1 Приступаем к работе

Эта книга очень похожа на поваренную и предназначена для решения конкретных распространенных проблем, которые могут возникнуть во время разработки игры для платформы Android. Все рекомендации испытаны, им легко следовать, а также можно без труда адаптировать к разным ситуациям.

Допустим, вы знаете, что следует класть в куриный суп, но не знаете, как превратить курицу и овощи в суп. С точки зрения стандартной поваренной книги, следует довериться пошаговому рецепту для приготовления супа. Во многом таким же образом вы можете использовать эту книгу, чтобы узнать, как закодировать конкретные сценарии игры от создания заставки до обнаружения столкновений при уничтожении врага.

Перед переходом к рецептам важно установить соответствующие фреймворки, чтобы получить от них максимальную пользу. В этой главе мы обсудим, какие навыки и инструменты понадобятся, чтобы получить от этой книги полную отдачу.

1.1. Что вам понадобится

Программирование игр как дисциплина довольно сложно, и его освоение может занять годы. Однако базовые понятия программирования игр на самом деле довольно просты в освоении и могут быть применены повторно во многих ситуациях. Количество времени, которое вы вкладываете в игры и код, в конечном счете определяет, насколько успешными будете вы и ваши игры. Каждый сталкивается с какой-нибудь проблемой при кодировании, и независимо от того, как долго вы будете почесывать затылок и использовать Google, вы не сможете получить готовое решение для нее. Данная же книга как раз сможет дать ответы на многие вопросы.

1.2. Навыки и опыт

Это издание не предназначено для новичков или пользователей, которые не имеют опыта разработки игр. Читая эту книгу, вы не узнаете, как написать игру с нуля. Это не означает, что вы должны быть профессиональным разработчиком игр, чтобы иметь возможность использовать это пособие. Напротив, предполагается, что

если вы его читаете, то, скорее всего, являетесь любителем, тем, кто, возможно, пытался создать одну-две игры (быть может, именно для Android) и столкнулся с проблемой преобразования ваших знаний разработки для платформы Android.

Эта книга призвана оказать вам помощь в решении конкретных вопросов или сценариев. Таким образом, вы должны иметь по крайней мере практические знания разработки игр и по меньшей мере быть знакомыми с базовыми сведениями о разработке для Android. Ни одна тема не будет рассматриваться с нуля.

Платформа Android была разработана на базе Java, поэтому вы должны также обладать достаточными практическими знаниями о разработке на этом языке. В данной книге вы не встретите разъяснений, как работает Java, и в некоторых сценариях будет подразумеваться, что вы знаете, как организован этот язык.

Впрочем, возможно, у вас есть некоторый опыт разработки игр на другой платформе, такой как Windows, или даже бизнес-приложений на Java, но нет такового для OpenGL ES. Большую часть времени разработка игры для Android представляет собой работу именно с OpenGL ES. По этой причине вторая часть данной главы посвящена знакомству с библиотекой OpenGL ES, где объясняется, почему эта библиотека так важна для Android. Если у вас уже есть опыт работы с OpenGL ES, полагаю, что вы можете пропустить раздел «Первый взгляд на OpenGL ES» данной главы.

В общем, если у вас есть страсть к разработке игр и платформе Android, но во время работы вы столкнулись с проблемами, эта книга для вас. Если вы уже начали создавать игру и возникли какие-то вопросы или вы находитесь в начальной стадии разработки и не знаете точно, что делать дальше, это пособие поможет вам преодолеть наиболее распространенные препятствия.

1.3. Версии программного обеспечения

В данный момент вы готовы погрузиться в поиск решений для вашего сценария игры для Android. Какие инструменты понадобятся, чтобы начать ваше путешествие?

Эта книга ориентирована на версии Android 4.1 и 4.2 Jelly Bean. Если вы не работаете в Jelly Bean, рекомендуется обновить SDK, перейдя по ссылке http://developer.android.com/sdk/. Тем не менее приведенные примеры должны работать и на Android 4.0 Ice Cream Sandwich. Существует множество ресурсов, которые помогут вам скачать и установить SDK (и соответствующие необходимые Java-компоненты). В этой книге установка SDK рассматриваться не будет.

Вам также понадобится использовать версию Eclipse, которая называется Kepler. Одной из главных особенностей среды разработки Eclipse является тот факт, что она будет поддерживать несколько версий Android SDK. Таким образом, можно быстро проверить свой код в Jelly Bean, Ice Cream Sandwich или даже Gingerbread, если это необходимо. Можете использовать практически любую интегрированную среду разработки на языке Java или текстовый редактор для написания кода на Android, однако я предпочитаю Eclipse из-за ее особенностей, позволяющих при-

менить хорошо продуманные надстройки, которые тесно интегрированы со многими рутинными ручными операциями компиляции и отладки кода для Android. Eclipse является официальной средой, рекомендованной для разработки на Android компанией Google, создателем платформы Android.

Если у вас нет Eclipse Kepler и вы хотите попробовать, ее можно бесплатно загрузить с сайта http://eclipse.org.

Эта книга не будет углубляться в процесс загрузки или установки Eclipse. Есть много ресурсов, в том числе на собственном сайте Eclipse и форуме разработчиков Android, которые помогут вам установить эту среду.

СОВЕТ

Если вы никогда не устанавливали Eclipse или аналогичную среду разработки, тщательно следуйте инструкциям по инсталляции. Последнее, что вам нужно, — это некорректно установленная среда, которая помешает писать отличные игры.

В следующем разделе мы рассмотрим один из наиболее часто используемых при создании игр на Android инструментов, OpenGL ES.

1.4. Первый взгляд на OpenGL ES

OpenGL ES (OpenGL for Embedded Systems — OpenGL для встраиваемых систем) является API для работы с графикой с открытым исходным кодом, который поставляется вместе с Android SDK. Ядро платформы Android предоставляет ограниченную поддержку работы с графикой, поэтому крайне сложно, если вообще возможно, создать всю игру без использования OpenGL ES. Вызовы методов работы с графикой из ядра Android медленны, неуклюжи и за редким исключением не должны использоваться для создания игр. Здесь вступает OpenGL ES.

OpenGL ES был включен в Android в той или иной форме с самого начала существования платформы. В более ранних версиях Android реализация OpenGL ES представляла собой ограниченную версию OpenGL ES 1. По мере развития Android и появления ее версий добавлялись многофункциональные реализации OpenGL ES. Начиная с Jelly Bean, разработчики игр получили доступ к OpenGL ES 2.

Давайте выясним, что именно OpenGL ES делает для разработки и каким образом.

1.5. Как OpenGL ES работает с Android

Open GL ES общается с графическим аппаратным обеспечением гораздо более непосредственным образом, чем ядро Android. Это означает, что вы отправляете данные непосредственно оборудованию, которое отвечает за их обработку. Вызовы ядра Android проходят через основные Android-процессы, потоки и интерпретатор, прежде чем добраться до графического оборудования. Игры, написанные для платформы Android, могут достичь приемлемого уровня скорости и воспроизведения,

общаясь непосредственно с GPU (Graphics Processing Unit — графический процессор).

Текущие версии Android имеют возможность использовать вызовы OpenGL ES 1 или OpenGL ES 2/3. Между двумя версиями есть большая разница, поэтому выбор сыграет важную роль в определении, кто сможет запустить игру, а кто — нет.

ПРИМЕЧАНИЕ

Все примеры этой книги, которые включают код работы с OpenGL ES, представлены в двух вариантах — для OpenGL ES 1 и для OpenGL ES 2/3.

OpenGL ES облегчает взаимодействие между вашей игрой и графическим оборудованием одним из двух способов. Тип GPU Android-устройства определит, какую версию OpenGL ES следует использовать, что обозначит и способ взаимодействия OpenGL с оборудованием. На современном рынке существуют два основных вида графического оборудования, и, поскольку они очень отличаются друг от друга, различные версии OpenGL ES обязаны взаимодействовать с ними.

Эти типы оборудования содержат средства фиксированной функциональности либо шейдеры. В следующих разделах приведен обзор взаимодействия OpenGL ES и средств фиксированной функциональности, а также OpenGL ES и шейдеров. Имейте в виду: OpenGL ES версии 1 работает со средствами фиксированной функциональности, а OpenGL ES 2/3 — с шейдерами.

1.6. Конвейеры с фиксированной функциональностью

У более старых устройств будет оборудование, которое использует средства фиксированной функциональности. В подобных графических процессорах имеется конкретное специализированное аппаратное средство для выполнения функций. Такие функции, как преобразования, осуществляются специальными частями GPU, над которыми разработчик зачастую не имеет контроля. Это означает, что вы должны просто передать свои вершины в GPU, указать ему выполнить преобразование, и на этом все.

Примером этого может быть случай, когда у вас множество вершин, представляющих куб, и вы хотите переместить его из одного места в другое. Это достигается путем передачи вершин в средство фиксированной функциональности, а далее следует указать оборудованию преобразовать эти вершины. Аппаратная часть затем выполнит для вас все необходимые действия и определит окончательное расположение куба.

В следующем фрагменте кода вы увидите очень упрощенную версию того, что вы могли бы сделать с помощью конвейера с фиксированной функциональностью. Вершины `myVertices` отправляются в конвейер. Затем используется метод `GlTranslatef()` для перемещения вершин на новые позиции. Последующие действия матричной математики будут сделаны в GPU за вас.

```
private float myVertices[] = {
  0.0f, 0.0f, 0.0f,
  1.0f, 0.0f, 0.0f,
  1.0f, 1.0f, 0.0f,
  0.0f, 1.0f, 0.0f,
};

// Другой код для работы с OpenGL или бизнес-логика игры

gl.glMatrixMode(GL10.GL_MODELVIEW)
gl.glLoadIdentity();
gl.glTranslatef(0f, 1f, 0f);
```

Преимущество этого подхода заключается в том, что с помощью специализированных аппаратных средств функция может быть выполнена очень быстро. Оборудование может исполнять функции довольно быстрыми темпами, а специализированные аппаратные средства или аппаратные средства, имеющие очень ограниченный набор функций, могут выполнять функции еще быстрее.

Недостаток данного подхода со средствами фиксированной функциональности состоит в том, что аппаратные средства не могут быть изменены или переконфигурированы, в отличие от программного обеспечения. Это ограничивает полезность аппаратного средства. Кроме того, специализированное аппаратное средство может выполнять свои функции только из одной очереди. Это означает, что конвейер часто можно замедлить, если поместить в очередь на обработку большое количество элементов.

Новые устройства, с другой стороны, имеют графические процессоры, использующие шейдеры. Шейдер — это также специализированное оборудование, но гораздо более гибкое, чем его предшественник с фиксированной функциональностью. OpenGL ES работает с шейдерами, используя язык программирования GLSL (OpenGL Shading Language), который может выполнять любое число программируемых задач.

1.7. Шейдеры

Шейдер — это программа, написанная на языке шейдеров, выполняющем все те же функции, которые может осуществить средство с фиксированной функциональностью. OpenGL ES 2/3 работает с двумя различными типами шейдеров — вершинными и фрагментными.

Вершинные шейдеры

Вершинные шейдеры выполняют операции над вершинами, такие как изменение цвета, расположения и текстуры вершины. Шейдер будет работать над каждой переданной вершиной. Это означает, что, если у вас есть фигура, сделанная из 256 вершин, такой шейдер будет обрабатывать каждую из них.

Вершины могут быть маленькими или большими. Тем не менее во всех случаях вершины будут состоять из множества пикселов. Вершинные шейдеры будут работать над всеми пикселами одной вершины одним и тем же образом. Все пикселы одной вершины рассматриваются как единое целое. Когда вершинный шейдер закончен, он передает вершину растеризатору, а затем в пиксельный шейдер.

Следующий код представляет собой вершинный шейдер:

```
private final String vertexShaderCode =
        "uniform mat4 uMVPMatrix;" +
        "attribute vec4 vPosition;" +
        "attribute vec2 TexCoordIn;" +
        "varying vec2 TexCoordOut;" +
        "void main() {" +
        "  gl_Position = uMVPMatrix * vPosition;" +
        "  TexCoordOut = TexCoordIn;" +
"}";
```

Фрагментные шейдеры

В то время как вершинные шейдеры обрабатывают данные для целой вершины, фрагментные шейдеры (их иногда называют пиксельными шейдерами) работают с каждым пикселом. Фрагментные шейдеры выполняют расчеты для освещения, затенения, тумана, цвета кожи и всего прочего, что может повлиять на отдельные пикселы вершины. Обработка градиентов и освещения выполняется на пиксельном уровне, поскольку они могут оказать разные эффекты на различные части вершины.

Следующий код представляет собой фрагментный шейдер:

```
private final String fragmentShaderCode =
    "precision mediump float;" +
    "uniform vec4 vColor;" +
    "uniform sampler2D TexCoordIn;" +
    "varying vec2 TexCoordOut;" +
    "void main() {" +
    "  gl_FragColor = texture2D(TexCoordIn, TexCoordOut);" +
    "}";
```

ПРИМЕЧАНИЕ

Существуют и другие типы шейдеров, включая шейдеры замощения и геометрические. Они могут быть необязательными и обрабатываются в аппаратной части устройства. Вы практически не знаете, как их использовать.

Большинство устройств Android теперь может обрабатывать комбинацию вызовов OpenGL ES 1 и OpenGL ES 2. Некоторые разработчики, плохо знакомые с программированием шейдеров, будут продолжать использовать средства с фиксированной функциональностью для выполнения действий над видимой областью и прочих динамических операций. Следует помнить, что по мере развития OpenGL ES поддержка конвейеров с фиксированной функциональностью по-

степенно сокращается. В самом ближайшем будущем наступит время, когда для работы с OpenGL ES вы будете вынуждены использовать только шейдеры, поэтому, если вы только начали работать с OpenGL ES, я бы посоветовал использовать шейдеры как можно чаще.

1.8. Как работают игры

При разработке игры или игрового цикла код должен быть выполнен в установленном порядке в определенное время. Знание потока выполнения имеет решающее значение в понимании, как должен быть написан ваш код.

В основе каждой видеоигры лежит игровой движок, частью которого является игровой цикл. Как следует из названия, движок игры — это код, который управляет игрой. Каждая игра независимо от жанра — RPG, шутер, платформер или даже стратегия в реальном времени — требует полнофункциональный игровой движок для запуска.

Движок игры обычно работает в собственном потоке, используя максимально возможное количество ресурсов. Все задачи, которые должна выполнять игра, от графики до звука, выполняются в игровом движке.

ПРИМЕЧАНИЕ

Движок любой игры намеренно построен так, чтобы быть универсальным. Это позволит использовать его в разных ситуациях и, возможно, для различных игр.

Один из наиболее популярных многоцелевых игровых движков — Unreal Engine. Впервые разработанный примерно в 1998 году компанией Epic для их шутера от первого лица, Unreal, он был использован в сотнях игр.

Unreal Engine легко адаптируется и работает с различными типами игр, не только с шутерами от первого лица. Общая структура и гибкость делают его популярным как у профессионалов, так и у разработчиков-любителей.

Скорее всего, вы уже использовали сторонние игровые движки. Существует много бесплатных и платных движков, доступных для Android. Эта книга, однако, станет хорошим подспорьем для создания собственного игрового движка.

Многие процессы игровых движков сторонних разработчиков намеренно запутаны, и вы не можете отладить движок или изменить его код. Если у вас возникли проблемы, вы, как правило, должны обратиться к компании, которая создала движок, и разработчику может потребоваться некоторое время, чтобы все исправить, если, конечно, он вообще собирается это делать. Это может стать одним из основных недостатков использования стороннего игрового движка.

Никакой опыт не может заменить создание собственного игрового движка. Эта книга предполагает, что вы собираетесь делать именно это. Многие задачи, которые будут решаться далее, предполагают, что вы пытаетесь написать игровой движок на Android и столкнулись с какими-то распространенными проблемами.

Что же делает движок игры? Он выполняет всю тяжелую работу, начиная с воспроизведения звуковых эффектов и фоновой музыки и заканчивая выводом графики

на экран. Ниже приводится частичный список функций, выполняемых типичным движком игры:

- отрисовка графики;
- анимация;
- звук;
- определение столкновений;
- искусственный интеллект (ИИ);
- физика (без учета столкновений);
- многопоточность и управление памятью;
- работа с сетью;
- интерпретатор команд (ИК).

В основе игрового движка находится игровой цикл. В то время как движок может справиться с чем угодно, начиная с единоразовой установки буферов вершин и получения изображений, игровой цикл предоставляет фактический код выполнения игры. Все игры выполняются в цикле кода. Чем быстрее будет этот цикл, тем лучше игра будет работать, тем быстрее реагировать на действия игрока и тем более плавно будет происходить действие на экране. Весь код, необходимый для отрисовки игры на экране, перемещения игровых объектов, определения счета, обнаружения столкновений и подтверждения или аннулирования элемента, выполняется в игровом цикле.

Игровой цикл — это фрагмент кода, который выполняется в непрерывном цикле. Цикл начинается вместе с запуском игры и не останавливается (за некоторыми исключениями), пока игра не завершится. Рассмотрим действия, которые должен выполнять игровой цикл на каждой итерации. Типичный цикл игры может делать следующее:

- интерпретацию команд устройства ввода;
- отслеживание символов и/или фона, чтобы гарантировать, что они не переместятся куда не нужно;
- определение столкновений между объектами;
- перемещение фона по мере необходимости;
- отрисовку фона;
- отрисовку любого количества неподвижных объектов;
- расчет физики подвижных объектов;
- передвижение всего оружия/пуль/объектов, которые были перемещены;
- отрисовку оружия/пуль/объектов;
- перемещение независимых объектов;
- отрисовку символов;
- воспроизведение звуковых эффектов;

- ответвление потоков для непрерывного воспроизведения фоновой музыки;
- отслеживание счета игрока;
- отслеживание и управление сетью или несколькими игроками.

Это далеко не полный список, но данные действия выполняются в игровом цикле наиболее часто.

Это очень важно для оптимизации вашего игрового кода. Чем больше вы сможете оптимизировать код игрового цикла, тем быстрее он будет выполнять все вызовы, что поможет улучшить впечатление от игры в целом. В следующем разделе мы рассмотрим, как платформа Android работает с игровыми движками и игровыми циклами.

1.9. Android и игровые движки

Android поставляется с мощным, полнофункциональным графическим API под названием OpenGL ES. Однако так ли необходим этот OpenGL ES для разработки игр? Вместо того чтобы проходить через трудности овладения довольно низкоуровневым API, таким как OpenGL ES, не проще ли написать игру, которая использует вызовы ядра Android?

Ответ следующий: ожидая эффективного запуска игры, нельзя рассчитывать на вызовы ядра Android для выполнения этой тяжелой работы. Да, большинство версий Android действительно имеет методы ядра, которые могут позаботиться о каждом пункте приведенного выше списка. Тем не менее системы отрисовки, работы со звуком и памятью в Android рассчитаны на общие задачи и могут адаптироваться к любому количеству непредсказуемых вызовов, не уделяя какой-либо определенной ситуации особого внимания. Непредсказуемость заключается в одном — в затратах времени. Методы API ядра Android, которые могли бы позаботиться о задачах, необходимых для запуска игры, содержат большое количество постороннего кода. Это допустимо, если вы пишете бизнес-приложение, но не для игр. Подобные «накладные расходы» замедляют выполнение кода, для написания игр требуется что-то более мощное.

Чтобы игра работала бесперебойно и быстро, код должен избегать этих огрехов, присущих методам API ядра Android, то есть игра должна общаться непосредственно с графическим оборудованием для отрисовки графики, со звуковой картой, чтобы воспроизводить звуковые эффекты, и т. д. Если бы вы использовали стандартные системы памяти, графики и звука, которые доступны в API ядра Android, задачи вашей игры могли бы помещаться в одну очередь с задачами других Android-приложений, запущенных в системе. Это предполагает, что игра выполнялась бы очень медленно.

По данной причине игровые движки и игровые циклы почти всегда написаны на языках низкого уровня или с помощью особых API, таких как OpenGL ES. Как вы увидите в гл. 2, низкоуровневые языки предлагают более прямой путь к аппаратной части системы. Игровой движок должен быть в состоянии принять код

и команды от движка и передавать их непосредственно аппаратному обеспечению. Это позволяет игре выполняться быстро и управлять всем необходимым для обеспечения хорошего впечатления.

1.10. Резюме

В данной главе мы рассмотрели инструменты, которые вы должны будете использовать, чтобы получить максимальную отдачу от этой книги. Версия Android Jelly Bean, Eclipse Kepler и опыт работы с Java и/или опыт разработки игр будут помогать вам на протяжении всей книги. Мы также рассмотрели различия между OpenGL ES версий 1 и 2/3 и разницу между средствами с фиксированной функциональностью и шейдерами.

В следующих главах описаны проблемы, возникающие при создании игрового движка. В частности, будут рассмотрены вопросы, которые могут возникнуть с различными способами загрузки изображения. Существует множество различных форматов изображений и всего несколько способов их загрузки и отображения на экране. Скорее всего, если вы уже пытались сделать это, то столкнулись с довольно неожиданными результатами.