для оптимального кода** 13](#_Toc76160264)

1. **ПК 1.6 Текстовая описание задачи.**

Функциональное описание и управление.

В данном проекте будет представлено управление игроком, управление было сделано для клавиатуры с использованием 8 клавиш.

|  |  |
| --- | --- |
| Клавиша | Результат |
| ESC | Выход из игры без подтверждения |
| A | Идти влево |
| D | Идти вправо |
| W | Ползти по лестнице в верх |
| S | Ползти по лестнице вниз |
| F | Атаковать врага |
| SPACE | Прыжок игрока |
| CTRL | Использование щита |
| SHIFT | Использование бега |

1. **ПК 1.6 Диаграмма прецедентов.**

На рисунке № 1 будет представлена диаграмма прецедентов.

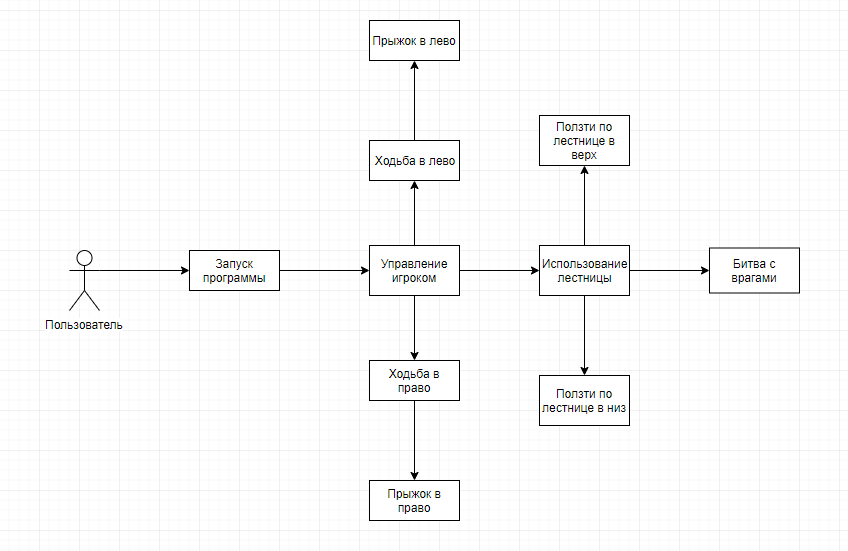


Рисунок № 1. (диаграмма прецедентов, управление игрока)

1. **ПК.1.1 Диаграмма классов.**

На рисунке № 2 будет представлена диаграмма классов.

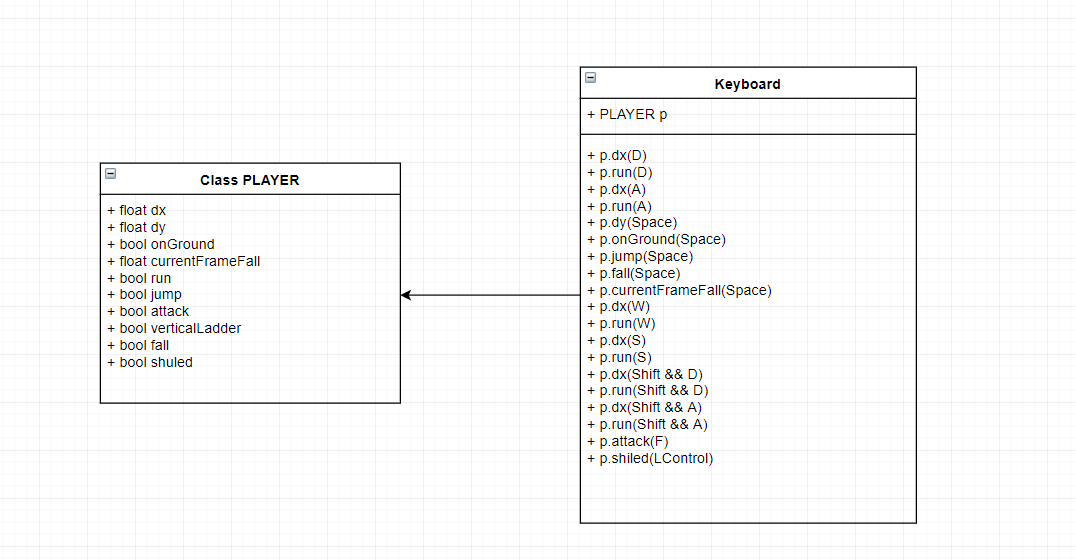


Рисунок № 2. (диаграмма классов, управление игроком)

1. **ПК 1.2 Определение классов по диаграмме.**

Листинг 1. Модуль PLAYER

class PLAYER {

public:

int level1();

float dx, dy, countDamage = 1, hud\_x, hud\_y;

FloatRect rect;

Texture hud\_r, hud\_hp, hud\_fon;

Sprite hud\_r\_sprite, hud\_hp\_sprite, hud\_fon\_sprite;

bool onGround, playerWalkLadderCollision = false, ladderWalk = false, animationDamage = false, jump, fall, run = false, verticalLadder = false, shield = false, gravity = true, death = false, attack = false, endGame = false, potion = true, potion2 = true, level2 = false, level3 = false, sound4 = false;

int left, position\_x = 50, position\_y = 580, i = 0;

Sprite sprite;

float currentFrameRun, currentFrameIdle, currentFrameJump, currentFrameShield, currentFrameDeath, currentFrameAttack, currentFrameFall, currentFrameDamage, currentFrameAttack2, hp = 100;

Листинг 2. Управление игроком

while (window.isOpen())

{

keyboard();

//...

Vector2f player\_position(p.position\_x, p.position\_y);

player\_position += Vector2f(10, 10);

ofstream ofs{ "positions.sav" };

ofs << player\_position.x << ' ' << player\_position.y << endl;

}

//...

if (p.endGame == false)

{

if (Keyboard::isKeyPressed(Keyboard::D))

{

p.dx = 0.15;

p.run = false;

}

if (Keyboard::isKeyPressed(Keyboard::A))

{

p.dx = -0.15;

p.run = false;

}

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

{

if (p.onGround)

{

p.dy = -0.5;

p.onGround = false;

p.jump = true;

p.fall = false;

p.currentFrameFall -= 5;

}

}

if ((p.onGround == true) and (p.currentFrameFall <= 14))

p.fall = true;

if (p.dx != 0)

{

// soundWalk = false;

}

if (p.verticalLadder == true)

{

if (Keyboard::isKeyPressed(Keyboard::W))

{

p.dy = -0.15;

p.run = false;

// p.onGround = true;

}

if (Keyboard::isKeyPressed(Keyboard::S))

{

p.dy = 0.15;

p.run = false;

// p.onGround = true;

}

if (!Keyboard::isKeyPressed(Keyboard::W) && !Keyboard::isKeyPressed(Keyboard::S))

{

p.dy = 0;

}

}

if (Keyboard::isKeyPressed(Keyboard::LShift) && Keyboard::isKeyPressed(Keyboard::D) && p.onGround == true)

{

p.dx = 0.5;

p.run = true;

}

if (Keyboard::isKeyPressed(Keyboard::LShift) && Keyboard::isKeyPressed(Keyboard::A) && p.onGround == true)

{

p.dx = -0.5;

p.run = true;

}

if (!Keyboard::isKeyPressed(Keyboard::D) && !Keyboard::isKeyPressed(Keyboard::A) && p.onGround == true)

{

sound2.play();

}

if (Keyboard::isKeyPressed(Keyboard::F) && p.onGround == true && p.attack == false && p.shield == false)

{

p.attack = true;

sound.play();

}

if (Keyboard::isKeyPressed(Keyboard::LControl) && p.onGround == true)

{

p.shield = true;

}

}

if (e.onGround == false)

{

e.dy = 0.4;

}

if ((e.animationDamage == true) or (e2.animationDamage == true))

{

sound3.play();

}

//...

if (p.sound4 == true)

{

sound4.play();

p.sound4 = false;

}

//...

}

1. **ПК 1.2 Проект с классами для выполнения задания.**

Листинг 3. Главный модуль приложения

#include <iostream>

#include "SFML/Graphics.hpp"

#include "SFML/Audio.hpp"

#include <fstream>

#include <SFML/Window.hpp>

#include <Windows.h>

#include <math.h>

#include <thread>

#include <chrono>

#include <string>

using namespace sf;

using namespace std;

int ground = 400;

const int H = 128;

const int W = 366;

const int sizeMapCollisionA = 16;

const int sizeMapCollisionB = 16;

const int sizeMapCollisionL = 16;

const int sizeMapCollisionI = 16;

int choiseCharacterGame;

String level;

String TileMap[H];

String TileMap2[H];

String TileMap3[H];

1. **ПК 1.3 Отладка одного из модулей при разработке.**

В ходе написания кода при попытке запустить программу было получено данное сообщение (Рисунок № 3):

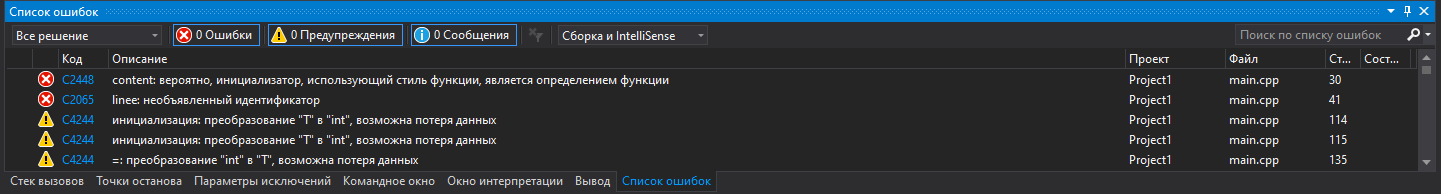


Рисунок № 3. (Сообщение об ошибке)

При получении данного сообщения были просмотрены строки 30, 41. Были обнаружены ошибки, которые впоследствии были исправлены, а после попытки запуска программы повторно было получено данное сообщение (Рисунок № 4):

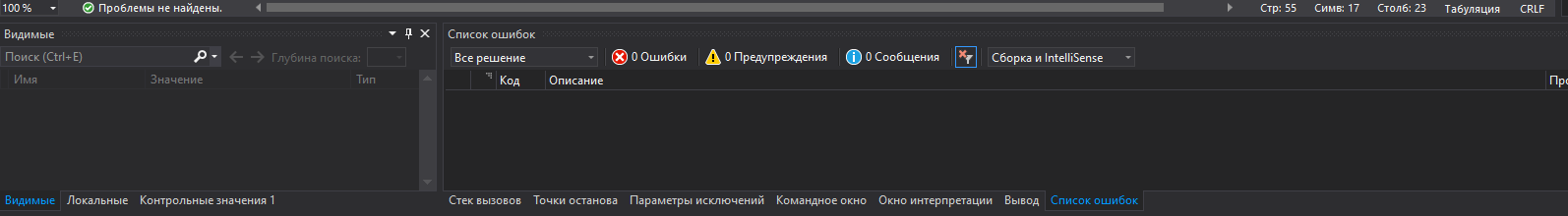


Рисунок № 4. (Ошибок нет)

1. **ПК 1.4 Подготовить тестовые наборы и провести тестирование одного из модулей.**

В виде тестового набора был использован специальный шаблон в Visual Studio под названием: «Проект модульного теста» в результате чего было выявлено 3 ошибки в функциях keyboard (Рисунок № 5):

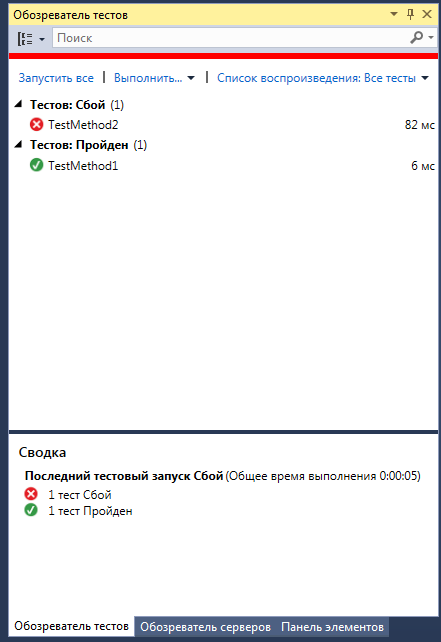


Рисунок № 5. (до теста)

Время компиляции было успешно улучшена на 0:00:01

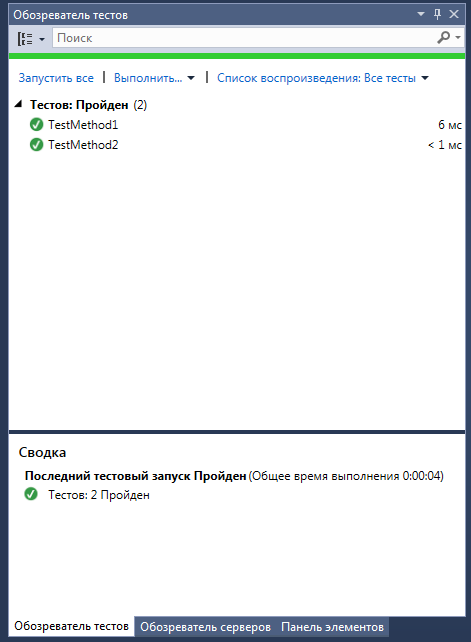


Рисунок № 6. (после теста)

1. **ПК 1.5 Значение временной сложности O для одного из методов. Значение O для оптимального кода**

О для модуля управления игроком = О(1), т.к вычислительная сложность алгоритма не зависит от входных данных. Однако, это не значит, что алгоритм выполняется за одну операцию или требует очень мало времени. Это означает, что время не зависит от входных данных.

Значение О для оптимального когда будет О(n), т.к сложность алгоритма линейно растёт с увеличением входных данных. Другими словами, удвоение размера входных данных удвоит и необходимое время для выполнения алгоритма.