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 Endless Survival»

Обучающийся _____ *Мяктинов Никита Алексеевич*
Подпись

Шифр 23К0049

Группа КМБО-02-23

Руководитель
работы _____ *Черноусов Игорь Дмитриевич*
Подпись

Москва 2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Данный документ представляет собой пояснительную записку к курсовой работе по предмету “Методы и стандарты программирования”. Темой курсовой работы является разработка игры “Endless Survivors”, представляющей собой rogue-like – игру, в которой пользователю предстоит сразить множество врагов, улучшить оружие и найти звезду Эдема для победы.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Техническое задание

Первоначально необходимо реализовать загрузку текстур для игровых объектов (трава, деревья, персонажи: игрок и враги) и шрифтов для интерфейса. Эти ресурсы должны быть загружены до начала игрового процесса и готовы для использования в ходе игры.

Игровой процесс происходит на карте, которая будет автоматически генерироваться по мере движения игрока. Карта должна состоять из различных биомов (лес, пустыня и т. д.) с уникальными объектами, чтобы каждая игровая сессия отличалась.

Игрок должен получать опыт за действия в игре. При достижении определенного количества опыта повышается уровень. Необходимо предоставить возможность улучшать характеристики персонажа (здоровье, скорость передвижения, урон) после каждого повышения уровня.

На карте должны быть разбросаны предметы, которые игрок может подбирать. Эти предметы должны давать временные или постоянные бонусы, такие как увеличение скорости, здоровья или урона. Для этого необходимо реализовать систему коллизий персонажа с объектами.

В игре должно быть три типа врагов:

- обычные зомби — базовые враги с низким уровнем сложности;
- танковые зомби — медленные, но выносливые враги, требующие большего урона для уничтожения;
- прыгающие зомби — враги, способные перемещаться на большие расстояния прыжками.

Враги должны появляться волнами, с каждой новой волной увеличивается их сложность и количество.

В игре будет реализован пользовательский интерфейс, который отображает:

- уровень здоровья персонажа;
- накопленный опыт.

Во время игры игрок может найти уникальный игровой объект под названием "Звезда Эдема". Этот предмет должен быть интегрирован в игровую механику как завершающий компонент.

1.2 Требования к функциональным характеристикам программы

Программный продукт должен включать следующие возможности:

1. Игрок имеет возможность исследовать открытый мир, взаимодействовать с различными объектами и сражаться с врагами.

2. Система здоровья и опыта, отображаемую на экране. Полоска здоровья уменьшается при получении урона и восстанавливаться со временем.

3. Враги обладают уникальными способностями, такими как бег, прыжок, огромное значение здоровья.

4. Игрок имеет возможность стрелять. Снаряды могут быть особенными. Взрывы наносят урон врагам в радиусе, критический урон повышен.

5. В мире генерируются предметы, такие как зелья и Звезда Эдема. Игрок получает бонусы при сборе этих предметов.

6. Все игровые объекты имеют соответствующие текстуры и анимации для различных состояний.

7. Интерфейс отображает важную информацию, такую как здоровье, опыт, основные характеристики. Он интуитивно понятен и легко читаем, чтобы игрок мог быстро ориентироваться и принимать решения.

8. Враги появляются волнами, с увеличением сложности по мере прогресса игрока. Каждая волна включает различных врагов и увеличивает их количество, чтобы поддерживать интерес и вызов для игрока.

2 ОТЧЁТ О РАЗРАБОТКЕ ПРОГРАММЫ

2.1 Архитектура программы

Для упрощения и систематизации взаимодействий классов в игре, можно выделить ключевые функции и основные методы, которые позволяют понять, как объекты взаимодействуют друг с другом.

Класс Player:

- Конструктор Player(float x, float y, float size, const sf::Texture& texture): создает игрока с заданной позицией и текстурой;
- handleInput(): обрабатывает ввод для управления игроком;
- update(float dt, World& world): обновляет состояние игрока с учетом времени и взаимодействия с миром, включая обработку столкновений и перемещений;
- takeDamage(float amount): уменьшает здоровье игрока при получении урона;
- gainExperience(float amount): увеличивает опыт игрока, что может вызвать повышение уровня;
- draw(sf::RenderWindow& window): отображает игрока на экране.

Класс Enemy (абстрактный):

- update(float dt, const sf::Vector2f& playerPosition, const std::vector<std::unique_ptr>& enemies, Player& player, sf::RenderWindow& window): обновляет состояние врага, определяя его поведение (например, преследование игрока);
- takeDamage(float amount, sf::RenderWindow& window): уменьшает здоровье врага при получении урона;

- isDead() const: проверяет, мертв ли враг;
- draw(sf::RenderWindow& window): отрисовывает врага на экране.

Классы Runner, Jumper и Boss (наследники Enemy):

- update(float dt, const sf::Vector2f& playerPosition, const std::vector<std::unique_ptr>& enemies, Player& player, sf::RenderWindow& window): реализуют логику поведения каждого конкретного типа врагов (например, бегун преследует игрока, джампер прыгает к нему);
- draw(sf::RenderWindow& window): отображает конкретного врага.

Класс World:

- update(const sf::Vector2i& playerChunkPos, Player& player, sf::RenderWindow& window): обновляет мир в зависимости от положения игрока, генерируя новые чанки и обновляя объекты;
- checkCollision(const sf::FloatRect& bounds) const: проверяет столкновения игрока и врагов с элементами мира;
- draw(sf::RenderWindow& window, const sf::Vector2f& playerPos): отрисовывает мир, ориентируясь на позицию игрока;
- generateChunk(const sf::Vector2i& chunkPos): генерирует новые участки мира, которые активируются при перемещении игрока.

Класс ProjectileManager

- update(float dt, const std::vector<std::unique_ptr>& enemies): обновляет положение всех снарядов и проверяет их столкновения с врагами;
- shoot(const sf::Vector2f& startPosition, const sf::Vector2f& targetPosition): создает снаряд, выпущенный игроком в заданном направлении;

- draw(): отрисовывает все активные снаряды.

Класс UIManager

- update(): обновляет пользовательский интерфейс в зависимости от состояния игрока (здоровье, опыт).
- draw(): отображает интерфейс на экране (здоровье, опыт, статистика).

Файл Main.cpp – точка входа в приложение, инициализирует игру и содержит главный игровой цикл.

Функция main, запускает игру, обрабатывает события, обновляет и отрисовывает игровые объекты.

Файл Const.h – содержит глобальные константы, используемые по всему проекту.

Содержимое файла констант:

- окно (размеры, заголовок);
- пути к ресурсам (текстуры, шрифты, звуки);
- параметры игрока (скорость, здоровье, урон);
- параметры врагов (типы, здоровье, урон);
- параметры игровых объектов (размеры, шансы спавна);
- константы анимации (длительность кадров, количество кадров).

2.2 Алгоритмическая часть

Процедурная генерация мира в игре осуществляется с использованием шума Перлина для создания биомов и деталей. Мир состоит из чанков, каждый

из которых содержит множество блоков. Для каждого блока определяется биом (лес или пустыня) и цвет земли. Затем, на основе значений шума, в блоках появляются объекты, такие как деревья или кактусы, а также подбираемые предметы, такие как зелья опыта.

3 РУКОВОДСТВО ПО СБОРКЕ И ЗАПУСКУ

3.1 Требования

Для корректной сборки проекта требуются следующие средства:

- CMake (версия 3.28 или выше);
- MinGW (для сборки. Крайне желательно использовать MSYS2 установщик);
- Git (для клонирования репозитория).

3.2 Сборка и запуск проекта

Инструкция по сборке и запуске проекта:

1. Перейдите в директорию проекта: “cd EndlessSurvivors”.
2. Создайте директорию сборки и перейдите в неё: “mkdir build && cd build”.
3. Запустите CMake для генерации файлов сборки: “cmake -G "MinGW Makefiles" ..”.
4. Соберите проект с помощью Make: “mingw32-make”.
5. Запустите игру: “cd bin && main.exe”.

Полная инструкция по сборке проекта доступна в Readme файле проекта на GitHub репозитории. – URL: https://github.com/foblako/EndlessSurvivors (дата обращения 20.12.24).

4 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

4.1 Управление

Управление персонажем осуществляется через клавиши WASD. Стрельба по нажатию левой кнопкой мыши в точку, где находится курсор.

4.2 Графический интерфейс программы

Графический интерфейс представляет пользователю возможность отслеживать изменение характеристик (см. Рисунок 1)

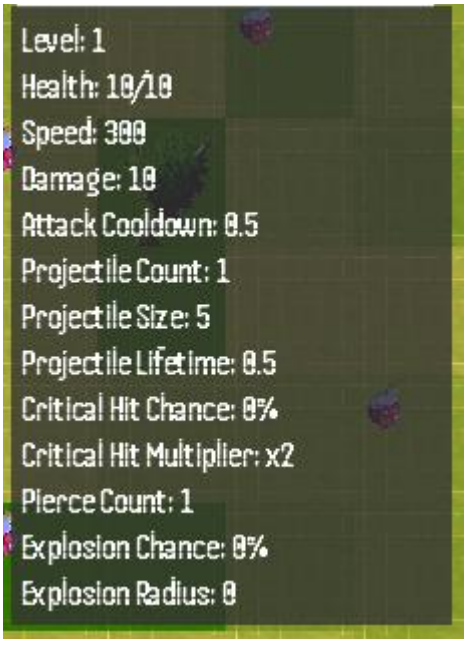

Рисунок 1 – Характеристики в UI

Также пользователь видит полоску, отображающую его текущее здоровье над окном характеристик (см. Рисунок 2)

Рисунок 2 - Полоса здоровья

Внизу экрана располагается полоска, показывая уровень опыт и сколько осталось до следующего уровня (см. Рисунок 3)

Рисунок 3 - Полоса опыта

При проигрыше игроку предлагается выбрать дальнейшие действия: выход или “restart” (см. Рисунок 4). При победе выбор аналогичен (см. Рисунок 5)

Рисунок 4 - Меню смерти игрока

Рисунок 5 - Меню победы игрока

При наборе достаточного количества очков опыта игрок может выбрать одну из двух обновлений его характеристик (см. Рисунок 6)

Рисунок 6 - Меню обновления характеристик

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсовой работы были успешно достигнуты поставленные цели по разработке игры "Endless Survivors". Проект позволил не только реализовать основные аспекты игровой механики, такие как управление персонажем, взаимодействие с врагами и окружающим миром, но и значительно углубить знания в области программирования на языке C++17.

Работа над игрой способствовала закреплению навыков объектно-ориентированного программирования, эффективного использования стандартной библиотеки C++ и освоению современных возможностей языка. Важным элементом стало знакомство с библиотекой SFML, которая предоставила необходимые средства для работы с графикой, звуком и взаимодействием с пользователем. Опыт разработки данного проекта помог лучше понять принципы работы с мультимедийными библиотеками и научиться решать задачи оптимизации производительности в игровых приложениях.

Кроме того, проект "Endless Survivors" дал возможность глубже погрузиться в структуру разработки игровых систем, управление состояниями объектов, обработку событий и анимаций, что расширило общее понимание создания сложных интерактивных приложений.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Официальный сайт языка C++. – URL: https://en.cppreference.com/w/https://en.cppreference.com/w/ (дата обращения 20.12.24).

2. Официальная документация библиотеки SFML. – URL: https://www.sfml-dev.org/documentation/2.6.2/ (дата обращения 20.12.24).

3. Дополнительная библиотека, реализующая шум Перлина. – URL: https://github.com/Reputeless/PerlinNoise/tree/master (дата обращения 20.12.24).