FIVE ACVU

Android Computer Vision Unity

ABSTRACT
Software development document for Unity Android Applications using OpenCV.

UT Dallas

Contents

1 Overview

FIVE ACVU (Android Computer Vision Unity) is an approach for developing Unity applications for android devices using OpenCV. The Unity application can be modeled for Gear VR development. The present version of FIVE ACVU integrates all the components into one Android application package (APK).

The integration of the above mentioned tools and library is based on the following approach.

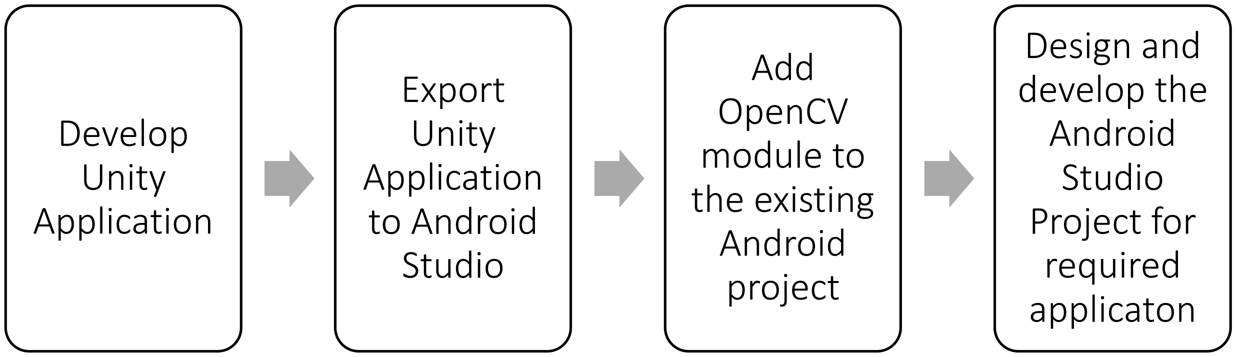

Develop Unity Application
Export Unity Application to Android Studio
Add OpenCV module to the existing Android project
Design and develop the Android Studio Project for required applicaton

Figure 1

As seen in Figure 1 we start by developing an application in Unity. The Unity application along with required Gameobjects and scripts implements a pre-defined set of functionalities to fetch required data from android modules. The application is then exported to an android project. The generated project is imported into Android Studio. In this project OpenCV for Android is added as a library. Finally, the Android Studio project is designed and developed as per the required features like sensors, services and image processing modules, OpenCV NDK modules(if any).

The current version of FIVE ACVU follows this approach rather than developing an android plugin as this gives more flexibility for adding android modules.

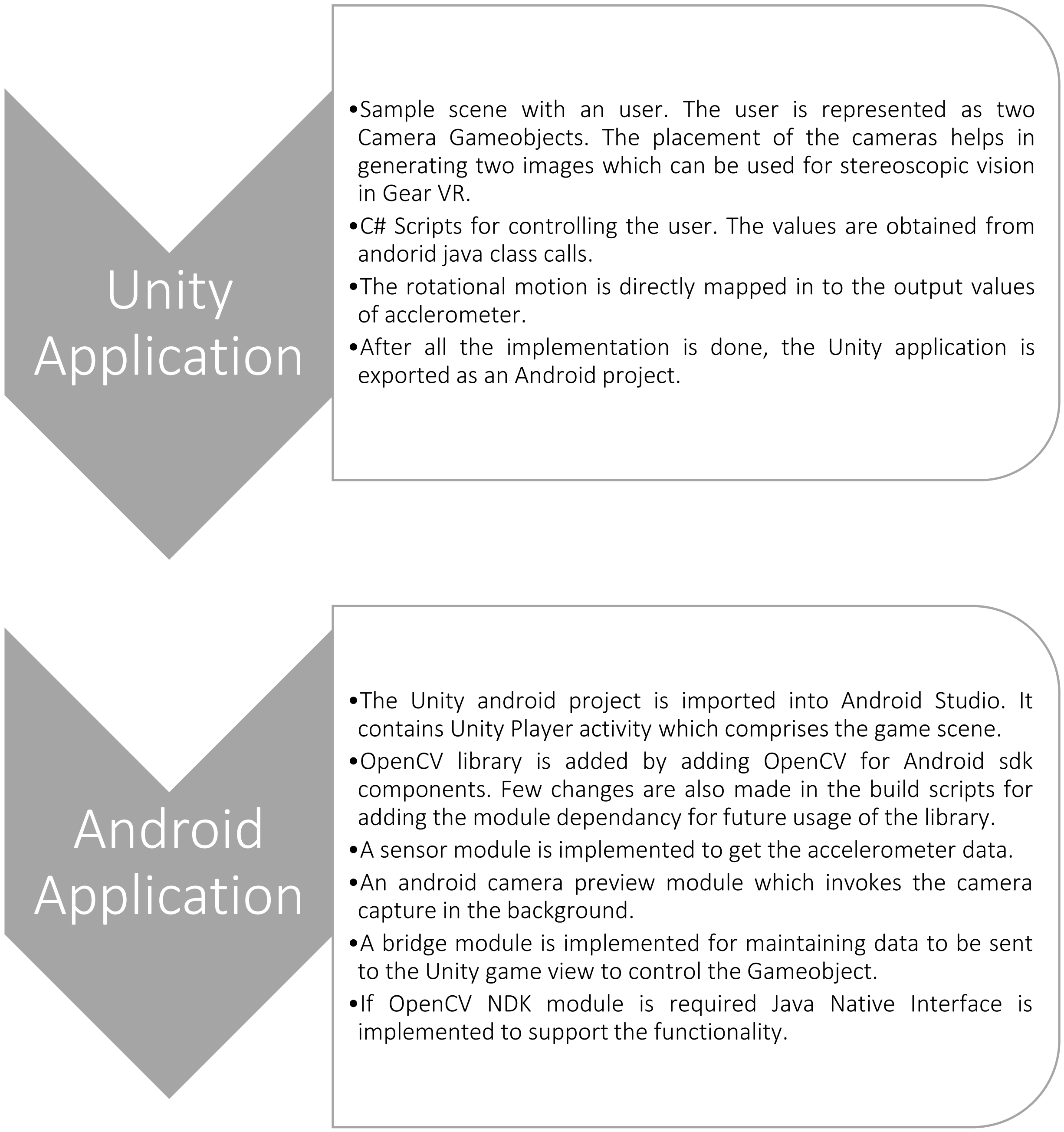
2 Purpose of the Document

The purpose of this document is to describe the steps for developing a sample android applications which has a game scene developed in Unity and takes input from the modules like android sensors (Accelerometer in this case) to control Gameobjects in the scene. The Unity game scene adds an invisible surface texture to the leaf view of Unity. It also invokes the camera module for capturing frames in the background.

3 Scope of the Document

This document presents one approach for developing application involving Unity, Android and OpenCV. Since there are various methods in which the integration of the above mentioned tools are possible, this document states one which is suitable for specific project requirements. The sample project presented in this document can be modeled for Gear VR development. Since this is work in progress, in the future versions there will be various design changes for the integration procedure as well as added functionalities in the sample application.

4 Sample Project Details



Unity Application

- Sample scene with an user. The user is represented as two Camera Gameobjects. The placement of the cameras helps in generating two images which can be used for stereoscopic vision in Gear VR.
- C# Scripts for controlling the user. The values are obtained from andorid java class calls.
- The rotational motion is directly mapped in to the output values of acclerometer.
- After all the implementation is done, the Unity application is exported as an Android project.

Android Application

- The Unity android project is imported into Android Studio. It contains Unity Player activity which comprises the game scene.
- OpenCV library is added by adding OpenCV for Android sdk components. Few changes are also made in the build scripts for adding the module dependancy for future usage of the library.
- A sensor module is implemented to get the accelerometer data.
- An android camera preview module which invokes the camera capture in the background.
- A bridge module is implemented for maintaining data to be sent to the Unity game view to control the Gameobject.
- If OpenCV NDK module is required Java Native Interface is implemented to support the functionality.

Figure 2

5 Required Components

1. Unity3D environment. It can be downloaded from http://unity3d.com/get-unity/download/archive . The sample Unity project is built in Unity 4.5.2. But any later version can also be used. Follow the installation instruction in the site to install Unity.
2. Android Studio IDE. It can be downloaded from http://developer.android.com/sdk/index.html. Follow the installation instruction in the web to install Android Studio. This requires Java Development Kit, if not pre-installed it can be downloaded from http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html and installed using instructions in the web. Also make changes to the System environment variables (Windows).
3. OpenCV Android SDK. It can be downloaded from http://sourceforge.net/projects/opencvlibrary/files/opencv-android/2.4.11/ . The sample project uses OpenCV 2.4.11 SDK for development. The procedure for adding SDK to android project will be explained in this document.
4. Android NDK. This will be required for developing OpenCV NDK modules. The android NDK can be downloaded from https://developer.android.com/tools/sdk/ndk/index.html . The installation procedure will be explained in this document.

6 Development Procedure

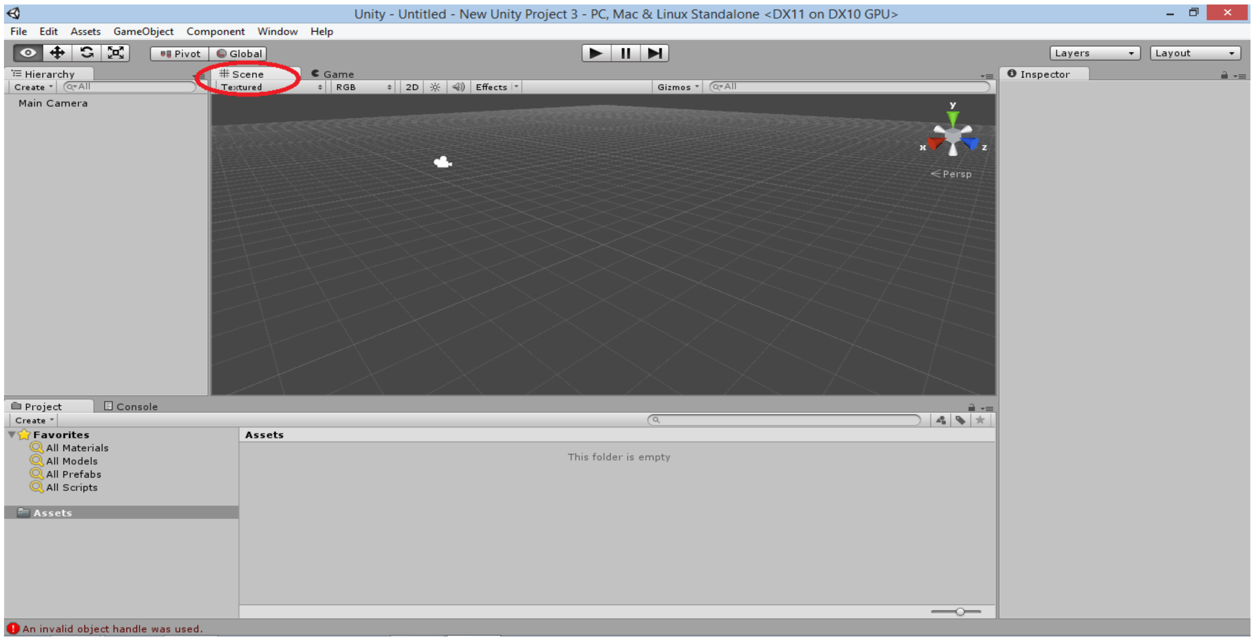
6.1 Unity Application

The Unity application will comprise the game scene. In the sample project, it just consists of a cube. But the scene can be designed as per the application requirement. The user in the sample project is represented by two camera Gameobjects renamed as left and right eye. They are placed few millimeters apart to represent inter pupillary distance (IPD). When developing stereoscopic application for Gear VR this will be taken care by the OVR plugin. The current setup is just for displaying two perspectives. The user's rotation is currently controlled directly by accelerometer values which is fetched by an android java call. This is also possible to handle inside Unity (Not mentioned in this sample project). These will be controlled by the computer vision modules in the future versions.

After the applications is created it is exported to an android studio project. Additional modules are added in the android project.

The step for creating and exporting the Unity application as android project are as follows.

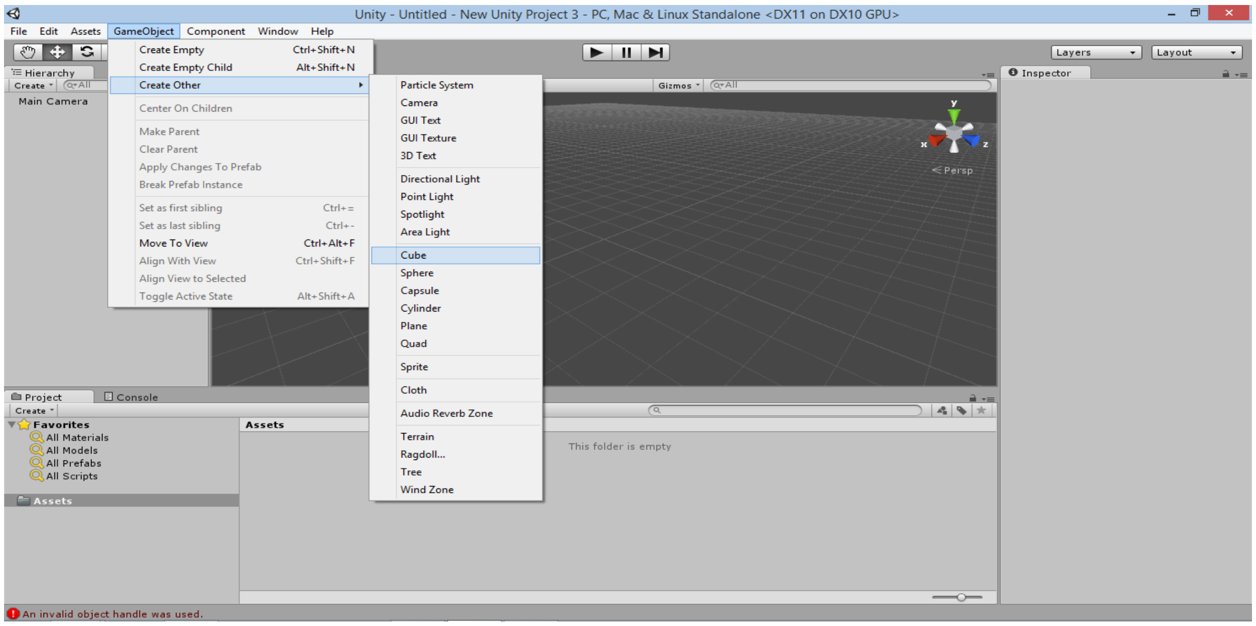
1. Open a new Unity project.
2. Once in the Unity environment select the scene view (in case not selected as default).



An invalid object handle was used.

Figure 3

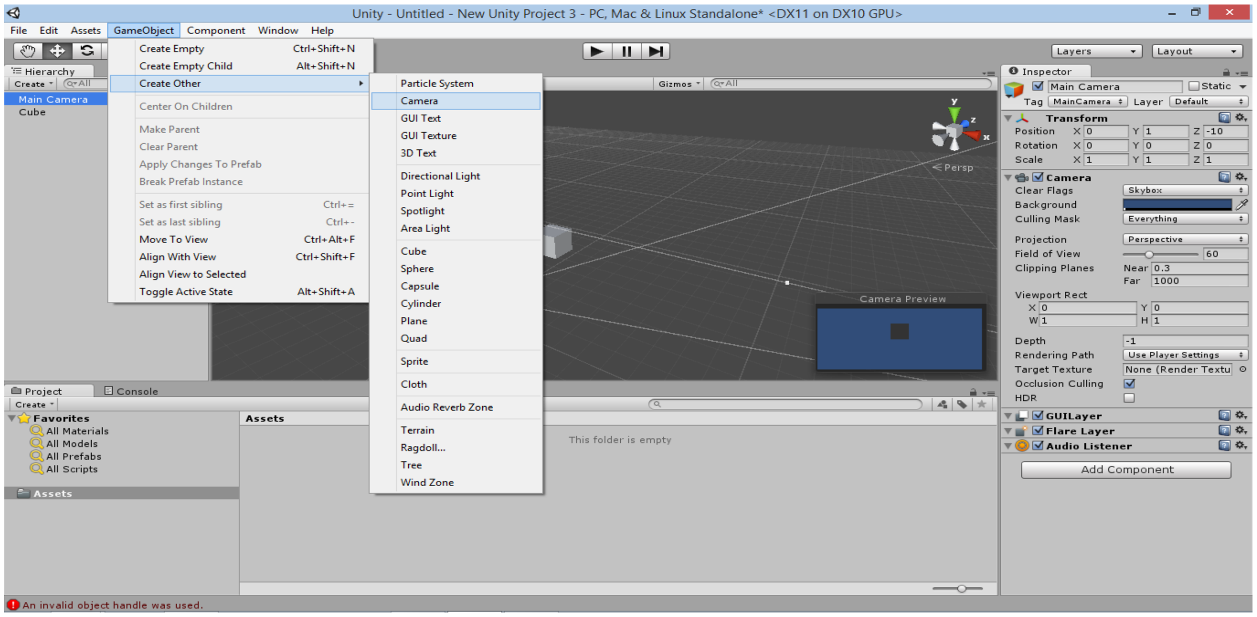
3. Add a Cube to the scene by clicking on GameObject->Create Other -> Cube. Resize the cube as desired. It can be resize using the scale in the inspector window or by pressing “r” on the keyboard and dragging the white square on the origin of cube.



An invalid object handle was used.

Figure 4

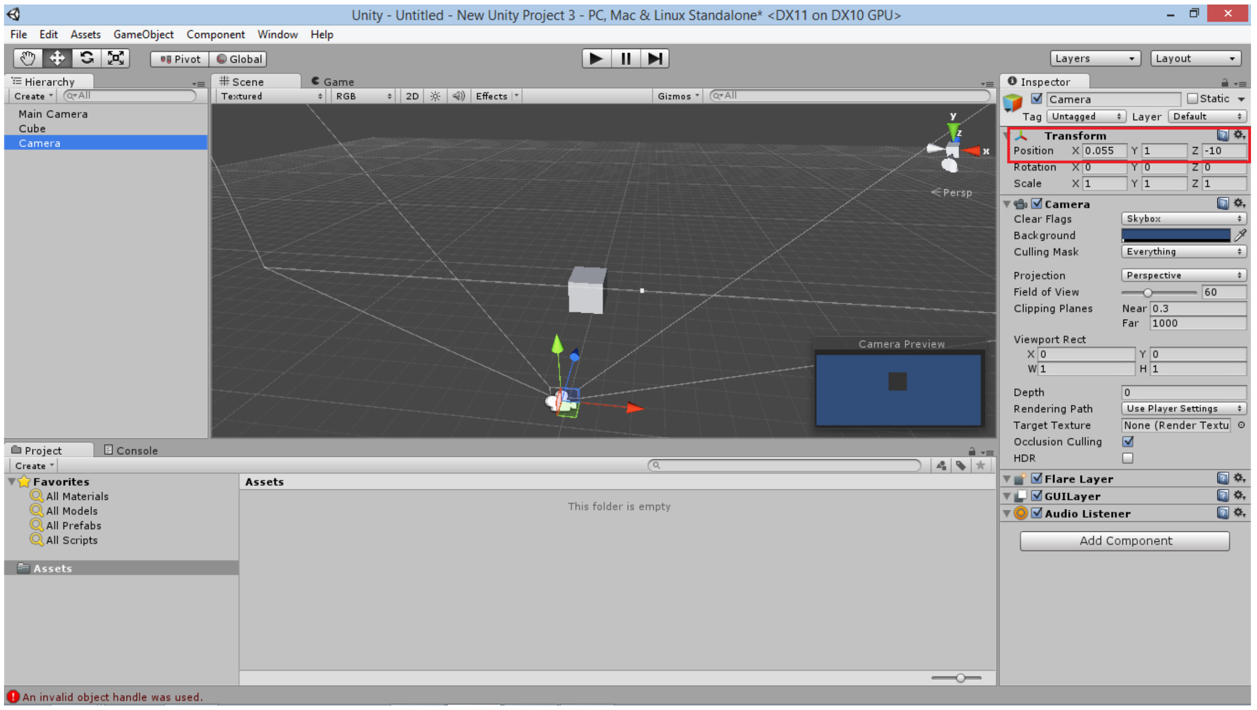
4. Place the cube such that it is in the field of view of the camera. The cube can be rotated or moved by pressing "e or w" on the keyboard and moving/dragging the arrow in the desired direction. The same can be achieved by changing the rotation or position in the inspector.
5. Create another camera Gameobject by clicking on Gameobject-> Create Other -> Camera.



Tag MainCamera Layer Default
Transform: Position X 0 Y 1 Z -10; Rotation X 0 Y 0 Z 0; Scale X 1 Y 1 Z 1
Camera: Clear Flags Skybox; Culling Mask Everything; Projection Perspective; Field of View 60; Clipping Planes Near 0.3 Far 1000; Viewport Rect X 0 Y 0 W 1 H 1; Depth -1; Rendering Path Use Player Settings; Target Texture None (Render Textu
An invalid object handle was used.

Figure 5

6. Place the camera 0.055m horizontally apart. As mentioned earlier this IPD is used just as an example. It will be handled by the OVR plugin while developing for Gear VR application.


Unity - Untitled - New Unity Project 3 - PC, Mac & Linux Standalone* <DX11 on DX10 GPU>
File Edit Assets GameObject Component Window Help
Pivot
Global
Hierarchy
Main Camera
Cube
Camera
Scene
Game
Persp
Camera Preview
Inspector
Add Component
This folder is empty

Camera, Tag Untagged Layer Default
Transform: Position X 0.055 Y 1 Z -10; Rotation X 0 Y 0 Z 0; Scale X 1 Y 1 Z 1
Camera: Clear Flags Skybox; Culling Mask Everything; Projection Perspective; Field of View 60; Clipping Planes Near 0.3 Far 1000; Viewport Rect X 0 Y 0 W 1 H 1; Depth 0; Rendering Path Use Player Settings; Target Texture None (Render Textu
An invalid object handle was used.

Figure 6

7. Rename the cameras as LeftEye and RightEye to depict the images that will be rendered as viewed by these camera on the respective stereoscopic displays.
8. Create an empty Gameobject by clicking on GameObject-> Create Empty and rename it as user.

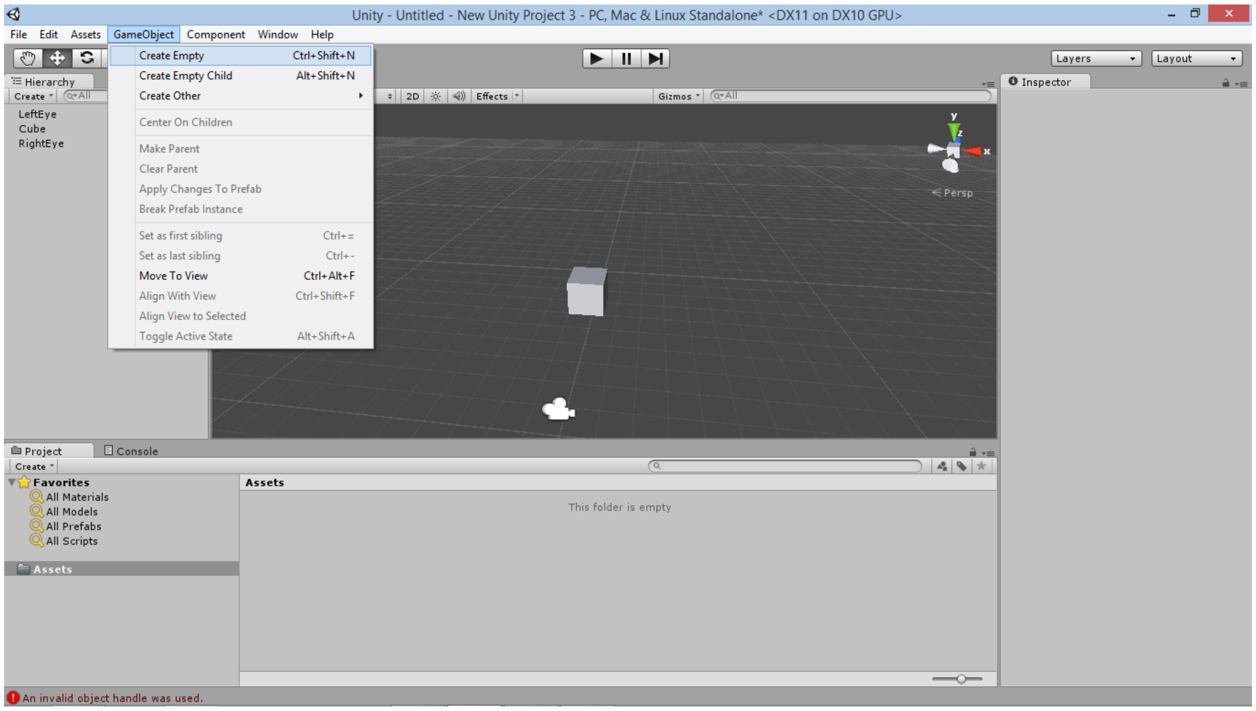
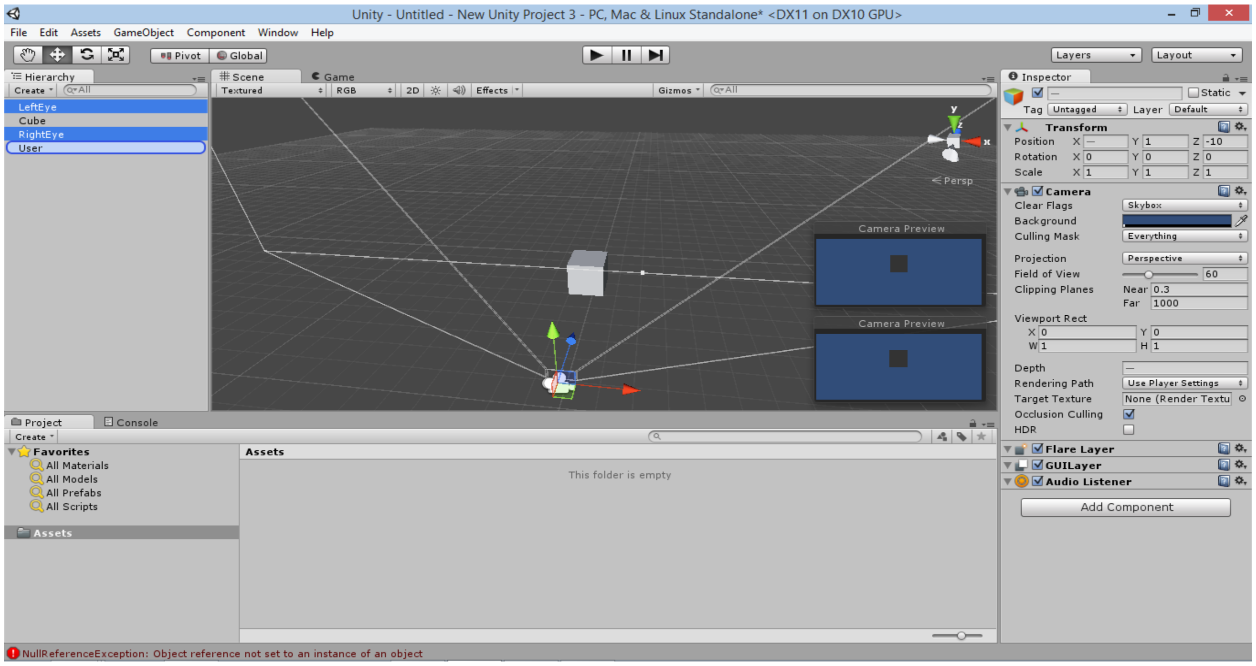

Unity - Untitled - New Unity Project 3 - PC, Mac & Linux Standalone* <DX11 on DX10 GPU>
File Edit Assets GameObject Component Window Help
Create Empty Ctrl+Shift+N
Create Empty Child Alt+Shift+N
Create Other
Center On Children
Make Parent
Clear Parent
Apply Changes To Prefab
Break Prefab Instance
Set as first sibling Ctrl+=
Set as last sibling Ctrl+-
Move To View Ctrl+Alt+F
Align With View Ctrl+Shift+F
Align View to Selected
Toggle Active State Alt+Shift+A
Hierarchy
LeftEye
Cube
RightEye
Inspector
Layers
Layout
Effects
Gizmos
Persp
Project
Console
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
This folder is empty
An invalid object handle was used.

Figure 7

9. Select the RightEye and LeftEye Gameobject and drag it on "User".

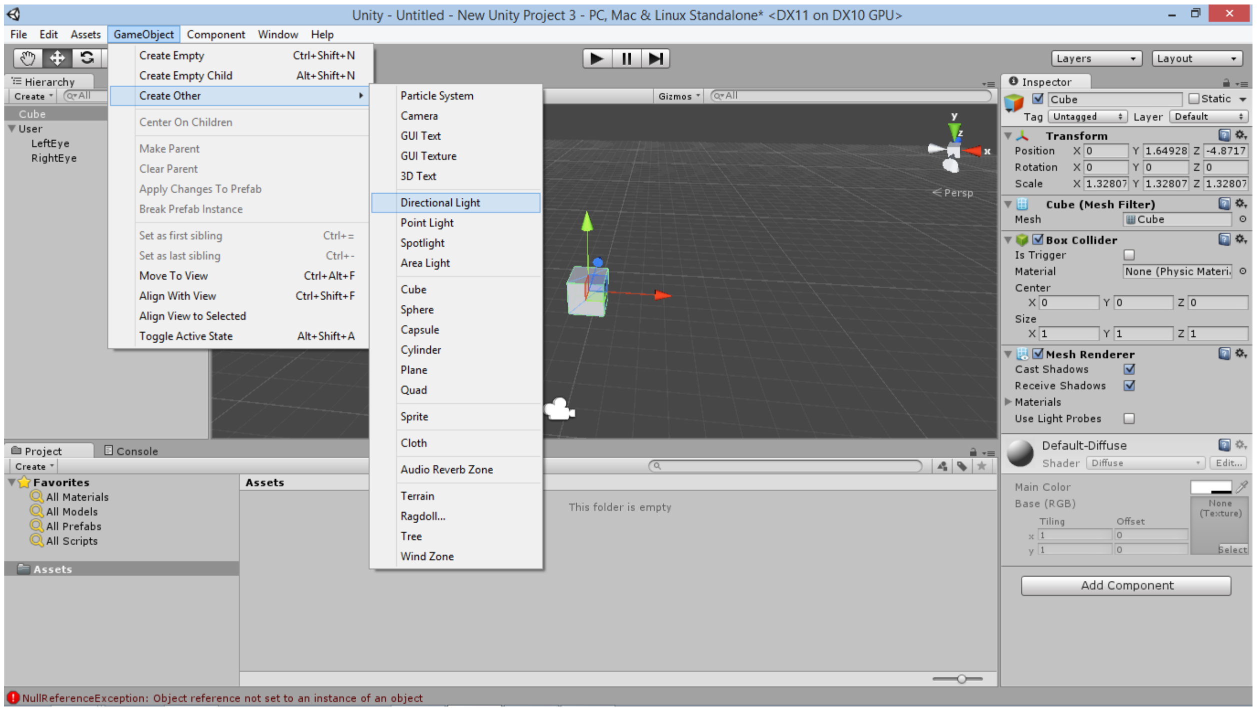


Static
Tag Untagged Layer Default
Transform
Position X — Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Camera
Clear Flags Skybox
Background
Culling Mask Everything
Projection Perspective
Field of View 60
Clipping Planes Near 0.3 Far 1000
Viewport Rect X 0 Y 0 W 1 H 1
Depth —
Rendering Path Use Player Settings
Target Texture None (Render Textu
Occlusion Culling ✓
HDR
Flare Layer
GUILayer
Audio Listener

NullReferenceException: Object reference not set to an instance of an object

Figure 8

10. For better clarity in the scene add a directional light by clicking GameObject -> Create Other -> Directional Light. Adjust the pose as desired.



NullReferenceException: Object reference not set to an instance of an object

Figure 9

11. Change the viewport values as follows in the LeftEye and RightEye GameObject.

LeftEye X = 0 Y = 0 W = 0.5 H = 1 RightEye X = 0.5 Y = 0 W = 0.5 H = 1

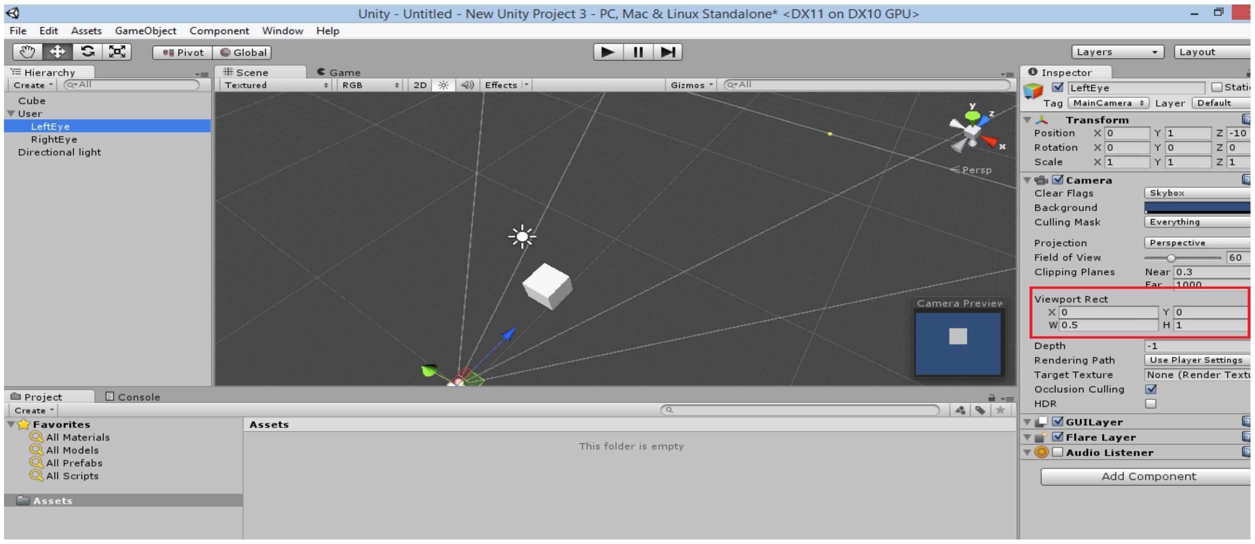
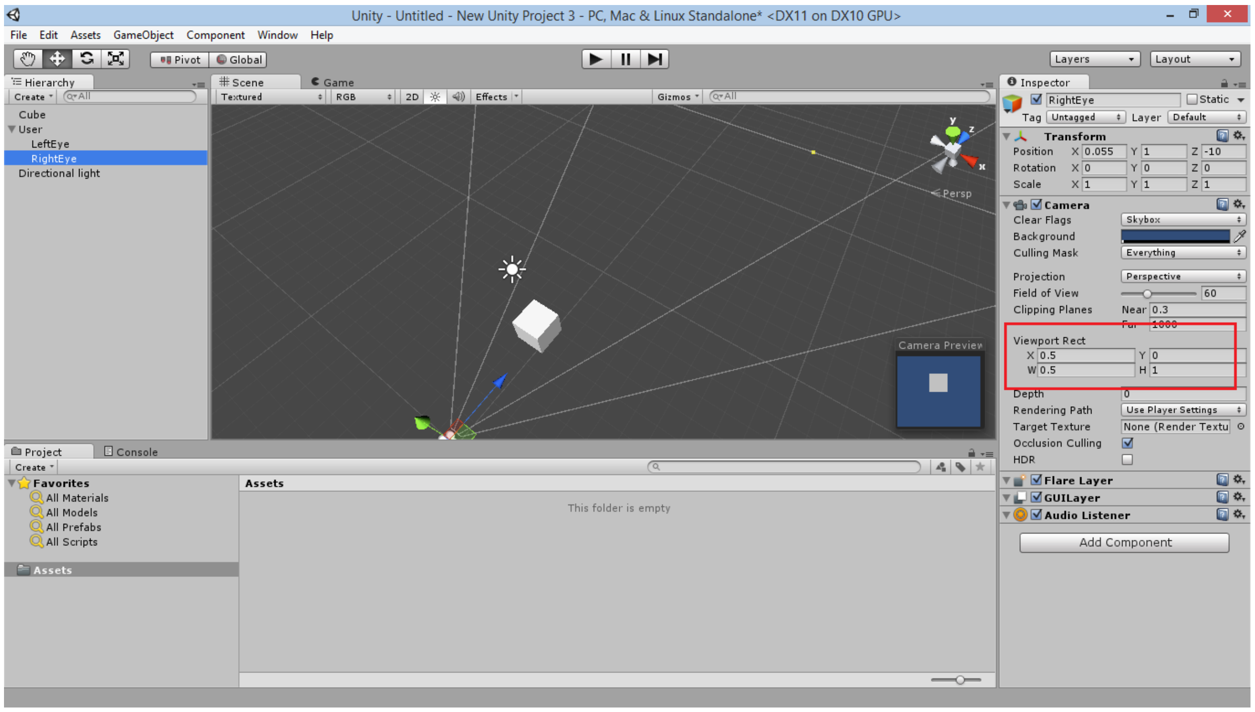


Figure 10



Unity - Untitled - New Unity Project 3 - PC, Mac & Linux Standalone* <DX11 on DX10 GPU>

Hierarchy: Cube; User (LeftEye, RightEye); Directional light

Inspector: RightEye — Tag Untagged — Layer Default

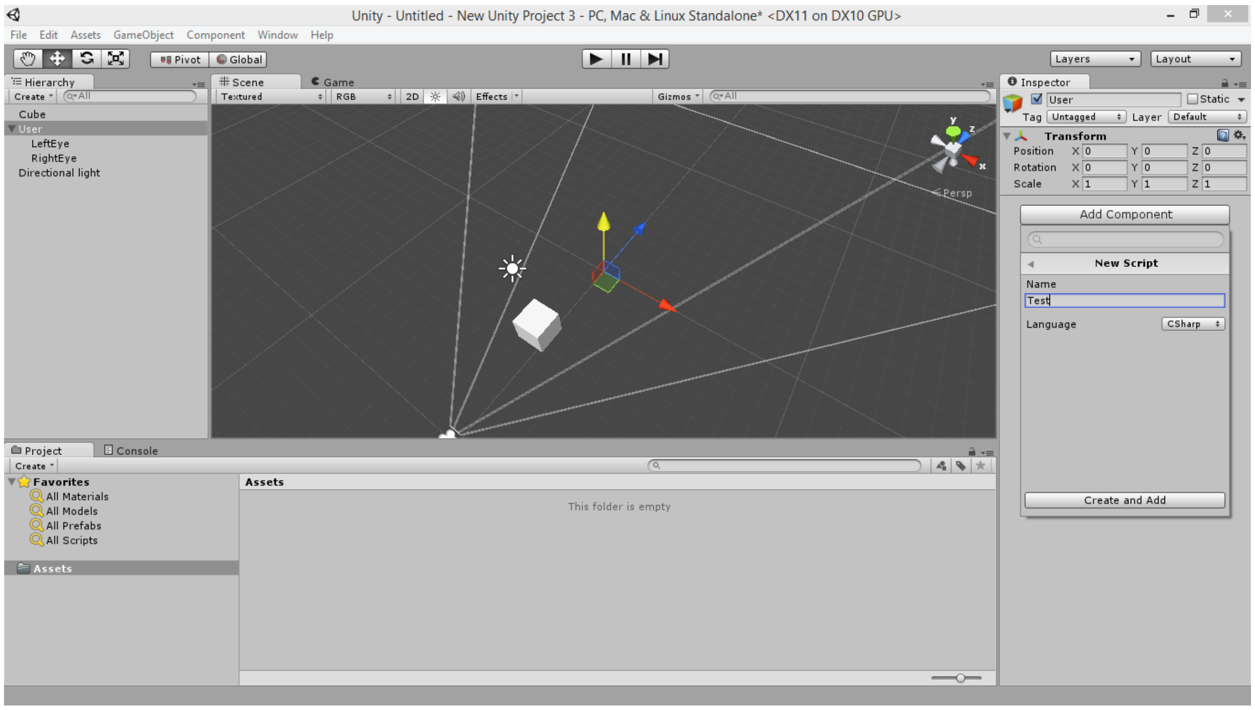
Transform
Position X 0.055 Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags: Skybox
Background
Culling Mask: Everything
Projection: Perspective
Field of View: 60
Clipping Planes: Near 0.3, Far 1000
Viewport Rect: X 0.5 Y 0, W 0.5 H 1
Depth: 0
Rendering Path: Use Player Settings
Target Texture: None (Render Textu
Occlusion Culling: ✓
HDR

Flare Layer
GUILayer
Audio Listener

Figure 11

12. Add a new C# script by selecting the user and clicking on Add Component -> New Script -> Create and Add. Rename the script as Test.



Unity - Untitled - New Unity Project 3 - PC, Mac & Linux Standalone* <DX11 on DX10 GPU>

Hierarchy: Cube; User (LeftEye, RightEye); Directional light

Inspector: User — Tag Untagged — Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

New Script
Name: Test
Language: CSharp

Figure 12

13. Modify the script as follows:

```
using UnityEngine;
using System.Collections;
public class Test : MonoBehaviour {
        //Positional and Rotational Data
        public static float GetRotationX(){
                AndroidJavaClass ajc = new AndroidJavaClass ("com.example.FiveACVU.UnityDataMgr");
                return ajc.CallStatic<float> ("GetRotationX"); }
        public static float GetRotationY(){
                AndroidJavaClass ajc = new AndroidJavaClass ("com.example.FiveACVU.UnityDataMgr");
                return ajc.CallStatic<float> ("GetRotationY"); }
        public static float GetRotationZ(){
                AndroidJavaClass ajc = new AndroidJavaClass ("com.example.FiveACVU.UnityDataMgr");
                return ajc.CallStatic<float> ("GetRotationZ"); }
        public static float GetTransformX(){
                AndroidJavaClass ajc = new AndroidJavaClass ("com.example.FiveACVU.UnityDataMgr");
                return ajc.CallStatic<float> ("GetTransformX"); }
        public static float GetTransformY(){
                AndroidJavaClass ajc = new AndroidJavaClass ("com.example.FiveACVU.UnityDataMgr");
                return ajc.CallStatic<float> ("GetTransformY"); }
        public static float GetTransformZ(){
                AndroidJavaClass ajc = new AndroidJavaClass ("com.example.FiveACVU.UnityDataMgr");
                return ajc.CallStatic<float> ("GetTransformZ"); }

        // Update is called once per frame
        void Update () {
                transform.position += new Vector3(GetTransformX(),GetTransformY(),GetTransformZ());
                transform.rotation =  Quaternion.Euler(GetRotationX(),GetRotationY(),GetRotationZ()); }
}
```

14. Edit the player setting for exporting the project as android project. Click on Edit -> Project Settings -> Player.

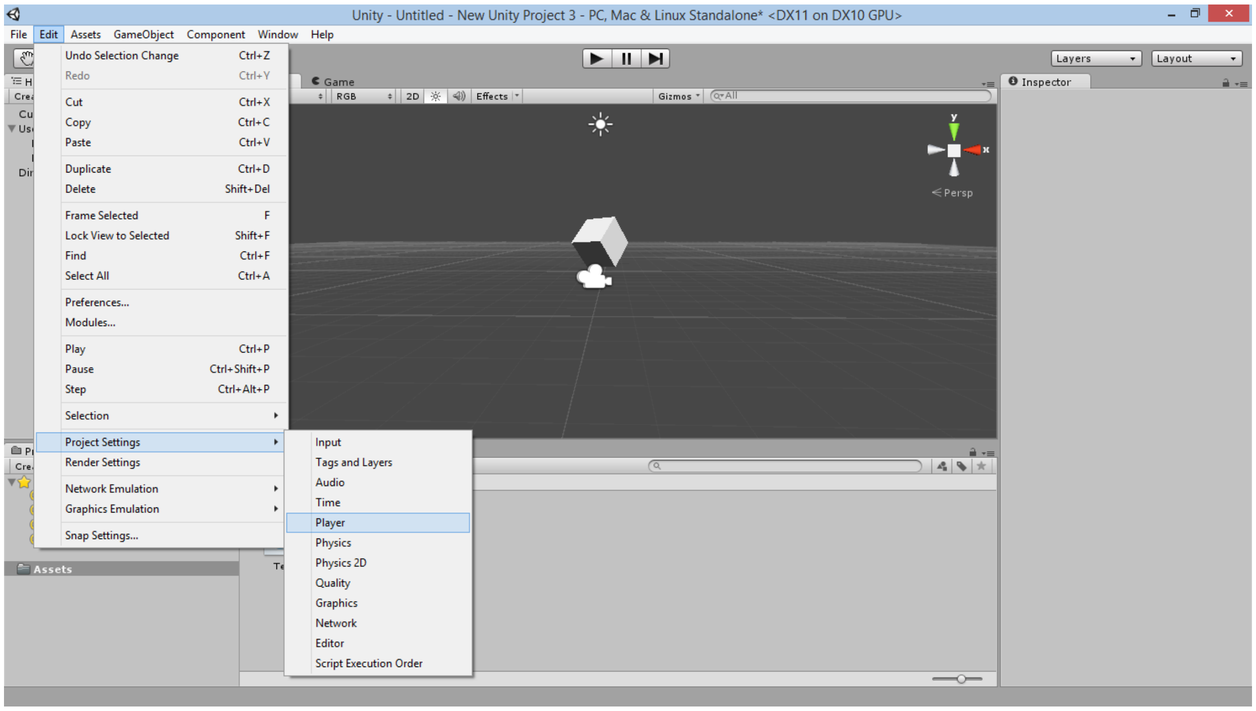
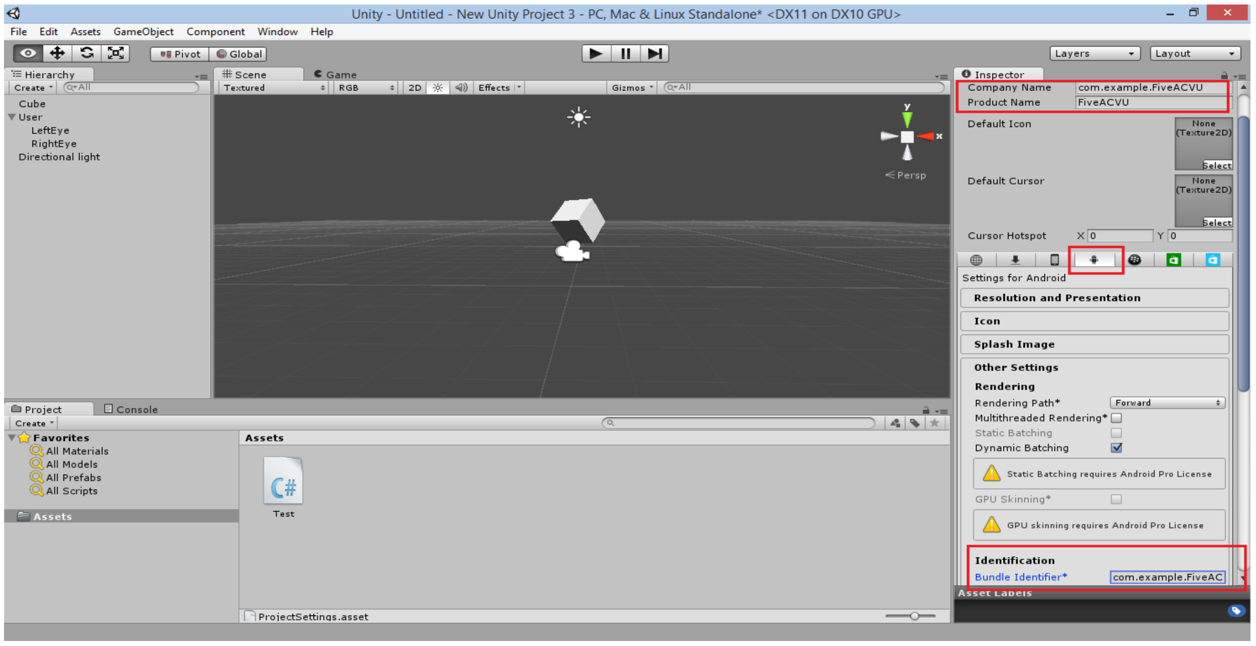


Figure 13

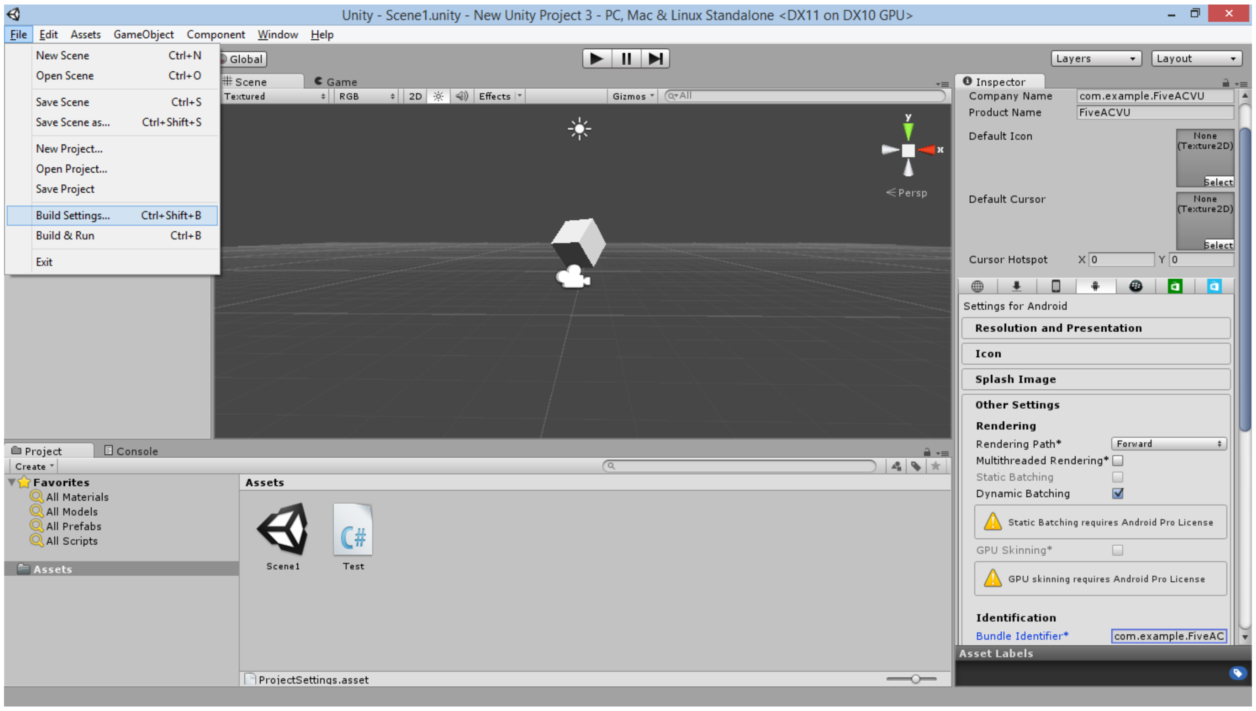
15. Change the Company Name to “com.example.FiveACVU” and the Product name to “FiveACVU”. Select Android symbol in the inspector and change the Bundler Identifier as “com.example.FiveACVU”.



Company Name com.example.FiveACVU
Product Name FiveACVU
Default Icon None (Texture2D) Select
Default Cursor None (Texture2D) Select
Cursor Hotspot X 0 Y 0
Settings for Android
Resolution and Presentation
Icon
Splash Image
Other Settings
Rendering
Rendering Path* Forward
Multithreaded Rendering*
Static Batching
Dynamic Batching ✓
Static Batching requires Android Pro License
GPU Skinning*
GPU skinning requires Android Pro License
Identification
Bundle Identifier* com.example.FiveAC
Asset Labels

Figure 14

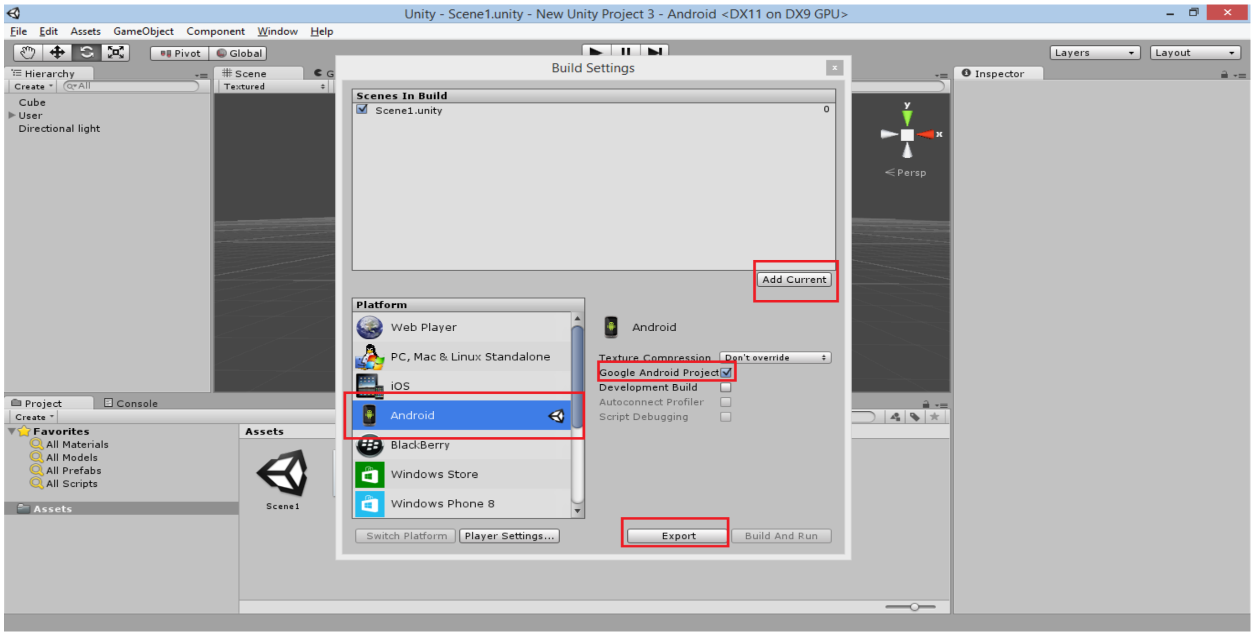
16. Save the Scene with a desired name and Go to File -> Build Settings for exporting the project to android.



Company Name: com.example.FiveACVU
Product Name: FiveACVU
Default Icon: None (Texture2D)
Default Cursor: None (Texture2D)
Cursor Hotspot: X 0 Y 0
Rendering Path*: Forward
Multithreaded Rendering*: ☐
Static Batching: ☐
Dynamic Batching: ✓
GPU Skinning*: ☐
Bundle Identifier*: com.example.FiveAC

Figure 15

17. On the Build Setting Menu Click “Add Current” to add the scene. Select Android from the platform list on the left. Check the “Google Android Project” to enable export of the project. Select Export and select the location to complete the procedure.



Scenes In Build: ✓ Scene1.unity 0

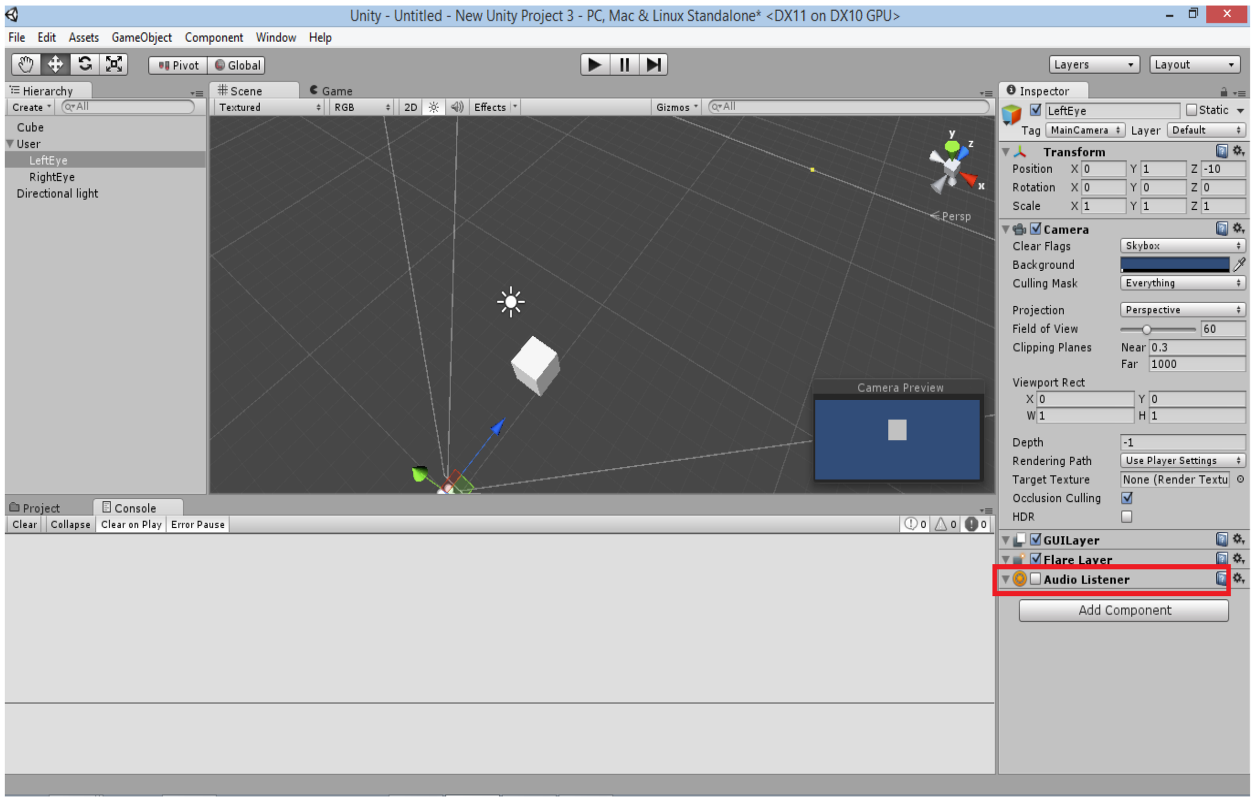
Android
Texture Compression: Don't override
Google Android Project: ✓
Development Build: ☐
Autoconnect Profiler: ☐
Script Debugging: ☐

Figure 16

6.2 Troubleshooting Unity Applications

1. As it is assumed that the tools mentioned in the required components are installed prior to starting the development. The export procedure will prompt one time for the android SDK path. Provide the SDK path which will be similar to this “C:\Users\<User Name>\AppData\Local\Android\sdk”.

2. To remove the warning of multiple audio listener uncheck the audio listener from one of the cameras.



LeftEye Static
Tag MainCamera Layer Default

Transform
Position X 0 Y 1 Z -10
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags: Skybox
Background
Culling Mask: Everything
Projection: Perspective
Field of View: 60
Clipping Planes: Near 0.3, Far 1000
Viewport Rect: X 0, Y 0, W 1, H 1
Depth: -1
Rendering Path: Use Player Settings
Target Texture: None (Render Textu
Occlusion Culling: ✓
HDR

GUILayer
Flare Layer
Audio Listener

Figure 17

3. If the project is run on Windows desktop environment it will throw error. This can be avoided using the preprocessor directives. These help avoiding runtime errors by capsuling code in platform based preprocessor directive.

4. For debugging the application while running on device using Android Studio, print statements could be added.

5. For making application Gear VR compatible the integration steps can be found in https://www.youtube.com/watch?v=0vkRtsaP3co . The steps in the link can be used on the top of the steps mentioned for making the sample application.

6.3 Android Application

The development is started by importing the android project generated in the Unity environment. By default the game scene in Unity will be the parent activity in android. The hierarchy can be changed as desired, but we will stick to the default in the sample project.

The other modules of the sample project app are sensor module which gets the accelerometer data and feeds to the bridge module responsible for sending the data into Unity. The bridge module consists of the functionalities which are called from the Unity C# scripts.

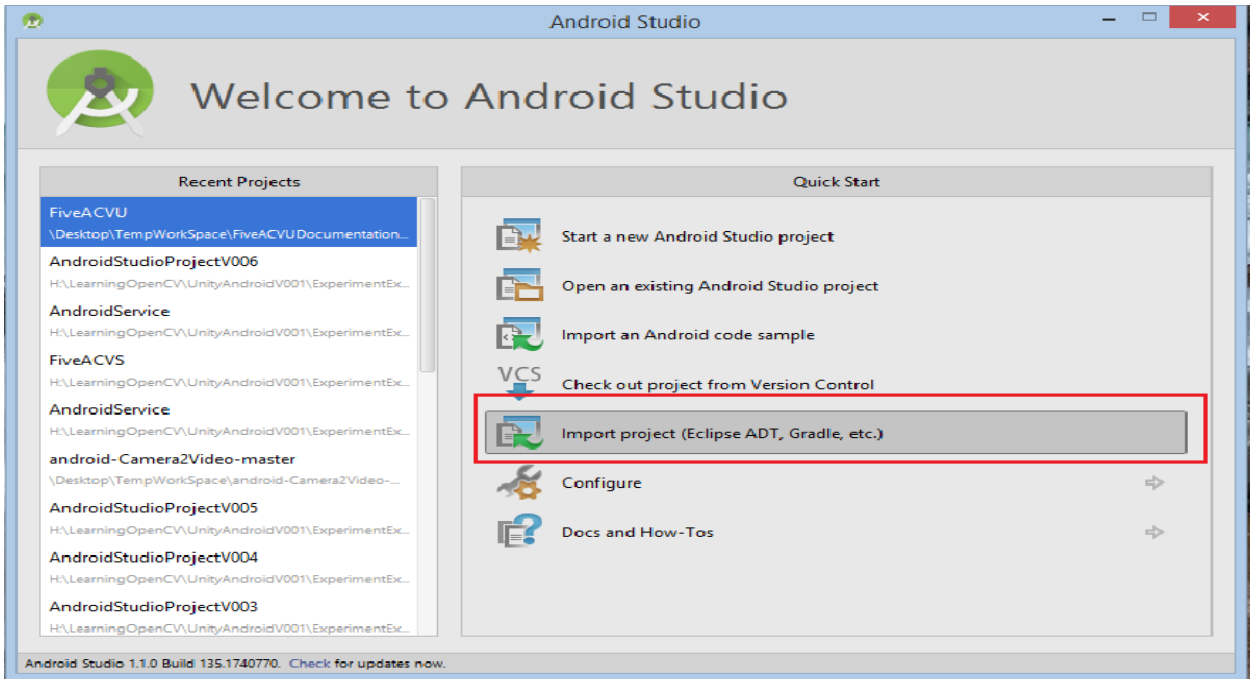
For Computer Vision development the sample project uses OpenCV for Android SDK. The first step for this is to integrate the SDK so that CV modules can use the functionalities in the OpenCV library. As the OpenCV library doesn't support all the functionalities in JAVA, so for processing such functions we can use Android Native Development Kit (NDK). This uses JAVA Native Interfaces (JNI) for implementing the required functions.

For CV application in Unity game scene the camera frames must be captured and processed in the background.

This module can be further enhanced for major processing like positional tracking, gesture detection and so on. If required this module will also use the JNI calls for native functions in OpenCV.

The steps for the development and integration are as follows:

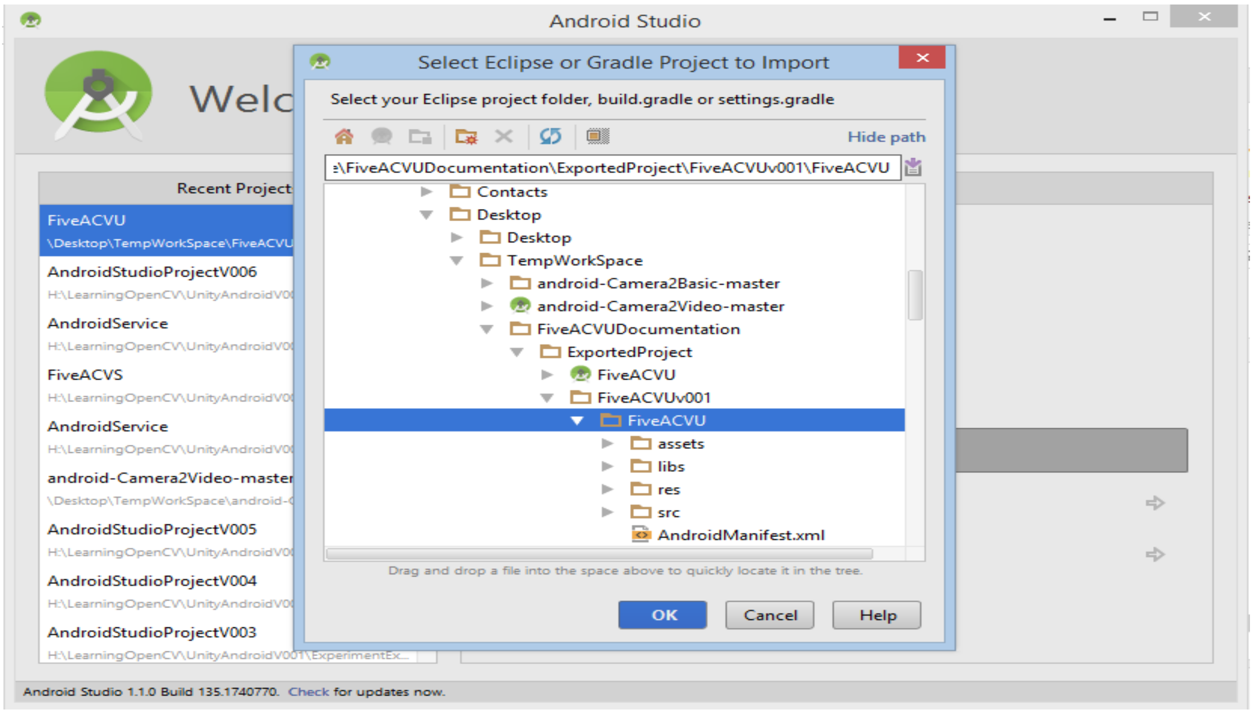
1. Open the Android Studio IDE and Click "Import Project"



Android Studio
Welcome to Android Studio
Recent Projects
FiveACVU
\Desktop\TempWorkSpace\FiveACVU Documentation...
AndroidStudioProjectV006
H:\LearningOpenCV\UnityAndroidV001\ExperimentEx...
AndroidService
H:\LearningOpenCV\UnityAndroidV001\ExperimentEx...
FiveACVS
H:\LearningOpenCV\UnityAndroidV001\ExperimentEx...
AndroidService
H:\LearningOpenCV\UnityAndroidV001\ExperimentEx...
android-Camera2Video-master
\Desktop\TempWorkSpace\android-Camera2Video-...
AndroidStudioProjectV005
H:\LearningOpenCV\UnityAndroidV001\ExperimentEx...
AndroidStudioProjectV004
H:\LearningOpenCV\UnityAndroidV001\ExperimentEx...
AndroidStudioProjectV003
H:\LearningOpenCV\UnityAndroidV001\ExperimentEx...
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos
Android Studio 1.1.0 Build 135.1740770. Check for updates now.

Figure 18

2. Browse to the folder where the Unity application was exported as android project, select and Click OK


Select your Eclipse project folder, build.gradle or settings.gradle

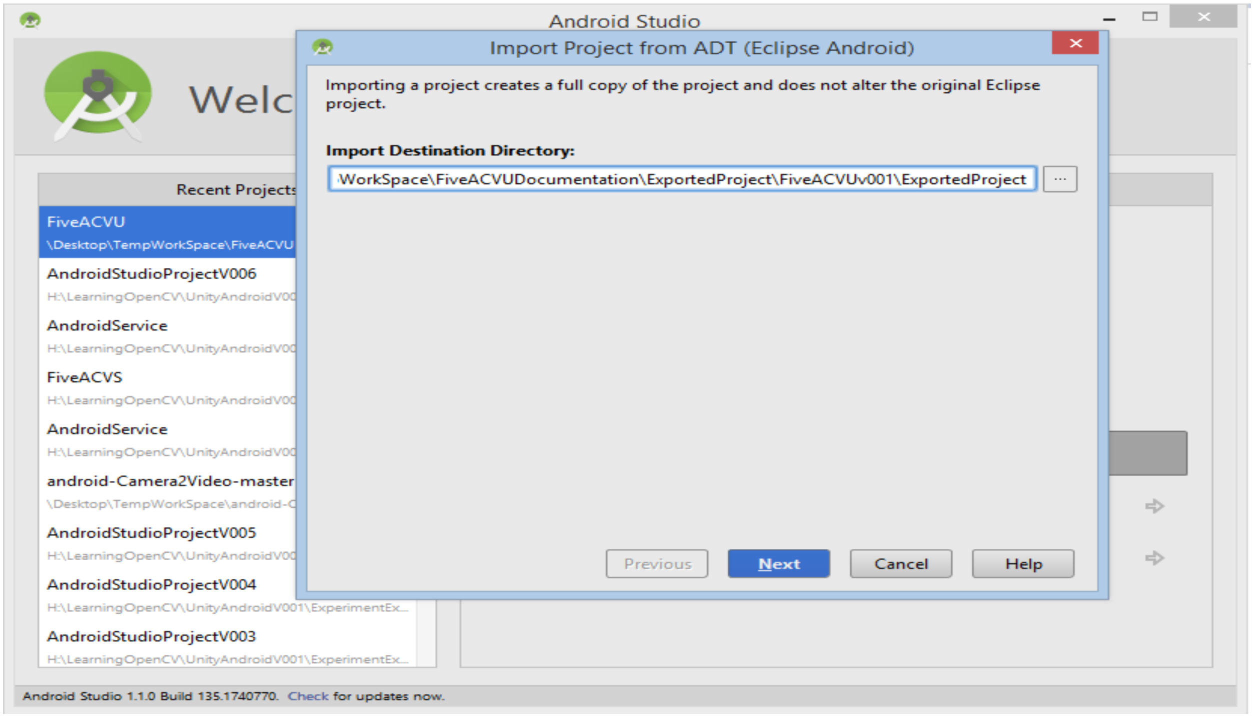
:\FiveACVUDocumentation\ExportedProject\FiveACVUv001\FiveACVU

Drag and drop a file into the space above to quickly locate it in the tree.

Android Studio 1.1.0 Build 135.1740770. Check for updates now.

Figure 19

3. Select the Destination Folder and click “Next” and then “Finish”.

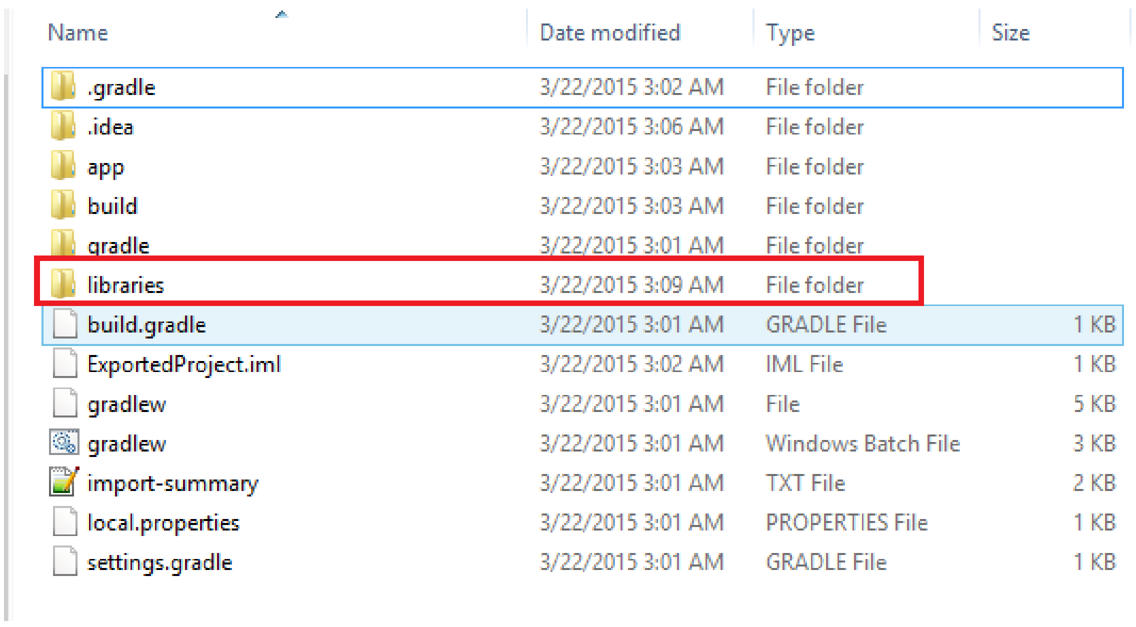

Importing a project creates a full copy of the project and does not alter the original Eclipse project.

Import Destination Directory: WorkSpace\FiveACVUDocumentation\ExportedProject\FiveACVUv001\ExportedProject

Android Studio 1.1.0 Build 135.1740770. Check for updates now.

Figure 20

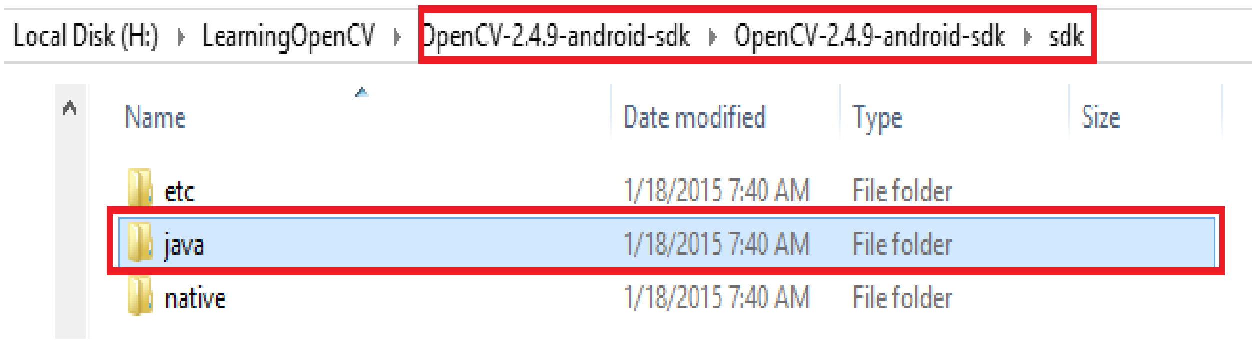
4. Create a folder "libraries" inside the Android Studio Folder. This folder corresponds to the Destination folder mentioned in the previous step



Name	Date modified	Type	Size
.gradle	3/22/2015 3:02 AM	File folder	
.idea	3/22/2015 3:06 AM	File folder	
app	3/22/2015 3:03 AM	File folder	
build	3/22/2015 3:03 AM	File folder	
gradle	3/22/2015 3:01 AM	File folder	
libraries	3/22/2015 3:09 AM	File folder	
build.gradle	3/22/2015 3:01 AM	GRADLE File	1 KB
ExportedProject.iml	3/22/2015 3:02 AM	IML File	1 KB
gradlew	3/22/2015 3:01 AM	File	5 KB
gradlew	3/22/2015 3:01 AM	Windows Batch File	3 KB
import-summary	3/22/2015 3:01 AM	TXT File	2 KB
local.properties	3/22/2015 3:01 AM	PROPERTIES File	1 KB
settings.gradle	3/22/2015 3:01 AM	GRADLE File	1 KB

Figure 21

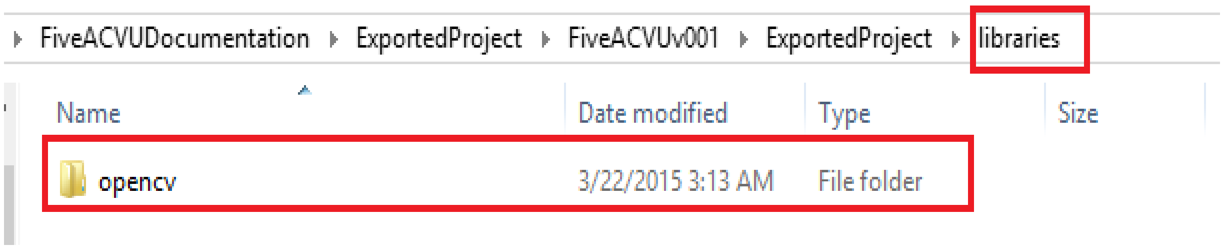
5. After extracting the OpenCV for Android project copy the <OpenCV SDK Folder>/sdk/java folder into libraries folder of the step 4 and rename the same to opencv



Local Disk (H:) ▸ LearningOpenCV ▸ OpenCV-2.4.9-android-sdk ▸ OpenCV-2.4.9-android-sdk ▸ sdk

Name	Date modified	Type	Size
etc	1/18/2015 7:40 AM	File folder	
java	1/18/2015 7:40 AM	File folder	
native	1/18/2015 7:40 AM	File folder	

Figure 22



▸ FiveACVUDocumentation ▸ ExportedProject ▸ FiveACVUv001 ▸ ExportedProject ▸ libraries

Name	Date modified	Type	Size
opencv	3/22/2015 3:13 AM	File folder	

Figure 23

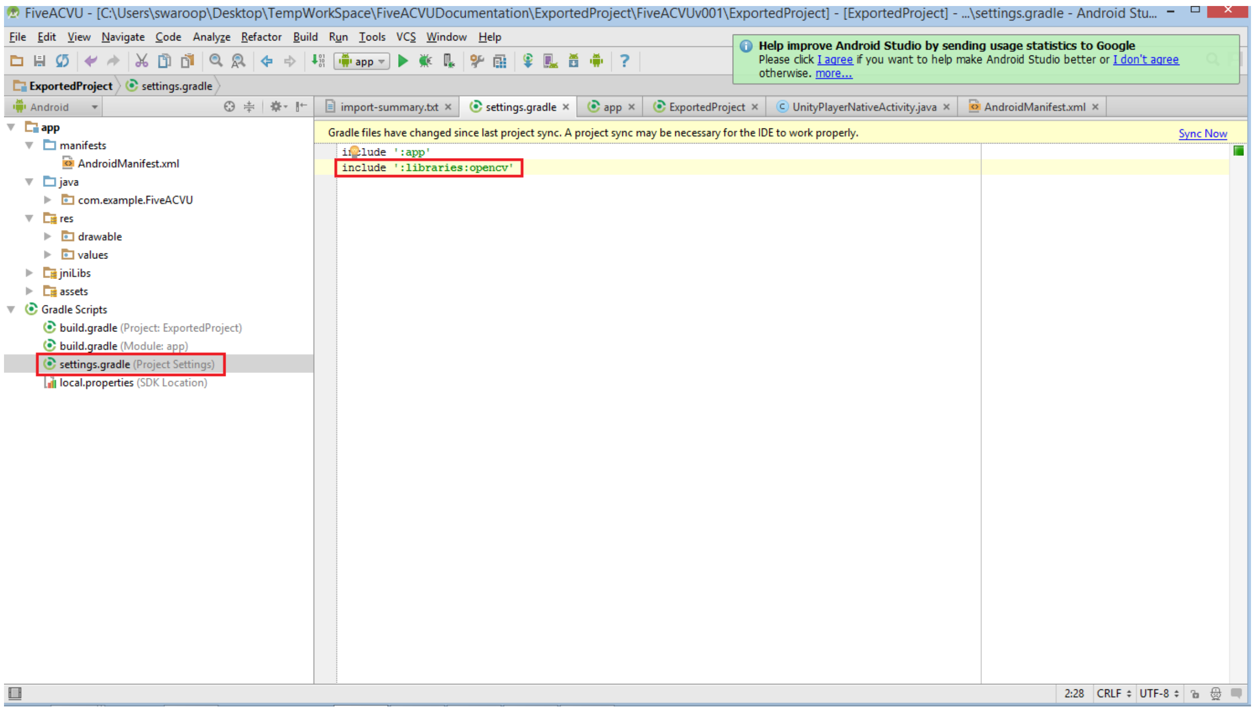
6. Create a “build.gradle” file inside the opencv directory as mentioned in the step 5. The gradle file will have the following instructions. This should be changed as per the targeted version, build tools and OpenCV SDK version.

```
apply plugin: 'android-library'

buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.1.0'
    }
}

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"
    defaultConfig {
        minSdkVersion 8
        targetSdkVersion 21
        versionCode 2490
        versionName "2.4.9"
    }
    sourceSets {
        main {
            manifest.srcFile 'AndroidManifest.xml'
            java.srcDirs = ['src']
            resources.srcDirs = ['src']
            res.srcDirs = ['res']
            aidl.srcDirs = ['src']
        }
    }
}
```

7. Add “include ':libraries:opencv'” to the settings.gradle file in the Android Studio IDE.


FiveACVU - [C:\Users\swaroop\Desktop\TempWorkSpace\FiveACVUDocumentation\ExportedProject\FiveACVUv001\ExportedProject] - [ExportedProject] - ...\settings.gradle - Android Stu...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ExportedProject
settings.gradle
Android
import-summary.txt
settings.gradle
app
ExportedProject
UnityPlayerNativeActivity.java
AndroidManifest.xml
app
manifests
AndroidManifest.xml
java
com.example.FiveACVU
res
drawable
values
jniLibs
assets
Gradle Scripts
build.gradle (Project: ExportedProject)
build.gradle (Module: app)
settings.gradle (Project Settings)
local.properties (SDK Location)
Sync Now
2:28 CRLF UTF-8

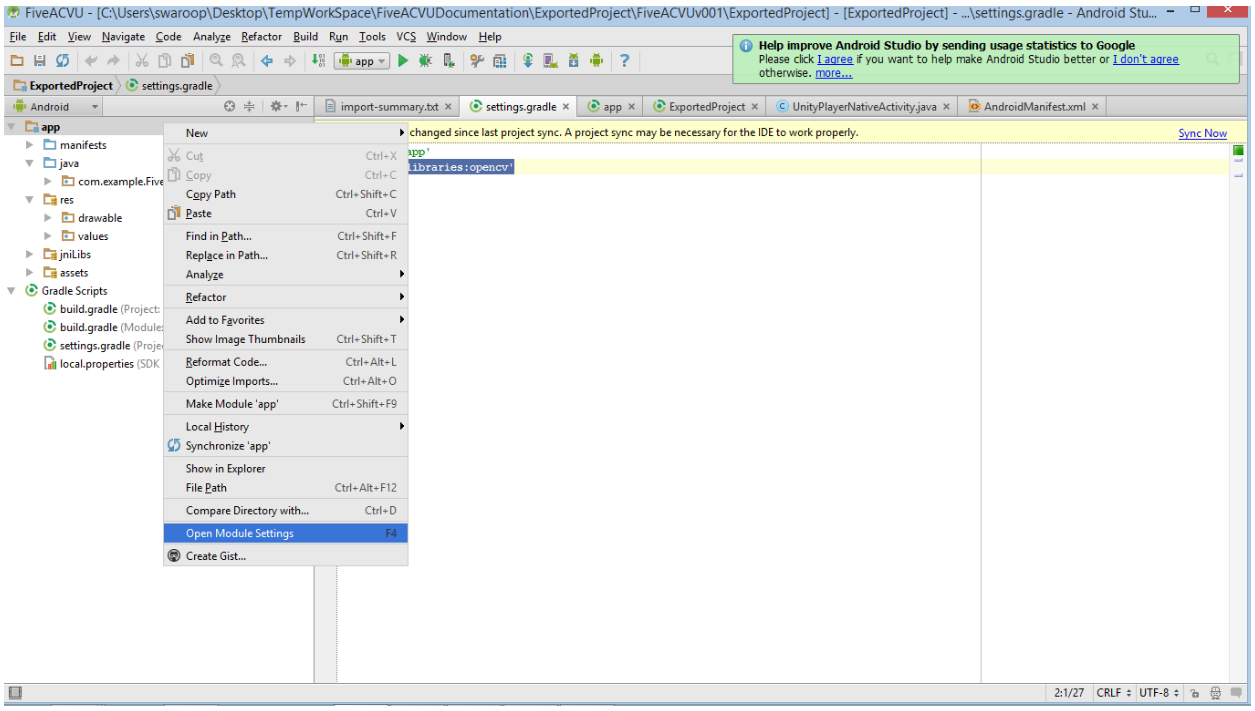
Help improve Android Studio by sending usage statistics to Google
Please click I agree if you want to help make Android Studio better or I don't agree otherwise. more...

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
include ':app'
include ':libraries:opencv'
```

Figure 24

8. Right Click on the app in the project structure and click in the “Öpen Module Settings”


FiveACVU - [C:\Users\swaroop\Desktop\TempWorkSpace\FiveACVUDocumentation\ExportedProject\FiveACVUv001\ExportedProject] - [ExportedProject] - ...\settings.gradle - Android Stu...
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
ExportedProject
settings.gradle
Android
app
manifests
java
com.example.Five
res
drawable
values
jniLibs
assets
Gradle Scripts
build.gradle (Project:
build.gradle (Module:
settings.gradle (Proje
local.properties (SDK
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path Ctrl+Shift+C
Paste Ctrl+V
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Analyze
Refactor
Add to Favorites
Show Image Thumbnails Ctrl+Shift+T
Reformat Code... Ctrl+Alt+L
Optimize Imports... Ctrl+Alt+O
Make Module 'app' Ctrl+Shift+F9
Local History
Synchronize 'app'
Show in Explorer
File Path Ctrl+Alt+F12
Compare Directory with... Ctrl+D
Open Module Settings F4
Create Gist...
Sync Now
2:1/27 CRLF UTF-8

Help improve Android Studio by sending usage statistics to Google
Please click I agree if you want to help make Android Studio better or I don't agree otherwise. more...

changed since last project sync. A project sync may be necessary for the IDE to work properly.

```
app'
ibraries:opencv'
```

Figure 25

9. Go the Dependencies tab in the app present in Modules. Click on the "+" symbol and click on Module dependency.

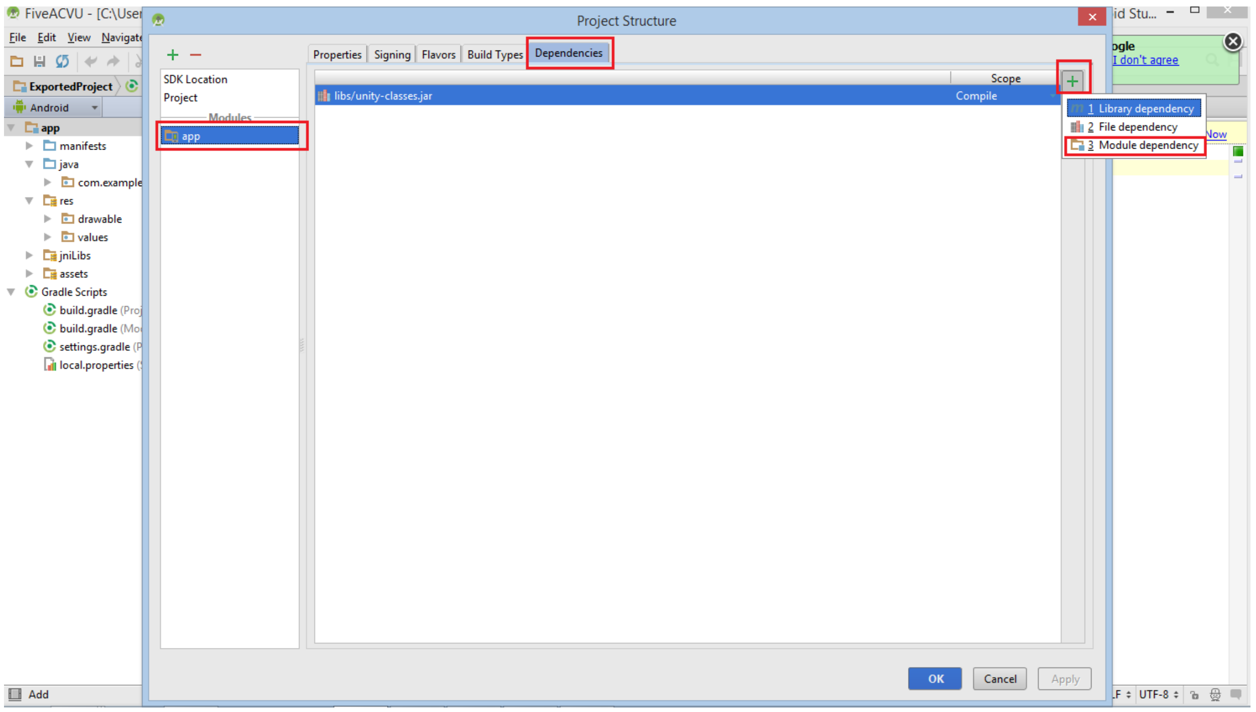


Figure 26

10. Select "libraries:opencv" and select OK -> Apply -> OK to add the dependency. This will sync the gradle again.

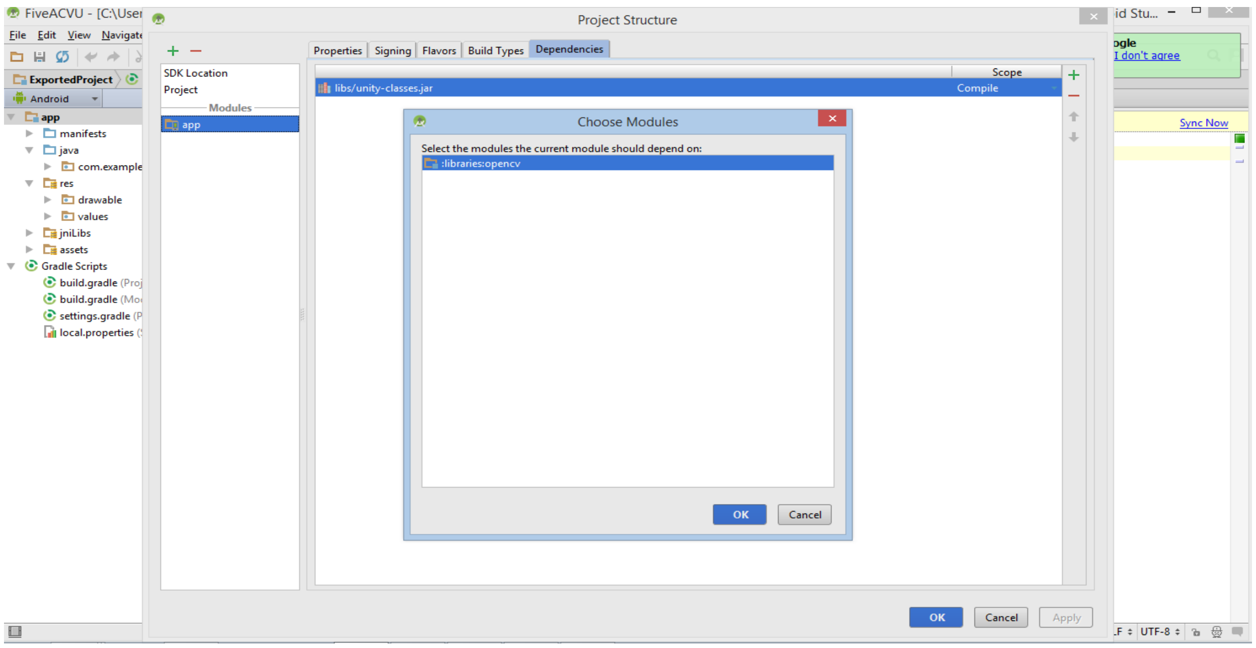
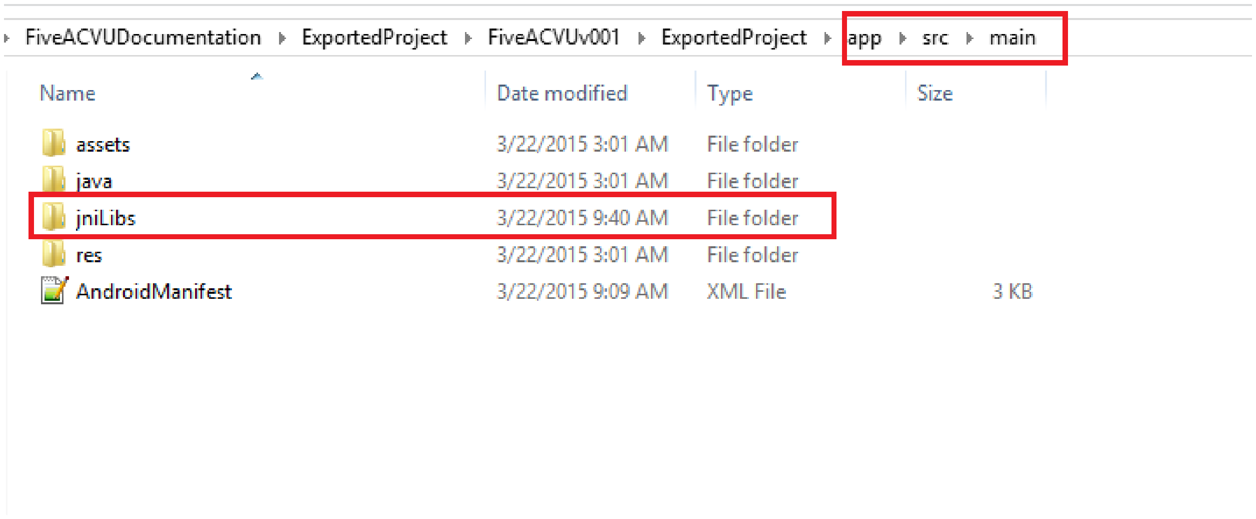


Figure 27

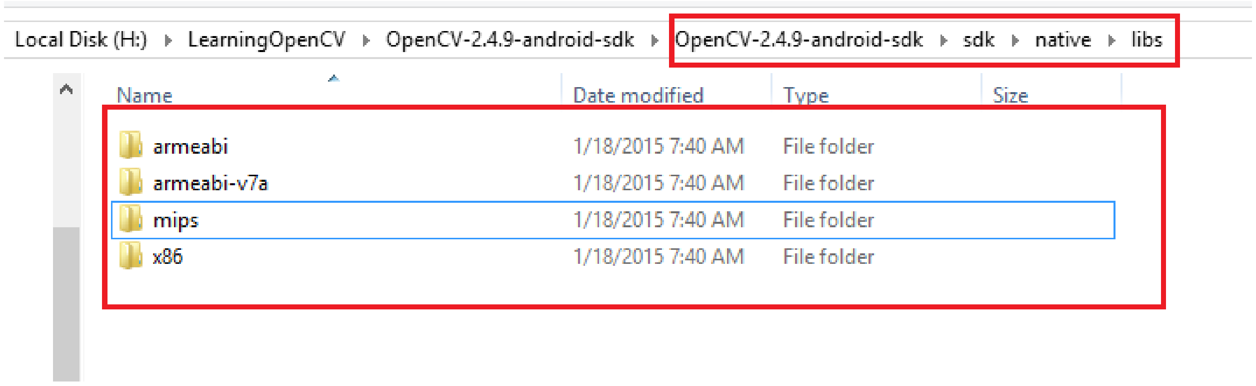
11. Create a folder “jniLibs” inside <Path to the Android Studio Project>/app/src/main. Copy the folders “armeabi”, “armeabi-v7a”, “mips”, “x86” from <OpenCV SDK Folder>/sdk/native/libs to the jniLibs folder created. The “armeabi-v7a” folder contents must be merged as it also contains libraries related to Unity.



FiveACVUDocumentation ▸ ExportedProject ▸ FiveACVUv001 ▸ ExportedProject ▸ app ▸ src ▸ main

Name	Date modified	Type	Size
assets	3/22/2015 3:01 AM	File folder	
java	3/22/2015 3:01 AM	File folder	
jniLibs	3/22/2015 9:40 AM	File folder	
res	3/22/2015 3:01 AM	File folder	
AndroidManifest	3/22/2015 9:09 AM	XML File	3 KB

Figure 28



Local Disk (H:) ▸ LearningOpenCV ▸ OpenCV-2.4.9-android-sdk ▸ OpenCV-2.4.9-android-sdk ▸ sdk ▸ native ▸ libs

Name	Date modified	Type	Size
armeabi	1/18/2015 7:40 AM	File folder	
armeabi-v7a	1/18/2015 7:40 AM	File folder	
mips	1/18/2015 7:40 AM	File folder	
x86	1/18/2015 7:40 AM	File folder	

Figure 29

12. Add a Java Class "UnityDataMgr" in the package. This class is the interface class for Unity C# script. The sample project feeds data into this class from the sensor module and the swipe gestures. In the future versions this class will be modified as per requirement.

```
package com.example.FiveACVU;

public class UnityDataMgr {

        //Rotational Data

        static float RotationX = 0.0f;

        static float RotationY = 0.0f;

        static float RotationZ = 0.0f;


        //Translational Data

        static float TransformX = 0.0f;

        static float TransformY = 0.0f;

        static float TransformZ = 0.0f;


      //Methods to get the rotational and transformation Data in t

      public static float GetRotationX(){ return RotationX; }

      public static float GetRotationY(){ return RotationY; }

      public static float GetRotationZ(){ return RotationZ;}

      public static float GetTransformX(){ return TransformX; }

      public static float GetTransformY(){ return TransformY; }

      public static float GetTransformZ(){ return TransformZ; }

}
```

13. Add a Java Class "SensorMgr" in the package. The class is responsible for initializing accelerometer. The class gets the accelerometer values and updates in the UnityDataMgr Rotational values. These are updated in the rotational values of the user transform. In the future version these may be deprecated if the rotational values are taken from the OVR Plugin for the Gear VR. The code for the SensorMgr is as follows.

```
package com.example.FiveACVU;
import android.content.Context;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
public class SensorMgr implements SensorEventListener {
        private SensorManager sensorManager = null;
        Context m_Context = null;
        SensorMgr(Context context){
        // Get a reference to a SensorManager
        m_Context = context;
        sensorManager = (SensorManager) context.getSystemService(context.SENSOR_SERVICE);}
        public void initSensorManager(){
                sensorManager.registerListener(this,
                                sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER),
                                SensorManager.SENSOR_DELAY_NORMAL);}
        public void closeSensorManager(){ sensorManager.unregisterListener(this); }
        public void onSensorChanged(SensorEvent sensorEvent){
                switch (sensorEvent.sensor.getType()){
                case Sensor.TYPE_ACCELEROMETER:{
                        UnityDataMgr.RotationX = sensorEvent.values[0];
                        UnityDataMgr.RotationY = sensorEvent.values[1];
                        UnityDataMgr.RotationZ = sensorEvent.values[2]; }
                break;
                default:
                break; } }
        public void onAccuracyChanged(Sensor arg0, int arg1){ } }
```

14. Create a CamPreview.java class in the package with the following code. This class extends the TextureView and implements the SurfaceTextureListener. This class also sets the surface texture as invisible when it is available. This helps in keeping the camera capture from overlapping the Unity scene view and supports the background capture mechanism mentioned in the following points.
    This class starts the camera preview and the frames will be received in a separate class "CamCallback"

```
package com.example.FiveACVU;
import android.content.Context;
import android.hardware.Camera;
import android.graphics.SurfaceTexture;
import android.view.Gravity;
import android.view.TextureView;
import android.widget.FrameLayout;
import java.io.IOException;

public class CamPreview extends TextureView implements TextureView.SurfaceTextureListener {
  private Camera mCamera;
  public CamPreview(Context context, Camera camera) {
    super(context);
    mCamera = camera;}
  @Override
  public void onSurfaceTextureAvailable(SurfaceTexture surface, int width, int height) {
    Camera.Size previewSize = mCamera.getParameters().getPreviewSize();
    setLayoutParams(new FrameLayout.LayoutParams(
        previewSize.width, previewSize.height, Gravity.CENTER));
    try{ mCamera.setPreviewTexture(surface); } catch (IOException t) {}
    mCamera.startPreview();
    this.setVisibility(INVISIBLE); // Make the surface invisible as soon as it is created
  }
  @Override
  public void onSurfaceTextureSizeChanged(SurfaceTexture surface, int width, int height) {}
  @Override
  public boolean onSurfaceTextureDestroyed(SurfaceTexture surface) {return true;}
  @Override
  public void onSurfaceTextureUpdated(SurfaceTexture surface) {}}
```

15. Add a class CamCallback.java in the package which implements Camera.PreviewCallback. This class receives the individual frame captured using the android device camera. The frame captured is changed in to MAT format of OpenCV for further processing.

```
package com.example.FiveACVU;

import android.graphics.ImageFormat;

import android.hardware.Camera;

import android.os.Environment;

import org.opencv.core.CvType;

import org.opencv.core.Mat;

import org.opencv.highgui.Highgui;

import org.opencv.imgproc.Imgproc;

import java.io.File;

public class CamCallback implements Camera.PreviewCallback{

  Mat mRgba,mYuv;

  public static boolean Save = false;

  static{ System.loadLibrary("opencv_java");}

  public void onPreviewFrame(byte[] data, Camera camera){

    // Process the camera data here

    int format = camera.getParameters().getPreviewFormat();


      int width = camera.getParameters().getPreviewSize().width;

      int height = camera.getParameters().getPreviewSize().height;

      mRgba = new Mat();

      mYuv = new Mat(height + height/2, width, CvType.CV_8UC1);

      mYuv.put(0, 0, data);

      Imgproc.cvtColor(mYuv, mRgba, Imgproc.COLOR_YUV2BGRA_NV21);

      File path = new File(Environment.getExternalStorageDirectory() + "/Images/");

      path.mkdirs();

      String filename = "TestImage.png";

      File file = new File(path, filename);

      filename = file.toString();

      Boolean bool = Highgui.imwrite(filename, mRgba);}}
```

The OnPreviewFrame function is called every frame. The byte[] data contains the captured frame detail matrix. This matrix is of NV21 image format. The data is stored in single channel grayscale format, so it can be converted to OpenCV Mat image with CvType as CV_8UC1. It can then be converted into desired format for further processing. In the sample application it is further converted to BGRA_NV21 format and stored on the device for verification of the captured image every frame. The image can be found in the Images folder in the disk storage of the android device.

It must be noted that while specifying the parameters for converting byte[] data into mat image the height should be used as height + height/2 and width should be the default as received in the camera parameters.

Also the coordinate where the data is being put is the origin hence 0,0 is used for the same reason. This method of capture is valid till the Android API level 19. So, the devices running API level 21 and later, should use the new capture method.

16. The class UnityPlayerNativeActivity is generated by Unity which contains the game scene related data and functions. It also injects most of the input, orientation and similar android activity related details into the Unity player, so it could be handled inside Unity modules if needed.

    As we need to add a SurfaceTexture on the top of Unity game scene for the background camera capture, we need to find the leaf view in the hierarchy of the Unity activity.

    To implement this mechanism a class “VRActivity” is created which extends UnityPlayerNativeActivity. We need to remove the setContentView method in the class to ensure it doesn’t overwrite the base view of the Unity player which contains the game scene. The mechanism to add the surface texture is to find the leaf view and add the surface texture to this view in a separate thread.

    The CamPreview class and the CamCallback is invoked from within this thread which begins the background capture. As mentioned earlier the CamPreview class is responsible for making the surface texture invisible and to start the camera preview.

    The CamCallback will receive the every captured frame and process it to an OpenCV format image. This image can be used for further processing as required by the application.

    For adding any additional android GUI feature like button, text or other view the same method can be used.

    The sample code for this class is as follows.

```
package com.example.FiveACVU;
import android.app.Activity;
import android.content.Context;
import android.hardware.Camera;
import android.os.Bundle;
import android.os.Handler;
import android.util.Log;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.FrameLayout;
import android.widget.Toast;
import java.util.Calendar;
public class VRActivity extends UnityPlayerNativeActivity {
  SensorMgr objSensorMgr;
  static Context VRActivityContext;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    objSensorMgr = new SensorMgr(getApplicationContext());
    VRActivityContext = this.getApplicationContext();
    final long delay = 5000;//ms
    Handler handler = new Handler();
    Runnable runnable = new Runnable() {
        public void run() {
            // Setup the camera and the preview object
             Camera mCamera = null;
             mCamera = Camera.open(0);
             CamPreview camPreview = new CamPreview(VRActivity.this,mCamera);
             camPreview.setSurfaceTextureListener(camPreview);
            ViewGroup rootView = (ViewGroup)VRActivity.this.findViewById
                    (android.R.id.content);
            // find the first leaf view (i.e. a view without children)
             // the leaf view represents the topmost view in the view stack
             View topMostView = getLeafView(rootView);

             // let's add a sibling to the leaf view
             ViewGroup leafParent = (ViewGroup)topMostView.getParent();
             FrameLayout preview = new FrameLayout(VRActivity.this);
            leafParent.addView(preview, new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
                    ViewGroup.LayoutParams.WRAP_CONTENT));
            // Connect the preview object to a FrameLayout in your UI
             // You'll have to create a FrameLayout object in your UI to place this preview in
             preview.addView(camPreview);

             // Attach a callback for preview
             CamCallback camCallback = new CamCallback();
             camCallback.SetContext(VRActivityContext);
             mCamera.setPreviewCallback(camCallback);}};
        handler.postDelayed(runnable, delay);
    }
```

```
private View getLeafView(View view) {
        if (view instanceof ViewGroup) {
            ViewGroup vg = (ViewGroup)view;
            for (int i = 0; i < vg.getChildCount(); ++i) {
                View chview = vg.getChildAt(i);
                View result = getLeafView(chview);
                if (result != null)
                    return result;
            }
            return null;
        }
        else {
            Log.v("VRActivity", "Found leaf view");
            return view;
        }
    }

// Quit Unity
    @Override protected void onDestroy ()
    {super.onDestroy();}
    // Pause Unity
    @Override protected void onPause()
    {super.onPause();objSensorMgr.closeSensorManager();}

    // Resume Unity
    @Override protected void onResume(){super.onResume();objSensorMgr.initSensorManager();}}
```

6.4 Troubleshooting Android Application

1. For running the application on Note4 OpenCV 2.4.11 version should be used. The version can be downloaded from http://sourceforge.net/projects/opencvlibrary/files/opencv-android/2.4.11/
2. OpenCV_2.4.11_binary_pack_armv7a.apk and OpenCV_2.4.11_Manager_2.20_armv7a-neon.apk must be installed on the Note4 device. The apk can be found in the apk folder inside the sdk directory.
3. For devices running on Android version prior to 4.4*, the videocapture of OpenCV can be used to capture the frame. The details is mentioned is the FIVE ACVUv1.0.
4. The manifest file used for the sample application is as follows.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.example.FiveACVU"
  android:installLocation="preferExternal"
  android:versionCode="1"
  android:versionName="1.0" >
<uses-sdk
    android:minSdkVersion="19"
    android:targetSdkVersion="21" />
<uses-feature android:glEsVersion="0x00030000" />
  <uses-permission android:name="android.permission.CAMERA" />
  <uses-feature android:name="android.hardware.camera" />
  <uses-feature android:name="android.hardware.camera.autofocus" />
  <uses-feature android:name="android.hardware.camera.front" />
  <uses-feature android:name="android.hardware.camera.front.autofocus" />
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  <uses-feature
    android:name="android.hardware.touchscreen"
    android:required="false" />
  <android:uses-permission
    android:name="android.permission.WRITE_EXTERNAL_STORAGE"
    android:maxSdkVersion="18" />
  <android:uses-permission android:name="android.permission.READ_PHONE_STATE" />

  <application
    android:debuggable="false"
    android:icon="@drawable/app_icon"
    android:label="@string/app_name"
    android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" >
    <meta-data
      android:name="com.samsung.android.vr.application.mode"
      android:value="vr_only" />
    <activity
      android:name=".VRActivity"
android:configChanges="mcc|mnc|locale|keyboard|keyboardHidden|navigation|orientation|screenLay
out|uiMode|screenSize|smallestScreenSize|fontScale"
      android:label="@string/app_name"
      android:launchMode="singleTask"
      android:screenOrientation="landscape" >
      <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
      <meta-data
        android:name="unityplayer.UnityActivity"
        android:value="true" />
      <meta-data
        android:name="unityplayer.ForwardNativeEventsToDalvik"
        android:value="true" />
    </activity>
    <activity
      android:name="com.oculusvr.vrlib.PlatformActivity"
      android:configChanges="screenSize|orientation|keyboardHidden|keyboard"
      android:launchMode="singleTask"
      android:screenOrientation="landscape"
      android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" >
    </activity>    </application></manifest>
```