УДК 378.016: 004.031

Газейкин Е.В., Газейкина А.И.

ОБУЧЕНИЕ БУДУЩИХ ИТ-СПЕЦИАЛИСТОВ РАЗРАБОТКЕ ИГРОВЫХ ПРИЛОЖЕНИЙ ДЛЯ ОПЕРАЦИОННОЙ СИСТЕМЫ ANDROID

Аннотация

Статья посвящена технологии разработки игровых приложений под операционную систему Android посредством libGDX. Рассматривается специфика разработки приложений на платформе Android, исследуются инструменты и сервисы для разработки Android-приложений.

Ключевые слова: геймплей, фреймворк, игровые приложения, технологии разработки приложений, программирование, информационные технологии, подготовка специалистов, обучение программированию.

Gazeykin E.V., Gazeykina A.I.

TRAINING FUTURE IT SPECIALISTS DEVELOPMENT GAME APPLICATIONS FOR OPERATING SYSTEM ANDROID

Abstract

The article is devoted to the technology of developing gaming applications for the Android operating system through libGDX. The specifics of developing applications on the Android platform are examined, and the tools and services for developing Android applications are explored.

Keywords: gameplay, framework, gaming applications, application development technologies, programming, information technologies, training of specialists, programming training.

Курс программирования является одним из основных курсов предметной подготовки будущего специалиста в области информационных технологий. В процессе освоения содержания этой дисциплины у студента формируется представление о технологии создания программных средств современных информационно-коммуникационные технологий. Курс формирует у студентов представление о способах разработки программных средств современных информационных технологий, учит разрабатывать эти средства на основе использования современных языков, методов и технологий программирования. Поэтому содержание, формы и методы обучения программированию должны соответствовать современному состоянию языков, методов и технологий программирования и перспективам их развития [1].

При этом в процессе обучения программированию важно выполнять не только учебные задания. Полезным и сточки зрения формирования профессиональных компетенций, и сточки зрения поддержки мотивации учения является выполнений относительно крупных проектов практической направленности. К таким проектам можно отнести и разработку игровых приложений. В процессе этой разработки интегрируются знания и умения студентов как в области программирования, так и в области информационных технологий вообще. Особый интерес представляет разработка приложений для мобильных устройств, в частности, для устройств, работающих под управлением опера-

ционной системы Android.

В настоящее время Android является универсальной операционной системой, на которой работают различные устройства: от смартфонов и планшетов до часов и автомобилей. Android является самой популярной мобильной операционной системой в мире. При этом пользователи Android загружают миллиарды приложений и игр в Google Play каждый месяц, причем большая доля загрузок приходится на игровые приложения [2].

Такая популярность игровых приложений закономерна. На данный момент игровая индустрия становится весьма популярной, ведутся споры о том, что видео игры являются отдельной областью искусства, как кино, литература, живопись, музыка.

В связи с вышесказанным, объектом проводимого исследования является процесс разработки игровых приложений, работающих на смартфонах и планшетах под управлением операционной системы Android. Цель исследования: обосновать технологию разработки игровых приложений для операционной системы Android с использованием библиотеки libGDX.

В процессе исследования были проанализированы особенности разработки приложений для операционной системы Android; обоснован выбор технологии разработки приложений в IDE Android Studio с использованием библиотеки libGDX, разработан прототип игрового приложения для операционной системы Android.

Как показал выполненный анализ, Android – это не просто еще один дистрибутив Linux для мобильных устройств. При разработке приложений для Android разработчику, скорее всего, не придется иметь дело с самим ядром Linux. С точки зрения программиста, Android – это платформа, абстрагирующая разработчика от ядра и позволяющая ему создавать код на Java. Android обладает несколькими полезными возможностями. Во-первых, это фреймворк, предлагающий большой набор API для создания различных типов приложений и, кроме того, обеспечивающий возможности повторного использования и замены компонентов, которые предлагаются платформой и сторонними приложениями. Во-вторых, наличие виртуальной машины Dalvik, которая отвечает за запуск приложений на Android. Кроме того, к услугам разработчика набор графических библиотек для разработки 2D- и 3D-приложений, поддержка мультимедиа-форматов (Ogg Vorbis, MP3, MPEG-4, H.264, PNG), API для доступа к камере, GPS, компасу, акселерометру, сенсорному экрану, джойстику и клавиатуре. Имеется даже специальное API для воспроизведения фоновых звуковых эффектов. Список возможностей Android не исчерпывается выше сказанным. Однако перечисленные будут наиболее важны для разработки игр [2].

Архитектура Android формируется из набора компонентов. Каждый компонент построен на основе элементов более низкого уровня. Ядро Linux предлагает основные драйверы для аппаратных компонентов системы, а также отвечает за память, управление процессами, поддержку сети и т. д. Среда выполнения Android, являющаяся надстройкой над ядром, отвечает за порож-

дение и выполнение приложений Android. Каждая программа работает в собственном процессе со своей виртуальной машиной Dalvik. Помимо библиотек ядра, предлагающих некоторую функциональность Java SE, существует также набор нативных библиотек на C/C++, создающих основу для фреймворка приложения. Эти системные библиотеки, как правило, отвечают за сложные в вычислительном смысле задачи (прорисовка графики, воспроизведение звука, доступ к базе данных), не очень подходящие для виртуальной машины Dalvik. API в них «обернуты» с помощью классов Java во фреймворк приложения. Фреймворк приложения связывает вместе системные библиотеки и среду выполнения, создавая таким образом пользовательскую сторону Android. Фреймворк управляет приложениями и предлагает продуманную среду, в которой они работают. Разработчики создают приложения для этого фреймворка с помощью набора программных интерфейсов на Java, охватывающих такие области, как разработка пользовательского интерфейса, фоновые службы, оповещения, управление ресурсами, доступ к периферии и т. д. Все ключевые приложения, поставляемые вместе с ОС Android (например, почтовый клиент), написаны с помощью этих API.

Для разработки игровых приложений под операционную систему Android можно использовать стандартный инструмент для разработки приложений Android Studio. Также существует несколько коммерческих и открытых игровых движков и фреймворков, такие как Unity, Unreal engine, Torque2D, Cocos 2D-X, libGDX, Xamarin, Corona SDK [4, 5].

Фреймворк отличается от игрового движка тем, что он позволяет контролировать разработчику все аспекты разработки игр. При этом разработчику приходится разбираться, как именно решить ту или иную задачу (например, как организовать игровой мир, как обрабатывать экраны и переходы и т. д.).

Игровой движок, с другой стороны, ориентирован на решение специфических задач и, по сути, диктует разработчику, как тот должен решать те или иные проблемы, предоставляя для этого простые в использовании модули для типовых задач, а также общую архитектуру разрабатываемой игры. Некоторые движки позволяют создавать игры практически не прибегая к написанию кода на языке программирования.

Однако использование игровых движков имеет и некоторые недостатки: например, исходный размер игры, после установки на устройство, будет занимать больше памяти. Разрабатываемая игра может не соответствовать предлагаемым готовым решениям, присутствующим в игровом движке. Разработчикам часто приходится самостоятельно модифицировать движок для того, чтобы достичь своих целей, что может быть невозможно, если его исходный код для разработчика недоступен. Движки могут значительно сократить время разработки, но могут и увеличить его в том случае, если разработчик встретит проблему, которая не была предусмотрена создателями игрового движка.

Также можно отметить и тот факт, что не все игровые движки являются абсолютно бесплатными.

Выбирая между фреймворком и движком, следует основываться на пер-

сональных предпочтениях, бюджете и целях. При работе над данным проектом были использованы IDE Android Studio и фреймворк LibGDX. Далее будут рассмотрены именно эти инструменты.

Android Studio, основанная на программном обеспечении IntelliJ IDEA от компании JetBrains, – это официальное средство разработки Android приложений. Данная среда разработки доступна для Windows, OS X и Linux [5]. libGDX – кроссплатформенный фреймворк для разработки игр и визуализации, основанный на языке программирования Java с некоторыми компонентами, написанными на C и C++ для повышения производительности определенного кода. В настоящее время поддерживает Windows, Linux, Mac OS X, Android, iOS и HTML5 как целевые платформы. LibGDX распространяется в соответствии с лицензией Apache 2.0 (лицензия на свободное программное обеспечение Apache Software Foundation), что дает право использовать libGDX для любых целей, свободно распространять, изменять, и распространять изменённую копию [4].

Игровой движок libGDX позволяет разработчику писать, тестировать и отлаживать код на собственном компьютере и затем переносить его на другие платформы. Для этого фреймворк использует отдельные модули для сборки приложения под каждую платформу, а также независимый модуль, который содержит основной код приложения. libGDX позволяет написать код и затем развертывать игру или приложение на нескольких платформах без модификации. К достоинствам libGDX можно отнести также возможность использования всех инструментов языка программирования Java [4].

Помимо этого, libGDX для своей функциональности использует множество сторонних библиотек и может быть интегрирован со многими сторонними инструментами. С помощью libGDX можно создавать 2D и 3D игры.

Создание игрового приложения – продолжительный и трудоёмкий процесс, состоящий из самых разнообразных этапов, включающий в себя как технические, так и творческие моменты. В ходе исследования был структурирован существующий материал и на его основе обоснована технология разработки игровых приложений.

Первое, что нужно сделать – это определиться с целью. Этапом концепции и определения цели занимается руководитель проекта. На ранних этапах разработки можно представлять в мельчайших деталях готовую игру, так же есть возможность создавать сюжет, стилистику, особенности игры в ходе самой разработки. В этом деле не обязательна излишняя точность, но, как минимум, нужно задать направление развития игрового проекта.

Выбор жанра игры осуществляется в самом начале. Жанр будет основным направлением развития игры. На данный момент в Google play в категории игр, существуют 17 подкатегорий, где основными являются аркады, головоломки, казуальные, обучающие и ролевые игры, стратегии, Экшен-игры. Выбранный жанр можно немного корректировать в процессе разработки, но его сущность должна оставаться прежней. Жанр – это своеобразный фундамент всей игры.

Игровые жанры определяют основные действия, которые будут совершать игроки в процессе игры. Время и место этих действий в игре, определяется основной характеристикой, называемой сеттинг. Сеттинг – это принадлежность игры к определенной сюжетной тематике или к определённому виртуальному миру. В среде компьютерных игр сформировалось несколько наиболее популярных сеттингов: фэнтези, научная фантастика, вторая мировая война, средневековье, стимпанк, постядерный мир, аниме, комиксы. Создание игры в популярном сеттинге обеспечивает её собственную популярность, игроки чувствуют себя уютно и комфортно в уже знакомом мире. Некоторые игры создаются в своих уникальных сеттингах или в необычных сочетаниях стандартных тем. Такие игры менее популярны, но, тем не менее, они имеют свою аудиторию особых игроков.

После задания цели игрового проекта, необходимо выбрать средства (материалы и инструменты) для её достижения. Инструментом и материалом игрового проекта является одна и та же сущность – программный код. Создание игрового материала – это творческая часть процесса. Прежде всего, нужно выбрать язык программирования. После этого предстоит кропотливая работа по написанию программного кода, способного оперировать двухмерными или трехмерными объектами в пространстве, привязкой изображений и звуков. Сейчас существуют готовые программные модули (игровые движки), где уже реализованы базовые функции, способные связать воедино графику, звук, объекты и их движения. Таким образом, выбор языка программирования заменяется другой дилеммой – выбором готового игрового движка. Применение игровых движков сводит к минимуму работу программиста.

Самая важная творческая часть любой игры – игровая механика (геймплей). Игровая механика – это свод правил, по которым будет функционировать игра. Основой всей механики являются игровые объекты. Главный герой игры, компьютерные соперники, второстепенные персонажи, бонусы, подвижные объекты, декорации – всё это игровые объекты со своими свойствами и возможными действиями. Игровая механика определяет, какими клавишами будет управляться главный герой или основной игровой объект, какое действие будет происходить после нажатия той или иной кнопки. Сюда же относится законы поведения игровых объектов (физический движок) и поведение врагов (искусственный интеллект).

В готовых игровых движках чаще всего реализованы и физические движки. Разработчику остается присвоить своим уникальным объектам уже готовые физические характеристики: вес, плотность, эластичность, разрушаемость. Если же создавать собственный физический движок, то для этого понадобится талантливый программист, хорошо понимающий принципы объектно-ориентированного программирования (ООП) и немного разбирающийся в классической физике.

Искусственный интеллект (ИИ) отвечает за поведение компьютерных врагов или союзников. Роль ИИ значительно разнится в зависимости от жанра игры. В экшенах действия врагов крайне примитивны; в стратегиях достаточно

пары десятков скриптов, чтобы придать сопернику кажущуюся разумность; в стелс-экшенах, слешерах и файтингах необходимо создать уникальную систему поведения для каждого типа врагов, иначе игра станет неинтересной. Серьезная стратегическая игра требует колоссальной работы над ИИ, а в простых казуальных играх и в онлайн-проектах, ориентированных на сражения только между реальными игроками, искусственный интеллект отсутствует.

После создания правил игры в виде игровой механики создаются площадки, где эти правила начнут работать. Созданные игровые объекты расставляются в отдельных виртуальных пространствах – уровнях (локациях). Игры чаще всего содержат множество отдельных уровней, переход между которыми происходит по ходу сюжета. Но в последнее время, благодаря возросшей производительности компьютеров, выпускаются игры с одним большим цельным миром, лишь условно разделяемом на различные локации. Построением уровней занимаются левелдизайнеры.

Далее нужно нарисовать текстуры, изображения для игры. Созданием графической составляющей занимаются художники, геймдизайнеры. При разработке простой 2D-игры, разработчик может самостоятельно создать графическую составляющую. Но в более серьезных и крупных проектах лучше нанять профессиональных художников и дизайнеров.

В начале создаются образы героев, врагов, игровых предметов, задних фонов. Для игровых объектов, которые будут передвигаться в ходе игры, создаются анимации. Особенно для героев и врагов, количество анимации которых иногда превышает целую сотню различных движений.

Визуальные спецэффекты – это, по сути своей, те же анимации, только вместо перемещения объектов в них используются перемещения частиц и светофильтров. Лучи света в разные стороны при взятии бонусов, огонь на горящем здании, дымовая завеса после взрыва гранаты, лазерные лучи из дула винтовок, наложение фильтров размытия при нахождении под водой и фильтров затемнения в плохо освещённых местах – всё это спецэффекты. Без подобных эффектов игра будет казаться пресной и слишком обыденной. Использование спецэффектов добавляет игре яркости, сочности и экспрессивности.

Оформить нужно не только игровые уровни, но и систему, объединяющую их в единое целое – игровое меню (строчки, кнопки, страницы настроек). Начальное меню – это визитная карточка игры, и выглядеть она должна идеально. На игровом экране так же есть множество элементов, к которым можно применить оформление – количество жизней, миникарта, меню быстрого выбора действий, инвентарь героя, списки заданий, экраны диалогов.

После графического оформления в игру добавляют звуковые эффекты. Для любого игрового движения добавляется соответствующий звук. Чаще всего в качестве звуковых эффектов используются реальные звуки, записанные в цифровом виде. Также звуковой оставляющей игры является озвучивание игровых диалогов и монологов.

Процесс разработки большой игры построен таким образом, что различными её элементами занимаются различные специалисты. На начальном

этапе игра представляет собой разрозненный набор творческих наработок в различных областях искусства: изображения, звуки, 3D-модели, архитектура, тексты, сценки, видеовставки, оформление. После создания всего необходимого игрового контента, с помощью программных средств, все объекты соединяются в единую сложную систему.

После того как игра готова, остается решить, как доставить игру конечным пользователям. Классический способ (выпуск большого тиража компьютерных дисков, и продажа их через розничные магазины) всё ещё актуален, но подходит лишь для крупных компаний, и для игр, имеющих изначальную популярность. Для небольших групп разработчиков подходит распространение игры через системы цифровой дистрибьюции (крупные онлайн-магазины). Такой вариант обеспечивает для игры уже готовую аудиторию покупателей, которая сформировалась вокруг сервиса. Самым известным примером является сервис Steam, имеющий огромную аудиторию игроков. Поэтому почти каждая игра, вышедшая в этом онлайн магазине, сразу же приобретает мировую известность. Также разработчик может создать собственный интернет-магазин с одним единственным товаром – созданной игрой. Но в таком случае придётся рекламировать не только игру, но еще и интернет-адрес магазина, и завоёвывать аудиторию самостоятельно.

Создание игры и её продажа – это не конец жизненного цикла игрового проекта. Предшествующий бета-тест устранивший из игры самые очевидные ошибки, не означает, что ошибок совсем в игре не осталось. Это могут быть проблемы из-за несовместимости с мало популярными марками оборудования, или ошибки из-за неестественного использования игровых возможностей. Все это способствует тому, что часто приходится вносить исправления ошибок в уже готовую игру. Такие исправления называются патчами, и этот термин очень распространен в игровой индустрии. Мало кому удаётся сразу же выпускать идеальные игры, чаще всего игры доводятся до идеала уже после своего официального релиза.

Представленная технология разработки игровых приложений может предполагать также групповое выполнение проекта студентами, что будет способствовать формированию важной профессиональной компетенцией – умение работать в команде.

На основе представленной выше технологии был разработан прототип игрового обучающего приложения с использованием IDE Android Studio и инструмента libGDX. Игра создана с использованием 2D графики и относится к жанру головоломок и приключений. Главным героем является робот, которым управляет игрок, перемещая его по лабиринту. Цель игры – пройти все лабиринты. Лабиринты генерируются по алгоритму Эллера, в лабиринтах случайным образом встречаются стены, которые можно убрать, тем самым сократив себе путь до выхода или же наоборот сделать путь более длинным. Чтобы убрать стену, нужно решить математическую задачу, при этом примеры и числа генерируются случайным образом.

При создании этого игрового приложения был реализован объектно-

ориентированный подход. Приложение состоит из 3 интерфейсов и 22 классов. Большая часть классов являются «экранами» (меню, игровой процесс и т. д.), шесть классов используется для создания экземпляров игровых объектов (робот, лабиринт, математическая задача), создан также класс для обработки действий на сенсорных экранах.

Реализован класс, который представляет собой модель лабиринта. В нем реализован метод для генерации лабиринта алгоритмом Эллера. Алгоритм Эллера позволяет создавать лабиринты, имеющие только один путь между двумя точками. Сам по себе алгоритм очень быстр и использует память эффективнее, чем другие популярные алгоритмы (такие, как Прима и Крускала), требуя памяти пропорционально количеству строк. Это позволяет создавать лабиринты большого размера при ограниченных размерах памяти, что важно для устройств, ограниченных в ресурсах. После генерации лабиринта вызывается метод, который преобразует лабиринт в удобный вид для использования его при рисовании тайловым методом. Также в данном классе есть методы для генерации стен с математическими задачами, входа выхода из лабиринта.

В настоящее время происходит тестирование разработанного приложения.

В ходе исследования были рассмотрены различные инструменты и сервисы, предоставляемые операционной системой Android, выделены и конкретизированы этапы технологии разработки игровых приложений с использованием инструмента libGDX, который имеет большое количество достоинств и функциональных возможностей, востребованных при создании игровых приложений.

ЛИТЕРАТУРА:

1. Газейкина А. И. Формирование научного мировоззрения будущего ИТ-специалиста в процессе обучения программированию // Педагогическое образование в России, 2015. № 7. С. 36-41.

2. Дейтел П., Дейтел Х., Дейтел Э. Android для разработчиков. 2-е изд. СПб.: Питер, 2015. 384 с.

3. Цехнер М. Программирование игр для Android. СПб.: Изд-во «Питер», 2013. С. 28-31.

4. libGDX – фреймворк для разработки игр // libGDX: libGDX – фреймворк для разработки игр. URL: http://www.libgdx.ru/2013/08/introduction.html (дата обращения: 24.04.2019).

5. Meet Android Studio // Meet Android Studio | Android Studio. URL: https://developer.android.com/studio/intro/index.html#the_user_interface (дата обращения: 20.04.2019).