**Lessons**

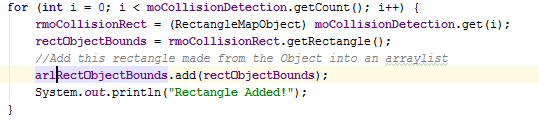
**Lesson 1: Tiled and Collision Detection**

Tiled contains a feature in which you are able to add object layers as well as tile layers. We used these object layers in order to perform our collision detection.

After creating our map in tiles, we created an Object Layer of rectangles on top of the tiles that we didn’t want the player to make contact with (e.g. rocks, trees).

In Android Studio, we then created a MapObject, whose purpose was to get all of the objects from the object layer we created in Tiled.

We then looped through all of the MapObjects, and created a RectangleMapObject out of each one. These RectangleMapObjects create a Rectangle object from the MapObject at the current index of the loop, that contains the x-coordinate, y-coordinate, length, and width of the MapObject.

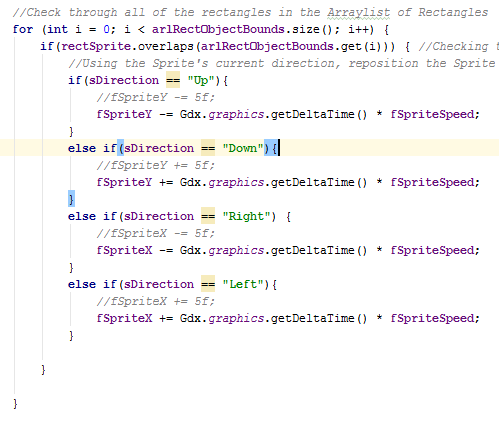


The next step was to convert the RectangleMapObject into a Rectangle, again with those same dimensions and location, and added them into an ArrayList of Rectangles.

Within the render function, we created a Rectangle that had the same height, width, x-coordinate, and y-coordinate as the player sprite. As the player moved and the x and y-coordinates changed, the rectangle would change with them.



Also within the render function we constantly looped through the entire ArrayList of collision detection Rectangle. If the player Rectangle ever intersected one of the Rectangles in the Arraylist, the game would reset the player back to a point before the player rectangle made contact with that rectangle.

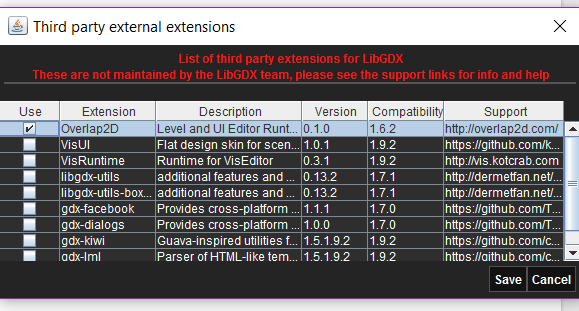


**Lesson 2: Overlap2D**

Overlap2D is at its core a fancy visual assets manager, allowing for quick placement and loading of objects to a game screen originally using Scene2D but in recent versions upgrading to Ashley.

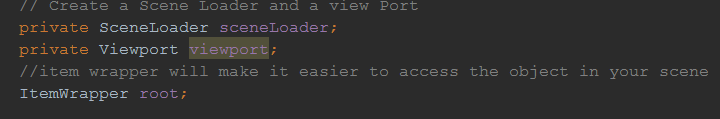
The first thing you need to do is download the newest version of the overlap2D jar and opening that. You will then import all assets that you intend to use, this can include images, sprite animations from either an atlas or a sequence of pngs, and particle effects.

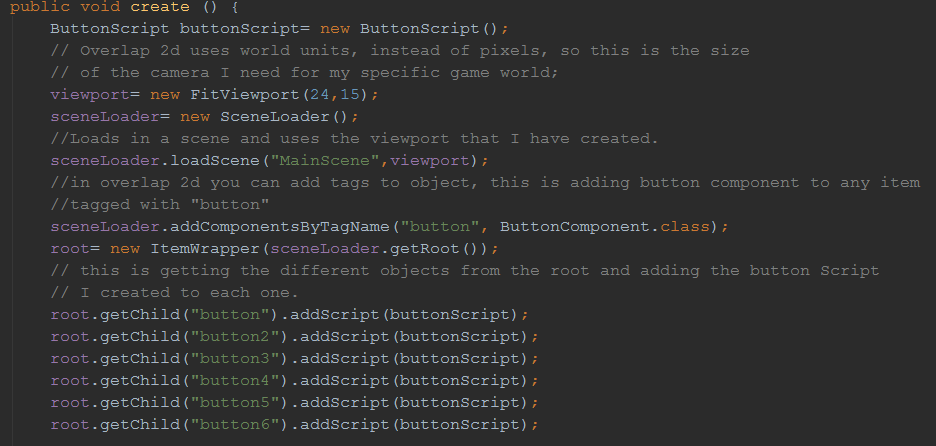
The interface is drag and drop, you can drag the images into the screen, by default (0,0) is in the middle of the screen and you can scroll by holding space then click and dragging. When Objects are dragged onto the screen they are added to the scene (Default is MainScene).

Next Step is to set up a Libgdx Project which is the same as usual except you have to make sure that click on 3rd party extensions and make sure that overlap2D is checked, also it is a good idea to check Ashley, Box2d.

Once you have created your project, go into Overlap2D once again and click on file>exportsettings and set the export directory to the assets folder of your libgdx project. This way you can click export and all your assets in overlap2D are automatically sent into your libgdx project.

Now let's get to the the actual code part of using Overlap2D with Libgdx. To make things easier I already commented instructions into my project code and will be using screenshots to quickly explain each section.



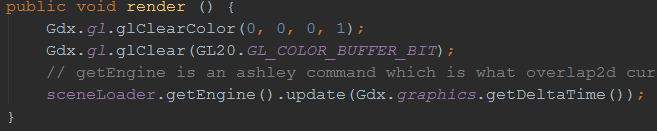
Then in create method:

Overlap2D uses world units in their scenes, when you create an overlap2d project you can choose how many pixel per world unit, for 1:1 with pixels choose 1. In the fit viewport you set it to the world units you want show which can be found by using the rulers/scales in overlap2D.

The scene is auto moved to the assets folder when you Export in overlap2d and is how changes made in overlap 2d are recreated in libgdx.

Item wrapper allows you to easily call the scene, and using getChild calling the object with whatever you named it in the overlap 2d editor identifier box

Finally in the render method:



You call the sceneloader and getEngine(), this is a Ashley feature. Then update with the

Gdx DeltaTime.