In May 2016, Google introduced Google VR (https://developers.google.com/vr/), including two mobile virtual reality (VR) platforms consisting of **a).** an update on Cardboard VR and introducing the new higher quality **b).** Daydream platform.

Just as the well-known `cardboard` had beem rebranded to "Google VR", or GVR, we require the following GVR-sdk to work on:

Google VR SDK 0.8 -> 1.10.0

accessible via the following commands:

```
git clone https://github.com/googlevr/gvr-android-sdk.git
```

Note

Version Conflit Problems

- The last Android build tools, since 24.0.0, works defauled with JDK-8. However, if there comes the error, JDK-7 used in build processing, even that the last JDK had been updated to JDK-8; then modify the "JDK location" setting in Android Studio:

  ```
  [File] -> [Project Structure] -> [SDK Location] -> [JDK Location]
  /Library/Java/JavaVirtualMachines/jdk1.8,0_101.jdk/Contents/Home
  ```

  in Mac Box, others also correct the right location where JDK was installed.
- Errors, "`... value not found...`", caused by SDK version, could be solved by changing SDK version from grandle (app).
- Should Android Studio run *off-line* possibly? Theoretically, yes. However, running online at the project first created is better since it could download the absent packages automatically if has. The error due to such kind of fail to create project can be removed by deleting uncessary depencies in grandle.build(app):

  ```
    ...
    androidTestCompile('com.android.support.test.espresso:espresso-core:
  2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotatio
  ns'
    })
    ...
  ```

```
Application name: Panorama
Company domain: com.gvr.widgets
---  creat project ---
```

```
add new modules: sdk-base-1.10, sdk-common-1.10, sdk-panowidget-1.10
from [open module setting] add module depencies created above
AndroidManifest.xml:

  <uses-sdk android:minSdkVersion="19" android:targetSdkVersion="22" />
  <category android:name="com.google.intent.category.CARDBOARD" />
Add assests directory include one picture, Andes.jpg
```

Native Cardboard App With Android Studio

First Step To Android Gvr Example

pltform

- Android Studio-2.3 beta1
- Gradle-3.2 [gradle-wrapper.properties] with build tool 25.0.2 [build.gradle (Module:app)]

Step 1. New Project Created

Open Android Studio and create a new project with empty content,

```
Project: gvrviewdemo
structure: com.cgu.gvrviewdemo
```

Step 2. Add Gvr Android SDK modules

Add the gvr android SDK as usual and check, [build.gradle (Module:app)], confirm the following setting changed:

```
...
dependencies {
    ...
    compile 'com.google.vr:sdk-base:1.10.0'
    compile 'com.google.vr:sdk-common:1.10.0'
}
```

Step 3. Modify AndroidManifest.xml

add staff for gvr android SDK:

```
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

```
    ...
    <activity android:name=".MainActivity" >
            <intent-filter>
                ...
                <category android:name="com.google.intent.category.CARDBOARD" />
                ...
            </intent-filter>
        </activity>
```

Step 4. Create Cardboard layout in [activity_main.xml]:

Delete all and add the following content:

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android_
(http://schemas.android.com/apk/res/android)"
    xmlns:tools="http://schemas.android.com/tools_
(http://schemas.android.com/tools)"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.vr.sdk.base.GvrView
        android:id="@+id/gvr_view"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true" />

</RelativeLayout>
```

Step 5. Main Java Code

```
package gvrviewdemo.cgu.com.gvrviewdemo;

import android.os.Bundle;
// modules for gvr Android SDK
import com.google.vr.sdk.base.Eye;
import com.google.vr.sdk.base.GvrActivity;
import com.google.vr.sdk.base.GvrView;
import com.google.vr.sdk.base.HeadTransform;
import com.google.vr.sdk.base.Viewport;

import javax.microedition.khronos.egl.EGLConfig;
```

```
//  change the native Android Activity to GvrActivity
public class MainActivity extends GvrActivity implements GvrView.StereoRenderer
{
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);


       GvrView gvrView = (GvrView) findViewById(R.id.gvr_view);
       gvrView.setRenderer(this);
       setGvrView(gvrView);
   }

   // following generated by the on-line assistant

   @Override
   public void onNewFrame(HeadTransform headTransform) {

   }

   @Override
   public void onDrawEye(Eye eye) {

   }

   @Override
   public void onFinishFrame(Viewport viewport) {

   }

   @Override
   public void onSurfaceChanged(int i, int i1) {

   }

   @Override
   public void onSurfaceCreated(EGLConfig eglConfig) {

   }

   @Override
   public void onRendererShutdown() {

   }

}
```

Note

Note

Sometimes, there are warnings by Android system which can solved by recompiling the project by

- [Build] -> [Clean Project] -> [Rebuild Project]
- [File] -> [Invalidate Caches and Restart...]

Simple Panorama project

Prepare

- Gvr Android SDK (https://github.com/googlevr/gvr-android-sdk) known as google cardboad SDK, version 1.1.0 up to 2016-12-19; especially, the sub-folder, [library], and [samples/simplepanowidgets];
- a pretty figure.

Step 1. Create New project

Open Android Studio and start a new project,

```
Project: vrViewDemo
structure: com.cgu.vrViewDemo
```

As explained previously, make sure to specify a Minimum SDK no less than API 19 (Android 4.4, KitKat). Choose Empty Activity and name it MainActivity. Click on Finish.

Note

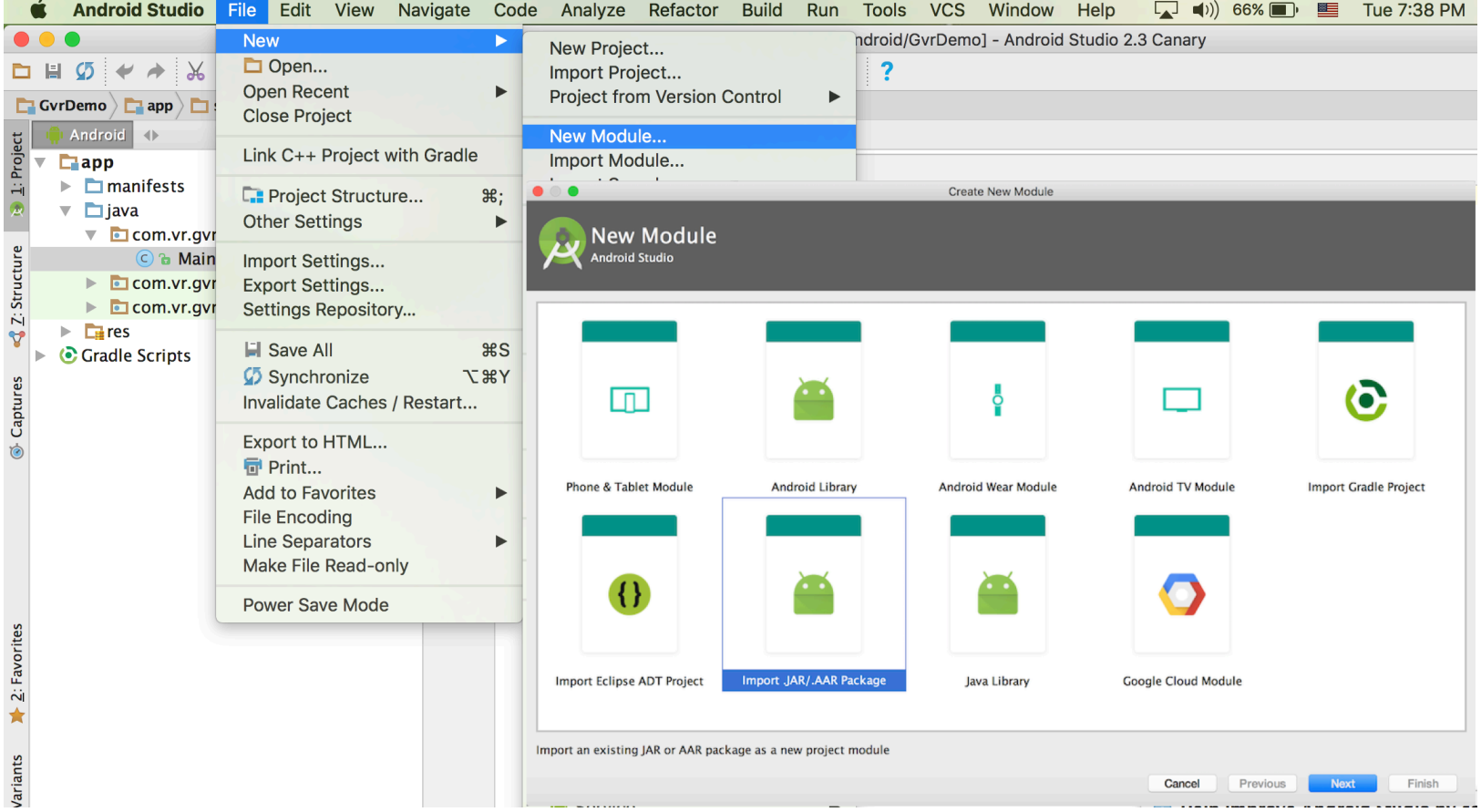
If found the error "Grandle too old", this is caused by the wrong integration og Android studio, general appeared after upgrading the Android studio. If has the problem, open the setting, gradle.build under [app] folder, modify the line within *buildscript* block:

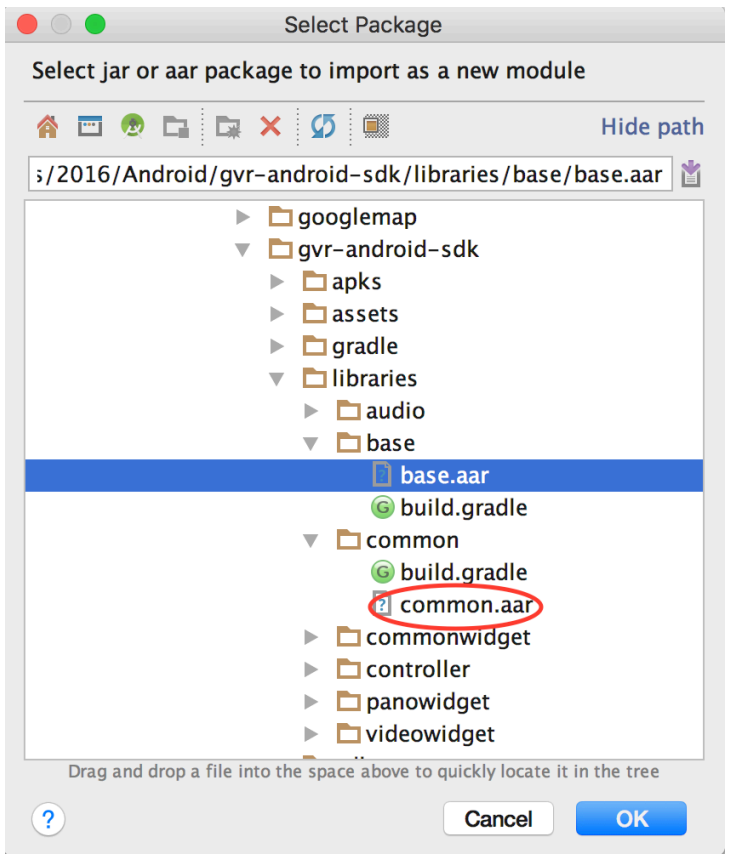
```
buildscript {
    ...
    dependencies {
        classpath 'com.android.tools.build:gradle:2.2.3'
    ...
}
```

bascially, change the version of gradle plugin, i.e. 2.2.3 for instance.

Step 2. Add GVR-Android-SDK

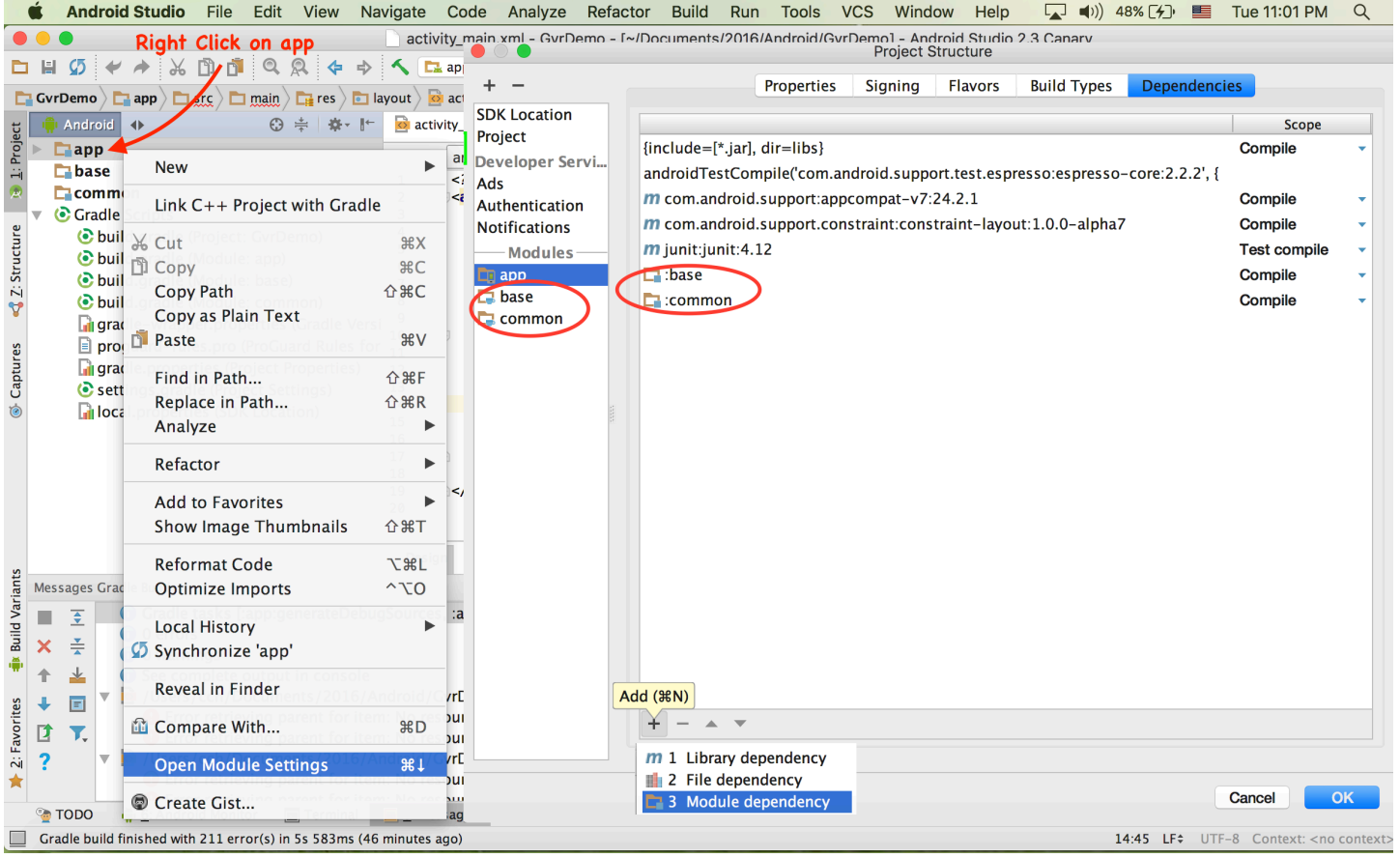
Add the .aar GVR SDK library les to your project. For each, the base.aar and common.aar libraries perform the following steps:





1. Navigate to [File | New | New Module...]
2. Select Import .JAR/.AAR Package, click on **[Next]**.
3. Use the file browser (... or press the Shift and Enter keys) to find and select the base.aar file. Then click on **[OK]**.
4. Click on **[Finish]**.
5. Repeat these steps for the base,arr, common.arr, commonwidget.arr, panowidget.arr, modules,.
6. also creat sub-folder, assets, as [app/src/main/assets], under project folder to store the backgroud figure, for example , 2012-1.jpg.

Next, we'll add the new modules as dependencies to our main app:



1. [Right-click] on your newly created application, **app**, in the project view and choose [Open Module Settings] or press [F4].
2. Click the [Dependencies] tab on the top right, click the [+]-button (on the left buttom bar) and select [Module Dependency] to add the modules, base and common, we just imported.
3. After adding the modules, we can find that *build.gradle (Module:app)* had been modified as follows:

```
...
dependencies {
  ...
  compile project(':gvr-android-sdk/libraries:common')
  compile project(':gvr-android-sdk/libraries:commonwidget')
  compile project(':gvr-android-sdk/libraries:panowidget')
}
```

Step 3. AndroidManifest

1. the permissions needed are as follows: INTERNET, [READ/WRITE] External storage reading and vibration.
2. minimum SDK version needs to be 19, 4.4 KitKat,
3. - the device must support OpenGL ES 2.0 to run the sample app*,
4. - Make accelerometer and gyroscope hard requirements for good head tracking*,
5. - activity's required screen orientation is "landscape",*
6. The intent-filter and specifically com.google.intent.category.CARDBOARD state that this activity is compatible with VR viewers.

```
<?xml version="1.0" encoding="utf-8"?>
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.cgu.vrViewDemo">
    <uses-sdk android:minSdkVersion="19" android:targetSdkVersion="22" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORA
GE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="com.google.intent.category.CARDBO
ARD" />
                <category android:name="android.intent.category.LAUNCHER"
/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Step4. Layout The activity_main.xml

All user interface elements in an Android app are built using views. The Google VR SDK for Android provides its own view, GvrView, which can be used for VR rendering. GvrView renders content in stereo, (make sure "base" module added); here, we use "com.google.vr.sdk.widgets.pano.VrPanoramaView":

```
<?xml version="1.0" encoding="utf-8"?>
   <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/andr
oid"
       xmlns:tools="http://schemas.android.com/tools"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       tools:context=".MainActivity">

       <com.google.vr.sdk.widgets.pano.VrPanoramaView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/vr_view"
        android:scrollbars="none"/>
   </RelativeLayout>
```

Step 5. Main Code,

there are two java files we use here:

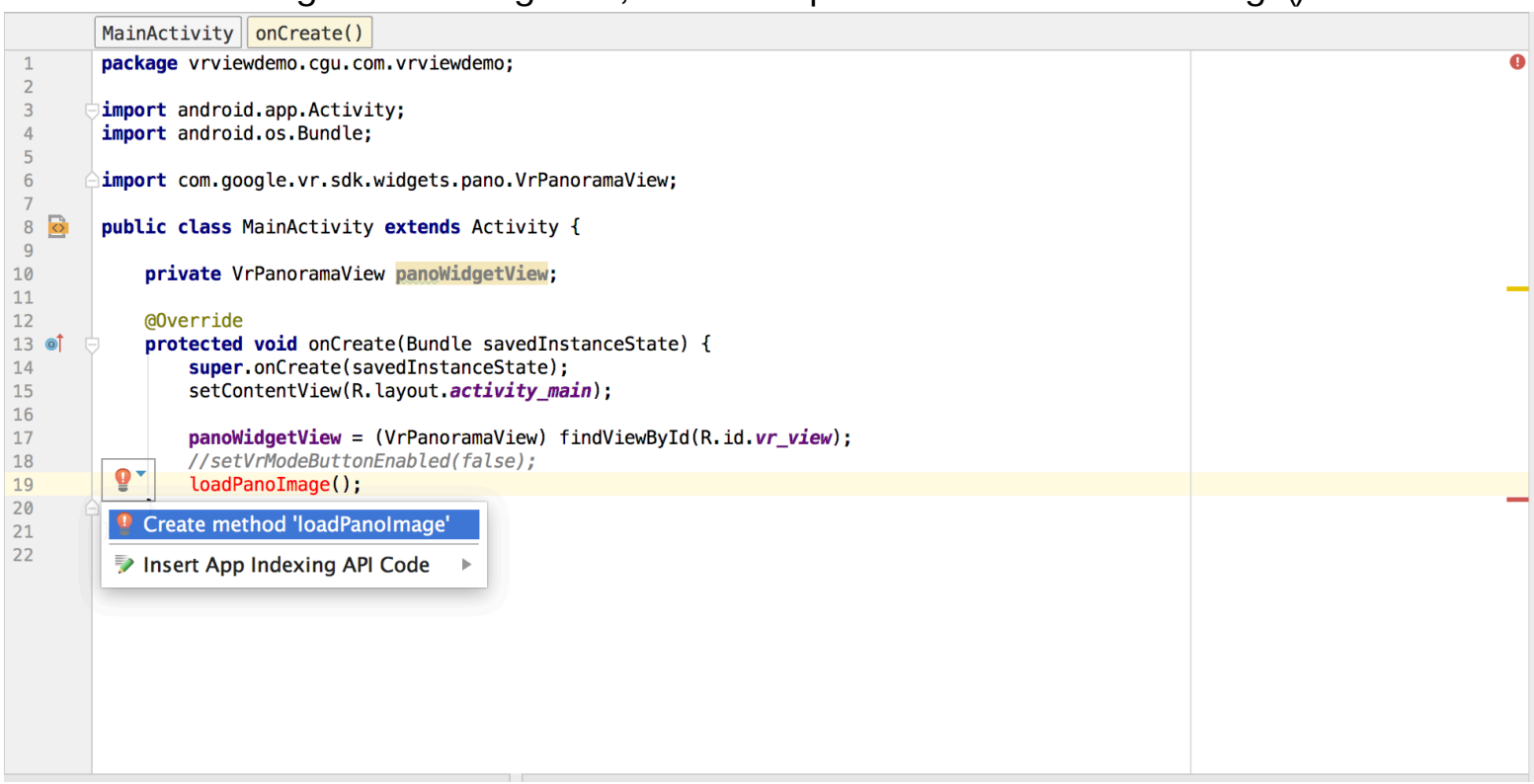
- MainActivity.java
- import VrPanoramaView to make the View of app:

```
import com.google.vr.sdk.widgets.pano.VrPanoramaView;

public class MainActivity extends Activity {

    private VrPanoramaView panoWidgetView;
    // class defned later
    private ImageLoaderTask backgroundImageLoaderTask;
    ...
```

- edit onCreate() by adding the GvrView instance, as follows:

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
     setContentView(R.layout.activity_main);

     panoWidgetView = (VrPanoramaView) findViewById(R.id.vr_view);
        //setVrModeButtonEnabled(false);
        loadPanoImage();
}
...
```

Android Studio should give a warning error, and as helper to define loadPanoImage().



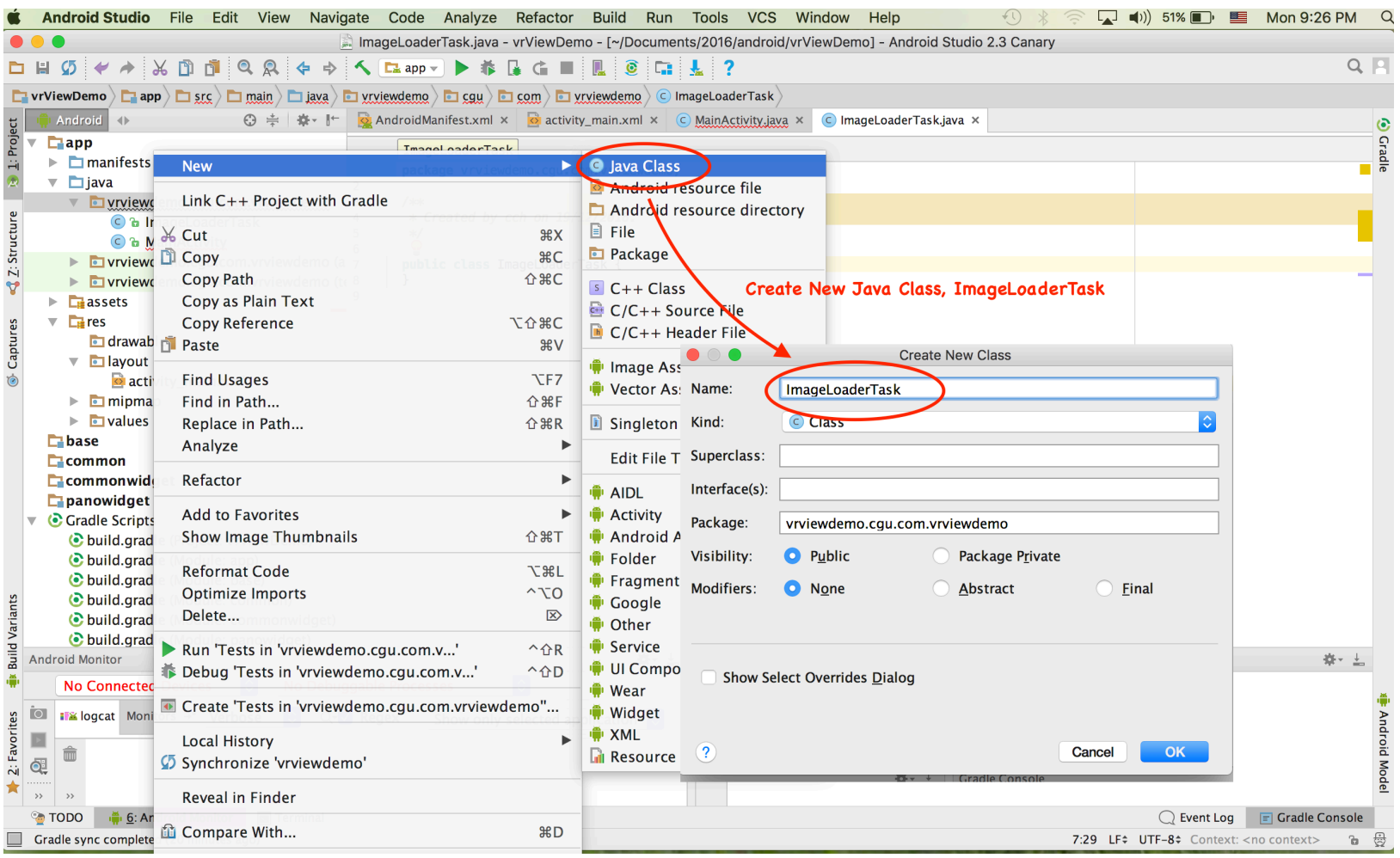
The complete code is:

```
package com.cgu.vrViewDemo;

import android.app.Activity;
import android.os.Bundle;

import com.google.vr.sdk.widgets.pano.VrPanoramaView;

public class MainActivity extends Activity {

    private VrPanoramaView panoWidgetView;
    private ImageLoaderTask backgroundImageLoaderTask;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        panoWidgetView = (VrPanoramaView) findViewById(R.id.vr_view);

        //setVrModeButtonEnabled(false);
        loadPanoImage();
    }

    @Override
    public void onPause() {
        panoWidgetView.pauseRendering();
        super.onPause();
    }

    @Override
    public void onResume() {
        panoWidgetView.resumeRendering();
        super.onResume();
    }

    @Override
    public void onDestroy() {
        // Destroy the widget and free memory.
        panoWidgetView.shutdown();
        super.onDestroy();
    }

    private synchronized void loadPanoImage() {
        ImageLoaderTask task = backgroundImageLoaderTask;
        if (task != null && !task.isCancelled()) {
            // Cancel any task from a previous loading.
            task.cancel(true);
        }
```

```
        // pass in the name of the image to load from assets.
        VrPanoramaView.Options viewOptions = new VrPanoramaView.Options()
;
        viewOptions.inputType = VrPanoramaView.Options.TYPE_MONO;
        //viewOptions.inputType = VrPanoramaView.Options.TYPE_STEREO_OVER
_UNDER;
        // use the name of the image in the assets/ directory.
        String panoImageName = "2012-1.jpg";

        // create the task passing the widget view and call execute to st
art.
        task = new ImageLoaderTask(panoWidgetView, viewOptions, panoImage
Name);
        task.execute(getAssets());
        backgroundImageLoaderTask = task;
    }
}
```

- ImageLoaderTask.java, for handling the image manipulation:

Create a new class as follows:



complete code is as follows:

```
package com.cgu.vrViewDemo;

import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.AsyncTask;
import android.util.Log;
```

```
import com.google.vr.sdk.widgets.pano.VrPanoramaView;

import java.io.IOException;
import java.io.InputStream;
import java.lang.ref.WeakReference;


public class ImageLoaderTask extends AsyncTask<AssetManager, Void, Bitmap
> {
    private static final String TAG = "ImageLoaderTask";
    private final String assetName;
    private final WeakReference<VrPanoramaView> viewReference;
    private final VrPanoramaView.Options viewOptions;
    private static WeakReference<Bitmap> lastBitmap = new WeakReference<>
(null);
    private static String lastName;

    @Override
    protected Bitmap doInBackground(AssetManager... params) {
        AssetManager assetManager = params[0];

        if (assetName.equals(lastName) && lastBitmap.get() != null) {
            return lastBitmap.get();
        }

        try(InputStream istr = assetManager.open(assetName)) {
            Bitmap b = BitmapFactory.decodeStream(istr);
            lastBitmap = new WeakReference<>(b);
            lastName = assetName;
            return b;
        } catch (IOException e) {
            Log.e(TAG, "Could not decode default bitmap: " + e);
            return null;
        }
    }

    @Override
    protected void onPostExecute(Bitmap bitmap) {
        final VrPanoramaView vw = viewReference.get();
        if (vw != null && bitmap != null) {
            vw.loadImageFromBitmap(bitmap, viewOptions);
        }
    }

    public ImageLoaderTask(VrPanoramaView view, VrPanoramaView.Options vi
ewOptions, String assetName) {
        viewReference = new WeakReference<>(view);
```

```
            this.viewOptions = viewOptions;
            this.assetName = assetName;
        }
    }
```

Step 6. Build and Test

As usual ...: Intel-based AVM can not run the apk; use arm-based avm

In [ ]:

OpenCV For Android

```
        OpenCV, vision-aware,
               +                          ---> Sense for World
      Android, intelligent applications
```

Requirements

- OpenCV, OpenCV4Android SDK (http://sourceforge.net/projects/opencvlibrary/files/opencv-android)
- Scenarios
  - With Android SDK, use manually;
  - Without Android SDK, use Tegra Android Development Pack (TADP (https://developer.nvidia.com/tegra-android-development-pack));
- Native Development Kit (NDK), install it by Android Studio.

First OpenCVForAndroid project

1. Launch Android Studio.
2. Create a new Android Studio project and name it HelloOpenCVWorld with Company Domain set to cv0.app.com.
3. Choose MinimumSDK. To build with OpenCV4AndroidSDK, the MinimumSDK version is 11.
4. Create a blank activity and name it `HelloVisionActivity`.
5. To add OpenCV as a dependency to the project, navigate to `[File|New|Import Module]` and `<OpenCV4Android_Directoy>\sdk\java`. Then, click `OK`.
6. [Right-click] on your newly created application in the project view and choose [Open Module Settings] or press [F4].
7. In the [Dependencies tab],press the [+]-button and select [Module Dependency].
8. Choose the [OpenCV library] and press [Add]. Now, you should be able to import the `OpenCV classes` to your project.

In [ ]:

```
%jupyter nbconvert GVR-Android-2016-12-19.ipynb
```