Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Севастопольский государственный университет»

Кафедра информационных систем

КУРСОВАЯ РАБОТА

ПО ДИСЦИПЛИНЕ «ОБЪЕКТНО-ОРИЕНТИРОВАНИЕ ПРОГРАММИРОВАНИЕ»

ПРОГРАММА «Super Pahom – History of Green Elephant»

Пояснительная записка

Листов

Студентов 2 курса группы ИС/б-21-о

направление подготовки 09.03.02

*(подпись)\_\_\_\_\_\_\_* Акименко В.А.

*(подпись)\_\_\_\_\_\_\_* Волков А.А.

*(подпись)\_\_\_\_\_\_\_* Корж А.И.

« » \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(должность, ученое звание, фамилия и инициалы)

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Члены комиссии \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия и инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия и инициалы)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (фамилия и инициалы)

г. Севастополь – 2016 г.

# **Аннотация**

В данном курсовом проекте рассматривается разработка игрового приложения жанра ‘*аркада’ (Super Mario Bros)*. Так же присутствует пояснение взаимодействие классов в разработанной системе, отображенное в виде UML диаграмм.

В данном проекте так же рассмотрен разработанный интерфейс взаимодействия пользователя с приложением.

Содержание

[Введение 3](#_Toc452074714)

[Список исполнителей 4](#_Toc452074715)

[Постановка задачи 5](#_Toc452074716)

[Проектное решение 6](#_Toc452074717)

[Программная реализация 8](#_Toc452074718)

[Заключение 10](#_Toc452074719)

Дата

Корж А.И..

Волков А.А.

Куркчи А.Э

Мжачев И.А.

Изм.

Лист

№ докум.

Подпись

Дата

Лист

3

КУРСОВАЯ РАБОТА

Выполнили

Акименко В.А.

Провер.

Пелипас В.О..

Утверд.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Лит.

Листов

Кафедра ИС

Группа ИС/б-21-о

# **Введение**

Целью написания курсового проекта является закреплением, углублением и обобщение знаний и навыков по пройдённому курсу «Объектно-Ориентированное Программирование». Задача была такова, разработать игровое приложение по жанру ‘*аркада*’, которая была выполнена, название получившегося приложение – *‘Super Pahom – History of Green Elephant’*.

В ходе разработки были проведены операции проектирования архитектуры приложения, применены все основные принципы ООП, была детально освоена библиотека SFML.

Так как велась командная разработка, мы использовали распределенную систему управления версиями Git, данные хранили на облаке и каждый из нас работал с локальным репозиторием, который был клонирован с облака.

# **Список исполнителей**

Команда разработавшая данное приложение состоит из трёх человек, а именно: Акименко В.А., Волков А.А., Корж А.И. В проектировании архитектуры принимали участие все из перечисленных людей, также совместно решались основные проблемы и риски проекта, происходил выбор языка программирования, вопросы организации разработки. Перечислим личный вклад каждого участника команды:

* Акименко В.А. – реализация взаимодействия персонажа с окружающим миром, противниками и боссом; реализация музыкального сопровождения проекта.
* Волков А.А. – организация основных принципов ООП в проекте; отладка и проектирование диаграмм.
* Корж А.И. – реализация библиотеки SFML; написание уровней; реализация перехода по уровням; создание классов и организация полей и методов.

# **Постановка задачи**

**Цель проекта:** написать программу, в которой происходит управление пользователем игровым персонажем (вид сбоку), соблюдая основные принципы объектно-ориентированного программирования.

**Цель игры:** дойти до финального уровня и победить главного противника (босса), не погибнув по пути.

Программа должна включать в себя:

* Отображение и взаимодействие персонажа, противников, окружающей среды, и бонусных блоков;
* Создание панели меню, для получения информации о программе, сохранения результата и выхода из программы;
* Возможность передвижения персонажем по карте вперед, вверх, вниз и назад;
* Визуализация предметов в игровом поле.

*Карты* представляют собой переменные типа string, и прорисовываются ежесекундно. На каждой карте изначально находится определенное количество противников (мобов) и на последней карте существует босс.

*Игрок* на каждом уровне имеет определенное количество здоровья, которое уменьшается при взаимодействии с мобами и боссом. Если здоровье игрока опустится до нуля – игра информирует о завершении, проигрыше. Так же, игрок имеется возможность выпить *бутыль энергии*. После этого он становится невосприимчивым некоторое время к атакам противников.

*Мобы* имеют одну единицу здоровья, для того чтобы уничтожить моба необходимо прыгнуть на него сверху. При каждом взаимодействии с героем мобы наносят ему 1 единицу урона.

*Босс* – особый вид моба, имеющий в 10 раз больше здоровья. Он наносит игровому персонажу 3 единицы урона.

# **Проектное решение**

Были выделены основные классы: Переменные, Игрок, Монстр, Босс, Бутыль энергии. Такие классы как, Игрок, Монстр, Бутыль энергии, наследуются от класса переменные (наследуют основные переменные). Класс Босс наследует методы и поля классов Переменные и Монстр, имеющих идентификатор доступа public, переопределяя некоторые методы и расширяя классы-родители своими методами. Все производные классы имеют как поля с открытым доступом, так и поля с закрытым. Все возможные действия с последними осуществляются благодаря таким свойствам, как get и set. В классе Игрок имеется поле clocks с типом доступа private, действующее в пределах данного класса. Вся иерархия наследования представлена на диаграмме классов (рисунок 1).

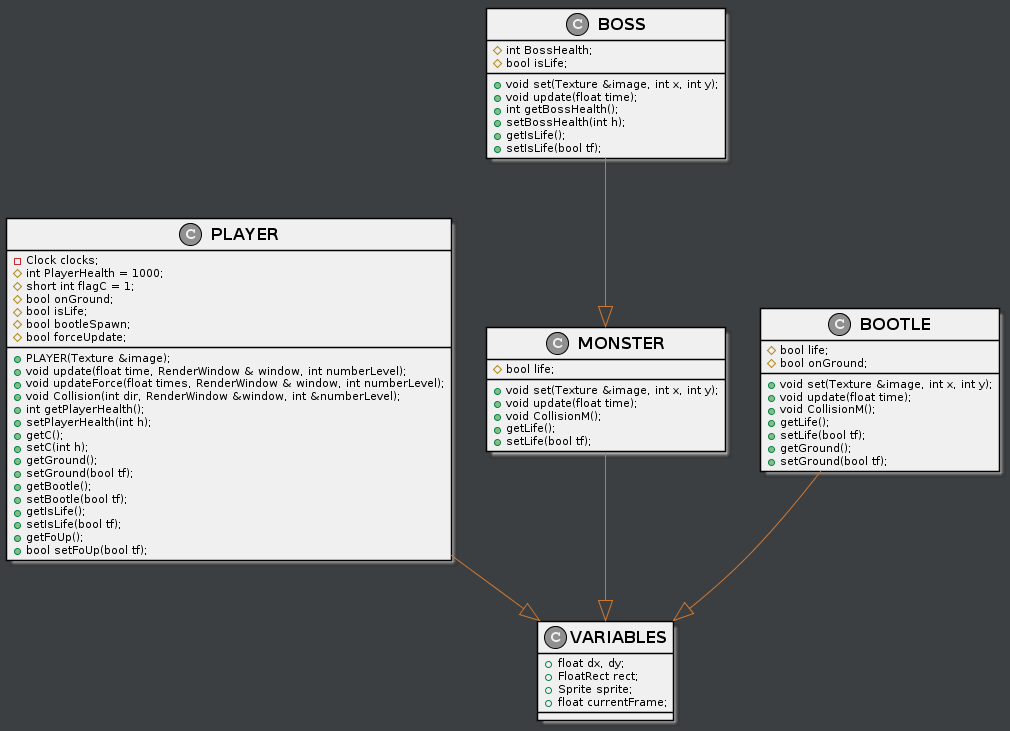


Рисунок 1 – Диаграмма классов

Жизненный цикл программы начинается с главного меню, где имеются кнопки «Новая игра», «Продолжить», «Справка», «Выход». После нажатия кнопки начала новой игры запускается первый уровень игры, при прохождении точки конца карты уровень автоматически меняется на следующий. Наша программа предоставляет возможность пользователю регулировать звук во время игры (уменьшать(O)/увеличивать(I) громкость, переключать музыку (N,M). Так же, есть возможность автоматического переключения уровней (D).

В любой момент пользователь имеет возможность выйти из игры (Escape) и продолжить ее через некоторое время, нажав кнопку продолжения игры.

Выход из игры (закрытие приложения) осуществляется через главное меню, т.е. пользователь сначала должен выйти в главное меню (Escape) и затем выбрать в нем пункт выхода.

Конец жизненного цикла приложения достигается в двух случаях:

* Когда здоровье игрока становится равным нулю (приложение уведомляет о конце игры, пользователь должен выйти в главное меню и начать новую игру либо выйти из нее);
* Когда здоровье босса становится равным нулю (приложение уведомляет о победе, пользователь должен выйти в главное меню и начать новую игру либо выйти из нее);

На рисунке 2 представлена диаграмма вариантов использования.

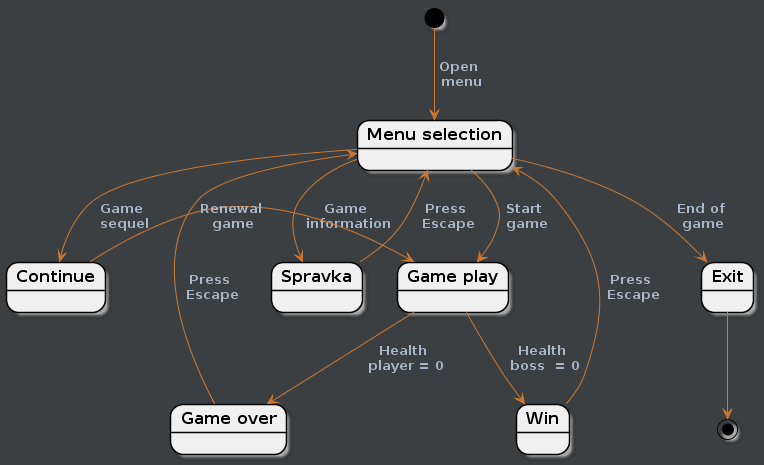


Рисунок 2 – Диаграмма вариантов использования

# **Программная реализация**

При разработки данного проекта первым делом мы определились с языком программирования. Выбор стоял между языками *C++*, *C#*, *Java*. Наш выбор остановился на языке C++ по следующим причинам:

* Данный язык сочетает как высокоуровневые средства программирования, так и низкоуровневые;
* Простота в реализации основных принципов ООП;
* Код эффективно упрощается за счет перегрузки функция и операторов;
* Кроссплатформенность;
* Высокая производительность кода;
* Низкая требовательность к ресурсам (наш исполнительный файл весит всего 154 КБ);
* Поддержка физической и логической константности.

При работе над проектом нами была детально изучена мультимедийная библиотека SFML, предоставляющая ряд модулей для простого и быстрого создания игр. Полное название библиотеки - **Simple and Fast Multimedia Library**, что в переводе с английского означает «простая и быстрая мультимедиа библиотека». Данная библиотека является объектно-ориентированным аналогом библиотеки SDL. В нашей работе мы пользовались следующими модулями:

1. Graphics, благодаря которому возможно отображение графических примитивов и изображений. А он в свою очередь требует для своей работы модуль Window;
2. Audio – модуль, благодаря которому мы смогли реализовать все действия с музыкой во время игры.

Наш выбор пал именной на SFML, так как:

* Данная библиотека является кроссплатформенной;
* Имеется ее открытый код;
* Библиотеки работают как на x64, так и на x86;
* Поддержка различных компиляторов;
* Имеет лицензию zlib/png license;
* Простота в эксплуатации;
* Независимость библиотек друг от друга.

**Выполнение проекта**

На рисунке 1 изображен вид главного меню:

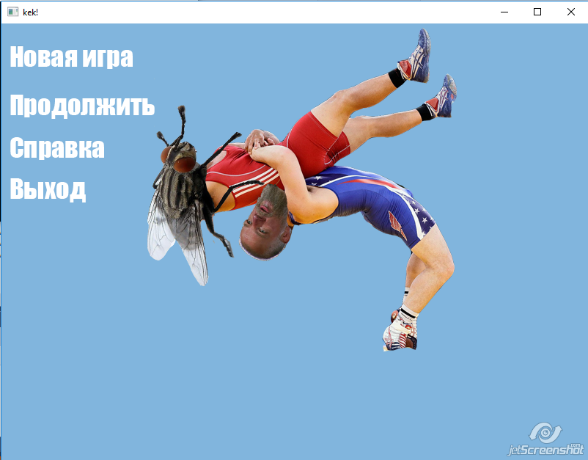


Рисунок 1 – Главное меню

На рисунке 2 изображена модификация персонажа после употребления бутылки энергии:



Рисунок 2 – Персонаж после употребления бутылки энергии

На рисунке 3 изображен один из вариантов окончания игры – смерть игрового персонажа:

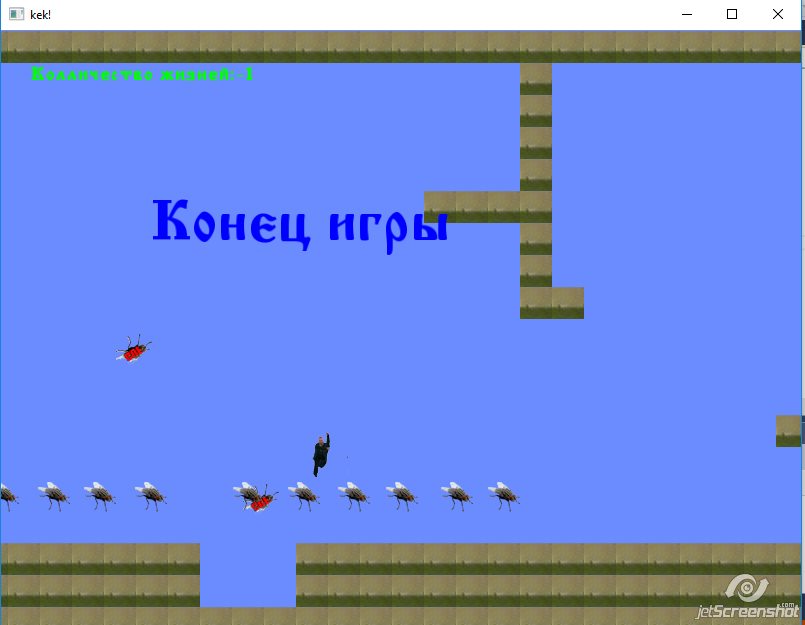


Рисунок 3 – Конец игры (проигрыш)

На рисунке 4 изображен один из вариантов окончания игры – смерть босса:



Рисунок 4 – Конец игры (победа)

# **Заключение**

В ходе выполнения курсового проекта было закреплены, углублены и усовершенствованы знания и навыки работы по дисциплине «Объектно-Ориентированное Программирование».

Было проведено архитектурное проектирование приложения с помощью основных принципов ООП, в ходе которого были построены UML-диаграммы.

Также в ходе разработки игрового приложения были сформированы навыки работы в команде, работы с системой контроля версий Git и мультимедийной библиотекой SFML.

Были получены практические навыки применения принципов объектно-ориентированного программирования, наследования, полиморфизма, инкапсуляции.

Приложение А

/\*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Севастопольский государственный университет

Кафедра информационных систем

Проект по разработке игрового приложения жанра “аркада”

Текст программы

РАЗРАБОТАЛ

Студенты гр. ИС-21

Акименко В.А., Волков А.А., Корж А.И.

2016

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++\*/

#include <SFML/Graphics.hpp>

#include <SFML/Audio.hpp>

#include <sstream>

#include <Windows.h>

using namespace sf;

float offsetX = 0, offsetY = 0;

const int H = 19;

const int W = 135;

int muss = 1;//переменная для переключения треков

Music music;

void changeLevel(int &numberLevel); //функция перехода между уровнями

void beggin(RenderWindow &, int &numberLevel);

int Muzika(int a);//функция переключения треков

String TileMap[H] = {

"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"B B B BB B B BB",

"B B B BB B B BB",

"B B B BB B B BB",

"B 0000 B BB B B BB",

"B BBBB BB B B BB",

"B B E",

"BBBBB C B B E",

"B BB BB BB E",

"B BBB BB E",

"B B BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BB BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BB BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BBBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BBBB BB BBBBB BB BBBB BB BBBBB BB",

"BBBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"BBBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

};

String TileMap1[H] = {

"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"B B B BB B B BB",

"B B B BB B B BB",

"B B B BB B B BB",

"B 0000 B BB B B BB",

"B BBBB BB B B BB",

"B B E",

"BBBBB C B B E",

"B BB BB BB E",

"B BBB BB E",

"B B BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BB BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BB BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BBBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BBBB BB BBBBB BB BBBB BB BBBBB BB",

"BBBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"BBBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

};

String TileMap2[H] = {

"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"B B B BB B B BB",

"B B B BB B B BB",

"B B B BB B B BB",

"B 0000 B BB B B BB",

"B BBBB BB B B BB",

"B B E",

"BBBBB C B B E",

"B B BB BB BB E",

"B B B BBB BB E",

"B B B B BBBB BB BBBBB BB",

"B BB B BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BB BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BBBB BB BBBBB BB",

"B BBBB BB BBBBB BB BBBB BB BBBBB BB",

"B BBBB BB BBBBB BB BBBB BB BBBBB BBB",

"BBBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"BBBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

};

String TileMap3[H] = {

"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"B B B BB B B BB",

"B B B BB B B BB",

"B B B BB B B BB",

"B 0000 B BB B B BB",

"B BBBB BB B BBBBBBBBBBBBB BBBBBB BBBBB",

"B B BBBB BB",

"BBBBB C B B BB BB",

"B BB BB BB BBBB BB",

"B BBB BB BBBB BB",

"B B BBBB BB BBBBB BB BBBBBB BB",

"B BB BBBB BB BBBBB BB BBBB BB",

"B BB BBBB BB BBBBB BB BBBB BB",

"B BBBBB BB BBBBB BB BBBB BB",

"B BBBB BB BBBBB BB BBBB BB",

"B BBBB BB",

"BBBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"BBBBBBBBBBBBBBBBBBBBBBB BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB",

};

class VARIABLES {

public:

float dx, dy;

FloatRect rect;

Sprite sprite;

float currentFrame;

};

class PLAYER : public VARIABLES {

private:

Clock clocks;

protected:

int PlayerHealth = 1000;

short int flagC = 1;

bool onGround;

bool isLife;

bool bootleSpawn;

bool forceUpdate;

public:

PLAYER(Texture &image)

{

sprite.setTexture(image);

rect = FloatRect(7 \* 32, 9 \* 32, 40, 50);

dx = dy = 0.1;

currentFrame = 0;

}

int getPlayerHealth() {

return PlayerHealth;

};

int setPlayerHealth(int h) {

PlayerHealth = PlayerHealth - h;

return PlayerHealth;

};

int getC() {

return flagC;

};

int setC(int h) {

flagC = flagC - h;

return flagC;

};

bool getGround() {

return onGround;

};

bool setGround(bool tf) {

return (onGround = tf);

};

bool getBootle() {

return bootleSpawn;

};

bool setBootle(bool tf) {

return (bootleSpawn = tf);

};

bool getIsLife() {

return isLife;

};

bool setIsLife(bool tf) {

return (isLife = tf);

};

bool getFoUp() {

return forceUpdate;

};

bool setFoUp(bool tf) {

return (forceUpdate = tf);

};

void update(float time, RenderWindow & window, int numberLevel)

{

rect.left += dx\*time;

Collision(0, window, numberLevel);

if (!getGround()) dy = dy + 0.005\*time;

rect.top += dy\*time;

setGround(false);

Collision(1, window, numberLevel);

currentFrame += 0.005\*time;

if (currentFrame >= 6) currentFrame -= 6;

if (dx > 0) sprite.setTextureRect(IntRect(43 \* int(currentFrame), 257, 43, 50));

if (dx < 0) sprite.setTextureRect(IntRect(43 \* int(currentFrame) + 43, 257, -43, 50));

sprite.setPosition(rect.left - offsetX, rect.top - offsetY);

dx = 0;

}

void updateForce(float times, RenderWindow & window, int numberLevel)

{

if (forceUpdate == true)

{

times += clocks.getElapsedTime().asSeconds();

rect.left += dx\*times;

Collision(0, window, numberLevel);

if (!getGround()) dy = dy + 0.005\*times;

rect.top += dy\*times;

setGround(false);

Collision(1, window, numberLevel);

currentFrame += 0.005\*times;

if (currentFrame >= 3) currentFrame -= 2;

if (currentFrame <= 1) currentFrame += 1;

if (dx > 0) sprite.setTextureRect(IntRect(39 \* int(currentFrame) + 39, 1002, -39, 57));

if (dx < 0) sprite.setTextureRect(IntRect(39 \* int(currentFrame), 1002, 39, 57));

sprite.setPosition(rect.left - offsetX, rect.top - offsetY);

dx = 0;

if (times >= 30) { forceUpdate = false; }

}

}

void Collision(int dir, RenderWindow &window, int &numberLevel)

{

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

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

if (TileMap[i][j] == 'B' || TileMap[i][j] == 'S' || TileMap[i][j] == 'W' || TileMap[i][j] == 'F') {

if ((dx > 0) && (dir == 0)) rect.left = j \* 32 - rect.width;

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

if ((dy > 0) && (dir == 1)) { rect.top = i \* 32 - rect.height; dy = 0; setGround(true); }

if ((dy < 0) && (dir == 1)) { rect.top = i \* 32 + 32; dy = 0; }

}

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

if ((dx > 0) && (dir == 0)) rect.left = j \* 32 - rect.width;

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

if ((dy < 0) && (dir == 1)) { rect.top = i \* 32 + 32; dy = 0; TileMap[i][j] = ' '; }

if ((dy > 0) && (dir == 1)) { rect.top = i \* 32 - rect.height; dy = 0; setGround(true); }

}

if (TileMap[i][j] == '0')

{

if ((dx > 0) && (dir == 0)) rect.left = j \* 32 - rect.width;

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

if ((dy < 0) && (dir == 1)) { rect.top = i \* 32 + 32; dy = 0; TileMap[i][j] = ' '; }

if ((dy > 0) && (dir == 1)) { rect.top = i \* 32 - rect.height; dy = 0; setGround(true);}

}

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

{

if ((dy < 0) && (dir == 1))

{

rect.top = i \* 32 + 32;

dy = 0;

if (getC() > 0) {

bootleSpawn = true;

setC(1);

}

}

}

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

{

if (numberLevel != 3) {

numberLevel = numberLevel + 1;

beggin(window, numberLevel);

}

else {

numberLevel = 1;

beggin(window, numberLevel);

}

}

}

}

};

};

class MONSTER : public VARIABLES {

protected:

bool life;

public:

void set(Texture &image, int x, int y)

{

sprite.setTexture(image);

rect = FloatRect(x, y, 42, 50); //первые 2 цифры за начальное положение отвечают

dx = dy = 2;

currentFrame = 0;

life = true;

}

bool getLife() {

return life;

};

bool setLife(bool tf) {

return (life = tf);

};

void update(float time)

{

rect.left += dx \* time;

CollisionM();

currentFrame += time \* 0.005;

if (currentFrame > 2) currentFrame -= 2;

sprite.setTextureRect(IntRect(43 \* int(currentFrame), 319, 43, 35));

if (!life) sprite.setTextureRect(IntRect(86, 319, 43, 35));

sprite.setPosition(rect.left - offsetX, rect.top - offsetY);

}

void CollisionM()

{

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

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

if (TileMap[i][j] == 'B' || TileMap[i][j] == 'C')

{

if (dx>0)

{

rect.left = j \* 32 - rect.width; dx \*= -1;

}

else if (dx<0)

{

rect.left = j \* 32 + 32; dx \*= -1;

}

}

}

};

class BOOTLE : public VARIABLES {

protected:

bool life;

bool onGround;

public:

void set(Texture &image, int x, int y)

{

sprite.setTexture(image);

rect = FloatRect(x, y, 42, 50); //первые 2 цифры за начальное положение отвечают

dx = dy = 0.1;

setGround(false);

life = true;

}

bool getLife() {

return life;

};

bool setLife(bool tf) {

return (life = tf);

};

bool getGround() {

return onGround;

};

bool setGround(bool tf) {

return (onGround = tf);

};

void update(float time)

{

rect.left += dx \* time;

CollisionM();

sprite.setTextureRect(IntRect(89, 1075, 17, 64));

if (!life) sprite.setTextureRect(IntRect(451, 300, 32, 32));

sprite.setPosition(rect.left - offsetX, rect.top - offsetY);

}

void CollisionM()

{

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

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

if (TileMap[i][j] == 'B' || TileMap[i][j] == 'C' || TileMap[i][j] == '0')

{

if (dx>0)

{

rect.left = j \* 32 - rect.width; dx \*= -1;

}

else if (dx<0)

{

rect.left = j \* 32 + 32; dx \*= -1;

}

}

}

};

class BOSS: public MONSTER {

protected:

int BossHealth = 10;

bool isLife;

public:

void set(Texture &image, int x, int y)

{

sprite.setTexture(image);

rect = FloatRect(x, y, 42, 50); //первые 2 цифры за начальное положение отвечают

dx = 0.5;

dy = 2;

currentFrame = 0;

life = true;

}

bool getIsLife() {

return isLife;

};

bool setIsLife(bool tf) {

return (isLife = tf);

};

void update(float time)

{

rect.left += dx \* time;

CollisionM();

currentFrame += time \* 0.005;

if (currentFrame > 2) currentFrame -= 2;

sprite.setTextureRect(IntRect(24 \* int(currentFrame), 1322, 97, 168));

if (!isLife) sprite.setTextureRect(IntRect(231, 1345, 167, 97));

sprite.setPosition(rect.left - offsetX, rect.top - offsetY);

}

int getBossHealth() {

return BossHealth;

};

int setBossHealth(int h) {

BossHealth = BossHealth - h;

return BossHealth;

};

void CollisionM()

{

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

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

if (TileMap[i][j] == 'B' || TileMap[i][j] == 'C')

{

if (dx>0)

{

rect.left = j \* 32 - rect.width; dx \*= -1;

}

else if (dx<0)

{

rect.left = j \* 32 + 32; dx \*= -1;

}

}

}

};

void menu(RenderWindow & window, int numberLevel) {

music.stop();

Texture menuTexture1, menuTexture2, menuTexture3, menuTexture4, aboutTexture, menuBackground;

menuTexture1.loadFromFile("fang.png");

menuTexture2.loadFromFile("fang.png");

menuTexture3.loadFromFile("fang.png");

menuTexture4.loadFromFile("fang.png");

aboutTexture.loadFromFile("fang.png");

menuBackground.loadFromFile("fang.png");

Sprite menu1(menuTexture1), menu2(menuTexture2), menu3(menuTexture3), menu4(menuTexture4), about(aboutTexture), menuBg(menuBackground);

music.openFromFile("Polkovnik.ogg");

music.play();

music.setLoop(true);

float vol = 100;

Music flint;

flint.openFromFile("Flint.ogg");

bool isMenu = 1;

int menuNum = 0;

menu1.setTextureRect(IntRect(30, 9, 216, 35));

menu2.setTextureRect(IntRect(30, 46, 216, 45));

menu3.setTextureRect(IntRect(30, 91, 216, 45));

menu4.setTextureRect(IntRect(30, 138, 216, 36));

about.setTextureRect(IntRect(16, 1151, 444, 300));

menuBg.setTextureRect(IntRect(0, 439, 444, 447));

menu1.setPosition(0, 30);

menu2.setPosition(0, 90);

menu3.setPosition(0, 150);

menu4.setPosition(0, 210);

about.setPosition(100, 100);

menuBg.setPosition(180, 0);

//////////////////////////////МЕНЮ///////////////////

while (isMenu)

{

menu1.setColor(Color::White);

menu2.setColor(Color::White);

menu3.setColor(Color::White);

menu4.setColor(Color::White);

menuNum = 0;

window.clear(Color(129, 181, 221));

if (Keyboard::isKeyPressed(Keyboard::I))

{

vol = vol + 0.1;

if (vol>100) vol = 100;

music.setVolume(vol);

}

if (Keyboard::isKeyPressed(Keyboard::O))

{

vol = vol - 0.1;

if (vol<0) vol = 0;

music.setVolume(vol);

}

if (IntRect(0, 40, 216, 19).contains(Mouse::getPosition(window))) { menu1.setColor(Color::Red); menuNum = 1; }

if (IntRect(0, 90, 216, 40).contains(Mouse::getPosition(window))) { menu2.setColor(Color::Red); menuNum = 2; }

if (IntRect(0, 150, 216, 40).contains(Mouse::getPosition(window))) { menu3.setColor(Color::Red); menuNum = 3; }

if (IntRect(0, 210, 216, 40).contains(Mouse::getPosition(window))) { menu4.setColor(Color::Red); menuNum = 4; }

if (Mouse::isButtonPressed(Mouse::Left))

{

if (menuNum == 1) { music.stop(); isMenu = false; beggin(window,numberLevel); }//если нажали первую кнопку, то выходим из меню

if (menuNum == 2) { isMenu = false; }

if (menuNum == 3) {

music.stop(); window.draw(about); window.display(); flint.play(); while (1) {

if (Keyboard::isKeyPressed(Keyboard::Escape)) { flint.stop(); music.play(); break; }

}

}

if (menuNum == 4) { window.close(); isMenu = false; }

}

window.draw(menuBg);

window.draw(menu1);

window.draw(menu2);

window.draw(menu3);

window.draw(menu4);

window.display();

}

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

}

void changeLevel(int &numberLevel) {

for (int i = 0; i < H; i++) {

if (numberLevel == 1) { TileMap[i] = TileMap1[i]; }

if (numberLevel == 2) { TileMap[i] = TileMap2[i]; }

if (numberLevel == 3) { TileMap[i] = TileMap3[i]; }

}

}

int Muzika(int a) {

music.setLoop(false);

music.stop();

switch (a) {

case 1:

music.openFromFile("Myhi.ogg");

break;

case 2:

music.openFromFile("Enjoy.ogg");

break;

case 3:

music.openFromFile("Voodoo.ogg");

break;

case 4:

music.openFromFile("TriReki.ogg");

break;

case 5:

music.openFromFile("Tetris.ogg");

break;

case 6:

music.openFromFile("ScholDance.ogg");

break;

case 7:

music.openFromFile("PolyBit.ogg");

break;

case 8:

music.openFromFile("Polka.ogg");

break;

case 9:

music.openFromFile("Lambada.ogg");

break;

case 10:

music.openFromFile("Gryppa.ogg");

break;

case 11:

music.openFromFile("Bratishka.ogg");

case 12:

music.stop();

return 0;

default:

if ((muss > 11) && (muss !=12)) muss = 1;

else if (muss < 1) muss = 11;

}

music.play();

music.setLoop(true);

return 0;

}

void beggin(RenderWindow &window, int &numberLevel) {

offsetX = 0;

offsetY = 0;

float Zvuk = 100;

//Level lvl;

changeLevel(numberLevel);

Muzika(rand() % 10);

Texture t;

t.loadFromFile("fang.png");

float currentFrame = 0;

//шрифты

Font font;

font.loadFromFile("CyrilicOld.ttf");

Text texts("", font, 20);

texts.setColor(Color::Green);

texts.setStyle(sf::Text::Bold);

Font gont;

gont.loadFromFile("CyrilicOld.ttf");

Text textd("", gont, 60);

textd.setColor(Color::Blue);

textd.setStyle(sf::Text::Bold);

Font font1;

font1.loadFromFile("CyrilicOld.ttf");

Text textf("", font1, 200);

textf.setColor(Color::Red);

textf.setStyle(sf::Text::Bold);

Font font2;

font2.loadFromFile("CyrilicOld.ttf");

Text textb("", font2, 20);

textb.setColor(Color::Blue);

textb.setStyle(sf::Text::Bold);

BOOTLE b;

BOSS bos;

MONSTER m;

MONSTER monster;

MONSTER monster1;

MONSTER monster2;

MONSTER monster3;

MONSTER monster4;

MONSTER monster5;

MONSTER monster6;

MONSTER monster7;

MONSTER monster8;

MONSTER monster9;

MONSTER monster10;

MONSTER monster11;

PLAYER p(t);

p.setIsLife(true);

p.setFoUp(false);

p.setBootle(false);

m.set(t, 1000, 300);

monster.set(t, 45, 50);

monster1.set(t, 450, 450);

if (numberLevel == 3) {

bos.set(t, 4100, 300);

}

monster2.set(t, 500, 450);

monster3.set(t, 400, 450);

monster4.set(t, 350, 450);

monster5.set(t, 300, 450);

monster6.set(t, 250, 450);

monster7.set(t, 200, 450);

monster8.set(t, 150, 450);

monster9.set(t, 100, 450);

monster10.set(t, 550, 450);

monster11.set(t, 600, 450);

Sprite tile(t);

SoundBuffer buffer;

buffer.loadFromFile("kyrlik.ogg");

Sound sound(buffer);

Music vilgelm;

vilgelm.openFromFile("Vilgelm.ogg");

Music music;

Clock clock;

RectangleShape rectangle(Vector2f(32, 32));

while (window.isOpen())

{

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

clock.restart();

time = time / 800;

texts.setPosition(30, 30);

textd.setPosition(150, 150);

if (time>20) time = 20;

Event event;

while (window.pollEvent(event))

{

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

window.close();

}

if (Keyboard::isKeyPressed(Keyboard::M))

{

muss++;

Muzika(muss);

}

if (Keyboard::isKeyPressed(Keyboard::N))

{

muss--;

Muzika(muss);

}

if (Keyboard::isKeyPressed(Keyboard::Escape))

{

music.pause(); menu(window,numberLevel);

}

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

{

music.pause();

numberLevel = (numberLevel + 1) % 3 + 1;

beggin(window, numberLevel);

}

if (p.getIsLife()) {

//move left

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

{

p.dx = -1;

}

//move right

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

{

p.dx = 1;

}

//move up

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

{

if ((p.getGround()) && (!p.getFoUp())) { sound.play(); p.dy = -1.5; p.setGround(false); }

if ((p.getGround()) && (p.getFoUp())) { sound.play(); p.dy = -2; p.setGround(false); }

}

}

//звучок громкость

if (Keyboard::isKeyPressed(Keyboard::I))

{

Zvuk = Zvuk + 1;

if (Zvuk>100) Zvuk = 100;

music.setVolume(Zvuk);

sound.setVolume(Zvuk);

vilgelm.setVolume(Zvuk);

}

if (Keyboard::isKeyPressed(Keyboard::O))

{

Zvuk = Zvuk - 1;

if (Zvuk<0) Zvuk = 0;

music.setVolume(Zvuk);

sound.setVolume(Zvuk);

vilgelm.setVolume(Zvuk);

}

if (p.getBootle() == true) {

b.set(t, 100, 100);

p.setBootle(false);

}

if ((p.getIsLife()) && (p.getFoUp() == false) ) {

m.update(time);

monster.update(time);

p.update(time,window,numberLevel);

b.update(time);

monster1.update(time);

monster2.update(time);

monster3.update(time);

monster4.update(time);

monster5.update(time);

monster6.update(time);

monster7.update(time);

monster8.update(time);

monster9.update(time);

monster10.update(time);

monster11.update(time);

bos.update(time);

}

else if ((p.getIsLife()) && (p.getFoUp() == true)) {

m.update(time);

monster.update(time);

monster1.update(time);

monster2.update(time);

monster3.update(time);

monster4.update(time);

monster5.update(time);

monster6.update(time);

monster7.update(time);

monster8.update(time);

monster9.update(time);

monster10.update(time);

monster11.update(time);

bos.update(time);

p.updateForce(time,window, numberLevel);

}

if (p.rect.intersects(monster.rect))

{

if (monster.getLife()) {

if (p.getFoUp() == true) { vilgelm.play(); monster.dx = 0; p.dy = -0.2; monster.setLife(false); }

if ((p.getFoUp() == false) && (p.dy>0)) { vilgelm.play(); monster.dx = 0; p.dy = -0.2; monster.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(m.rect))

{

if (m.getLife()) {

if (p.getFoUp() == true) { vilgelm.play(); m.dx = 0; p.dy = -0.2; m.setLife(false); }

if (p.dy>0) { vilgelm.play(); m.dx = 0; p.dy = -0.2; m.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster1.rect))

{

if (monster1.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster1.dx = 0; p.dy = -0.5; monster1.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster1.dx = 0; p.dy = -0.2; monster1.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(bos.rect))

{

if (bos.getIsLife()) {

if (p.getFoUp() == true) { vilgelm.play(); bos.dx = 0; p.dy = -0.5; bos.setIsLife(false); }

if ((p.dy>0) && (bos.getIsLife())) { vilgelm.play(); bos.setBossHealth(1); p.dy = -0.2; }

if (bos.getBossHealth() <= 0) { Muzika(12); bos.dx = 0; bos.setIsLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(5); }

}

}

if (p.rect.intersects(monster2.rect))

{

if (monster2.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster2.dx = 0; p.dy = -0.5; monster2.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster2.dx = 0; p.dy = -0.2; monster2.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster3.rect))

{

if (monster3.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster3.dx = 0; p.dy = -0.5; monster3.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster3.dx = 0; p.dy = -0.2; monster3.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster4.rect))

{

if (monster4.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster4.dx = 0; p.dy = -0.5; monster4.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster4.dx = 0; p.dy = -0.2; monster4.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster5.rect))

{

if (monster5.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster5.dx = 0; p.dy = -0.5; monster5.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster5.dx = 0; p.dy = -0.2; monster5.setLife(false);}

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster6.rect))

{

if (monster6.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster6.dx = 0; p.dy = -0.5; monster6.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster6.dx = 0; p.dy = -0.2; monster6.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster7.rect))

{

if (monster7.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster7.dx = 0; p.dy = -0.5; monster7.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster7.dx = 0; p.dy = -0.2; monster7.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster8.rect))

{

if (monster8.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster8.dx = 0; p.dy = -0.5; monster8.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster8.dx = 0; p.dy = -0.2; monster8.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster9.rect))

{

if (monster9.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster9.dx = 0; p.dy = -0.5; monster9.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster9.dx = 0; p.dy = -0.2; monster9.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster10.rect))

{

if (monster10.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster10.dx = 0; p.dy = -0.5; monster10.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster10.dx = 0; p.dy = -0.2; monster10.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(monster11.rect))

{

if (monster11.getLife())

{

if (p.getFoUp() == true) { vilgelm.play(); monster11.dx = 0; p.dy = -0.5; monster11.setLife(false); }

if (p.dy>0) { vilgelm.play(); monster11.dx = 0; p.dy = -0.2; monster11.setLife(false); }

else { if ((p.getIsLife()) && (p.getFoUp() == false)) p.setPlayerHealth(1); }

}

}

if (p.rect.intersects(b.rect))

{

if (b.getLife()) {

b.dx = 0; p.dy = -0.2; b.setLife(false); p.setFoUp(true);

}

}

if (p.rect.left>300) offsetX = p.rect.left - 300;

window.clear(Color(107, 140, 255));

for (int i = 0; i<H; i++) {

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

{

if (TileMap[i][j] == 'B') tile.setTextureRect(IntRect(13, 366, 33, 33));

if (TileMap[i][j] == '0') tile.setTextureRect(IntRect(70, 366, 32, 32));

if (TileMap[i][j] == 'C') tile.setTextureRect(IntRect(26, 1077, 32, 32));

if (TileMap[i][j] == ' ') continue;

tile.setPosition(j \* 32 - offsetX, i \* 32 - offsetY);

window.draw(tile);

}

}

std::ostringstream PlayerHealthString;

PlayerHealthString << p.getPlayerHealth();

texts.setString("Колличество жизней:" + PlayerHealthString.str());

std::ostringstream BossHealthString;

BossHealthString << bos.getBossHealth();

textb.setString("Колличество жизней ПОЛКОВНИКа:" + BossHealthString.str());

textd.setString("Конец игры");

textf.setString("ПОБЕДА");

if (p.getPlayerHealth() <= 0) { p.setIsLife(false); window.draw(textd); }

if (!bos.getIsLife()) { window.draw(textf); }

if (b.getLife() == true) {

window.draw(b.sprite);

}

if (numberLevel == 3) {

window.draw(textb);

}

window.draw(bos.sprite);

window.draw(m.sprite);

window.draw(p.sprite);

window.draw(monster1.sprite);

window.draw(monster2.sprite);

window.draw(monster3.sprite);

window.draw(monster4.sprite);

window.draw(monster5.sprite);

window.draw(monster6.sprite);

window.draw(monster7.sprite);

window.draw(monster8.sprite);

window.draw(monster9.sprite);

window.draw(monster10.sprite);

window.draw(monster11.sprite);

window.draw(monster.sprite);

window.draw(texts);

window.display();

}

}

int main()

{

int numberLevel = 1;

RenderWindow window(sf::VideoMode(800, 600), "kek!");

menu(window,numberLevel);

return 0;

}