|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
| Институт искусственного интеллекта | | |
| Кафедра программного обеспечения систем радиоэлектронной аппаратуры | | |

КУРСОВАЯ РАБОТА

по дисциплине «Методы и стандарты программирования»

на тему: «Создание компьютерной игры на языке программирования C++»

|  |  |  |  |
| --- | --- | --- | --- |
| Обучающийся | \_\_\_\_\_\_\_\_\_\_\_\_\_ | \_ Забора Георгий Данилович | |
|  | *Подпись* | *Фамилия Имя Отчество* | |
| Шифр |  | 22К0386 |  |
| Группа |  | КМБО-02-22 |  |
|  |  |  |  |
| Руководитель  работы | \_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*\_\_\_*\_\_\_\_\_\_\_*\_\_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_* | |
|  | *Подпись* | *Фамилия Имя Отчество* | |

Москва 2023s

[ВВЕДЕНИЕ 3](#_Toc176643165)

[ПОСТАНОВКА ЗАДАЧИ 3](#_Toc855821054)

[ОТЧЕТ О РАЗРАБОТКЕ 3](#_Toc823232836)

[Архитектура программы 3](#_Toc2127998628)

[Описание классов 3](#_Toc1074450785)

[Диаграмма наследования 3](#_Toc1870802884)

[Используемые алгоритмы 3](#_Toc1969753299)

[РУКОВОДСТВО ПО СБОРКЕ И ЗАПУСКУ 3](#_Toc161645544)

[РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 3](#_Toc198954804)

[Базовое управление и навигация в меню игры 3](#_Toc1003561157)

[Цель игры 3](#_Toc1208377586)

[ЗАКЛЮЧЕНИЕ 3](#_Toc548655120)

ВВЕДЕНИЕ

Целью представимой курсовой работы является создание оригинальной компьютерный игры под названием “Project Pentagram”. Во время игрового процесса игрок управляет персонажем, который отображается на экране как 2d проекция вида сверху. Персонаж может передвигаться по карте с помощью ввода с клавиатуры, и атаковать с помощью мыши. Во время прохождения уровней игры игроку, в качестве основной угрозы, будут встречаться различные враги, а также некоторые ловушки и головоломки, исход которых будет влиять на прогресс игрока. Целью игры является прохождение всех доступных уровней, хотя разнообразие содержания игры предлагает большую вариативность для импровизированных испытаний. К примеру, игрок может попробовать пройти все уровни на тяжелой сложности, не используя стандартную атаку персонажа. Основным жанром игры является “Hack & Slash”.

Для создания проекта автору потребовалось изучить большое количество материалов по дисциплине разработки игр, охватывая следующие темы: основы графического дизайна, основы звукового дизайна, дизайн уровней, современный стандарт C++17. Более широкий список представляется читателю ниже.

Настоящая пояснительная записка содержит в себе базовое устройство архитектуры разработанной игры, методы, использованные при создании проекта, отчет о разработке, руководство сборки, а также руководство пользователя.

ПОСТАНОВКА ЗАДАЧИ

Необходимо реализовать код на языке программирования C++, результатом сборки которого будет компьютерная игра, полностью отвечающая всем пунктам технического задания, указанного в данном файле.

Передвижение по уровню осуществляется с помощью ввода с клавиатуры (W, A, S, D), также доступны 2 различных варианта атаки игроком противников.

Уровни игры относительно схожи друг с другом и представляют из себя 2d поверхность (карту), по которой может передвигаться игровой персонаж. Во время старта каждого уровня в специально отведенных местах карты появляются враги, которые будут атаковать игрового персонажа, как только он окажется в радиусе видимости. В игре также присутствуют механики лечения посредством подбирания появляющихся аптечек, и опасность во время прохождения заранее заданных ловушек.

Если здоровье игрового персонажа дойдет до нуля, прогресс уровня будет потерян, и игроку придется начинать его заново.

Всего имплементировано 3 основных заранее заданных уровня. Присутствует выбор уровня сложности из легкого, нормального и сложного, который влияет на параметры появляющихся врагов и их атак, как ближних, так и дальних.

В игре реализован следующий пользовательский интерфейс:

* Главное меню, где игрок может выбрать любой из уже открытых уровней и сложность. В нем реализованы соответствующе реагирующие на нажатие кнопки:
  + Играть
  + Настройки
  + Выход
* Меню уровней, которое открывается по нажатию кнопки “Играть”
* Меню сложности, которое открывается по нажатию кнопки “Сложность”
* Графическое отображение здоровья игрока, его абсолютной позиции на карте и прочность его щита.

ОТЧЕТ О РАЗРАБОТКЕ

## Архитектура программы

Весь код приложения написан на языке C++ (стандарт C++17).

В качестве движка игры была выбрана библиотека на языке C с открытым исходным кодом - Raylib. Для удобства разработки и соблюдения принципов ООП были написаны оберточные классы и методы исходных структур и функций движка.

За создание исходного окна игры отвечает объект класса Game. Класс Game реализует всю внутреннюю логику программы, включая приватные методы создания и загрузки уровня, закрытые поля, хранящие в себя прогресс игрока, объекты классов графического и звукового обработчиков, а также публичный метод run(), который запускает игру.

Классы GraphicsHandler и AudioHandler являются первичной границей приложения с его движком. Согласно архитектуре, все внутренние функции Raylib должны вызываться только внутри методов данных классов.

## Описание классов

Экземпляр класса GraphicsHandler отвечает за отображение всех моделей и анимаций внутриигровых объектов. Ключевыми полями класса являются objectList и objectMap - два контейнера stl, которые хранят в себе указатели на объекты, которые необходимо отобразить. ObjectMap предоставляет доступ к указателю по ключу (long long int = object.id), в то время как objectList является связанным списком этих указателей. Каждую итерацию основного цикла окна игры происходит последовательное отображение объектов по всем указателям из objectList.

Экземпляр класса AudioHandler отвечает за звуковой вывод программы. Он хранит указатели на поля GraphicsHandler objectList и objectMap, и обладает собственным полем soundListPair (std::pair), где soundListPair.first - список всех аудио файлов, которые необходимо начать проигрывать на итерации, а soundListPair.second - список всех, которые необходимо остановить.

Все классы объектов игры представляют из себя структуры с общедоступными полями. Базовой структурой архитектуры является GameObject. От GameOjbect наследуются структуры Entity и Projectile.

В структуре GameObject хранится базовая информация об объекте: его текстура, уникальный ID, индикатор коллизии, а также указатель на его базовую анимацию.

Структура Entity используется для описания всех объектов в игре, способных на динамическое взаимодействие с игроком, и обладающих нефиксированным сроком существования (в отличии от Projectile). Entity обладает полями урона, здоровья, а также указателями на вспомогательные анимации атаки и движения.

Projectile отвечает за все снаряды, запущенные дальними атаками Entity. Структура обладает вектором направления, целью, анимацией попадания и уроном.

Также реализованы вспомогательные пространства имен, одно из которых отвечает за отображение внутриигрового меню.

## Диаграмма наследования

Рисунок 1 - Схема наследования объектов

## Используемые алгоритмы

В процессе разработки были написаны шаблонные функции циклического сдвига контейнеров stl (vector, list).

Реализовано пространство имен ObjectHandler. В нем имплементированы функции нахождения центра объекта и быстрые проверки на коллизию. Также в это пространство имен входят функции векторной нормировки, векторной разности и измерения длины.

Для разрешения проблем с передвижением персонажа по карте, был реализован класс абелевой группы. В частности, в секции кода, отвечающего за ввод направления движения, используется следующая мультипликативная группа:

# РУКОВОДСТВО ПО СБОРКЕ И ЗАПУСКУ

Список минимальных требований для сборки и запуска программы:

* Наличие утилиты CMake версии не ниже 3.11
* Наличие утилиты GitBash, или любого другого эмулятора терминала Linux
* Наличие mingw g++ версии не ниже 13.1.0

В директории файлов игры необходимо создать папку “build”, в которой будет происходить сборка.

Выполнить следующие две строки в терминале bash:

* cmake .. -G "MinGW Makefiles"
* cmake --build .

После их выполнения, в директории build появится исполняемый файл, после запуска которого откроется окно игры.

# РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

## Базовое управление и навигация в меню игры

Чтобы начать играть, игрок должен нажать Play, затем Levels, и выбрать уровень из списка доступных.

При желании игрок может поменять сложность уровня во вкладке Play, Difficulty.

Передвижение персонажа по карте осуществляется с помощью клавиш (W, A, S, D).

Персонаж атакует при нажатии левой кнопки мыши.

## Цель игры

Цель игры - пройти все доступные уровни. Прохождение уровня означает уничтожение всех врагов на карте. Выбор сложности и вариативность тактик боя с врагами предоставляют игроку дополнительные испытания, также игрок может сам придумать себе испытание на уровне.

# ЗАКЛЮЧЕНИЕ

Курсовая работа представляет из себя готовый продукт в виде компьютерной игры, которая отвечает требованиям технического задания. Во время написания проекта автор приобрел множество полезных знаний в области разработки видеоигр, а также лучшее понимание архитектур кода больших проектов в целом. Многие вещи из университетской программы первого и второго курсов были использованы в реализации проекта. Из математики автору оказалась незаменима аналитическая геометрия, а также пригодились основы общей алгебры. Большинство пройденных в университете тем по предмету программирования были использованы в коде в той или иной форме, а знания автора в области разработки игр и ООП значительно продвинулись вперед.