ВВОДНЫЙ КУРС ПО РАЗРАБОТКЕ НА UNREAL ENGINE 4-5

Магнитогорск 2024

СОДЕРЖАНИЕ

1 Создание Примитивного Проекта с Нуля

1.1 Шаг 1. Project Browser

Первым делом для создания проекта необходимо открыть Unreal Project Browser.

Unreal Project Browser представляет собой конфигуратор для создания новых проектов. В нем вы можете увидеть различные шаблоны (например, шутер от первого лица, игра с видом сверху и другие).

Мы выберем пустой (Blank) шаблон с такими конфигурациями:

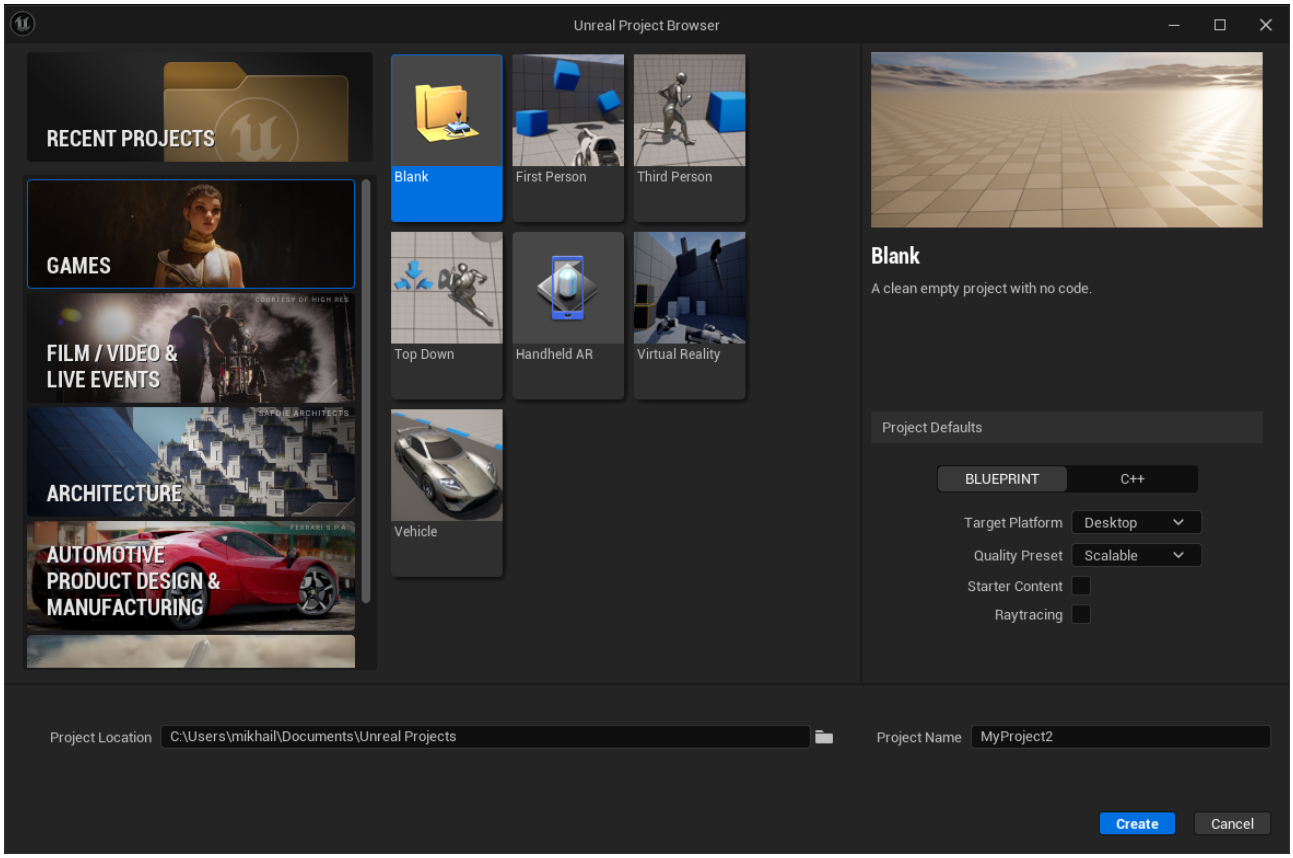

Рисунок 1 Интерфейс Unreal Project Browser с выбранным пустым шаблоном

1.2 Шаг 2. Настройка интерфейса и создание уровня

Для начала установим классический интерфейс редактора через верхнее меню **Window → Layout → UE4 Classic Layout**.

На старте нам дается уровень с большой картой. Давайте создадим новый уровень с упрощенной картой.

В верхнем меню переходим в **File → New Level → Basic** и нажимаем **Create**. Теперь нужно сохранить этот уровень в папку проекта с помощью сочетания клавиш

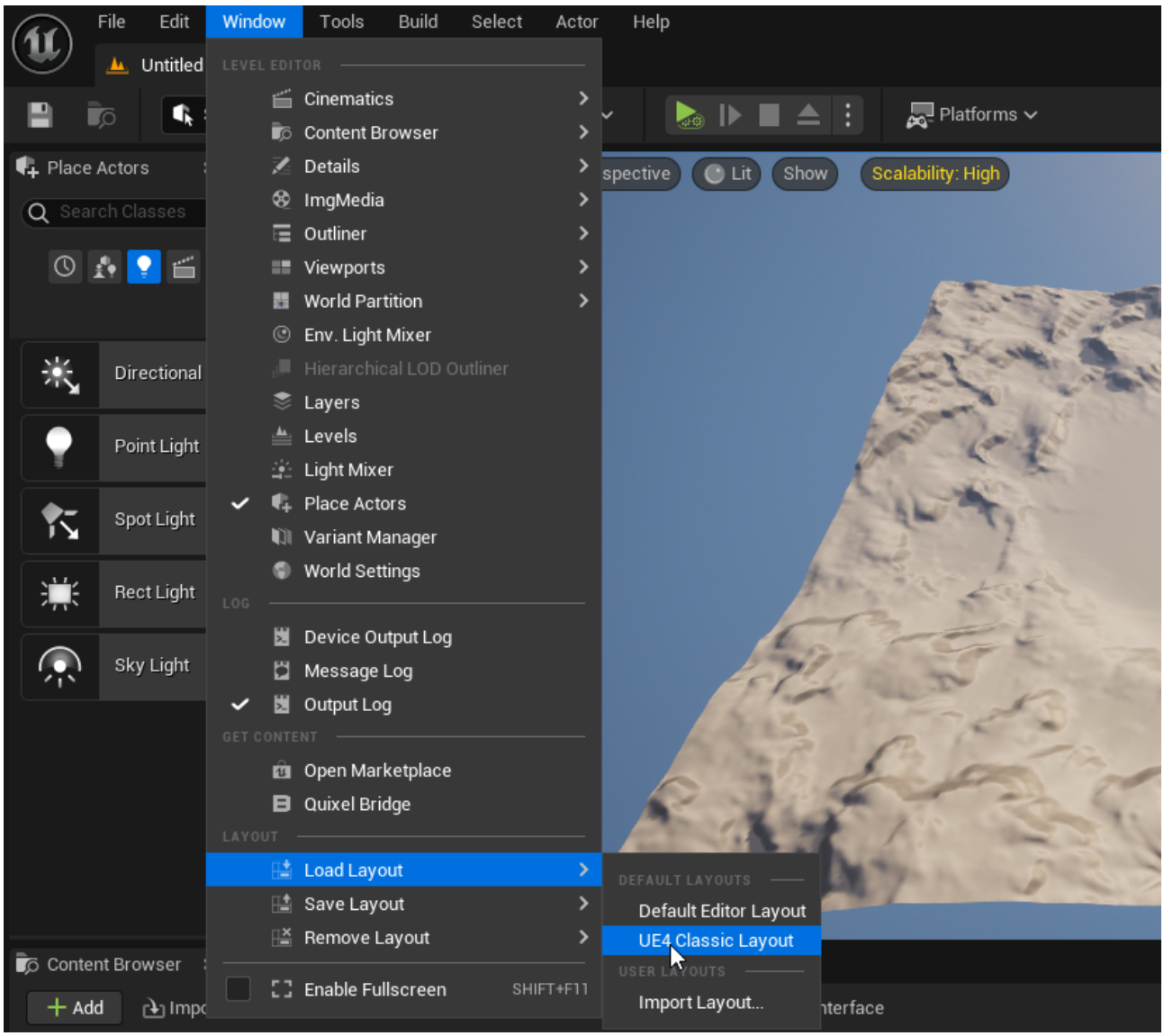

Рисунок 2 Установка классического интерфейса редактора

Ctrl+S. Для удобства лучше выделить отдельную папку **Levels** и сохранить новый уровень туда с префиксом **L_**.

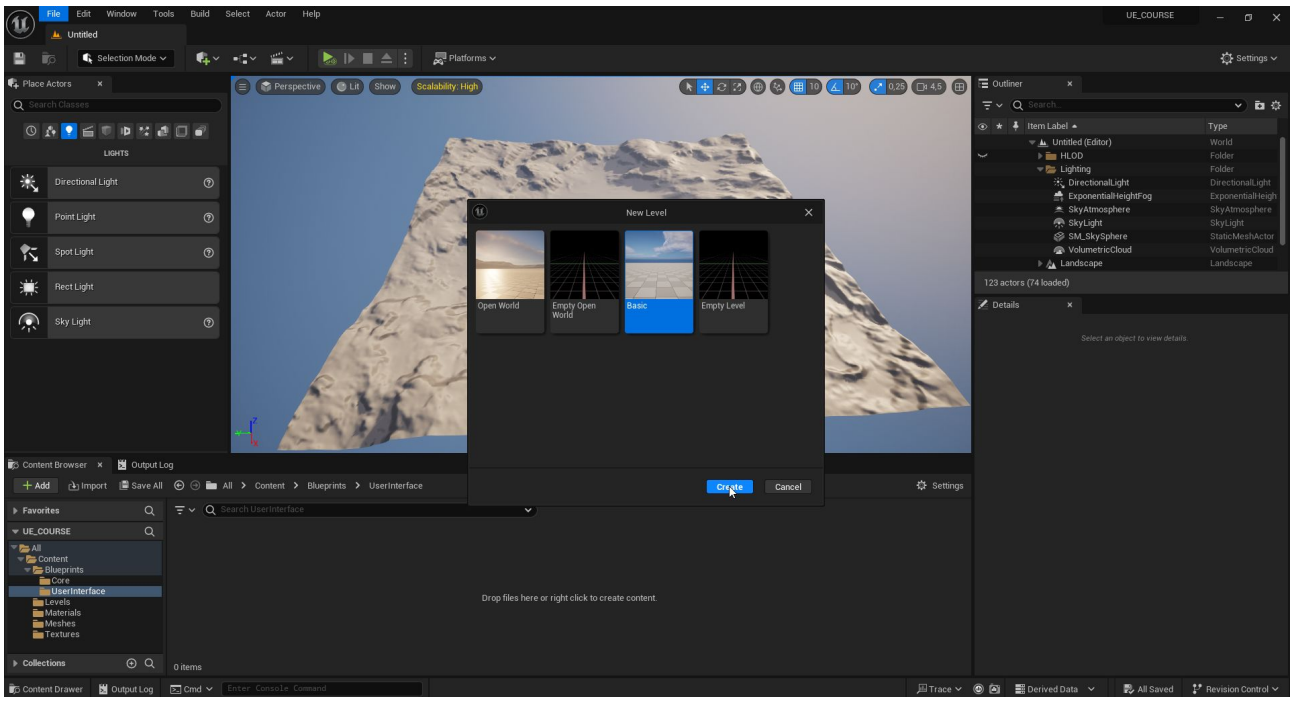

Рисунок 3 Создание и сохранение нового уровня

1.3 Шаг 3. Работа с базовыми объектами

В классическом отображении интерфейса слева можно увидеть панель **Place Actors**.

Actor — это любой объект, который может быть размещен на уровне (**Level**).

Actor состоит из компонентов и может управлять их поведением.

Components (компоненты) — это объекты с определенной функциональностью, которые можно добавить к **Actor**, чтобы расширить его возможности.

Unreal Engine поддерживает вложенно-агрегированную[1] структуру, позволяя подключать компоненты к actor'ам, изменяя функционал и поведение actor'ов.

Примером такой структуры является отношение уровня к акторам, размещенным в нем. Уровень может существовать независимо от actor'ов, размещенных на нем.

Для примера разместим на уровне куб (Cube) из меню **Place Actors**. Теперь, используя горячие клавиши (**W, E, R**) или кнопки в окне **Viewport**, можно переключаться между режимами изменения трансформации (**Transform**) объекта.

Transform — это структура, содержащая три вектора (**Location** — расположение, **Rotation** — поворот, **Scale** — масштаб), которые определяют положение и трансформацию объекта в пространстве.

Для изменения других свойств актера можно воспользоваться меню **Details**.

Details — здесь собраны параметры актера и его компонентов, размещенного на

[1] Агрегация — это способ организации объектов, при котором вложенные объекты и основной объект могут существовать независимо друг от друга.

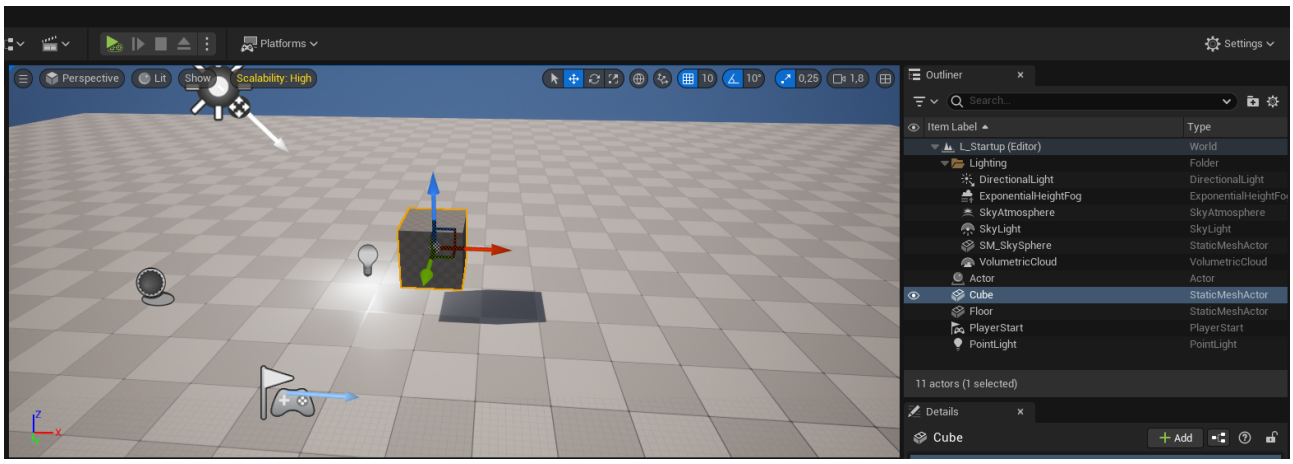

Рисунок 4 Пример использования агрегированной структуры с объектами на уровне

сцене.

Поскольку у компонента **StaticMeshComponent** есть возможность включения физики, включим её и поднимем куб над полом. Теперь запустим проект в режиме **Simulate**.

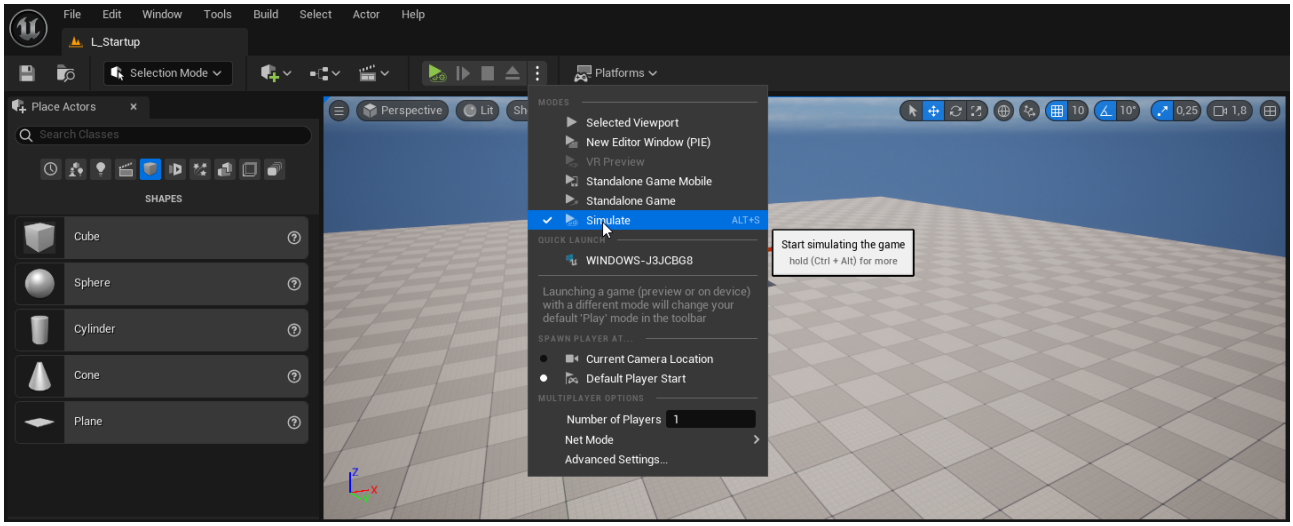

Рисунок 5 Запуск проекта в режиме **Simulate**

1.4 Шаг 4. Режимы запуска проекта

Для тестирования и отладки проекта существует несколько вариантов запуска игры:

- **Selected Viewport** — классический запуск игры в окне Viewport.
- **New Editor Window** — запуск игры в отдельном окне редактора.
- **VR Preview** — запуск игры в режиме виртуальной реальности (если подключена VR-гарнитура).
- **Standalone Game или Mobile** — запуск в отдельном процессе, имитирующем реальное устройство.

- **Simulate** — запуск в режиме наблюдателя, когда вы видите физическое и логическое поведение объектов.

1.5 Шаг 5. Базовая логика и Level Blueprint

Для реализации базовой логики и взаимодействия объектов на уровне используется **Level Blueprint**. Это визуальный скриптовый инструмент, позволяющий создавать логику без написания кода, связывая различные объекты и их события.

Прежде чем открыть Level Blueprint, стоит напомнить о том, что было сказано ранее. Поскольку структура игрового движка основана на логике вложенности и взаимодополнения, эта парадигма сохраняется и для логики. Каждый элемент в движке имеет свой код и может влиять на другие элементы игрового и неигрового окружения.

Когда мы говорили об акторах (**Actors**) и их компонентах, мы рассматривали их как отдельные логические единицы, каждая из которых содержит свою логику и может дополнять другие. Но как тогда воспринимать уровень (**Level**) — как актор или как компонент?

На данном этапе, не углубляясь в ООП[2], можно сказать, что уровень является вершиной иерархии игровых объектов. Level содержит акторов, а акторы, в свою очередь, содержат компоненты.

Таким образом, как высший элемент объектной иерархии, Level получает возможность управлять всеми actor‘ами, размещенными в нем.

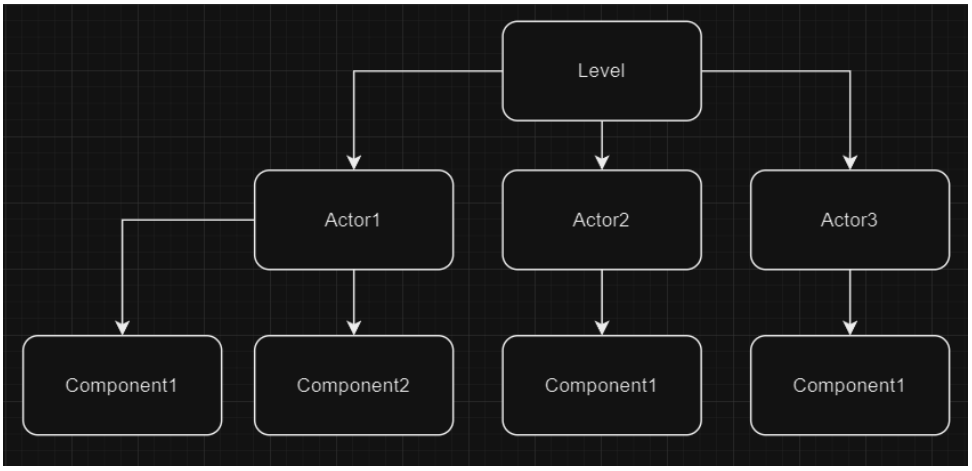

Рисунок 6 Структура вложенности уровня

После всего вышесказанноро можем перейти к **Level Blueprint**.

[2]ООП (Объектно-ориентированное программирование) — это парадигма программирования, в основе которой лежит взаимодействие объектов (сущностей), наделенных набором свойств и методов.

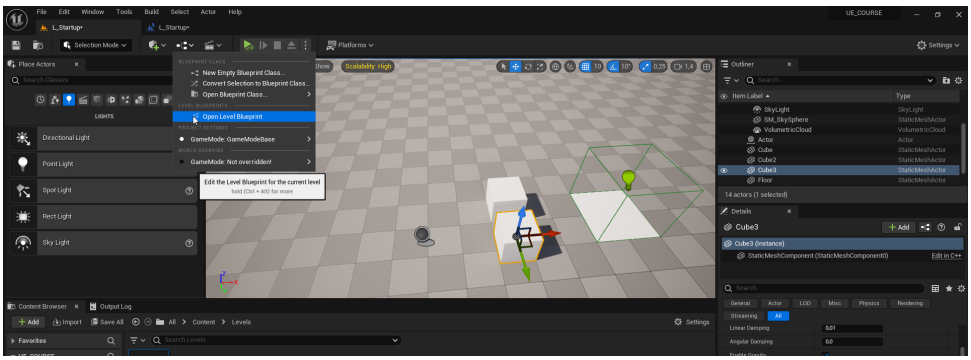

Рисунок 7 Level Blueprint

1.5.1 Редактор Blueprints

Blueprint (Чертеж) — это шаблон, содержащий скриптовую логику, описывающую поведение объектов.

Редактор Blueprints представляет собой редактор кода с визуальным программированием. В нем можно создавать функции, переменные и события (**Events**). Его функционал позволяет описывать логику объекта, компилировать код и отлаживать программу.

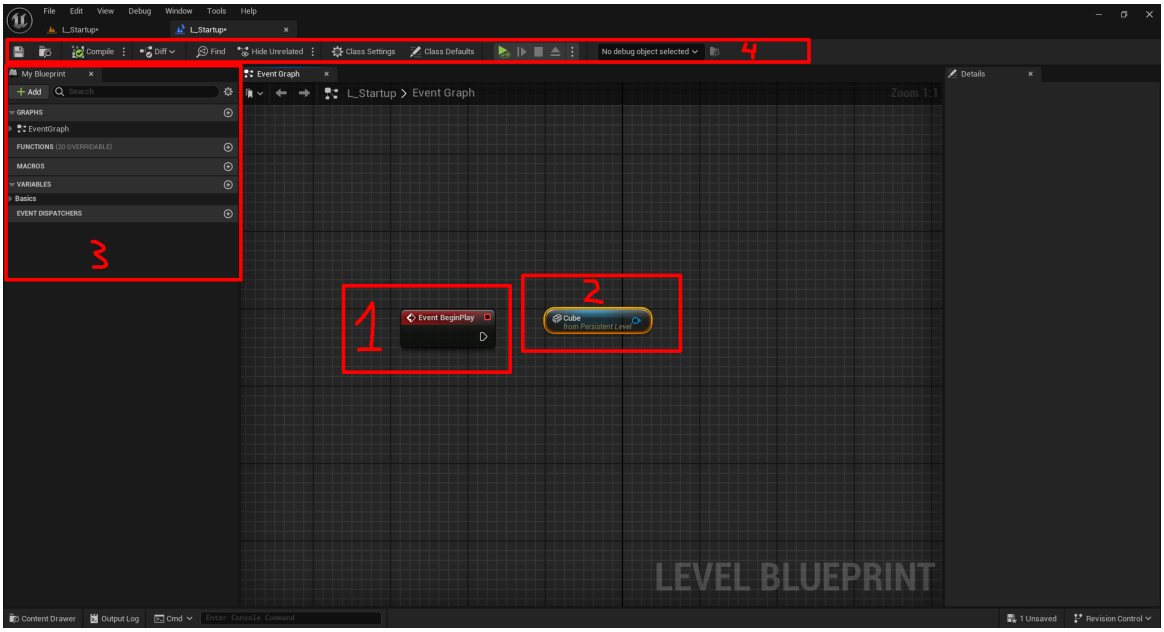

Рисунок 8 (1. BeginPlay событие начала игры; 2. Ссылка на куб; 3. Панель MyBlueprint - здесь отображаются основные элементы кода; 4. Панель управления - здесь можно компилировать, отлаживать, и запускать код.)

Панель **MyBlueprint** отражает основные элементы кода, переменные и функции. Также в нем находятся **Графы (Graphs)**, которые являются окружением для написания кода.

Сверху находится панель инструментов, содержащая основные действующие кнопки: компиляция, сохранение и отладка (**debug**).

Event (Событие) — это элемент кода, который вызывается actor'ами или внешними системными объектами движка.

Сравнение событий Unreal Engine с работой предприятия:

1. Старт предприятия → запуск движка (вызов всех конструкторов).
2. Все работники на своих местах и готовы приступить → завершение инициализации.
3. Звонок о начале работы предприятия → запуск игры (вызов **Begin Play**).
4. Пока предприятие работает, подгонять работников как можно чаще → работа события **Tick**.
5. Завершение работы предприятия → вызов события **EndPlay**.

В редакторе **Level Blueprint** мы можем обращаться к любым объектам на уровне, менять их свойства и взаимодействовать с ними.

Давайте обратимся к ранее созданному кубу на уровне, отключим физику и выделим его. С выделенным кубом перейдем в редактор **Level Blueprints**. Правой кнопкой мыши нажмем по пустому полю. В выпадающем меню выберем **Create a Reference to Cube**.

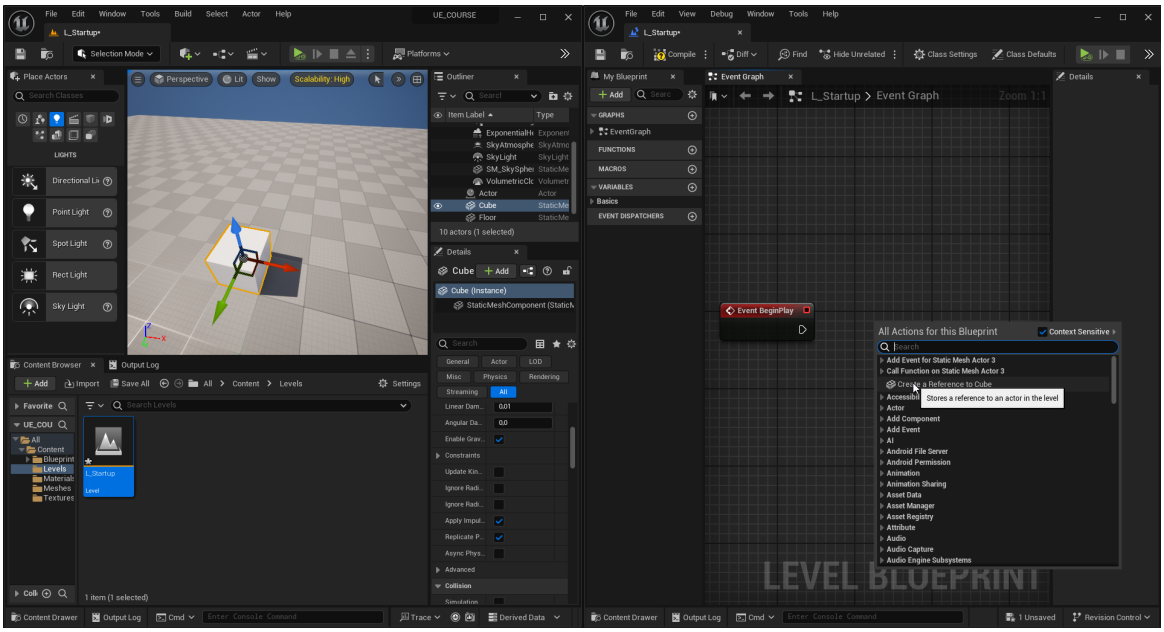

Рисунок 9 Создание ссылки на куб

Reference (Ссылка) — это переменная-ссылка, содержащая доступ к объекту, находящемуся по указанному адресу.

С помощью ссылок на объекты мы можем изменять сам объект и его свойства.

К примеру, уничтожим этот объект с помощью метода **Destroy** с самого начала игры. Для того чтобы обращаться к методам объекта, нужно потянуть за ссылку и отпустить в пустом поле, чтобы появилось контекстное меню. Далее введем в поиске **Destroy Actor** и нажмем **Enter**.

Теперь остается только вызвать этот метод в нужный момент, а именно при вызове события **EventBeginPlay**. Протянем сигнальный узел от пина **EventBeginPlay** к пину **DestroyActor**. Скомпилируем скрипт с помощью кнопки **Compile**. После запуска в режиме **Simulate** куб уничтожится.

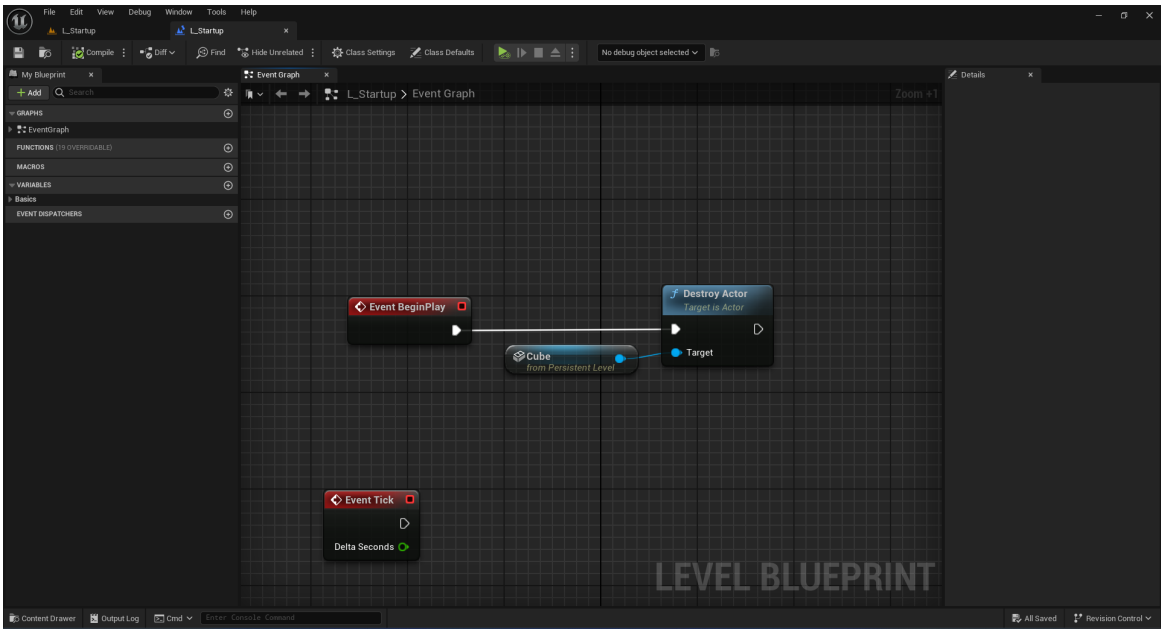

Рисунок 10 Скрипт уничтожения куба