**содержание**

[Введение 3](#_Toc500310488)

Глава [1. Теоретическая часть 5](#_Toc500310489)

[1.1 Презентация в маркетинге 5](#_Toc500310490)

[1.2 Основные сведения об Unreal Engine 4 5](#_Toc500310491)

[1.3 Основные инструменты 8](#_Toc500310492)

[1.3.1 Режимы размещения 8](#_Toc500310493)

[1.3.2 Контент-браузер 10](#_Toc500310494)

[1.4 Инструмент визуального скриптинга Blueprint 10](#_Toc500310495)

Глава [2. Практическая часть 12](#_Toc500310496)

[2.1 Установка движка Unreal Engine 12](#_Toc500310497)

[2.3 Помещение на уровень игрового персонажа и привязка клавиш перемещения 14](#_Toc500310498)

[2.4 Создание класса Blueprint Actor Component для отслеживания статистики персонажа 18](#_Toc500310499)

[2.5 Смена положения камеры 21](#_Toc500310500)

[2.6 Добавление функционирующих дверей 22](#_Toc500310501)

[2.7 Активация Ragdoll при смерти персонажа 24](#_Toc500310502)

[2.8 Добавление GUI 25](#_Toc500310503)

[2.9 Итог 25](#_Toc500310504)

[Заключение 26](#_Toc500310505)

[Список литературы 27](#_Toc500310506)

# **введение**

Предметной областью курсовой работы является презентация, сопровождение и тестирование программного продукта.

Одним из условий эффективной деятельности любой компании является формирование её привлекательного облика в глазах клиентов. В большинстве случаев для того чтобы представить продукт или компанию используют мультимедийные презентации. Преимущества этой технологии состоят в современной форме презентации с использованием компьютера.

Кроме рекламных целей презентации могут выполнять и другие практические функции: обучение и проверка знаний сотрудников, клиентов или потребителей продукции, распространение нормативно-технической документации методических материалов и т.д.

Проект курсовой работы выполнен в программе Unreal Engine 4, выпущенной компанией Epic Games. Этот игровой движок распространяется на бесплатной основе, так же он имеет открытый код, который можно посмотреть на сайте GitHub, это нужно, если разработчик захочет сделать свою модификацию игрового движка.

Актуальность курсовой работы можно объяснить увеличением роста популярности разработки игр независимыми компаниями. Indie-игры – это компьютерные игры, созданные отдельным разработчиком или небольшим коллективом без финансовой поддержки издателя компьютерных игр. Ввиду своей независимости Indie-разработчики не имеют операционных ограничений со стороны издателей или технических ограничений и не нуждаются в одобрении издателя, что является обязательным для разработчиков массовых игр. Решения гейм дизайнера также не ограничивается бюджетом проекта, более того, чем меньше коллектив, тем ярче выражается индивидуальность конкретного разработчика. В поисках источника финансирования новой игры Indie-разработчики могут прибегать к краудфандингу или созданию вспомогательного сообщества для разработки игры. Краудфандинг – способ коллективного финансирования, основанный на добровольных взносах. Самые популярные краудфандинговые компании: Kickstarter, IndieGoGo. Для того чтобы продвинуть игру, разработчики используют предлагаемые в интернете службы цифровой дистрибуции. Цифровая дистрибуция – один из современных методов распространения легального электронного контента по интернет-каналам без использования материальных носителей. Преимущество заключается в легком и быстром поиске и приобретении копий необходимого программного обеспечения. Самыми знаменитыми способами цифровой дистрибуции игр Indie-разработчиков являются Steam и GOG.Техническая демоверсия – предварительная (ограниченная и, как правило, бесплатная) версия продукта, например, компьютерной игры. Название термина – сокращение от слова «демонстрация».

Цель работы: разработать техническую демоверсию игры на движке Unreal Engine 4.

Для достижения цели необходимо решить следующие задачи:

* Изучить инструмент визуального скриптинга Blueprint.
* Создать свои классы Blueprint для написания логики к различным объектам на сцене.
* Разместить графическую составляющую.
* Скомпилировать демоверсию.

# **Глава 1. теоретическая часть**

## **1.1 Презентация в маркетинге**

Презентация – документ или комплект документов, предназначенный для представления чего-либо. Целью презентации является донесение полноценной информации об объекте презентации в удобной форме. Презентация является одним из маркетинговых и PR инструментов.

Типы презентаций:

* По форме подачи информации:
  + Статическая форма – классический доклад с иллюстрациями на доске.
  + Динамическая форма – мультимедийная презентация с использованием ИТ технологий.
* По спецификации содержания:
  + Стандартная – распространение строгой деловой информации, яркие и креативные детали неуместны.
  + Концептуальная – усиление интереса аудитории посредством шоу программы и общения с аудиторией.

## **1.2 Основные сведения об Unreal Engine 4**

Unreal Engine 4 – это движок нового поколения для создания видеоигр, центральный программный компонент компьютерных и видеоигр или других интерактивных приложений с графикой, обрабатываемой в реальном времени. Он предоставляет большой набор инструментария для создания 2D и 3D проектов. Это могут быть игры, архитектурная визуализация и различные демоверсии или видеоролики. Unreal Engine 4 распространяется бесплатно.

Системные требования Unreal Engine 4:

* Настольный ПК или MacOS.
* Windows 7 64-bit или MacOS X 10.9.2 или старше.
* Четырехядерный процессор Intel или AMD, 2.5 GHz или лучше.
* NVIDIA GeForce 470 GTX или AMD Radeon 6870 HD или лучше.
* 8 Гб ОЗУ.

Этот движок доступен для создания проектов для Windows PC, Linux, iOS, Mac, Android. Также есть встроенная поддержка виртуальной реальности для Oculus Rift. Помимо этого, UE4 поддерживает Xbox One и PlayStation 4.

Для того чтобы скачать Unreal Engine 4, нужно зайти на официальный сайт производителя и скачать клиент-программу, в который есть список платных и бесплатных программ. После установки клиента нужно зарегистрироваться и подтвердить свою почту. В списке доступных приложений выбрать Unreal Engine 4 и место для скачивания, на данный момент самая актуальная версия это 4.18.

Для того чтобы начать разрабатывать проект, нужно его создать. При открытии, Unreal Engine 4 предлагает выбрать способ программирования Blueprint или С++, шаблон видеоигры или выбрать пустой бланк. После выбора шаблона, программа предлагает выбрать место хранения проекта, его название, платформу и начальный контент, включающий в себя стандартный набор 3D моделей, материалов и архитектурные модели (см. рис. 1.1).

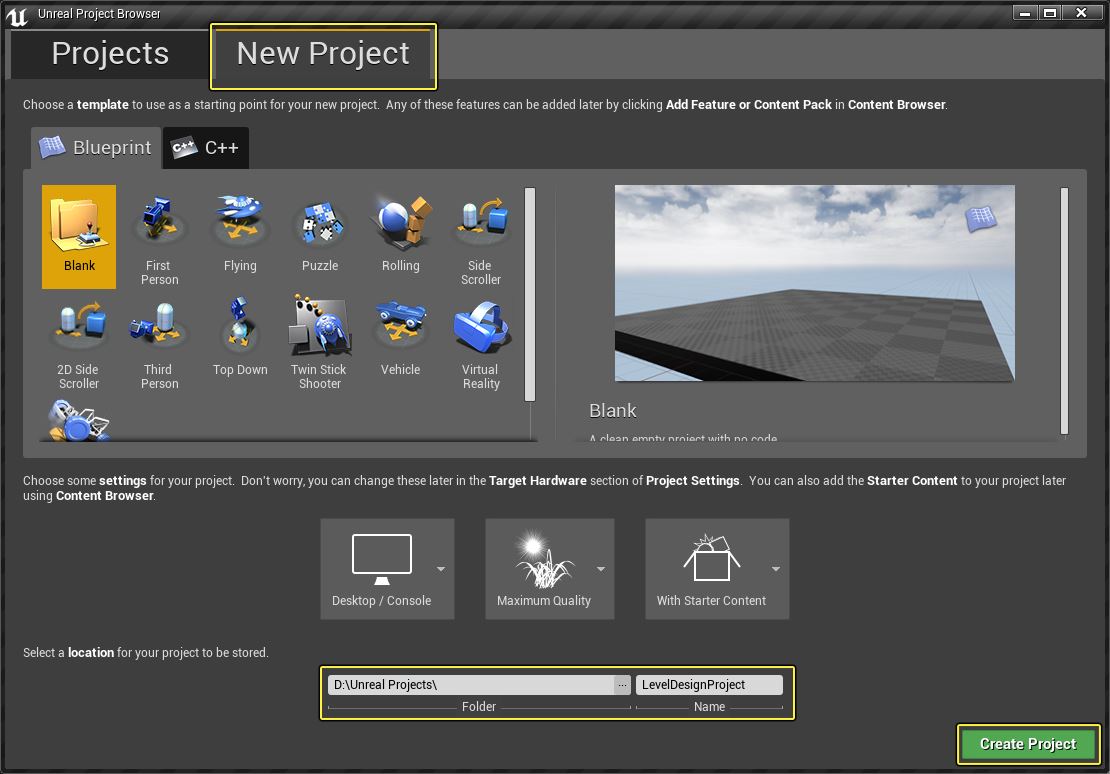


Рис. 1.1. Браузер проектов

Проект представляет собой автономный блок, который содержит все содержимое и код, которые составляют индивидуальную игру, и совпадает с набором каталогов на диске. Например, на рисунке 1.2 изображено Древо иерархии браузера содержимого, который содержит ту же структуру каталогов, которая находится внутри папки Project на диске.

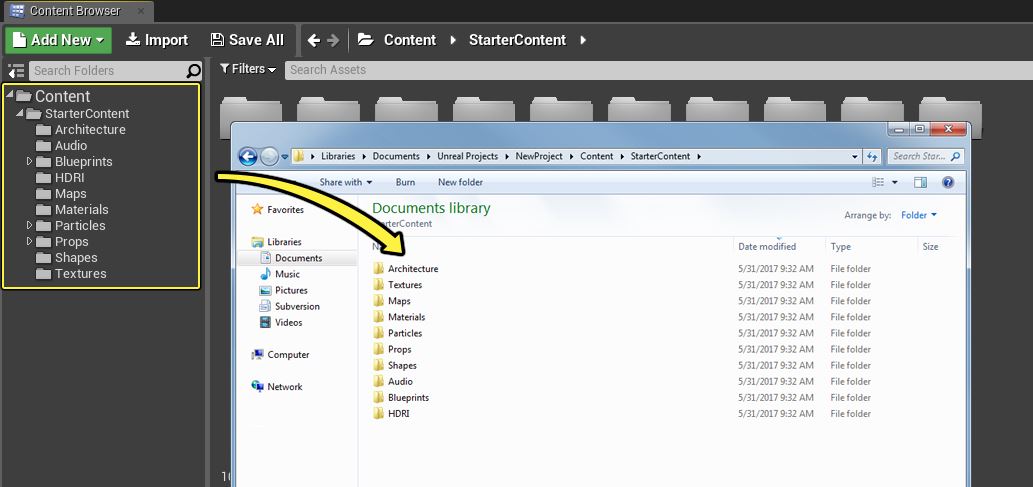


Рис. 1.2. Древо иерархии

В Unreal Engine 4 возможно создать любое количество различных проектов, которые могут поддерживаться и развиваться параллельно. Unreal Engine может легко переключаться между ними, что позволит одновременно работать с несколькими играми или иметь несколько тестовых проектов. В Unreal Editor сцены, в которых создается игровой опыт, обычно называют уровнями. Любой объект, помещённый в мир, будь то свет или персонаж, считаются актером. Actor – это класс программирования, используемый в Unreal Engine для определения объекта, который имеет данные о трехмерном положении, вращении и изменении масштаба. Для того чтобы видеть, что создается в Unreal Editor, существуют видовые экраны. Viewport – это окно для управления уровнем. Видовые экраны содержат множество инструментов и визуализаторов, которые помогут точно определить нужные данные.

## **1.3 Основные инструменты**

Для размещения любого объекта, в Unreal Editor есть специальная панель, которая называется “Modes” или режимы. Панель «Режимы» содержит ряд различных режимов работы редактора. Они изменяют основное поведение редактора уровней для специализированной задачи, например, размещения новых актёров в мире, создание геометрических кистей, рисование, создание растений, скульптурирование.

### **1.3.1 Режимы размещения**

Place Mode – это специализированный инструмент для ускорения и упрощения создания среды. Этот режим фокусируется только на тех активах, которые могу быть непосредственно размещены на уровне. Режим Place также облегчает добавление специализированных типов актеров, таких как   
«точечный свет». Кроме того, он упрощает создание специальных подтипов актеров, таких как создание субъектов физики из выбранных статических сеток. Для того чтобы разместить актера из режима Place, нужно открыть проект и в окне режимы выбрать Place Mode (1), далее нужно перейти на вкладку «Основные» (2), затем щелкнуть левой кнопкой мыши и перетащить актер (3) на уровень (см. рис. 1.3).

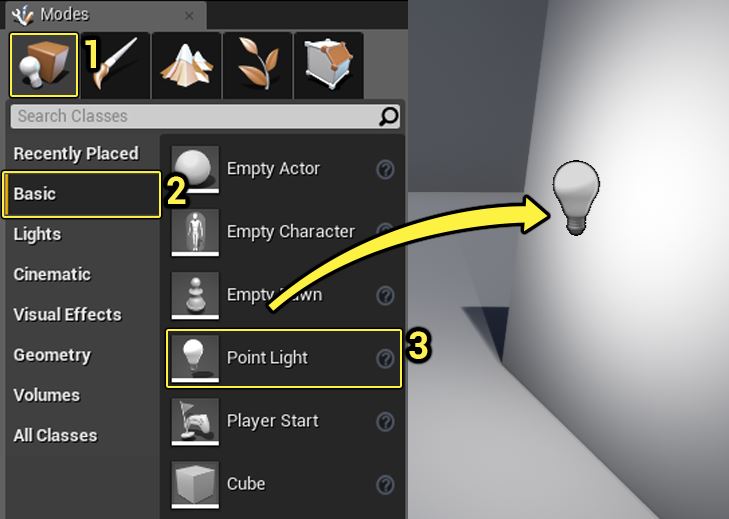


Рис. 1.3. Размещение специального актера «Точечный свет»

Режим Mesh Paint – обеспечивает быстрый и простой способ регулировки цвета и материала на актерах.

Режим Landscape tool – эта система позволяет создавать рельеф местности для уровня – горы, долины, неровную или наклонную землю, даже отверстия для пещер. В этом режиме существуют три инструмента для настройки пейзажа.

* Инструмент Manage – позволяет создавать новые ландшафты и изменять ландшафтные компоненты.
* Инструмент Sculpt – позволяет изменить форму ландшафта, добавить впадины, горы и пещеры.
* Инструмент Paint – позволяет изменить внешний вид частей пейзажа, нанести на него материал на основе слоёв.

Режим Foliage – эта система позволяет быстро нарисовать или стирать наборы статических ландшафтных актеров. Этот режим содержит пять инструментов.

* Инструмент Paint – этот инструмент предназначен для размещения мелких актеров на большой площади, например, травы.
* Инструмент Reapply Tool – позволяет выборочно изменить определенные параметры для актеров, уже размещенных на уровне.
* Инструмент Select – позволяет выбирать единичные размещенные объекты.
* Инструмент Lasso – позволяет одновременно выбирать количество большого количества актёров.
* Инструмент Fill – позволяет залить всю область статической сетки ландшафта различными актерами.

Режим Geometry Brush Actor – геометрические кисти являются самым основным инструментом для построения уровней в Unreal.

На базовом уровне создания уровней всё сводится к размещению элементов на карте. Эти предметы могут представлять собой геометрию, украшения в виде кистей, статических сеток, света, старта игрока, транспортных средств.

### **1.3.2 Контент-браузер**

Контент-браузер является основной областью в Unreal Editor для создания, импорта, организации, просмотра и изменения содержимого активов в Unreal Editor. Он также предоставляет возможность управлять папками содержимого и выполнять другие полезные операции с активами, такие как переименование, перемещение, копирование и просмотр.

## **1.4 Инструмент визуального скриптинга Blueprint**

Система визуальных сценариев Blueprints, разработанная на ящике программирования С++ представляет собой полноценную систему сценариев, основанную на концепции использования интерфейса на основе узлов создания элементов игрового процесса. Эта система чрезвычайно гибкая и мощная, поскольку она позволяет дизайнерам использовать весь спектр инструментов, которые обычно доступны только программистам. Кроме того, специфичная Blueprint разметка, доступная в реализации Unreal Engine C++, позволяет программистам создавать базовые системы, которые могут быть расширены. Blueprints – это визуальное дополнение к вашей игре, для подключения узлов, событий, функций и переменных. С помощью связей можно создавать сложные элементы игрового процесса. Чертежи работают с использованием графиков и узлов для различных целей: построения объектов, отдельных функций и общих событий игрового процесса, которые специфичны для каждого чертежа. Есть два наиболее распространённых типа чертежа, это «Чертеж уровня» и «Чертеж класса»

* Чертеж уровня – представляет собой специализированный тип, который действует в качестве глобального графа событий в масштабах уровня. Каждый уровень в проекте имеет свой чертеж. Чертежи уровня также обеспечивают механизм управления потоками уровня и Matinee, а также привязки событий к актерам, размещенным на уровне.
* Чертеж класса – Классы Blueprint идеально подходят для создания интерактивных активов, таких как двери, переключатели, предметы коллекционирования и разрушаемые пейзажи.

Сценарий Blueprint также можно использовать для создания HUD игры, который аналогичен классам Blueprint, поскольку он может содержать последовательности событий и переменные, но присваивается активу GameMode проекта, а не добавляется непосредственно на уровень. HUD можно настроить для чтения переменных из других чертежей и использовать их для отображения панели работоспособности, обновления значения оценки, отображения объективных маркеров и т. Д. Также можно использовать HUD для добавления хит-боксов для таких элементов, как кнопки, на которые можно щелкнуть, или, в случае мобильных игр, ответить на сенсорный ввод.

# **Глава 2. практическая часть**

## **2.1 Установка движка Unreal Engine**

Для создания первого проекта нужно скачать клиент программу с официального сайта компании Epic Games <https://www.epicgames.com/ru>. На сайте в правом верхнем углу будет кнопка Get Epic Games (см. рис. 2.1), при нажатии на неё начнется скачивание программы клиента.

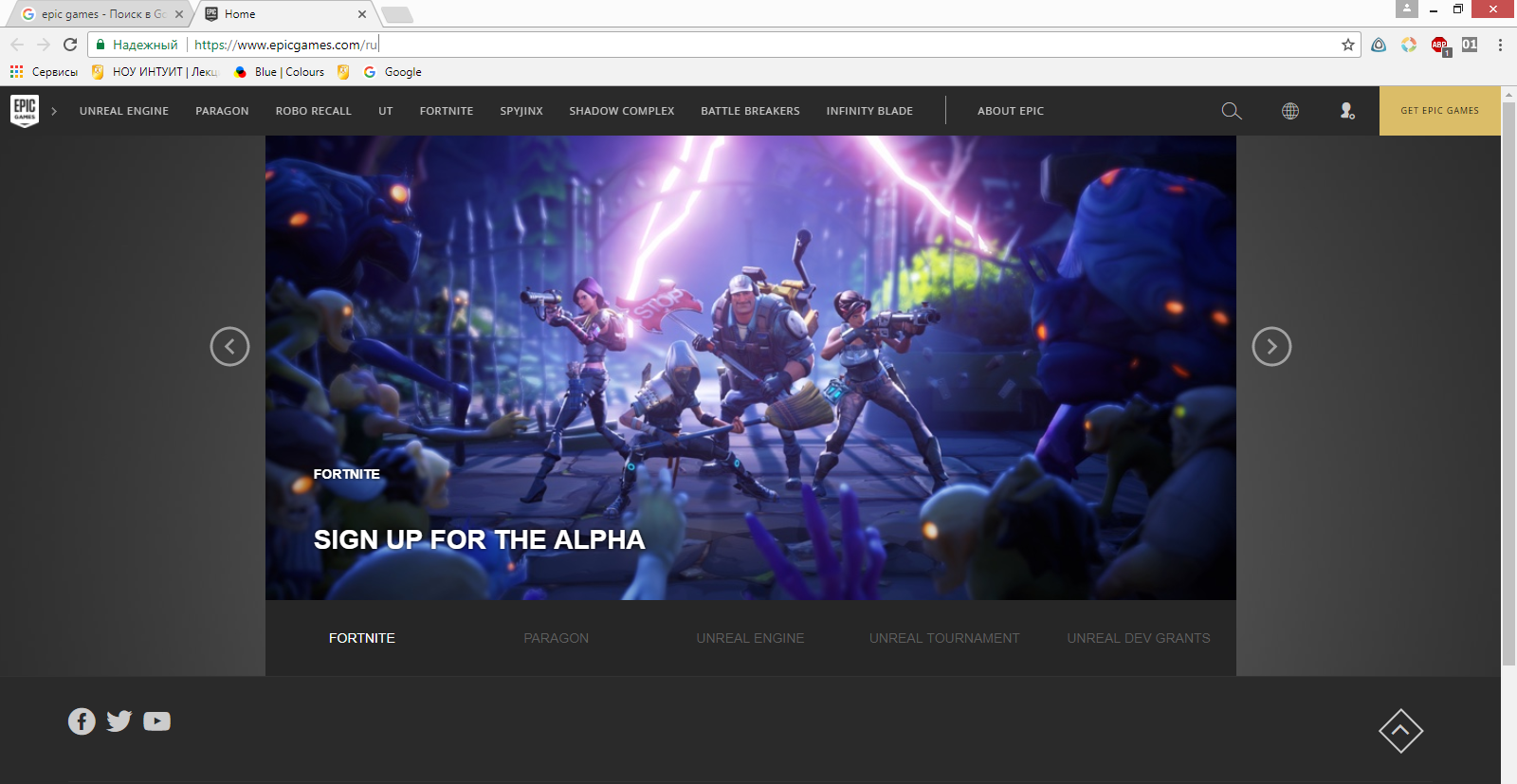


Рис. 2.1. Официальный сайт компании Epic Games

При запуске установщика, программа попросит указать её расположение на диске, так как это всего лишь клиент, то много места он не займет в отличии от самого движка, который желательно ставить на отдельный диск или раздел диска, на котором много места. После установки клиент потребует зарегистрироваться или войти, если аккаунт уже есть. Далее в главном меню нужно выбрать Unreal Engine и нажать скачать (см. рис. 2.2).

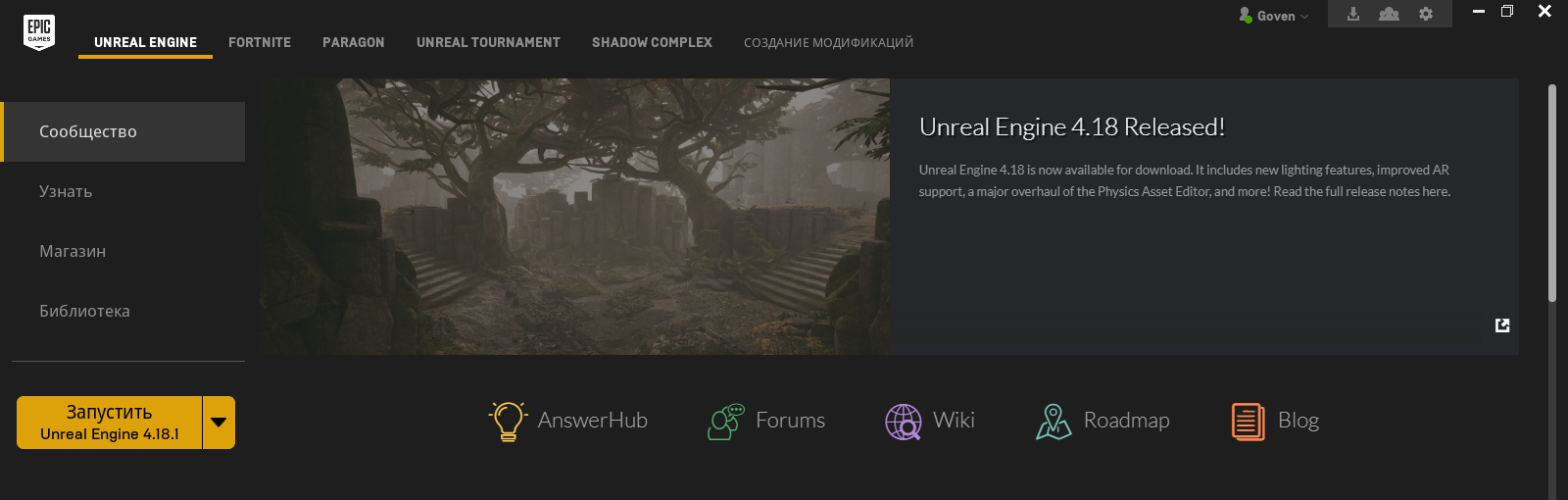


Рис. 2.2. Клиент-программа, позволяющая скачать Unreal Engine

При нажатии кнопки скачать, программа выдаст окно, в котором нужно выбрать место хранения движка (см. рис. 2.3), желательно выбрать более свободный диск или раздел, так как программа будет весить примерно 20 гигабайт.

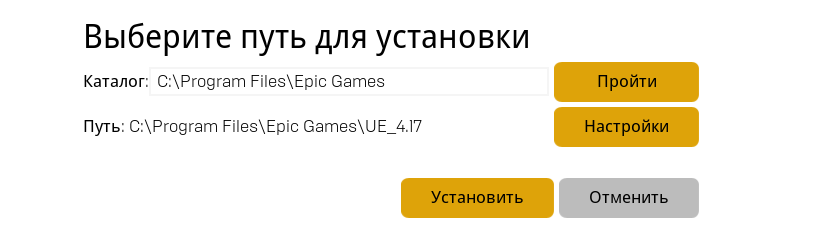
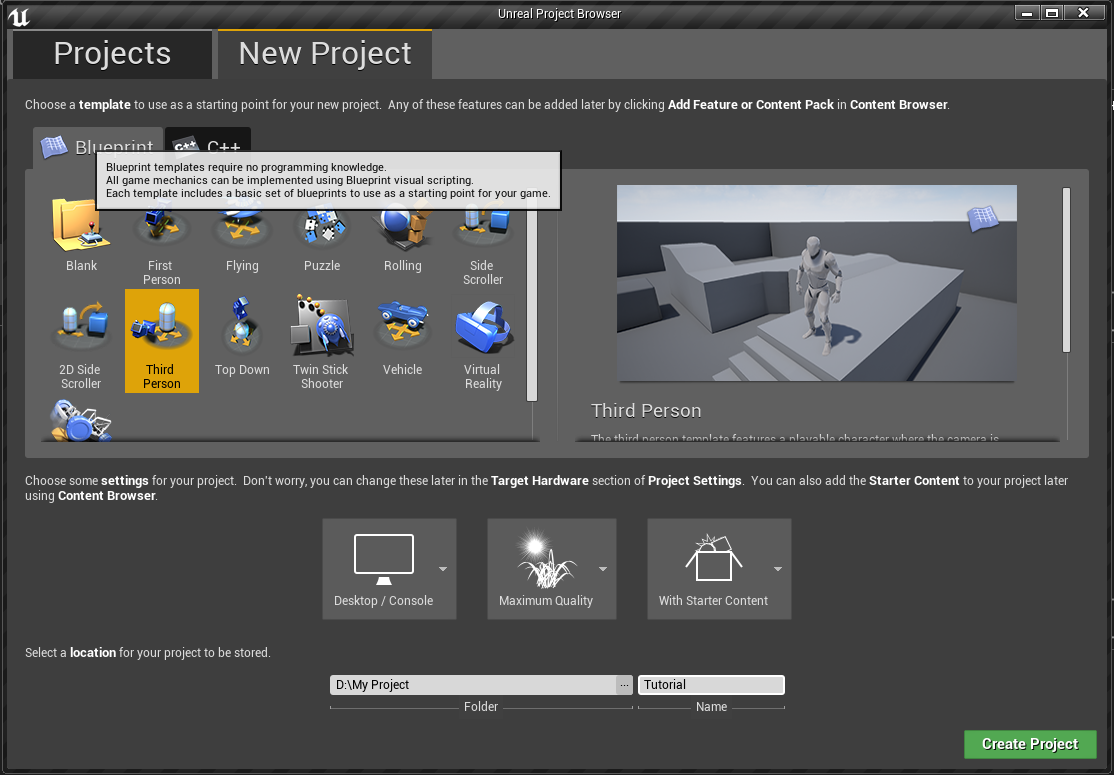


Рис. 2.3. Выбор места хранения

После установки движка, его можно запускать и создавать свой первый проект.

**2.2 Создание проекта и его первоначальная настройка**

Чтобы начать разрабатывать проект, его нужно создать и настроить. После запуска программы нужно выбрать среду программирования (1), начальный шаблон (2), тип платформы (3), качество графической части (4), начальный контент (5), место хранения (6). Для своего проекта я выбрал шаблон игры от третьего лица (см. рис. 2.4).



(4)

(3)

(5)

(6)

(2)

(1)

Рис. 2.4. Настройка проекта

Шаблон игры от третьего лица и включенный начальный контент позволит ознакомиться с игровым движком. Этот шаблон включает в себя небольшую карту и набор анимации для манекена, а начальный контент дает доступ к стандартным объектам, таким как стены, двери, небольшое количество материалов.

## **2.3 Помещение на уровень игрового персонажа и привязка клавиш перемещения**

Так как шаблон дает манекен и набор анимаций, то его нужно использовать и сделать так, чтобы за него можно было ходить, бегать и прыгать по доступной локации. Для этого из Браузера содержимого нужно перенести Blueprint персонажа на уровень (см. рис. 2.5).

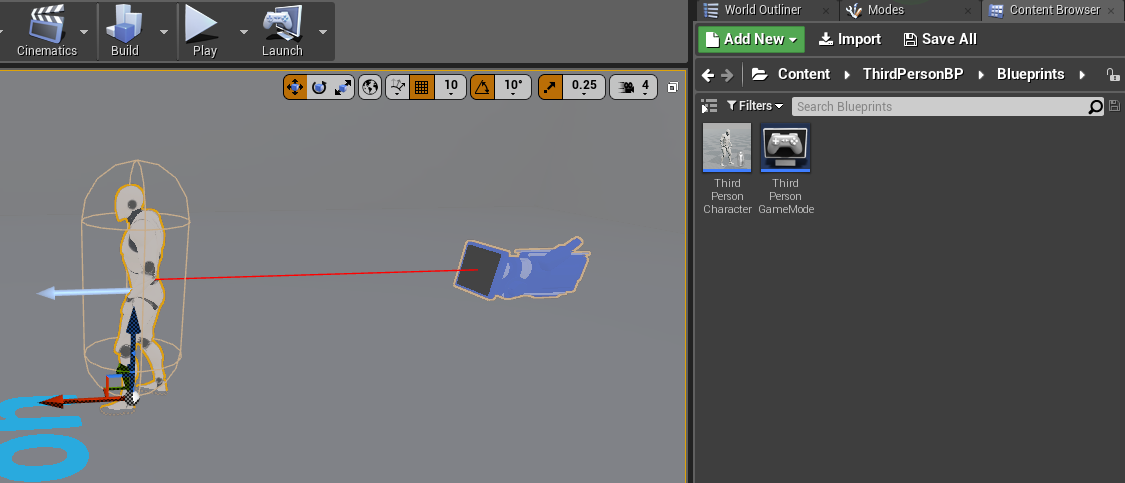


Рис. 2.5. Помещение персонажа на уровень

После этого нужно зайти в настройки проекта, раздел «Клавиши Ввода». Перейти в раздел перемещения и ввести нужные клавиши для определенных векторов передвижения. «Движение вперед» – клавиша W для движения вперед и S для движения назад. «Движение в сторону» – клавиша А для движения влево и клавиша D для движения вправо. Так же нужно настроить осмотр мышкой и добавить для неё сетки перемещения по X и Y, для этого настраиваем другие разделы, такие как «Поворот» и «Взгляд Вверх» (см. рис. 2.6).



Рис. 2.6. Настройка клавиш перемещения

После привязки клавиш, нужно перейти в Blueprint персонажа и создать небольшую логику. Для этого двойным кликом открываем Blueprint персонажа от третьего лица. В новом окне откроется Граф для написания логики, этим и отличается режим программирования Blueprint от C++ в нём всё наглядно, и все функции сразу видны и ими легко управлять. Так же такой способ программирования очень легкий для новичков или дизайнеров, которым главное простота управления логикой.

Итак, в новом окне будет пустой граф, для того чтобы прописать любую логику нужно добавить Ноды. Ноды – это функции, события, переменные или любой другой предмет программирования в среде Blueprint. Для начала настроим ввод с мыши, для этого нужно добавить событие, которое ссылается на привязанную до этого сетку перемещения мыши. Нажав правую кнопку мыши, откроется окно выбора всех доступных Нод. В поиске нужно набрать слово «Turn» и тогда будут написаны все доступные Ноды с этим словом, нам нужна Нода, которая находится в категории «Axis Input» и отвечает за ввод по оси X. После выбора, она появится на графе, так же нам нужна Нода, которая отвечает за ввод по оси X, в нашем случае это Нода LookUp (см. рис. 2.7). Но этих событий будет недостаточно, их нужно соединить с функциями, которые считывают поведение мыши. Для этого нужно от выхода Ноды Turn провести узел и выбрать функцию Add Controller Yaw Input (см. рис. 2.8), а от Ноды LookUp вывести функцию Add Controller Pitch Input (см. рис. 2.9).



Рис. 2.7. Добавленные Ноды событий

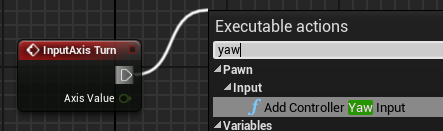


Рис. 2.8. Привязка мыши по сетке Х

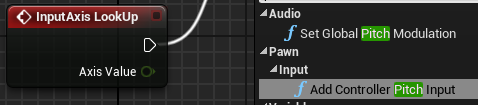
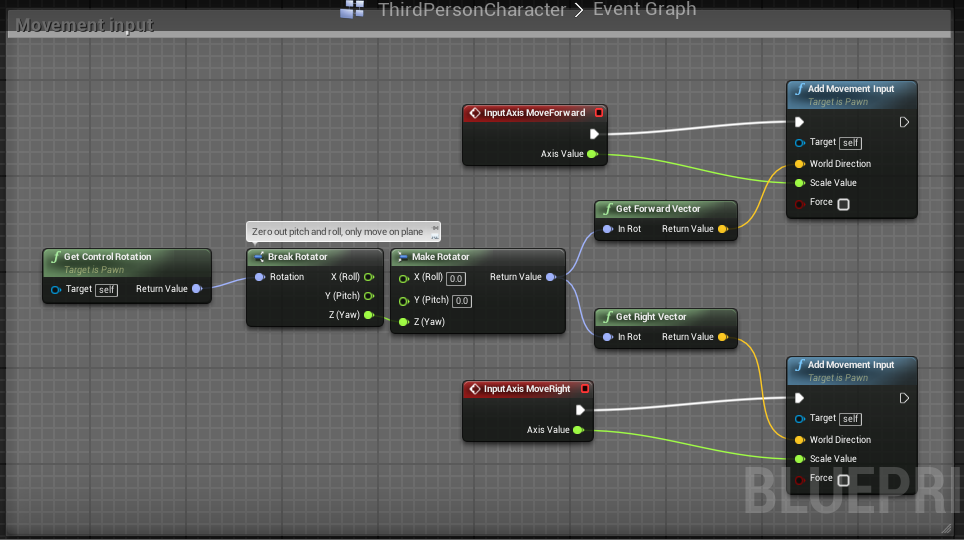


Рис. 2.9. Привязка мыши по сетке Y

Далее нужно привязать клавиши передвижения. Привязка клавиш перемещения немного сложнее привязки мыши, но итог будет таким (см. рис. 2.10). Для того, чтобы привязать клавиши перемещения нужно добавить Ноду «Get Control Rotation» (1) – она отвечает за то, чтобы взять функцию вращения манекена, далее идет Нода «Break Rotator» и «Make Rotator» (2) – она нужна чтобы разбить значения вращения и взять только вектор по сетке Z, так как игра разрабатывается в трехмерном пространстве. Далее идут Ноды «Get Forward Vector» и «Get Right Vector» (3), они нужны для взятия векторов перемещения. Ноды событий Input Axis (4), привязаны к нашим клавишам. И конечная Нода «Add Movement Input» (5) добавляют к клавишам перемещения вектора, взятые до этого.



(5)

(4)

(3)

(2)

(1)

Рис. 2.10. Привязка клавиш перемещения

После написания этой логики, персонажем можно побегать по готовому уровню.

## **2.4 Создание класса Blueprint Actor Component для отслеживания статистики персонажа**

Создание личного класса Blueprint Actor Component нужно для подключения логики к любому актёру. Этот компонент нужен для того, чтобы упростить написание и использование логики для актёров одного класса. Например, данный компонент будет отслеживать количество жизни персонажа, эту логику можно написать и в самом персонаже, но если похожих актёров много, то придётся писать логику для каждого актёра. Таким образом, этот компонент уменьшит длительность разработки.

Для того чтобы создать этот компонент нужно нажать правой клавишей в контент браузере и выбрать «создать Blueprint Class» в списке самых часто используемых компонентов выбрать Actor Component, набрать имя компонента и открыть его (см. рис. 2.11).



Рис. 2.11. Выбор компонента из списка

В новом окне будет стандартный граф событий с двумя изначальными функциями это Event Begin Play и Event Tick. Чтобы отслеживать состояние здоровья персонажа, нужно создать переменную Health(Float), Max Health(Float), Regeneration(Boolean), Regeneration Speed(Float), No Health(Boolean) и Auto Regeneration(Boolean), эти переменные создаются во вкладке «переменные» при нажатии клавиши «+» (см. рис. 2.12).

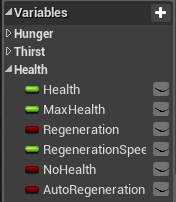


Рис. 2.12. Создание переменных

Для того, чтобы эти переменные изменялись нужно создать функцию «Change Health» (см. рис. 2.13).

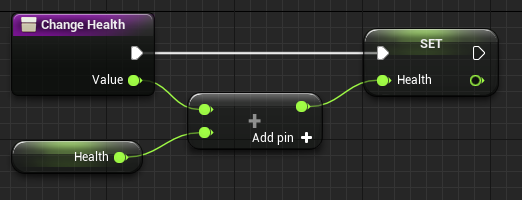


Рис. 2.13. Создание функции

Далее нужно создать логику, которая будет отображать эту переменную во время тестов. Для этого нужно создать функцию «Display Stats» из которой вывести Ноду «Print String» и при помощи Ноды «Append» привязать переменную (см. рис. 2.14).



Рис. 2.14. Функция «Display Stats»

Для того, чтобы эта переменная изменялась, то есть наносился урон персонажу, нужно на уровень добавить триггер «Pain Causing Volume», при пересечении его персонажем, ему будет наноситься урон, но для этого нужно написать логику, чтобы этот урон засчитывался и изменял переменную (см. рис. 2.15).

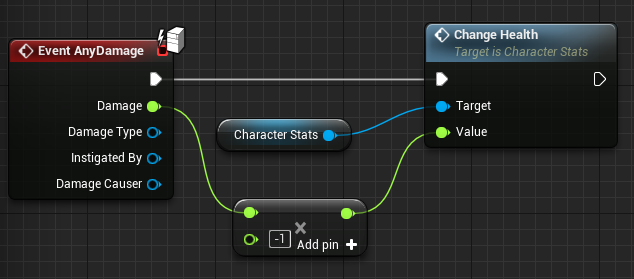


Рис. 2.15. Событие запоминания урона

Для восстановления здоровья нужно написать функцию «Health Regeneration», данная функция будет включать регенерацию здоровья, как только его значение упадет ниже 30, так же оно выключится, как только здоровье будет равно 100 (см. рис. 2.16).

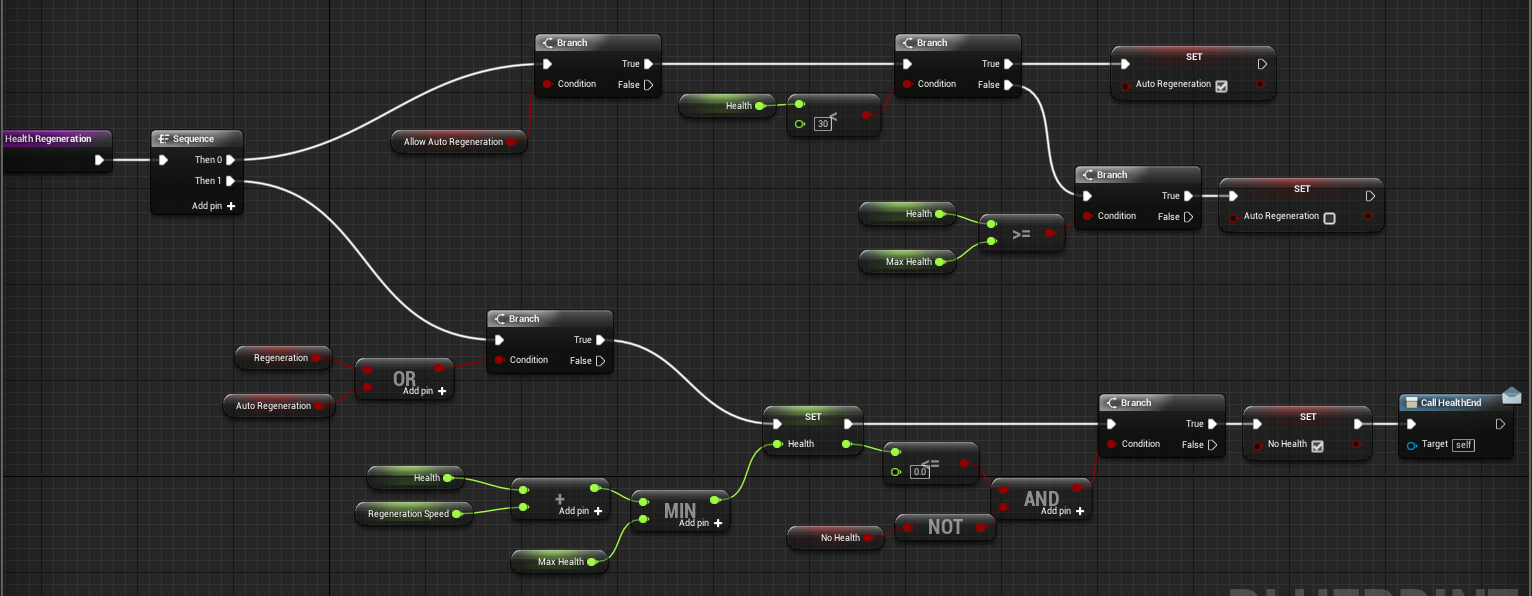


Рис. 2.16. Функция регенерации

## **2.5 Смена положения камеры**

Чтобы сменить положение камеры, нужно добавить её к мешу актёра и настроить её положение (рис. см. 2.17).

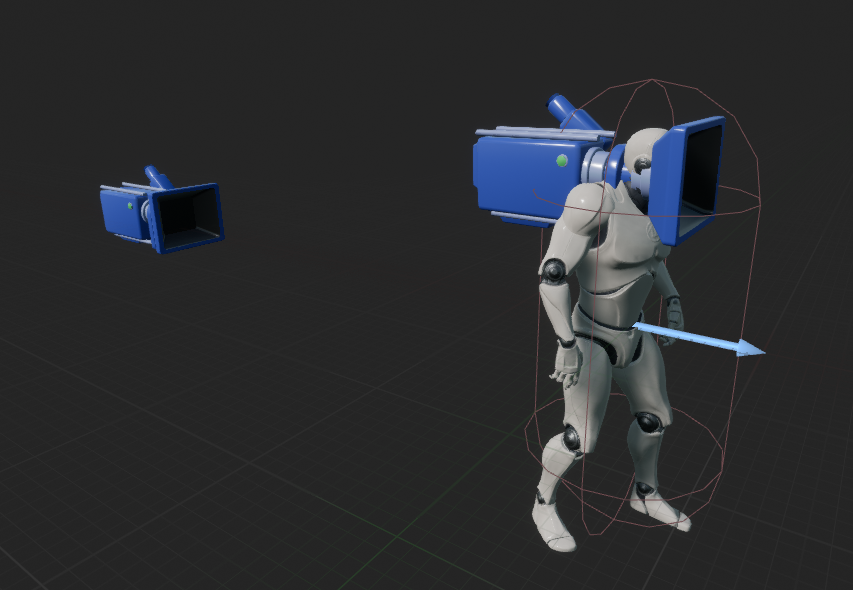


Рис. 2.17. Добавление камеры

Далее нужно настроить камеру так, чтобы при перемещении мышкой положение персонажа менялось от положения мыши. Для этого нужно в настройках камеры поставить галочку (см. рис. 2.18).

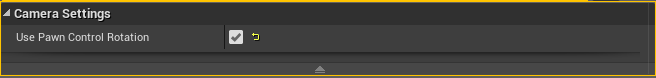


Рис. 2.18. Настройка камеры

Для того, чтобы менять положение камеры при нажатии на кнопку, нужно создать функцию и написать логику в графике персонажа, так же привязать клавишу с клавиатуры (см. рис. 2.19, 2.20).

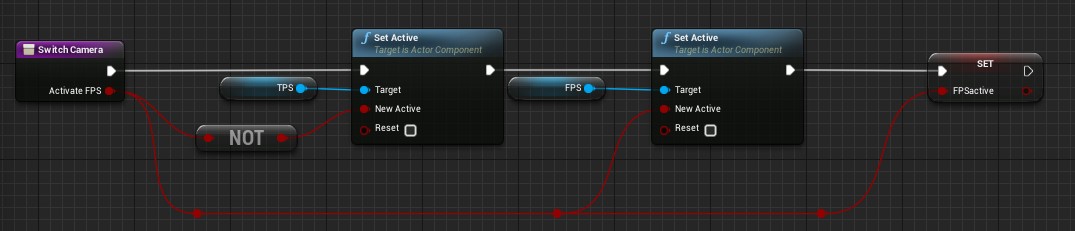


Рис. 2.19. Логика для смены положения камеры

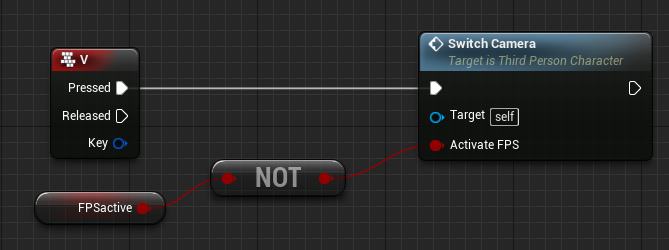


Рис. 2.20. Привязка клавиши для смены камеры

## **2.6 Добавление функционирующих дверей**

Для того чтобы создать функционирующую дверь для начала её нужно добавить на уровень, так же нужно добавить Box Trigger (см. рис. 2.21).

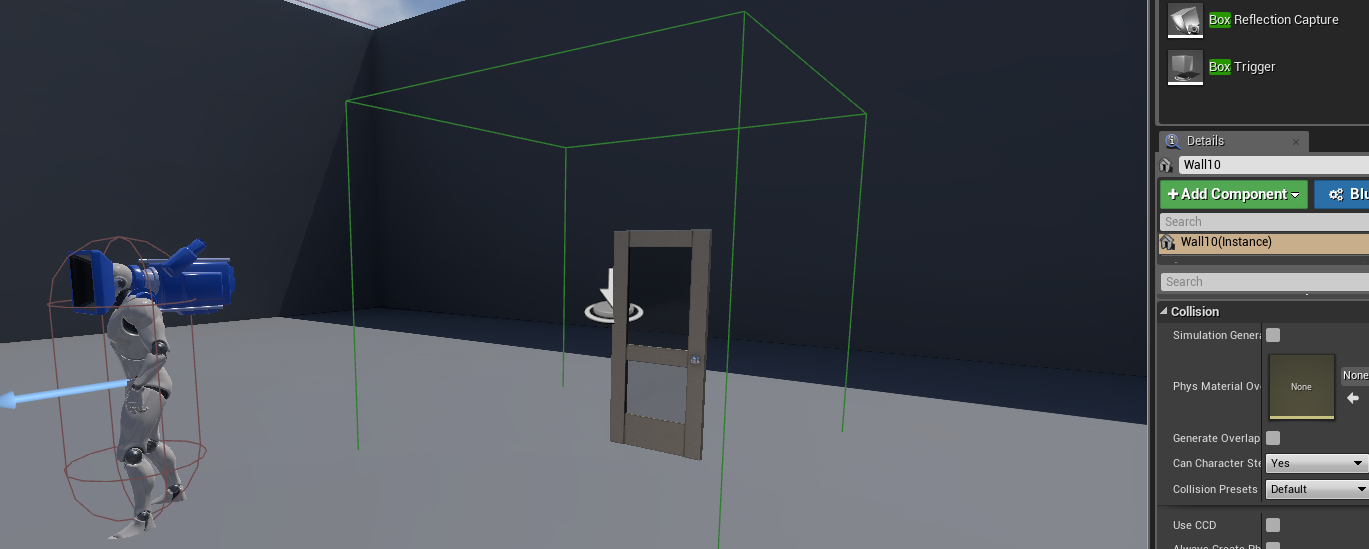


Рис. 2.21. Перемещение на уровень триггера и двери

Далее нужно зайти в редактор Matinee, он позволит редактировать положение двери по линии времени. В редакторе нужно на линию времени добавить актёр двери и назначить первую точку, далее вернуться в редактор, повернуть дверь на 120 градусов и поставить вторую точку (см. рис. 2.22).

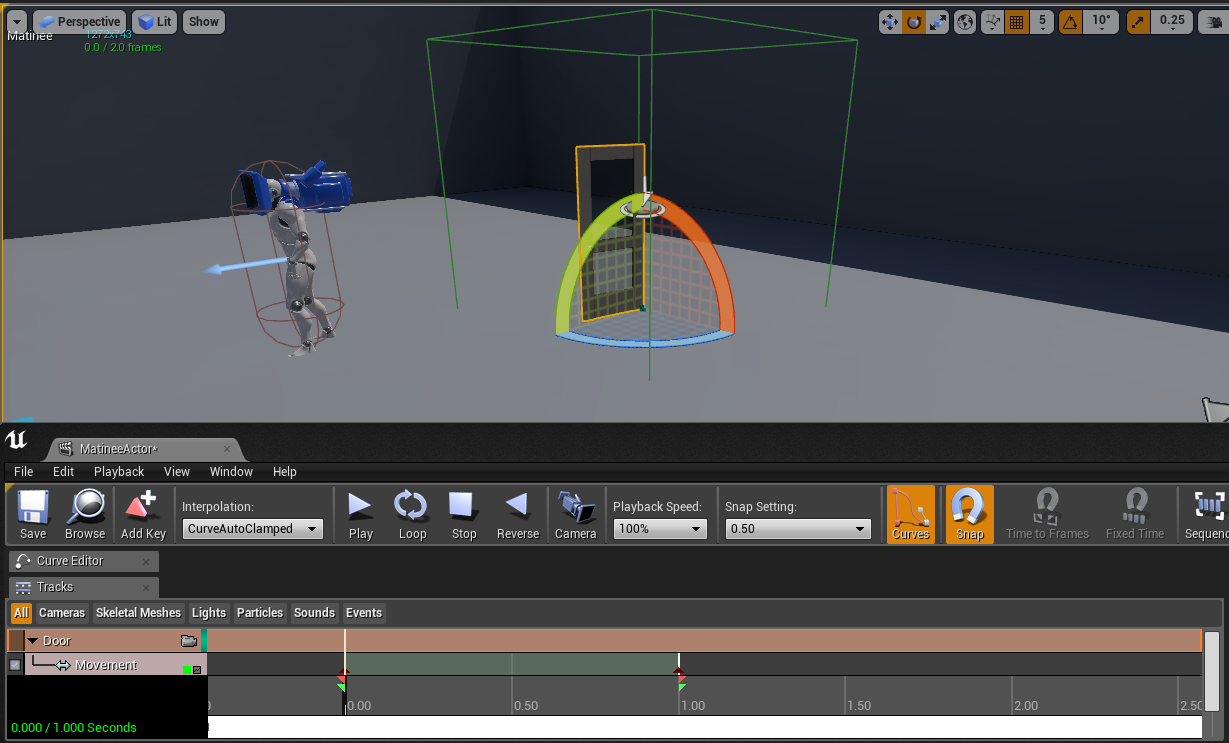


Рис. 2.22. Перемещение двери по линии времени в редакторе Matinee

Далее, чтобы дверь открывалась при пересечении триггера и нажатии клавиши нужно написать логику (см. рис. 2.23).

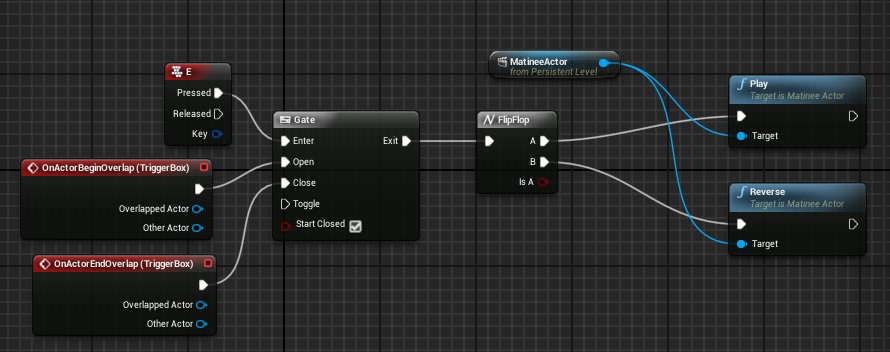


Рис. 2.23. Логика для открытия двери

## **2.7 Активация Ragdoll при смерти персонажа**

Чтобы было невозможно управлять персонажем после того, как закончатся жизни, нужно включить Ragdoll персонажа и сменить камеру на третье лицо. Для этого нужно создать функцию и прописать логику (см. рис. 2.24, 2.25).

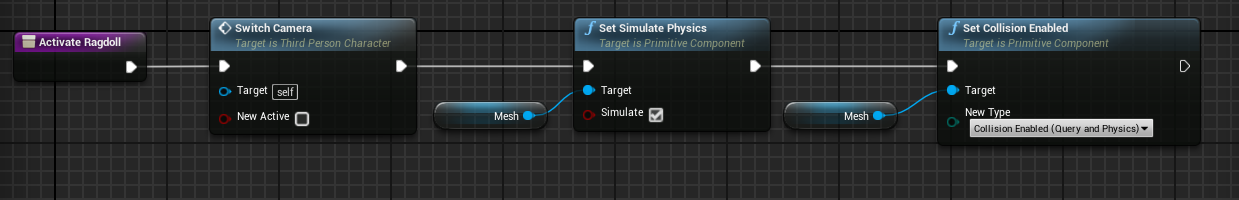


Рис. 2.24. Функция Ragdoll

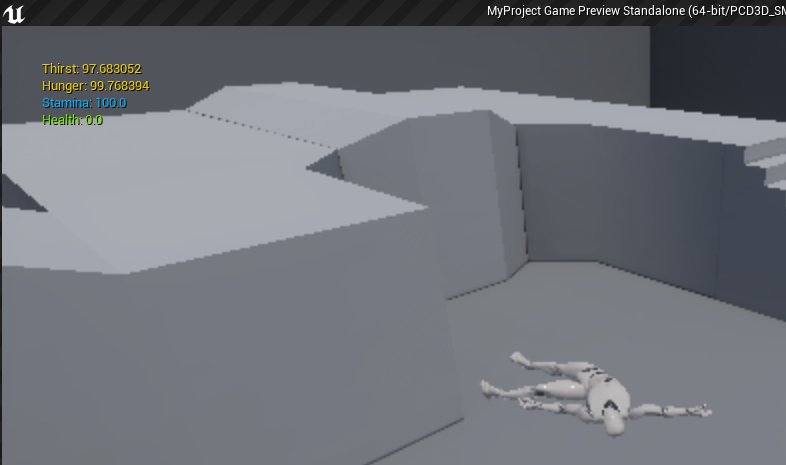


Рис. 2.25. Включение Ragdoll, когда закончились жизни персонажа

## **2.8 Добавление GUI**

Благодаря встроенному редактору можно нарисовать простой графический интерфейс, на данном интерфейсе отображены потребности и статистика персонажа, такие как Здоровье, Голод, Жажда и Выносливость (см. рис. 2.26).

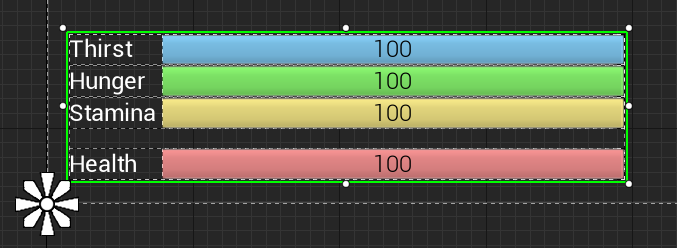


Рис. 2.26. Графический интерфейс

## **2.9 Итог**

На данный момент разработана система перемещения, смена камеры, несколько анимаций, простой графический интерфейс, инвентарь (см. рис. 2.27). В дальнейшем будет добавлено строительство, простые автомобили, враждебные боты и дружественные с системой диалогов и торговли.

Рис. 2.27. Экспортированный меш персонажа из программы Fuse

# **заключение**

Во время разработки своего проекта я изучил теоретический материал по теме презентации в маркетинге, а также узнал больше о игровом движке Unreal Engine 4. Узнал, что каждый объект размещенный на сцене называется актёром, что актёры – это класс, зависимый от встроенного графического языка программирования, который разработан на языке С++ для более удобной работы с движком Unreal Engine 4.

Blueprint – позволяет выстраивать логику для абсолютно любого актера, из-за гибкости системы и разнообразия классов графического языка программирования. В своем проекте я использовал несколько различных классов Blueprint для создания логики перемещения, взаимодействие с триггерами в игровом мире, создания статистики, смены камеры и включения Ragdoll. Так же разместил объекты на уровне и прописал логику для взаимодействия с ними. После этого занялся графической частью, чтобы техническая демоверсия выглядела более привлекательно. С сайта Tirbosquid скачал бесплатные объекты и расставил на уровне. Написал логику для выключателей, дверей, инвентаря, графического интерфейса. Создал персонажа в программе Fuse Maximo, привязал анимированный скелет на сайте Maximo и экспортировал в Unreal Engine 4.

# **список литературы**

1. Джейсон Грегори Игровые движки – 2014. – 1052 с.
2. Ламот А. Программирование трехмерных игр для Windows. Советы профессионала по трехмерной графике – 2015. – 1407 с.
3. Роман Сакутин Разработка игр для начинающих – 2016. – 50 с.
4. Роман Сакутин Обучение программированию на C# – 2015. – 268 с.
5. Роман Сакутин Создание игры без программирования – 2016. – 164 с.
6. Трэйси Фуелтон Гейм-дизайн – 2012. – 496 с.

Электронные ресурсы:

Официальная документация от Epic Games [Электронный ресурс] // URL: <https://docs.unrealengine.com/latest/INT/>

Официальные видео уроки от Epic Games [Электронный ресурс] // URL: <https://docs.unrealengine.com/latest/INT/Videos/>

Видео уроки по созданию Survival Shooter [Электронный ресурс] // URL: <https://www.youtube.com/channel/UCLbkGIcYJxxL0tciH9RVebg>