# The LibGDX Game Engine

## Contents



# 1 Skills/Concepts

This module will involve using the LibGDX engine to create and build a new game. This will involve the following concepts:

- software development kits (SDKs)
- modules/libraries
- console use

## 1.1 Software Development Kits (SDKs)

From Wikipedia:

> A software development kit (SDK or devkit) is typically a set of software development tools that allows the creation of applications for a certain software package, software framework, hardware platform, computer system, video game console, operating system, or similar development platform.

Or more simply: It's the stuff you code with to make coding more convenient. It's anything from specialized tools to just libraries you can include in your own code that makes development easier.

## 1.2 Modules/Libraries

Java statements are placed in methods that are encapsulated by classes that exist in packages. Libraries in java are collections of packages that contain classes. It's a way of organizing code to make it more convenient for doing other code. Generally, SDKs provide one or more libraries.

### 1.3 Console Use

You will be using a terminal emulator to execute commands to help you compile and run your code.

| Command | Description |
|---|---|
| wget | The non-interactive network downloader |
| java | The Java Virtual Machine |
| gradlew | The bundled Gradle build system |
| git | Source code versioning tool |

## 2 Preparation

You should be fine booting up any Linux distribution as long as it has a Java SDK.

## 3 Getting LibGDX and Creating a Project

### 3.1 Background

From the web site:

> LibGDX is a relatively low level, free, open source cross platform game development framework. The goal of the project is to assist you in creating games/applications and deploy to desktop and mobile platforms without getting in the way and letting you design however you like.

This module will cover fetching the engine and creating a project.

### 3.2 Process

0. Get the eninge using `wget`:

   wget https://libgdx.badlogicgames.com/nightlies/dist/gdx-setup.jar

Success looks something like this:

```
Resolving libgdx.badlogicgames.com (libgdx.badlogicgames.com)... 95.216.8.184
Connecting to libgdx.badlogicgames.com (libgdx.badlogicgames.com)|95.216.8.184|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://libgdx.badlogicgames.com/ci/nightlies/dist/gdx-setup.jar [following]
--2019-04-22 22:18:35--  https://libgdx.badlogicgames.com/ci/nightlies/dist/gdx-setup.jar
Reusing existing connection to libgdx.badlogicgames.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 1729604 (1.6M) [application/java-archive]
Saving to: ‘gdx-setup.jar’

gdx-setup.jar                     100%[============================================================>]   1.65M  1.52MB/s

2019-04-22 22:18:36 (1.52 MB/s) - ‘gdx-setup.jar’ saved [1729604/1729604]
```

1. Run the downloaded jar file:

   java -jar gdx-setup.jar

2. Set the basic project properties (Name, Package, etc.) to anything you want (provided you use valid values) but uncheck the Android and iOS-related *Sub Projects* since we're only focusing on Java/Desktop, and so the download wont take as long.

3. Click *Generate* and wait patiently for the downloading to finish. You'll want to look for the "Done!" in the box below the generate button.

4. Verify that your project will build and run by invoking the bundled `gradlew` command.


Figure 1: LibGDX Project Generator Screenshot


Figure 2: LibGDX Project Generator Screenshot - Done

Note: Here I have to cd into the new directory that I chose when generating the project.

```
cd test
./gradlew run
```

Successful execution will show the following at the console:

```
> Task :core:compileJava
warning: [options] bootstrap class path not set in conjunction with -source 1.6
1 warning

> Task :desktop:compileJava
warning: [options] bootstrap class path not set in conjunction with -source 1.6
1 warning
<===========--> 87% EXECUTING [40s]
> :desktop:run
```

Which may be obscured by the window that you really care about, that looks like this:


Figure 3: LibGDX Project - Successful build & run

### 3.3 Conclusion

You now have a fairly plain project that's ready for editing and customizing. If you need to, you can repeat the steps to create new projects.

# 4 Getting the Box2D Editor

## 4.1 Background

To maximize code re-use and limit reinventing the wheel, you can simplify the use of physics in your own creations by leveraging tools that allow you to create objects that can be reduced into regions for collision detection without having to code every single possibility on your own. This module covers fetching one such tool that will be used in future modules.

## 4.2 Procedure

0. Clone the repository (do this somewhere OTHER than in your current project):

   git clone https://github.com/julienvillegas/box2d-editor

1. Run the embedded `gradlew` script:

   cd box2d-editor bash ./gradlew run

Note: Some features of the editor are buggy. First image loaded will not display unless you create a dynamic object to give focus to temporarily. You can create one temporarily to get the image to display, then delete it later. Also, the advanced options for automatically tracing objects may not work.

## 4.3 Conclusion

There wasn't much to this module, but now the tool will be available for other modules and future projects.

# 5 Fetch and Try some samples

## 5.1 Background

Any decent SDK or Engine will have samples available. If not directly provided by the project itself, then provided somewhere by the community. You can find samples for libgdx on the tutorials web site.

Github projects with sample code:

- basic image
  - https://github.com/julienvillegas/libgdx.info-Basic_Image
- basic label
  - https://github.com/julienvillegas/libgdx.info-Basic_Label
- buttons-scene2d
  - https://github.com/julienvillegas/libgdx.info-Basic_Button
- basic action
  - https://github.com/julienvillegas/libgdx.info-Basic_Action
- runnable action
  - https://github.com/julienvillegas/libgdx.info-Runnable_Action
- basic screen
  - https://github.com/julienvillegas/libgdx.info-Basic_Screen
- basic camear
  - https://github.com/julienvillegas/libgdx.info-Basic_Camera
- parallax
  - https://github.com/julienvillegas/libgdx.info-Parallax
- ui multiplexing
  - https://github.com/julienvillegas/libgdx.info-inputMultiplexing
- particle effects
  - https://github.com/julienvillegas/libgdx.info-particleEffect
- box2d-basic
  - https://github.com/julienvillegas/libgdx.info-Box2D-basic
- box2d complex bodies
  - https://github.com/julienvillegas/libgdx.info-Box2d-importing-models

- box2d light pooled particle effects…
  - https://github.com/julienvillegas/libGDX.info-Box2DLight_Basic
- shaders
  - https://github.com/julienvillegas/libgdx.info-Shader-Shockwave

You can generally fetch a sample via a `git clone` and then use the bundled `gradlew` to execute them.

## 5.2 Procedure

0. Clone a sample as listed above:

   git clone https://github.com/julienvillegas/libgdx.info-Basic_Image

1. Run the cloned sample:

   cd libgdx.info-Basic_Image/ ./gradlew run

2. Repeat with more samples from above.

Alternately, each of the pages listed above in the background have a video or animation of some kind illustrating the sample.

## 5.3 Conclusion

Now you have the basic tools necessary to fetch and run a sample, as well as resources that explain those samples. With this information you can start to get an appreciation for what’s possible with the engine,