ISEL / ADEETC

Master in Communication Networks and Multimedia Engineering

Interactive Multimedia Applications

Tutorial 5

Interactive Multimedia Applications

libGDX - 2nd

(Background Scrolling Parallax, Animações e Áudio)

Rui Jesus

Introduction

This work aims to introduce a set of techniques commonly used in the development of games for Android using the game engine libGDX. The application developed in this tutorial includes animations with Sprites, using multiple background images in order to produce the effect of parallax and the use of audio. Below is a set of links that you should consult during the development of this work.

Android Developers
http://developer.android.com/training/index.html

libGDX
https://libgdx.badlogicgames.com/index.html

libGDX Wiki
https://github.com/libgdx/libgdx/wiki

Box2D Manual
https://github.com/libgdx/libgdx/wiki/Box2d

Laboratory Work

Code Structure of a Game

1. Create a new libGDX project. Copy the following files to the "assets" directory of the project:

   "*cowboy_run_sprite.png*"

   "*parallax_background_layer_back.png*"

   "*parallax_background_layer_front.png*"

   "*parallax_background_layer_mid.png*"

AssetLoader

2. Repeat the class structure of tutorial 4. That is, create an "AssetLoader" class to load the data into memory. Create two methods (as in the previous tutorial): "`load`", "`dispose`".

For now they can be empty. Then create a "GameScreen" class that implements the "Screen" interface. This class makes the rendering on the screen and therefore typically contains the "world" and "camera" objects. For now, create the methods necessary to implement "Screen" and the "camera" object (as in the previous tutorial). We also need the "SpriteBatch" attribute and associate it with the "camera" object. Typically in a game we will have several screens. Right now we just have this one. Other screens (implementations of the "Screen" interface) must be created in parallel with this class. Finally, we need to create the launcher class, that is, the class that is called when the program starts. In this tutorial we will call it "Parallax". This class extends the "Game" class and will load the assets and manage the screens. This class is exactly the same as the previous tutorial. To complete the basic structure of a game, it is necessary to create a class for each actor in the game. There is an "Actor" class which should be the basis of the classes of each actor in the game. In this tutorial the actors are simple so we will not create this class.

3. The "AssetLoader" class should load the 4 images from point 1, construct "TextureRegion" objects with these textures and create their animations. Follow the example below for an image. Do not forget the "`dispose`" method that is identical to the previous tutorial.

```
//AssetLoader
public class AssetLoader {
	public static Animation cowboyAnimation;
	public static TextureRegion [] cowboy = new TextureRegion[12];
	public static Texture backTexture,frontTexture,
midTexture,cowboyTexture;

public static void load() {
	// Images load to textures
	…
	// Creating TextureRegion for the Animation
	int step = 175; int x = 0; int y = 0;
	for(int i=0; i<cowboy.length; ++i){
		TextureRegion txtreg = new TextureRegion(cowboyTexture,
x, y, step, step);
	txtreg.flip(true, false);
		cowboy[i] = txtreg;
		if(x>=2*step){
			x = 0;
			y += step;
		}else{
			x +=step;
		}
	}
	// Animating the cowboy
	cowboyAnimation = new Animation(0.05f, cowboy);
	cowboyAnimation.setPlayMode(Animation.PlayMode.NORMAL);
}
…
```

4. In the "GameScreen" class, put on the screen the textures and animation (as in tutorial 4). Run the program.

Background Scrolling Parallax

5. Let's introduce and apply the "parallax background" to make a nice perspective effect that will give the illusion of depth. This feat is obtained by using parallax values that will determine the speed of movement of Sprites/Textures based on the movement of the camera. Three .png image files larger than the application layout (preferably) are required. An image to represent the closest scenario (called the front). Another is to designate the more further away part of the scenery (called back) and finally, an image (called mid) to place objects between the two previous ones.

Classe "Scrollable"

6. All three images will be moving from right to left. So let's start by constructing the "Scrollable" class that lets you manage this movement. Create the class and copy the code below. The class is composed of attributes and methods required for this example. In general, this class should be more generic to allow, for example, vertical movements. The sketch is for this example and the challenge to students who need this movement in their games, to build a more generic class or even a class hierarchy.

```
public class Scrollable {
    protected Vector2 position;
    protected Vector2 velocity;
    protected int width;
    protected int height;
    protected boolean isScrolledLeft;

    public Scrollable(float x, float y, int width, int height, float
scrollSpeed) {
        position = new Vector2(x, y);
        velocity = new Vector2(scrollSpeed, 0);
        this.width = width;
        this.height = height;
        isScrolledLeft = false;
    }

    public void update(float delta) {
        position.add(velocity.cpy().scl(delta));

        // If the Scrollable object is no longer visible:
        if (position.x + width < 0) {
            isScrolledLeft = true;
        }
    }
   public void reset(float newX) {
        position.x = newX;
        isScrolledLeft = false;
    }


    public void stop() {
        velocity.x = 0;
    }

    // Getters for instance variables
    public boolean isScrolledLeft() {
        return isScrolledLeft;
    }

    public float getTailX() {
        return position.x + width;
    }

    public float getX() {
        return position.x;
    }

    public float getY() {
        return position.y;
    }

    public int getWidth() {
        return width;
    }

    public int getHeight() {
        return height;
    }

}
```

7. In the "GameScreen" class we will create instances of the "Scrollable" class and move them on the screen. We start with the texture *backTexture*. Copy the code below and run the program.

```
// GameScreen constructor
// To each texture we create two Scrollable instances one follow the
// other to give the notion of continuity
background = new Scrollable(0, 0, AssetLoader.backTexture.getWidth(),
AssetLoader.backTexture.getHeight(), -5.0f);

background2 = new Scrollable(background.getTailX(), 0,
AssetLoader.backTexture.getWidth(),
AssetLoader.backTexture.getHeight(), -5.0f);

// render method
background.update(delta);
background2.update(delta);
if (background.isScrolledLeft()) {
      background.reset(background2.getTailX());

} else if (background2.isScrolledLeft()) {
      background2.reset(background.getTailX());
}
…
batch.begin();

batch.draw(AssetLoader.backTexture, background.getX(),
background.getY());
batch.draw(AssetLoader.backTexture, background2.getX(),
background.getY());

batch.end();
```

Background Parallax

8. To make the background parallax effect simply repeat the previous code for the remaining background textures with the proper order and speeds of 5, 20, 75 respectively. Do not forget the animation that should stay between the textures *midTexture* and *frontTexture*. Run the program.

Audio (New Project)

9. Create a new libGDX project. Copy the following files to the "assets" directory of the project:

   images - "*back.png*", "*notes.png*", "*tank.png*";

   text fonts - "*text.fnt*" / "*text.png*", "*shadow.fnt*" / "*shadow.png*";

   audio - "*explosion.ogg*", "*wagner_the_ride_of_the_valkyries.ogg*".

10. Repeat step 2. Set the launcher class with the name "SoundTutorial". This class remains the same as the "Parallax" class. The "AssetLoader" class retains the same structure. Loading the images is the same. For the remaining assets, copy the code below. In the "GameScreen" class, show the textures on the screen. Run the program. Do not forget to do the "`dispose`" in class "AssetLoader".

```
// sounds, LibGDX supports 3 audio formats: ogg, mp3 and wav
explosion = Gdx.audio.newSound(Gdx.files.internal("explosion.ogg"));
music =
Gdx.audio.newSound(Gdx.files.internal("wagner_the_ride_of_the_valkyri
es.ogg"));

// fonts
font = new BitmapFont(Gdx.files.internal("text.fnt"));
shadow = new BitmapFont(Gdx.files.internal("shadow.fnt"));
```

11. Let's create two buttons to receive input from the user using the two images "tank.png" and "notes.png". To do this, we'll create two rectangles around the images, then we'll set the "touchUp" event to receive the user's touch / click and start playing the sounds. Copy the code below to the "GameScreen" class.

```
// GameScreen constructor
public GameScreen() {
      camera = new OrthographicCamera();
      camera.viewportHeight = 480;
      camera.viewportWidth = 640;
      camera.position.set(camera.viewportWidth * .5f,
camera.viewportHeight * .5f, 0f);
      camera.update();

      batch = new SpriteBatch();
      batch.setProjectionMatrix(camera.combined);

// To render shapes like a SpriteBatch
      shapeRenderer = new ShapeRenderer();
      shapeRenderer.setProjectionMatrix(camera.combined);

// To enable the touch user input (do not forget to implement
// InputProcessor
      Gdx.input.setInputProcessor(this);

// 2 retangles to used around the textures in order to built a button
      tankButton = new Rectangle(camera.viewportWidth / 4
        - AssetLoader.tank.getWidth(), camera.viewportHeight / 2-
AssetLoader.tank.getHeight()/2, AssetLoader.tank.getWidth(),
AssetLoader.tank.getHeight());
      notesButton = new Rectangle(3 * camera.viewportWidth / 4
        - AssetLoader.notes.getWidth() / 2, camera.viewportHeight /
2, AssetLoader.notes.getWidth(), AssetLoader.notes.getHeight());
}
```

12. In the "`render`" method of the "GameScreen" class we will render the rectangles and display the text with the font and the shadow on the screen.

```
 …

batcher.begin();
batcher.draw(AssetLoader.background, 0, 0);
batcher.draw(AssetLoader.tank, tankButton.x, tankButton.y);
batcher.draw(AssetLoader.notes, notesButton.x, notesButton.y);


// Draw shadow first
AssetLoader.shadow.draw(batch, "Touch for Sound!", 16,
AssetLoader.font.getCapHeight()+1);
// Draw text
AssetLoader.font.draw(batch, "Touch for Sound!",
15 ,AssetLoader.font.getCapHeight());

batcher.end();

// shapes render
shapeRenderer.begin(ShapeType.Line);
shapeRenderer.setColor(new Color(1, 0, 0, 1));
shapeRenderer.rect(tankButton.x,tankButton.y,tankButton.width,tankBut
ton.height);
shapeRenderer.setColor(new Color(0, 1, 0, 1));
shapeRenderer.rect(notesButton.x,notesButton.y,notesButton.width,note
sButton.height);
shapeRenderer.end();
}
```

13. To finish, copy the code below to play the sounds. Run the program.

```
public boolean touchUp(int screenX, int screenY, int pointer, int
button) {

    Gdx.app.log("touchUp", "x:" + screenX + " y:" + screenY);

    int viewportX = (int) (screenX * camera.viewportWidth /
Gdx.graphics.getWidth());
    int viewportY = (int) ((Gdx.graphics.getHeight() - screenY) *
camera.viewportHeight / Gdx.graphics.getHeight());

    if(notesButton.contains(viewportX, viewportY)){
        Gdx.app.log("RectContains", "Notes button");
        AssetLoader.music.stop();
        AssetLoader.music.play();
    }

    if(tankButton.contains(viewportX, viewportY)){
        Gdx.app.log("RectContains", "Tank button");
        AssetLoader.explosion.stop();
        AssetLoader.explosion.play();
    }
    return false;
}
```