**Lesson**

Box2D

Box2D is a complete realistic physics simulator, which can apply and emulate different characteristics of objects. It is useful as a form of template, as even at the very simple stages of creation, it handles various aspects according to a default configuration that include (but are not limited to) collisions, velocity, and gravity. The most difficult part about this library is that any desired change to the way the system works requires a strong knowledge of how the engine works as well as the ability to create heavy modifications for the purpose of making the changes not conflict with the rest of the engine.

It has a similar method of creation to Scene2D; in Scene2D, you first create a Stage and add a variety of Actors onto it. In Box2D, however, you first create a World (which handles collision and gravity), then add various combinations of Bodies and Fixtures (from hereon called “objects”) to the World to allow for the physics engine to both create and apply the assigned physics to the bodies.

By also including the external library Box2DUtils, you also receive the functionality of loaders that can read in various types of files, such as the Box2DMapObjectParser, which loads in TiledMaps in a way that is compatible with Box2D. Assuming that the TiledMap (.tmx) file has an object layer with shapes drawn, the Box2DMapObjectParser will load in those shapes and place them accordingly, while also applying the custom properties that you set in the Tiled program. However, inclusion of this external library will also not allow you to build the project to an html project under any circumstance (both gradle and Android Studio will not compile).

Collisions between two bodies is very complicated. An automated process by the World will return a Contact (which includes the two fixtures that have collided) in the instance that a collision has occurred. It will then go through four stages - BeginContact, PreSolve, EndContact, PostSolve. It is here that you can decide what should happen when a contact happens, with the possibility of making it specific as to what action (or lackthereof) to take in the occurrence of a collision between two specific fixtures.

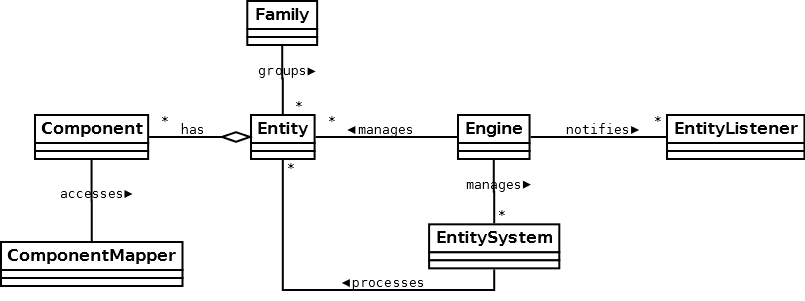
Tiled

Tiled is a map creator/editor that is compatible with libGDX thanks to the Box2dUtils external library mentioned above. The map editor handles both spritesheet importing, drawing of the map, and optionally, the Box2D body creation and placement. It very much simplifies the process of creating a map and adding appropriate collision management, as all of that can be handled right in the program itself.

For the purposes of compatibility with Box2D, Tiled allows you to create bodies, determine their shape and size, and add various properties to them. Not only can you name them and decide whether or not it is a body, fixture, or object, but you can also add custom properties in the case that you want to add something that Tiled does not handle (including custom variables and other aspects of Box2D, such as userData or the categoryBits). This allows you to create things such as doorways, spawnpoints, and environmental hazards. All that is needed is a clear idea of how you want to approach the situation.

Ashley

Ashley is a library independent of Box2D. It is an ECS (Entity Component System), which assists in the separation of different parts of the project. In Ashley, you have three steps: you first add Components to an Entity, then you create a System that can handle the Entity.



You can create a variety of Systems using Ashley. For instance, in our game, we have three systems running: MovementSystem, GravitySystem, and a RenderingSystem. These three systems run in sequence to create a game similar to our original non-Ashley project.