Содержание

Об авторе

Крис Дикинсон (Chris Dickinson) вырос в Англии, увлекается естественными науками, математикой и компьютерными играми. Получив в 2005 году степень магистра в области физики и электроники, сразу же переехал в Калифорнию, чтобы заниматься научными исследованиями в самом сердце Кремниевой долины. Затем, разочаровавшись в своем выборе, переключился на работу в области разработки программного обеспечения.

За последнее десятилетие он многого достиг, сделал карьеру в области разработки программного обеспечения, стал ведущим разработчиком. Крис в основном занимался программным обеспечением для автоматизации и создавал инструменты для внутреннего тестирования в процессе разработки, но страсть к видеоиграм не потерял. В 2010 году он занялся изучением секретов разработки игр и трехмерной графики и получил степень бакалавра в области разработки игр и программного моделирования. Написал учебник о применении физических законов в играх (*Learning Game Physics with Bullet Physics and OpenGL*, издана в Packt Publishing). В настоящее время продолжает разрабатывать программное обеспечение, свободное время посвящая работе над независимыми игровыми проектами, использующими Unity 3D.

Благодарности

Всего за 5 лет я сумел усвоить огромный объем знаний. Сделать это было бы невозможно без постоянной поддержки коллег, преподавателей, друзей и семьи.

Спасибо моим коллегам по работе, что с пониманием отнеслись к моему перидическому отсутствию, пока я обучался разработке игр в колледже.

А также спасибо моим преподавателям за то, что смогли оперативно донести до меня самое основное и помогли так много узнать о разработке игр за такое короткое время.

Спасибо друзьям, которые всегда были и остаются постоянным источником вдохновения, за их интерес к моей работе.

Спасибо моей семье за предоставленную возможность получить так много знаний, ощущений и любви за такое короткое время.

И конечно, спасибо моей прекрасной жене и лучшему другу Джейми, окружающей меня заботой и поддержкой и помогающей мне творить.

О технических рецензентах

Клиффорд Чампион (Clifford Champion) обладает обширным, многолетним опытом разработки программного обеспечения, полученным во время работы над 3D-играми, интернет-приложениями и программами искусственного интеллекта. Имеет ученые степени в области математики и информатики, присвоенные в Калифорнийском университете в Сан-Диего и в Калифорнийском университете в Лос-Анджелесе. В прошлом Клиффорд работал в компании Havok (ныне часть Microsoft), занимающейся разработкой игровых технологий, и в известной дизайн-студии PlainJoe Studios. В настоящее время возглавляет группу разработки программного обеспечения в компании zSpace (`zspace.com`), специализирующейся на 3D-приложениях виртуальной реальности для учебных заведений и промышленности.

Клиффорда можно найти в Twitter (`@duckmaestro`), и он готов вести дискуссии на любые темы.

Доктор Себастьян Т. Кёниг (Dr. Sebastian T. Koenig) получил степень доктора наук в области систем «человек–машина» в университете Кентербери, Новая Зеландия, за разработку основы системы виртуальной реальности для индивидуализированной когнитивной реабилитации. Имеет диплом психолога в области клинической нейропсихологии и реабилитации с применением систем виртуальной реальности, полученный в университете Регенсбурга, Германия.

Является основателем и генеральным директором Katana Simulations, где курирует проектирование, разработку и проведение когнитивной экспертизы систем моделирования для профессиональной подготовки. Его профессиональный опыт включает более 10 лет клинической работы в сфере когнитивной реабилитации и более 8 лет в области исследования, разработки и тестирования пользователями виртуальной реальности. Он часто выступает на международных конференциях и приглашается в качестве рецензента научных публикаций в области реабилитации, когнитивной психологии, нейропсихологии, программной инженерии, разработки игр, исследований восприятия пользователями игр и виртуальной реальности.

Разработал ряд прикладных программ для когнитивной оценки и профессиональной подготовки. За работу над задачей виртуальной памяти в 2011 году был удостоен престижной премии Laval Virtual в категории медицины и здравоохранения. Другими его достижениями являются: приложение виртуальной реальности для оценки восприятия, разработанное в сотрудничестве с Фондом Кесслера (Нью-Джерси, США), и запатентованное приложение для моторной и когнитивной подготовки JewelMine/Mystic Isle, основанное на Microsoft Kinect и созданное в Калифорнийском институте креативных технологий (Калифорния, США).

Поддерживает веб-сайт www.virtualgamelab.com, где представлены результаты его исследований и проекты разработки программного обеспечения. Сайт также содержит исчерпывающий список учебных материалов, посвященных игровому движку Unity.

Предисловие

Привлекательность – чрезвычайно важный компонент любой игры. Она определяется не только сценарием игры и игровым процессом, но также гладкостью графики, надежностью соединения с многопользовательскими серверами, скоростью реакции на действия пользователя и даже величиной конечного файла приложения, что важно при распространении из хранилищ или из облака. Барьер включения в разработку игр был значительно снижен благодаря появлению недорогих и качественных инструментов, таких как Unity. Однако требования к конечному продукту, предъявляемые игроками, растут с каждым днем. Следует иметь в виду, что все аспекты игры могут и будут тщательно изучены как игроками, так и критиками.

Цели оптимизации производительности тесно связаны с восприятием игры пользователями. Плохо оптимизированная игра, как правило, имеет низкую частоту кадров, зависает, сбоит, реагирует на действия пользователя с опозданием, долго загружается, ведет себя непоследовательно и дергано во время выполнения, работа физического движка часто нарушается и даже имеет чрезмерно высокое энергопотребление (что крайне важно в эпоху мобильных устройств). Наличие хотя бы одной из этих проблем должно приводить в ужас разработчиков, поскольку авторы обзоров склонны делать упор на отрицательных качествах, оставляя в стороне положительные.

Высокая производительность обеспечивается оптимальным использованием всех имеющихся ресурсов, включая ресурсы центрального процессора, такие как частота процессора и объем оперативной памяти, ресурсы графического процессора, такие как объем видеопамяти и ее пропускная способность, и т. д. Поэтому основной задачей оптимизации является такое распределение ресурсов, которое обеспечит эффективное их использование и выполнение высокоприоритетных задач в первую очередь. Даже небольшие, кратковременные задержки и потери производительности снижают привлекательность игры, ухудшают погружение в нее и ограничивают потенциал разработчиков.

Важно также выбрать момент, когда следует остановиться и прекратить заниматься улучшением производительности. В идеальном мире, где время и ресурсы ничем не ограничены, всегда найдется еще один способ достичь более удачного, быстрого и простого решения.

Но в реальном мире должен существовать момент в процессе разработки, когда следует заявить, что продукт достиг приемлемого качества. Если этого не сделать, дальнейшие усилия принесут мало пользы или вообще не дадут ощутимых результатов.

Лучший способ выбрать момент прекращения работ над улучшением производительности – ответить на вопрос: «А пользователь это заметит?» Если ответ на этот вопрос – нет, дальнейшая оптимизация не стоит затраченных на нее усилий. Существует старая поговорка, относящаяся к разработке программного обеспечения:

Преждевременная оптимизация – корень всех зол.

Под преждевременной оптимизацией понимается переработка и рефакторинг кода для повышения производительности, когда нет твердой уверенности в необходимости этих действий, например при полном отсутствии проблем с производительностью (отвечаем на вопрос о заметности для пользователя) или при наличии предположений, что проблема производительности вызвана определенной областью, но нет доказательств, что так и есть на самом деле. Такие ошибки обходятся разработчикам удручающе большим количеством рабочих часов, потраченных впустую.

Цель этой книги – дать вам инструменты, знания и навыки, необходимые для выявления и исправления проблем производительности, независимо от их причин. Подобные проблемы могут вызывать компоненты оборудования, такие как центральный процессор, графический процессор или оперативная память; подсистемы программного обеспечения, такие как подсистема физики, визуализации или же сам движок Unity. Кроме того, чем больше ресурсов удастся сэкономить, тем больше можно будет сделать с помощью движка Unity на одном и том же оборудовании, что позволит реализовать более интересный и динамичный процесс игры.

А это даст вашей игре больше шансов на успех и позволит ей выделиться из массы других, представленных на рынке, куда ежедневно выбрасываются новые, высококачественные игры.

О чем рассказывается в этой книге

Глава 1 «Выявление проблем с производительностью» посвящена знакомству с профилировщиком Unity Profiler и методами профилирования приложений, выявлению узких мест с точки зрения производительности и анализу основных причин их возникновения.

Глава 2 «*Приемы разработки сценариев*» описывает эффективные приемы разработки сценариев для Unity на языке C#, минимизации дублирования компонентов, оптимизации взаимодействий между объектами и многое другое.

Глава 3 «*Преимущества пакетной обработки*» посвящена изучению применения систем динамической и статической пакетной обработки в Unity для снижения нагрузки на систему визуализации.

Глава 4 «*Привнесение искусства*» поможет разобраться в закулисных процессах работы с ресурсами и узнать, как избежать распространенных ошибок при их импорте, сжатии и перекодировании.

Глава 5 «*Разгон физического движка*» посвящена особенностям физической системы Unity для 3D- и 2D-игр и правильной организации физических объектов для повышения производительности.

Глава 6 «*Динамическая графика*» подробно рассматривает систему визуализации и способы решения проблем производительности приложений, возникающих в системе визуализации, узкими местами которой являются графический процессор или центральный процессор, а также специальные методы для мобильных устройств.

Глава 7 «*Мастерство управления памятью*» анализирует внутренние рабочие процессы движка Unity, фреймворка Mono и управление памятью в этих компонентах, чтобы защитить приложение от распределения памяти в куче и сборки мусора во время выполнения.

Глава 8 «*Тактические советы и приемы*» знакомит со множеством полезных методов, используемых профессионалами для улучшения управления рабочими процессами и сценами.

Что потребуется для работы с книгой

Основное внимание в этой книге уделяется возможностям версии Unity 5.x. Большинство методов, описанных в книге применимы к проектам, созданным в Unity 4.x, но для некоторых из них требуется обновить Unity 4 до версии Pro (например, окклюзивная выбраковка, статическая пакетная обработка и даже сам профилировщик).

Кому адресована эта книга

Эта книга адресована разработчикам среднего и продвинутого уровня, уже имеющим опыт работы с большей частью функций Unity, заинтересованных в улучшении производительности игр или в решении конкретных проблем. Если вы испытываете проблемы, связанные

с перегрузкой процессора, скачками нагрузки во время выполнения, замедленным доступом к памяти, фрагментацией, уборкой мусора, снижением частоты графического процессора или пропускной способностью памяти, эта книга научит вас выявлять источники проблем и выбирать способы уменьшения их воздействия на приложение.

Вам потребуется знание языка C# при чтении разделов, содержащих сценарии и посвященных использованию памяти, а также базовое понимание библиотеки Cg – при рассмотрении оптимизации шейдеров.

Соглашения

В этой книге используется несколько разных стилей оформления текста для выделения разных видов информации. Ниже приводятся примеры этих стилей и объясняется их назначение.

Программный код в тексте, имена таблиц баз данных, имена папок, имена файлов, расширения файлов, фиктивные адреса URL, пользовательский ввод и ссылки в Twitter будут выглядеть так: «Главной функцией обратного вызова Unity является функция обновления `Update()`.»

Блоки программного кода оформляются так:

```
public class TestComponent : MonoBehaviour {
    void Update() {
        if (Input.GetKeyDown(KeyCode.Space)) {
            PerformProfilingTest();
        }
    }
}
```

Когда потребуется привлечь ваше внимание к определенному фрагменту в блоке программного кода, он будет выделяться жирным шрифтом:

```
public class TestComponent : MonoBehaviour {
    void Update() {
        if (Input.GetKeyDown(KeyCode.Space)) {
            PerformProfilingTest();
        }
    }
}
```

Новые термины и **важные слова** будут выделены жирным. Текст, отображаемый на экране, например в меню или в диалогах, будет

оформляться так: «Пороговое значение для перехода в состояние сна может быть изменено при помощи выбора пункта **Edit ⇨ Project Settings ⇨ Physics ⇨ Sleep Threshold** (Редактирование ⇨ Параметры проекта ⇨ Физика ⇨ Интервал для перехода в состояние сна)».

Так будут оформляться предупреждения и важные примечания.

Так будут оформляться советы и рекомендации.

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 прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Загрузка исходного кода примеров

Загрузить файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф в разделе «Читателям – Файлы к книгам».

Список опечаток

Хотя мы приняли все возможные меры, чтобы удостовериться в качестве наших текстов, ошибки всё равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии этой книги.

Если вы нашли опечатку, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в Интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и «Packt» очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в Интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли принять меры.

Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Глава 1

Выявление проблем с производительностью

Оценка производительности большинства программных продуктов является исследовательским процессом, включающим: определение максимальных поддерживаемых параметров (количества одновременно работающих пользователей, максимального объема доступной памяти, ресурсов центрального процессора и т. д.); выполнение нагрузочного тестирования путем воссоздания реальных ситуаций; инструментальный сбор результатов тестирования; анализ данных для выявления узких мест; комплексный анализ причин; внесение изменений в конфигурацию или код приложения для исправления проблем; повторение всего перечисленного еще раз.

Разработка игр является творческим процессом, но это вовсе не означает, что ее не следует рассматривать с объективной и исследовательской точек зрения. Игра направлена на конкретную целевую аудиторию, что и определяет ее аппаратные ограничения. Нужно протестировать приложение, собрать данные о работе нескольких компонентов (центральный процессор, графический процессор, память, физика, визуализация и т. д.) и сравнить их с требуемыми характеристиками. Эти данные используются для выявления узких мест в работе приложения, с помощью дополнительных инструментов определяются главные причины проблем, после чего проблемы рассматриваются с разных точек зрения.

Для успешного выполнения этого процесса необходимы инструменты и знания, методы, с которыми знакомит эта глава. Они будут использоваться на протяжении всей книги для выявления проблем с производительностью и определения их главных причин. Такие навыки составляют методику выявления и анализа проблем с произ-

водительностью **Unity-**приложения и выбора мест для внесения изменений. В этой главе будет проделана подготовительная работа для освоения материала остальных глав, описывающего пути решения выявленных проблем.

Начнем с изучения **профилировщика Unity Profiler** и его многочисленных функций. Затем опишем несколько способов выявления узких мест и в завершение приведем несколько советов, касающихся применения этих технологий.

Профилировщик Unity Profiler

Профилировщик Unity Profiler встроен в **редактор Unity** и обеспечивает целенаправленный поиск узких мест производительности путем сбора информации о работе компонентов **Unity3D** во время выполнения, то есть:

- использование центрального процессора отдельно каждым компонентом движка Unity3D;
- затраты на визуализацию;
- использование графического процессора GPU на разных этапах работы графического конвейера;
- использование памяти;
- затраты на воспроизведение аудиоэффектов;
- использование физического движка.

Начиная с **Unity 5.0**, компания Unity Technologies открыла доступ к профилировщику в редакции Personal Edition (прежде эта редакция называлась Free Edition).
Пользователям, использующим редакцию Unity 4 Free Edition, необходимо обновить ее до Unity 5 или приобрести лицензию для редакции Unity 4 Pro Edition.

Однако использование профилировщика влечет дополнительные накладные расходы. При включении флагов профилирования в настройках компилятора будут генерироваться журнал событий времени выполнения и прочий код поддержки профилирования, что вызовет дополнительные затраты ресурсов центрального процессора и памяти во время выполнения. Однако затраты на профилирование не очень значительны и не приводят к непредсказуемой разнице в поведении при включенном и отключенном профилировщике.

Кроме того, всегда избегайте использования режима редактирования во время профилирования и проведения сравнительного анализа

из-за накладных расходов на работу интерфейса редактора и затрат памяти на размещение различных объектов и компонентов. Для получения более точных и надежных данных проверку приложения лучше выполнять автономно и на целевом устройстве.

Читатели, знакомые с процедурой подключения профилировщика к приложениям, могут пропустить следующий раздел и перейти сразу разделу «*Окно профилировщика*».

Запуск профилировщика

Начнем с краткого обзора способов подключения профилировщика Unity в различных контекстах:

- к локальным экземплярам приложения через редактор или автономное профилирование самого редактора;
- к локальным экземплярам приложения в **веб-плеере Unity**;
- к удаленным экземплярам приложения на iOS-устройствах (iPad или iPhone);
- к удаленным экземплярам приложения на Android-устройствах (планшет или телефон с ОС Android).

Далее мы кратко рассмотрим особенности настройки профилировщика в каждом из этих контекстов.

Редактор или автономные экземпляры

Запустить профилировщик можно только из редактора Unity, после чего его следует подключить к запущенному экземпляру приложения. Здесь возможны случаи, когда игра запущена из редактора или как автономное приложение на локальном или удаленном устройстве, или когда требуется выполнить профилирование самого редактора.

Чтобы запустить профилировщик, выберите в меню редактора пункт **Window ⇨ Profiler** (Окно ⇨ Профилировщик), как показано на рис. 1.1. При работе редактора в режиме воспроизведения в окне профилировщика появится отчет.

Для профилирования автономных проектов при их сборке следует установить флаги Use Development Mode (Использовать режим разработки) и Autoconnect Profiler (Автоматически подключать профилировщик).

Выбор экземпляра приложения, запущенного в редакторе (в режиме воспроизведения) или автономного (собранного отдельно и рабо-

тающего в фоновом режиме), производится с помощью меню **Active Profiler** (Активный профилировщик) в окне профилировщика, как показано на рис. 1.2.

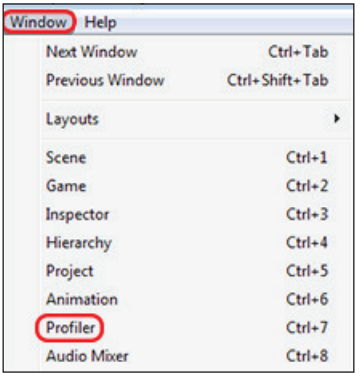

Рис. 1.1 ❖ Запуск профилировщика Unity

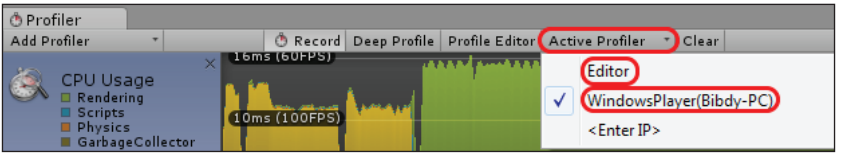

Рис. 1.2 ❖ Профилировщик Unity

Профилирование редактора

Профилирование редактора, например редактора сценариев, можно включить выбором пункта меню **Profile Editor** (Профилирование редактора) в окне профилировщика, как показано на рис. 1.3. Обратите внимание, что для этого требуется отметить параметр **Editor** (Редактор) в меню **Active Profiler** (Активный профилировщик).

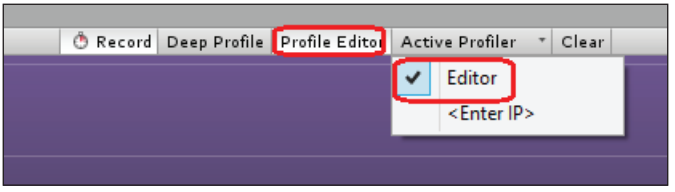

Рис. 1.3 ❖ Профилирование редактора сценариев

Подключение к веб-плееру Unity

Профилировщик можно также подключить к экземпляру приложения в веб-плеере Unity, запущенном в браузере. Это позволит выполнить профилирование веб-приложения в наиболее реальной среде с помощью целевого браузера и протестировать несколько типов браузеров для выявления несоответствий.

1. Перед сборкой приложения для веб-плеера включите флаг **Use Development Mode** (Использовать режим разработки).
2. Запустите скомпилированное для веб-плеера приложение в браузере, активируйте окно браузера и, удерживая нажатой клавишу **Alt** (*Option* на компьютере Mac), щелкните правой кнопкой мыши на объекте веб-плеера в браузере, чтобы открыть меню **Release Channel Selection** (Выбор канала вывода). Затем выберите канал **Development** (Разработка), как показано на рис. 1.4.

Обратите внимание, что изменение параметра Release Channel (Канал вывода) вызовет перезапуск приложения в веб-плеере.

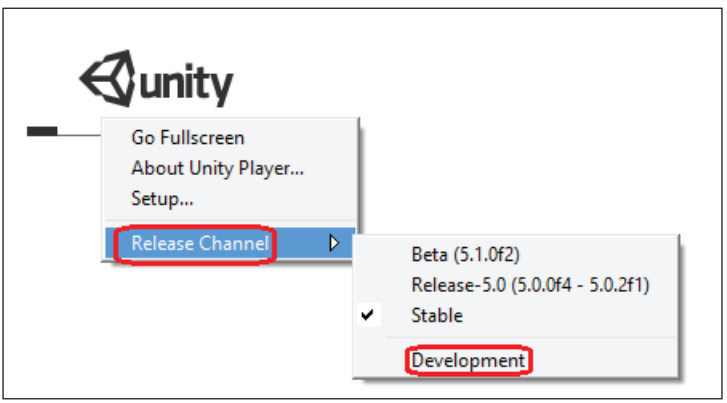

Рис. 1.4 ❖ Выброр канала вывода

3. Откройте профилировщик в окне редактора Unity и выберите пункт **Active Profiler ⇨ WindowsWebPlayer(COMPUTER-NAME)** или **Active Profiler ⇨ OSXWebPlayer(COMPUTER-NAME)** в зависимости от используемой операционной системы, как показано на рис. 1.5.

После этого в окне профилировщика должна появиться собранная им информация.

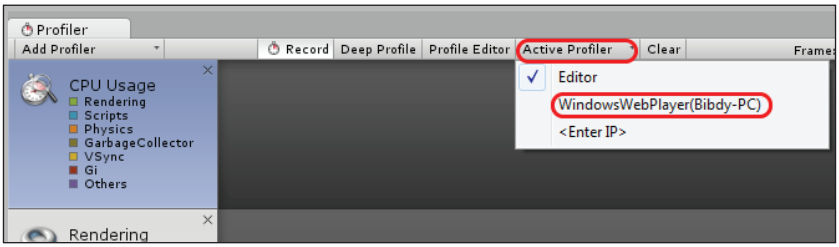

Рис. 1.5 ❖ Выбор веб-плеера

Подключение к удаленным iOS-устройствам

Профилировщик можно также подключить к активному экземпляру приложения, запущенному на удаленном iOS-устройстве, таком как iPad или iPhone, например через Wi-Fi-подключение. Чтобы подключить профилировщик к устройству Apple, выполните следующие действия:

Обратите внимание, что подключение к устройству Apple возможно, только если профилировщик выполняется на устройстве Apple Mac.

1. Перед сборкой приложения включите флаги **Use Development Mode** (Использовать режим разработки) и **Autoconnect Profiler** (Автоматически подключать профилировщик).
2. Подключите оба устройства, iOS и Mac, к локальной сети или сети ADHOC WiFi.
3. Соедините устройства iOS и Mac кабелем USB или Lightning.
4. Запустите сборку приложения, выбрав в меню пункт **Build & Run** (Собрать и запустить), как обычно.
5. Откройте окно профилировщика в редакторе Unity и выберите устройство в меню **Active Profiler** (Активный профилировщик).

После этого в окне профилировщика должна появиться собранная им информация.

Для передачи данных профилировщик использует порты с номерами от 54998 до 55511. Убедитесь, что эти порты доступны для исходящего трафика, если в сети используется брандмауэр.

Подключение к удаленным Android-устройствам

Существуют два способа подключения профилировщика к Android-устройству: через Wi-Fi-соединение или с помощью инструмента отладки **Android Debug Bridge (ADB)**. ADB – это набор инструментов отладки, входящий в состав пакета Android SDK.