Objective:

Create an Augmented reality library for mobile devices that does the following:

- Render 3D models (.obj) fast and efficiently.
- Support object render on runtime in order to download objects from internet.
- Use a marker as a positioner for the render.
- Scale the object according to the camera view.
- Keep track of the object even when the marker it's out of sight.

Justification:

The library can be used for whatever purpose. But the inspiration came from previewing furniture with the app before actually acquiring the piece.

Resources needed for this project:

Aside from the obvious (Android device, SDK, etc…)

- Libraries:
    - Vuforia [**https://developer.vuforia.com/**]
    - Crystax NDK [**https://www.crystax.net/en/android/ndk**]
    - libzip [**http://www.nih.at/libzip/**]
    - OpenGL ES [**https://www.khronos.org/opengles/**]

Vuforia:

Licencie: GPL

Used to keep track of markers and camera visualization.

Crystax NDK:

License: GPL

Better compiler but google ndk works fine too.

libzip:

License: Free to distribute as long as keeping the copyright.

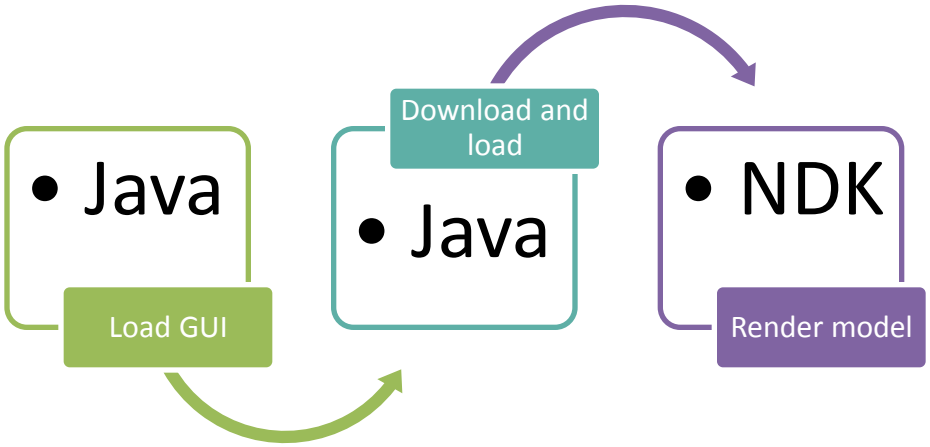
Used to decompress downloaded .zip files containing the .obj.

OpenGL ES:

License: Divese.

Used to render 3d models.

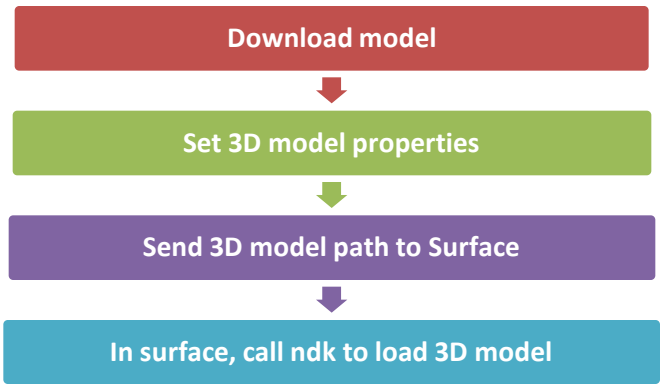
Methodology:



Load GUI

Load native library modules (C++).
Start main activity, set up **Vuforia** and GUI components. Then initialize camera.

Download mesh model and load



Download process:

```
private class DownloadTask extends AsyncTask<String, Integer, Model3D>
```

Once GUI has been initialized select an object mesh and download it in the background. This way camera view will not freeze.

After file has been downloaded fire interface method to render the model ASAP.

3DModel object (POO object):

This part is important because since every object have different rotations, scales, sizes etc.. some might render too big others too small.

This is where Model3D enters. This is Server-side job, and it gets the right positions, scales, rotations needed in order as desired. (This is a manual job and has not been automatized).

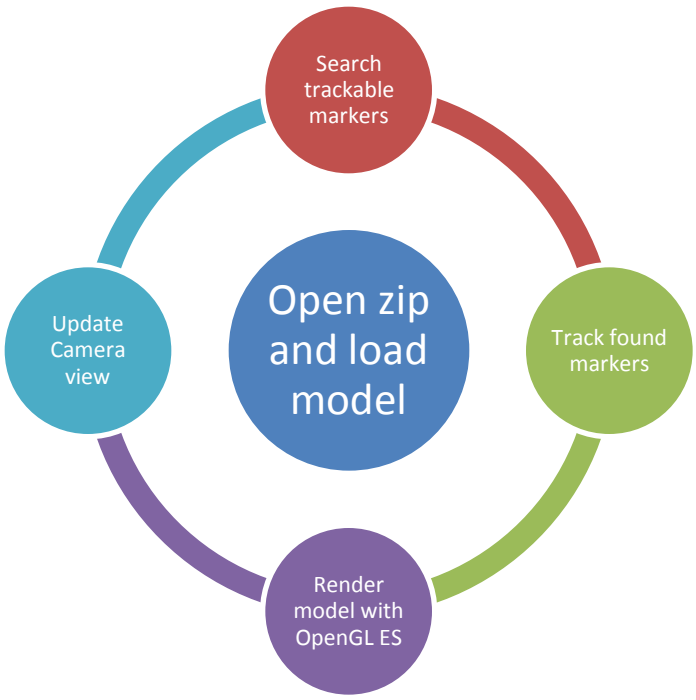
```
Model3D{

        /** Properties */
        private InputStream stream;
        private String streamName;
        private String modelName;
        private InputStream texture;
        private float objectScale;
```

Load 3D Model:

Send downloaded model into native code so OpenGL ES can render it by passing it to the `GLSurfaceView.Renderer.` Load the model mesh into NDK and loop

```
public native void loadModel(String path, String model, float scale, int bnn);
```

Download mesh model and load



Tests

There are several popular libraries made to facilitate OpenGL rendering on Java.

	Quality	**Scope**	**Rendering time**
Jpct-ae	Low	Low	Fast
Rajawali	High	Medium	Slow
Native loader	High	High	Fast

Additionally, the same project on **Unity SDK** was made and these were the results.

The object renders are great and it's very easy to develop there. But theres less control