МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

СОЗДАНИЕ ИГРЫ НА ИГРОВОМ ДВИЖКЕ UNREAL ENGINE

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Павловского Владислава Юрьевича

Научный руководитель
доцент, к. ф.-м. н. _____ А. С. Иванова

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разработка игр — это сложный и многогранный процесс, который требует глубоких знаний программирования и умения работы с игровыми движками. В настоящее время игры являются неотъемлемой частью развлекательной индустрии и пользуются большой популярностью среди широкой аудитории.

Unreal Engine (UE) — один из самых популярных игровых движков, который предоставляет разработчикам мощный набор инструментов для создания игр разного жанра и уровня сложности. UE был создан компанией Epic Games в 1998 году и с тех пор значительно эволюционировал. Сегодня UE является первоклассным решением для разработки игр, благодаря своей функциональности, производительности и качеству графики.

UE используется для создания различных типов игр, от маленьких независимых проектов до крупных AAA-игр. поддерживает использование языков программирования `C++` и `Blueprint`.

UE имеет мощный редактор, который позволяет создавать игровое окружение и персонажей, управлять логикой игры и настраивать детали графики. Разработчики также могут использовать различные инструменты, такие как физическая симуляция, система частиц, звуковые эффекты и другие, чтобы создать более реалистичный и захватывающий игровой мир.

Кроме того, UE предоставляет разработчикам широкие возможности для создания многопользовательских игр, благодаря встроенным инструментам для работы с сетями и мультиплеером.

В целом, Unreal Engine является одним из наиболее полных и мощных инструментов для разработки игр на сегодняшний день. Он предоставляет разработчикам все необходимые инструменты и функциональность для создания высококачественных и захватывающих игр.

Целью работы является создание игры. Для этого были поставлены следующие задачи:

1. Разработать игрового персонажа.
2. Реализовать минимальный игровой интерфейс.
3. Создать две локаций, а именно: остров и замок.
4. Добавить врага и его поведение.
5. Разработать анимации: получение урона, смерти, удара мечем.
6. Создать инвентарь персонажа.

1 Практическая часть

1.1 Создание главного персонажа

Создание персонажа происходит в два этапа:

1. Создать класс игрока на C++ и описать в нём все ключевые атрибуты и функции.
2. Создать чертеж Blueprint игрока, для более удобных манипуляций персонажем.

Для главного персонажа нужно реализовать функции, которые выполняют следующие действия:

1. Передвижение вперед и назад;
2. Перемещение по сторонам;
3. Бег;
4. Уворот;
5. Повороты камеры.

После реализации на C++ необходимо создать класс Blueprint для персонажа. В нем нужно придать персонажу внешний вид:

1. Капсулу, которая отвечает за столкновение персонажа со всеми другими объектами;
2. Mesh, который в последствии поможет реализовать анимации персонажа, а также установить внешний вид;
3. Компонент Character␣Movement, отвечающий за передвижение героя (см. рис 1).

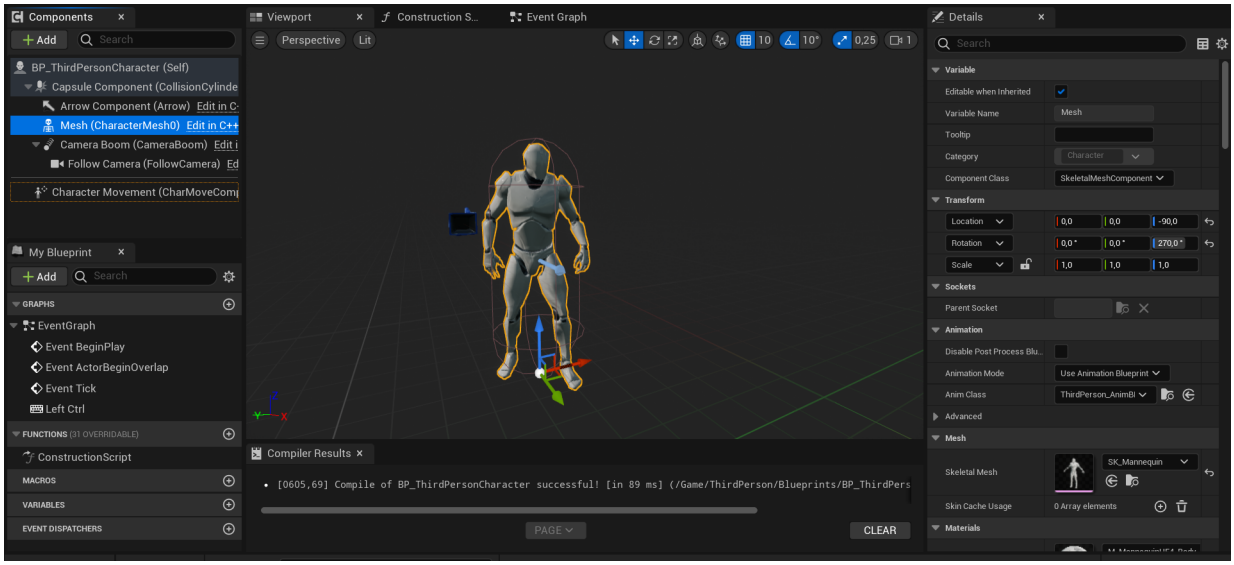

Рисунок 1 – Главный персонаж

1.2 Реализация работы анимаций

Для того чтобы анимации персонажа работали корректно, нужно предусмотреть различные ситуации, когда анимации могли бы прерваться. Для предотвращения такой проблемы для каждой анимации были добавлены различные проверки и задержки, а также запрет на обработку событий.

1.3 Добавление меча

Для добавления меча нужно сделать несколько вещей:

1. Создать сокет на скелете игрока, для крепления меча к персонажу;
2. Добавить компонент `ChildActor` к `Mesh` персонажа;
3. Разместить в правильном положении на игроке (см. рис 2).

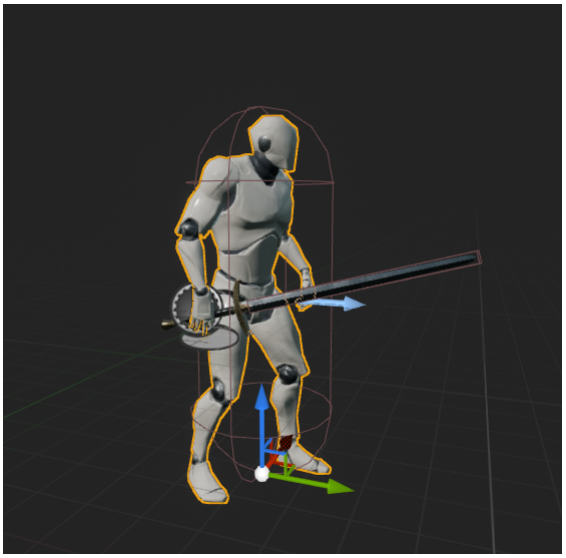

Рисунок 2 – Персонаж с мечем

1.4 Пользовательский интерфейс

В пользовательском интерфейсе будет отображаться полоска здоровья и магической энергии. Для реализации этого создается виджет, в котором будет отображаться две полоски (см. рис 3).

Рисунок 3 – Здоровье и магическая энергия

1.5 Новые анимации

Для создание анимаций использовалась программа `Cascadeur`. С её помощью были созданы анимации:

1. Удара мечем (см.рис 4);
2. Получение урона (см.рис 5);
3. Смерть персонажа (см.рис 6).

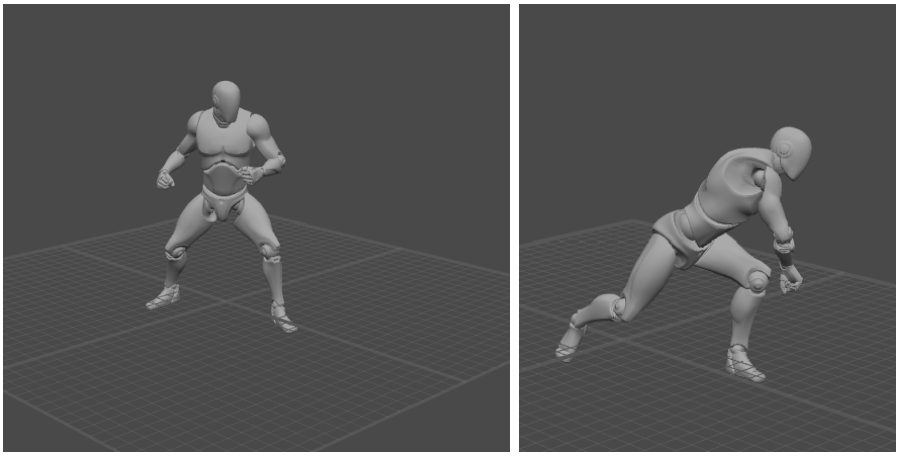

Рисунок 4 – Анимация атаки от начального ключа до конечного удара

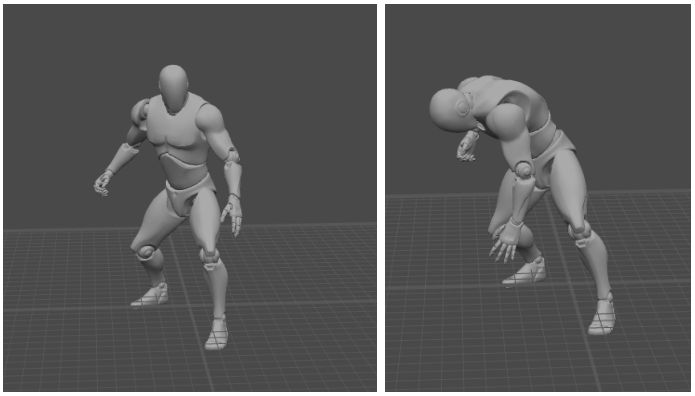

Рисунок 5 – Анимация получения урона от начального ключа до конечного

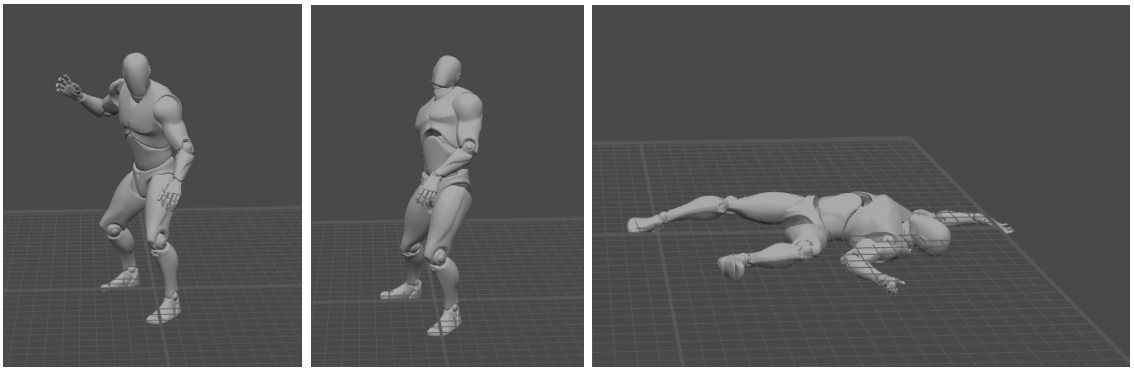

Рисунок 6 – Анимация смерти персонажа по ключевым кадрам

1.6 Создание врага

Создание вражеского персонажа происходит в три этапа:

1. Написание класса врага на C++;
2. Реализация AI-Controller для патрулирования локации;
3. Создать класс Blueprint для более удобных манипуляций персонажа.

Для вражеского персонажа нужно реализовать следующие функции:

1. Для перемещения к главному персонажу;
2. Для поиска игрока;
3. Для нахождения пересечения между двумя компонентами персонажей;
4. Для вызова анимаций атаки;

1.7 Реализация AI-Controller

AI-Controller отвечает за управление поведением искусственного интеллекта в игре. Для этого нужно наследовать класс от базового AI-Controller и реализовать функцию, которая позволяет врагу патрулировать на всей карте.

1.8 Реализация класса персонажа в Blueprint

После реализации врага на C++ необходимо создать класс Blueprint. В него нужно добавить:

1. Капсулу, которая отвечает за столкновение персонажа со всеми другими объектами;
2. Mesh, который в последствии поможет реализовать анимации персонажа, а также установить внешний вид;
3. Компонент Character␣Movement, отвечающий за передвижение героя.
4. А также во вкладке Pawn нужно выбрать созданный AI-Controller (см.рис 7).

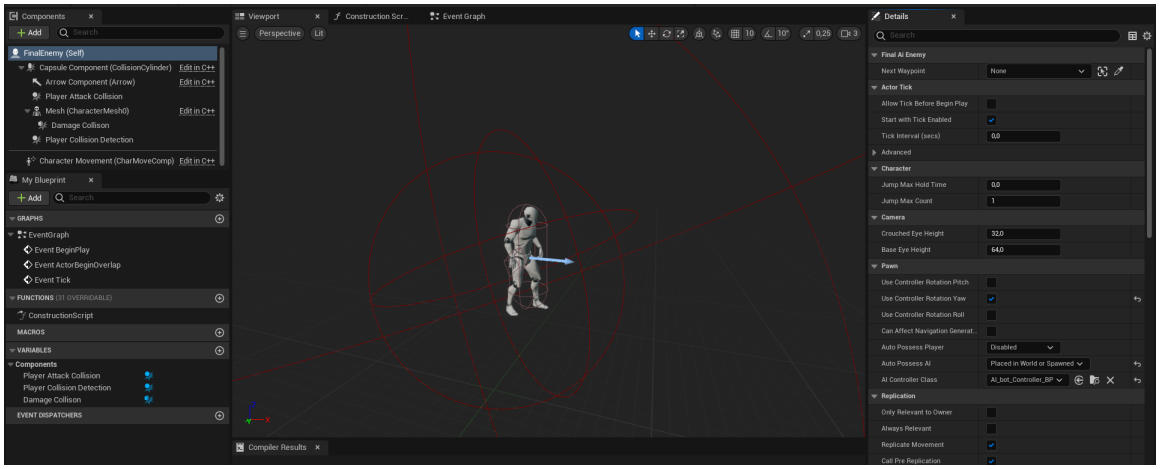

Рисунок 7 – Input

1.9 реализация анимаций

При помощи встроенного в движок инструмента для создания анимаций, были созданы различные состояния: спокойствия, ходьбы, бега, прыжка, уворота, удара. Далее при помощи класса `Blueprint`, который работает с анимациями, нужно указать условия перехода из одной анимации в другую.

1.10 Создание новой локации

Создание локаций происходит во внутреннем редакторе уровней. С помощью инструментов скульптинга создается ландшафт карт. В встроенном магазине бесплатного контента нужно скачать материалы для ландшафта и замка. При помощи специального плагина на локацию добавляется озеро. Для того чтобы построить замок, нужно соединять материалы так чтобы образовались части замка, а потом уже соединить их в единое целое (см. рис 8).

Рисунок 8 – Финальный вид новой локации

Также была создана локация остров (см. рис 9).

Рисунок 9 – Финальный вид острова

1.11 Создание инвентаря

Инвентарь — это место, где хранятся предметы, которые персонаж может использовать в процессе игры. Он представляет из себя массив предметов, которые могут быть доступны в игры.

Прежде чем создавать инвентарь, нужно создать класс предмета, который будет являться родительским классом для всех последующих предметов. В нем нужно указать общие переменные, а именно:

1. Текстовое описание при использовании;
2. Текстура для отображения предмета в инвентаре;
3. Название;
4. Описание;
5. Вес;

Далее создать предмет еды, который будет восполнять здоровье персонажа при использовании. Для этого нужно унаследовать родительский класс предмета и добавить переменную здоровье. Потом нужно реализовать функцию, которая лечит персонажа, который использовал данный предмет.

Для создания инвентаря используется класс `Actor␣Component`. Его особенность заключается в том, что он может быть прикрепленным к персонажу. Далее нужно создать переменные:

1. `DefaultItems` — массив предметов по умолчанию;

2. `Capacity` — определяет вместимость;
3. Переменную, которая обновляет виджет инвентаря;
4. Текущий массив предметов.

А также надо добавить функции, которые реализуют такие действия как: добавление и удаление предмета.

Далее нужно создать два виджета, для отображения инвентаря и предметов в нем (см. рис 10).

Рисунок 10 – Виджет инвентаря и предмета

Также было реализована подсказка, когда наводишь мышкой на предмет в инвентаре, то выскакивает текстовое сообщение о названии, описании и о том как его можно использовать.

ЗАКЛЮЧЕНИЕ

В заключении можно отметить, что разработка игр является сложным и трудоемким процессом, который требует от разработчиков глубоких знаний программирования и умения работы с игровыми движками. Unreal Engine (UE) является одним из самых популярных игровых движков, который предоставляет мощный набор инструментов для создания игр разного жанра и уровня сложности. UE имеет мощный редактор, который позволяет создавать игровое окружение и персонажей, управлять логикой игры и настраивать детали графики.

Целью данной работы является расширение функционала игры, что включает в себя устранение существующих багов, добавление врагов и их поведения, а также разработку новых анимаций. В свою очередь, Unreal Engine предоставляет разработчикам все необходимые инструменты и функциональность для создания высококачественных и захватывающих игр. Благодаря широким возможностям для создания многопользовательских игр, UE значительно расширяет аудиторию игр, которые могут быть созданы на этой платформе.

В результате выполнения работы была создана, а также выполнены следующие задачи:

- Разработан игровой персонаж.
- Реализовован минимальный интерфейс.
- Созданы две локаций, а именно: остров и замок.
- Добавлен врага и его поведение.
- Созданы новые анимации, а именно:
  - Получение урона
  - Смерть
  - Удара мечем
- Реализован инвентарь.

Из чего можно сделать вывод, что все задачи были реализованы, а это значит, что поставленная цель была успешно выполнена.