Anton Kornholt

# libGDX

## En kort artikel om libGDX egenskaber


**Introduktion**

I denne artikel vil jeg diskutere, hvordan libGDX kan bruges til, at lave mobilspil. Hvorfor dette framework udskiller sig fra andre, og hvorfor og hvornår man kan bruge libGDX under ens udvikling.

Jeg vil herunder komme ind på fordele og ulemper ved at bruge dette framework.

Derudover vil der også være eksempler fra et simpel “Raindrop Bucket” spil.

## Hvad er libGDX?

libGDX er et java udviklings framework, der ved hjælp af en fælles API kan udvikle spil på tværs af forskellige platforme. Disse platforme er:

Windows

Linux

Mac OS X

Android (2.2+)

Blackberry

iOS

Java Applet

Javascript/WebGL

Udover at tilbyde support til tvær-platforms udvikling kan libGDX også integrere en række forskellige tredjeparts tilføjelser. På den måde kan man selv customize og personliggøre ens brug af frameworket.

Grafikken der renders med libGDX er OpenGL ES 2.0 på samtlige platforme. Det er både muligt at lave 2D og 3D spil med denne grafik.

Man kan godt sige, at med libGDX, så bliver ens programmering ret high level. Hvilket vil sige, at man abstrahere en del væk fra standard java kode og prøver at gøre udviklingen mere brugervenlig og let forståelig. Der kræves ikke kendskab til flere forskellige kodesprog end java. Alt tvær-platform programmering klarer libGDX selv.

## Hvordan bruger man libGDX?

Et libGDX projekt er temmelig ligetil at starte. Du skal først og fremmest downloade en libGDX setup jar fra følgende link:

https://libgdx.badlogicgames.com/download.html

Dernæst skal man navigere derhen, hvor man har placeret ens jar fil på computeren og via commandlinjen taste:

java -jar ./gdx-setup.jar

Herefter popper der en setup-skærm frem, der beder dig udfylde en række informationer om dit projekt. Dette inkluderer hvilke platforme, du gerne vil have projektet skal være kompatibelt med. Det er her også muligt at vælge ekstra extensions til ens projekt.

Det eneste, der tit kan volde lidt problemer her, er ved valg af SDK. Det er nemlig ikke alle, der ved, hvor deres SDK er placeret på computeren.

En let måde at finde dette sted er ved Android Studio opstart at trykke på configure og dernæst vælge “system settings” og så “Android SDK”. Herinde kan man finde frem til ens SDK placering.

Når alt er udfyldt og man trykker generate så genereres, der et tvær-platforms projekt på den valgte lokation. For at importere ens nye projekt i Android Studio, skal du vælge “Import Projects” og dernæst navigere hen til build.gradle filen i ens nye project. Så simpelt er det.

Lige nu er der dog en lille fejl med Gradle versionen, der bliver brugt i libGDX. Det er nødvendigt at downgrade denne til 3.3 fra 4.1. I følgende “stackoverflow” tråd diskuteres problemet og der nævnes en simpel fix til at downgrade gradle versionen i ens projekt:

https://stackoverflow.com/questions/46975883/error2-0-plugin-with-id-jetty-not-found

## At kode med libGDX

Som et eksempel på et spil lavet ved hjælp af libGDX gennemgik jeg følgende tutorial:

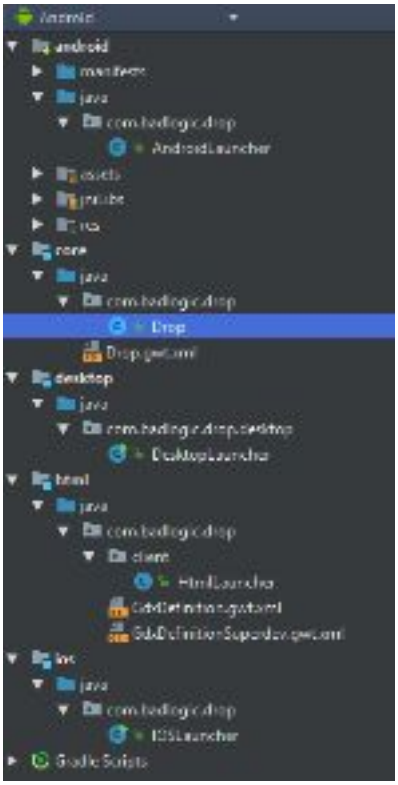
https://github.com/libgdx/libgdx/wiki/A-simple-game

Her går spillet i al sin enkelhed ud på, at man har en spand i bunden af skærmen, der skal samle regndråber før de ryger ud af billedet. Et simpelt koncept, men som der kan bygges en del videre på. Det var en god start for at se, hvad libGDX kan tilbyde.

Det første man skal bestemme sig for, er hvordan spillet skal vises på tværs af de forskellige platforme. Man skal altså f.eks gerne vælge, hvordan ens resolution skal konfigureres. Her i starten skal man også tænke over, om der er andre platforms specifikke ting man vil tage højde for når spillet afspilles. På mobil kunne man f. eks slå kompas og accelerometer fra ved spil launch. Herunder ses et eksempel på en modificeret startklasse for ens android udgave:

```
public class AndroidLauncher extends AndroidApplication {
    @Override
    protected void onCreate (Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        AndroidApplicationConfiguration config = new AndroidApplicationConfiguration();
        config.useAccelerometer = false;
        config.useCompass = false;
        initialize(new Drop(), config);
    }
}
```

Her initialiserer man spillet med den platforms specifikke config. Således kan man ændre configs i alle ens start/launch-klasser, således at ens spil er ordentlig tilpasset samtlige platforme. Her ses et overview af ens forskellige platforme og deres folders i projektet:



Når man så har konfigureret alle start/launch-klasserne, så man er tilfreds med deres start konfigurationer, så kommer man til selve spiludvikling. Der tager man gerne og initialisere alle ens assets i starten i ens OnCreate metode. Her arbejder man udelukkende på udviklingen i ens core folder og i dette tilfælde i den markerede "Drop"-klasse, som der ses i billedet ovenfor.

Det er det som vi skriver her, der initialiseres i de enkelte start/launch-klasser.

```
public void create() {
    // load the images for the droplet and the bucket, 64x64 pixels each
    dropImage = new Texture(Gdx.files.internal( path: "droplet.png"));
    bucketImage = new Texture(Gdx.files.internal( path: "bucket.png"));

    // load the drop sound effect and the rain background "music"
    dropSound = Gdx.audio.newSound(Gdx.files.internal( path: "drop.wav"));
    rainMusic = Gdx.audio.newMusic(Gdx.files.internal( path: "rain.mp3"));

    // start the playback of the background music immediately
    rainMusic.setLooping(true);
    rainMusic.play();

    // create the camera and the SpriteBatch
    camera = new OrthographicCamera();
    camera.setToOrtho( yDown: false,  viewportWidth: 800,  viewportHeight: 480);
    batch = new SpriteBatch();

    // create a Rectangle to logically represent the bucket
    bucket = new Rectangle();
    bucket.x = 800 / 2 - 64 / 2; // center the bucket horizontally
    bucket.y = 20; // bottom left corner of the bucket is 20 pixels above the bottom screen edge
    bucket.width = 64;
    bucket.height = 64;

    // create the raindrops array and spawn the first raindrop
    raindrops = new Array<Rectangle>();
    spawnRaindrop();
}
```

Her ses hvordan de forskellige assets initialiseres i OnCreate metoden i "Drop"-klassen.

Dernæst kan man skrive metoder for ens spillogik. Som et eksempel her viser jeg, hvordan jeg får regndråber til at opstå tilfældige steder i toppen af spillets grænseflade:

```
private void spawnRaindrop() {
    Rectangle raindrop = new Rectangle();
    raindrop.x = MathUtils.random(0, 800-64);
    raindrop.y = 480;
    raindrop.width = 64;
    raindrop.height = 64;
    raindrops.add(raindrop);
    lastDropTime = TimeUtils.nanoTime();
}
```

Et eksempel på, hvordan gdx tillader support til mange forskellige platforme ses tydeligt i render metoden, hvor der defineres to forskellige måder at bevæge ens regndråbesamlende spand på:

```
// process user input
if(Gdx.input.isTouched()) {
    Vector3 touchPos = new Vector3();
    touchPos.set(Gdx.input.getX(), Gdx.input.getY(), z: 0);
    camera.unproject(touchPos);
    bucket.x = touchPos.x - 64 / 2;
}
if(Gdx.input.isKeyPressed(Keys.LEFT)) bucket.x -= 200 * Gdx.graphics.getDeltaTime();
if(Gdx.input.isKeyPressed(Keys.RIGHT)) bucket.x += 200 * Gdx.graphics.getDeltaTime();
```

Alt i alt, er det meget hurtigt at supportere ens spil med forskellige inputs ved hjælp af GDX.

## Konklusion

Det har været en god oplevelse at prøve at bruge libGDX til udviklingen af et simpelt mobilspil. Jeg kan se et virkelig stort potentiale i det, hvis målet er at udvikle et spil, der skal kunne spilles på tværs af en masse forskellige platforme.

Det er dejligt, at man kun skal skrive ens kodelogik en gang og at GDX sørger for resten. Hvis man skulle være lidt kritisk vil jeg dog sige, at man overgiver en del af kontrollen til GDX, når man begynder at bruge det. Man sidder som sagt på et ret højt plan og koder, og selvom man får meget hjælp af programmet, så skal man stadig sætte sig ind i de forskellige værktøjer som libGDX tilbyder før man for alvor bliver en haj til at bruge alle funktionaliteterne.

Alt i alt ville jeg dog anbefale dette framework, hvis man står og er interesseret i at prøve kræfter med for alvor at lave tvær-platformsspil. Det er meget nemt at komme i gang med, hvis man følger den samme guide, som der blev postet ovenfor, og man ser meget hurtigt, at man potentielt kan lave ret komplicerede spil uden den store erfaring indenfor hverken Android eller general spiludvikling.

Hvis man dog er interesseret i at rode med mere specifikke android spil, så kunne man meget vel også tage et kig på androids indbyggede canvas grafik, da det måske sagtens kan opfylde de krav man har til ens spil. libGDX er gerne til, hvis du vil lave lidt mere avanceret grafik og især, hvis det skal laves til flere forskellige platforme.

## Ressourcer

Github til Drop-spil med links til guide og setup i readme:

https://github.com/AntonKornholt/LibGDX

libGDX dokumentation:

https://libgdx.badlogicgames.com/documentation/

Download libGDX:

https://libgdx.badlogicgames.com/download.html