Оглавление

Предисловие от издательства

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте *www.dmkpress.com*, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу *dmkpress@gmail.com*; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в издании новой книги, заполните форму на нашем сайте по адресу *http://dmkpress.com/authors/publish_book/* или напишите в издательство по адресу *dmkpress@gmail.com*.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от непонимания текста и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу *dmkpress@gmail.com*, и мы исправим их в следующих тиражах.

Нарушение авторских прав

Пиратство в сети Интернет по-прежнему является насущной проблемой. Издательство «ДМК Пресс» очень серьезно относится к вопросам защиты авторских прав и лицензирования. Если вы знаете о незаконной публикации какой-либо из наших книг в сети Интернет, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу *dmkpress@gmail.com*.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Об авторах

Энтони Дэвис (Anthony Davis) – старший технический художник компании Accelerate Solutions из Орландо, штат Флорида. До Unity он работал в нескольких сферах: от военного ветерана и тренера по гимнастике до открытия студии разработки инди-игр и других сфер деятельности. Его работа в независимой и внештатной работе позволила многим изучить все аспекты разработки игр. В свободное время предпочитает играть в Dungeons and Dragons, заниматься искусством и планировать свой следующий проект.

Хочу поблагодарить свою семью за предоставленное пространство для работы над этим проектом. Тика (Tica), спасибо, что заставляешь все двигаться. Джерин (Jehryn) и Кембер (Kember), спасибо за понимание, когда я работал над книгой вместо того, чтобы поиграть с вами в игры. Мосс (Mohss), работая над книгой, я думал о тебе без остановки. Очень надеюсь, что книга поможет вам при создании собственных игр.

Трэвис Батист (Travis Baptiste): художник, ученый на протяжении всей жизни и геймер-любитель — вот лишь некоторые из терминов, которые можно использовать для описания Трэвиса. После службы в армии Трэвис поступил в Full Sail University, где изучал игровое искусство. После выпуска в 2015 году он работал внештатным 3D-моделлером, одновременно обучая своих двоих детей на дому. Между обслуживанием клиентов и личными проектами Трэвис поддерживал свои 3D-таланты в нескольких программах.

Моя семья, спасибо вам за понимание, когда мне приходится работать. Моим сыновьям: я надеюсь, что этот опыт еще больше вдохновит вас обоих на достижение собственных целей, несмотря на трудности. Моей жене Альмире (Almira) спасибо за поддержку. Работать по ночам стало намного легче с помощью, которую ты оказала с детьми. Спасибо, Дэвид Нгуен (David Nguyen), за жизненные ориентиры и за то, что ты рядом. Спасибо, Энтони (Anthony), за предоставленную возможность. Я благодарен, что у меня была возможность работать с вами над проектом.

Рассел Крейг (Russel Craig) – старший инженер-программист в Unity Technologies. На момент написания книги имеет 10-летний опыт профессионального моделирования на Unity в таких областях, как разработка приложений, аппаратное и микропрограммное обеспечение продукта, моделирование сенсорных систем, симуляция обучения в медицине и разработка AR/VR. Спикер конференций Unite и мастер на все руки в Unity. Свободное время проводит с женой и детьми, увлекается сборкой компьютеров, автомобилей и просто играет в видеоигры.

Я хотел бы поблагодарить мою семью, друзей и коллег за то, что они мирятся с моим напряженным графиком.

Райан Станкел (Ryan Stunkel) – профессиональный звуковой дизайнер видеоигр, руководитель собственной студии-подрядчика Blipsounds в Остине, штат Техас. Помимо Blipsounds, Райан является учителем и наставником сообщества звукорежиссеров на YouTube-канале Blipsounds. Путешествуя по миру, делился своими знаниями о звуковом дизайне видеоигр для компаний Google, PAX и Schools.

О рецензентах

Гиртс Кестерис (Ģirts Ķesteris) – руководитель студии @HyperVR Games, а также внештатный инженер-программист XR/Unity в Ubiquity Inc. Лектор по разработке интерактивных 3D-сред (включая игровой движок Unity) в Vidzemes Augstskola (Видземский университет прикладных наук) (по совместительству) и независимый разработчик игр в NYAARGH! (частный предприниматель). Имеет большой опыт работы с Unity в нескольких компаниях.

Спасибо моей жене за терпение и поддержку!

Монтика «Тика» Сансук (Montica «Tica» Sansook) – серийный предприниматель азиатско-американского происхождения, спикер, художник по играм и блогер. Тика является соучредителем Defy Esports Bar, места проведения ночных киберспортивных клубов, и Defy Games, независимой студии по разработке игр. Она дважды окончила университет Full Sail со степенью магистра наук в области индустрии развлечений и степенью бакалавра наук в области игрового искусства. Ее обширный и разнообразный карьерный опыт варьировался от креативного развития бренда и бизнеса в области игр до управления сообществом блогеров и профессиональных киберспортивных организаций. Тика получает огромную радость и удовлетворение от поддержки других посредством наставничества, продвижения разнообразия и консультирования. Ей нравится мотивировать людей к раскрытию их потенциала и подчеркивать достижения своих коллег. В своем жизненном пути она хочет оказать положительное влияние на развитие экосистемы игровой индустрии в лучшую сторону.

Введение

Книга представляет собой детальный обзор дизайна и разработки трехмерной игры-головоломки в Unity. Мы займемся дизайном, созданием и реализацией персонажей, окружения, пользовательского интерфейса, звука и игровой механики.

Кому предназначена книга?

В первую очередь данная книга для тех, кто заинтересован в создании 3D-игр, но еще не начал свой путь. Мы изучим все – от самой базы до продвинутых техник.

Также мы поможем всем, кто уже начал свой путь и желает изучить новые для себя аспекты разработки игр, поскольку в книге мы охватываем широкий спектр навыков и знаний.

О чем эта книга

Часть I – Планирование и проектирование

*Г*лава 1 «Введение в трехмерное пространство» знакомит с трехмерной терминологией и начальным жаргоном того, через что будет проходить книга.

Глава 2 «Дизайн и прототип» познакомит пользователя с кроличьей норой дизайна и заканчивается установкой Unity для создания вашего первого проекта.

Глава 3 «Программирование» закладывает основы программирования. Эта глава опирается на мощь C# (C Sharp), объясняя основы логики и первоначальное использование Visual Studio.

Часть II – Сборка

В главе 4 «Персонажи» рассматривается проектирование 3D-персонажей, а также обдумывается, как они будут использоваться для риггинга и анимации.

Глава 5 «Окружающая среда» проведет вас через размышления об окружающей среде для вашей игры, а также о том, что мы сделали для ее проектирования и создания.

Глава 6 «Взаимодействия и механика» посвящена тому, как следует думать о механике и что представляет собой взаимодействие для пользователя, а также охватывает программирование, необходимое для взаимодействия в нашем проекте.

Глава 7 «Взаимодействие Rigidbody и физики» добавляет сложности к взаимодействию с физикой и более сложные концепции программирования.

В главе 8 «Пользовательский интерфейс и меню» рассказывается о компоненте холста Unity и о том, как разрабатывается общий игровой интерфейс в любом проекте.

Часть III – Полировка и уточнение

В главе 9 «Визуальные эффекты» рассказывается, как можно работать с системами визуальных эффектов, чтобы создать дополнительную эмоциональную связь с вашим миром. Это делается путем объяснения основ рендеринга и окружающих его систем.

Глава 10 «Звуковые эффекты» рассказывает о звуковых системах в Unity, а также закладывает прочную основу звукового дизайна.

В главе 11 «Сборка и тестирование» рассказывается, как Unity создает окончательный исполняемый файл игры, и объясняются методы тестирования для устранения ошибок, которые можно исправить с целью сделать продукт лучше.

Глава 12 «Последние штрихи» выглядит как набор полезных инструментов для того, чтобы сделать вашу игру настолько идеальной, насколько это возможно. Там мы полируем наш проект, что включает в себя специальные системы частиц, освещение, доработка арта и улучшенную звуковую полировку.

Дополнительная глава «Бонус: другие инструменты Unity» — это глава, в которой рассматриваются некоторые сервисы, которые Unity может предложить на тот случай, если данная книга вдохновит вас на работу над проектом, который мы не смогли охватить, например над многопользовательской игрой или смешанной реальностью.

Чтобы получить максимальную отдачу от этой книги

- Смотрите на эту книгу не как на учебник, а как на множество в одном месте инструментов, используемых для разработки 3D-игры. Мы рассмотрим только несколько простых примеров. Избавьтесь от логики при чтении, чтобы применить ее к своим проектам в максимально возможной степени.
- Будьте готовы делать собственные заметки по рассматриваемым темам. Мы немного усложняем программирование в части физики.
- Задавайте вопросы в канале Discord, который прикреплен через QR-код.

Загрузите файлы примеров кода

Набор кодов для книги размещен на GitHub по адресу *https://github.com/PacktPublishing/Unity-3D-Game-Development*. У нас также есть другие наборы из нашего богатого каталога книг и видео, доступных по адресу *https://github.com/PacktPublishing/*. Загляните!

Загрузите цветные изображения

Мы также предоставляем PDF-файл с цветными изображениями скриншотов и диаграмм, использованных в этой книге. Вы можете скачать его здесь: *https://static.packt-cdn.com/downloads/9781801076142_ColorImages.pdf*.

Организация текста

В этой книге используется ряд текстовых соглашений.

`CodeInText`: обозначает слова из кода в тексте, имена таблиц базы данных, имена папок, имена файлов, расширения файлов, пути, пустые URL-адреса, пользовательский ввод и дескрипторы Twitter. Например: «Смонтируйте загруженный файл образа диска `WebStorm-10*.dmg` как другой диск в вашей системе».

Блок кода устанавливается следующим образом:

```
void OnStartGameButtonPressed()
    {
        SetPlayerEnabled(true);
        Cursor.lockState = CursorLockMode.Locked;
        Cursor.visible = false;
        this.gameObject.SetActive(false);
    }
```

Полужирный: обозначает новый термин, важное слово или слова, которые вы видите на экране. Например, слова в меню или диалоговых окнах также появляются в тексте таким образом, допустим: «Выберите **System info** в панели **Administration**».

Предупреждения или важные примечания выглядят так.

Советы и рекомендации выглядят следующим образом.

Связь с нами

Отзывы наших читателей всегда приветствуются.

Общий отзыв: напишите на почту *feedback@packtpub.com* и укажите название книги в теме сообщения. Если у вас есть вопросы по какому-либо аспекту этой книги, пожалуйста, напишите нам по адресу *questions@packtpub.com*.

Исправления: хотя мы приложили все усилия, чтобы обеспечить точность нашего контента, ошибки случаются. Если вы нашли ошибку в этой книге,

мы будем признательны, если вы сообщите нам об этом. Пожалуйста, посетите *http://www.packtpub.com/submit-errata*, нажмите **Submit Errata** и заполните форму.

Пиратство: если вы столкнетесь с незаконными копиями наших работ в любой форме в интернете, мы будем признательны, если вы сообщите нам адрес или название веб-сайта. Пожалуйста, свяжитесь с нами по адресу *copyright@packtpub.com* со ссылкой на материал.

1

Введение в трехмерное пространство

Добро пожаловать!

Мы рады, что вы присоединились к нам в этом путешествии, чтобы изучить основы разработки 3D-игр. Для начала мы познакомим вас с командой, написавшей эту книгу.

- **Трэвис Бапист** (Travis Bapiste) (3D-художник) руководил оформлением, смоделировал каждую модель в игре, сделал риг персонажа и помог определить дизайн истории.
- **Рассел Крейг** (Russel Craig) (старший инженер по программному обеспечению) создал скрипты для механик игры.
- **Райан Станкел** (Ryan Stunkel) (звуковой дизайнер) создал и реализовал все звуки в проекте.
- **Энтони Дэвис** (Anthony Davis) (старший технический художник) написал книгу, руководил проектом, создавал эффекты и шейдеры и полировал наш проект.

Мы извлекли лучшее из нашего коллективного опыта за более чем 50 лет (четыре мозга за каждой страницей этой книги), каждый день превращался в американские горки (слишком много веселья!). На написание книги потребовалось более шести месяцев и две редакции всей книги (а также сотни GIF-файлов, которыми мы обменялись в процессе), чтобы включить наиболее подходящие практические примеры, объясняющие новые концепции и, что наиболее важно, предлагающие подход к обучению, который работает. В конце концов, мы считаем, что нам удалось создать книгу, которая определила бы траекторию нашей карьеры в разработке игр и продвинула нас вперед как минимум на 3–5 лет.

Наша книга снабдит вас всеми инструментами, которые понадобятся, чтобы начать создавать игры; однако, чтобы превратить идеи в творения, вам может понадобиться дополнительная поддержка и совет на пути.

На данном моменте наш сервер Discord вступает в игру. Он вводит элемент интерактивности, чтобы мы могли общаться, вместе читать книгу и обсуждать ваши проекты 3D-игр. На нашем Discord-канале мы доступны почти всегда,

чтобы помочь вам с легкостью пройти книгу, поэтому, пожалуйста, не стесняйтесь заходить, чтобы поздороваться и задать любые вопросы!

Не забудьте кратко написать о себе в разделе *#introduce-yourself*, когда присоединитесь: *https://packt.link/unity3dgamedev*.

Ну что же, давайте начнем!

Цель книги

Наша цель – дать каждому читателю возможность сформировать правильное мышление о 3D-играх, а затем показать все шаги, которые мы предприняли, чтобы вы смогли создать свои собственные проекты. Абсолютный новичок может работать с этой книгой, однако чем дальше, тем сложнее вам могут показаться некоторые темы. Даже если это случится, то это лишь шаги к мастерству в разработке игр. Основная целевая аудитория этой книги – те, кто уже имеет некоторые знания в области разработки игр, хотя, независимо от вашего опыта, мы надеемся создать для вас увлекательное путешествие по обучению. Концепции с персонажами, программированием, шаблонами проектирования и многим другим, которые мы рассмотрим, будут усложняться.

Чтобы извлечь максимальное количество пользы от книги, рекомендуем вам следующий подход.

- Прочитав главу, сделайте паузу, чтобы тщательно обдумать прочитанное.
- Если что-то оказалось непонятным, взгляните на наши проекты в GitHub, потому что практический взгляд может ответить на возникшие вопросы. В случае, если все равно останутся сомнения, то, как вариант, можно воспользоваться поиском в интернете.
- Если вдруг произошли какие-то ошибки в проекте, отправьте мне сообщение в Discord или обратитесь за помощью к коллегам на сервере сообщества – ссылка опубликована выше.
- Возможно, освежить взгляд, пропустив камень преткновения и пройти дальше, а затем снова к нему вернуться, также поможет вам.

Данный подход поможет вам разобраться с подводными камнями на пути; как только вы почувствовали себя в тупике, обратитесь за помощью к коллегам. Проблемы, с которыми вы, возможно, столкнетесь, обязательно у кого-то уже были и еще будут. Их решения и обсуждение в нашем Discord укрепляют общие знания всего сообщества на любых уровнях.

Концепция этой книги создана для данного подхода, соответственно, в первую очередь нужно понять, почему мы сделали так, как мы сделали. Нам потребуется некоторое время, чтобы изучить основы интерфейса Unity, но в дополнение будет полезным воспользоваться онлайн-ресурсами по обучению, их очень много.

Обратите внимание, что вы не найдете здесь детализированной информации о том, как моделировать персонажей, риггить или анимировать их. Мы очень мало будем говорить об этом процессе, поэтому это пойдет на вашу самостоятельную тренировку. Мы расскажем, почему создали нашего персонажа именно так, как мы это сделали, с целью помочь вам научится делать что-то похожее. В проекте есть все уже готовые анимации и синематики, поэтому финальные продукты доступны для ознакомления вместе с результатами нашей работы. Такой подход – отличный способ учиться, и мы научим вас, почему все делается именно так, а не иначе. Таким образом, вы увидите конечный результат, но также будет прекрасно, если вы проявите творческий подход и придумаете собственные идеи для дизайна. И не бойтесь самостоятельно экспериментировать в новых для вас софтах по ходу чтения, в этом нет никаких ограничений.

Наконец, прежде чем погрузиться в то, ради чего мы здесь собрались, мы хотели бы посоветовать вам открыть репозиторий GitHub, перейти в папку `Builds` и поиграть в игру для себя. Это поможет вам увидеть то, что наша небольшая команда собрала воедино в конечный результат. Поиграв в нее сейчас, вы сможете визуализировать то, через что мы прошли, соответственно, будет легче представлять результат во время чтения.

Темы, в которые мы погрузимся в этой главе:

- основы 3D,
- основные концепции Unity,
- интерфейс Unity.

Начнем со знакомства с основными компонентами разработки 3D-игр.

Основы 3D

В этом разделе мы рассмотрим базовое понимание работы с 3D – от систем координат до визуализации 3D-модели, чтобы вы ясно понимали основы перед продвижением к сложным разделам. Здесь вы получите четкое представление о том, как Unity отображает элементы.

Система координат

В каждом 3D-софте своя система координат. Unity имеет левостороннюю систему координат, в которой `+y` направлен вверх. На рис. 1.1 представлено сравнение левосторонней и правосторонней систем.

Работая в данных системах, вы будете видеть положение объекта относительно трех значений:

(0, 100, 0).

Так мы обозначили (x, y, z) соответственно. Применять такой синтаксис будет полезным, поскольку в программировании используется похожий синтаксис положения в скриптах, как в примере. Как правило, говоря о положении, имеется ввиду `transform` (преобразование) внутри любого используемого **Создателя цифрового контента** (Digital Content Creator – DCC). В Unity под преобразованием (`transform`) предполагается положение (position), поворот (rotation) и масштаб (scale) объектов.

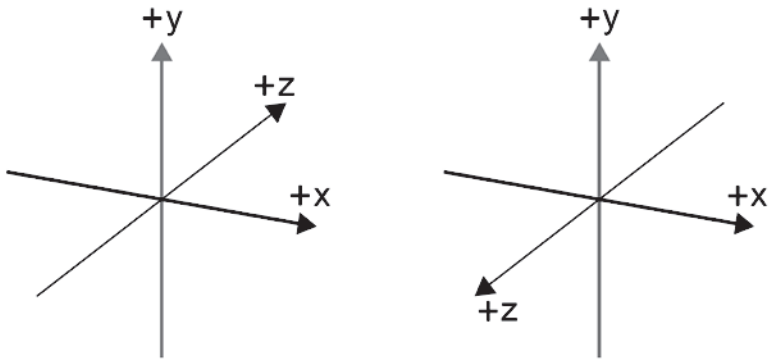

Рис. 1.1. Сравнение систем координат

Таким образом, мы выяснили, что (x, y, z) – это мировые координаты с начальным положением 0, соответственно, (0, 0, 0). Ниже на рис. 1.2 мы видим общую точку всех трех направлений, которая имеет координаты (0, 0, 0) в мире. Куб имеет собственный `transform`, включающий в себя положение, вращение и масштабирование. Обратите внимание, что в первую очередь `transform` обозначает локальное положение, поворот и масштаб объекта, а мировые `transform` рассчитываются исходя из этого в соответствии с их иерархией.

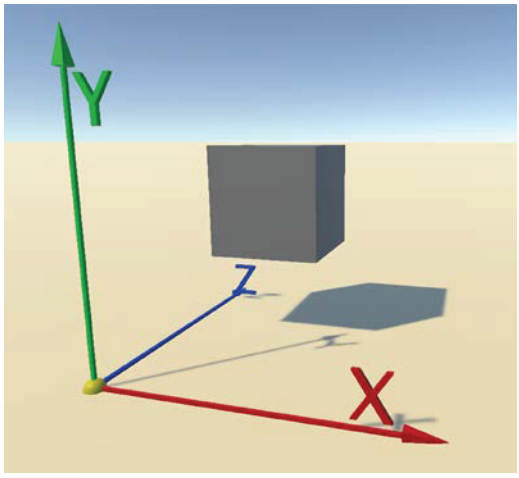

Рис. 1.2. Система координат в 3D

Куб на рис. 1.2 находится в точках (1, 1.5, 2) **мирового пространства**, поскольку `transform` элемента представлен через мировые координаты относительно (0, 0, 0).

Теперь, когда мы знаем, что преобразование куба происходит от мирового (0, 0, 0), рассмотрим взаимосвязь родитель–потомок (parent-child), наглядно описывающую локальное пространство. На рис. 1.3 сфера является потомком

куба. Локальное положение сферы (0, 1, 0) по отношению к кубу. Если вы начнете перемещать куб, то сфера последует за ним, поскольку она всего лишь смещена от куба, а ее преобразования останутся (0, 1, 0) по отношению к кубу.

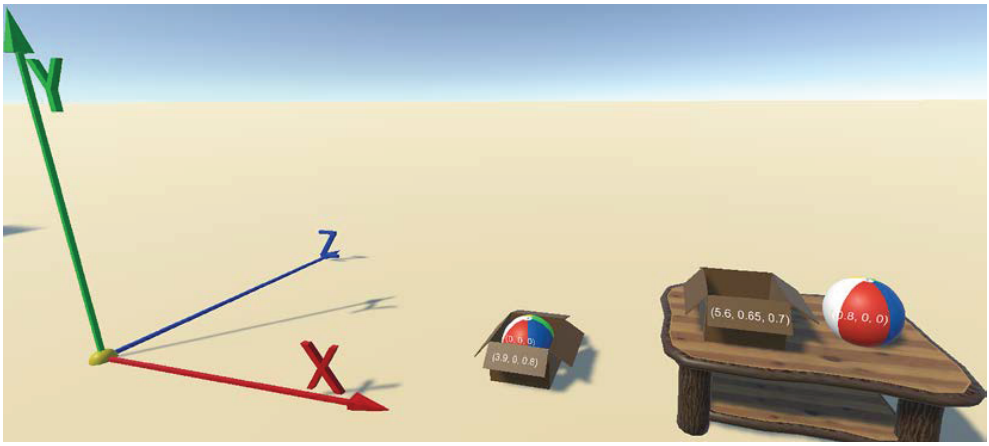

Рис. 1.3. Локальное и мировое пространство

Векторы

Вектор – это единица, которая имеет более одного элемента с направлением. Далее `Vector3` будет выглядеть очень похоже на то, с чем мы только что работали. (0, 0, 0) – это `Vector3`! Векторы используются в очень многих решениях для игровых элементов и логики. Обычно разработчик нормализует векторы таким образом, что величина всегда будет равна 1. Это позволяет разработчику очень легко работать с данными, так как 0 – это начало, 0.5 – это половина пути, а 1 – это конец вектора.

Камеры

Камеры – невероятно важные компоненты! Они смиренно показывают наш взгляд на объекты, что позволяет игрокам испытать то, что мы желаем им передать. Как вы уже могли подумать, у камеры тоже есть свой `transform`, как и у всех игровых объектов (которые мы изучим позже в этой главе) в иерархии. Камеры также имеют несколько параметров, которые можно изменять для получения различных визуальных эффектов.

В разных игровых элементах и жанрах камеры используются по-разному. Например, в игре Resident Evil используются статичные камеры, чтобы создать ощущение напряжения, и игрок не знает, что находится за окном или за углом, в то время как Tomb Raider приближает камеру, пока игровой персонаж Лара проходит через пещеры, создавая ощущение близости и эмоциональное понимание ее неудобства в ограниченном пространстве.

Камеры необходимы для передачи определенного опыта, который вы будете создавать для своих игроков. Будет важным уделить этому время и изучить композиционные концепции, чтобы передать максимальные ощущения игрокам.

Faces, edges, vertices, meshes

3D-объекты состоят из нескольких частей, как показано на рис. 1.4. Вершины (vertices), обозначенные зеленым цветом, являются точками объекта в пространстве относительно мира (0, 0, 0). Каждый объект имеет некоторое количество вершин и соответствующих им связей.

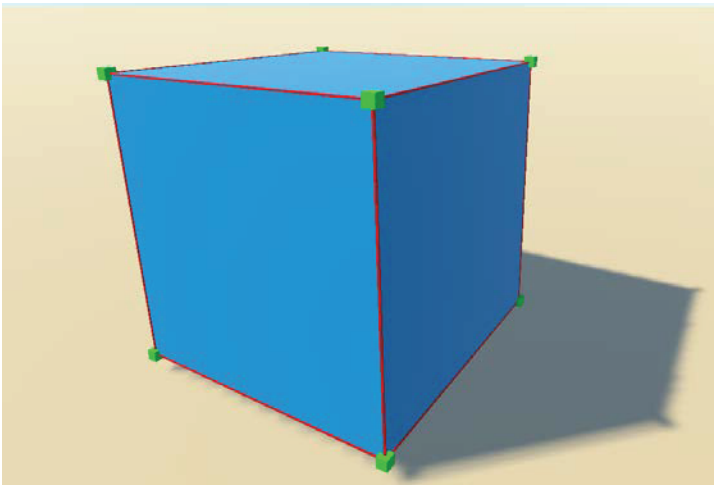

Рис. 1.4. Вершины, ребра, грани и меши

Две соединенные вершины образуют ребро (edge), обозначенное красной линией. Когда три или четыре ребра соединяются, образуя треугольник или четырехугольник, мы получаем грань (face). Если один четырехугольник не соединен с другими гранями, его называют плейн (plane). Соединив все три части вместе, мы получаем меш (mesh).

Материалы, текстуры и шейдеры

Теперь, когда вы знаете, из чего состоит меш во всех инструментах DCC, давайте посмотрим, как это отображает конкретно Unity. На базовом уровне находится шейдер. Шейдеры можно рассматривать как небольшие программы, которые имеют собственный язык и работают на графическом процессоре, поэтому Unity может отображать объекты сцены на вашем экране. Вы можете думать о шейдере как о большом шаблоне для создаваемых материалов.

Следующий уровень – материалы. Материал – это набор атрибутов, которыми манипулирует шейдер, с помощью чего мы показываем, как выглядит объект. Каждый конвейер рендеринга будет иметь отдельные шейдеры: **Встроенный конвейер рендеринга** (Built-in), **Универсальный конвейер рендеринга** (URP – Universal Rendering Pipeline) или **Конвейер рендеринга высокого разрешения** (HDRP – High Definition Rendering Pipeline). В этой книге мы используем второй, самый используемый вариант: URP.

На рис. 1.5 показан пример материала с использованием шейдера URP **Standard Lit**. Это позволяет нам управлять параметрами поверхности, входными данными для этой поверхности и некоторыми дополнительными параметрами.

А пока давайте просто поговорим о **Base Map**, первом элементе в разделе Surface Inputs. Термин **Base Map** используется здесь как комбинация **Diffuse**/**Albedo** и **Tint**. **Diffuse**/**Albedo** используется для определения базового цвета (красное выделение), который будет применяться к поверхности, – в нашем случае белого.

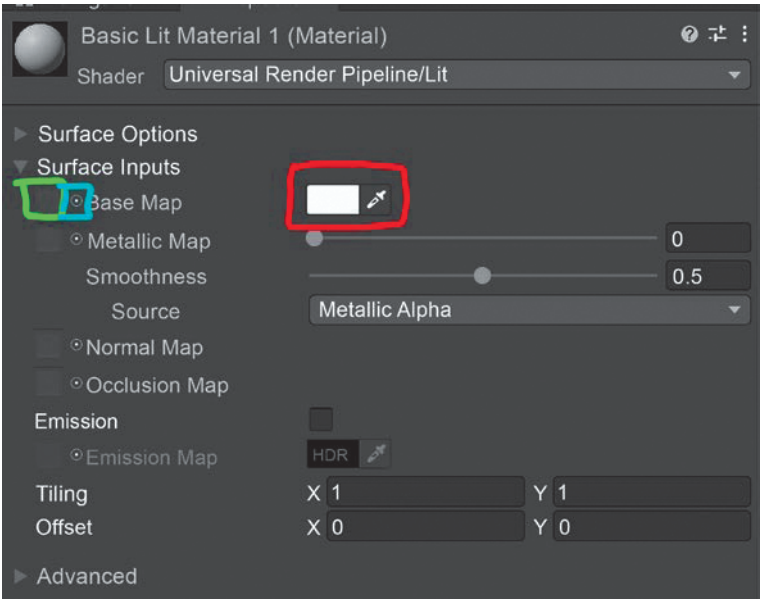

Рис. 1.5. Базовые параметры материалов

Если вы поместили текстуру на эту карту, либо перетащив текстуру на квадрат (зеленое выделение) слева от базовой карты, либо щелкнув по кружку (синее выделение) между полем и названием, вы сможете подкрасить поверхность цветом, если нужны какие-либо корректировки.

На рис. 1.6 показан простой пример того, как будет выглядеть куб с оттенком, текстурой и той же самой текстурой с измененным оттенком. По мере прохождения книги мы будем открывать все больше и больше функций материалов, шейдеров и текстур.

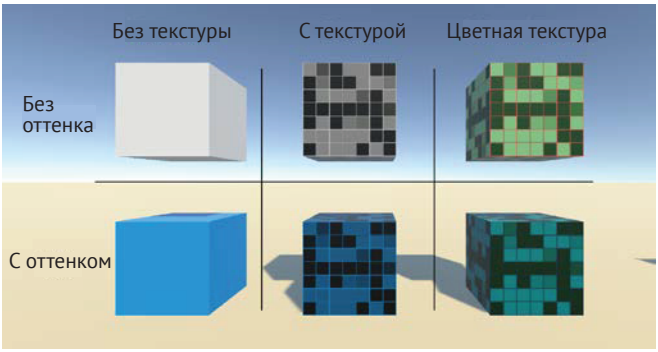

Рис. 1.6. Оттенок и текстура основного цвета

Текстуры могут обеспечить невероятную детализацию вашей 3D-модели.

При создании текстуры важным фактором является разрешение. Первая часть раздела разрешения, которую необходимо понять, – это размер «power of 2» (кратное двум), например:

2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096…

Эти числа представляют размер пикселя как по ширине, так и по высоте. Если вам понадобится использовать разные размеры для ширины и высоты, то только если числа будут кратны двум. Примеры:

- 256×256,
- 1024×1024,
- 256×1024 (редко встречается в использовании, но все равно корректен).

Второй пункт относительно разрешения – это размерность. Самый простой способ – подумать о том, насколько большим будет 3D-объект на вашем экране. Если у вас разрешение экрана 1920×1080, то это 1920 пикселей в ширину и 1080 пикселей в высоту. Если рассматриваемый объект будет занимать только 10 % экрана и его вряд ли можно будет близко рассмотреть, вы можете использовать текстуру 256×256. Другой вариант – если вы делаете игру, в которой важны эмоции и выражение лица, может понадобиться текстура 4096×4096 или 4K только на лице во время каких-либо кат-сцен.

Физика Rigidbody

Unity предполагает, что для физики нам не нужно вычислять каждый кадр игрового объекта. Для моделирования физики Unity использует движок Nvidia PhysX. Чтобы получить какие-либо рассчитанные физические отклики, в игровой объект необходимо добавить компонент Rigidbody (твердое тело).

Как только вы добавили компонент Rigidbody в игровой объект, появляется меню с некоторыми возможностями воздействия на данный игровой объект.

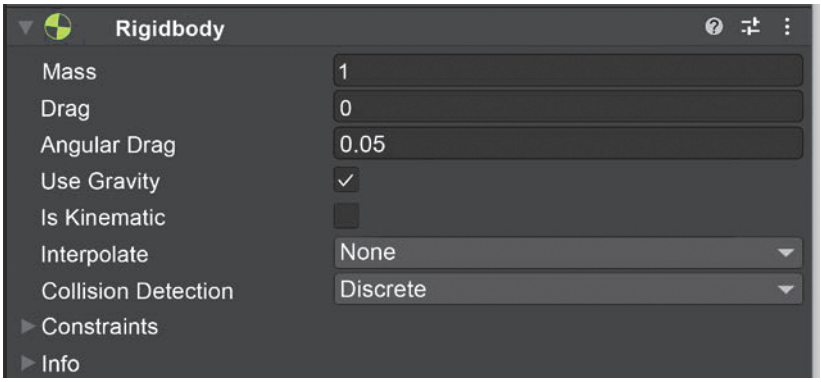

Рис. 1.7. Rigidbody

В Unity одна единица массы равна 1 кг. Это влияет на физические решения при коллизиях. Параметр **Drag** добавляет трение, уменьшая скорость относительно времени. В **Angular drag** аналогично, но ограничено только скоростью

вращения. **Use Gravity** включает или выключает гравитацию, равную стандартной земной гравитации (0, –9.81, 0), поэтому масса крайне важна! А если вам не нужна конкретно земная гравитация, можете изменить настройки физики, чтобы сделать гравитацию такой, какой хотите.

Подробное объяснение темы Rigidbody будет рассмотрено в главе 7. Там мы будем использовать Rigidbody при создании персонажей, а также окружающей среды и интерактивного игрового процесса.

Обнаружение столкновений

Игровой объект с Rigidbody без каких-либо коллайдеров не будет полностью использовать физику и с включенной гравитацией просто провалится сквозь мир. Существует довольно много типов коллайдеров, с которыми можно поиграть, чтобы удовлетворить потребностям ваших игр. На рис. 1.8 вы можете видеть, что для 2D существуют отдельные коллайдеры. Они используют систему физики, отличную от 3D. Если вы делаете игру в 2D, обязательно используйте 2D-коллайдеры.

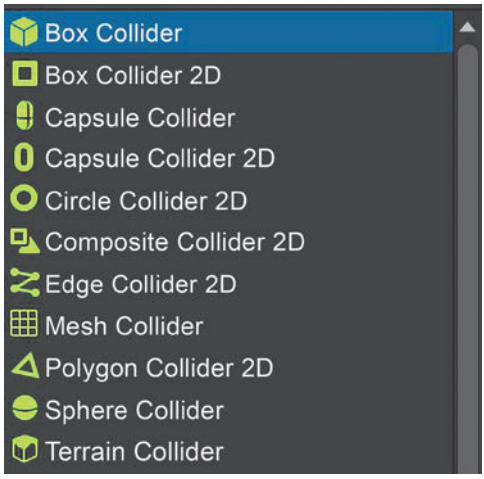

Рис. 1.8. Вариативность коллайдеров

Вы также можете добавить сразу несколько коллайдеров – с основными параметрами, показанными на рис. 1.8, – к объекту, чтобы он наилучшим образом соответствовал форме игрового объекта. Очень часто можно увидеть коллайдеры на пустых игровых объектах, которые находятся в связи объект–ребенок у основного объекта-родителя, чтобы упростить преобразование коллайдеров. Мы рассмотрим это на практике в главе 4 «Персонажи» и в главе 5 «Окружающая среда».

Интерфейс Unity

Интерфейс Unity разделен на несколько основных компонентов. На рис. 1.9 мы видим окно Сцены (**Scene**) (красный цвет) и элементы в ее интерфейсе, а также меню управления свойствами инспектор (**Inspector**) (оранжевый цвет). Затем мы перейдем к элементам, которые неактивны в сцене, но доступны для добав-

ления в Окне проекта (**Project Window**) (фиолетовый цвет). Наконец, рассмотрим окно Игра (**Game**) (зеленый) и Менеджер пакетов (отдельно от рис. 1.9).

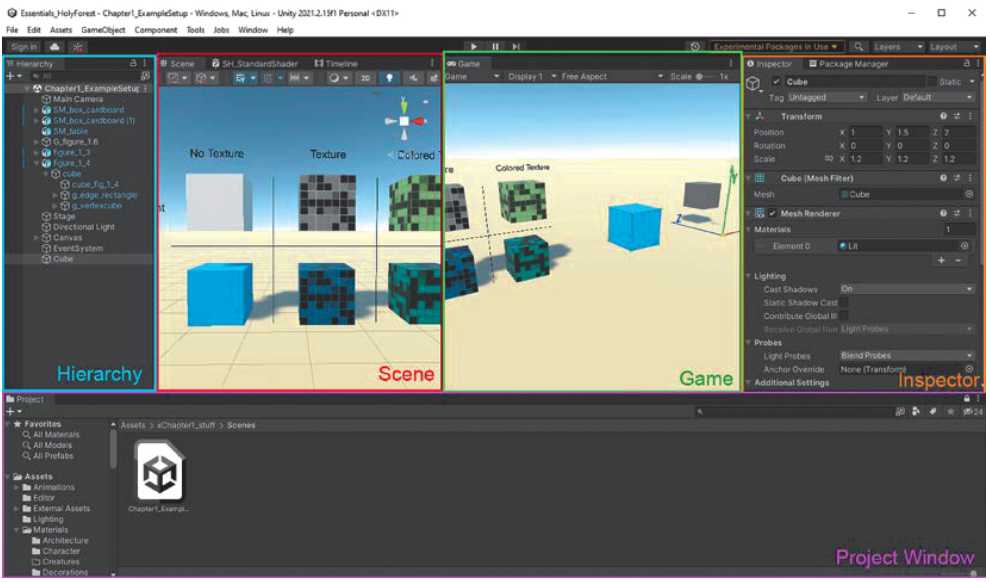

Рис. 1.9. Общий интерфейс

Окно сцены и иерархия

Окно сцены и иерархия работают в тандеме. Иерархия – это то, как сцена будет отображаться во время игры. Окно сцены позволяет вам манипулировать игровыми объектами и их значениями в режиме реального времени. Кроме того, когда редактор находится в режиме воспроизведения, игре можно вносить изменения в игровые объекты в иерархии.

Когда игровым объектом манипулируют в режиме воспроизведения, включая манипуляции вручную в окне сцены, как только вы остановите воспроизведение, все игровые объекты автоматически вернутся в исходное положение до запуска.

На первый взгляд вам может показаться, что на рисунке слишком много информации. Слева в иерархии видно, что в сцене есть объекты. Все эти объекты имеют `transform`, который помещает их в мир. Если дважды щелкнуть по объекту или же сначала щелкнуть, навести указатель мыши на окно сцены, а затем нажать клавишу **f**, вы сфокусируетесь на этом объекте, что поместит его в центр окна просмотра сцены (viewport).

После выбора объекта вы увидите его точку пивот (pivot) с цветными стрелками, по стандарту он находится в центре объекта. Данный инструмент позволяет позиционировать игровой объект в пространстве. Вы также можете расположить объект на плоскости, выбрав маленький квадрат между двумя осями.

В правом верхнем углу рис. 1.10 вы увидите гизмо камеры. Эта маленькая штуковина позволит вам легко сориентировать камеру вьюпорта на вид спереди, сбоку, сверху, снизу или изменить ее на изометрическую или перспективную камеру одним щелчком мыши.

Теперь, когда вы видите объект в сцене, выбранный щелчком левой кнопки мыши в сцене или в иерархии, вы можете изменить некоторые свойства или добавить компоненты к этому игровому объекту. Тут в игру вступает инспектор.

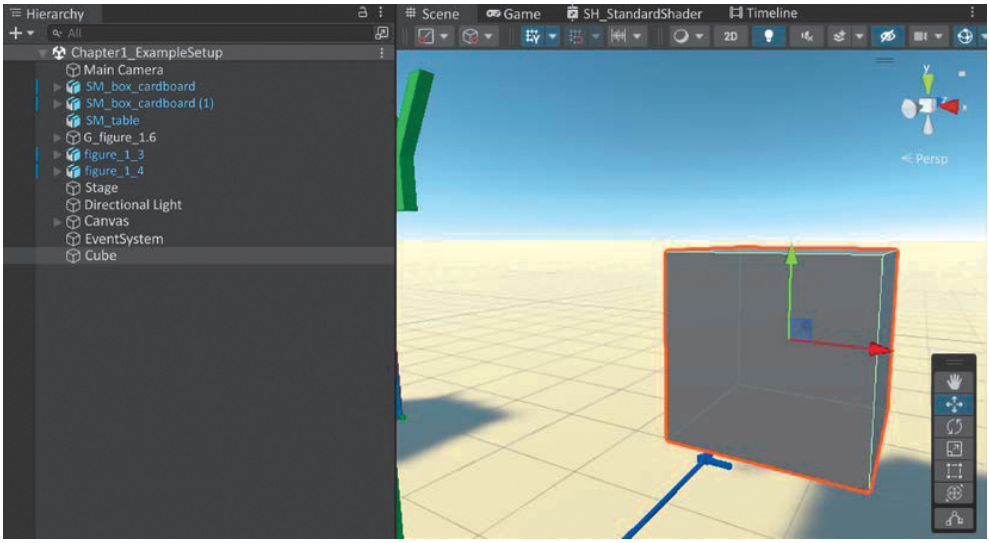

Рис. 1.10. Сцена и иерархия

Inspector

Чтобы манипулировать игровым объектом, когда вы выбираете его в сцене или иерархии, инспектор будет меняться, чтобы показать вам возможные варианты изменения для каждого объекта в отдельности.

Окно инспектора на рис. 1.11 показывает некоторое количество параметров для манипуляции с объектом. Вверху имя **Cube**, а синий куб слева обозначает префабный (шаблонный) тип данных. Вы можете внести изменения в сам префаб, нажав кнопку **Open** чуть ниже имени. Это создаст новый вид сцены, который показывает только префаб. Когда вы вносите изменения в префаб, эти изменения будут дублироваться во все экземпляры данного префаба в любой сцене, которая на него ссылается.

Компонент **Transform** показывает положение, поворот и масштаб префаба в сцене.

Mesh filter показывает вершины, ребра и плоскости, из которых состоит полигон.

Ниже находится **Mesh renderer**. Этот компонент позволит визуализировать меш, отображаемый в компоненте **Mesh filter**. Здесь мы можем задать материал и другие параметры, относящиеся к конкретному освещению и датчикам этого предмета, которые мы рассмотрим в главе 12 «Последние штрихи».

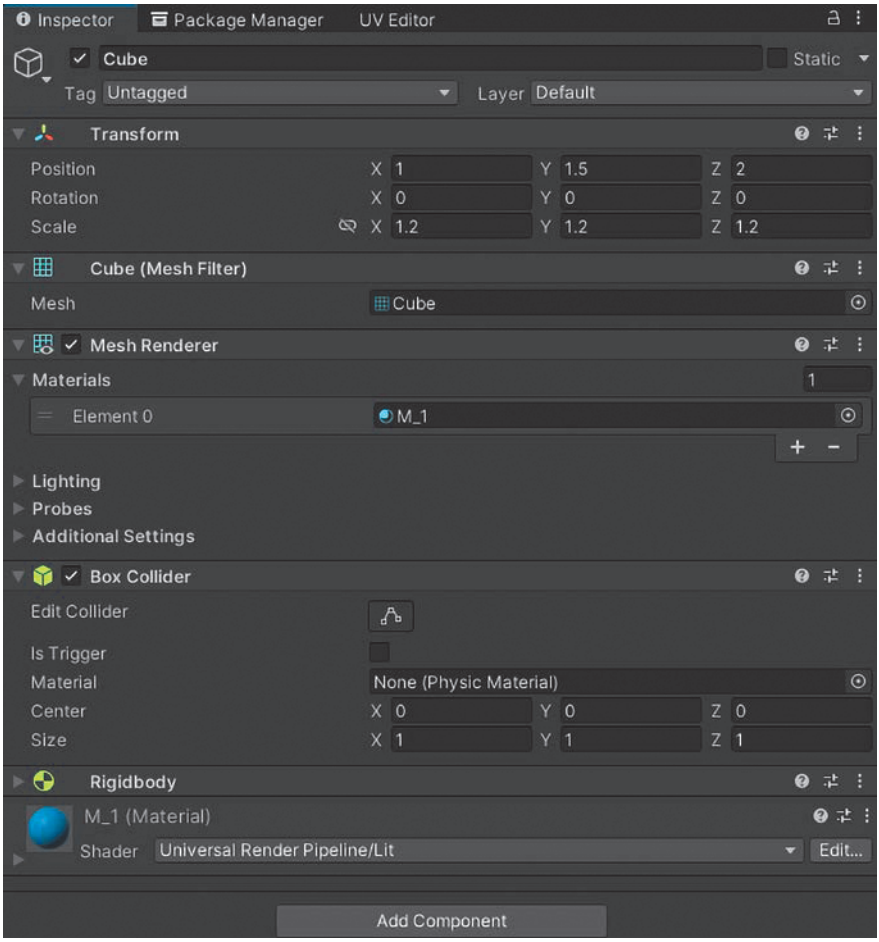

Рис. 1.11. Окно **Inspector**

Далее по списку мы видим **Collider** и **Rigidbody**. Они работают в тандеме и помогают объекту реагировать на физику в реальном времени в соответствии с настройками данных компонентов.

Мы много говорили об элементах сцены и их свойствах, но где они размещаются, находясь за пределами сцены? Окно **Project** ответит на этот вопрос.

Project window

Здесь вы найдете ассеты, которые будут добавлены в сцену или использованы в качестве составного элемента для полной реализации создаваемой вами игры.

Это окно является физическим представлением игровых объектов, на которые вы ссылаетесь. Все объекты в папке ассетов, показанной на рис. 1.12, физически находятся на вашем жестком диске. Unity создает метафайлы, содержащие все свойства объектов.

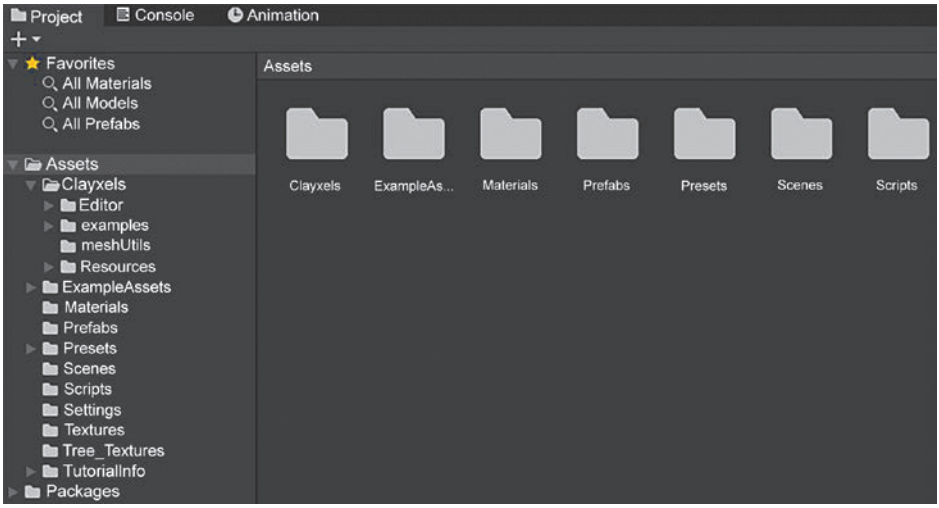

Рис. 1.12. Окно **Project**

Благодаря наличию первоначальных файлов в окне **Project** вы можете изменять объекты, а когда вы выбираете проект Unity (нажмите на программу Unity), все метафайлы перенастроятся и объекты в сцене перезагрузятся.

Мы посмотрели на игровые объекты в сцене, разместили их, манипулируя преобразованиями, и знаем, откуда на них ссылаются. Теперь мы должны посмотреть на окно **Game**, чтобы узнать, как выглядит сама игра.

Окно Game

Окно игры похоже на окно сцены; однако он следует правилам, встроенным в представление сцены. Игра будет автоматически рендерить содержимое сцены через основную камеру или же через ту, которую вы укажете.

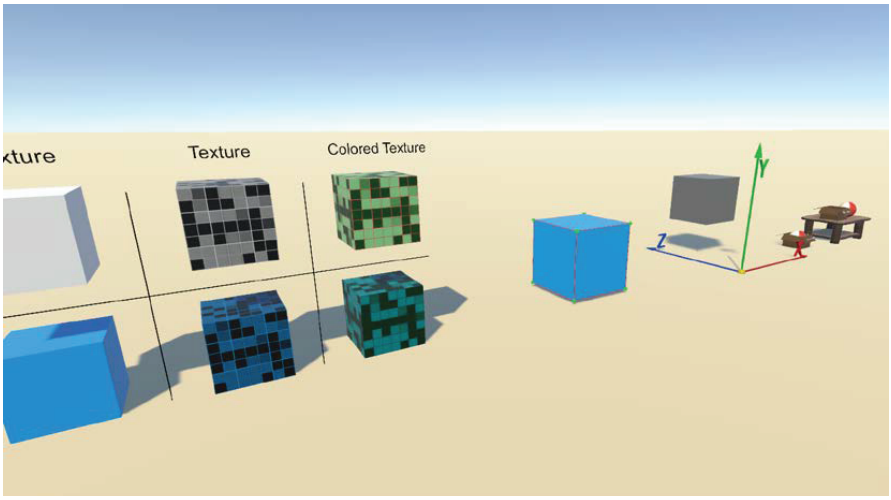

Рис. 1.13. Окно **Game**

Здесь мы видим сходство с окном сцены, но в верхней части отличаются параметры. В левом верхнем углу есть раскрывающийся список **Display**, в котором можно менять камеры, если в сцене их несколько. Правее находится соотношение сторон экрана для настройки таргета на определенные объекты. Далее **Scale**, с помощью которого можно, к примеру, быстро увеличить окно или приблизить для удобства отладки. **Maximize On Play** развернет экран во время проигрывания, чтобы использовать все преимущества полноэкранного режима. **Mute Audio** отключает звук в игре. **Stats** дает небольшой обзор статистики в игровом окне.

Позже в нашем проекте во время оптимизации мы проведем профилирование, чтобы более подробно рассмотреть, что может вызывать проблемы в игровом процессе с точки зрения использования памяти и других возможностей оптимизации.

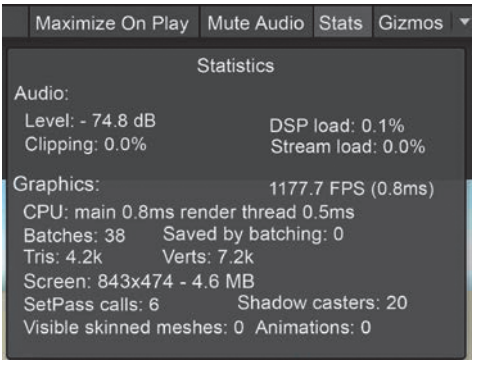

Рис. 1.14. Статистика игры

Продолжая движение направо, вы увидите **Gizmos**. Это набор элементов, которые отображаются в окне игры. В этом меню вы можете отключить или включить их в зависимости от ваших потребностей.

Package Manager

В вашем Unity ID будут храниться пакеты, которые вы купили в Unity Asset Store, а также пакеты, которые могут быть у вас на жестком диске или на GitHub! С помощью менеджера вы можете импортировать пакеты в свой проект.

Чтобы найти менеджер пакетов, пройдите **Windows > Package Manager**, как показано на рис.1.15.

После того как вы откроете менеджер, сначала вам будет показано, какие пакеты уже есть в проекте. Вы можете изменить раскрывающийся список в верхнем левом углу, чтобы увидеть, что является стандартным в Unity или какие пакеты вы купили в Unity Asset Store.

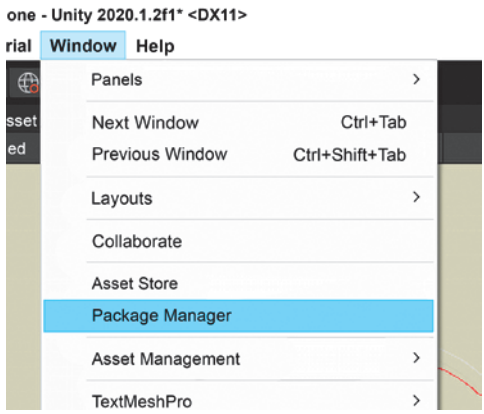

Рис. 1.15. Путь в Package Manager

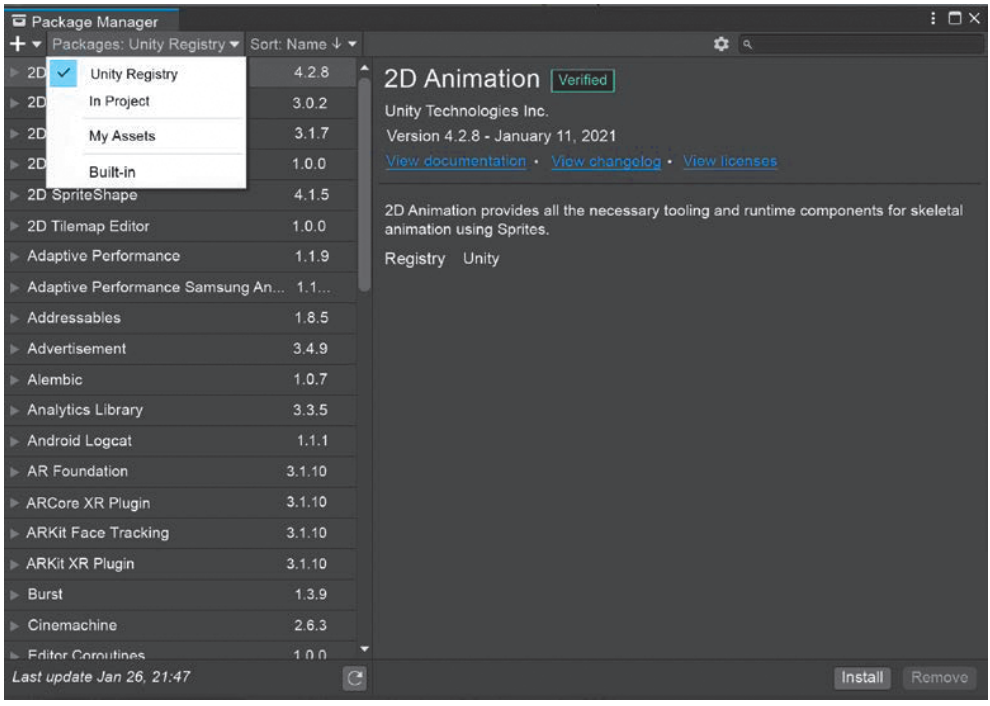

Рис. 1.16. Package Manager

Выбрав Unity Registry, вы увидите список проверенных пакетов Unity, которые предоставляются бесплатно и являются базовой частью платформы Unity. Выбрав пакет, можете прочитать о нем документацию, представленную по ссылке **View documentation**.

Если вы выберете список **In Project**, он покажет вам, какие пакеты уже установлены в текущем загруженном проекте. Так вы сможете удалить ненужные пакеты в вашем проекте.

My Assets – это ассеты, которые вы купили для проекта, а также те, которые связаны с вашим Unity ID.

Built-in стандартен для любого проекта. Возможно, в одном из проектов потребуется включить или отключить какой-либо встроенный пакет в зависимости от ваших потребностей. Изучите их и отключите то, что не нужно; компактный проект в начале приводит к меньшей оптимизации в будущем.

Основные концепции Unity

Мы уже рассмотрели, где используются некоторые концепции, а теперь более подробно раскроем их. В Unity реализован очень модульный подход к элементам, размещенным в среде разработки игр.

Ассеты

Unity рассматривает каждый файл как ассет – все, включая 3D-модель, файл текстуры, спрайт, систему частиц (партиклов) и т. д. В проекте у вас будет папка **Assets** в качестве базовой папки для размещения всех элементов. Это могут быть текстуры, 3D-модели, системы частиц, материалы, шейдеры, анимация, спрайты и т. д. По мере того как мы увеличиваем количество элементов проекта, наша папка должна быть организована и готова к росту. Настоятельно рекомендуется поддерживать организованную структуру папок, чтобы вы или ваша команда не тратили время впустую, пытаясь найти тот единственный элемент текстуры, который был случайно оставлен в случайной папке.

Сцены

Сцена содержит всю логику игрового процесса, игровые объекты, синематики и все остальное, на что ваша игра будет ссылаться для рендеринга или взаимодействия.

Сцены также используются для разделения игрового процесса на части с целью сократить время загрузки. Представьте, что при каждом включении современной игры вы пытаетесь загружать абсолютно все ассеты. Это отнимет слишком много драгоценного игрового времени.

Игровые объекты

Большинство ассетов, которые представлены в сцене, являются **игровыми объектами** (GameObject, далее GO). В некоторых случаях ассет может быть лишь компонентом GO. Один общий фактор, который вы увидите во всех GO, заключается в том, что у них есть компонент **Transform**. Как мы читали в начале главы, `transform` содержит локальное положение, поворот и масштаб. Мировые преобразования рассчитываются исходя из локальных в соответствии с их иерархией. GO может иметь длинный список связанных для функциональности компонентов или данных, которые будут использоваться в скриптах для развития механики.

Компоненты

GO могут содержать несколько функциональных частей в виде компонентов. Каждый компонент имеет свои уникальные свойства. Весь список компонентов, которые вы можете добавить, довольно обширен, как можно видеть на рис. 1.17.

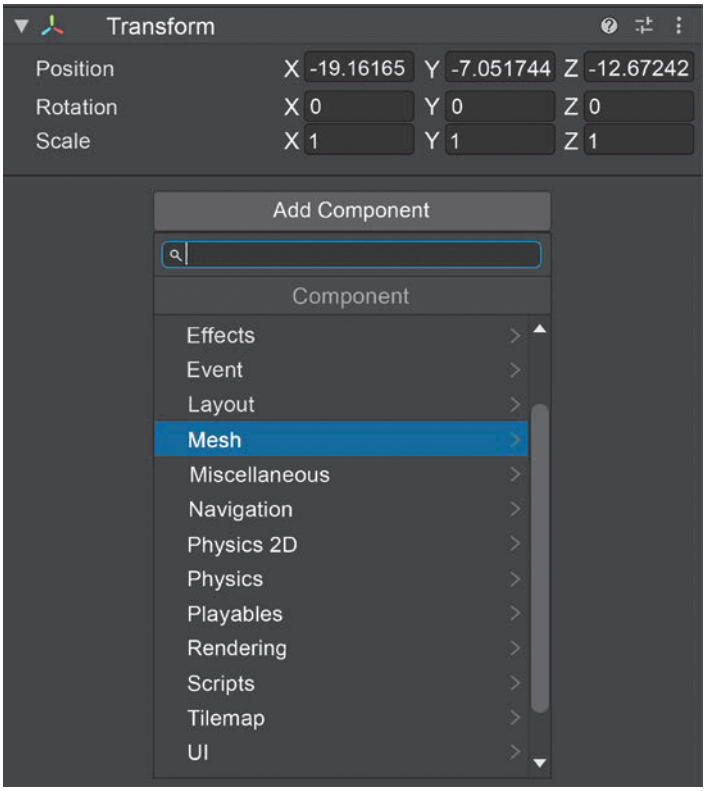

Рис. 1.17. Список компонентов

Каждый из этих разделов имеет свои подразделы. В книге мы рассмотрим большую часть. Когда вы добавляете ассет в иерархию сцены, для которого требуются компоненты, Unity добавит их по умолчанию. Например, когда вы перетаскиваете 3D-меш в иерархию, к GO автоматически прикрепляется компонент отображения сетки.

Скрипты

Одним из компонентов, который часто используется в GO, является скрипт. В скриптах будет встроена вся логика и механика игровых объектов. Если вы хотите изменить цвет, прыгнуть, изменить время суток или собрать предмет, вам нужно будет добавить эту логику в скрипт объекта.

В Unity основным языком является C# (произносится как «C sharp»). Это строго типизированный язык программирования, а это означает, что любой переменной, с которой манипулируют, должен быть присвоен тип.

Мы с вами будем использовать скрипты множеством способов, и я уверен, что вы уже были бы рады приступить к кодированию, но перед этим нам еще необходимо разобраться в других стандартных процессах Unity.

Префабы

Используя модульную и крайне объектно ориентированную природу Unity, мы можем собрать группу элементов со значениями по умолчанию, установленными для их компонентов, которые могут быть добавлены в сцену в любое время и содержать свои собственные значения.

Чтобы создать префаб, вы перетаскиваете GO из иерархии сцены в браузер ассетов. Объект в иерархии будет обозначен синим цветом по умолчанию, как показано на рис. 1.18.

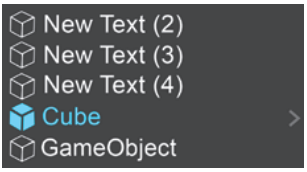

Рис. 1.18. Префаб в иерархии

Пакеты

Чтобы вывести модульные компоненты на совершенно новый уровень, Unity может взять пакет со всеми его зависимостями и экспортировать их для использования в других проектах. Более того, вы можете продавать свои пакеты другим разработчикам игр через Unity Asset Store.

Теперь, когда у вас есть прочная основа в терминах 3D и Unity, давайте рассмотрим сам интерфейс. В следующем разделе мы изучим все наиболее распространенные элементы интерфейса Unity.

Заключение

Итак, мы с вами рассмотрели несколько ключевых областей, чтобы начать свой путь в разработке игр. В этой главе мы заложили основу того, что будет происходить дальше, рассмотрев некоторые фундаментальные особенности трех основных тем. Для третьего измерения мы рассмотрели систему координат, векторы, камеры, 3D-сетки, основы физики Rigidbody и обнаружения коллизий. Этой основы, чтобы мы могли разобраться в концепциях Unity, таких как ассеты и игровые объекты основы префабов, а затем написать скрипты на C#, на данном этапе достаточно. В завершение этой главы мы прошли виртуальный тур по интерфейсу Unity – сценам, иерархии, инспекторам и диспетчеру пакетов.

В следующей главе мы рассмотрим основы дизайна и прототипирования. Там вы начнете понимать, как мы описываем наши мыслительные процессы для проекта, создаваемого в этой книге. Кроме того, она заложит фундаментальные знания, которым вы будете следовать при создании собственных проектов после прочтения этой книги.