1 Введение

Когда у меня в 1979 году появилась первая игровая консоль — крутая система Intellivision от компании Mattel, термина «игровой движок» не существовало. Тогда аркадные и видеоигры, по мнению большинства взрослых, были не более чем игрушками, и запускающее их программное обеспечение разрабатывалось специально для конкретной игры и оборудования, на котором они запускались. Сегодня игры — это господствующая многомиллиардная индустрия, конкурирующая с Голливудом по масштабу и популярности. И программное обеспечение, которое лежит в основе этих теперь уже повсеместно распространенных трехмерных миров, называется *игровыми движками*. К ним относятся Unreal Engine 4 (компания Epic Games), Source (компания Valve), CRYENGINE® 3 (компания Crytek), Frostbite™ (компания DICE (Electronic Arts)), а также Unity. Эти игровые движки стали полнофункциональными средствами разработки программного обеспечения многократного использования, которые могут лицензироваться и применяться для разработки практически любой игры.

В то время как игровые движки значительно различаются в деталях архитектуры и реализации, благодаря как движкам с публичной лицензией, так и их коммерческим аналогам появились узнаваемые паттерны. Фактически все игровые движки содержат схожие наборы основных компонентов, включая движок рендеринга, движок коллизий и физики, объектную модель игрового мира, системы анимации, аудио, искусственного интеллекта и т. д. В каждом из этих компонентов также начинает появляться относительно небольшое количество вариантов дизайна, постепенно становящихся стандартными.

Существует огромное множество книг, которые детально описывают отдельные подсистемы игровых движков, например трехмерную графику. Другие издания освещают приемы работы и дают ценные советы во всем разнообразии областей игровых технологий. Однако я не смог найти такого, которое представляет читателю полную картину всей гаммы компонентов, составляющих современный игровой движок. Цель этой книги и состоит в том, чтобы провести читателя по обширному сложному пейзажу архитектуры игрового движка.

Прочитав эту книгу, вы узнаете:

- как спроектированы реальные мощные игровые движки;
- как организована работа команды разработки игр в реальном мире;

- какие главные подсистемы и паттерны проектирования повторяются вновь и вновь практически в каждом игровом движке;
- каковы типичные требования к любой ключевой подсистеме;
- какие подсистемы не зависят от жанра и игры, а какие разрабатываются явно для определенных жанра и игры;
- где движок обычно заканчивается и начинается собственно игра.

Вы получите непосредственное представление о внутренней работе некоторых популярных игровых движков, таких как Quake, Unreal и Unity. Также мы изучим известные пакеты промежуточного программного обеспечения, такие как библиотека физики Havok Physics, движок рендеринга OGRE и Granny 3D от компании Rad Game Tools — набор инструментов управления анимацией и геометрией. Мы исследуем множество коммерческих игровых движков, с которыми я имел удовольствие работать, включая движок Naughty Dog, разработанный для игровых серий *Uncharted* и *The Last of Us*.

Прежде чем начать, рассмотрим некоторые методы и инструменты крупномасштабной разработки программного обеспечения в контексте игровых движков, а именно:

- разницу между логической и физической архитектурой программного обеспечения;
- управление конфигурациями, контроль версий и системы сборки;
- советы и приемы работы с одной из наиболее часто используемых сред разработки для C и C++ — Microsoft Visual Studio.

Я полагаю, что вы хорошо знаете C++ (язык, который выбирает большинство современных игровых программистов) и понимаете основные принципы проектирования программного обеспечения. Кроме того, я считаю, что вы имеете представление о линейной алгебре, трехмерной векторной и матричной математике и тригонометрии (речь о некоторых ключевых понятиях пойдет в главе 5). В идеале вы должны иметь начальное представление о фундаментальных понятиях программирования систем реального времени и событийно-ориентированного программирования. Но если это не так, не переживайте — я кратко рассмотрю эти темы и укажу вам правильное направление, если вы почувствуете, что должны усовершенствовать свои навыки, прежде чем мы начнем углубленное изучение.

1.1. Структура типичной игровой команды

Прежде чем погрузиться в изучение структуры типичного игрового движка, кратко рассмотрим структуру типичной команды разработки игры. Игровые студии обычно формируются из специалистов пяти основных направлений: разработчиков, художников, геймдизайнеров, продюсеров и другого управленческого персонала и персонала поддержки (маркетинговая, правовая, информационно-техническая и технологическая поддержка, администрация и т. д.). Каждая должность может иметь различные специализации. Мы коротко поговорим о каждой из них далее.

1.1.1. Разработчики

Разработчики проектируют и создают программное обеспечение, заставляющее игру и инструменты работать. Они зачастую делятся на две группы: *разработчики среды выполнения* (работают над движком и игрой как таковой) и *разработчики инструментов* (создают офлайновые инструменты, позволяющие остальным разработчикам действовать более эффективно). Члены обеих групп имеют различные специальности. Некоторые из них сосредотачиваются на одной определенной системе движка, например рендеринге, искусственном интеллекте, аудио или коллизиях и физике, некоторые — на программировании геймплея и скриптинге. А другие предпочитают работать на системном уровне и не слишком вникают в то, как на самом деле работает игра. Некоторые разработчики — универсалы, мастера на все руки, которые могут переключиться на работу над любой проблемой, требующей решения во время разработки.

Старших разработчиков иногда просят взять на себя техническую руководящую роль. Ведущие разработчики обычно продолжают проектировать и писать код, а также помогают управлять работой команды, принимают решения относительно общего технического направления проекта и иногда даже напрямую руководят другими сотрудниками.

В некоторых компаниях также есть один или несколько технических директоров, работа которых сводится к ведению одного или нескольких проектов на высоком уровне, информированию команд о потенциально возможных технических проблемах, предстоящих событиях в индустрии, новых технологиях и т. д. Самая высокая должность, связанная с программированием в игровой студии, — CTO (если она вообще имеется). Роль CTO — быть своего рода техническим директором для всей студии и выполнять ключевую управляющую роль в компаниии.

1.1.2. Специалисты творческих профессий

В игровой индустрии говорят: «Контент — это главное». Художники создают весь визуальный и аудиоконтент игры, и качество их работы может буквально сделать или уничтожить ее. Для работы требуются художники и дизайнеры всех направлений.

- *Создатели концепт-артов* создают скетчи и зарисовки, позволяющие команде увидеть итог разработки игры. Они начинают раньше других — на стадии разработки концепции, но обычно обеспечивают визуальную направленность проекта на протяжении всего жизненного цикла. Как правило, скриншоты, сделанные в процессе игры, имеют странное сходство с произведениями концептуального искусства.
- *Разработчики 3D-моделей (3D-моделлеры)* создают трехмерную геометрию для всего в виртуальном мире игры. Сюда обычно входят разработчики моделей переднего и заднего плана. Те, кто работает над передним планом, создают объекты, персонажей, транспорт, оружие — все то, что наполняет игровой мир. Разработчики моделей заднего плана создают геометрию статичного фона — растительность, здания, мосты и т. д.

- *Художники по текстурам* создают двухмерные изображения, называемые текстурами, которые накладываются на поверхности 3D-моделей для обеспечения детализации и реализма.
- *Специалисты по освещению* создают в игровом мире источники света, как статичные, так и динамичные, работают с цветом, интенсивностью и направлением освещения, чтобы обеспечить максимальную художественность и эмоциональное воздействие каждой сцены.
- *Аниматоры* придают движущимся персонажам и объектам игры динамичность. Они буквально выступают актерами в производстве игры, как это происходит в кинопроизводстве с компьютерной графикой. Однако игровой аниматор должен иметь уникальный набор навыков, чтобы выполнить анимацию, которая будет безупречно согласовываться с технологическими особенностями игрового движка.
- *Актеры захвата движения* обычно используются для обеспечения грубого набора данных о движении, которые затем обрабатывают аниматоры, прежде чем интегрировать их в игру.
- *Звукорежиссеры* работают вместе с разработчиками, чтобы создать и смешать звуковые эффекты и музыку в игре.
- *Актеры озвучки* наделяют персонажей голосами в большинстве игр.
- Во многих играх есть один или несколько *композиторов*, сочиняющих оригинальное звуковое сопровождение для игры.

Как и разработчиков, старших художников часто призывают стать тимлидерами. Некоторые игровые команды имеют одного (или нескольких) арт-директора, который является главным художником, управляющим всей игрой и обеспечивающим упорядоченность деятельности всех членов группы.

1.1.3. Геймдизайнеры

Работа геймдизайнера заключается в разработке интерактивной составляющей пользовательского взаимодействия с игрой, называемой *геймплеем*. Различные дизайнеры работают на разных уровнях детализации. Некоторые (как правило, старшие) геймдизайнеры действуют на макроуровне, определяя сюжет игры, общую последовательность глав или уровней, а также основные цели и задачи игрока. Другие дизайнеры работают над определенными уровнями или географическими областями внутри виртуального мира игры, создавая слои геометрии статичного фона, определяя, когда и откуда появятся враги, размещая такие припасы, как оружие и восстанавливающие здоровье аптечки, разрабатывая элементы внутриигровых загадок и т. д. В то же время другие дизайнеры работают на более высоком техническом уровне, тесно взаимодействуя с разработчиками геймплея, и/или пишут код (как правило, на высокоуровневом языке скриптинга). Многие геймдизайнеры в прошлом являлись разработчиками, решившими играть более активную роль в определении того, как игра будет выглядеть.

Некоторые игровые команды нанимают одного или нескольких игровых писателей. Работа этого специалиста может варьироваться от сотрудничества со старшими геймдизайнерами в целях конструирования сюжета всей игры до написания отдельных линий диалогов.

Некоторые старшие дизайнеры играют роль менеджеров. Во многих игровых командах есть геймдиректор, работа которого заключается в наблюдении за всеми аспектами геймдизайна, помощи в управлении сроками и обеспечении того, чтобы деятельность всех дизайнеров была упорядочена на протяжении всей работы над продуктом. Старшие дизайнеры иногда вырастают в продюсеров.

1.1.4. Продюсеры

Роль продюсеров в различных студиях определяется по-разному. В одних игровых компаниях их работа заключается в управлении сроками и человеческими ресурсами. В других продюсер служит старшим геймдизайнером. В третьих он является связующим звеном между командой разработчиков и подразделениями компании (финансовыми, правовыми, маркетинговыми и т. д.). В некоторых небольших студиях вообще нет продюсеров. Например, в студии Naughty Dog буквально каждый, включая обоих сопрезидентов, играет ведущую роль в создании игры, управление командами и деловые обязанности поделены между старшими специалистами.

1.1.5. Другой персонал

Группу людей, непосредственно создающих игру, как правило, поддерживает ключевая команда технического персонала. В нее входят исполнительное руководство студии, маркетинговый отдел (или команда, которая кооперируется с внешней маркетинговой группой), административный штат и IT-отдел, работа которого состоит в том, чтобы купить, установить и настроить аппаратное и программное обеспечение и оказывать техническую поддержку сотрудникам.

1.1.6. Издатели и студии

Маркетингом, выпуском и распространением игры обычно занимается *издатель*, а не сама студия. Издатель — это, как правило, крупная корпорация, такая как Electronic Arts, THQ, Vivendi, Sony, Nintendo и др. Многие игровые студии не аффилированы с каким-то определенным издателем. Они продают производимые игры любому издателю, предложившему более выгодные условия. Другие студии работают только с определенным издателем, либо заключив с ним долгосрочный контракт, либо являясь его дочерней компанией. Например, игровые студии THQ управляются независимо от издателя THQ, но они принадлежат ему и полностью контролируются им. Издатель Electronic Arts имеет более тесные отношения со студиями и напрямую управляет ими. Разработчики первого эшелона — это игровые студии, которыми владеют непосредственно производители консолей (Sony,

Nintendo и Microsoft). Например, Naughty Dog — разработчик первого эшелона Sony. Эти студии создают игры эксклюзивно для игровых консолей, выпускаемых родительской компанией.

1.2. Что такое игра

Все мы имеем вполне четкое интуитивное представление о том, что такое игра. Общий термин «игра» охватывает настольные игры, такие как шахматы и «Монополия», карточные игры, например покер и блек-джек, игры в казино, такие как рулетка и автоматы, военные игры, компьютерные игры, различные виды детских игр, и этот список можно продолжать очень долго. Научный термин «теория игр» используется, когда в рамках четко установленного набора игровых правил требуется определить стратегию и тактику поведения нескольких участников для получения максимальной выгоды. Когда термин «игра» применяется в сфере консольных и компьютерных развлечений, он обычно вызывает в памяти изображения трехмерного виртуального мира с гуманоидом, зверем или транспортом в качестве главного персонажа, управляемого игроком. (У более старшего поколения он может вызывать воспоминания о двухмерной классике, такой как *Pong*, *Pac-Man* или *Donkey Kong*.) В прекрасной книге Рэфа Костера «Разработка игр и теория развлечений» игра определена как интерактивный опыт игрока, реализованный в виде усложняющейся последовательности шаблонных действий, которые он или она осваивает и в конечном счете выполняет [30]. Костер утверждает, что действия освоения и выполнения в игре являются главными в том, что мы называем «развлечение», так же как анекдот становится забавным в тот момент, когда мы улавливаем его смысл.

В данной книге мы сконцентрируемся на наборе игр, включающем двух- и трехмерные виртуальные миры с небольшим количеством игроков — от 1 до 16. К большей части того, что мы изучим, можно также отнести игры, написанные на HTML5/JavaScript в Интернете, чистые головоломки, такие как «Тетрис», или массовые многопользовательские онлайн-игры (MMOG). Но основное внимание будет нацелено на игровые движки, с помощью которых создают шутеры от первого лица, экшен-игры от третьего лица, гонки, файтинговые игры и т. п.

1.2.1. Видеоигры как мягкая симуляция реального времени

Большинство двух- и трехмерных видеоигр являются примерами того, что разработчики назвали бы *мягкой компьютерной симуляцией на основе интерактивного взаимодействия агентов в реальном времени*. Давайте разберем эту фразу по частям, чтобы лучше понять, что все это значит.

В большинстве видеоигр некоторая часть реального или воображаемого мира *моделируется* математически, поэтому им можно управлять с помощью компьютера. Модель является приближением и упрощением реальности (даже если это *воображаемая* реальность), потому что совершенно непрактично переносить в нее

каждую деталь вплоть до атомов или кварков. Следовательно, математическая модель является симуляцией реального или воображаемого игрового мира. Приближение и упрощение — два самых мощных инструмента разработчика игр. При умелом использовании даже очень упрощенная модель иногда почти неотличима от реальности — и даже немного веселее.

Симуляция *на основе агентов* — это такая, в которой взаимодействует ряд различных сущностей, известных как агенты. Под это описание подходит большинство трехмерных компьютерных игр, где агентами являются автомобили, герои, огненные шары, точки силы и т. д. Учитывая агентную природу множества игр, неудивительно, что большая их часть в настоящее время реализована на объектно-ориентированном или по крайней мере близком к таковому языке программирования.

Все интерактивные видеоигры являются *временными симуляторами*. Это означает, что модель виртуального игрового мира *динамична* — состояние игрового мира со временем меняется по мере развития событий и истории игры. Видеоигра также должна реагировать на непредсказуемые входные данные от игрока (игроков) — *интерактивные временные симуляции*. Наконец, большинство видеоигр представляют свои истории и реагируют на действия игроков в реальном времени, превращая их в интерактивные симуляции в реальном времени. Исключением может быть категория пошаговых игр, таких как компьютеризированные шахматы или пошаговые стратегии. Но даже эти типы игр обычно предоставляют пользователю некоторую форму графического пользовательского интерфейса в среде выполнения. Таким образом, для целей этой книги мы будем предполагать, что все видеоигры имеют как минимум *некоторые* ограничения, связанные с работой в реальном времени.

В основе любой системы реального времени лежит концепция *предельного значения (deadline)*. Наглядным примером является требование, чтобы экран обновлялся не менее 24 раз в секунду, чтобы создать иллюзию движения. (Большинство игр обновляют экран с частотой 30 или 60 кадров в секунду, потому что это кратно частоте обновления монитора NTSC.) Конечно, в видеоиграх много и других предельных значений. Симуляция физики может требовать обновления 120 раз в секунду, чтобы оставаться стабильной. Системе искусственного интеллекта персонажа может понадобиться «думать» по крайней мере раз в секунду, чтобы персонаж не показался глупым. Аудиобиблиотека может вызываться не реже одного раза в 1/60 секунды, чтобы аудиобуферы были заполнены и не было слышимого заикания.

«Мягкая» система реального времени — это система, в которой несоблюдение сроков не является катастрофическим. Следовательно, все видеоигры — это *мягкие системы реального времени*: если требование к частоте кадров не соблюдается, то с игроком, как правило, ничего не случается. Сравните это с *жесткой системой реального времени*, в которой пропущенный срок может означать серьезную травму или даже смерть оператора. Система авионики на вертолете или система управления стержнем на атомной электростанции — это примеры жестких систем реального времени.

Математические модели бывают *аналитическими* или *численными*. Например, аналитическая (замкнутая) математическая модель твердого тела, падающего

с постоянным ускорением под действием силы тяжести, обычно записывается следующим образом:

$$y(t) = \frac{1}{2}gt^2 + v_0 t + y_0. \tag{1.1}$$

Аналитическая модель может быть вычислена для любых значений ее независимых переменных, таких как время t в уравнении (1.1), только с учетом начальных условий v_0 и y_0 и постоянной g. Когда такие модели можно создать, пользоваться ими очень удобно. Однако многие математические задачи не имеют решения в аналитическом виде. А в видеоиграх, где ввод пользователя непредсказуем, мы не можем полагаться на аналитическое моделирование всей игры.

Численная модель движения того же твердого тела под действием силы тяжести может быть выражена следующим образом:

$$y(t + \Delta t) = F\left(y(t), \dot{y}(t), \ddot{y}(t)\ldots\right). \tag{1.2}$$

Итак, высота твердого тела в некоторый следующий момент времени ($t + \Delta t$) может быть найдена как функция высоты и ее производной от времени первого, второго и, возможно, более высокого порядка в текущий момент времени t (1.2). Численное моделирование обычно выполняется путем многократных вычислений, чтобы определить состояние системы на каждом дискретном временно́м шаге. Игры работают так же. Основной игровой цикл запускается многократно, и во время каждой итерации различные игровые системы, такие как искусственный интеллект, игровая логика, физическая симуляция и др., получают возможность рассчитать или обновить свое состояние для следующего дискретного временно́го шага. Затем результаты визуализируются путем отображения графики, воспроизведения звука и, вероятно, создания других выходных данных, таких как отдача на джойстике.

1.3. Что такое игровой движок

Термин «игровой движок» появился в середине 1990-х годов в отношении игр-шутеров от первого лица (first-person shooter, FPS), таких как безумно популярная игра *Doom* компании id Software. *Doom* был спроектирован с достаточно четким разделением между основными программными компонентами (такими как система рендеринга трехмерной графики, система обнаружения коллизий или аудиосистема) и графическими ресурсами, игровыми мирами и правилами игры, которые вместе составляли игровой опыт. Ценность этого разделения стала очевидной, когда разработчики начали лицензировать игры и переделывать их в новые продукты, создавая новые изображения, макеты миров, оружие, персонажей, транспортные средства и правила игры при минимальных изменениях в самом движке. Это ознаменовало рождение сообщества моддеров — групп отдельных игроков и небольших независимых студий, которые создавали новые игры, модифицируя

уже существующие с помощью бесплатных наборов инструментов, предоставляемых разработчиками оригинала.

К концу 1990-х годов некоторые игры, такие как *Quake III Arena* и *Unreal*, разрабатывались с учетом возможности повторного использования кода и моддинга. Движки стали делать настраиваемыми с помощью языков сценариев, таких как Quake C компании id Software, а лицензирование движков стало хорошим вторичным источником дохода для их разработчиков. Сегодня разработчики игр могут лицензировать игровой движок и повторно задействовать значительную часть его ключевых программных компонентов для создания игр. Хотя эта практика по-прежнему требует значительных инвестиций в разработку программного обеспечения на заказ, она гораздо более экономична, чем разработка всех основных компонентов движка с нуля.

Граница между игрой и ее движком часто размыта. В некоторых случаях она довольно четкая, в других разработчики даже не пытаются ее провести. В одной игре код рендеринга может «точно знать», как рисовать орка. В другой игре движок рендеринга способен предоставлять универсальный материал и средства затенения и «оркность» может быть полностью определена в данных. Ни одна студия не проводит абсолютно четкого разделения между игрой и движком, и это понятно, учитывая, что определения этих двух компонентов часто меняются по мере разработки дизайна игры.

Вероятно, *архитектура, управляемая данными*, — это то, что отличает игровой движок от программного обеспечения, которое представляет собой игру, но не является движком. Когда игра содержит жестко запрограммированную логику или правила игры либо задействует специальный код для отображения определенных типов игровых объектов, становится трудно или невозможно повторно использовать это программное обеспечение для создания другой игры. Вероятно, нам следует зарезервировать термин «игровой движок» для программного обеспечения, которое расширяемо и может применяться в качестве основы для множества различных игр без значительных модификаций.

Понятно, что такое определение не подразумевает разделения на черное и белое. Мы можем представить шкалу многократного использования, на которой находится каждый движок. На рис. 1.1 показано расположение некоторых известных игр/движков на этой шкале.

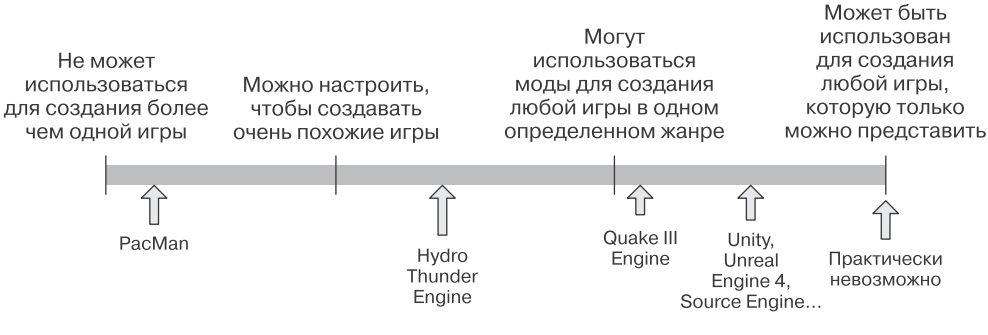

Рис. 1.1. Градация игровых движков по возможности их повторного применения

Можно подумать, что игровой движок должен быть чем-то похожим на Apple QuickTime или Microsoft Windows Media Player: являться универсальным программным обеспечением, способным воспроизводить практически *любой* игровой контент, который только можно себе представить. Однако этот идеальный вариант еще не достигнут и, вероятно, никогда не будет достигнут. Большинство игровых движков разработаны и тщательно настроены для запуска конкретной игры на конкретной аппаратной платформе. И даже самые универсальные кроссплатформенные движки действительно подходят только для создания игр в одном определенном жанре, таких как шутеры от первого лица или гонки. Можно с уверенностью сказать, что чем более универсален игровой движок или компонент промежуточного программного обеспечения, тем менее он оптимален для запуска конкретной игры на конкретной платформе.

Так происходит потому, что разработка любого эффективного компонента программного обеспечения неизменно влечет за собой компромиссы, основанные на предположениях о том, как будет использоваться программное обеспечение, и/или о целевом оборудовании, на котором оно будет работать. Например, движок рендеринга, созданный для работы с закрытыми помещениями, вероятно, не очень подойдет для рендеринга больших открытых пространств. Движок для помещений может задействовать систему порталов или дерево с двоичным разбиением пространства (binary space partitioning, BSP), чтобы гарантировать, что предметы не будут пересекаться стенами или объектами, находящимися ближе к камере. А движок для открытых пространств может применять менее точный механизм наложения или вообще не использовать его. Но он, вероятно, активно задействует техники уровня детализации (level-of-detail, LOD), чтобы гарантировать, что удаленные объекты отображаются минимальным количеством треугольников, а для объектов, которые находятся близко к камере, используется сеть треугольников, реализующая большее разрешение.

Появление более быстрого компьютерного оборудования и специализированных видеокарт наряду со все более эффективными алгоритмами рендеринга и структурами данных постепенно стирает различия между графическими движками для разных жанров. Например, теперь можно использовать движок для шутера от первого лица для создания стратегической игры. Однако компромисс между обобщением и оптимизацией все еще существует. Игру всегда реально сделать более впечатляющей, настроив движок под требования и ограничения конкретной игровой и/или аппаратной платформы.

1.4. Различия движков для разных жанров

Игровые движки, как правило, специфичны для конкретных жанров. Движок, разработанный для файтинга на боксерском ринге, будет сильно отличаться от движка для многопользовательской онлайн-игры (MMOG), или шутера от первого лица, или стратегии в реальном времени (real-time strategy, RTS). Тем не менее они имеют

и много общего — все 3D-игры, независимо от жанра, требуют некоторой формы низкоуровневого пользовательского ввода с джойстика, клавиатуры и/или мыши, определенной формы рендеринга трехмерного меша, некоторой формы отображения информации на экране (heads-up display, HUD), включая отображение текста различными шрифтами, мощную аудиосистему, и этот список можно продолжать. Так, например, несмотря на то что в свое время Unreal Engine был разработан для шутеров от первого лица, позже он успешно использовался для создания игр в ряде других жанров, в том числе в популярной франшизе шутеров от третьего лица *Gears of War* компании Epic Games, экшен-приключении *Batman: Arkham*, представленном Rocksteady Studios, известной игре-файтинге *Tekken 7* компании Bandai Namco Studios и первых трех ролевых шутерах от третьего лица серии *Mass Effect* компании BioWare.

Поговорим о некоторых наиболее распространенных игровых жанрах и рассмотрим примеры технологических требований, характерных для каждого из них.

1.4.1. Шутеры от первого лица

К жанру шутера от первого лица относятся такие игры, как *Quake*, *Unreal Tournament*, *Half-Life*, *Battlefield*, *Destiny*, *Titanfall* и *Overwatch* (рис. 1.2). Исторически они предусматривали относительно медленное пешее передвижение в потенциально большом, но в основном коридорном мире. Тем не менее в современных FPS действие происходит в самых разных виртуальных средах, включая обширные открытые и закрытые помещения. Современная механика перемещения в FPS может включать в себя передвижение пешком, на наземных транспортных средствах, как рельсовых, так и нет, судах на воздушной подушке, лодках и самолетах. Полный обзор этого жанра вы найдете на страницах en.wikipedia.org/wiki/First-person_shooter и https://ru.wikipedia.org/wiki/Шутер_от_первого_лица.

Игры от первого лица, как правило, одни из наиболее технологически сложных в создании. С ними конкурируют только шутеры от третьего лица, игры-платформеры и многопользовательские игры. Это вызвано тем, что FPS стремятся дать игрокам иллюзию погружения в детальный, гиперреалистичный мир. Неудивительно, что многие крупные технологические инновации игровой индустрии возникли из игр этого жанра.

Отличительными чертами движков FPS являются:

- эффективный рендеринг больших трехмерных виртуальных миров;
- отзывчивая механика управления камерой/прицеливания;
- высококачественная анимация виртуального оружия игрока;
- широкий ассортимент мощного оружия в руках;
- нестрогая модель движения персонажа и коллизий, которая часто придает этим играм ощущение плавания;