Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

Дисциплина: Основы алгоритмизации и программирования (ОАиП)

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

**ПРОГРАММНОЕ ИГРОВОЕ СРЕДСТВО**

**«Assassin's Creed: in the search for keys»**

БГУИР КП 1-40 01 01 134 ПЗ

Студент: гр. 951008 Курбацкий И.Д.

Руководитель: Шулицкий Д.С.

Минск 2020

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc7740842)

[1 Анализ предметной области 6](#_Toc7740843)

[1.1 Обзор аналогов 6](#_Toc7740847)

[1.2 Цели и задачи 8](#_Toc7740847)

[2 Разработка программного средства 10](#_Toc7740849)

[2.1 Структура программы 10](#_Toc7740850)

[2.2 Интерфейс программного средства 10](#_Toc7740851)

[2.3 Отрисовка карт. Звуковые и графические ресурсы 11](#_Toc7740851)

[2.4 Игровая логика 13](#_Toc7740851)

[3 Тестирование программного средства 17](#_Toc7740849)

[4 Руководство пользователя 18](#_Toc7740849)

[4.1 Правила игры 18](#_Toc7740851)

[4.2 Интерфейс программы 18](#_Toc7740851)

[Заключение 23](#_Toc7740849)

[Список использованных источников 24](#_Toc7740849)

[Приложение. Исходный код программы 25](#_Toc7740849)

# ВВЕДЕНИЕ

Ни для кого не секрет, что видеоигры прочно заняли своё место в современной индустрии развлечений. Существуют попытки выделить ком- пьютерные игры как отдельную область искусства, наряду с театром, кино и т.п. Первые примитивные компьютерные и видеоигры были разработаны в 1950-х и 1960-х годах. Они работали на таких платформах, как осциллографы, университетские мейнфреймы и компьютеры EDSAC. Самой первой компью- терной игрой стал симулятор ракеты, созданный в 1942 году Томасом Голд смитом младшим и Истл Рей Менном. Позже, в 1952 году, появилась программа «OXO», имитирующая игру «крестики-нолики», созданная Александром Шафто Дугласом как часть его докторской диссертации в Кембриджском университете. В настоящее время разработка игры – это масштабный процесс, в котором задействована целая команда разработчиков, сложные современные технологии и даже маркетинговые ходы.

Платформер — жанр [компьютерных игр](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%B8%D0%B3%D1%80%D0%B0), в которых основной чертой игрового процесса является прыгание по платформам, лазанье по лестницам, собирание предметов, обычно необходимых для завершения уровня.

Игры подобного жанра характеризуются нереалистичностью, рисованной мультяшной графикой. Героями таких игр обычно бывают мифические существа.

Целью данного курсового проекта является разработка игрового программного обеспечения жанра платформер.

# АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В настоящее время существует огромное количество видов реализации ишр данного жанра. «Super Mario» – один из самых популярных представителей платформеров. Данная серия игр является одной из самых известных во всем мире. Существует множество компьютерных аналогов, немного отличающихся между собой правилами и способами реализации.

* 1. **Обзор аналогов**

## «SuperTux 0.3.0»

Программное средство «SuperTux 0.3.0» – один из примеров реализаций игр данного жанра, созданный SuperTux Development Team, выпущенный 11 апреля 2000 года. В данной программе реализован довольно обширный функционал, позволяющий совершать множество различных действий. Также к плюсам можно отнести постоянные анимации фона. Интерфейс «SuperTux 0.3.0» отображён на рисунке 1.1.



## Рисунок 1.1 – Интерфейс приложения «SuperTux 0.3.0»

* + 1. «Super Mario 1985»

Данное игровое программное средство является классическим пердставителам платформера. В программе реализован классический функционал и пиксельная графика. В данной программе есть множемтво дополнительного функционала, такого как телепорты и смысловая связанность раундов. Из минусов можно отметить качество графики, но это связано с возрастом игры и платформой, для которой она создавалась. Интерфейс «Super Mario 1985» представлен на рисунке 1.2.



Рисунок 1.2 – Интерфейс приложения «Super Mario 1985»

* + 1. «Electro Man»

В данной программе реализован функционал, входящий в основы этого жанра игр: множество платформ и сбор ключей. В этой вариации игры используется более качественная графика, которая улучшает впечатления от игрового процесса. К минусам можно отмести систему реализации собираемых игроком карточек-ключей. Интерфейс «Electro Man» показан на рисунке 1.3.



Рисунок 1.3 – Интерфейс приложения «Elrctro Man»

* 1. **Цели и задачи**
     1. Цель курсовой работы

Целью данной курсовой работы является создание игрового развлекательного приложения «Assassin's Creed: in the search for keys» на основе игры «Elrctro Man», созданной Epic MegaGames в 1991 году.

* + 1. Задачи курсовой работы

В результате сравнения аналогов программного обеспечения и анализа предметной области в данном курсовом проекте поставлены следующие задачи:

* разработать:
  1. алгоритм для выбора возможных позиций для ключей;
  2. алгоритм отрисовки только части видимой карты;
  3. алгоритм системы жизней
* реализовать:
  1. разного рода анимацию;
  2. звуковое сопровождение;
* создать:
  1. игровое поле с ключами;
  2. стартовое меню;

Задача пользователя состоит в том, чтобы пройти все доступные уровни с наименьшим количеством попыток.

* + 1. Необходимые ресурсы для курсовой работы

Для разработки программы необходимо использовать язык программирования С++ с библиотекой SFML. Среда разработки – Microsoft Visual Studio 2019. Использование данной среды разработки дает множество возможностей в создании оконных приложений. С помощью данных библиотеки и среды можно очень легко и быстро разработать качественное приложение, за счет легкости для работы и освоения функционала библиотеки.

Для создания графики были использованы приложения Adobe Photoshop и Adobe Illustrator, которые позволяют с легкостью cоздавать графические файлы разных форматов, а также их редактировать.

1. **РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**
   1. **Структура программы**

В данном приложении необходимо использовать следующие функции и классы:

* класс PLAYER используется для отрисовки и обработки действий, совершаемых игроком ;
* printMap отвечает за отрисовку отображаемой на экране карты;
* gameplay используется обработки всего игрового процесса, путем вызова сторонних функций;
* readMap считывает отрисовываемую выбранныую карту из текстового файла;
* printMenu осуществляет отрисовку игрового меню с выбором дальнейшего действия пользователем;
* checkHp используется для контроля количества жизней персонажа;
  1. **Интерфейс программного средства**

Одним из наиболее важных критериев качества каждой игры является интуитивно понятное меню. Если правила игры можно прочитать, то в начальном экране пользователю придется разбираться самому.

Интерфейс игры оформлен в стандартном стиле для игр из 2000-х. Цветовая палитра приложения реализована с помощью темных цетов: используются для оформления большей части интерфейса, для заливки фона – и белого: основного для всего текста. Он был выбран по причине того, что является самым читабельным на таком цвете фона.

Формирование всех игровых меню реализовано в главном классе в функциях main и printMenu, которые лежат в основе управления всей программой. Главное меню изображенно на рисунке 2.1. Каждый пункт меню вызывает другие функции.

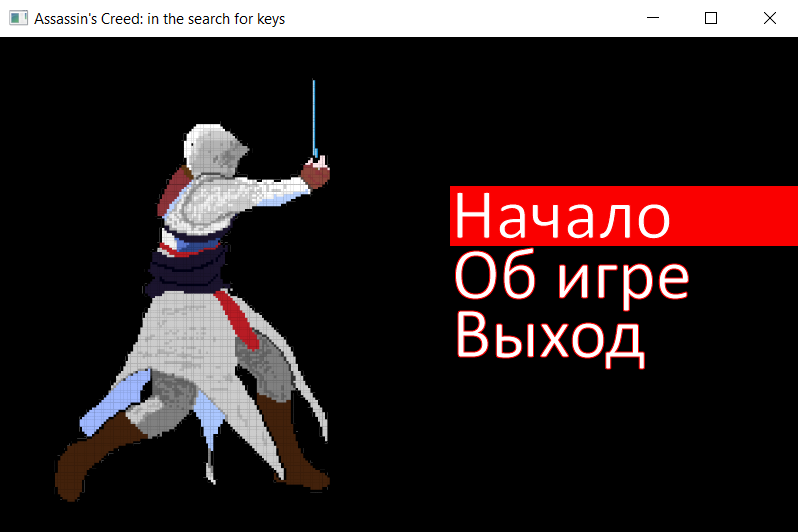


Рисунок 2.1 – Основное меню программы

* 1. **Отрисовка карт. Звуковые и графические ресурсы**

Главными игровыми объектами для отрисовки в данной курсовой работе являются персонаж, карта и множество ключей. За основу для отображения персонажа был взят принцип передвиженгия по текстуре. Каждую секунду происходит сдвиг по текстуре, с которой происходит смена позы персонажа на экране. При прыжке или нахождении в воздухе герой становится в позицию прыжка, по принципу, описанному до этого. Также для выбора стороны, в которую сторону смотрит персонаж, использован принцип отображения текстуры в обратную сторону.

Для решения задачи с анимацией о выборе пункта меню было решено использовать FillTransition, который позволяет изменять заполнение фигуры через равные промежутки времени. FillTransition использован в классе MenuItem, который отвечает за добавление кнопок для пунктов меню. При наведении курсора на пункт меню происходит смена цвета с темно-серого к золотому и обратно. Решение данной задачи приведено ниже:

if (currentFrame > 5) currentFrame -= 5; // обнуление

времени

if (currentFrame > 2) {positionHero(currentFrame, 44);}

else { positionHero(currentFrame, 41); } //изменение

позиции персонажа

if ((!Keyboard::isKeyPressed(Keyboard::Right)) &&

(!Keyboard::isKeyPressed(Keyboard::Left)) &&

(!Keyboard::isKeyPressed(Keyboard::Up))){

//персонаж в позиции без ходьбы

if (isCheckRight) //если персонаж смотрит вправо

{sprite.setTextureRect(IntRect(39 \*

int(currentFrame), 1, 32, 60));}

else {sprite.setTextureRect(IntRect(39 \*

int(currentFrame) + 32, 1, -32, 60));}

}

if (isCheckRight){jump(1);} //вызов обработки прыжка

else {jump(0);}

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

= false;} //персонаж смотрит влево

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

{isCheckRight = true;} // смотрит влево

sprite.setPosition(rect.left - offsetX, rect.top - \

offsetY); //изменение позиции перчонажа на карте

Почти все изображения изначально прогружаются в функции setSprite(): указывается путь к текстурам, после чего текстуры загружаются в спрайти, в которых выбираются части загруженных изображений. Пример реализации этого показан ниже:

pr.loadFromFile("Graphics\\2.png"); //загрузка изображения

sprite0.setTexture(pr); //загрузка текстуры в спрайт

sprite0.setTextureRect(IntRect(0, 1000, 800, 500)); //выбираем

часть текстуы, отображаемую спрайтом

sprite0.setPosition(-offsetX, -offsetY); //установка позиции

спрайта

Для проигрывания фоновой музыки было решено применить класс Music, который производит непосредственно из источника. именён AudioClip, поскольку он достаточно прост в использованиии. Для зацикливания аудиофайлов в программе ввелена проверка на статус воспроизведения, которая в случае остановки воспроизведения, возобновляет его. Пример представлен ниже:

musicMenu.openFromFile("Audio\\menu.wav");//загрузка аудиофайла

musicMenu.setVolume(2.5);//установка громкости воспроизведения

musicMenu.setLoop(true); //зацикливание

if (!musicMenu.getStatus()) { musicMenu.play();} //если мелодия не

проигрывается – то насинается воспроизведение

Для проигрывания звука прыжка было решено применить класс Sound, который изначальномзагружает звуки в память. Он был использован, поскольку требуется чтобы звук воспроизводился без опоздания. Пример показан ниже:

SoundBuffer buffer;//создание буфера

buffer.loadFromFile("Audio\\jump.wav");//загрузка мелодии

Sound sound; //создание звука

sound.setBuffer(buffer);//загрузка буфера в звук

sound.play();//воспроизведение мелодии

* 1. **Игровая логика**

Игровая логика программы реализована в функции gameplay(). Она вызывается при выборе карты, после чего игровой процесс продолжается, пока не будут собраны все ключи или не кончатся жизни. Часть реализации представлена ниже:

while (hp > 0 && score < 3)

{

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

//получение времени с секундомера

clock.restart();//перезапуск секундомера

time = time / 600;//кменьшение времени для анимации

eventWindow();//возможность изменения размера окна

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

{ p.dx = -0.1; }//изменение позиции при нажатии влево

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

{ p.dx = 0.1;}}//изменение позиции при нажатии вправо

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

//обработка прыжка

if (p.onGround) {

x=p.rect.left / 32;y = (p.rect.top / 32) - 1;

if ((TileMap[y][x + p.isCheckRight] != 'B') &&

(TileMap[y][x + p.isCheckRight] != 'b')) {

sound.play();

p.dy = -0.35; p.onGround = false;

}

}

}

p.update(time);//обновление позиции персонажа

if (p.rect.left > 400) offsetX = p.rect.left - 400;

if (p.rect.left > 32 \* W - 400) offsetX = p.rect.left –

400 - (p.rect.left - 32 \* W + 400);

offsetY = p.rect.top - 250;

window.clear(Color::Black);//очистка экрана

window.draw(sprite1); //печать фона

printMap(window, sprite2, rectangle, noKey, keys, p);

//печать карты

printInfo(window);//печать жизней и очков

window.draw(p.sprite);//печать персонажа

window.display();//обновление дисплея

while (Keyboard::isKeyPressed(Keyboard::Escape))

{ hp = 0; }//выход из игры по нажатию клавиши

}

При вызове отрисовываемой карты отрисовывается только та часть карты, которая видная на экране. Код для отрисовки карты показан ниже:

void printMap(RenderWindow& window, Sprite sp1, int& noKey, int&

keys, PLAYER p) {

int startHorisontal = (p.rect.left / 32)-13;

int stopHorisontal = startHorisontal + 2 \* 13;

if (startHorisontal < 0) {startHorisontal = 0;}

if (startHorisontal >= W - 2 \* 13) { startHorisontal = W - 2

\* 13; stopHorisontal = W; }

int startVertical = (p.rect.top / 32)-8;

if (startVertical < 9) { startVertical = 0;}

int stopVeertical = (p.rect.top / 32)+9;

if (stopVeertical > H) { stopVeertical = H; };

for (int i = startVertical; i < stopVeertical; i++) {

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

{

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

else {

sp1.setTextureRect(spriteMap[TileMap[i][j]]);

//текстура для платформы }

sp1.setPosition(j \* 32 - offsetX, i \* 32 –

offsetY);//позиция платформы

window.draw(sp1);//отрисовка платформы

}

}

}

Случайный выбор положения ключа реализован при помощи удаления двух ключей, за счет смены значения ключа на символ, который появляется после сбора ключа. Код реализации этого представлен ниже:

char positionKey(int& keys, int& noKey, char TileMap, int vib) {

if ((keys == 2 && noKey == 0) || (keys == 1 &&

noKey == 1)) { TileMap = '+'; }

else{

if (noKey != 2 && vib == 0) {

TileMap = '+';//замена значения ключа на

символ

noKey++;//изменение значения счетчика убраных

ключей

}

}

keys--;

return TileMap;

}



Рисунок 2.2 – блок-схема выбора позиции ключа

Система жизней реализована таким образом, что определяются изначальная высота с которой начинается прыжок и высота, на которой осуществляется приземление, после чего происходит сравнивание этих двух позиций. Если начальная высота выше конечной большее, чем конечная на 4 платформы, то происходит отнятие очков жизни. Осуществление этого показано ниже:

void checkHp(int i) {

int x, y = (rect.top / 32) + 2;

if (isCheckRight) { x = (rect.left - 29) / 32; }

else { x = (rect.left + 29) / 32; }

if ((TileMap[y][x + i] != 'B') && (TileMap[y][x + i] !=

'b')) //право

{

if (counter == 0) { positionJump1 = rect.top; }

positionJump2 = rect.top;

counter++;

}

else {

if (positionJump1 != 0) {

int n = positionJump2 - positionJump1;

int num = (positionJump2 - positionJump1

+ 10) / 32;

switch (num) {

case 5: {hp -= 20; break; }

case 6: {hp -= 28; break; }

default: {

if (num > 6) {

int i = 5 + rand() % 3;

hp -= num \* i - rand() % 8;

}

break;

}

}

positionJump1 = positionJump2 = counter = 0;

}

}

}

1. **ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА**

В ходе тестирования приложения были выявлены некоторые ошибки и недочеты в работе программного средства.

Была обнаружена ошибка в системе контроля жизней: для её решения было решено изменить точку считывания позициия персонажа, так как при быстрых прыжках между платформами при спуске происходило уменьшение количества очков.

Так же была убрана возможность прыжка, если прямо над персонажем находится платформа. Реализация этого приведена ниже:

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

if (p.onGround) {

x = p.rect.left / 32;

y = (p.rect.top /32) - 1;

if ((TileMap[y][x + p.isCheckRight] != 'B')

&& (TileMap[y][x + p.isCheckRight] !=

'b')) {

sound.play();

p.dy = -0.35; p.onGround = false;

}

}

}

Была найдена ошибка в выборе места появления ключа, после оптимизации производительности программы. Этот недочет был решен путем выбора мест появления ключей на стадии считывания карты из текстового документа.

1. **РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ**

В данном разделе приводится вся необходимая информация, которая поможет пользователю в работе с приложением «Assassin's Creed: in the search for keys».

* 1. **Правила игры**

Суть игры заключается в следующем. Каждый раунд игрок может выбрать одну из доступных карт. После выбора карты пользователь попадает на карту, передвигаясь по которой игрок должен собрать все ключи. Игроку доступно в начале каждого раунда 100 очков hp. В случае падения с высоты, большей чем 4 платформы. Цель игры является сбор трех ключей на выбранной карте. Если у игрока заканчивается 100 hp, то игок проигрывает, после чего попадпет в главное меню. Также игрок может выйти из игры, нажав escape.

* 1. **Интерфейс программы**

На стартовом экране, изображенном на рисунке 4.1, представлено превью программы, после нажатия на которое произойдёт переход в главное игровое меню, которое изображено на рисунке 4.2.

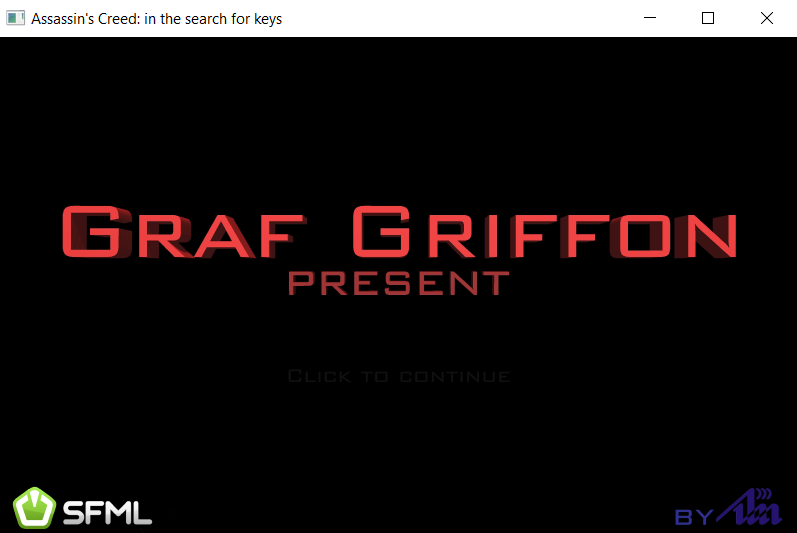


Рисунок 4.1 – Стартовый экран

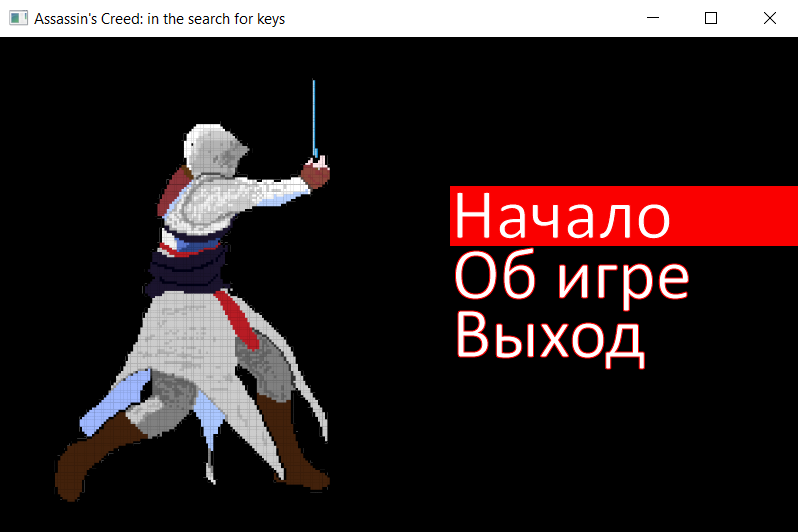


Рисунок 4.2 – Главное игровое меню

При выборе пункта «Начало» и нажатии Enter отображается меню выбора карты, представленное на рисунке 4.3, на котором пользователь аналогично выбирает карту.

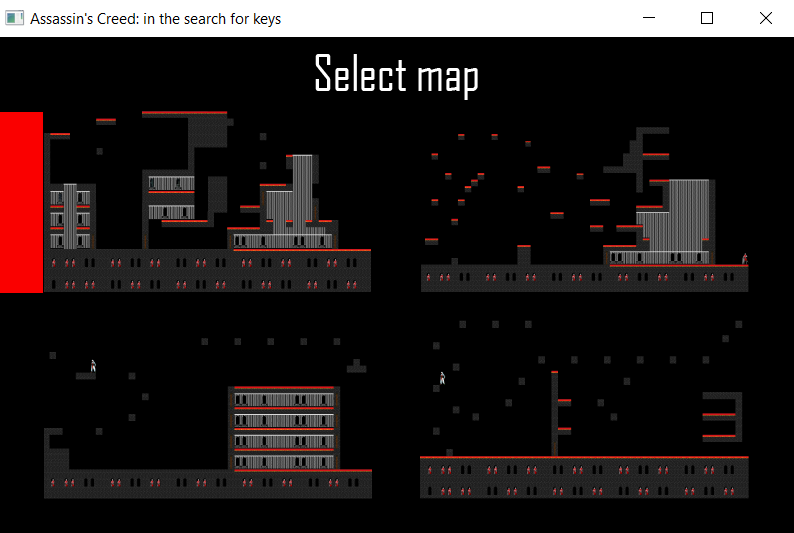


Рисунок 4.3 – Меню выбора карты

После выбора карты пользователь попадпет на игровое поле, на котором может быть дневное или ночное ремя суток, изображенные на рисунках 4.4 и 4.5.



Рисунок 4.4 – дневное игровое поле

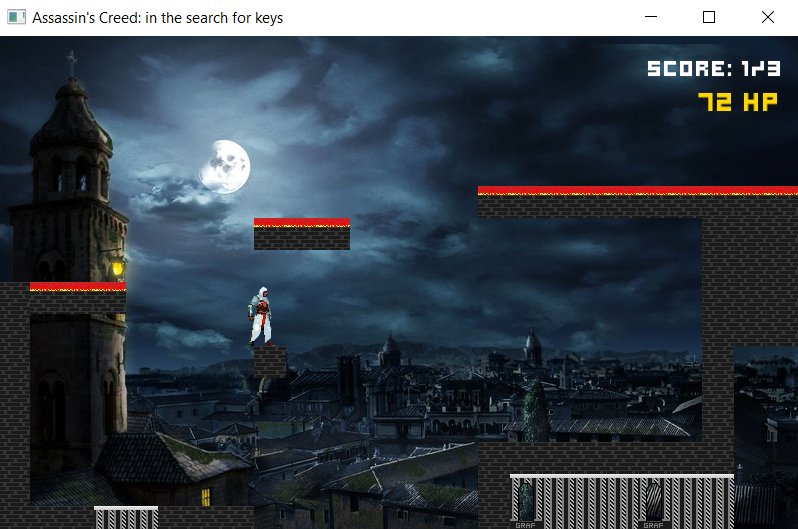


Рисунок 4.5 – ночное игровое поле

По окончанию игры, собрав три ключа, запускается экран с сообщением о победе, который изображен на рисунке 4.6.

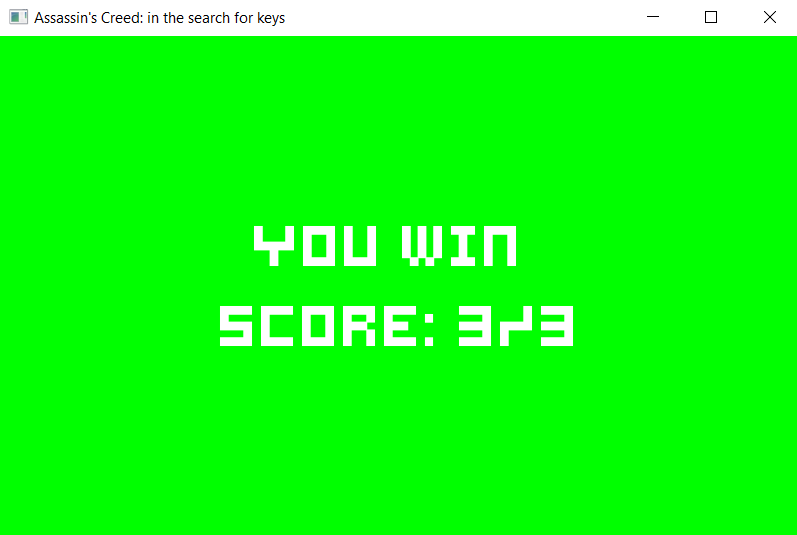


Рисунок 4.6 – Победный экран

В случае нажатия на Escape или утери ста очков hp – игрок попадёт на экран, сообщающий о проигрыше. Пример изображения представлен на рисунке 4.7.

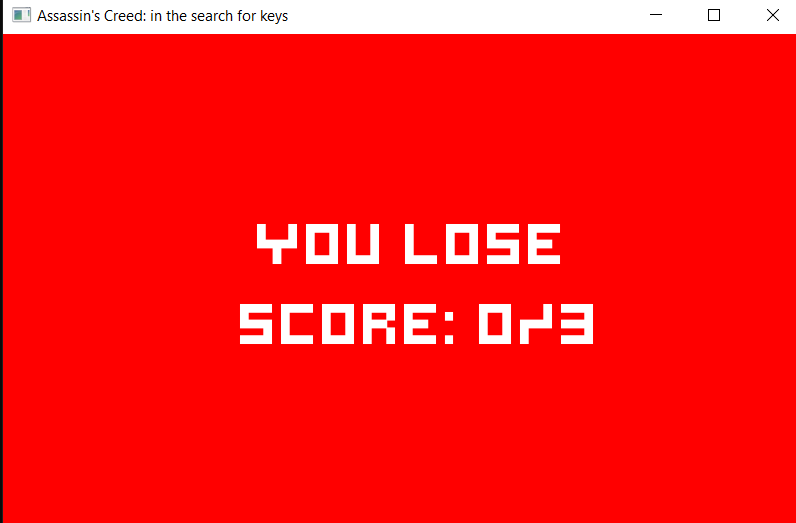


Рисунок 4.7 – Экран, сообщающий о проигрыше

При нажатии пункта «Об игре» в главном меню игрок переходит на экран, изображённый на рисунке 4.8, на котором сообщаются данные об игре, создателе игры и ссылка для обратной связи.

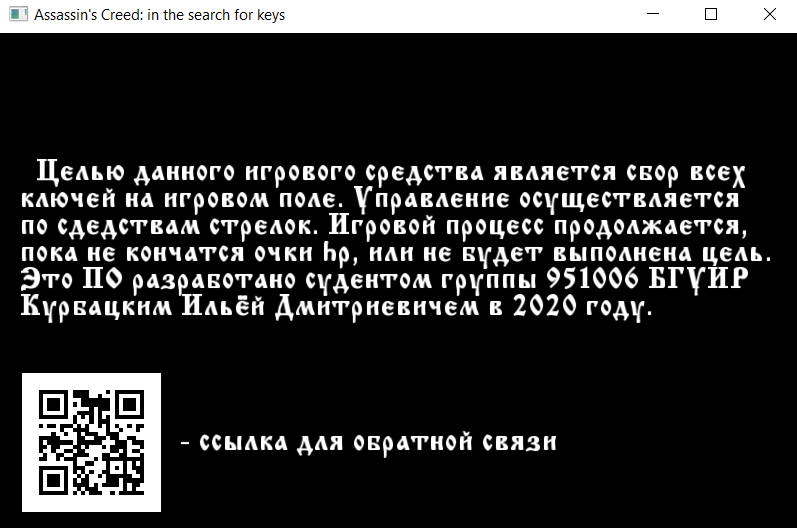


Рисунок 4.9 – Экран, сообщающий игформацию об игре

ЗАКЛЮЧЕНИЕ

Для приятного времяпровождения существует множество способов, один из которых – видеоигры. Одними из самых распростроненных являются платформеры, которых существует огромное количество различных реализаций.

В рамках данного курсового проекта было разработано игровое программное средство «Assassin's Creed: in the search for key». Согласно поставленным задачам, в данном приложении были реализованы следующие функции:

* выбор позиции ключей;
* отрисовка только видимой части карты;
* игровая система жизней;
* всевозможные анимации персонажа;
* звуковое сопровождение;
* построение выбранного игрового поля;
* обработка игрового меню.

Для успешного выполнения всех поставленных целей потребовалось изучить возможности библиотеки SFML, а также освоить некоторые особенности работы с данным расширением.

Существует много возможностей для дальнейшего улучшения при-ложения. Например, можно создать ещё несколько карт, реализовать возможность менять главного персонажа и статистики об игровом процессе.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Artur Moreira. Sfml Game Development. – Birmingham: Packt Publishing, 2013.
2. Скотт Майерс, «Эффективный и современный С++». – Москва: изд. «И.Д.Вильямс», 2019.
3. Учебники для SFML 2.5 [Электронный ресурс]. – Электронные данные. – Режим доступа: https://www.sfml-dev.org/tutorials/2.5/.
4. Страуструп Бьерн, «Программирование. Принципы и практика с использованием C++. – Москва: изд. «И.Д.Вильямс», 2016.
5. Герберт Шилдт, «С++ для начинающих. Шаг за шагом.» – Москва: изд. «ЭКОМ Паблишерз», 2013.
6. Липпман Стенли, «Язык программирования C++. Базовый курс» – Киев: изд. «Диалектика», 2018.

ПРИЛОЖЕНИЕ

Исходный код программы