Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Программирование на языках высокого уровня

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

Игра «OneBitAdventure»

Студент С.Гусаков

Руководитель Е.В.Богдан

МИНСК 2023

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой ЭВМ

*––––––––––––––––––––––––* (подпись)

––––––––––––––––––2023 г. 6

ЗАДАНИЕ

по курсовому проектированию

Студенту *Гусакову Святославу*

1. Тема проекта *Игра OneBitAdventure*
2. Срок сдачи студентом законченного проекта *15 декабря 2023 г.–––*

3. Исходные данные к проекту: картинки png для объектов игры каталоге GameImages, в ее дочерних папках

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке)

1. Введение.

2. Задание.

3. Обзор литературы.

3.1Обзор методов и алгоритмов решения поставленной задачи.

4. Функциональное проектирование.

4.1. Структура входных и выходных данных.

4.2. Разработка диаграммы классов.

4.3.Описание классов.

5. Разработка программных модулей.

5.1. Разработка схем алгоритмов (два наиболее важных метода).

5.2. Разработка алгоритмов (описание алгоритмов по шагам для двух методов).

6. Результаты работы.

7. Заключение

8. Литература

9. Приложения

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма классов. ––––––––––––––––––––––––––––––*

*2. Схема алгоритма MAIN\_GAME\_PROC()*

*3. Схема алгоритма BOSSFIGHT()*

6. Консультант по проекту (с обозначением разделов проекта) Е.В.Богдан

1. Дата выдачи задания 15 *сентября 2023 г.–*
2. Календарный график работы над проектом на весь период проектирования (с обозначением сроков выполнения и трудоемкости отдельных этапов):

*1. Выбор задания. Разработка содержания пояснительной записки. Перечень графического материала – 15 %; ––––––––––––––––––––––––––––\_\_\_\_\_\_\_\_\_* *разделы 2, 3 – 10 %; ––––––––––––––––––––––––––––––––––––– \_\_\_\_\_\_\_\_\_\_*  *разделы 4 к –20 %; –––––––––––––––––––––––––––––––––––––––\_\_\_\_\_\_\_\_\_* *разделы 5 к – 35 %; –––––––––––––––––––––––––––––––––––––\_\_\_\_\_\_\_\_\_\_\_–* *раздел 6,7,8 – 5 %; ––––––––––––––––––––––––––––––––––––––\_\_\_\_\_\_\_\_\_\_–* *раздел 9 к – 5%; ––––––––––––––––––––––––––––––––––––––––– \_\_\_\_\_\_\_\_*  *оформление пояснительной записки и графического материала к 15.12.23 –*

*10 %*

*Защита курсового проекта с 21.12 по 28.12.23г.*

РУКОВОДИТЕЛЬ Е.В.Богдан

#### *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. Гусаков

(дата и подпись студента)

[СОДЕРЖАНИЕ   
ВВЕДЕНИЕ 5](#_Toc44548)

[1 ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ 7](#_Toc44549)

[2. ОБЗОР ЛИТЕРАТУРЫ 9](#_Toc44550)

[2.1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ](#_Toc44551)

[ЗАДАЧИ 10](#_Toc44552)

[3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ 13](#_Toc44553)

[4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ 17](#_Toc44554)

[4.1 Разработка схем алгоритмов 17](#_Toc44555)

[4.2 Разработка алгоритмов 17](#_Toc44556)

[5 РЕЗУЛЬТАТ РАБОТЫ 19](#_Toc44557)

[ЗАКЛЮЧЕНИЕ 23](#_Toc44558)

[СПИСОК ЛИТЕРАТУРЫ 25](#_Toc44559)

[ПРИЛОЖЕНИЕ А 26](#_Toc44560)

[ПРИЛОЖЕНИЕ Б 27](#_Toc44561)

[ПРИЛОЖЕНИЕ В 28](#_Toc44562)

[ПРИЛОЖЕНИЕ Г 29](#_Toc44563)

[ПРИЛОЖЕНИЕ Д 30](#_Toc44564)

**ВВЕДЕНИЕ**

Язык программирования С++ представляет высокоуровневый компилируемый язык программирования общего назначения со статической типизацией, который подходит для создания самых различных приложений. На сегодняшний день он является одним из самых популярных и распространенных языков.

С++ унаследовал от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. В общем С++ - язык широкого пользования, на котором можно создавать практически любые виды программ.

Объектно-ориентированное программирование (ООП) это совокупность понятий (класс, объект, инкапсуляция, полиморфизм, наследование), приемов их использования при проектировании программ, а Си++ - инструмент этой технологии.

Строгое следование технологии ООП предполагает, что любая функция в программе представляет собой метод для объекта некоторого класса. Это не означает, что нужно вводить в программу какие попало классы ради того, чтобы написать необходимые для работы функции. Наоборот, класс должен формироваться в программе естественным образом, как только в ней возникает необходимость описания новых физических предметов или абстрактных понятий (объектов программирования). С другой стороны, каждый новый шаг в разработке алгоритма также должен представлять собой разработку нового класса на основе уже существующих.

Изучение ООП помогло мне посмотреть на разработку программного обеспечения под другим углом. Использование ООП в программировании может стать мощным инструментом в руках опытного разработчика. И на своей практике, при написании курсовой работы, я понял, что использование данной методологии действительно упрощает работу с кодом даже на интуитивном уровне, так как она обеспечивает более высокий уровень абстракции. Например, использование классов и объектов позволяет нам сразу же понять, какие функции и для чего предназначены.

В завершение необходимо сказать, что изучение ООП в наше современное время, несомненно, открывает перед разработчиком множество возможностей для создания инновационных и эффективных решений. ООП становится неотъемлемым инструментом в индустрии программирования в целом, обеспечивая не только технические навыки у разработчиков, но и фундаментальное понимание архитектурных принципов, необходимых для успешного участия в динамичном мире разработки программного обеспечения.

И вот, используя все эти инструменты в своей курсовой работе, я создал фактически свой первый пробный продукт, о котором тоже стоит сказать пару слов.

«OneBitAdventure» – это динамичная, пошаговая игра, где главный герой продвигается как можно дальше, чтобы прокачаться и сражаться с монстрами, а в конце встретится с главным злодеем, с которым будет битва на смерть за справедливость. В зависимости от наших успехов игру можно пройти на две концовки. Это одно из основных отличий моей версии игры от игры-тезки, вышедшей на Андроид. А об остальных деталях разработки будет говориться дальше.

# 1 ЗАДАНИЕ НА КУРСОВУЮ РАБОТУ

Овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня С++ ,овладеть практическими навыками проектирования и разработки законченного, отлаженного и протестированного программного продукта с использованием языка высокого уровня С++[1]. Разработать игру OneBitAdventure с использованием библиотеки Raylib. Вот несколько общих целей, которые могут быть установлены для освоения ООП:

1.Приобретение понимания основных принципов ООП: освоение базовых концепций ООП, такие как инкапсуляция, наследование и полиморфизм, что обеспечит более глубокое понимание организации кода и его взаимодействия.

2.Развитие навыков проектирования классов и Объектов: умение создавать классы и объекты, определять их атрибуты и методы. Это является основой ООП и позволяет структурировать код для более легкого понимания и поддержки.

3.Применение инкапсуляции: использование инкапсуляции для скрытия внутренних деталей реализации классов и предоставления публичного интерфейса. Почему это важно: это способствует безопасности кода и облегчает его сопровождение.

4.Мастерство в обработке наследования: понимать, как использовать наследование для создания иерархии классов и расширения функциональности. Позволяет эффективно использовать и пере использовать код.

5.Овладение исключениями и обработкой ошибок: знанием, как обрабатывать исключения и ошибки в объектно-ориентированных программах. Почему это важно: это повышает устойчивость программы к ошибкам и улучшает ее отказоустойчивость.

Эти цели могут служить отправной точкой для разработки программистом плана обучения и практического применения концепций ООП в реальных проектах.

Raylib – это высокомодульная библиотека с открытым исходным кодом для разработки 2D/3D-игр и простых приложений. Всё содержится в небольшом количестве четко определенных, конкретных и автономных модулей, названных в соответствии с его основными функциями. Некоторые из этих модулей можно использовать даже в автономном режиме.

Преимущества Raylib:

1.Высокая производительность: Raylib использует оптимизированный код и низкоуровневые библиотеки, что позволяет создавать игры с высокой производительностью.

2.Кроссплатформенность: Raylib поддерживает несколько платформ, включая Windows, Linux, macOS, Android и iOS.

3.Бесплатность и открытый исходный код: Raylib является бесплатной и открытой для разработчиков библиотекой, что позволяет им создавать игры без каких-либо ограничений.

4.Обширная документация: Данная библиотека имеет обширную документацию и большое сообщество разработчиков, которые готовы помочь новичкам. На главном сайте библиотеки можно найти шпаргалки, в которых перечислены все основные функции для разработки приложений и небольшие описания к ним. А еще на сайте имеется раздел с примерами программ, где можно изучить подробнее особенности работы и различные возможности этой библиотеки.

5.Поддержка 2D и 3D графики: Данная особенность библиотеки позволяет разработчикам создавать разнообразные игры с многочисленными возможностями. Не смотря на долгий процесс создания, в итоге можно получить довольно неплохой продукт.

6.Разнообразие функций: Как я уже упоминал ранее, в Raylib присутствуют шпаргалки, где кратко описаны функции, используемые для создания игр. Этих функций очень много, и каждая относится к определенному разделу: основные, для работы с формами, текстурами, звуком, текстом и моделями.

# 2 ОБЗОР ЛИТЕРАТУРЫ

Перед началом разработки ознакомимся с принципами игровой механики игры OneBitAdventure. При ознакомлении был использован сайт [10].

OneBitAdventure – это игра в жанре приключения, в которой мы развиваемся по мере прохождения, убиваем враждебных существ, которые мешают нам добраться до главного злодея, а в конце сражаемся с тем самым антагонистом.

Игра начинается с создания игры, где мы должны выбрать персонажа. У каждого из них свои игровые показатели, которые отвечают за изначальный уровень сложности в игре. Выбрав персонажа, мы создаем игровой слот, в который сохраняется время выбора персонажа и нулевые показатели нашей игры. Затем начинается сам игровой процесс. Мы оказываемся в лесу, где нам по пути будут встречаться монстры, которые будут мешать нам добраться до главного антагониста. Чтобы скорее победить босса данной игры нам необходимо собирать ресурсы из сундуков, которые будут улучшать наши показатели жизни и урона, иначе прохождение игры будет довольно трудной задачей.

Реализовать все эти игровые особенности мне помогла библиотека Raylib, документацию которой можно найти в качестве мануала на сайте [11], а так же в качестве примеров и шпаргалок, которые находятся на официальном сайте библиотеки [12].

Мануал на сайте [11] разделен на 4 части:

1. Library Design, где описывается, как проектируется и структурируется raylib, решения, принимаемые в процессе разработки, как все работает внутри.
2. Development Platforms, где скомпилировать и использовать raylib на нескольких платформах разработки.
3. IDE Configurations: специальная информация для настройки raylib в нескольких IDE.
4. Misc Help: различные практические документы по конкретным темам на часто задаваемые вопросы.

## 2.1 Обзор методов и алгоритмов решения поставленной задачи

В основе игры OneBitAdventure лежит объектно-ориентированный подход. Я буду использовать принципы ООП для создания различных игровых объектов, таких как персонажи, враги и предметы. Это позволит мне эффективно организовать код и обеспечить его гибкость и масштабируемость.

Язык программирования C++ был выбран для реализации игры из-за его производительности и богатых возможностей. С помощью C++ и принципы ООП я планирую создать классы и структуры данных, необходимые для работы игры и для более простого и понятного в интуитивном плане кода.

Для создания графического интерфейса игры потребуется использование возможностей библиотеки Raylib. Она предоставляет простой и понятный API для работы с графикой, что значительно упрощает процесс разработки игровых приложений. А множество функций для графической части во многом упростит задачу реализации.

В процессе разработки игры я так же буду использовать различные методы и алгоритмы для обработки нажатий на клавиши и объекты. Например, для обработки столкновений между объектами я буду использовать объекты Rectangle и функцию CheckCollisionPointRec.

# 3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВНИЕ

В данном разделе описываются входные и выходные данные программы, диаграмма классов, а также приводится описание используемых классов и их методов.

##### 3.1 Входные данные

Для работы программы и отображения всех объектов игры были сделаны специальные png картинки с помощью приложения PixelStudio, которые располагаются в каталоге GameImages, в ее дочерних папках.

**3.2 Разработка диаграммы классов**

Диаграмма классов данной работы показана в приложении А.

##### 3.3 Описание классов

###### 3.3.1 Класс button

button – класс, в котором реализуется объекты-кнопки, для более удобной работы с ними.

Описание полей класса:

Texture2D buttonTexture – пременная, хранящая текстуру объекта,

float button\_x, button\_y – пременные, хранящие координаты x и y объекта,

float button\_scale – пременная, хранящая площадь текстуры объекта.

Описание методов:

button(const char\* textureAdress, float x, float y, float scale) {…} – конструктор класса button, в котором мы загружаем базовую информацию в поля объекта;

void show() {…} – рисуем (показываем) объект в окне;

Rectangle rectangleButton() {…} – получаем объект типа Rectangle, который используется для того, чтобы проверять совпадение позиции одного объекта с другим;

~button() {…} – деструктор, в котором мы разгружаем текстуру объекта.

###### 3.3.2 Класс TickButton

TickButton – класс, в котором реализуется объекты-кнопки, но с изменяемой картинкой и запоминанием состояния. Для более удобной работы мы делаем данный класс наследником основного класса button.

Описание полей класса:

int tickButtonStatus – пременная, хранящая состояние объекта TickButton.

Описание методов:

TickButton(const char\* textureAdress, float x, float y, float scale) : button(textureAdress, x, y, scale) {…} – конструктор класса TickButton, в котором мы загружаем базовую информацию в поля объекта;

void changeTickStatus(int tickButtonStatus) {…} – меняем информацию о состоянии объекта;

~TickButton() {…} – деструктор, в котором мы разгружаем текстуру объекта.

###### 3.3.3 Класс SavingSlot

SavingSlot – класс, в котором реализуется слот для сохранения игры.

Описание полей класса:

Texture2D slotTexture – пременная, хранящая текстуру объекта;

float slot\_x, slot\_y – пременные, хранящие координаты x и y объекта;

float button\_scale – пременная, хранящая площадь текстуры объекта;

string slotText – это поле хранящее текст слота;

string filename – поле хранящее имя файла, куда мы сохраняем наши данные;

DataToSave dataToSave – поле хранящее структуру, в который полями являются основными данные для созранения в файл;

int slotStatus – статус слота (пустой, непустой).

Описание методов:

SavingSlot(const char\* textureAdress, float x, float y, float scale, string filename, int slotNum) {…} – конструктор класса SavingSlot, в котором мы загружаем базовую информацию в поля объекта;

void show() {…} – рисуем (показываем) объект в окне;

Rectangle rectangleSavingSlot() {…} – получаем объект типа Rectangle, который используется для того, чтобы проверять совпадение позиции одного объекта с другим;

void clearFile() {…} – очищаем файл;

void loadingGame() {…} – загружаем игру;

void savingGame() {…} – сохраняем игру;

void prepairingSavingFile(int characterType) {…} – подготавливаем файл к сохранению;

~SavingSlot() {…} – деструктор, в котором мы разгружаем текстуру объекта.

###### 3.3.4 Класс Chunk

Chunk – класс, в котором реализуется чанк карты для игры.

Описание полей класса:

Texture2D mainChunkTexture – пременная, хранящая текстуру объекта;

float x, y – пременные, хранящие координаты x и y объекта;

float Texture2D textures[5] – массив, хранящий текстуры объекта;

string slotText – это поле хранящее текст слота;

int textureNum – поле хранящее номер текстуры из массива textures[5].

Описание методов:

void setCoordinates(int i, int j) {…} – загружаем координаты объекта;

void show() {…} – рисуем (показываем) объект в окне;

void changeChunkTexture(int chunkNum) {…} и void changeChunkTexture(Texture2D texture) {…} – перегруженная функция для изменения текстуры объекта;

~Chunk() {…} – деструктор, в котором мы разгружаем текстуры объекта.

###### 3.3.5 Класс Hero

Hero – класс главного героя игры.

Описание полей класса:

Texture2D mainHeroTexture – пременная, хранящая текстуру объекта,

int x = 4, y = 6 – координаты x и y объекта;

Texture2D heroTextures[3] – массив, хранящий обычные текстуры героя;

Texture2D heroDamageTextures[3] – массив, хранящий поврежденные текстуры героя;

int maxHealthPoints – пременная, хранящая максимальное здоровье героя;

int ammountOfDamagePoints – это поле хранящее общее количество урона;

int ammountOfHealthPoints – пременная, хранящая текущее здоровье героя;

Vector2 heroPosition – структура, которая хранит положение героя в окне.

Описание методов:

Hero(int heroType) {…} – конструктор класса в котором мы загружаем базовую информацию в поля объекта в зависимости от выбранного типа героя;

void show() {…} – рисуем (показываем) объект в окне;

void movingUp(), void movingDown(), void movingDown(), void movingLeft() {…} – функции для перемещения персонажа на игровом поле;

void gettingDressed(int extra) {…} – увеличение максимального количества жизней у героя;

void useWeapon (int extra) {…} – увеличение общего количества урона у героя;

void gettingDressed(int extra) {…} – увеличение текущего количества жизней у героя;

bool gettingDammage(int ammOfDamage) {…} – получение урона.

###### 3.3.6 Класс Villain

Villain – класс злодеев игры.

Описание полей класса:

Texture2D mainVillainTexture – пременная, хранящая текстуру злодея;

int v\_x, v\_y – координаты x и y объекта;

Texture2D villainTextures[3] – массив, хранящий текстуры злодеев;

bool visible = false – пременная, хранящая видимость злодея в окне;

bool attack = false – пременная, хранящая готовность злодея к атаке;

int ammountOfDamagePoints – это поле хранящее общее количество урона;

int ammountOfHealthPoints – пременная, хранящая текущее здоровье героя;

Vector2 vPos – структура, которая хранит положение злодея в окне.

Описание методов:

void generatingVillain(int villainType, int steps) {…} – метод, в котором мы загружаем базовую информацию в поля объекта в зависимости от выпавшего типа злодея;

void checkHeroPosition(int h\_x, int h\_y) – сравниваем позиции персонажей(героя и злодея) и устанавливаем режим атаки в зависимости от ответа;

void show() {…} – рисуем (показываем) объект в окне;

void movingDown() {…} – функции для перемещения персонажа на игровом поле;

void gettingDressed(int extra) {…} – увеличение максимального количества жизней у героя;

void useWeapon (int extra) {…} – увеличение общего количества урона у героя;

void startPosition(int i) {…} – установление начальной позиции злодея;

bool gettingDammage(int ammOfDamage) {…} – получение урона.

###### 3.3.7 Класс Boss

Boss – класс главного злодея игры.

Описание полей класса:

Texture2D krackenMainTexture – пременная, хранящая текстуру злодея;

Texture2D textures[4] – массив, хранящий текстуры здля анимации главного злодея;

float scale – пременная, хранящая площадь текстуры объекта;

float scale – пременная, хранящая площадь текстуры эмблемы;

int ammOfDamage – это поле хранящее общее количество урона;

int textureNum – поле хранящее номер текстуры из массива textures[5];

int ammOfHP – пременная, хранящая текущее здоровье героя;

int step – пременная, хранящая специальный показатель;

Vector2 krackenPos – структура, которая хранит положение злодея в окне;

Vector2 emblemPos – структура, которая хранит положение эмблему жизней злодея в окне.

Описание методов:

Boss() {…} – конструктор класса в котором мы загружаем базовую информацию в поля объекта в зависимости от выбранного типа героя;

void show() {…} – рисуем (показываем) объект в окне;

bool gettingDammage(int ammOfDamage) {…} – получение урона;

~Boss() {…} – деструктор, в котором мы разгружаем текстуры объекта.

###### 3.3.8 Класс Food

Food – класс еды.

Описание полей класса:

Texture2D foodTexture – пременная, хранящая текстуру еды;

int ammOfFood – это поле хранящее общее количество еды;

int maxAmmIncHP – пременная, хранящая количество восстанавливаемого здоровья героя;

Vector2 position – структура, которая хранит положение еды в окне;

Описание методов:

Food() {…} – конструктор класса в котором мы загружаем базовую информацию в поля объекта;

void show() {…} – рисуем (показываем) объект в окне;

int takeFood() {…} – возвращает количество восстанавливаемого здоровья и уменьшает ammOfFood;

void chestCheck(int h\_x, int h\_y) {…} – рандомно генерирует количество подбираемой еды и дополнительное значение maxAmmIncHP;

~Food() {…} – деструктор, в котором мы разгружаем текстуру объекта.

###### 3.3.9 Класс Weapons

Weapons – класс оружия.

Описание полей класса:

Texture2D weaponsTexture– пременная, хранящая текстуру оружия;

int ammOfExtraDamage – пременная, хранящая количество дополнительных баллов для урона героя;

Vector2 position – структура, которая хранит положение оружия в окне;

Описание методов:

Weapons (int exAmmOfExtraDamage = 0) {…} – конструктор класса в котором мы загружаем базовую информацию в поля объекта;

void show() {…} – рисуем (показываем) объект в окне;

void chestCheck(int h\_x, int h\_y) {…} – рандомно генерирует показатели подбираемого оружия;

~Weapons () {…} – деструктор, в котором мы разгружаем текстуру объекта.

###### 3.3.10 Класс Armory

Armory – класс броня.

Описание полей класса:

Texture2D weaponsTexture– пременная, хранящая текстуру брони;

int incMaxHP – пременная, хранящая количество дополнительных баллов для максимального здоровья героя;

Vector2 position – структура, которая хранит положение брони в окне;

Описание методов:

Armory (int exIncMaxHp = 0) {…} – конструктор класса в котором мы загружаем базовую информацию в поля объекта;

void show() {…} – рисуем (показываем) объект в окне;

void chestCheck(int h\_x, int h\_y) {…} – рандомно генерирует показатели подбираемой брони;

~Armory () {…} – деструктор, в котором мы разгружаем текстуру объекта.

# 4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

## 4.1 Разработка схем алгоритмов

Метод generatingMap(Chunk\*\* chunks) генерирует текстуру для каждого чанка игры. Схема метода generatingMap(Chunk\*\* chunks)показана в приложении Б.

Метод show() класса Boss обрабатывает отображение главного злодея на карте. Схема метода show() показана в приложении В.

## 4.2 Разработка алгоритмов

Функция CreatingGame(int WindowWidth, SavingSlot \* slot) обрабатывает процесс создания игры.

Шаг 1. Загружаем требуемые текстуры и информацию для фона;

Шаг 2. Загружаем дисплей и инициализируем координаты дисплея;

Шаг 3. Создаем игровую кнопку;

Шаг 4. Загружаем картинку с показателями персонажей;

Шаг 5. Инициализируем кнопки для персонажей;

Шаг 6. Загружаем текстуру ошибки;

Шаг 7. Инициализируем переменную, отвечающую за ошибку;

Цикл while (пока выполняется условие !WindowShouldClose):

Шаг 8. Начинаем рисование;

Шаг 9. Очищаем задний фон;

Шаг 10. Отображаем фон меню;

Если error > 0:

Шаг 11. Рисуем дисплей с ошибкой;

Иначе:

Шаг 12. Рисуем обычный дисплей;

Шаг 13. Показываем остальные объекты меню;

Шаг 14. Инициализируем “переменные-прямоугольники”, которые будут помогать нам проверять касание мышки и объектов меню;

Если левая кнопка мыши нажата:

Шаг 15. Определяем координаты указателя мыши;

Если координаты мыши совпадают с “переменной-прямоугольником” кнопки рыцаря, то:

Шаг 16. Меняем статус рыцаря на избранный, а статусы других персонажей обнуляем;

Если координаты мыши совпадают с “переменной-прямоугольником” кнопки эльфа:

Шаг 17. Меняем статус эльфа на избранный, а статусы других персонажей обнуляем;

Если координаты мыши совпадают с “переменной-прямоугольником” кнопки гнома, то:

Шаг 18. Меняем статус гнома на избранный, а статусы других персонажей обнуляем;

Если координаты мыши совпадают с “переменной-прямоугольником” кнопки начинающей игру:

Если статус рыцаря – избранный, то:

Шаг 19. Статус выбранного слота равен 1;

Шаг 20. Выгружаем все текстуры данной функции и заканчиваем рисование;

Шаг 22. Возвращаемое функцией значение равно полученному из функции MAIN\_GAME\_PROCESS(WindowWidth, slot);

Если статус эльфа – избранный, то:

Шаг 23. Статус выбранного слота равен 2;

Шаг 24. Выгружаем все текстуры данной функции и заканчиваем рисование;

Шаг 25. Возвращаемое функцией значение равно полученному из функции MAIN\_GAME\_PROCESS(WindowWidth, slot);

Если статус гнома – избранный, то:

Шаг 26. Статус выбранного слота равен 3;

Шаг 27. Выгружаем все текстуры данной функции и заканчиваем рисование;

Шаг 28. Возвращаемое функцией значение равно полученному из функции MAIN\_GAME\_PROCESS(WindowWidth, slot);

Иначе:

Шаг 29. Переменная error становится равна нулю;

Шаг 30. Конец рисования;

Конец цикла while.

Шаг 31. Возвращаем false.

Функция main()обрабатывает начало игры.

Шаг 1. Создаем переменные: bool exitStatusCheck , хранящая статус выхода из функции; WindowWidth и WindowHeight, которые показывают ширину и высоту окна соответственно;

Шаг 2. Инициализируем окно программы;

Шаг 3. Устанавливаем количество кадров в секунду;

Шаг 4. Загружаем требуемые текстуры и информацию для фона;

Шаг 5. Загружаем текстуру названия игры;

Шаг 6. Создаем кнопку начала игры;

Шаг 7. Создаем кнопку выхода;

Цикл while (пока выполняется условие !WindowShouldClose):

Шаг 8. Начинаем рисование;

Шаг 9. Очищаем задний фон;

Шаг 10. Отображаем все объекты меню;

Шаг 12. Инициализируем “переменные-прямоугольники”, которые будут помогать нам проверять касание мышки и объектов меню;

Если левая кнопка мыши нажата:

Шаг 14. Определяем координаты указателя мыши;

Если координаты мыши совпадают с “переменной-прямоугольником” кнопки выхода, то:

Шаг 15. Прерываем цикл while;

Если координаты мыши совпадают с “переменной-прямоугольником” кнопки начала игры:

Шаг 16. Статус выхода exitStatusCheck равен полученному значению из функции LoadAndCreateGameMenu(WindowWidth);

Если exitStatusCheck равно true:

Шаг 17. Прерываем цикл while;

Шаг 18. Конец рисования;

Конец цикла while.

Шаг 19. Выгружаем все текстуры данной функции;

Шаг 20. Возвращаем 0.

# 5 РЕЗУЛЬТАТ РАБОТЫ

На рисунке 5.1 изображена начало работы программы. При старте программы запускается главное меню, в котором у нас есть фон, название и две кнопки: StartGame и Exit.

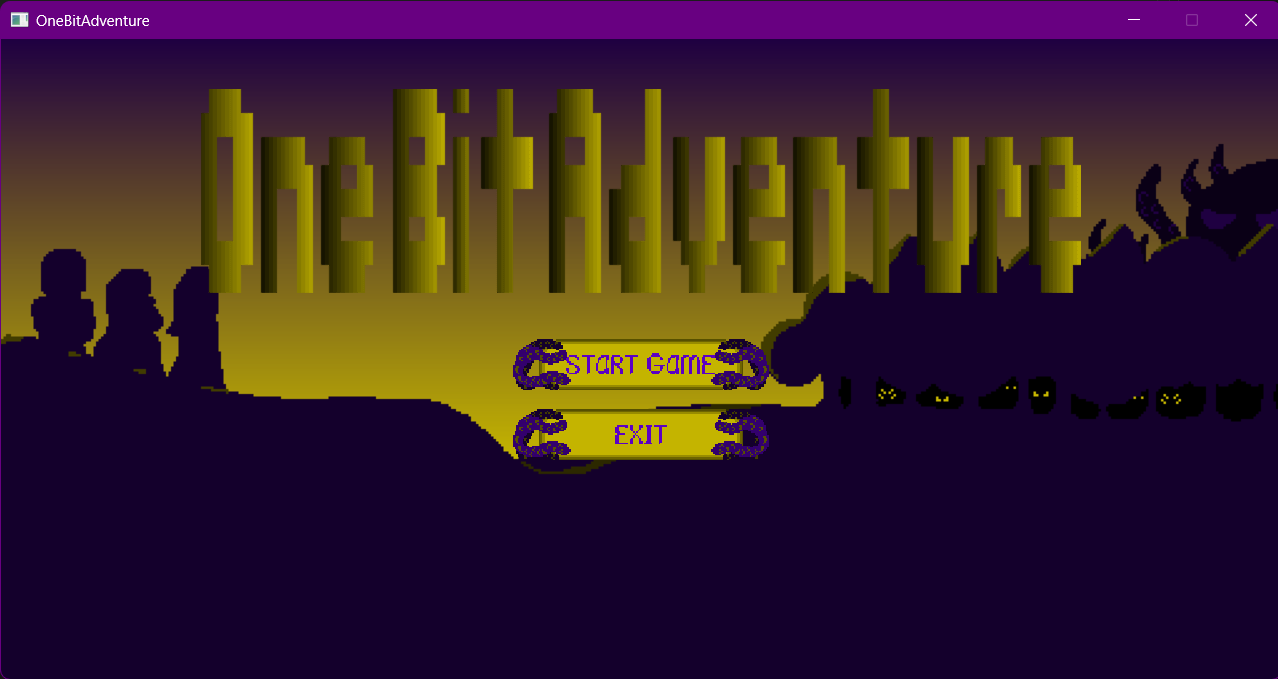


Рисунок 5.1 – Главное меню

На рисунке 5.2 изображено меню с игровыми слотами. Оно нам будет показано при нажатии на кнопку StartGame. Слоты будут хранить данные сохраненных игр. Сбоку будут находиться кнопки Создание игры (сверху) и Выход в главное меню (снизу).

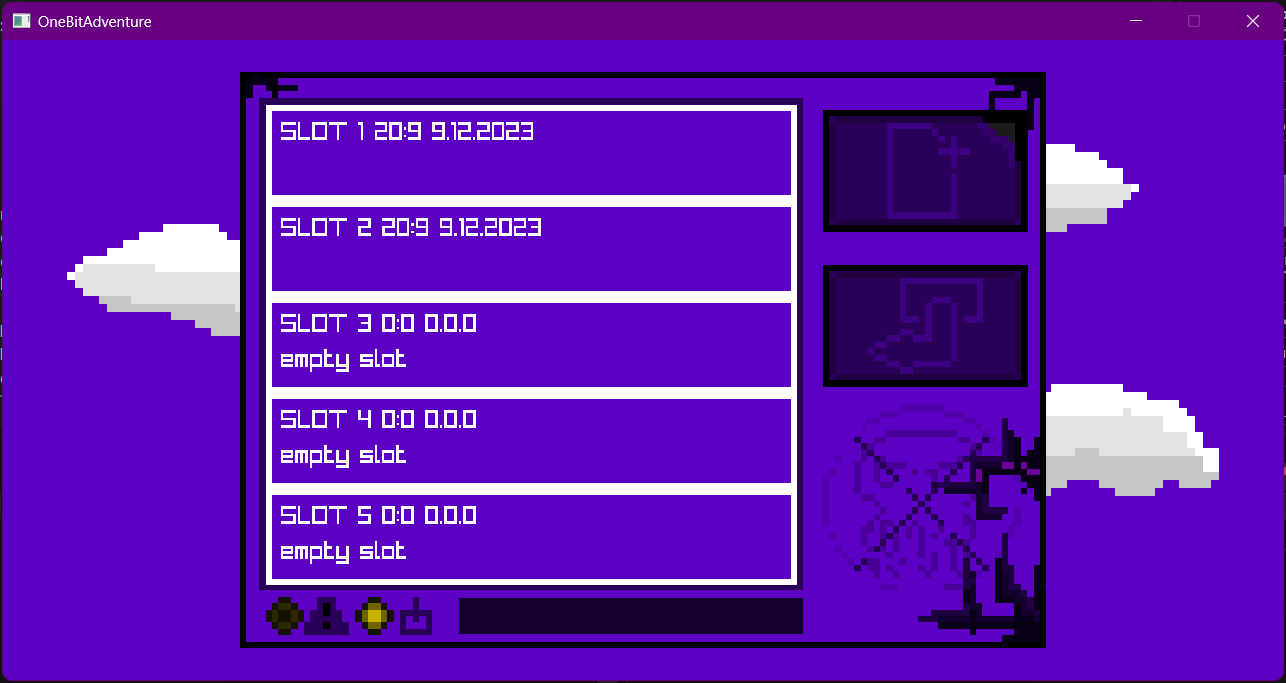


Рисунок 5.2 – Меню игровых слотов

На рисунке 5.3 отражено меню выбора персонажа при создании новой игры, то есть оно будет показано при нажатии на кнопку Создание игры в меню слотов. Выбрать персонажа можно нажав на соответствущую кнопку снизу. После выбора персонажа нажимаем кнопку Начало игры.



Рисунок 5.3 – Окно конца раунда

На рисунке 5.4 показан основной игровой процесс. На заднем фоне показана сгенерированная матрица чанков игры. Посередине находится выбранны й нами персонаж. По бокам видны элементы интерфейса: показатели еды, оружия, брони находятся слева от матрицы; кнопка сохранения находится в правом верхнем углу; количество HealthPoints пероснажа находиться в нижнем правом углу. При нажатии клавиши W, если персонаж стоит в середине матрицы чанков, верхняя линия чанков будет обновляться генерируя при этом сундуки и враждебных существ, показанных на рисунке. Сверху будет отображаться количество шагов, которые мы прошли, и, в зависимости от их количества, некоторые показатели в игре будут усложняться.



Рисунок 5.4 – Главный процесс игры

На следующем рисунке 5.5 можно увидеть одно из главных действий – битву с боссом. Данная сцена отображается при прохождении 1000 шагов. При выполнении каких-либо действий, на матрице новых чанков будут появляться красные клетки, которые, в зависимости от степени их прозрачности, будут показывать приближение атаки босса. Если игрок не успевает отбежать от опасной клетки, то босс наносит урон в 100 жизненных очков.

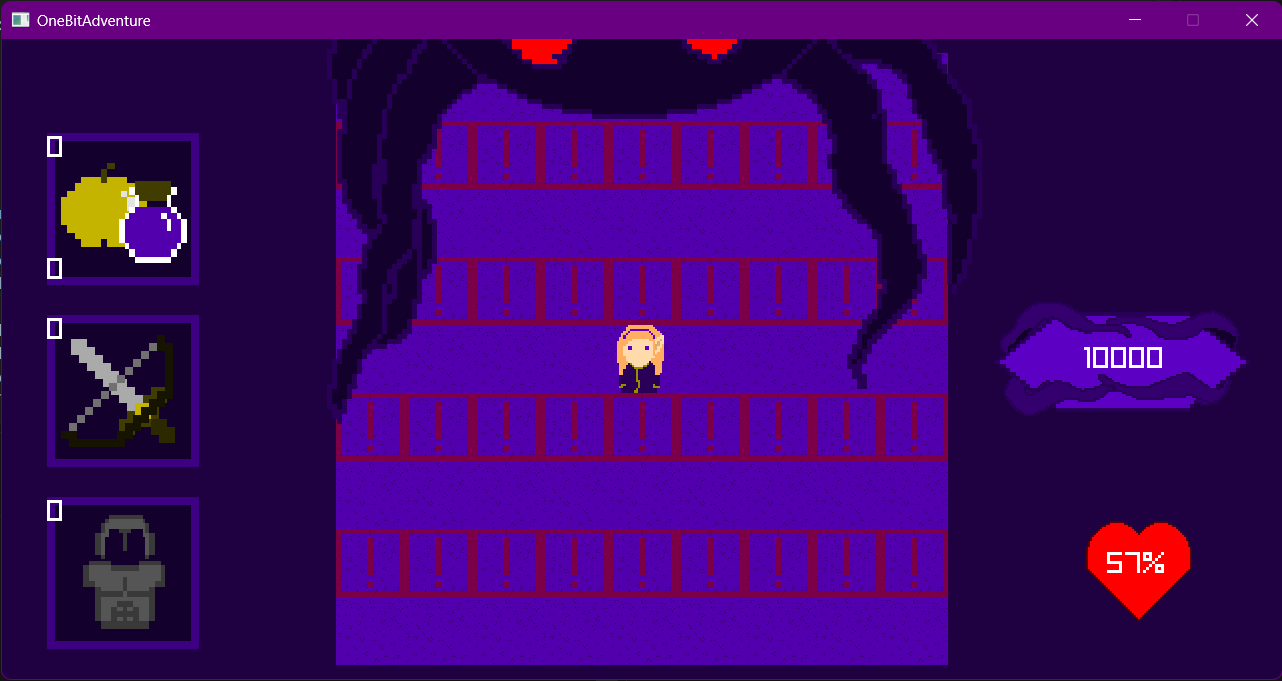


Рисунок 5.5 – Битва с боссом

Далее, в зависимости от ваших успехов будут отображаться различные концовки. Если персонаж умирает в главном процессе игры, то на экране будет показан рисунок 5.6. Однако, если перед этим вы сохранили игру, у вас будет возможность начать с сохраненного места.



Рисунок 5.6 – Сцена Game Over

Если вы одерживаете победу над боссом, то на экране будет показан рисунок 5.7. Спустя 5 секунд вы будете перемещены в главное меню, а слот данной игры будет очищен.



Рисунок 5.7 – Сцена YouWin

Если вы умираете в битве с боссом, то на экране будет показан рисунок 5.8. Спустя 5 секунд вы будете перемещены в главное меню, и слот данной игры так же будет очищен, как и при прохождении на хорошую концовку.



Рисунок 5.6 – Сцена YouLose

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы по теме ООП, C++ и Raylib, я получил ценный опыт и углубил свои знания в области объектно-ориентированного программирования. Изучение основных принципов ООП, таких как инкапсуляция, наследование и полиморфизм, позволило мне эффективно организовать код, что упрощает его понимание и увеличивает эффективность.

Язык программирования C++ оказался мощным инструментом для реализации сложных алгоритмов и структур данных. Он предоставляет гибкость и контроль над системными ресурсами, что является важным преимуществом для разработки высокопроизводительных приложений.

Библиотека Raylib позволила мне создавать интерактивные графические интерфейсы с простым и понятным API. Это значительно упрощает процесс разработки графических приложений и позволяет сосредоточиться на основной логике программы.

В целом, реализация игры OneBitAdventure была полезным опытом разработки в ООП. Это позволило мне применить и углубить свои знания в применении языка C++, дало мне навыки в использовании библиотеки Raylib, а также получить практический опыт разработки игровых приложений. Этот опыт будет полезен мне в дальнейшем обучении и профессиональной деятельности.

# СПИСОК ЛИТЕРАТУРЫ

1. "Объектно-ориентированное программирование на С++" Бьярн Страуструп
2. "Язык программирования С++" Герберт Шилдт
3. Руководство по языку программирования C++[Электронный ресурс]. - Электронные данные. - Режим доступа: https://metanit.com/cpp/tutorial/ Дата доступа: 25.11.2023
4. "Алгоритмы. Построение и анализ" Кормен, Лейзерсон, Ривест, Штайн
5. "Введение в алгоритмы" Кормен, Лейзерсон, Ривест, Штайн
6. "Алгоритмы на C++" Роберт Седжвик, Кевин Уэйн
7. C++: эффективное программирование. 55 способов улучшения структуры программ и стиля кода" Scott Meyers
8. "Алгоритмы. Построение и анализ" Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein
9. "Структуры данных и алгоритмы в C++" Robert Lafore
10. One Bit Adventure Wiki [Электронный ресурс]. - Электронные данные. - Режим доступа: https://onebitadventure.fandom.com/wiki/OneBit\_Adventure\_Wiki - [Дата](https://www.gipsyteam.ru/poker/pravila-blekdzheka%20-%20Дата) доступа: 25.11.2023
11. Raylib Wiki Homepage [Электронный ресурс]. – Электронные данные. – Режим доступа: https://github.com/raysan5/raylib/wiki -

# ПРИЛОЖЕНИЕ А

*(обязательное)*

Диаграмма классов

# ПРИЛОЖЕНИЕ Б

##### *(обязательное)*

Схема метода generatingMap(Chunk\*\* chunks)

# ПРИЛОЖЕНИЕ В

##### *(обязательное)*

Схема метода show() класса Boss

# ПРИЛОЖЕНИЕ Г

*(обязательное)*

Код программы

# ПРИЛОЖЕНИЕ Д

*(обязательное)*

Ведомость документов