РАЗРАБОТКА **ИНТЕРАКТИВНОЙ ВИЗУАЛИЗАЦИИ, АНИМАЦИИ** И **РЕНДЕРИНГА** С ИСПОЛЬЗОВАНИЕМ UE4

EPIC GAMES

UNREAL ENGINE

UNREAL® ENGINE 4 для ДИЗАЙНА и ВИЗУАЛИЗАЦИИ

Том **ШЭННОН**

Том **ШЭННОН**

UNREAL® ENGINE 4 для ДИЗАЙНА и ВИЗУАЛИЗАЦИИ

БОМБОРА™
Москва 2021

ОГЛАВЛЕНИЕ

Предисловие 11

Для кого эта книга? 11
Как организована эта книга? 12
Файлы ресурсов 15
Обозначения, используемые в этой книге 16

Благодарности 17

Об авторе 18

Часть I Обзор Unreal Engine 4 19

Глава 1 Начало работы с Unreal Engine 4 21

Что такое Unreal Engine 4? 22
Краткая история Unreal Engine 23
Представление Unreal Engine 4 24
Основные моменты UE4 для визуализации 25
Разработка интерактивной визуализации с Unreal Engine 4 26
Требования к разработке Unreal Engine 4 33
Командная работа в Unreal Engine 4 35
Стоимость разработки на UE4 36
Экономия затрат UE4 38
Ресурсы и обучение 39
Заключение 41

Глава 2 Работа с Unreal Engine 4 43

Компоненты Unreal Engine 4 44
Структура проекта 50
Строение файлов .uasset 52
Организация производственного процесса в Unreal Engine 4 . . 53
Заключение 56

Глава 3 Производственный процесс . . . 57
Обзор производственного процесса . . . 58
Настройка 3D-сцены . . . 60
Подготовка геометрии к UE4 . . . 61
Конвейер работы с FBX-мешами . . . 72
Процесс работы с текстурами и материалами . . . 74
Импорт в Content Library . . . 77
Процесс работы с камерой . . . 80
Заключение . . . 81

Глава 4 Освещение и рендеринг . . . 83
Понимание физически корректного рендеринга UE4 (PBR) . . . 84
Освещение в UE4 . . . 87
Понимание подвижности света (параметр Mobility) . . . 88
Отражения в реальном времени . . . 93
Post-Processing (постобработка) . . . 94
Заключение . . . 102

Глава 5 Материалы . . . 103
Обзор материалов . . . 104
UE4 Material Editor . . . 105
Как работают материалы в Unreal Engine 4 . . . 108
Типы поверхностей . . . 113
Экземпляры материалов . . . 114
Простой материал . . . 117
Заключение . . . 124

Глава 6 Blueprints . . . 125
Введение в Blueprints . . . 126
Объекты, класс и акторы . . . 127
Игрок . . . 128
Контроллер игрока . . . 128
Pawns (павны) . . . 130
Мир . . . 131
Уровни . . . 131
Компонеты . . . 132
Переменные и их типы . . . 132
Tick . . . 133
Наследование классов . . . 134

Spawning и Destroying (спавн и уничтожение) 134
Взаимодействие между Blueprints 136
Компиляция скрипта 137
Заключение 138

Часть 2 Ваш первый проект на UE4 139

Глава 7 Настройка проекта 141
Объем проекта 142
Создание нового проекта из Launcher 142
Заключение 146

Глава 8 Заполнение мира 147
Создание и сохранение нового пустого уровня 148
Расстановка и модификация ассетов 149
Да будет свет 152
Перемещение по сцене 155
Создание архитектуры 155
Добавление деталей к структуре 157
Заключение 160

Глава 9 Создание интерактивности с Blueprints 161
Настройка проекта 162
Нажатие кнопки Play 162
Создание Pawn 164
Привязка ввода данных 168
Создание Player Controller Class 170
Добавление Input с Blueprints 170
Поворот взгляда 172
Передвижение игрока 173
GameMode 178
Размещение Player Start Actor 181
Заключение 182

Глава 10 Упаковка и распространение 183
Упаковка против сборок из Editor 184
Упаковка проекта 184
Настройки упаковки 185
Как запустить упаковку 186
Запуск вашего приложения 187

Ошибки упаковки . . . 187
Распространение проекта . . . 188
Использование установщиков . . . 188
Заключение . . . 189

Часть 3 Архитектурная визуализация . . . 191

Глава 11 Настройка проекта . . . 193
Объем проекта и его требования . . . 194
Настройка проекта . . . 195
Работа с Project Settings . . . 199
Заключение . . . 202

Глава 12 Процесс работы с данными . . . 203
Организация сцены . . . 204
Материалы . . . 205
Архитектура и Арматура . . . 206
Экспортирование сцены . . . 208
Импорт сцены . . . 210
Декоративные меши (реквизит) . . . 214
Заключение . . . 218

Глава 13 Наполнение сцены . . . 221
Сборка сцен для визуализации . . . 222
Настройка Level . . . 222
Размещение Architecture Static Meshes . . . 224
Размещение декоративных мешей . . . 225
Организация сцены . . . 226
Заключение . . . 230

Глава 14 Архитектурное освещение . . . 233
Получение большего от Lighting . . . 234
Static Lighting с Lightmass . . . 235
Настройка Sun и Sky Lights . . . 235
Билд освещения . . . 240
Настройка Lightmass с архитектурной визуализацией . . . 242
Настройка Lightmap UV Density . . . 245
Размещение интерьерного света . . . 248
Размещение Light Portals . . . 250
Использование Reflection Probes . . . 251

Post-Process Volume 252
Заключение 259

Глава 15 Архитектурные материалы 261

Что такое мастер-материал? 262
Создание мастер-материала 264
Создание Material Instances 272
Сложные материалы 276
Заключение 284

Глава 16 Создание кинематики с Sequencer 285

Начало работы с Sequencer 286
Анимация камеры 289
Редактирование кадров 291
Сохранение 292
Совместная работа 292
Рендеринг в видео 292
Заключение 295

Глава 17 Подготовка уровня к интерактивности 297

Настройка Level 298
Добавление Player Start Actor 298
Добавление коллизий 299
Включение курсора мыши 304
Создание Post-Process Outlines 306
Заключение 308

Глава 18 Более сложные Blupritns: взаимодействие с UMG 309

Переключение данных 310
Создание вариаций уровней 310
Level Streaming 314
Определение Player Start Actor 317
Настройка Level Blueprint 318
Настройка Level Blueprint 322
Время тестов 325
Unreal Motion Graphics (UMG) 326
Вернемся к Level Blueprint 333
Заключение 337

Глава 19 Дополнительный уровень Blueprints: переключатель материалов . . . 339

Настройка цели . . . 340
Создание Actor Blueprint . . . 341
Создание Variables . . . 342
Добавление компонентов . . . 347
Создание функции Change Material . . . 349
Понимание Construction Script . . . 350
Понимание Event Graph . . . 355
Заполнение уровней . . . 359
Запуск приложения . . . 362
Заключение . . . 364

Глава 20 Финальные размышления . . . 365

UE4 продолжает меняться . . . 366
Будущее визуализации . . . 366
Следующие шаги . . . 367
Виртуальная реальность . . . 367
Создание фильмов . . . 368
Создание контента . . . 368
Спасибо . . . 368

ПРЕДИСЛОВИЕ

Unreal Engine (UE4) быстро становится звездой в области разработки игр, визуализации и даже создания фильмов. Компании, занимающиеся визуализацией, профессионалы и энтузиасты восхищаются его возможностями. Unreal Engine 4 — внушительный комплекс программных средств с тысячами функций, про которые сняты сотни часов обучающего видео и создано бессчетное множество туториалов, wiki-статей, разъяснений и указаний. Все эти ресурсы помогают ответить на огромное количество вопросов по UE4, но поиск материала по важным аспектам визуализации может оказаться ошеломляюще сложным. Практически все доступные обучающие платформы ориентированы на создание игр, а не на визуализацию.

UE4 может показаться как очень знакомым, так и абсолютно чужим для людей, уже работавших с традиционной визуализацией. Сходство с привычными инструментами упрощает его изучение, в то же время это лишь сходство в названиях, и на практике есть много существенных отличий. Часто эти ложные аналогии становятся причиной страшных разочарований и кажутся непреодолимыми трудностями.

Цель этой книги — создать руководство для студий визуализации и отдельных пользователей с реальными примерами надежных рабочих процессов, а также продемонстрировать некоторые инструменты и уловки. Unreal Engine 4 (UE4) быстро становится будущим игровой индустрии, визуализации и даже художественной кинематографии. Любители и профессионалы стремятся исследовать его возможности. Данная технология имеет огромное количество функций, сотни часов обучающих видео, бесчисленное количество руководств, учебников и wiki-сайтов. Отсеять среди всей этой информации полезную для визуализации крайне сложно, так как большая ее часть ориентирована на создание игр.

Для кого эта книга?

Данная книга предназначена для профессиональных визуализаторов, стремящихся создать впечатляющие интерактивные инновационные приложения, работающие в режиме реального времени, рендеры и анимации в Unreal Engine 4. Она также нацелена на руководителей команд разработки, которые хотят знать, что надо делать, чтобы ускорить работу команд визуализации и даже программистов.

Вы уже должны понимать, как работает 3D-рендеринг на профессиональном уровне, прежде чем приступать к этой книге. Такие понятия, как материалы, глобальное освещение (GI), полигональное моделирование и UV mapping, должны быть вам знакомы.

Я не буду освещать конкретные процессы визуализации данных за пределами UE4. Предполагается, что вы, взявшись за эту книгу, умеете работать с сырыми данными и преобразовывать их в формат, пригодный для 3D-рендеринга. Также необходимо хорошо разбираться в методах визуализации и терминологии, например вы должны понимать, что такое refraction и Fresnel falloffs.

Если вам интересны Blueprints, то желательно иметь некоторый опыт в написании скриптов, или программировании, или компьютерной логике. Понятия цикла, оператора if, переменной и ее типа — Boolean, Floats и Strings широко используются в Blueprints, поэтому должны быть вам понятны. Данная книга содержит обзор методов программирования в Blueprints, но это не учебник по программированию. Тем не менее Blueprints крайне удобны для начинающего пользователя, и любой, кто имел опыт работы со скриптами, особенно в 3D-приложениях, сможет легко их освоить.

Если вы продвинутый программист или разработчик UI, Blueprints покажутся вам родными. Функции, типизированные переменные и наследование работают как любой объектно-ориентированный язык. Как и многие языки программирования или платформы, Unreal имеет определенную специфику, поэтому стоит взглянуть на более ранние главы, чтобы лучше понять, что происходит «под капотом». Создание визуализации требует сочетания технических навыков для анализа предоставленных данных, умения использовать доступные инструменты, художественного и дизайнерского вкуса, чтобы оправдать ожидания требовательных клиентов.

Как организована эта книга?

Эта книга разделена на три части. Первая часть — это технический обзор UE4 — его основных функций, систем и рабочих процессов. Вторая часть демонстрирует создание простых интерактивных приложений при помощи UE4 Editor и первых Blueprints. В третьей части мы от начала до конца рассмотрим архитектуру реального проекта, начиная с клиентских данных в 3ds Max, добавим освещение при помощи Unreal Lightmass, создадим и применим материалы, анимацию камеры — Sequencer, отрендерим ее на диск и, наконец, закончим созданием приложения с UI, интерактивными элементами и фотореалистичным рендером.

В первой части мы изучим UE4 с концептуальной и технической точки зрения — основы, такие как установка лаунчера и движка, создание проекта и такие понятия, как Level, Maps и Assets. Вы изучите важную терминологию и технологии, гарантирующие, что, читая книгу или другие ресурсы по визуализации, вы хорошо поймете, о чем идет речь. Я попытаюсь объяснить, как все работает с технической точки зрения, чтобы вы могли применить полученные знания к собственному проекту визуализации. Мы посмотрим на материал книги с точки зрения пользователя V-Ray или mental ray, который впервые открыл Unreal Engine 4 и столкнулся с разницей между офлайн-рендерингом и рендерингом в реальном времени.

После того как мы изучим основы и получим четкое представление, что происходит в UE4, мы начнем погружаться в реальные примеры. Они описаны в учебном стиле с подробными пошаговыми инструкциями. Все файлы проектов (.MAX и файлы UE4) доступны для скачивания по ссылке **www.TomShannon3D.com/UnrealForViz**. Таким образом вы можете шаг за шагом следить за созданием проекта.

Часть I. Обзор Unreal Engine 4

1. **Начало работы с Unreal Engine 4.** С помощью подробного обзора UE4 вы узнаете, что UE4 добавляет в визуализации, с какими проблемами вы столкнетесь и как их преодолеть, где получить помощь и как начать планировать свои первые проекты на UE4.

2. **Работа в UE4.** Рабочий процесс UE4, скорее всего, сильно отличается от всего, что вы как профессионал в визуализации использовали раньше. Узнайте, как основные элементы UE4 функционируют вместе для создания, редактирования и распространения интерактивных приложений.

3. **Конвейер контента.** Добавление вашего контента в UE4 может быть одним из самых запутанных и сложных аспектов обучения. Узнайте, как UE4 импортирует и обрабатывает 2D- и 3D-контент из других приложений, а также изучите некоторые идеи о том, как интегрировать его в существующие конвейеры.

4. **Освещение и рендеринг.** UE4 представляет революционный физически основанный рендерер, который дает удивительные результаты за долю секунды на кадр. Вы научитесь использовать свое мастерство рендеринга и применять его к системе освещения UE4 на основе физически корректного рендеринга (PBR).

5. **Материалы.** Создание богатых, реалистичных материалов имеет важное значение для достижения фотореалистичности. Материалы в UE4 являются неотъемлемой частью рабочего процесса PBR и отличаются от любой системы материалов, которую вы ранее использовали. Узнайте, как создаются материалы, работают различные компоненты PBR, и начните изучать, как экземпляры материалов делают создание материалов в UE4 интерактивным и увлекательным.

6. **Blueprints.** Blueprints — это революция в написании скриптов и программировании игр. Теперь можно разрабатывать богатые передовые приложения, не написав ни одной строки кода. Тем не менее Blueprints по-прежнему являются языком программирования, и изучение основ позволит вам начать развиваться в этом направлении.

Часть II. Ваш первый проект в UE4

1. **Настройка проекта.** Научитесь определять цели проекта, а затем узнайте, как создать новый проект и настроить его основные параметры, чтобы начать с выбора предварительно созданного Starter Content для построения вашего уровня.

2. **Заселение мира.** Используя Starter Content, вы исследуете использование редактора в первый раз, помещая ассеты в мир, чтобы они стали акторами, перемещая и меняя их, а затем размещая источники света, чтобы осветить сцену.

3. **Использование Blueprints.** Создайте свои первые классы Blueprints, Player Controller, Pawn и GameMode. Вы назначите Input Mappings и запрограммируете входные данные игрока, позволяя ему перемещаться по уровню в режиме просмотра от первого лица.

4. **Упаковка и распространение.** После того как ваш проект заработает, настанет время подготовить его к распространению в виде отдельного приложения. В UE4 это называется упаковкой, и она создает оптимизированное, простое в установке и запуске приложение, которое вы можете легко заархивировать и отправить.

Часть III. Архитектура проекта визуализации

1. **Настройка проекта.** Вы снова определите цели своего проекта — на этот раз для создания высококлассной архитектурной визуализации с несколькими ключевыми результатами: интерактивным приложением и предварительно отрисованной пошаговой анимацией, выполненной с помощью Sequencer.

2. **Конвейер данных.** Вы узнаете, как подготовить и организовать ваши 3D-данные перед экспортом их в FBX. Вы поймете разницу между архитектурой и реквизитом. Затем вы исследуете несколько методов для получения ваших данных в UE4, сосредоточившись на рабочем процессе импорта и экспорта FBX.

3. **Наполнение сцены.** После того как ваши данные будут импортированы в UE4, вы сможете узнать, как перенести их на ваш уровень UE4. Есть несколько стратегий, которые нужно изучить, и здесь вы увидите несколько штук, используемых, чтобы получить как архитектуру, так и реквизит на сцене, там, где вы хотите.

4. **Архитектура освещения.** Решение Lightmass Global Illumination от UE4 прекрасно, но в то же время сильно отличается от того, как вы когда-либо визуализировали освещение сцены раньше. Вы научитесь получать потрясающее освещение с высоким динамическим диапазоном для ваших сцен, которые визуализируются за доли секунды.

5. **Архитектура материалов.** Опираясь на основы, которые вы изучили о материалах в первой части книги, вы начнете создавать основные материалы, программируемые параметры и другую логику шейдера, чтобы получить гибкий, быстрый и красивый набор экземпляров материала для применения к сцене.

6. **Создание Cinematics при помощи Sequencer.** Возможность рендеринга фотореалистичной анимации за долю секунды открывает целый мир творческих

возможностей. Научитесь создавать анимацию Sequencer, используя Cine Cameras для достижения физически правильного, кинематографичного вида, включая такие эффекты, как глубина резкости, размытие и виньетка. После завершения вы сможете записать 90-секундное пошаговое руководство на диск с разрешением 4k и 60 кадрами в секунду в течение нескольких минут.

7. **Подготовка Level для интерактива.** Столкновение (коллизия) как способ физического взаимодействия между объектами абсолютно необходимо для получения отличного интерактивного опыта, а также является одной из самых сложных и непонятых областей разработки UE4. Поскольку видеоигры требуют много интерактивных элементов, столкновение было разработано, чтобы быстро реализовать такие объекты, но иногда их трудно настроить. Узнайте, как легко подготовить ваш уровень в игре, чтобы по нему можно было передвигаться, не проваливаясь под текстуры.

8. **Более сложные Blueprints: взаимодействие UMG.** Одна из величайших возможностей интерактивной визуализации — это возможность сравнивать варианты в контексте и реальном времени. Вы узнаете, как настроить стриминг уровней, а затем поменять их местами во время выполнения с помощью Blueprints. Вы сможете предоставить эту функциональность игроку, создав в UMG простой пользовательский интерфейс.

9. **Дополнительный уровень Blueprints: Material Switcher.** Если вы действительно ищете испытаний, эта глава для вас. Здесь вы увидите, как разрабатывается готовый к выпуску переключатель материалов. Он предоставляет расширенные функциональные возможности не только игроку, но и дизайнеру уровней (LD), позволяя визуализировать настройку в Editor, создавая полный набор инструментов, который может быть повторно использован в любом проекте.

10. **Финальные размышления.** Эта книга только царапает поверхность айсберга, которым является UE4. Я надеюсь дать вам хороший фундамент для разработки и немного вдохновения для дальнейшего самостоятельного развития. В заключение я расскажу, куда движутся UE4 и отрасли, которые его используют, и как это повлияет на будущее визуализации.

Файлы ресурсов

Все файлы проекта UE4 и исходные файлы 3ds Max доступны по адресу **www.TomShannon3D.com/UnrealForViz**. Мы также предоставим ссылки на дополнительные ресурсы для каждой главы. UE4 — быстро развивающаяся технология, но я постараюсь поддерживать информацию актуальной, насколько это возможно.

Обозначения, используемые в этой книге

В этой книге будут использоваться следующие типографические обозначения.

- **Жирным** шрифтом выделены новые термины, имена переменных и параметров
- *Курсивом* выделены свойства и параметры

Вы можете зарегистрировать свою копию **Unreal Engine 4 для дизайна и визуализации** по адресу informit.com, с ее помощью вы получите более удобный доступ к загрузкам, обновлениям и исправлениям по мере их появления.

Для начала процесса регистрации перейдите по ссылке informit.com/register и войдите в систему или создайте учетную запись. Ведите ISBN продукта (9780134680705) и нажмите кнопку «отправить».

БЛАГОДАРНОСТИ

Я хотел бы выразить благодарность Лоре Левин, ответственному редактору Pearson Education Technology Group, за предоставленную возможность опубликовать эту книгу. Я также хотел бы поблагодарить Шери Реплин, моего потрясающего редактора разработки, за ее обратную связь и руководство; Поле Лоуэлл, которая занималась редактированием копий, и Лори Лайонс за помощь. Я также хотел бы поблагодарить Оливию Баеджио за поддержку во время всего процесса.

Технические рецензенты этой книги сделали все возможное для того, чтобы предоставить надежные источники информации и дать самую разную обратную связь. Никаких слов не хватит, чтобы поблагодарить Дэвида Спаркса за его советы и опыт, которым он щедро делился со мной. Я также хотел бы выразить благодарность Тиму Хобсону и Сэму Дейтеру из Epic Games за их время и усилия, направленные на предоставление максимально точной информации.

Я хотел бы поблагодарить Epic Games за их видение и энтузиазм, а также за то, что они взяли меня с собой в поездку к будущему визуальной коммуникации. Это честь — работать вместе с лучшими разработчиками игр в мире.

Я также хотел бы поблагодарить команду Hoyt Architecture Labs и Imerza, Дориана Ви и Гэри Хойта, за их щедрость и поддержку, а также за предоставление читателям этой книги доступа к их исходникам.

Наконец, я хочу поблагодарить мою жену за ее бесконечную поддержку и вдохновение и моих детей за их любовь и за то, что они приносят радость и игры в мою повседневную жизнь.

ОБ АВТОРЕ

Том Шэннон — технический художник с более чем десятилетним профессиональным опытом разработки видеоигр и визуализации, эксперт в Unreal Engine и просто фанат своего дела. Он увлечен играми и связанными с ними технологиями, а также визуализацией и ее влиянием на реальный мир. Он проводит свои дни, балансируя между строительством мира вместе с архитекторами, инженерами и дизайнерами, и его разрушением гигантскими роботами вместе с программистами, аниматорами и художниками по эффектам.

Том живет в Колорадо со своей прекрасной женой Серин, которая остается его вдохновителем, и своими замечательными и скромными детьми Эммой и Декстером.

Вы можете найти его сайт по адресу www.TomShannon3D.com.

ЧАСТЬ I

ОБЗОР UNREAL ENGINE 4

1 Начало работы с UE4

2 Работа с UE4

3 Процесс работы с контентом

4 Освещение и рендеринг

5 Материалы

6 Blueprints

ГЛАВА 1

НАЧАЛО РАБОТЫ С UNREAL ENGINE 4

Unreal Engine 4 — это мощная программа для разработки и платформа создания контента. Именно графические возможности, мощные инструменты и прекрасная расширяемость делают движок одной из популярных и доступных платформ. Игровые разработчики, архитекторы, ученые, любители, создатели фильмов и визуальные художники используют UE4 для создания ошеломляющего интерактивного опыта, который переопределяет состояние искусства. Все эти люди начинали с чего-то свои первые проекты в UE4, и, вероятно, это было в www.unrealengine.com.

Что такое Unreal Engine 4?

Unreal Engine 4 (UE4) — это самая последняя версия много лет развивающегося инструмента для разработки игр Unreal Engine от компании Epic Games. UE4 — профессиональный набор инструментов для разработки видеоигр, который содержит в себе современный рендер физически корректных материалов в реальном времени, отражений и освещения с мощными инструментами для создания непередаваемого интерактивного опыта. Инструменты включают в себя симуляторы физики, освещения и теней, пользовательских интерфейсов, листвы и рендеринга, массивных террейнов, сложных материалов, визуальных скриптов, анимации персонажа, моделирование системы частиц, кинематики, мультиплеера, и все, что нужно команде опытных разработчиков игр для создания многомиллионной игры-блокбастера AAA-класса. Любая возможность, которой не хватает в UE4, может быть добавлена благодаря открытому доступу к полному, модифицируемому исходному коду на C++.

Несмотря на свою мощь и модульность, UE4 невероятно прост. Редактор в UE4 (рисунок 1.1) может похвастаться современным интерфейсом, прекрасными инструментами, удобной документацией, полным доступом к исходному коду и процветающим, постоянно растущим комьюнити. Даже в одиночку человек очень быстро может начать создавать удивительные игры, симуляции и визуализации на профессиональном уровне.

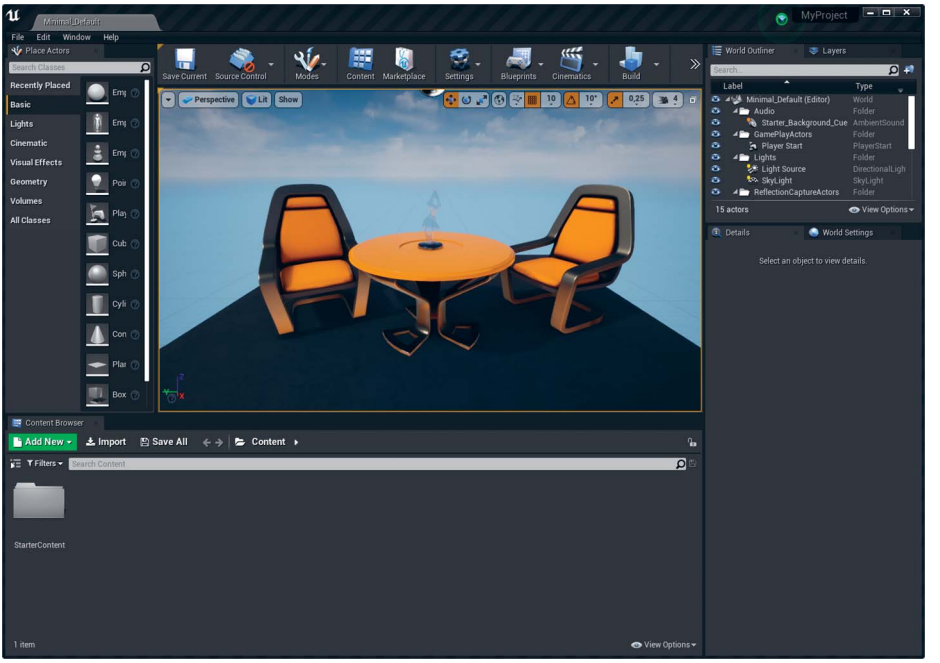


Рисунок 1.1 Интерфейс редактора Unreal Engine 4

Маленькие команды могут легко увеличиваться, продолжая совместно работать над одним проектом, совершая невероятные подвиги, которые когда-то считались возможными только в огромных студиях с большим бюджетом. Теперь любой, имеющий доступ к достаточно мощному компьютеру, может скачать бесплатно Unreal Engine 4 и начать создавать собственные удивительные интерактивные миры.

Краткая история Unreal Engine

С конца 1990-х Epic Games и другие профессиональные разработчики игр использовали различные версии движка Unreal Engine, чтобы создать сумасшедшее количество самых продаваемых и отмеченных наградами игр и симуляторов. Такие игры, как Gears Of War, Mass Effect, Batman Arkham Knight и, конечно же, Unreal Tournament, разработаны с помощью Unreal Engine.

Доступ к игровым движкам профессионального уровня традиционно был очень технически сложным и дорогостоящим предложением, недоступным для большинства маленьких студий и недостижимым даже для некоторых крупных, куда более технически подкованных профессиональных студий. Это вынудило многие маленькие инди-студии и разработчиков визуализации создавать собственные движки или переходить на менее дорогие и сложные решения. Подобные решения включали в себя ранние промежуточные рендеры, физические библиотеки, аудиосистемы и другие инструменты, которые можно было собрать вместе для создания проектов, но в целом им не хватало изящества и качества визуализации. Ну и они все еще оставались чрезвычайно сложными для разработчиков.

За последнее десятилетие популярность независимых игр[1], разрабатываемых в маленьких, в основном самофинансируемых студиях, резко возросла, и они часто оказывались чрезвычайно успешными. Тысячи самостоятельных разработчиков и крошечных студий сейчас создают игры и новый игровой опыт для огромной аудитории геймеров.

Хотя немало инди-студий разрабатывали собственные движки, многие полагаются на готовые решения, такие как Unity 3D, DX Studio и Torque. Они создавались, чтобы предложить полный, интегрированный набор инструментов для разработки игр, но значительно дешевле, чем Unreal Engine 3 и CRYENGINE. Даже Epic Games приняли идею с бесплатным использованием (free-to-use) для binary-only версии UE3 под названием Unreal Developers Kit (UDK).

И хотя эти движки доказывали, что очень успешны, они всегда играли в догонялки с премиальными инструментами, сталкивая независимых разработчиков и мелкие студии с серьезными технологическими трудностями.

[1] Имеются в виду инди-команды. — *Прим. пер.*

Представление Unreal Engine 4

Спустя годы разработки за закрытыми дверями Epic Games в штаб-квартире в Кэри, Северная Каролина, Epic (всего лишь с несколькими скудными демо и закрытой бета-версией) представили Unreal Engine 4, следующую версию игрового движка и приемника Unreal Engine 3, имевшего колоссальный успех.

Релиз был абсолютно поразительный и ошеломил всю индустрию.

UE4 — это не подгруппа движков или демонстрационная версия; это доступ к исходному коду на C++ и инструментам, *ко всему*, что стоит десятки тысяч долларов, и все за небольшую ежемесячную плату. Движок включал в себя расширенную документацию, прямой контакт с разработчиками и доступ к их кодовой базе.

Реакция сообщества разработчиков игр была мгновенной и чрезвычайно позитивной. Сотни людей скачивали движок и создавали игры и демки. Это изменило взгляд на видеоигры: на их вид, звучание и игровой процесс.

Unreal Engine 4 представил совершенно новый, построенный с нуля, кросс-платформенный, расширяемый и современный редактор. Процесс рендеринга был полностью перестроен на основе использования физически корректных материалов и освещения, позволявших создавать невероятно живые и реалистичные сцены, которые соперничали с качеством рендеринга с трассировкой лучей.

Новые системы освещения и отражений, совмещенные с полностью линейным конвейером рендеринга и новейшими эффектами постобработки, создали визуальные эффекты такого качества, которые сравнимы с лучшими примерами в кинематографии — и все это с непревзойденной простотой использования, никогда прежде невиданной в игровых движках.

Кроме того, был представлен новый язык визуального программирования, Blueprints, позволяющий тестировать «на лету» абсолютно все реализованное средствами движка в игровом мире без необходимости написания и строчки кода. Целые игры, включая мультиплеер и HUD, могут быть созданы с использованием Blueprints, освобождая разработчиков от необходимости программировать на C++.

В момент запуска Unreal Engine 4 опережал на несколько лет своего ближайшего конкурента, даже на самом высоком профессиональном уровне, и он предлагался *для всех* по ежемесячной подписке всего за $20.

Меньше года спустя Epic анонсировали отмену ежемесячной подписки и объявили, что теперь движок доступен бесплатно для всех. Самый мощный движок на рынке внезапно стал также самым дешевым, самым простым в использовании, самым открытым и самым хорошо поддерживаемым.

Однако Epic Games не почивала на лаврах. С момента первого выпуска в 2014 году UE4 получил 23 основных обновления. Каждое из которых принесло сотни новый функций, тысячи улучшений и исправлений ошибок. Например, основные функции рендеринга, такие как затемнение волос и кожи, а также основные функции движка, такие как интеграция VR и постоянная поддержка для почти всей современной аппаратуры. Благодаря доступу к коду движка работа над UE4 ведется открыто между Epic Games и членами сообщества разработчиков игр, которые непосредственно участвуют в разработке движка, помогая обеспечить максимальную надежность, безошибочность и полезность каждой функции. Никогда еще такое масштабное программное обеспечение не разрабатывалось настолько открыто и свободно, и результаты говорят сами за себя.

Не только крупные и мелкие игровые компании перешли на UE4, но и неигровые разработчики начали осваивать этот движок, чтобы легче достигать своих целей. Кинематографисты, музыканты, архитекторы, инженеры и многие другие использовали мощь и простоту UE4. Креативные профессионалы и команды, которые ранее никогда и не мечтали использовать или даже иметь доступ к игровому движку, начали работать с Unreal Engine 4 для создания собственных уникальных, интерактивных и визуальных историй.

Основные моменты UE4 для визуализации

У UE4 есть все необходимое для создания AAA-игры или высококлассной визуализации. На самом деле он предлагает десятки инструментов, преимуществ и функций; однако следующие являются самыми важными при разработке визуализации.

1. **UE4 Editor:** динамический, современный, приятный в использовании набор для разработки игр, созданный профессионалами, которые полагаются на движок, чтобы создавать собственные игры. Редактор (Editor) связывает все инструменты движка вместе с контентом, чтобы создать прекрасный опыт.

2. **Освещение и материалы:** удивительно реалистичное, предварительно запеченное глобальное освещение с использованием Lightmass или высококачественное динамическое освещение с реалистичными тенями, отражениями и материалами. Все это достигается благодаря современной и легкой в использовании системе физически корректного рендеринга.

3. **Сиквенсер:** инновационная анимация камеры и объектов, а также инструмент сиквенирования, который сочетает в себе легкое в освоении приложение для редактирования видео с мощью интерактивного игрового движка.

4. **Рабочий процесс FBX:** повсеместный формат файлов FBX, который используется исключительно для 3D-геометрии и анимации в UE4.

5. **Blueprint:** визуальное программирование, которое не требует построчного написания кода и компилируется напрямую в редакторе для немедленной обратной связи.

6. **UMG (Unreal Motion Graphics):** основанная на Blueprint разработка простых в использовании, привлекательных пользовательских интерфейсов, необходимых для успешной интерактивной визуализации.

7. **Виртуальная реальность:** как только это стало предметом научной фантастики, VR активно используется и преобразует все формы медиа, включая визуализацию. UE4 интегрировал VR в себя и устанавливает стандарт для создания интерактивного VR-контента.

8. **Поддержка платформ:** единоразовая разработка и выпуск приложений на ПК, Mac, Linux, iOS, Android, VR и многое другое с очень небольшим количеством изменений в проектах.

9. **Лицензирование и стоимость:** либеральные условия лицензирования и свободный доступ к редактору, поддержке и удивительному сообществу UE4 в сочетании с недорогими требованиями к аппаратному обеспечению означают, что сегодня почти каждый может начать разработку в Unreal Engine 4.

Разработка интерактивной визуализации с Unreal Engine 4

Unreal Engine 4 может похвастаться одним из самых реалистичных и гибких конвейеров рендеринга. Несомненно, вас уже впечатляло потрясающее качество картинки в играх, визуализациях и VR-приложениях, созданных с помощью UE4.

Хотя UE4 и раздвигает границы визуализации и средств художественной выразительности повышенным качеством изображений, его главное преимущество — скорость, с которой он позволяет визуализировать, что делает его интерактивным. Как в редакторе, так и в созданном вами проекте UE4 позволяет рендерингу реагировать непосредственно на ваши команды или ввод игрока в режиме реального времени.

Скорость рендеринга и интерактивность, которую она привносит в процесс визуализации, являются двумя наиболее важными различиями между интерактивной визуализацией в UE4 и традиционной рендеринговой визуализацией, что влияет почти на каждый аспект работы с проектом. UE4 снимает многие творческие ограничения для студий визуализации и профессионалов, которые хотят научиться новым фишкам, избавиться от старых, вредных привычек и получить опыт интерактивной визуализации.

Что UE4 привносит в визуализацию

Несмотря на то что интерактивность требует серьезных усилий, времени, технических знаний и зачастую влечет за собой сложные рабочие процессы, она с лихвой оправдывает вложения. Она трансформирует ваше видение визуализации и меняет зрительные ощущения от готового проекта у ваших заказчиков и клиентов.

Unreal Engine 4 зарекомендовал себя как мощный инструмент для всех видов интерактивной визуализации: автомобильной, аэрокосмической, архитектурной, инженерной, научной — все это в режиме реального времени. Инструменты, предлагаемые UE4, не имеют себе равных в отрасли, а приложения, разрабатываемые с его помощью, являются доказательством его невероятных возможностей как платформы разработки интерактивной визуализации.

Нелинейное и реальное время

Приложения в Unreal Engine 4 работают в режиме реального времени. Это не только обеспечивает интерактивность, но и позволяет времени в нашем мире течь так же, как и в реальности. На фотографии время останавливается. На видео время статично и неизменно. В Unreal Engine 4 время и пространство изменчивы, и вы можете ими манипулировать. Весь мир может быть изменен в мгновение ока, высвечивая такие пространственные и временные отношения, которые просто невозможны в традиционной визуализации.

Вы можете создать визуализацию, которая будет работать каждый раз одинаково, используя заранее определенные траектории камеры и действия акторов, но волшебство начинается, когда вы превращаете зрителя в игрока, предоставляя ему контроль над временем и пространством, и этого можно добиться только при помощи интерактивной визуализации.

Скорость, скорость и еще раз скорость

Возможность рендеринга целых сцен менее чем за секунду меняет правила игры. Включив в свой проект обработку входных данных, физику, звук, пользовательский интерфейс и взаимодействия с игроком, вы создадите практически любой мир и игровой опыт, какой только вам заблагорассудится. Единственные ограничения — это ваши возможности и объем доступной вычислительной мощности.

Традиционная анимация опирается на заранее созданную анимацию с помощью офлайн-рендеринга: серии 2D-растровых кадров обрабатываются с помощью программ трассировки лучей, таких как V-Ray, Mental Ray или Maxwell, которые затем компилируются в приложении для редактирования видео, или композиторе, например After Effects.

Эти рендеры чрезвычайно детализированы, с корректным освещением и удивительно реалистичными материалами, которые имитируют стекло, мрамор, воду и листву с почти фотореалистичной точностью. Затем эти кадры собирают и накладывают на них

визуальные эффекты, аудиодорожку и графику движений, после чего происходит повторный рендеринг в видеофайл или изображение. Получившийся файл отправляют клиенту.

Несмотря на то что каждый кадр демонстрируется в течение дискретного отрезка времени, обычно 1/30 секунды каждый, для его рендеринга могут потребоваться часы, а для анимации — целые дни. После завершения подготовки каждого кадра часто требуется дополнительное время на постобработку и редактирование, прежде чем будет получен окончательный результат.

Unreal Engine рендерит каждый кадр в реальном времени, то есть для рендеринга со скоростью 30 кадров в секунду требуется всего 1/30 секунды на каждый кадр. UE4 не только справляется с этой непростой задачей, но и добавляет постэффекты и аудио, а также имитирует физику и логику геймплея вместе с каждым кадром.

Для интерактивной визуализации требуется частота кадров не меньше 30 кадров в секунду (fps). Это значит, что каждый кадр должен рендериться за 1/30 секунды или 33,3 миллисекунды (мс). Если сравнить с традиционной визуализацией, которая требует 20 минут на подготовку одного кадра, получается в 36 000 раз быстрее!

Например, вам необходимо создать трехминутную HD-анимацию. Это дает вам 5400 кадров, которые надо визуализировать. При 20 минутах на кадр (что является неплохой оценкой для многих средств визуализации) время на рендеринг составит 108 000 минут, или 1800 часов, или 75 дней. Конечно, большинство студий используют рендер-фермы, которые объединяют мощности нескольких компьютеров и сокращают время рендеринга до долей изначального, но даже довольно мощной рендер-ферме, состоящей из 25 узлов, потребуется целых три дня на трехминутную анимацию.

Рендеринг изображений и анимации

В дополнение к скорости, обеспечивающей интерактивность, UE4 может быть использован для рендеринга кадров и анимации с предварительно определенным путем камеры, который мы можем задать с помощью ключевых кадров непосредственно в Sequencer или импортировать путь из приложения по выбору.

Когда анимации рендерятся в реальном времени, вы можете использовать несколько приемов, чтобы сделать качество намного выше, чем обычно у интерактивных сцен; то есть вы можете пожертвовать скоростью для дополнительного разрешения и качества картинки. Вы можете легко отрисовывать видео с высоким разрешением и частотой кадров. UE4 делает это настолько быстро, что время записи на диск может быть дольше, чем непосредственно сам рендеринг.

То, что вы видите, это то, что вы получаете (WYSIWYG)

Рендеринг с трассировкой лучей может занимать часы, и большую часть времени в окне просмотра предварительного результата вашего редактора вы видите

предварительную, упрощенную версию сцены, которая неточно отражает окончательный рендер. Освещение показано приблизительно, а материалы вообще могут не отображаться. Тени и глобальное освещение очень редко показываются в окне предпросмотра. Каждое изменение материалов или освещения требует дополнительного времени рендеринга, чтобы вы могли просмотреть его. В большинстве сцен это делает процесс работы утомительным и трудоемким. С помощью редактора UE4 вы почти всегда получаете точное представление об освещении, положении камеры, эффектах постобработки и других элементах сцены — все сразу, в режиме реального времени. Окно просмотра, в котором вы редактируете свои сцены, и финальный проект запускает один и тот же движок, что позволяет видеть все изменения одновременно.

Природа WYSIWYG UE4 предоставляет огромную творческую свободу в производственном конвейере. Теперь директор и клиент могут увидеть, как будут выглядеть окончательные рендеры на ранних стадиях проекта. Это делает процесс подписания и утверждения документации быстрее и надежнее. Также это существенно упрощает поиск подходящих материалов и корректировку освещения, так как UE4 позволяет моментально увидеть изменения и подобрать необходимую конфигурацию.

Интерактивность оживляет мир в UE4

Превращение зрителей в игроков переосмысляет процесс визуализации. У данного подхода просто нет недостижимых пределов. По одному нажатию кнопки вы можете оказаться в любом месте театра или любоваться видом из любой комнаты в любом высотном здании. Вы без проблем можете настроить дизайн декора своего нового автомобиля вплоть до зернистости кожи сидений, и все это сделано с прекрасной точностью и реалистичностью.

Одним из наиболее привлекательных способов использования интерактивных визуализаций является исследование проектов во времени или сравнение альтернатив и вариантов проектирования в контексте и в режиме реального времени. Превращение «зрителя» в игрока невероятно расширяет возможности и предлагает новые способы визуализации и передачи сложных наборов данных.

Даже работа и редактирование миров в UE4 — это весело, интерактивно и визуально приятно. Вы летите по миру в Perspective Viewport с постоянным обновлением кадров, видя, как освещение отражается от материалов.

Когда вы размещаете свет, вы видите предварительный просмотр теней с высоким разрешением, показывающий вам то, что вы получите в конечном продукте. Blueprints позволяют вам интерактивно изменять мир без написания кода, компиляции или необходимости выходить из редактора, чтобы опробовать новые функции.

Камеры могут быть настроены на лету, обеспечивая точные эффекты постобработки, такие как глубина резкости и размытие движения, которые визуализируются в режиме реального времени. Для художников визуализации, которые потратили годы

на то, чтобы внедрить эти дорогостоящие эффекты в свои сцены, видеть такие высококачественные эффекты и иметь возможность корректировать их в реальном времени — освобождение.

Вау-эффект

Unreal Engine 4 производит визуальные эффекты, которые заставляют людей говорить «Вау». Визуализация всегда была очень технологичной отраслью. Достижения в области вычислений, изображений, моделирования и рендеринга быстро внедряются. Клиенты всегда ищут способы выделиться, как и студии визуализации. Использование игровых движков для визуализации не ново, но способность интерактивных приложений удивлять людей — это нечто новое. Ландшафт области еще только формируется, но люди уже начали задумываться о своих проектах и о том, как донести их до аудитории в интерактивном контексте.

Будущее

Виртуальная реальность, дополненная реальность, мобильные платформы Epic Games, ее партнеры и десятки тысяч разработчиков UE4 создают будущее, обладая почти неограниченной расширяемостью и постоянно растущим набором функций. Ни одна другая платформа разработки или рендеринга не может позволить вам воспользоваться преимуществами того, что будет дальше.

Предостережения по разработке на UE4

Unreal Engine 4 — это фантастическая платформа разработки, которая предлагает творческие и художественные возможности, невообразимые всего несколько лет назад. Хотя никогда не было такого игрового движка, как UE4, это все еще игровой движок в сердце, разработанный компанией по разработке видеоигр для разработки видеоигр.

Чтобы в полной мере использовать то, что предлагает UE4, и создавать наиболее привлекательные визуализации, вы должны научиться думать о своих историях, контенте и рабочих процессах разработки по-новому. Вы также должны научиться работать с ограничениями UE4, а иногда и обходить их, и вообще избегать их, если сможете.

Сложность разработки

Вы можете спросить: «Почему мы все до сих пор не использовали игровые движки для рендеринга?» Простая причина заключается в том, что это может потребовать гораздо больше усилий. Создание контента для быстрого рендеринга требует чрезвычайно агрессивной оптимизации, мощного оборудования и множества продуманных рабочих процессов для обеспечения точности и достоверности.

Разработчики используют уровень детализации (LOD), стриминг, тесселяцию, предварительный расчет, определение скрытой поверхности и кэширование, чтобы убедиться, что во время выполнения необходимо вычислить как можно меньше каждого

кадра. Текстуры предварительно обрабатываются и хранятся в форматах с аппаратным ускорением, отражения предварительно захватываются в зондах отражения, а освещение и глобальное освещение (GI) вычисляются и хранятся в текстурах с использованием Lightmass.

Все это требует времени, усилий и планирования, чтобы убедиться, что это работает. Хотя традиционный рендеринг требует большей части той же оптимизации, давление на рендеринг в доли секунды здесь отсутствует, и время рендеринга может быть компенсировано дополнительным оборудованием или хорошо спланированными графиками, которые позволяют рендерить больше времени. Хотя пятиминутное время рендеринга на кадр отлично подходит для автономного рендеринга, даже если он занимает более 1/15 секунды в интерактивной визуализации, это означает грубый, прерывающийся опыт для ваших игроков.

Нет права на ошибку

Имея всего 0,033 секунды (33,3 мс, или 1/30 секунды) для рендеринга каждого кадра, никакая ошибка или неэффективность не являются незначительными. Отставание только 0,001 секунды от времени, необходимого для рендеринга объекта, кажется слишком маленьким, чтобы прикладывать много усилий, чтобы устранить его, но, если у вас есть 30 таких объектов (например, деревья, цветы, автомобили, здания или люди) в вашей сцене, вы уже сэкономили 0,03 секунды (3 мс), или 2,5 кадра в секунду. Это крошечная погрешность, а небольшие недостатки быстро накапливаются.

Когда вы начинаете обсуждать VR, вы говорите как минимум о 90 кадрах в секунду, распределенных между двумя дисплеям. Это всего лишь 11 миллисекунд для рендеринга всей сцены дважды.

Требования других приложений

UE4 предлагает удивительный набор инструментов, которые вы можете использовать для достижения удивительных вещей с вашим контентом. Что он не может сделать очень хорошо, так это создать контент, который будет отображаться. В то время как UE4 предлагает десятки передовых инструментов, почти все они требуют контента, созданного в другом приложении, чтобы достичь наилучших результатов.

Этот «недостаток» имеет одно большое преимущество, по замыслу. Это позволяет вам делать почти все ваши разработки контента в приложениях, которые вы используете каждый день, и использовать разработанные вами рабочие процессы. Если ваше приложение поддерживает экспорт в FBX или может быть импортировано тем, что это поддерживает, вы можете получить свой контент в UE4.

Это разработка программного обеспечения

Интерактивные визуализации — это программные приложения. Они имеют программную логику и пользовательский интерфейс и работают в режиме реального времени. Это означает, что они также могут иметь ошибки, нуждаться в новых функциях

и требовать поддержки и обслуживания, которые существуют далеко за пределами срока поставки проекта.

Даже самые организованные студии обнаружат, что структура, сроки и управление разработкой интерактивной визуализации с помощью UE4 часто несовместимы с автономным конвейером рендеринга. Тестирование игр, программирование и отслеживание ошибок — вот некоторые из новых рабочих процессов, которые вы должны включить в ваш процесс, чтобы обеспечить успешную реализацию проекта.

Если ваше приложение предназначено для публичного использования, усилия по полному тестированию и исправлению приложения потребуют гораздо больше времени, чем вы изначально предполагали. Будьте готовы к этому, составьте бюджет и планируйте свои проекты соответственно.

Игроки могут смотреть все

Когда вы создаете статическую визуализацию или анимацию, у вас есть преимущество в том, что вы точно знаете, что зритель будет видеть в каждый момент. За исключением неизбежного изменения траектории камеры в последнюю минуту, вы можете выбрать, на чем сосредоточить свои усилия, игнорируя все, что вы не видите. Если он не визуализирован, он даже не должен быть там.

В интерактивном 3D-окне игрок может смотреть в любом направлении в любое время. Он может подобраться так близко, как ему нравится, почти к каждому углу и приспособлению. Любая ошибка может (и будет) обнаружена. Очень важны точность, аккуратность и внимание к деталям.

В виртуальной реальности это становится еще более очевидным, когда игрок использует позиционное отслеживание. Это означает, что он часто может буквально проникать внутрь стен и других объектов, видеть внешний мир, обнаруживая волшебника за занавесом.

Показатель отличных проектов UE4

Некоторые общие требования к проекту помогут вам определить, когда проект выиграет от использования UE4, а также несколько важных моментов, на которые следует обратить внимание.

Чтобы извлечь выгоду из UE4, проект должен использовать одну из сильных сторон Unreal Engine 4 в достаточной степени, чтобы уравновесить дополнительные усилия по разработке интерактивной визуализации по сравнению с традиционной визуализацией.

Интерактивность, скорость, программируемость и нелинейный характер интерактивных визуализаций в реальном времени дают безграничные возможности, но иногда UE4 просто не подходит для проекта и даже может быть неудачным выбором. Наличие четкой причины использовать UE4 очень важно, прежде чем идти по этому пути.

Сравнение вариантов

Безусловно, наиболее распространенной и убедительной причиной разработки интерактивной визуализации является необходимость сравнения наборов данных или вариантов проектирования в трехмерном пространстве и времени.

Сравнение альтернатив всегда было сложной задачей для традиционной визуализации и линейной доставки данных. Даже простое сравнение A/B может удвоить время рендеринга для традиционных визуализаций. Добавление всего нескольких вариантов может быстро умножиться на десятки или сотни вариаций и перестановок. Нелинейный характер интерактивной визуализации и мощные пользовательские интерфейсы (UI), которые вы можете разработать в UE4, позволяют игрокам исследовать данные личными, значимыми способами.

Предоставление игроку возможности переключать альтернативы, устанавливать переменные и управлять своим представлением в соответствии с его точными потребностями чрезвычайно расширяет его возможности. Это тоже происходит в реальном времени. Здесь нет фреймов для визуализации и нет презентаций PowerPoint для обновления; изменения в данных могут быть интегрированы немедленно.

Изменение набора данных

Проекты с постоянно меняющимися наборами данных или те, которые, вероятно, будут получать обновления в последнюю минуту, также являются отличными кандидатами для использования UE4. Почти исключив время рендеринга, обновления можно увидеть сразу.

Отличным примером такого рода проектов является разработка проектных предложений. Очень важно, чтобы дизайнеры, архитекторы и инженеры могли работать до последней минуты. Спецификации для проектов часто меняются за несколько дней до срока, или решения ключевых проблем решаются только за несколько дней или часов до презентации. С традиционной анимацией эти изменения могут быть огромной проблемой, потому что время рендеринга часто не позволяет визуализациям представлять окончательный предлагаемый дизайн или, что еще хуже, требования проекта. Это может сильно снизить ценность визуализаций, а иногда даже стать помехой. Наличие великолепных визуализаций, на которые команда может положиться в одиннадцатом часу, очень важно, и скорость UE4 может дать вашей команде такую возможность.

Требования к разработке Unreal Engine 4

Разработка на UE4 требует сочетания правильного оборудования, программного обеспечения и, конечно же, правильных людей. Самой большой проблемой для большинства художников и студий визуализации, несомненно, будет приобретение новых навыков, необходимых для разработки интерактивных визуализаций. Оборудование

и программное обеспечение можно легко приобрести и модернизировать; приобретение новых навыков и людей требует времени и настойчивости.

Будучи опытным разработчиком визуализации, вы уже являетесь опытным трейдером, владеющим искусством визуального повествования, света и цвета, анимации и вызывающим эмоциональные и мозговые реакции, которые делают визуализацию таким мощным инструментом коммуникации. Скорее всего, вы не будете искусным программистом, звукорежиссером или дизайнером пользовательского интерфейса, и ваш рабочий процесс почти наверняка не основан на отслеживании и обновлении сотен фрагментов контента и исходного кода.

На мой взгляд, найти кого-то, кто увлечен созданием интерактивного опыта, очень важно для студий, желающих пройти мимо создания статических визуализаций. Этот человек, возможно, уже существует в вашей команде и требует использовать UE4 или любой другой игровой движок для создания своего искусства. Сотни, если не тысячи профессионалов и студентов приняли UE4 и в сообществе стремятся присоединиться к любой команде, которая так же стремится, как и они.

Что касается аппаратного и программного обеспечения, то для разработки UE4 требуется достаточно мощный компьютер с хорошим объемом места на жестком диске. Наиболее важным компонентом является мощный «выделенный» графический процессор или видеокарта. Рабочие станции и ноутбуки часто поставляются без выделенных графических процессоров, работая на «интегрированных» графических чипах, которые находятся на материнской плате. Интегрированные графические процессоры не рекомендуются и не поддерживаются.

Для разработки я рекомендую мощный графический процессор, который намного мощнее вашей цели. Это позволяет вам выдержать накладные расходы редактора плюс любые другие приложения, которые вы можете запускать при разработке в UE4. Однако я должен предостеречь вас от лучших графических процессоров, потому что они могут вам дать очень неточное представление о производительности вашего приложения, что затруднит оценку того, как оно будет работать на вашей целевой платформе.

Требования к программному обеспечению для разработки UE4 варьируются в основном в зависимости от ваших конкретных потребностей, и у вас, вероятно, уже есть все необходимое для начала работы почти без каких-либо или с очень небольшими инвестициями. Редактор Unreal Engine 4 изначально работает на Windows, Mac OSX и Linux. Вы можете использовать коммерческие приложения, такие как Photoshop, Maya и 3DS Max, или бесплатные альтернативы, такие как GIMP и Blender, для создания контента для UE4. Unreal Engine 4 использует стандартные индустриальные форматы файлов для обмена данными (FBX, TARGA, EXR и т. д.), что позволяет вам адаптировать свои инструменты и рабочий процесс для работы с UE4.

Командная работа в Unreal Engine 4

То, как команды работают вместе в Unreal Engine 4, отличается от того типа рабочего процесса, к которому привыкло большинство команд визуализации. Корни Unreal Engine лежат не в анимации и видеопроизводстве большинства производственных программ, распространенных в визуализации, а в производственной среде разработки программного обеспечения для разработки видеоигр.

Обзор системы управления версиями

Типичный рабочий процесс проекта, размещенный на файловом сервере, к которому привыкло большинство художников визуализации — когда все файлы проекта хранятся на сервере и доступны одновременно, вообще не рекомендуется использовать для проектов Unreal Engine 4.

Наличие нескольких человек, работающих над одними и теми же файлами в проекте UE4, может привести к конфликтам и ошибкам и невероятно быстро сломать проект. Любой, кто когда-либо работал в команде, имел опыт, когда два человека работали над одним и тем же файлом одновременно. В большинстве случаев работа одного человека будет нарушена. В UE4 этот конфликт может не только означать потерю работы, но и привести к ошибкам ссылок, которые способны привести к полному отказу запуска проекта.

Чтобы избежать этих конфликтов и обеспечить резервное копирование и безопасность данных, вы всегда должны использовать систему управления версиями исходного кода или файлов.

Хотя многие «бренды» программного обеспечения для управления версиями (Perforce, SVN, Git и т. д.) доступны, все они имеют общий рабочий процесс: пользователи обновляют или загружают последнюю версию проекта с удаленного сервера на свою локальную машину, а затем, когда они вносят изменения в файлы, они «проверяются». Это говорит другим пользователям, что эти файлы изменены и находятся вне пределов досягаемости. После того как пользователь закончит вносить изменения, он может «зарегистрировать» или «отправить» файлы на сервер. Теперь другие пользователи могут «обновить» копии файлов на своих рабочих станциях.

Важно отметить, что ни одна передача файлов не происходит автоматически, как это могло бы быть с такой системой, как Dropbox. Она всегда инициируется пользователями. Это гарантирует, что не произойдет никаких неожиданностей; например, не произойдет случайного обновления файлов во время работы или случайного обновления, когда изменения не должны были быть сохранены.

Большинство систем управления версиями также требуют комментариев при каждом изменении файлов. Это позволяет каждому увидеть, кто сделал какую работу и какие новые функции были добавлены, просто взглянув на журнал версий.

Интеграция поддержки управления версиями в Unreal Engine 4

Unreal Engine 4 поддерживает несколько наиболее распространенных и подходящих платформ управления версиями файлов, встроенных непосредственно в редактор. Если вы включите поддержку управления версиями в UE4, редактор автоматизирует многие из наиболее распространенных задач, таких как извлечение, переименование и другие задачи управления файлами.

Вы все равно должны вручную отправить свою работу на сервер, когда закончите вносить изменения, но даже этот процесс интегрирован в редактор.

Эта интеграция с редактором UE4 обширна и делает использование управления версиями неотъемлемой частью разработки с помощью UE4. Наличие редактора, выполняющего большую часть тяжелой работы, освобождает пользователей от сложностей управления версиями, гарантируя, что все члены команды используют управление версиями.

Я настоятельно рекомендую командам, стремящимся разрабатывать приложения на UE4, как можно быстрее освоить системы управления версиями. Один разработчик также может извлечь выгоду из безопасности и удобства размещения всех своих проектов на внешнем сервере.

Из-за большого количества доступных систем управления версиями и сложности изучения любой новой системы пошаговое руководство по использованию систем управления версиями выходит за рамки этой книги; однако некоторые фантастические официальные и общественные ресурсы могут помочь вам настроить собственный сервер управления версиями и начать использовать UE4 в командной среде в течение нескольких часов. Посетите сопутствующий веб-сайт для этой книги по адресу www.TomShannon3d.com/UnrealForViz для получения обновленного списка и информации об интеграции управления версиями файлов UE4.

Стоимость разработки на UE4

Для большинства художников и студий визуализации UE4 несет очень мало дополнительных затрат на программное или аппаратное обеспечение для начала работы. Редактор можно скачать бесплатно, а другие необходимые инструменты являются отраслевыми стандартами. Большинство инструментов, аппаратных средств и программного обеспечения, которые вы используете для создания традиционных визуализаций, хорошо послужат вам для создания контента для UE4. Конечно, вы можете рассчитывать на то, что потратите больше времени и денег на некоторые области.

Оборудование

Первое, что вам нужно для запуска UE4, — это хорошая видеокарта — чем быстрее, тем лучше. Хорошей новостью здесь является то, что UE4 лучше всего работает на видеокартах игрового уровня. Хотя профессиональные карты, такие как Quadro и Fire GL, могут предложить некоторые преимущества надежности, высококачественная игровая карта обычно стоит гораздо дешевле и обеспечивает гораздо большую производительность.

Вам также нужно много быстрого локального хранилища. В то время как вы могли бы хранить большую часть своего контента и проектов на общем сервере, UE4 предназначен для запуска с локальной машины пользователя.

Проекты UE4 могут очень быстро увеличиваться в размерах, а некоторые проекты могут работать более чем на 50 ГБ. Даже небольшие проекты могут занять много места на диске.

Время разработки

Хотя большая часть вашего рабочего процесса останется неизменной — подготовка клиентских данных, организация сцены, создание и применение материалов и освещения, создание траекторий и видов камер и т. д. — как только вы добавите интерактивность, это будет совершенно новая игра.

Программирование — это требование, которое большинство традиционных визуализаций обычно не имеют, а большинство студий и частных лиц не имеют большого опыта с ним. Даже самая простая интерактивная визуализация требует некоторого уровня сценариев и логики пользовательского интерфейса. Вход игрока должен быть прочитан, а взаимодействие — запрограммировано. Программирование и написание сценариев быстро станут существенной частью большинства бюджетов на разработку.

Затраты на разработку могут быстро расти и умножаться. Сложность интерактивной визуализации возрастает, казалось бы, экспоненциально по мере добавления или развития объектов. При разработке интерактивных визуализаций возникает нечто вроде эффекта ящика Пандоры. Одна особенность неизбежно влияет на другую, увеличивая оба предполагаемых времени разработки. Определение того, когда функции будут взаимодействовать, и соответствующее увеличение бюджетов чрезвычайно важно и должно быть строго учтено при разработке сферы охвата вашего проекта.

Тестирование и QA

Наряду с накладными расходами на логику и программирование часто возникают скрытые или упускаемые из виду затраты: тестирование и контроль качества. По мере того как ваши приложения визуализации становятся все более сложными и/или когда

они становятся общедоступными, поиск, отслеживание и исправление ошибок могут стать неожиданно трудоемкой частью вашего цикла разработки.

Помните об этих затратах и не продавайте себя в убыток при составлении бюджета проектов с использованием UE4. Вы должны предоставить достаточно времени для разработки программного обеспечения части интерактивной визуализации. Эти затраты могут быть такими же, если не больше, чем затраты на создание 3D-моделей и сцен, особенно если вы начинаете и должны развивать свои навыки разработчика.

Экономия затрат UE4

Разработка в Unreal Engine 4 может показаться дорогостоящим предложением, особенно если учесть время, которое вы и ваша команда тратите на обучение и адаптацию к новому способу работы. Однако некоторая существенная экономия затрат происходит просто на основе конвейера разработки и технологии UE4.

Освободившись от интенсивного рендеринга с привязкой к CPU и массивных наборов данных, генерируемых рендерингом кадров, необходимость в выделенной или облачной рендер-ферме устраняется или значительно уменьшается. Одна рабочая станция может заменить десятки, а иногда и сотни тысяч долларов, потраченных на рендеринг оборудования, лицензионные платежи за программное обеспечение, техническое обслуживание и содержание или хостинг. Если у вас уже есть настроенная ферма рендеринга, вы можете использовать ее для ускорения рендеринга Lightmass с помощью агента Swarm и координатора для распределения рабочей нагрузки рендеринга так же, как и другие инструменты распределенного рендеринга, с которыми вы, возможно, знакомы.

Если вы не используете UE4 для создания визуализаций, вы также можете увидеть огромную экономию на хранении данных. Когда вы визуализируете интерактивное приложение, каждый кадр заменяет предыдущий и тот буквально выбрасывается, а не сохраняется на диске.

Unreal Engine 4 просто не требует массивной инфраструктуры рендеринга, необходимой для рендеринга с трассировкой лучей. Без необходимости в рендер-фермах, огромных решениях для хранения данных или сверхскоростных сетях вы можете достичь гораздо большего с меньшими капиталовложениями, используя UE4. Как и большинство аспектов разработки с Unreal Engine 4, существуют компромиссы, и стоимость разработки в UE4 ничем не отличается. Вы столкнетесь с увеличением времени производства и усилий, но будете вознаграждены рендерингом в реальном времени и всеми преимуществами, которые это приносит.

По мере того как вы и ваша команда освоитесь с инструментами и начнете создавать собственные надежные рабочие процессы с учетом ваших специализированных конвейеров визуализации, источников данных, требований клиентов и ожиданий, затраты

могут резко снизиться. Однако имейте в виду: как и все проекты визуализации, каждый проект UE4, который вы делаете, бросает вам вызов новыми и неожиданными способами. Это не недостаток инструментов; это потому, что Unreal Engine 4 настолько обширен, мощен и расширяем, что почти нет предела тому, чего вы можете достичь с его помощью. Воображение вас и ваших клиентов будет раскрыто, и вы станете подталкивать друг друга к достижению следующей великой цели, поэтому бюджетируйте соответственно.

Ресурсы и обучение

Два самых больших различия между программным обеспечением профессионального уровня и всеми другими программами — это документация и поддержка. Без всесторонней, актуальной документации и доступа к системе поддержки принятие любого программного приложения может быть сложным или совершенно невозможным. Способность обучать свою команду, находить своевременные решения проблем, отслеживать и вносить свой вклад в разработку используемых инструментов имеет важное значение для плавного и предсказуемого цикла разработки.

Unreal Engine 4 — одно из самых хорошо документированных приложений, которые я когда-либо использовал. Официальная документация является всеобъемлющей, хорошо написанной и актуальной, несмотря на огромные обновления, которые происходят в приложении на регулярной основе. Новые функции вводятся с подробной документацией и примерами проектов, и понятные, живые тренинги проводятся несколько раз в неделю, где фактические разработчики инструментов демонстрируют их. Во время этих прямых трансляций аудитория может задавать вопросы, получая немедленное индивидуальное обучение и обратную связь по движку непосредственно от разработчиков.

Несмотря на огромные усилия Epic Games по созданию документации мирового уровня и учебных материалов, включая эту книгу, проекты, над которыми работают разработчики, настолько уникальны, что невозможно охватить каждый случай использования без сотен людей, работающих над созданием документации. Именно это и произошло.

Поддержка сообщества

С самого начала Epic Games обратилась к сообществу разработчиков, чтобы помочь с Unreal Engine 4, и сообщество отреагировало на это огромным образом. Члены сообщества сразу же начали выпускать удивительные обучающие и демонстрационные материалы, чтобы помочь привлечь новых разработчиков как можно быстрее. Epic Games признала эти усилия и оказала огромную поддержку сообществу разработчиков UE4 в виде маркетинга для разработчиков, грантовой программы Unreal Dev, которая присуждает разработчикам сообщества денежные гранты, прямую поддержку группам пользователей по всему миру и многое другое.

Epic Games пошла на огромный риск, выпустив свое самое ценное в мир бесплатно. Я считаю, что одна из ключевых причин такого успеха UE4 заключается в том, что она не только приняла и взрастила сообщество, но и сама стала членом коммьюнити разработчиков.

Marketplace

Продолжением ориентированной на сообщество бизнес-модели Epic является Unreal Engine Marketplace. Marketplace позволяет разработчикам продавать контент непосредственно другим разработчикам. Сотни ассетов доступны во всех мыслимых категориях. 3D-модели, материалы, полные исходные файлы игр, анимация, музыка и звуковые эффекты, а также системы частиц могут быть приобретены на рынке один раз для каждой студии и использованы в любых проектах UE4 этой студией или отдельным лицом с этого момента.

Эти ассеты, конечно, различаются по качеству и уровню поддержки, потому что они не созданы Epic Games; однако Epic Games имеет строгие стандарты контроля качества и систему проверки сообщества, чтобы гарантировать, что все доступные ассеты имеют хорошее качество.

Ассеты из Marketplace могут не только сэкономить сотни часов разработки вашего проекта, но и стать удивительными учебными ресурсами. Вы можете разобрать каждый ассет, который покупаете на рынке, и подробно изучить, как он работает. По мере того как вы узнаете больше, вы можете модифицировать эти ассеты в соответствии с вашими потребностями или интегрировать их функции в собственные технологии.

Сообщество разработчиков игр всегда любило собираться вместе, чтобы поговорить о магазине, украдкой взглянуть на новые игры друг друга и учиться друг у друга. Встречи и конференции по разработке игр проходят почти в каждом крупном городе США, Европы и Азии. Разработка игр — это огромная индустрия. Только в США игроки тратят на видеоигры 25,3 миллиарда долларов в год. Быстрый поиск в Google может быстро выявить события рядом с вами. Хотя большинство из них не могут быть специфичными для UE4, безусловно, будут присутствовать некоторые разработчики UE4 и, возможно, даже некоторые разработчики интерактивной визуализации. Случались и более странные вещи.

Официальных групп пользователей, поддерживаемых непосредственно Epic Games, немного, но их число растет. Если вам посчастливилось получить доступ к этим группам, они являются отличными ресурсами для создания связей, обучения и непосредственного участия в сообществе разработчиков UE4. Тысячи разработчиков Unreal Engine 4 также встречаются неофициально, используя такие инструменты, как meetup.com, чтобы найти рядом с ними единомышленников-разработчиков.

Крупнейшим собранием разработчиков игр в США является Game Developer's Conference (GDC). Недельное собрание, проводимое в Сан-Франциско каждый год,

позволяет разработчикам из самых разных слоев общества собираться вместе, чтобы учиться, общаться и обмениваться идеями. Беседы, тренинги и круглые столы от разработчиков премьерных и инди-игр заполняют дни, в то время как вечера заполнены вечеринками и событиями по всему городу. GDC также имеет обширную выставочную площадку с производителями и разработчиками, демонстрирующими все — от новейшего программного обеспечения для управления разумом до новейшего игрового движка или устройства ввода. Ярмарка вакансий привлекает сотни нетерпеливых художников, программистов и дизайнеров, ищущих свой первый большой прорыв в индустрии интерактивных развлечений. Если вы можете позволить себе присутствовать, я настоятельно рекомендую ее каждому серьезному разработчику интерактивной визуализации. Несмотря на то что GDC не сфокусирована только на индустрии визуализации, участие даст беспрецедентный опыт взаимодействия с сообществом для последующих переговоров.

Где бы вы ни жили, я надеюсь, что вы сможете попасть на собрание сообщества разработчиков интерактивных приложений и игр, а также, надеюсь, на группу, ориентированную на UE4. Все они очень страстная компания, и эти события всегда помогают прекрасно проводить время.

Заключение

Начало работы с разработкой интерактивной визуализации и Unreal Engine 4 в частности — это процесс с фантастическим вознаграждением, который дает вам, вашей команде и вашим клиентам творческие возможности, а вашей аудитории — новые способы узнать и испытать истории, которые вы рассказываете как профессионалы в области дизайна.

Теперь у вас должно быть лучшее понимание того, что такое UE4, что он предлагает для визуализации, и обзор того, как UE4 отличается от того, что вы можете использовать для разработки визуализаций. Вы также должны иметь некоторое представление о том, какие проекты являются хорошими вариантами для вашего первого проекта UE4, и понимать некоторые проблемы, с которыми вы можете столкнуться в будущем, и как избежать некоторых из них.

ГЛАВА 2

РАБОТА С UNREAL ENGINE 4

Разработка интерактивной визуализации в UE4 — это искусство разработки ПО в той же мере, что и искусство визуальной коммуникации. Рабочие процессы и инструменты, на которые вы полагались в течение многих лет, могут не работать или даже усложнять ситуацию. Unreal Engine 4 предполагает, что вы будете делать все определенным образом, чтобы весь контент рендерился настолько быстро, насколько это возможно в движке. Если вы знаете, чего ожидать (и что UE4 ожидает от вас и вашего контента), вы можете максимально использовать возможности Unreal Engine 4 для разработки динамичных, притягательных визуализаций, которые находят отклик у вашей аудитории.

Компоненты Unreal Engine 4

UE4 — это не одно приложение, а скорее набор инструментов, приложений и данных, которые работают совместно, помогая управлять, разрабатывать и развертывать приложения реального времени на различных платформах. Среда разработки UE4 удивительно хорошо организована, последовательна и наполнена учебными и справочными ресурсами. Editor — это хорошо реализованная часть UE4, построенная вокруг идеи сделать разработку интерактивного контента такой же увлекательной и очаровательной, как использование проектов и игр, разработанных с его помощью.

Понимание компонентов Editor и то, как они действуют вместе, может дать вам более полное представление о «закулисной» работе UE4. Это также позволит вам лучше понять, как подходить к ежедневной работе с UE4.

Epic Games Launcher

Если вы используете компьютеры на Mac или Windows, когда впервые устанавливаете UE4 с unrealengine.com, скачивается **Epic Games Launcher** (рисунок 2.1). Это легкое приложение служит в качестве универсального магазина для всех ваших нужд Unreal Engine 4. Он предоставляет доступ к огромному разнообразию информации, контента, проектов с разными примерами и многое другое.

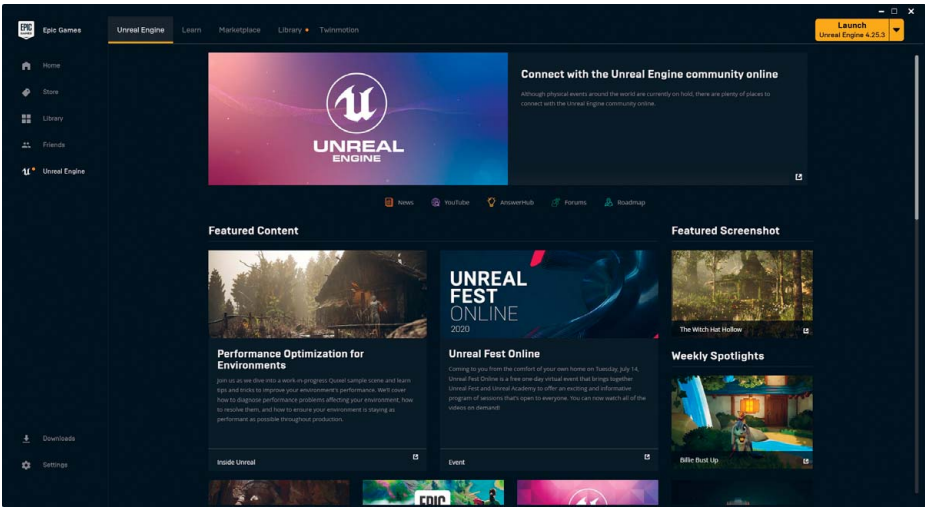


Рисунок 2.1 Epic Games Launcher

Сообщество

Секция **Сообщество** включает в себя последние новости и основные моменты сообщества UE4.

Это прекрасное место, чтобы познакомиться с новшествами, которые вносятся в UE4, и фантастический источник вдохновения для ваших собственных проектов. Epic Games стремится помочь своим разработчикам добиться успеха, и продвижение контента в Launcher и через социальные сети Epic является одним из лучших способов достижения этой цели.

Здесь также есть ссылки на форумы, онлайн-документацию и другие сообщества, и официальные ресурсы, такие как план разработки движка и AnswerHub.

Learn

Секция **Learn** — это сокровищница учебных ресурсов, проектов с разными примерами и контента. Каждый из этих проектов предлагается на тех же либеральных условиях, что и остальной контент, входящий в состав движка; вы можете использовать что угодно из доступного в ваших личных и коммерческих проектах бесплатно.

В секции есть туториалы, wiki, видео и доступный для скачивания контент: эффекты, материалы, текстуры и т. д. Я призываю вас скачать Launcher и исследовать это все самостоятельно. Вы узнаете удивительные вещи, наблюдая, как люди, создавшие движок, используют его, чтобы воплотить в жизнь собственные видения.

Marketplace

Marketplace — это место, где разработчики и художники могут покупать и продавать почти любой тип контента (3D-модели, анимации, эффекты и т. д.) и плагины, которые увеличивают возможности UE4.

Тысячи разработчиков на UE4 создают бесплатный и платный контент практически всех видов, которые только можно себе представить. Дополнительные материалы доступны на сторонних веб-сайтах, таких как Gumroad.com или собственных сайтах разработчиков.

Прежде чем разрабатывать что-то в UE4 самостоятельно, сделайте поиск в Google; высока вероятность, что вы сразу найдете что-то доступное «с полки» для использования.

Library

Library — это место, где вы управляете, устанавливаете, удаляете или обновляете ваши версии движка. Вы также можете просматривать и управлять своими проектами UE4 в Library и управлять вашим Marketplace-контентом.

Заметка

Лаунчер — это фактически приложение Unreal Engine 4! Вы можете найти его вместе с другими установками UE4 в папке приложения.

Прочий контент в UE4 и от Epic

В верхней части Launcher доступны для загрузки другие игровые проекты Epic Games и популярные проекты для UE4. Некоторые из них являются коммерческими играми, например Paragon и Fortnite; другие — играми с моддинговыми компонентами, написанными UE4 и поддерживаемые через UE4 Marketplace.

Epic Games разрабатывает новую версию Unreal Tournament и предоставляет общественности полный доступ к исходному коду игры по мере ее разработки. Сообщество активно помогает развивать игру вплоть до уровня кода и художественного уровня. Если вы когда-нибудь хотели увидеть, как профессиональная компания по разработке видеоигр делает массовую AAA-игру, вы можете посмотреть или даже принять участие в проекте, внеся свой вклад.

Движок UE4

Движок находится в папке приложения и представляет собой набор кода и ресурсов, необходимых для запуска приложений UE4. Движок — это базовый код, на основе которого строятся Launcher и Editor, он содержит весь рендеринг, физику, пользовательский интерфейс и другие коды и инструменты, необходимые для сборки и запуска приложений, разработанных с помощью Editor.

Версии движка

Unreal Engine 4 регулярно обновляется как с основными апдейтами, так и с незначительными исправлениями ошибок. UE4 использует десятичную схему нумерации для описания каждого выпуска. Обновления версий обозначаются первым десятичным числом. Например Unreal Engine 4.24 представляет собой крупное обновление с версии 4.23. Новые функции, обновленные основные компоненты и другие большие изменения обычно включаются в эти выпуски.

Хотфиксы и исправления ошибок описываются с помощью второго десятичного числа: Unreal Engine 4.24.2 содержит небольшие исправления ошибок и другие незначительные изменения в версии 4.24.1.

Заметка

Launcher позволяет установить более одной основной версии UE4 одновременно, но только один экземпляр каждой версии. Это означает, что вы можете установить 4.23.3 и 4.24.1 одновременно, но не 4.23.3 и 4.23.2[2].

[2] Это относится только к установкам движка, управляемым пусковой установкой. Пользователям Linux и проектам, требующим модификации движка, может потребоваться специальная сборка, которая обычно компилируется из исходного кода и управляется вне Launcher. Нет никаких ограничений на количество, расположение или номера версий этих сборок ядра.

Обновление проекта

Обновление проектов UE4 до более новых версий движка поддерживается, но понижение — нет. Файлы, сохраненные в новой версии редактора, не могут быть открыты более старыми версиями. Обычно вы должны запускать новые проекты в последней доступной версии движка и обновлять их только по мере необходимости. Если что-то не сломано, не чините это, иначе вы можете сломать.

Обновление проектов и контента с одной версии движка на другую, как правило, проходит гладко, но может иметь некоторые неожиданные сбои. Сторонние плагины, например, часто требуют времени для обновления до последней версии движка.

Epic Games предоставляет четкие указания в примечаниях к выпуску, описывающие внесенные изменения, которые повлияют на обновленные проекты. Всегда подходите к задаче обновления существующих проектов с осторожностью и постарайтесь делать это только в том случае, если ваш проект существенно выиграет от функций или исправлений в обновленном движке.

UE4 Editor

UE4 Editor — это основной интерфейс для создания, тестирования, упаковки и программирования проектов UE4. Editor — это набор инструментов и интерфейсов для импорта, организации, оптимизации, программирования и создания привлекательного интерактивного контента.

Он предлагает инструменты для создания систем частиц, надежных систем искусственного интеллекта, моделирования транспортных средств, сетей, мультиплеера, виртуальной реальности, реалистичного освещения и материалов, пользовательских интерфейсов и кинематографического секвенсора. Он обеспечивает почти неограниченную функциональность с помощью визуального программирования и — хотите верьте, хотите нет — гораздо больше. Редактор UE4 — это *массивная* платформа для разработки на профессиональном уровне.

Заметка

Как и Launcher, UE4 Editor — это приложение, построенное с помощью Unreal Engine 4 в качестве графического интерфейса для создания других приложений Unreal Engine 4!

Редактор также предоставляет вам легкий доступ ко множеству параметров конфигурации, инструментов для построения мира и развертывания, а также инструментов отладки и профилирования, которые помогут вам диагностировать проблемы производительности, чтобы сделать визуализации максимально привлекательными.

Проекты UE4

Большинство приложений, традиционно применяемых для визуализации, используют один файл для описания всей сцены. Этот файл обычно сильно увеличивается по мере роста сцены. Файл Photoshop становится больше по мере добавления слоев, а файл Max растет по мере добавления геометрии. Этот единственный файл может быть скопирован с диска на диск и часто содержит *все* необходимое для визуализации сцены[3].

Каждый **проект** в Unreal Engine 4 представляет собой не один файл, а целый набор — исходного кода, файлов контента, плагинов, конфигурационных файлов и других вспомогательных файлов, хранящихся в одном каталоге. Этот каталог и содержащиеся в нем компоненты являются проектом, а не просто одним файлом (рисунок 2.2).

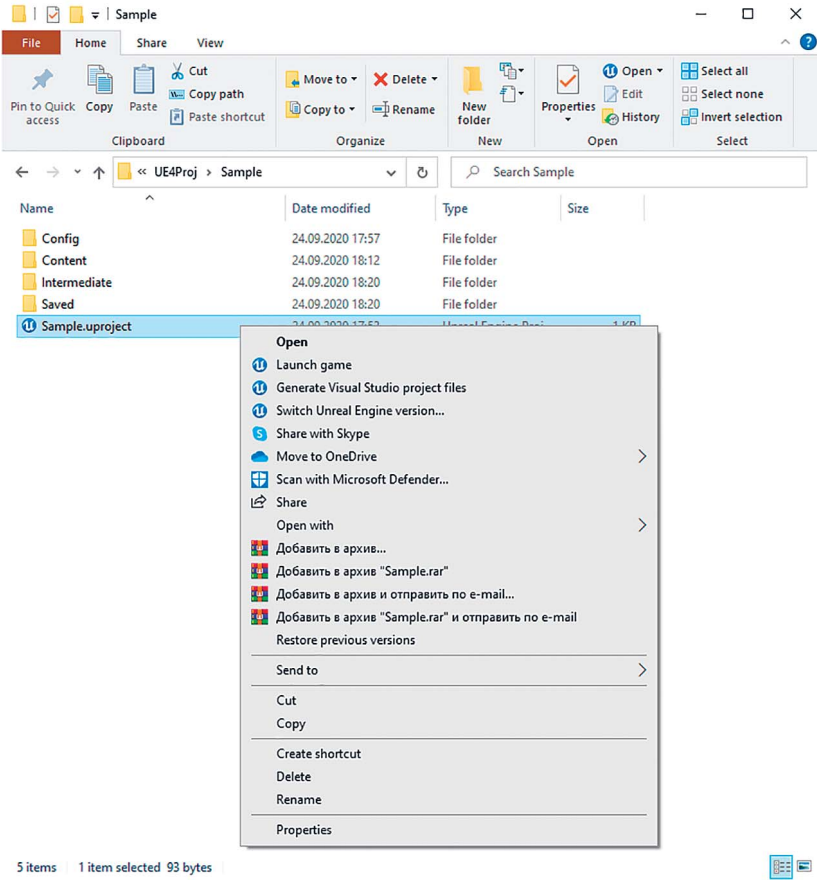


Рисунок 2.2 Каталог проекта Unreal и контекстное меню (Windows 10)

[3] Конечно, и Photoshop, и 3ds Max позволяют создавать ссылки на внешние файлы (например, текстуры), но эта структура не применяется. Вы можете хранить свои текстуры на отдельном диске или в сетевом расположении, которое находится в тысячах миль от вас.

В корневой директории проекта есть файл с расширением **.uproject**, который сопоставляет ваш проект с определенной версией UE4 и установленными плагинами. Этот файл не растет и даже не сохраняется или меняется очень часто. На самом деле это просто текст, который можно открыть и просмотреть с помощью текстового редактора.

Ниже представлен типичный файл **.uproject**:

```
{
        "FileVersion": 3,
        "EngineAssociation": "4.25",
        "Category": "",
        "Description": "",
        "Plugins": [
                {
                        "Name": "Substance",
                        "Enabled": true,
                        "MarketplaceURL":
                        "com.epicgames.launcher://ue/marketplace/
                                content/2f6439c2f9584f49809d9b13b16c2ba4"
                }
        ]
}
```

Двойное нажатие по файлу запускает редактор UE4 версии, к которой привязан проект, если данная версия у вас установлена, или предлагает установить привязку с другой версией движка.

Щелчок правой кнопкой мыши по файлу **.uproject** дает вам несколько удобных опций, например запуск проекта без Editor при помощи команды контекстного меню Launch Game.

Это отличный способ обойти Editor в повседневной работе, и это мой любимый способ запуска проектов.

Source Art

Unreal Engine удивителен и способен помочь вам создать потрясающие миры. Однако он не заточен на создание основной части контента, наполняющего эти миры. И, разумеется, это наполнение делаете вы при помощи 3D- и 2D-приложений, которые знаете и любите (например, Photoshop, 3ds Max и Maya). UE4 не имеет никаких оснований конкурировать с возможностями этих приложений или требовать от вас изучения совершенно нового способа создавать и подготавливать контент.

В любом случае, скорее всего, вы создадите почти все ассеты в программах, которые вам хорошо знакомы, или в других, с которыми вы, возможно, знакомы не так близко, например Substance Painter, ZBrush и Blender. Каждый день разрабатываются новые инструменты; это очень быстро развивающаяся экосистема, и, передавая

создание контента на аутсорс этим приложениям, Epic Games может сосредоточиться на развитии возможностей UE4 для создания привлекательных интерактивных миров.

В визуализации исходными данными могут быть также данные САПР, научные данные или GIS-информация. Независимо от исходного приложения или формата вы можете рассматривать все это как часть своей арт-галереи.

Вы создаете, храните, имеете доступ и меняете исходные арт-галереи *полностью за пределами* проекта UE4. Сохраните существующие проверенные рабочие процессы и поменяйте их в соответствии с соглашениями и требованиями UE4. Unreal Engine 4 не знает, откуда берутся данные, пока они экспортируются или сохраняются в формат, который может импортировать UE4.

Структура проекта

Структуру папок проекта UE4 необходимо строго соблюдать. Проект не может загружать или ссылаться на любой файл вне каталога проекта, именно поэтому они являются портативными и автономными. Проекты могут быть скопированы и синхронизированы между рабочими станциями легко и надежно.

В каждой папке проекта находится еще несколько папок. Понимание того, что они такое, а также что вы должны и не должны делать с каждой из них, очень важно.

Config

Каталог **Config** содержит настройки для каждого проекта в текстовых файлах с расширением **.ini**. Настройки для каждой функции в движке представлены в этих файлах. Существуют *тысячи* отдельных настроек, и вы можете оставить большинству из них значение по умолчанию.

Почти все наиболее важные параметры конфигурации отображаются на вкладках Preference в Editor (рисунок 2.3). Вы можете непосредственно менять настройки, используя удобный пользовательский интерфейс с полезными контекстными подсказками, которые описывают функцию и часто используют варианты для каждой настройки. **.ini**-файлы обновляются на лету, и только некоторые настройки требуют перезапуска редактора для вступления в силу. Если вы измените параметр, требующий перезагрузки, редактор уведомит вас об этом.

Вы *всегда* должны включать папку Config при копировании, совместном использовании или синхронизации вашего проекта с другими пользователями. Если между копиями проектов будут разные файлы конфигурации, это может вызвать множество труднодиагностируемых проблем.

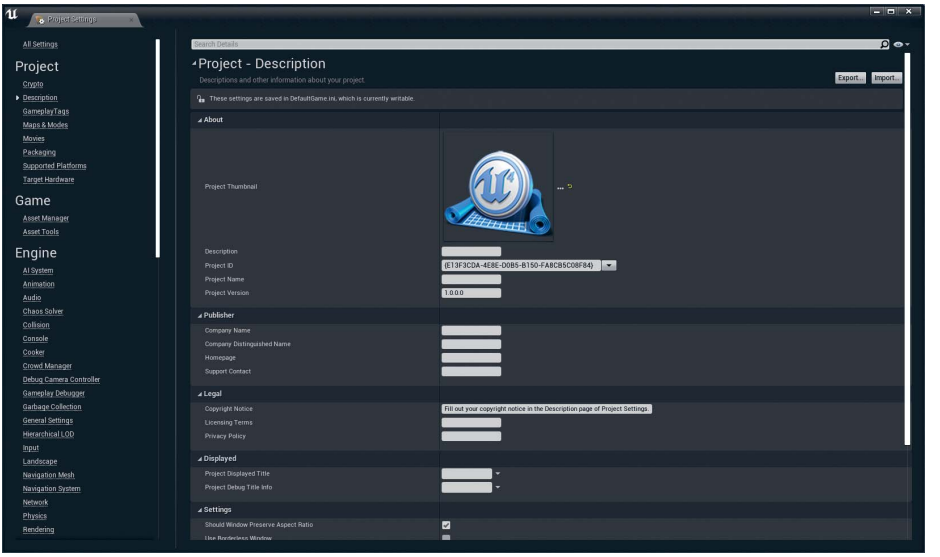


Рисунок 2.3 Диалог настроек проекта UE

Saved

Saved — это временная папка, которая генерируется во время запуска проекта. Скриншоты, логи и другие файлы, которые сохраняются во время игры, лежат здесь. Эта папка также содержит файлы автоматического резервного копирования.

Обратите внимание, что папка может серьезно увеличиться в размерах и стать одной из основных причин, по которым не получается разместить проект в сети или общей папке. Аналогично, если вы копируете свой проект или используете систему управления версиями (вы же используете систему управления версиями, не так ли?), то нужно исключить эту папку, чтобы сэкономить время и пространство и избежать конфликтов конфигурации.

Plugins (Опционально)

Много плагинов доступно для UE4, и пока некоторые из них основаны на контенте и требуют просто расположения в папке Content, то другие должны располагаться в директории Plugins и/или быть скомпилированы из папки Source.

Эти плагины в итоге, как и следовало ожидать, попадают в каталог **Plugins**. Здесь общие файлы для конкретной платформы (.dll на ПК, .so на Linux и так далее), которые сгенерированы из C++ или загружены, устанавливаются. Эти файлы читаются Editor и игрой и могут быть изменены практически любым способом, который только можно себе представить — например, добавлением дополнительной поддержки импорта файлов или новых функций рендеринга.

Все члены команды должны иметь полное содержимое этой папки.

Content

В **Content** хранятся все ресурсы, которые используются для создания игры. Обычно там лежат только два типа файлов: **.uasset** и **.umap**. Эти файлы являются обобщенными обертками и представляют все ассеты и Levels в проекте.

> **Заметка**
>
> *Никогда* не меняйте папку Content вне редактора UE4. *Все* управление файлами выполняется из UE4 Editor. Перемещение, переименование, удаление или иное изменение объекта **.uasset** и файлов **.umap** *создадут* пустые ссылки. Это очень плохо. Вдвойне плохо.

Видеть папку, полную файлов с одинаковым расширением, и думать о них как о текстурах, материалах, анимации, 3D-моделях, чертежах, звуках и т. д. может быть нелогичным. Мы часто используем расширения файлов в нашей ежедневной работе с компьютерами, чтобы различать типы файлов и их функции. Unreal Engine 4 использует данные внутри каждого из файлов **.uasset**, которые позволяют определить, что это такое и как он должен отображаться в редакторе и использоваться в игре во время выполнения.

Intermediate

Директория **Intermediate**, как и Saved, генерируется во время запуска. Она содержит файлы, необходимые для компиляции и запуска игр или ускорения повседневной работы Editor, например с шейдерами или кэшем геометрии.

Как и Saved, эту папку следует исключать и не передавать, когда копируете или версионируете ваш проект.

Строение файлов .uasset

UE4 требует, чтобы почти все содержимое — модели, текстуры, аудио и анимация — было импортировано в движок и сохранено в папке Content проекта как файлы **.uasset**. Каждый файл **.uasset** также называется **Content Package**. В каждом пакете находится исходный файл, который был импортирован, а также куча другой информации, относящейся к конкретным ассетам. Движок конвертирует эти **.uasset** файлы на лету в правильный формат для платформы, на которой вы работаете и развертываетесь.

Это означает, что один файл **.uasset** можно использовать для развертывания контента на Mac, PC, mobile, VR или любой другой платформе, поддерживаемой UE4, без

каких-либо изменений **.uasset** или доступа к исходным кодам. Движок делает конвертацию в нужный формат для вас в фоновом режиме. Вы даже можете экспортировать исходный файл (изображение, модель и т. д.) обратно из **.uasset**-файла в Editor для изменения, а затем повторно импортировать измененный файл обратно в проект.

В Content Browser движок и Editor UE4 будут читать и отображать только файлы **.uasset**.

Организация производственного процесса в Unreal Engine 4

У UE4 отшлифованные и простые в освоении рабочие процессы почти для каждого аспекта и функции движка. Несмотря на то что некоторые рабочие процессы могут быть требовательными или трудоемкими, они последовательны. Если вы будете следовать правилам и делать все так, как требует UE4, то получите высокую частоту кадров, надежные результаты и потрясающие визуализации.

С точки зрения управления производственными процессами, важно, что типичный проект UE4 имеет предсказуемый рабочий процесс. Практически каждый проект требует от вас выполнения нескольких задач (очень похожих), приводящих к созданию продукта, готового к передаче в руки клиентов.

Создание нового проекта

UE4 предлагает несколько способов настройки нового проекта. Вы можете создать его, запустив движок непосредственно из раздела Library в Launcher. Или вы можете создать новый проект из меню File в Editor UE4.

Заметка

Я *не рекомендую* использовать сетевой диск для размещения вашего проекта индивидуальной или общей разработки. UE4 не предназначен для работы подобным образом и не доставит вам никакого удовольствия. Используйте **программное обеспечение контроля версий**, чтобы позволить нескольким разработчикам работать над одним проектом.

Храните свой проект и работайте над ним, используя быстрый локальный диск. Использование жесткого диска позволяет загрузить самый большой уровень или ассеты в считаные секунды, а также сокращает время, затрачиваемое на импорт и обработку ассетов, что значительно ускоряет рабочий процесс.

Создание контента

Прежде чем вы сможете приступить к большей части визуализации, необходимо создать контент. Вы уже профессионал в этом деле. Существующие рабочие процессы и системы, как правило, продолжают работать с незначительными изменениями. Если вы создаете 3D-геометрию, вы, вероятно, можете импортировать ее в UE4.

Экспорт контента

Вы должны экспортировать почти весь контент, который хотите открыть в UE4, в понятном для него формате. С каждый днем в Unreal Engine 4 импортируется все больше стандартных отраслевых форматов файлов, таких как FBX, TGS, PNG, PSD и другие. Однако UE4 не поддерживает импорт специфичных для приложения двоичных файлов, таких как у 3ds Max.

Многие из этих файлов являются временными, промежуточными и просто используются для обеспечения интероперабельности между различными инструментами создания контента и UE4. Вы должны сохранять исходные файлы и реэкспортировать их по мере необходимости.

Импорт контента в UE4

После создания и экспорта контента необходимо импортировать его в UE4. Это процесс создания **.uasset** Content Package для каждого импортированного ассета. Пакет **.uasset** обычно содержит копию импортированных данных и выше крыши других специфичных для ассетов метаданных и другой информации.

Вы можете автоматизировать большую часть этого процесса, используя скрипты и инструменты для создания экспорта файлов; импорт — обычно безболезненный процесс с поддержкой пакетного импорта.

Наполнение уровня контентом

После импорта контент будет отображен в **Content Browser**. Простым перетаскиванием можно переместить контент из Content Browser в 3D Viewport и начать наполнение мира.

Каждый добавляемый на уровень Mesh является экземпляром импортированного ассета, который ссылается на этот **.uasset**. Если вы перетаскиваете модель на сцену 30 раз, то создается не 30 копий, а только 30 идентичных *ссылок* на ассет в библиотеке, в то время как в оперативную память загружается только одна копия. Если вы поменяете **.uasset**, все ссылки на этот ассет будут обновлены, чтобы соответствовать изменениям.

Добавление интерактивности (программирование!)

На этом этапе вы сможете создавать статические траектории камеры и рендерить сцену в кадрах, которые затем будут соединены на видео, и качество отрендеренных сцен может быть очень высоким.

Однако вы много потеряете, если просто перетащите все ассеты в UE4 без добавления интерактивности.

Даже добавление простой орбитальной камеры позволяет пользователю управлять фотореалистичной визуализацией способами, невозможными на любой другой платформе.

В UE4 вы можете создавать удивительно богатые интерактивные миры, не написав ни одной строки кода, а используя только системы визуального программирования Blueprint. Вы также можете использовать C++ или один из доступных сценариев интеграции.

Тестирование и шлифовка

Если вы решите добавить интерактивности для повышения вовлеченности пользователей, то неизбежно столкнетесь с возникающими ошибками. Вероятно, вы привыкли работать с ошибками при работе по визуализации: от сбоев рендеринга до ошибок моделирования и ошибок данных. При работе с интерактивной визуализацией в UE4 они тоже будут. Кроме того, вам придется работать с новой проблемой взаимодействия с пользователем.

Создание гладко работающих, простых в использовании интерфейсов, сцен и приложений не так просто, как кажется. Пользователи традиционно много ожидают от создаваемых вами интерфейсов, и процесс точной настройки всех систем занимает гораздо больше времени и ресурсов, чем можно было бы подумать.

Чем больше возможностей вы добавляете, тем больше тестов и итераций потребуется, и чем шире ваша аудитория, тем более тщательным должно быть тестирование.

Отдайте приложение в руки вашего клиента/аудитории как можно скорее и, если возможно, понаблюдайте, как они его используют. Прислушайтесь к ним и соберите максимум отзывов о приложении как можно раньше. Это лучший способ избежать того, что процесс разработки растянется до 11 ночи в день дедлайна.

Упаковка проекта

После того как ваша интерактивная визуализация протестирована и отшлифована, созданный интерфейс аккуратен и удобен, пришло время запустить ее на компьютерах, планшетах и телефонах пользователей!

Для этого необходимо **упаковать проект**. Это многоэтапная операция. Во-первых, UE4 «готовит» весь контент проекта. Процесс просто берет все файлы **.uasset**, которые вы создали, и подготавливает их для любой платформы, на которой вы развертываете проект, и удаляет любое исходное содержимое или другой хлам, оставляя только самый эффективный файл, который и будет загружен системой.

Приготовленное содержимое объединяется с версией движка UE4, из которой были удалены Editor и другие инструменты разработки. Создается исполняемый файл, в результате чего получается автономная, полностью переносимая версия проекта, которую можно установить на машину, способную все это запустить!

Распространение

Чтобы ваше приложение было доступно клиентам или широкой публике, требуется немного больше усилий, чем отправка изображения по электронной почте или загрузка фильма на YouTube.

Установщики приложений UE4 обычно составляют несколько сотен мегабайт, часто больше гигабайта. Доступ по простой ссылке на Dropbox хоть и достаточен для отправки файлов нескольким людям, быстро перегружается, если даже несколько сотен человек единовременно получат к ней доступ.

Осуществление технической поддержки

Как только вы выпустите приложение, оно будет протестировано и использовано так, как вы и представить себе не могли. Ваша аудитория найдет несовместимость с удивительным набором аппаратных и программных средств, о существовании которых вы даже не подозревали, а тем более не проверяли совместимость с ними. Всегда закладывайте дополнительное время на тестирование и поддержку для любого проекта и масштабируйте сроки в зависимости от того, как сильно меняется размер вашей аудитории.

Заключение

Там, где вы, возможно, когда-то считали проект выполненным после наполнения сцены визуализации и нажатия кнопки Render, разработка интерактивной визуализации в UE4 вводит новую часть в процесс разработки визуализации. Вы должны учитывать это в своих проектах, в навыках сотрудников, а также в установленных временных рамках для разработки интерактивной визуализации.

ГЛАВА 3

ПРОИЗВОДСТВЕННЫЙ ПРОЦЕСС

Перевод существующих сцен и контента в UE4, скорее всего, будет одним из первых шагов, которые захочется совершить после запуска редактора. Это также может стать первым серьезным препятствием, особенно если вы хотите преобразовать большие существующие сцены визуализации с трассировкой лучей в UE4. Понимая, что нужно сделать для подготовки контента для UE4, вы можете существенно сэкономить время, сделав это «как требует Unreal» в первый раз, и получить качество и производительность уровня, ожидаемого от Unreal Engine 4.

Обзор производственного процесса

Вы, вероятно, привыкли работать в 3D-приложении с интегрированным моделлером, системой материалов и визуализатором. После того как данные занесены в 3D-приложение, вы можете визуализировать сцену одним нажатием кнопки. Вы применяете материалы и освещение, создаете анимацию и свои истории, используя инструменты в выбранном 3D-приложении. Вы можете подготовить текстуры и другие данные во внешних приложениях, но большую часть повседневной работы делаете именно в 3D-приложении.

После рендеринга изображения или анимации вы загружаете их в приложение для последующего редактирования — чтобы добавить эффекты и заголовки, отредактировать отснятый материал и добавить аудио. Когда вы завершите работу, окончательная версия будет отрендерена в видеофайл или изображение и передана вашей аудитории или клиенту (рисунок 3.1).

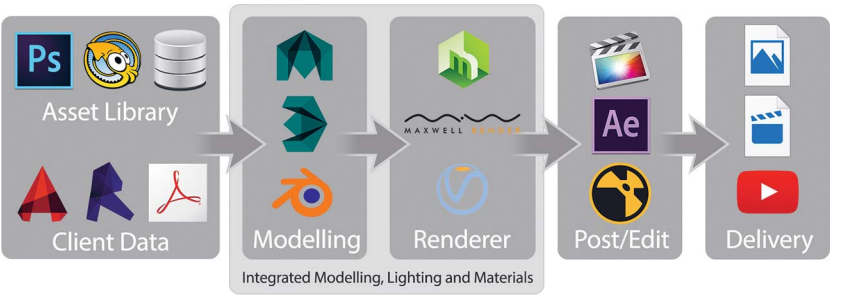


Рисунок 3.1 Традиционный процесс визуализации

UE4 является автономным приложением и не интегрируется непосредственно в любое 3D-приложение. Вместо этого вы используете 2D- и 3D-приложения для создания 3D-моделей и текстур, которые необходимо экспортировать в формат для обмена (FBX, TGA и т. д.) и импортировать в UE4 (рисунок 3.2).

Рисунок 3.2 Процесс визуализации с Unreal Engine 4

После импорта контента в UE4 вы создаете миры в UE4 Editor: добавляете освещение, материалы и интерактивность. Здесь нет кнопки рендеринга; Viewport рендерит все в реальном времени, производя изображения финального качества, включая эффекты постобработки, такие как размытие, глубина резкости и цветовая градация.

Чтобы добавить интерактивность, вы можете использовать Blueprints, C++ и Unreal Motion Graphics (UMG) для создания пользовательских интерфейсов и других экранных титров. Sequencer позволяет создавать фильмы голливудского качества с использованием редактора нелинейного монтажа, физических камер.

После завершения проекта вы можете запустить свое приложение на любых платформах: Mac, Windows, VR, мобильных и так далее. Вы также можете рендерить высококачественные кадры и анимации.

Подготовка контента

Будьте особенно осторожны при подготовке контента для использования в UE4. Для обеспечения плавного, быстрого рабочего процесса, высокой производительности и качественного рендеринга требуются соответствующие соглашения об именовании, Lightmap, UV map, коллизии, LOD и многое другое. Поскольку для подготовки ваших сцен нужно сделать еще очень многое, автоматизация и согласованность необходимы для обеспечения быстрого производства, стабильности и производительности вашего проекта, а также снижения производственных затрат.

Рабочий процесс UE4 требует намного больше времени выполнения, чем трассировка лучей. Почему? Вспомните это невероятно быстрое время рендеринга (0,016–0,033 секунды на кадр обсуждается в главе 1)! Один из лучших способов сделать процесс визуализации быстрее — выполнить обработку заранее и сохранить ее различными способами. Запекание освещения, запись деталей с использованием карт нормалей или даже скромный процесс присвоения UVW координат — все это освобождает ресурсы обработки.

Прекрасно, что время простоя в течение работы UE4 практически равно нулю. Экспорт, импорт и обработка данных в UE4 происходят *очень* быстро. Единственные процессы, которые занимают серьезные промежутки времени, обычно являются потоковыми и могут выполняться параллельно, что позволяет вам продолжать работать в Editor, пока запекается освещение или компилируется проект для запуска на другой платформе.

Заметка

Некоторые из наиболее долгих и трудоемких вычислений могут быть выполнены не только параллельно с Editor, но и распределенно по сети. Вы можете легко использовать существующие рабочие станции или рендер-ферму, чтобы обеспечить существенное повышение производительности рендеринга Lightmass, приготовления контента, компиляции и многого другого.

И вам не придется тратить большую часть дня на ожидание рендеров. Viewports в Max и Maya, как известно, ужасны для передачи того, как будет выглядеть освещенная визуализированная сцена, часто даже не показывая должным образом материалы и текстуры. Это заставляет вас визуализировать свою сцену для предварительного просмотра каждого изменения, затрачивая минуты, а иногда и часы, чтобы получить хорошее представление о том, как будет выглядеть окончательное изображение.

UE4 быстро загружает контент. Content Browser открывает ассеты и позволяет просматривать их в режиме реального времени, а различные редакторы быстро открываются даже со сложным контентом. Массивные уровни сохраняются и открываются в считаные секунды, даже не минуты. Результатом является интерактивный рабочий процесс, что означает, что вы можете сделать больше за меньшее время и, если станете следовать некоторым нехитрым рекомендациям, большая часть работы будет выполнена за вас автоматически волшебными средствами.

Настройка 3D-сцены

Вы всегда должны следить за тем, чтобы контент соответствовал стандартам и соглашениям, принятым в UE4. Большинство из них носят технический характер, многие — стилистический, но все они имеют хорошую аргументацию.

Вы должны начать с организации исходных художественных файлов (файлов Max, Maya и Photoshop). Организация — это название игры, и очень важно найти время, чтобы сделать все правильно.

Изменение имен, масштаба, количества вершин, точек разворота и т. д. — все это очень легко сделать в приложениях для создания цифрового контента (DCC). Просто нажмите, переименуйте, переместите и масштабируйте. Эти приложения разработаны для изменения мешей и растровых изображений и хороши в этом, и вы, скорее всего, отлично владеете ими. Делать все это будет гораздо сложнее (если вообще останется возможно) после того, как вы разместите свой контент в UE4.

Единицы измерения

Unreal Engine 4 по умолчанию использует сантиметры в качестве **единицы измерения** на сцене. В идеале всегда устанавливайте свои приложения для работы в масштабе 1 единица измерения равна 1 см. Однако для многих визуализаций это не так просто, как кажется. Мы часто обязаны ограничивать исходные данные или другие императивы рабочего процесса, которые мешают нам легко выполнять подобное преобразование.

Существует несколько способов решить эту проблему. Экспортер FBX позволяет применять масштабирование к мешам во время экспорта, а UE4 предлагает настройку масштаба импорта (и поворота) для мешей. К сожалению, эти настройки могут

сработать или не сработать в зависимости от контента, поэтому не рекомендованы к использованию, если вы в силах самостоятельно найти способ изменить масштаб содержимого перед экспортом.

Max также предлагает возможность установки **Display Units**. Этот параметр *ничего* не делает с данными или сценой. Это просто преобразование масштаба единицы измерения сцены в визуальные элементы пользовательского интерфейса, что позволяет вам сохранять содержимое в сантиметрах и продолжать работать со знакомыми единицами измерения.

Статистика

Получение точной **статистики** очень важно для оптимизации вашего контента. Самое важное оттуда — число треугольников и вершин. Обратите внимание, что я сказал *треугольник*, а не *полигон* или *грань*, и не использовал любую другую терминологию. UE4 разбивает всю геометрию на треугольники, определенные вершинами для визуализации, поэтому все остальное не имеет значения для наших нужд.

Видимость полигонов и нормали

Backface Cull — это оптимизация рендеринга, при которой треугольники не отображаются, если они обращены в другую сторону от камеры. Этот эффект может быть вам известен как **two-sided** или **double-sided** (двусторонний) рендеринг.

Многие 3D-приложения имеют двустороннюю визуализацию, включенную по умолчанию, показывая обе стороны объектов. Это может дать вам неправильное представление геометрии, поскольку UE4 по умолчанию убирает повернутые назад треугольники. Вы можете настроить отдельные материалы и экземпляры материалов в UE4, чтобы они были двусторонними. Такое исправление, однако, не эффективно и не предназначено для исправления плохого контента, а является функцией для визуализации определенных поверхностей, таких как листва.

Вы должны отключить двустороннюю визуализацию в ваших 3D-приложениях, чтобы убедиться, что вы видите ту же модель, что и в UE4. Исправьте проблемы с гранями и вершинами в вашем 3D-приложении перед экспортом в UE4.

Подготовка геометрии к UE4

Unreal Engine 4 позволяет работать такими 3D-объектами, как к **Static Meshes** (статические меши) и **Skeletal Meshes** (скелетные меши). Один меш может иметь Smoothing Groups, несколько материалов и цветов вершин. Меши могут быть жесткими (статические меши) или деформирующимися (скелетные меши). Вы экспортируете каждый меш в виде файла FBX из 3D-приложения, а затем импортируете его в свой проект UE4.

Заметка

В этой книге для построения сцен использованы исключительно статические меши. Статические меши, несмотря на название, не являются неспособными к перемещению в приложениях. «Статические» части статического меша — это его вершины. Статические меши не могут деформироваться, как меши, построенные на основе костей. Вот для чего нужна скелетная сетка.

Архитектура и реквизитные меши

Я думаю о своих ассетах статических мешей в двух широких категориях: архитектура и реквизит. Каждый из них следует схожим правилам, но относительно обоих есть соображения, которые могут значительно облегчить работу с каждым из них.

Архитектура

Архитектурные меши — это уникальные объекты, расположенные в определенном месте, например стены и полы, рельеф местности, дороги и т. д. Как правило, у вас на сцене находится только одна копия этих объектов, и они должны располагаться в определенном месте.

Чаще всего вы сохраняете соотношение 1:1 между объектами в исходном изображении, папкой Content и уровнем UE4. Например, у вас будет один SM_Wall01 в каждом из них в одном и том же месте.

Вы экспортируете эти объекты на место, импортируете их и размещаете на сцене в точке 0,0,0 (или в другой определенной исходной точке). Синхронизировать содержимое UE4 со сценой с помощью этого метода очень просто, поскольку вы можете импортировать или повторно импортировать меши или целые сцены без необходимости их перемещения.

Реквизит

Реквизитные меши — это повторяющиеся или многократно используемые статические меши, которые вы размещаете в архитектуре или на уровне. Примером могут служить тарелки на столе. У вас может быть уникальный ассет статического меша для каждого объекта сцены или, поскольку все они абсолютно одинаковы, вы можете ссылаться на один ассет и перемещать его на нужную позицию на Viewport. Стол также был бы реквизитом, и его можно было бы легко передвигать.

Наличие нескольких реквизитов, ссылающихся на один ассет, снижает нагрузку на память и упрощает обновление и повторение контента.

Именование

Как и во всех цифровых проектах, разработка и соблюдение твердых соглашений об именовании в проектах является важной задачей для всех, кто работает с UE4. Проекты заканчиваются с тысячами отдельных ассетов, и все они имеют расширение **.uasset**. Присвоение имен вашим объектам в 3D-приложении происходит легко и быстро, но в UE4 это не совсем так.

Вы, несомненно, придумаете собственные стандарты и систему для обеспечения ваших конвейеров для данных, но единообразие поможет вам быть на одной волне с остальными участниками сообщества Unreal Engine 4, поскольку вы, вероятно, станете использовать контент и делиться им.

Основы

Не используйте пробелы или специальные символы/символы Юникода (например, национальные алфавиты) при именовании чего-либо (меши, файлы, переменные и т. д.). Иногда вам может сойти с рук нарушение этих правил, но это усложняет ситуацию позже, когда вы сталкиваетесь с системой, которая не принимает пробел или специальный символ.

Соглашение об именовании UE4

Базовая схема именования для контента в Unreal использовалась в течение последних 15 лет Epic Games и разработчиками, использующими Unreal Engine. Оно следует за основной Конвенции:

Prefix_AssetName_Suffix

Если вы просматривали любое содержимое, включенное в UE4, вы уже видели именования, отвечающие этой конвенции.

> **Заметка**
>
> Движок и Editor **не применяют** никаких правил к соглашениям об именовании. Вы можете называть контент так, как вам нравится.

Prefix

Prefix (префикс) — это короткий одно- или двухбуквенный код, определяющий тип содержимого ресурса. Распространенными примерами являются M_ для материалов и SM_ для статических мешей. Обычно это простое сокращение или аббревиатура, но иногда конфликты или традиции требуют других, таких как SK_ для скелетных мешей. Полный список доступен на сопутствующем сайте по адресу www.TomShannon3D.com/unreal4Viz.

AssetName

BaseName описывает объект простым и понятным способом. WoodFloor, Stone, Concrete, Asphalt и Leather — все это прекрасные примеры.

Suffix

Отдельные классы ассетов имеют некоторые вариации, которые важно отметить в названии. Самый распространенный пример — текстуры. Хотя у всех у них один и тот же префикс T_, различные типы текстур предназначены для конкретных применений в движке, таких как карты нормалей и карты Roughness (шероховатостей). Использование суффикса помогает описать эти различия, сохраняя при этом все объекты текстуры аккуратно сгруппированными.

Примеры

Примеры, приведенные в таблице 3.1, можно найти в любом распространенном проекте UE4.

Таблица 3.1 Примеры названий ассетов

Название ассета	Заметка
T_Flooring_OakBig_D	Текстура с базовым цветом дубового паркета с крупным рисунком
T_Flooring_OakBig_N	Карта нормалей дубового паркета с крупным рисунком
ML_Flooring_OakBig	Экземпляр материала с использованием текстур дубового паркета с крупным рисунком
M_Flooring_MasterMaterial	Родительский материал, к которому относится экземпляр материала дубового паркета с крупным рисунком
SM_Floor_1stFloor	Статический меш с примененным экземпляром материала MI_Flooring_OakBig

В UE4 есть десятки типов контента и вариантов, каждый из которых имеет различную возможную схему именования. Вы можете найти на www.TomShannon3D.com/UnrealForViz прямую ссылку на управляемый сообществом список всех предложенных префиксов и суффиксов.

Хотя этот список является исчерпывающим, ни одно из этих правил именования не применяется в произвольном случае. Главное — это последовательность. Разработайте систему и придерживайтесь ее.

UV-отображение

UV-отображение — это вызов для всех 3D-художников и тем более для визуализации. Большая часть визуализации рендеринга может обойтись очень плохим отображением UVW, а иногда и вовсе без него. UE4 без UV не может.

Координаты UVW используются для множества функций и эффектов; от очевидных, таких как применение текстур на поверхности, до менее понятных, но не менее важных вещей, как, например, координаты Lightmap.

Всем вашим 3D-ассетам потребуются качественные, последовательные координаты UVW. Для простой геометрии это может быть так же просто, как применение модификатора UVW-карты к вашей геометрии; более сложная геометрия превратится в более сложный ручной процесс. Не бойтесь; в большинстве случаев вы легко получите хорошие результаты. Просто следите за своими UV-координатами, когда работаете с UE4. Если у вас есть проблемы с визуализацией, которые трудно диагностировать, проверьте UV-координаты.

Реальный масштаб

Многие 3D-приложения могут использовать в реальном масштабе систему UV-координат, где текстура масштабируется в материале, а не в UV-координатах. Хотя такое вполне возможно сделать в UE4, это не лучший вариант.

Если ваши сцены уже сопоставлены с реальным масштабом, вы должны масштабировать UV-координаты с помощью модификатора (Scale UVW в Max) или вручную масштабировать UV-вертикали в UV-Editor.

Текстурирование с помощью мировой проекции

Знаменитый чит визуализатора состоит в текстурировании в мировых координатах (текстуры проецируются вдоль мировых координатных осей XYZ) для быстрого покрытия сложных статических моделей тайловыми текстурами. Такие методы не требуют качественных UV-координат для получения приличного результата. UE4 предоставляет методы для такой работы, и лично я на них полагаюсь в собственных проектах.

Важно помнить, что если вы планируете использовать предварительно рассчитанное освещение с Lightmass, то все-таки потребуется предусмотреть хорошие, явные UV-координаты, даже если применяется текстурирование с помощью мировой проекции. Это происходит потому, что Lightmass хранит информацию об освещении в текстурных картах, которым нужно уникальное пространство UC для правильного отображения.

Тайлинг против уникальных текстурных координат

Для большинства визуализаций активно используется тайлинг. В действительности, конечно, текстура не тайлится, скорее UV-координаты устанавливаются или изменяются в зависимости от материала. Тайловые наборы UVW-координат позволяют

UV-граням перекрываться, а вершинам выходить за пределы 0–1 UV-пространства. Такова природа UV-координат, что выход за пределы 0–1 приводит к повторению текстуры (тайлингу), то есть 0.2, 1.2 и 2.2 — это один и тот же пиксель.

Уникальные UV-координаты определяются отсутствием координат вне диапазона 0–1, и каждая грань занимает уникальную область в пространстве UV-карты без перекрытий. Это полезно, когда вы запекаете информацию в текстурную карту, поскольку каждый пиксель может соответствовать определенному местоположению на поверхности модели (рисунок 3.3).

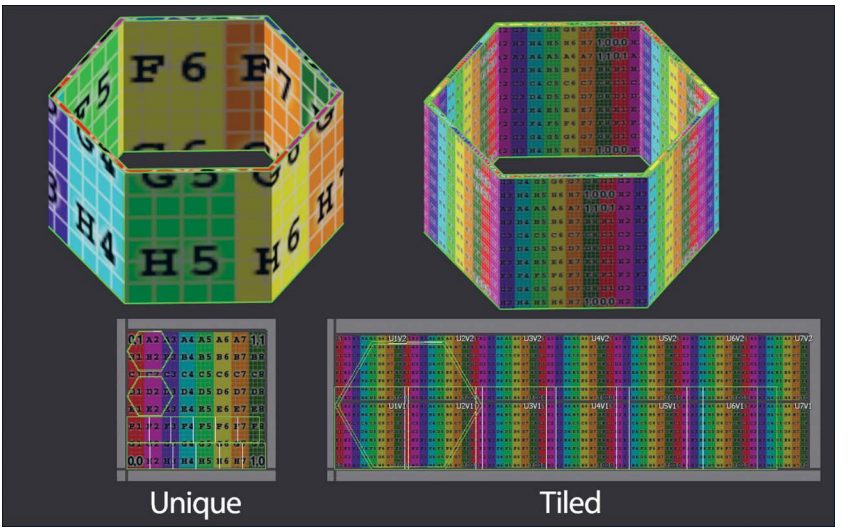


Рисунок 3.3 Уникальные координаты по сравнению с плиточными UV-координатами

Распространенными примерами преобразования данных в текстуры являются карты нормалей, созданные из высокополигональных моделей с использованием репроекции или, в UE4, Lightmass для записи информации об освещении и Global Illumination (GI) в текстурные карты, называемые **Lightmaps**.

Множественные UV-каналы

Unreal Engine 4 поддерживает и даже поощряет использование нескольких UV-каналов для смешивания текстур и других функций на уровне движка, таких как запеченное освещение с помощью Lightmass.

В 3D-приложениях часто возникают проблемы с визуализацией нескольких UV-каналов в Viewport, а рабочий процесс обычно неуклюж и откладывается в пользу использования более крупных и детализированных текстур. Это не вариант; массивные текстуры могут потреблять огромное количество VRAM и неприемлемо перегружать сцены.

Вместо этого используйте несколько UV-каналов и многослойные UV-координаты в материалах для добавления деталей крупным объектам.

Заметка

UE4 использует основание 0 для UV-координат. Многие приложения используют 1. Другими словами, если ваше приложение (любители Max, я смотрю на вас!) использует основание 1 UV-канала, UE4 импортирует его как координатный индекс 0.

Lightmap-координаты

Создание UV-координат Lightmap может показаться обременительной задачей. Не бойтесь: координаты Lightmap легко создаются и являются просто вторым набором UV-координат, которые обеспечивают соблюдение нескольких правил.

- Если в вашем проекте используется исключительно динамическое освещение, вам, скорее всего, вообще не нужно беспокоиться о координатах Lightmap!
- Координаты должны быть уникальными, а не перекрываться или накладываться друг на друга. Если грани перекрываются, Lightmass не может решить, информацию об освещении какой грани записать в пиксель, порождая ужасно выглядящие ляпы.
- Между UV-диаграммами (группами прикрепленных граней в UV-Editor) должно существовать пространство. Это называется **padding** (заполнение) и гарантирует, что пиксели из одного треугольника не перетекают в соседние треугольники.
- Последнее основное правило состоит из двух частей: избегайте разбиения UV-карт Lightmap на гладких гранях, поскольку это может привести к некрасивому шву и разбивайте координаты вдоль границ Smoothing Group, чтобы избежать перетекания света через них.

Заметка

Иногда у вас не будет другого выбора, кроме как разместить UV-шов на гладкой поверхности. Сделайте все возможное, чтобы создать его «лицом» в другую сторону от того места, где находится или с наибольшей вероятностью будет находиться зритель (обычно выбирается задняя или нижняя часть объекта).

Автоматическое создание Lighmap UVs

Опция Auto Generate Lightmap UVs при импорте позволяет получить существенный выигрыш по времени при подготовке моделей к освещению с помощью Lightmass.

Я не устану его рекомендовать. Он быстро генерирует высококачественные координаты Lightmap при импорте (или после) и настраивается художником для получения оптимальных результатов (рисунок 3.4).

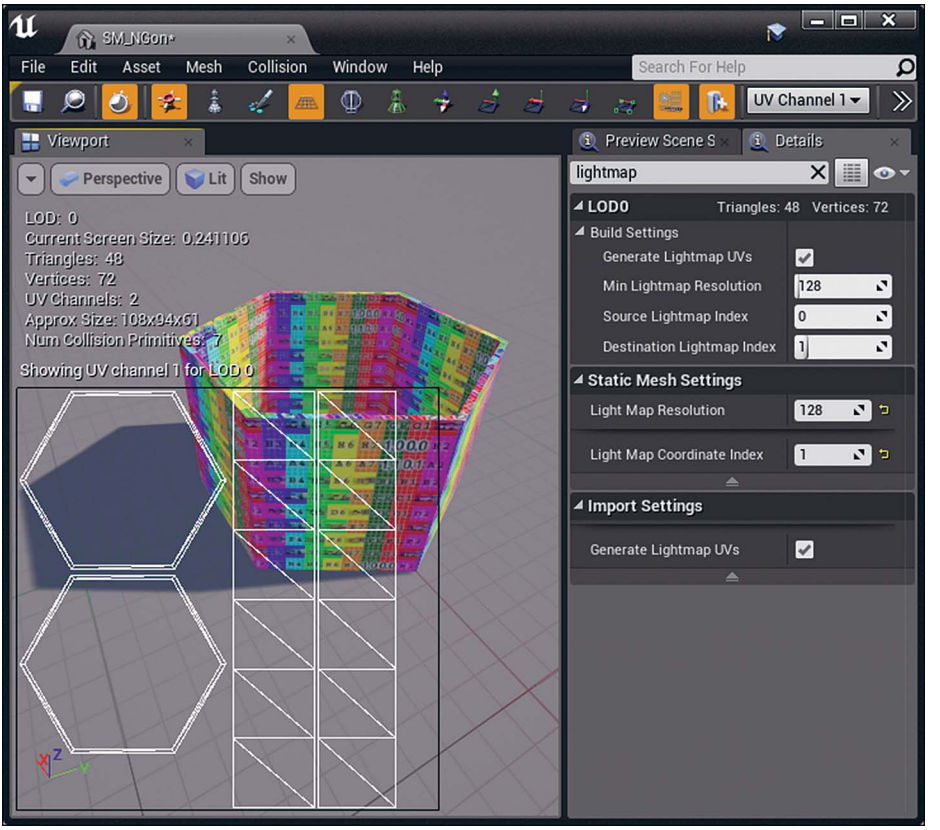


Рисунок 3.4 Автогенерированные UV-координаты с наложенными UV-координатами Lightmap (обратите внимание на эффективность по сравнению с рисунком 3.3)

В широком смысле система Auto Generate Lightmap UVs — это просто операция переупаковки и нормализации, которая берет исходный канал и упаковывает существующие UV-диаграммы в UV-пространство 0–1 так эффективно, как только возможно. Он также добавляет правильное количество отступов пикселей между диаграммами на основе предполагаемого разрешения Lightmap, заданного Source Lightmap Index. У Lightmap с более низким разрешением пиксели крупнее, и они требуют большего пространства между диаграммами. При более высоком разрешении пиксели мельче и требуют меньшего пространства.

Для этого нужно только, чтобы координаты в вашем базовом UV-канале следовали правилам разделения по Smoothing Groups и не были растянуты. Они могут быть любого масштаба, перекрываться и многое другое, если они изначально четко сопоставлены.

Совет

Наложение карт освещения на геометрию часто можно осуществлять на скорую руку тривиальной автоматической разверткой (box mapping) или инструментами UV-развертки в вашем приложении перед экспортом (например, с разбиением по группам сглаживания (smoothing group) или углам между гранями). В сочетании с 3D-мэппингом в мировых координатах рабочий процесс создания UV-разверток может быть эффективным и автоматизироваться без особых усилий.

Уровень детализации (LOD)

UE4 имеет фантастическую поддержку **LOD**. LOD — это процесс переключения 3D-моделей на более низкую детализацию по мере их удаления на расстояние и уменьшения на экране.

Если вы разрабатываете контент для мобильных устройств или VR, оптимизация LOD и модели невероятно важны. Даже если вы работаете на высококлассном оборудовании, подумайте о создании моделей LOD для высокополигональных реквизитов. Для таких мешей, как транспортные средства, люди, деревья и другая растительность, которые используются сотни раз каждый в большой сцене, наличие эффективных моделей LOD будет означать, что вы сможете сохранить больше ресурсов на экране, создавая детальную симуляцию.

UE4 полностью поддерживает цепочки LOD из приложений, поддерживающих их экспорт в FBX. Это означает возможность создавать свои LOD в любимом 3D-пакете и импортировать их все сразу в UE4. Вы также можете вручную импортировать LODs, экспортировав несколько файлов FBX и импортировав их из Editor.

Автоматическое создание LOD

UE4 поддерживает создание LOD непосредственно в Editor. Как и генерация Lightmap, эти LOD специально созданы для UE4 и великолепны без почти какого-либо вмешательства художника.

Чтобы создать LOD-меши в Editor, просто назначьте LOD Group в Editor статического меша или LOD Group при импорте. Я рекомендую это для мешей с более высокой плотностью, особенно если вы ориентируетесь на мобильные или VR-платформы.

Несколько сторонних плагинов, таких как Simplygon (https://www.simplygon.com) и InstaLOD (http://www.instalod.io/), может дополнительно автоматизировать весь процесс генерации LOD и иметь более мощные возможности уменьшения, чем встроенное решение UE4.

Если вы регулярно имеете дело с сильно тесселированными мешами, я рекомендую рассмотреть возможность интеграции этих плагинов в ваш конвейер UE4, потому что

они гораздо надежнее систем оптимизации в большинстве 3D-приложений, и время, сэкономленное с помощью автоматической генерации LOD, может оказаться значительным, не говоря уже об очевидных преимуществах производительности за счет уменьшения общего числа вершин, преобразуемых и визуализируемых каждым кадром.

Обнаружение столкновений

Обнаружение столкновений (для уточнения, факт пересечения оболочек мешей будем называть коллизией) — это сложный предмет в UE4. Интерактивные акторы сталкиваются и моделируют физику, используя комбинацию для каждого полигона и прокси геометрии (для скорости) с низким разрешением и различные настройки, чтобы определить, с чем они сталкиваются и через что могут пройти (например, ракета блокируется щитом, через который может пройти ваш персонаж).

К счастью, поскольку большая часть визуализации не требует сложных физических взаимодействий между враждующими фракциями, вы можете использовать упрощенный подход к подготовке столкновения.

Коллизия архитектурных мешей

Для крупной уникальной архитектуры вы можете просто использовать per-polygon (сложное) столкновение. Стены, полы, дороги, ландшафты, тротуары и так далее могут использовать это легко без существенной просадки в производительности. Имейте в виду, что чем плотнее меш, тем дороже эта операция, поэтому последуйте дружескому совету и разбейте более крупные меши, чтобы избежать проблем с производительностью.

Коллизия реквизитных мешей

Для небольших или более детализированных мешей сложное столкновение слишком затратно по памяти и производительности. Вместо этого следует использовать Collision Proxy Mesh с низким разрешением или Simple Collision (простое столкновение).

Вы можете выбрать создание собственных низкополигональных прокси **Collision Meshes**[4] в 3D-пакете или в Editor UE4, используя вручную размещенные примитивы (коробки, сферы, капсулы). Кроме того, следует взглянуть на параметры автоматической генерации столкновений, которые предоставляет Editor.

Convex Decomposition (Auto Convex Collision)

UE4 предоставляет в Editor отличную автоматическую систему генерации столкновений под названием **Convex Decomposition**. Эта функция использует причудливую систему вокселизации, которая разбивает полигональные объекты на 3D-сетки для создания качественных примитивов столкновения. Эта система работает и для сложных

[4] Низкополигональные примитивы экспортируются вместе с вашими мешами, которые следуют определенному соглашению об именовании. Дополнительные сведения см. в разделе документации Static Mesh по ссылке https://docs.unrealengine.com/latest/ INT/Engine/Content/FBX/StaticMeshes/index.html#collision.

мешей, и я очень рекомендую использовать ее для уменьшения количества полигонов на ваших реквизитах.

Не путайте Convex Decomposition с возможностью автоматического создания коллизии во время процесса импорта. Эта система унаследована от UE3 и непригодна для многих ситуаций визуализации, потому что она плохо обрабатывает большие, неравномерные или удлиненные формы.

Convex Decomposition не включена при импорте, поскольку она может быть очень медленной для больших мешей или мешей с большим количеством треугольников. Для этого вы должны создать традиционную оболочку столкновения в своем 3D-приложении и импортировать ее вместе с моделью.

Дополнительную информацию о создании коллизий можно найти в официальной документации и по адресу www.TomShannon3d.com/UnrealForViz.

Pivot Point (поворотная точка)

UE4 импортирует поворотные точки (пивоты) не так, как это можно было бы ожидать. Движок позволяет выбрать один из двух вариантов для **пивотов** при импорте через параметр Transform Vertex в Absolute в импорте статического меша. Вы можете либо иметь набор пивотов, полученный из 3D-приложения (путем установки Transform Vertex to Absolute на false), либо можно указать pivot point быть импортированным на сцены в точку 0,0,0, полностью игнорируя определенный pivot point (путем установки Transform Vertex to Absolute на true).

Вероятно, у вас возник вопрос, зачем может потребоваться переопределение pivot point вашего объекта? В действительности это может обеспечить существенную экономию времени. Используя общий пивот, вы можете легко и надежно разместить все **архитектурные меши** на вашей сцене, поместив их на уровень и установив положение 0,0,0; все они будут точно выровнены.

И это работает, потому что вы обычно не перемещаете свои архитектурные меши, поэтому то, где они вращаются и масштабируются, не имеет значения. Что важно, так это их точное положение в трехмерном пространстве.

Реквизитные меши — это противоположность. Когда вы размещаете реквизит на своих сценах, пивот имеет важное значение для перемещения, поворота и масштабирования их на месте.

Чтобы получить pivot point в UE4 в соответствии с вашим 3D-приложением, у вас есть несколько вариантов. Вы можете смоделировать все ваши реквизиты на уровне в 0,0,0 или переместить их при экспорте. Более новые версии UE4 (4.13 и выше) позволяют переопределить поведение импорта по умолчанию и использовать созданный pivot point меша. Как правило, это лучший вариант для обеспечения соответствия ваших pivot points.

Конвейер работы с FBX-мешами

После подготовки ассетов вам нужно экспортировать их в FBX. У этого формата очень долгая история, которая привела к тому, что он стал одним из наиболее распространенных форматов, доступных для 3D-данных.

Если ваша сцена отмасштабирована не в сантиметрах, вы можете легко перемасштабировать ваши ассеты при экспорте, установив **Scene units to converted to:** *Centimeters*.

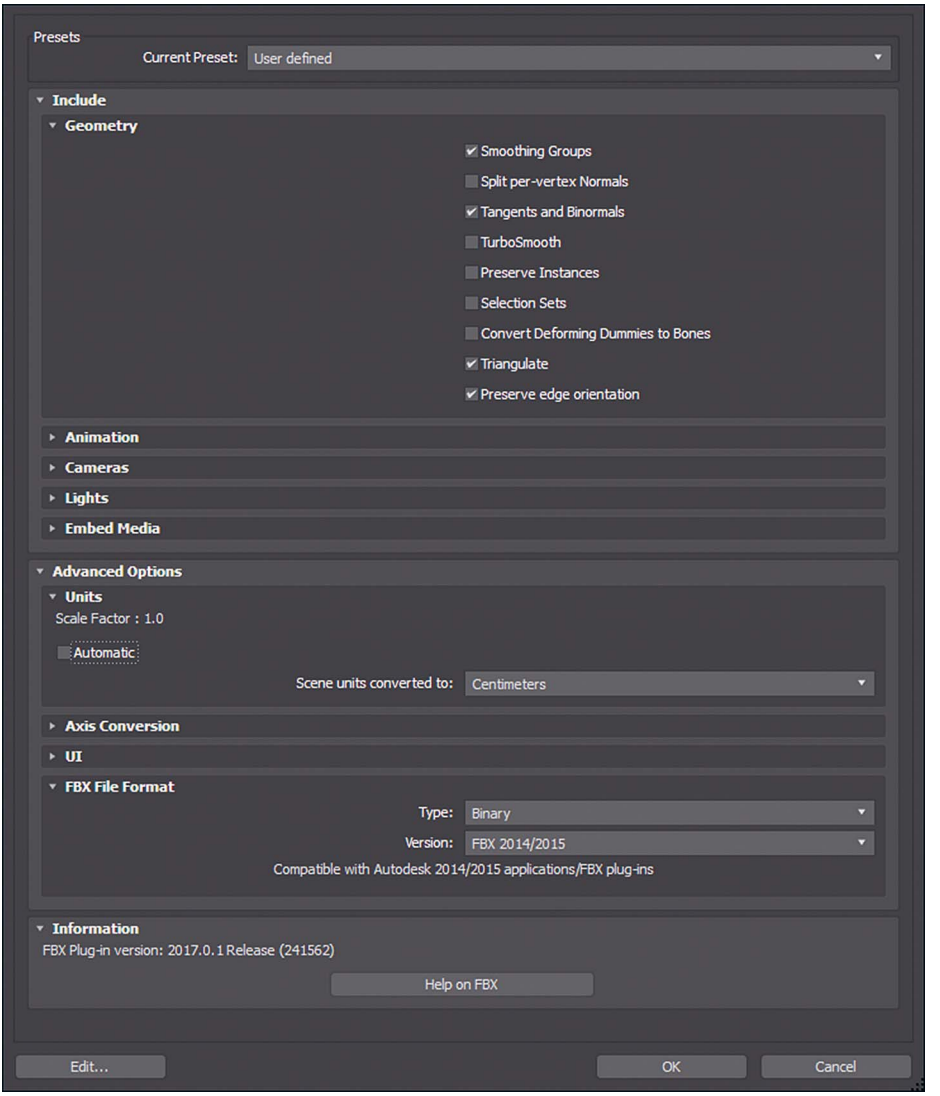


Рисунок 3.5 Рекомендуемые настройки экспорта FBX для статических мешей (3D Studio Max)

Экспорт нескольких мешей

Вы можете использовать разные способы экспорта нескольких мешей. У каждого есть свои недостатки и преимущества.

Один FBX-файл

Несколько мешей могут быть включены в один файл FBX. Когда вы импортируете его в UE4, будет предложена опция объединения мешей в один или импорт каждого из них как отдельных мешей.

Я не фанат этого метода. Некоторые проблемы могут сделать этот способ неудобным и иногда нестабильным.

Частая проблема — это что вам нужно экспортировать точно такой же набор мешей каждый раз, когда вы хотите обновить только один. Это правило очень трудно соблюдать, а его невыполнение может привести к сбоям и другим проблемам.

Импорт сцены

В UE4 был добавлен способ для облегчения вышеупомянутой проблемы. **Scene Import** принимает единственный FBX-файл, импортирует все меши и располагает все на Level. Этим способом также можно импортировать камеры, освещение и анимации.

Множество FBX-файлов

Экспорт множества файлов FBX — моя любимая опция. Хотя большинство 3D-приложений не поддерживают нативно этот способ, однако доступно несколько сценариев и инструментов, которые позволяют выполнить экспорт множества. Вы можете найти часть этих скриптов на www.TomShannon3D.com/UnrealForViz.

UE4 может быстро импортировать полную структуру файлов FBX. Таким путем, если вам нужно обновить ваш единственный ассет или группу ассетов, вы можете легко и надежно заменить эти FBX-файлы и переимпортировать в UE4. Этот подход дает вам прекрасную возможность контроля, и это наименее рискованный и наименее технически сложный метод.

Переимпортирование

Легкий итеративный процесс — это когда очень легко можно перезаписать ваш уже экспортированный FBX с обновлением геометрии. Это позволяет выбрать файл в Content Browser и переимпортировать его, обновив ассеты в UE4.

Автоматический реимпорт

Я часто использую автоматический реимпорт в UE4. Он позволяет выбрать директорию, и каждый раз, когда вы обновляете или создаете новый файл, который можно импортировать, Editor находит эти изменения и автоматически обновляет существующие ассеты или импортирует и создает новые.

Чтобы включить этот способ, откройте Editor Preferences и определите папку для просмотра и соответствующую папку в Content (рисунок 3.6).

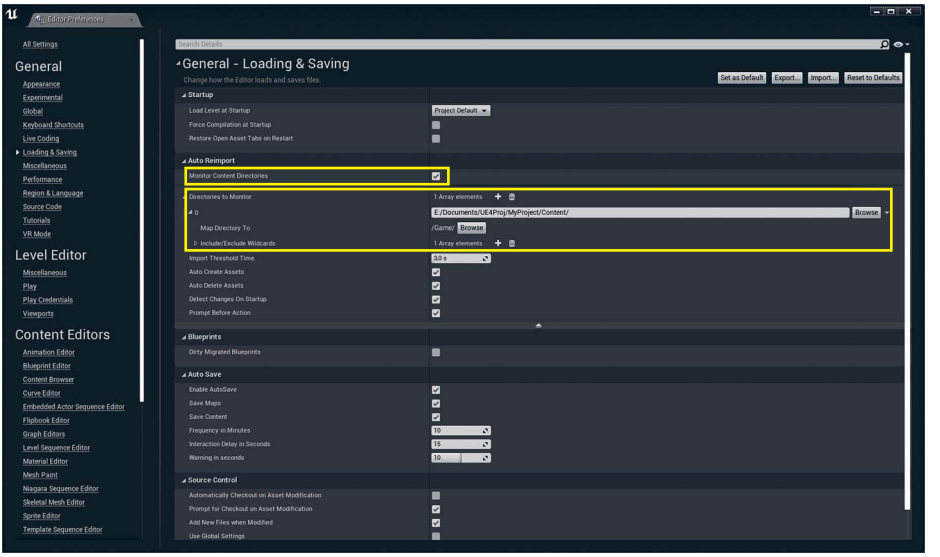


Рисунок 3.6 Настройка автоматического импорта (Auto Import) в Editor Preferences

Заметка

По умолчанию автоматический импорт включен для папки Content. Хотя это легко запомнить и использовать, я не рекомендую помещать ваши исходные файлы в эту папку. Никогда не меняйте папку Content вне Editor — UE4 это не нравится и может привести к повреждению проекта.

Куда помещать FBX-файлы

Обычно вам не нужно поддерживать экспортированные FBX-файлы, и можно считать их временными. Вы можете сохранять их где угодно. Чтобы использовать функцию переимпортирования, очень важно быть последовательным.

Процесс работы с текстурами и материалами

Как и всякий раз, когда вы переходите от одного программного обеспечения для рендеринга к другому, вам придется столкнуться с необходимостью повторного создания ваших материалов, чтобы воспользоваться преимуществами новых возможностей рендера, и UE4 в этом плане ничем не отличается от другого ПО.

UE4 позволяет вам импортировать материалы с мешами, но функции для импорта очень ограниченны и требуют, чтобы ваши исходные материалы были сделаны определенным образом.

Это не значит, что импортировать материалы в UE4 не стоит — это может сохранить много времени для прототипирования и создания материала в UE4. Не возитесь с материалами в 3D-приложении. Простой диффузной карты или цвета будет достаточно (и это все, что придет в голову!).

Текстуры

UE4 накладывает некоторые строгие правила и ограничения на текстуры. Некоторые правила применяются настолько строго, что являются причинами провала импорта текстур, созданных во внешних приложениях; другие просто ухудшают качество или производительность вашего проекта.

Большинство правил — это технические ограничения, наложенные на UE4 аппаратным или программным обеспечением, на котором запускается UE4. Graphical Processing Units (GPU) на графических картах специфичны тем, как текстуры хранятся, как к ним получать доступ и как они рендерятся, и UE4 должен учитывать все эти ограничения.

Поддерживаемые форматы

UE4 поддерживает следующие форматы картинок:

- **.bmp**
- **.float**
- **.pcx**
- **.png**
- **.psd**
- **.tga**
- **.jpg**
- **.exr**
- **.dds**
- **.hdr**

Наиболее общие в использовании — BMP, TGA и PNG.

TGA файлы — это стандарт 8-битных картинок в игровой индустрии. Популярность TGA связана с возможностями явного контроля художниками изображений по каждому из каналов. UE4 принимает 24-битные (RGB) и 32-битный (RGBA) TGA-файлы[5].

[5] Форматы картинок могут быть непонятными. 32-битные TGA-файлы содержат четыре 8-битных канала (RGBA), в отличие от 32-битных картинок формата HDR или EXR, который представляет 32 бита на канал.

PNG-файлы наиболее распространены для UI-элементов, потому что они используют предварительно умноженный альфа-канал, который позволяет правильно блендить их.

Я рекомендую использовать TGA и BMP для всех текстур, которые будут использоваться в материалах.

MIP-текстурирование

Mip-карты — версия текстур с уменьшенным разрешением, которые рассчитываются заранее и хранятся как часть текстуры. Это позволяет GPU использовать маленькие текстуры тогда, когда объект, на который они наложены, отдаляется от камеры. Это позволяет избежать сглаживания и повысить производительность.

Разрешение

Все текстуры должны быть масштабированы до разрешения степени двойки, такие как 64, 128, 256, 512, 1024 и т. д. С максимальным разрешением 8192 на 8192, высокое разрешение текстуры требует модификации исходных кодов и может не поддерживаться на всех видеокартах.

Текстуры могут быть не квадратными, но все еще должны оставаться степенью двойки по каждой стороне; например 128 на 1024 будет нормально.

Мы можете импортировать текстуры не степени двойки, но они не будут производить Min-Maps, что добавляет различные визуальные артефакты в проект. Вы должны всегда тратить время на масштабирование текстур до нужных размеров перед импортом.

Alpha-каналы (альфа-каналы)

Вы можете включать альфа-каналы во многие форматы; однако имейте в виду, что альфа-каналы вдвойне накладывают влияние на размер текстур, поэтому включайте его, только если он используется.

Другой вариант — это включить альфа-канал в качестве отдельной текстуры. Это поможет, когда вам нужно поделиться альфа-каналом между разными текстурами или вы хотите иметь уменьшенное разрешение альфа-канала по сравнению с остальными каналами.

Сжатие

UE4 использует сжатие текстур на аппаратном уровне. В большинстве случаев это формат DDS с некоторыми значительными артефактами сжатия блоков, поэтому очень важно иметь качественное исходное изображение (то есть избегать использования сжатых исходных изображений).

Вы можете переопределить сжатие на основе каждой текстуры, но вы должны это делать только при необходимости. Несжатые текстуры могут быть до 8 раз больше памяти.

Несколько материалов

UE4 полностью поддерживает множество материалов на одном меше. В вашем 3D-приложении назначьте материалы на каждую поверхность и используйте обычный процесс работы с FBX. Только одно предостережение: каждый материал увеличивает расходы на рендеринг на каждого актора. Для акторов, которые используются нечасто, как архитектура или здания, это хорошо. Ассетам, которые используются на сценах (транспортные средства, растительность, реквизит и т. д.), лучше иметь как можно меньше материалов, потому что невыполнение этого быстро приведет к появлению проблем с производительностью в вашем проекте.

Импорт в Content Library

Импортирование контента — это наименее трудоемкий с технической стороны процесс, в отличие от подготовки и экспорта вашего ассета. Обычно вас интересует только несколько опций, если вы работаете с визуализацией.

Инициирование импорта

У вас несколько простых путей для инициирования импорта.

1. Перетаскиванием ассета из вашего проводника в Content Browser. Вы можете перетаскивать сразу несколько файлов, несколько директорий прямо в Content Browser.

2. Кликом правой кнопкой мышки в Content Browser и выбрать Import.

3. Использованием кнопки **Import** в Content Browser.

4. Автоматически импортировать ассет путем указания директории для отслеживания и каталог назначения в вашем проекте.

В зависимости от того, что вы импортируете, UE4 предоставит вам разные наборы опций импорта.

Опции импорта меша

Опции, которые вы выберете, будут немного отличаться в зависимости от конкретных нужд, но в большинстве случаев вам подойдут следующие (рисунок 3.7).

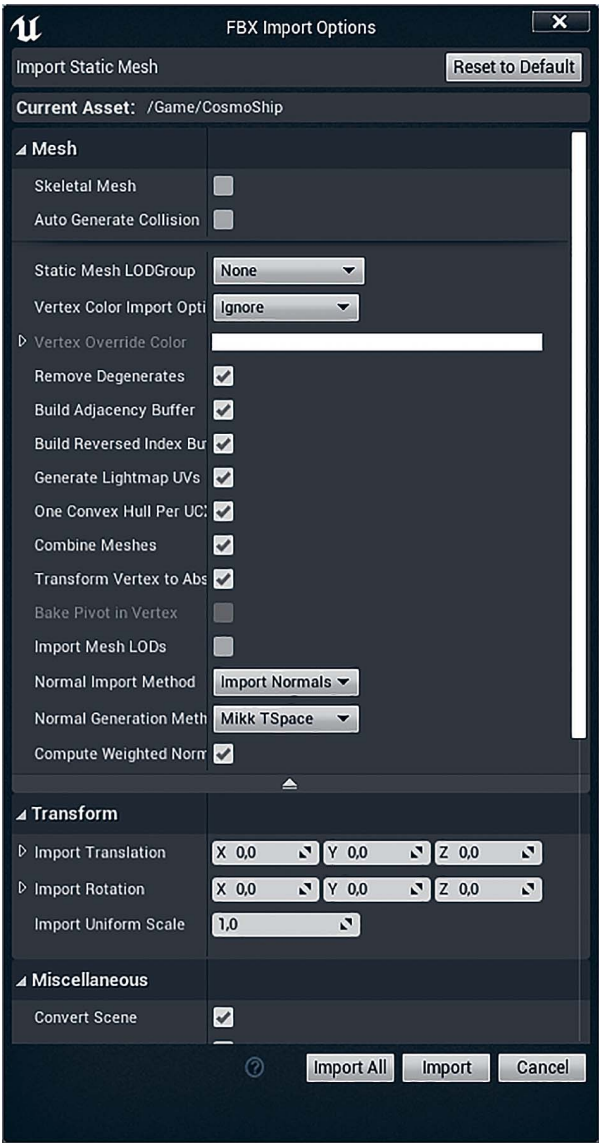


Рисунок 3.7 Предлагаемые настройки FBX-импорта для статических мешей

Auto Generate Collision

Я не рекомендовал бы использовать **Auto Generate Collision** для большинства мешей. Это устаревшая система, которая выдает плохой результат для контента, не созданного специально для нее.

Вы должны создавать собственное столкновение (collision) (в вашем 3D-пакете или в Editor, разместив примитивы), полагаясь на межполигональное столкновение

(медленное на высокополигональных сетках) или используя Convex Decomposition Collision), создаваемое в Editor.

Generate Lightmap UV

Я рекомендую использовать **Generate Lightmap UVs** и отказаться от создания собственного Lightmap канала. Лучше сосредоточиться на очистке базового UV-канала и позволить Editor поработать над вашим контентом.

Import Materials and Textures

Обычно я оставляю включенным Import Materials and Textures, потому что очень удобно иметь сразу назначенный материал и получить различные текстуры, которые были применены до импорта, а также это дает хорошую отправную точку для создания и назначения материалов позже.

Однако, если вы будете неосторожны, эта опция может создать огромный беспорядок. Чтобы избежать этого, нужно обеспечить правильное расположение ваших материалов в используемом 3D-приложении перед экспортом. Также вы можете использовать Material Instance assignment system для создания экземпляра материала, а не самого материала.

Transform Vertex to Absolute

Когда опция **Transform Vertex to Absolute** *включена*, UE4 заменит созданный pivot point на 0,0,0 координаты на сцене. Если опция *выключена*, Mesh будет использовать созданный pivot point. Как упоминалось в предыдущем разделе «Подготовка геометрии к UE4», этот параметр может сэкономить много времени при размещении и обслуживании ваших архитектурных мешей.

Опции текстур

Когда вы импортируете текстуры, вам не предоставляются параметры импорта; однако вы должны убедиться, что на текстурах правильно установлены флаги. Это может значительно сказаться на производительности и визуальной точности ваших проектов.

После импорта откройте Texture Editor с помощью двойного нажатия на ассет текстуры (Texture asset) в Content Browser. Следующие настройки имеют наибольшую важность.

- **Texture Group:** большинство ваших текстур могут быть оставлены в World Group. Карта нормалей, HDR- и UI-изображения и другие специальные текстуры, такие как LUT и векторные карты, должны быть расположены в соответствующих группах. Это устанавливает множество внутренних флагов, чтобы гарантировать корректное чтение и отображение текстур.

- **Compression Settings:** установка Texture Group часто будет корректно устанавливать Compression Settings; часто — нет. Большинство текстур использует

настройки по умолчанию (Default settings). Карты нормалей всегда должны использовать настройки карт нормалей. Для UI-текстур следует установить User Interface setting, чтобы обеспечить правильное альфа-смешивание (alpha blending) и масштабирование.

- **sRGB:** sRGB флаг должен быть снят почти для всех текстур, которые содержат информацию о цвете. Это сообщает механизму рендеринга, чтобы гамма-корректировать эту текстуру для точного отображения на сцене (UE4 использует конвейер линейного рендеринга и требует, чтобы текстуры были гамма-скорректированы для линейного цветового пространства). Для текстур, используемых как маски или другие карты информации, такие как карта нормалей, этот параметр должен быть выключен для обеспечения точного считывания данных без применения к ним гамма-кривой.

Процесс работы с камерой

Вы наверняка эксперт в визуализации анимации с быстрой, плавной анимацией камеры, подсвечивающей каждый аспект проекта. Вы потратили годы, оттачивая свое мастерство, и разработали инструменты и техники, которые выделяют вас среди конкурентов.

UE4 предлагает удивительную систему кинематографии (cinematics) и анимации с Sequencer, но изучение полностью новой системы может быть излишним и напугать. UE4 предлагает мощный набор инструментов для импорта и экспорта камер как в, так и из вашего приложения, а также редактор, позволяющий легко выполнять итерации, используя похожие инструменты.

Просто выберите ваш объект Camera в 3D-приложении, которым пользуетесь, и экспортируйте его как FBX. Возможно, вам понадобится запечь анимацию, если ваша камера подключена к сплайну (spline), или использует другие нестандартные преобразования (например, присоединение к другим объектам, использование Look-at-контроллеров или модификаторов, или является частью рига.

В UE4 создайте новый Level Sequence, нажав на кнопку Cinematic на toolbar в Editor, и выберите Add Level Sequence, а затем выберите местоположение для сохранения нового ассета (рисунок 3.8).

После того как новый Sequencer откроется, щелкните правой кнопкой мышки на динамически созданном акторе Camera и выберите Import.

Добавится ваша анимация и применится к камере. Я столкнулся с некоторыми несоответствиями с типами камер, отличными от значений по умолчанию в приложении, такими как пивоты или неправильное позиционирование. Возможно, вам придется

переработать свои камеры или преобразовать их в стандартные типы, чтобы полностью использовать эту функцию.

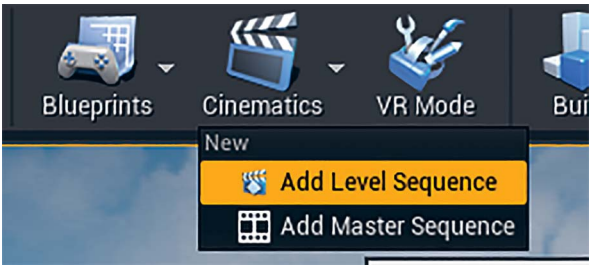


Рисунок 3.8 Добавление нового Level Sequence

Вы также можете экспортировать анимации камеры из Sequencer обратно в FBX-файл и импортировать их в 3D-приложение или даже в пакет постредактирования, такой как Nuke, чтобы завершить снимок, отрендеренный в UE4.

Заключение

Подготовка вашего контента к экспорту и импорту в UE4 — это самое большое препятствие, с которым многие столкнутся. Однако затраты времени на подготовку вашего контента для UE4 окупятся с лихвой, поскольку вы восстановите свои данные в UE4 Editor. Даже большие наборы данных могут быть аккуратно перенесены в UE4 и на ваши сцены, если вы уверены в успехе и инструментах, позволяющих его достичь.

ГЛАВА 4

ОСВЕЩЕНИЕ И РЕНДЕРИНГ

Одна из самых привлекательных особенностей UE4 — потрясающее качество визуализаций, которого можно достичь. Используя физически корректный рендеринг (PBR), UE4 достигает ошеломляющего реализма. Поверхности реагируют на свет и тень удивительно реалистично. Material Editor использует визуальную графическую систему, которая позволяет создавать сложные, динамичные, физически корректные материалы, в то время как экземпляры материалов (Material Instance) и функции позволяют повторно использовать ваши материалы и менять их в режиме реального времени. Возможно, вам больше никогда не захочется возвращаться к своему старому рендереру.

Понимание физически корректного рендеринга UE4 (PBR)

В последнее десятилетие компьютерная графика пережила Ренессанс качества и точности. Это в значительной степени связано с развитием физически корректного рендеринга (PBR). PBR заменяет традиционное зеркальное затенение Phong/Blinn более точным представлением того, как реальные поверхности реагируют на свет, основываясь на таких параметрах поверхности, как шероховатость (рисунок 4.1).

Рисунок 4.1 Одинаковые материалы в четырех разных световых окружениях

Рендер UE4 использует PBR-технологию, созданную для голливудских блокбастеров, и представляет собой мощную, удобную и простую в изучении систему материалов и освещения, объединяющую физически корректные материалы, освещение и отражения в единую систему для создания изображений высокого качества на основе трассировки лучей в реальном времени.

Те, кто использовал физически корректные рендеры, такие как Maxwell, должны очень быстро понять концепции рендера UE4. Другие поначалу будут немного сбиты с толку, но после того, как увидят, насколько хорошо выглядят материалы и освещение, насколько они просты в настройке и как замечательно они выдерживают почти все условия освещения, им будет трудно вернуться к вашим старым материалам на основе отражений.

Base Color

Base Color (основной цвет) определяет цвет материала со всеми тенями и бликами (UE4 предполагает, что это заполнит информацию об освещении).

Эти значения почти никогда не бывают полностью черными или полностью белыми. Уголь или свежий асфальт — около 0,02 (по шкале интенсивности от 0 до 1), тогда как свежевыпавший снег — 0,81, чистый бетон и песок — где-то посередине 0,51 и 0,36 соответственно.

Roughness

UE4 почти полностью отказывается от канала Specular. Хотя вы все еще можете получить доступ к этому значению, его использование предназначено только для особых случаев, когда система PBR не совсем выдерживает.

Вместо этого для определения яркости и плотности зеркальных отражений используется одно значение: **Roughness** (шероховатость; рисунок 4.2). Вы можете управлять шероховатостью либо с помощью одного значения, либо с помощью текстуры. На рисунке 4.2 изменены только значения шероховатости. Обратите внимание, как зеркальные блики и отражения становятся размытыми по мере увеличения значения.

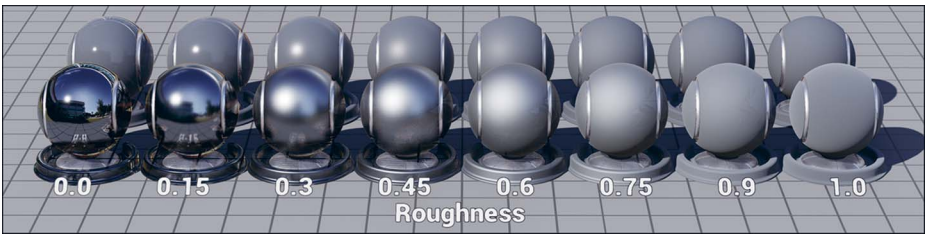


Рисунок 4.2 Roughness, как она влияет на неметаллические (верхний ряд) и металлические (нижний ряд) материалы

В реальном мире чем грубее поверхность, тем более рассеянным или размытым является отраженный свет. Это происходит из-за микроскопических дефектов на поверхности. Эти несовершенства рассеивают свет, который попадает на них, тогда как

гладкие поверхности без этих несовершенств не рассеивают свет, что приводит к резкому, зеркальному отражению.

> **Заметка**
>
> У вас может возникнуть соблазн уменьшить зеркальное значение материала, чтобы удалить блики, но не делайте этого! Вместо этого сделайте свой материал грубее.

Metallic

Metallic задает, насколько похож на металл ваш материал (рисунок 4.3). Это **скалярный**[6] **вход** (0,0–1,0); однако почти все поверхности будут либо на уровне 0, либо вблизи него, либо на уровне 1. Только в некоторых случаях металлическое значение находится между 0 и 1; например, в таких случаях, как слегка загрязненный металл, причудливая керамическая глазурь и т. д., или когда вы используете маску для определения областей, которые являются металлическими или нет.

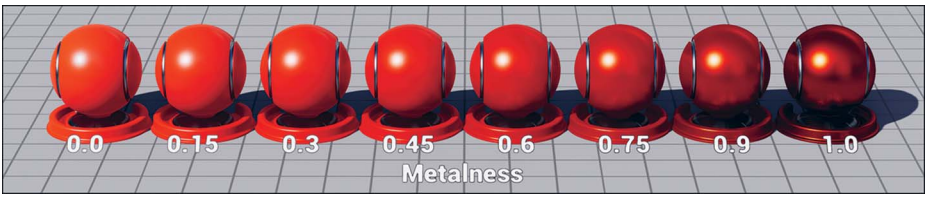


Рисунок 4.3 Matellic. Цвет отражения контролируется Base Color материала

Получение большего от PBR

PBR требует хороших входных данных. Тонны ресурсов теперь доступны для загрузки очень качественных текстур PBR в комплекте с картами Roughness, Normal и другими картами PBR.

Такие инструменты, как Substance Bitmap 2 Material (B2M), помогут вам подготовить существующие диффузные текстуры для PBR-рендеринга.

Также вы можете создавать эти карты вручную в Photoshop или другом программном обеспечении для редактирования изображений, но такие инструменты, как Substance, Quixel Suite или Substance B2M, могут делать гораздо более красивые наборы карт в разы быстрее.

[6] Scalar, скаляр — это просто материальный язык шейдеров для чисел с плавающей запятой.

Эти инструменты используют методы обработки изображений для создания Normals, Roughness и Base Color карт путем анализа форм и освещения в исходных изображениях.

Добавление деталей в Roughness

Roughness Texture карта — это ваш ключ к лучшим возможным материалам PBR. Осмотритесь. Вариации шероховатости говорят вам все, что нужно знать о поверхности, и все имеет некоторые вариации. Именно здесь должны жить царапины, грязь и пятна ваших материалов.

Сохранение Base Color простым

Держите свою Base Color карту простой. Она не должна содержать никакой информации об освещении или тени, только о цвете. Пусть рендер заполнит затенение и освещение.

Освещение в UE4

Освещение в Unreal Engine 4 очень похоже на освещение в любом другом 3D-пакете. Вы размещаете конусные, направленные, небесные и точечные огни, а также контролируете яркость, падение и цвет. Чем они отличаются, так это строгими ограничениями, наложенными на вас UE4, чтобы сбалансировать качество и функции с производительностью.

Заметка

UE4 не имеет встроенного динамического решения освещения GI. Единственная готовая к производству глобальная система освещения в UE4-это Lightmass, которая опирается на предварительно рассчитанные карты света и тени и вычисляется автономно в Editor.

Light Propagation Volumes

UE4 действительно предлагает ограниченный GI в реальном времени в **Light Propagation Volumes**, экспериментальную функцию, которую вы можете включить в настройках проекта. Поскольку она экспериментальная, она не совсем готова к выпуску и не имеет поддержки, но многие использовали ее для ошеломляющего эффекта.

NVIDIA VXGI

NVIDIA также предлагает решение GI для UE4 под названием VXGI. Для этого необходимо загрузить и укомплектовать пользовательскую сборку движка, модифицированную NVIDIA, что выходит за рамки данной книги. Это решение может обеспечить очень качественные результаты, но не поддерживается непосредственно Epic Games.

Освещение в движке трассировки лучей медленное, но обычно очень динамичное. Вы можете легко перемещать свет, модулируя цвет, чтобы имитировать заход солнца. Безупречный GI заполняет каждый уголок, поворот и щель богатым светом и тенью. Цена, которую вы платите, — время. Даже простая сцена с несколькими базовыми примитивами занимает несколько секунд для визуализации. Большинство реальных сцен занимают от нескольких минут до нескольких часов. Чтобы достичь частоты кадров в реальном времени, у нас есть только доля секунды, чтобы визуализировать кадр.

Ограничивая то, что свет может и не может делать, были реализованы огромные преимущества производительности. Поначалу с этими ограничениями бывает трудно справиться, но, зная, чего ожидать и чего нельзя делать, вы можете избежать некоторых дорогостоящих ошибок.

Понимание подвижности света (параметр Mobility)

Каждый Light и Mesh Actor, который вы помещаете на свой уровень, имеет параметр **Mobility** (мобильность). Три режима подвижности света в UE4 являются Moveable (подвижными), Stationary (стационарными) и Static (статическими). Каждое состояние имеет определенные возможности и ограничения, которые вы должны тщательно учитывать при освещении ваших сцен.

Подвижные источники света (Moveable)

Как следует из названия, если вам нужен свет или меш, которые вы можете перемещать, вам понадобится **Moveable**-тип. Используйте подвижные источники света осторожно. Хотя UE4 может визуализировать сотни незатененных, подвижных источников света одновременно, они являются одним из наиболее интенсивных эффектов производительности для визуализации в UE4, особенно если они отбрасывают тени.

Тени

Moveable Light (подвижные источники света) могут непосредственно освещать и отбрасывать тени на подвижные и статические меши. Это чрезвычайно важно. Если ваша геометрия сцены должна быть динамичной — например, с большим количеством двигающихся людей или автомобилей или с движущейся или изменяющейся геометрией сцены — вам нужно будет использовать Moveable.

Затенение дорого[7] стоит для динамического освещения, поэтому используйте его экономно. Чем больше радиус света, тем дороже его визуализация и тем больше акторов потребуется для создания теней.

[7] Термин «дорого» часто используется для описания штрафа за качество эффекта. Каждый кадр часто рассматривается как имеющий определенный «бюджет», и каждый свет, полигон и тень имеют стоимость против этого бюджета.

Динамические тени имеют низкое разрешение и, как правило, очень четкие края. Вы не можете легко получить мягкие ослабленные края теней, как это было бы с заранее рассчитанными статическими тенями (рисунок 4.4).

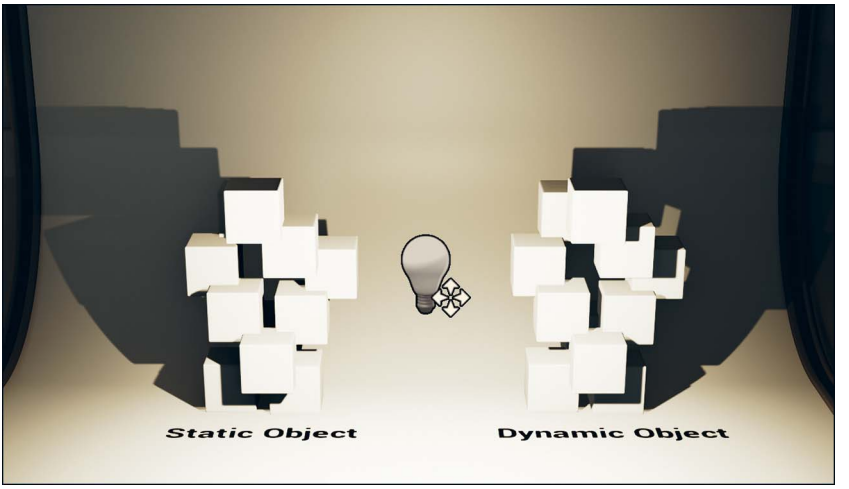


Рисунок 4.4 Тени Moveable Light на Moveable и Static объектах с динамическими тенями (без поддержки GI)

Global Illumination

Moveable Light не поддерживаются глобальным освещением (GI). Некоторые сторонние плагины и интеграции могут достичь этого, но их использование и интеграция выходят за рамки этой книги.

Specular Reflection

Moveable Light добавляют прямые зеркальные отражения на поверхности. Это дает им приятную подсветку на поверхностях PBR и поможет вашим материалам стать лучше.

Стационарные источники света (Stationary)

Средний ребенок семейства освещения UE4, Stationary Lights, использует статические и динамические пути освещения вместе, чтобы создать свет, который не может двигаться, но который может отбрасывать динамические тени и модулировать их цвет и интенсивность во время выполнения. Они могут вносить свой вклад в GI, но изменение интенсивности света влияет только на термин прямого освещения, а не на статическое освещение GI, хранящееся в световых картах (Lightmaps).

Stationary Lights полезны для статически освещенных сцен, поскольку они могут добавить детали и позволить динамическим акторам отбрасывать динамические тени;

однако вы должны использовать их осторожно, чтобы избежать некоторых серьезных штрафов за производительность.

Тени

Тени от статических мешей и на них запекаются в теневые карты с помощью Lightmass, но прямое освещение и тени от подвижных объектов и на них вычисляются динамически (рисунок 4.5).

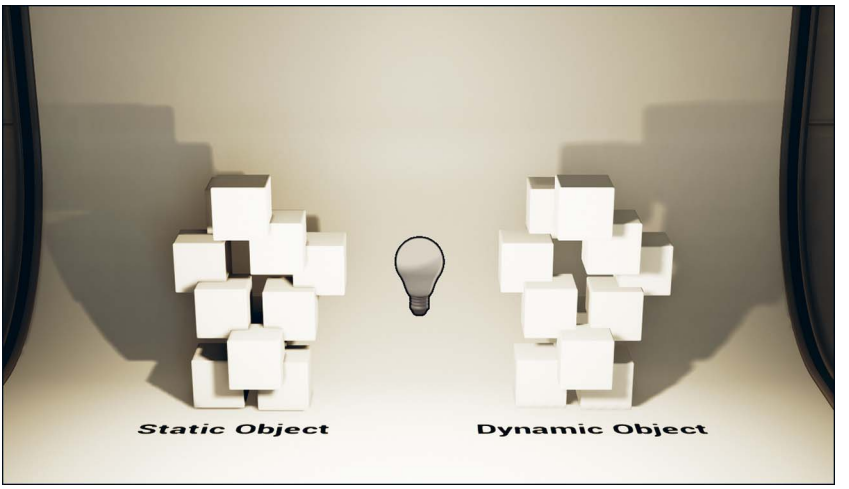


Рисунок 4.5 Stationary Lights, затеняющий Static Mesh слева с помощью статической теневой карты, и динамические тени, затеняющие Dynamic Mesh справа

Хотя стационарные источники света могут обеспечить лучшее из обоих миров, существует серьезное ограничение для этой магии. Каждый статически освещенный меш может быть подвержен влиянию только *четырех* стационарных огней одновременно.

Если более четырех стационарных источников света воздействуют на один меш, дополнительные Stationary Lights вернутся к Moveable Lights с динамическими тенями и будут иметь экстремальный штраф за производительность.

Чтобы избежать этой ситуации, UE4 предупреждает вас как описательными значками на Viewport, так и предупреждениями при сборке вашего освещения.

Вы можете просмотреть неподвижные Lightmap Overlaps, выбрав соответствующий режим просмотра в Editor. На рисунке 4.6 все четыре актора Spot Light Stationary, как и Directional Light солнечного света, поступающий из окна. Дополнительный Spot Light не может быть визуализирован как Stationary и вместо этого будет визуализирован как более дорогой Moveable.

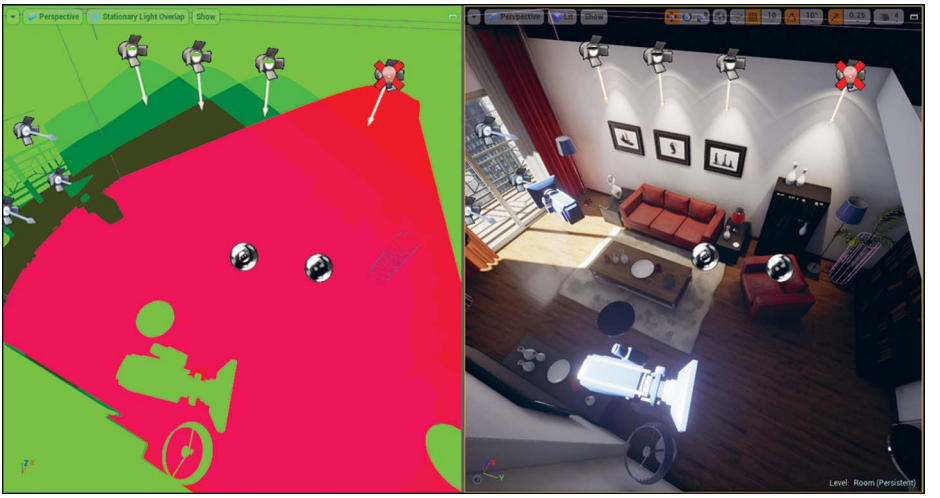


Рисунок 4.6 Слева режим просмотра Stationery Light Overlap

Global Illumination

Stationary Lights могут вносить свой вклад в GI, но если вы поменяете цвет или интенсивность света, GI не изменится вместе с ним. Вы можете управлять количеством GI, используя значение **GI Contribution** в свойствах Light Actor.

Specular and Reflection

Поскольку компонент прямого освещения Stationary Lights визуализируется динамически, он может добавлять зеркальные отражения на поверхности, как это делает Moveable Ligth.

Совет

Stationary Lights создают идеальный солнечный свет в ваших статически освещенных визуализациях.

Статические источники света (Static)

Static Lights, как следует из названия, полностью статичны. Они вообще не могут двигаться или меняться в реальном времени. Вся их информация об освещении и тени запекается в текстурах. Статические источники света используются исключительно системой Lightmass GI.

Заметка

Static Lights широко используются в архитектурной визуализации, где качество освещения важнее гибкости. Вы можете иметь практически неограниченное количество статических огней в ваших сценах, потому что путь статического освещения имеет фиксированную стоимость рендеринга после того, как Lightmass запекла информацию об освещении в световые карты.

Тени

Static Lights используют Lightmass для визуализации **световых карт**, текстур, содержащих информацию об освещении и затенении. Из-за этого статические источники света не могут непосредственно освещать или затенять динамические объекты (рисунок 4.7).

Вы можете настроить мягкость статических теней, изменив свойство радиуса источника света Light Actor. Вы не увидите эффекта, пока не отрендерите освещение в Lightmass.

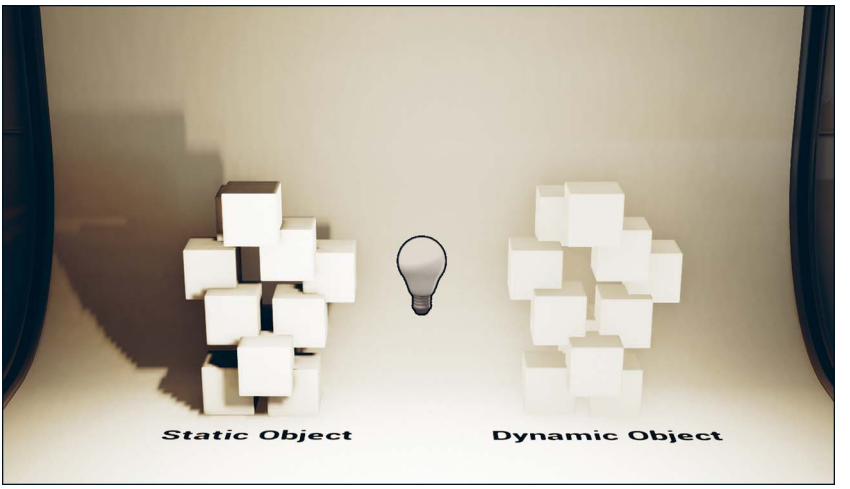


Рисунок 4.7 Static Light, затеняющий Static Mesh с запеканием Lightmap, и отсутствие тени от Dynamic Mesh справа

Global Illumination

Вы также не увидите никаких эффектов GI, пока не отрендерите свое освещение с помощью Lightmass. Статические источники света также хранят свое прямое освещение и информацию о GI в световых картах.

Specular and Reflections

Static Lights не создают зеркальных отражений. Однако они очень хорошо поддерживаются системой Reflection Capture, которая использует предварительно захваченные кубические карты HDR для применения зеркальных отражений к материалам PBR.

Отражения в реальном времени

Отражения уже давно стали одним из самых востребованных и труднодостижимых визуальных эффектов для графики в реальном времени. Сотни приемов и трюков были использованы, чтобы попытаться имитировать внешний вид и качество отражений от трассировки лучей.

Отражения необходимы для функционирования системы PBR. Без отражений материалы теряют глубину, и вместо этого приходится рассчитывать на прямые зеркальные блики для отображения информации о поверхности. Из-за этого все выглядит пластиковым и блестящим, что мы все привыкли ассоциировать с компьютерной графикой.

Отражения — это то, в чем трассировка лучей великолепна, но невероятно медленна, особенно когда вы начинаете размывать эти отражения. Как UE4 справляется с этим? В основном хитростями.

Два основных отражения в сцене UE4 — это статические кубические карты, генерируемые Reflection Capture Actors и Post Process Effect отражения в пространстве экрана. Оба они автоматически применяются к материалам и соответствующим образом модулируются. Это помогает при создании материалов и освещении сцен, потому что вам не придется настраивать материалы специально для сцен, как это могло бы быть сделано с предыдущими игровыми движками или другими рендерами.

Reflection Probes

UE4 использует reflection probes, называемые Reflection Capture Actors, вручную размещенные вокруг сцены для создания большинства поверхностных отражений. Эти датчики берут кубическую карту окружения с высоким динамическим диапазоном (HDR) и применяют ее к любому материалу, входящему в их радиус.

Они также вносят свой вклад в окружающее освещение как на динамических, так и на статических мешах, что делает их незаменимыми для высококачественного освещения.

Reflection Capture Actors не могут быть изменены во время выполнения.

Post Process Reflection

Screen Space Reflection (SSR) — это прекрасный метод постобработки, который использует информацию из G-буфера для восстановления отражений от уже отрисованной

2D-сцены. Это имеет некоторые серьезные ограничения, такие как раздражающий мельтешащий шум и невозможность отображения информации вне экрана, но он обеспечивает резкие, динамические отражения, которые большую часть времени выглядят фантастически и имеют относительно низкую стоимость рендеринга. Однако при более высоких разрешениях, таких как 4K-дисплеи и VR, SSR может стать проблемой производительности. Если у вас возникли проблемы с производительностью, можно попробовать отключить его в первую очередь в качестве варианта решения.

Post-Processing (постобработка)

Unreal Engine 4 в значительной степени полагается на постобработку всего: от сглаживания до отражения, от окклюзии до размытия.

С традиционно отрендеренными визуализациями вы обычно применяете эти эффекты к вашим визуализированным кадрам в пакете редактирования видео или эффектов, таких как After Effects или Nuke.

В UE4 вы применяете эффекты постобработки в реальном времени, определяя их либо для акторов на сцене в **Post Process Volume**, либо в настройках Post Process в Camera Actor. Эти настройки могут оказать значительное влияние на внешний вид и качество сцены и необходимы для того, чтобы ваши визуализации выглядели как можно лучше (рисунки 4.8 и 4.9).

Рисунок 4.8 Сцена UE4 без постобработки

Рисунок 4.9 Сцена UE4 с тяжелой постобработкой (блум, окклюзия, screen-space reflection и сглаживание)

Anti-Aliasing (сглаживание)

Сглаживание — одна из самых медленных операций рендеринга. Большинство рендеров используют форму super- или multi-sampling (рендеринг одного пикселя с более высоким разрешением для более точного среднего цвета, сглаживание краев). Этот метод просто слишком медленный для реального времени, потому что каждый визуализируемый пиксель уменьшает частоту кадров.

Unreal Engine 4 использует специально построенную систему temporal anti-aliasing (TAA), которая производит почти идеально сглаженные изображения за счет некоторой резкости изображения и некоторых возможных артефактов в движении[8]. TAA последовательно отсчитывает несколько отрисованных кадров и сравнивает их, чтобы создать среднее значение.

Менее эффективно сглаживание экранного пространства FXAA, который хотя и лучше, чем отсутствие сглаживания, обычно слишком груб для использования в визуализации.

Мягкость TAA с использованием других эффектов постобработки может придать вашему изображению кинематографичный, реалистичный вид и обеспечить высочайшее качество (рисунок 4.10).

[8] Ghosting чаще всего возникают в высокочастотных или шумных областях, которые меняются от кадра к кадру. Избегание шума в ваших сценах помогает скрыть этот артефакт.

Рисунок 4.10 Сравнение методов сглаживания, показывающее изменение качества краев

Bloom, Glare, and Lens Flares (блум, блеск и блики)

Поскольку большинство дисплеев не могут отображать яркость более 1, **Bloom** был разработан, чтобы позволить чрезмерно ярким пикселям быть яркими или блестящими, дабы помочь увеличить кажущийся динамический диапазон изображения. Блум эффективен, потому что он имитирует эффект объективов камер и наших глаз, когда они подвергаются воздействию очень яркого света.

Lens Flares (блики) имитируют межлинзовые отражения, которые появляются, когда камеры подвергаются воздействию высококонтрастного освещения.

UE4 визуализирует сцену с помощью линейного рабочего процесса и использует эту информацию для определения яркости для создания высококачественных зеркальных блумов бликов. Эти эффекты при экономном использовании могут помочь подчеркнуть HDR-освещение в вашей сцене и повысить качество ваших визуализаций.

Eye Adaption (Auto Exposure) (автоэкспозиция)

Общий эффект, используемый в фильмах и играх, заключается в регулировке экспозиции, когда камеры входят и выходят из тусклых и ярко освещенных областей. Это создает драматический блум света, и обычно сложно создать его в традиционно отрендеренной визуализации (рисунок 4.11).

UE4 обладает высоким динамическим диапазоном рендеринга, позволяя существовать как темным, так и очень ярким областям в пределах одной сцены. Когда игрок перемещается из помещения на улицу, перед его взглядом плавно регулируется яркость, имитируя человеческий глаз и/или камеры с помощью автоматической экспозиции.

Рисунок 4.11 Создание драматических эффектов позволяет камере изменять экспозицию при изменении интенсивности освещения

Depth of Fiel (глубина резкости)

DOF (глубина резкости) — это еще один эффект, который чрезвычайно трудоемок для визуализации с помощью большинства рендеров трассировки лучей, но может быть достигнут в режиме реального времени с помощью UE4 (рисунок 4.12).

Рисунок 4.12 Физически корректная глубина резкости с помощью Circle DOF

В UE4 существует несколько различных типов DOF: Gaussian, Bokeh, и Circle.

Gaussian

Gaussian DOF быстр, но имеет много артефактов. Сцена размыта, нереалистична и физически неточна.

Bokeh

Bokeh также размыта и неточна, но выглядит лучше благодаря включению формы Bokeh (апертурной маски) для высококонтрастных пикселей. Каждая форма Bokeh — это частица, визуализируемая для каждого пикселя, который ярче порога; как таковой, этот эффект может быть очень дорогим для визуализации.

Circle

Circle DOF — это относительно новая функция, и этот Sequencer и кинематографические камеры часто используют для достижения физически точной глубины резкости, основанной на диафрагме камеры и поле зрения.

Этот эффект является быстрым и придает вашим изображениям реалистичный, кинематографический вид. Вы можете использовать его практически во всех своих визуализациях с минимальными затратами на рендеринг.

Film Effects

Фильм и видео имеют много интерпиксельных шумов и других эффектов объектива, таких как виньетка и хроматическая аберрация. UE4 предлагает широкий спектр корректировок и эффектов постобработки, которые помогут вам достичь

фотореалистичного или стилизованного внешнего вида практически без снижения производительности.

Motion Blur (размытие)

Motion Blur — еще один эффект, который традиционно очень дорог в трассировке лучей. Интерполяция объекта во времени и пространстве происходит очень медленно. И снова в игру вступает постобработка. UE4 рендерит скоростной G-буфер, который позволяет применять очень качественное, но очень быстрое размытие движения ко всей сцене, а также к движущимся объектам (рисунок 4.13).

Рисунок 4.13 Высококачественное размытие

Screen Space Ambient Occlusion (SSAO)

Screen space ambient occlusion (окклюзия окружающего пространства экрана) — это один из самых важных графических методов, доступных для движков видеоигр, его используют почти все движки 3D-игр. SSAO создается с использованием Depth и World Normal G-буферов для определения того, где края и объекты находятся близко, рендеринга карты AO и композиции ее в реальном времени.

В сценах с динамическим освещением SSAO обеспечивает огромный визуальный импульс. Объекты получают контактные тени, что помогает им читать, как связаны поверхности, на которых они располагаются, и это значительно увеличивает глубину затененных областей. Хотя SSAO не является заменой трассировки лучей или предварительно рассчитанного AO, это огромное улучшение.

Screen Space Reflections

Преобразуя и искажая визуализируемое изображение на основе информации G-буфера, такой как World Normal и Depth, UE4 может создавать фальшивое, но хорошо выглядящее и быстро визуализируемое отражение.

Screen-space reflections помогают дополнить Reflection Capture Actors, обеспечивая динамические отражения с высокой детализацией (рисунок 4.14). Это помогает заземлить объекты и обеспечить максимально точный внешний вид и динамические отражения.

Как и все другие эффекты экранного пространства, эти отражения не могут отображать что-либо вне экрана. Это может создать некоторые артефакты перехода и другие странные эффекты при перемещении по сцене. Использовать ли их — это художественный выбор.

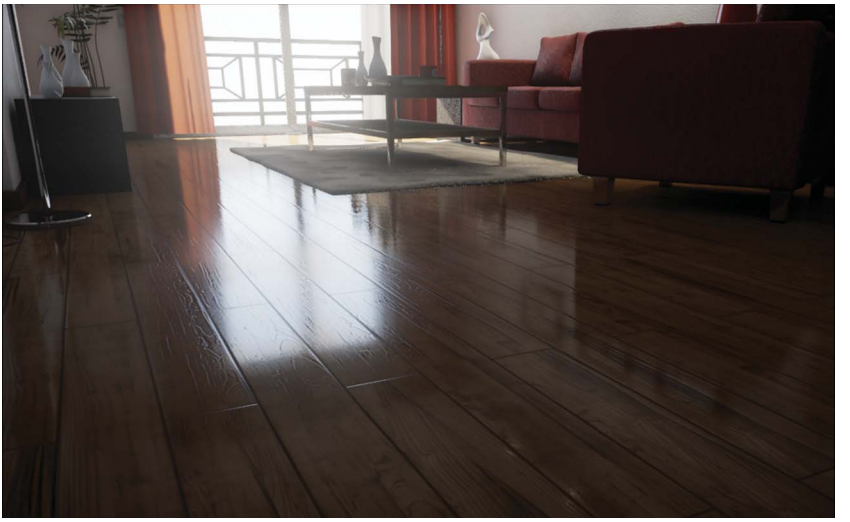


Рисунок 4.14 Резкие отражения экранного пространства работают в гармонии с захваченными отражениями кубической карты от Reflection Capture Actors

Screen Percentage

Хотя UE4 не предлагает multi- или super-sampling-сглаживания[9], это действительно так. UE4 обладает очень качественным инструментом для масштабирования изображений и возможностью рендеринга в любом разрешении, независимо от экрана.

[9] UE4 предлагает Forward Rendering, который позволяет использовать MSAA при проигрывании многими преимуществами Deferred Rendering, такими как динамическое освещение.

По мере увеличения разрешения проекта экранно-пространственные эффекты становятся все более медленными в использовании. Пользователи мониторов с высоким разрешением более остро ощутят эту настройку.

Post Process Materials

UE4 предоставляет G-буферы, которые он использует для визуализации и компоновки своего конечного изображения в Material Editor. UE4 позволяет создавать пользовательские эффекты постобработки, расширяющие возможности рендеринга UE4, используя тот же Material Editor и систему Material Instancing, что и в остальной части движка.

Они могут варьироваться от простых, таких как создание собственной системы виньеток, до сложных, таких как написание эффектов затенения ячеек и выделения контуров (рисунок 4.15). Небо — это ваш предел благодаря гибкости редактора материалов.

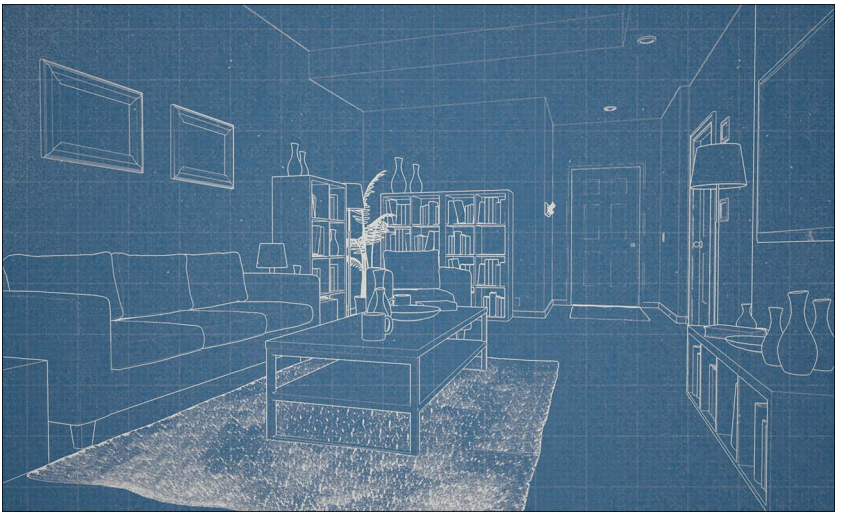


Рисунок 4.15 Пример использования материалов после обработки для полного изменения способа визуализации сцен UE4

Хотя они называются Post Process Effects, на самом деле они рассчитываются во время выполнения проекта, а не после него. Они рендерятся в конце каждого кадра, в то время как следующий кадр начинает визуализироваться. Они работают в реальном времени, реагируя на динамично меняющуюся сцену и точку зрения.

Post Process Effects могут сделать вашу сцену похожей на первое задание студента-искусствоведа по редактированию фотографий с яркими цветами, виньетками и вспышками линз, доминирующими над видом. Если использовать их с умом, они также могут придать ему кинематографичный, реалистичный вид.

Заключение

UE4 имеет одну из самых надежных доступных систем рендеринга. Реализм и простота использования просто непревзойденные, а качество говорит само за себя. Большинство систем основаны на техниках и системах, которые вы использовали в течение многих лет, так что они знакомы и просты и для понимания.

PBR вводит совершенно новый способ смотреть и определять материальные поверхности, которые удобны для художника и выглядят великолепно. В сочетании с арсеналом световых трюков и инструментов вы можете получить изображения, схожие по качеству с трассировкой лучей в реальном времени.

ГЛАВА 5

МАТЕРИАЛЫ

Материалы в UE4 подобны визуальным программам, которые запускаются на каждом пикселе (пиксельный шейдер) и вершине (вершинный шейдер) на сцене. Используя визуальный Material Editor, вы можете создать ошеломляющий массив поверхностей, с помощью лишь нескольких параметров и текстур. Даже очень простые материалы выглядят поразительно благодаря PBR. По мере совершенствования навыков вы можете опираться на эти материалы, делая их более гибкими и динамичными, включая интерактивность и программируемость.

Обзор материалов

Материалы являются одним из наиболее важных элементов для создания убедительного интерактивного контента. Материалы определяют, как каждый отдельный пиксель в вашей сцене реагирует на свет, тень и отражение.

Создание материалов в реальном времени может сильно отличаться от создания материалов в 3D-приложениях и рендерах. Как и большинство процессов UE4, работа с материалами сосредоточена на интерактивности и производительности.

UE4 имеет WYSIWYG, предварительный просмотр материалов в реальном времени и — с помощью визуального Material Editor — возможность программировать буквально на основе каждого пикселя и каждой вершины, как ведут себя материалы. Благодаря интерактивному характеру UE4 вы также получаете мгновенную обратную связь на сцене о том, как выглядят ваши материалы (рисунок 5.1).

Рисунок 5.1 Предварительный просмотр материалов в UE4

Создание материалов

Материалы в UE4 — это ассеты, такие как меши и текстуры, хранящиеся в папке Content. Они создаются исключительно в Editor и не могут быть отредактированы вне Material Editor. Material Editor — это визуальный редактор скриптов на основе узлов, который позволяет создавать чрезвычайно эффективные шейдеры на High-Level Shading Language (HLSL) с помощью простого, удобного для художника интерфейса.

Чтобы создать новый Материал, щелкните правой кнопкой мыши в Content Browser, выберите пункт Create Material или нажмите кнопку **Add New** в Content Browser и выберите пункт Material в появившемся меню.

Применение материалов

Вы можете применять материалы к мешам различными способами: перетаскиванием из Content Browser на меш в вашей сцене, через Static и Skeletal Editor или через диалоговые окна Object Property.

Вы можете выбрать метод, который лучше всего подходит для вас, но я настоятельно рекомендую применять материалы непосредственно к вашим ассетам (а не к их ссылкам на сцене) с помощью Static и Skeletal Editor. Это гарантирует, что каждый раз, когда вы помещаете свой меш на уровень, будут применены соответствующие материалы.

Изменение материалов

Изменение материалов во время выполнения является важным инструментом в арсенале инструментов художника визуализации UE4. Гибкость редактора материалов не имеет себе равных и позволяет создавать почти бесконечный набор визуальных эффектов, которые могут принести живость, интерактивность и реализм вашим сценам.

Чертежи усиливают эту гибкость, предоставляя вам возможность добавлять логику и интерактивность к материалам, динамически устанавливая Material Property во время выполнения.

UE4 Material Editor

Unreal Engine Material Editor — это немного инженерной магии и блеска пользовательского интерфейса, соединенных вместе, чтобы создать удобный для художника способ создания сложных пиксельных и вершинных HLSL-шейдеров без необходимости написать даже одной строки кода.

Используя визуальный редактор скриптов, вы создаете материалы через сеть узлов Material Expression. Каждый узел представляет собой фрагмент кода HLSL, и когда вы соединяете их, движок записывает код HLSL в фоновом режиме. Вы можете заглянуть в Material Editor и увидеть, как в режиме реального времени выглядит написанный код.

Визуальная природа Editor делает его доступным в использовании, а PBR-рендеринг делает создание реалистичных материалов очень простым. Вы также можете создавать сложные динамические интерактивные материалы, которые включают в себя передовые методы, такие как тесселяция, параллаксная окклюзия, деформация вершин и анимация.

Открытие Material Editor

Вы можете получить доступ к Material Editor только двойным щелчком мыши на ассете материала в Content Browser. Если у вас нет материала в проекте, нужно создать его, как описано ранее в разделе «Создание материалов».

Интерфейс Editor

Редактор материалов состоит из обычной строки menu и toolbar, которые являются общими для большинства редакторов UE4 (рисунок 5.2). Вы увидите предварительный просмотр материала в режиме реального времени на Viewport, динамической Details Panel, которая представляет доступные параметры и опции для выбранных узлов Expression. На панели Palette отображается список доступных Expressions.

Рисунок 5.2 Интерфейс Material Editor

В центре находится **Graph Editor**. Вот тут-то и происходит волшебство. В каждом материале находится узел Base Material. Этот узел имеет входные данные для каждого аспекта материала, которые можно изменить, подключив к ним другие узлы.

Размещение узлов

Существует несколько способов размещения узлов в Graph View. Самое очевидное — это **Palette**. Оттуда вы можете просто перетащить узлы Expressions в Graph Editor.

Вы также можете щелкнуть правой кнопкой мыши в пустом пространстве в Graph Editor, чтобы открыть контекстную Palette, дающую вам доступ к возможным Expressions.

Для вас доступно множество Expressions, с помощью которых вы можете создавать свои материалы. В обеих Palette вы можете легко фильтровать узлы по названиям, используя поле поиска, чтобы быстро найти нужное.

Использование Viewport для предпросмотра

Предварительный просмотр — одна из лучших функций Material Editor. Вы получаете мгновенную обратную связь почти по каждому изменению вашего материала и воспроизведение в реальном времени эффектов, таких как панорамирование текстур и рябь волн.

Preview Viewport — это Viewport живой игры со всеми теми же функциями постобработки и рендеринга. Это означает, что вы получаете представление 1:1 о том, как будет выглядеть ваш материал в проекте.

Preview Viewport также позволяет выбрать стандартные примитивы, такие как кубы и сферы, для предварительного просмотра вашего материала. Вы также можете загрузить собственный пользовательский меш, сначала выбрав нужный меш в Content Browser, а затем щелкнув значок чайника в Preview Viewport Material Editor.

Меши предварительного просмотра по умолчанию имеют высоту 500 см, поэтому меши гораздо большего размера может быть трудно увидеть поначалу. Если у вас возникли проблемы с видимостью меша, вы можете нажать клавишу F, пока Viewport активен, чтобы центрировать вид на меше и увеличить масштаб до ее границ.

Обратите внимание, что материал использует орбитальную камеру, которая ведет себя немного иначе, чем обычная камера Viewport. Просто нажмите и перетащите, чтобы вращаться вокруг вашего объекта, и используйте колесо мыши для увеличения и уменьшения масштаба. Если вам нужно изменить угол освещения, удерживайте нажатой клавишу L при перетаскивании влево с помощью мыши.

Хотя Viewport обеспечивает отличный предварительный просмотр вашего материала, вы всегда должны тестировать его в своих сценах, потому что изменения в освещении, постобработке и т. д. могут повлиять на внешний вид ваших материалов.

Компилирование шейдеров

По мере размещения и подключения узлов в Editor вы будете видеть обновление Preview Viewport в режиме реального времени. Этот предварительный просмотр перекомпилируется относительно быстро по мере внесения изменений. Вы заметите, что в мешах сцен не происходит изменений, которые могли бы привести к применению материала, пока вы не нажмете кнопку Apply на toolbar для компиляции материала. Вы также можете использовать кнопку Save, чтобы сохранить свой материал. Нажатие этой кнопки вызывает принудительную компиляцию, если она требуется и, как следствие, сохранение.

Компиляция создает код HLSL и кэширует аппаратные шейдеры для конкретной платформы, которые используются для отображения материала на мешах на вашей сцене. Этот шаг необходим, потому что компиляция некоторых материалов может занять более минуты в зависимости от того, как они используются и сколько Material Parameters они содержат.

Применяя материалы к различным типам мешей (статические, скелетные, частицы, листва, рельеф, экземпляры и т. д.), вы увеличите количество требуемых перестановок шейдеров в вашем материале и время компиляции.

Во время компиляции материала или при наличии ошибок материал, примененный к мешам на сцене, будет отображаться как серый материал по умолчанию. Обязательно следите за Material Editor после компиляции, чтобы убедиться в отсутствии ошибок. Если они есть, то они появятся в окне Output Material Editor.

Сохранение

Вы еще не закончили! Сохраните этот материал! Вид материалов, примененных к вашей сцене, может дать ложное ощущение, что все сохранилось. Компиляция не сохраняет ваши материалы; вы должны вручную сохранить ваши материалы (которые затем компилируют шейдер перед сохранением).

Как работают материалы в Unreal Engine 4

Материалы в UE4 являются продолжением пайплайна PBR-рендеринга и плотно интегрированы в пайплайн освещения и отражения. Материалы определяют, как каждая поверхность в мире реагирует на свет, отражения и тени.

Пиксельный и вершинные шейдеры

Первое, что нужно понять о работе материалов, — это как пиксельные и вершинные шейдеры используются для рендеринга изображений. Это, похоже, ренедеринг большинства изображений, но существует более тесная интеграция с оборудованием

рендеринга (вашим графическим процессором), и материалы легче раскрывают этот аппаратный интерфейс.

Когда каждый кадр рендерится, сначала он проходит через вершинный шейдер. Этот шейдер преобразует вершины сцены в трехмерном пространстве, назначает материалы и подготавливает сцену к оценке пиксельным шейдером. Здесь UV-координаты преобразуются и поворачиваются, применяются смещение и тесселяция и любые другие вычисления вершин и геометрических уровней выполняются перед отправкой в пиксельный шейдер.

После завершения прохода вершинным шейдером, пиксель за пикселем изображение рендерится пиксельным шейдером. Когда каждый пиксель рендерится, он получает информацию от вершинного шейдера, такую как направление нормали поверхности, UV-координаты, а также данные о материале и текстуре, необходимые для рендеринга пикселя.

Пиксельный шейдер использует эту информацию для выборки текстур, выполнения математических операций и возврата одного линейного HDR-значения пикселя. После того как все пиксели для одного кадра будут отрендерены, изображение передается в цепочку постобработки для отображения тона и применения к нему других эффектов. Затем это изображение выводится на экран в качестве окончательного изображения.

Материалы — это математика

Вы быстро заметите, что большинство узлов Material Expression, доступных в Material Editor, являются математическими терминами. Выражения (Expression) существуют почти для всех обычных математических операций: сложения, вычитания, синуса, квадрата, степени и т. д.

По своей сути каждый материал представляет собой сложное математическое выражение. Значения цвета пикселя добавляются, вычитаются и модулируются другими способами на основе входных данных, таких как положение света, мировое положение и направление, UV-координата пикселя, цвет вершин и так далее.

Цвета как числа

Вы, вероятно, не привыкли думать о цифрах, стоящих за изображениями в вашей повседневной работе. Для большинства художников текстура RGBA означает цветное изображение с непрозрачностью. Для художника UE4 изображения RGBA часто означают четыре значения оттенков серого, которые можно получить и изменить.

Эта концепция может показаться запутанной, но именно так вы работали в Photoshop и других программах для редактирования изображений в течение многих лет. Операции со слоями, такие как сложение, умножение и так далее, — это все пиксельные математические операции, которые UE4 просто предоставляет более непосредственно.

Линейный цвет

Материалы в UE4 рендерятся внутренне с плавающей точкой, поэтому, хотя текстуры могут быть ограничены значениями 0–255 для каждого канала, материал, в котором они рендерятся, может иметь значения с плавающей точкой от 0 до 1 и выше. Если что-то возможно с числами, это возможно в Material Editor.

Однако ваш монитор, вероятно, не может отображать информацию выше или ниже яркости от 0 до 1. Например, значение цвета ниже 0 будет просто отображаться черным, а значение выше 1 может отображаться как белый (возможно, с некоторым блумом постобработки).

Карты нормалей

Ничто не иллюстрирует тезис «Материалы — это математика» лучше, чем карты нормалей (рисунок 5.3). Карты нормалей — это 3D-векторы (значения направлений X, Y, Z), хранящиеся в каждом пикселе в виде красных, зеленых и синих значений. XYZ = RGB. Движок использует эту информацию и изменяет видимое направление мирового пространства каждого пикселя, чтобы изменить его реакцию на освещение окружающей среды.

На рисунке 5.3 фиолетовый цвет — это нейтральная нормаль, которая вообще не меняет нормаль пикселя во время рендеринга; красный канал меняет направление X, а зеленый — Y.

Заметка

Вы когда-нибудь замечали, что Transform Gizmo в UE4 и многих других 3D-приложениях использует красный цвет для оси X, зеленый-для оси Y и синий — для оси Z? RGB = XYZ = UVW

Эти значения взаимозаменяемы и рассматриваются материалами как общие числовые массивы и могут быть математически оценены как таковые.

Когда каждый пиксель рендерится, значение карт нормалей в этом месте буквально добавляется (+) к касательной-пространственной вершинной нормали поверхности модели, возвращаемой вершинным шейдером, возвращая новую нормаль и, следовательно, другой цвет после применения к ней освещения.

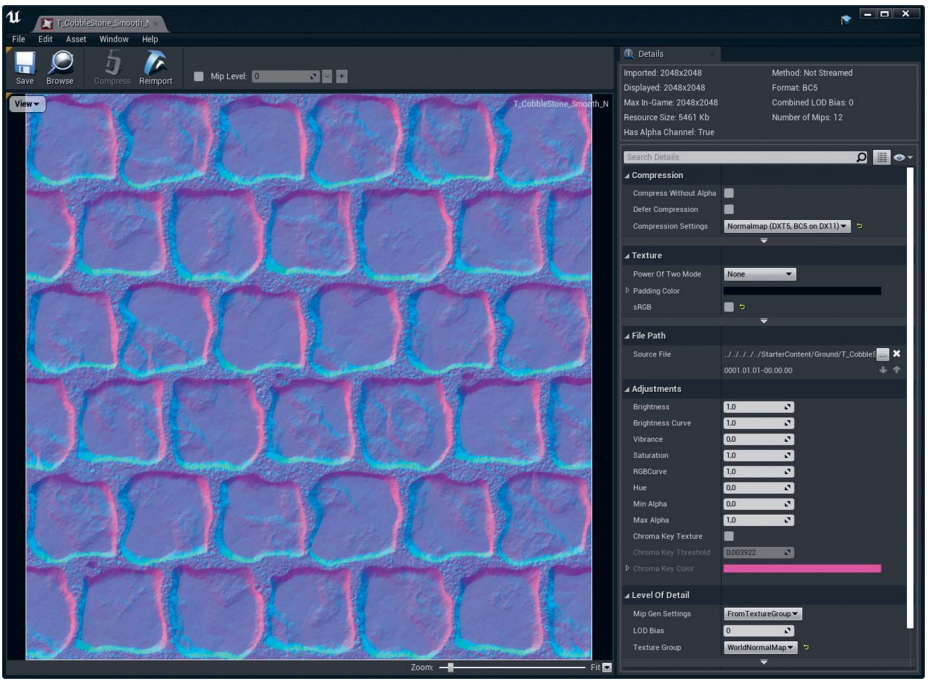


Рисунок 5.3 Карты нормалей хранят данные о направлении движения

Заметка

Карты нормалей, созданные различными программными пакетами, дают несколько иные результаты и могут потребовать корректировки. Например, Maya выводит карты нормалей с зеленым каналом, перевернутым со стандарта UE4. Вы можете легко исправить это в диалоговом окне Texture Editor, установив опцию Flip Green Channel на true.

Как и все карты данных, карты нормалей вы не должны хранить с коррекцией sRGB на них.

Упаковка маски текстур

Точно так же, как вы можете хранить свою маску непрозрачности в альфа-канале текстуры, распространенный метод, используемый в UE4, заключается в использовании других трех каналов (RGB) для хранения трех дополнительных масок вместо информации о цвете (рисунок 5.4).

В этой картинке каждый текстовый слой — это просто сложение. Хотя комбинированное изображение бесполезно, есть четыре четких изображения в оттенках серого.

Рисунок 5.4 Изображение RGB, используемое для того, чтобы содержать четыре маски, удобно названные RGB и A

Поскольку каждая маска должна иметь только значение от 0 до 1, популярный метод сокращения чтения текстур и памяти заключается в объединении нескольких текстур масок в одно изображение RGBA. Это означает, что вы можете иметь четыре маски в одной текстуре. Это может быть немного сложно для автора, но способно уменьшить память и помочь производительности в сложных сценах и материалах.

Мощность рендеренга на современных GPU вкупе с относительно малым объемом требующейся для визуализации большинства текстур делает этот метод бессмысленным. Тем не менее для продолжительных и очень больших проектов паковка текстур позволяет серьезно сэкономить на объемах памяти и времени рендеренга.

Заметка

Текстуры данных, такие как маски и карты нормалей, не должны иметь настройки sRGB gamma, применяемой к ним. Вы должны убедиться, что этот параметр имеет значение false в UE4.

Функции материалов

Вы часто будете хотеть создать функциональные возможности в ваших материалах, которые захотите повторно использовать в других материалах. Material Functions позволяют свернуть сети кода материала в ассеты, которые затем могут быть добавлены к любому материалу в вашем проекте в качестве узла Expression.

Вы можете добавить входы и выходы, позволяющие Material Functions обрабатывать данные и возвращать преобразованные данные. Обычная функция может быть

такой же простой, как взятие нескольких цветовых входов и возврат их среднего значения, или такой же сложной, как нанесение мозаичных волн на поверхность океана.

Functions могут даже возвращать целые определения материалов, включая текстуры и другие параметры и значения. Помещая две функции в материал, вы можете смешать их вместе с маской, чтобы определить, где будет отображаться та или иная функция, сохраняя код графического редактора материала чистым и читаемым.

Типы поверхностей

Материалы UE4 используют несколько различных Surface Domains или Types. Каждый тип представляет собой совершенно иную кодовую базу рендеринга, чем другие, и часто имеет совершенно разные возможности и параметры.

Понимание того, когда и почему использовать каждый тип, очень важно.

Непрозрачность

Большинство материалов, которые вы будете использовать в своих проектах, непрозрачные. Непрозрачные материалы, как вы уже догадались, непрозрачны! Вы никогда не сможете увидеть пиксель, который находится за непрозрачным пикселем. Он просто никогда не будет отрендерен.

Непрозрачные материалы — самые эффективные, полностью освещенные, самые надежные доступные материалы. Используйте непрозрачные материалы в большинстве случаев в настройке Materials.

Маскируемые материалы

Маскируемые материалы подобны непрозрачным, но могут иметь форму прозрачности, применяемой к ним в виде 1-битной маски непрозрачности. Это означает, что они могут быть либо полностью непрозрачными, либо полностью прозрачными с жестким краем, разграничивающим их.

Маскированные материалы выглядят великолепно и почти так же быстро визуализируются, как и непрозрачные материалы, но могут подвергаться тяжелой перерисовке[10] там, где они прозрачны. Маскированные материалы чаще всего используются для листвы в UE4, обеспечивая очень высокое качество освещения и обеспечивая некоторый уровень прозрачности.

[10] Перерисовка — это когда несколько поверхностей визуализируются в одном пикселе. Когда непрозрачная поверхность визуализируется, объекты за ней не оцениваются, поэтому визуализируется только один пиксель. Прозрачные поверхности должны также отображать любые пиксели позади них в дополнение к их собственным. Это приводит к увеличению сложности рендеринга этого пикселя.

Полупрозрачность

Полупрозрачные материалы визуализируются с использованием совершенно другого пути рендеринга, чем большинство материалов в UE4, и составляются на конечном изображении. Это создает несколько проблем для их использования на сценах.

Полупрозрачные поверхности получают очень приблизительную информацию об освещении и не являются физически корректными. Они в основном предназначены для таких эффектов, как дым и огонь, и, хотя вы можете сделать из них стекло, воду и другие полупрозрачные поверхности, они никогда не приблизятся к качеству этих материалов в рендерах с трассировкой лучей.

Непрозрачность

Когда вы устанавливаете свой материал на полупрозрачный, вы включаете вход непрозрачности. Этот входной сигнал принимает значение от 0 до 1 и просто определяет, насколько поверхность будет сливаться с пикселями позади нее. Поверхности с непрозрачностью 1 будут выглядеть непрозрачными, но все равно будут визуализировать пиксели позади них, вызывая перерисовку и штраф за производительность.

Преломление

Светопрозрачные материалы позволяют использовать **преломление**. Этот эффект является эффектом экранного пространства и ограничен в UE4, но при экономном использовании может добавить много визуального интереса к прозрачным поверхностям в ваших сценах.

Рефракция использует нормаль экранного пикселя и смещает образец пикселя позади него на это нормированное направление, умноженное на величину рефракции. Это немного противоречит здравому смыслу, но 1,0 — это не преломление 0,9 искажает пиксели на –10%, а 1,1 — это искажение на 10%.

Значений между 0,95 и 1,05 достаточно для большинства целей, чтобы получить реалистичное искажение.

Экземпляры материалов

Ключом к динамическим управляемым материалам являются **Material Instances**: облегченные, инстанцированные версии материалов с наследованием, не отличающимся от отношений «ребенок — родитель» между классами в языках программирования (рисунок 5.5).

Material Instances не требуют компиляции, как это делают полные материалы при их изменении. Используя **Material Parameters** в ваших материалах, вы можете динамически обновлять их на лету как в Editor, так и во время выполнения вашего проекта.

Изменение свойств Material Instances во время редактирования сцен обеспечивает удивительную обратную связь о внешнем виде ваших сцен в режиме реального времени. Пока вы совершенствуете настройки своих материалов, вы видите, как они мгновенно обновляются на вашей сцене с применением финального освещения и эффектов постобработки.

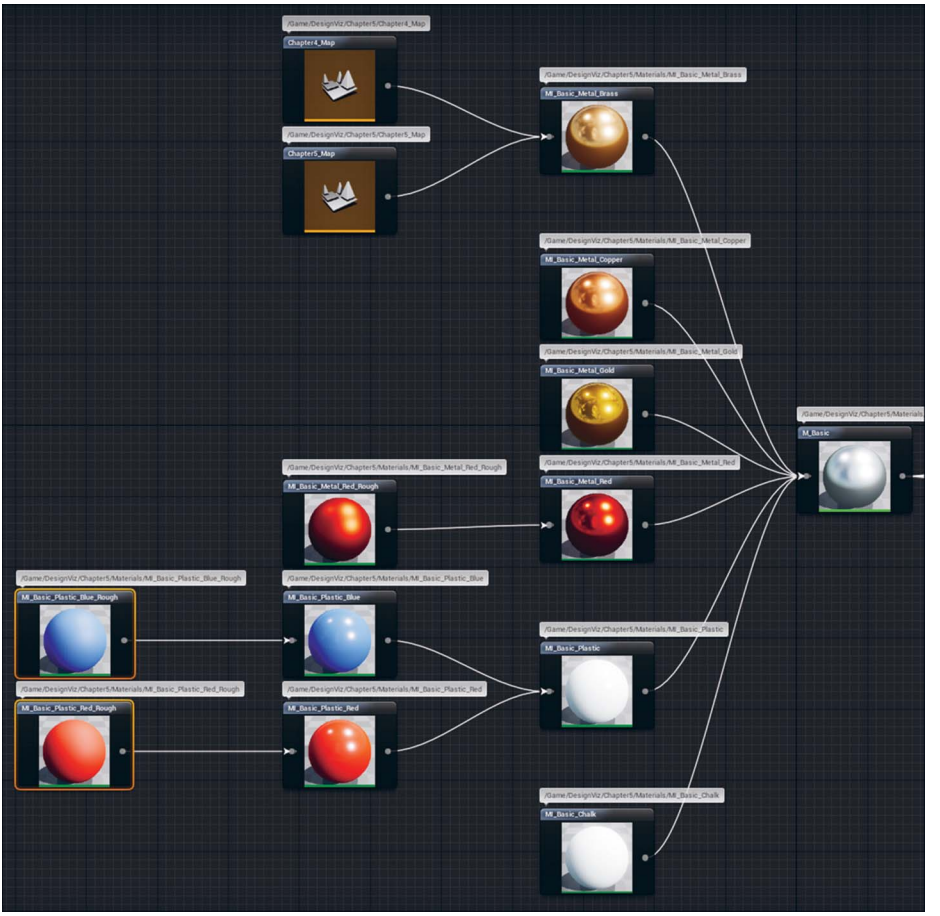


Рисунок 5.5 График наследования, показывающий, как различные Material Instances являются потомками для одного материала (M_Basic), в крайнем правом углу

Параметры материалов

Параметры — это специальные входные узлы в материалах, которые подвергаются воздействию Material Instances. Это могут быть цвета (векторы), текстуры или простые скалярные (с плавающей точкой) значения. Они отображаются в Material Instance Editor как отдельные элементы строки, которые могут быть переопределены или

унаследовать свойство своего родителя. Это позволяет вам создавать материалы, которые могут быть повторно использованы и изменены самыми разными способами.

Наследование

На верхнем уровне Material Instance являются облегченными экземплярами или ссылочными копиями материалов. Они не могут быть отредактированы с помощью Material Editor, а редактируются с помощью специального Material Instance Editor, который открывается при двойном щелчке по ассету Material Instance.

Material Instances наследуют все значения параметров родительского материала или Material Instance.

Это означает две вещи: если вы изменяете свойство в экземпляре, его родительский материал всегда остается незатронутым, и если вы изменяете родительский материал, то все экземпляры, связанные с этим материалом, будут обновлены.

Вы также можете делать экземпляры других Material Instances, создавая сложные цепочки наследования материалов. Такая практика способна привести к замусориванию проекта, особенно если вы точно не знаете, для чего создаете экземпляры экземпляров. Обязательно составьте хороший план, прежде чем идти по этому пути. Часто простое копирование экземпляра и изменение одного свойства лучше, чем продуцирование экземпляров экземпляров.

Переопределение параметров

Переопределение Material Parameter в экземпляре позволяет этому экземпляру иметь другое значение, чем его родительский элемент. Вы должны явно определить каждое свойство, которое переопределяете, щелкнув флажок рядом с ним, что позволяет редактировать его. Все дочерние элементы этого экземпляра также будут обновляться при изменении этих значений.

Организация параметров

Каждый параметр имеет Group-значение, которое можно задать. Это используется для организации списка параметров Material Instance Editor. Параметры, не имеющие набора групп, помещаются в группу Default.

Когда у вас есть материалы с большим количеством параметров, важно иметь привычку назначать группы для ваших входных данных, потому что они могут легко стать хаотичными и трудными в использовании.

Мастер-материалы

Используя Material Instances, можно создать единый материал, который способен обрабатывать каждый тип поверхности в проекте. Хотя на первый взгляд это может показаться хорошей идеей, существуют веские причины для создания нескольких Master Materials для различных типов поверхностей и акторов.

Например, вы, вероятно, не хотите, чтобы один и тот же мастер-материал обрабатывал деревянные полы и дым, исходящий от свечи.

Хотя теоретически можно было создать один материал с достаточным количеством параметров для обработки всех возможных событий, этот единственный материал стал бы проблемой, порождая длительное время компиляции и огромные кэши шейдеров, поскольку движок создает слишком много перестановок шейдеров, чтобы справиться со сложностью шейдера.

Создавайте мастер-материалы, которые просты, но могут выполнить максимально возможное количество вариаций, при этом не становясь причиной проблем.

Простой материал

Вы можете достичь отличного качества и разнообразия с помощью очень простых материалов в UE4. В этом разделе вы увидите, как простой материал настраивается с параметрами для Material Instance. Затем мы рассмотрим построение массива таких экземпляров, демонстрируя, как даже несколько параметров могут создавать множество поверхностей.

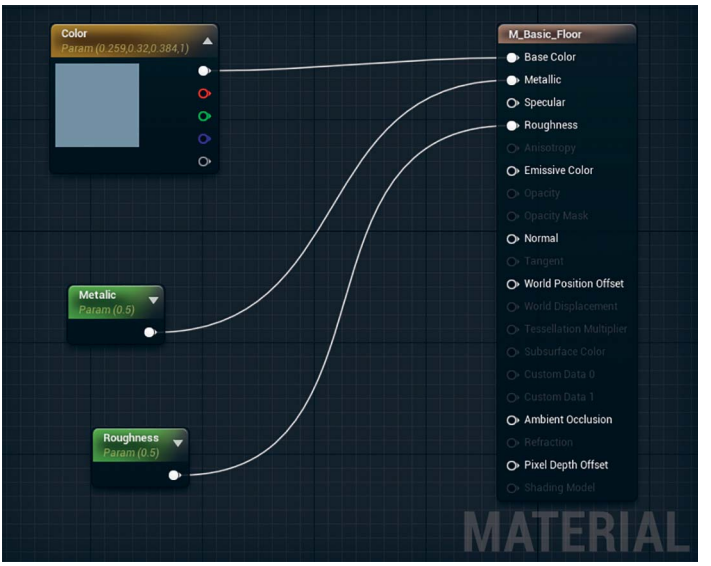


Рисунок 5.6 Самый простой материал

Рисунок 5.6 показывает наиболее простую настройку материала, практичную в UE4. В граф материала помещены только векторный параметр (RGBA), называемый Base Color, скаляр (float), называемый Metallic, и еще один скаляр, называемый Roughness. Они подключены к одноименным входам.

Размещение узлов параметров

Вы можете просмотреть все доступные параметры узлов Выражений (Expression), выполнив поиск слова Parameter в Pallets узлов. Есть довольно много доступных, но нам потребуются только скалярные и векторные типы, чтобы сделать этот материал.

При создании параметра необходимо указать его имя (Name) и значение Default. Вы также можете назначить группу каждому параметру, если нужно организовать много параметров.

Затем просто перетащите провод от выходного узла по каждому параметру и подключите его к соответствующему входному контакту другого узла — в этом случае атрибуты материалы будут входами материала.

По мере добавления новых узлов в материал вы обнаружите, что иногда требуется преобразовать статическое значение, например Texture Sample или скалярное значение, в параметр. Щелкнув правой кнопкой мыши на определенных типах узлов, таких как векторы, текстуры и скалярные узлы, и выбрав **Convert to Parameter**, вы можете быстро конвертировать статические и динамические параметры и обратно.

Вы также можете разместить много узлов, используя сочетания клавиш. Удерживайте нажатой определенную клавишу, например **T**, и щелкайте ЛКМ в графе, чтобы создать узел Texture Sample. Держите **S** и щелкните ЛКМ, чтобы разместить скалярный параметр, или удерживайте **V** и нажмите, чтобы установить вектор.

С помощью этих трех простых параметров можно создать огромный ассортимент Material Instances. Даже эти простые материалы будут выглядеть фантастически в UE4, подчиняясь физическим законам освещения в соответствии с требованиями PBR.

Создание экземпляров материала

Создание экземпляра материала можно провернуть несколькими способами. Один из способов — создать экземпляр материала с помощью меню Add New или щелкнуть правой кнопкой мыши в Content Browser и выбрать Material Instance из группы Materials & Textures. Затем вам нужно открыть этот ассет и вручную назначить материал, который вы хотите видеть в качестве родительского.

Проще всего щелкнуть правой кнопкой мыши на материале, экземпляр которого вы хотите создать, и выбрать в меню пункт **Create Material Instance**. Это создаст новый экземпляр, назовет его и назначит ему родителя.

После того как родительский материал будет назначен экземпляру материала, вы увидите параметры, определенные в вашем материале, перечисленные на панели Details (рисунок 5.7). Установка флажка рядом со свойством объявляет его переопределенным, и теперь экземпляр больше не будет использовать параметр, определенный в родительском материале, даже если этот параметр изменится в родительском материале. На рисунке 5.7 видно, что M_basic определяется как Parent. Base Color переопределен так же, как и Roughness, в результате чего получился совсем другой материал, чем тот, с которого вы начали (рисунок 5.8).

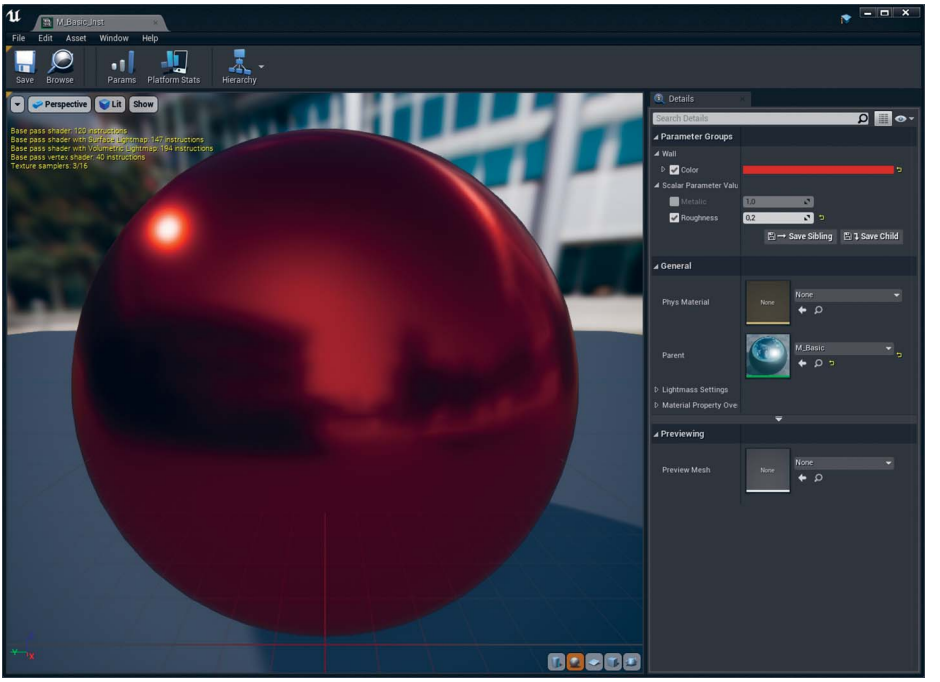


Рисунок 5.7 Material Instance, созданный от M_Basic

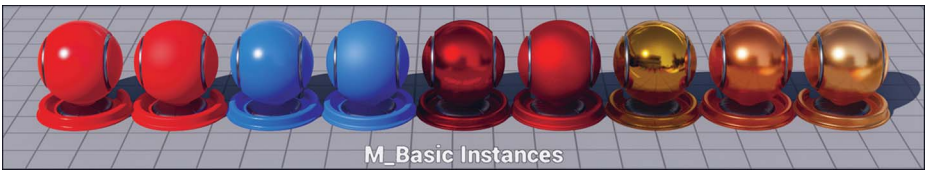


Рисунок 5.8 Массив Material Instance, созданных от M_Basic, показывающий много вариаций

Добавление карты шероховатостей

Простое добавление карты шероховатости к вашему материалу (рисунок 5.9) придает гораздо больше глубины и детализации поверхности, как видно на рисунке 5.10.

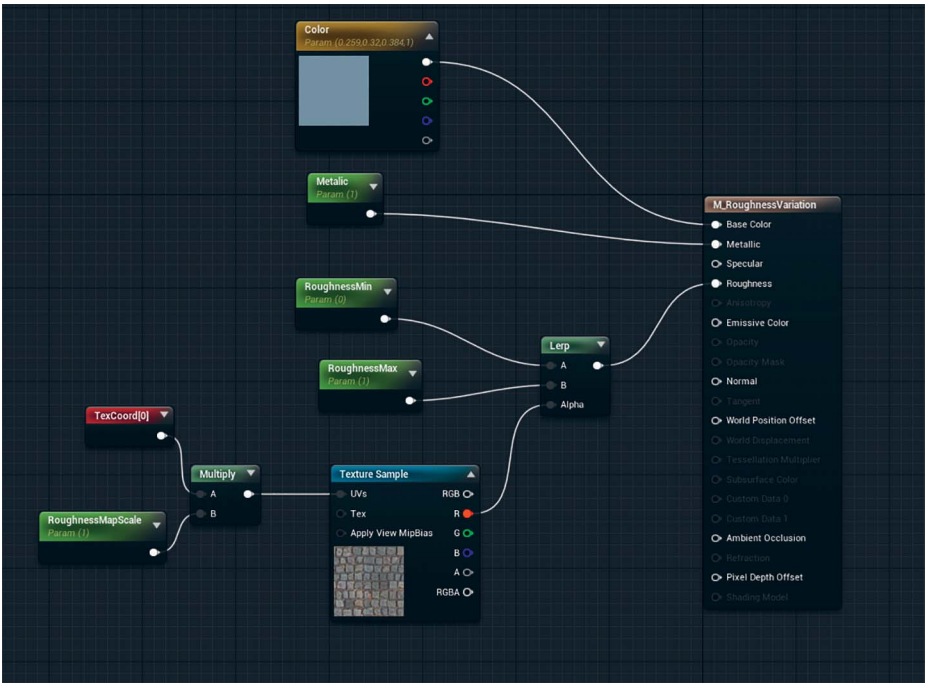


Рисунок 5.9 Добавление RoughnessMap Parameter к материалу для еще больших возможностей

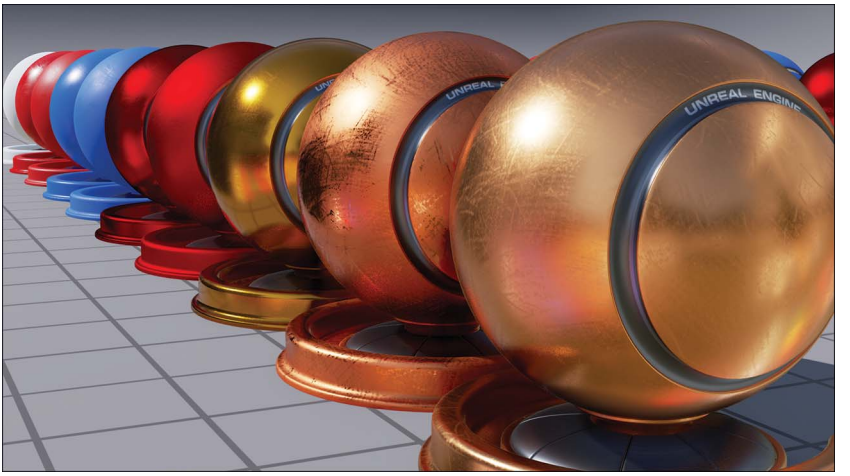


Рисунок 5.10 Material Instances с Roughness maps

Создание парамтеров текстур

В дополнение к ранее описанным методам добавления параметров можно также создать узел Texture Sample, перетащив текстуру в Material Editor из Content Browser. Когда вы делаете это, текстура еще не является параметром, и если она скомпилирована, то никак не может быть изменена в экземплярах. Чтобы преобразовать его в параметр, щелкните правой кнопкой мыши по узлу текстуры и выберите **Convert to Parameter**.

Editor даст его … уникальное имя; но вам, вероятно, следует присвоить ему более вдумчивое название. Теперь экземпляры этого материала могут определять собственную текстуру или изменять ее во время выполнения с помощью Blueprints.

Lerping It

Одной из наиболее важных особенностей Material Instance является возможность регулировать внешний вид материала на лету. Один из лучших способов сделать это — использовать узел Lerp (линейная интерполяция).

Добавление Min и Max Salar Parameters позволяет художникам динамически изменять значения карты шероховатости для каждого экземпляра.

Это один из моих любимых узлов для создания простых в настройке материалов. Вы даже можете легко инвертировать альфа-значение, установив вход A на 1, а B на 0. Lerp работает со всеми видами входных сигналов, включая цвета RGB, и является отличным способом смешивания между двумя значениями.

Заметка

Входы A и B в узле Lerp могут иметь практически любой тип данных (скалярный, векторный и т. д.); однако альфа-вход может быть только одним каналом оттенков серого. Возможно, вам придется использовать узел Component Mask, чтобы выбрать, какой канал вы отправляете на этот вход.

Масштабирование карты текстур

Чтобы масштабировать текстуры и создавать плитки в UE4, вы не масштабируете текстуру так, как, возможно, привыкли делать; скорее вы масштабируете UV-координаты, умножая их на скалярное значение.

Создание экземпляров материала

Как вы можете видеть, теперь у нас есть еще много параметров, перечисленных в нашем Material Instance Editor, включая возможность полностью переопределить текстуру, используемую для модуляции шероховатости (рисунок 5.11).

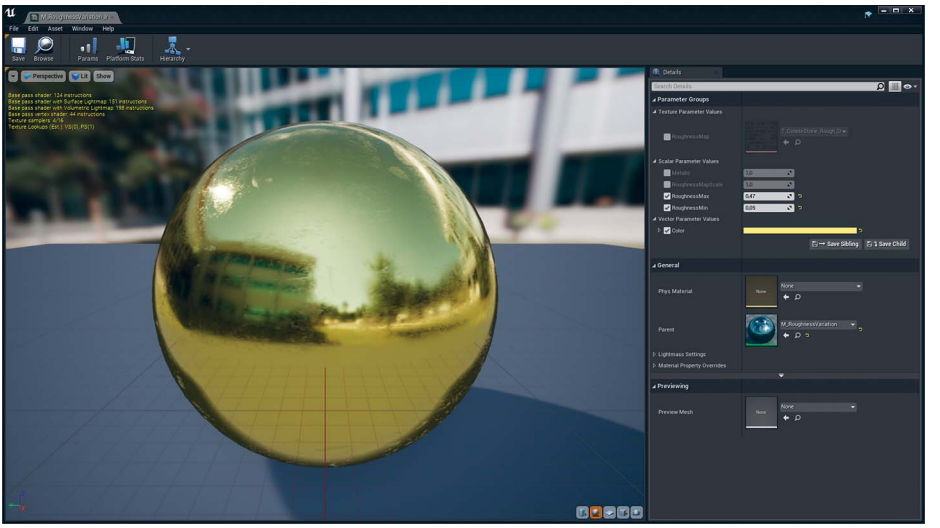


Рисунок 5.11 Material Instances Editor показывает группу Roughness Parameter

Добавление карты нормалей

Карта нормалей очень похожа на карту рельефа (bump map), но гораздо более эффективна и гибка. Карты нормалей определяют угол в пространстве, который помогает добавить детали поверхности к другим плоским полигонам (рисунок 5.12).

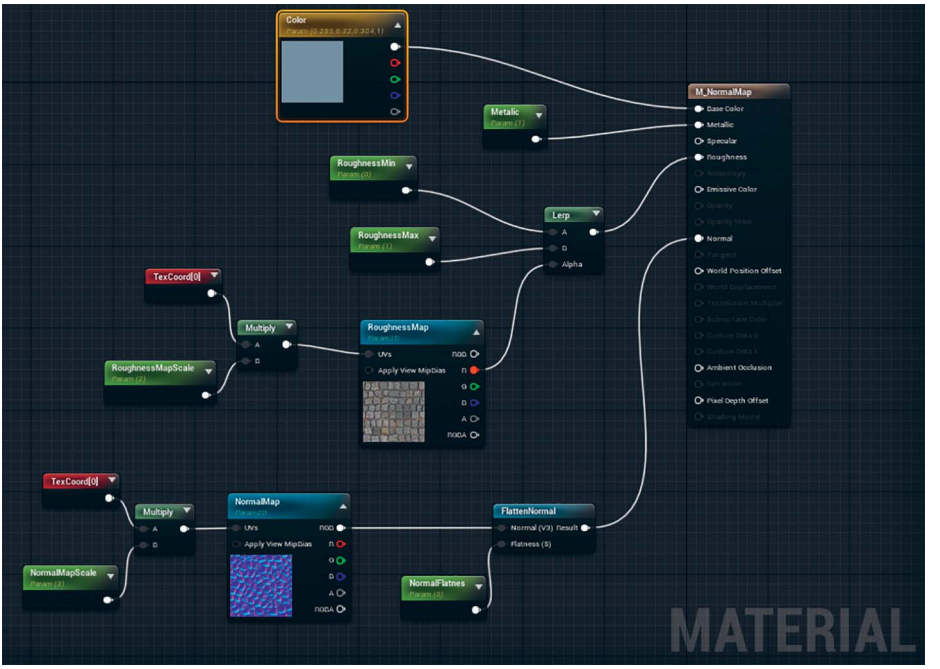


Рисунок 5.12 Добавление карты нормалей к материалу

Добавление карты нормалей к материалу добавляет еще один слой визуальной глубины и насыщенности (рисунок 5.13). Масштабирование нормалей и использование Material Funcation FlattenNormal позволяет художникам смягчить карту нормалей, если она слишком «твердая».

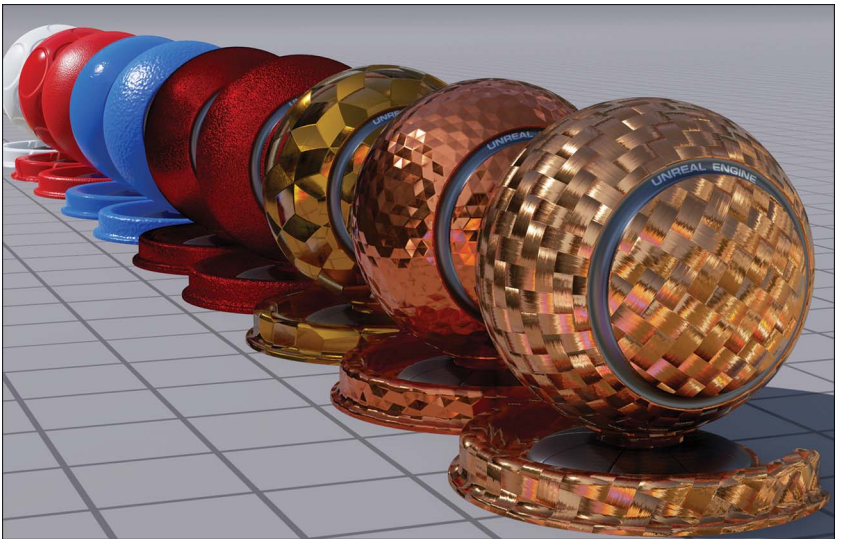


Рисунок 5.13 Экземпляры материала с картой нормалей, показывающего разные вариации применения этой карты

Понимание карты Base Color

Base Color на самом деле менее важен, чем вы, возможно, привыкли считать. Вы даже можете получить некоторые отличные материалы, не используя базовую цветовую текстуру. Конечно, вы наверняка хотите иметь цветовые вариации во многих ваших материалах, поэтому все равно очень важно, чтобы она у вас была.

Поскольку система освещения в UE4 настолько сильна, карты шероховатости и нормалей могут определить огромное количество поверхностной информации, которая раньше хранилась в Diffuse Texture.

Это оставляет Base Color только для передачи цветовой информации. На этой карте не должно быть никакой информации об освещении или затенении. Плоские цвета со средней и низкой детализацией работают лучше всего. Сохраните ваши детали для шероховатости и карт нормалей.

Base Color также определяет цвет Metallic, когда Metallic вход материала установлен в 1 (Metallic вход является скалярным или только в оттенках серого). Это позволяет вам получить металлическую зеркальную окраску для фантастических металлических эффектов (рисунок 5.14).

Рисунок 5.14 Экземпляры финального материала, показывающие различные полные наборы текстурных карт (Base Color, Normal Map и Roughness)

Заключение

Разработка материалов в UE4 — это фантастический опыт. Она сделана очень удобной для художников и проста в освоении. Однако она также невероятно мощная, что дает вам гибкость для создания практически любого вида материала с качеством, близким к качеству трассировки лучей.

Использование Material Instances и Functions добавляет организацию, вариативность и возможность повторного использования к уже мощной системе Material, а Blueprints и Material Parameters гарантируют, что ваши материалы будут такими же динамичными и интерактивными, как и остальная визуализация.

После того как вы начнете создавать авторские работы с помощью Material Editor UE4, интерактивных Viewports и PBR рендеринга, вам будет трудно использовать что-то другое.

ГЛАВА 6

BLUEPRINTS

UE4 представляет язык визуального программирования Blueprints, высвобождая мощь программирования, но не требуя от вас написания ни одной строки кода. Используя воображение, Blueprints и некоторые базовые концепции программирования, вы можете создавать что угодно — от простых взаимодействий, таких как выключатели света или двери, до целых систем моделирования трафика или многопользовательских VR-игр.

Введение в Blueprints

Хотя визуальные элементы UE4 привлекают наибольшее внимание, визуальный язык программирования **Blueprints** — это то, что сделало UE4 феноменом, которым он остается сегодня. Без написания кода любой человек может создавать собственные функции и добавлять возможности в приложения и игры.

Команда больше не должна нанимать программиста или жертвовать одним из своих, чтобы овладеть языком программирования. Почти каждый может научиться использовать Blueprints, чтобы вдохнуть жизнь в свои миры.

Blueprints — это реальный язык программирования, гораздо более надежный, чем большинство скриптовых систем, и он опирается на концепции программирования, такие как классы, наследование, циклы, переменные и функции. Если вы уже писали скрипты в своих 3D-приложениях или программировали раньше, вы будете чувствовать себя как дома, собирая визуальные скрипты. Если нет, то не бойтесь.

Blueprints упрощают программирование и устраняют многие барьеры, которые делают программирование недоступным для многих людей. Визуальный характер, дружелюбный интерфейс, доступ к исходному коду и обширная документация делают изучение Blueprints доступным и простым.

Вы в силах достичь почти всего, что способны себе представить, с помощью UE4, но правильное понимание основ может сделать даже самое простое интерактивное приложение блестящим и настроить вас на решение более серьезных проблем в будущем. Надежная производительность и хорошо продуманные интерфейсы отличают профессиональный и привлекательный опыт от того, что расстраивает или сбивает с толку ваших пользователей.

Как дизайнеры визуализации мы стремимся донести как можно больше информации до наших зрителей и игроков, причем наиболее четко и реалистично. С помощью Blueprints мы можем представлять данные нелинейными и не заданными заранее способами. Побуждение игроков экспериментировать, исследовать и открывать что-то новое самостоятельно — это то, что делает интерактивные визуализации такой мощной коммуникационной средой и дает нам силы как художникам и дизайнерам.

Эта глава предназначена как для тех, кто не имеет никакого опыта программирования, так и для тех, у кого такой опыт есть. Понимание того, как разрабатывается приложение, которое вы используете, и терминологии, используемой разработчиками и сообществом, всегда играет вам на руку, даже если вы знакомы с основными концепциями.

Объекты, класс и акторы

Вы встретите термины *объекты* (Object), *классы* (Class) и *акторы* (Actor), часто используемые в UE4 и вспомогательной документации. Они могут казаться синонимами или почти синонимами, но это не так. Хотя они тесно связаны, каждый из них отличается друг от друга, и знание этого различия имеет важное значение для обучения программированию в UE4.

Объекты

UE4 — это объектно-ориентированное приложение. Объектно-ориентированное программирование (ООП) — это способ представления программы в терминах реального мира и разбиения работы программы на логические единицы. Вместо одного огромного скрипта, который запускает все приложение, каждый элемент или **объект** содержит собственный код и возможности и взаимодействует с другими объектами в программе для обмена информацией или запуска событий.

Объекты могут быть любой частью приложения: кнопка или выпадающее меню — это отдельный тип объекта. Эффект частицы — это объект, который производит другие объекты.

Реальный пример — это ваза с яблоками. Компоненты — чаша и несколько отдельных яблок — являются полностью независимыми друг от друга (объектами), но взаимодействуют вместе, образуя целое.

Классы

Свойства каждого объекта определяются его **классом**, набором программных правил, функций и переменных, определяющих его поведение.

Объекты — это сущности, или **экземпляры**, класса. Каждый объект содержит собственные уникальные параметры или переменные (например, положение и вращение), которые он получил от своего **родительского** класса. Все объекты одного класса имеют те же инструкции и возможности, что и другие экземпляры класса объекта на сцене.

Пример с яблоками имеет два класса: Apple и Bowl. Существует один экземпляр класса Bowl (чаша) и несколько объектов, созданных из класса Apple (яблоко).

Один класс может иметь сотни объектов, созданных из него, каждый со своими уникальными данными и свойствами.

Акторы (Actors)

Акторы — это особый вид объекта. Эти объекты могут существовать в трехмерном мире ваших уровней и быть отрендерены. Распространенными примерами являются

Static Mesh Actors и Point Light Actors. Акторы могут иметь коллизии, материалы, анимацию, запрограммированную логику и симулировать физику.

В этом примере чаша и яблоки, которые в ней лежат, — все акторы. Каждый из них занимает определенное место в пространстве и имеет 3D-преобразования (положение, вращение и масштаб).

В UE4 свет, звуки, системы частиц и камеры — все это разные примеры акторов.

> **Заметка**
>
> Если вы можете видеть, размещать или взаимодействовать с чем-то в игровом мире, это актор. Свет, объемы, триггеры, чайники и ракеты — все это примеры.

Игрок

Большинство визуализаций линейны и неизменны. Они обычно доставляются в виде изображений, анимаций или других линейных форматов, которые практически никогда не меняются каким-либо существенным образом. Каждый человек, который смотрит на них **(зритель)**, будет иметь тот же опыт и, по большей части, видеть те же вещи, что и *любой другой*, кто смотрит на них.

Интерактивная визуализация нелинейна. Каждый **игрок** (Player, человек за контроллерами, обеспечивающими ввод информации) имеет свободу исследовать мир так, как ему захочется. Вы, как дизайнер, можете ограничить и контролировать то, что игрок может сделать; но в конечном счете игрок будет определять, как станет развиваться история. Игрок является центром вселенной UE4. Все, что делает UE4, он делает на благо игроков и их опыта.

Контроллер игрока

Player Controller или **PC** — это класс, ответственный за изменение мира на основе входных данных от игрока.

Для каждого игрока в вашем приложении будет создан один PC. PC автоматически создается миром, когда игрок присоединяется к игре, и уничтожается, когда он выходит.

Player Controller не имеет формы и не может быть виден (это объект в мире, а не актор). Вы можете думать о PC как о виртуальном представлении игрока в мире — его духе или призраке, если хотите.

Один PC может одновременно владеть одним Pawn (павн). Когда PC «вселятся» во что-то, он имеет прямой доступ к этому объекту и может передавать ему команды или иным образом менять его.

Это позволяет одному игроку радикально изменить то, как он играет в игру от одного момента к другому, «вселяясь» в различные павны с различными возможностями.

Обработка входных данных

PC — это первое место, куда система ввода пересылает входные данные после обработки. PC Blueprints имеют доступ к огромному массиву входных данных от мыши и клавиатуры до VR Motion Controller и сенсорных систем ввода для мобильных устройств.

Хотя объекты Pawn и другие акторы также могут сами получать входные события, PC часто используется для передачи этих входных данных игрока объектам Pawn и другим системам, которым нужны эти входные данные, чтоб уменьшить вероятность конфликта между различными классами.

Хотя существуют веские причины для того, чтобы другие объекты обрабатывали некоторые входные данные игрока (например, кнопка в пользовательском интерфейсе, показывающая, что она нажата), я стараюсь сохранить почти всю обработку моих входных данных в PC.

Данные игрока

Одна из самых замечательных вещей в PC — это то, что он всегда есть. Если есть игрок, то есть и PC. Это означает, что каждый раз, когда системе нужно взаимодействовать с одним надежным объектом, приходит PC.

Все объекты Actor в Blueprints имеют быстрый и легкий доступ к PC с помощью простых функций Get. Это делает его отличным местом для хранения данных об игроке, таких как имя, цвет команды и т. д. Более сложные игры могут использовать специализированные классы (классы инвентаря и т. д.), но они обычно создаются и управляются с помощью Player Controller.

Вращение

Одна из самых важных функций Player Controller — следить за вращением игрока. У PC нет 3D-позиции — нет камеры или другого 3D-представления. Однако у него есть переменная Controller Rotation.

Отслеживание вращения игрока отдельно от позиции имеет несколько преимуществ, особенно когда речь заходит о представлении взгляда игрока и представлении последовательной и плавной точки зрения игрока. Наличие одного типа вращения также легко запрограммировать, потому что есть одна переменная, которую можно отслеживать и изменять по мере необходимости.

Интерфейс мыши

Класс Player Controller имеет несколько важных настроек для визуализации. Наиболее важной является возможность отображения курсора мыши.

По умолчанию UE4 и большинство видеоигр не показывают курсор мыши. Однако многие проекты визуализации полагаются на надежные пользовательские интерфейсы, и наличие курсора для них очень важно.

Вы также можете включить события наведения (mouse over) и щелчка мыши (mouse-click) для ваших сцен, позволяя вам напрямую взаимодействовать с 3D-акторами в вашем мире. Это может повлиять на производительность и на то, как игрок взаимодействует с HUD и другими элементами, поэтому включайте события мыши только в том случае, если они необходимы для вашего проекта.

Другие контроллеры

Вы можете создавать или добавлять другие Controllers в свои приложения. Наиболее распространенным является AI Controller (искусственный интеллект). Эти контроллеры действуют точно так же, как Player Controller, но вместо того чтобы использовать входные данные от игрока, они полагаются на запрограммированные правила и последовательности для имитации человеческого поведения и входных данных.

Pawns (павны)

Объекты Pawns — это акторы в мире, в которые можно вселяться[11] контроллером (игрока или ИИ). Когда, контроллер посылает входные и другие команды Pawn, инструктируя ее двигаться или запускать ракеты по врагам.

Объектом типа Pawn могут быть любые персонажи, животные или транспортные средства, которыми можно управлять. Контроллер дает объекту Pawn простые команды, такие как Move forward (двигаться вперед), и тот интерпретирует их соответствующим образом, чтобы имитировать тип движения и взаимодействия, которые вы хотите увидеть.

Используя PC и Pawn вместе, вы можете гарантировать, что игрок имеет как последовательный набор возможностей, которыми управляет PC, так и конкретную интерпретацию этих входных данных каждым Pawn-объектом. Это позволяет одному PC, а следовательно и игроку, владеть любым видом объекта типа Pawn, который только можно себе представить.

[11] Вселение — это термин, используемый, когда контролер берет под контроль актора. Вы можете думать о контролере как о призраке, который способен прыгать от актора к актору или даже существовать без актора вовсе.

Заметка

Вы можете спавнить и вселяться в новые Pawn-объекты в любое время. Это отличный способ создать новый режим просмотра или метод управления для игры. Например, вы можете переключаться между ходячим объектом и летающим беспилотным объектом, каждый из которых имеет совершенно разные возможности, внешний вид, эффекты, звуки и даже элементы пользовательского интерфейса.

Мир

World (мир) — это место, где существуют все объекты, акторы и данные в приложении UE4. Когда вы загружаете **Level** (уровень) (также называемый **картой**), создается новый мир с акторами и объектами, которые были помещены для спавна в нем.

Все, что существует в мире (объекты), может получить доступ и быть доступно для любого другого объекта в мире. Объекты также могут спавнить и уничтожать другие объекты и акторов.

Мир, как и почти все остальное в UE4, может быть запрограммирован с помощью Blueprints, а также может спавнить, уничтожать и изменять акторов и объекты в нем.

Уровни

Каждый уровень (Level) в UE4 похож на автономное мини-приложение. Каждый уровень может иметь совершенно разные правила, геометрию и т. д. На самом деле каждый уровень может быть совершенно другой игрой. Возможно, вы видели это в играх, в которые играли, когда один уровень — это вождение, а следующий — пешая погоня с очень разными элементами управления, моделями и геймплеем.

Заметка

Уровни часто называют картами. Эти два термина по большей части взаимозаменяемы.

Уровень может иметь несколько подуровней, которые он может стримить. Эта система является отличным способом загрузки определенного контента в определенное время в вашем приложении. Вы можете легко загружать, выгружать и устанавливать видимость целых уровней из Blueprints. Это отличный способ управления информацией, как в редакторе, так и во время выполнения.

Компонеты

Акторы Blueprints могут иметь любое количество **компонентов** (Components): специальные классы, которые создаются вместе с актором и могут взаимодействовать непосредственно с ним. Распространенными примерами являются статические и скелетные меши, свет и системы частиц.

Компоненты могут быть унаследованы от родителя ребенком. Они могут быть созданы во время выполнения или в Contraction Script. Они также могут быть определены в Blueprints Editor как Component Parameters.

Компоненты — это отличный способ повторного использования кода. Например, можно создать компонент, который перемещается с определенной скоростью по траектории. Вы можете прикрепить этот компонент к любому актору в вашем проекте, чтобы дать ему такую возможность.

Переменные и их типы

Все объекты могут хранить данные с помощью **переменных** (Variables), также называемых **свойствами**. Переменные могут содержать такие значения, как числа и строки, или даже ссылки на другие объекты, классы или акторы.

Переменные в Blueprints имеют строгий тип, что означает, что каждая переменная относится к определенному классу и это невозможно изменить. Если вы занимались чистым написанием скриптов, это может быть сложно для вас, но это позволяет проводить более сложную компиляцию, проверку ошибок и обработку переменных. Одна из самых крутых вещей в нем заключается в том, что каждый тип может иметь собственные функции и переменные!

Распространенными типами переменных являются следующие.

1. **Boolean:** простая переменная true или false.
2. **Float or Scalar:** числа с десятичными дробями, такие как 0.984, 4356.234 или –34.2 Integer: числа без десятичных дробей, такие как 23 или –2354.
3. **String:** буквенно-цифровые последовательность символов, например Hello World.
4. **Text:** текст похож на строку, но в первую очередь предназначен для локализации.
5. **Vector:** массив из трех значений с плавающей точкой, представляющих значения X, Y и Z, часто используемых для записи положения, масштаба или направления в 3D-пространстве.
6. **Rotator:** массив из трех значений с плавающей точкой, представляющих значения вращения Roll, Pitch и Yaw в 3D-пространстве.

7. **Transform:** объединяет данные о положении (Vector), вращении (Rotator) и масштабе (Vector).

8. **Object:** ссылка на объект или актор в мире, включая статические меши, камеры, источники света и Player Controllers.

Tick

Каждый кадр, отрендеренный в UE4, следует точному порядку операций. Этот набор операций часто называют игровым циклом или основным циклом. В UE4 это называется **Tick** (тиком).

На каждом тике входные данные от игрока считываются и направляются в систему сценариев, где они могут инициировать события и функции. Персонажи могут перемещаться, орудия — стрелять, ну или шкафы могут менять материалы. Различные системы в игре могут затем реагировать на эти данные: физика, освещение и искусственный интеллект собирают информацию об окружающем мире и выполняют собственные действия, созданные при помощи программирования. После того как все входные данные и логика обработаны, геометрия сцены обновляется и может быть визуализирована и представлена игроку.

Delta Seconds

Время, необходимое для рендера одного кадра или тика, называется **Delta Seconds** или **Delta Time**. Игра, работающая со скоростью 60 кадров в секунду, имеет среднее Delta Time 0,0167 секунды.

Delta Time важно, потому что каждый тик может занять разное количество времени для рендеринга от последнего кадра (сцены могут резко меняться от одного кадра к следующему) или от одного компьютера к другому.

Отслеживание этого значения и его применение очень важно при выполнении любой задачи, которая имеет место во времени и пространстве.

Например, если вы хотите вращать актора с постоянной скоростью, вы можете просто добавлять значение поворота на каждом тике. Компьютер, работающий со скоростью 30 кадров в секунду, будет вращать актора на 360 градусов в секунду, если его повернуть на 12 градусов за тик, в то время как компьютер, работающий со скоростью 120 кадров в секунду, вызовет вращение со скоростью 1440 градусов в секунду.

Именно здесь появляется Delta Time: вы можете просто умножить предполагаемое преобразование на Delta Time, чтобы убедиться, что ваше движение основано на времени, а не на частоте кадров.

В этом примере умножьте ваши предполагаемые градусы в секунду на Delta Seconds. На ПК с 30 кадрами в секунду это выглядит как 360*0,033 градуса на тик (около 12 градусов на тик), в то время как при 120 кадрах в секунду это 360*0,0833, или 3 градуса на тик. Даже если частота кадров сильно колеблется, актор будет вращаться с той же скоростью.

Наследование классов

Каждый класс может служить шаблоном **(родителем)** для другого класса **(дочернего)**. Ребенок **наследует** все функциональные возможности и структуру своего родительского класса. Любые изменения в родительском классе также наследуются дочерними классами.

Возвращаясь к предыдущему примеру с яблоками, класс Apple может иметь несколько дочерних классов: Red Delicious, Granny Smith и так далее. Каждый из них наследует функциональность и описание яблока (хрустящее, сладкое, круглое), но переопределяет определенные свойства, такие как цвет, размер и параметры вкуса.

Дочерние классы могут переопределять функции родительского класса, добавляя или уточняя функциональность. Например, и Point Light Actor, и Spot Light Actor являются «детьми» класса Light Actor. Оба класса имеют много одинаковых свойств, таких как яркость, цвет и затухание.

Вместо того чтобы оба класса определяли каждое из этих свойств, родительский класс (Light Actor) определяет их, а дочерние классы их наследуют. Каждый класс может затем расширить родительские функции и свои возможности.

Класс Apple также имеет родительский класс: Fruit. Fruit класс включает в себя яблоки, апельсины, виноград, груши и многое другое. Хотя каждый из них отличается и вы не можете сделать яблоко из винограда, они наследуют много общих свойств и методов от класса Fruit.

UE4 использует концепцию **наследования** почти во всех аспектах своего дизайна (Material Instances отличный пример). Понимание этого необходимо для того, чтобы иметь возможность разрабатывать более сложные системы и приложения. Это также отличный способ избежать дублирования уже проделанной работы.

Spawning и Destroying (спавн и уничтожение)

Каждый объект в вашем игровом мире входит в этот мир с помощью **спавна**. Когда объект порождается, он создается из своего определенного класса и появляется в игровом мире, что позволяет ему «видеть» все остальное в мире.

Объекты могут быть созданы во время выполнения с помощью Blueprints. Практически любой другой объект в мире может породить новый объект.

Вы можете спавнить акторов в свой игровой мир в Editor до выполнения, перетаскивая их из Content Browser в игровой мир.

Вы также можете уничтожить акторов, удалив их из мира. Это может вызвать проблему в Blueprints, которые ссылаются на уничтоженный объект, поэтому будьте осторожны и проверяйте свои ссылки. Функции в Blueprints, возвращающие none, могут привести к нестабильности или сделать проект некомпилирующимся.

Спавн — это дорогостоящая операция, и спавн большого количества акторов может замедлить работу вашего приложения. Если вы обнаружите, что спавните и уничтожаете много акторов одного и того же класса, вам стоит потратить время на переработку этих акторов, а не на уничтожение и создание новых.

Construction Script

Акторы делают что-то особенное, когда они спавнятся либо в Editor, либо во время выполнения: они запускают Construction Script.

Эта функция очень мощная, потому что она позволяет Blueprint Actor выполнять действия из скрипта до запуска игры в Editor (если он помещается на уровень) или до его появления (если он спавнится во время выполнения). Это может включать изменение самого себя, создание новых мешей и других компонентов, а также выполнение другой обработки.

Вы можете определить переменные в вашем акторе в качестве изменяемых параметров. Вы можете редактировать выставленные параметры на размещенных акторах, и Construction Script тут же перезапустится.

Вы можете создавать сложные Construction Script, которые способны создавать целые структуры или системы на основе параметров. В качестве примера можно привести создание актора, который размещает меши вдоль сплайна (фонарные столбы, перила, дороги) или рассеивает объекты по мешу (трава, листва и другие объекты), или систему времени суток, которая обновляет свои различные компоненты, основываясь исключительно на положении солнца. Возможности безграничны.

Begin Play Event

После запуска Construction Script объект инициализируется и добавляется в окружающий мир. Как только он доступен в мире и может видеть окружающий мир, запускается событие Begin Play.

Именно здесь вы можете настроить любое поведение или взаимодействие, которое должно быть установлено до того, как будет показан первый кадр или

непосредственно перед тем, как актор станет виден на экране. Акторы на этой стадии не будут отрендерены, но получат доступ к миру и содержащимся в нем объектам.

Begin Play начинается только после того, как все акторы и объекты созданы для этого тика, поэтому можно безопасно получить доступ к другим объектам, которые создавались в этом фрейме (в отличие от Construction Script, который выполняется до инициализации объекта).

Взаимодействие между Blueprints

Очень важно, чтобы ваши Blueprints могли коммуницировать друг с другом и с миром. UE4 предоставляет несколько способов заставить ваши объекты, акторы и миры, которые они населяют, обмениваться информацией и создавать богатые взаимодействия.

Direct Blueprint Communication

Когда вы хотите, чтобы один Blueprint напрямую связывался с другим Blueprint на индивидуальной основе, вы чаще всего будете использовать Direct Blueprint Communication. Этот метод требует, чтобы вы явно определили, на какой объект в мире ссылаются.

Например, игрок щелкает по элементу Light Switch Blueprint Actor, у которого есть определенный Light Bulb Blueprint Actor на уровне, назначенный переменной. Затем выключатель сообщает лампочке, чтобы она выключилась и включилась непосредственно. Эффект подействовал только на эту лампочку.

Event Dispatcher

Event Dispatchers позволяют вам иметь несколько Blueprints, прослушивающих одно событие, и каждый из них реагирует индивидуально, когда оно вызывается.

В примере с Light Switch Light Bulbs могут прослушивать событие OnSwitch у Switch Blueprint и соответствующим образом изменять себя, когда это событие инициируется взаимодействием игрока с выключателем света.

Blueprint Interfaces

Blueprint Interface предназначен для тех случаев, когда у вас есть события, которые являются общими для разных классов, например щелчок мышью или указание на контроллере, но каждый класс должен делать что-то отличающееся или вообще ничего.

Интерфейсы — это просто списки ожидаемых событий. Если в Blueprint реализован интерфейс, можно выбрать любой из перечисленных методов, все или ни одного.

Вы должны использовать интерфейсы, когда у вас есть функциональные возможности, которые *похожи* в нескольких Blueprints, но должны выполняться *по-разному* в каждом из них.

Интерфейсы[12] являются наименее часто используемым методом связи между Blueprints и обычно не рекомендуются для большинства вариантов использования. Вместо этого используйте Event Dispatchers.

Blueprint Casting (приведение типов)

Приведение типов использует специальные узлы в Graph Editor. Эти узлы принимают входные данные одного класса, а затем пытаются получить доступ к этому узлу как к определенному классу. Если попытка будет успешной, вы сможете получить прямой доступ к переменным, событиям и функциям целевого объекта.

Это позволяет запрашивать общий набор акторов и классов. И если они принадлежат к определенному классу, они успешно будут приведены к этому классу, предоставляя вам доступ ко всем переменным и методам, доступным в этом классе.

Например, Light Bulb Blueprint может расширить класс актора, на котором он основан, чтобы иметь функцию ToggleLight. Switch Blueprint может запросить каждого актора на сцене и попытаться привести их к классу Light Bulb. В случае успеха он может напрямую запустить функцию ToggleLight.

Вы также можете использовать приведение типов для запуска событий и изменения свойств для всех акторов определенных классов на уровне. Используя узел Get All Actors of Class, вы можете получить доступ к массиву акторов, которые легко изменить.

Компиляция скрипта

Перед запуском Blueprint необходимо его скомпилировать. Компиляция преобразует графы узлов в код, который выполняется в Editor и во время работы в игре.

Компиляция также проверяет ваш код на наличие ошибок, пытаясь найти ошибки еще до запуска игры. Она не улавливает все, но обычно находит самые опасные типы ошибок до того, как может произойти сбой.

Чтобы скомпилировать Blueprint, просто нажмите кнопку Compile на toolbar. Любые найденные ошибки появляются в Compiler Results и обычно включают удобную ссылку, которая ведет вас непосредственно к проблеме.

[12] Интерфейсы — неотъемлемая часть ООП, которую поддерживает движок, и почти ни один средний и крупный проект не обходится без них. Также они очень часто применяются в разных системах взаимодействий в играх и других проектах на UE4. — *Прим. пер.*

Заключение

Термины и понятия, представленные в этой главе, широко распространены в интерфейсе UE4, документации и сообществе. Вы должны лучше понимать, как приложение UE4 работает «под капотом», и должны быть в состоянии начать изучать Blueprints и создавать приложения в UE4 увереннее.

В следующих главах вы будете часто сталкиваться с этими концепциями, даже в системах, не относящихся к Blueprints. Такие понятия, как наследование, переменные и свойства, присутствуют во всем UE4 и определяют почти каждый аспект его дизайна.

ЧАСТЬ 2

ВАШ ПЕРВЫЙ ПРОЕКТ НА UE4

7 Настройка проекта

8 Заполнение мира

9 Создание интерактивности с Blueprints

10 Упаковка и распространение

ГЛАВА 7

НАСТРОЙКА ПРОЕКТА

Ваш первый проект в UE4 фокусируется на акклиматизации к UE4 Editor, Content Browser и Viewport уровня. Эта глава посвящена определению объема проекта и созданию всего необходимого для начала создания вашего первого интерактивного мира в UE4.

Объем проекта

Прежде чем приступить к работе над любым интерактивным приложением, важно сначала определить цели проекта и записать их. Солидный объем работ и четко определенные минимальные требования гарантируют, что вы будете разрабатывать свои проекты целенаправленно и методично.

Первый свой проект вы будете делать простым. UE4 включает в себя множество примеров контента, который легко доступен и является отличным способом начать изучение основ UE4. Вы будете использовать этот контент, чтобы построить простой уровень, заполнить его и настроить павн, который может плавно проходить через уровень, сталкиваясь со стенами и полами, давая захватывающее ощущение прочности.

Наконец, вы протестируете приложение с помощью Play-In-Editor (PIE). После того как вас все устроит, вы соберете и подготовите свое творение для распространения в качестве автономного приложения.

К концу этой главы вы будете иметь хорошее представление о том, как разработать простое приложение UE4 с нуля. Давайте разберем это и составим список.

1. Разработайте приложение, которое позволит игроку проходить через виртуальную среду.
2. Используйте Starter Content для создания простой среды со светом, дверью и переключателем для переключения света.
3. Используйте Dynamic Lighting.
4. Сделайте вид от первого лица и ходите в медленном и устойчивом темпе.
5. Убедитесь, что высота вида соответствует архитектурным визуализациям.
6. Дайте игроку возможность сталкиваться со стенами и полами, придавая симуляции солидности.

Создание нового проекта из Launcher

Если вы еще не создали учетную запись на сайте unrealengine.com, не скачали и не установили Epic Games Launcher, а также не установили движок версии 4.25 или более поздней (эта книга использует 4.25 во всех примерах), то нужно сделать это перед началом работы.

Самый простой способ создать новый проект UE4 — это использовать окно New Project. Вы можете получить доступ к нему, запустив движок из Launcher (рисунок 7.1).

После запуска **Unreal Project Browser** выполните следующие действия.

1. В разделе **New Project Categories** выберите Games.

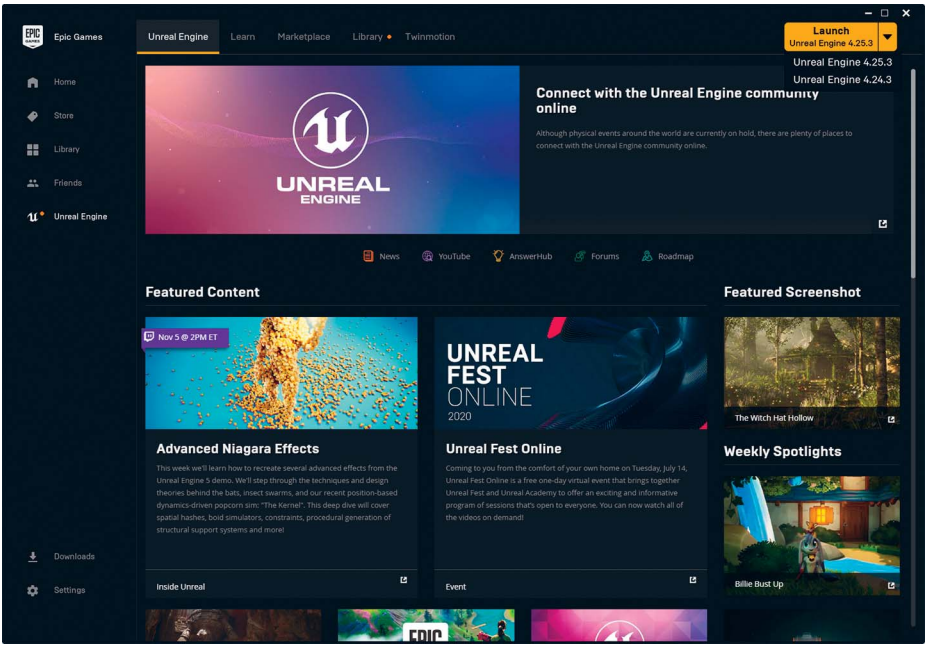


Рисунок 7.1 Запуск движка из Launcher, используя кнопку Launch

Здесь вы можете выбрать один из широкого спектра стартовых **шаблонов**[13] (в других разделах помимо Games содержатся другие шаблоны, которые в книге не рассматриваются). Каждый из них хорошо написан и может стать отправной точкой для многих проектов. Они также являются отличным и простым способом играть с различными игровыми режимами и стилями (от первого лица, полет, боковой скроллер и так далее) в UE4.

Вы не будете использовать ни один из них для вашего проекта. Хотя они являются отличными отправными точками, все они ориентированы на игру и вводят вещи, которые вам придется удалить, и другие вещи, которые вы должны уметь настраивать самостоятельно.

2. Выберите шаблон **Blank**. Этот шаблон проекта не содержит никакого контента или кода вообще — другими словами, это совершенно чистый лист.

[13] Не все из них соблюдают принципы про Player Controller, так как шаблоны — это очень маленькие проекты. При их использовании рекомендую вынести логику обработки входных данных в PC. — *Прим. пер.*

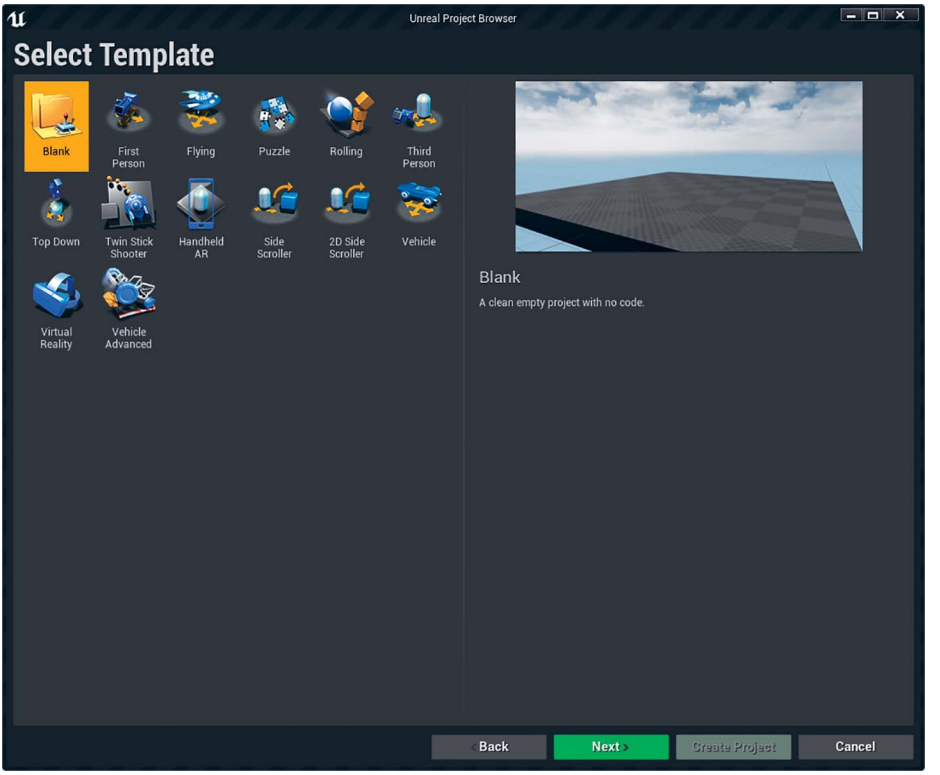


Рисунок 7.2 Панель выбора шаблона в Unreal Project Browser показывает все доступные шаблоны

3. Выберите тип проекта Blueprint и **With StarterContent**, что добавит в проект несколько простых готовых ассетов, они помогут вам быстро построить свой уровень. Остальные параметры оставьте без изменений (рисунок 7.3).

4. Выберите место для хранения вашего проекта и дайте ему имя. Я храню свои проекты вне своей пользовательской папки; однако только вам решать, где хранить проект. Просто не забудьте выбрать быстрый локальный жесткий диск или SSD с большим количеством свободного места (рисунок 7.3).

5. Нажмите зеленую кнопку **Create Project**. Движок создает новую папку и структуру папок, необходимую для запуска вашего проекта.

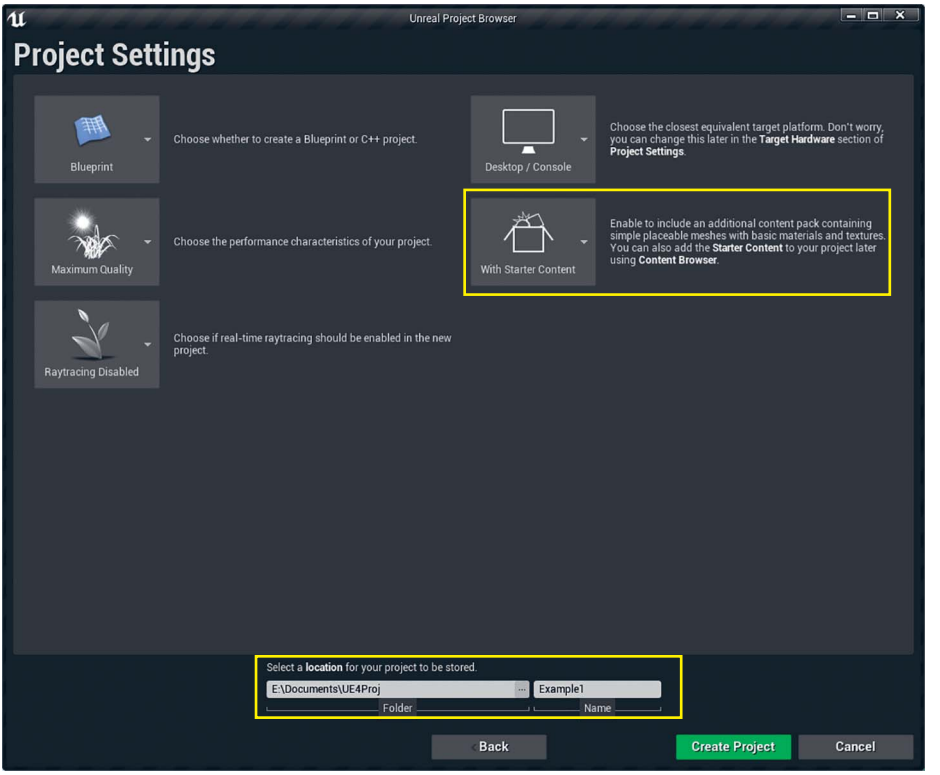


Рисунок 7.3 Панель настроек проекта в Unreal Project Browser

После создания файлов проекта запускается новый экземпляр UE4 Editor (рисунок 7.4), отображающий имя вашего нового проекта.

Рисунок 7.4 Первая загрузка нового проекта

После короткой последовательности загрузки в Editor появляется уровень (рисунок 7.5).

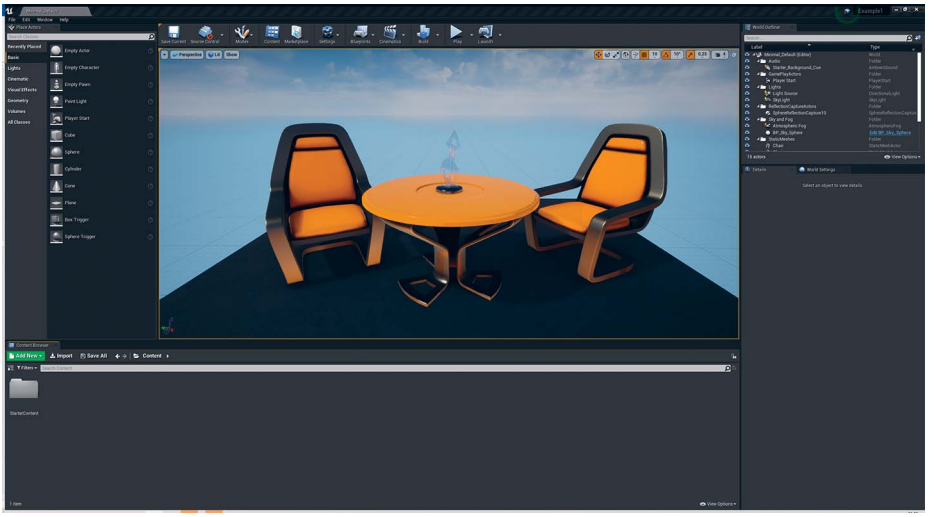


Рисунок 7.5 Открытый Unreal Engine 4 Editor показывает стартовый уровень и папку Starter Content

Когда вы включите Starter Content, уровень по умолчанию вашего проекта установится на простую сцену с размещенными примерами ассетов.

Заключение

Ваш новый проект готов к работе. Вы узнали, как создать новый проект из шаблона и настроить исходную файловую структуру в папке Content. Именно так вы, скорее всего, будете начинать большинство своих новых проектов, так что это рабочий процесс, с которым стоит познакомиться.

Различные шаблоны позволяют легко опробовать готовые настройки, и я призываю вас изучить их, потому что они являются отличным учебным ресурсом, а также отличным способом начать работу над конкретным типом проекта.

ГЛАВА 8

ЗАПОЛНЕНИЕ МИРА

Получение контента на вашем уровне является простым и удобным для художника. UE4 Editor предлагает множество инструментов и функций, которые помогут вам разместить, поменять и организовать ваши ассеты в 3D-пространстве. В этой главе вы будете использовать уже созданные ассеты Starter Content, включенные в ваш проект, чтобы построить простой уровень, на котором можно ходить.

Создание и сохранение нового пустого уровня

На мой взгляд, начать с нового уровня и построить уровень с нуля — это лучший способ получить желаемые результаты. Цели визуализации отличаются от целей большинства игр, и большинство доступных шаблонов просто не подходят для типов взаимодействий, необходимых при создании интерактивных визуализаций.

В меню File выберите пункт **New Level** и выберите опцию **Empty Level**.

Сохраните свой уровень, выбрав пункт **Save Current** в меню File или нажав кнопку **Save Current** на панели инструментов. Когда появится диалоговое окно **Save Level As** (рисунок 8.1), назовите уровень **MyFirstMap** и нажмите кнопку **Save**, чтобы сохранить карту на диск.

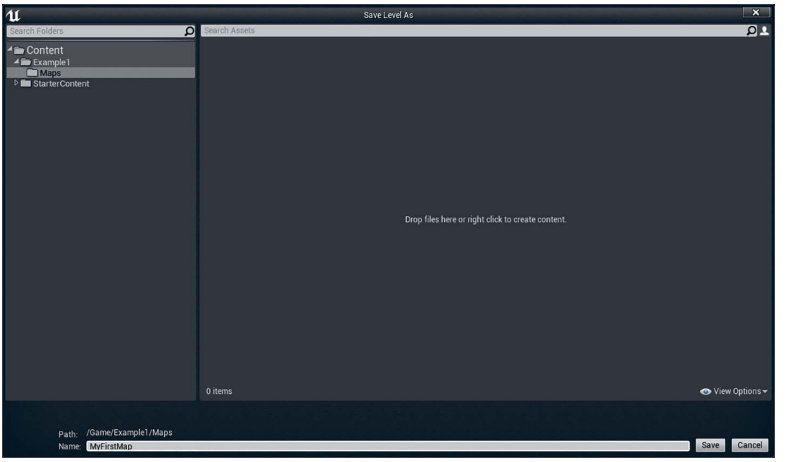


Рисунок 8.1 Диалог сохранения уровня

Как видно из этого примера, я создал папку в корне каталога Content под названием **Example1**. Вы можете создать новую папку, щелкнув правой кнопкой мыши в различных логических местах Content Browser или используя кнопку Add New в Content Browser. Помните, что вы можете и должны выполнять все функции управления файлами из Content Browser.

Если вы допустили ошибку и нужно переименовать карту или папку, вы можете воспользоваться контекстным меню или выбрать ассет или папку и нажать клавишу F2 на клавиатуре.

Создание каталога для конкретного проекта в папке Content — это обычная практика в UE4, поскольку она гарантирует, что все содержимое, созданное для этого проекта, будет автономно содержаться в его собственном каталоге, в то время как содержимое

от третьих лиц или других проектов может быть объединено в ваш проект без опасений, что они будут конфликтовать.

Помните, что уровни и карты — это одно и то же, и термины используются взаимозаменяемо для представления файлов UMAP.

Расстановка и модификация ассетов

Наиболее распространенный метод расстановки ассетов из Content Browser на уровень — простое перетаскивание ассетов из Content Browser во Viewport (рисунок 8.2).

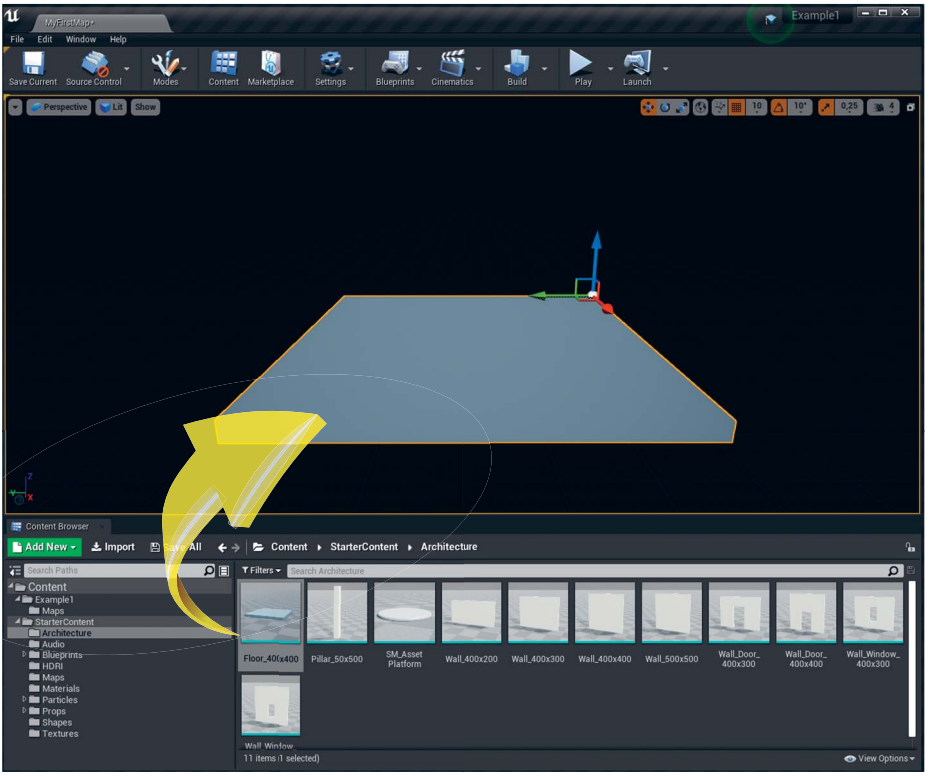


Рисунок 8.2 Перенос ассета из Content Browser на уровень

Перемещение, масштабирование и вращение

После размещения акторов вы можете легко перемещать, масштабировать и вращать их с помощью знакомых Gizmo. Вы можете легко переключаться между режимами перемещения, поворота и масштабирования с помощью пробела или значков в верхней части Viewport.

Заметка

Вы также можете использовать клавиши W, E и R для переключения между перемещением, масштабированием и вращением или нажать пробел для циклического переключения между каждым из режимов.

Использование панели Details

На панели Details отображаются все свойства для каждого выбранного актора (рисунок 8.3). Здесь вы можете непосредственно задать свойства расположения, масштаба и поворота для акторов, а также специфические для класса настройки, такие как Lightmap Resolution, параметры тени и переопределения материала.

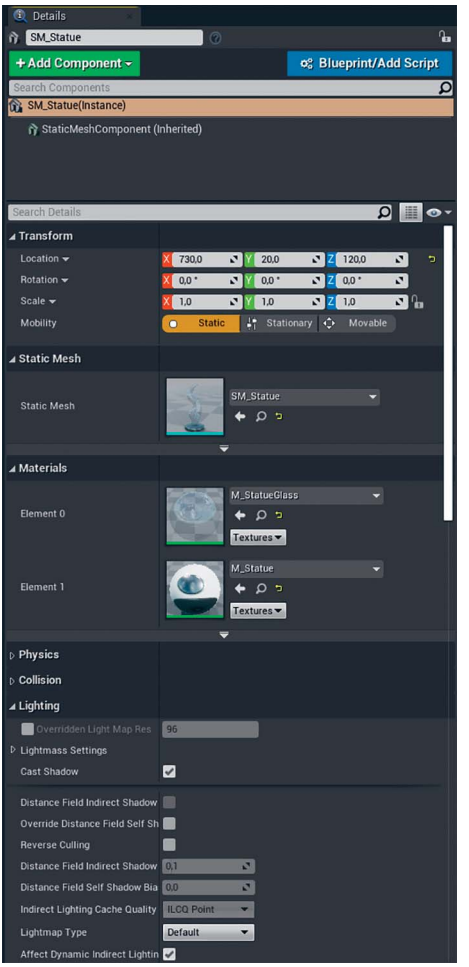


Рисунок 8.3 Панель Details для выбранного Static Mesh Actor

Snapping

Ассеты в Starter Content построены с размером меша 100 units. По этой причине включение Snap в Viewport и установка его на 100 units является хорошей идеей. Таким образом, когда вы перемещаете акторов, они привязываются к сетке единиц измерения 100 x 100 x 100.

Вы также можете установить привязку для Rotation и Scale в настройках Viewport с помощью кнопок-переключателей, расположенных в правом верхнем углу каждого видового экрана (рисунок 8.4). Каждый из них отображает интервал привязки справа. Вы можете настроить этот параметр, нажав на интервал.

Рисунок 8.4 Настройки Snap в Viewport

Дублирование

У вас есть несколько способов дублировать акторов. Выполнение копирования-вставки или использование команды дублирования в меню на правый клик и Edit — хорошие варианты.

Я часто использую сочетание клавиш Alt+D для мгновенного создания копий выбранного объекта. Удерживайте Shift, пока вы дублируете актора, чтобы камера следовала за актором при его перемещении. Вы также можете при помощи Alt + перетащить сразу несколько акторов. Для этого вам нужно ухватиться за трансформирующую gizmo во Viewport, а не просто в любом месте модели. Вы также можете клонировать акторов, подобных этому, с помощью вращающихся и масштабируемых gizmo.

Добавление акторов из Content Browser

Content Browser предназначен для размещения сгенерированного или импортированного контента на уровне, но вы будете размещать в своих игровых мирах много вещей, которые недоступны в Content Browser.

Вы размещаете такие классы, как Lights, перетаскивая их из Class Browser с помощью Place Mode. Оба метода создают новый актор уровня, который создает экземпляр выбранного класса.

Окно Place Actors расположено в левом верхнем углу редактора по умолчанию. Вы также можете открыть Class Browser, перейдя в пункт Window в верхнем меню и выбрав **Place Actors**.

Да будет свет

Прежде чем вы начнете всерьез расставлять акторов, нужно немного света на сцене. В отличие от Max или Maya, здесь нет встроенной сцены по умолчанию или подсветки Viewport. Требуется создать системы освещения вручную.

Если на сцене нет света, она будет использовать Unlit режим просмотра. В этом режиме просмотра отображается только незатененный Base Color объектов. Это рабочий вариант, но так будет трудно работать в долгосрочной перспективе, потому что вы не можете легко увидеть глубину, а объекты, как правило, сливаются друг с другом.

UE4 может похвастаться отличным освещением и тенями в реальном времени. Независимо от того, используете ли вы Lightmass для высокой производительности, статическое освещение GI или динамические тени и системы освещения для прямого освещения, пара ключевых акторов освещения необходимы для достижения наилучших результатов от UE4.

Sun (солнце)

Для солнца давайте выберем Directional Light Actor. Чтобы добавить его, используйте панель Place Actor. Нажмите на Lights и перетащите Directional Light на сцену.

Установите солнце в положение Moveable на панели Details. Это позволяет ему динамически перемещать и менять свойства и заставляет его использовать динамические тени.

В большинстве архитектурных визуализаций вы использовали бы статическое освещение с Lightmass, но я перейду к этой теме в следующем разделе. А пока сосредоточьтесь на главном. Динамическое освещение предлагает быстрый, простой в редактировании способ WYSIWYG экспериментировать и учиться без сложностей Lightmass.

Атмосферный туман

UE4 включает в себя **моделирование атмосферного тумана**, или математический способ затенения неба и ослабления и тонирования света на больших расстояниях. Это сродни эффекту величия пурпурных гор, если смотреть на них с расстояния нескольких миль, когда свет рассеивается в атмосфере.

Вы размещаете Atmospheric Fog Actor, как Directional Light, используя Place Mode.

Как только вы помещаете Fog Actor на сцену, сразу же появляется основное небо и горизонт, и если вы смотрите в правильном направлении, то вы увидите солнечный диск (рисунок 8.7, далее в этой главе).

Прямо сейчас вы заметите, что солнце находится в неправильном месте, и небо выглядит так, как будто солнце садится. Вы можете либо настроить положение солнца и цвета неба вручную, либо назначить Directional Sun Light для определения этих параметров, что даст вам хорошее динамическое небо.

Установка Солнца в атмосфере

Вы должны вручную определить, какой Directional Light Actor используется для определения цветов неба. Сделать это можно в свойствах Directional Light Actor с флажком **Atmosphere Sun Light** (рисунок 8.5).

Этот флажок немного скрыт в расширенных свойствах Light Actor. Чтобы открыть его, выберите маленькую стрелку вниз в свойствах света на панели Details (рисунок 8.5). Вы также можете использовать строку поиска на панели Details для быстрой фильтрации списка свойств (рисунок 8.6).

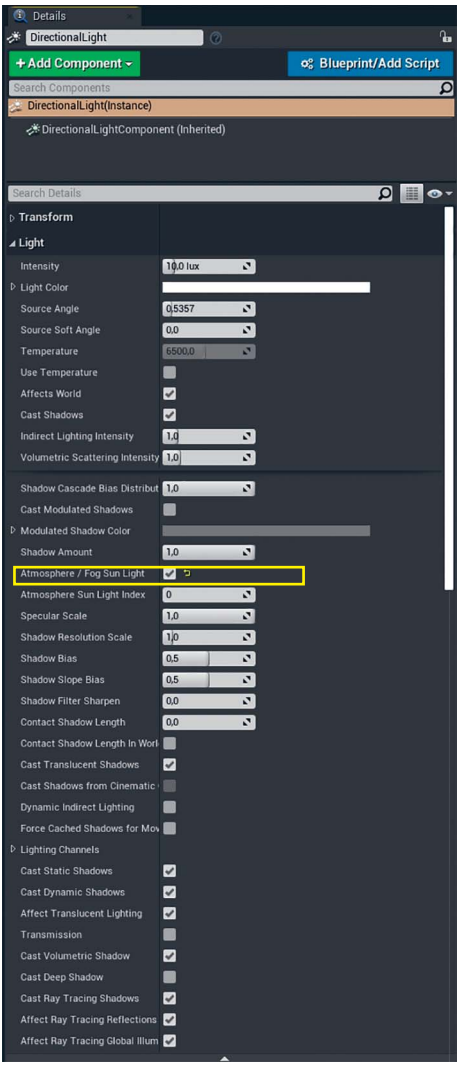


Рисунок 8.5 Детали Directional Light с отображаемыми расширенными свойствами и Atmosphere Sun Light, установленным на true

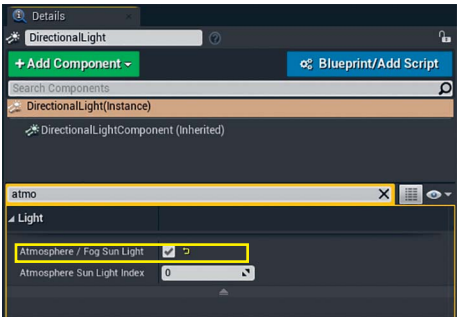


Рисунок 8.6 Отфильтрованные настройки Directional Light

Sky Light

Sky Light Actor — это третий и последний актор, которого вам нужно добавить. Этот актор захватывает HDR-кубическую карту сцены или использует определенную текстуру HDR-кубической карты для освещения сцены.

После добавления этого актора установите для параметра «мобильность» значение Moveable. Ваши затененные области должны быть заполнены синим цветом с неба. Если он слишком яркий, уменьшите Intensity в SkyLight.

Ваша сцена должна выглядеть немного более похожей на то, что показано на рисунке 8.7. Directional Light, Sky Light и Atmospheric Fog Actors — каждый установлен на Moveable, чтобы использовать производственный процесс динамического освещения. Макет Editor настроен таким образом, чтобы обеспечить большой Viewport и больше места для изменения свойств.

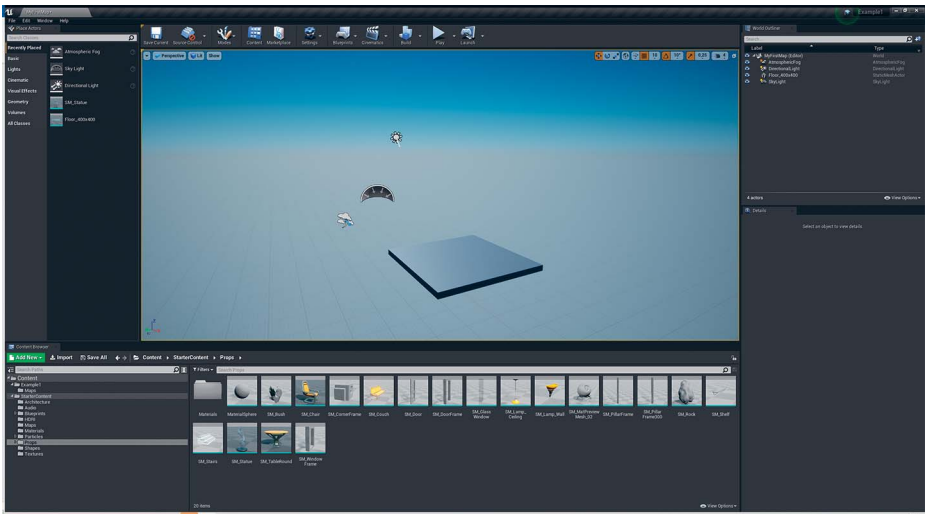


Рисунок 8.7 Основное освещение сцены с Directional Light, Sky Light и Atmospheric Fog Actors

Перемещение по сцене

Теперь, когда на сцене что-то есть, вы, вероятно, захотите немного переместиться и повернуть камеру, чтобы осмотреться.

UE4 имеет отличную комбинацию методов навигации по Viewport, полученных как из игр, так и из приложений 3D-дизайна.

Game Style

Самый распространенный способ перемещения по сцене — это использование игровой навигационной системы.

Удерживая нажатой правую кнопку мыши на Viewport, можно включить Game Navigation Mode. Перетаскивание мыши вокруг (при этом удерживая правую кнопку мыши) вращает вашу камеру. Нажатие **W** на клавиатуре переместит вас вперед, а **S** — назад; нажатие **A** и **D** переместит вас влево и вправо соответственно.

Вы также можете двигаться вверх и вниз с помощью клавиш **E** и **Q**.

Для настройки скорости движения вы можете использовать колесо мыши, чтобы ускорить и замедлить, как быстро вы перемещаетесь по уровню.

Object Focused

Вы также можете сфокусировать камеру на любом акторе (или группе выбранных акторов) с помощью сочетания клавиш **F**. Эта клавиша центрирует и приближает камеру к выбранным объектам. Когда это произойдет, вы можете вращаться вокруг выбранных акторов, удерживая нажатой клавишу Alt и перетаскивая ее левой кнопкой мыши.

Вид орбиты отлично подходит для осмотра объектов в 3D-режиме.

Вы можете легко увеличивать и уменьшать масштаб сфокусированных акторов, используя колесо прокрутки на вашей мыши (на этот раз без удержания правой кнопки мыши).

Создание архитектуры

Используя различные Static Meshes в папке Architecture в Starter Content, давайте построим простую квартиру или дом.

Если у вас есть Snapping, установите интервал в 100 units. Это делает каждый меш в этой папке привязанным к сцене как строительные блоки.

Вам понадобится пол, стены и все остальное, что вы придумаете; повеселитесь.

Убедитесь, что есть дверь, которая соединяет разные комнаты, чтобы можно было прогуляться. Каждая часть пола составляет 400 x 400 см, поэтому каждая комната должна быть не менее 2 x 2.

На рисунке 8.8 показано, что я придумал после того, как немного повеселился, собирая кучу блоков. Я попытался сделать что-то, что удержит игрока в небольшой области, но будет достаточно открытым для изучения. Используются только статические меши стен и пола из Starter Content, привязанного к сетке 100 × 100. Я выставил Orthographic Viewports, используя кнопку Maximize/Minimize в правом верхнем углу каждого Viewport. Я также настроил второй перспективный вид, позволяющий мне видеть сцену из нескольких 3D-видов.

Ваша область, конечно, не должна быть такой сложной или может быть гораздо более сложной — зависит от вас. Однако убедитесь, что у вас есть пол, чтобы игрок мог стоять на нем, и некоторые стены, чтобы они не могли уйти в пустоту.

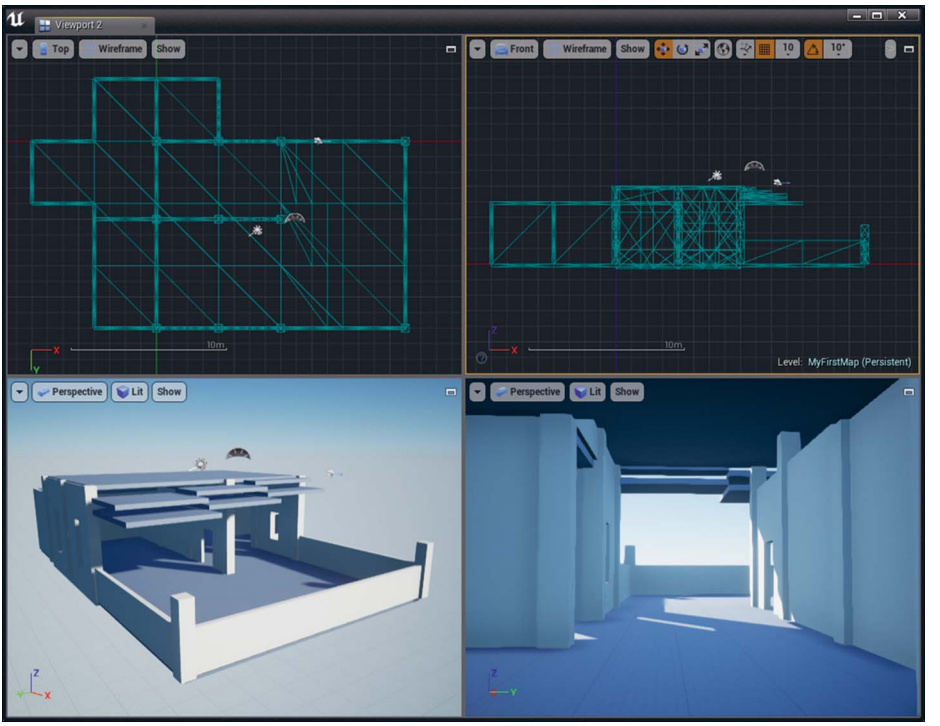


Рисунок 8.8 Дом плавает в бесконечной пустоте

Добавление деталей к структуре

Теперь, когда у вас есть структура, давайте рассмотрим некоторые детали. Разместим несколько Static Mesh Actors из Content Browser, а затем добавим Spot Light Actors с профилями освещения и различными цветами, чтобы задействовать забавные световые эффекты.

Placing Props

Props (реквизит) — это Static Mesh Actors на сцене, которые составляют декорации и другие неархитектурные элементы. Sample Content поставляется с хорошим выбором мешей для игры.

Как и другие меши, просто перетащите их из Content Browser, чтобы разместить на вашем уровне. Скопируйте и вставьте, а затем переместите и масштабируйте по вкусу.

Вы также можете разместить материалы и системы частиц на своем уровне. Сходите с ума, держите его минимальным — на ваш вкус. Все, что действительно нужно, — это пол и несколько стен; остальное зависит от вашего воображения.

Рисунок 8.9 Моя сцена после нескольких минут размещения мешей

Вот что я придумал еще через несколько минут перетаскивания, копирования, вставки, привязки поверхности и клонирования (рисунок 8.9). Я потратил некоторое время на то, чтобы сделать свой плавучий дом домом, поместив различный реквизит из Content Browser, а также дублируя и манипулируя мешами. Камни помогают визуально закрепить конструкцию.

Самое время попробовать различные варианты привязки. Я призываю вас изучить возможность Surface Snapping, чтобы помочь вам разместить акторов непосредственно на других акторах.

Размещение света

Как и в случае с Directional Sun Light и Sky Light Actors, используйте Class Browser для добавления света на сцену.

При нажатии на вкладку Lights в Class Browser отображается список доступных классов light (рисунок 8.10). Просто перетащите нужный класс источника света на Viewport, чтобы поместить Light Actor на свой уровень.

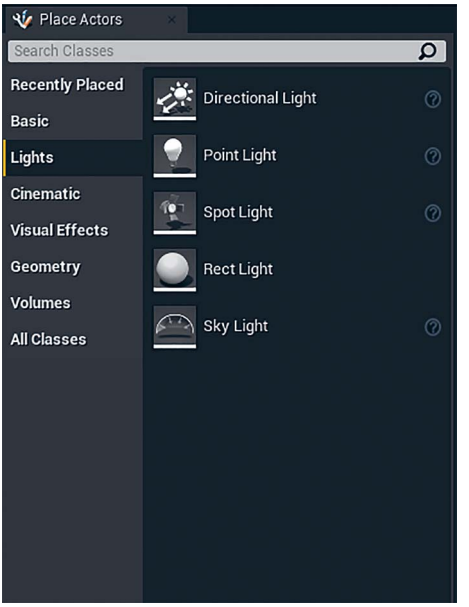


Рисунок 8.10 Light Classes из Placement Actors

Как и в случае со Static Mesh Actors, вы можете вращать и перемещать источники света с помощью Gizmo или элементов управления трансформацией на панели Details. Вы можете копировать, вставлять и дублировать таким же образом.

Свойства света

Потратьте время на изучение свойств Light Actor на панели Details. Оно предлагает варианты яркости, тени и цвета. Многие из этих вариантов являются чисто визуальными, в то время как многие из них тесно связаны с производительностью.

Динамическое освещение и производительность

Поскольку вы используете динамическое освещение, будьте осторожны с количеством источников света. Хотя deferred renderer UE4 позволяет использовать гораздо больше динамических источников света, чем методы рендеринга предыдущего поколения, они являются дорогостоящими эффектами, особенно когда речь заходит о затенении.

Тени

Динамические тени добавляют много накладных расходов на рендеринг, и вы должны использовать их экономно. Затененные точечные источники света являются самым дорогим видом света для визуализации и должны использоваться наиболее экономно.

Радиус затухания

Вы также можете уменьшить радиус затухания (Attenuation Radius) огней настолько, насколько это возможно, чтобы повысить производительность. Акторы вне радиуса не будут затронуты светом, и освещение и тени от этого источника света не будут вычисляться.

Добавление профилей IES

UE4 поддерживает 2D-профили IES для точечных и прожекторных источников света. Профили IES модулируют яркость света с помощью текстуры, сгенерированной из импортированного файла IES. UE4 поставляется с несколькими профилями IES, или вы можете импортировать собственный в Content Browser, как и в случае с любым другим типом контента.

Вот моя сцена после размещения нескольких источников света (рисунок 8.11). Я также добавил некоторые профили IES к точечным источникам света, чтобы сделать их немного более интересными. Вы можете найти свойство IES на панели Details, когда настраиваете Light Actor на своем уровне.

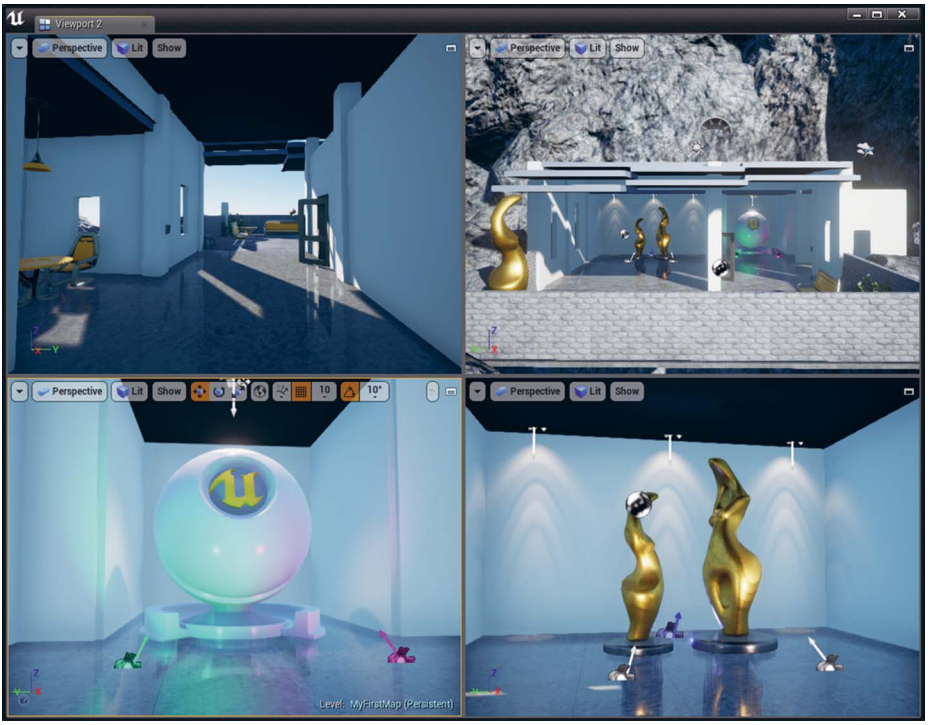


Рисунок 8.11 Финальная сцена в Editor со всеми четырьмя Viewports, настроенными на перспективу, что позволяет видеть уровень во время работы в нескольких проекциях

Заключение

Вы видели, как легко заполнить уровни ассетами. Источники света, материалы и акторов можно перетаскивать и удалять, копировать и перемещать, поворачивать и масштабировать так же легко, как в нашем любимом 3D-приложении. Построение уровней в UE4 — это весело и интерактивно, и получить отличную настройку освещения очень легко с помощью Atmospheric Fog и Skylight.

Иметь полный контроль над виртуальным пространством и создавать что-то без каких-либо реальных ограничений, кроме простых требований стен и пола, — это весело.

Я включил эту сцену в материалы на www.TomShannon3d.com/UnrealForViz для вас, чтобы вы могли открыть ее самостоятельно и играть.

ГЛАВА 9

СОЗДАНИЕ ИНТЕРАКТИВНОСТИ С BLUEPRINTS

Вы создали свой первый проект, наполнили мир и установили некоторые источники света и реквизит, чтобы он выглядел красиво. Это хорошо, но чтобы сделать проект экстраординарным, нужно добавить ему интерактивности. В этой главе вы узнаете, как создать первые классы Blueprints, настроить первый Game Mode и заставить игрока перемещаться по миру с помощью системы ввода.

Настройка проекта

Требования проекта включают вид от первого лица и движение в стиле ходьбы. К счастью, создание персонажа от первого лица и движения по типу ходьбы — это то, что Unreal Engine может сделать действительно хорошо.

Сначала вы создадите павн, Blueprint Class, который получает входные данные, обрабатываемые классом Player Controller, который вы научитесь делать позже. Наконец, вы создадите пользовательский класс Game Mode, чтобы определить эти классы как значения по умолчанию для вашего проекта.

С этими объектами вы сможете легко разместить Player Start Actor, чтобы определить, где ваш игрок будет спавниться, когда он загрузит этот уровень.

В конце главы ваш павн сможет плавно ходить по уровню, исследуя все вокруг и озираясь.

Процесс настройки для создания Game Mode, в комплекте с Player Controller, отображением входных данных и павном, довольно длительный, но в конце концов вы поймете, как UE4 обрабатывает входные данные и как он ожидает от вас перемещения своих игроков. Вы также сможете перенести этот контент в новый проект, что позволит вам развивать свою работу по мере добавления возможностей к вашим собственным интерактивным визуализациям.

Нажатие кнопки Play

Вы, вероятно, уже нажали кнопку Play в Editor до этого момента. Все в порядке, трудно этого не делать. В этом вся причина существования UE4: запуститься!

Нажатие кнопки Play в Editor запускает вашу игру в специальном режиме под названием **Play in Editor** (PIE), который использует уже загруженные ассеты, чтобы мгновенно запустить мир и заставить вас играть как можно быстрее. Это фантастический способ протестировать ваше приложение без необходимости каждый раз загружать его с нуля.

Режимы игры

Вы можете выбрать Option справа от кнопки Play, чтобы открыть некоторые дополнительные функции — режимы игры (Play Mode; рисунок 9.1).

Режим по умолчанию — запуск игры на **Selected Viewport**. Вы также можете запустить его в **New Editor Window**. Это полезно, если ваш Viewport скрыт или вы хотите протестировать его в определенном разрешении или соотношении сторон.

Выбор **Standalone Game** запускает сырую сборку вашей игры из командной строки. Это занимает больше времени для запуска, чем PIE (который работает почти

мгновенно), но дает более точное представление о том, как ваше финальное приложение будет вести себя вне редактора.

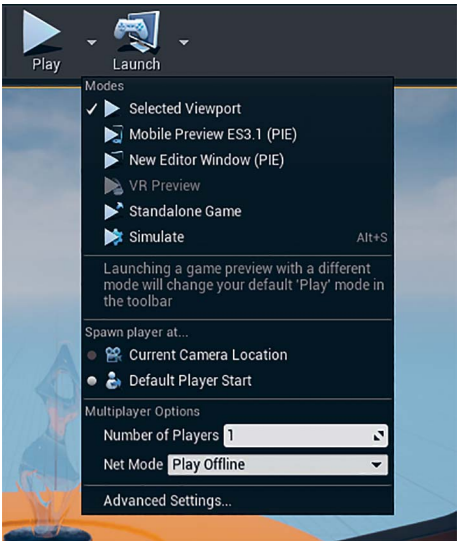


Рисунок 9.1 Параметры PIE

Simulate — это интересный режим, который запускает игру, но не порождает типичный Player Controller и павн, оставляя вам свободу исследовать уровень, поскольку он работает без ограничений, наложенных на обычные павны.

После выбора одного из этих параметров кнопка Play изменится в соответствии с последним выбранным режимом.

Если вы хотите изменить разрешения и другие параметры запуска, выберите **Advanced Settings**. После этого откроется диалоговое окно Editor Settings, в котором вы сможете изменить их.

Default GameMode

Нажатие кнопки Play на уровне в этот момент спавнит PC по умолчанию и павн, и позволяет вам летать по уровню, используя летающего персонажа с игровым управлением. Это **GameMode** по умолчанию в UE4.

GameMode — это класс, который определяет, какой Player Controller, Pawn и другие классы будут использоваться при запуске игры.

Обратите внимание, что в этом режиме вы являетесь полноценным игроком с реакциями на столкновения и физику, в отличие Viewpoint в Editor, где вы рассматриваетесь как

призрак. Однако вы также можете летать и двигаться очень быстро. Это не совсем подходящий способ для исследования собственного мира.

Вы должны создать собственный Pawn, PC и GameMode, а затем назначить их своему проекту. После этого вы сможете обойти свой уровень так, как определено в области проекта.

Создание Pawn

Класс, который представляет ваше физическое присутствие в мире, — это **Pawn**. Pawn управляет физикой, столкновением и взаимодействием с миром и другими участниками уровня.

Pawn — это Blueprint-классы. Они содержат компоненты, переменные и Event Graph. Чтобы создать его, выполните следующие действия.

1. Выберите **Add New** в Content Browser и **Blueprint Class** (рисунок 9.2).

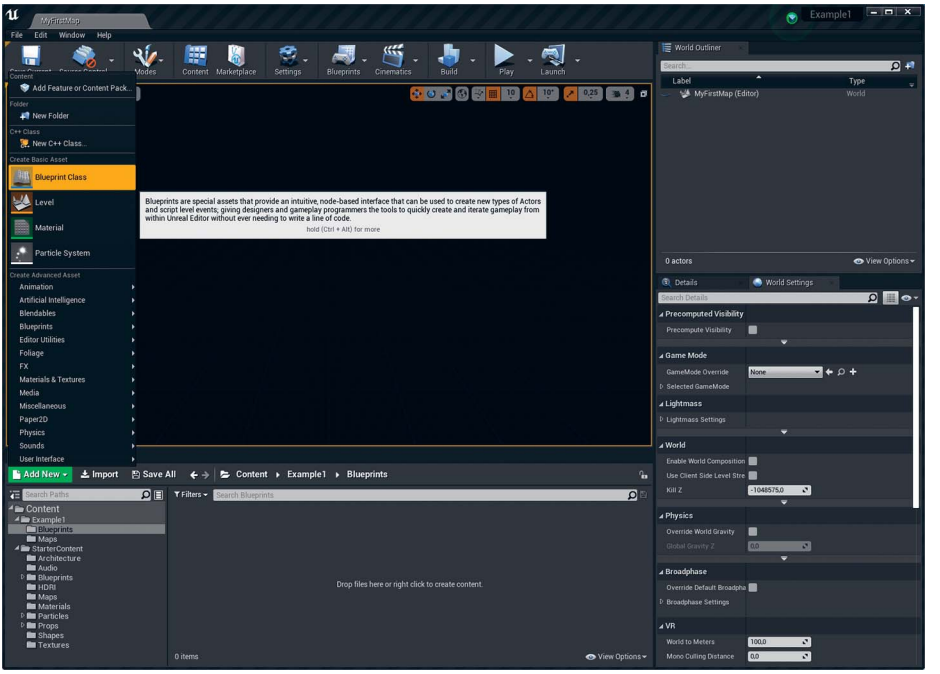


Рисунок 9.2 Создание нового ассета Blueprint в Content Browser

2. Выберите кнопку **Character** в открывшемся диалоговом окне **Pick Parent Class**. Класс Character — это улучшенный класс павна, который работает для широкого спектра приложений персонажей от первого и третьего лица, поэтому он

идеально подходит для нужд этого проекта и экономит много усилий с вашей стороны. (Нет *никакой необходимости* постоянно изобретать велосипед!)

3. Назовите свой ассет. Я выбрал BP_Interior_Pawn, следуя общепринятым соглашениям об именовании UE4 (рисунок 9.3).

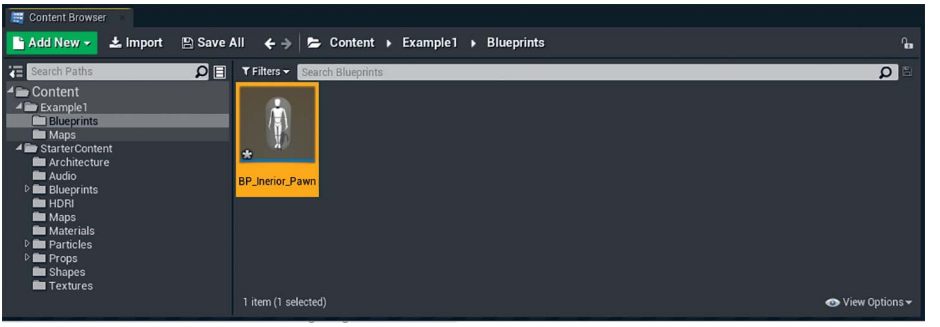


Рисунок 9.3 Созданный и названный BP_Interior_Pawn Blueprint Asset в Content Browser

4. Дважды щелкните по ассету, чтобы открыть Blueprint Editor.

5. Выберите компонент BP_Interior_Pawn из списка Components и убедитесь, что на toolbar выбрана кнопка Class Defaults (рисунок 9.4).

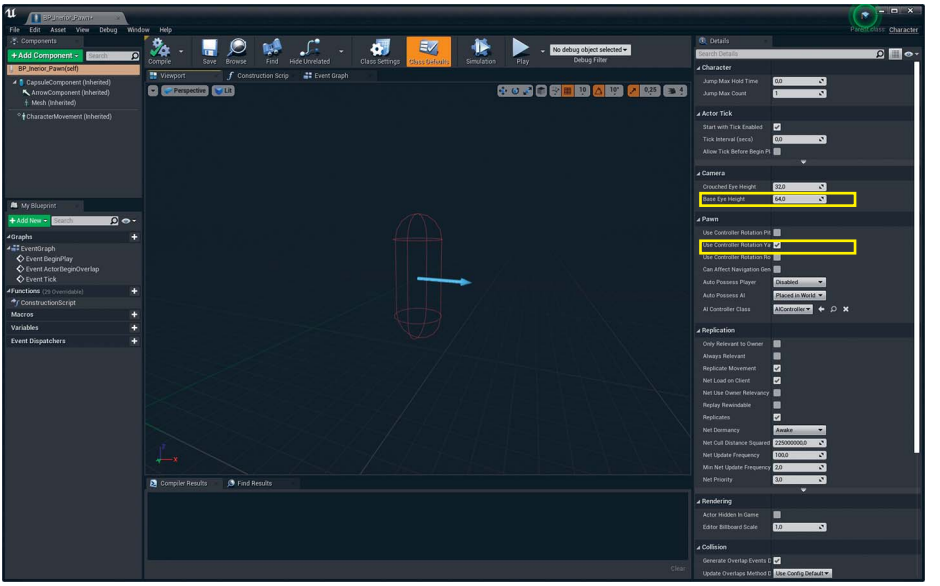


Рисунок 9.4 Значения по умолчанию класса Pawn, отображаемые в Blueprint Editor

Настройка View Height игрока

Чтобы установить точку зрения игрока на разумную высоту, вам нужно настроить Base Eye Height вместе с Half Height у Capsule Component.

Уровень глаз игрока устанавливается путем добавления свойства Base Eye Height со свойством Half-height у Capsule Component. Вам нужно настроить эти параметры как для достижения нужной высоты, так и для столкновения игрока.

Регулирование Capsule Component

Pawn использует Capsule Collision Component для имитации тела игрока в мире. Вы можете увидеть Capsule Component на вкладке Viewport и в списке компонентов (рисунок 9.5 далее в этой главе).

Выберите Capsule Component из списка компонентов, чтобы получить доступ к двум его переменным: Radius и Half-height. Как следует из названия, полувысота представляет собой расстояние от пола до середины Capsule Collision Component. Это значение по умолчанию составляет 88 см, или 176 см (около 5'9") от пола до верха капсулы. Это низко для большинства людей, но лучше ошибиться немного, чем позволить голове игрока зацепиться за дверные проемы и светильники. Вы можете сохранить это значение по умолчанию.

Настройка Base Eye Height

UE4 вычисляет высоту взгляда персонажа, добавляя свойство Half-Height Capsule Component к Base Eye Height, чтобы получить окончательную высоту.

Значение базовой высоты взгляда по умолчанию 64 дает высоту всего 152 см, что кажется слишком низким для большинства людей. Средняя высота глаз у мужчин составляет около 175 см, а у женщин — около 160 см. Цель на 168 дает почувствовать разницу, оставляя ее в 80 см Base Eye Height.

Чтобы вернуться к свойствам класса Character по умолчанию, щелкните на корневой компонент (BP_Interior_Pawn) в списке компонентов и установите Base Eye Height равной 80.

Использование Controller Rotation Yaw

Следующая настройка в классе нужна, чтобы знать о том, как будет определяться поворот взгляда игрока. Yaw — это вращение игрока по оси Z, контролирующее, как он поворачивается влево и вправо.

Хотя вы оставите это значение по умолчанию, важно знать, что здесь происходит. Movement component не управляет вращением; это делает Player Controller. Кроме того, поскольку вы не хотите, чтобы вся капсула качалась вверх и вниз, когда вы смотрите вверх и вниз, вы используете только Yaw (вращение по оси Z), оставляя Pitch для применения к камере, которая является родительским павном.

Наличие двух компонентов, независимо обрабатывающих pitch и yaw, позволяет избежать интерактивной проблемы камеры, обычно известной как **Gimbal Lock**, когда камера не поддерживает горизонтальную стабильность при вращении.

Включение Controller Rotation Yaw автоматически соотносит yaw объекта Pawn с Control Rotation Parameter у Player Controller.

Настройка Movement Speed

Компонент **CharacterMovement** отвечает за перемещение персонажа по сцене и управление состояниями движения. Здесь же вы найдете настройки по умолчанию, относящиеся к движению персонажа, такие как скорость ходьбы, режим движения по умолчанию и так далее.

Найдите компонент CharacterMovement на панели Components и выберите его, чтобы просмотреть свойства по умолчанию на панели Details (рисунок 9.5). Вы можете фильтровать панель Details с помощью значка глаза рядом со строкой поиска в верхней части панели Details.

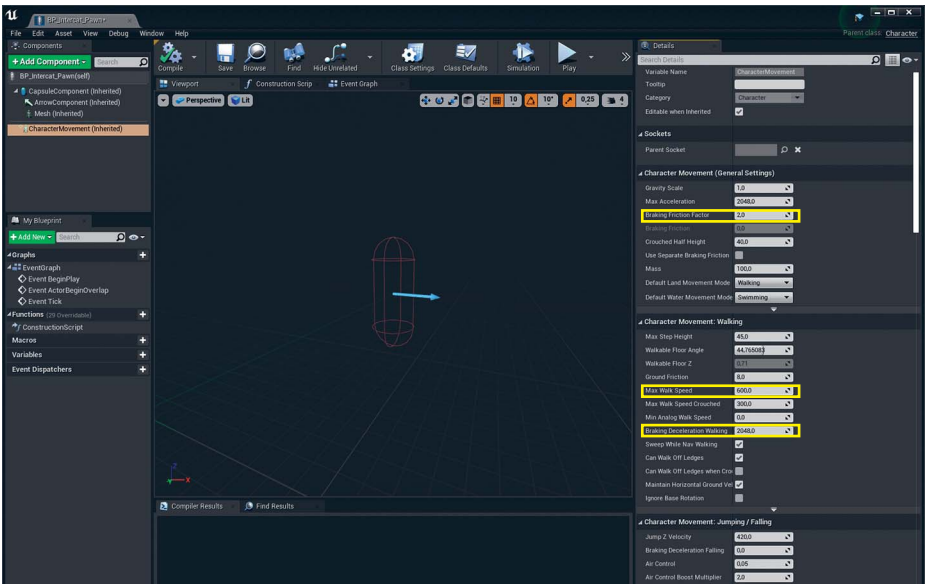


Рисунок 9.5 Измененные свойства CharacterMovement Component

Вам не так уж много нужно изменить для этого простого проекта, но обратите внимание, что варианты обширны и вы можете использовать их для создания тонны различных типов движений, от полета до падения, плавания и лазания.

Max Walk Speed

По умолчанию игрок должен двигаться со скоростью 600 см в секунду (13 миль в час). Это может быть слегка быстро для большинства людей для движения по комнате. Установите этот параметр где-то в диапазоне 150–175 для медленной прогулки.

Braking Friction и Deceleration

Для замедления персонажа доступны две настройки, и обе по умолчанию имеют высокие значения (предположительно, для игр с быстрым действием, где люди ходят со скоростью 6 метров в секунду).

Установите Braking Friction Factor 1, а Braking Deceleration Walking — 0.0.

Это позволяет игроку прийти к плавной остановке, а не к резкой, необходимой в большинстве игр для точного управления. Это личное предпочтение и демонстрация того, как вы можете действительно настроить «ощущение» вашего персонажа, используя класс Character.

Теперь, когда вы создали и изменили свой класс павна, сохраните его, прежде чем продолжить. Помните, что вновь созданные ассеты не записываются автоматически на диск, поэтому вы должны сохранить их вручную.

Привязка ввода данных

Привязка ввода данных (Input Mappings) — это параметры всего проекта, которые позволяют настраивать общие события ввода, такие как щелчки мыши или нажатия клавиш, и получать к ним доступ из Blueprints.

Чтобы получить доступ к Input Mappings, перейдите в меню Edit и выберите пункт Project Settings.

Выберите Input в левом столбце (рисунок 9.6), чтобы просмотреть Input Mappings.

Action и Axis Mapping

Action Mappings — это одноразовые события, вызванные событиями одного действия, такими как нажатие кнопок и щелчки мыши. Эти сопоставления срабатывают за один тик.

Axis Mappings представляют собой события, которые могут иметь числовое значение и способны вызываться от кадра к кадру. Это было бы похоже на то, как быстро вы хотите двигаться вперед или повернуть налево или направо. Их называют *Axis Mappings*, потому что они традиционно относились к игровым джойстикам, которые определяли их движение по своей оси.

В современных интерфейсах это может быть количество пикселей, перемещенных мышью в последнем тике, нажатой и удерживаемой клавишей или положением аналогового джойстика на геймпаде.

Для перемещения и вращения игрока вы будете использовать исключительно Axis Mappings.

Настройка Mappings (отображений)

Массив Axis Mappings состоит из списка Mappings. Каждый Mapping имеет Label и массив входных данных, каждый из которых имеет значение Scale.

Label — это то, на что ссылается Mapping в Blueprints. Вы можете назвать этот Label как угодно: Walk, Turn, EatShrimp. Нет никакого установленного стандарта. Все, что вы назначаете этому полю, становится доступным в качестве Input Event и переменной Axis Value, которая легко доступна в Player Controller.

Каждый именованный Input Axis может иметь несколько входов. Например, Mapping «MoveForward» может иметь назначенные как клавиши W, так и S. У клавиши S Scale –1, тогда как у W Scale +1. Это означает, что, когда это событие вызывается игроком, нажимающим W, оно возвращает значение 1, а игроком, нажимающим S, возвращает значение –1. Таким образом, вы можете сгруппировать входные данные, чтобы избежать слишком большого количества событий в вашем игровом коде.

Прежде чем продолжить, настройте Input Mappings так, чтобы они выглядели как на рисунке 9.6. Настройки в Project Settings проекта автоматически сохраняются в конфигурационных файлах проекта. Вам не нужно нажимать кнопку Save, когда вы закончите; вы можете просто закрыть окно настроек.

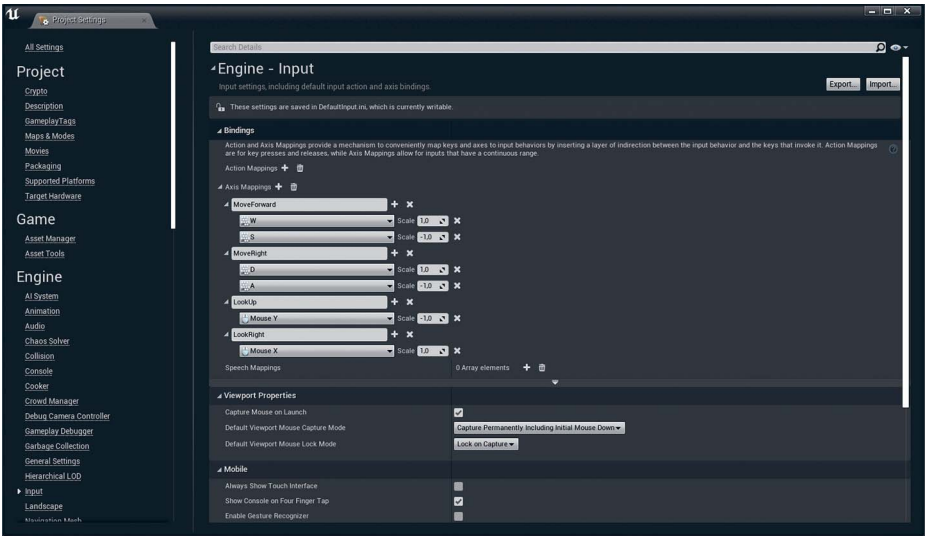


Рисунок 9.6 Диалог настройки Input Mapping

Input Device Flexibility

Еще одним преимуществом такой настройки ввода является возможность использовать ввод с нескольких устройств в едином виде. Нажатие Enter может вызвать то же событие, что и нажатие кнопки X на геймпаде или щелчок правой кнопкой мыши. Это полностью зависит от вас.

Из-за этой гибкости UE4 не поставляется с настройками по умолчанию. Вы можете сделать Input Mappings точно в соответствии с потребностями вашей системы ввода. Эти настройки хранятся в файле Saved/Config/DefaultInput.ini вашего проекта, и вы можете импортировать и экспортировать его с помощью кнопок в интерфейсе Project Settings или путем копирования и вставки текстового содержимого из других входных файлов.

Создание Player Controller Class

Теперь, когда у вас есть павн и настроен Input Mappings, вам нужно собрать их вместе. Хотя вы *можете* поместить свою логику ввода и движения непосредственно в класс павна, это не лучший шаблон проектирования.

Для этого у вас есть Player Controller. Как следует из названия, одной из основных функций Player Controller является обработка входных данных от игрока. Помните, что при запуске приложения UE4 *всегда* есть Player Controller, поэтому это отличное место для размещения кода, который *всегда* должен работать, например обработка входных данных.

Создайте свой Player Controller точно так же, как Pawn и GameMode, которые вы сделали раньше, щелкнув правой кнопкой мыши в **Content Browser** и выбрав пункт **Blueprint** в меню **Create Basic Asset**.

Выберите класс **Player Controller** в окне **Pick Parent Class** и назовите вновь созданный ассет **BP_UE4Viz_PlayerController**.

Обязательно сохраните свой ассет на диске в первый раз.

Добавление Input с Blueprints

Теперь, когда вы определили все входные значения, которые вам нужны, вы можете начать использовать их для управления событиями в вашей игре с помощью сценариев Blueprint! Давайте начнем с открытия Blueprint Editor. Вы делаете это, как и во многих редакторах в UE4, просто дважды щелкнув на свой ассет Player Controller в Content Browser.

Когда ваш PC открыт, вам нужно добраться до Event Graph чтобы начать написание кода обработки входных данных. Если в окне Blueprint Editor отсутствует Event Graph и вверху отображается сообщение о том, что это только Data Only Blueprint, вам нужно нажать

кнопку **Open Full Blueprint Editor**, чтобы увидеть полный интерфейс Blueprint Editor. Когда он будет доступен, перейдите на вкладку Event Graph.

Добавление Axis Events

Вам нужно определить Input Axis Mappings, которые вы настроили ранее. Когда срабатывает один из Input Axis Mappings (обычно при выполнении игроком выбранного действия), оно запускает событие, доступ к которому можно получить из Blueprints. Это называется **Axis Event**, и оно передает одну переменную: **Axis Value**.

Axis Value представляет значение ввода, умноженное на Scale, которое вы установили в Input Mappings dialog (см. рисунок 9.6). Итак, нажатие на W запускает input event, возвращающий 1.0 в течение каждого Tick удерживания кнопки, и нажатие S также запускает input event, но возвращает значение –1.0 потому, что Mapping устанавливает Scale value –1. Также эти значения являются совокупными, если Player нажмет W и S одновременно, input axis value вернет 0 потому, что вводы отменяют друг друга.

Клавиши на клавиатуре, конечно, цифровые и могут вернуть только 1 или 0. Тем не менее Axis Value могут работать с аналоговым дивайсом, таким как геймпад, который может вернуть любое значение между –1 и 1 по каждой оси, или с вводом мышью, который возвращает количество пикселей, пройденных мышью с последнего момента считанного ввода (иногда называемых **input delta**, или разницей между двумя выборками). Эта настройка позволяет множеству различных типов вводов использовать одну и ту же координатную ось и соответствующее событие, упрощая код обработки ввода.

Для добавления Axis event в ваш Player Controller, нажмите правую кнопку мыши в Event Graph's Graph Editor и найдите Axis name, определенное Settings (рисунок 9.7). Просто начните печатать, и список будет отфильтрован динамически.

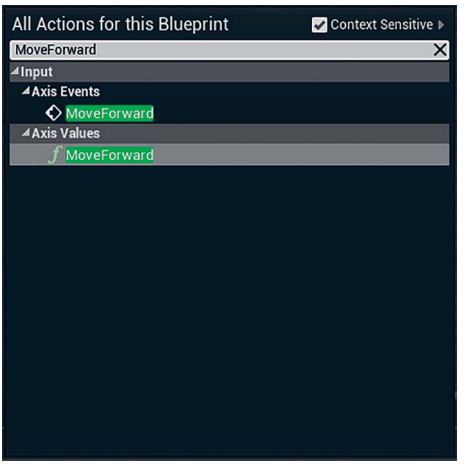


Рисунок 9.7 Поиск Move Forward Axis Event с использованием контекстного меню, доступного при правом клике в Blueprint Event Graph

Добавьте события для каждого из ваших движений Axis Mappings: MoveForward и MoveRight (рисунок 9.8). Добавьте еще два для Rotation Mappings: LookUp и LookRight.

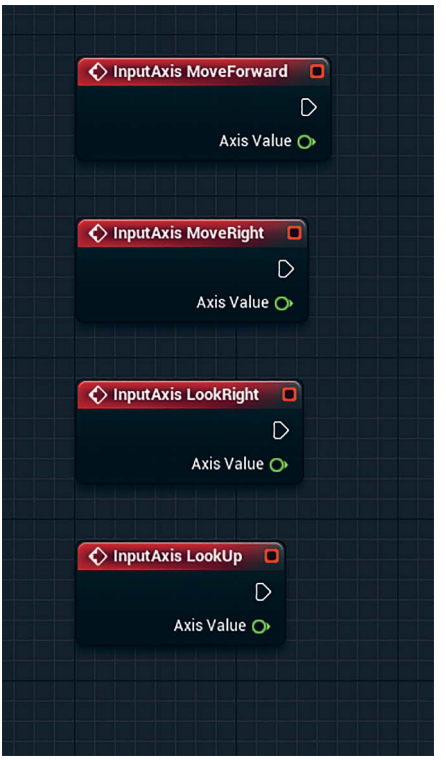


Рисунок 9.8 Добавленные Axis Events

Обратите внимание на вывод Axis value на рисунке 9.8. Оно возвращает значение с плавающей точкой, которое представляет Player input value масштабируемое Input Scale назначенный в Input Mapping.

Поворот взгляда

Вращать поле обзора действительно просто. Благодаря Character Class, от которого вы наследуете ваш Pawn, вам достаточно вращать Player Controller (помните, что ваш Player Controller управляет полем зрения), и взор будет следовать за ним.

Поскольку это часто используется, UE4 предлагает несколько быстрых решений. **Add Yaw Input** и **Add Pitch Input** две готовые функции, принимающие входное значение и записывающие изменение значения вращения в переменную PC's Control Rotation.

Выполните следующие шаги для создания этих узлов.

1. Нажмите правой кнопкой мыши в the Event Graph и найдите Add Yaw Input и Add Pitch Input.

2. Подключите вывод **Exec Out** (белая стрелка в правой части узла) от InputAxis LookRight Event и подключите его к входу Add Yaw Input узла **Exec In** (белая стрелка в левой части узла), перетаскивая из одного в другой. Вы можете перетащить выходы в любом направлении.

3. Соедините выход **Axis Value** вершины InputAxis LookRight с контактом **In Val** вершины Add Yaw Input.

4. Повторите эти действия для LookUp axis и функции Add Pitch (рисунок 9.9).

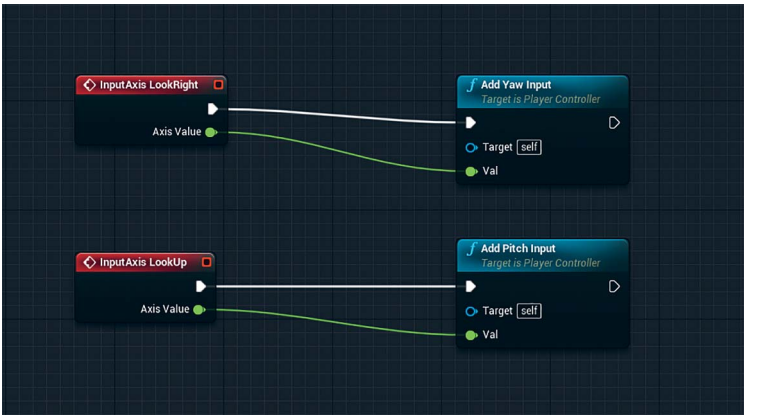


Рисунок 9.9 Завершенный Rotation Input скрипт

Передвижение игрока

Движение игрока немного сложнее настроить, нежели вращение обзора. Player Controller должен сообщить объекту Pawn о движении. Чтобы сделать это, вам нужно использовать модель связи Blueprint для передачи входных данных героя в Pawn, чтобы он мог двигаться.

Ссылка на Pawn

Когда Player нажимает на одно из Axis Mappings и запускает эти события, вы хотите, чтобы Pawn двигался. Чтобы сделать это, вам необходимо связаться с ним и первым шагом получить **Reference** на Pawn.

Базовый PC-класс, от которого вы наследуете ваш PC, содержит переменную **Player Character**, заполняемую автоматически с Reference на Character Pawn, которой владеет в данный момент. Это позволяет вам напрямую получить доступ к вашему Character и легко добавить ввод движения.

Вы можете получить ссылку на Player Character, щелкнув правой кнопкой мыши в Event Graph и найдя **Character** во всплывающем Action List. Выберите функцию **Get Player Character**.

Is Valid

Так как Player Controller может обладать практически любым типом Actor в мире, переменная Player Character может вернуть **none**, если она не контролирует Actor или Pawn, наследуемый от Character Class.

Дабы избежать этого, используйте **Is Valid** branch, чтобы убедиться, что этого не произойдет (рисунок 9.10). Сценарий будет продолжаться только в том случае, если значение Player Character is valid и не выдает ошибку при обращении к нему.

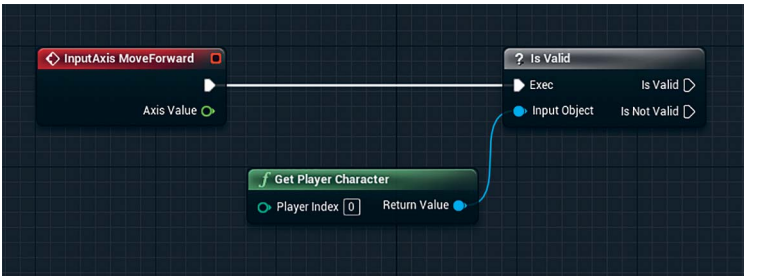


Рисунок 9.10 Получение ссылки на Player Character и проверка его валидности

Всегда проверять, когда «получаете» другие Actor или Objects, которые ненадежно существуют в мире, — это хорошая идея.

Add Movement Input

Character Class уже имеет функции для перемещения Player по сцене с помощью Movement Component. Функция называется Add Movement Input, но вам нужно получить доступ к ней иначе, чем вы делали.

До сих пор вы нажимали правой кнопкой мыши в Viewport и использовали контекстное меню для создания узлов. В этом меню отображаются только те узлы, которые доступны для Blueprint, в котором он находится. Functions, Events и Variables, которые хранятся в других Blueprints, должны быть доступны по ссылке.

Для этого протяните провод от синего контакта Return Value из Get Player Controller в Graph Editor и отпустите. Появится контекстное меню, заполненное Functions, Events и Variables, связанными с указанным классом (рисунок 9.11). Обратите внимание, что в контекстном меню написано Actions taking a(n) Character Reference. Это говорит о том, что вы успешно ссылаетесь на другой Blueprint Class.

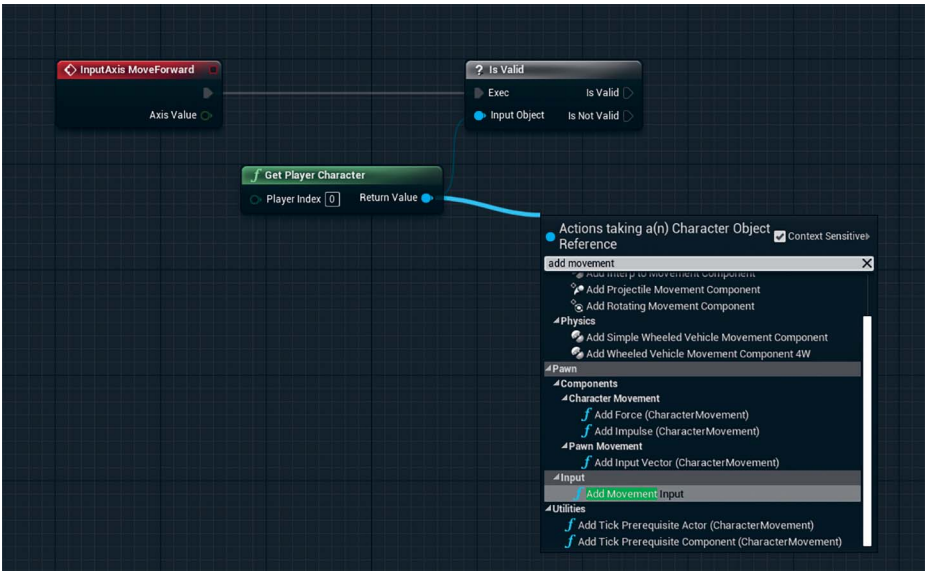


Рисунок 9.11 Добавление Add Movement Input из Character Class Reference

Найдите Add Movement Input и нажмите на него, чтобы разместить в Event Graph, подключив, как показано на рисунке 9.12

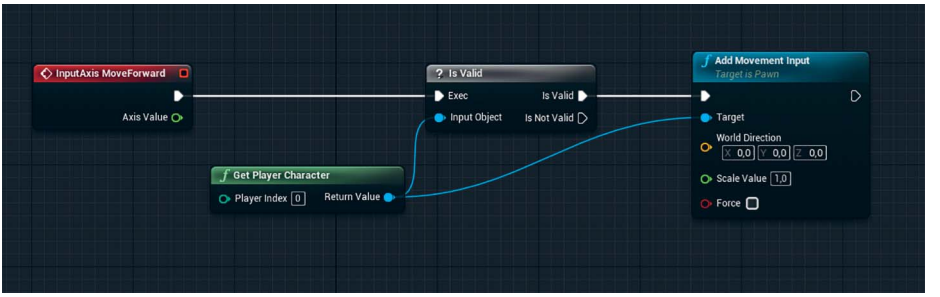


Рисунок 9.12 Размещенная функция Add Movement Input

На рисунке 9.12 обратите внимание на синий провод, соединяющий узел Get Player Controller с Target Add Movement Node. Этот провод представляет собой ссылку на Player Character.

Get Forward и Right Vectors

Функция **Add Movement Input** использует мировой вектор для перемещения вашего персонажа. Этот вектор меняется в зависимости от того, куда направлен Player, поэтому вам необходимо выяснить и использовать это для определения направления движения.

Вот где функции **Get Actor Forward Vector** и **Get Actor Right Vector** применяются. Они преобразуют world rotation в нормализованные векторы XYZ для каждого направления.

Как и в случае с узлом Add Movement Input, вы должны вытянуть ссылку из Return Value узла Get Player Character's, чтобы получить доступ к списку функций. Найдите узлы и разместите их в Event Graph (рисунок 9.13). Вы можете увидеть ссылки к Components из Character Class; убедитесь, что выбираете корневой компонент, как показано.

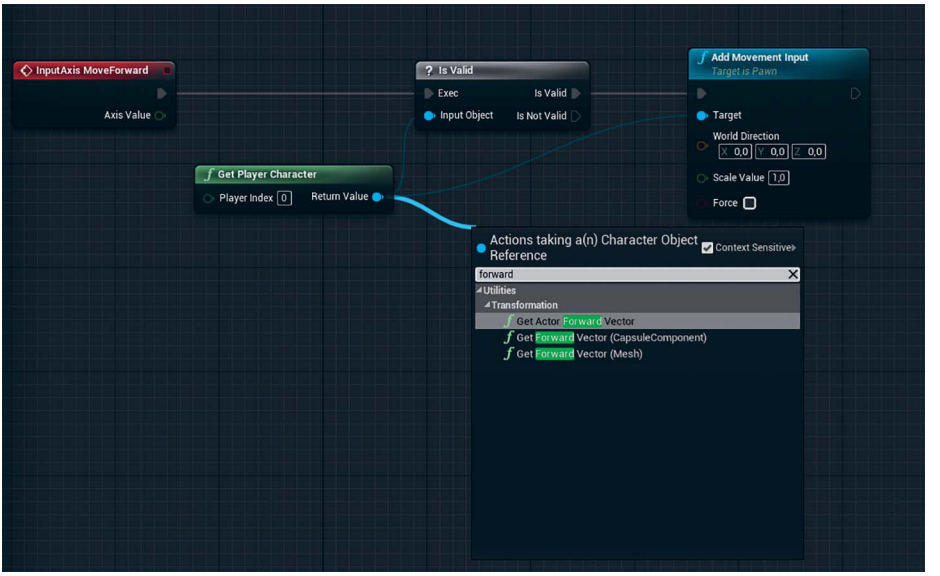


Рисунок 9.13 Добавление функции Get Actor Forward Vector

Scaling Input

Не вдаваясь сильно в векторную математику, вы можете просто умножить вектор (три числа с плавающей запятой, такие как X, Y, Z) по Axis Value из InputAxis Events, масштабируя каждое направление на введенную сумму.

Перетаскивание соединения из Return Value узла Get Actor Forward Vector в Event Graph и отпускание, позволит вам увидеть методы, доступные для Vector Reference. Найдите *(звездочку), чтобы увидеть доступные функции умножения и выберите **vector * float** из списка (рисунок 9.14).

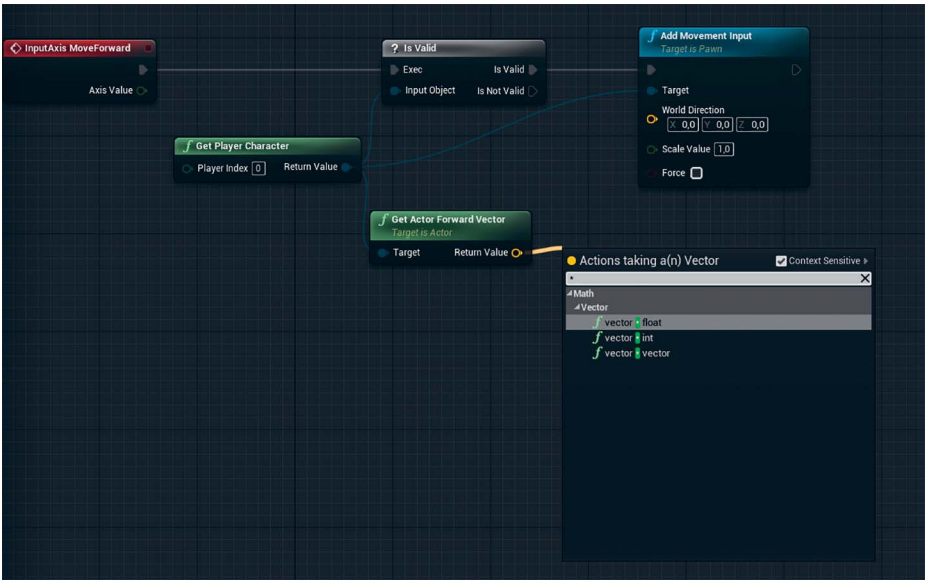


Рисунок 9.14 Добавление математического узла **vector * float**

Затем вы можете подать масштабируемое значение Vector на входные контакты World Direction функции Add Movement Input, завершив ваш код движения вперед. Вам нужно повторить этот процесс для Move Right InputAxis Event. Единственное отличие состоит в том, что вы заменяете функцию Get Actor Forward Vector на узел Get Actor Right Vector (рисунок 9.15).

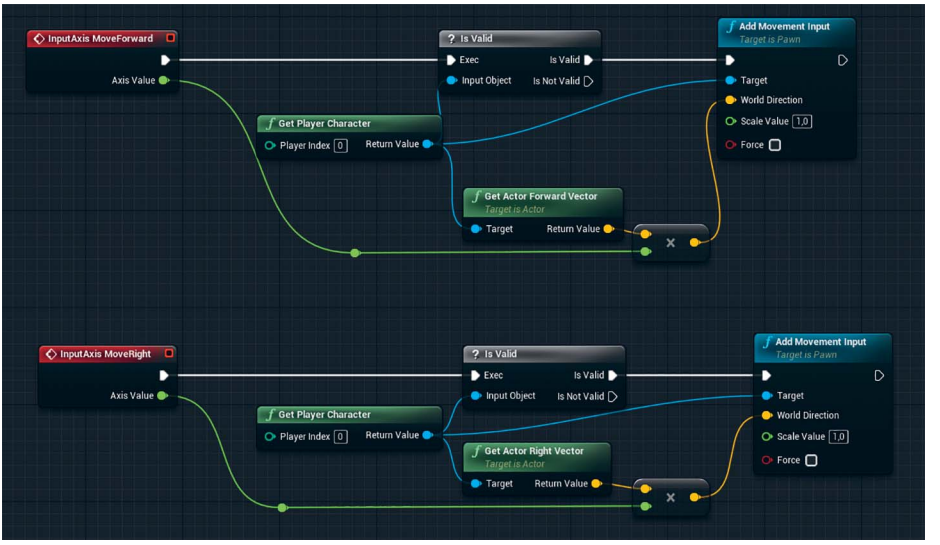


Рисунок 9.15 Завершенный сценарий movement input

Заметка

Вы можете копировать и вставлять узлы в BluePrint Editor, используя стандартные комбинации клавиш, а также контекстное меню, когда нажимаете правой кнопкой мыши по выбранным узлам в Graph Editor. Это может значительно сэкономить время при сборке похожих блоков кода.

Однако если вы обнаружите, что копируете и вставляете один и тот же код снова и снова, рассмотрите возможность создания функции для повторного использования.

Поздравляю. Это все, что нужно сделать, чтобы получить ваш Player's input для вращения PC и перемещения Player's Pawn. Теперь нужно заставить игровой мир использовать ваши Classes, когда вы нажимаете Play, и вы будете готовы к работе.

GameMode

GameMode Class — это место, в которое заглядывает движок при загрузке уровня (Level), чтобы определить, какой выбрать Player Controller, Pawn и другие необходимые классы.

Создание GameMode

Создайте новый GameMode Asset, выполнив следующие действия.

1. Нажмите Add New button в Content Browser и выберите Blueprint Class
2. Выберите **Game Mode Base** Class и назовите ваш Asset **BP_UE4Viz_Interior_Game**. Нажмите Enter для создания Asset.
3. Откройте ваш только что созданный GameMode Asset с помощью двойного нажатия в Content Browser. Вы увидите список Classes, который можете определить для GameMode. В этом случае назначьте Pawn и Player Controllers, которые вы создали (рисунок 9.16).
4. Закройте Blueprint Editor и сохраните ваш прогресс, выбрав **File** > **Save All**.

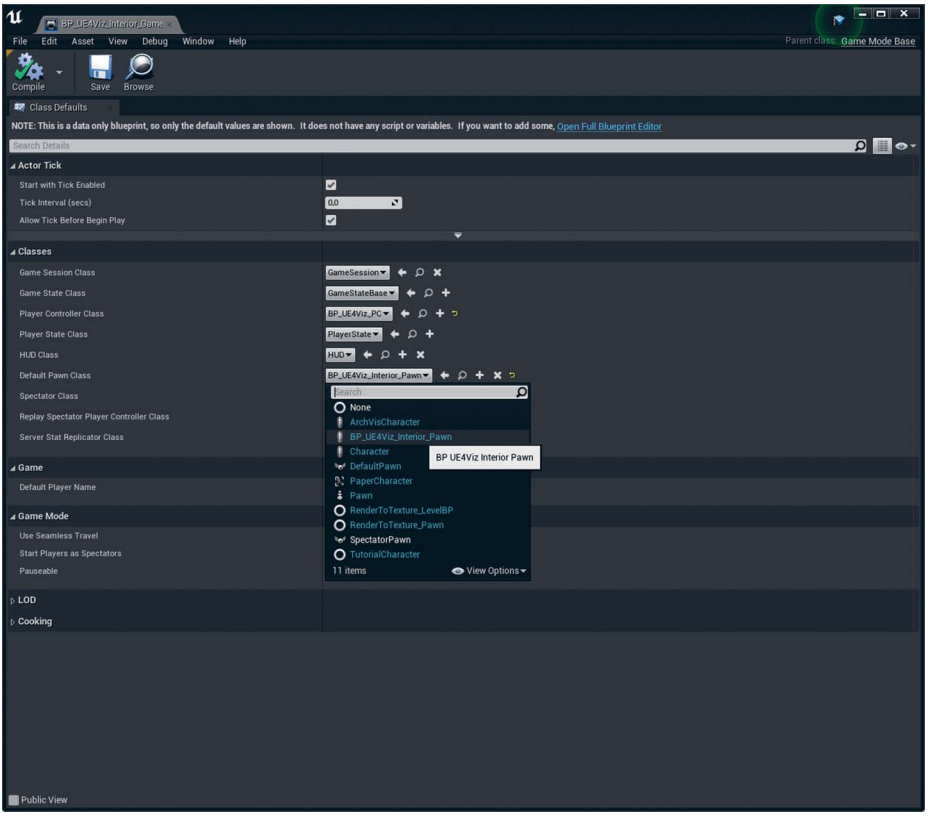


Рисунок 9.16 Назначение Pawn и Player Controller в GameMode Class

Назначение GameMode проекту

У вас есть два способа определить, какой GameMode Class применяется в UE4 Level: как для всего проекта по умолчанию, используя диалоговое окно Project Settings, или через переопределение для Level. Так как вы наверняка хотите запускать новый GameMode на любых Level, которые создадите в этом проекте, установите его как проект по умолчанию.

Откройте диалоговое окно Project Settings, выбрав **Edit** > **Project Settings** (рисунок 9.17). Edit menu доступно практически из всех окон Editor.

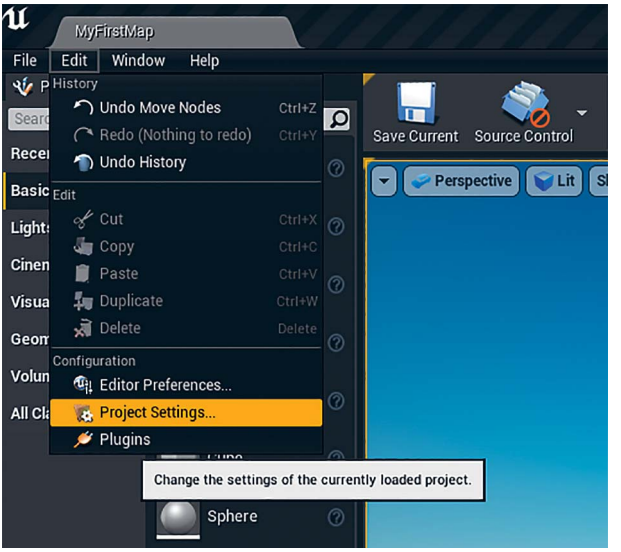


Рисунок 9.17 Переход к диалоговому окну Project Settings

Выберите Maps & Modes под заголовком Project. В разделе Default GameMode выберите только что созданный BP_UE4Viz_Game Class (рисунок 9.18).

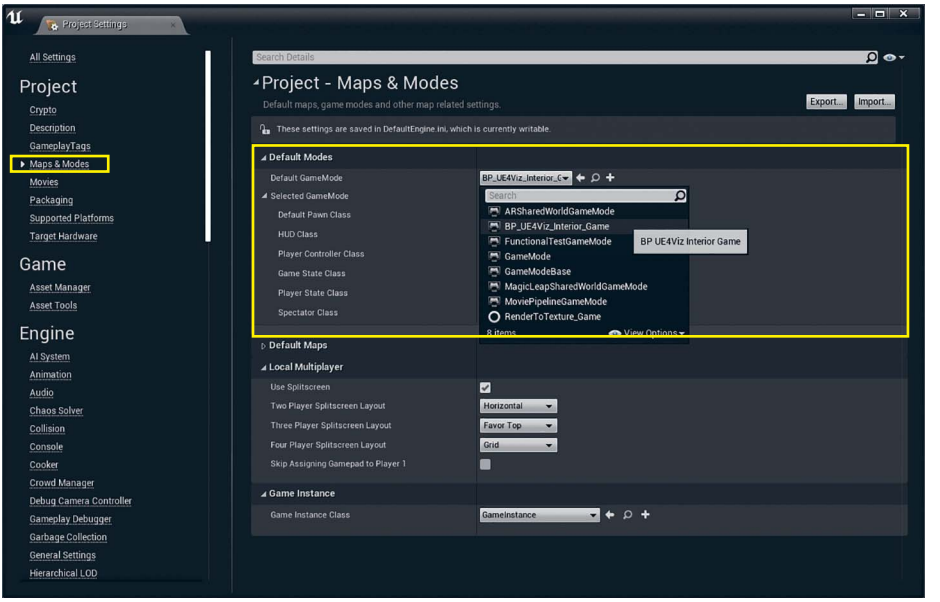


Рисунок 9.18 Назначение проекту по умолчанию BP_UE4Viz_Game Class

Размещение Player Start Actor

Есть одна вещь, которая важна и которую легко забыть, — это Player Start Actor.

Без Player Start Actor игра не будет знать, где появится ваш Pawn class, когда она загружена в качестве standalone или packaged build. Где вы in-Editor и используете PIE без Player Start, ваш Pawn появится по умолчанию в текущей позиции камеры в активном Viewport. Можно легко забыть разместить Player Start так, как ваш Level будет выглядеть рабочим, когда он действительно установлен некорректно.

Для размещения Player Start Actor перетащите его из Class Browser на панель Place Mode (нажмите Shift+1), как показано на рисунке 9.19. Синяя стрелка (3D-стрелка, которая является частью Player Start Actor, а не желтая стрелка) указывает, в каком направлении будет смотреть Player, когда появится.

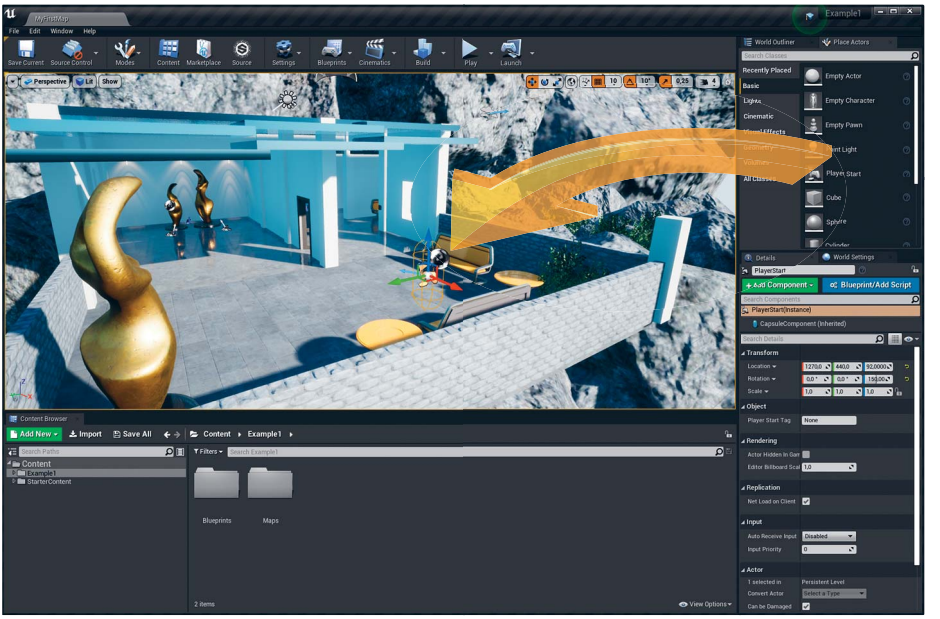


Рисунок 9.19 Поместите Player Start Actor и поверните его лицом к сцене

Теперь вы можете нажать кнопку Play и появиться в игре в качестве вашего пользовательского Character Pawn. Вы сразу же должны заметить, что привязаны к земле с помощью физики и не можете проходить сквозь стены.

Когда вы закончите ходить и захотите вернуться в Editor, вы можете просто нажать Esc на вашей клавиатуре для завершения сессии.

Убедитесь, что сохранили весь ваш прогресс.

Заключение

Поздравляю. Вы создали пользовательский UE4 GameMode с Player Controller, Pawn и пользовательским Input Mappings и настроили проект для использования новых Classes в качестве проекта по умолчанию.

Эти Classes будут служить основой для добавления новых функций и дальнейшей работы в UE4 в соответствии с потребностями вашего проекта.

ГЛАВА 10

УПАКОВКА И РАСПРОСТРАНЕНИЕ

Теперь, когда у вас есть уровень и связанные с ним объекты Pawn, Player Controller и Game Modes, пришло время подготовить его к запуску в качестве отдельного приложения. Это, как правило, называется упаковкой. После упаковки вы сможете поделиться вашим приложением, как любым другим устанавливаемым приложением. Эта глава познакомит вас с упаковкой приложения.

Упаковка против сборок из Editor

Когда вы запускаете приложение с помощью кнопки Play в Editor, вы на самом деле просто запускаете другое окно в Editor, а не отдельное приложение. Editor использует Assets и Maps из вашей папки Content и то, что уже загружено в оперативную память, чтобы заполнить мир, и код Editor запускает симуляцию. Это позволяет очень быстро запускать, тестировать и выполнять итерации, но для этого требуется установить UE4 Editor целиком.

Это явно не то, как вы хотите передавать приложение своим клиентам или общественности.

Чтобы создать приложение под конкретную платформу и облегчить его распространение, ваш контент должен быть **упакован**. Упаковка обрабатывает контент и создает отдельное приложение, которое легко распространять как любую другую игру или программу.

Вы также можете протестировать свое приложение в Standalone Mode, выбрав эту опцию в меню параметров Editor на панели инструментов. Это запустит загруженный в данный момент уровень в отдельном процессе, который больше похож на упакованную сборку, но это не так. Важно собирать и тестировать свои приложения, поскольку ошибки и несоответствия могут возникать между Editor и упакованными сборками.

Упаковка проекта

В общих чертах, упаковка берет ваш контент и код игры, оптимизирует их и создает исполняемое приложение для выбранной вами платформы.

Упакованные сборки пытаются как можно меньше загружать оперативную память. Это обеспечивает максимальную производительность и совместимость. Однако это может привести к проблемам, особенно с Level streaming и загрузкой Assets в реальном времени. Для простых проектов это не столь важно, но крупные проекты должны регулярно тестироваться с помощью упакованной сборки, чтобы избежать проблем.

Готовка контента

Готовка контента (Content Cooking) также проходит через несколько процессов: компиляции всех шейдеров, сжатия всех всех карт текстур и исправления всех редиректов в проект. Подобный подход избавляет игроков от ожидания завершения всех этих процессов во время загрузки приложения.

Готовка может занять некоторое время и серьезно загрузить CPU и RAM, поэтому запускайте ее на мощной машине. Чем больше ассетов, материалов и текстур содержит ваш проект, тем больше времени уйдет.

В завершение готовки все обработанные ассеты и классы копируются в файл **.pak**. Этот файл **.pak** предназначен для очень быстрого чтения даже на самых медленных дисках, что позволяет приложению загружаться быстро.

Развертывание

Упаковка берет готовый (cooked) контент, объединяет его с бинарными файлами движка и помещает их вместе в одну папку для распространения. Этот последний шаг создает отдельное приложение, которое можно запустить без установки редактора.

Настройки упаковки

UE4 может генерировать упакованные проекты для множества платформ и систем (iOS, Windows, Mac и т. д.). С такой гибкостью приходит несколько вариантов.

Платформы

Целевая платформа — это комбинация аппаратного (hardware)/программного обеспечения (software), для которой вы пытаетесь создать исполняемый файл проекта.

Вы можете упаковать ваш проект для множества платформ, поддерживаемых UE4. От мобильных девайсов до Windows, Mac и Linux до игровых консолей, UE4 может упаковать ваш проект для практически всех популярных устройств и операционных систем.

Каждый будет иметь собственную оптимизацию, методы ввода и другие ограничения, которые вы должны учитывать. Эта книга фокусируется только на настольных платформах (Mac и Windows).

Конфигурация сборки

Другим важным решением, которое вы должны принять, является конфигурация сборки. Два основных варианта — Shipping и Development.

Наибольшим различием между сборками Shipping и Development является доступ к инструментам отладки, введению логов и командной строке. Сборка Development позволяет это, в то время как сборка Shipping лишена этих возможностей.

Я обычно использую Development, если не планирую распространять пакет для публики в качестве широкого выпуска. Консольные команды, введение логов и отладочная информация полезны, если мне нужно отладить какую-нибудь проблему.

Сборка Shipping хороша, когда вы хотите распространять ваш пакет публично и не хотите, чтобы он имел доступ к Console.

Как запустить упаковку

Легче всего вы можете упаковать ваш проект через Editor. Просто выберите Package Project из File menu. Это меню содержит несколько настроек и обеспечивает быстрый доступ к более подробным настройкам в диалоговом окне Project Settings (рисунок 10.1).

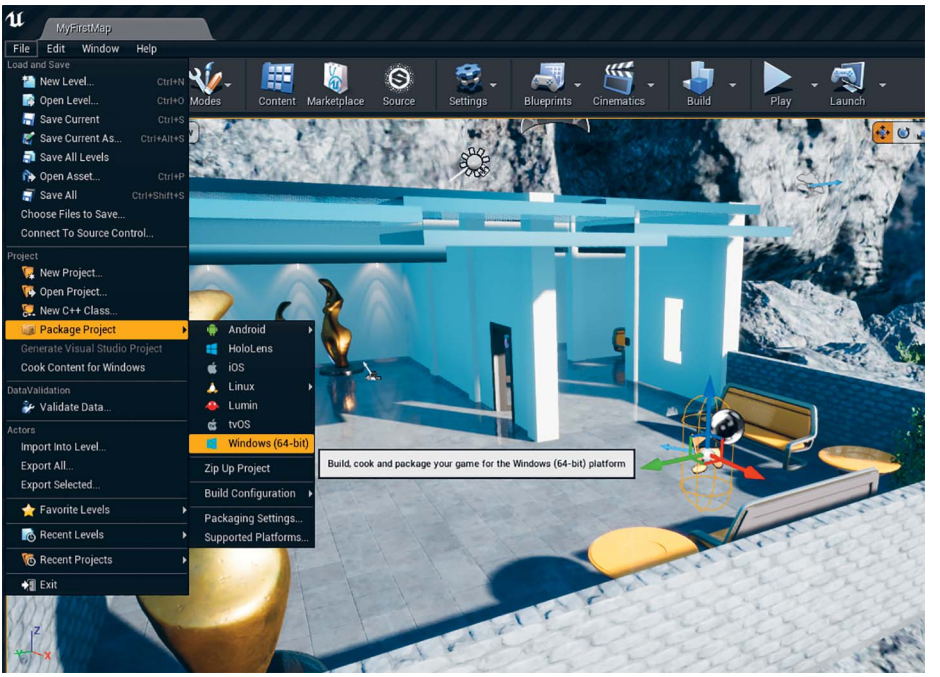


Рисунок 10.1 Меню упаковки в UE4 Editor

После выбора нужной платформы вам предложат указать место для сохранения вашего упакованного проекта. Не помещайте упакованный проект в папку вашего проекта — это может привести к путанице.

Упаковка занимает некоторое время. Появится уведомление, говорящее, что идет упаковка (рисунок 10.2). Этот процесс полностью фоновый, поэтому вы можете продолжить работу над вашим проектом, пока он обрабатывает контент в фоновом режиме.

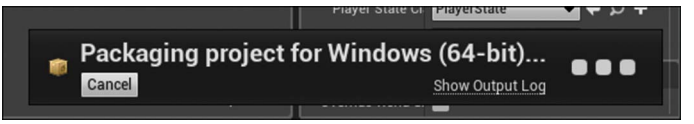


Рисунок 10.2 Уведомление об упаковке UE4 Editor

Запуск вашего приложения

После завершения упаковки Editor уведомит вас (рисунок 10.3).

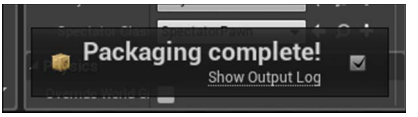
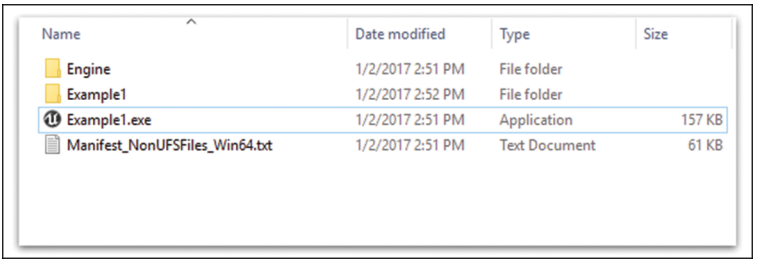


Рисунок 10.3 Уведомление о завершении упаковки в Editor

Пройдите к директории, которую вы определили ранее, и найдите папку с именем WindowsNoEditor (название может различаться на разных платформах). Вы можете переименовать эту папку.

Внутри папки вы найдете исполняемый файл и несколько папок, содержащих все данные и классы, а также облегченную бинарную версию движка (рисунок 10.4).



Name	Date modified	Type	Size
Engine	1/2/2017 2:51 PM	File folder	
Example1	1/2/2017 2:52 PM	File folder	
Example1.exe	1/2/2017 2:51 PM	Application	157 KB
Manifest_NonUFSFiles_Win64.txt	1/2/2017 2:51 PM	Text Document	61 KB

Рисунок 10.4 Упакованный проект в проводнике Windows

Можете запустить ваше приложение, просто дважды кликнув по исполняемому файлу. Приложение должно запуститься, загрузить Level, который вы создали, и вызвать соответствующий Player Controller и Pawn, предоставляя вам доступ к мыши.

Ошибки упаковки

Иногда упаковка сталкивается с ошибками. Простая ошибка компиляции Blueprint или нехватка места на диске могут остановить весь Packaging process.

Когда это происходит, вам следует обратиться к окну **Output Log**, которое в большинстве случаев может предоставить полезную информацию.

Для доступа к окну Output Log в Editor перейдите Window > Developer Tools > Output Log (рисунок 10.5). Я обычно прикрепляю это окно к моему Content Browser для быстрого доступа.

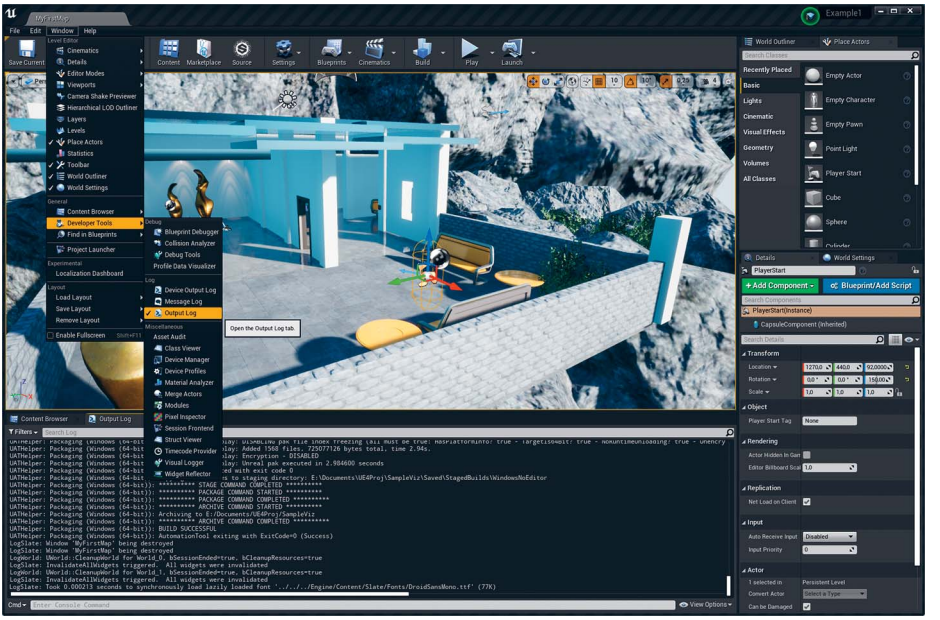


Рисунок 10.5 The Output log

Распространение проекта

Теперь вы можете копировать, архивировать и отправлять эту папку кому угодно. Это действительно просто (чаще всего).

Конечно, у вас еще есть проблемы с платформой и аппаратной совместимостью, но это в порядке вещей для любого приложения.

Некоторые компьютеры, на которых пытаются запустить ваше приложение, могут не иметь отдельных системных компонентов, которые требует UE4. Они называются **Prerequisites**. По умолчанию эти файлы включены в ваш упакованный проект. Если ваш упакованный UE4-проект обнаружит, что они не установлены, то предложит пользователю установить их, когда тот попытается запустить исполняемый файл.

Помимо пререквизитов, упакованное UE4 приложение не требует какого-либо формального процесса установки; оно может быть запущено из любого места.

Использование установщиков

UE4 не умеет встраивать установщик при упаковке. Разработка или использование установщика для приложения и платформы разработки зависит от вас. Однако, так как UE4 не требует формальной установки, предоставления файлов проекта в архиве с простыми инструкциями обычно достаточно.

Конечно, если вы хотите настроить ярлыки рабочего стола, возможности установки (install) и деустановки (uninstall), которые делает обычный установщик, то потребуется разработать его.

Хотя эти задачи выходят за рамки этого текста, их довольно просто выполнить, используя одно из множества бесплатных или коммерческих приложений для создания установщика.

Заключение

Теперь, когда вы создали в UE4 ваше первое приложение с нуля, вы должны чувствовать себя образованным. Поздравляем!

В следующих главах рассматривается более традиционный стиль визуализации с использованием конвейера данных и некоторые из наиболее важных систем для визуализации: Lightmass, Sequencer и особые Blueprints.

ЧАСТЬ 3

АРХИТЕКТУРНАЯ ВИЗУАЛИЗАЦИЯ

11 Настройка проекта

12 Процесс работы с данными

13 Наполнение сцены

14 Архитектурное освещение

15 Архитектурные материалы

16 Создание кинематики с Sequencer

17 Подготовка уровня к добавлению интерактивности

18 Более сложные Blueprints: взаимодействие с UMG

19 Дополнительный уровень Blueprints: Material Switcher

20 Финальные размышения

ГЛАВА 11

НАСТРОЙКА ПРОЕКТА

Теперь, когда у вас есть четкое представление об использовании UE4 для создания простой сцены, вы можете погрузиться в пример из реальной жизни. Визуализация интерьера важна для архитекторов, маркетологов, дизайнеров и потенциальных покупателей, это один из наиболее распространенных типов визуализаций, выполняемых в UE4. В этой части книги вы получите знания и исследуете использование Lightmass для создания высококачественного глобального освещения (global illumination (GI)), построения и рендеринга кинематографического пошагового руководства с использованием Sequencer. Затем вы погрузитесь в изучение того, как использовать Blueprints для создания динамических взаимодействий.

Объем проекта и его требования

Представьте, что вам предоставили 3D-модель архитектурного интерьера (рисунок 11.1) и вы согласились создать интерактивную визуализацию, которая позволит игроку свободно исследовать пространство от первого лица с помощью пешей прогулки. Игрок должен иметь возможность менять определенные материалы с помощью простого пользовательского интерфейса, управляемого мышью. Клиент также запросил архитектурную анимацию, которая будет разработана с использованием Sequencer и записана на видео для автономного воспроизведения и редактирования.

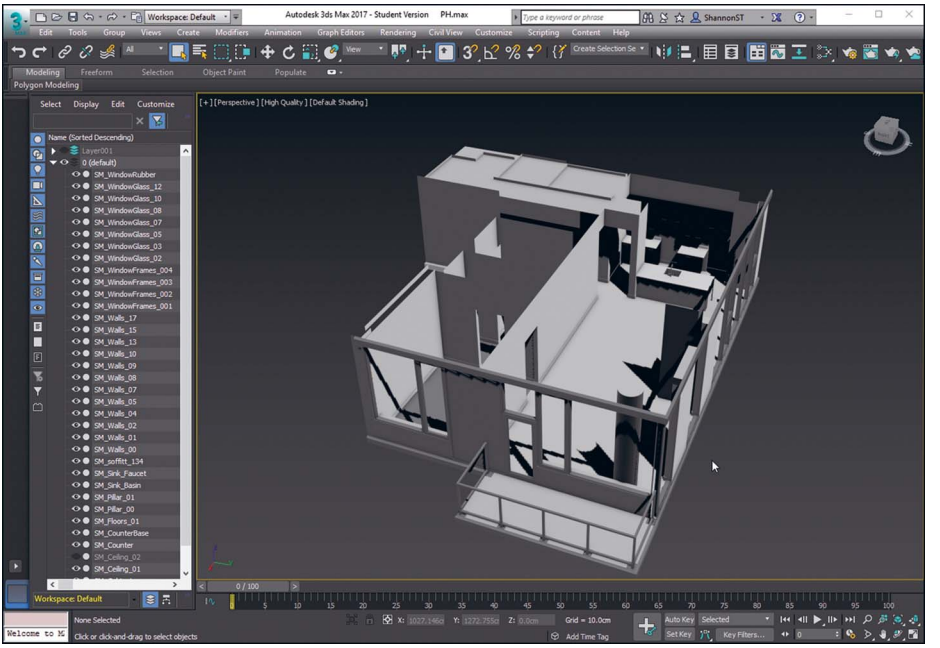


Рисунок 11.1 Предоставленная 3D-модель в 3D Studio Max

Первый шаг — подготовить 3D-модель для импорта в UE4, включая соответствующие материалы, UV-координаты и координаты Lightmap, а также сосредоточиться на создании чистой геометрии, которая импортируется на месте и легко обновляется и меняется. Я также покажу вам, как использовать автоматическую генерацию координат Lightmap UE4, чтобы сэкономить время и улучшить качество ваших сцен.

После импорта модели в UE4 пришло время начать итерационный процесс нанесения материалов и размещения света, отражающих точек, освещенных пространств и настройки постобработки, чтобы быстро получить отличные результаты.

После того как сцена готова, следующим шагом является создание анимированного пошагового руководства с помощью Sequencer. Вы узнаете, как легко создавать

привлекательные анимации и как заставить их проигрываться и прекратить использование входных данных, поступающих от игрока.

Затем происходит перенос объекта Pawn, разработанного в главе 9, в новый проект и настройка сцены, чтобы позволить игроку ходить по ней, включив столкновение и установив соответствующие акторы и параметры мира.

Завершающим этапом является создание систем переключения материалов и геометрии, которые используют Blueprints и UMG для добавления последнего слоя полировки и подготовки приложения к сборке и отправке заказчику.

Заметка

Не забывайте — вы можете скачать этот проект и все связанные с ним исходные файлы с www.TomShannon3D.com/UnrealForViz.

Вот требования к этому проекту.

1. Разработка приложения архитектурной визуализации интерьера с использованием предоставленных клиентом 3D-моделей.
2. Использование Lightmass для создания высококачественного освещения GI и стремление к фотореализму.
3. Создание заранее определенной архитектурной анимации с помощью Sequencer и использованием нескольких ракурсов и переходов камеры.
4. Объединение павна из главы 9 и подготовки сцены, чтобы игрок мог перемещаться по ней.
5. Разработка пользовательского интерфейса на основе UMG, который позволит игроку переключаться на различные версии сцены.
6. Разработка системы переключения материалов с помощью мыши.
7. Сборка проекта для отправки.

Настройка проекта

В этом разделе рассматривается создание пустого проекта из Epic Launcher и перенос объектов Player Controller, Game Mode и Pawn в новый проект вместе с необходимыми конфигурационными файлами.

Эта миграция позволяет использовать системы, созданные в главе 9, и сосредоточиться на подготовке исходных 3D и 2D данных для UE4.

Создание проекта

Используя **Epic Launcher**, запустите версию движка, которую хотите применить, и выберите вкладку **«новый проект»**, когда появится **браузер проекта**.

На этот раз вы не будете включать Starter Content, поэтому снимите флажок, но оставьте настройки **Target Hardware** на Maximum Quality.

Проект называется Example2. Сохраните его рядом с предыдущим проектом, Example1.

Заметка

Единственный способ открыть Editor — это открыть проект. Вы также можете создать новый проект из Editor, выбрав **File > New Project**.

Миграция ассетов

Вам наверняка понадобится взять ассеты, Blueprints и другую работу, выполненную в предыдущих проектах, и скопировать их в новые проекты.

UE4 опирается на концепцию ассетов, ссылающихся на другие ассеты. Например, материалы могут ссылаться на несколько текстур. На каждый материал, в свою очередь, будут ссылаться меши, к которым он применяется, и даже Material Instances, производные от него.

Из-за этого следует быть осторожным при ручном копировании содержимого между проектами, чтобы они сохраняли эти ссылки и правильную структуру папок; в противном случае ссылки перестанут работать, что может привести к ошибкам загрузки, багам или даже сбоям в вашем проекте.

Самый надежный способ копирования контента из одного проекта в другой — это использование функции **Content Migration**. Миграция гарантирует, что все ассеты, на которые ссылаются другие ассеты, которые вы хотите скопировать, будут скопированы в целевой проект.

Чтобы перенести содержимое из одного проекта в другой, откройте исходный проект (проект с файлами, которые вы хотите скопировать) в Editor. В данном случае это будет проект Example1.

После его открытия выберите ассеты, которые хотите скопировать в Content Browser. Вы можете выбрать отдельные ассеты, уровни или целые папки. В нашем случае выберите **павна**, **PC** и **GameMode** (рисунок 11.2). Щелкните правой кнопкой мыши и выберите **Asset Actions > Migrate**.

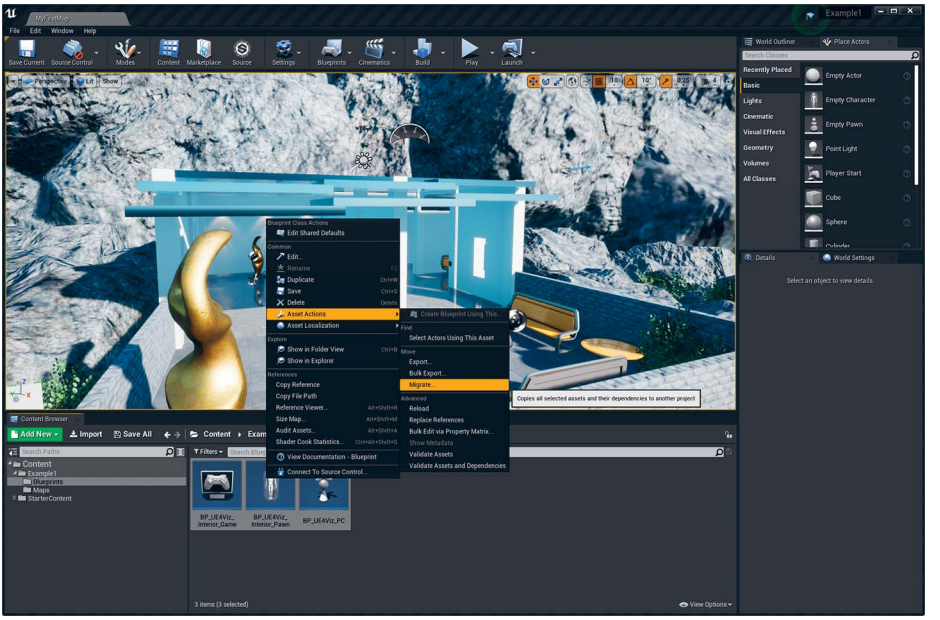


Рисунок 11.2 Миграция ассетов из проекта Example1

Откроется сводка файлов, которые будут скопированы в новый проект. Просмотрите ее и убедитесь, что у вас есть все, что вам нужно, и ничего такого, чего вы не хотите переносить (рисунок 11.3).

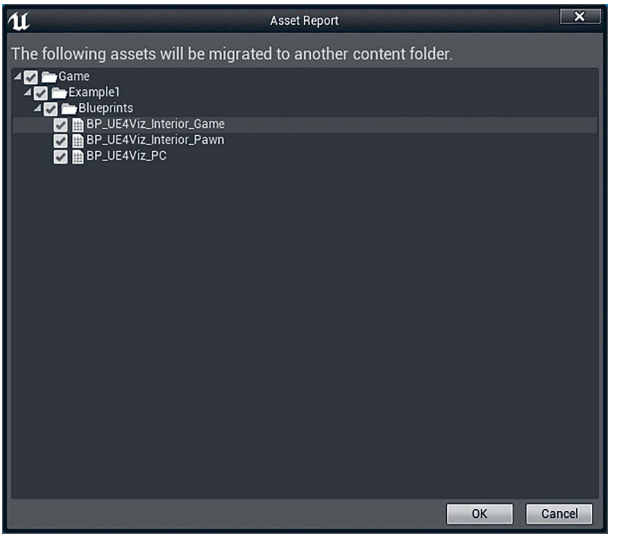


Рисунок 11.3 Список файлов, которые будут мигрированы

После подтверждения сводки появится экран с запросом выбрать каталог **Content** в рамках нового проекта. Перейдите в папку **Content** нового проекта и нажмите кнопку OK (рисунок 11.4).

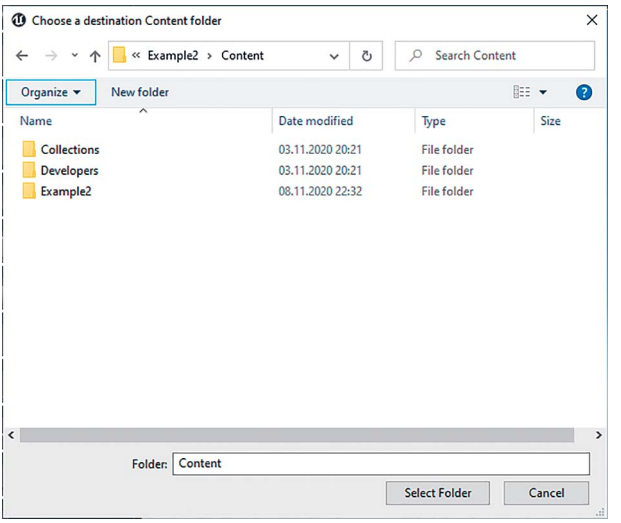


Рисунок 11.4 Указание Content каталога в новом проекте Example2

UE4 скопирует файлы и сообщит об успешном выполнении. Вы должны немедленно увидеть скопированный контент в каталоге Content нового проекта.

Не забудьте про настройку Input

Большая часть работы была сосредоточена на входных данных первого проекта, и получить их в новом проекте легко. На самом деле PC и объект Pawn не смогли бы скомпилироваться, потому что они ссылаются на input bindings, созданные в Example1.

Эти параметры хранятся в папке Config. Эти файлы не управляются из Content Browser и не копируются при использовании инструмента Content Migration, поэтому их необходимо переместить вручную.

Для этого найдите файл DefaultInput.ini в каталоге Config исходного проекта (Example1\Config\) и скопируйте его в каталог Config целевого проекта (Example2\Config\).

Копирование, переименование и перемещение ассетов

При открытии проекта Example2 обратите внимание на папку Example1, содержащую павн, PC и GameMode.

Вы хотите переместить их в новую папку, которая не зависит от проекта и ее легко найти, и сейчас самое подходящее время избавиться от папки Example1 в Content Browser. Теперь вы находитесь в Example2!

Всегда выполняйте перемещение, копирование, переименование или любые другие операции с файлами в папке Content с помощью Content Browser. Это гарантирует, что ссылки на перемещенные или переименованные ассеты сохраняются. При необходимости создается **redirector**. Это небольшие файлы размером 1–2 КБ, которые служат указателем на правильное расположение ассета.

Выполните следующие действия.

1. Создайте новую папку в Content directory под названием **UE4Viz**, а в этой папке — другую под названием **Blueprints**. Я выбрал эту папку, чтобы содержать повторно используемый код, который я мог бы переносить из проекта в проект без необходимости переименовывать или копировать что-либо.

2. Перетащите ассеты Pawn, PC и Game Mode в папку Blueprints, которую вы только что создали. UE4 спросит, хотите ли вы переместить или скопировать ассеты. Выберите Move.

3. Щелкните правой кнопкой мыши на папке Example1 и выберите **Fix Up Redirectors in Folder**.

4. Удалите старую папку Content1, щелкнув по ней правой кнопкой мыши в Content Browser и выбрав пункт Delete.

5. Выберите **Save All** на панели инструментов обозревателя содержимого, чтобы убедиться, что все ваши изменения сохранены на диске.

Работа с Project Settings

Несколько вариантов рендеринга помогут оптимизировать Engine для использования в визуализациях архитектуры. Эти сцены, как правило, гораздо менее интенсивны, чем сцены видеоигр, что позволяет вам увеличить некоторые настройки и все еще поддерживать стабильную частоту кадров.

Откройте диалоговое окно Project Settings из меню Edit и перейдите к кнопке Rendering в левом столбце (рисунок 11.5). Измените настройки, как показано на рисунке и описано в следующем списке.

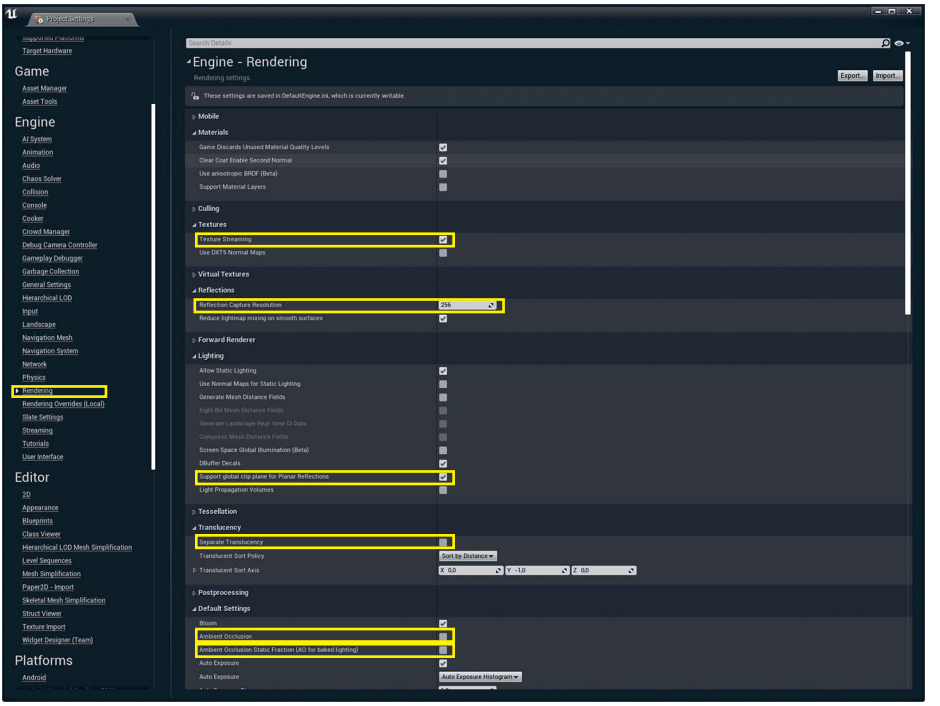


Рисунок 11.5 Настройки рендеринга проекта

1. **Disable Texture Streaming:** этот параметр предназначен для включения больших игровых миров в ограниченной памяти на консолях и более старом оборудовании ПК. Это может привести к ошибкам отображения света при использовании архитектурных мешей и Lightmaps высокого разрешения Disabling Texture Streaming заставляет все текстуры, используемые на уровне, загружаться в оперативную память сразу. Используйте эту настройку с некоторой осторожностью. Потоковая передача текстур является важной оптимизацией для сложных сцен, и ее отключение может привести к проблемам с производительностью.

2. **Set Reflection Capture Resolution:** чтобы получить самые качественные отражения, вы можете увеличить разрешение Reflection Capture Actors, которые размещаете на своих сценах. Будьте осторожны, устанавливая его слишком большим: это может съесть много памяти, если у вас много отражений с высоким разрешением. Я не рекомендую делать значение более 512 для этого параметра, если у вас только не одно или два отражения на каждом из уровней.

3. **Turn on Support Global Clip Plane for Planar Reflections:** зеркала и отражения являются основным элементом визуализации, и UE4 поддерживает зеркальные Planar Reflections. Это очень дорогая настройка, и вы должны включить ее только

в том случае, если она вам абсолютно необходима. Даже если отражения не отображаются на экране, они оказывают значительное влияние на производительность рендеринга в масштабах всего проекта.

Вы можете смягчить это воздействие с помощью панели Details, чтобы отключить множество более тяжелых функций, таких как некоторые настройки постобработки, например Screen-space Ambient Occlusion и Reflections. Вы даже можете ограничить видимых акторов или расстояние, на котором будет отображаться отраженная сцена.

4. **Turn off Separate Translucency:** Separate Translucency — это повышение производительности для игр с большим количеством эффектов альфа-прозрачности, таких как искры и взрывы. Эти эффекты могут быть очень дорогими для применения таких эффектов, как Depth of Field, поэтому эти эффекты визуализируются отдельно и игнорируются постобработкой. Вам не нужно, потому что вы наверняка хотите, чтобы полупрозрачные поверхности вели себя как можно реалистичнее.

   Эта настройка не влияет на материалы Masked, поэтому такие вещи, как растительность, обычно не затрагиваются.

5. **Turn off Ambient Occlusion and Ambient Occlusion Static Fraction:** эти параметры относятся к Screen-space Ambient Occlusion (SSAO) и эффектам постобработки. SSAO — это отличный способ добавить детали к динамически освещенным сценам и объектам. Поскольку ваша сцена почти полностью статична и вы используете высококачественные настройки Lightmass, вы можете отключить эту настройку для значительного повышения производительности, особенно при более высоких разрешениях.

   Эти настройки влияют только на эффект SSAO и не влияют на AO, рассчитанный по Lightmass. Static Fraction контролирует, насколько эффект SSAO сочетается с запеченными световыми картами, а также имеет накладные расходы на производительность, которые можно уменьшить, отключив эффект.

Настройки проекта автоматически записываются в файл проекта DefaultEngine.ini, и нет необходимости сохранять его перед закрытием окна настроек. Однако, поскольку мы изменили такие настройки, как Planar Reflections, вам нужно будет перезапустить Editor, чтобы эти настройки можно было инициализировать. Есть вероятность, что при повторном открытии Editor вы столкнетесь с перекомпиляцией шейдеров, которая может занять некоторое время, но должна произойти только один раз.

Заключение

Теперь ваш проект готов к запуску. Включение некоторых расширенных функций рендеринга UE4 и отключение отдельных ненужных функций улучшает качество изображения и позволяет избежать проблем с потоковой передачей текстур, SSAO и полупрозрачными материалами.

Вы также узнали, как использовать функцию Content Migration UE4 для перемещения готовой работы из одного проекта в другой, гарантируя, что вам не придется заново изобретать колесо для каждого проекта. Входные настройки также были скопированы через .ini-файл.

Теперь вы готовы начать учиться освещать свой мир, применять материалы и использовать Blueprints и Unreal Motion Graphics (UMG), чтобы позволить игроку вносить интерактивные изменения.

ГЛАВА 12

ПРОЦЕСС РАБОТЫ С ДАННЫМИ

Получение точных архитектурных данных в UE4 на первый взгляд может показаться сложной задачей и противоречить вашим рабочим процессам. Но с помощью нескольких стратегий и понимания всего процесса работы вы сможете легко получить любой тип данных в UE4. Вы будете импортировать массивные наборы данных в UE4 без труда и в кратчайшие сроки.

Организация сцены

Для этого проекта вы потратите много времени вне UE4, внутри ваших приложений DCC, создавая 3D-модели и текстуры, которые будут импортированы в UE4.

Сценарий: данные, предоставленные вашим клиентом, оказались хорошего качества. Их можно использовать почти «из коробки», но вы все равно хотите потратить время на подготовку контента, чтобы он импортировался в UE4 как можно проще и надежнее.

Начните с организации сцены в вашем 3D-приложении. САПР и инженерные приложения часто не генерируют наиболее элегантно названные или организованные сцены. Часто специфическая логика используется для именования объектов, которые трудно читать, или они несовместимы с UE4 или конвейером FBX. Длинные или содержащие сложные символы имена могут вызвать проблемы с экспортером FBX и UE4.

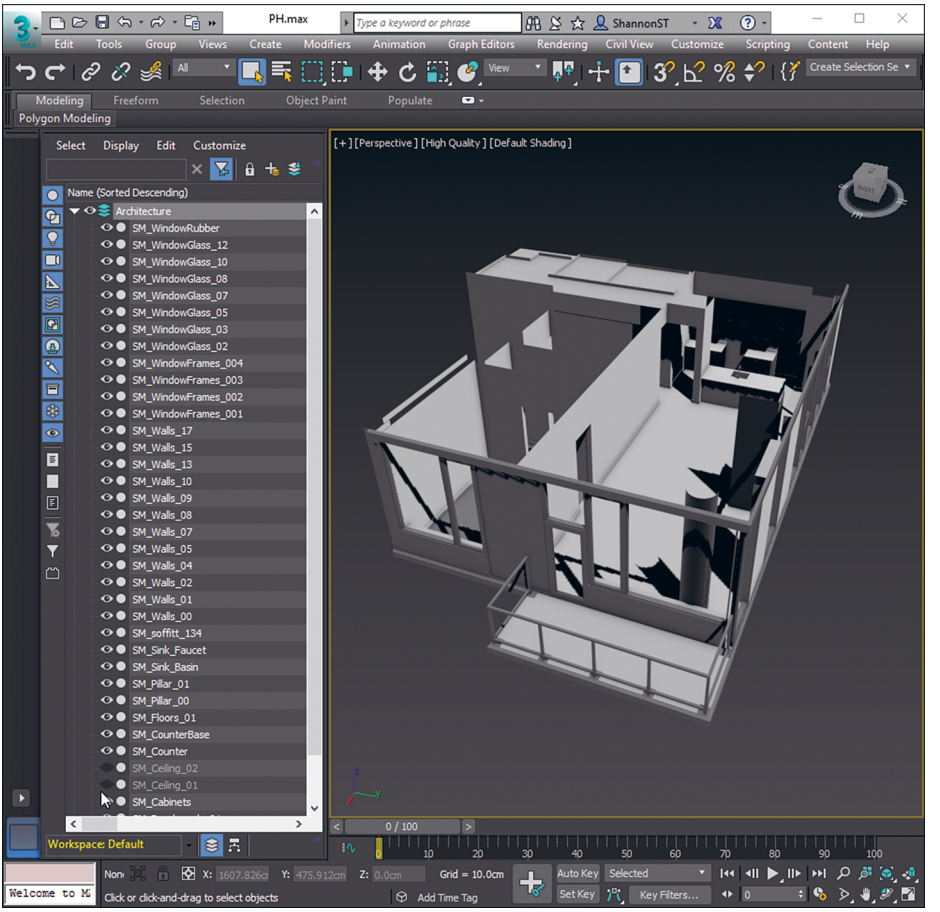


Рисунок 12.1 Организация 3D-сцены в 3D Studio Max

Определение наилучшей организации для вашего конкретного контента и адаптация ваших инструментов и процессов к потребностям конвейера FBX зависят только от вас.

В этом случае вы можете использовать простой сценарий переименования, чтобы упростить имена объектов на сцене и привести их в соответствие с принятой схемой именования UE4, в которой префикс SM_ ставится перед статическими мешами (рисунок 12.1).

Теперь это делается быстро и легко по сравнению с попыткой переименовать все, что было экспортировано, или в UE4, где переименование гораздо более важно.

Вы также можете отсортировать свою сцену по слоям в зависимости от типа модели, наиболее распространенными типами которой являются реквизит (Props) и архитектура (Architecture). Это позволяет легко выбирать слои для экспорта или управлять видимостью объекта на сцене.

Материалы

Хотя UE4 не поддерживает полный импорт/экспорт материалов из 3D-приложений, он поддерживает импорт основных функций материала, таких как цветные и Texture Maps. Импорт даже основных материалов экономит много времени на первоначальной настройке сцены.

UE4 также поддерживает назначение атрибутов вашего материала определенным параметрам в указанном родительском материале. Вместо создания материала для каждого импортированного материала делается Material Instance Asset и настраиваются его параметры. Это отличный рабочий процесс для продвинутых пользователей, и я призываю вас вернуться к нему, когда вы захотите повысить эффективность своих рабочих процессов.

Обратите внимание, что UE4 поддерживает только стандартные материалы для приложений. Материалы из пользовательских рендеров, таких как V-Ray, могут вообще не импортироваться или импортироваться плохо.

В этом случае в исходной модели используются Autodesk Architectural Materials, которые не придерживаются текстурных форматов и входных данных, поддерживаемых UE4. Я использовал простой скрипт, который заменял каждый из них стандартным материалом 3D Studio Max с тем же именем, отбрасывая все остальные параметры и назначая простые растровые изображения диффузным контактам входа, когда это было возможно.

Multi-subobject (MSO) Materials также поддерживаются UE4, но мешу нужен как Material ID, назначенный граням, так и материал MSO с уникальным материалом, назначенным каждому ID. Например, меш с четырьмя Material ID и только одним материалом, назначенным в Max, будет импортироваться как меш только с одним Material ID.

Вы также должны найти время, чтобы переименовать свои материалы и текстуры в соответствии с вашим соглашением об именовании. Опять же, это гораздо проще сделать в вашем 3D-приложении, чем в UE4.

Архитектура и арматура

Сначала вам стоит получить основные Architectural Meshes — стены, полы, потолки, светильники и технику — импортированные в UE4 и помещенные на нашу сцену. Эти меши уникальны и не нуждаются в перемещении. Также важно, чтобы они располагались точно в том же месте, что и исходные данные.

Обеспечение чистой геометрии

Чистое освещение требует чистой, хорошо подогнанной геометрии. Мерцания, пятна и ошибки освещения деморализуют, поскольку требуют повторной сборки после импорта геометрии и уже проведенной длительной сборки света.

Два самых больших виновника — плохие нормали у граней (перевернутые грани) и совпадающие или сложенные грани. И то и другое вызывает мерцание и плохое освещение. Lightmass не может правильно рассчитать отраженный свет от этих граней и часто возвращает черный цвет.

Чтобы найти перевернутые нормали, отключите backface culling или double-sided Materials в вашем 3D-приложении. Некоторые приложения также предлагают инструменты визуализации, которые помогут вам искать перевернутые грани.

Найти совпадающие грани может быть сложнее. Ради производительности UE4 использует 16-битный буфер глубины, который более склонен к мерцанию, когда грани расположены более близко, чем вы могли бы использовать с рендерингом трассировки лучей. Эта модель имела много перекрывающихся граней, которые требовали значительной очистки. Многие ошибки не были очевидны до тех пор, пока их не импортировали в UE4 и не увидели там мерцание. Будьте терпеливы, вы получите их все. Обеспечение хорошей итеративной настройки рабочего процесса делает устранение подобных проблем гораздо менее болезненным.

Вы также наверняка захотите избежать односторонних мешей. Свет будет проходить через заднюю сторону и вызывать утечки света и другие ошибки рендеринга. В этой сцене я должен был создать внешние стены, а также потолочные и напольные сооружения, чтобы обеспечить хорошее освещение.

Использование хороших UV-координат

Наличие хороших UV-координат очень важно в UE4. Плохие UV-координаты не только затрудняют нанесение материалов и приводят к плохому внешнему виду, но и могут негативно повлиять на качество рендеринга объектов.

Базовый канал (Channel 0)

Канал 0, базовый UV-канал, является каналом по умолчанию для применения текстур к мешам.

Предоставленная модель содержит меши с хорошими UV-координатами, но выглядит как плитка, повторяясь каждый 1 см (данные CAD часто будут сопоставляться с реальными единицами измерения). Хотя вы можете масштабировать свои текстуры в материалах UE4 для компенсации, я предпочитаю этого не делать, потому что это затрудняет чтение меша для предварительного просмотра — текстура растягивается в большом масштабе, необходимом для компенсации.

Я предпочитаю масштабировать свои UV-координаты примерно до 1 м. Я делаю это либо с помощью box UV modifier (рисунок 12.2), либо, если уже есть реальные масштабируемые координаты, подобные тем, что в этой модели, я использую модификатор UV map scaler для масштабирования координат до более удобного масштаба UE4.

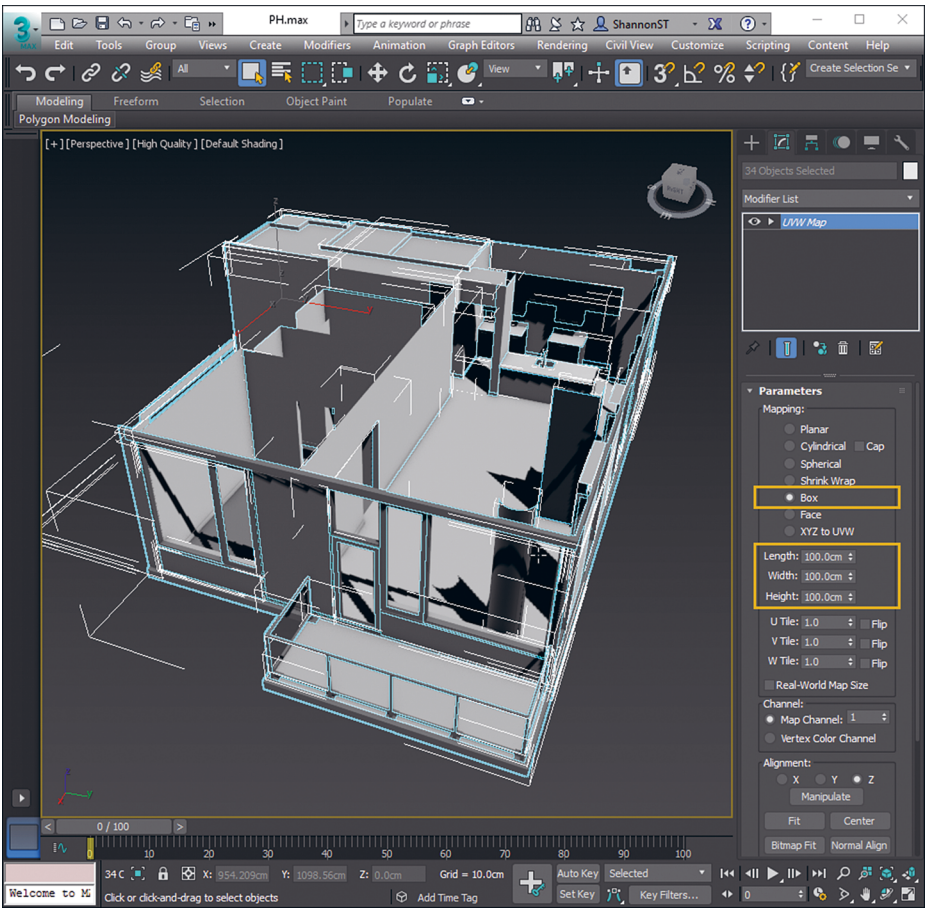


Рисунок 12.2 Применение одного экземпляра Box UVW map modifier ко всем стенам, полам и потолкам

Lightmass (Channel 1)

Lightmass требует чистых, неискаженных UV-координат, чтобы быть эффективным. Вы можете либо повторно упаковать свои координаты в 3D-приложении, либо положиться на систему переупаковки UE4.

Я обычно опираюсь на Automatic Lightmap Coordinate Packing UE4, гарантируя, что мои модели имеют хорошие базовые координаты канала UV перед экспортом. Более сложные меши могут потребовать ручного редактирования этих координат для достижения наилучших результатов, но большинство мешей работают очень хорошо.

Эта система не разделяет края, не применяет и не меняет ваши UV-диаграммы каким-либо образом. Она просто берет UV-данные канала 0 и переупаковывает их.

Для сцен, которые не имеют UV-координат, или когда вы моделируете собственные сцены, вы все еще можете использовать автоматическую упаковку в UE4, но вам нужно убедиться, что ваши базовые UV-координаты чисты, без UV-искажений и имеют хорошо расположенные границы.

Большинство стен, полов и потолков можно отобразить с помощью a box-style mapping. Конечно, неквадратные грани могут потребовать дополнительного внимания, но если ваши базовые Texture Maps выглядят хорошо, UE4 должен быть в состоянии сделать хороший набор координат Lightmap для вас.

Экспортирование сцены

Теперь, когда ваши меши подготовлены, пришло время импортировать их в UE4. Для этого сначала нужно сохранить их в файл формата UE4, который можно импортировать. Для статических мешей предпочтительным форматом файла является FBX.

Для экспорта вашей модели в FBX доступно несколько методов. У каждого из них свои преимущества и недостатки, и каждый может работать лучше для вашей ситуации или сцены.

Использование нескольких FBX-файлов

Я предпочитаю экспортировать объекты на сцене в отдельные файлы FBX. Это позволяет мне максимально контролировать данные, и я могу легко и надежно обновлять, экспортировать и повторно импортировать свои данные.

Большинство 3D-приложений не поддерживают пакетный экспорт файлов FBX, что вынуждает вас утомительно экспортировать их один за другим. Из-за этого я разработал сценарий Max, который позволяет мне быстро сделать это (рисунок 12.3). Вы можете скачать его по адресу www.TomShannon3D.com/UnrealForViz.

Важно отметить, что UE4 использует имя файла FBX в качестве имени импортированного меша и игнорирует имя, назначенное в вашем 3D-приложении.

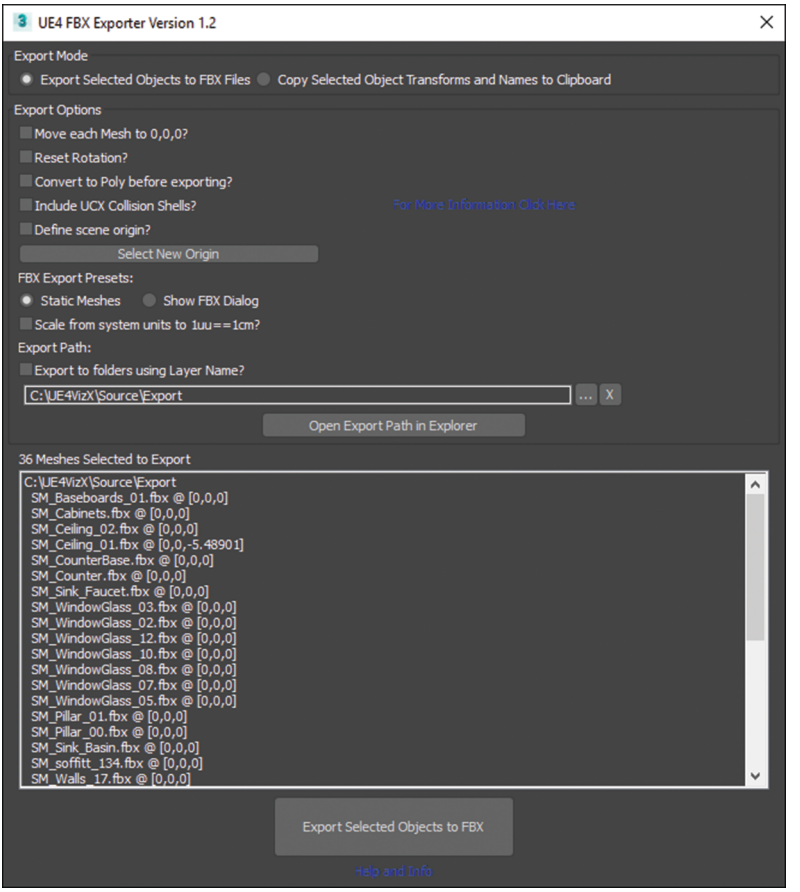


Рисунок 12.3 Скрипт для Max используется для пакетного экспорта файлов FBX для UE4; для получения дополнительной информации перейдите по ссылке www.TomShannon3D.com/UnrealForViz

Использование одного FBX-файла

Вы также можете экспортировать всю свою геометрию в один файл FBX. UE4 может импортировать файл FBX с несколькими объектами в нем разными способами. Параметры варьируются от создания единого монолитного статического меша до импорта каждого объекта в виде статического меша в файл по отдельности с помощью Content Browser. UE4 также предлагает функцию **Import to Level** в меню File, которая позволяет автоматизировать импорт сложных иерархических моделей в комплекте с камерами, подсветкой и анимацией.

Эти опции очень просты в использовании, и это отличный способ быстро импортировать большие наборы данных в UE4.

Самый большой недостаток, с которым я столкнулся с этими методами, — это трудности с обновлением конкретных мешей или наборов мешей из файла FBX. UE4 ожидает, что структура файла FBX останется неизменной между реимпортами, поэтому реимпорт статических мешей легко вызовет ошибки, если вы не будете очень осторожны с экспортом и не станете поддерживать строгий набор экспортируемых объектов для каждого файла FBX.

Импорт сцены

В зависимости от того, как вы экспортировали свои данные и как хотите управлять ими на собственной сцене, у вас есть несколько вариантов, когда вы импортируете свой контент в UE4.

Импортирование в Content Browser

Если вы экспортировали содержимое в виде отдельных файлов FBX, вы можете импортировать их по одному или в пакетном режиме, перетаскивая файлы FBX из системных файлов (рисунок 12.4) или используя кнопку **Import** в Content Browser.

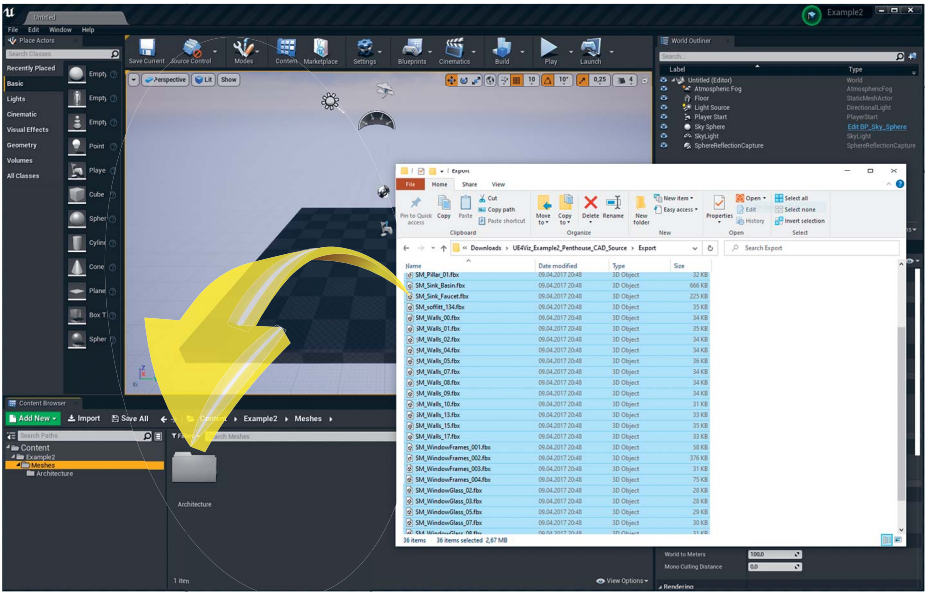


Рисунок 12.4 Импорт нескольких файлов FBX путем перетаскивания из Windows Explorer в Content Browser

Если вы экспортировали ваш контент в один FBX-файл с большим количеством мешей, вы можете импортировать эти файлы в Content Browser как один совмещенный

статический меш или в виде отдельных мешей, выбрав **Combine Meshes** в диалоге FBX Import Options, который появляется при попытке импортировать FBX ассет.

Как видно на рисунке 12.5, этот диалог предлагает множество опций, большинство из которых вы можете оставить по умолчанию, за несколькими заметными исключениями.

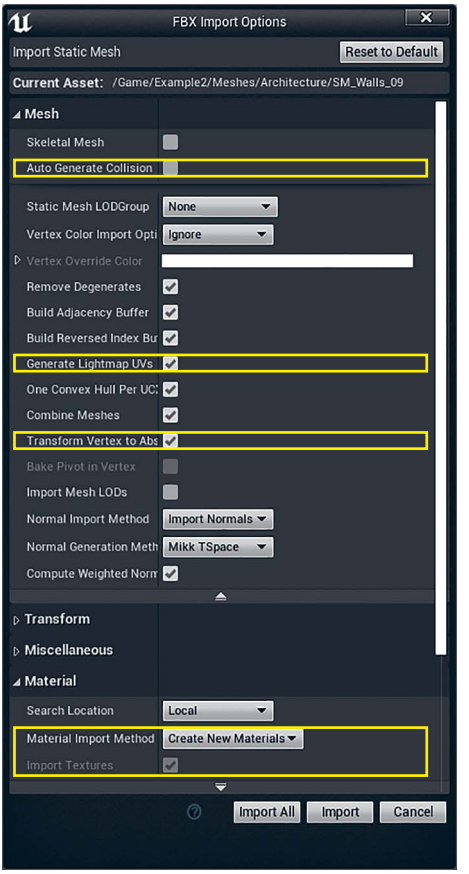


Рисунок 12.5 Диалоговое окно FBX Import Options. Auto Generate Collision отключена, в то время как Generate Lightmap UVs включена, как и параметры Material and Texture import

Auto Generate Collision

Опция Auto Generate Collision создает очень упрощенное поле столкновений вокруг вашего меша и обычно не подходит для визуализации данных. Поскольку эта сцена проста, вы можете положиться на столкновение по полигонам. Это дороже, чем использование примитива коллизии с низким количеством полигонов, и может вызвать проблемы с производительностью в более сложных сценах. Для этого проекта вы должны установить значение **false**.

Generate Lightmap UVs

Вы уверены, что ваши меши имеют хорошие, чистые UV-координаты, поэтому система Automatic Lightmap Generation должна давать хорошие результаты. Установите для параметра Generate Lightmap UVs значение **true**.

Transform Vertex to Absolute

Transform Vertex to Absolute позволяет вам решить, использует ли UE4 pivot point, созданную в вашем 3D-приложении, или она будет установлена в значение 0,0,0. Для ваших архитектурных мешей установите этот параметр в значение **true**.

Импорт материалов и текстур

Установите значение **true**, чтобы импортировать применяемые материалы и текстуры вместе с моделью.

Importing into the Level

Другой вариант — это новый **Import into Level**. Это позволяет импортировать содержимое файла FBX, включая связанные иерархии, анимацию, освещение и камеры.

В UE4 выберите **File > Import into Level** и выберите файл FBX (рисунок 12.6). Выберите папку для импорта (рисунок 12.7).

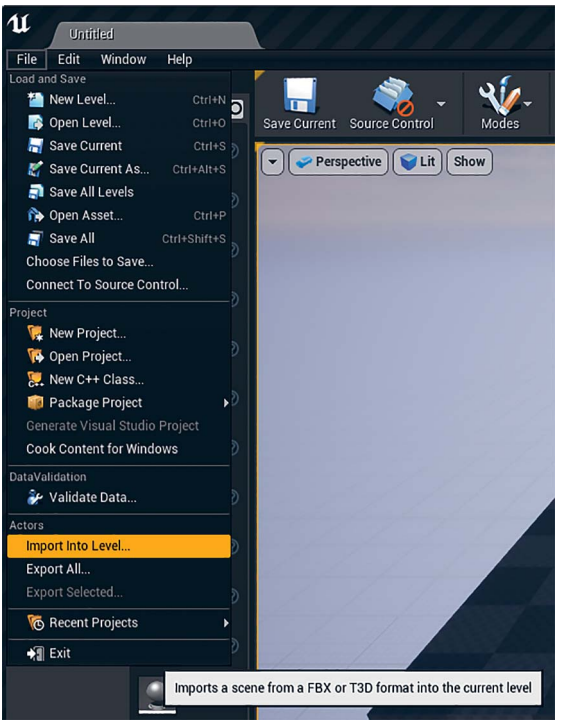


Рисунок 12.6 Импорт сцены FBX непосредственно на уровень

Появится диалоговое окно импорт / реимпорт (рисунок 12.8), позволяющее предварительно просмотреть меши и материалы, которые будут импортированы. Это хорошее время, чтобы быстро взглянуть и убедиться, что вы импортируете то, что хотите.

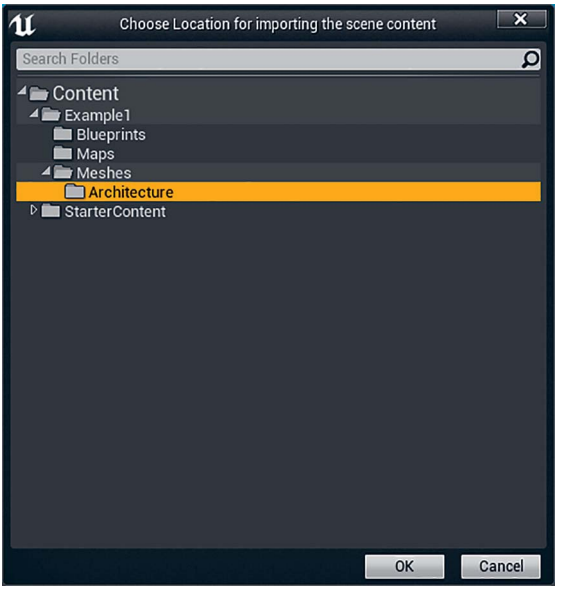


Рисунок 12.7 Выбор целевой папки

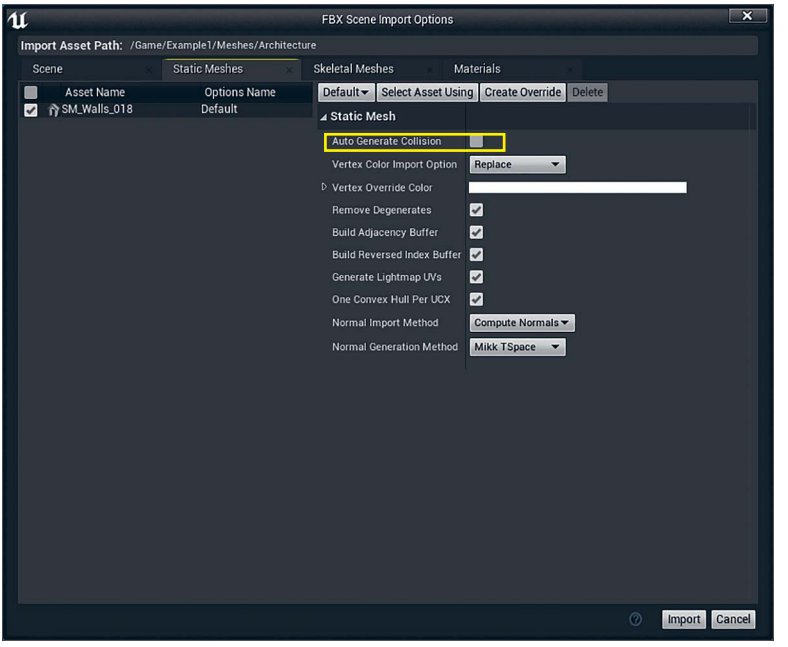


Рисунок 12.8 Диалоговое окно FBX Scene Import Options

Обязательно отключите функцию **Auto Generate Collision**, поскольку она несовместима с мешами в визуализации архитектуры. Вы будете использовать per-poly collision на своих стенах, а не полагаться на прокси-объекты коллизии.

Каждый меш импортируется как единое целое в файл **.uasset** вместе с любыми материалами и текстурами, примененными к этим материалам. Вы также можете определить другую папку для импорта ваших материалов.

Создается один Blueprint Actor и помещается в загруженную сцену со ссылками на каждый меш. Этот Blueprint работает в сочетании с FBX Import Data Asset, позволяющим повторно импортировать файл FBX, а также добавлять, удалять и изменять меши и материалы в соответствии с файлом FBX при повторном импорте.

После импорта не забудьте сохранить свою работу, потому что импортированные меши не сохраняются на диске до тех пор, пока вы не сохраните их сами.

Декоративные меши (реквизит)

Стулья, стаканы и тарелки — это все, что я считаю **декоративными мешами**, или **реквизитом**: меши, которые дублируются и/или должны быть перемещены или помещены в Editor.

В модели, представленной клиентом в примере этой главы, отсутствовала какая-либо мебель, но вам были даны рекомендации и справочные материалы. Оттуда вы строили 3D-модели специально для UE4 или находили меши в своей библиотеке контента, которые можно было импортировать.

Заметка

Часто вам дают наборы данных с реквизитами в определенных местах. Их могут быть сотни. Пример — уличные фонари в генеральном плане или растительные меши в ландшафтном плане.

Этот сценарий является общим для визуализации, которая не имеет очевидного решения рабочего процесса в UE4. Я написал скрипт 3DS Max, который копирует местоположение мешей в буфер обмена для копирования и вставки в UE4. Вы можете скачать его с сопутствующего сайта этой книги по адресу www.TomShannon3D.com/UnrealForViz.

Лучше всего разместить свой реквизит в UE4, а не в 3D-приложении. FBX не поддерживает инстанцирование и будет рассматривать каждую тарелку, стакан и стул как уникальный объект, импортируя каждый из них как уникальный **.uasset**, каждый из которых занимает память, дисковое пространство и вычислительную мощность. Это также

означает, что вы должны обновить каждый экземпляр объекта, если есть что-то, что нужно изменить.

Когда у вас есть много повторяющихся объектов на сцене, имея лишь один **.uasset** можно использовать его с большим количеством ссылок на этот единственный файл ассета на уровнях.

Кроме того, как будет показано далее, создавая хорошие pivot points, вы можете сделать изменение, перемещение и даже анимацию ваших мешей намного проще, чем их явное размещение.

Установка точек вращения

Вы должны создавать реквизит иначе, чем архитектурные меши, с самой большой разницей в точках вращения (пивотах). Хотя архитектурные меши экспортируются и импортируются на месте, с их пивотами, установленными на 0,0,0 при импорте, опорные меши должны быть импортированы с хорошо авторизованными пивотами.

Пивоты не могут быть отредактированы непосредственно в UE4, поэтому настройка точек разворота в 3D-приложении очень важна.

Поместите свой пивот (точку вращения) туда, где вы хотите, чтобы ваш меш прикрепился к миру. Создание правильного pivot point может сэкономить огромное количество усилий, когда речь заходит о размещении, перемещении и даже анимации ваших мешей в UE4.

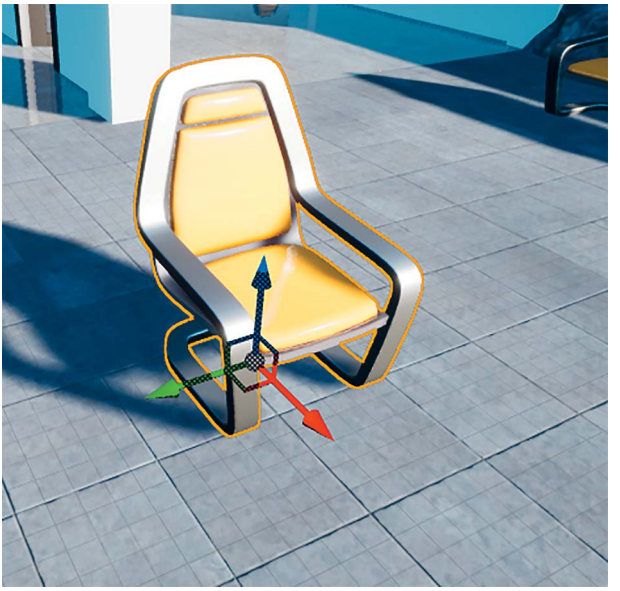


Рисунок 12.9 Pivot point объекта в UE4, где красный — Roll или X, зеленый — Pitch или Y, а синий — Yaw или вращение Z

Наличие пивота в центре объекта, как правило, не лучшее место. Пивот — это место, где актор будет масштабироваться, вращаться и двигаться в UE4. Это также место, где он будет привязываться к поверхностям других акторов, когда вы поместите его на уровень. Для стульев и другой мебели это означает, что вы должны разместить ось на высоте, где мебель ударяется о землю. Вы должны установить у картины в рамке pivot сзади, где она будет прикреплена к стене.

UE4 относится к вращениям как комбинации осей Roll, Pitch и Yaw. Крен — это вращение влево и вправо вокруг оси X, а тангаж — вверх и вниз вокруг оси Y; рыскание вращает актора вокруг оси Z (рисунок 12.9).

Это означает, что вы должны моделировать акторов в своем 3D-приложении, обращенных к положительному X, точно так же, как они появляются в UE4. UE4 будет управлять переводом оси Y.

Использование осмысленной геометрии

Реквизит, предоставленный клиентом, является отличной отправной точкой, но он часто бывает недостаточно хорош, чтобы использоваться для визуализации высокого класса.

Я использовал модели, предоставленные вместе с руководством от клиента, в качестве эталона и переделал большую часть мебели. Вы видите, что большинство моделей очень просто моделируются. Хотя современные видеоигры используют очень трудоемкий рабочий процесс для мешей, персонажей и окружающей среды, проекты визуализации обычно не имеют времени, бюджета или потребности в таком уровне внимания, чтобы тратить его на каждый ассет.

Я сосредоточился на том, чтобы сделать геометрию чистой, хорошо UV-преобразованной и довольно низкополигональной. UE4 и современное оборудование позволяет обрабатывать миллионы треугольников в секунду, и большинство сцен визуализации относительно редки по сравнению с окружением видеоигр. Это позволяет вам использовать больше полигонов для определения ваших мешей, но будьте внимательны, тотальная оптимизация все еще важна.

Экспорт

Хотя вы можете экспортировать один **.FBX** со всеми реквизитами в одном файле, я не рекомендую это делать. Это неэффективно и может затруднить рабочий процесс, так как нужно повторно экспортировать *все* одинаковые меши в файл FBX, если нужно внести изменения в один меш.

Используйте следующие настройки или аналогичные настройки в экспортере вашего 3D-приложения (рисунок 12.10).

1. Убедитесь, что вы экспортируете Smoothing Groups, Tangents и Binormals, а также что Triangulate и Preserve Edge Orientation установлены на true.

2. Если вы работаете на сцене, которая не масштабируется до сантиметров, вы также можете масштабировать данные с помощью системы масштабирования единиц измерения экспортера. Для этого установите **Automatic** переключатель в положение **false** и установите параметр **Scene units converted to** в **Centimeters**.

Вы должны экспортировать каждый реквизит как один файл FBX, называя каждый файл FBX так, как вы хотите. UE4 использует имя файла для установки имени меша.

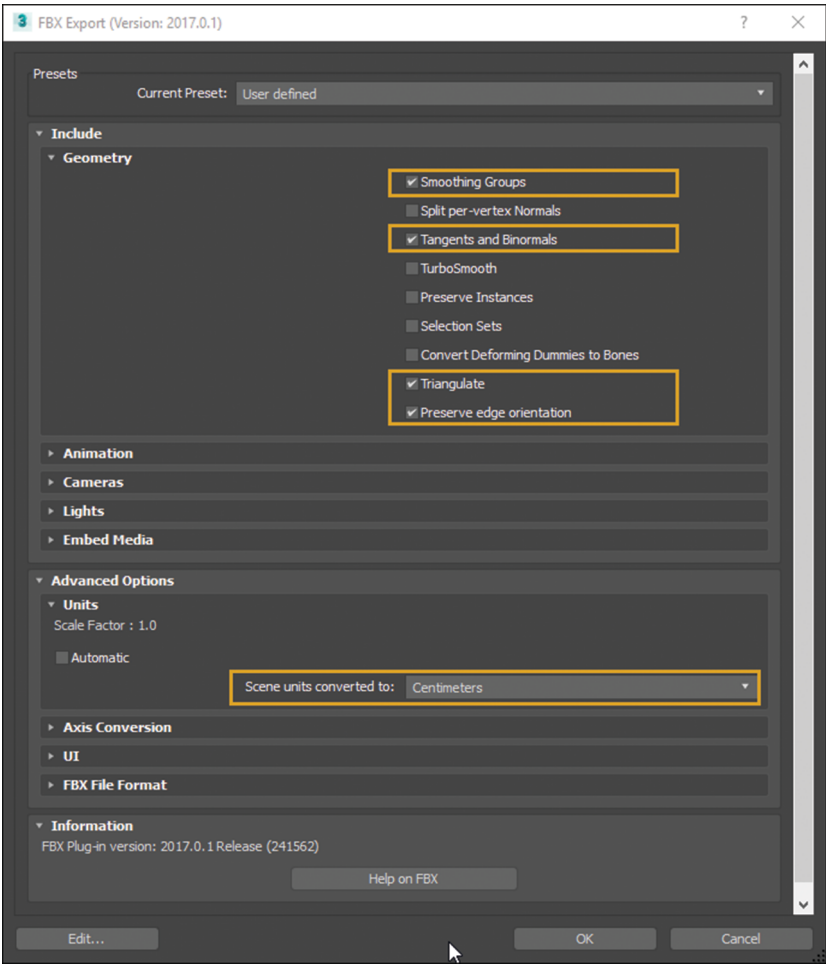


Рисунок 12.10 Диалоговое окно 3D Studio Max FBX Export, показывающее настройки, которые обеспечат возможность импорта вашей модели в UE4 со всеми smoothing groups, UV-координатами и правильной ориентацией ребер, чтобы ваши ассеты отображались в UE4 точно так же, как в вашем 3D-приложении

Импорт

Как и в случае с мешами архитектуры, импортируйте реквизит с помощью кнопки **Import** в Content Browser или перетаскивая файлы в Content Browser. Вы можете импортировать все ваши файлы сразу или один за другим.

Как и прежде, появится диалоговое окно Import Dialog for Static Meshes. Используйте настройки, показанные ранее на рисунке 12.5.

Auto Generate Collision

Не рекомендую использовать автоматически создаваемые столкновения. Я обычно не использую столкновение на своих мешах для реквизита для архитектурных визуализаций, потому что это часто слишком ограничивает движение игрока.

Transform Vertex to Absolute

Установка Transform Vertex to Absolute в значение **false** позволяет UE4 использовать авторский пивот для поворота в вашей модели, а не использовать 0,0,0 сцены и запекать вращение и другие преобразования в вершины меша. Если вы не переместили свой реквизит в 0,0,0 в 3D-приложении перед экспортом, то потребуется установить для него значение **false**.

Заключение

Импорт данных и контента в UE4 может показаться пугающим на первый взгляд, но подготовка и организация вашего контента, а также обеспечение правильных настроек импорта и экспорта для него делают процесс плавным и предсказуемым.

Теперь у вас должно быть хорошее понимание различий между объектами реквизита и архитектуры и того, как надежно поместить их в UE4 (рисунок 12.11). Теперь вы готовы разместить эти ассеты и начать строить интерактивный мир.

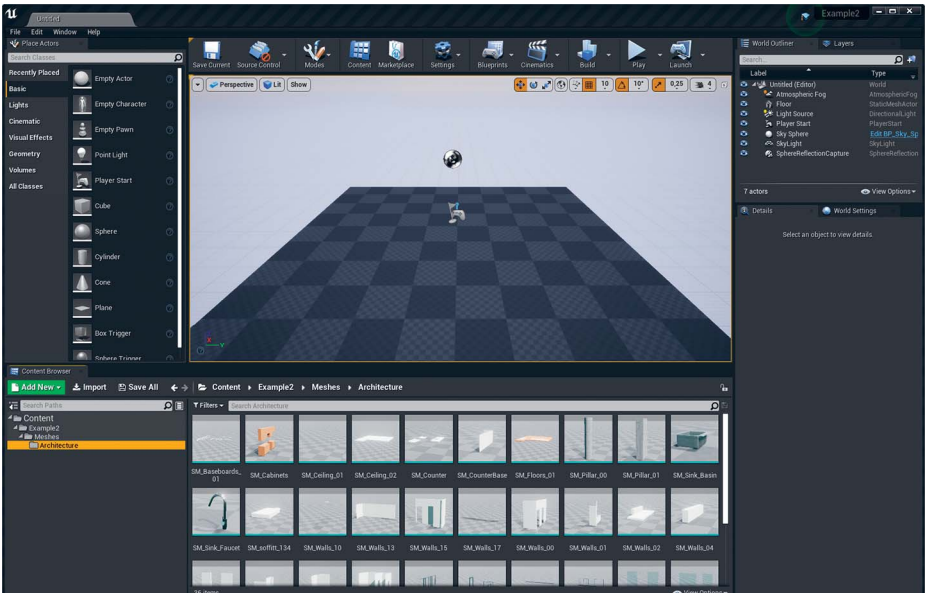


Рисунок 12.11 Content Browser, показывающий импортированные в Architecture Static Mesh Assets

ГЛАВА 13

НАПОЛНЕНИЕ СЦЕНЫ

Заполнение сцен в UE4 может быть сложной задачей для визуализации. Проектировщикам часто поручают представлять данные или проекты, которые требуют точности и аккуратности. В этой главе я покажу вам, как поместить статические реквизитные и архитектурные ассеты статических мешей на сцену именно там, где вы хотите.

Сборка сцен для визуализации

UE4 предлагает мощный набор инструментов для левел-дизайнеров (Level Designers, LDs) и художников, чтобы быстро создавать фантастические игры и визуализации. Художники могут перетаскивать ассеты, огни и другие активы в мир с помощью Content Browser; легко перемещать, масштабировать и поворачивать их с помощью инструментов преобразования, а также легко менять свойства с помощью Details Panel.

Построение сцен для визуализации может быть сложной задачей, поскольку вам часто нужно очень точно отображать данные, ставя точность выше свободы творчества.

Для архитектурных визуализаций часто приходится балансировать между потребностью в точности и художественной свободой для создания невероятно красивых пространств.

Вы должны точно разместить свои архитектурные меши в пределах уровня. Здесь нет никакой художественной свободы, потому что это данные, которые вам дали визуализировать.
Однако ваш реквизит должен быть вручную помещен на уровень, перемещен, повернут и свободно масштабирован, наполняя все жизнью и смыслом.

В этой главе я покажу, как я настраиваю свои уровни и наполняю их своей архитектурой и реквизитом. Я покажу инструменты Viewport и шорткаты, которые использую чаще всего и в итоге получаю сцену, готовую к применению освещения и материалов.

Настройка Level

Прежде чем вы построите уровень, нужен сам уровень. По умолчанию UE4 предоставляет простой уровень с облаками, источником света и Player Start Actor, расположенным на Box Static Mesh Actor.

Я предпочитаю начинать все заново с совершенно нового уровня, избегая любых проблем, которые могут возникнуть при использовании пресета.

Создание New Level

Чтобы начать с чистого листа для нового проекта, выберите **File > New Level** в Editor (рисунок 13.1).

Это создает новый, совершенно пустой уровень. Сохраните свой уровень, выбрав **File > Save Current As**.

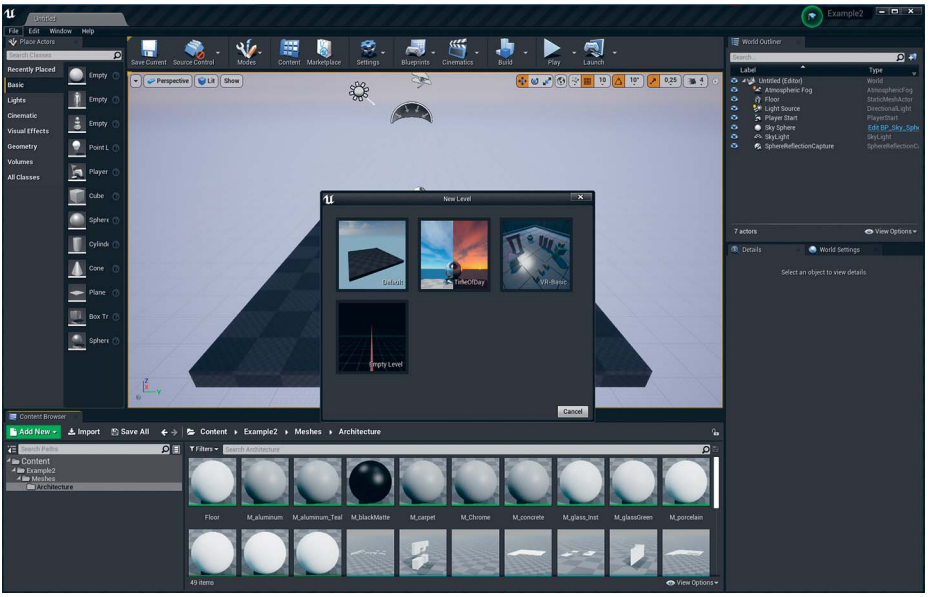


Рисунок 13.1 Создание нового пустого Level; уровень по умолчанию отображается во Viewport

Добавление простых источников света

Как и в первом проекте, первая задача состоит в том, чтобы поставить несколько простых источников света, прежде чем размещать какие-либо меши. Без света UE4 делает сцену либо черной, либо в Unlit View Mode, что затрудняет просмотр того, что вы делаете. Добавление ряда основных источников света в первую очередь позволяет проще видеть некоторые вещи и работать с ними.

Atmospheric Fog

На панели **Place Actors** выберите пункт **Visual Effects** и перетащите **Atmospheric Fog** в Viewport. Это обеспечивает горизонт, атмосферное рассеивание (голубое небо) и солнечный диск, который делает небо простым и приятным.

Directional Light

Выберите **Lights** на панели **Place Actors** и перетащите **Directional Light** в Viewport. Это будет работать как ваше Солнце.

Выберите **Directional Light**, найдите опцию **Atmosphere / Fog Sun Light** и установите ее в **true** (возможно, вам придется посмотреть в разделе Advanced Options rollout). Это позволяет направленному свету контролировать цвета актора атмосферного тумана и положение солнечного диска.

Sky Light

Перетащите Sky Light на сцену. Это образец окружающей среды вокруг него и использует данную информацию для обеспечения косвенного освещения сцены. Теперь у вас должна быть базовая настройка освещения (рисунок 13.2).

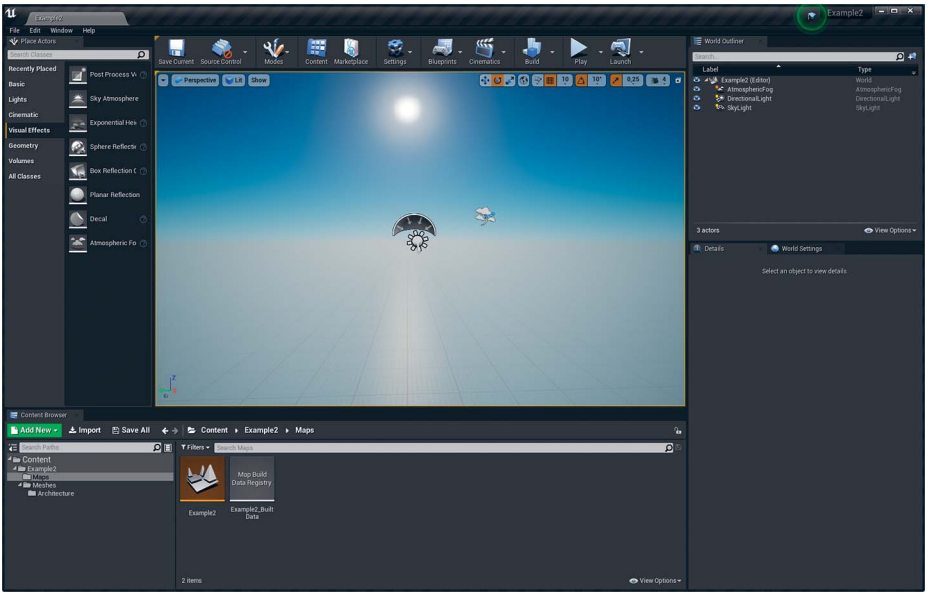


Рисунок 13.2 Базовая настройка освещения показывает выбранный Directional Light и Atmosphere / Fod Sun Light, установленный в true на панели Details, что позволяет установить положение Солнца в небе.

Размещение Architecture Static Meshes

Подготовив свои архитектурные ассеты к использованию origin FBX в качестве pivot point (0,0,0), вы легко разместите их на уровне в том самом месте, куда вы их экспортировали.

Dragging и Dropping мешей

Перетащите ваши меши на Viewport из Content Browser. Вы можете выбрать один или несколько Static Mesh Actor с помощью обычных сочетаний клавиш (Shift-click, Ctrl / Cmd-click и т. д.) и перетащить их в любое место Viewport (рисунок 13.3).

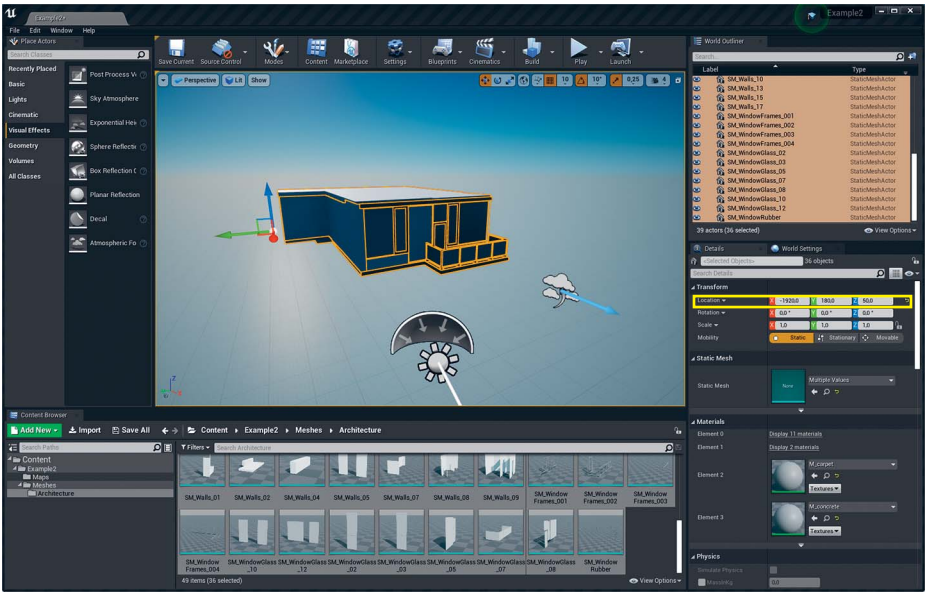


Рисунок 13.3 Результаты операции перетаскивания, а затем установка свойства Location размещенных мешей на 0,0,0

Настройка Location на ноль

Когда вы перетаскиваете эти меши на уровень, созданные Static Mesh Actors выбираются в World Outliner и Level. Если вы посмотрите на панель Details, то увидите, что можете редактировать свойства всех мешей одновременно. Это позволяет легко установить местоположение на 0,0,0 (рисунок 13.3).

Теперь все ваши архитектурные меши размещаются точно там, где они были в вашем 3D-приложении.

Размещение декоративных мешей

Теперь, когда стены, полы и другая архитектура размещены, пришло время расставить реквизит.

Как и в большинстве случаев с контентом, просто перетащите реквизит, чтобы извлечь его из Content Browser на карту и поместить на свой уровень.

В этом примере мне дали свободу на заполнение сцены мебелью и реквизитом, выбранной клиентом и уже готовым реквизитом из моей библиотеки моделей.

Привязка к поверхности

Когда вы впервые перетаскиваете модель на уровень, обратите внимание, что она проецируется на сцену и «прилипает» к тому, на которое приземляется. Такая привязка отлично подходит для того, чтобы поместить реквизит на уровне, но что произойдет после подобного размещения?

Включение Surface Snapping на панели Viewport блокирует актора, перемещаемого по поверхности так, как если бы он летал над ней (рисунок 13.4).

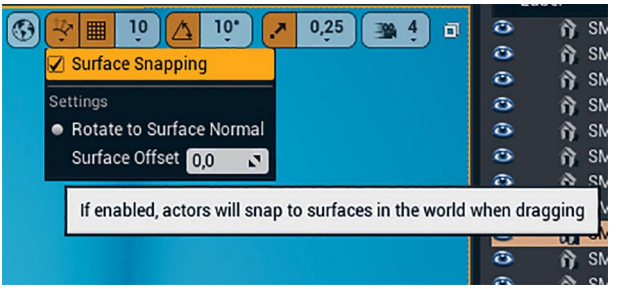


Рисунок 13.4 Выпадающее меню Surface Snapping

Вы также можете включить **Rotate to Surface Normal**, чтобы размещаемый меш переориентировался на полигональную поверхность, на которой вы его размещаете. Это отлично подходит для размещения акторов на стенах, потому что они будут вращаться, когда вы перетаскиваете их от стены к стене.

Клонирование и копирование

Обязательно используйте инструменты клонирования, такие как copy-paste и метод Alt-Drag для создания копий. Эти методы делают заполнение сцены очень быстрым.

Shift+Dragging

Если вы удерживаете нажатой клавишу Shift во время перемещения актора с помощью механизма Transform, вид будет следовать за этим актором. Вы можете использовать эту технику в сочетании с методом Alt-Drag — это отличный способ перемещения акторов по большим уровням с помощью перспективного Perspective Viewport.

Организация сцены

Добавление меша, света и других акторов на сцене вскоре приводит к сотням акторов на ней. Их организация необходима для быстрого рабочего процесса, и UE4 предлагает инструменты, помогающие управлять вашей сценой.

World Outliner

Когда вы добавляете акторов на свой уровень, они появляются в списке **World Outliner**. Этот список может быстро стать громоздким и трудным для навигации. К счастью, у вас есть несколько вариантов для его очистки.

Вы можете легко создавать папки с помощью кнопки New Folder (расположенной в правом верхнем углу World Outliner рядом со строкой поиска), но я предпочитаю использовать метод щелчка правой кнопкой мыши. Выберите ассеты, которые хотите поместить в папку в World Outliner, и щелкните правой кнопкой мыши на одном из них, что откроет контекстное меню (рисунок 13.5).

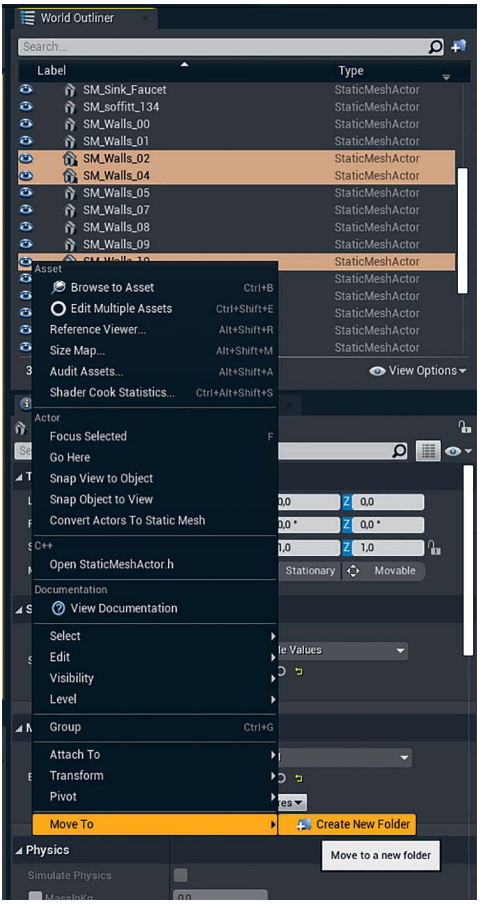


Рисунок 13.5 Добавление акторов в новую папку в World Outliner с помощью меню правой кнопки мыши

Здесь вы можете добавить их в новую папку или в существующую и не искать ее в World Outliner.

Слои

Менее очевидным организационным методом, но, вероятно, удобным для художников визуализации, является система слоев (Layer). Как и большинство 3D-приложений, UE4 имеет полноценную систему слоев, в которой вы можете легко скрывать, отображать и выбирать акторов через слой.

Интерфейс слоев не открыт по умолчанию, но вы можете получить доступ к нему через меню Window (рисунок 13.6).

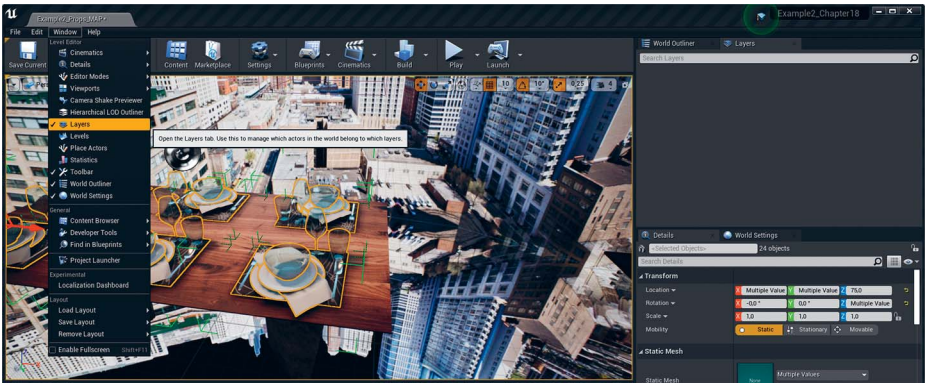


Рисунок 13.6 Включение вкладки Layers; обратите внимание на вкладку Layers справа с размещенными в ней реквизитами

Вы также можете использовать как системы World Outliner, так и системы слоев, поскольку они не влияют на производительность среды выполнения или функции; они просто помогают вам организовать свою сцену.

Я считаю, что система слоев трудна в использовании и легко ломается. Я предпочитаю использовать систему папок World Outliner, чтобы сохранить свою сцену организованной.

Группировка

Группировка (Groupping) — это отличный способ сделать многоразовые или легко выбираемые наборы акторов на вашем уровне.

Выберите акторов, которые хотите сгруппировать, щелкните правой кнопкой мыши и выберите пункт **Group** (рисунок 13.7). Создается Group Actor, и выбранные акторы привязываются к нему. Выбор любого из акторов или группы акторов выбирает их всех.

Группы отлично подходят для построения многоразовых наборов геометрии в ваших сценах.

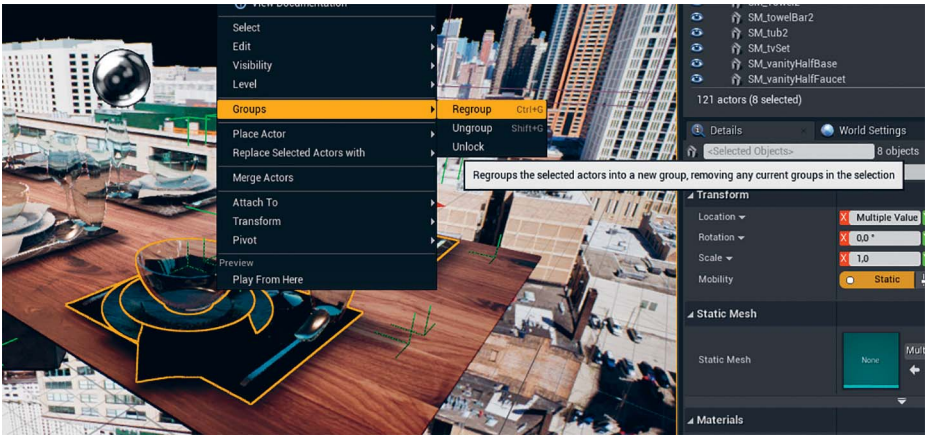


Рисунок 13.7 Группировка мешей вместе для дублирования вокруг сцены

После того как акторы сгруппированы, вы можете **Ungroup** или **Unlock** группу для редактирования с помощью контекстного меню. Вы не можете назвать или изменить Group Actor.

Actor Blueprints

Вы можете взять набор акторов (включая свет, частицы, звуки и т. д.) и создать из них объект Blueprint. У этого есть несколько преимуществ перед группами.

Чтобы создать Actor Blueprint из акторов вашего уровня, выберите ассеты, которые вы хотите преобразовать, и выберите **Convert Selected Components to Blueprint Class** (рисунок 13.8).

Затем вы должны выбрать, где сохранить новый Blueprint Asset в Content Browser. Этот Blueprint содержит компоненты, которые ссылаются на статические меши и другие выбранные вами классы. Он не создает копию или не объединяет их в новый меш.

Теперь вы можете изменить один Blueprint Asset, и все Level Actors, основанные на нем, будут обновлены. Actor Blueprint отлично подходит для создания таких вещей, как свет или другие сложные наборы акторов или даже просто группы акторов, которые вы хотите повторно использовать.

Рисунок 13.8 Преобразование выбранных компонентов в класс Blueprint

Заключение

Размещение контента в UE4 — это одновременно весело и сложно. Теперь вы должны лучше понимать, как поместить архитектурные меши на сцены в точном положении, в котором они были в вашем 3D-приложении, и как организовать и поддерживать сцену по мере добавления все большего количества контента.

Когда сцена заполнена (рисунок 13.9), пришло время заполнять ее освещением, материалами и эффектами постобработки, чтобы достичь своей цели создания фотореалистичной сцены.

Рисунок 13.9 Наполненная сцена с предварительным освещением

ГЛАВА 14

АРХИТЕКТУРНОЕ ОСВЕЩЕНИЕ

Разработка чистого, точного, реалистичного освещения очень важна для создания визуализаций. Взаимодействие света, теней, отражений и материалов оживляет миры, создавая глубину и рассказывая истории. В этой главе вы узнаете, как настроить великолепное освещение Global Illumination (GI) с помощью Lightmass.

Получение большего от Lighting

Системы Lighting и Materials в UE4 тесно связаны между собой. Вам нужно работать между ними для достижения конечного результата. Эта глава демонстрирует, как освещать интерьер сцены с помощью UE4.

Высокий динамический диапазон конвейера рендеринга Unreal'a сочетается с такими эффектами, как auto-exposure и bloom, для создания реалистичной, теплой и динамической среды освещения. Правильно настроить освещение с помощью этих методов может быть сложно, но, используя некоторые базовые настройки, вы сможете быстро получить потрясающий результат.

В этой главе вы изучите, как настраивать свойства и параметры Level и Static Mesh для достижения отличного освещения с приемлемым временем на его расчеты.

В этой главе также будет показано, как использовать эффекты post-processing, такие как vignette, color grading и depth of field, для создания более фотореалистичного изображения (рисунок 14.1).

Рисунок 14.1 Предпросмотр финального освещения

Наконец вы изучите, как расположить Reflection Capture Actors на уровне для улучшения освещения и обеспечения более точно выглядящих зеркал.

Материалы и освещение

Как и весь рендеринг Global Illumination (GI), освещение в UE4 напрямую зависит от материалов, и наоборот. Материалы окрашивают и смягчают отраженный свет, который информирует нас о свойствах поверхности, отражаясь от них.

Из-за этого вы не можете разделить их полностью. Эта и следующая главы предназначены, чтобы рассмотреть обе темы отдельно; однако для получения лучшего результата вы будете использовать методы из этих глав параллельно.

Static Lighting с Lightmass

UE4 произвел огромный резонанс в индустрии архитектурной визуализации благодаря своим впечатляющим визуальным эффектам. Один из способов, благодаря которому UE4 справляется с рендерингом невероятно хорошо освещенных сцен так быстро, является предварительное вычисление освещения с помощью GI рендера, называемого *Lightmass*.

Lightmass рассчитывает попадающий на поверхность свет для каждого Static Mesh в вашей сцене, используя, к примеру, Mental Ray для вычисления Global Illumination, основанного на фотонах. Затем он записывает эти данные для нескольких Textures, называемых Lightmaps и Shadowmaps. Этот процесс называется *построением Static Lighting*. После завершения рендеринга освещения Textures автоматически импортируются и применяются.

Как следует из названия, вы не можете перемещать или менять объекты, освещенные Static Lighting или источниками света, после их построения. Если вы добавите, удалите, переместите или каким-либо еще способом измените Static Lights или Static Meshes на сцене, то *нарушите* освещение и должны будете повторно построить его.

Обратите внимание, что вы можете *добавить* источники света к уже построенной сцене с освещением. Несмотря на то что вам нужно будет построить освещение снова, чтобы увидеть влияние GI недавно размещенных источников света, существующий static lighting останется. Это может быть полезно, так как вы добавляете все больше и больше источников света на вашу сцену.

Lightmaps и Shadowmaps хранятся в специальном **BuildData** UMAP Level файле (пользовательский файл формата UE4, хранящийся как Levels) вместе с вашими сохраненными уровнями. Этот файл и ваш проект будут значительно увеличиваться по мере того, как вы станете добавлять больше Assets, так как каждый из них увеличивает расход памяти на Lighting Textures.

Настройка Sun и Sky Lights

В предыдущей главе вы разместили Directional Light Actor, Sky Light Actor, и Atmospheric Fog Actor. Эти три актора обеспечивают основу для построения освещения, но они требуют некоторой настройки для работы со Static Lighting.

Солнечный свет

Directional Light будет действовать как солнечный свет, обеспечивая интенсивный прямой свет, который отлично подходит для решения задачи с глобальным светом (GI).

Тремя основными типами или классами источников света в UE4 являются Point, Spot, и Directional. Освещения Point и Spot излучают из одной точки в пространстве. Они отлично подходят для близкого освещения, такого как лампы, светильники, и точечных фонарей. Но они не очень подходят для солнца, которое, с нашей точки зрения, на Земле испускает лучи так, как если бы они шли параллельно друг другу. Для этого вам понадобится свет, который симулирует единственный источник прямого освещения. Внесите класс Directional Light Actor.

Как и большинство Actors, вы можете легко разместить освещение на вашей сцене, перетащив его из Class Browser в Viewport.

Вы можете разместить освещение в любом месте на сцене, так как оно не излучается из этой точки; важен только поворот. Я разместил свое там, где я могу легко найти его снова.

Найдите приятный или реалистичный поворот для освещения. Динамические тени действуют как предпросмотр и дают вам представление, где будут статические тени (рисунок 14.2). Оглянитесь вокруг, когда закончите. Динамические тени UE4 обеспечивают хорошую приблизительную точность, показывая, где будут располагаться запеченные тени на сцене.

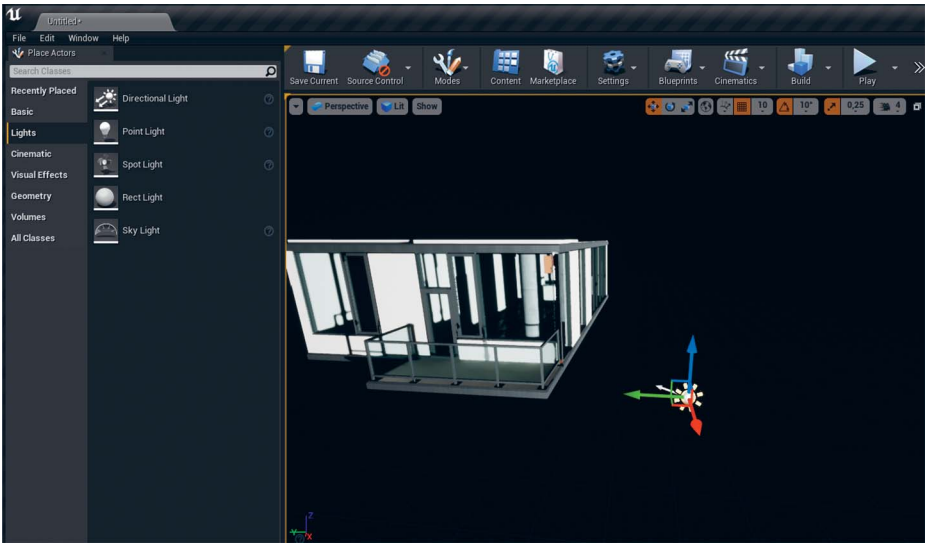


Рисунок 14.2 Размещение Directional Light

В Details Panel показаны различные доступные для выбранного источника света параметры (рисунок 14.3). Вам нужно отрегулировать несколько настроек для получения лучшего качества и производительности от этого Light Actor.

Рисунок 14.3 Свойства Directional Light в Details Panel

Mobility

Первым и наиболее важным параметром, который нужно изменить, является **Mobility**. Для этого источника света вам нужно использовать Stationary Light. Это как Static Light (он не может перемещаться), но он не записывает свое прямое освещение в Texture; скорее динамически рассчитывает его. Это означает, что вы можете менять яркость и цвет после построения освещения.

Обратите внимание, что изменение Intensity не влияет на освещение GI, так как оно запекается в Static Textures.

Intensity

Для получения эффекта яркого солнечного света используйте значение **Intensity** 8–10. Это гарантирует, что солнце является самым ярким источником освещения на вашей сцене.

Indirect Lighting Intensity

Поскольку интенсивность Stationary Lights может быть изменена, вам нужно сообщить Lightmass, насколько ярким рассчитывать это освещение. Для этого вы можете использовать параметр **Indirect Lighting Intensity** для регулировки яркости GI этого источника света. Установите его где-нибудь между 1.0 и Intensity, установленной для вашего источника освещения.

Temperature

Проверьте галочку на **Use Temperature** и установите **Temperature** около 5000–5500. Это обеспечит слегка желтое солнце, которое все еще значительно «белее», чем обычный свет от ламп накаливания, у которых значение 2700–3500.

Atmosphere Sun Light

Убедитесь, что выбран параметр **Atmosphere Sun Light**, который обеспечит совместную работу с Atmosphere Fog Actor для создания физически корректного освещенного неба, которое обеспечит цвет окружения для сцены. Если вы не видите его в начале, то этот параметр может быть скрыт в Advanced Options источника света.

UE4 оснащен системой атмосферного освещения рассеянного тумана, которая симулирует атмосферу земли. Управляемая с помощью Atmospheric Fog Actor и Directional Light Actor. Просто добавьте Environmental Fog Actor на вашу сцену. Если у вашего Directional Light Actor включен Atmosphere Fog Light, он будет определять местонахождение солнечного диска и устанавливать цвета неба в зависимости от позиции солнца, создавая простой, но эффектный skybox.

Sky Light

Для симуляции окружающего света атмосферы UE4 использует Sky Light Actor. Этот специальный источник света может запечатлеть существующую сцену как Cubemap, для использования непрямого освещения, или вы можете предоставить ему HDR Texture.

В данном примере давайте просто запечатлеем сцену.

Разместите этот источник света на вашей сцене там, где вы сможете легко его найти. Вам также нужно убедиться, что он не находится в какой-нибудь геометрии, таким образом захватывая незакрытое небо. Sky Light Actor содержит несколько настроек для получения оптимальных результатов (рисунок 14.4).

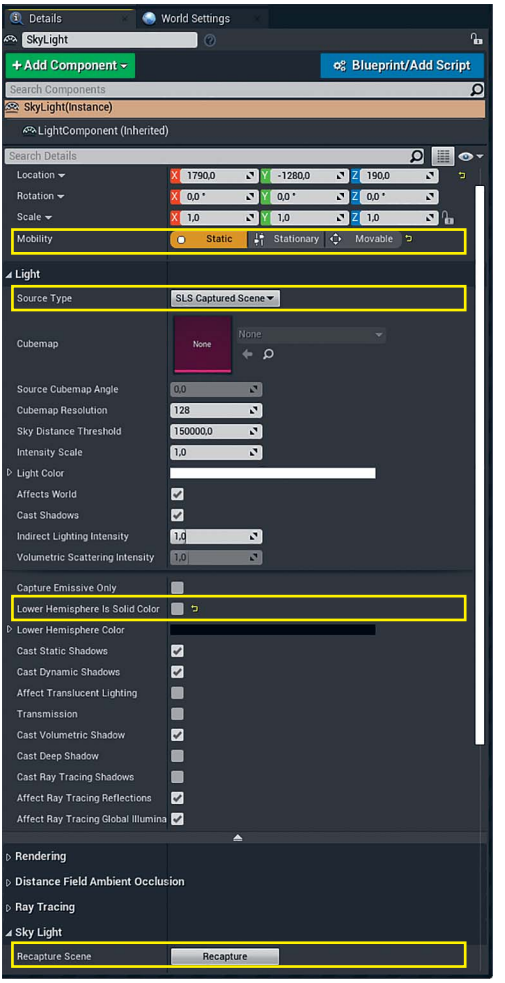


Рисунок 14.4 Расположение Sky Light

Mobility

Sky Light установлен на Static для этой сцены. Вы также можете выбрать Stationary для этой сцены, но вы лишитесь определенного уровня качества и детализации вашего непрямого освещения неба с возможностью динамического изменения свойств Intensity, Source Cubemap Angle и Cubemap Texture без перерасчета освещения.

При установке Mobility в значение Static запекается все непрямое освещение карт света и тени, созданных для источника света Lightmass. Это делается, чтобы избежать проблем мобильного пайплайна, например, утечек света или неточного освещения.

Source Type

Вы можете выбрать между импортированной HDR Cubemap или Cubemap сцены, взятую Sky Light Actor в Editor. В данном случае используйте сцену в качестве освещения, установив Source Type на SLS Captured Scene.

Lower Hemisphere Is Solid Color

Настройка Lower Hemisphere Is Solid Color является личным предпочтением. Установка его на true заменяет нижнюю половину захваченного Cubemap одним сплошным цветом. Это может ограничить количество света, падающего на площадь, поэтому я предпочитаю оставлять его на false для интерьера сцены, где важно получать как можно больше освещения на сцене.

Recapture Scene

Если вы измените освещение или сцену, Sky Light не будет динамически обновлять Cubemap, используемую для освещения сцены. Вы должны вручную повторно захватить Cubemap с помощью кнопки Recapture Scene.

Билд освещения

Теперь, когда сцена содержит элементарное освещение, вы можете построить освещение в первый раз.

Для построения освещения нажмите кнопку Build в панели инструментов Editor. Вы можете нажать на стрелку рядом с ним, чтобы увидеть настройки. Наиболее важным параметром является Lighting Build Quality. Он позволяет перейти от быстрого превью к хорошо выглядящему, но медленному для рендеринга production quality build (рисунок 14.5).

Lightmass предлагает несколько предустановок для качества освещения от preview до production. Production lighting builds могут занимать на порядок больше времени, чем preview builds, но билды низкого качества могут содержать артефакты и несоответствия, которые трудно отличить от реальных проблем с геометрией или освещением.

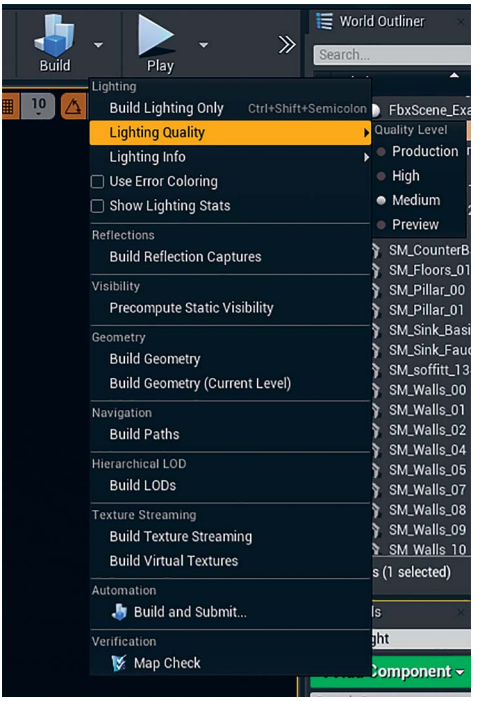


Рисунок 14.5 Настройка Lighting Quality из выпадающего списка Lighting Build

Заметка

Для оптимизации времени на построение освещения ваших сцен вы можете скрыть Actors, оставив только Architecture и Fixtures. Это значительно уменьшит время, необходимое для построения освещения, позволяя вам iterate faster. Для этого выберите Actors, которые не хотите освещать, и установите свойство Visibility в Details Panel на false.

Чтобы сделать их видимыми снова, вам нужно найти Actors, перечисленные в World Outliner, так как они больше не видны в Viewport.

Кроме того, вы можете выбрать Static Meshes, для которых хотите избежать расчета освещения, и установить их Mobility на Moveable.

В любом случае вам нужно использовать группы, слои или папки для организации вашей сцены, чтобы облегчить переключение этих свойств.

Рисунок 14.6 показывает сравнение построений освещения между preview и production. Вы можете заметить, что точность значительно уменьшена в построении preview, но время на расчет освещения также значительно сократилось.

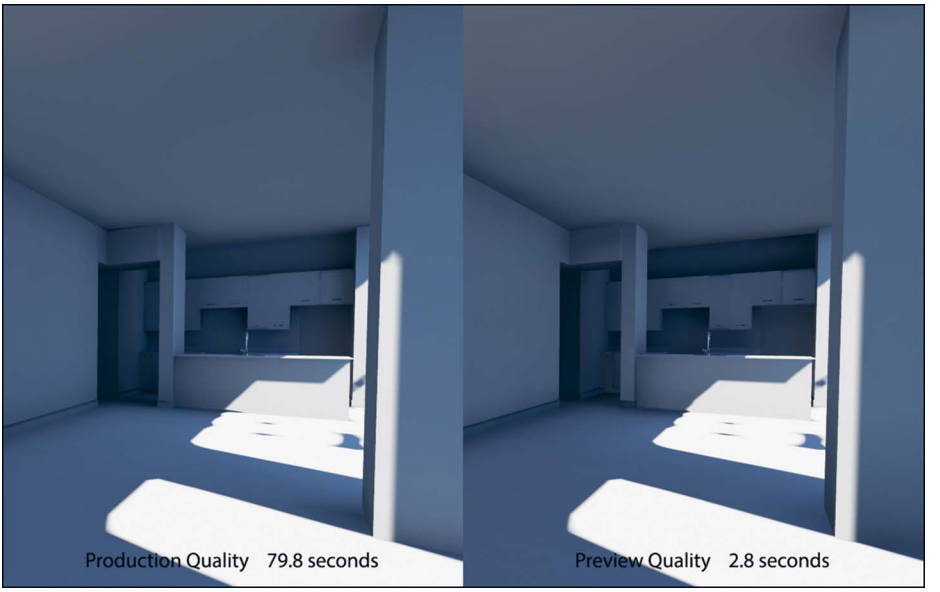


Рисунок 14.6 Первое построение освещения в качествах production и preview

Как вы можете заметить, даже при использовании качества production освещение остается все еще не очень хорошим. Тени являются неопределенными, блочными и в низком разрешении, а также содержат ошибки на плинтусе и в темных областях. Сделав несколько настроек, вы можете достаточно быстро заставить эту сцену выглядеть гораздо лучше.

Настройка Lightmass с архитектурной визуализацией

Первым шагом для «причесывания» вещей является повышение настроек Lightmass. Architecture Visualizations определяет приоритет качества изображения над временем рендеринга, и в UE4 это ничем не отличается. Вы увеличиваете построение освещения, но в итоге получаете сцену, которая выглядит гораздо лучше.

Настройки для Lightmass хранятся на каждом основном Level и могут быть изменены в настройках свойств World. Панель Lightmass предлагает множество настроек (рисунок 14.7). К счастью, настройки по умолчанию UE4 хороши для большинства обстоятельств. Каждый параметр существенно влияет на время построения, поэтому всегда стоит начинать с малого и продвигаться дальше.

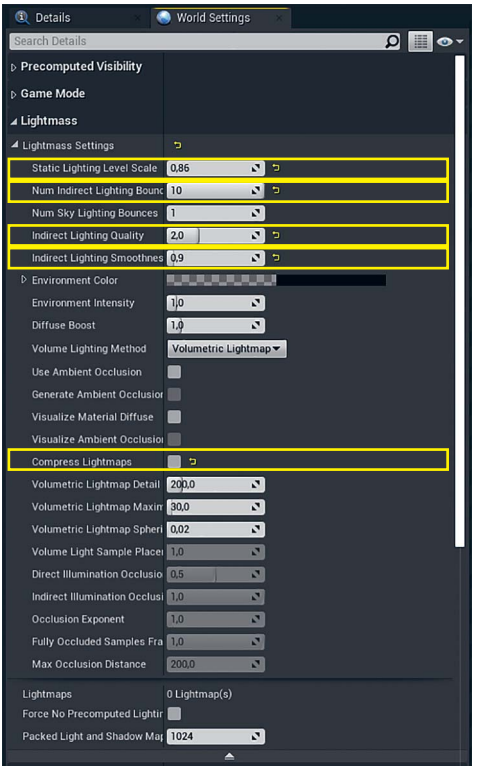


Рисунок 14.7 Настройки Lightmass для мира

Заметка

Если вы исследовали Lightmass онлайн, вероятно, вы видели некоторые сложные настройки файла **.ini** для получения Lightmass, которая производит чистые lighting solutions для архитектурных визуализаций.

Эти настройки в значительной степени стали ненужными в более поздних версиях движка (возможно, это вредит качеству изображения и, безусловно, времени на построение). Epic Games проделали огромную работу по улучшению Lightmass для визуализации, и вам нужно всего лишь откорректировать настройки, выставленные в World Settings.

Static Lighting Level Scale

Параметр Static Lighting Level Scale контролирует плотность фотонов на сцене. Маленькие числа означают более плотные фотоны с большей детализацией. Это может значительно повысить время на построение. Вы можете оставить этот параметр

на значении по умолчанию, равном 1,0, или немного уменьшить его. Любое значение ниже 0,8 повысит время на построение, но может появиться шум.

Num Indirect Lighting Bounces

Num Indirect Lighting Bounces фиксирует, сколько раз фотон может отразиться от поверхности на сцене до уничтожения. Значение 10 обеспечивает очень хороший результат, позволяя фотонам проникать в самые темные области вашей сцены. Повысьте этот параметр, если у вас есть пятна в темных областях.

Indirect Lighting Quality

Параметр Indirect Lighting Quality оказывает наибольшее влияние на время построения и общее качество. Это увеличивает количество семплирований, сделанных для сглаживания GI solution. Очень высокое значение обеспечивает очень плавный GI. Оставьте этот параметр в диапазоне от 1 до 4; выше только при шумном GI.

Indirect Lighting Smoothness

Используйте параметр Indirect Lighting Smoothness, чтобы контролировать, насколько мягким или резким является ваше непрямое освещение GI. Маленькие числа делают более резкими, но шумными непрямые тени, тогда как высокие значения дают более мягкие тени. Немного уменьшите этот параметр, если хотите получить эту резкость, но не сильно ниже 0,75, или придется повышать общее качество вашего решения для удаления шумных артефактов.

Compress Lightmaps

Наиболее важным параметром для архитектурной визуализации является Compress Lightmaps, который отключает сжатие Texture для Lightmass Textures. Как видно на рисунке 14.8, сжатие может привести к соединению и блокировке артефактов. Это применяется в большинстве игр, которые не полагаются на чистое освещение, но не для архитектурных визуализаций, где точность освещения и поверхности имеют наивысший приоритет.

Вам не следует трогать другие настройки, пока не отключите этот параметр. Шум, который он вносит, сделает ваше освещение непригодным для архитектурных визуализаций, и никакие доводки не смогут устранить это.

Использование несжатых Lightmaps сказывается на расходе памяти. Несжатые Lightmaps занимают значительно больше дискового пространства, нежели сжатые Lightmaps. Они также используют больше памяти видео при запуске в игре.

Если ваш Level очень большой (стадион или целое здание) или вы ориентируетесь на нижний сегмент аппаратных средств (ноутбуки, мобильные устройства и т. д.), вы

можете использовать параметр Compressed Lightmaps, чтобы позволить этим Levels загружаться в RAM.

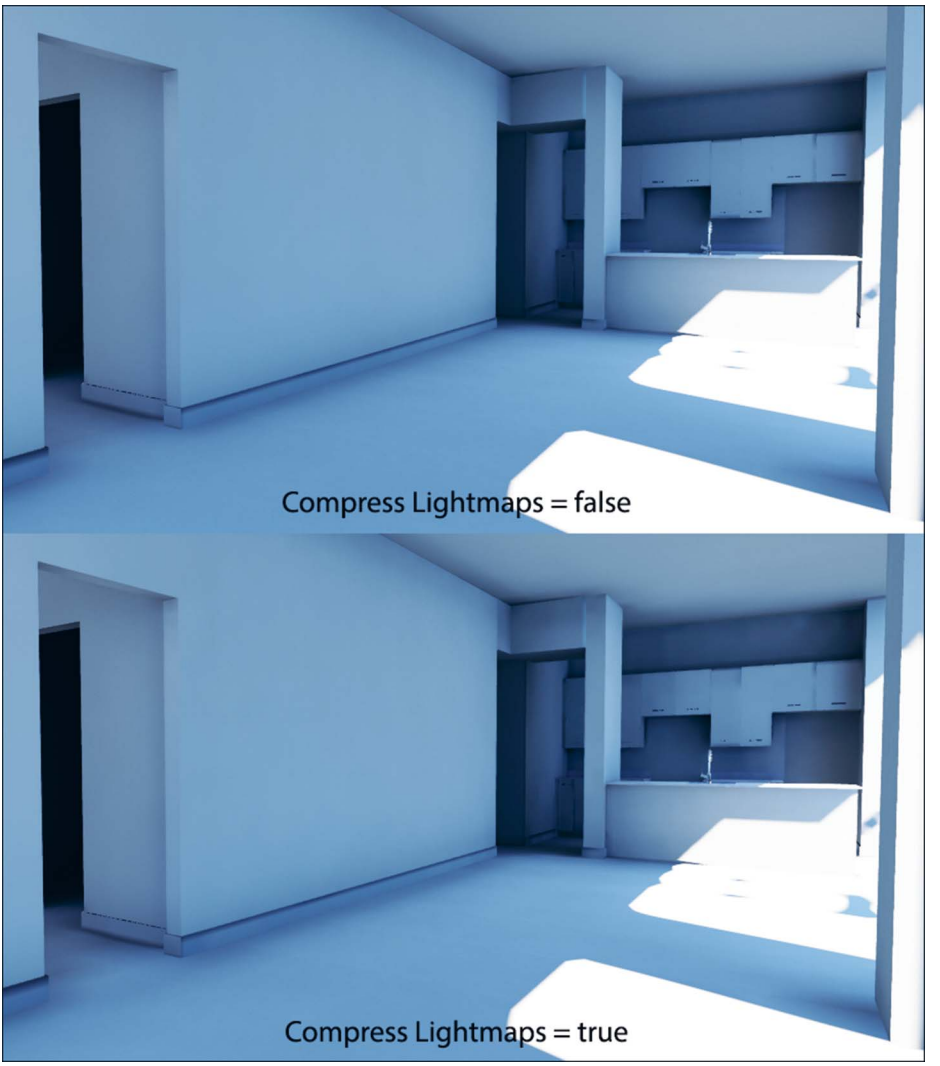


Рисунок 14.8 Сравнение Compress Lightmaps (with increased contrast to enhance artifacts), показывающее артефакты на шкафах, дверных рамах и гладких поверхностях, таких как стена слева

Настройка Lightmap UV Density

Придание вашим Lightmaps достаточного высокого разрешения важно для получения ровного, детализированного освещения. Однако слишком высокое разрешение может испортить вашу сцену из-за длительного построения и огромного количества используемой памяти.

Найти баланс может быть непросто, но вот один из способов настроить разрешение Lightmap в UE4.

1. Включите визуализацию Lightmap Density, выбрав **View Mode** > **Optimization Viewmodes** > **Lightmap Density** (рисунок 14.9).

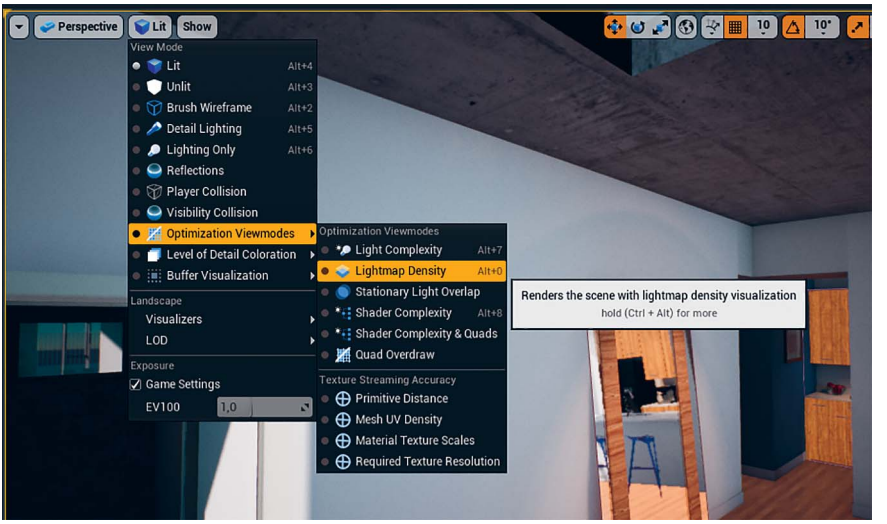


Рисунок 14.9 Переключение режима визуализации на Lightmap Density

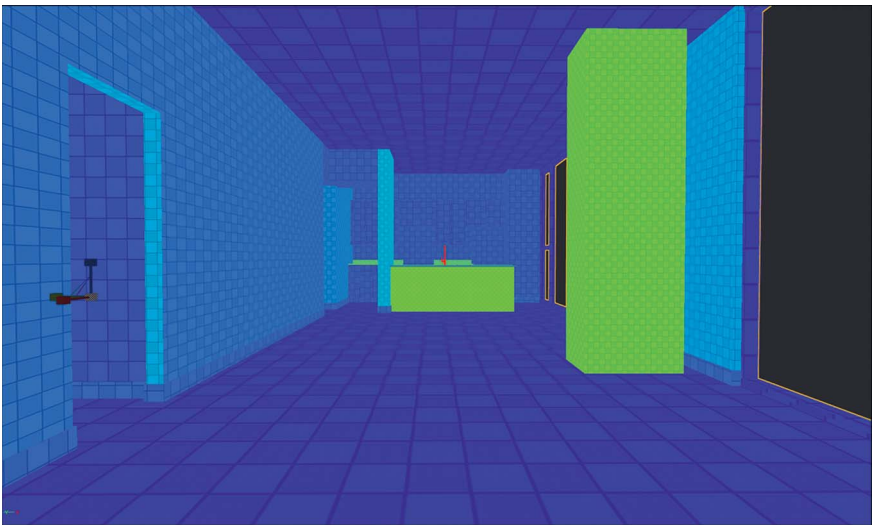


Рисунок 14.10 Визуализация Lightmap Density, показывающая слишком низкое разрешение, которое не позволяет получить очень детализированные тени и освещение

Когда вы впервые попадете в этот режим, вы увидите что-то, как рисунке 14.10, — множество сеток разных размеров и цветов без информации об освещении. То, на что вы смотрите, является визуализацией пикселей Lightmap для каждого объекта на сцене. Это поможет вам быстро и интерактивно настроить разрешение.

Синий означает, что разрешение Lightmap слишком низкое, а красный — слишком высокое. Вам нужно стремиться к зеленому или оранжевому на всех ваших поверхностях для получения лучшего качества с приемлемым временем построения.

2. После их настройки Lightmaps должен стать более равномерным по плотности и выше по разрешению, как показано на рисунке 14.11.

   Вам не нужно устанавливать разрешение, кратное двойке (64, 128 и т. д.), но не нужно отклоняться слишком сильно. Разрешение более 1024 поддерживаются, но должны быть использованы с осторожностью. Чем выше разрешение, тем дольше время построения и тем больше памяти Level занимает на диске. Лучше всего держать его как можно ниже без нежелательных артефактов.

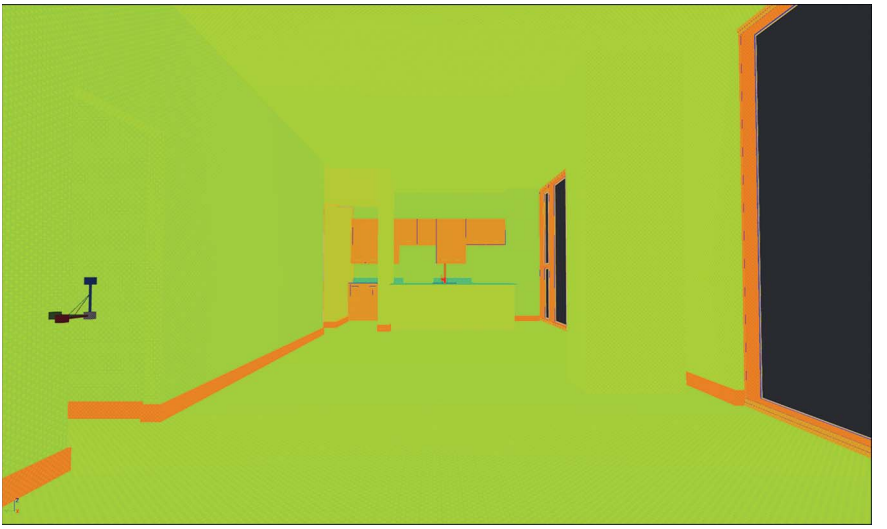


Рисунок 14.11 Режим визуализации с лучшими настройками Lightmap Density visualization

Настройка разрешения Lightmap для Props отличается, поэтому вы настраиваете ее в Asset Level. Это гарантирует, что для каждого реквизита на уровне будет правильное разрешение, так как сделанные вами изменения для исходного Asset отразятся на всех сценах объектов, которые ссылаются на эти Assets.

Имейте в виду, что любые изменения настроек Static Lighting для Mesh Asset нарушат освещение на каждом Level, на который ссылается Asset.

Также вы можете настроить плотность Props для каждого Object basis на вашем уровне с помощью свойства Overridden Light Map Res в Details Panel. Я иногда задаю своим реквизитам (особенно если они содержат множество деталей или органических форм) более высокую плотность, нежели Architecture Meshes, как показано на рисунке 14.12, что делает их красноватыми.

Будьте осторожны; повторяющиеся реквизиты могут быстро генерировать много Texture-данных в высоком разрешении.

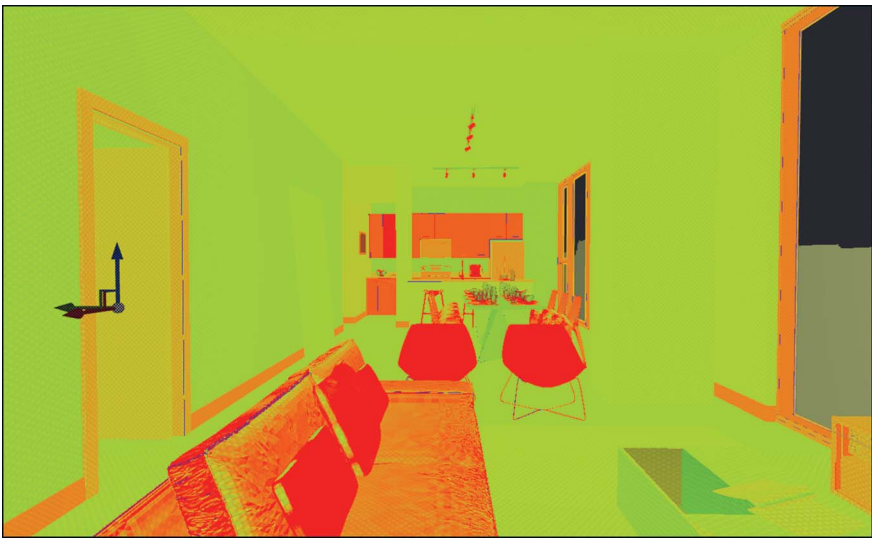


Рисунок 14.12 Конечная плотность Lightmap, показывающая более высокую плотность для органических Props, таких как диван или стулья

Билд и сохранение

Теперь самое время заново построить ваше освещение. Вы сразу заметите, что с увеличением плотности ваших Lightmaps, время построения также значительно увеличилось. Ваше общее качество также должно было значительно вырасти с более детализированными тенями и непрямым светом.

А еще сейчас самое время сохранить вашу работу.

Размещение интерьерного света

Теперь, когда у вас есть Солнце и небо на сцене, вы можете начать размещать источники света внутри, чтобы помочь осветить некоторые темные уголки.

Spot Lights

Точечные источники света и прожектора имеют те же настройки, что и свет в рендерах с трассировкой лучей, и вы должны чувствовать себя как дома (рисунок 14.13). Есть только несколько различий, о которых следует знать.

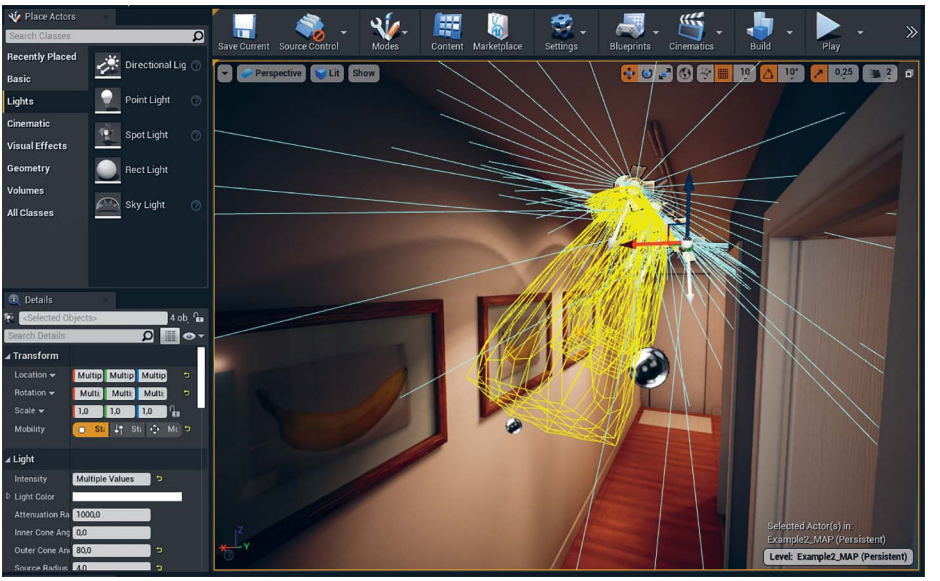


Рисунок 14.13 Размещение Spot Light Actors

Mobility

Я рекомендую использовать Static Lights почти исключительно для внутреннего освещения. Хотя стационарные и динамические источники света дают некоторые преимущества, они добавляют сложность и значительную стоимость производительности вашим сценам.

Intensity

Постарайтесь помнить, что внутреннее освещение должно быть гораздо менее ярким, чем солнечный свет. Даже самый яркий внутренний свет, кажется, не дает много света в залитой солнцем комнате.

Получение правильной яркости освещения — это искусство. Это зависит от количества ИС в области, яркости солнечного света и даже цвета материалов, от которых он отражается. Яркость имеет критически важное влияние на окончательный вид вашего освещения.

Вам нужно повторять настройку освещения, чтобы найти подходящий баланс для вашей сцены и вкусов.

Temperature

В этом примере я использовал настройку температуры для своего света. Это дает реалистичный цвет, основанный на яркости света. Я использовал более теплый тон для внутреннего освещения, который позволяет предметам быть более контрастными по отношению к Солнцу и небу.

Attenuation

Ограничение расстояния, на которое может перемещаться прямое освещение, используется как для производительности, так и для художественных целей. Ограничение ослабления света ограничивает количество мешей, на которые он влияет в мире. Это имеет решающее значение для производительности, особенно для динамически затененных огней. В Lightmass ограниченный радиус затухания повлияет на меньшее количество мешей, что ускорит построение освещения; но поскольку данные о свете и тени запекаются в текстуры, нет никакой выгоды для этого в реальном времени.

IES Files

UE4 поддерживает световые профили IES (Illuminating Engineering Society) для точечных светильников и прожекторов. Вы можете импортировать файлы IES и применять их к лампам, чтобы получить интересные паттерны освещения (как показано на рисунке 14.13). Вы можете найти профили IES по всему интернету от реальных производителей и поставщиков освещения или даже из Unreal marketplace.

Auto Exposure

Временное отключение Auto Exposure может быть большой помощью при настройке освещения на вашей сцене. Автоэкспозиция по умолчанию очень агрессивна и может легко стать очень яркой или очень темной.

Блокировка экспозиции с помощью меню View Mode на Viewport гарантирует, что яркость освещения не будет регулироваться при переходе от светлых областей к темным, и позволяет легче сбалансировать освещение.

Размещение Light Portals

Light Portals — это специальные акторы, которые помогают Lightmass фокусировать фотоны к отверстиям, в которые поступает много света. Это улучшает время сборки и качество освещения.

Разместите этих акторов на ваших окнах и других отверстиях и свободно отрегулируйте коробку по размеру. Вам не нужно быть ужасно требовательным; это просто помощник для фотонов, и он не нуждается в точности.

Использование Reflection Probes

Отражения являются неотъемлемой частью PBR. UE4 использует акторов захвата отражения для захвата статических кубических карт сцены и автоматически применяет их к материалам, находящимся под их влиянием.

Хотя вы можете уйти, не размещая их вокруг своего уровня, вы достигнете гораздо более высокого качества с ними.

Два типа Reflection probes — сфера и куб. Проще говоря, для областей квадратной формы используйте Box Reflection Capture Actor. Для всех остальных областей используйте Sphere. Как видно из рисунка 14.14, для точной настройки отражений на сцене полезно смешение обоих видов.

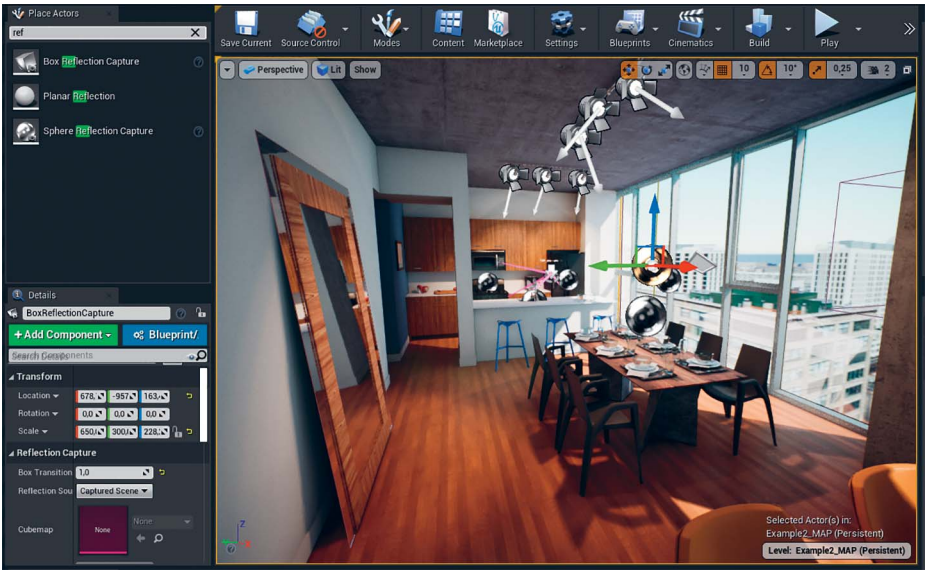


Рисунок 14.14 Reflection Probes

Размеры

Sphere Reflection Actors используют радиус влияния, чтобы определить, на какие пиксели они будут влиять. Как правило, Reflection Actors меньшего радиуса имеют приоритет над Reflection Actors большего радиуса. Это позволяет вам «гнездить» меньшие Reflection Actors внутри больших, позволяя добавлять детальные отражения в области, которые могут в этом нуждаться.

Box Reflection Capture Actors выставляют значение 3D-масштаба. Вы должны отрегулировать масштаб коробки до тех пор, пока углы коробки, показанной в окне просмотра, не будут максимально приближены к углам вашей комнаты.

Производительность

В дополнение к накладным расходам памяти, захваченной Cubemaps, Reflection Capture Actors автоматически применяют свои отражения к любым пикселям, которые визуализируются внутри них. По этой причине перекрывающиеся захваты могут повлиять на производительность. Вы должны избегать слишком большого количества накладывающихся друг на друга элементов.

Также стоит отметить, что Box Reflection Capture Actors оказывают более серезное влияние на производительность, чем Sphere Reflection Capture Actors.

Post-Process Volume

Последняя часть головоломки освещения состоит в том, чтобы применить эффекты постобработки, такие как виньетка, блум, размытие движения и глубина резкости к изображению, чтобы придать ему кинематографичный, реалистичный вид.

При создании нового уровня в UE4 настройки постобработки по умолчанию применяются к сцене автоматически. Эти настройки — хорошее начало, но вы почти всегда будете хотеть настроить их в каждой из ваших сцен, чтобы соответствовать ее уникальному освещению и достичь того внешнего вида, который вам нужен.

Чтобы получить доступ к этим настройкам, вам нужно добавить специальный класс актора на вашу сцену: Post-Process Volume.

Volume Actors

Volume — это особый тип класса акторов в UE4, который может определить, находятся ли другие акторы в его пределах. Post-Process Volumes используют эту способность смешиваться между различными постпроцессными настройками, когда камера игрока входит и выходит из каждого тома.

Вы можете настроить Volumes, установив их настройки Brush. Вы можете определить форму объема, а также его размер. Вы также можете масштабировать, вращать и перемещать их, как и любой другой актор.

Размещение Post-Process Volume

Вы найдете список различных типов классов объемов, доступных на панели Place Actors в разделе Volumes (рисунок 14.15). Перетащите Post-Process Volume в Viewport. Вы можете разместить его в любом месте сцены — просто убедитесь, что он легко выбирается.

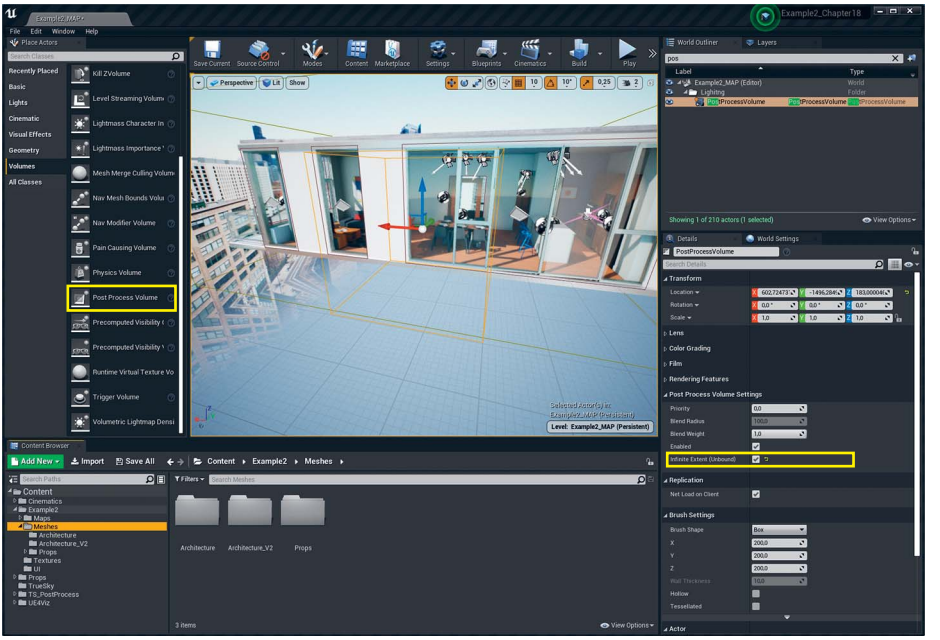


Рисунок 14.15 Размещение Post-Process Volume на Level

Настройка Post-Process Volume

Post-Process Volume имеет множество настроек и позволяет значительно улучшить ваше изображение. В то время как многие настройки — такие как настройка цвета и эффекты вроде цветения — почти полностью зависят от ваших личных вкусов и стилей, другие оказывают прямое влияние на производительность и качество изображения.

Чтобы получить доступ к настройкам, просто выберите Post-Process Volume на уровне и посмотрите на панель Details.

Ниже приведены настройки, которые вы будете настраивать чаще всего, а также те, которые были скорректированы для этого примера.

Unbound

Вы можете настроить свой Post-Process Volume, чтобы пропустить проверку границ и применить его настройки по всему уровню, включив опцию Unbound. Это гораздо проще, чем установить объем, чтобы охватить весь уровень.

Priority

Вы можете иметь Post-Process Volumes, которые перекрывают друг друга. Чтобы определить, какие настройки будут использоваться, примените опцию «приоритет».

Объемы с более высоким приоритетом будут переопределять объемы с более низким приоритетом.

Важно отметить, что будут применены только те свойства, которые переопределены вручную (установив флажок слева от свойства на панели Details для перекрывающегося объема). Это отличный способ настроить только одно свойство, не проверяя соответствие всех остальных параметров.

Blend Radius

Это радиус мирового пространства вокруг объема, который интерполируется между настройками двух объемов. По мере того как камера игрока перемещается в Volume и из него, настройки будут плавно переходить от одной к другой. У объемов, настроенных как несвязанные, этот параметр неактивен.

White Balance

В UE4 есть много настроек для цветокоррекции, но вам, вероятно, следует начать с баланса белого. Это значение по умолчанию равно 6500, что является очень чистым, сине-белым цветом, который не очень распространен в реальном мире. Это может придать вашим сценам слишком холодный тон и затруднить достижение правильного баланса между теплым светом и прохладными тенями. Вы можете установить значение примерно между 5000 и 6000 для более теплого тона на вашей сцене и выше — для более холодного взгляда.

Saturation and Contrast

Настройки **Saturation**, **Contrast**, **Crush Highlights** и **Crush Shadow** работают вместе, чтобы помочь контролировать баланс изображения. Вы, вероятно, знакомы с контрастом и насыщенностью, но, возможно, не с настройками подавления. Они просто обрезают черно-белые точки, давая больший видимый контраст.

Настройки по умолчанию часто слишком контрастны, что делает затененные области очень темными. Это очень кинематографичный взгляд, но результат может быть слишком темным для архитектурной визуализации, при чем совсем не там, где предпочтительнее иметь очень темные области.

Эти настройки очень чувствительны, поэтому небольшие корректировки имеют большое значение.

Vignette, Noise и Fringe

Используйте эти эффекты для имитации эффектов объектива камеры.

Виньетка добавляет темный градиент к краям вашего изображения, что может позволить вам увеличить общую яркость изображения — единственные полностью яркие пиксели находятся в самом центре экрана.

Бахрома имитирует эффект хроматической аберрации света, проходящего через объектив камеры, разделяя цвета ближе к краям изображения.

Зернистость добавляет анимированный шум к изображению. Интенсивность зерен управляет непрозрачностью наложения шумовой текстуры, в то время как дрожание зерен управляет тем, насколько зерно смещает изображение. Используйте их осторожно, так как они могут быстро стать сильно заметными.

Color Grading (LUT)

Система Color Grading в UE4 использует специальную текстуру, называемую Color Lookup Table (LUT), для изменения цвета сцены. LUT генерируется путем применения цветовой градации к базовому изображению в приложении для композиции или редактирования изображений.

UE4 считывает разницу между базовым изображением и модифицированным LUT и применяет эту дельту к сцене. Это отличный способ перенести существующий конвейер цветокоррекции в UE4. Вы можете найти более подробную информацию о LUTs и некоторых файлах LUT для загрузки на сопутствующем сайте для этой книги по адресу www.TomShannon3D.com/UnrealForViz

Bloom и Lens Flares

Некоторые из наиболее распространенных эффектов постобработки как в играх, так и в традиционно визуализируемом контенте, цветение и вспышки линз помогают изображать чрезмерно яркие области, имитируя объективы камер и человеческий глаз.

Настройки цветения и бликов объектива по умолчанию в UE4 немного агрессивны и могут уменьшить контрастность и четкость сцен, если они используются слишком часто. Уменьшение интенсивности или увеличение порога являются хорошими способами уменьшения общего количества цветения и бликов объектива в сцене.

Вы также можете настроить размер эффектов. Большие размеры будут иметь более значительное влияние на производительность.

Я часто полностью отключаю вспышки объектива в сценах или выключаю их до тех пор, пока они происходят только с очень яркими пикселями.

Auto Exposure

Поскольку UE4 визуализирует сцены с использованием среды HDR-освещения, вы можете иметь очень разные уровни яркости освещения от одной области вашей карты до другой. По этой причине UE4 имеет сложную систему автоматической экспозиции.

Эта система поможет создать динамическое, привлекательное взаимодействие освещения, когда игрок перемещается из одной области в другую и видит, как экспозиция настраивается в ответ так же, как человеческий глаз или камера с автоматической

экспозицией. Это также может сделать настройку вашего освещения довольно сложной задачей.

Рекомендую настроить освещение с отключенной автоматической экспозицией, а затем включить его по мере необходимости для сцены. Этот эффект можно отключить либо с помощью настроек экспозиции в раскрывающемся списке View Mode на Viewport, либо установив параметры Min and Max Brightness в Post-Process Volume равными 1,0. Затем вы можете использовать Exposure Bias, чтобы вручную установить экспозицию камеры.

После настройки освещения начните играть с Min and Max Brightness, чтобы настроить экспозицию камеры. Чтобы разрешить камере переэкспонировать, осветляя темные области, установите минимальную яркость на значение ниже 1,0. Чтобы уменьшить экспозицию камеры при ярком освещении, увеличьте максимальную яркость выше 1,0.

Ambient Occlusion

Хотя в данном примере мы не используем Screen Space Ambient Occlusion (SSAO), это очень важный эффект, который широко используется во всех видах визуализаций и игр. Если вы создаете сцены с динамическим освещением или большим количеством динамических акторов, вы, вероятно, захотите включить это, потому что это значительно увеличивает глубину сцены и качество освещения.

Глобальное освещение

Это управляет интенсивностью и цветом Lightmaps, генерируемых Lightmass. Вы можете использовать это, чтобы быстро изменить свое испеченное освещение. Это не управляет никаким видом GI в реальном времени в UE4.

Глубина резкости (Depth of Field, DOF)

UE4 предоставляет несколько методов для создания эффектов глубины резкости. Для визуализации **круг DOF** легко является лучшим. Это физически точный эффект, который имитирует фактические характеристики размытия диафрагмы объектива и создает очень тонкий, реалистичный эффект. Он также довольно эффективен по сравнению с другими эффектами. Однако, как и многие эффекты постобработки, он может быть дорогим при более высоких разрешениях. Я вообще не рекомендую его для виртуальной реальности.

Чтобы увеличить эффект размытия, уменьшите значение параметра **Aperture F-Stop**. Более низкие настройки приведут к большему размытию изображения (рисунок 14.16). Важно отметить, что это реалистичный эффект, поэтому вы можете не заметить его, пока не подойдете очень близко к объекту.

Для внутренней сцены вам нужно будет убедиться, что ваше **фокальное расстояние** установлено примерно на 300–500 (от 3 до 5 метров), а **Aperture F-Stop** довольно велика на уровне 4–8.

Рисунок 14.16 Circle Depth сравнение Field F-Stop

Размытие изображения при повороте камеры (Motion Blur)

UE4 использует высококачественную систему размытия движения, которая генерирует карту скорости каждого кадра и использует ее для соответствующего размытия сцены. Настройки по умолчанию, как правило, очень хороши для большинства сцен; однако, если вы работаете с более высокой частотой кадров или хотите более чистую презентацию, то можете отключить эту функцию или уменьшить ее значение, уменьшив параметр **Max**.

Отражения в экранном пространстве (Screen Space Reflections)

Screen Space Reflections обеспечивают детальные динамические отражения, основанные на визуализированном изображении. Они необходимы для достижения самого высокого качества, но вы захотите увеличить Quality и Max Roughness. Установите

Quality на 100 и Max Roughness между 0,6 и 1,0. Более высокие значения дороже визуализировать, но они могут выглядеть более точными. Устанавливайте значения как можно ниже, сохраняя при этом внешний вид вашей сцены.

Anti-Aliasing

UE4 предлагает несколько методов сглаживания (AA). Для большинства визуализаций Temporal AA (TAA) система в UE4 дает превосходные результаты и имеет очень мало накладных расходов на производительность.

В следующем примере сцены показан результат включения параметров окружающей окклюзии, виньетки, зернистости и глубины резкости. Каждый из них улучшает внешний вид изображения, добавляя несовершенства и другие эффекты, которые можно увидеть на фотографии.

Как видите, настройки в постобработке пространства оказывают серьезное влияние на внешний вид сцены. Даже тонкие настройки, используемые здесь, значительно изменяют внешний вид сцены (рисунки 14.17 и 14.18).

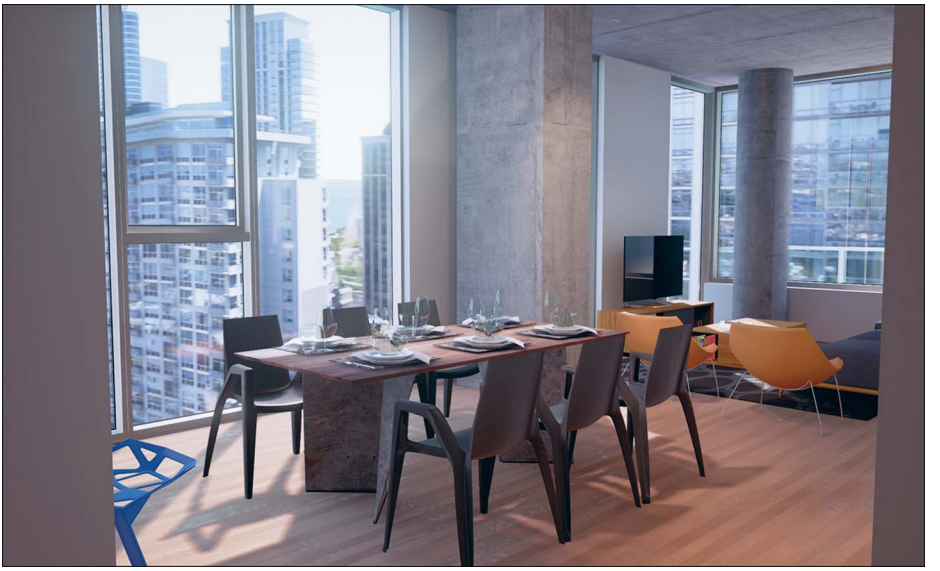


Рисунок 14.17 Сцена с настройками по умолчанию у постобработки

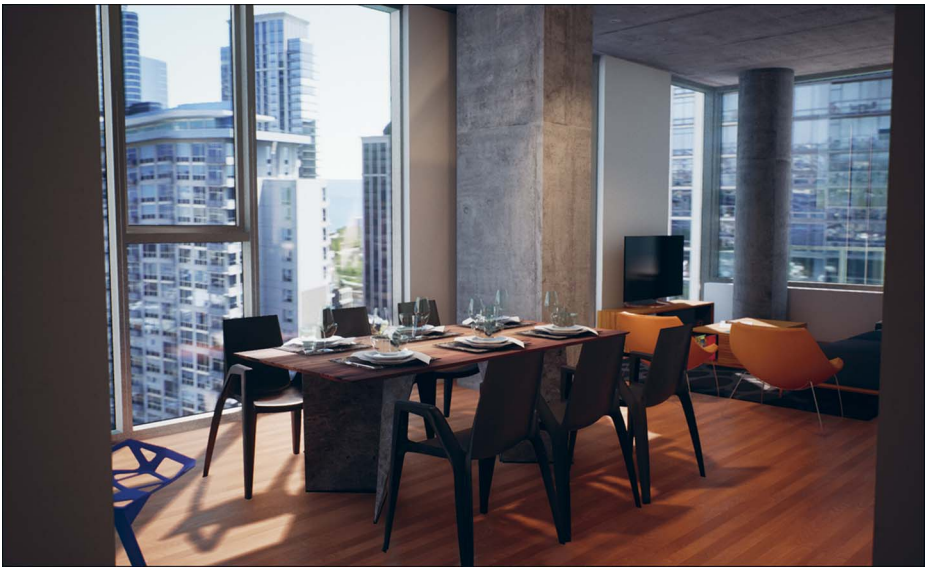


Рисунок 14.18 Сцена с настройками постобработки, показывающими заметное изменение контраста, баланса белого и насыщенности

Заключение

Освещение в UE4 имеет много компонентов, которые работают вместе, чтобы создать единое целое. Хотя эти понятия были представлены в том порядке, в котором вы могли бы с ними столкнуться, по мере изучения и повторения на вашей сцене вы станете перемещаться между их настройками, потому что настройка одного влияет на другое.

Подобное взаимодействие между светом, цветом и материалами знакомо художникам визуализации, и именно в этой точке технология расходится с художественностью. Благодаря практике вы научитесь понимать свой инструмент и начнете легко создавать теплое освещение в UE4.

ГЛАВА 15

АРХИТЕКТУРНЫЕ МАТЕРИАЛЫ

Система материалов в UE4 как хорошая видеоигра: проста для изучения, но занимает целую жизнь, если ты надумал стать мастером. Создание хороших материалов UE4 для производства — это не только о качествах художника, но и о создании повторно используемых материалов, получении лучшей из возможных производительности и изучении новых техник, которые не используются при рендеринге с трассировкой лучей, таких как Parallax Occlusion Mapping.

Материалы и освещение работают рука об руку для создания богатого и реалистичного окружения для игрока. Вы можете достичь потрясающих результатов с помощью простых материалов. Теперь, когда запущено освещение, вы можете добавить несколько материалов на сцену и оживить их с помощью цвета, отражения, деталями и вариациями поверхности.

Получить хорошо выглядящий материал в UE4 просто, потому что использовать PBR (physically based rendering) легко. Определите параметры Base Color, Roughness, Metallic, Normal, а движок сделает сложную работу и породит физически корректный материал для потрясающих поверхностей ваших сцен.

Обязательно ознакомьтесь с главой 5, «Материалы», чтобы узнать об основах создания материалов в UE4, особенно о создании сущностей материалов. Вы будете полагаться на них для быстрого создания больших библиотек материалов и их применения к сценам.

Что такое мастер-материал?

Как уже обсуждалось в главе 5, вы можете создать уникальный материал для каждой поверхности, как и в обычном 3D-редакторе. Это может занять очень много времени, особенно когда вы начинаете добавлять в материалы много функционала.

Вместо этого вы будете использовать специальные параметры, **Material Parameters,** для создания одного материала (**Material**) и затем нескольких экземпляров материала (**Material Instances**), назначая текстуры и переопределяя свойства для создания практически каждого материала, используемого на сцене.

Этот единственный материал часто называют Master Material. Master Material не является конкретным ассетом в Unreal Engine. Скорее это идея. Любой материал может быть Master Material, если вы создадите его с параметрами Material Parameters. Material Parameters предоставляют переменные для Material Instances, которые могут быть изменены на лету, как в редакторе, так и в процессе работы с помощью Blueprints.

Благодаря простоте основанного на физике рендеринга сеть ваших материалов не будет слишком сложной. Используя только Color, Normal, Metallic и Roughness Textures вместе со случайной картой высот, вы сможете определить практически любую поверхность.

Вы даже сможете создать обычно сложные поверхности, такие как металл, стекло, и архитектурные настенные покрытия с минимальными усилиями или без необходимости создавать сложные пользовательские материалы. Вы можете продолжить создать собственный материал или скачать готовые файлы проекта на www.TomShannon3D.com/UnrealForViz.

Обзор работы материалов

На рисунке 15.1 представлен Master Material. На первый взгляд он может показаться сложным, но на самом деле все просто.

Материалы похожи на Blueprints и используют основанный на узлах граф (Material Graph), чтобы помочь визуализировать то, что на самом деле является понятиями из программирования. Узлы, соединенные с другими узлами и потоками данных слева направо, в конце сходятся в одном из различных **атрибутов** материала, таких как **Base Color** или **Roughness**.

Вы возьмете концепции, описанные в главе 5, и расширите их, добавив немного новых типов узлов, которые обеспечат большую гибкость для экземпляров материалов или предоставят продвинутые функции рендеринга, такие как Parallax Occlusion Mapping.

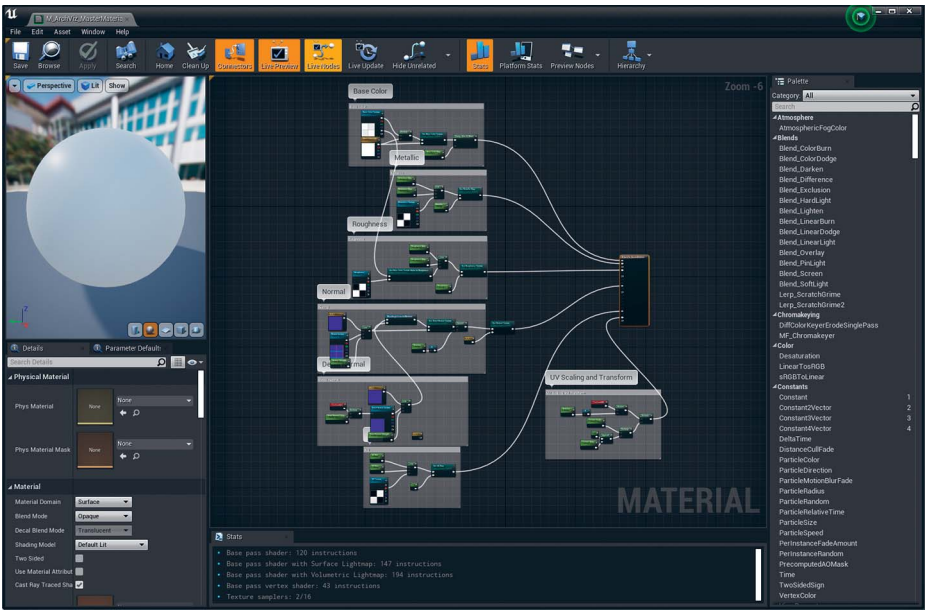


Рисунок 15.1 Завершенный M_ArchViz_MasterMaterial shader graph, представляющий один Material, который может быть использован для почти каждого Material на сцене через Material Instances

Узлы параметров

Существуют особые виды вершин, Material Expression Nodes, которые вы можете разместить в Material Graph, называемые Parameters. Эти узлы позволяют вам раскрывать некоторые аспекты вашего материала, который может быть динамически изменен с помощью Material Instance assets или в процессе работы с помощью Blueprints.

Параметры, созданные в материале, отображаются как редактируемые Properties в Material Instances, полученные из этого материала. В Material Instance Editor (рисунок 15.2), вам нужно установить галочку слева от свойства, которые вы хотите переопределить до изменения значения. Для возврата значения по умолчанию нажмите на маленькую желтую стрелку рядом с измененным свойством. Вы также можете убрать галочку для отмены переопределения.

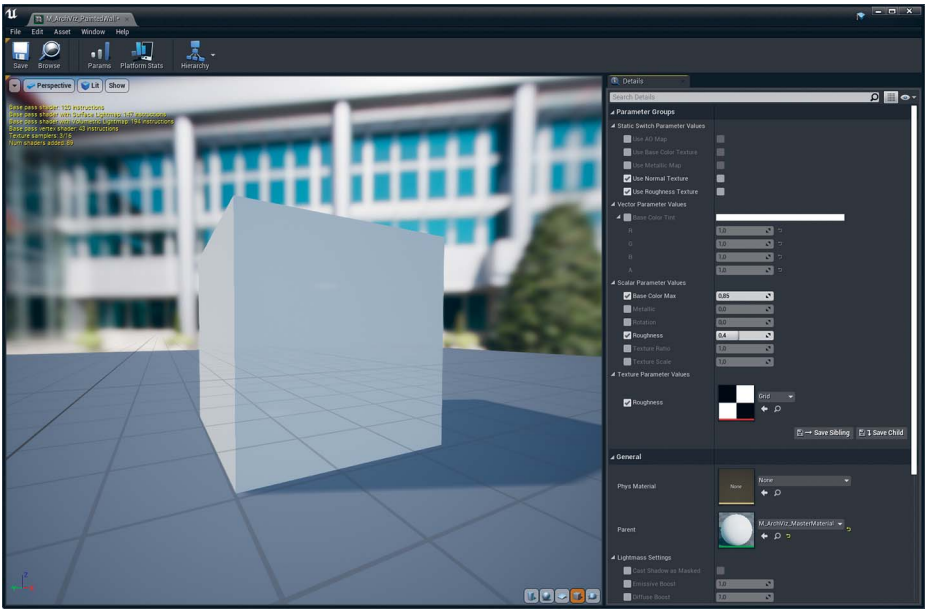


Рисунок 15.2 Раскрашенный Wall Material Instance, основанный на M_ArchViz_MasterMaterial

Создание мастер-материала

Как показано на рисунке 15.1, нужно очень многое настроить для создания хорошего Master Material. Каждый Material input (Base Color, Metallic, Roughness, and Normal) имеет ряд узлов, соединенных с ним, что позволяет вам создавать бесконечные вариации ваших материалов с Material Instances.

Давайте пройдемся по каждому input и посмотрим, как он настроен. Вы можете продолжить создавать ваш собственный Material или получить эти файлы на www.TomShannon3D.com/UnrealForViz.

Base Color

На рисунке 15.3 показан **Texture Parameter 2D**, называемый **Base Color Texture**, который умножается на **Vector Parameter**, называемый **Base Color Tint**. Помните, что Materials — это Math, а Colors как RGB Vectors. Это значит, что вы можете выполнять все

возможные математические операции для вектора цвета в текстурах, и это позволит вам делать корректировки на лету.

Adding Parameters

Для размещения узла **Texture Parameter** просто используйте палитру или меню, вызванное нажатием правой кнопкой мыши, и выберите Texture Parameter из списка. Затем назовите Parameter чем-нибудь значимым. Parameters могут содержать пробелы и знаки препинания в их названиях, делая их легко читаемыми, но это может затруднить доступ к этим переменным кода.

Узлы Parameter могут быть переименованы и изменены с помощью панели Details. Используя панель Details, вам также нужно определить Texture asset, который будет использоваться для этого узла по умолчанию. Вы можете либо нажать по миниатюре для вызова browser, либо перетащить Texture из Content Browser в свойства[14] узла.

Разместите **Vector Parameter** таким же образом, снова присвоив ему осмысленное название. Vector Parameter можно отредактировать, изменив значения RGBA в панели Details, или двойным нажатием на образец цвета в узле. Это вызовет палитру цветов, которой гораздо проще пользоваться.

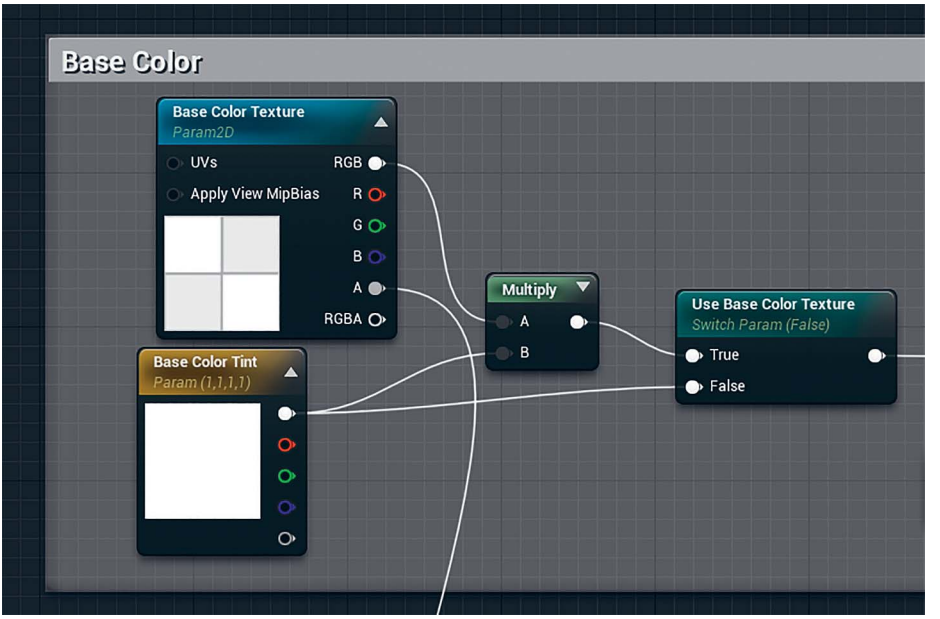


Рисунок 15.3 Узел Base Color

[14] Кроме того, вы можете создать вершину Texture 2D, перетащив текстуру из Content Browser в граф материала. Сразу такая вершина не станет вершиной Parameter, но ее можно легко преобразовать в таковую двойным щелчком ПКМ на вершене Texture 2D и выбором команды Convert to Parameter в контекстном меню.

Multiplying Colors

Добавьте узел **Multiply** с помощью Palette, или меню, вызываемого двойным нажатием правой кнопки мыши, и подключите выход Base Color Texture к входу A и выход Base Color Tint к входу B.

Узел Multiply умножает каждый канал входа A на вход B и возвращает результат. Умножение вектора (выходной узел base Color Texture's RGB) на другой вектор (выходной узел BaseColor Tint's RGB) означает, что каждый из отдельных RGB-каналов умножается друг на друга (RedA x RedB, BlueA x BlueB, GreenA x GreenB). Вы также можете умножить один тип данных на другой, к примеру Vector и Scalar (float), — в данном случае будет умножаться каждый канал Vector на значение Scalar.

Использование узла Multiply для цветов аналогично Multiply blend mode, с которым вы, возможно, сталкивались во многих приложениях. Черный (0,0,0), полностью окрашивает ваш Base Color в черный, так как все умноженное на 0 равно 0. Чистый белый (1,1,1) не изменяет входящее значение вообще. Конечно, вы можете выбрать цвет, тонируя пиксели. Вы также можете превысить 1 (или 0) в вашем Vector Parameter, действуя как регулировка яркости для Texture. Это иногда может привести к физически неточным результатам, поэтому будьте осторожны, когда переопределяете ваши входные данные этим способом.

Static Switch Parameters

Узел **Static Switch Parameter** отображает Boolean флажок в Material Instances. Если параметр установлен на true (в Material или Material Instances, от которого получает его), Material будет оценивать путь кода, соединенный с входом True (и с False в противном случае).

Изменение значения Static Switch также может обновить интерфейс Material Instance. Parameters, которые не вызываются, такие как Base Color Texture, не будут показаны в Material Instance Editor, что приведет к уменьшению помех и избеганию показывания пользователю параметров, которые ничего не делают.

Соедините результат узла Multiply с входом True, а выход Base Color Tint Parameter — с входом False в Static Switch Parameter.

В данном случае, если **Use Base Color Texture** равен *false*, Material будет использовать Base Color Tint для определения Base Color, минуя узлы Texture и Multiply. Это экономит на чтении Texture и вычислении Shader, делая более производительный Material.

Metallic

Входной канал **Metallic** является одним из наименее используемых атрибутов и может быть установлен на 0 или 1 через **Metallic Scalar Parameter** (рисунок 15.4).

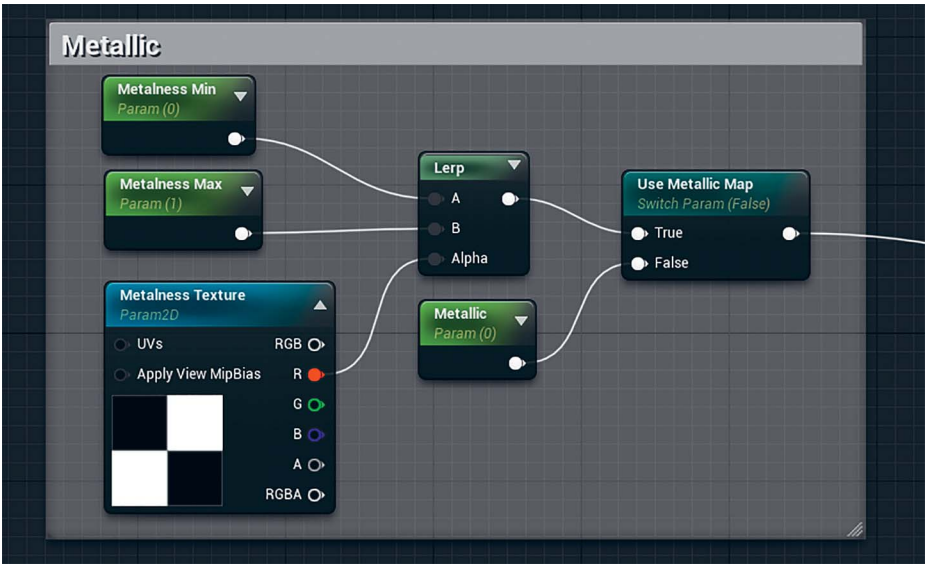


Рисунок 15.4 Узел Metallic

Однако иногда вы можете захотеть использовать Texture как маску для определения области модели, какие являются metallic, а какие нет; к примеру, текстура древесины с видимыми гвоздями или шурупами или материал, представляющий сколотую краску с металлической поверхностью под ней.

Для учета этого создайте узел **Texture Parameter 2D** под названием **Metalness Texture**.

Расположите узел **Linear Interpolate** (или **Lerp**, как его обычно называют), который позволит вам переназначать значения в другие значения, используя простые Min и Max Parameters для каждого. Alpha-вход действует как процентный вес между входами A и B, значение 0.0 соответствует входу A и значение 1.0 соответствует входу B. Регулируя эти значения, вы можете легко интерактивно настроить поверхности ваших сцен.

Создайте два Scalar Parameters, назовите их **Metalness Min** и **Metalness Max** и подключите их к входам A и B узла Lerp. Установите Default Value для Metalness Max на 1.0 с помощью панели details. Настройка параметров **Metalness Min** и **Metalness Max** позволит вам легко изменять значение маски без изменения Texture (см. главу 5 для дополнительной информации).

Чтобы полностью отключить Texture, разместите другой Static Switch Parameter в Graph и назовите его. Если для этого Parameter установлено *true*, то выбирается красный канал **Metalness Texture** и затем изменяется с помощью узла Lerp. Поскольку для входа

Metallic требуется только градация серого или скаляр (0–1), только красный канал текстуры используется, как альфа узла[15] Lerp.

Если **Use Metallic Map** установлен на false, он будет использовать **Metallic Scalar**, который установлен на 0,0 и подключен к выходу False.

Roughness

Канал Roughness, возможно, является самым важным каналом, не считая Base Color (иногда больше, как обсуждалось в главе 5). Однако, как вы можете заметить на рисунке 15.5, он настроен так же, как и другие каналы, с **Use Roughness Texture** Static Switch Parameter переключаясь между Texture-based roughness и **Roughness Scalar** Parameter.

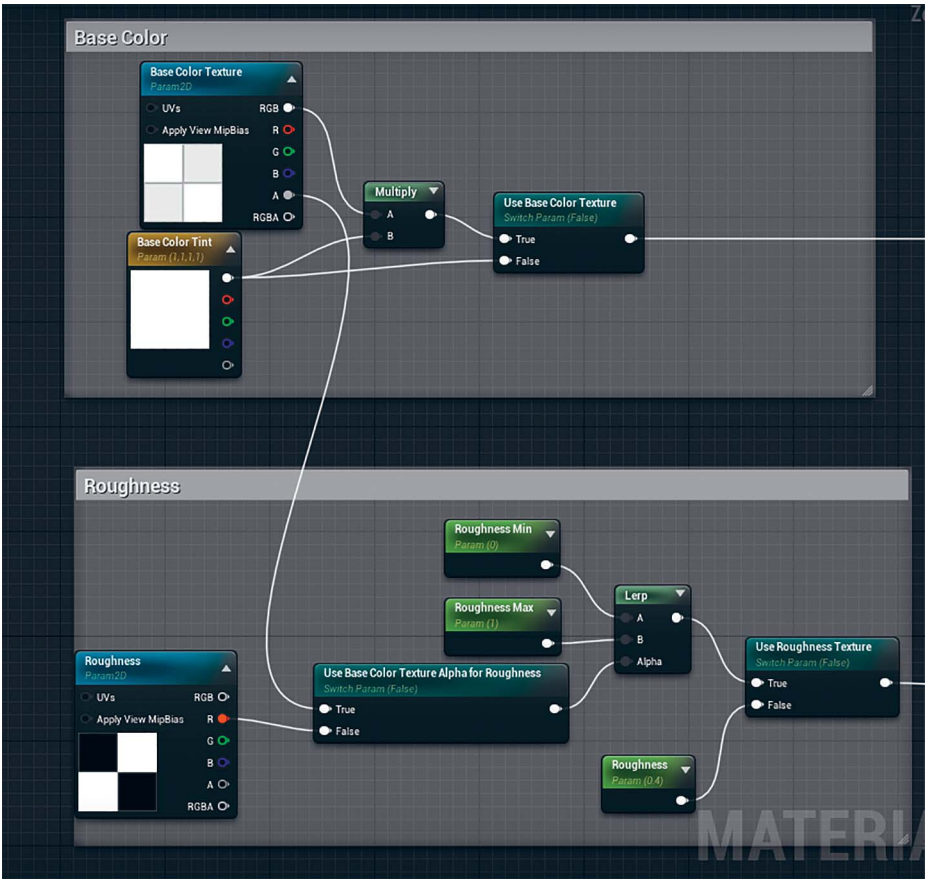


Рисунок 15.5 Узел графа Roughness также показывает Base Color, поэтому вы можете заметить, как Alpha из Base Color Texture может быть использована в качестве Roughness map

[15] Использование только одного канала из Texture позволяет продвинутым пользователям применять каждый канал для хранения RGB-текстуры различных изображений в градациях серого.

Создайте ваш Material Graph, как показано на рисунке 15.5. Обратите внимание на значение по умолчанию для каждого узла Parameter при подготовке построения вашей сети.

Use Base Color Texture Alpha for Roughness Static Switch Parameter позволяет использовать Alpha канал **Base Color Texture** как маску Roughness или красный канал **Roughness Texture Parameter**. Использование Alpha из Base Color Texture является обычным рабочим процессом, и многие Assets, доступные в Marketplace и в сообществе, используют этот метод для определения Roughness в Materials.

Другой Static Switch Parameter, называемый **Use Roughness Texture**, позволит вам переключаться между одним скалярным значением Roughness, который соединен с входом False, и путем на основе Texture, который подходит к входу True. Если Use Roughness Texture установлен на *true*, то данные текстуры вернутся к Use Base Color Texture Alpha для Roughness Parameter, управляемым с помощью узла Lerp и **Roughness Min** и **Roughness Max** Scalar Parameters.

Normal

Канал **Normal**, вероятно, наименее знаком большинству визуализирующих художников. Большинство 3D-приложений полагаются на карты рельефа и высот для определения поверхности. Работающие в реальном времени приложения, включая UE4, используют карту нормалей, потому что они быстрее вычисляются, чем карта рельефа, и могут определить кривизну поверхности, что делает их более качественными, чем карты рельефа.

Создайте узел Normal Texture Parameter, как и любой другой Texture Parameter. Однако вы должны установить **Sampler Type** на *Normal* в панели Details узла. Это делается автоматически, если вы назначили карту нормалей текстуре для свойства узла Texture.

Для контроля интенсивности карты нормалей используйте Lerp между значением Normal Texture, подключенным к входу A, и значением Constant Vector 0,0,1 (значение для не измененной нормали), подключенным к входу B (рисунок 15.6).

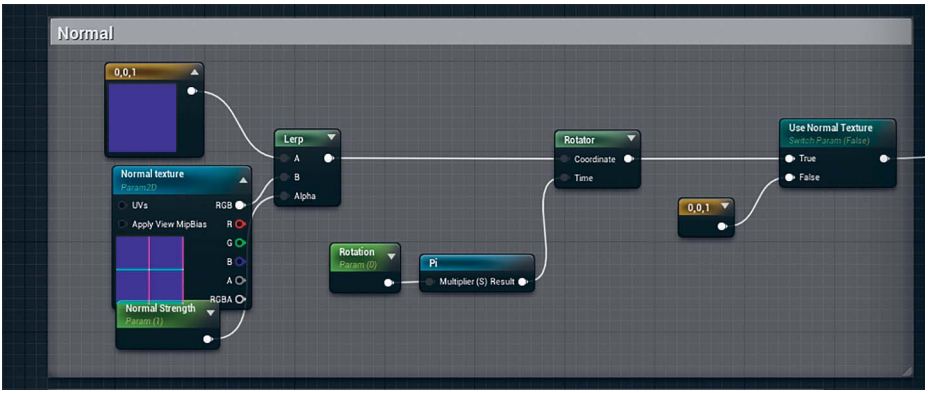


Рисунок 15.6 Узел Normal

Для учета вращения карты нормалей требуется изменить сохранения правильной векторной математики. Используйте узел **Rotator** для вращения данных, возвращенных из Lerp. Scalar Parameter, называемый **Rotation**, умножается на Pi и используется как Time input для узла Rotator, связанного с UV и Texture.

Преобразования (объясню позже). Чтобы было понятно, вы поворачиваете *значение* карты нормалей, а не Texture.

Векторная математика, как эта, является основой 3D-приложений. Все, от перемещения в 3D-пространстве до цветов материалов, является векторами. Изучение векторной математики является одной из лучших вещей, которую вы можете сделать как художник, стремящийся улучшить свои возможности и навыки в каждом аспекте для UE4.

Ambient Occlusion

Дополнительная карта Ambient Occlusion (AO) предназначена для ручного определения микроповерхностной ambient occlusion of a Material (рисунок 15.7). Если этот ввод не определен, UE4 будет динамически генерировать эти данные с помощью карты нормали, но иногда они не совсем точные или это не то, что хочет художник. Канал AO чаще всего используется, когда наборы текстур уже запечены с помощью таких программ, как Substance designer или XNormal, и может быть отключен в большинстве случаев.

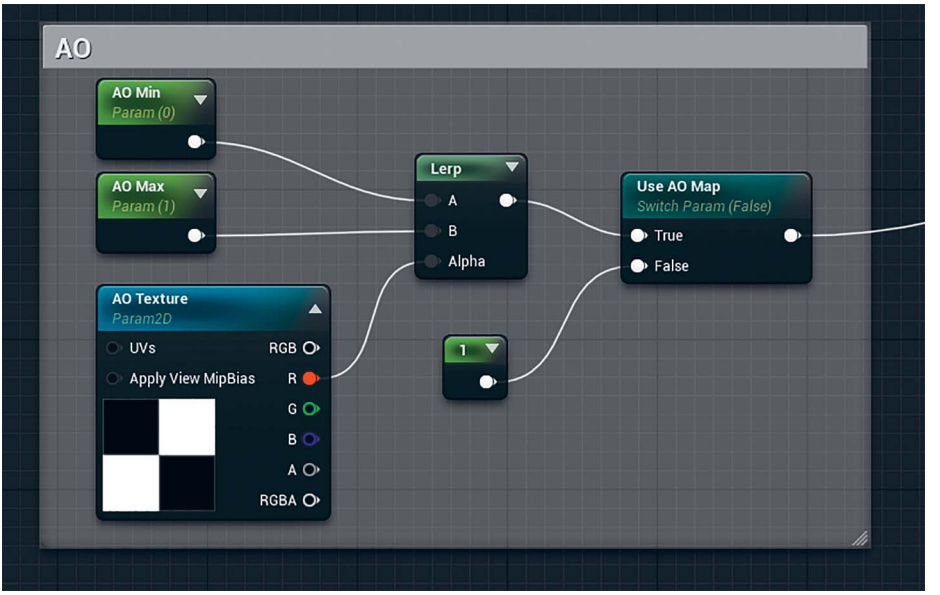


Рисунок 15.7 Узел Ambient Occlusion обычно используется только при особых обстоятельствах, когда материалы UER4 нуждаются в небольшой помощи

Настройте граф, как показано на рисунке 15.7. Как входы roughness и Metallic, AO нужен только скаляр или градация серого; следовательно, используется только красный канал **AO Texture** Parameter.

Использование **AO Map** Static Switch Parameter позволяет Material Instances пропустить использование AO map вообще. В отличие от некоторых других Switch Parameters, где на вход False подается Scalar Parameter, здесь мы будем просто назначать узел **Constant** и устанавливать его 1.0. В отличие от Scalar Parameters, Constant переменные не могут быть изменены в процессе работы.

Texture Scaling и Transform

Настройка масштаба и позиций Textures необходима для правильного оформления материалов. В UE4 вместо настройки масштаба Texture, как в 3D-приложении, вы изменяете UV-координаты поверхности в реальном времени с помощью shader network (рисунок 15.8).

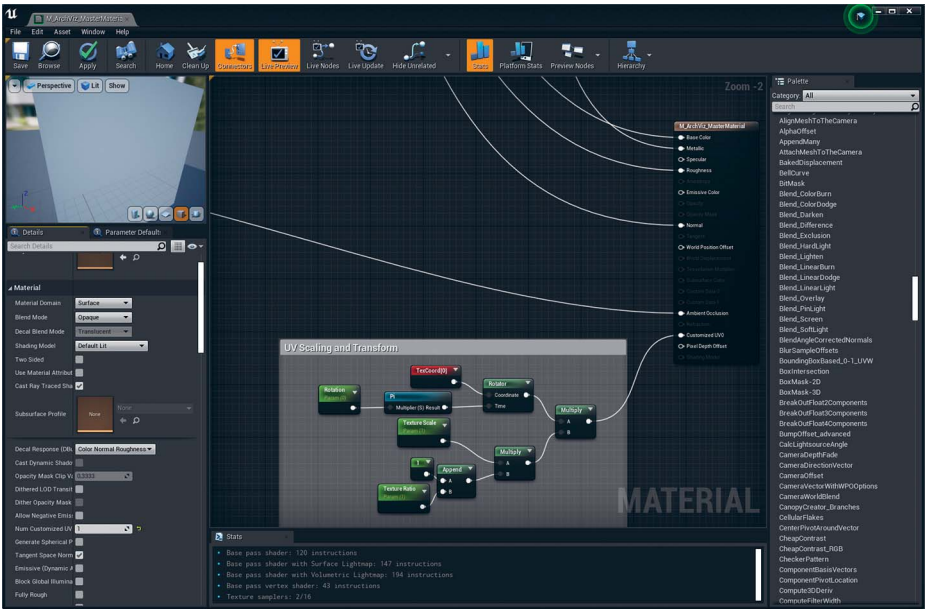


Рисунок 15.8 UV Scaling и Transform

Начните с добавление узла **Texture Coordinate** (названного TexCoord[0] на рисунке 15.8). Этот узел возвращает UV-координаты в определенном UV-канале.

Затем изменяете координаты UV, которые он возвращает от первого поворота его вокруг центра Texture, используя **Rotator**. Этот узел приводится в действие путем

подключения **Rotation** Scalar Parameter к Time input[16]. Для упрощения создания Rotation Parameter в Material Instances он умножается на число pi, превращая **Rotation** в диапазон 0,0–1,0, где 0,5 равно 180 градусам.

Для масштабирования координат и, следовательно, масштабирования/наложения Texture, они должны быть умножены. **Texture Scale** Parameter выполняет эту задачу. Значения, превышающие 1, приводят к более крупным тайлам в мозаичной Texture, в то время как значения меньше 0 приводят к тому, что Texture кажется больше, а тайлы меньше.

Texture Ratio Parameter позволяет текстуре быть неоднородно масштабированной. Узел **Append** создает значение *Vector2* (значение с двумя float; в данном случае каналы U и V, например 0, 0 или 0,2, 1,0) с **Constant** 1,0 в качестве первого значения и Texture Ratio в качестве второго. Это возвращает значение vector2D, например 1,0, 0,5, которые затем умножаются на Texture Scale Parameter, и этот результат умножается на rotated coordinates.

Результатом всего этого являются измененные UV-координаты, которые вы затем подключаете к атрибуту **Customized UV0** для Material. Этот вход не виден по умолчанию. Для включения этой опции вы должны сначала установить свойство **Num Customized UVs** для Material. Установка его на 1 или больше добавит входные узлы Customized UV для Material.

Custom UVs позволяет избежать необходимости соединять преобразованные UV-координаты к каждому входу UV всех узлов Texture в вашем Material. Вместо этого данные, поступающие к входам Custom UV, меняют координаты канала UV. Теперь к любой Texture, которая настроена для использования UV-координат канала, будут применены эти изменения значений UV.

Создание Material Instances

На рисунке 15.9 показывается множество Material Instances, используемых на сцене. Почти все из этих Instances основаны на одном Master Material, описанном ранее. К каждому Material Instance применили новые Textures и настройки Parameters, создавая разнообразную Material Library, доступную здесь.

[16] Rotation Parameter используется несколько раз в этом шейдере. Если у параметров одинаковые названия, изменение одного значения повлияет на все узлы с этим названием.

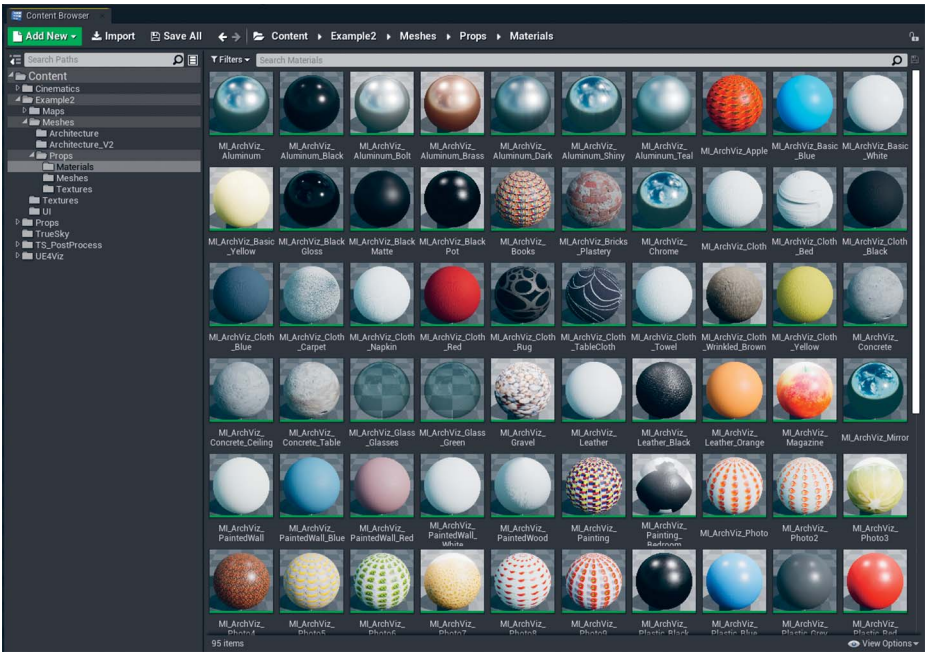


Рисунок 15.9 Content Browser с Materials и Material Instances, унаследованные от Master Material

Раскраска стен

Для этой сцены потребуется множество красок разных цветов. Для каждого создается Material Instance, каждый цвет определяется с помощью изменения Base Color Tint Parameter. Однако, как вы можете заметить на рисунке 15.10, мне также пришлось настроить множество Parameters, чтобы получить правильно выглядящий Material Instance: мне пришлось назначить Textures, переключить Static Switch Parameters и изменить скалярные значения, что в результате дало богато выглядящую поверхность стены.

Все различные цветовые вариации окрашенных стен происходят от одного Master Material; однако они унаследованы от другого Material Instance, а не напрямую от Master Material (рисунок 15.11). Это демонстрирует одну из самых мощных возможностей Material Instances: способность наследовать Material Instances от других Material Instances.

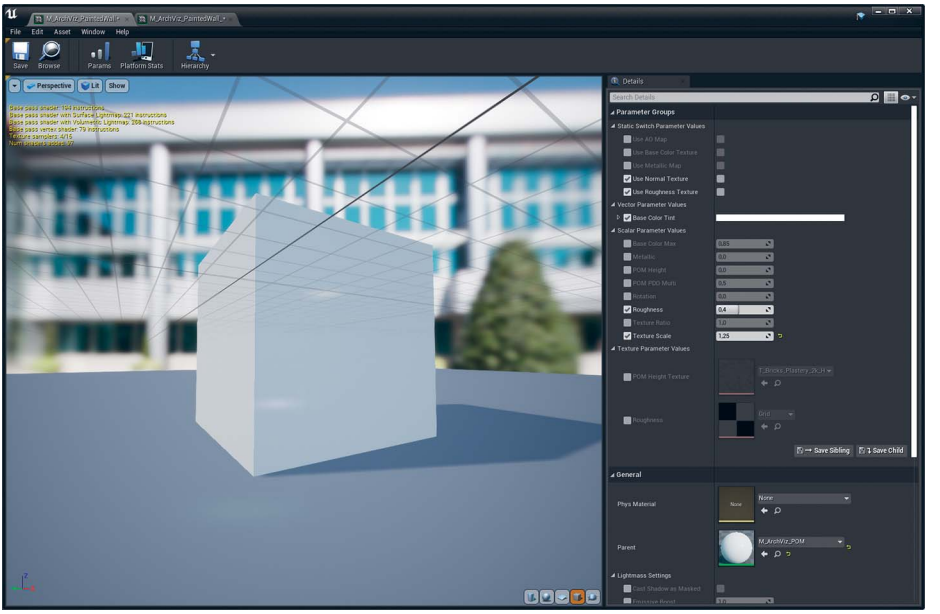


Рисунок 15.10 Раскрашенный Wall Material Instance с множеством измененных Parameters, точно настроенных для нужного вида Material

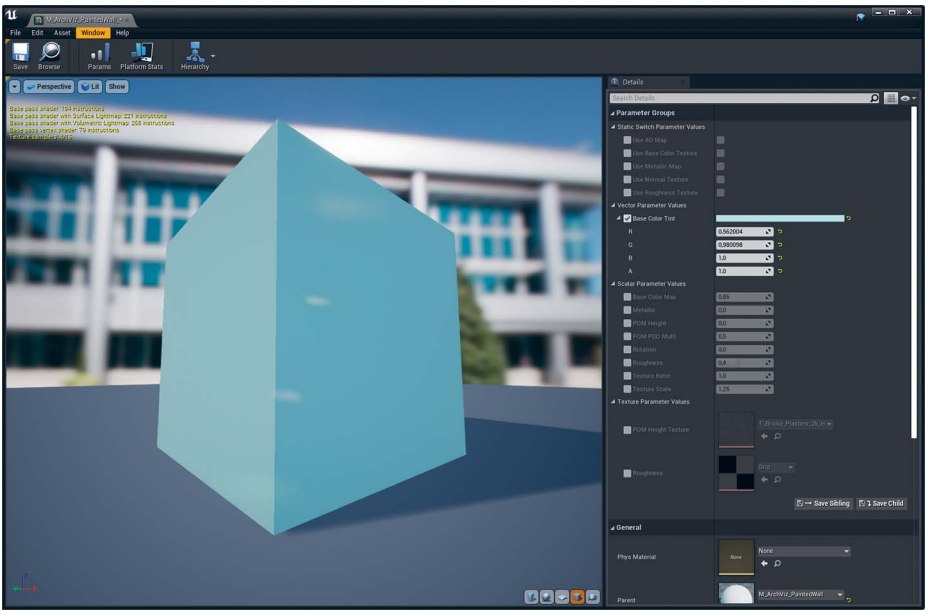


Рисунок 15.11 M_ArchViz_PaintedWall _Blue Material Instance, унаследованные от другого Material Instance: M_ArchViz_PaintedWall

Создание Material Instances от другого Material Instances позволяет вам настроить один Master Material Instance, изменение которого будет затрагивать все Material Instances, унаследованные от него.

Это позволяет создавать бесконечные вариации по данной теме и затем корректировать их все на лету без необходимости изменять каждый отдельно. Свойства, переопределенные в наследном Material Instances, такие как Base Color Tint, не будут изменены изменениями в родителе и сохранят любые сделанные вами изменения.

Полы

Полы могут быть одними из самых простых Material Instances, но в тоже время одними из самых важных. Получение отражений и деталей прямо на полах в визуализациях важно, так как это обеспечивает контакт с мебелью, стенами и другими Props (рисунок 15.12). Пол составляет огромный процент от привычного вида окружения, таким образом настройка и улучшение напольных материалов стоит усилий.

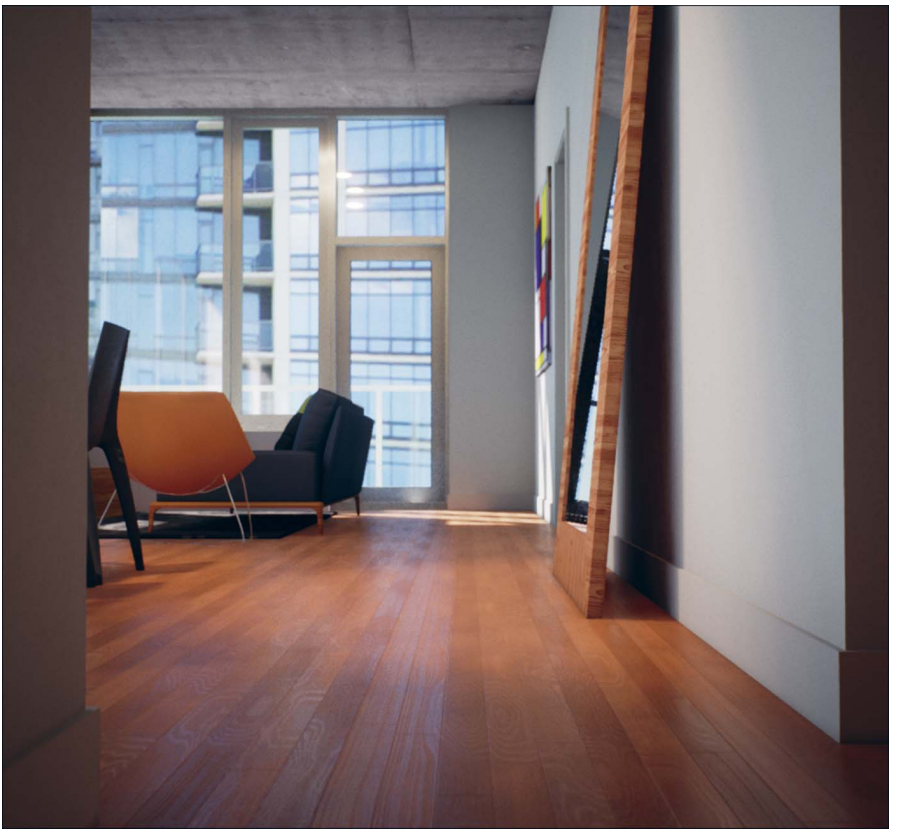


Рисунок 15.12 Напольный Material Instance зависит от хороших карт нормалей и Roughness, которые были сильно изменены с помощью Material Instance Parameters

Используя образец Texture для половых досок, которые использовались клиентом, я сгенерировал карты Normal, Base Color и Roughness Texture с помощью Substance Bitmap 2 Material (B2M). Это одно из множества доступных платных приложений, которое использует методы обработки на простом изображении и пытается сгенерировать PBR Textures для использования в таких игровых движках, как UE4.

Как вы можете заметить на рисунке 15.13, несмотря на то что Textures, созданные с B2M, хороши, их все еще нужно много дорабатывать. Вот почему настройка Parameters в Materials, позволяющая вносить изменения, так важна. Без нее вам нужно будет изменять Texture Assets, повторно импортируя каждый раз, чтобы увидеть изменения. Используя этот способ, вы можете посмотреть, как должен выглядеть материал в контексте.

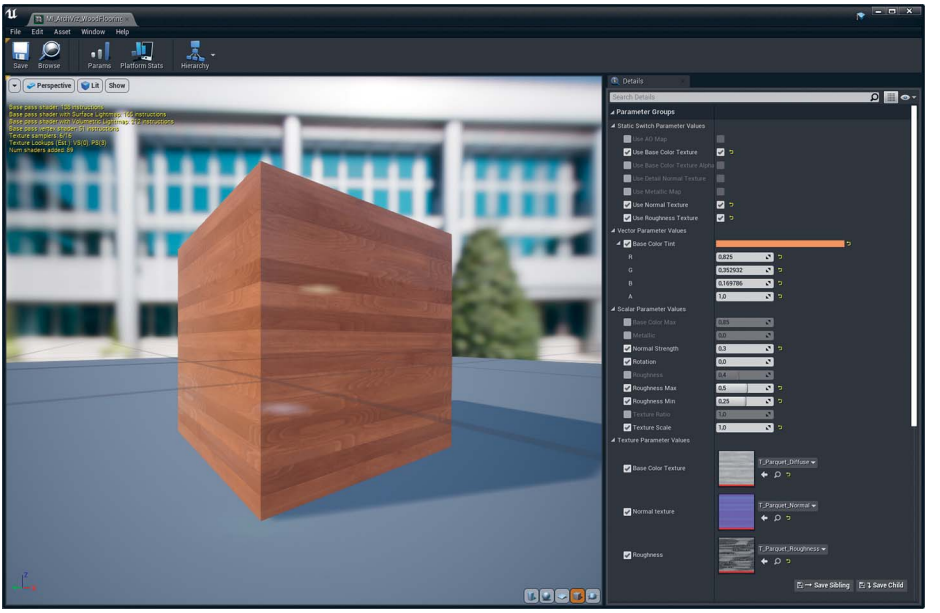


Рисунок 15.13 M_ArchViz_WoodFlooring Material Instance

Сложные материалы

Несмотря на то что мастер-материал превосходно подходит для большинства непрозрачных поверхностей, он не может справиться со всем. Такие материалы, как стекло или кирпич, требуют более специфичных материалов, чтобы выглядеть лучше, и использования преимущества от возможностей рендеринга UE4.

Parallax Occlusion Mapping

Parallax Occlusion Mapping (POM) является методом для создания похожих на смещение эффектов, на который уходит часть от стоимости рендеринга. Посредством некоторой причудливой математики можно использовать карту высот для смещения Textures, создавая иллюзию глубины (рисунок 15.14).

Рисунок 15.14 Тот же Material, только с включенным POM, показывает, как создается глубина, и помогает лучше определить поверхность, чем только одна карта нормалей

POM создает искаженные UV-координаты, которые затем подаются в каждый Texture's UV input. Вся эта сложная математика сделана для вас и настроена Material Function, которой требуется только несколько входных данных, например **Height Map** для корректной работы.

Так как POM влияет на каждый канал вашего Material, разбить его на собственный Material обычно является хорошей идеей. Настройка Material graph с учетом всех ваших других параметров может привести к чрезмерно сложному Material, который выполняет слишком многое и может стать бременем для обслуживания.

Как вы можете заметить на рисунке 15.15, версия POM для Master Material почти такая же, как и у стандартного Master Material. По факту я создал его, дублировав Master Material в Content Browser и затем отредактировав его. Самым большим дополнением является Parallax Occlusion Mapping Function вдалеке слева. Если вы присмотритесь (рисунок 15.16), то увидите, что его легко настроить.

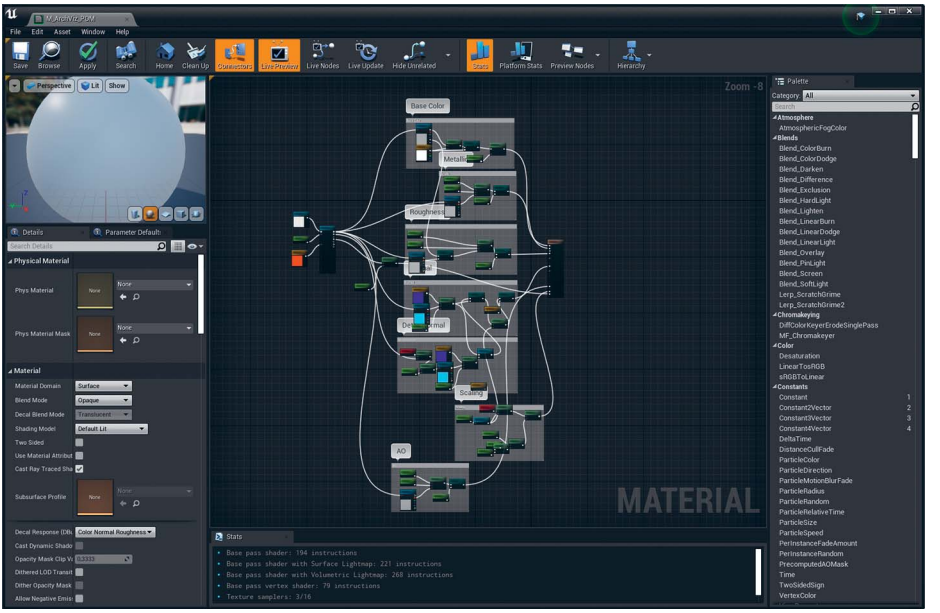


Рисунок 15.15 Обзор POM-версии для Master Material

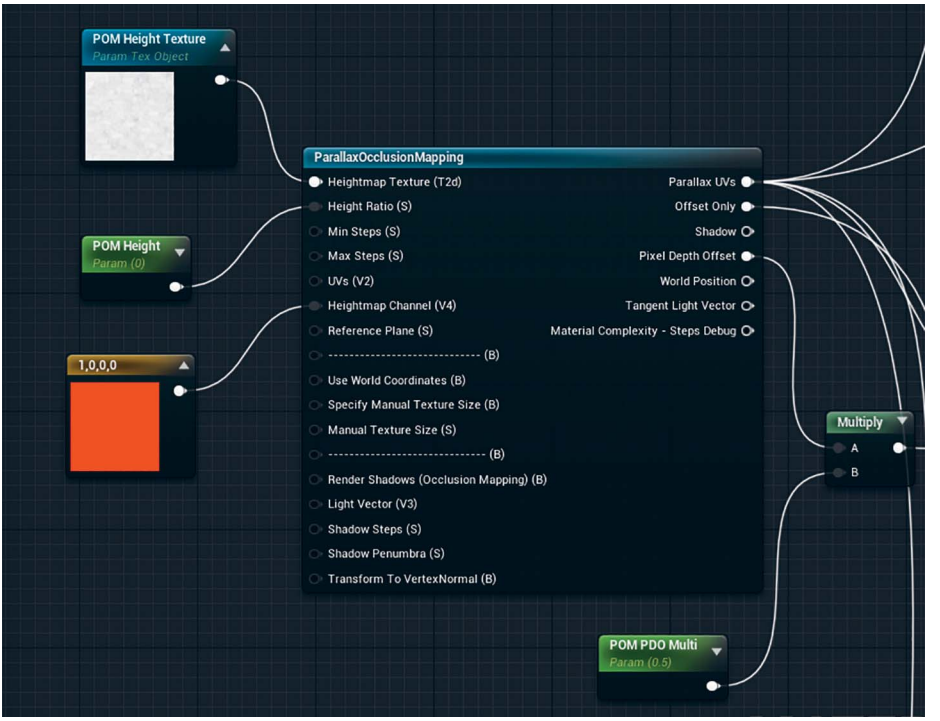


Рисунок 15.16 Детальный обзор функции Parallax Occlusion Mapping; получившиеся искажения UV затем подключаются к каждому входу Texture Sample Parameter's UV

Для настройки функции подключите **Texture Object** Parameter к входу Heightmap Texture. Texture Object Parameter отличается от других Texture Parameters, которые вы применяли где-то в другом месте. Texture Parameters, который вы используете, возвращают цвет, основанный на назначенной Texture; Texture Object Parameter выдает ссылку на Texture, и ее используют для взаимодействия с функциями материала, а не напрямую для рендеринга.

Для его создания вы должны явно выполнить поиск **Texture Object** Parameter в Palette или открыть правой кнопкой мыши меню в Material Editor.

Подключите **POM Height** Scalar Parameter, соединенный с входом **Height Ratio**, и настройте силу воздействия. Оно должно быть очень низким. Если установить 0,01, то это приведет к очень сильному эффекту. Хорошие значения обычно находятся в диапазоне от 0,001 до 0,005.

Подключите узел **Constant Vector 3** к входу **Heightmap Channel**. Это определит, какой цветовой канал будет выбран из узла POM Height Texture. В данном случае установите его на красный канал, установив для Constant 3 значение 1,0,0.

Ковры

Использование POM с высокочастотной Texture, такой как карта высот ковра, прекрасно работает, давая глубину и насыщенность даже простым материалам (рисунок 15.17).

Рисунок 15.17 Material ковра использует POM для смещения волокон, что помогает придать ковру глубину и мягкость, особенно в движении и в VR

Кирпичи

Кирпичи являются классическим примером для демонстрации возможностей рендеринга движка— по веской причине. Их обманчиво сложно рендерить. POM дает хорошие результаты и обеспечивает качество, которое устраивает большинство художников визуализации (рисунок 15.18).

Есть много онлайн-источников для отличных кирпичей и других Texture ресурсов. Главное — иметь точную карту высот. Получение определения между кирпичом и осадком имеет огромное значение. Посетите www.TomShannon3D.com/UnrealForViz, где можно найти несколько ссылок на мои любимые места с высококачественными Textures.

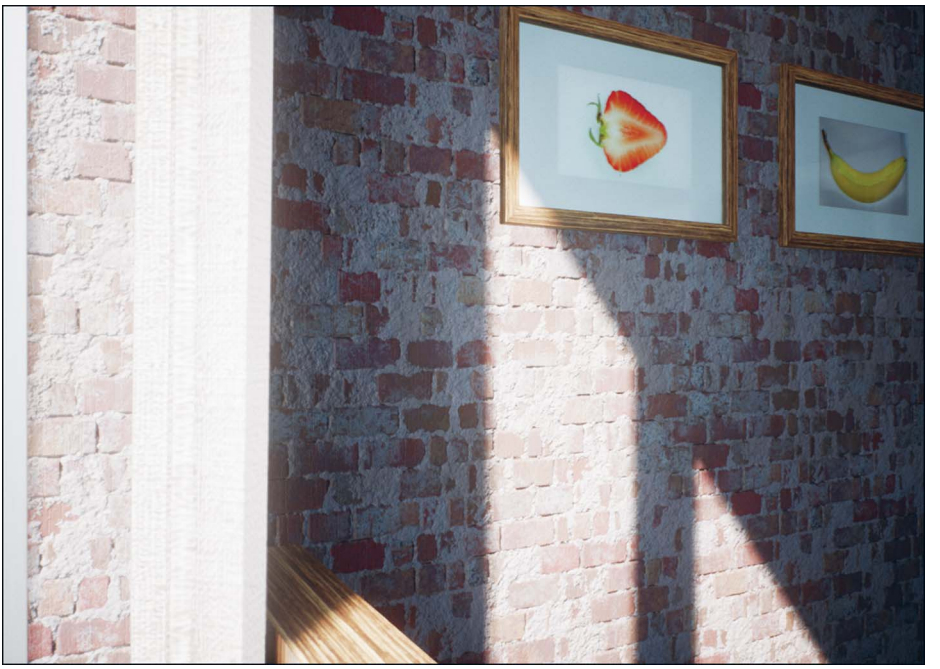


Рисунок 15.18 Brick Wall с использованием POM

Стекло

Игровые движки долго боролись за создание убедительных полупрозрачных эффектов стекла. Эффективный рендеринг стекла основан на преломлении и отражении, и оба являются дорогими и затратными по времени эффектами для рендеринга (рисунок 15.19). С осторожностью вы можете достичь потрясающего стеклянного Material в UE4.

Рисунок 15.19 Один стеклянный Material, используемый для стаканов, мисок и окон из листового стекла на заднем плане

Существует несколько способов создания Glass Materials, каждый со своими преимуществами и недостатками. Обычно чем точнее материал, тем больше времени требуется для рендеринга. Материал, показанный на рисунке 15.19, является довольно дорогим для рендеринга, но это того стоит, так как повышается качество простых полупрозрачных материалов.

Так как нам понадобится установить Material на Translucent, мы не можем просто использовать Master Material в качестве основы для Glass Materials. Нам нужен новый Material.

Первое, что вам нужно сделать, — Material, который вы создаете, Translucent. Это позволит рендерить его после основной сцены и смешивать поверх после. В панели Material's Detail без выбранных узлов установите Material's **Blend Mode** на **Translucent** и **Lighting Mode** на **Surface Translucency Volume** (рисунок 15.20).

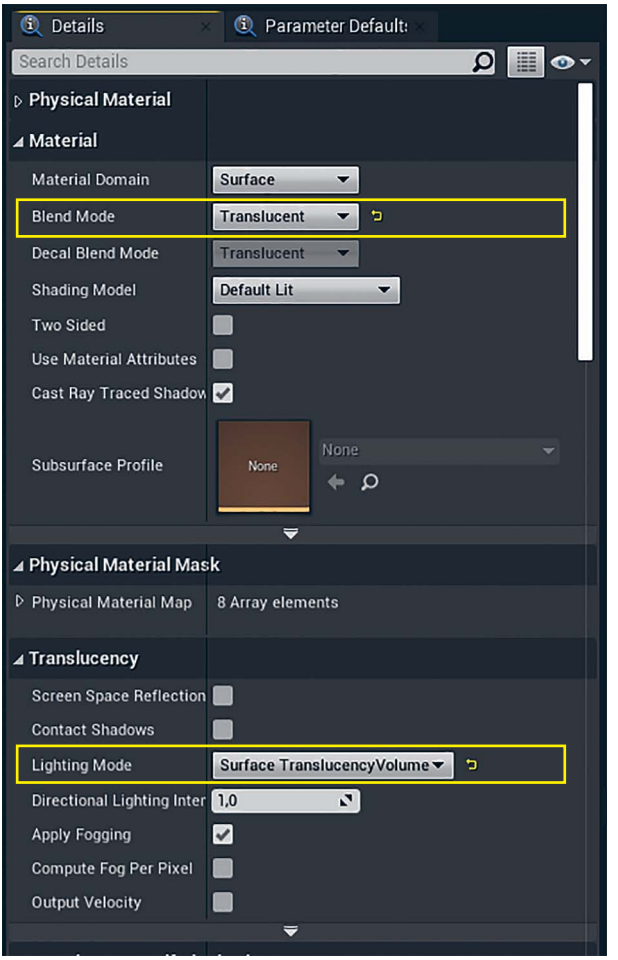


Рисунок 15.20 Detail Panel стекла с Blend Mode, установленным на translucent, и Lighting Mode, установленным на Surface Translucency Volume

Большинство прозрачных материалов полагаются на атрибут Opacity, чтобы модулировать непрозрачность объекта, позволяя отрендерить сцену позади него. В данном случае вы обойдете это (установите Opacity почти на 1,0) и соедините со своей версией сцены за стеклами, раскрасив и исказив.

Для этого нужно выбрать **Scene Texture** (рисунок 15.21).

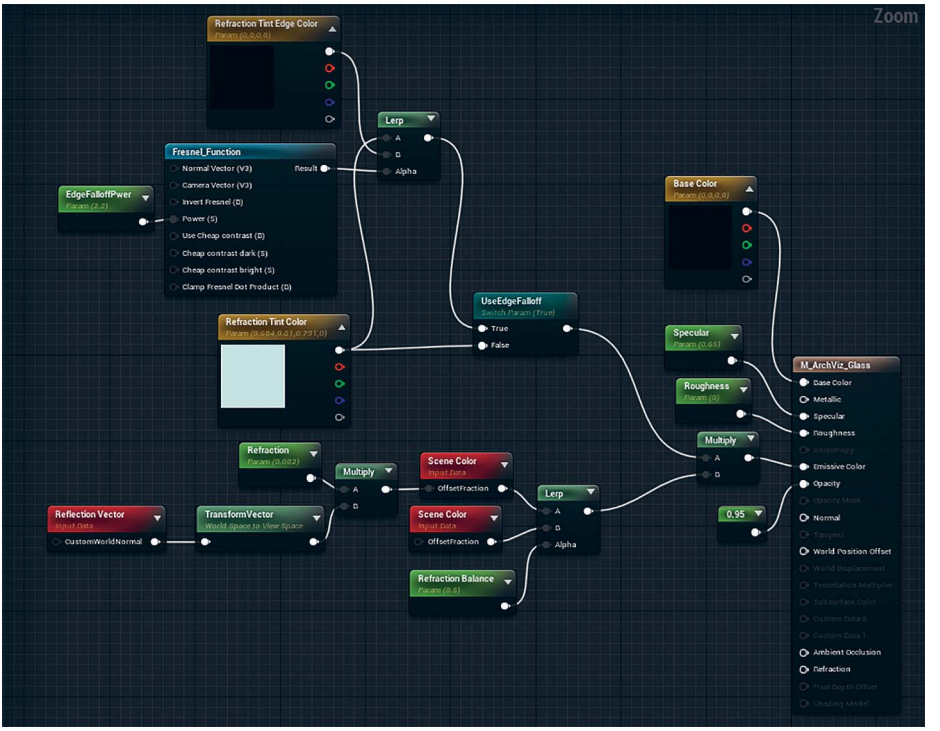


Рисунок 15.21 Сеть ArchViz Glass Material, минуя канал Opacity и создавая вашу собственную измененную Scene Texture, позволяет вам более эффективно контролировать преломление, а отражениям поверхности оставаться яркими

Scene Texture — это отрендеренная сцена до применения полупрозрачности (полупрозрачные объекты в UE4 рендерятся после остальной части сцены и затем компонуются в финальный кадр). Если бы вы передавали Scene Color напрямую атрибуту Emissive для материала, это заставило бы стекло выглядеть полностью прозрачным, не считая отражения от поверхности.

Для искажения Scene Texture возьмите Reflection Vector (входной Vector предоставленный движком), преобразуйте его к вашему виду (переводя его из 3D мирового пространства в значение 2D пространства обзора) и затем используйте эти данные для искажения Scene Color с помощью Offset Fraction. Offset fraction — это процент (0,0–1,0), который позволяет вам выбрать Scene Color в другом месте пикселя на экране.

Это абсолютно физически не точная техника преломления, но она обеспечивает хорошо выглядящие результаты.

Вы заметите два узла Scene Color на рисунке 15.21, которые связаны с помощью **Reflection Balance**. Один из них — искаженный Scene Color, а другой без изменений.

Это дает двойной вид панели, который отлично подходит для архитектурной визуализации листового стекла.

Затем вы можете легко подкрасить искаженные и удвоенные образцы Scene Color.

В предположении, что вы установили **Use Edge Falloff** на true в экземпляре материала, основанном на некотором материале, **Freshnel** обеспечивает простой спад по Френелю, основанный на нормали поверхности относительно камеры. Используя этот спад, Tint интерполируется между определенным **Refraction Tint Color** Parameter к **Refraction Tint Edge Color** Parameter. В противном случае Scene Color окрашивается **Refraction Tint Color** Parameter.

Установите атрибуты материала **Roughness** и **Specular** с помощью простого Scalar Parameters. Вы можете расширить этот материал для использования текстур, чтобы управлять любым из этих параметров; однако это сделает материал более дорогим для рендеринга и должно использоваться с осторожностью.

Заключение

Благодаря системе PBR в UE4, визуальному редактору материалов и мощности, а также удобству экземпляров материалов, создание красивых материалов происходит быстро и легко. Это может быть так весело, что возврат к созданию материалов в 3D-приложениях может быть трудным.

Использование параметров материалов и экземпляров материалов делает повторно используемыми ваши материалы по щелчку, а интерактивный основанный на узлах редактор материалов позволяет вам экспериментировать, изучать и расширять используемые материалы по мере роста ваших способностей и требований к проекту.

ГЛАВА 16

СОЗДАНИЕ КИНЕМАТИКИ С SEQUENCER

Интерактивность и исследование являются отличительными чертами интерактивной визуализации в UE4. Однако UE4 также хорошо приспособлен для создания предварительно обработанных статических анимаций, которые соперничают с качеством рендеринга, полученного с использованием трассировки лучей и могут сделать это быстро. С помощью Sequencer, инновационного инструмента анимации Unreal Engine 4, вы можете комбинировать интерактивные миры с ключевыми кадрами анимации камер и акторов для создания превосходных анимаций.

Начало работы с Sequencer

Sequencer Editor является кинематографическим инструментом для редактирования в UE4, позволяющим вам редактировать **Sequences**.

Sequences являются ассетами, которые вы можете размещать на уровнях, как актора, содержащего ключевые кадры анимационных **Tracks**. Сиквенсер черпает вдохновение из таких инструментов, как After Effects, Final Cut и других приложений для редактирования и создания видео. Знакомство с этими приложениями облегчает изучение многим художникам визуализации.

Сиквенсор заменяет предыдущий инструмент кинематографического редактирования в UE4, **Matinee**. Matinee продолжает сосуществовать вместе с Sequencer, но он устарел и в сравнении с Sequencer очень ограничен.

Master Sequence

В отличие от большинства ассетов, которые вы либо импортируете, либо создаете в Content Browser, Sequences вы создаете на уровне с помощью раскрывающегося меню Cinematics (рисунок 16.1).

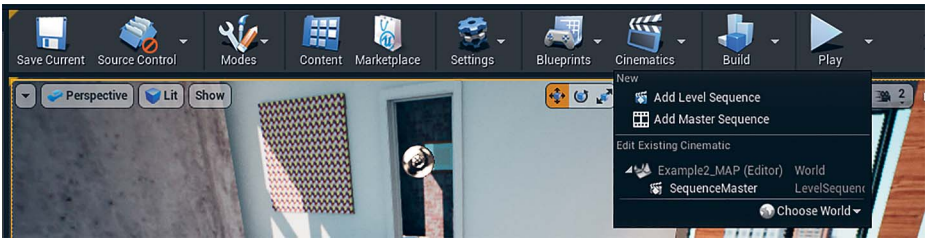


Рисунок 16.1 Создание Master Sequence из выпадающего меню Cinematics в Editor

Вам предоставляется возможность создать **Master Sequence**, одиночный **Level Sequence** или устаревший сиквенсор Matinee. Master Sequence — это в основном мастер, который создает Sequence с несколькими sub-sequences (рисунок 16.2). Мастер также создает различные Sequence Assets и сохраняет их в специальном месте в Content directory.

Вы также можете начать с одним Level Sequence, если нужно создать что-то простое. Вы можете включить отдельные Sequences в другие Sequences в любое время.

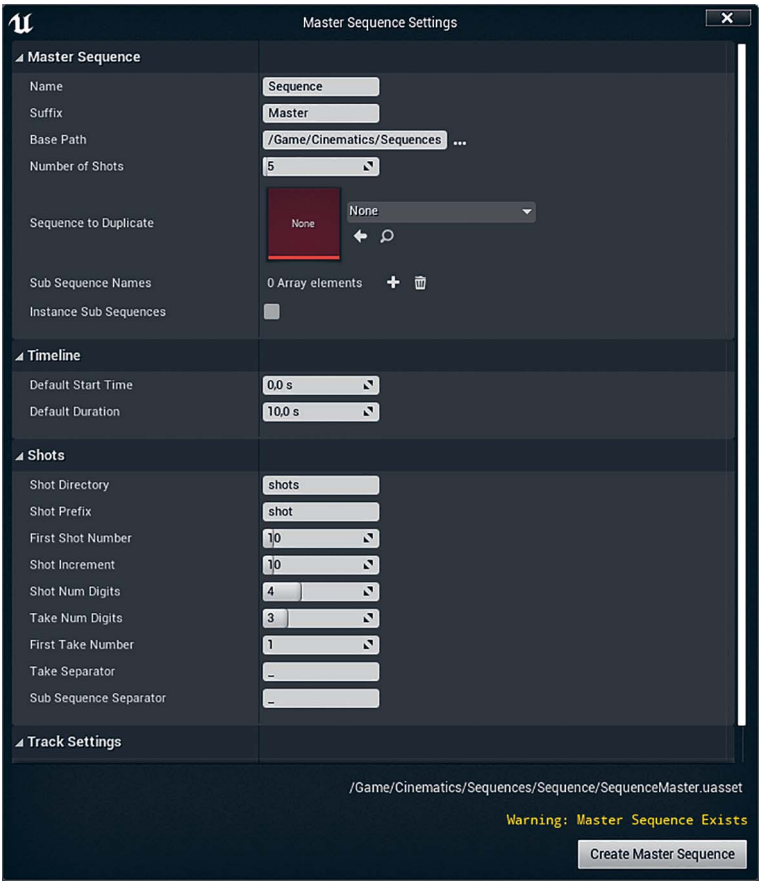


Рисунок 16.2 Настройки мастера Master Sequence, показывающие, что я изменил только Default Duration и установил его на 10 секунд от заданных по умолчанию 5

Вы можете принять стандартные настройки для большинства проектов, если не нужно следовать схеме именования или другим стандартам. Единственной настройкой, которую я изменил здесь, является Default Duration, которую я установил на 10 секунд.

Когда вы создаете Master Sequence, появится окно Sequencer показывающее все дорожки Sequences, расположенные в ряд (рисунок 16.3).

Любой, кто знаком с приложениями нелинейного редактирования, почувствует себя как дома. Вы можете перемещать кадры (**Shots**) и добавлять, удалять и просто менять продолжительность каждого трека, что делает Shots Track Editor похожим на Adobe Premiere: быстрый, оптимизированный, но с нехваткой точного контроля.

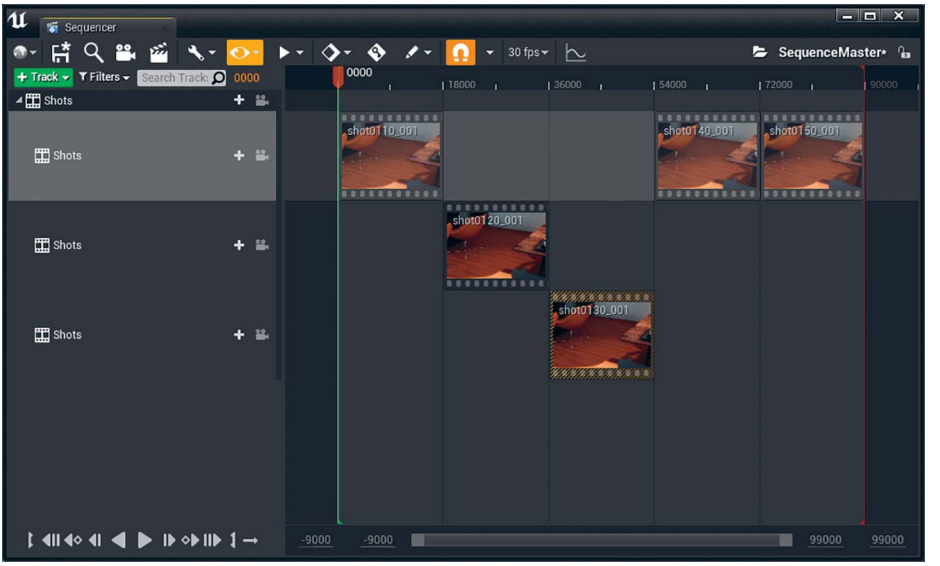


Рисунок 16.3 Master Sequence Editor, в котором вы можете перемещать и редактировать Shots как в программе для нелинейного редактирования видео

Чтобы получить доступ к управлению, дважды нажмите на любой Sequences, чтобы открыть кадр в Sequence Editor (рисунок 16.4). Хотя это тот же Sequencer Editor, но он выглядит по-другому и служит другой цели. Предоставляя Camera Cuts и Camera Actor Tracks, этот редактор больше похож на Adobe After Effects: управляемый ключевыми кадрами интерфейс анимации и эффектов.

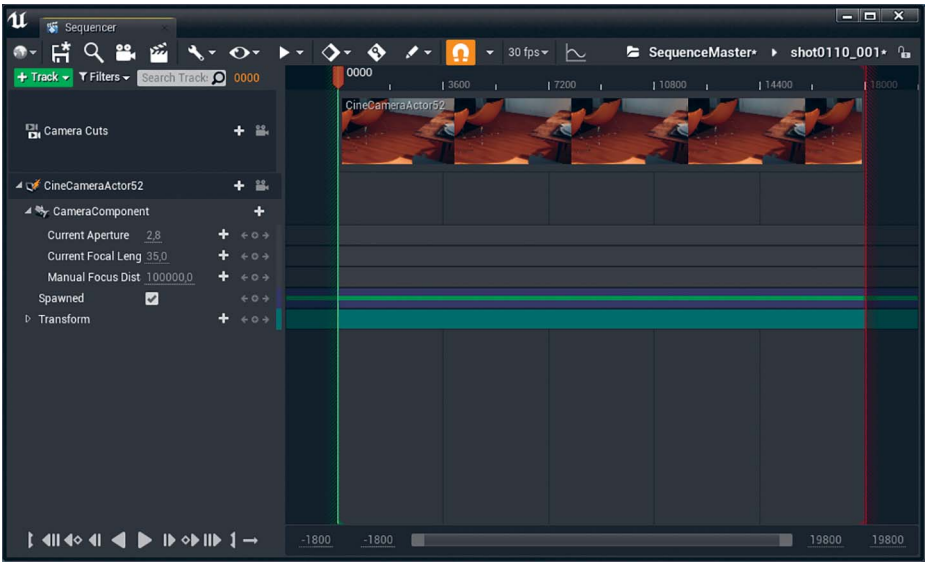


Рисунок 16.4 Единственный кадр, открытый для редактирования в Sequence Editor, показывающий, как меняется редактор, чтобы выглядеть более похожим на пакет эффектов ключевых кадров, например After Effects

Динамические камеры

В каждом из этих кадров Sequences, **CineCameraActor** появляется динамически в Sequence и уничтожается, когда Sequence завершается. Эта особенность по-настоящему мощная и означает, что не нужно размещать одиночного Camera Actor на вашем уровне, так как они создаются и уничтожаются на лету в сиквенсоре.

Эти камеры действуют как любой другой Camera Actor и отображают настройки объектива, глубину резкости и настройки фокуса, как и все доступные эффекты постпроцессинга, позволяя настроить каждую камеру, как вам нужно (рисунок 16.5).

Обратите внимание на Details Panel и кнопку Add Keyframe, которая появляется слева для каждого из связанного с анимацией свойства. Как и большинство окон в UE4, вы можете закрепить сиквенсор в окне UE4 для наведения порядка.

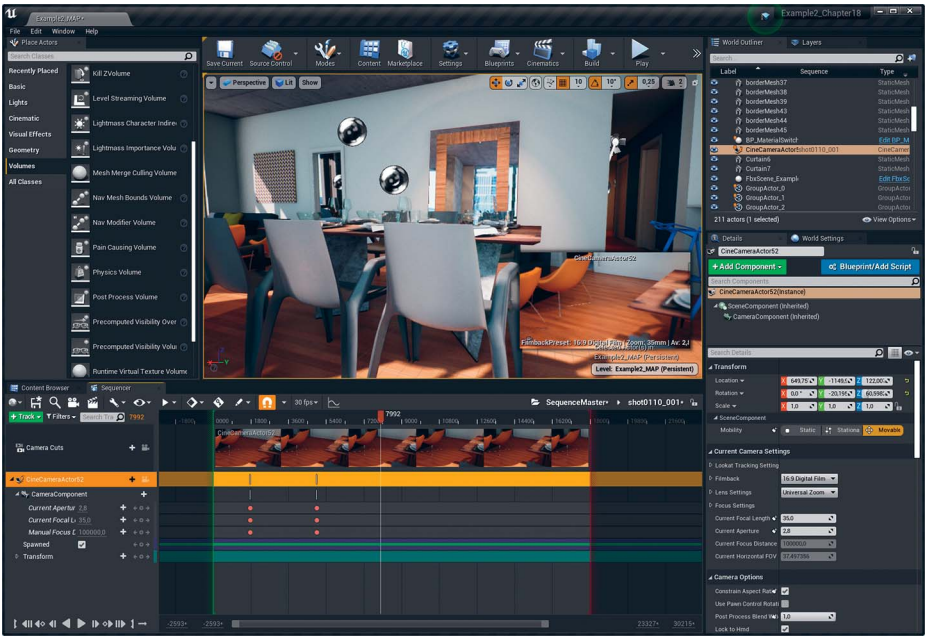


Рисунок 16.5 Закрепленный Sequencer в UE4 Editor

Анимация камеры

Заставить камеру передвигаться и меняться со временем — следующий очевидный шаг, но сделать это может быть не так легко.

Настройка ключевых кадров

У вас есть несколько способов установить ключевые кадры в сиквенсоре. Вы можете включить Auto-Key, создавая ключевой кадр каждый раз, когда меняете свойство или перемещаете, поворачиваете или масштабируете актора. Это может быстро привести к беспорядку.

Вы также можете вручную установить ключевые кадры. Вы можете сделать это либо в интерфейсе сиквенсора, либо напрямую в свойствах и Viewports. Когда сиквенсор активирован, связанные с анимацией свойства для объектов Actors показывают кнопку Add Keyframe (рисунок 16.5), позволяя вам легко добавлять ключевые кадры, когда это нужно.

Третий способ является смесью из предыдущих вариантов. Вы можете автоматически устанавливать ключевые кадры, но ограничивать их свойствами, которые вы уже добавили к ключевым кадрам (рисунок 16.6). Я обычно использую этот вариант, так как он сочетает простоту в использовании auto-keyframing, не беспокоясь о случайном добавлении ключевых кадров на дорожки, которые не нужно анимировать.

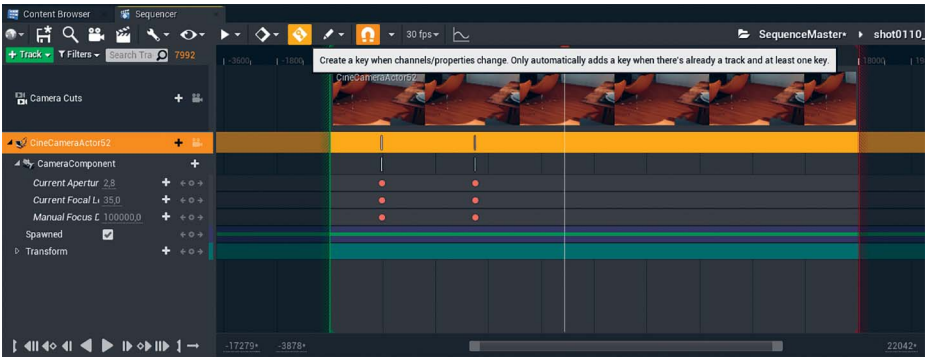


Рисунок 16.6 Настройка параметров Auto-Keyframe для добавления только ключевых кадров к уже анимированным дорожкам и свойствам

Установка камеры

Теперь, когда вы можете установить несколько ключевых кадров, давайте поставим камеру на место.

Для просмотра через камеру выберите справа значок камеры, которую хотите анимировать. Это установит Viewport для управление камерой, позволит летать Actor по сцене, прикрепив его к вашему обзору.

Значки камеры появляются во многих областях интерфейса сиквенсора, и это может немного сбивать с толку (рисунок 16.6). Вы заметите, что дорожка Camera Cuts также содержит значок камеры. Если вы нажмете на нее, вам откроется вид с перспективы Sequence, включающий любые camera cuts, которые вы создали.

Как правило, вы будете переключаться между этими видами, пока работаете, настраивая ключевые кадры камеры и просматривая контекстный результат с другими кадрами.

При управлении камерой вы смотрите через объектив камеры и перемещаете ее по сцене с помощью стандартного управления Perspective Viewport. Вы можете настроить ключевые кадры камеры в Orthographic-видах или свойства Camera Actor в панели Details.

Дорожки и название камеры

Вы можете обратить внимание на название дорожки в сиквенсоре, отличное от метки в предпросмотре ленты времени. Это немного сбивает с толку, так как они именуются независимо (вы можете, к примеру, использовать один и тот же Camera Actor для различных дорожек в разных Sequences). Название в предпросмотре является названием актора в мире. Вы можете переименовать ваш Cine Camera Actor с помощью панели Details для обновления названия в ленте времени.

Название в левой панели является названием дорожки. Вы можете дважды нажать на это имя, чтобы переименовать дорожку.

Переходы

Единственный **переход**, который предлагает UE4, — дорожка **Fade**. Недоступны cross-dissolve или другие эффекты перехода. Это в основном из-за соображений о производительности. Выполнение cross-dissolve заставляет дважды отрисовывать сцену и совмещает их во время растворения. Хотя некоторые сцены могут справиться с этим, но большинству будет тяжело отрендерить так много информации.

Редактирование кадров

Для возврата к Master Sequence, выберите заголовок в правом верхнем углу окна сиквенсора (рисунок 16.7).

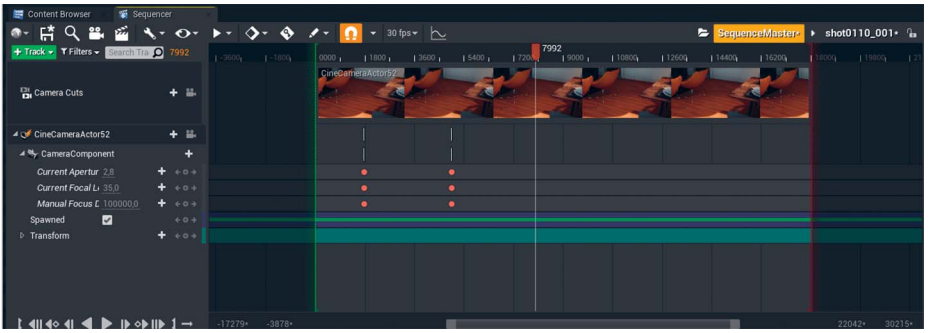


Рисунок 16.7 Завершенный кадр с SequenceMaster-треком, выделенным в правом верхнем углу окна секвенсора

Вы увидите, что предварительный просмотр снимка был обновлен. Вы можете использовать это как руководство для настройки входных и выходных точек в Sequence. Чтобы лучше увидеть, увеличьте масштаб timeline с помощью ползунка диапазона внизу окна сиквенсора.

Продолжайте редактирование оставшихся снимков. Отредактируйте снимок, затем вернитесь к основной последовательности, чтобы увидеть ее в контексте, измените ее и повторяйте до тех пор, пока не будете довольны всеми кадрами в последовательности.

Сохранение

Не забывайте периодически сохраняться. Sequences сохраняются как UASSET-файлы в Content directory вашего проекта.

Вам не нужно сохранять уровень, на котором находится Sequence после того, как вы впервые разместили Sequence Actor. Этот актор ведет себя как ссылка на ваш Master Sequence, позволяя инициализировать данные Sequence, когда уровень загружается, и тем самым дает Blueprints получить доступ к нему, просто ссылаясь на объект Actor.

Совместная работа

Так как кадры хранятся как отдельные пакеты UASSET, несколько членов команды могут одновременно работать с одним Sequence. Также установка света, реквизита и другое декорирование может продолжаться, пока анимация разрабатывается параллельно. Это экономит время и позволяет быстрее создавать наброски в производственном конвейере.

Рендеринг в видео

После усовершенствования Sequence время вывести его на диск. Именно здесь скорость UE4 поразительна. Целые анимации могут быть доставлены (не только отрисованы, но и на YouTube) за время, необходимое одному кадру для отрисовки в стандартном рендеринге с трассировкой лучей, в таком как Mental Ray или V-Ray.

Открыв ваш Sequence, нажмите на кнопку Render to Video (рисунок 16.8) в панели инструментов Sequencer. Откроется диалоговое окно Render Movie Settings (рисунок 16.9). Здесь вы можете установить широкий спектр параметров рендеринга и экспорта. От рамок бампера до предварительной прокрутки сжигаемых модулей UE4 предлагает инструменты профессионального уровня для видеоредакторов и композиторов.

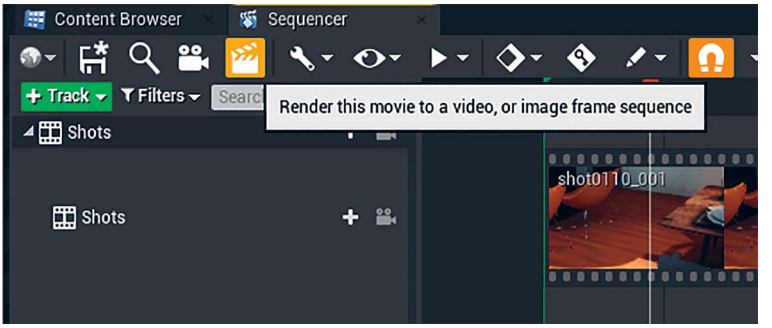


Рисунок 16.8 Выбор кнопки Render to Movie в Sequencer

Render Movie Settings

Как вы можете заметить, я установил его на рендеринг в разрешении 4K с 60 кадрами в секунду (рисунок 16.9). Это создает шелковисто-гладкие, резкие анимации, которые действительно привлекают внимание. Эти файлы весят более 100 Гб за минуту без сжатия.

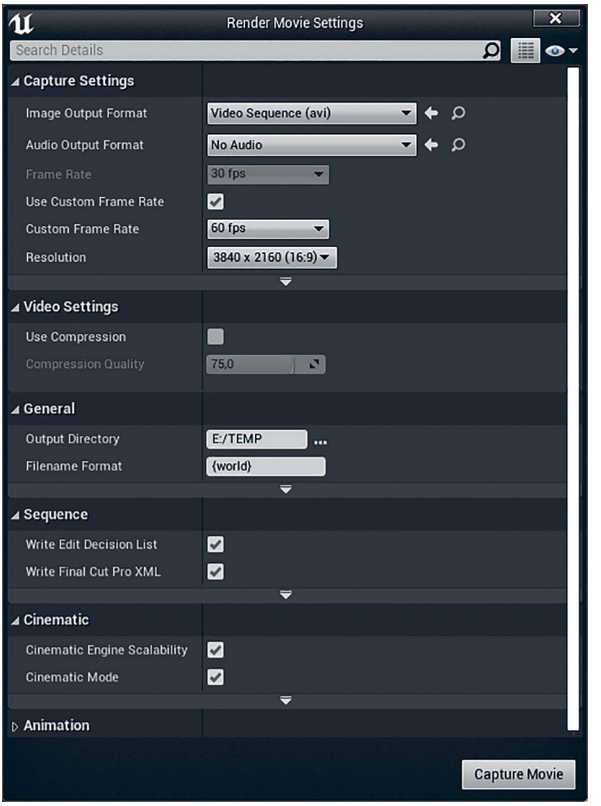


Рисунок 16.9 Диалоговое окно Render Movie Settings, установленное на разрешение 4K (3840×2160) и 60 кадров в секунду

Я рендерю напрямую в AVI (Video Sequence), но, если в вашей студии есть более обширный процесс постпродакшена, основанный на линейном цвете, вы также можете рендерить буферы HDR-изображений отдельно, используя опцию Custom outputs.

Вы также можете выбрать сжатие ваших AVI для сохранения места (не рекомендуется) или экспортировать на отдельные кадры в распространенных форматах, таких как BMP и EXR.

Процесс рендеринга

Когда UE4 начинает рендеринг, появляется маленькое окно рендеринга и начинается запись на диск (рисунок 16.10). Окно рендеринга расположено в верхнем левом углу окна, а редактор находится в фоновом режиме. Обратите внимание на уведомление Capturing video в нижнем правом углу. Также обратите внимание на высокое использование диска для записи — чем быстрее, тем лучше.

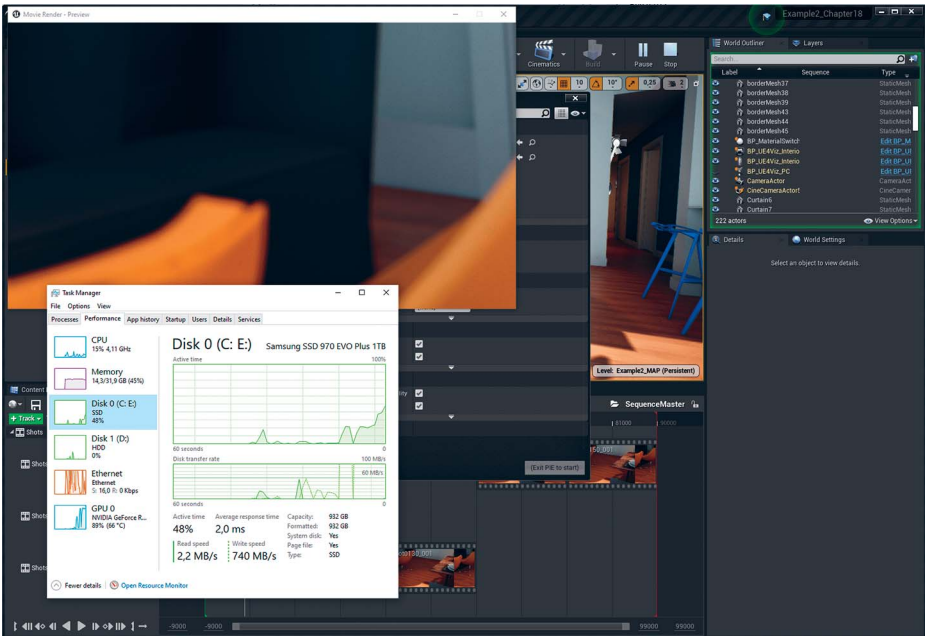


Рисунок 16.10 UE4 рендерит Sequence на диск

На данном этапе рендеринг ограничивается не визуальными эффектами, а скоростью жесткого диска, на который он записывается. Я не могу увеличить это достаточно. Запись кадра на диск длится дольше, чем рендеринг кадра.

Записывайте на диск, который дает самую быструю и стабильную производительность. Обязательно проверьте. Я был удивлен, узнав, что мой SSHD значительно быстрее

в долгосрочной перспективе, чем мой обычный очень быстрый SSD, который я считал лучшим выбором. Диски с более устойчивой скоростью записи принесут огромные выгоды.

После завершения анимации обработайте ее так же, как и любой другой видеофайл. Он может напрямую отправиться в интернет или включаться в более крупное видео, точно так же, как и ваши стандартные видео визуализации.

Заключение

Возможность адаптироваться к нуждам клиента в момент обращения является еще одной причиной, почему так много студий визуализации переходят на UE4 не только для создания интерактивности, но и для предоставления потрясающего предварительно отрендеренного контента.

Эта 90-секундная последовательность требует примерно 10 минут на рендеринг — в разрешении 4K с 60 кадрами в секунду. Больше времени уходит на сжатие, копирование и загрузку видеофайлов, созданных с помощью UE4, нежели на их рендеринг.

Благодаря мощным возможностям Sequencer, способности точно видеть, что будет в кадре вашей камеры все время, и физически корректировать модель Cine Camera, вы можете начать создавать видеоконтент в UE4, который соответствует, а иногда и превосходит качество традиционных рендеров.

ГЛАВА 17

ПОДГОТОВКА УРОВНЯ К ИНТЕРАКТИВНОСТИ

Основываясь на простом Pawn, Game Mode и Player Controller из первого примера проекта, мы легко настроим возможность исследования уровня. Однако вы должны настроить ваш уровень с учетом коллизий, запуска игрока и других настроек, прежде чем игрок начнет успешно перемещаться по нему.

Настройка Level

Для настройки интерактивности уровня вам нужно настроить использование курсора мыши и сенсорного ввода для Level и Player Controller. Вам также станут доступны события взаимодействия с мышью, которые позволят вам выделять и нажимать на акторов в игровом мире в процессе работы.

Конечно, вы захотите открыть уровень, прежде чем продолжить. Вы, возможно, как и я, захотите сохранить новую версию уровня, которая позволит легко вернуться к его неизмененной версии.

Добавление Player Start Actor

Как описывалось в главе 9, каждый UE4 Level требует Player Start Actor. Это простой Actor, который сообщает UE4, где расположить игрока на уровне при начале игры.

Перетащите Player Start Actor из панели Place Actors в Level рядом с тем местом, где вы хотите, чтобы игрок появлялся в мире каждый раз (рисунок 17.1).

Также вы должны повернуть Player Start Actor лицом по направлению, в котором хотите чтобы игрок начинал. Синяя стрелка в середине Player Start Actor указывает на Player Start Actor's forward vector.

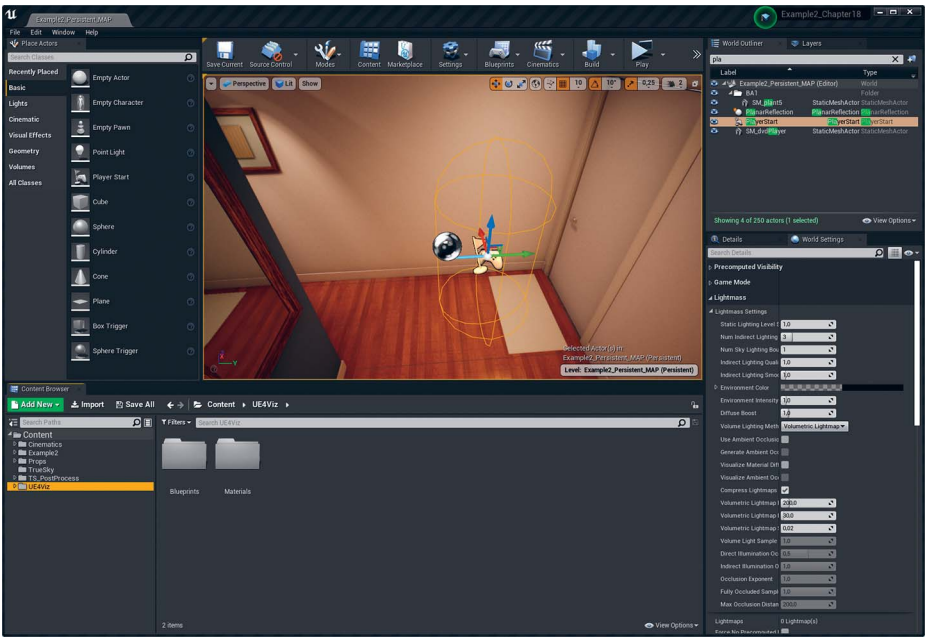


Рисунок 17.1 Расположение Player Start на Level

Добавление коллизий

Теперь вы можете протестировать ваше приложение, нажав кнопку Play in Editor. Однако высока вероятность, что вы обнаружите себя упавшим ниже вашего пола на уровне, когда сделаете это. Это может быть связано с отсутствием информации о коллизии.

Обнаружение коллизий является серьезной задачей в интерактивных играх и симуляциях. Знание, когда один актор «пересекся» или столкнулся с другим актором или зашел в некоторую область, является важным для создания живых миров.

В UE4 вы можете использовать **Player Collision** View Mode, как показано на рисунке 17.2, для предпросмотра столкнувшихся акторов. В Viewport выберите кнопку ViewMode и из выпавшего списка — Player Collison. Чтобы вернуться к вашему обычному виду, установите обратно Lit для Viewmode.

Как видите (или, точнее, не видите), на рисунке нет этажей или стен, но у некоторого реквизита и дверных мешей уже есть коллизии. Они представлены с помощью плоских затененных версий ваших примитивов коллизий.

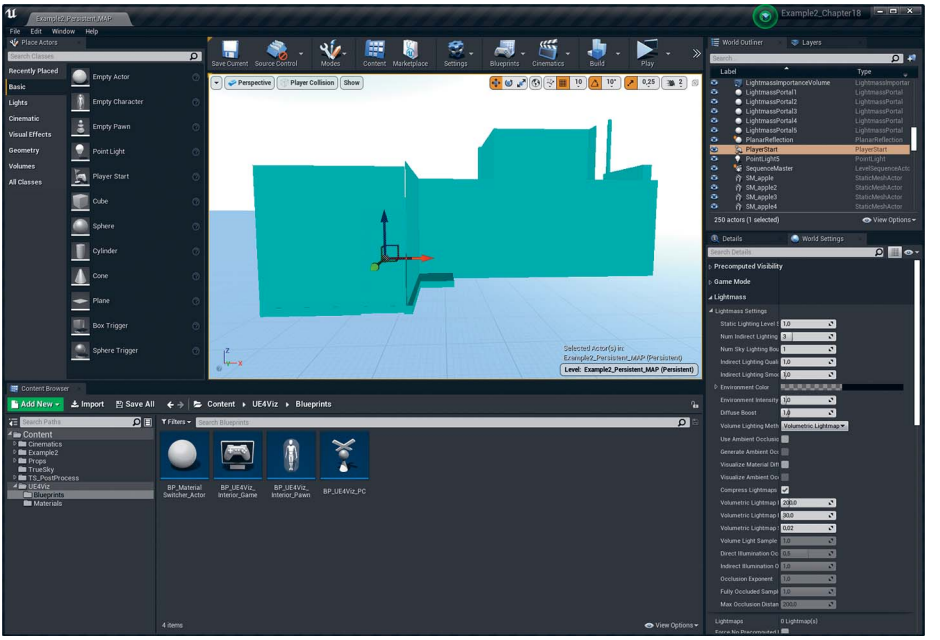


Рисунок 17.2 Collision, отображаемая с помощью Player Collision View Mode

Collision в UE4 может быть сложной для понимания на первый взгляд. К счастью, визуализации имеют тенденцию становиться более простыми, статичными, с небольшим количеством динамичных Actors (в сравнении с игрой с десятками или сотнями героев

на экране и в игровом мире одновременно), чтобы помочь вам избежать некоторых сложностей, с которыми вы можете столкнуться при настройке коллизий.

Сложные коллизии против простых

Игры, как правило, полагаются на упрощенную версию моделей для выполнения расчетов столкновений. Это связано с тем, что эти вычисления являются и становятся все более дорогостоящими, чем больше полигонов и информации требуется обработать.

UE4 использует **collision primitives** — простые фигуры, такие как кубики, сферы, и капсулы, — которые можно создать в Engine или с помощью низкополигональных фигур, сделанных в 3D-приложении, для использования в качестве **simple collision**. Это позволяет графически применить на несколько порядков больше полигонов, так как физический движок использует оптимизированную версию сцены для своих вычислений.

UE4 также может выполнять **per-polygon collision** по мере необходимости. Это часто косметически используют в играх для добавления определенных графических эффектов, таких как повреждение стен или транспортных средств, которые происходят точно в месте видимого удара, в то время как фактическая физическая коллизия использует упрощенную модель для скорости.

Так как сцены визуализации могут быть относительно простыми с несколькими взаимодействующими элементами, вы часто можете использовать per-poly или **complex collision** (как это известно в UE4) вместо простого collision, позволяющего вам пропустить время и усилия по созданию собственной геометрии коллизий.

Стены и пол

Главное, что вы наверняка хотите, — это чтобы ваши игроки не проходили через стены и полы в симуляции. Так как эти меши геометрически просты и имеют неудобную форму, мы можем безопасно установить их для использования обычного per-polygon collision.

Для этого откройте Static Mesh Asset в Static Mesh Editor. Во вкладке Details раскройте Static Mesh Settings, установите Collision Complexity на **Use Complex Collision as Simple** и убедитесь, что Collision Preset установлен на **BlockAll** (рисунок 17.3).

Вы можете переопределить Collision Preset для каждого основного Actor с помощью панели Details; однако нельзя изменить свойство Collision Complexity. Можно сделать это только в интерфейсе Static Mesh редактора или в матрице свойств.

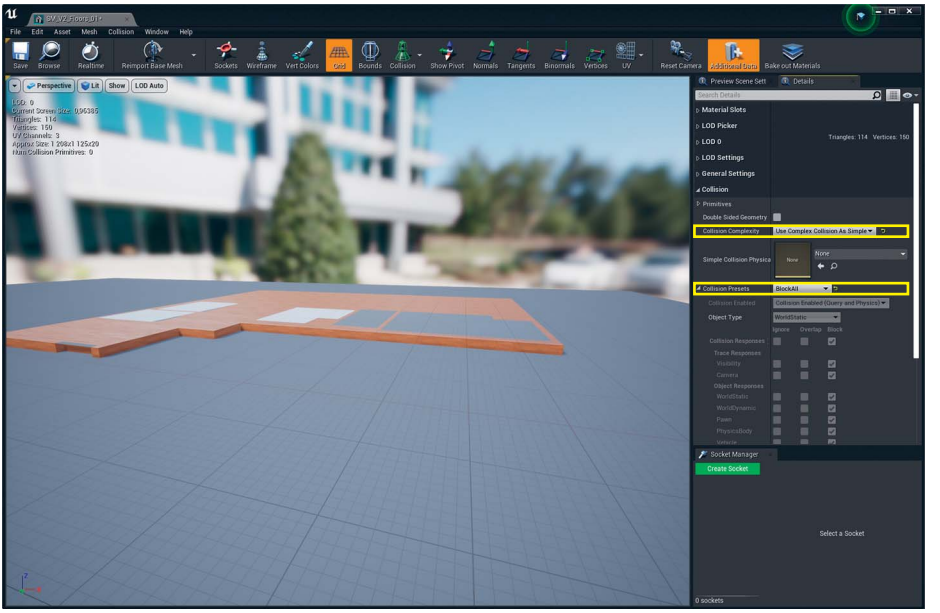


Рисунок 17.3 Настройка Floor Mesh для использования per-polygon collision

Bulk Editing с Property Matrix

Вы должны настроить collision complexity для каждого объекта Mesh, с которым хотите столкнуть вашего игрока. Это займет очень много времени, если вам придется открывать все ассеты один за другим.

К счастью, UE4 содержит функцию Bulk Edit.

Выберите все меши, которые хотите включить для коллизий в Content Browser. Нажмите правой кнопкой мыши на один из значков Asset для открытия контекстного меню. В разделе **Asset actions** выберите **Bulk Edit via Property Matrix** (рисунок 17.4).

Матрица свойств показывает все общие свойства в измененной Details Panel. Отсюда вы можете установить Collision Complexity для всех ваших Meshes за раз (рисунок 17.5). Вы также можете отобразить свойства в виде столбцов в списке слева и таким образом визуально сравнить Assets и изменить отдельные Assets, как в электронной таблице.

Иногда свойства содержат разные названия в разных интерфейсах. Collision Preset является одним из таких. Для изменения этого свойства с помощью матрицы свойств посмотрите на свойство под названием Collision Profile Name. Фильтрация свойств для **collision** помогает сузить список. Для настройки Collision Preset введите **BlockAll** в текстовое поле свойства Collision Profile Name.

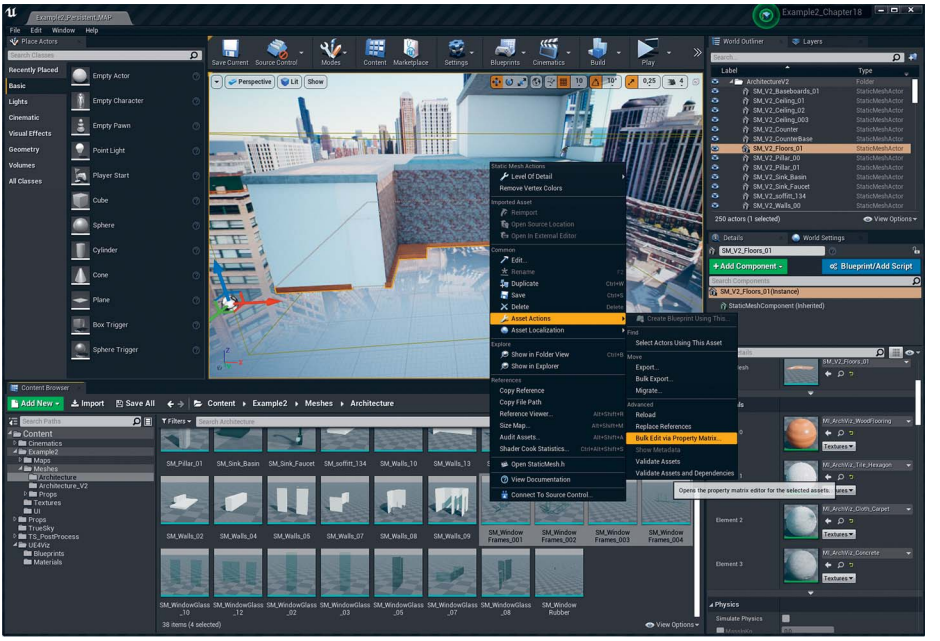


Рисунок 17.4 Выбор нескольких Assets в Content Browser и их одновременное редактирование с помощью матрицы свойств

Когда вы закончите редактирование ваших Assets, обязательно сохраните их, чтобы изменения записались на диск.

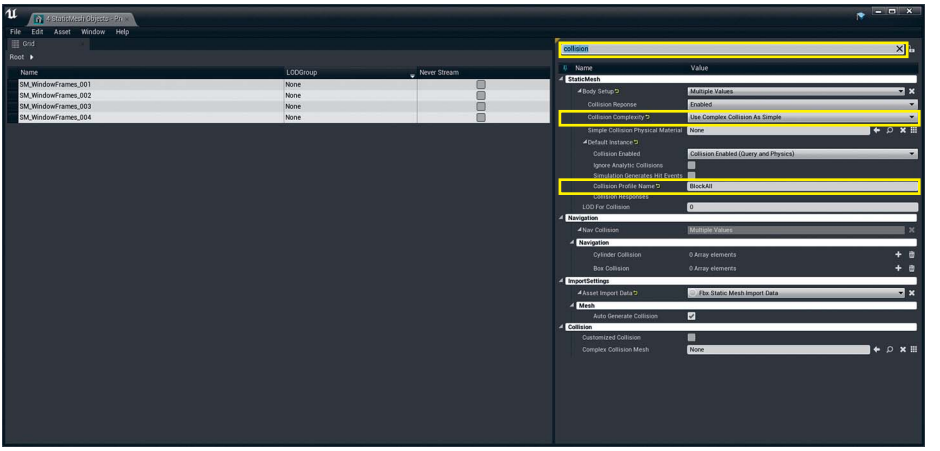


Рисунок 17.5 Использование матрицы свойств для настройки Collision Complexity для всех стен и полов одновременно

Визуализация столкновения

После того как вы настроили столкновения, проверьте вашу инфраструктуру коллизий, установив Perspective Viewport для Player Collision View mode (рисунок 17.6). Это позволит вам увидеть инфраструктуру коллизий, которую физический движок UE4 использует для расчетов столкновений.

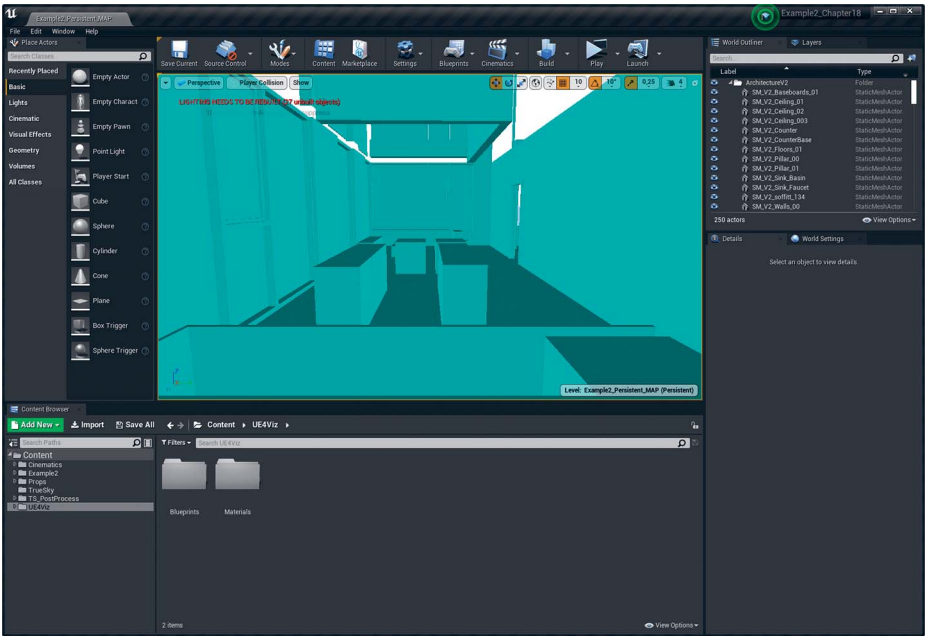


Рисунок 17.6 Player Collision View mode показывает per-polygon collision, успешно включенную для стен и полов.

Настройка Prop Collision

Настройка столкновений для реквизита немного сложнее. Лично я предпочитаю отключать коллизии на большинстве объектов Props в моих симуляциях, позволяя игроку свободно проходить через и над большинством препятствий. Я предпочитаю останавливать своего игрока только объектами по пояс или выше. Это позволяет лучше прочувствовать мобильность в маленьком пространстве и делает перемещение более легким для начинающих игроков.

На рисунке 17.7 вы можете увидеть, что большая часть мелкого реквизита все еще имеет коллизии, что потенциально может затруднить перемещение игрока в этом пространстве.

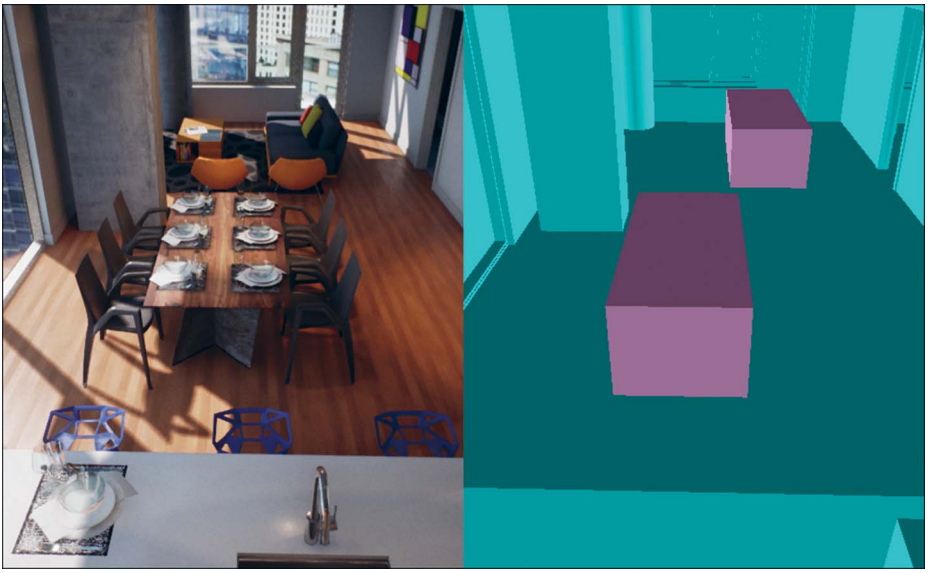


Рисунок 17.7 Окончательная настройка Collision, показывающая только стены, полы, окна и огромную мебель содержащую player collision enabled, позволяющая игроку свободнее перемещаться по пространству.

Чтобы разрешить реквизиту иметь коллизию (важно для взаимодействий, таких, например, как нажатие мышью), измените **Collision Preset** в Static Mesh Editor (или **Collision Profile Name** в матрице свойств) на **Ignore Only Pawn**. Это позволит статическому мешу реагировать на все другие физические события, но не помешает перемещению игрока. Вы также можете установить это свойство каждому актору через панель Details.

Теперь вы можете нажать Play и походить вокруг по уровню без страха навсегда упасть в пустоту. Стены и пол должны быть цельными, и на уровне должно быть легко ориентироваться.

Если вы застреваете или падаете, вам нужно посмотреть на настройку коллизий на уровне.

Включение курсора мыши

Поскольку вы хотите, чтобы это было управляемое мышью приложение, вам нужно видеть курсор. Откройте Player Controller и найдите группу **Mouse Interface** в свойствах по умолчанию класса (рисунок 17.8). Включите **Show Mouse Cursor**, **Enable Click Events** и **Enable Mouse Over Events**, которые позволят 3D Actors на сцене взаимодействовать с курсором мыши.

Если вы сделаете тест сейчас, вы обратите внимание на то, что можете вращать камерой, только когда удерживаете кнопки мыши. Отпускание кнопки отобразит курсор и отключит контроль вращения.

Вы также можете заметить, что ось вращения ощущается «выключенной». Камера повернет направо, когда вы ожидаете, что она повернет влево или вверх, когда вы хотите посмотреть вниз.

Это из-за того, что игроки ожидают управления камерой. Когда игроку не требуется удерживать кнопку для поворота, это привычно для горизонтальной оси, следовать направлению мыши. Правый поворот — направо, левый поворот — налево.

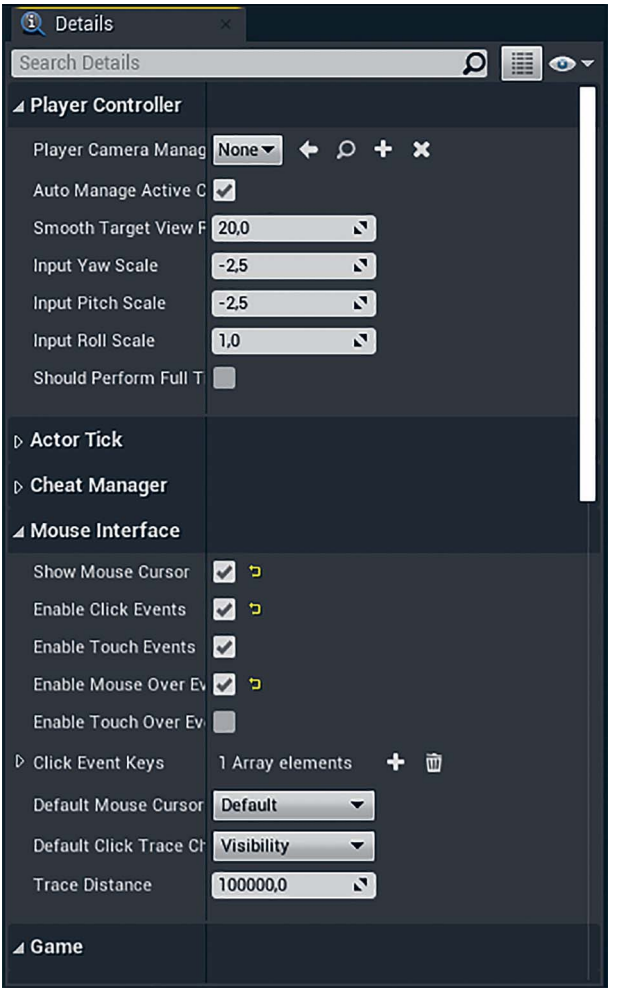


Рисунок 17.8 Настройки Player Controller для использования Mouse Cursor, Mouse Click, и Mouse Over events, а также Input Yaw и Pitch для компенсации ощущения инвертированного поворота

Однако, перетаскивание и отпускание (drag and drop) для поворота камеры ощущается естественно и выглядит как работа на тачскрине с привязкой курсора к точке в 3D-пространстве и поворотом камеры, как если бы пользователь использовал трекбол. Это означает, что перетаскивание мышью вправо должно поворачивать обзор влево, когда перетаскивание вниз должно поднять камеру.

Вы можете регулировать эти изменения в зависимости от нужд игрока через значения Input Yaw и Input Pitch Scale в Player Controller (рисунок 17.8). Причина, по которой Yaw (горизонтальное вращение) быстрее, связана с человеческим восприятием и «правильным ощущением». Наличие симметричных осевых скоростей «ощущается» неправильно.

Стоит отметить, что интерактивность камеры действительно зависит от личных предпочтений. Возраст, опыт работы с различными технологиями и даже любимая игра людей сообщит им, как интерактивная камера должна вести себя. Умение прислушаться к вашим игрокам по этому поводу и адаптация их потребностей в каждом проекте очень важны.

Создание Post-Process Outlines

Чтобы было понятно, какой актор выбран, я хочу использовать очертание, похожее на используемое в редакторе. Этот эффект недоступен во время работы, так как использует совершенно другую систему рендеринга, чем главный Viewport.

Взамен вы можете использовать post-process Material для рисования контура. Я (как и ожидалось) слил собственный Marketplace content в этот проект, создал экземпляр материала и изменил его под свои потребности (рисунок 17.9).

Чтобы назначить post-process Material, вы должны сделать это в настройках Post-Process Volume, включив **Blendables array**.

Для добавления записи в массив задайте для нее Asset Reference и либо выберите материал постобработки из списка, либо перетащите его из Content Browser в свойство в панели Details.

Этот материал использует **Custom Depth** буфер для определения, какие объекты выбраны, а какие нет. Вы устанавливаете Custom Depth для каждого актора в Levels Editor (рисунок 17.9).

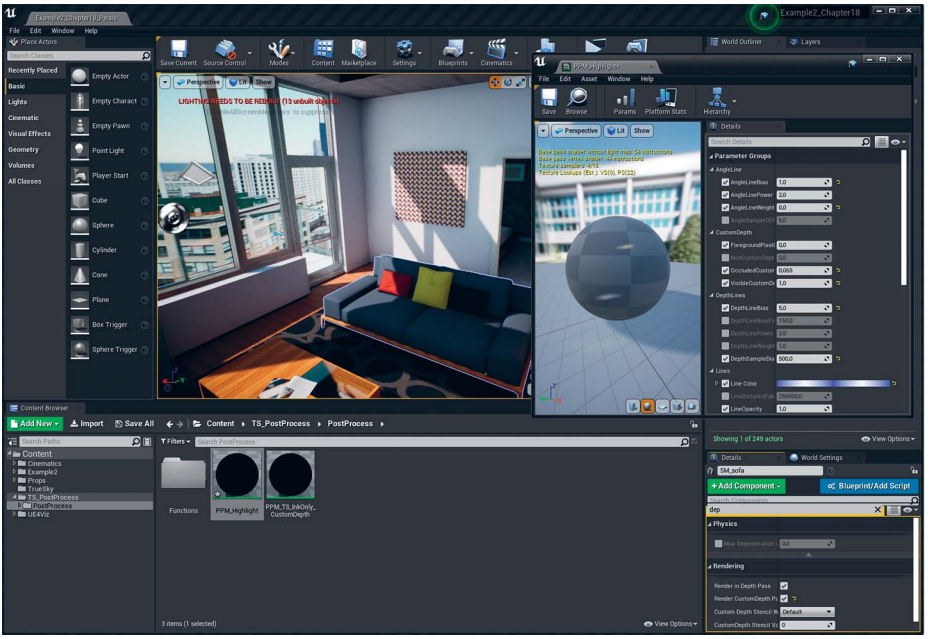


Рисунок 17.9 Тестирование Post-Process Material с помощью включения свойства Render Custom Depth для Static Mesh Actor дивана

Вам может потребоваться включить этот эффект в ваших Project Settings в разделе Rendering (рисунок 17.10).

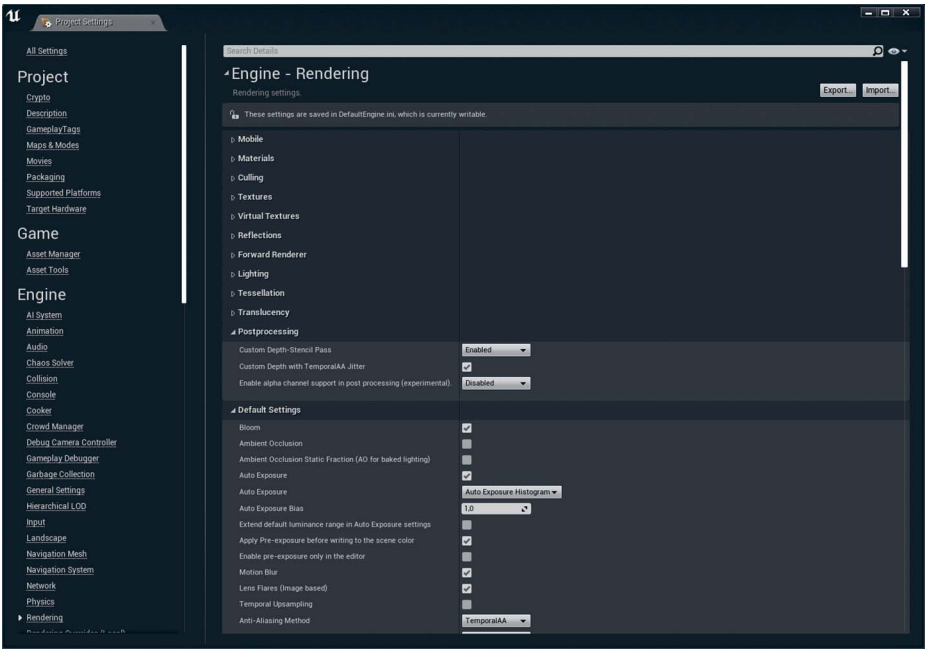


Рисунок 17.10 Включение Custom Depth буфера в диалоговом окне Project Settings

Заключение

Настройка ваших данных, позволяющая взаимодействовать с игроком, является важным процессом и обеспечивает ему или ей отличный опыт. Установка коллизий гарантирует, что Player может гулять по миру без страха упасть или застрять в геометрии, что испортило бы захватывающий опыт.

Включение курсора мыши открывает множество возможностей взаимодействия Player, включая разработку интерфейса в UMG и взаимодействие с внутриигровыми мешами с использованием событий ввода.

Теперь ваш проект готов, чтобы начать программировать интерактивность с помощью Blueprints.

ГЛАВА 18

БОЛЕЕ СЛОЖНЫЕ BLUEPRITNS: ВЗАИМОДЕЙСТВИЕ С UMG

Создание пользовательских интерфейсов — это вызов, с которым большинству профессионалов визуализации, вероятно, не приходилось сталкиваться раньше. Без правильных инструментов это может стать непростой задачей. UE4 представляет UMG (Unreal Motion Graphics), комплексное решение, которое может быть использовано для разработки полноценных data-driven интерфейсов или простых кнопок переключения и логотипов, которые необходимы для создания отточенной и простой в использовании интерактивной визуализации.

Переключение данных

Следующая установленная цель этого проекта позволит игроку переключаться на альтернативный вариант вашего уровня, сохраняя одно и то же положение камеры. Такое контекстное переключение — одно из самых мощных преимуществ интерактивной визуализации. Предоставление игроку возможности сравнивать различные данные с одного и того же вида является отличным способом анализировать альтернативы и позволяет ему выбирать, где разместить камеру, тем самым сильно расширяя возможности.

UE4 содержит систему под названием **Level Streaming**, предназначенную для загрузки и выгрузки целых уровней во время работы. Она разработана, чтобы позволить играм иметь очень большие уровни, разбитые на секции, которые загружаются и выгружаются, когда игрок перемещается по ним без экранов загрузки, прерывающих ход игры.

В этой главе вы изучите, как пользоваться этой системой для одновременной загрузки двух версий вашей карты и скрытия и отображения их сначала с помощью простого переключения клавишами для тестирования, а затем путем создания UMG UI для использования игроками.

Чтобы сделать это, вы сначала должны разработать новый уровень, основанный на ваших новых данных, вместе с Lighting, Materials и Props. Вы будете использовать уже существующий уровень как основу, многократно используя ваши работы из предыдущих глав, сохраняя много времени, которое тратится на разработку.

После этого вы создадите на основе UMG (Unreal Motion Graphics) пользовательский интерфейс, который позволит игроку просто нажимать кнопки для мгновенного переключения между различными вариантами. Вы изучите, как создавать UMG Widget Blueprint и связывать его с Viewport и как он принимает пользовательский ввод и выдает команды для изменения игрового мира.

Создание вариаций уровней

Клиент предоставил вам альтернативную планировку пространства. В ней присутствует превосходный лофт над спальней (рисунок 18.1).

Это значительное изменение, которое повлияет на внешний вид, ощущение и освещение на уровне, поэтому лучше всего создать совершенно новый уровень, с собственными освещением и геометрией.

К счастью, вам не нужно это делать с нуля, можно использовать уже существующий уровень и использовать его за основу. Сначала загрузите уровень в редактор, если еще не сделали это. Вы можете загружать уровни, перейдя в **File** > **Open Level** или найдя UMAP Asset в Content Browser и дважды нажав на него.

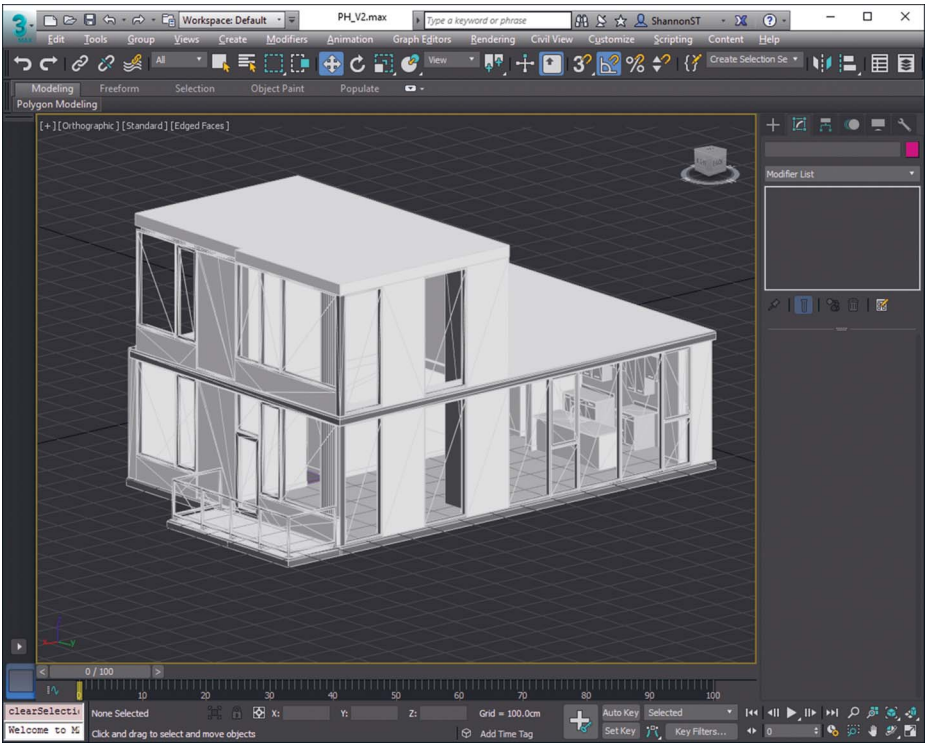


Рисунок 18.1 Обновленные данные в 3DS Max

Making a Copy with Save As

Создайте копию уровня, выбрав **File** > **Save Current As,** и затем назовите его как-нибудь наглядно, например **Example2_V2_MAP**.

Так вы получите два уровня в вашем проекте: **Example2_MAP** и **Example2_V2_MAP**.

На данном этапе уровни идентичны, за исключением названия.

Импорт новой архитектуры

Поскольку ваши уровни имеют много общего, включая расположение опор, освещение и другие элементы, вам нужно только поменять архитектурные меши. Вы также можете выбрать для замены только меши, которые изменяются, но для примера давайте создадим полностью уникальный уровень с полностью уникальной геометрией.

Как в главе 12 «Процесс работы с данными», приготовьте ваш контент в 3D-приложении. Примените UVW mapping для проверки на плохую геометрию и упорядочьте ваш

контент, как вы делали ранее. Вам следует придумать новый вариант наименования для этих мешей, чтобы избежать возможных конфликтов.

После подготовки экспортируйте в формате FBX и импортируйте в UE4.

Когда вы импортируете FBX-файлы в UE4, вам следует переместить контент в новую папку, отдельно от предыдущих архитектурных мешей; это позволит поддерживать два набора данных раздельно, избегая конфликтов данных.

Как и раньше, импортируйте FBX-файлы в Content Browser используя предложенные параметры для статических архитектурных мешей — очень важно убедиться, что **Auto Generate Collision** установлен на false, **Generate Lightmap UVs** установлен на true, и для **Transform Vertex to Absolute** также стоит true (рисунок 12.5 в главе 12).

После импорта FBX-файлов не забудьте сохранить Static Mesh Assets, которые вы только что создали, прежде чем двигаться дальше.

Замена архитектурных мешей

В вашем *новом* уровне (Example2_V2_MAP) выберите все Architecture Static Mesh Actors на сцене. Нажмите кнопку Del или правой кнопкой мыши на акторах и выберите Delete для их удаления, оставив только Props, Lights и других Actors.

Для расположения новых архитектурных мешей перетащите их во Viewport и затем переустановите их позицию на 0,0,0 с помощью свойства Location в панели Details. Теперь вы обновили архитектуру.

Пока ваши недавно расположенные Static Mesh Actors все еще выбраны, самое время также систематизировать их в папку в **World Outliner**, чтобы вы могли в будущем легко выбрать их.

Настройка Lightmap Density

Теперь, когда вы заменили стены, полы и потолки, они вернулись к своему стандартному Lightmap-разрешению. Вы должны просмотреть и изменить во вновь размещенных мешах свойство **Overridden Light Map Res**, используя Lightmap Density Optimization View Mode.

Вы должны попытаться сделать так, чтобы все как можно лучше соответствовало плотности, которую вы установили на предыдущей карте. Для гарантии, что билд света согласуется между двумя уровнями.

Применение материалов

Ваши недавно импортированные меши, несомненно, будут либо лишены материалов, либо будут применены импортированные материалы по умолчанию. Потратьте время, чтобы применить материалы на первом уровне.

Включение Collision

Последнее, что вам осталось сделать, — это убедиться, что ваша Collision установлена правильно. Используйте Player Collision View Mode, чтобы определить, для каких мешей нужно включить Collision.

Декорации (опционально)

Воспользуйтесь этой возможностью, чтобы внести столько изменений уровня, сколько захотите, — попробуйте различные Lighting, Materials, Props, назовите его. В этом примере Props, Lighting и Materials остались прежними, изменившись только в архитектуре. Это позволит игрокам сконцентрироваться на различиях, не отвлекаясь на другие изменения.

Билд освещение

Освещение хранится отдельно для каждого уровня. Таким образом у вас могут быть одинаковые ассеты для разных уровней с сильно различающимися настройками освещения и Lightmaps. Это позволяет запекать освещение для каждого уровня, пока используются одинаковые ссылки на ассеты.

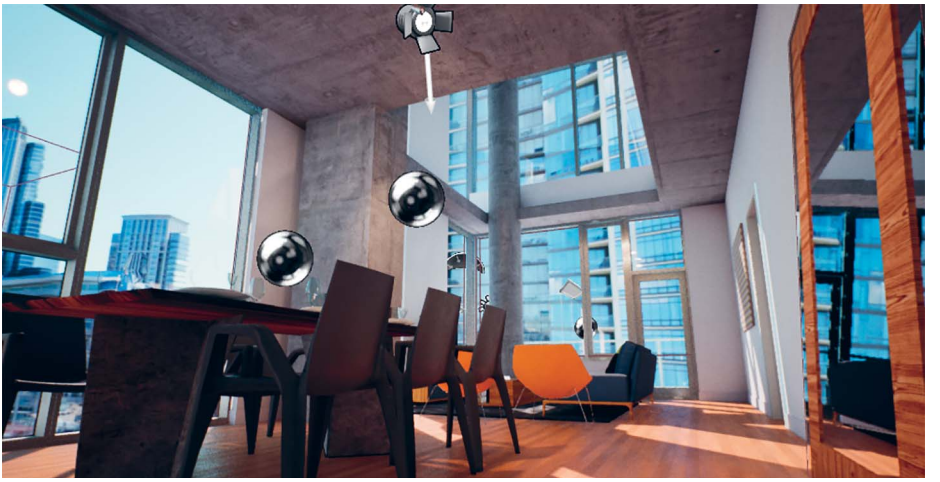


Рисунок 18.2 Готовое освещение, построенное для новых данных

Преимущество использования метода Save As для создания нового уровня состоит в том, что он сохраняет все World Settings, которые вы применили ранее, включая настройки Lightmass. Это делает создание освещения действительно очень простым; вы просто должны иметь возможность установить Lighting Build Quality для какого хотите уровня и нажать кнопку Build.

После создания освещения у вас должен быть новый вариант вашего уровня с потрясающими сводчатыми потолками и соответствующими изменениями освещения (рисунок 18.2).

Сохраните уровень. Карты света и теней для уровня будут записаны (или, начиная с версии 4.15, в отдельный файл Build Data рядом с уровнем, который виден только в Content Browser или File Explorer).

Level Streaming

UE4 может загружать и выгружать целые уровни на лету. Это называется **Level Streaming,** и эта система может использоваться для перемещения больших наборов с помощью команд Blueprint.

Способ работы Level Streaming довольно прост. Сначала загружается уровень, который называется **Persistent Level**. Этот уровень часто очень прост или даже почти полностью пустой.

Сам этот уровень или объект, или актор внутри него (к примеру, Blueprint или Player Controller), может загрузить другой уровень или уровни в или из него. Вы также можете переключать видимость загруженных уровней в редакторе и в процессе исполнения, обеспечивая удобный способ быстрого переключения наборов данных.

Для настройки вашего Level Streaming сначала создайте Persistent Level, затем добавьте ваши две версии в качестве Streaming Levels с помощью интерфейса **Levels** (рисунок 18.3).

Создание нового уровня

Давайте сначала создадим новый, пустой уровень для вашего Persistent Level. Перейдите в **File** > **New Level** и выберите **Empty Level** из доступных вариантов.

Когда ваш новый, пустой уровень откроется, сохраните и назовите его (к примеру, Example2_Persistent_ MAP). Эта карта содержит только некоторый код Blueprint в Level Blueprint для управления переключения ваших Streaming Levels.

Доступ к Levels Interface

Editor предоставляет доступ к системе Level Streaming через список Levels. Вы можете получить доступ к нему, перейдя в **Window** > **Level**. Это откроет окно Levels (рисунок 18.3)

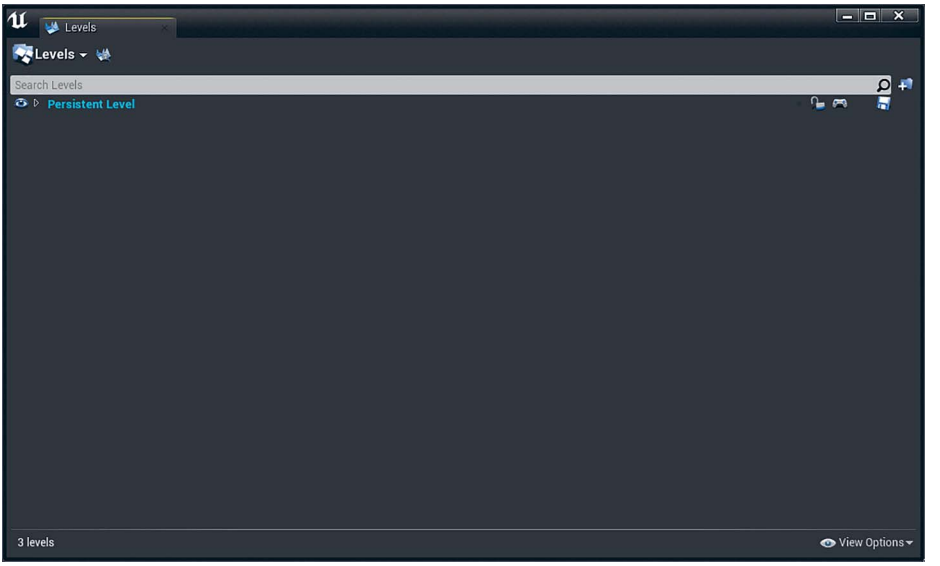


Рисунок 18.3 Окно Levels

Вы увидите текущий загруженный уровень в списке как *Persistent Level*.

Добавление Streaming Levels

Нажмите кнопку Levels в левом верхнем углу окна Levels и выберите **Add Existing** из раскрывшегося меню. Выберите исходный уровень (Example2 _MAP), который вы сделали (рисунок 18.4). Вы также можете создать новую, пустую карту streaming или новую карту с выбранными Actors.

Карта загрузится в Viewport, и вы увидите ее в списке в окне Levels.

Проделайте это снова для добавления новой версии (Example2 _V2_MAP) в окно Levels.

Теперь вы должны увидеть оба уровня в окне Levels, как и во Viewport.

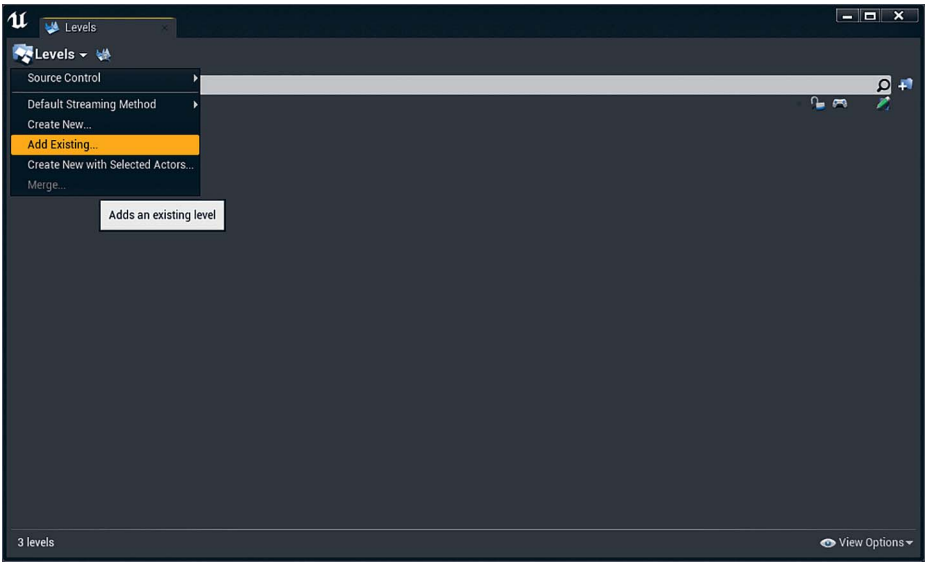


Рисунок 18.4 Добавление streaming Level с существующей картой

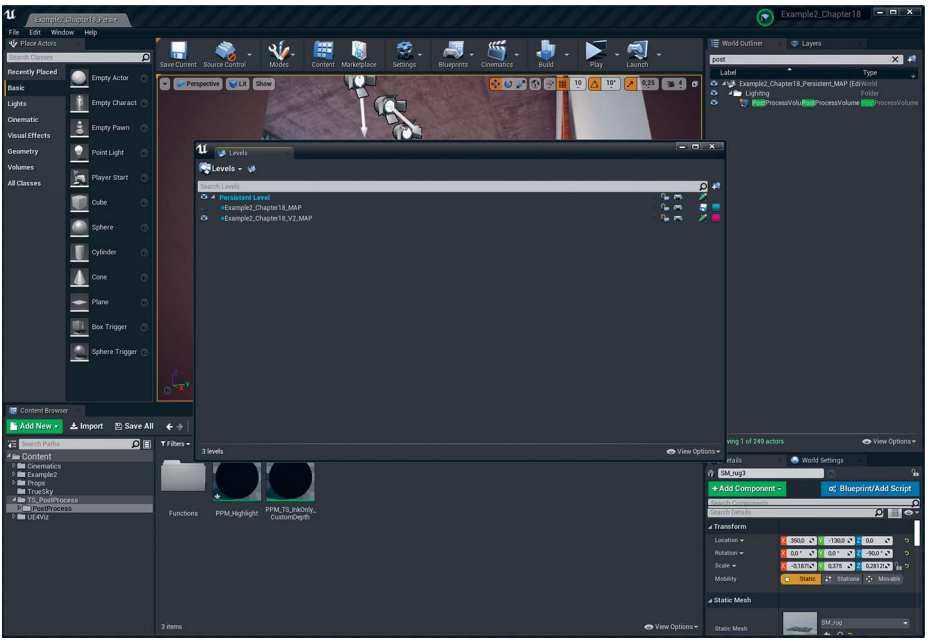


Рисунок 18.5 Обе версии уровня загружены в окно Levels

Вы можете скрывать и отображать уровни с помощью значка глаза для каждого уровня. Вы также можете сохранять уровни, открыв их Level Blueprints и переключив

блокировку редактирования для предотвращения изменений. Даже если ваш проект не использует Level Streaming в процессе работы, его использование может быть хорошим способом разбить большую сцену на маленькие файлы или для организации всего (рисунок 18.5). Обратите внимание, что на рисунке Example2_MAP скрыт с помощью значка глаза, который теперь отображается как закрытая иконка глаза.

Помните, что переключатели обзора, установленные для каждого уровня в окне Levels, доступны только для редактора и что видимость уровня в процессе работы обрабатывается с помощью Blueprint.

Теперь вам нужно сохранить ваш Persistent Level, так как вы изменили его добавлением Streaming Levels с помощью интрейфейса Levels.

Использование Blueprints и постоянно загруженные уровни

Обратите внимание на маленькую синюю точку рядом с вашими двумя новыми уровнями в окне Levels. Это значит, что уровни загружаются с помощью Blueprints и что они не будут загружаться до тех пор, пока вы не сообщите им об этом с помощью Blueprints. Вы также можете выгружать, скрывать и отображать эти уровни в процессе работы[17].

Кроме того, можно установить уровни в состояние Always Loaded. Как следует из названия, эти уровни всегда загружаются и не могут быть скрыты или выгружены в процессе работы.

Уровни, которые необходимо всегда загружать, можно установить с помощью двойного нажатия по уровню в окне Levels, выбрав **Change Streaming Method** > **Always Loaded**. Эти уровни будут загружаться так, как если бы они были частью Persistent Level, когда игра запустилась.

Сохраняйте ваш Streaming Method для уровня как Blueprints, чтобы иметь возможность скрыть или отобразить их во время работы с помощью Blueprints.

Определение Player Start Actor

Даже несмотря на наличие Player Start Actors на двух ваших Streaming Levels, нужно определить одного для нового Persistent Level. Иногда это несколько сбивает с толку, так как вы можете ясно видеть Player Start Actors и выбрать их в Editor Viewport, но они не появятся здесь при начальной загрузке уровня. Это связано с тем, что Player Controller создается до того, как Streaming Level сможет загрузиться.

[17] Выгрузка и загрузка Levels удаляет их из памяти и может занять некоторое время в процессе работы. Скрытие или отображение сохраняет Level загруженным, но просто включает рендеринг содержимого этого Level. Это происходит мгновенно.

Вы можете легко копировать и вставлять один за другим ваших Player Start Actors с одного из ваших Streaming Levels в Persistent Level. Выберите Player Start из Content Browser или Viewport и затем перейдите в **Edit** > **Copy** или выберите Edit > Copy из меню, открывающегося нажатием правой кнопки.

Вставка требует немного больше внимания. Когда у вас есть несколько уровней, загруженных в редактор, вам нужно определить, какой уровень является *активным*, прежде чем ставить Player Start.

Чтобы этот Player Start был расположен в Persistent Level, нажмите дважды по Persistent Level в окне Levels, сделав его активным. Вы сможете узнать активный слой, так как название станет синим.

Вставьте Player Start Actor в Persistent Level используя меню Edit, меню, вызванное правой кнопкой мыши, или просто используя Crtl/Cmd+V.

Вы также можете разместить Player Start используя Class Browser, как вы делали раньше, но вам все равно нужно убедиться, что активен правильный уровень.

Теперь у вас есть Player Start, размещенный в вашем Persistent Level. Однако если вы нажмете Play сейчас, то вы просто загрузитесь в темный пустой мир. Вам нужно настроить Level Blueprint для загрузки streaming Levels.

Настройка Level Blueprint

Теперь, когда ваши Streaming Levels настроены в редакторе, вы можете написать Blueprint логику для их переключения с помощью **Level Blueprint**.

Открытие Level Blueprint

Каждый уровень содержит собственный Blueprint Event Graph, называемый **Level Blueprint**. Этот Blueprint является отличным способом сделать Level-specific действия; вещи, которые будут происходить только на одном уровне, основанном на особых акторах или событиях этого уровня. В качестве примера можно привести срабатывание открытия двери игроком или настройку воспроизведения конкретной музыки, когда уровень открывается.

Любая функциональность, которая должна быть общей от уровня к уровню (например, движение игрока) должна обрабатываться обычным Blueprint Class, так как вам не нужно дублировать и поддерживать этот код для каждого созданного Level[18].

[18] Во время работы все ваши уровни загружены в единый мир, и все могут получить доступ ко всему остальному. Однако в редакторе акторы и уровни могут получить доступ только к другим акторам на том же уровне.

Для доступа к Level Blueprint вы можете нажать на значок Gamepad в окне Levels, соответствующий уровень, который хотите редактировать, или нажать кнопку Blueprints из панели инструментов и выбрать **Open Level Blueprint** (рисунок 18.6).

Вы также можете получить доступ к Level Blueprints загруженных уровней и получить быстрый доступ к вашим Game Mode's Classes.

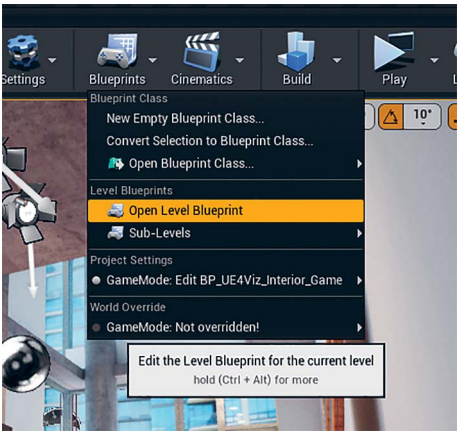


Рисунок 18.6 Открытие Persistent Level Blueprint

Заметка

Streaming Levels могут содержать собственные Level Blueprints. Так как Level Blueprint может только ссылаться на Actors, которые находятся на своем уровне.

Level Blueprint открывается в окне Blueprint Editor (рисунок 18.7). Этот редактор немного отличается от Blueprint Editors, который вы использовали до сих пор. В нем отсутствуют вкладки Viewport, Components и Construction Script. Level Blueprints содержит только Event Graph, потому что они не могут иметь компоненты и не собираются, как классы Actor.

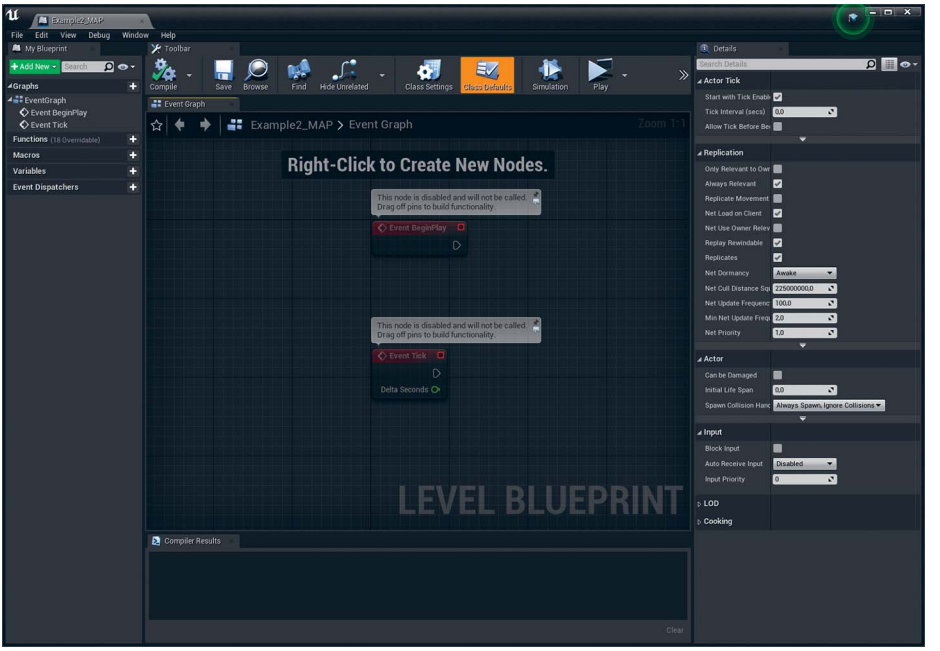


Рисунок 18.7 Level Blueprint, открытый в Blueprint Editor

Использование Events

Event является специальным узлом, который вызывается из кода игрового процесса. При вызове он запускает граф узла, соединенный с ним выходным контактом выполнения (белая стрелка). Эти события могут вызываться в ответ на различные события игрового процесса, такие как начало игры, перезапуск уровня или если игрок нажал на определенную клавишу.

UE4 содержит множество предопределенных узлов событий. Два наиболее распространенных изображены на рисунке 18.7 выше: **BeginPlay** и **Tick**. Также помните, что ранее вы использовали **InputAxis** Events для настройки ввода вашего Player Controller.

BeginPlay

BeginPlay Event вызывается автоматически игрой один раз, когда уровень впервые загрузился, после загрузки и инициализации всех акторов и мировых объектов.

Здесь вы устанавливаете ваш начальный код Level Streaming. Как уже упоминалось, ваш Persistent Level пуст и должен содержаться в Levels streamed.

Для этого используйте функцию **Load Stream Level**. Вы можете увидеть на рисунке 18.8, что я расположил два этих узла. Как и в большинстве узлов, нажмите правой кнопкой мыши в Event Graph и найдите узел Load Stream Level.

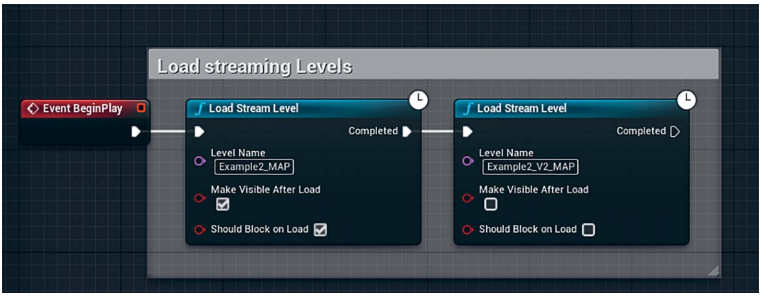


Рисунок 18.8 Узлы Load Stream Level, соединенные с узлом BeginPlay Event

После размещения обоих узлов вам нужно заполнить настройки для каждого из них, чтобы они соответствовали рисунку 18.8. Example2_MAP загружается первым, используя узел Load Stream Level с **Should Block on Load** и **Make Visible After Load,** установленным на **True**. Затем загружается Example2_V2_MAP, но изначально с помощью параметра Make Visible After Load, установленного на *False*.

Вы должны установить для свойства Name точно такое же имя, как и для уровня в окне Levels. Также уровень уже должен быть загружен как Streaming Level в окне Levels для доступа к потоковой передаче Blueprints.

Should Block on Load вынуждает UE4 ожидать загрузки первого уровня, прежде чем продолжить выполнение любого игрового кода (это и есть block — он блокирует продолжение игры). Блокировка предотвращает падение вашего объекта Pawn в пустой Persistent Level до загрузки Streaming Level и связанной с ним коллизии.

Заметка

Загрузка нескольких streaming Levels одновременно может потребовать очень много памяти. Если у вас возникают сбои или снижается производительность, попробуйте уменьшить нагрузку на память, особенно на VRAM вашей видеокарты. Начните с уменьшения разрешения Reflection Capture Actors и/или уменьшите разрешение Lightmaps.

Latent-функции

Обратите внимание на значок **Clock** на функциях **Load Stream Level**. Это указывает на то, что узел является **Latent-**функцией. Latent-функции требуют времени для выполнения и продолжают рабочий процесс по графу событий (Event Graph), только когда их задача выполнена.

Загрузка уровней может занять несколько секунд. Большие уровни и медленные жесткие диски могут увеличить это время. Вот почему мы решили загружать их

одновременно при начальной загрузке Persistent Level и переключать их видимость. Сохраняя уровни в памяти и просто меняя их видимость, мы можем избежать задержек и переключаться почти мгновенно.

Время тестов

Если нажать Play сейчас, ваше приложение должно загружать карту так же, как и раньше. Однако это, возможно, заняло немного больше времени по сравнению с тестированием с PIE до этого.

Это потому, что вы ждете, пока загрузятся уровни; даже карта V2, которая еще не отображается, требует времени для загрузки.

После загрузки уровней вы сможете перемещаться в пространстве, как это было в предыдущих главах, но на этот раз — с помощью Level Streaming.

Настройка Level Blueprint

Теперь, когда у вас есть загруженные карты, давайте заставим их переключаться. Это получится всего с несколькими узлами в вашем Level Blueprint. Вы также увидите, как создавать простые сочетания клавиш для тестирования переключения уровня, прежде чем приступить к разработке UMG-интерфейса.

Создание Custom Events

Вы можете создать собственный узел **Custom Event**, который можно вызвать в любой момент в Blueprint из другого Blueprint в вашем мире, тем самым предоставляя возможность для организации Event Graphs и позволяя Blueprints взаимодействовать друг с другом.

Вам требуется несколько Custom Events, которые могут переключать видимость уровней. Вы будете использовать эти события для тестирования переключения уровней и затем используете тот же Events позже, когда разработаете UMG-интерфейс.

Двойным нажатием по заднему плану Event Graph создайте в Persistent Level Blueprint узел Custom Event и выберите **Add Custom Event** из контекстного меню (рисунок 18.9).

Когда вы создаете Event, присвойте ему уникальное имя и нажмите Enter для подтверждения.

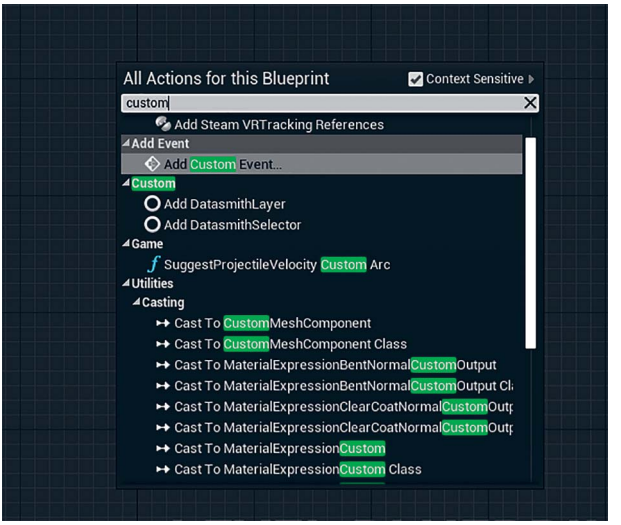


Рисунок 18.9 Добавление Custom Event для Level Blueprint

Вам требуется три события для работы системы переключения, поэтому разместите три узла Custom Event, назвав первый *ShowVersion1*, второй *ShowVersion2* и третий — *ToggleVersions*. Ваш Event Graph должен выглядеть так, как показано на рисунке 18.10.

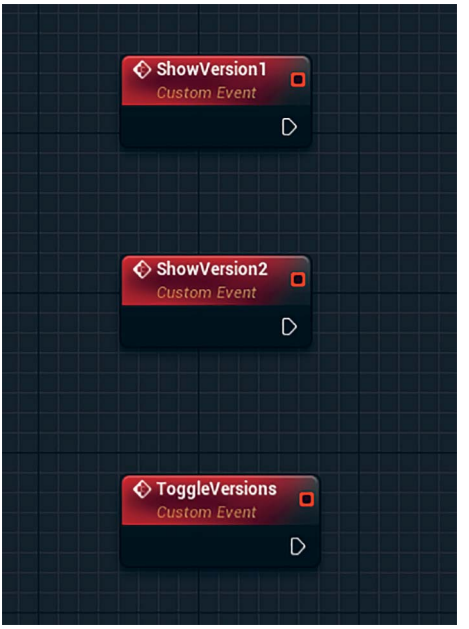


Рисунок 18.10 Три узла Custom Event, добавленные в Event Graph

Настройка Show Versions Events

Функции Show Version 1 и Show Version 2 просто устанавливают видимость для Streaming Levels, позволяя переключаться между ними.

Настройте Event Graph, как показано на рисунке 18.11.

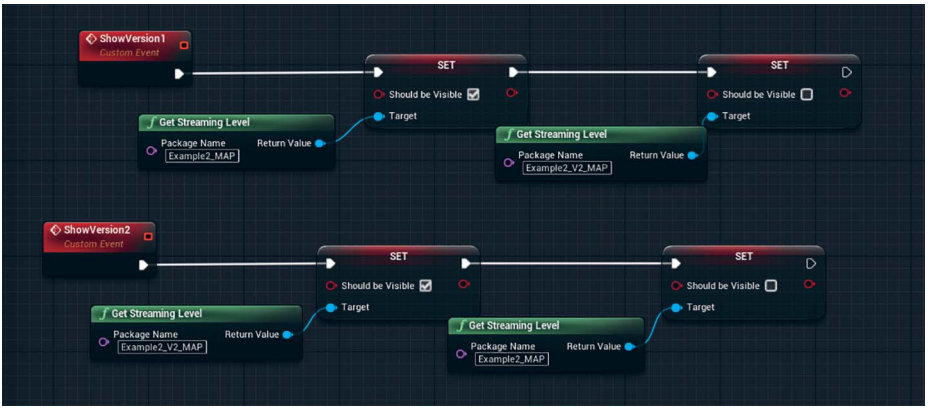


Рисунок 18.11 ShowVersion1 и ShowVersion2 custom Events и их графы выполнения

Эти два почти зеркальных изображения графов выполнения скрывают один уровень, пока показывают другой. Функция **Get Streaming Level** возвращает ссылку на уровень, определенный в свойстве Package Name. Вы должны вручную ввести это имя, и оно должно в точности совпадать со Streaming Levels.

Для доступа к свойству **Set Should be Visible** для уровня протяните соединение из синего контакта Return Value узла функции Get Streaming Level в Event Graph и отпустите для доступа контекстному меню для Streaming Levels, затем найдите **Set Visible**.

Настройте ваш первый не скрытый уровень так, чтобы скрывать другие. Вы увидите, что название уровня объявлено в Package Name узла Get Streaming Level.

Toggle Versions Event

Toggle Versions Event переключает между двумя вариантами при каждом вызове (рисунок 18.12). Узел **Flip Flop** специально разработан для этого, а еще он смешно читается.

При первом вызове Flip Flop он вызовет только вывод A. В следующий раз он вызовет только выход B, затем A и так далее, чередуясь между вызовами двух Custom Events, которые вы создали ранее.

Для создания вызова Custom Event нажмите правой кнопкой мыши в Event Graph и найдите имя, которое дали вашему Custom Event, когда создали его. Чтобы убедиться, что ваш Custom Events находится в списке, вы должны скомпилировать и сохранить Blueprint.

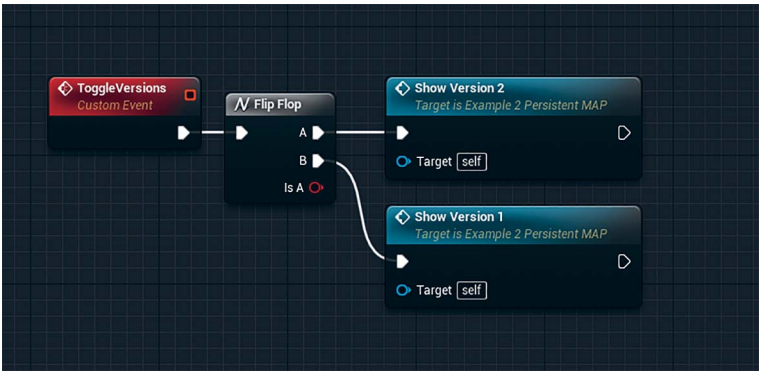


Рисунок 18.12 ToggleVersions и узел Flip Flop для переключения между двумя Custom Events

Время тестов

Теперь, когда вы все настроили, потратьте время на тестирование, прежде чем приступать к UMG-интерфейсу. Как вы можете проверить свою работу сейчас без UMG-интерфейса? Проще всего — создайте сочетание клавиш в Level Blueprint.

Создание шорткатов

Level Blueprints умеет перехватывать входные события от игрока так же, как Player Controller. Вы можете использовать это, чтобы легко настроить сочетания клавиш для вызова Toggle Versions Event.

В вызванном нажатием правой кнопки мыши контекстном меню найдите Input L (рисунок 18.13) и выберите L из списка Keyboard Events.

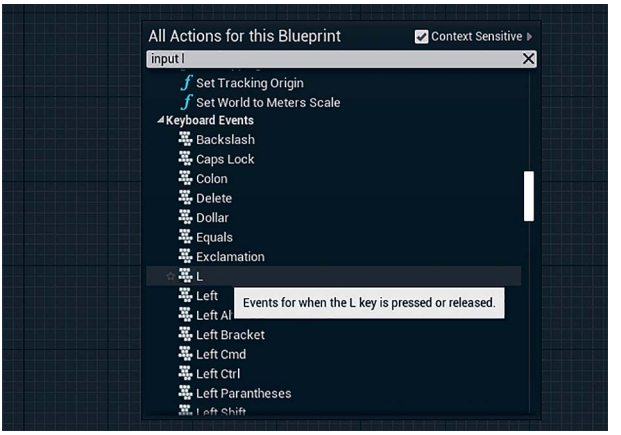


Рисунок 18.13 Создание Input Event для клавиши L

Затем просто соедините вызов **ToggleVersions** Event с контактом выполнения **Pressed** в **L Keyboard Input** Event, как показано на рисунке 18.14.

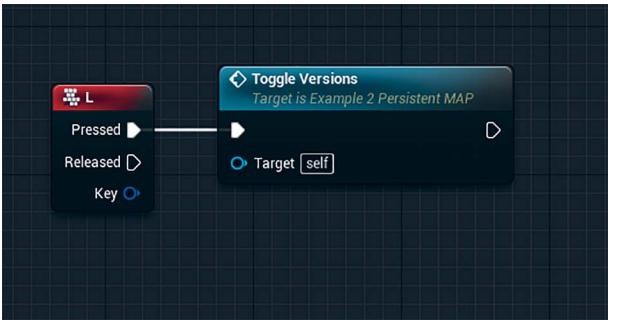


Рисунок 18.14 Абсурдно сложный код сочетания клавиш

Компиляция и сохранение

Теперь вы должны скомпилировать Level Blueprint, убедившись, что в нем нет ошибок или предупреждений, затем сохраните Persistent Level для сохранения кода, который написали.

Нажатие Play

Теперь, когда вы нажимаете Play, вам должно быть доступно нажатие кнопки L на клавиатуре, и ваш уровень немедленно переключится между двумя версиями.

Вы заметите, какое существенное изменение вносят дополнительные окна для всей сцены, даже в коридорах, далеко от основных изменений. Такая чрезвычайно мощная возможность переключения опций в контенте, как эта, дает игроку личный опыт. Нет двух игроков, которые будут видеть одно и то же место и переключаться в разные места одновременно.

Однако не ждите, что все знают, что нужно нажать L, чтобы эти изменения произошли. Вам нужно предоставить пользовательский интерфейс для игрока, который даст ему понятные опции. Для этого примените встроенную в Unreal систему пользовательского интерфейса UMG.

Unreal Motion Graphics (UMG)

Unreal Motion Graphics UI Designer (UMG) является инструментом визуализации UI, который вы можете использовать для создания игровых элементов, например меню, названий и кнопок. UMG аппаратно-ускоренный, современный и не зависящий от платформы, что означает, что он работает быстро, отлично выглядит и может быть применен

на любой платформе, поддерживающей UE4. Вы можете создать один интерфейс для использования на чем угодно, от PC и Mac до Nintendo Switch и всем, что между ними.

Использование Widgets

UMG полагается на виджеты (**Widgets**), готовые элементы, которые вы можете использовать для создания вашего интерфейса. Предопределенные виджеты доступны для большинства элементов UI, включая кнопки, ползунки, выпадающие списки и текстовые метки, а также виджеты, помогающие организовать и упорядочить другие виджеты в UI.

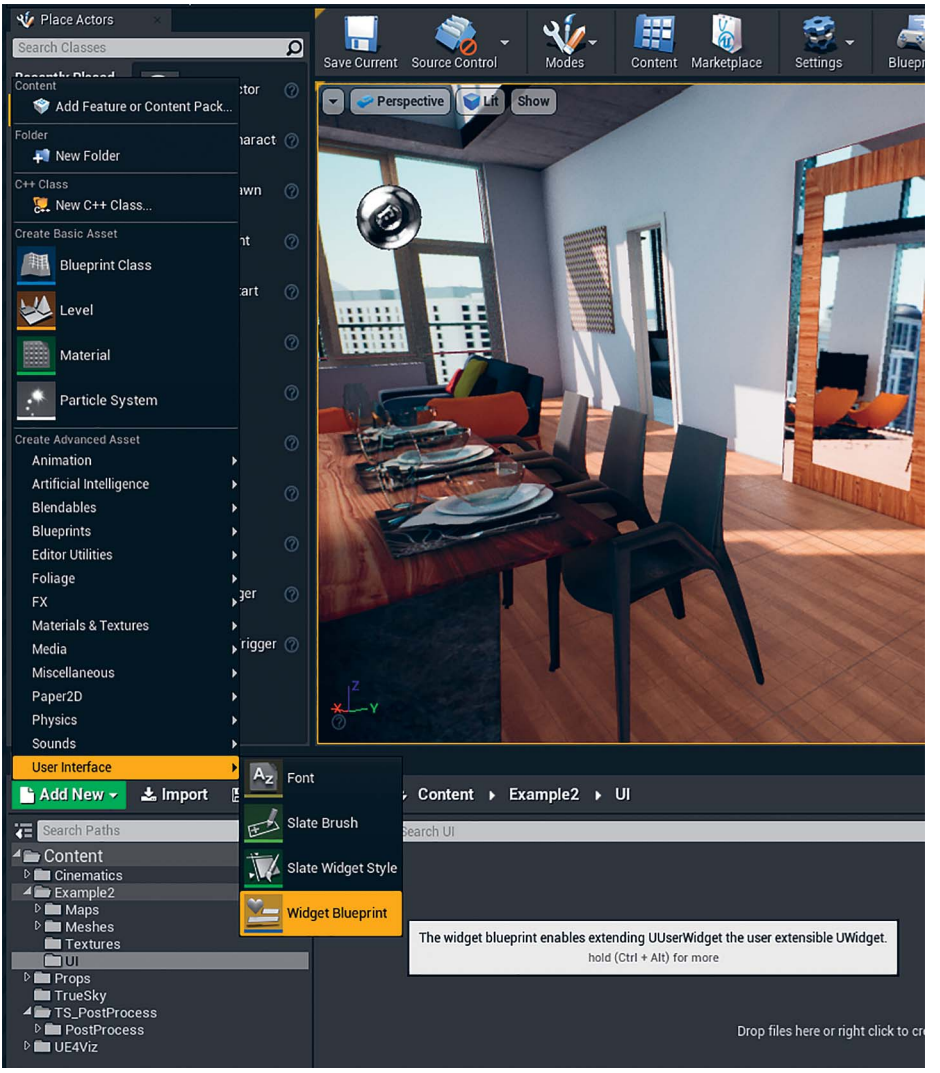


Рисунок 18.15 Создание Widget Blueprint

Виджеты собраны в **Widget Blueprint**, специальном Blueprint Class с настраиваемым Editor.

Вы создадите Widget Blueprints, как и большинство UE4 Classes, в Content Browser. В меню Add New перейдите в **User Interface** > **Widget Blueprint** (рисунок 18.15). Назовите ваш новый Widget **UI_Example2_HUD**. HUD означает Heads up Display и обычно называется интерфейсом, который всегда показывается во время игры. Другими распространенными примерами UI могут быть главное меню или меню паузы.

Теперь нажмите дважды по недавно созданному Widget Blueprint, чтобы открыть его для редактирования (рисунок 18.16).

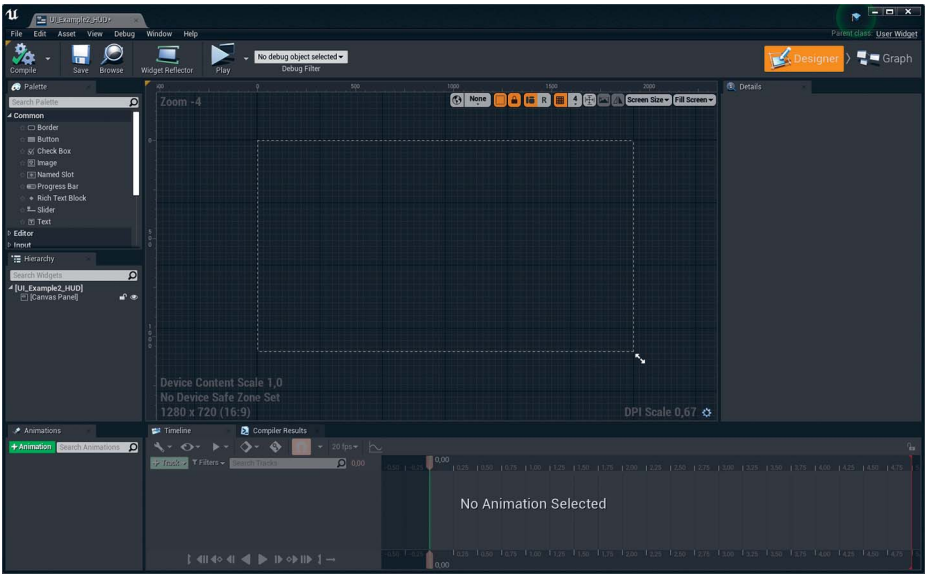


Рисунок 18.16 Окно Widget Blueprint Editor

Этот редактор состоит из двух основных разделов: вкладки Designer, которая позволяет вам визуально настраивать элементы UI, и вкладки Graph, где вы можете добавить функциональность для ваших UIs. В правом верхнем углу вы можете увидеть вкладки **Designer** и **Graph**. Нажмите на них, чтобы переключиться между двумя режимами интерфейса. В центре **Stage**, а слева **Palette** со всеми виджетами, доступными создания вашего UI. Ниже находится **Hierarchy**, которая показывает расположение Widgets во вложенном списке. Внизу находится **Animation** List и **Timeline**. Вы можете использовать их для разработки ключевой анимации для ваших элементов UI. Наконец, справа находится **Details Panel**. Как и все остальные панели Details в UE4, она контекстная и отображает детали выбранного в данный момент виджета.

Horizontal Box

Для этого UI вам нужны две кнопки, позволяющие переключаться между вашими вариантами. Вы наверняка хотите, чтобы кнопки были аккуратно расположены вдоль нижней части экрана с равномерными интервалами, чтобы все смотрелось чисто и профессионально.

UE4 поставляется с виджетом, который помогает с этим: **Horizontal Box** Widget. Он может содержать несколько дочерних виджетов, вложенных в него, равномерно распределяя каждый по горизонтали.

Найдите **Horizontal Box** Widget, указанный в окне **Palette,** под группой Panel. Перетащите его в **Stage**. Виджет появится в Stage и в списке Hierarchy.

Ваш виджет, вероятно, не попал в действительно правильное место, таким образом, вам нужно передвинуть его в нижнюю часть экрана, куда вы хотите. Вы можете просто перетащить его, но это не лучшая практика.

UMG поддерживает произвольное разрешение и масштабирование, поэтому он может быть использован на достаточно большом количестве устройств с любыми экранами.

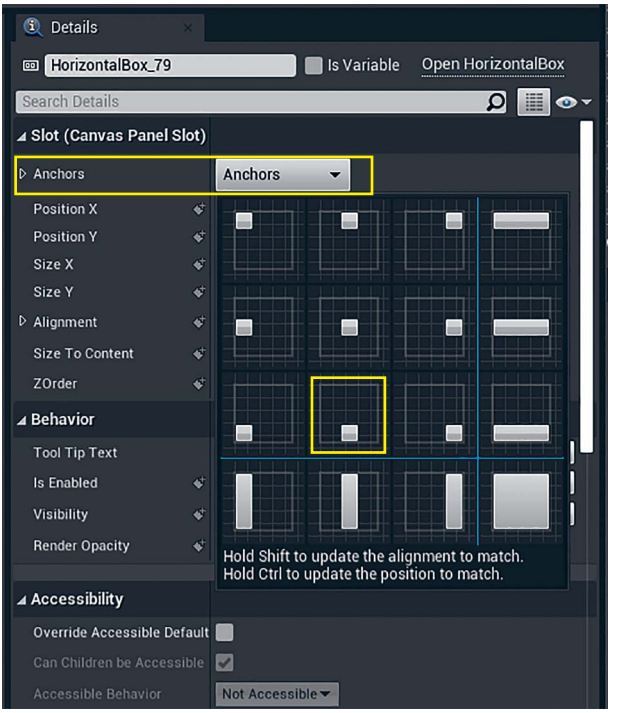


Рисунок 18.17 Настройки Widget Anchor

Одним из способов, применяемых для работы UMG, является использование Anchors для Widgets. Anchor гарантирует, что Widget всегда привязан к относительной части экрана: стороне, углу или центру.

В данном случае прикрепите его к нижнему центру, нажав на раскрывающийся список Anchors в панели Details (рисунок 18.17).

Когда вы сделаете это, то не увидите больших изменений в Viewport, потому что UE4 пытается регулировать все свойства расположения так, чтобы Widget оставался в том же месте. Вы можете изменить эти свойства, чтобы получить Horizontal Box Widget там, где хотите, и такого размера, который хотите (рисунок 18.18).

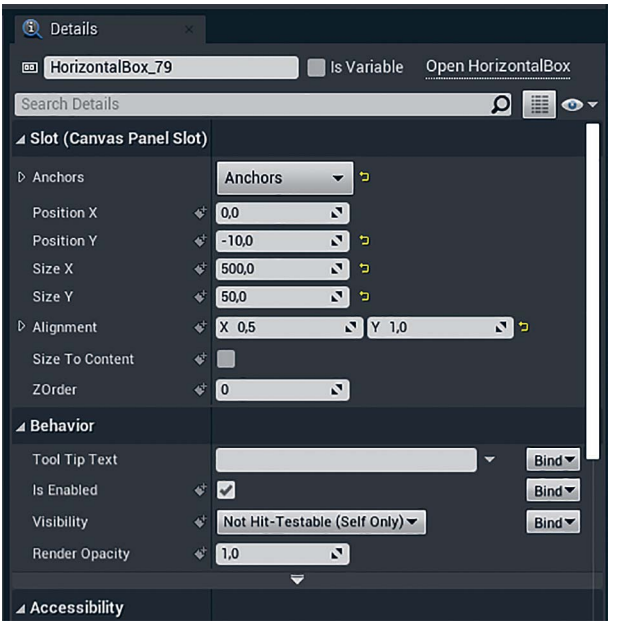


Рисунок 18.18 Настройка свойств Slot для Horizontal Box Widget's

1. Установите **Position X** на 0 и **Position Y** на –10. Это центр Widget, но сдвинутый на 10 пикселей от нижней части экрана.

2. Установите **Size X** на 500 и **Size Y** на 50 для определения размера поля.

3. Установите **Alignment**. Это в основном точка опоры смещения. Если вы установите ее на 0,0, Widget будет перемещен к верхнему левому углу, а при 1,1 будет перемещен к нижнему правому углу. Alignment установленный на 0,5; 1,0 переместит опорную точку Widget к центру низа.

Теперь у вас должно быть поле, готовое к организации некоторого контента, так что давайте дадим ему немного контента!

Кнопки

Просто перетащите два Button Widgets из Palette в Horizontal Box Widget. Вы можете сделать это либо в Stage, либо в окне Hierarchy (рисунок 18.19). Перетаскивание в окно Hierarchy может очень помочь, когда ваши UI Widget становятся сложными.

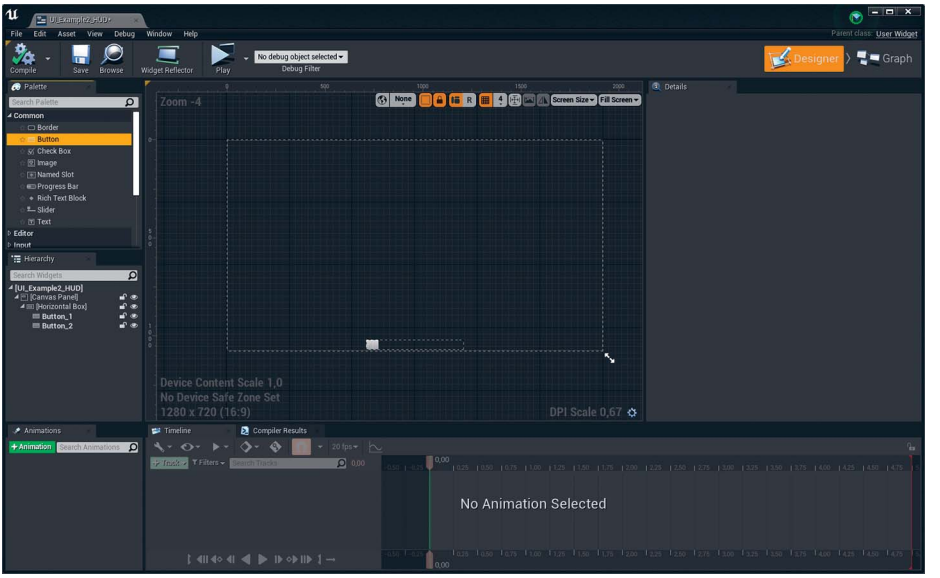


Рисунок 18.19 Кнопки в Horizontal Box Widget выглядят неидеально

Вам нужно немного стилизовать и настроить кнопки, чтобы они выглядели и вели себя правильно.

Выбирайте кнопки одну за другой и настройте их следующим образом (рисунок 18.20).

1. Главное — вы должны дать каждой вашей кнопке уникальное название. Если вы этого не сделаете, попытка получить доступ к Widgets из Blueprints может стать проблемой.

2. Установите для **Padding 10** пикселей слева и справа. Вам, возможно, потребуется развернуть параметр Padding, нажав по стрелке рядом с ним, чтобы отобразить отдельные параметры.

3. Установите **Size** для **Fill** в значение **1,0**. Это заставит кнопки заполнять любую доступную площадь, а не пытаться быть как можно меньше (Auto).

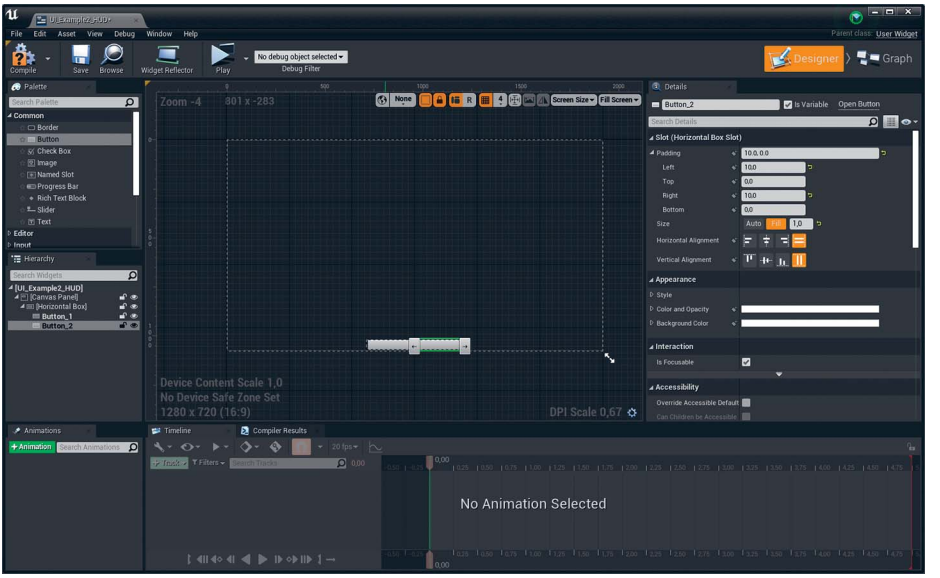


Рисунок 18.20 Настройка свойств Slot для Button Widget's

Метки

Вам нужны некоторые метки (Labels) для ваших кнопок. К счастью, Button Widget Class может иметь один дочерний класс — Text Widget, который идеально подходит для использования в качестве метки.

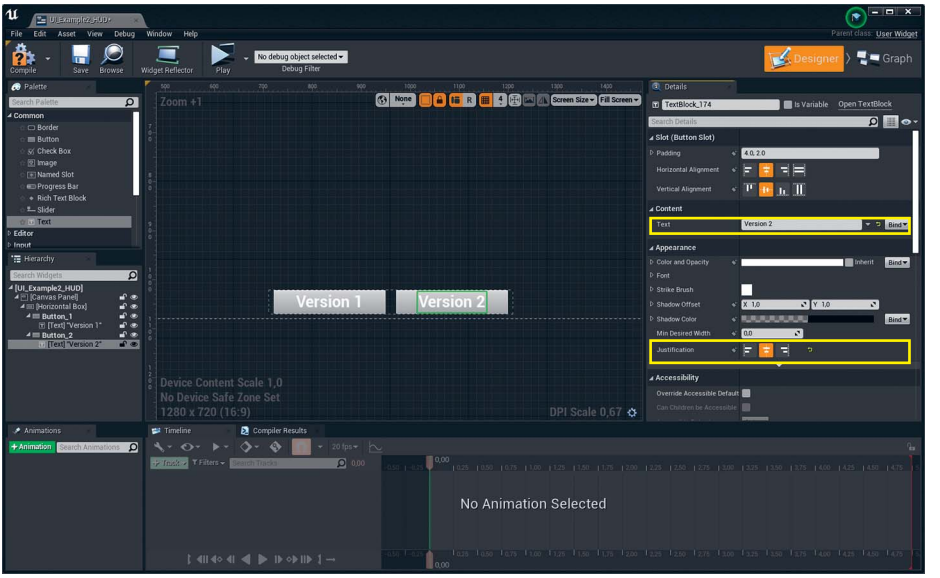


Рисунок 18.21 Добавление Labels

Найдите Text Widget в Palette и перетащите его в каждую из ваших кнопок. Установите в Label свойство Text для чтения Version 1 и Version 2. Вы также, возможно, захотите добавить небольшую тень для текста, чтобы сделать его более разборчивым (рисунок 18.21).

Самое время сохранить вашу работу, если вы еще не этого сделали. Ваш интерфейс готов, и вам не надо добавлять какой-либо код непосредственно в графе этого Widget Blueprint, так как вы будете обрабатывать все это в Level Blueprint.

Вернемся к Level Blueprint

У вас есть все части, необходимые для работы этой системы; теперь вам осталось только собрать их. Всю финальную сборку вы сделаете в Level Blueprint.

Откройте Level Blueprint, нажав кнопку **Blueprints** в **Toolbar**, и выберите **Open Level Blueprint**.

Наиболее очевидный вопрос на данном этапе: как получить мой UI, который я только что создал, и отобразить его? Для этого в UE4 присоедините ваш Widget Blueprint к Viewport игрока с помощью Blueprints, как показано на рисунке 18.22[19].

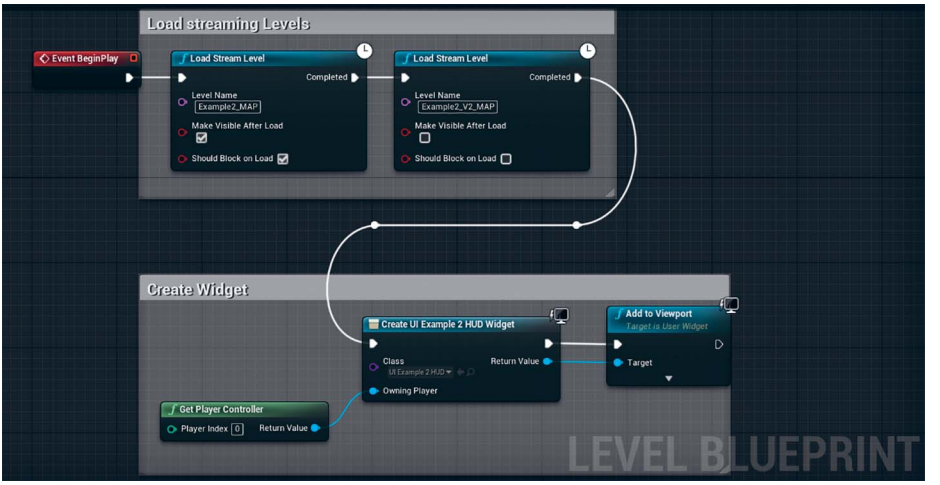


Рисунок 18.22 Создание экземпляра вашего Widget Blueprint и добавление его к Viewport

Лучшим местом, чтобы сделать это, является **Begin Play Event**, так как вы наверняка хотите, чтобы HUD отображался при загрузке уровня. Добавьте этот код после

[19] Обратите внимание на узлы Redirect в белом потоке исполнения. Они позволяют программисту сделать узлы графов более читабельными, определяя положение исполнительных линий. Для добавления одного дважды нажмите на соединение.

завершения загрузки вашего уровня, таким образом вы не будете показывать UI игроку прежде, чем тот сможет нажать на него. (Это грубо!)

Следующие разделы проведут вас через соединение всех компонентов вместе.

Создание Widget

Первым шагом является создание объекта Widget, который загружает класс виджета с диска и создает его экземпляр в памяти (вы, конечно, можете создать больше одного экземпляра любого Widget, все они являются экземплярами одного и того же Widget Blueprint Class).

Как обычно, нажмите правой кнопкой мыши в Event Graph и найдите **Create Widget** для создания этого узла.

После размещения узла Create Widget вам нужно сделать две вещи.

1. Назначить класс, который вы хотите создать; в данном случае UI_Example2_HUD, выбрав из раскрывающегося списка Class.

2. Предоставьте этой функции ссылку на ваш Player Controller. Просто нажмите правой кнопкой мыши в Viewport, найдите Get Player Controller из списка и затем соедините его с контактом Owning Player.

Вам необходимо это сделать, так как в многопользовательской среде может быть больше одного Player Controller, и вам нужно знать, кому принадлежит HUD. В данном случае у вас есть только один, поэтому вы можете просто ссылаться на первый доступный Player Controller.

Добавление Widget во Viewport

Чтобы прикрепить Widget к Viewport и быть видимым и не взаимодействующим, вам нужно добавить его в Viewport.

Для доступа к узлу Add to Viewport перетащите соединение из Return Value узла Create Widget (который теперь должен читаться как Create UI Example 2 HUD Widget) и отпустите в Graph Editor, который откроет контекстное меню для вашего Widget Class. Найдите узел **Add to Viewport** и добавьте его в граф. Синее соединение автоматически подключится к контакту Target недавно созданной функции Add to Viewport.

А теперь пора нажать на кнопку Compile и, если ошибок нет, сохраните ваш Level Blueprint.

Если вы запустите игру сейчас, вы увидите, что ваши две кнопки отображаются внизу экрана. Если вы щелкните по ним, то ничего не произойдет. К счастью, вы наконец-то

настроили функции переключения; вам осталось только установить вызов этих функций при нажатии кнопок.

Event Binding

Мощной особенностью UE4 является способность для одного Blueprint привязываться к Event в другом. Это позволяет одному Blueprint обрабатывать несколько взаимодействий или Events с помощью одного унифицированного кода.

Для обнаружения событий вы сначала должны получить ссылку на кнопки, которые сделали (рисунок 18.23). Это легко, потому что метод Create Widget возвращает ссылку на Object, который создал. Таким образом, перетащите соединение из Return Value в graph для доступа к контекстному меню вашего Widget Class.

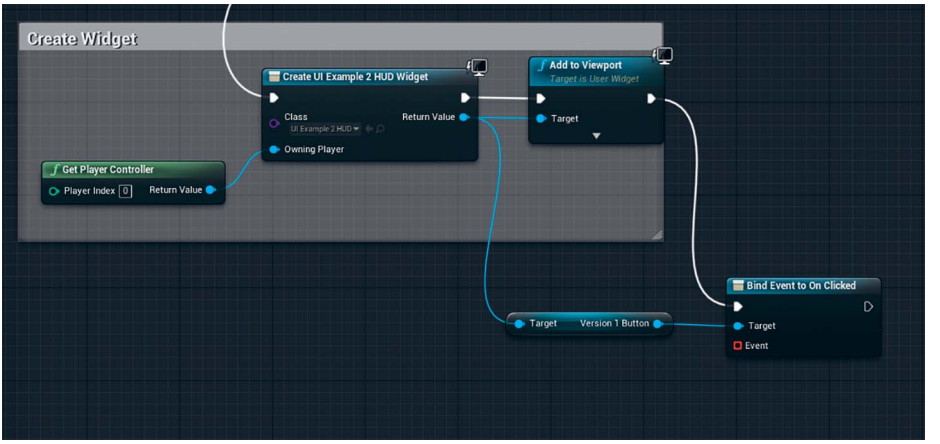


Рисунок 18.23 Получение ссылки на кнопку, которую вы создали в HUD Widget

Найдите **Get Version** и выберите **Get Version 1 Button**. Из этого узла **Get** снова вытяните синее соединение и найдите **Bind**. Вы можете привязать несколько событий, и так же вы можете отвязаться от Events. Выберите **Bind Event to OnClicked** для вашей кнопки. Подключите его к функции Add To Viewport, потому что он должен быть вызван для Bind, которую вы настроите для получения эффекта.

Теперь вам нужно определить, какое событие будет вызваться при запуске привязанного OnClicked Event. Вы сделаете это с красным/оранжевым контактом ссылки на Event.

Перетащите из этого контакта, похожего на контакт в ShowVersion1 Custom Event. Вам может понадобиться передвинуть ваши узлы поближе, чтобы делать это было проще (рисунок 18.24).

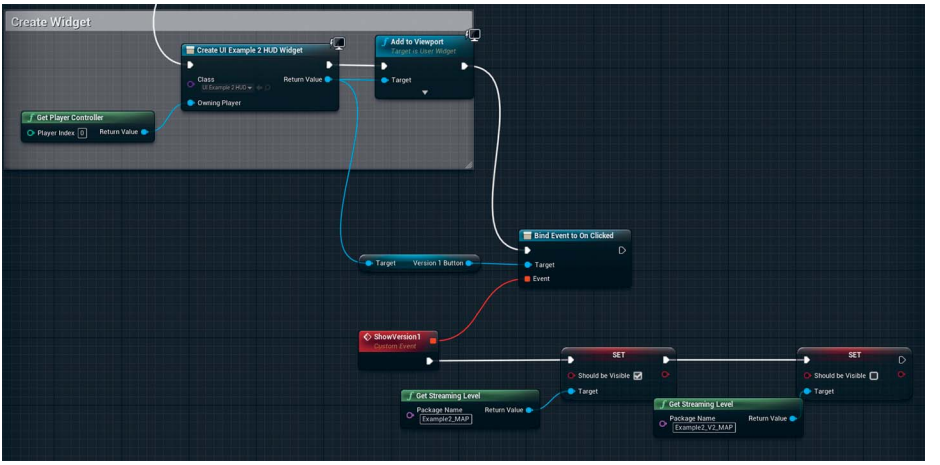


Рисунок 18.24 Привяжите ваш ранее созданный **ShowVersion1** Event к **OnClicked** Event для **Version1Button** Widget

Теперь, конечно, повторите для другой кнопки, получив ссылку из узла Create Widget и назначив ваш другой Custom Event для OnClicked Event этой кнопки (рисунок 18.25).

Теперь нажатие кнопок вызывает Events, переключающие уровни.

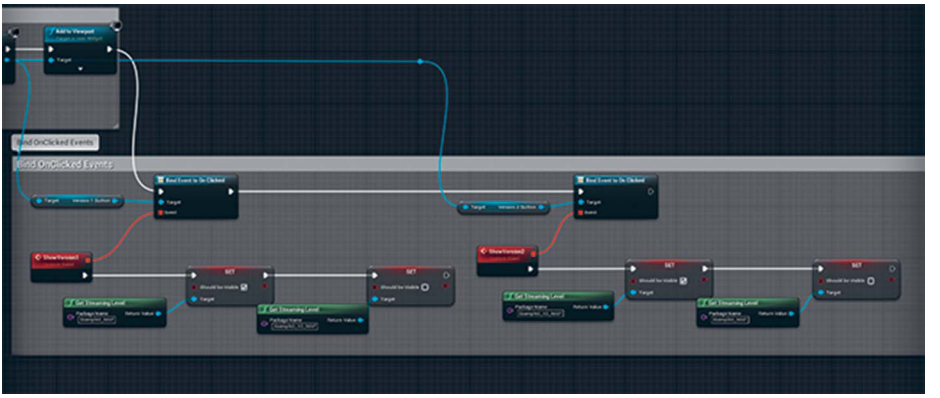


Рисунок 18.25 Обе кнопки связаны с переключающими Events в Persistent Level Blueprint

Компиляция и сохранение

Скомпилировав ваш Level Blueprint, убедитесь, что ошибок нет, и сохраните вашу работу.

Запуск игры

Теперь вы можете нажать Play на уровне и испытать полностью собственный, управляемый мышью пользовательский интерфейс. Вы должны быть способны перемещаться с помощью клавиатуры и мыши, переключаться между двумя версиями с помощью UI (рисунок 18.26).

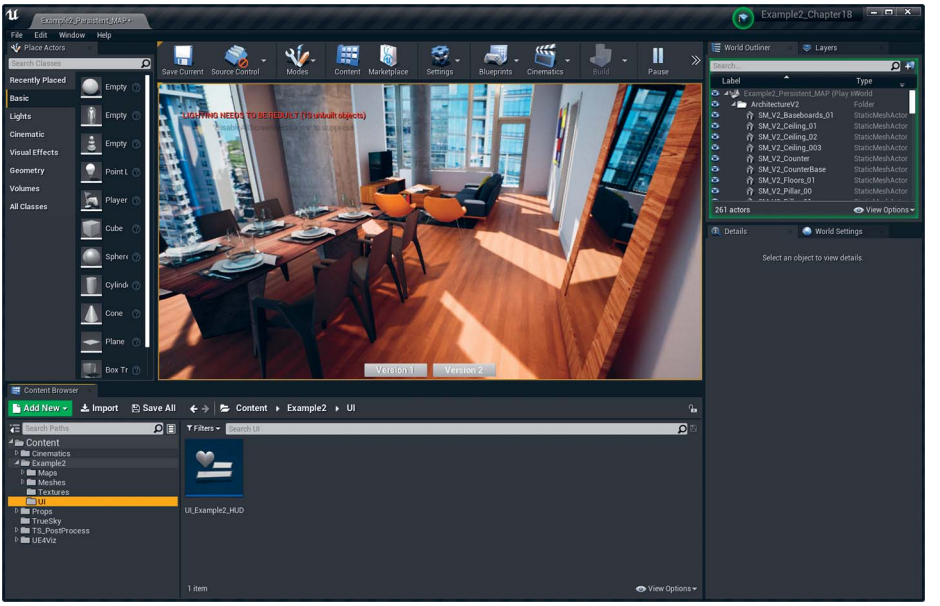


Рисунок 18.26 Запущенная игра в PIE, с полностью функциональными кнопками и мгновенным переключением

Заключение

Вы проделали огромный путь и узнали много нового в этой главе.

Вы изучили систему Level Streaming в UE4 и использовали ее для мгновенного переключения целых уровней в процессе работы. Это отличный способ для смены наборов данных в приложениях для визуализации, который прост в настройке и поддержке.

Вы также рассмотрели, как Blueprints могут взаимодействовать друг с другом с помощью привязки события и использования этой способности для построения простого, но эффективного пользовательского интерфейса с UMG.

UMG является отличным ресурсом и одним из лучших доступных инструментов создания пользовательского интерфейса. В сочетании с гибкостью и мощностью UE4,

вы в силах разработать огромное количество приложений, которые только можете себе представить.

Хотя эта игра является лишь простым примером, UMG использовался для таких успешных проектов, как полномасштабная AAA-видеоигра, блокбастер с простым инструментом сравнения A/B, как этот. Его производительность, гибкость и простота в использовании — одна из причин, по которой UE4 становится доминирующей силой практически в каждой визуальной индустрии.

ГЛАВА 19

ДОПОЛНИТЕЛЬНЫЙ УРОВЕНЬ BLUEPRINTS: ПЕРЕКЛЮЧАТЕЛЬ МАТЕРИАЛОВ

Создание готовых Blueprints необходимо для успешной работы с UE4. Хорошие системы Blueprint создают новую функциональность для приложения и построены как инструмент, который прост в использовании для вас или вашей команды. В этой главе вы узнаете, как создать один Actor Blueprint, который позволит игроку нажимать на любой меш на сцене и переключаться на следующий определенный материал в списке. Рассматривая этот Blueprint, вы сможете заметить, как Blueprints могут взаимодействовать друг с другом и менять свойства на лету.

Теперь, когда вы увидели, как создать приложение для интерактивной визуализации, создать Player Controller и Pawn, переключение между уровнями и как разрабатывается пользовательский интерфейс в UMG, настало время сделать следующий шаг — ну ладно, может быть, короткий лестничный пролет.

Эта глава предназначена для людей, у которых есть опыт программирования в UE4 в целом, или для тех, кто хочет увидеть, как создается сложная система. В любом случае, вы должны иметь четкое представление обо всех темах, затронутых ранее в этой книге, прежде чем пытаться их воссоздать или следовать этой главе.

Эта глава демонстрирует наиболее важную часть этой системы и дает рекомендации о том, как выполнить некоторые обычные программные паттерны UE4 и Blueprints.

Конечно, вы можете скачать файлы для проекта этой главы на www.TomShannon3D.com/UnrealForViz, ну или анализировать, отслеживать или копировать и вставлять в собственные проекты. (Я не против, правда.)

Настройка цели

Цель — создать Blueprint Actor Class, который:

- позволит левел-дизайнеру (LD) разместить Blueprint на уровне, используя редактор;
- определит список сцен с Static Mesh Actors, которые меняют материалы при нажатии;
- позволит LD определять список материалов, которые циклически повторяются, когда игрок кликает на один из мешей;
- выделит перечисленных Actors, когда курсор игрока находится над Actor, демонстрируя, что он может быть изменен.

Я могу придумать пять способов заставить систему работать так же. В программировании редко бывают идеальные решения. Каждый программист подходит к проблеме по-своему.

Для этого проекта мы создадим один Actor Blueprint, содержащий несколько переменных для хранения списка мешей и материалов. Этот Blueprint четко покажет LD, что настраивается с помощью визуальных подсказок, что сделает настройку простой и надежной, а также облегчит процесс отладки (рисунок 19.1).

Мы также перенесем часть контента с Marketplace в наш проект и модифицируем его, для размещения объектов с помощью Post-process Domain Material, который будет переключен с использованием Blueprint.

Рисунок 19.1 Material switcher Blueprint, расположенный на Level, показывает LD-friendly design

Создание Actor Blueprint

Вы можете использовать один Blueprint для добавления всего функционала, который нужен для создания простой в использовании системы для игрока и дизайнера, которому поручено настроить уровень (**Level Designer** или LD).

Actor Blueprint — это то, что вы можете разместить на уровне и у чего есть 3D-transform, который вы можете изменить. Вы можете прикрепить Components к Blueprint Actors для расширения их возможностей. Components включают в себя Meshes, Particle Effects, Sound Cues и многое другое.

Actor Blueprints также могут сделать что-то особенное. Они могут ссылаться на других акторов уровня. Это важно, потому что вам нужен этот Blueprint для взаимодействия с различными статическими мешами на уровне, чтобы менять их.

Создайте Blueprint в Content Browser. Когда появятся опции выбора Class, от которого вы наследуете ваш новый Blueprint, выберите Actor Class (рисунок 19.2).

Я назвал свой Blueprint BP_MaterialSwitcher_Actor и сохранил его в месте с моими PC, Game Mode и Pawn в папке Blueprints.

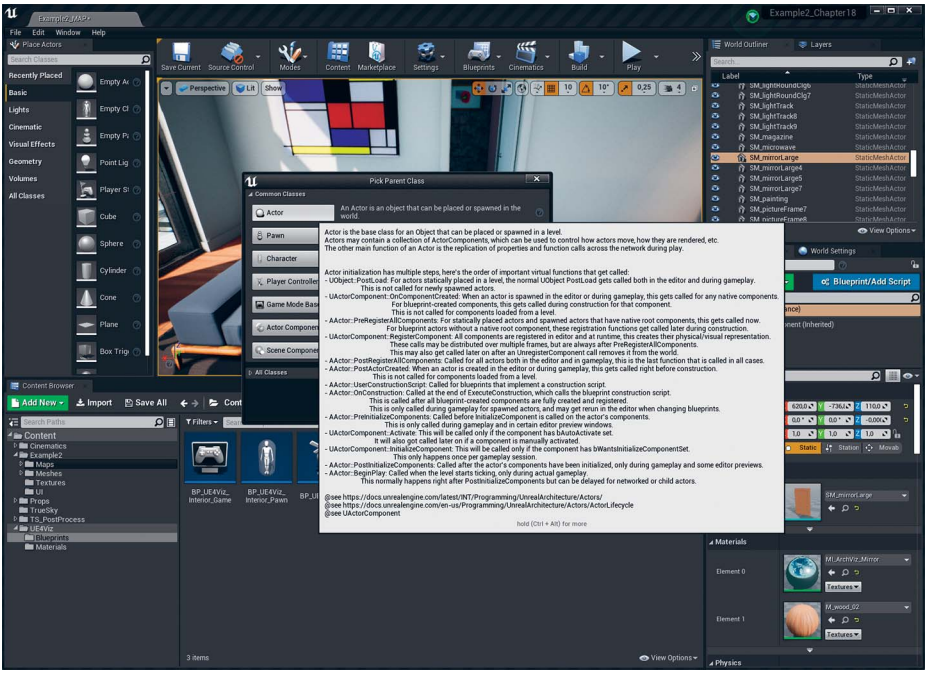


Рисунок 19.2 Создание нового Actor Blueprint из Content Browser

Создание Variables

Вам нужно хранить некоторую информацию или данные в Blueprint Class. Для сохранения данных в Blueprint используйте переменную. Переменные бывают множества разных типов (Classes), включая простые, с которыми вы, скорее всего, знакомы, такие как булевые (boolean) или число с плавающей точкой (float), а также любых других классов, включенных в ваш проект.

Эта возможность использовать классы и акторы в качестве переменных позволяет вам ссылаться на Static Mesh Actors, которые вы хотите изменить. Вы также можете ссылаться на Assets, хранящиеся в Content Browser. Это позволит вам определять различные материалы, которые будут применены к статическим моделям. Для работы Switcher Blueprint требуется всего несколько переменных (рисунок 19.3).

Вы создаете переменные в Blueprint Editor с помощью кнопки Add New на панели My Blueprint или нажатием на иконку + рядом с разделом Variables на этой панели (рисунок 19.3).

Когда вы создаете переменную, нужно назвать ее и присвоить тип. Вы можете переименовать и присвоить тип позже в панели My Blueprint или в панели Details переменной.

Вы можете не только определить **тип** (float, string, vector и т. д.) переменной, вы также можете определить ее как **массив**. Массив переменных становится списком любого типа, на который она установлена. Вы можете редактировать или менять этот список в Details актора, если включен параметр переменной **Allow Editing**.

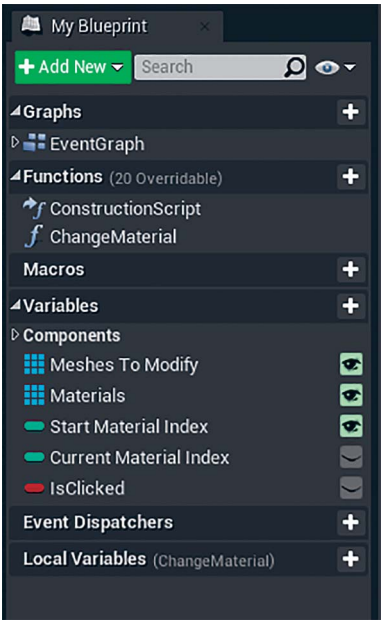


Рисунок 19.3 Переключатель материалов переменных и функций Blueprint

Каждый раз, когда вы создаете новую переменную, у нее по умолчанию будет тип и настройки массива, которые вы установили для последней созданной переменной.

Meshes To Modify Array

The Meshes To Modify Variable — это массив Static Mesh Actors (Static Meshes, который размещен на уровне, а не ссылка на Static Mesh Asset из Content Browser), который должен менять материал как группу при нажатии. Это позволяет LD определить несколько предметов мебели (например, все стулья для столовой) в массиве, и при нажатии на один все они совместно изменят материал.

Массив Variable может содержать список Level Actors, к которому обращаются узлы Event Graph nodes (рисунок 19.4).

После создания, присвоения имени и назначения типа переменной нажмите на значок Grid рядом с ним, чтобы конвертировать в массив. Обратите внимание, что при нажатии на Variable Details Panel заполняется свойствами этой переменной.

Если для переменной установлено **Editable,** то она отобразится в панели Details в Level Actors этого класса. Это позволяет каждому экземпляру уровня содержать различный набор мешей, назначенных массиву. Вы можете увидеть на рисунке 19.4, что свойство Meshes To Modify отображается в панели Details.

Expose on Spawn является специальным параметром, который позволяет вам легко установить переменную, если вы программно спавните этого актора в процессе исполнения. Так как вы не будете этого делать, ни у одной из ваших переменных не должно быть установлено это.

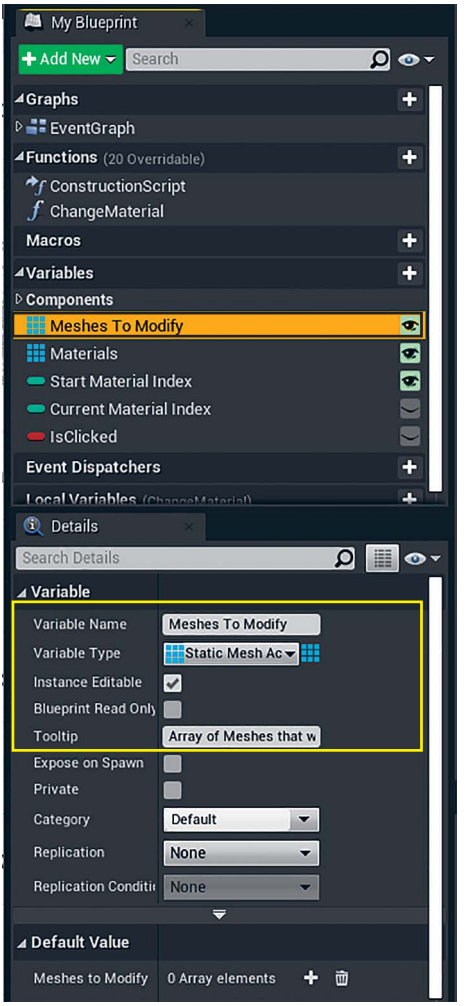


Рисунок 19.4 Свойства Meshes To Modify Variable

Массив материалов

Materials Variable — это массив ссылок на **Material Interface**. Тип Material Interface для переменной позволяет использовать как обычные материалы, так и объекты материалов.

В отличие от Meshes to Modify Variable, которые ссылаются на Static Mesh Actors на уровне, массив Materials Variable ссылается на Assets в Content Browser. Это позволяет системе считывать из библиотеки материалов в вашем проекте, даже если материалы не были назначены для чего-нибудь на сцене.

Установите Editable для этой переменной, позволяющий LD изменять эти значения для каждого актора в пределах каждого уровня.

Переменная Start Material Index

Свойство переменной **Start Material Index** является типом **Integer**. Integers — это все целые числа, такие как 1, 2 и 8675309. Они не могут иметь дробной части, но могут быть отрицательными. В этом случае вы можете использовать их для определения, какой материал из массива материалов выбрать.

Индекс — это термин, который относится к числовой позиции значения в массиве. В UE4 индекс начинает отсчет с 0, поэтому индекс 0 является первым элементом в массиве, а индекс 3 — четвертым. Это немного сбивает с толку, и сбило меня с толку больше, чем я думал.

В этом случае вы будете использовать его, чтобы определить, какой материал применить для вашего **Meshes to Modify** Actors из массива **Materials**.

Также установите для него **Editable** (рисунок 19.5), так вы можете решить, как создатель вашего уровня, с какого индекса начинать и какой материал применять по умолчанию.

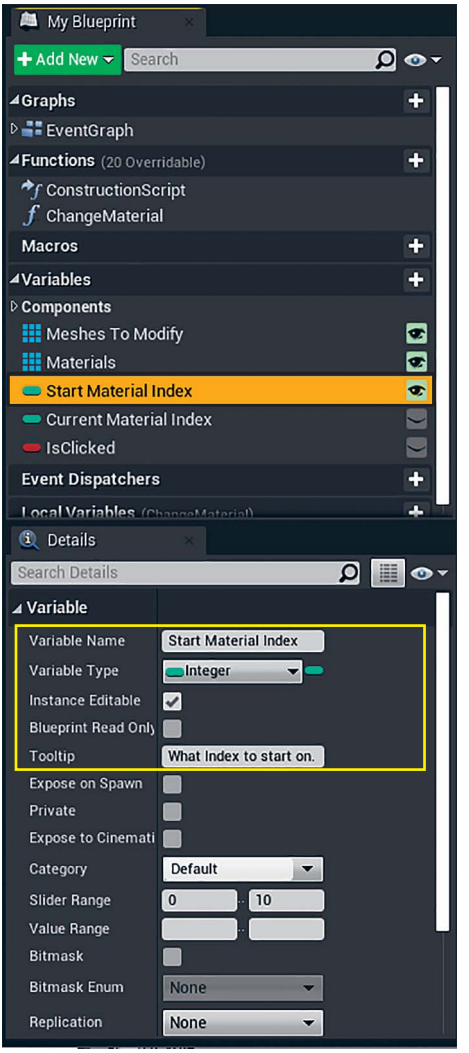


Рисунок 19.5 Свойства Start Material Index Variable, установленные как Integer (целые числа) и Editable

Current Material Index Variable

Установите для **Current Material Index** другой Integer точно так же, как вы делали в Start Material Index, но отключите свойство Editable, потому что Current Material Index используется только кодом Blueprint в процессе исполнения и никогда не будет напрямую устанавливаться пользователем. Вы станете применять эту Variable для отслеживания пользовательских нажатий, увеличивая его на единицу каждый раз при нажатии пользователя.

IsClicked Variable

Свойство **IsClicked** переменной имеет тип Boolean. Boolean определен только двумя возможными значениями: true или false. Как и Current Material Index, это не редактируемая переменная (свойство Editable отключено), так как код использует ее во время выполнения для определения, является ли клик игрока по Actor намеренным или случайным нажатием.

Добавление компонентов

Components похожи на sub-Actors для ваших Blueprints. Это C++ и Blueprint-классы, которые могут содержать все, что может другой Blueprint-класс, включая код, эффекты и ввод.

Есть два способа создать Components в Blueprints — программно во время исполнения или вручную, используя Components и окна Viewport в Blueprint Editor (рисунок 19.6).

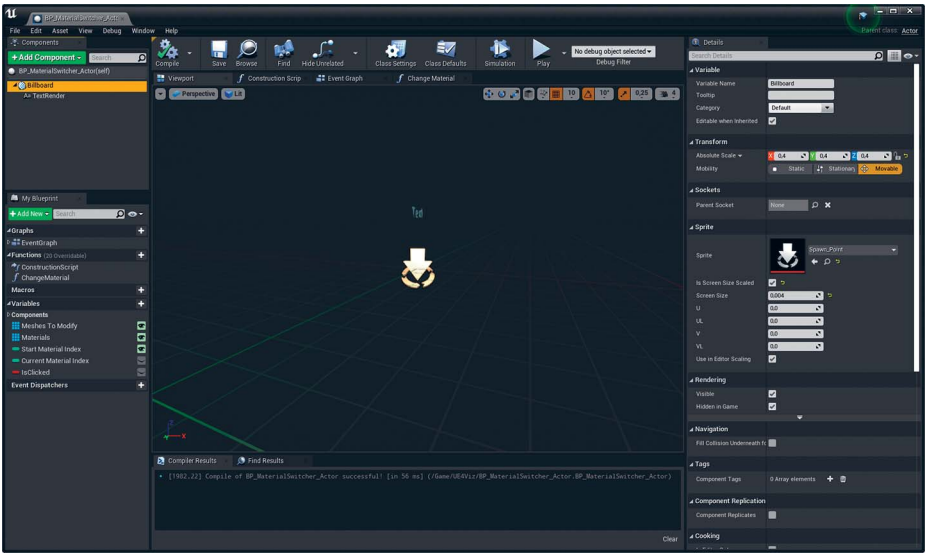


Рисунок 19.6 Детали Billboard Component

Billboard Component

Billboard Component выступает в качестве основы вашего Actor. Billboard Class отображает Texture, которую всегда видит игрок. Вы можете использовать это для таких вещей, как вспышка объектива или спецэффектов. В этом случае у вас есть что-то, что LD может увидеть и выбрать в уровне.

Для создания компонентов используйте раскрывающийся список Add Component и выберите тип, который вам нужен, в данном случае Billboard Component.

Для замены DefaultSceneRoot Component, который установлен по умолчанию, перетащите только что созданный Billboard Component в DefaultSceneRoot Component, заменив его.

Вы также можете определить собственную **Sprite** Texture для отображения, как показано на рисунке 19.6. Показанный Spawn_Point из содержимого движка поставляется вместе с UE4[20]. Вы также можете создать и импортировать вашу собственную текстуру для использования здесь.

Параметр **Is Screen Size Scaled,** установленный на true, гарантирует, что ваш спрайт никогда не будет превосходить определенного размера экрана. Это полезно, чтобы экран не заполнялся огромными спрайтами, так как Viewport camera приближается к ним.

TextRender Component

TextRender Component удобен — он отображает текстовые 2D-строки в 3D-пространстве, что может быть изменено на лету (рисунок 19.7). Я использую его для отображения названия первого выбранного меша в массиве Meshes To Modify. Это только помогает LD при создании уровня.

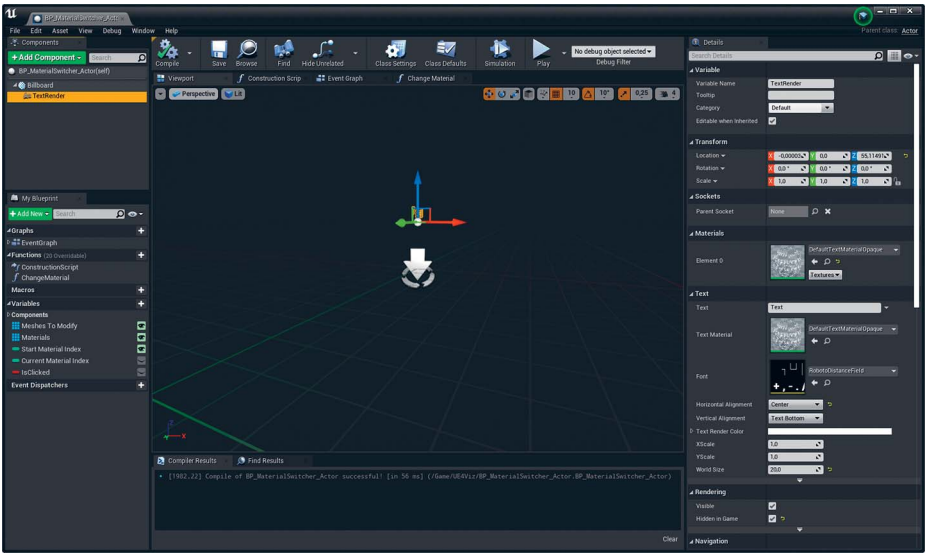


Рисунок 19.7 Детали TextRender Component

[20] Вы можете получить доступ к контенту, который входит в состав Engine, выбрав значок Eye в Content Browser, а затем — Show Engine Content. Будьте осторожны, чтобы ничего не изменить в содержимом вашего Engine, так как это может привести к проблемам со стабильностью и обмену контентом между членами команды.

Создание функции Change Material

Функция **Change Material** (рисунок 19.8) действительно является ядром всего этого Blueprint. Она обрабатывает работу по назначению материалов для назначенных Static Meshes. Также она содержит некоторую логику для избегания ошибок и может гарантировать, что материалы всегда назначены.

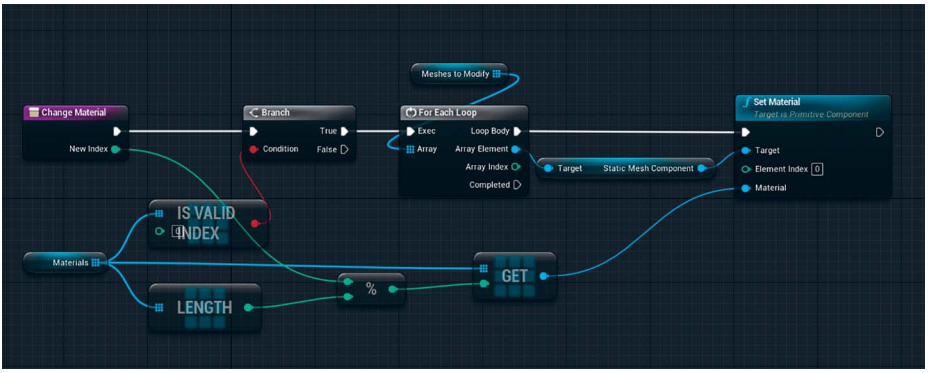


Рисунок 19.8 Изменение Material Function

Функции используются для инкапсуляции определенного функционала, особенно когда вы хотите его повторно использовать. В данном случае вам нужно изменить материал в процессе Construction Script, после Begin Play Event в Event Graph и во время выполнения, когда игрок нажимает на перечисленный меш.

Вы создаете функции аналогично тому, как создаете переменные, используя панель My Blueprint.

Нажмите на кнопку Add New и выберите Function из выпадающего меню. Назовите ее Change Material.

После создания функции она сразу же откроется для редактирования в пустом Event Graph. Чтобы открыть уже созданные функции, дважды нажмите на них для входа в панель My Blueprints.

New Index Input

Функция может содержать входные и выходные параметры, которые позволяют обрабатывать и возвращать данные. Входные и выходные параметры даже могут быть разных типов данных. Простой функцией, которая делает это, является функция **Get** массива. Вы предоставляете ей целочисленный индекс, и она вернет что угодно, что массив определил как Static Mesh или Material.

Для добавления входных параметров нажмите на узел функции в Graph Editor; это отобразит свойства функции, включающие область для добавления входных и выходных параметров.

Нажмите на кнопку + в Details Panel рядом с Inputs для добавления нового входного параметра. Назовите его New Index и установите ему тип Integer.

Этот входной параметр будет определять, какой индекс массива Materials использовать. Это позволит функции стать многофункциональной. Вместо того чтобы писать новую функцию для каждого номера материала, вы можете просто использовать эту переменную для определения желаемого индекса.

Is Valid Index

Проверка Is Valid Index выполняется в массиве материалов перед продолжением, чтобы удостовериться, что ваш массив валидный и что вы не обращаетесь к пустой записи массива.

Цикл For Each

Если Is Valid прошла, тогда функция проходит через каждый меш в массиве **Meshes To Modify** и вызывает для него функцию Set Material.

Чтобы определить, какой материал применить, вы должны получить ссылку на него из массива материалов с помощью предоставленного значения New Index.

Узел Modulo (%) используется, чтобы гарантировать, что любое целое число будет в пределах доступного диапазона в массиве Materials. Он делает это, возвращая остаток от деления A на B — к примеру, 1 mod 4 = 1, 4 mod 4 = 0, 5 mod 4 = 1 и 55 mod 4 = 3.

Понимание Construction Script

Два случая, когда код Blueprint запускается во время конструирования акторов, **Construction Script** и в процессе исполнения **Event Graph**.

Construction Script запускается только при появлении класса впервые в мире. Это может случиться либо когда вы размещаете или меняете актора в редакторе, либо когда актор или объект появятся в процессе выполнения. Construction Script — это место, где вы можете запрограммировать Blueprint до того, как игра запустится. Это включает появление компонентов, изменение других акторов и так далее.

Акторы, размещенные на уровне (в отличие от появляющихся во время выполнения), запускают только их Construction Script в Editor, и результаты сохраняются в файл, когда вы сохраняете уровень. Construction Script больше не запустится, даже если перегрузить уровень.

Construction Script в нашем классе делает три основных действия: устанавливает text label, рисует линии от актора до мешей, которые будет изменять, и устанавливает материалы для меша в соответствии со Start Index Variable (рисунок 19.9).

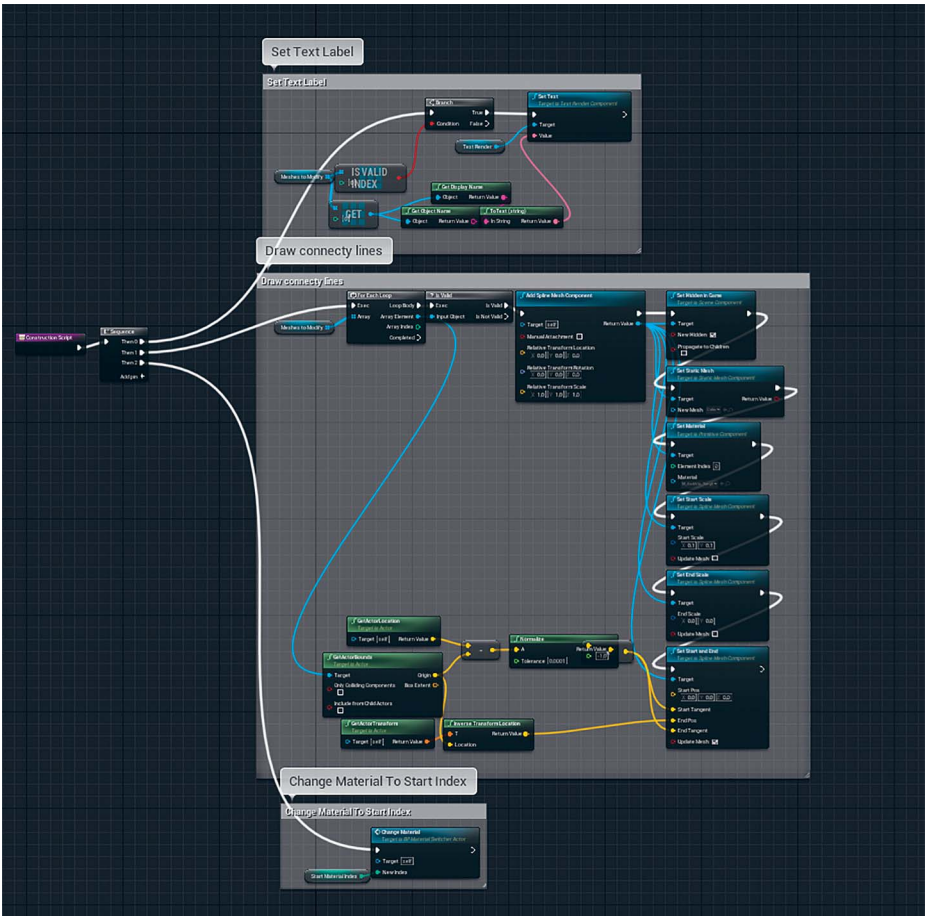


Рисунок 19.9 Окончательный Construction Script с тремя основными функциями, выполняемыми Construction Script

Установка Text Label

Construction Script сначала пытается установить в TextRender свойство Text на основе выбранных мешей (рисунок 19.10). Если Meshes не выбраны, он определяет как None, помогая LD понять, правильно ли настроены его системы.

Чтобы сделать это, Construction Script должен получить ссылку на первый элемент массива Meshes To Modify, для этого получите имя с помощью Get Display Name и соедините его с Set Text Function с текстом в Render Component.

Как и множество узлов Blueprint, которые ссылаются на определенный класс, вам нужно сначала получить ссылку на ваш Text Render Component и затем перетащить соединение из него в Editor, чтобы увидеть контекстный список узлов, доступных для этого класса.

Вершина Branch требуется, чтобы гарантировать, что вы не вызовете **Get** для пустого массива (это плохо, и могут быть сбои), таким образом, исполнение продолжится, только если первая запись в массиве Meshes To Modify (Index 0) действительна.

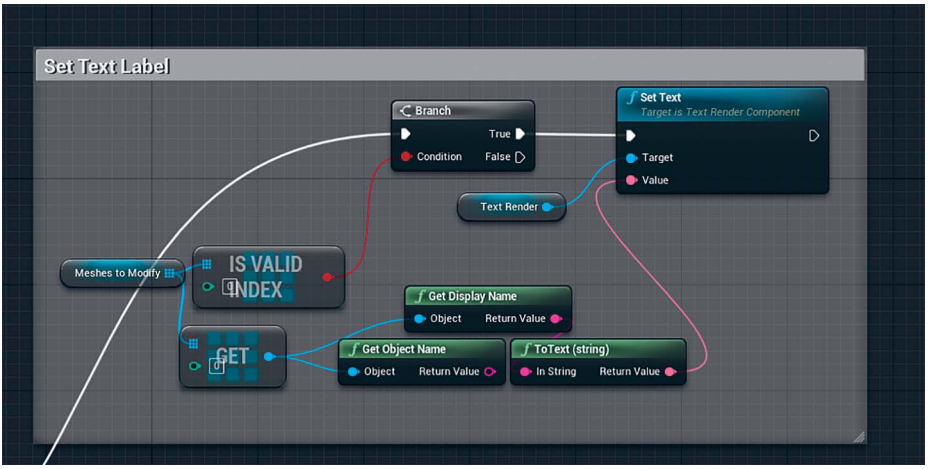


Рисунок 19.10 Детали TextRender Component

Рисование коннекторов

При разработке Blueprints для использования в производстве вы должны учитывать не только как игрок взаимодействует с Blueprints, но еще и как человек, настраивающий уровень, должен взаимодействовать с ним. Чрезмерно сложные или трудные в использовании инструменты не востребованы, и время, затраченное на их создание, теряется впустую, поэтому наведение лоска функциональности в редакторе может гарантировать, что практически каждому будет легко использовать.

Чтобы дать возможность LD способы визуализации того, как Static Mesh Actors ссылается на каждый Material Switcher Actor, их расположили в специально выделенной планке для каждого перечисленного меша из массива Meshes to Modify (рисунок 19.11).

Так как вы наверняка хотите нарисовать линию для каждого меша в вашем массиве Meshes To Modify, вам нужно перебрать массив, используя цикл For Each.

Каждый цикл сначала проверяет, возвращает ли допустимый **Array Element** (возможно, в вашем массиве могут быть пустые записи). Если допустимый, то используется узел **Add Spline Mesh Component** для добавления нового Spline Mesh Component в ваш Blueprint.

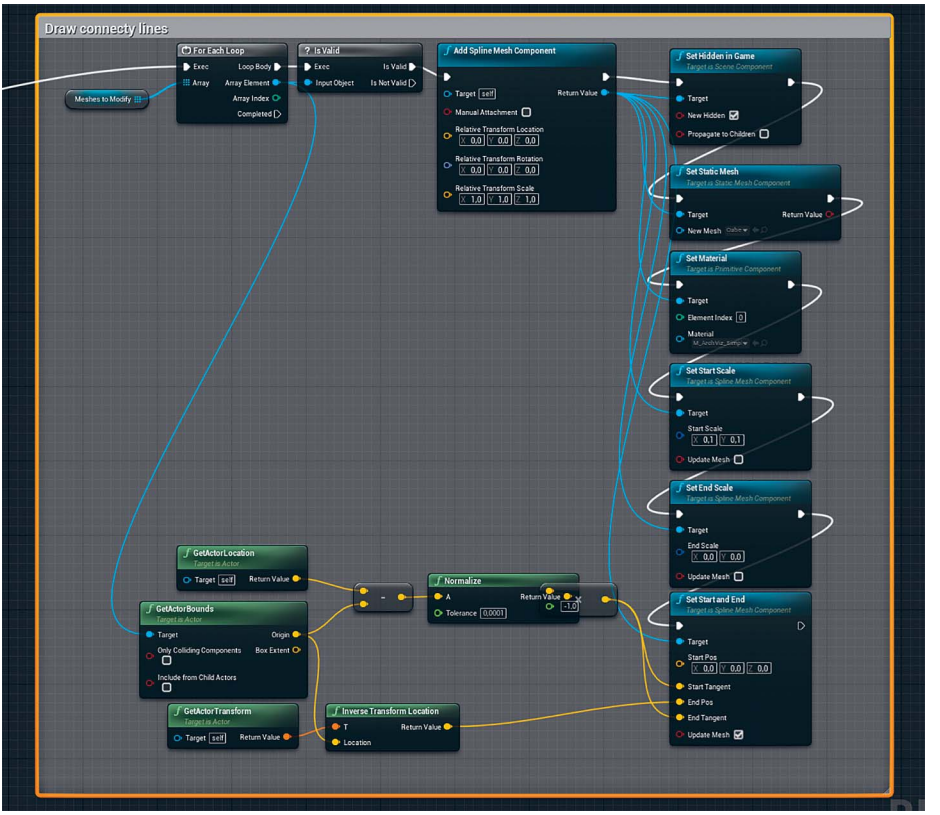


Рисунок 19.11 Рисование 3D-линий для визуализации назначенных Meshes

Затем каждая линия оформляется, так чтобы она указывала от Switcher Blueprint на каждый ссылочный объект Mesh.

Set Hidden in Game позволяет Component быть видимым в редакторе, но, когда вы переходите в Game Mode (с помощью симуляции, нажатия на Play или нажатия G для предпросмотра Game Mode), он будет скрыт. Множество Editor-only-акторов используют этот параметр, например значки и стрелки в Light Actors.

Затем Cube Mesh Static Mesh Asset присваивает Spline Mesh Component вместе с материалом, используя **Set Static Mesh** и **Set Material** Functions.

Начальный и конечный масштаб устанавливается, создавая форму, подобную стрелке, используя **Set Start Scale** и **Set End Scale**.

Наконец, используя функцию **Set Start and End**, чтобы установить начальные и конечные позиции сплайна относительно мировой позиции Blueprint Actor. Так как эти местоположения относятся к позиции Blueprint в мире, значение 0,0,0 представляет место точки опоры вашего Blueprint.

Мировая позиция Static Mesh Actor запрашивается с помощью функции Get Actor Bounds. Затем она преобразуется в локальные с помощью узла Inverse Transform Positon и передается в свойства End узла Set Start and End.

Сплайн в этой точке будет выглядеть очень странно, так как сначала вы должны определить направление касательной для каждой точки на сплайн. Поскольку вы наверняка хотите, чтобы касательные просто шли вниз по направлению сплайна, вы вычисляете касательную между ними, вычитая их позиции и затем нормализуя результат для получения Unit vector, который можете соединить с двумя свойствами касательной Start and End.

Использование Context Sensitive Checkbox

Помните, что для доступа к этим функциям вам нужно перетащить соединение из **Return Value** в **Add Spline Component** для получения доступных для Class-specific методов. Вы также можете не проверять флажок Context Sensitive во всплывающем окне. Это перечислит все доступные методы, которые вы можете расположить в Blueprint; однако вы можете запутаться, так как вам будет предложено гораздо больше вариантов, чем если бы вы использовали контекстно отсортированный список.

Техника перетаскивания соединений полезна для всех видов узлов. Например, чтобы легко найти математический узел Vector * vector, вы можете перетащить желтое соединение из выхода vector и напечатать *, чтобы получить список всех возможных функций умножения, которые может использовать vector.

Замена материала по Start Material Index

Чтобы разрешить переключателю материалов (Material Switcher) работать в редакторе, вы можете вызвать Change Material Function, которую вы создали в Construction Script (рисунок 19.12). Теперь, когда LD устанавливает Start Material Index Variable, он увидит соответствующее изменение материала в редакторе.

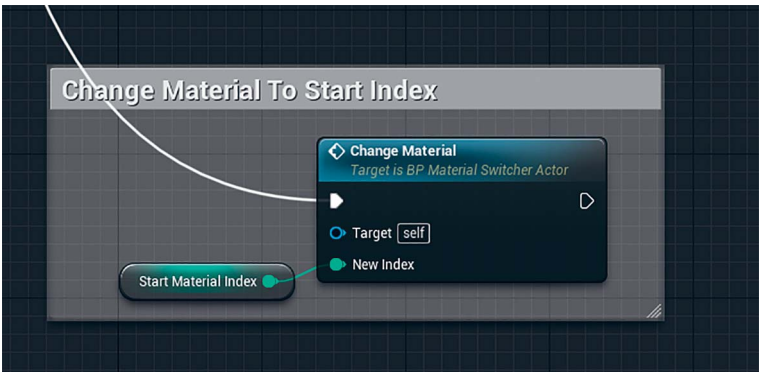


Рисунок 19.12 Вызов функции Change Material в Construction Script

Добавление функции Change Material в Construction Script также является удобным способом протестировать вашу функцию, поскольку вы можете вручную установить Index и увидеть, действительно ли она изменит материалы, как предполагается.

Понимание Event Graph

Граф событий (Event Graph) является частью Blueprint, которая запускается во время игры, в котором у вас есть функции, события и узлы, отвечающие за разные игровые события, такие как Tick и Input Events. В данном Blueprint, Event Graph обрабатывает выделение объектов Actors и меняет материалы на них, если игрок нажимает на актора (рисунок 19.13).

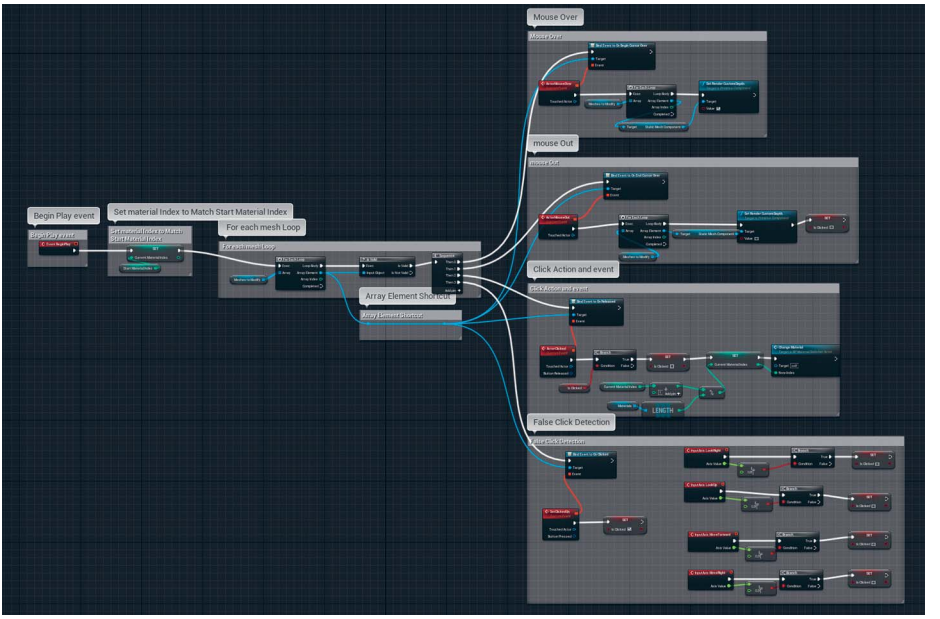


Рисунок 19.13 Обзор Event Graph с показом главных Functions

Прежде чем двигаться дальше, убедитесь, что вы работаете во вкладке Event Graph в вашем Blueprint Editor.

Основное назначение Event Graph в классе Material Switcher Blueprint — настроить все Meshes To Modify Actors для получения Events ввода мыши. Для этого код циклом проходит через каждый меш в массиве Meshes To Modify и использует Event Binding для сообщения этим мешам, как вести себя, когда на них нажимает игрок.

Событие Begin Play

Код Event Graph в этом Blueprint начинается с события Begin Play. Это событие вызывается только один раз, когда Actor или Object впервые появляются в игре в процессе исполнения.

Первая часть Graph из узла Event Begin Play, скорее всего, теперь понятна. Для каждого Static Mesh Actor в Meshes To Modify циклом проходит каждый меш в массиве, выполняя проверку на доступность, чтобы избежать обращения к null указателя. Если он доступен, то функционал доступен различным событиям взаимодействия мыши.

Привязка события

Вы уже знаете из предыдущей главы, что Blueprints можно связать с событиями, которые происходят в других классах вашей игры, и это очень полезно для сохранения простоты и краткости ваших Blueprints. Вам не надо добавлять код в каждый объект Mesh, в котором вы хотите зарегистрировать события мыши, — просто используйте функционал Bind to Event.

Как и прежде, вы должны перетянуть соединение из красного/оранжевого Bind To Function's поля делегата в Graph Editor, отпустить и выбрать **Add Custom Event**. Это создаст новый Custom Event, который содержит Inputs и другие параметры, правильно настроенные для этого Event. В данном случае вы можете видеть, что Custom Event возвращает ссылку на Touched Actor, который добавлен автоматически.

Mouse Over и Mouse Out

Подобно множеству приложений, которые содержат управляемые мышью интерфейсы, UE4 предлагает Events, когда курсор мыши начинает и заканчивает висеть на объекте Actor.

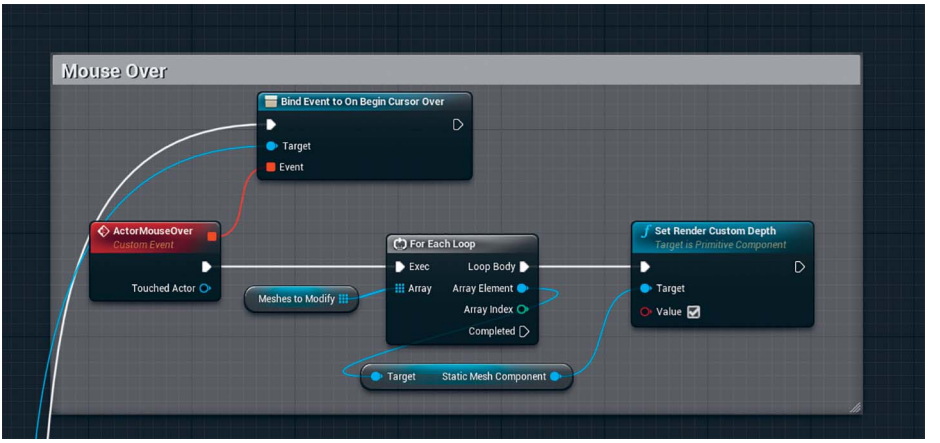


Рисунок 19.14 Граф события Mouse Over Event

В данном случае Event используется для установки Custom Depth на каждый меш в массиве Meshes To Modify, что позволяет post-process Material обнаруживать их. Вы можете заметить на картинках (рисунок 19.14 и 19.15), что, как только игрок наводит курсор на акторов, код в цикле просто перебирает массив Meshes To Modify, устанавливает свойство Custom Depth для каждого меша на true и возвращает false, когда курсор покидает.

Custom Depth затем считывается Post-Process Material, и генерируется контур вокруг поверхности актора.

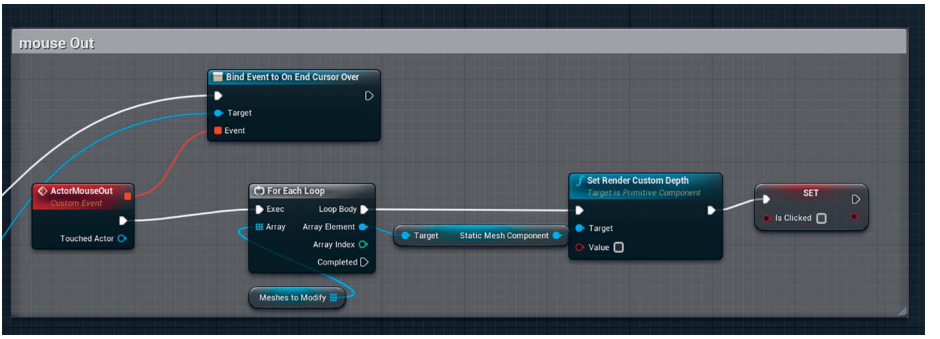


Рисунок 19.15 Mouse Out Event Graph

Узел, устанавливающий значение **Is Clicked** переменной на false. Это гарантирует, что, если курсор игрока когда-либо покинет периметр актора, даже с зажатой кнопкой он не вызовет изменение материала, когда игрок отпустит кнопку мыши.

Этот вид полировки делает разницу между плохой системой и той, которая ведет себя так, как ожидает игрок, заметной. Эта переменная добавлена после тестирования, которое показало, что игрок может случайно изменить материалы для мешей, когда отпускает кнопку мыши после обзора.

Click Action

Когда игрок нажимает на конкретный меш в массиве Meshes To Modify, нужно изменить материалы. Это делается с помощью увеличения Current Material Index Variable, снова используется оператор взятия по модулю (%) (рисунок 19.16), чтобы убедиться, что значение остается в пределах диапазона; в основном зацикливая целое число по определенному номеру.

Затем результат взятия по модулю отправляется в Change Material Function, которая, в свою очередь, меняет все меши в массиве Meshes To Modify.

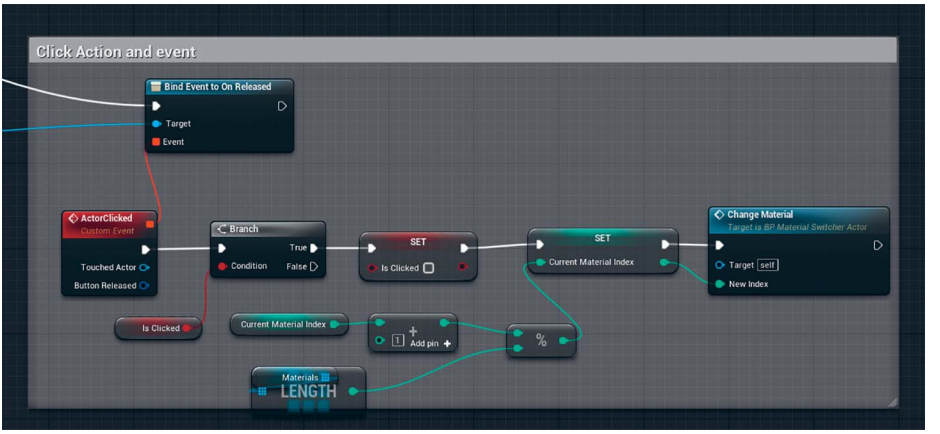


Рисунок 19.16 Click Action и Event

Определение False Clicks

Иногда игрок захочет повернуть камеру с помощью мышки и не нажимать на интерактивные меши, размещенные на уровне. Если вы позволяете менять стены и полы, могут существовать несколько мест, куда игрок может нажать без случайного переключения материала.

Для решения этой проблемы Blueprint обнаруживает входные данные, которые могут указывать игроку не только нажимать, но нажимать и перетаскивать, предположительно для поворота обзора (рисунок 19.17).

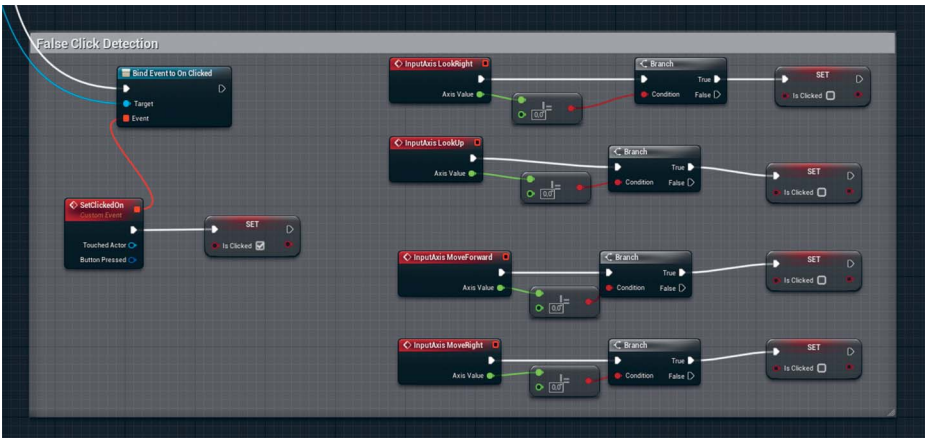


Рисунок 19.17 Обнаружение ложного нажатия

Это происходит, когда IsClicked Variable вступает в игру. Присвоение значения true этой переменной происходит, только когда игрок напрямую нажимает на меш, и после

этого присваивается false для любых других вводов мыши, избегая вызова функции **Change Material**.

Заполнение уровней

Как и все ассеты, которые вы хотите взять из Content Browser для уровня, лучше всего просто перетащить их из Content Browser в Viewport.

Начните с размещение одного из Material Switcher Blueprints рядом с Meshes, которые хотите изменить (рисунок 19.18). Обратите внимание на Details Panel справа, что Variables, которые помечены как Editable, видны и готовы для изменений LD.

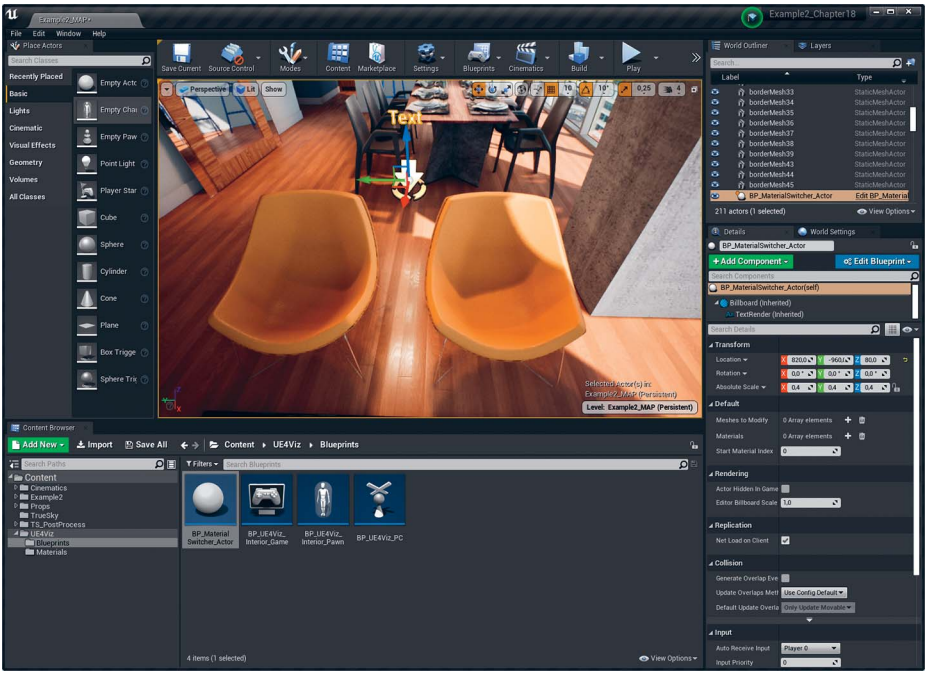


Рисунок 19.18 Material Switcher добавлен на Level рядом с парой стульев

Добавление мешей

Теперь определите меши, которыми хотите повлиять на Blueprint. В Details Panel, найдите массив Meshes To Modify и нажмите на кнопку «+» для добавления нового элемента в массив. Появится выпадающий список, показывающий каждый статический меш на вашем уровне. Это может быть трудный способ выбора мешей; попробуйте лучше возможность использования инструмента eyedropper (рисунок 19.19).

Как только вы добавите меши в список, запустится Construction Script, и вы увидите линию, проведенную между Blueprint и каждым мешем в списке (рисунок 19.20).

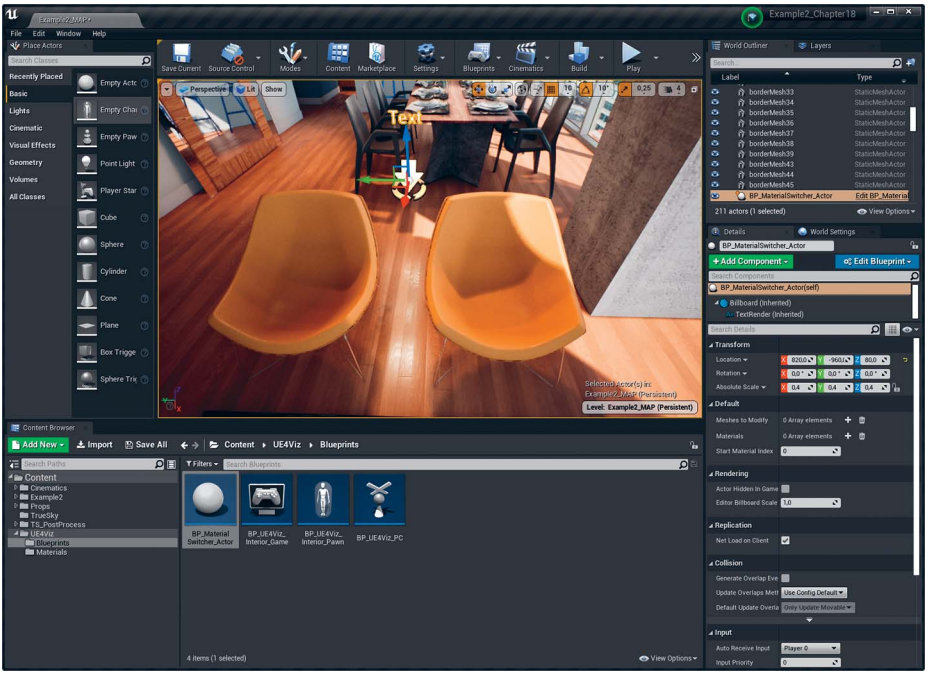


Рисунок 19.19 Использование инструмента Eyedropper для выбора Meshes

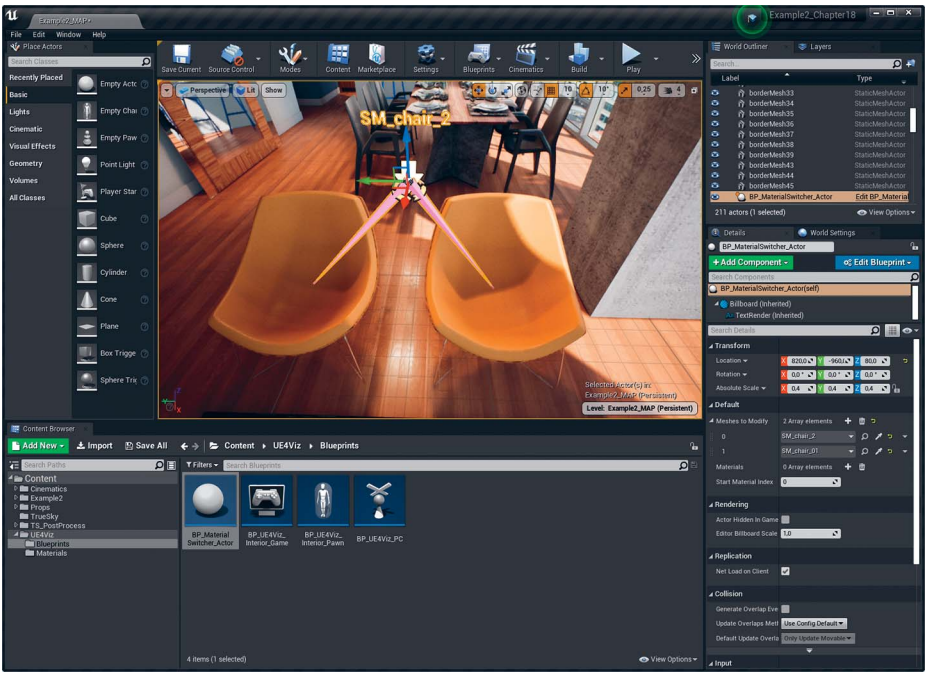


Рисунок 19.20 Добавленный Meshes в массив Meshes to Modify. Линии соединения и Text label также видны

Добавление материалов

После того как вы заполнили список Mesh, проделайте аналогичную настройку для материалов. Вместо выбора мешей из мира используйте eye dropper, как раньше, выберите ваш материал или объект материала из Content Browser, так как массив Materials скорее всего является ссылкой на Asset, нежели на Actor (рисунок 19.21).

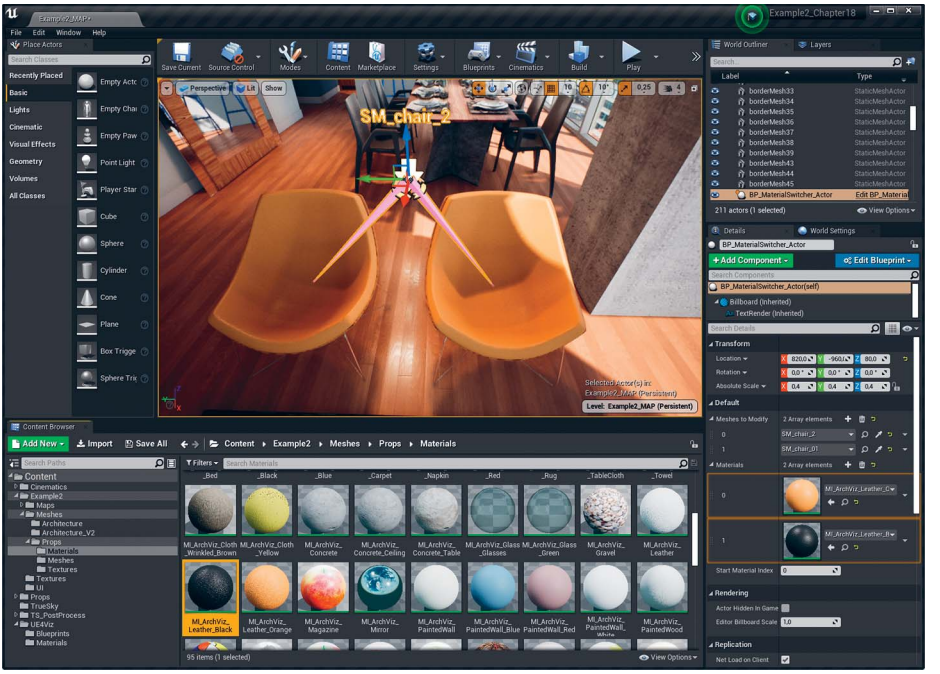


Рисунок 19.21 Полностью настроенный Blueprint с Materials, назначенными массиву Materials

Настройка параметров по умолчанию

Не так много параметров доступно другим в этом Blueprint, не считая массивы. Конечно, вы можете назначить переменную Start Material Index. Это легкий способ настроить внешний вид вашей области по умолчанию и способ предпросмотра Materials до нажатия Play.

Copying и Pasting

Отличная короткая дорожка для настройки акторов — возможность копировать и вставлять их как переменные. К примеру, вы можете выбрать запись массива Materials в Details Panel и выбрать Copy перед вставкой в то же поле свойства другого Actor.

Этот трюк не ограничивается Blueprints. Вы будете удивлены, как много вещей можно копировать и вставлять в UE4.

Запуск приложения

Все, что осталось сделать, — это протестировать, нажав кнопку Play. Вы должны иметь возможность перемещаться в пространстве, нажимать на различных акторов на уровне и менять их материалы, создавать персонализируемое внутреннее пространство (рисунок 19.22).

Рисунок 19.22 Использование Play в Editor для тестирования системы — успех!

После установки достаточного количества мешей и материалов вы можете резко сменить внешний вид сцены с помощью всего лишь нескольких нажатий (рисунки 19.23 и 19.24).

Рисунок 19.23 До нажатия

Рисунок 19.24 После нажатия

Заключение

Теперь у вас есть законченное приложение с полностью проработанным функционалом, которое можно отправить вашему клиенту, радуясь тому, что выполнили все свои обязанности. Вы увидели, как простой Blueprints может содержать мощные эффекты в игровом мире только с несколькими узлами.

Вы увидели два способа создания интерактивности: используя UMG и напрямую взаимодействуя с акторами на уровне.

Вы также увидели пару способов переключения данных UE4, оба с их собственными преимуществами и рабочими процессами. Я надеюсь, вы воодушевлены и готовы начать исследовать и создавать собственные методы для решения конкретных задач вашей индустрии, рабочего процесса и наборов данных.

ГЛАВА 20

ФИНАЛЬНЫЕ РАЗМЫШЛЕНИЯ

Эта книга только поверхностно затрагивает UE4. С доступом к исходному коду C++, огромному массиву плагинов и интеграций, а также лучшим разработчикам на Земле, которые улучшают его каждый день, UE4 является действительно мощной силой, способной изменить индустрию развлечений и индустрию визуализации. От визуализации до фильмов и спецэффектов, у него блестящее будущее.

UE4 продолжает меняться

Unreal Engine 4 развивается быстрее, чем любой другой программный пакет, который я когда-либо использовал. Эволюция редактора и инструментов с моего первого знакомства с UE4 Rocket Beta поразительна.

Постоянные улучшения, инновации и поддерживаемые рабочие процессы добавляются с такой регулярностью, что весомой частью моей работы стало просто поспевать за всем, что выходит.

Очень интересно работать с таким ярким сообществом и видеть, как каждый день тысячи людей и команд со всего мира делают невероятные вещи.

Я не могу предвидеть будущее, но я регулярно получаю доступ ко многим инструментам планирования Epic Games, смотрю бесчисленное множество видео и общаюсь непосредственно с разработчиками и сообществом, получая хорошее представление о том, каким сообщество и Epic хотят его видеть.

Будущее визуализации

У визуализации всегда было несколько основных проблем. Предоставление качественного продукта, который эффективно рассказывает истории и сообщает сложную информацию визуально, может стать чрезвычайно сложной задачей. Визуализация часто разрабатывается в немыслимо короткие сроки для сильно ограниченных бюджетов с удивительно неполными или меняющимися (или оба варианта) наборами данных.

UE4 поможет значительно облегчить это противостояние. Интерактивность позволяет игроку визуализировать данные на свой лад в собственном и впечатляющем стиле. Как и в игре игрок использует интерактивную визуализацию для создания своей уникальной истории и делает это со своей точки зрения.

Также благодаря сокращению времени на рендеринг изменение данных или запросы клиентов в последнюю минуту проще обрабатывать. Это приводит к улучшению, большей точности, более полезному продукту, делает клиента более счастливым, а работу студии — более продуктивной: меньше времени уходит на рендеринг и ожидание и больше — на созидание и улучшение.

Виртуальная реальность, интерактивность и настраиваемость с помощью Blueprints и C++ означает, что UE4 является наиболее эффективным средством визуализации, доступным сегодня, и сильно опережает конкурентов, которые еще, судя по всему, не пошли по этому пути.

Следующие шаги

Мы коснулись только самых основ того, что может сделать UE4 и что вы можете с ним делать. Показанные примеры также представляют только маленькую часть типов визуализации, создаваемых сегодня. Наборы данных и требования клиентов широко варьируются от одной индустрии к другой и даже от одного проекта к другому.

Я надеюсь, что заложил основы и теперь вы способны взять данные и свои представления и перенести их в UE4. Вы, несомненно, встретитесь с трудностями, с которыми я не сталкивался и о которых не думал и не писал на этих страницах. Когда это произойдет, вы сможете найти множество доступных ресурсов. Потому что многие люди используют и изучают UE4, так что вы никогда не останетесь один на один с трудностями. И Epic Games, и сообщество UE4 стремятся сделать UE4 лучшим движком для всех.

Поскольку вы все больше и больше будете использовать UE4, я рекомендую взглянуть на ваш процесс работы и подумать, как можно его адаптировать из UE4 в общий студийный подход. Подобно тому как объединяются подходы Голливуда и игр, вы можете извлечь много уроков из разработки игр и интерактивных визуализаций, которые можно применить к повседневной работе, что не только улучшит работу, но и упростит ее интеграцию в UE4.

Виртуальная реальность

Наиболее выдающейся и многообещающей технологией ближайшего будущего, которую активно продвигают, является виртуальная реальность. Почти каждая индустрия на Земле включает в себя VR. Это только начало жизни VR, но, как и UE4, она очень быстро развивается.

UE4 играет серьезную роль в разработке VR, так как начал поддерживать VR очень рано, и Epic Games продолжают продвигать VR-инновации.

Сейчас UE4 поставляется с режимом для Editor, который позволяет дизайнерам и художникам создавать VR-миры непосредственно в VR практически со всеми возможностями Desktop-версии — это удивительное достижение.

Эта функция не была просто введена и забыта. Она активно и быстро прошла путь от красивой технической демонстрации до действительно жизнеспособного метода работы.

Даже если вы не используете VR Editor, Epic проделала фантастическую работу, раскрывая все вводы, отслеживания и другие функции различных VR API и платформ, что делает разработку для VR быстрой.

Инструменты, визуальная составляющая и страсть Epic для VR в совокупности делают UE4 лучшей VR-платформой для разработки.

Создание фильмов

Голливуд проявил большой интерес к UE4. Создатели фильмов встречаются со схожими трудностями, такими как визуализация домов, но в гораздо больших масштабах. Эффективность имеет огромное значение, так что скорость рендеринга и набор инструментов UE4 делают его неоспоримо привлекательным.

Разработчики игр и создатели фильмов годами поддерживают друг друга, перенимая техники и технологии друг у друга для улучшения собственных продуктов.

UE4 представляет собой вершину развития всех этих интеграций. UE4 кажется знакомым создателям игр и фильмов. Это подход к сотрудничеству между создателями игр и фильмов, что в конце концов размывает границы между двумя областями.

Через несколько лет мы не увидим различий между множеством игр и фильмов и наши представления о них изменятся кардинально.

Создание контента

По мере развития UE4 разрабатывается все больше и больше инструментов для создания контента. В последнем обновлении Epic добавили full polygon modelling с subdivision surfaces (и это даже работает в режиме VR Editor), а также инструменты для рисования и скульптинга.

Также UE4 теперь поддерживает Render to Texture (RTT). Используя Material Editor, Particle Systems и любые другие инструменты, доступные в UE4, вы можете создать flipbook анимации, текстуры и другие эффекты так же, как и в других приложениях, таких как Substance Designer или Photoshop.

Замена всех нужных вам инструментов займет много времени для UE4, но лично я все меньше и меньше провожу времени в моих 3D-приложениях и все больше в UE4.

Спасибо

Под конец спасибо тебе, добрый читатель. Я взволнован существованием этой книги. Я пытался объединить свои более чем десятилетние увлечения игровыми технологиями и визуализацией, и это невероятно захватывающе — наблюдать, как индустрия визуализации начала использовать рендеринг в реальном времени, и в частности в UE4.

Я желаю вам всего наилучшего в ваших начинаниях и с нетерпением жду все ваши удивительные творения и инновации.

Официальное руководство по разработке визуализации на Unreal Engine 4.

Эта книга содержит сведения, необходимые любому профессионалу в области визуализации, чтобы раскрыть огромную мощь UE4. Том Шэннон – эксперт мирового класса по UE4 – не только разбирает основные понятия и компоненты Unreal Engine 4, но и обучает целостному процессу создания выдающихся шедевров визуализации, проводя пошаговый разбор реалистичных и подробно документированных учебных проектов.

- Изучите структуру UE4 и овладейте средой разработки
- Освойте профессиональный UE4-конвейер от исходных данных до самого приложения
- Научитесь различать подходы UE4 и традиционные методы визуализации и рендеринга
- Разработайте яркое, теплое освещение для архитектурной визуализации
- Создайте предварительную анимацию с помощью секвенсора
- Используйте визуальное программирование на Blueprints для создания сложных взаимодействий без единой строчки кода
- Научитесь работать с ограничениями UE4 и использовать его преимущества для достижения невероятных результатов
- Достигайте ошеломляющего реализма

ТОМ ШЭННОН профессионально разрабатывал игры, занимался визуализацией и интерактивными приложениями на протяжении более десяти лет. Он работал со студиями видеоигр, домами визуализации, архитектурными и инженерными компаниями, используя Unreal Engine 4.

Эта книга – просто идеальная возможность попасть в уникальный мир визуализации и дизайна с помощью технологии Unreal Engine. Написанная простым языком, она будет полезна как новичкам, так и тем, кто уже начал пробовать свои силы в этой сфере. Книга позволит раскрыть и понять основные преимущества технологии Unreal Engine, а также собрать свой первый проект!

Стас Старых,
основатель и генеральный директор
компании VR Professionals

ISBN 978-5-04-108632-9

9 785041 086329 >

БОМБОРА
ИЗДАТЕЛЬСТВО

БОМБОРА – лидер на рынке полезных и вдохновляющих книг.
Мы любим книги и создаем их, чтобы вы могли творить, открывать мир, пробовать новое, расти. Быть счастливыми. Быть на волне.

bomborabooks bombora.ru