



Урок 1

Установка и настройка инструментов разработки

Установка и настройка инструментов разработки. Создание шаблонного проекта и его запуск. Структура проекта. Обзор модулей LibGDX

[Инструменты разработчика](#)

[Начало работы](#)

[Архитектура фреймворка](#)

[Базовые возможности по работе с графикой](#)

[SpriteBatch](#)

[Texture](#)

[TextureRegion](#)

[Закраска](#)

[Домашнее задание](#)

[Дополнительные материалы](#)

[Используемая литература](#)

Инструменты разработчика

LibGDX – это фреймворк для создания кроссплатформенных игр и приложений, написанный на Java, с использованием C и C++.

Для работы необходимо скачать и установить следующие инструменты:

- JDK - <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>;
- Android Studio - <https://developer.android.com/studio/index.html>;
- LibGDX Framework - <https://libgdx.badlogicgames.com/download.html>.

Возможные проблемы:

- 1) Во избежание проблем с OpenAL, необходимо убедиться, что имя учетной записи написано только с использованием латинских букв. Так как драйвер OpenAL устанавливается в папку «Users».
- 2) На 32-битных системах возможна ошибка импорта проекта, связанная с нехваткой памяти, поэтому перед импортом нужно подкорректировать файл «gradle.properties», заменив значение «-Xmx1500m» на «-Xmx512m».
- 3) При возникновении исключения «OpenGL is not supported by the video driver» возможно, что ваша видеокарта не поддерживает OpenGL, либо необходимо обновить драйверы.

Начало работы

Первым шагом сформируем шаблонный проект с помощью утилиты «gdx-setup.jar».

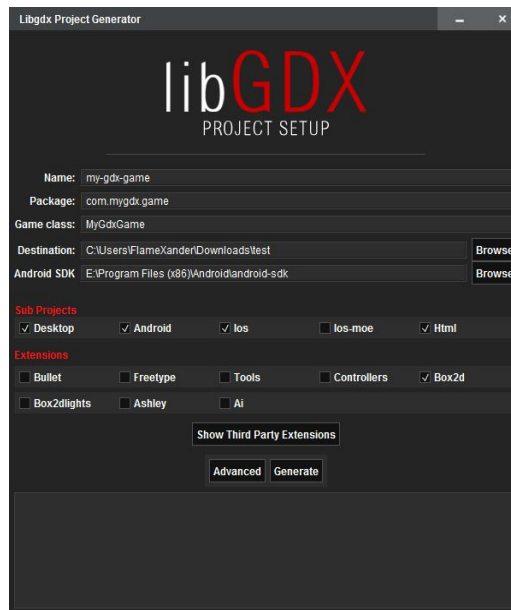


Рисунок 1. Утилита «gdx-setup».

1. Указываем имя проекта, пакеты, путь к проекту и Android SDK:

- Name: StarGame.
- Package: ru.geekbrains.stargame.
- Game class: StarGame.
- Destination: Путь к проекту.
- Android SDK: Расположение Android SDK.
- Sub Projects: Desktop, Android.
- Extensions: все пункты выключены.

Утилита создаст 3 проекта (Core, Desktop и Android) в папке, указанной в пункте «Destination». Core - проект, где будет содержаться код с основной логикой нашей игры. Desktop и Android - проекты, реализующие Core-проект на конкретных платформах.

2. Нажимаем кнопку «Generate», после чего установщик скачает все необходимые файлы и сконфигурирует проекты.
3. После генерации импортируем проект в Android Studio.
4. Заходим в Edit Configurations и настраиваем в соответствии с рисунком 2.

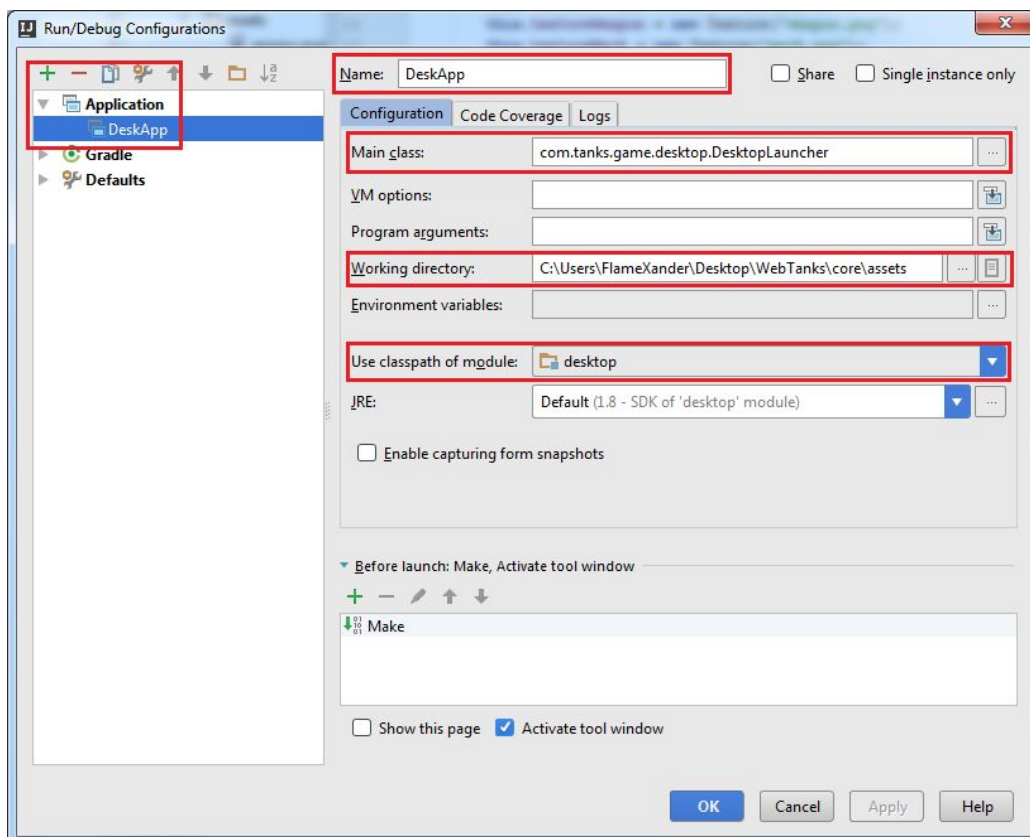


Рисунок 2. Edit Configurations.

5. Если все сделано верно, при запуске появится окно:



Рисунок 3. Базовый шаблонный проект.

Архитектура фреймворка

```
public class MyGdxGame extends ApplicationAdapter {
    SpriteBatch batch;
    Texture img;

    @Override
    public void create () {
        batch = new SpriteBatch();
        img = new Texture("badlogic.jpg");
    }

    @Override
    public void render () {
        Gdx.gl.glClearColor(1, 0, 0, 1);
        Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);
        batch.begin();
        batch.draw(img, 0, 0);
        batch.end();
    }

    @Override
    public void dispose () {
        batch.dispose();
        img.dispose();
    }
}
```

Начальный проект делится на несколько методов, каждый из которых выполняет определенную функцию.

- «create()» занимается инициализацией проекта, срабатывает при запуске проекта.
- «render()» выполняется 60 раз в секунду и отвечает за отрисовку игры, это ограничение при необходимости можно изменить (до 30 к/с, или 90 к/с), если мощности устройства не будет хватать для обсчета логики или отрисовки, частота кадров в секунду может падать.
- «dispose()» отвечает за освобождение ресурсов после того, как они перестали быть нужны.

Именно с такой заготовки мы и начнем работу. Перед описанием работы с отдельными блоками фреймворка LibGDX, рассмотрим его структуру.

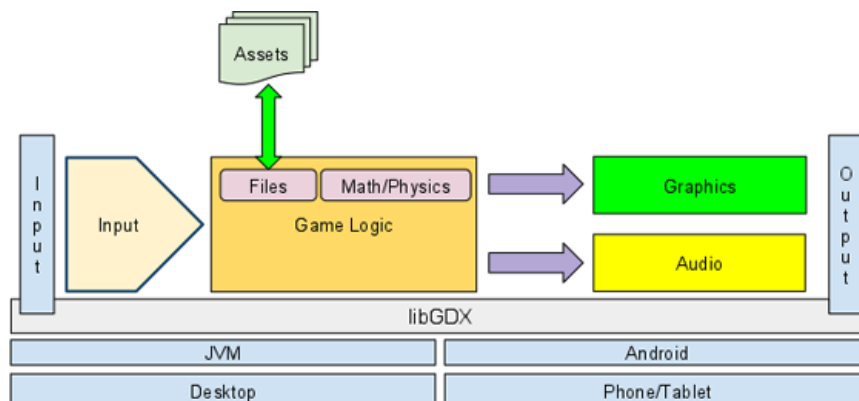


Рисунок 4. Структура LibGDX.

LibGDX включает в себя несколько модулей:

- «Input» - предоставляет унифицированную модель ввода для всех платформ. Поддерживает клавиатуру, сенсорный экран, акселерометр и мышь. Модель позволяет работать с устройствами ввода на любой платформе. В desktop-приложениях сенсорный экран заменяет мышь.
- «Graphics» - позволяет выводить изображение на экран, используя OpenGL ES, реализованный на аппаратном уровне.
- «Files» - предоставляет абстрагированный доступ к файлам на различных платформах, реализуя типичные, для работы с файлами, функции ввода/вывода информации.
- «Audio» - упрощает создание и использование аудио файлов, предоставляет прямой доступ к устройству аудио-вывода и позволяет работать со звуковыми файлами двух типов: музыка и звук. Оба типа поддерживает форматы: WAV, MP3, OGG, mid.
- «Math» - предоставляет математические функции, которые могут понадобиться в игре.

Базовые возможности по работе с графикой

Изображение, полученное из его оригинального формата (например, PNG) и загруженное в GPU, называется текстурой. Текстуры рисуются по некоторой спецификации, которая представляет собой описание геометрической фигуры, и каким образом текстура накладывается на ее вершины.

SpriteBatch

Очень часто рисование текстур происходит в прямоугольной геометрии. Также очень часто одна текстура или ее разные части рисуются много раз. Неэффективно отсылать один прямоугольник для отрисовки в GPU. Вместо этого много прямоугольников для одной текстуры могут быть описаны и отправлены в GPU все вместе. Этим и занимается класс «SpriteBatch».

«SpriteBatch» получает текстуру и координаты каждого прямоугольника, куда будет выведена эта текстура. Он накапливает эту информацию без отсылки в GPU. Когда он получает текстуру, которая отличается от последней загруженной, он делает активной последнюю загруженную, отправляет накопленную информацию по рисованию в GPU и начинает накапливать данные по отрисовке для следующей текстуры.

Изменение текстуры каждые несколько прямоугольников, которые должны отрисовываться, мешает «SpriteBatch» группировать достаточно много прямоугольников. Также привязка текстуры - довольно дорогая операция. Учитывая эти причины, часто хранят много мелких изображений в одном большом и потом рисуют его части, максимизируя этим количество накопленных прямоугольников для отрисовки и избегая смены текстуры.

Использование класса «SpriteBatch» в приложении выглядит следующим образом:

```
public class MyGdxGame extends ApplicationAdapter {
    SpriteBatch batch;
    ...
    @Override
    public void render () {
        Gdx.gl.glClearColor(1, 1, 1, 1);
        Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);
        batch.begin();
        batch.draw(img, 0, 0);
        batch.end();
    }
}
```



```
...  
}
```

Все вызовы «SpriteBatch» для отрисовки должны быть заключены между методами «begin()» и «end()». В данном примере производится рисование текстуры в точке с координатами 0, 0.

Texture

Класс «Texture» получает изображение из файла, размещенного в папке «assets», и загружает его в GPU. Размеры изображения должны быть степенью двойки (16x16, 64x128, 128x128 и т. д.). При отрисовке текстуры указываются координаты ее нижнего левого угла.

Пример:

```
public class MyGdxGame extends ApplicationAdapter {  
    SpriteBatch batch;  
    Texture img;  
    @Override  
    public void create () {  
        batch = new SpriteBatch();  
        img = new Texture("square.png");  
    }  
    @Override  
    public void render () {  
        Gdx.gl.glClearColor(1, 1, 1, 1);  
        Gdx.gl.glClear(GL20.GL_COLOR_BUFFER_BIT);  
        batch.begin();  
        batch.draw(square, 50, 50);  
        batch.end();  
    }  
    ...  
}
```

Здесь создается текстура и отправляется классу «SpriteBatch» для отрисовки. Текстура рисуется в прямоугольнике, левый нижний угол которого размещенный в точке (50, 50), с шириной и высотой, равной размерам текстуры. Метод «draw()» класса «SpriteBatch» имеет несколько перегрузок:

```
//Рисует часть текстуры с возможностью масштабирования, вращения вокруг  
точки и отражения  
draw(Texture texture, float x, float y, float originX, float originY,  
float width, float height, float scaleX, float scaleY, float rotation,
```

```

int srcX, int srcY, int srcWidth, int srcHeight, boolean flipX, boolean
flipY)

//Рисует текстуру в точке x, y, используя размеры исходной текстуры
draw(Texture texture, float x, float y)

//Рисует часть текстуры
draw(Texture texture, float x, float y, int srcX, int srcY, int
srcWidth, int srcHeight)

//Рисует часть текстуры, растянутую до размеров width и height, с
возможностью отражения
draw(Texture texture, float x, float y, float width, float height, int
srcX, int srcY, int srcWidth, int srcHeight, boolean flipX, boolean
flipY)

//Рисует часть текстуры, растянутую до размеров width и height.
Координаты указываются не в пикселях, а в действительных числах от 0.0f
до 1.0f
draw(Texture texture, float x, float y, float width, float height,
float u, float v, float u2, float v2)

```

TextureRegion

Класс «TextureRegion» описывает прямоугольник внутри текстуры и используется для рисования только части текстуры.

Пример:

```

private Texture texture;
private TextureRegion region;
...
texture = new Texture("image.png");
region = new TextureRegion(texture, 20, 20, 50, 50);
...
batch.begin();
batch.draw(texture, 50, 10);
batch.draw(region, 50, 10);
batch.end();

```

Здесь 20, 20, 50, 50 описывает часть текстуры, которая потом будет нарисована в точке (10, 10). Это же действие может быть выполнено передачей текстуры и дополнительных

параметров классу «SpriteBatch», но класс «TextureRegion» делает это удобнее, поскольку проще определить отдельный объект и работать с ним, чем помнить про множество дополнительных параметров. «SpriteBatch» имеет несколько перегрузок для рисования «TextureRegion», которые аналогичны перегрузкам при рисовании обычных текстур.

Закраска

Когда текстура нарисована, она может быть закрашена определенным цветом:

```
private Texture texture;
private TextureRegion region;
...
texture = new Texture("image.png");
region = new TextureRegion(texture, 20, 20, 50, 50);
...
batch.begin();
batch.setColor(1, 0, 0, 1);
batch.draw(texture, 10, 10);
batch.setColor(0, 1, 0, 1);
batch.draw(region, 50, 10);
batch.end();
```

Этот код показывает, как нарисовать текстуру и ее регион с окраской цветом. Цвет описывается в RGBA-модели, где каждая компонента лежит в пределах от 0.0f до 1.0f. Альфа-канал игнорируется, если смешивание отключено.

Домашнее задание

1. Все домашние задания курса выкладываются на страницах соответствующих занятий, это связано с тем, что формулировка задания зависит от прогресса написания проекта на каждом конкретном уроке.

Дополнительные материалы

1. <http://www.gamefromscratch.com/page/LibGDX-Tutorial-series.aspx>
2. <https://github.com/libgdx/libgdx/wiki>
3. <https://github.com/libgdx/libgdx/wiki/External-tutorials>

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. <https://habrahabr.ru/post/143405/>
2. <https://github.com/libgdx/libgdx/wiki/The-application-framework>

