СОЗДАНИИЕ КРОССПЛАТФОРМЕННЫХ ПРИЛОЖЕНИЙ НА JAVA С ПОМОЩЬЮ ФРЕЙМВОРКА LIBGDX НА ПРИМЕРЕ НАПИСАНИЯ ПРОСТОЙ ИГРЫ

Копасов Д. В., Масалевичюте О. В.
Чердынцев Е. С.
Томский Политехнический Университет
Kopasov.dima@mail.ru

Введение

В настоящее время популярны разного рода приложения (в том числе и игры), как для настольного компьютера, так и для мобильных устройств. Существует много различных сред разработки со встроенными инструментами, облегчающих разработку требуемых продуктов (Visual Studio, Unity, Android Studio, GameMaker). Также для разработки приложений существует свободно распространяемый фреймворк LibGDX, который без проблем встраивается в любую среду разработки, поддерживающую язык программирования Java (Eclipse, Android Studio, IntelliJ IDEA).

LibGDX является одновременно эффективным и лёгким в использовании, позволяет создать множество различных игр быстро и результативно. LibGDX доступный и имеет открытый исходный код. Он используется для создания 2D и 3D игр, при этом легко совмещается с другими библиотеками для поддержки дополнительных возможностей. Приложения, созданные с использованием LibGDX кроссплатформенны и поддерживаются различными системами, включая Windows, Mac OC X, Linux, Android, IOS и HTML5/WebGL [1].

Возможности LibGDX

LibGDX это Java фреймворк для разработки игр, обеспечивающий единый API, работающий на всех поддерживаемых платформах. Он также предлагает следующие возможности [2]:

- Кроссплатформенная разработка для Android, IOS, Windows, Linux и HTML5;
- Визуализация посредством OpenGL ES 1.0, 1.1 и 2.0 на всех платформах;
- Высокоуровневое 2D и 3D;
- Воспроизведение музыки и звуковых эффектов из WAV, MP3 и OGG файлов;
- Доступ к прикосновениям сенсорного экрана, мыши и клавиатуры основанных на обработке событий и очереди;
- Классы линейной алгебры в 2D и 3D.

Кроссплатформенность

Java, как известно, является кроссплатформенным языком программирования, то есть программы, написанные на Java, могут работать на разных платформах. Однако на разных платформах запуск приложения осуществляется по-разному, и поэтому в LibGDX используется разбиение всего проекта создаваемого приложения на подпроекты (рис. 1):

- основная часть, которая содержит в себе весь исполняемый код;
- дополнительные проекты, содержащие в себе классы соответствующей платформы, отвечающие за инициализацию основного проекта. Так для Desktop платформы — класс, содержащий метод main. Для Android проекта это класс, наследуемый от AndroidApplication и переопределяющий метод onCreate этого класса. Для HTML проекта это класс, наследуемый от GwtApplication. Для IOS платформы это класс, наследуемый от IOSApplication.

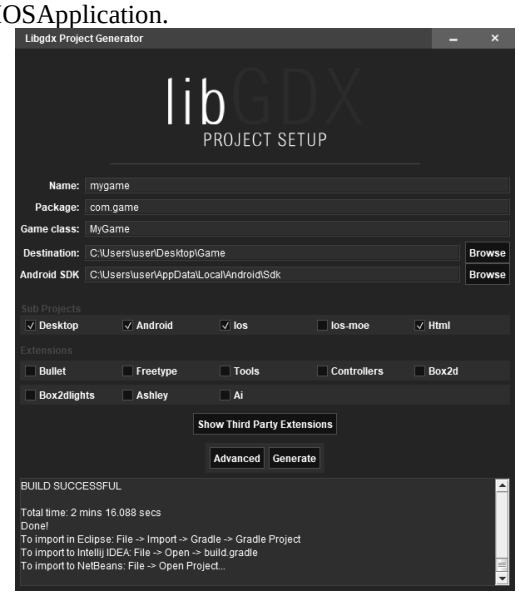

Рис. 1. Меню создания проекта с выбором разных целевых платформ.

Сложности в разработке

Несмотря на то, что LibGDX является кроссплатформенным и должен одинаково обеспечивать работу приложения на разных платформах, существуют некоторые сложности, которые стоит учитывать при проектировании программного продукта. Самые заметные из них это различия в файловой системе и системе ввода разных платформ.

В случае обработки ввода для Desktop и HTML проектов можно использовать клавиатуру и мышь. При использовании мыши можно отслеживать положение курсора и реагировать на появление его в какой-либо области (не актуально для мобильных платформ), а также клики мыши, воспринимаемые, как прикосновение к экрану (по аналогии с сенсорными телефонами).

Для мобильной платформы (Android и IOS) почти весь ввод происходит через прикосновения к

экрану, а также вполне возможно обработать нажатие физических кнопок смартфонов.

Также стоит не забывать про сложности, связанные с различиями в файловых системах платформ. Самое заметное отличие это чтение и запись файлов на Desktop и Android платформах. Проблема заключается в том, что в Android, в отличие от Desktop, внутренние файлы приложения защищены от записи, доступно только чтение.

Поэтому нужно реализовывать проект с алгоритмами либо универсальными для использования на всех системах, либо отдельно используемыми на разных платформах.

Основные используемые инструменты

Для создания относительного простого, но в то же время функционального приложения достаточно использовать четыре основных инструмента фреймворка LibGDX: SpriteBatch, Texture, TextureRegion, OrthographicCamera.

1. SpriteBatch – упаковщик спрайтов, который позволяет рисовать различные графические элементы в заданных координатах в двумерной системе координат.
2. Texture – графическая единица, взятая из картинки в формате .png, .jpg и другие. Используется для рисования на SpriteBatch.
3. TextureRegion – определённый фрагмент текстуры, имеющий те же самые свойства, что и текстура. И так же рисуется на SpriteBatch.
4. OrthographicCamera – камера прямоугольной проекции, предназначенная для визуального отображения фрагмента области рисования.

Реализация

Для демонстрации основных возможностей LibGDX была создана небольшая игра под целевые платформы Desktop и Android (рис. 2). Как описывалось ранее, для каждой платформы создан класс, инициализирующий основной класс игры MyGame.

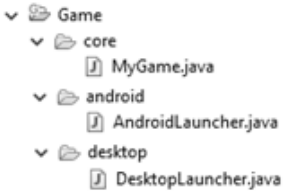

Рис. 2. Краткая структура проекта с классами лаунчерами под разные платформы.

MyGame в свою очередь реализует интерфейс ApplicationListener, обеспечивающий работу приложения по определённому жизненному циклу, регулирующему основные состояния приложения, такие как создание, приостановка, возобновление, визуализация и утилизация. Этотинтерфейс реализует методы create, render, resize, pause и dispose. Каждый из этих методов [3, 4]:

- Метод create вызывается один раз при запуске приложения;
- Метод render вызывается циклом приложения каждый раз, когда должна быть выполнена визуализация;
- Метод resize вызывается при каждом изменении размера экрана в игре;
- Метод pause вызывается, когда приложение переходит в фоновый режим работы;
- Метод dispose вызывается при закрытии приложения.

Программно игра состоит из основного класса MyGame, обеспечивающего переключение и работу окон состояния игры. Каждое окно состояния игры имеет три метода, соответствующие распространённой модели построения проекта MVC. Это методы:

- Обновить – соответствует внутренней работе с данными игры и их обновлению с течением времени (Model);
- Отрисовка – соответствует визуальному представлению всех графических элементов приложения (View);
- Ввод – соответствует реакции на внешнее воздействие на игру (Control).

Игра (рис. 3) представляет собой небольшую пошаговую стратегию, которую я в дальнейшем планирую развивать, выполненную в 2D графике.

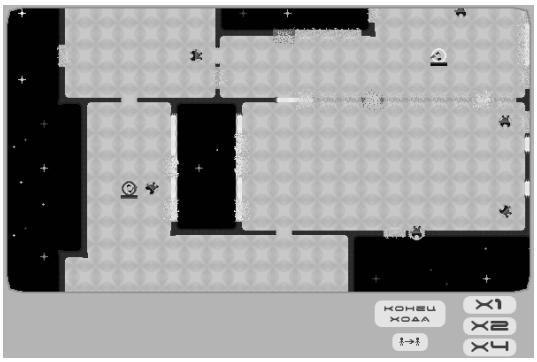

Рис. 3. Кадр из написанной игры, демонстрирующий сгенерированное поле и прошедшие на нём боевые действия.

Заключение

Рассмотренный фреймворк LibGDX прост в использовании и позволяет с лёгкостью создать приложения на распространённые компьютерные платформы, что является весьма полезным и востребованным в настоящее время.

Список использованных источников:

1. Beginning Java Game Development with LibGDX/ Lee Stemkoski. – 2015. – 325 с.

2. libGDX [Электронный ресурс]. – URL: https://libgdx.badlogicgames.com/features.html

3. libGDX: Жизненный цикл [Электронный ресурс]. – URL: http://www.libgdx.ru/2013/09/application-life-cycle.html

4. Создание игр с использованием libGDX [Электронный ресурс]. URL – https://www.youtube.com/watch?v=yUgitdAcTwo