AndEngine

Eine 2D-Spiel-Engine für Android

Henrik Voß, 2012

Übersicht

- AndEngine
  - Einleitung
  - Theoretischer Hintergrund
  - Praktische Vorstellung

Was ist eine Spiel-Engine?

- Laufzeitumgebung für Spiele
- Bietet verschiedene Funktionalitäten
  - Grafik-Engine
  - Physik-Engine
  - Soundsystem
  - Mehrspielerunterstüzung
  - ...
- Bereits der Standard MDIP 2.0 von 2002 schreibt eine Game API vor

Was ist die AndEngine?

- 2D-Spiel-Engine
- Benutzt die Hardwarebeschleunigung OpenGL ES
- Nur für Android
- Entwickelt von Nicolas Gramlich

Webseite:
http://www.andengine.org

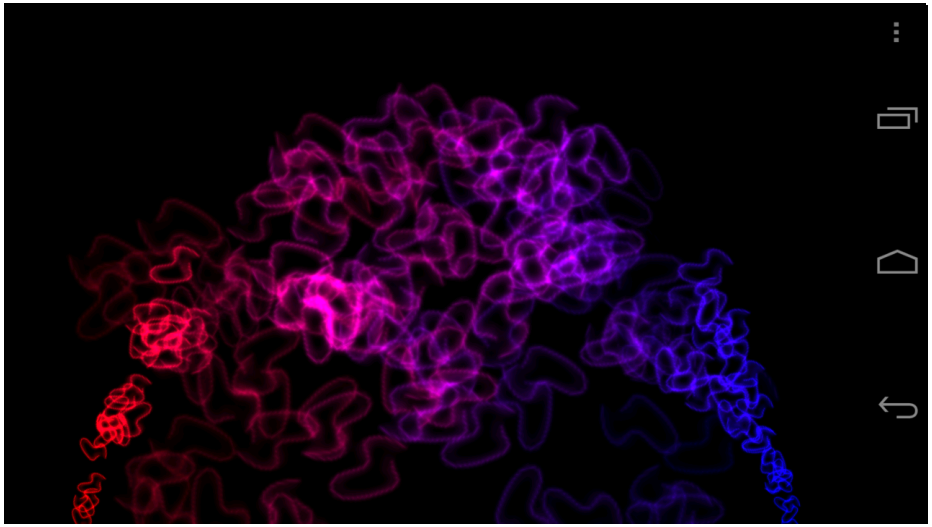
Bildquelle: http://www.andengine.org/blog/about/

Warum AndEngine?

- Einfach zu benutzen
- Sehr bekannt
  - Die Wikiseite listet über 170 Apps
- Kostenlos & Open Source (LGPL)
- Einschränkungen
  - Nur 2-dimensionale Grafik
  - Nicht platformunabhängig
  - Hierfür wäre z.B. das kostenpflichtige Unity3D geeignet

Features

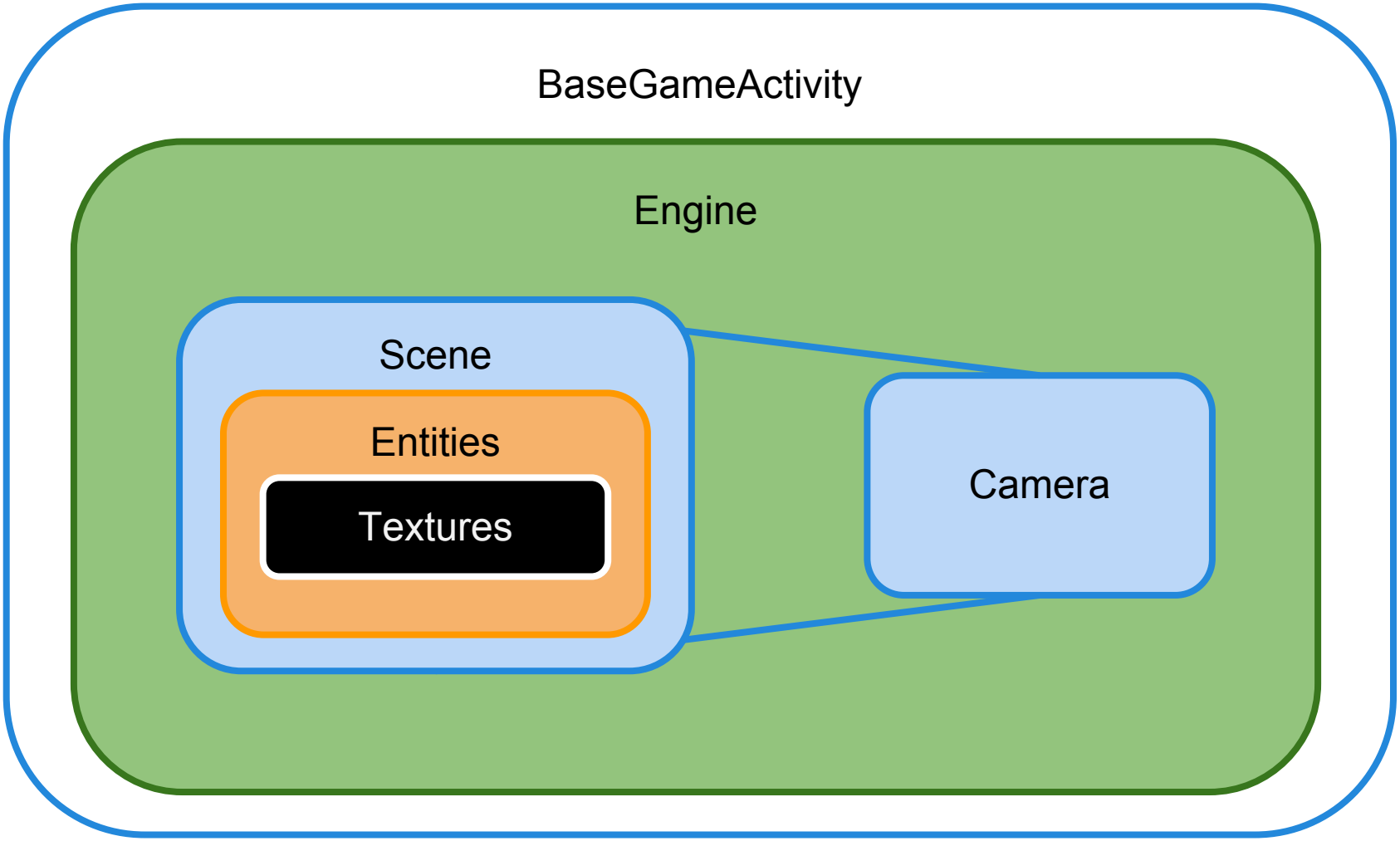
- Alle in Spiel-Engine erwähnten Funktionalitäten
- Grafik-Engine bietet u.a.
  - Animierte Sprites
  - Textrendering
  - Partikelsystem
- Und weiteres
  - On Screen Joysticks
  - Multitouch
  - ...



Partikel System der AndEngine Examples

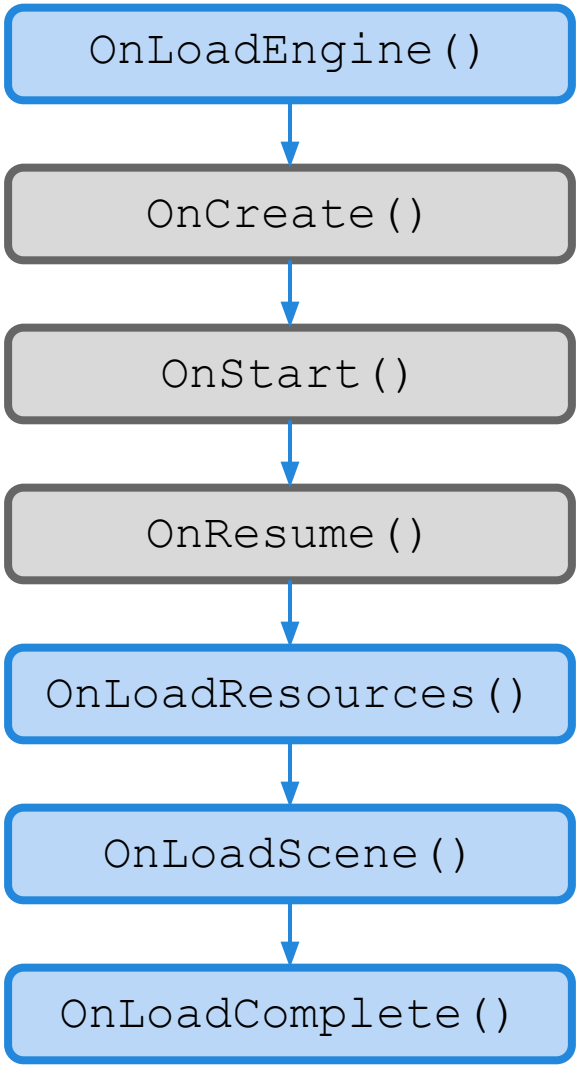
Quelle Partikelsystem: http://code.google.com/p/andengineexamples/

Aufbau eines Spiels



Bildquelle: http://www.slideshare.net/ANDLABSMUNICH/andengine Folie 14

Erweiterter Lifecyle



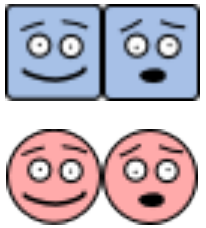
Quelle: http://www.slideshare.net/ANDLABSMUNICH/andengine
Folie 15

Grafiken

- `TextureAtlas`
- `TextureAtlasRegion`
- `Sprite`
  - `TiledSprite`
  - `AnimatedSprite`

Livecoding

- Stufe 1
  - Darstellen eines Rahmens
- Stufe 2
  - Laden von Grafiken
  - Hinzufügen durch Berührung
- Stufe 3
  - Einbinden der Physik-Engine
  - Smilies fallen nach unten
- Stufe 4
  - Einbinden des Accelerometers
  - Smilies bewegen sich durch Gerätbewegung



Quellcode: http://code.google.com/p/andengineexamples/

Stufe 1 1 / 2

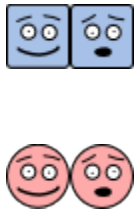
```
public class Iteration1Activity extends BaseGameActivity {
    /* [...] Member-Variablen */
    public Engine onLoadEngine() {
        Camera camera = new Camera(0, 0, 720, 480);

        EngineOptions engineOptions = new EngineOptions(
            true,
            ScreenOrientation.LANDSCAPE,
            new RatioResolutionPolicy(720, 480),
            camera);

        engineOptions.getTouchOptions().setRunOnUpdateThread(true);

        return new Engine(engineOptions);
    }
```

Stufe 1 2 / 2

```
    public Scene onLoadScene() {
        this.mScene = new Scene();
        this.mScene.setBackground(new ColorBackground(0, 0, 0));

        Shape ground = new Rectangle(0, 480 - 2, 720, 2);
        this.mScene.attachChild(ground);
        /* [...] Weitere Rechtecke erstellen */

        return this.mScene;
    }
}
```

Stufe 2 1 / 3

```
public class Iteration2Activity extends BaseGameActivity
                implements IOnSceneTouchListener {
    /* [...] Member-Variablen */
    public void onLoadResources() {
        this.mBitmapTextureAtlas = new BitmapTextureAtlas(
            64, 64, TextureOptions.BILINEAR_PREMULTIPLYALPHA);
        /* [...] */
        this.mBoxFaceTextureRegion =
            BitmapTextureAtlasTextureRegionFactory.
                createTiledFromAsset(
                    this.mBitmapTextureAtlas, this,
                    "face_box_tiled.png", 0, 0, 2, 1);

        this.mEngine.getTextureManager().loadTexture(
            this.mBitmapTextureAtlas);
}
```

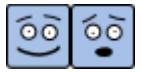


Stufe 2 2 / 3

```
@Override
public boolean onSceneTouchEvent(
    final Scene pScene, final TouchEvent pSceneTouchEvent) {

    if (pSceneTouchEvent.isActionDown()) {
        this.addFace(
            pSceneTouchEvent.getX(),
            pSceneTouchEvent.getY());
            return true;
    }

    return false;
}
```

Stufe 2 3 / 3

```
private void addFace(final float pX, final float pY) {
    this.mFaceCount++;

    final AnimatedSprite face;

    if (this.mFaceCount % 2 == 0) {
        face = new AnimatedSprite(pX, pY,
            this.mBoxFaceTextureRegion);
    } else {
        face = new AnimatedSprite(pX, pY,
            this.mCircleFaceTextureRegion);
    }

    face.animate(200);

    this.mScene.attachChild(face);
}
}
```



Stufe 3 1 / 3

```
public class Iteration3Activity
                extends BaseGameActivity
                implements IOnSceneTouchListener {

    private static final FixtureDef FIXTURE_DEF =
        PhysicsFactory.createFixtureDef(1, 0.5f, 0.5f);

    private PhysicsWorld mPhysicsWorld;
```

Stufe 3 2 / 3

```
@Override
public Scene onLoadScene() {
    /* [...] Initialisierung der Scene */
    this.mPhysicsWorld = new PhysicsWorld(
        new Vector2(0, SensorManager.GRAVITY_MOON), false);

    /* [...] Generierung des Rahmens */

    FixtureDef wallFixtureDef = PhysicsFactory.createFixtureDef(
                                                0, 0.5f, 0.5f);
    PhysicsFactory.createBoxBody(this.mPhysicsWorld, ground,
        BodyType.StaticBody, wallFixtureDef);
    /* [...] Die anderen Rahmenelemente werden auch mit Physik versehen */

    /* [...] Rahmen der Scene hinzufügen */

    this.mScene.registerUpdateHandler(this.mPhysicsWorld);
}
```

Stufe 3 3 / 3

```
    private void addFace(final float pX, final float pY) {
        this.mFaceCount++; final AnimatedSprite face; final Body body;

        if(this.mFaceCount % 2 == 0) {
            face = new AnimatedSprite(pX, pY, this.mBoxFaceTextureRegion);
            body = PhysicsFactory.createBoxBody(
                this.mPhysicsWorld, face, BodyType.DynamicBody,FIXTURE_DEF);
        } else {
            face = new AnimatedSprite(pX, pY, this.mCircleFaceTextureRegion);
            body = PhysicsFactory.createCircleBody(
                this.mPhysicsWorld, face, BodyType.DynamicBody, FIXTURE_DEF);
        }
        face.animate(200);

        this.mScene.attachChild(face);
        this.mPhysicsWorld.registerPhysicsConnector(
            new PhysicsConnector(face, body, true, true));
    }
}
```

Stufe 4 1 / 2

```
public class Iteration4Activity
            extends BaseGameActivity
            implements IAccelerometerListener, IOnSceneTouchListener {

    @Override
    public void onAccelerometerChanged(
                final AccelerometerData pAccelerometerData
            ) {
            final Vector2 gravity = Vector2Pool.obtain(
                pAccelerometerData.getX(),
                pAccelerometerData.getY()
            );
        this.mPhysicsWorld.setGravity(gravity);
        Vector2Pool.recycle(gravity);
    }
```

Stufe 4 2 / 2

```
    @Override
    public void onResumeGame() {
        super.onResumeGame();

        this.enableAccelerometerSensor(this);
    }

    @Override
    public void onPauseGame() {
        super.onPauseGame();

        this.disableAccelerometerSensor();
    }
}
```

Vielen Dank für Eure Aufmerksamkeit

Fragen?

Link zu den Unterlagen:
https://android.dsi8.de/