УДК 004.4
ББК 32.973.26-018
М17

Книга подготовлена в ходе работы в рамках
Программы фундаментальных исследований Национального исследовательского университета «Высшая школа экономики» (НИУ ВШЭ)

Максименкова, Ольга Вениаминовна.

М17 Программирование в Unreal Engine 5 для начинающего игродела : основы визуального языка Blueprint / Ольга Максименкова, Никита Веселко. — Москва : Эксмо, 2023. — 320 с. — (Российский компьютерный бестселлер. Гейм-дизайн).

ISBN 978-5-04-164196-2

Unreal Engine 5 — популярный игровой движок, который поддерживается всемирно известной компанией по производству игр Epic Games. Благодаря богатому функционалу и простоте в использовании им пользуются разработчики как крупных игровых проектов (Fortnite, Stalker, Bioshock), так и бесчисленных инди-игр. Руководство, которое вы держите в руках, поможет вам без труда понять внутреннюю логику UE 5, освоить его полный инструментарий и начать генерировать игровые миры и населяющих их персонажей не хуже легендарных геймдизайнеров.

УДК 004.4
ББК 32.973.26-018

ISBN 978-5-04-164196-2

Оглавление

Благодарности и посвящение

Посвящается нашему учителю Подбельскому Вадиму Валериевичу — доктору технических наук и ординарному профессору НИУ ВШЭ.

Никита Веселко, Ольга Максименкова

Мы выражаем глубокую признательность коллегам из игровой индустрии за обсуждения идеи этой книги, ценные комментарии и помощь в понимании особенностей отдельных направлений игровой разработки.

Авторы также благодарят студентов программы бакалавриата «Программная инженерия» факультета компьютерных наук НИУ ВШЭ Шалаеву Марину, Архарова Дмитрия, Панову Екатерину, Миронова Александра, Шепелина Дмитрия и Филатова Юрия за плодотворные обсуждения сложностей, возникающих у новичков, а также за оказанные авторам поддержку и активную помощь в подготовке некоторых текстов глав и примеров кода.

Об авторах

Ольга Вениаминовна Максименкова — кандидат технических наук, научный сотрудник международной лаборатории интеллектуальных систем и структурного анализа ФКН, Национальный исследовательский университет «Высшая школа экономики» (НИУ ВШЭ).

Никита Веселко — владелец продукта (Product Owner) в студии разработки игр IThub games, стажер-исследователь международной лаборатории интеллектуальных систем и структурного анализа ФКН, Национальный исследовательский университет «Высшая школа экономики» (НИУ ВШЭ).

Как устроена эта книга?

ДЛЯ КОГО ЭТА КНИГА

За последние несколько лет Unreal Engine эволюционировал из «просто» игрового движка в мощнейший инструмент для создания цифровых интерактивных миров. Сфера применения UE, куда входят, например, мобильные игры и разработка высокоточных физически корректных инженерных симуляций (какие незаменимы при создании авиационных двигателей, суперкаров и ядерных реакторов) поистине впечатляет!

Если вы хотите попробовать себя в качестве творца высококачественных синтетических миров и попытаться на новом уровне решить эту визуализационную задачу, но при этом не обладаете ни навыками программирования, ни опытом работы с таким огромным комплексным инструментом, как UE, эта книга поможет вам в осуществлении вашей мечты.

ЧТО ВЫ НАЙДЕТЕ В ГЛАВАХ?

Введение познакомит вас с областью геймдева и ее направлениями. Также здесь приведены списки основных интернет-ресурсов и мероприятий, посвященных игровой индустрии.

В первой главе содержится справочная информация о некоторых концепциях игрового дизайна. Это теоретический минимум, посвященный таким понятиям, как игры, игроки, игровая механика и игровой цикл. Здесь же вы узнаете о различных процессах разработки игр и разновидностях игровой документации.

Во второй главе вы познакомитесь с понятием игрового движка и его характеристиками, а также узнаете, чем отличается UE от других своих «собратьев».

Глава третья посвящена работе с контентом и ресурсами в UE: как добавить контент в проект, какие существуют особенности создания материалов и структур, подробный рассказ о статических и скелетных мешах и инструментах для работы с ними в UE. Уделено внимание и работе со звуком.

В четвертой главе находится информация о языке визуального программирования, который называется Blueprints. Чтобы начинающим было проще погрузиться в эту тему, мы собрали в этой главе материалы по основам алгоритмизации и программирования, а также сделали краткий экскурс в объектно ориентированную парадигму разработки, подкрепленный примерами диаграмм UML.

Глава пятая повествует о парадигме проектирования интерфейсов, делающей упор на пользовательский опыт, и кратко рассказывает о паттернах поведения и взаимодействия, важности их применения в процессе создания интерфейсов игр. Кроме того, глава познакомит вас с UMG — инструментом проектирования пользовательских интерфейсов движка UE.

В шестой главе центральной темой является искусственный интеллект (ИИ). Здесь обсуждаются общие вопросы применения ИИ в играх и особенности его реализации посредством встроенных средств движка UE.

В седьмой главе вы познакомитесь с понятием анимации и особенностями ее использования в игровой разработке, узнаете о секвенсоре (Sequencer) и о том, как реализованы анимации в UE.

Восьмая глава представляет собой небольшое введение в нюансы перехода на C++ проекты в UE.

В заключение изложена информация о перспективных направлениях развития движка.

ВВЕДЕНИЕ

Игровая индустрия

Мы назвали этот раздел «Игровая индустрия», подчеркнув природу появления UE. Большинство примеров и процессов, описанных в этой книге, будут относиться именно к играм. При этом фактор универсальности UE не позволит нам обойти вниманием и другие сферы применения движка.

Сообщество и события

Внутри любой серьезной индустрии всегда существует довольно много разнообразных сообществ. Игровая индустрия, которую нам с вами повезло застать в период взросления, тоже не является исключением. Специалисты различных направлений (художники, программисты, продюсеры, аналитики, гейм-дизайнеры и многие другие) регулярно встречаются на бизнес-конференциях как регионального, так и международного уровня. Самыми крупными событиями такого рода, конечно, являются ежегодные знаменитые Выставка электронных достижений (Electronic Entertainment Expo, E3) и Конференция разработчиков игр (Game Developers Conference, GDC) (http://www.gdconf.com), на которых презентуются инновационные технологии, а также демонстрируются самые свежие игры и их концепты.

Но чем могут быть полезны бизнес-конференции новичку? Важно понимать, что успех в делах зависит не только от ваших навыков, но и от актуальности профессиональных знаний, умения общаться и работать в команде, «ощущения» окружающего вас мира. Посещение индустриальных конференций и новые знакомства позволяют определиться и понять, хотите ли вы, чтобы этот мир стал вашим, готовы ли вы ко взаимодействию с людьми, работающими в этой области, к решению ее задач, графику работы и т. п. На мероприятия, проходящие в России и СНГ, например White Nights Conference for Gaming Industry (http://wnconf.com/en), можно зарегистрироваться волонтером и провести несколько дней в закулисье важного игрового события в качестве рядового участника конференции.

Современные технологии позволяют выстраивать сообщества и в виртуальном мире. Здесь нет границ, часовых поясов и необходимости физически присутствовать в определенное время в конкретном месте. Одним из самых старых сетевых ресурсов является база знаний, состоящая из статей специалистов по различным аспектам

разработки игр, Gamasutra (http://www.gamasutra.com), и если повезет, вы еще успеете застать этот сайт в его старом виде, так как сейчас площадка претерпевает ребрендинг и переезд на новый домен Game developer (https://www.gamedeveloper.com/). В русскоязычном сегменте также имеются как узкопрофильные, так и разнообразно специализированные сообщества, знакомство с которыми мы предоставим читателю осуществить самостоятельно, но все же отметим важный в контексте этой книги ресурс UE4 Daily (https://ue4daily.com/), где можно познакомиться с русскоговорящим тематическим сообществом посредством проектов и публикаций, узнать об образовательных возможностях, связанных с UE, а также просто почитать новости мировой игровой индустрии.

Отдельного упоминания заслуживают гейм-джемы (game jam) — события, в которых участвуют и профессионалы, и любители, и просто сочувствующие. Цель джема — за ограниченное время создать игру на заданную тему. Получившиеся в результате игры и их прототипы чаще всего попадают на своего рода игровой файлообменник itch.io — открытый магазин приложений независимых разработчиков видеоигр. Конечно, требования к играм могут разниться от джема к джему, но неизменной остается базовая идея: люди приходят туда, чтобы ощутить дух и радость творчества. Здесь можно протестировать любую геймплейную и гейм-дизайнерскую гипотезу, стихийно создать команду и расширить круг знакомств.

Существуют исследования в области образования, показывающие, что такие мероприятия заметно влияют на процесс обучения разработке игр. Кроме того, в гейм-джеме можно не только участвовать, но и самостоятельно его организовать. Начните знакомство с джемами, например с Global Game Jam (https://globalgamejam.org/) или Ludum Dare (https://ldjam.com/).

Зрелым индустриям сопутствуют развитые научные направления. Область научной коммуникации подразумевает существование социальных связей, накопление и формирование нового знания и его распространение. Те ученые и лидеры технологических компаний, что имеют возможность содержать собственные R&D-подразделения (Research and Development), изучают разные аспекты игр посредством тестирования гипотез, разработки новых алгоритмов и создания специализированной аппаратуры. Исследователи в области компьютерной графики могут похвастаться существованием такой мировой конференции по данной тематике, как ACM SIGGRAPH; сфера изучения человеко-машинного взаимодействия представлена не менее известной ACM SIGCHI; гуманитарные исследования обсуждаются в рамках мероприятий по game studies. Также в настоящее время продолжается формирование профильных конференций и появляются научные журналы в области игровой индустрии.

Unreal Engine 5

Эта книга ориентируется на самую свежую, пятую, версию движка UE. Этот раздел призван показать некоторые различия между четвертой и пятой версиями. И хотя движок в основном поддерживают совместимость для проектов, разработанных в младших версиях, также присутствует ряд особенностей, о которых полезно знать разработчику.

Примечания к выпуску этой версии движка можно прочесть по следующей ссылке (https://docs.unrealengine.com/5.0/en-US/ReleaseNotes/)

ВОЗМОЖНО ЛИ ОТКРЫТЬ В UE5 ПРОЕКТЫ, СОЗДАННЫЕ В UE4?

Пятая версия UE серьезно отличается от предыдущей, объем внесенных изменений равноценен продвижению движка на четыре версии вперед. Чем сложнее ваш проект и чем больше масштабных изменений было внесено в используемые в нем подсистемы UE, тем выше вероятность возникновения несовместимостей при попытке открыть ваши наработки в новой версии. Несовместимость может затронуть внешнюю интеграцию и плагины, поэтому обязательно следует проконсультироваться с разработчиками этого программного обеспечения, чтобы узнать, существует ли совместимая с UE5-версия их продукта. В документации UE5 присутствует отдельное руководство по миграции проектов с UE4 на UE5.

Unreal Engine 5 Migration Guide (https://docs.unrealengine.com/5.0/en-US/unreal-engine-5-migration-guide/)

ПЕРЕНОС ПРОЕКТОВ С UNREAL ENGINE 4 НА UNREAL ENGINE 5

Хотим заострить ваше внимание на том факте, что UE5 по своей сути является глобальным обновлением предыдущей, четвертой версии движка. Это означает, что при переходе с UE4 на UE5 не придется с нуля изучать работу с данным программным продуктом.

Blueprints и C++

Blueprints и C++ по-прежнему являются основными инструментами программирования в UE5. Новая версия движка принесла с собой много внутренних улучшений и изменений, но основной функционал остается прежним, поэтому любые учебные

материалы, ориентированные на Blueprints или C++ в UE4, будут актуальными и для UE5. Для студентов это означает, что они по-прежнему могут полагаться на широкий спектр учебных ресурсов, доступных в печатном виде и онлайн. Студентам, незнакомым с UE, может потребоваться дополнительная помощь при изучении более старого учебного материала. Если в процессе его изучения и адаптации к новой версии движка возникают трудности, вы всегда можете обратиться к предыдущим версиям UE4, которые доступны для загрузки через лаунчер Epic Games и Git.

Gameplay Framework

Gameplay Framework остается основой для разработки игр на UE. Все так же в силе такие концепции, как акторы, классы и интерфейсы, а также дизайн и принципы ООП в виде инкапсуляции и взаимодействия классов. А такие новые функции UE5, как модульный игровой процесс (Modular Gameplay) и реестры данных (Data Registries), дополнительно улучшают уже проверенные временем системы.

Синтаксис Blueprint не претерпел изменений и в UE5. Проекты, использующие Blueprint, не нуждаются в каких-либо модификациях, хотя производительность приложений может быть снижена. В этом случае разработчикам потребуется прибегнуть к оптимизации кода.

Структура проекта и контроль версий

В пятой версии движка базовая структура проектов Unreal сохранилась. Соответственно, не изменился и привычный набор практик и рабочих процессов с использованием систем контроля версий или других подходов для совместной работы над файлами. Улучшен процесс сохранения уровней на диск (система One File Per Actor), но основные концепции структуры папок (Content, Config, Binary и Source) остались прежними.

ГДЕ ИСКАТЬ УЧЕБНЫЕ МАТЕРИАЛЫ?

Компания Epic Games предоставляет множество качественных материалов для изучения UE5, которые помогут вам освоить новые инструменты движка.

Идущий вместе с движком демонстрационный проект Valley of the Ancient представляет собой короткий сегмент геймплея, иллюстрирующий новые возможности движка. Проект можно загрузить из маркетплейса, для запуска потребуется высокопроизводительный компьютер, работающий на ОС Windows (кроме того, потребуется ≈100 Гб свободного места на диске).

МОЖНО ЛИ ПЕРЕНЕСТИ ПРОЕКТЫ И КОНТЕНТ ИЗ UE5 ОБРАТНО НА UE4?

Нет, обратной совместимости движок не обеспечивает.

О новых возможностях интерфейса движка можно прочесть в соответствующем разделе документации UE (https://docs.unrealengine.com/5.0/en-US/EditorImprovements)

Какие платформы поддерживает Unreal Engine 5?

UE5 поддерживает те же платформы, что и UE4, — консоли нового поколения, консоли текущего поколения, Windows, macOS, Linux, iOS и Android. Исключением являются такие подсистемы движка, как Nanite и Lumen, которые поддерживаются только на консолях нового поколения и Windows.

Доступен ли Unreal Engine 5 на GitHub?

Да! Как и сборка раннего доступа, так и версия в UE5/Main отзеркалена на страницу GitHub, давая возможность загрузить и скомпилировать свою версию движка.

Отметим, что в отличие от регулярных релизов и патчей, эти билды находятся в текущей разработке и полная их работоспособность не гарантируется. Кроме того, для этих билдов не предоставляется техническая поддержка.

В пятой версии интерфейс движка существенно обновлен, процесс взаимодействия с ним улучшился, а также оптимизировано использование площади экрана, в результате чего работа с движком стала проще, быстрее и приятнее.

Рисунок 1. Вид нового интерфейса Unreal Engine

Как видно из рисунка 1, в окне всегда отображается командная строка (**cmd**), что упрощает и ускоряет работу с командным языком.

Чтобы разработчику было доступно больше экранного места для взаимодействия с областью просмотра, в пятой версии появилась возможность легко сворачивать и разворачивать **Content Browser** (комбинация клавиш **Ctrl+Spacebar** или клик по кнопке **Content Drawer**), а также прикреплять любую вкладку редактора к сворачиваемой боковой панели.

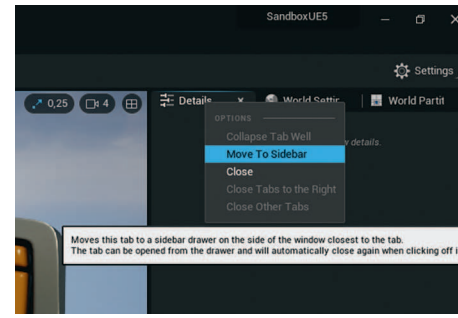

Рисунок 2. Контекстное меню вкладки, подсвечен пункт прикрепления ее к панели

Новая система добавления элементов в избранное обеспечивает быстрый доступ к часто используемым свойствам на панели **Details**, а новая кнопка **Create**, расположенная на главной панели инструментов, позволяет легко размещать акторов на сцене.

В UE5 значительно улучшен и вместе с тем упрощен процесс создания проектов.

В UE5 облегчен доступ к бесплатным сторонним ассетам, предоставляемый плагином Bridge (**Content › Quixel Bridge**, опция активирована по умолчанию). Библиотека ассетов доступна при условии авторизации в учетной записи Epic Games (на нее должна быть зарегистрирована лицензия UE), соответственно, контент может быть использован только в Unreal Editor.

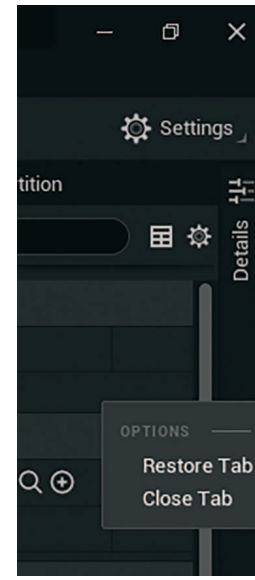

Рисунок 3. Восстановить/закрыть свернутую вкладку

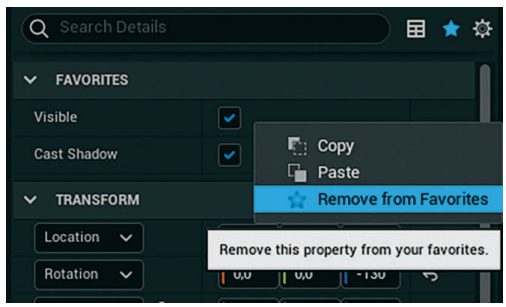

Рисунок 4. Пункт удаления элемента из избранного

В UE5 добавлена возможность автоматического преобразования типа в Blueprints. По умолчанию эта опция активна, но ее можно отключить в настройках редактора (**Edit › Editor Preferences › Content Editors › Blueprint Editor › Workflow › Enable Type Promotion**).

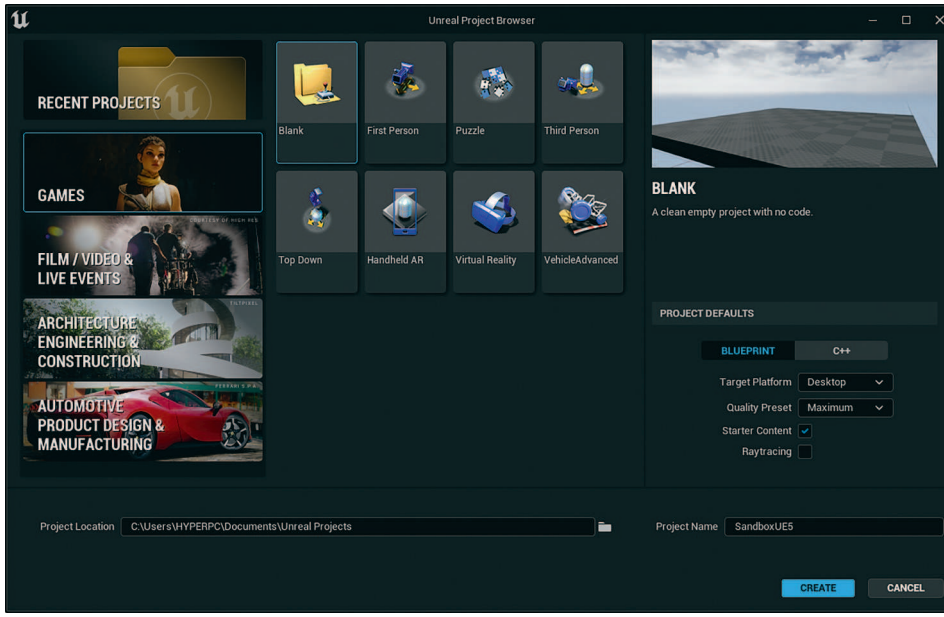

Рисунок 5. Новый интерфейс окна-проводника для работы с проектами

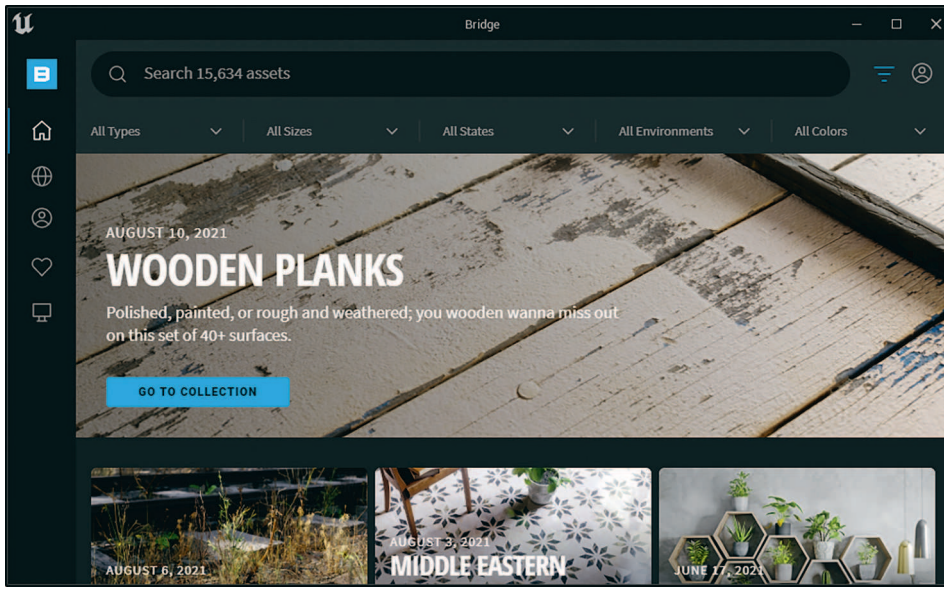

Рисунок 6. Окно плагина Bridge, открывающего доступ к библиотеке бесплатных ассетов

Несмотря на то что в новой версии движка представлено немало изменений в пользовательском интерфейсе, функционал во многом похож на тот, что был ранее. Все визуальные референсы для UE4 должны без особых проблем транслироваться на новую версию движка.

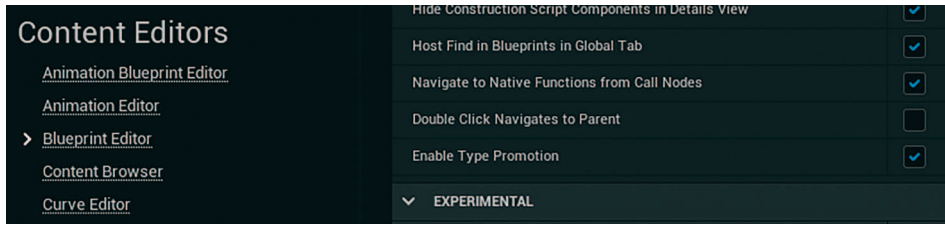

Рисунок 7. Параметр Type Promotion активирует автоматическое преобразование типа в Blueprints

Создание миров и совместная работа

Более подробно об этой теме можно прочесть в соответствующем разделе документации движка (https://docs.unrealengine.com/5.0/en-US/WorldFeatures)

Разработчики UE постоянно совершенствуют процесс создания динамических открытых миров, стараясь сделать его проще, быстрее и удобнее для совместной работы команд любой численности. В пятой версии движка представлена система World Partition, которая предлагает новый подход к управлению уровнями и их стримингом, для чего автоматически разбивает мир с помощью сетки, тем самым позволяя обрабатывать только необходимые в данный момент ячейки.

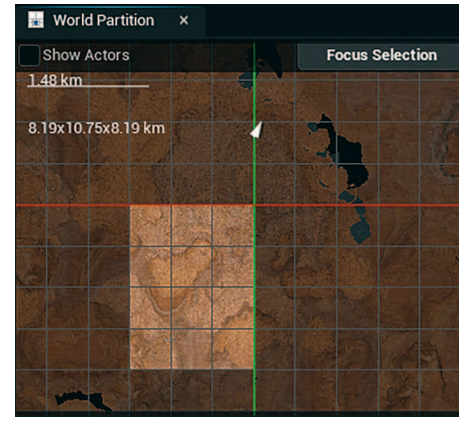

Рисунок 8. Ячейки сетки, на которую поделен мир

Активировать систему World Partition можно в настройках проекта (**Edit › Project Settings › World Partition › Enable World Partition**). По умолчанию панель World Partition скрыта, а открыть ее можно в меню **Window**. Далее нажатием левой кнопки мыши можно выделять области на **World Partition**, а нажатием правой кнопки мыши по ячейке или выделенной области можно загружать/

сбрасывать выделенные ячейки. Обратите внимание, что разрешение портирования доступно только на тех уровнях, которые были созданы после активации этой опции, а чтобы в панели World Partition отображалась корректная информации об уровне, он должен быть сохранен.

Отладка доступна с помощью консольной команды wp.Runtime.ToggleDrawRuntimeHash2D, она включает/отключает отрисовку карты с ячейками и загружаемыми в режиме реального времени объектами.

Благодаря новой системе One File Per Actor, которая разбивает каждый уровень на наборы файлов, группы разработчиков могут одновременно работать над игровым миром, не мешая друг другу. Это значит, что изменения, внесенные вами в уровень, будут сохраняться и отправляться только для тех акторов, которых вы редактировали или помещали на уровень.

Система One File Per Actor автоматически включается только в уровнях с разрешенной World Partition. Чтобы задействовать систему **совместной работы** в обычных уровнях для всех акторов, нужно в панели **World Settings** включить опцию **Use External Actors**. Кроме того, One File Per Actor можно включить и для отдельного актора, установив значение его параметра **Packaging Mode** в **External**.

Не обошли стороной изменения и в работе UE с системами контроля версий, в частности с Perforce, теперь разработчики могут выполнять почти все SCM-операции непосредственно в редакторе.

Использование рассмотренных новых инструментов движка позволяет значительно упростить процесс как командной разработки, так и персональной (например, для студентов, желающих попробовать себя в разработке игр с открытым миром).

ГЛАВА 1

Основы разработки игр

Поскольку речь в книге так или иначе будет идти о компьютерных играх, чтобы не заставлять читателя постоянно отвлекаться на поиск объяснения того или иного термина, эта глава создана как некоторый базовый справочник понятий из мира игровой разработки.

Вы узнаете:

- *формальные определения понятий «игра» и «игрок»;*
- *какие существуют способы удержания игрока в игре;*
- *что такое игровой цикл и игровая механика (на уровне понимания, достаточном для программиста);*
- *как правильно читать проектную документацию и работать с ней (что критически важно для тех, кто работает в составе команды);*
- *об особенностях игровой документации.*

Игры и игроки

Каких только не существует определений игры! Мы, как авторы этой книги, будем опираться на академическую трактовку, данную в знаменитой книге Йеспера Юля Half-Real. Video Games between Real Rules and Fictional: игра — это формальная, то есть основанная на правилах, система, в рамках которой игрок действует и прилагает усилия, чтобы повлиять на результаты игры, которые должны быть эмоционально важными для играющего, разнообразными, количественно измеримыми, обладать различной ценностью для игрока, быть необязательными и отвратимыми.

Йеспер Юль (1970 г. р. https://www.jesperjuul.net/) — доцент школы дизайна Датской королевской академии изящных искусств, гейм-дизайнер, преподаватель и исследователь в области видеоигр, автор и соавтор нескольких книг о видеоиграх, изданных в MIT Press

Понятие отвратимости известно каждому, кто хоть раз играл в игры (и не только компьютерные) — это возможность переигровки и сохранения промежуточных результатов игры, а также ненастоящие потери жизни. Важное

отличие игры от реального мира — это «понарошковость» всего происходящего. Игрок всегда может переиграть!

Поскольку в этой книге мы будем обсуждать преимущественно компьютерные (видео) игры, давайте познакомимся с еще несколькими определениями этого явления, данными другими исследователями.

Николас Эспозито, руководитель отдела исследований в школе визуальной коммуникации и искусства Gobelins, в своем докладе[1], сделанном на конференции Digital Games Research Conference в 2005 году, кратко охарактеризовал понятие видеоигры как разновидности игры, основанной на некоторой истории, в которую можно сыграть благодаря аудиовизуальной аппаратуре. Существуют и активно обсуждаются и другие варианты определений, которые любопытный читатель может найти в статье Jonne Arjoranta «How to define Games and why we need to»[2].

Игры создаются людьми для людей, поэтому разработчику очень важно понимать, что представляют собой игроки как явление.

Концепция игрока характеризуется несколькими связанными между собой понятиями. Чтобы в дальнейшем их было легко различать и использовать, давайте разберем каждое из них. Прежде всего следует поговорить о **пользователях**, так как именно они запускают на своих компьютерах программы и выполняют в них различные действия для решения своих задач. Иногда пользователи даже играют в компьютерные игры. Именно в этот момент пользователь становится **игроком**, то есть выступает в социальной роли «игрока» в реальном мире. Далее мы иногда будем взаимозаменять термины «пользователь» и «игрок», но только там, где это не вызовет путаницы и двусмысленности.

Следующее понятие — **аватар**. Это представитель игрока в мире игры, социальная роль, в игре называемая **персонажем**, например можно быть темным или светлым магом, морским пехотинцем или ученым. В каждом из этих случаев вы будете управлять соответствующим персонажем.

КАК ИГРА ВОЗДЕЙСТВУЕТ НА ИГРОКА?

Интересно отметить, что среди прикладных программ цифровые развлечения и игры обладают уникальным свойством: в их задачи входят вовлечение и развлечение пользователя. Подобные требования не предъявляются к калькуляторам, средам разработки программного обеспечения или системам бизнес-аналитики, а вот все происходящее в игре должно удерживать внимание игрока, мотивируя его не бросать

[1] *Esposito, N.* A short and simple definition of what a videogame is: In DiGRA 2005 conference: Changing views-worlds in play. 2005 (https://www.utc.fr/~nesposit/publications/esposito-2005definition.pdf).

[2] https://link.springer.com/article/10.1007/s40869-019-00080-6#ref-CR8.

игровой процесс. Если же в процессе игры игрок заскучает, то, скорее всего, он переключится на другое, более интересное занятие, поэтому разработчики игр стремятся создавать такие виртуальные миры, которые способны надолго увлечь игрока, не надоедая ему.

> **Прикладные программы** (application software, applications, apps) согласно IBM Research — это программное обеспечение, необходимое для решения или помощи в решении задач пользователей. К прикладным программам относятся офисные продукты, системы управления данными. Веб- и мобильные приложения также могут быть прикладными программами, например мобильные приложения торговых марок или супермаркетов, социальных сетей и т. п.
>
> **Системные программы** (system software) согласно тому же порталу представляют собой набор функций (или целый комплекс программ) по управлению подсистемами компьютера. К системным программам относятся операционные системы, системы управления дисками и другими аппаратными средствами, утилиты и т. п.

Так что же заставляет игрока продолжать играть? Большинство «хитрых манипуляций» можно разделить на три большие группы: позитивное подкрепление, негативное подкрепление и принуждение. Позитивное подкрепление (positive reinforcement) означает вознаграждение игрока за его действия. Негативное (отрицательное) подкрепление (negative reinforcement), наоборот, предполагает устранение негативного воздействия на игрока в случае выполнения им желаемого авторами игры действия. Не следует путать негативное подкрепление с принуждением (punishment), смысл которого именно в непосредственно негативном воздействии на играющего, чтобы заставить его действовать определенным образом. Забавно, что согласно исследованиям принуждение — самый эффективный способ сделать так, чтобы игрок избегал определенных действий. Вспомните, например, необходимость перемещаться по лаве, кислоте или другим токсичным субстанциям, отнимающим здоровье. Повторять подобные действия и попадать в соответствующие среды лишний раз не хочется, у игрока появляется желание избегать их, ища обходные пути, но «успешный» прорыв через опасные препятствия воспринимается как преодоление и маленькая победа. Если же цель — добиться от игрока постоянного повторения каких-либо действий, в этом случае стоит использовать положительное подкрепление. Игры жанра «три в ряд» не скупятся на похвалу за каждые три убранных объекта, даже если игрок проходит уже пятисотый по счету уровень.

В битве за удержание внимания игрока часто используют **постепенное открытие доступа к контенту**, это может быть любое содержимое игры, с которым сталкивается игрок. Если контент доступен сразу и полностью, он подобен прочитанной книге,

которую, может быть, никогда больше и не перечитают. Поэтому можно ограничить количество изначально доступного пользователю материала игры, открывая его по мере того, как игрок продвигается дальше. Это способствует появлению чувства преодоления и победы, тем самым поддерживая интерес к игре. И гейм-дизайнерам, и программистам во время проектирования продукта следует хорошо продумывать вопросы, связанные с постепенным доступом к контенту. Например, если в игре содержится много материала, то его поэтапное открытие будет работать не только на удержание внимания игрока, но и на скорость погружения в игру, что даст разработчикам различные лазейки для оптимизации программы.

Создание для игрока такой атмосферы, чтобы он ощущал себя частью созданного мира и максимально погрузился в игру, задача сложная, потому что каждый человек по-своему взаимодействует с виртуальным миром и может не «вжиться» в навязываемую ему роль. Разработчики по-разному подходят к решению этой проблемы — кто-то создает правдоподобные сценарии и персонажей, кто-то делает акцент на реалистичную графику, а кто-то думает о стиле и музыке, ясном культурном коде и паттернах. Кроме того, сейчас практически все игры стараются выстроить **интересную интерактивную историю**. Подчас успех действительно кроется в захватывающем повествовании, грамотно выписанных персонажах и умелой подаче сюжета через игровые механики, окружение и ощущения. Следует отметить, что создание осмысленной истории, раскрывающейся через интерактивное игровое окружение, — это нетривиальная и комплексная задача, поэтому обычно ее решением занимаются специальные гейм-дизайнеры — нарративные дизайнеры. Если вы хотите побольше узнать о том, как работают эти специалисты, начните с изучения материалов на портале Narratorika.

Из сказанного выше становится понятно, что многие решения, принимаемые на уровне гейм-дизайна, оказывают серьезное влияние на разработку. Например, **вариативность** может быть реализована разными способами — развилками в сюжете, предоставлением выбора вариантов окружения или каких-либо объектов и др. Некоторой части игроков вариативность настолько важна, что существует даже целый жанр игр, называемый «рогаликами» (roguelike и roguelite games), где частенько применяются алгоритмы процедурной генерации событий, игровых сценариев, ландшафтов и карт, что позволяет игроку получать уникальные эмоции от фактически каждого перепрохождения игры.

М. Качакова. «Кто такой “Нарративный дизайнер”?», перевод статьи Э. Макрея What is a Narrative Designer? (http://narratorika.com/whosnarrativedesigner)

Соревнования в играх, бесспорно, добавляют им интереса и фана. Как мы знаем, соревновательность в играх породила киберспорт. Так уж получилось, что людям нравится соревноваться друг с другом или развитыми NPC (Non-player Character), управляемыми искусственным интеллектом. Даже состязание с ИИ рождает новый игровой

опыт, поэтому что уж говорить о противостоянии формата «человек — человек» в навороченной интерактивной медиасреде. Игра против серьезных оппонентов обладает тонким соревновательным моментом в виде получения от процесса массы эмоций, хоть и не всегда позитивных. Например, хорошо запрограммированный искусственный интеллект может как развлекать игрока, так и заставлять его скучать, если ИИ будет играть слишком хорошо и человек будет постоянно ему проигрывать. Для решения этой проблемы используется **балансировка**, которую программисты либо разрабатывают, либо же только создают инструменты, позволяющие гейм-дизайнерам балансировать игру самостоятельно. Однако отметим, что тот же развитый искусственный интеллект может быть не противником, а союзником, который обучает игрока или подсказывает и дает рекомендации, но это уже совсем другое направление развития ИИ.

И раз уж мы заговорили о взаимодействиях, то упомянем здесь и **социальные элементы**, особенно характерные для многопользовательских игр. Использование социальных взаимодействий позволяет игрокам более эффективно и достоверно взаимодействовать друг с другом. Внутриигровые чаты, возможность объединяться с другими игроками для прохождения испытаний и прочие возможности, позволяющие хорошо проводить совместно время, переносят игру в реальную жизнь, становясь фундаментом для создания сообществ.

Почему мы говорим об этом в книге по программированию? Потому что нам предстоит программировать поведение, предопределенное гейм-дизайнерами, где обязательно будут использоваться перечисленные методы управления мотивацией и вовлеченностью игрока, поэтому знакомство с ними не будет лишним. Кроме того, авторы книги надеются, что читатели, успешно освоившие материал книги, станут завсегдатаями гейм-джемов и конкурсов, где функции гейм-дизайнера, вероятно, поначалу придется выполнять самостоятельно. Поскольку наша книга в основном повествует о движке UE и программировании, отметим, что читатель может самостоятельно познакомиться с наиболее популярными и простыми методиками развлечения игрока, начав, например, с чтения статьи на портале DTF.

В. Семыкин. «Методы удержания игрока» (https://dtf.ru/gamedev/38612-metody-uderzhaniya-igroka)

Что такое игровые механики и с чем их «едят»?

Поскольку реализация игровых механик при помощи кода является задачей программистов, давайте попытаемся разобраться, что же это такое. Сразу отметим, что мы не будем слишком углубляться в тему гейм-дизайна, оставляя это занятие профессионалам, поэтому в книге споров об определении понятия игровой механики не будет.

> **Игровая механика** (game mechanic) — способ взаимодействия с игровыми объектами в рамках установленных ограничений игры, изменяющий ее состояние, влияющее на принятие игроком последующих решений.
>
> *Сергей Гиммельрейх*

Тайнан Сильвестр в книге «Гейм-дизайн» пишет, что механика — это правила, по которым работает игра. Иногда говорят, что множество игровых механик формирует игровой процесс, или геймплей (gameplay), игры. Геймплей конструируется на базе заданных правил (механик) и ограничений игры, а также поведения игрока, поэтому результаты игрового процесса и ощущения от игры у разных людей могут быть различными.

Для настольной игры, такой как шахматы или карты, существуют правила, определяющие, что и как должен и может делать игрок, играя в игру. Правила устанавливают порядок ходов, разрешают спорные ситуации и даже определяют цель игры. Ходить пешкой только вперед — это правило, тонкости короткой и длинной рокировки — это тоже правило, возможность убрать из колоды шестерки для игры в преферанс — это ограничение и т. д. В данном случае правила задают механики для настольной игры. В компьютерных играх механики придется запрограммировать, например в шутере — механику стрельбы, в платформере — механику прыжка и т. д.

Игровые циклы и циклы разработки игр: смешать, но не путать

В основе любой игры лежит одно или несколько действий, которые игрок повторяет на протяжении всего игрового процесса. Каждое выполнение действия обязательно приводит к определенному результату (неважно, плохому или хорошему). Именно такие повторы основных игровых действий (механик) называются игровыми циклами (гейм-луп, gameplay loop). Игровые циклы конструируют гейм-дизайнеры, стараясь придумать увлекательные действия, которые затянут пользователя внутрь игры и будут его там удерживать. Денис Поздняков, специалист по нарративному дизайну, определяет игровые циклы как результат комбинирования целеполагания с игровыми механиками.

> **Денис Поздняков** — разработчик видеоигр, директор по продукту IThub games, преподаватель школы «Нарраторика».

Следующее ключевое понятие — это цикл разработки игр. Не путайте его с игровыми циклами! Это понятие относится не к игровому процессу, а к процессу разработки игры. Данный тип цикла можно условно разделить на пять основных этапов.

1. **Подготовка и анализ требований.** Это этап планирования будущего продукта, в случае создания игры сюда входит выбор жанра игры, платформы, под которую будет вестись разработка, определение целевой аудитории.
2. **Проектирование.** На этом этапе создается концепт игры, где «начерняются» игровые механики и процессы, а также прототипирование — создание «макета» какой-либо игровой механики или сценария для проверки того, как это «вписывается» в проект и что можно еще улучшить или изменить. В этой части разработки обычно создается игровая документация (ее еще называют «диздок» — сокращение от «дизайнерский документ»), она содержит полное описание игры. Получив подобный документ, компетентный в разработке специалист сможет создать игру, соответствующую задумке.
3. **Программирование.** Непосредственно разработка компонентов игры и их объединение. Данный этап включает в себя не только написание программного кода, но и работу с другими системами UE5, например с системой

визуального программирования Blueprint, разработку интерфейсов или создание игровых сцен. Все упомянутые виды разработки мы будем обсуждать в следующих главах.

4. **Тестирование.** Проверки на наличие ошибок и некорректной работы разработанных систем следует осуществлять не только на данном этапе, но и после каждого значимого изменения в проекте. Чтобы контроль ошибок был как можно более качественным, для различных систем и игровых механик разрабатываются всевозможные тесты и проводятся плейтесты среди коллег или знакомых.
5. **Внедрение.** Разработка проекта завершена и его можно представлять конечному потребителю — на физических носителях или посредством дистрибуции через онлайн-магазины.

Что такое игровая документация?

Говоря о процессе разработки, следует сказать пару слов о формальной стороне вопроса. Как и в случае создания любой другой информационной системы, у игрового приложения есть свой набор документации. Она может быть разной в зависимости от конечной аудитории: программисты, моделлеры, саунд-дизайнеры и т. д. Однако ядром любой игровой документации являются несколько основных «бумаг», за которые отвечает ведущий гейм-дизайнер проекта.

В перечень основных документов игровой разработки входят:

1) вижн игры;
2) концепт игры;
3) документация по гейм-дизайну.

Вижн (видение) игры — это краткое описание будущей игры. В этом документе необходимо отобразить то, как мы видим игру, о чем она и что будет делать наш игрок. Также в этом документе следует обозначить целевую аудиторию.

Концепт разрабатывается после вижна игры. Фактически вижн нужен для понимания того, какую игру мы будем делать и есть ли вообще смысл в ее создании. Далее на основании этого создается структурированный документ, в котором дается общее описание игры, обозначаются платформы, целевая аудитория и различные детали игры (2D или 3D, какой движок, размер игры и т. д.). Объем концепт-документа обычно составляет 1–2 страницы, из которых можно понять идею игры и ее перспективы.

Документация по гейм-дизайну — это набор документов с максимально подробным описанием всех деталей игры, так как именно на основе этого материала будут составляться задачи для всех участников создания проекта. Например, в документации должно содержаться описание основных механик, визуального вида игры, ИИ, сеттинга.

Пример гейм-дизайнерского документа для игры Fallout: Brotherhood of Steel 2 (https://fallout.fandom.com/wiki/Fallout:_Brotherhood_of_Steel_2_design_document)

ГЛАВА 2

Игровые движки

В этой главе мы поговорим об игровых движках. Вы узнаете, что представляет собой это программное обеспечение, из каких частей состоит игровой движок, в чем удобство его использования и какова связь движка с циклом игровой разработки. И, наконец, начнем знакомство с движком UE как с серьезным инструментом разработки трехмерных виртуальных интерактивных миров.

Как делают игры?

Итак, у нас есть идея игры, но каким образом ее можно реализовать? Начнем с того, что формально игра — это набор исполняемых файлов, запускаемых на некоторой платформе (например, на игровой консоли, смартфоне или компьютере), и файлов ресурсов. Все эти файлы входят в состав приложения для некоторой платформы. Его нам и требуется создать.

У каждой платформы есть своя специфика. Предположим, что нам удалось обуздать ту магию, которая создает интерактивное окно и рисует внутри него объекты. Теперь нам нужно заставить их перемещаться и реагировать на нажатие клавиш или какой-либо другой тип ввода команд от пользователя (то есть с объектами должна быть возможность взаимодействия). Для этого нам потребуется запрограммировать игровой цикл, в котором предусмотрены обновления окна и проверка данных, вводимых игроком.

Конечно, интерактивность — это только начало, нам понадобится написать еще кучу кода, который будет реализовывать логику игрового процесса, сетевые взаимодействия (если есть такое требование), запуск и синхронизацию звука и анимаций. Придется не забыть также и про ботов и реализовать их поведение в игре, а также не упустить из виду такие прозаические вещи, как изменение настроек игры и сохранение прогресса игрока. Сама по себе программная логика — это каркас игры, который только еще предстоит визуально оформить. Для этого потребуется создать различный контент в виде файлов с трехмерными моделями или двумерными спрайтами, текстом, анимацией, звуком и текстурами. Все это тоже должно обрабатываться, компоноваться и настраиваться.

Но игры — это нечто большее, чем просто программное обеспечение. Любая игра — это уникальный сплав технологий и средств художественной выразительности. И эта особенность игр как приложений ярко проявляется в знаменитом конфликте дизайнера (проектировщика) и инженера: не все, что красиво выглядит на бумаге, может быть реализовано в реальности. Можно создать чудесную трехмерную модель, состоящую из миллионов мелких деталей, но ее не получится вставить в игру, потому что объект слишком «тяжелый» и долго грузится в процессе игры на конкретной платформе. Можно придумать умный искусственный интеллект (ИИ), который будет хитрее любого игрока, но такой бот не будет включен в игру, потому что людям не очень нравится постоянно проигрывать. Можно придумать революционную идею для игрового процесса, но если ее невозможно реализовать с помощью имеющихся инструментов, она так и не увидит свет. Разработчикам приходится искать баланс между техническими возможностями, концептами и аудиовизуальными образами, чтобы проект и был реализуем, и позволял получить неповторимый интерактивный опыт.

Даже такое краткое описание позволяет ощутить, насколько игры как программный продукт сложны и сколько для своего появления на свет требуют согласованной работы многих подсистем. Разработчики должны предусмотреть эту слаженность, иначе видео будет расходиться со звуком, аватары будут проваливаться по щиколотку в твердые поверхности, да и мало ли какие еще неприятности могут возникнуть? Каждый, кто играл в игры, знает, как неприятно видеть недоделки. И здесь мы вплотную подходим к теме игровых движков.

Что такое игровой движок?

В предыдущем разделе вы узнали, из какого количества взаимодействующих (и весьма отличающихся друг от друга) частей состоят компьютерные игры. Настройка и синхронизация подсистем вручную — это, безусловно, заслуживающий похвалы труд, но в условиях реальной разработки он слишком ресурсоемок. Проще говоря, эта работа потребует очень много времени, высокого уровня подготовки разработчиков и немалых финансовых вливаний. Для того чтобы процесс шел быстрее, используют специальные инструменты разработки (да, их тоже иногда пишут с нуля и тогда формула «долго, дорого и тяжело» остается в силе). Готовый игровой движок также является таким специальным инструментом разработки. Поскольку игры — это программы, то игровые движки — это программы для создания других программ. Задача движка — снять с вас головную боль, связанную с согласованием работы подсистем игры и помочь специалистам разных направлений работать эффективнее. Кроме того, игра состоит из множества компонентов, часть

What is a Game Engine? (https://www.gamecareerguide.com/features/529/what_is_a_game_.php?page=2)

которых впоследствии можно использовать повторно при создании других игр. Джефф Уорд (сооснователь компании Orbus Gameworks) пишет, что игровой движок позволяет абстрагироваться от деталей выполнения общих задач, связанных с игрой, таких как рендеринг, физика и обработка пользовательского ввода, чтобы разработчики (а также художники, дизайнеры, сценаристы и даже другие программисты) могли концентрироваться только на тех вещах, что делают игру уникальной.

Порой непонятно, где начинается игра и заканчивается движок, однако на самом деле разница между этими понятиями не так важна. Важно, какие преимущества дает разработчику использование готового движка. Возможность опереться на уже написанный код и наличие удобного редактора значительно ускоряет процесс создания игры. Во-первых, разработчику не приходится писать многие вещи с нуля. Например, организация игрового цикла, отрисовка объектов и просчет их физики часто уже реализованы в движке. Во-вторых, за разработкой движка стоит команда специалистов, которая занимается его постоянной поддержкой. Это значит, что при создании игры разработчик может опираться на решения, уже реализованные в движке, и заниматься отладкой только той части игры, которая была написана поверх «фундамента». Вследствие этого упрощается прототипирование, но, разумеется, доведение игры до ума (оптимизация и прочие тонкие настройки) все равно потребует вмешательства разработчика. Выпуск игры на разные платформы часто отнимает много усилий, однако современные движки делают все возможное, чтобы максимально упростить решение этой задачи. А когда на рынке появляется новая платформа или технология (например, AR/VR), весьма вероятно, что в скором времени движок начнет поддерживать эти возможности и их можно будет использовать в игре, не прибегая к значительным переделкам проекта.

Основные элементы игрового движка:

— компонент **моделирования мира и синхронизации с игровыми событиями**. Отвечает за описание модели мира и привязку ее событий (как правило, выраженных в шкалах времени) к абсолютной шкале времени пользователя;

— компонент **синтеза изображений и звуков**. Отвечает за визуализации, видео и аудио, включая аппаратное ускорение;

— компонент **обеспечения взаимодействия с пользователем**. Позволяет обрабатывать события (аппаратные прерывания) устройств ввода и вывода.

Несмотря на то что перечисленного вполне достаточно для создания скелета игры, технологии не стоят на месте, и в настоящее время в движки включаются такие дополнительные компоненты, как:

— компонент непрерывной поддержки сессий игрока для возможности загрузки и сохранения игрового прогресса;

— компонент коммуникации в игровом мире, чтобы упростить взаимодействие с NPC, общение с другими игроками и ведение журналов событий;

— компонент для реализации подсистемы искусственного интеллекта;

— компонент симуляций (в том числе и корректных физически) в реальном времени;

— редактор игрового мира, из которого можно запускать редактор содержимого, редактор кода и скриптов, инструменты оптимизации;

— система управления конфигурациями для поддержки версионирования артефактов проектирования и упрощения документирования разработки;

— система обратной связи обеспечивает персонализированную и внешнюю обратную связь, генерацию отчетов об ошибках.

Как уже говорилось, игровой движок можно разработать самостоятельно, купить готовый или использовать его бесплатные версии.

Особенности Unreal Engine

Unreal Engine — это свободный для использования игровой движок, его можно бесплатно скачать, установить и начать разрабатывать игры прямо сейчас (https://www.unrealengine.com/). Бесплатность движка является условной, при определенных условиях разработчик должен начать выплачивать проценты с продажи игры, написанной с использованием UE. Подробнее об этом можно узнать на официальном сайте приложения. Давайте познакомимся с ключевыми компонентами движка (а он состоит из множества инструментов), их возможностями и особенностями.

Отрисовка цвета предмета, взаимодействие освещения и материала объекта, создание тени, способ использования прозрачных материалов, а также обсчет в реальном времени всей этой картинки с частотой 60 кадров в секунду (то есть экран обновляется каждые 16 миллисекунд), а то и быстрее, не создавая при этом угрозу жизни железу пользователя, — все это лежит на плечах рендер-движка. UE позволяет без особых усилий создавать и настраивать собственные материалы, конфигурировать объекты освещения, тонко регулировать тени, управлять настройками камеры и оптимизировать процесс отрисовки объектов, чтобы в результате игрок мог получить как можно более качественный визуальный опыт.

Для создания красивого изображения достаточно расставить объекты на сцене и настроить освещение, однако игра требует динамики, создание которой можно начать

с конструирования физики объектов, определяющей, как предметы в мире игры будут падать, сталкиваться, скользить или даже плавать или летать. В большинстве случаев пользователям UE не придется реализовывать математические модели, строить уравнения движения или оптимизировать вычисление столкновений. Вместо этого можно воспользоваться готовыми настройками и функциями, чтобы обеспечить реалистичное поведение игровых объектов.

Игровая логика и возможность чтения данных от пользователя позволяют сделать игру интерактивной. UE поддерживает множество моделей контроллеров, упрощая разработку для разных моделей управления, включая и мобильные устройства. Для задания логики игры движок предоставляет визуальный язык программирования под названием Blueprints, он позволяет интуитивно и наглядно настраивать процессы в игре.

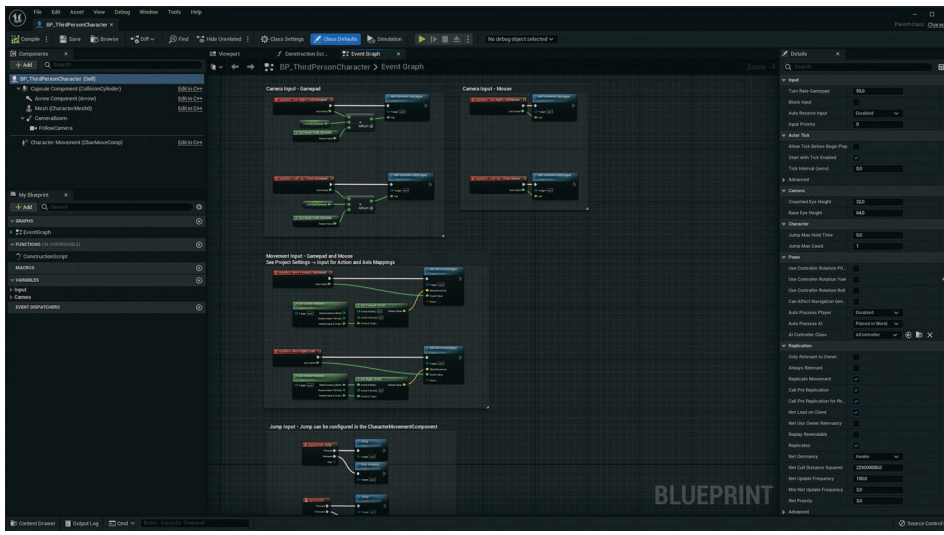

Рисунок 9

Анимация, эффекты частиц, звуковое оформление, визуальный интерфейс и искусственный интеллект также являются источниками динамики и интерактивности. Каждому из перечисленных компонентов в UE соответствует свой редактор. Но это только часть всего того набора инструментов, который предоставляется разработчику. Движок позволяет отслеживать производительность каждой по отдельности составляющей игры, тестировать функциональность на разных платформах, использовать визуальное логирование для записи игровых событий, настраивать локализацию игры на различные языки, выстраивать кинематографические сцены, изменять ландшафт крупных игровых пространств, работать со спрайтами и 2D-анимациями и многое-многое другое. Инструментарий Unreal Engine позволяет даже небольшой команде разработчиков быстро создавать крупные игровые проекты, отвечающие современным стандартам качества.

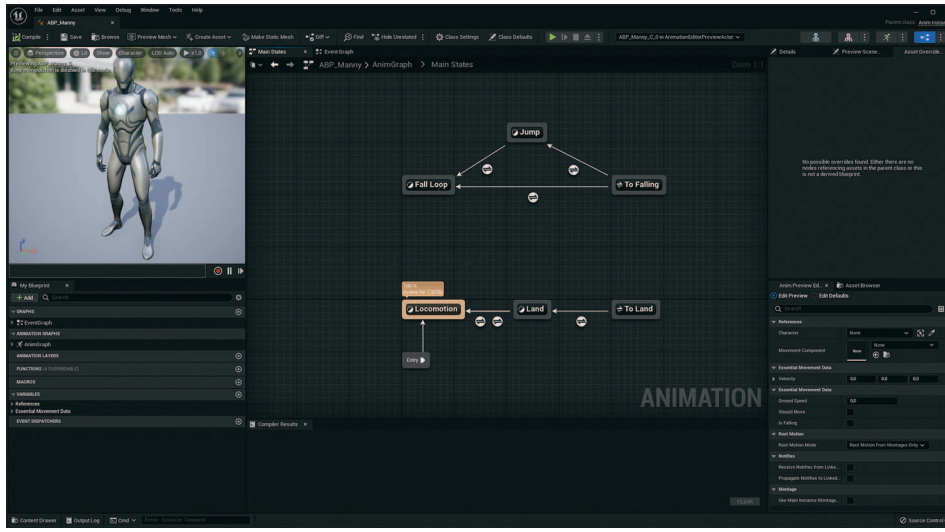

Рисунок 10

Возможности UE огромны, но, чтобы ими воспользоваться, нужны знания и опыт. За знаниями можно обратиться на официальный сайт движка, где имеются курсы, проекты-примеры и видеоматериалы для обучения. Для изучения особенностей UE также подойдет документация, которая наполнена не только описаниями предоставляемых UE функций, но и статьями, посвященными разным аспектам работы с программой. Кроме того, существуют официальные форумы, где пользователи UE обсуждают подходы к решению всевозможных задач или общаются с разработчиками по поводу желаемых нововведений в движке. YouTube-канал UE постоянно пополняется записями обучающих стримов, новостями и прочими полезными материалами. Если же вы захотите углубиться в мельчайшие детали работы движка, то к вашим услугам открытый исходный код программы, доступный для изучения и совершенствования.

ГЛАВА 3

Работа с контентом и ресурсами

Пора переходить от слов к делу: в этой главе мы скачаем и установим UE, научимся создавать проекты и загружать в них разнообразный контент, называемый ассетами. Вы познакомитесь с основными типами ассетов и способами работы с ними в UE.

Установка Unreal Engine и создание первого проекта

Для установки UE5 требуется наличие на компьютере Epic Games Launcher. Если у вас отсутствует эта программа, перейдите на страницу https://www.epicgames.com/store/ru/download и нажмите кнопку «Загрузить». После установки программы авторизуйтесь в ней. Epic Games предоставляет множество вариантов входа в лаунчер, вы можете выбрать для себя наиболее удобный. Если все прошло успешно, должен открыться Epic Games Launcher, который будет выглядеть примерно как показано на рисунке 11.

В боковом меню найдите раздел Unreal Engine и выберите его. После этого откроется окно, содержащее информацию о работе с движком. В этом окне расположено несколько вкладок. Откройте ту, которая называется Library. Отсюда вы сможете установить движок Unreal Engine. На выбор может быть предложена уже установленная версия UE или же можно просто установить новую как показано на рисунке 12, для чего следует кликнуть на кнопку Install («Установить движок»).

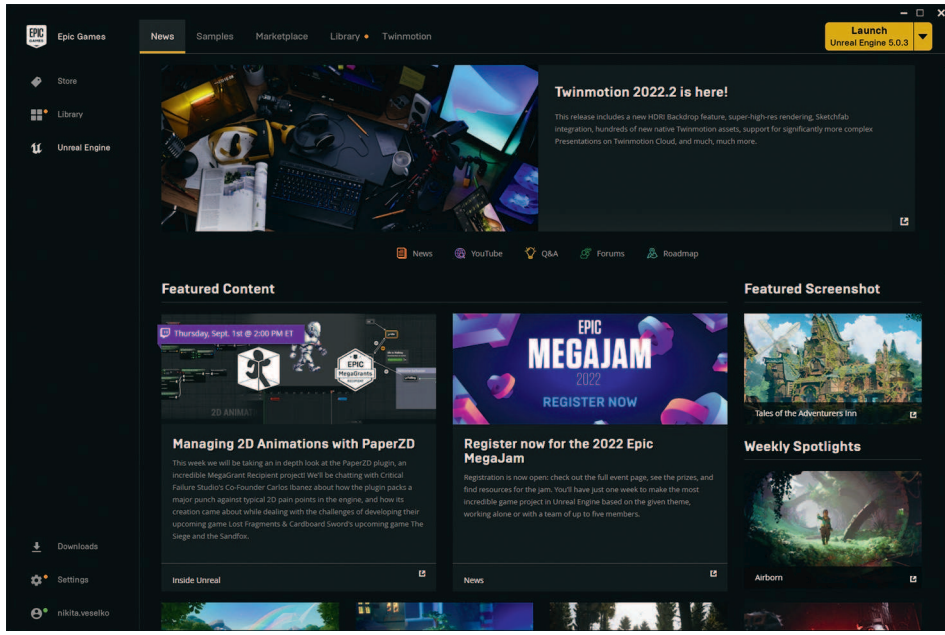

Рисунок 11. Epic Games Launcher

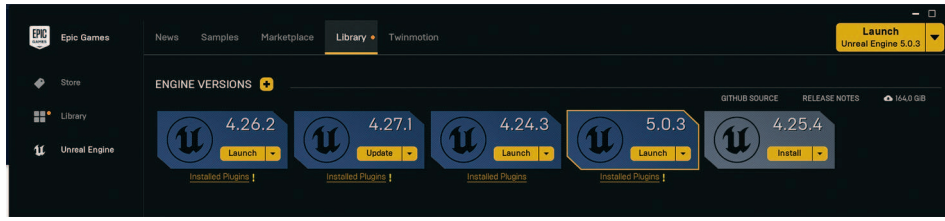

Рисунок 12. Различные версии движка в Epic Games Launcher

В появившемся окне выберите путь, по которому будет установлен движок, и нажмите Install. Вас вернет в окно Epic Games Launcher на вкладку Library. Здесь будет отображаться прогресс установки приложения.

После завершения установки появится кнопка Launch. Нажмите на нее и дождитесь открытия Unreal Project Browser. Появится окно, в котором можно выбрать уже существующий проект или же создать новый. В этой книге мы будем разбирать только категорию Games, она выделена синей рамкой на рисунке 13. Прочие варианты представляют собой преднастроенные для конкретных целей проекты, и в книге они не рассматриваются.

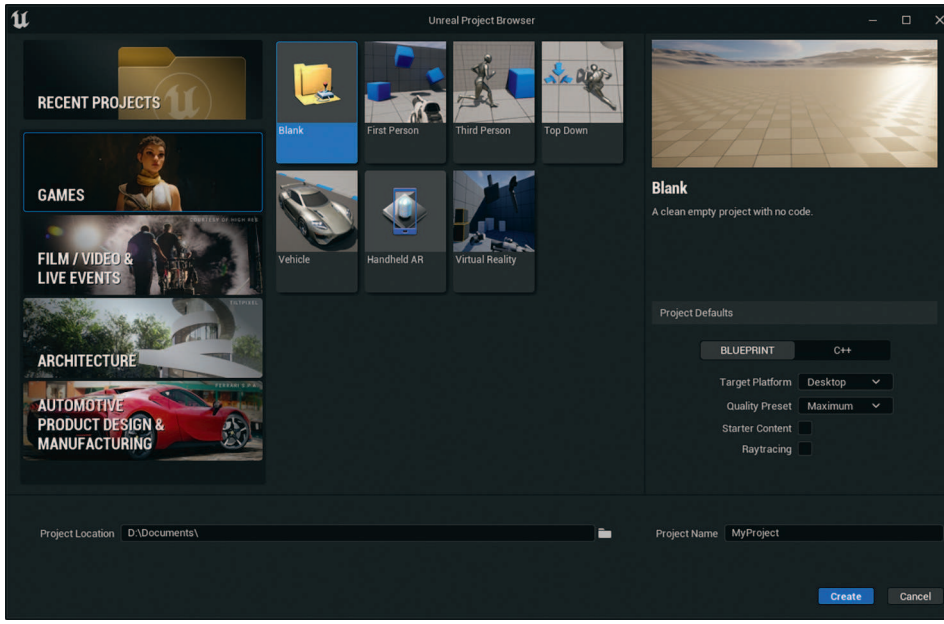

Рисунок 13. Unreal Project Browser: выбор категории проекта

Выбрав категорию Games, вы увидите список доступных для выбора шаблонов. Epic Games предлагает набор различных шаблонных проектов с минимальной логикой для быстрого создания приложения определенного типа. Часть из них будет рассмотрена в книге, а часть останется читателю на самостоятельное освоение. В качестве полигона для испытаний выступит Blank — это полностью пустой проект без каких-либо предустановок.

После выбора шаблона обратим внимание на правую нижнюю сторону, где под заголовком Progect Defaults расположены настройки проекта. Первый пункт предлагает указать язык программирования, на котором вы будете писать логику: Blueprint или C++. В рамках данной книги мы будем использовать только Blueprint. Тем не менее языковая настройка не является жесткой, в процессе разработки вы можете использовать как C++, так и Blueprint.

Дальше можно указать целевую платформу игры. Этот параметр тоже всегда доступен для изменения, но мы будем работать только с вариантом Desktop/Console.

Следующий пункт позволяет выбрать характеристики графики: Maximum Quality или Scalable 2D и 3D. В первом случае графика будет максимально возможного качества, второй же вариант понизит качество рендера. Параметр доступен для изменения в любой момент.

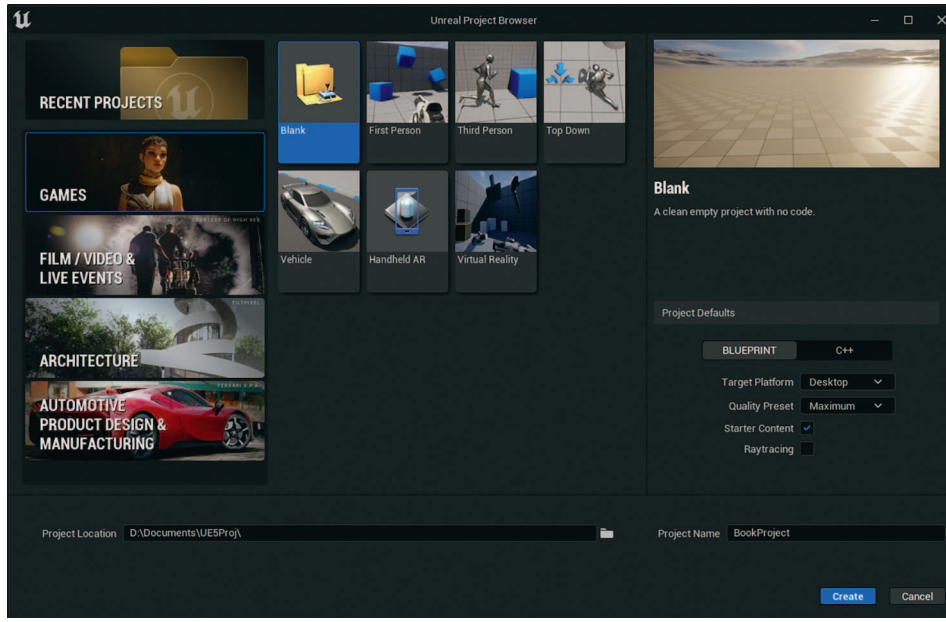

Рисунок 14. Unreal Project Browser: выбор шаблона проекта

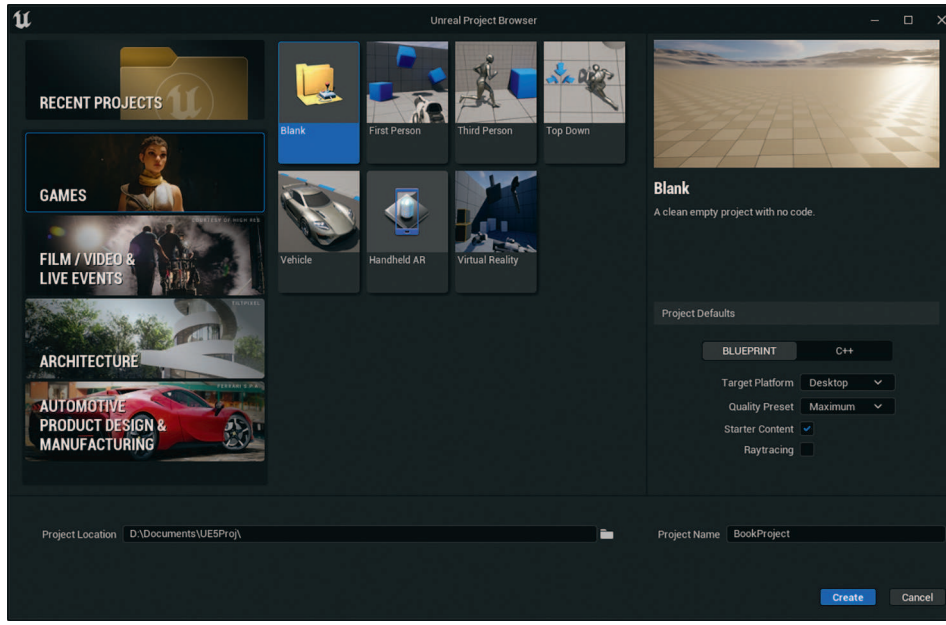

Рисунок 15. Unreal Project Browser: базовые настройки проекта

Следующая настройка позволяет задействовать в проекте стартовый контент. В нем содержится несколько моделей, материалов и эффектов. Рекомендуем отказаться от этой возможности, так как место на диске будет занято, а контент может вам и не пригодиться. При этом он доступен для добавления в проект в любой момент.

Последняя настройка позволяет задействовать в проекте поддержку **Raytracing**. Эта технология относится к визуализации и техническому арту, ее освоение является следующим уровнем овладения движком, поэтому в книге мы не будем касаться этой темы. **Для проектов, рассмотренных в книге, будем выбирать значение Raytracing disabled.**

И последнее, что требуется сделать, — это указать путь, по которому будет сохранен проект. После выбора директории нажмите **Create Project** и подождите, пока загрузится проект. Длительность загрузки зависит от мощности вашего ПК, но даже на очень производительных устройствах он займет продолжительное время.

Интерфейс редактора

После создания проекта вы увидите интерфейс основного редактора. Он состоит из нескольких блоков, давайте рассмотрим каждый из них по порядку.

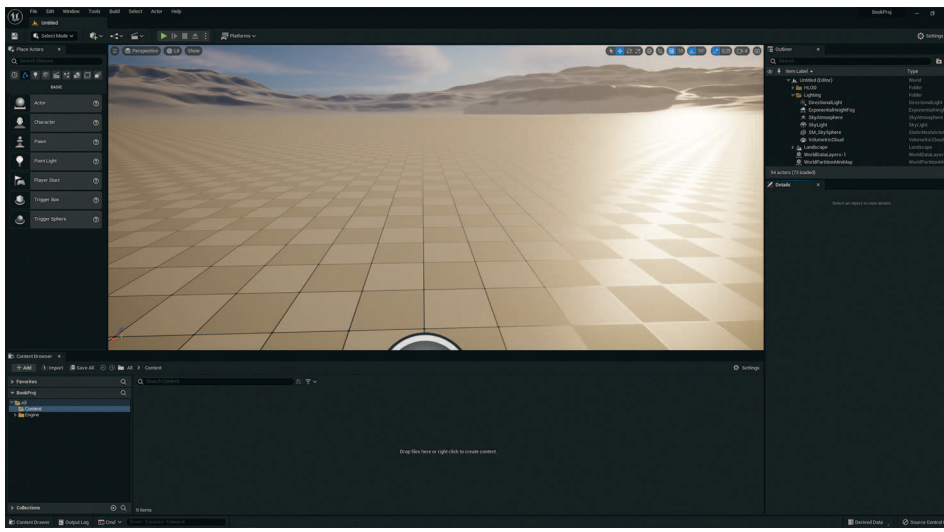

Рисунок 16. Unreal Engine 5 Editor

1. **Viewport** — окно просмотра. Здесь можно просматривать текущий уровень. Для перемещения по нему зажмите правую кнопку мыши и воспользуйтесь

клавишами WASDQE. Клавиши WASD отвечают за стандартное перемещение вперед, влево, назад и вправо, в то время как QE — за передвижение вниз и вверх.

2. **Place Actors** — окно акторов. Тут можно выбрать такие готовые объекты, как освещение, кубы, сферы и другие и расставить их на сцене. Для этого просто перетащите требуемый объект во Viewport.
3. **Content Browser** — окно просмотра контента. В нем будут храниться все ассеты проекта. Работа этого окна аналогична работе проводника в любой операционной системе.
4. **World Outliner** — здесь отображаются все объекты, которые находятся на уровне. Этим окном удобно пользоваться для поиска нужных элементов и создания иерархии с помощью папок.
5. **Details** — окно с подробной информацией об объекте. Здесь отображаются параметры выбранного объекта (с уровня).
6. **World Settings** — панель, предназначенная для настройки параметров уровня.

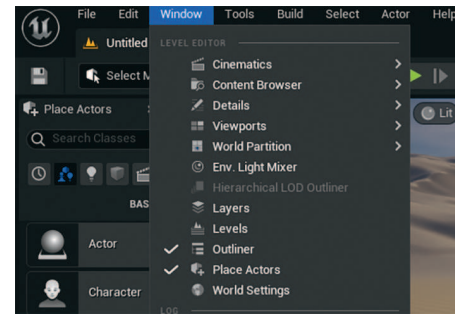

Рисунок 17. Добавление недостающего блока

Если какого-либо блока в вашем проекте не хватает, его всегда можно включить в панели Window путем нажатия на нужный элемент (рисунок 17).

Стандарты наименования ассетов

Прежде чем мы перейдем непосредственно к работе с контентом, давайте изучим правила именования ассетов в UE5.

Формат выглядит следующим образом: **Prefix_Assetname_Index_Suffix**, где:

— **Prefix** — краткое обозначение типа ассета (одна-две буквы);

— **Assetname** — имя ассета, кратко описывающее назначение ассета;

— **Index** — порядковый номер ассета; может не указываться, полезен для случаев наличия вариаций одних и тех же ассетов;

— **Suffix** — краткое обозначение вариации типа ассета (одна-две буквы); не у всех ассетов есть подтипы, но если есть, они должны быть обозначены.

Пример: **T_Wood_D** — диффузная (D) текстура (T) дерева.

Очень важно усвоить эти правила и применять их, иначе работать над проектом, как в одиночку, так и в команде, будет тяжело.

Рекомендации Epic Games по именованию ассетов (https://docs.unrealengine.com/5.0/en-US/recommended-asset-naming-conventions-in-unreal-engine-projects/)

Контент-браузер

Контент-браузер (Content Browser) — это инструмент для управления файлами пакетов, их организации, создания и выполнения действий с ассетами[3].

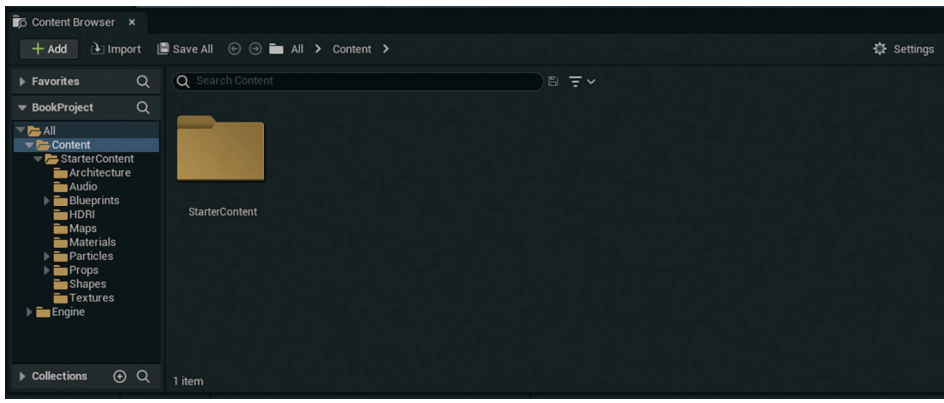

Рисунок 18. Внешний вид Content Browser в Unreal Editor

Контент-браузер предназначен для создания, импорта, организации, просмотра и изменения ассетов. Он также предоставляет возможность управлять папками с контентом и выполнять другие полезные операции с ассетами (например, переименование, перемещение, копирование или просмотр ссылок).

[3] Практически любые цифровые ресурсы, которые используются для создания игрового контента.

Рисунок 19. Варианты значка Browse

Именно благодаря Content Browser UE способен быстро определить местоположение ассетов, для этого нужно нажать на значок лупы (Browse), расположенный рядом с ассетом или в его панели инструментов (в браузере откроется папка с выделенным ассетом). Более того, контент-браузер помогает организовать взаимодействие всех ассетов приложения.

Импорт

Наиболее удобный способ создания контента состоит в использовании внешних инструментов и последующем импорте в движок получившегося результата.

Импортировать ассеты можно несколькими способами.

— Перетащите нужный ассет из файлового проводника операционной системы в контент-браузер.

— Нажмите правой кнопкой мыши (ПКМ) на нужную папку и выберите опцию Import to [path], далее укажите ассет, который хотите импортировать в проект.

— В окне контент-браузера нажмите на кнопку Add/Import и выберите опцию для импорта ассета (Import to [path]; импорт будет произведен в открытую в контент-браузере папку).

Для некоторых ассетов (например, мешей) при импорте нужно дополнительно указать значения некоторых настроек, список с ними будет открыт в новом окне при импорте.

СОХРАНЕНИЕ ИМПОРТИРОВАННЫХ АССЕТОВ

Несохраненные импортированные ассеты помечаются в контент-браузере звездочкой в левом нижнем углу. Сохранить ассеты можно путем нажатия кнопки Save All в контент-браузере (появится диалоговое окно для выбора контента, выберите тот ассет, который нужно сохранить). Также можно нажать комбинацию клавиш **Ctrl+S** для сохранения ассета, выделенного в контент-браузере

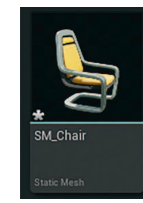

Рисунок 20. Несохраненный объект

ПРОВЕРКА ИМПОРТИРОВАННЫХ АССЕТОВ

Возьмите за правило обязательно проверять, что все ассеты при импорте действительно были скопированы на жесткий диск. Для этого откройте папку с импортированным ассетом и убедитесь, что все файлы на месте. Для перехода в нужную папку нажмите правой кнопкой мыши на файле и выберите опцию **Show In Explorer**. В открывшейся папке у каждого импортированного файла должен быть «близнец» с расширением .uasset.

ИМПОРТ СТАРТОВОГО КОНТЕНТА

При создании скриншотов к этой главе использовались ассеты из набора Starter Content и шаблона Third Person. При импорте необходимых элементов в свой проект пользуйтесь информацией из этого раздела.

После создания проекта в него можно дополнительно импортировать как стартовый контент, так и любой другой из различных шаблонов движка. Для этого нажмите на кнопку **Add** в Content Browser и выберите опцию **Add Feature or Content Pack**. В открывшемся меню укажите нужный контент и добавьте его в проект нажатием на кнопку Add to Project.

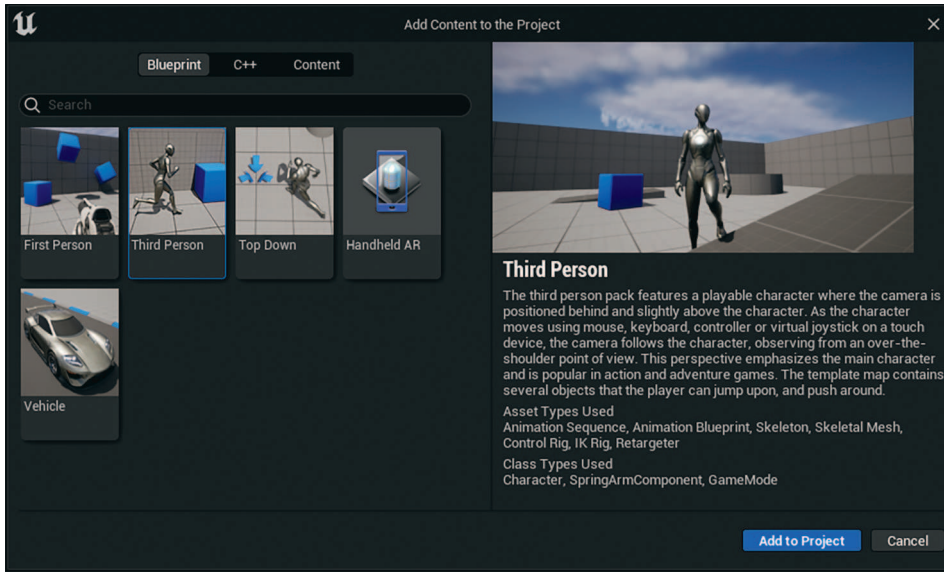

Рисунок 21. Окно добавления контента

Текстуры

Текстуры — это изображения, которые используются в Материалах (Material) путем наложения на поверхности, к которым применяется материал. Текстуры можно использовать в том виде, в каком они есть, например текстуры Base Color, или же для различных вычислений можно взять только значения (цвет) пикселей текстур (texels) внутри материала (это называется маской). Кроме того, текстуры можно использовать и напрямую, без материалов, например в HUD.

Как правило, текстуры создаются во внешних инструментах (например, в Photoshop или Blender) и импортируются в движок посредством Content Browser. Однако некоторые текстуры можно генерировать и с помощью движка, например Render Textures. Такие текстуры обычно берут некоторую информацию со сцены и запекают ее на текстуру для последующего использования (этот процесс называется texture baking).

В материале могут использоваться несколько текстур, каждая из которых имеет свое назначение. Например, обычный материал может иметь текстуры с базовым цветом (Base Color), карту отражений (Specular) и карту нормалей (Normal Map). В дополнение могут использоваться текстуры свечения (Emissive) и шероховатости (Roughness), которые хранятся в альфа-каналах указанных ранее текстур. Все базовые особенности текстур мы разберём в нижеследующих разделах этой главы.

Все эти текстуры могут иметь одну разметку, однако их цвет будет различаться в зависимости от типа текстуры.

СОЗДАНИЕ ТЕКСТУР

При создании текстур для материалов важно учитывать, что значение каждой стороны должно быть степенью двойки (например, будет корректным разрешение 64 × 256). Это нужно для правильной обработки текстур в реальном времени.

В UE разрешение текстуры меняется в зависимости от расстояния между ней и камерой игрока. По мере отдаления/приближения текстуры ее размер уменьшается/увеличивается в два раза. Этот процесс называется миппингом или множественным отображением (**mip mapping / mipping**).

ОСНОВНЫЕ ВИДЫ ТЕКСТУР

В UE5 существует несколько видов основных текстур для мешей:

1) диффузные текстуры;
2) карты нормалей;
3) маски.

Диффузные текстуры

Диффузные текстуры (base color/albedo/diffuse texture) содержат описание цвета и являются наиболее часто встречающимся типом текстур.

Как правило, диффузные текстуры не содержат информации о тенях и освещении. В альфа-канале[4] диффузных текстур часто хранят информацию о прозрачности.

Рисунок 22. Три разные текстуры, используемые для создания материала кирпичной стены. Слева направо: диффузная текстура, маска, карта нормалей

Карты нормалей

Карты нормалей[5] (**normal maps/normals**) позволяют создавать иллюзию рельефности и детальности текстур.

Нормали описывают направление, в котором источники света должны взаимодействовать с поверхностью, придавая ей таким образом объемный вид.

Чтобы понять, откуда берется информация об освещении пикселей текстуры, поговорим о цветовых каналах. Каждый канал карты текстуры (красный, зеленый и синий) используется для представления различных углов направлений поверхности.

- **Красный** — ось X, или направление света слева направо.
- **Зеленый** — ось Y, или направление сверху вниз.
- **Синий** — ось Z.

Важно отметить, что зеленый канал может быть отражен (зависит от программного обеспечения, с помощью которого создавалась карта нормалей). В таком случае после импорта в движок текстуры нужно в ее свойствах (их можно открыть двойным кликом по текстуре в контент-браузере) выбрать опцию **Flip Green Chanel**, чтобы текстура начала отображаться корректно.

Как правило, во время импорта движок автоматически распознает и настраивает нормали. Если же этого не происходит (текстура отображается некорректно), необходимо в свойствах текстуры отключить **sRGB** и установить корректную группу текстур для нормалей (свойство **Texture Group**).

---

[4] Текстура содержит информацию в четырех каналах (R, G, B, A); в обычных текстурах первые три канала хранят информацию о цвете, а последний, альфа-канал, содержит информацию о прозрачности.

[5] Традиционно нормалью называется вектор, перпендикулярный к поверхности в данной точке.

Маски

Маски используются для хранения информации, не имеющей отношения к цвету текстуры. **Маска** — это одноканальная текстура (то есть используется один цветовой канал). Таким образом, в отдельной текстуре можно хранить до четырех масок одновременно (R, G, B, A).

Также маски можно хранить в одном из каналов других текстур (например, в альфа-каналах нормали или диффузной текстуры).

РЕДАКТОР ТЕКСТУР И ИХ СВОЙСТВА

Просматривать и изменять свойства текстуры можно в редакторе текстур (открывается двойным кликом по текстуре в контент-браузере).

Свойства текстуры делятся на шесть категорий.

- **Compression** — позволяет указать значение сжатия текстуры.
- **Texture** — показывает информацию об исходном файле текстуры и содержит параметры, позволяющие настроить вид текстуры (цвет отступа (**Padding Color**) и метод тайлинга (**Tiling**).
- **File path** — информация о времени и источнике импортирования текстуры.
- **Adjustments** — настройка цветовых параметров текстуры (оттенок, яркость и т. п.).
- **Level Of Details (LOD)** — параметры детализации текстуры (такие как настройки мипмаппинга).
- **Compositing** — позволяет добавить текстуру для определения шероховатости (Roughness) данной текстуры и предоставляет некоторые возможности для ее настройки.

Подробнее о свойствах текстур можно прочесть в соответствующем разделе документации движка (https://docs.unrealengine.com/4.27/en-US/RenderingAndGraphics/Textures/Properties/)

ИМПОРТ

UE поддерживает следующие форматы текстур: .tga, .psd, .tiff, .bmp, .float, .pcx, .png, .jpg, .dds и .hdr.

Импорт текстур в UE5 не имеет настроек, однако у текстур есть несколько важных параметров, которые следует выставлять сразу. Если открыть текстуру в редакторе, на панели Details можно задать следующие важные параметры:

1. **Compression Settings** — настройки сжатия текстур. На первых порах вы можете оставить значение по умолчанию, но чем глубже вы будете погружаться в визуальную составляющую, тем серьезнее отнеситесь к изучению остальных параметров, так как это напрямую влияет на оптимизацию;
2. **Min Gen Settings** — настройка генерации минмап. Хорошей иллюстрацией того, что представляют собой минмапы, является любая 2D-игра, в которой можно приближаться или удаляться от объектов. Чем дальше от объекта вы находитесь, тем меньшее разрешение имеет текстура — и, к слову, это тоже параметр оптимизации;
3. **Texture Group** — текстурная группа. Эта настройка влияет на параметры LOD (**Level Of Details** — уровни детализации). Например, при выборе значения UI движок не будет использовать минмапы для этой текстуры, так как для интерфейсов они не применяются.

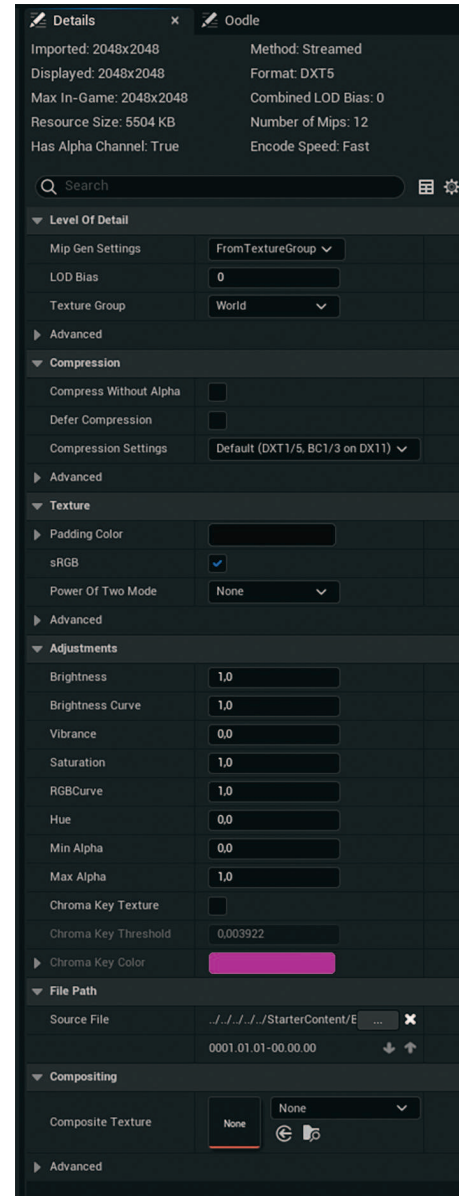

Рисунок 23. Окно Details текстуры

Материалы

В основном текстуры используются при создании материалов. Материал — это набор текстур и параметров, определяющих поведение света на поверхности. Проще всего думать о материале как о «краске», которая наносится на объект. Но это определение будет грубоватым, потому что фактически материал определяет тип поверхности, из которой создается объект. Вы можете указать его цвет, степень сияния, настроить возможность видения сквозь объект и многое другое.