УДК 621.9.048.7

Об одном способе работы с библиотеками при создании web-приложений для мобильных устройств

Дегалевич Е.В.

Белорусский Национальный Технический Университет, Минск, Беларусь,

E-mail: degalevich_evgeniy@mail.ru

Все, кто сталкивался с программированием на языке Java, прекрасно знают проблемы, связанные с созданием графического интерфейса. Аналогичные трудности придется преодолеть Вам, если Вы решили создать приложение для мобильной платформы. Представляется логичным, что интерфейс для мобильного устройства должен быть максимально информативным, простым и понятным, содержать элементы управления достаточно удобных размеров, при этом количество управляемых элементов не должно быть большим. Естественно полагать, что в процессе работы возникает необходимость использования специальной библиотеки. К большому сожалению, для многих из них разработчики программных продуктов «забывают» создавать инструктивные материалы для пользователей, что значительно затрудняет работу с ней в особенности малоопытным программистам. Столкнувшись с задачей создания Web-приложения на платформе Android и описанной проблемы, рассмотрим работу со вспомогательной библиотекой AndEngine, для которой нет документации, способствующей программисту понять принципы создания приложения. Справедливости ради отметим, что есть сборник примеров. Его можно найти по ссылке:

http://www.proandroid.net/download-854-andengine-examples.html

Данное приложение имеет открытый код и огромное количество примеров, но, в связи с отсутствием документации, приходится искать нужный пример, а потом разбираться в его коде. Основываясь на личном опыте, предлагаю инструктивный порядок установки библиотеки AndEngine в программной среде Eclipce с рассмотрением ее структуры в интересах использования гаджетов в ходе разработки приложения для мобильных устройств.

Установка AndEngine. На первом шаге следует установить плагин Mercurial для Eclipce, предварительно скачав его, перейдя по ссылке http://www.javaforge.com/project/HGE.

Далее выполняется следующая последовательность команд:

1. File — New Project — Clone Existing Mercurial Repository
2. В поле URL вставляем ссылку:

https://andengineexamples.googlecode.com/hg/.

3. Нажатием Next переходим к загрузке AndEngine.

4. Next и Finish, установка окончена.

5. Перезапускаем Eclipce.

Для дальнейшей работы с библиотекой необходимо подключить ее к проекту, выполнив следующие шаги:

1. Создать новый проект, например Project1.

2. Щелчок правой кнопкой мыши на папке проекта и создаем новую папку: New — Folder, папку назовем lib.

3. Открыть папку проекта AndEngineExamples и скопировать из папки lib файл andengine.jar в папку lib только что созданного проекта Project1.

4. Открываем папку lib проекта Project1.

5. Правая кнопка мыши на andengine.jar и выбираем Build path — Select build path.

Основные функции и структуры AndEngine

AndEngine использует для вывода графики библиотеку OpenGL, в составе которой находятся так называемые Текстуры или Атласы изображений. При выводе графики OpenGL использует Атласы изображений для рисования. OpenGL берёт из Атласа заданную прямоугольную область и копирует её на экран. Поэтому спрайты и прочие картинки необходимо предварительно загрузить в эти Атласы. В AndEngine они называются BitmapTextureAtlas. BitmapTextureAtlas должен иметь размеры, равные степеням 2 (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048). Например, 512×256 или 1024×1024. (Размер 2048 доступен только для OpenGL версии 2.0 и будет работать не на всех устройствах).

В играх графика – растровая. Создавать спрайты можно в чём угодно – от Max 3D и Flash до Фотошопа и Paint. AndEngine позволяет загружать графику в формате PNG и JPG, а также SVG (векторный формат).

Класс Engine – самый главный класс, который управляет процессом вывода графики, проигрывания звуков и музыки, обрабатывает касания экрана, воспроизводит вибрацию, реагирует на акселерометр (рис.1). И в своём основном цикле (update) заставляет работать все объекты в игре — от спрайтов до модификаторов.

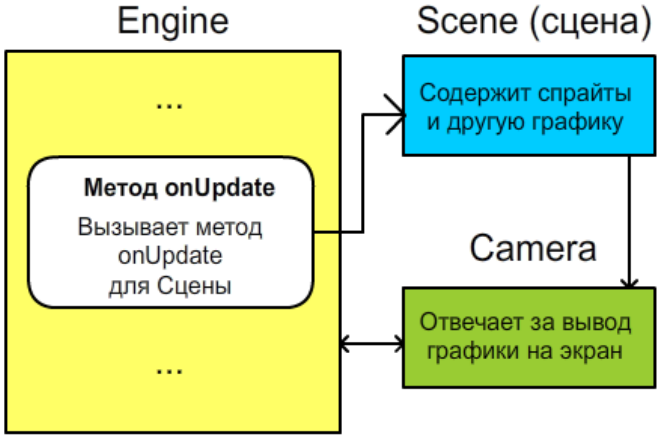

Рис. 1- Схема взаимодействий основных классов.

Класс Camera – не менее важный класс. Именно Camera отвечает за то, что вы увидите на экране. Всё, что за пределами камеры, на экране не видно. Размер камеры обычно устанавливается равным размеру экрана устройства.

Сцена (Scene) похожа на MovieClip во Flash. Она является контейнером для остальных объектов — других сцен, спрайтов, графических примитивов. При создании игры обязательно нужно создать хотя бы одну сцену (в методе onLoadScene () нашегоActivity).

Графику можно загружать как напрямую из файлов (из папки assets), так и из ресурсов (папка res).

Рассмотрим небольшой пример.

После создание проекта нам необходимо расширить основной класс приложения. Делается это с помощью базового класса BaseGameActivity.

```
import org.anddev.andengine.engine.Engine;
import org.anddev.andengine.entity.scene.Scene;
import org.anddev.andengine.ui.activity.BaseGameActivity;

public class T_RIP_SActivity extends BaseGameActivity {
@Override
public Engine onLoadEngine() {
// TODO Auto-generated method stub
return null;
}
@Override
public void onLoadResources() {
// TODO Auto-generated method stub
}
@Override
public Scene onLoadScene() {
// TODO Auto-generated method stub
return null;
}
@Override
public void onLoadComplete() {
// TODO Auto-generated method stub
}
	}
```

Как видно, в основном классе T_RIP_SActivity создаются различные перегружаемые методы, необходимость которых продиктована следующим.

Прежде всего, нам необходимо создать Камеру и соединить её с нашим «движком» - Engine. Для этого используем перегруженный метод `onLoadEngine()`:

```
public Engine onLoadEngine() {
   Camera mCamera = new Camera(0, 0, Camera_width, Camera_height);
     return          new          Engine(new          EngineOptions(true,
ScreenOrientation.LANDSCAPE,new     RatioResolutionPolicy(Camera_width,
Camera_height), mCamera).setNeedsMusic(true));
          }
```

Далее нам необходимо загрузить ресурсы в Атлас. Для этого используется метод `onLoadResources()`:

```
@Override
     public void onLoadResources() {
     BitmapTextureAtlasTextureRegionFactory.setAssetBasePath("car/");
     BitmapTextureAtlasmBitmapTextureAtlas            =            new
BitmapTextureAtlas(2048, 2048, TextureOptions.DEFAULT);
          TextureRegionweelreg                                          =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.mBitmapTex
tureAtlas,
               this, "w.png", 600, 0);
          TextureRegioncarreg                                           =
BitmapTextureAtlasTextureRegionFactory.createFromAsset(this.mBitmapTex
tureAtlas,
               this, "main.png", 1024, 1024);
          }
```

И в конце мы создаём сцену, на которую размещаем спрайты из атласа. Для этого используем метод `onLoadScene()`:

```
@Override
     public Scene onLoadScene() {
          this.mEngine.registerUpdateHandler(new FPSLogger());
          Scene mMainScene = new Scene();
          Sprite weelspr= new Sprite(313, 113, weelreg);
          Sprite weelspr1= new Sprite(663, 113, weelreg);
          Sprite carspr = new Sprite (250,0,carreg);
          mMainScene.attachChild(carspr);
          mMainScene.attachChild(weelspr);
          mMainScene.attachChild(weelspr1);
          returnmMainScene;
          }
```

И последний метод - `onLoadComplete()`. Данный метод вызывается после загрузки всех необходимых ресурсов. Именно с него и можно начинать написание функциональной части Вашего приложения.

В целом, это и есть основная идеология создания приложения с использованием AndEngine.

ВЫВОДЫ.

1. В докладе показана возможность широкого использования Web-библиотек, не имеющих достаточно значимой документации.
2. В помощь малоопытным программистам на примере AndEngine изложен инструктивный порядок ее установки.
3. Рассмотрены основные структурные компоненты, рекомендуемые к использованию при создании мобильных приложений.