Par : Mohamed Amine KHARRAT

Création et installation d'une application Android qui ouvre la caméra d'un smartphone en vision gray

Pour cela, on a besoin d'installer :

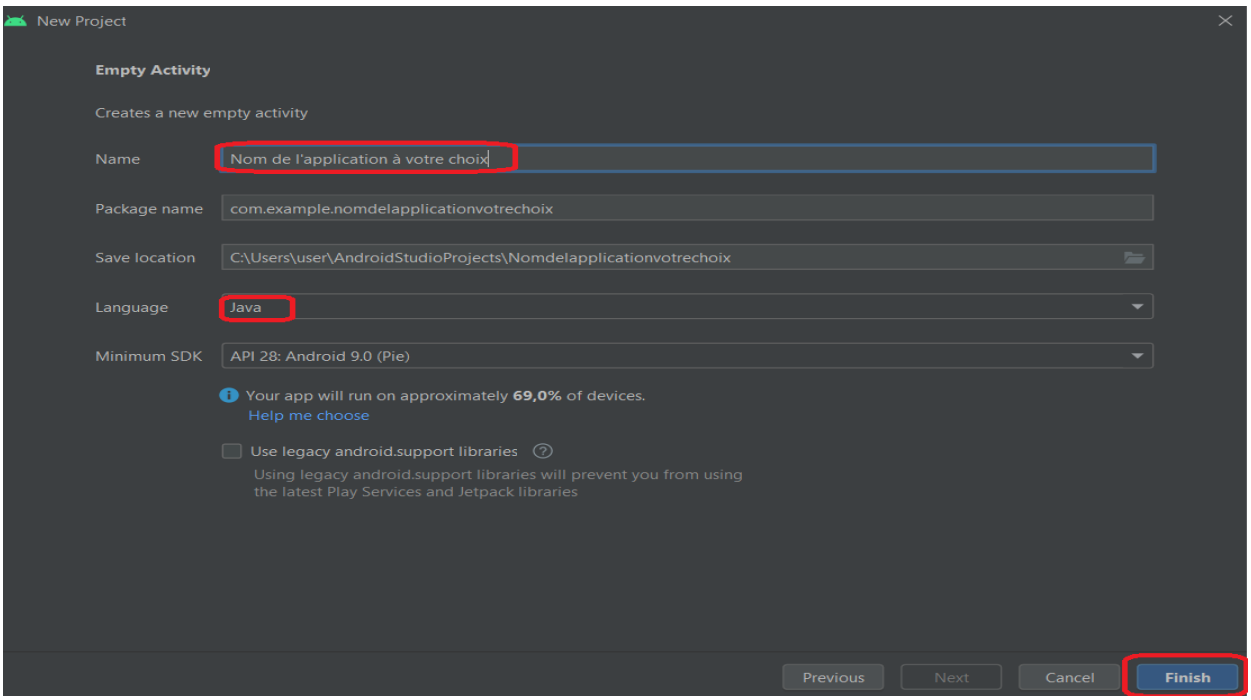
- Android Studio : est un environnement de développement pour développer des applications mobiles Android avec le langage Java.
  Lien de téléchargement :
  https://developer.android.com/studio?hl=fr&gclid=CjwKCAiAvriMBhAuEiwA8Cs5lTEo-U4fWLjKvX5wNbI99VH6FYWcs34fkG8EphI7Tt7hV9-bM1nzPBoCNdkQAvD_BwE&gclsrc=aw.ds

- openCV : (Open Source Computer Vision) est une bibliothèque graphique libre, spécialisée dans le traitement d'images en temps réel.
  Lien de téléchargement : (il faut installer la dernière version pour qu'elle soit compatible avec Android Studio)
  https://sourceforge.net/projects/opencvlibrary/files/4.5.4/opencv-4.5.4-android-sdk.zip/download

- **1ére étape** : Intégration de openCV dans Android Studio

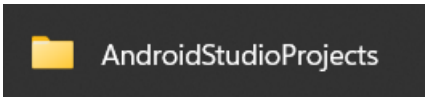
Après l'installation d'Android Studio, on crée un nouveau projet **''New Project''**

On choisit **'Empty Activity'.** On choisit le nom de l'application désirer et on choisit la langage Java.

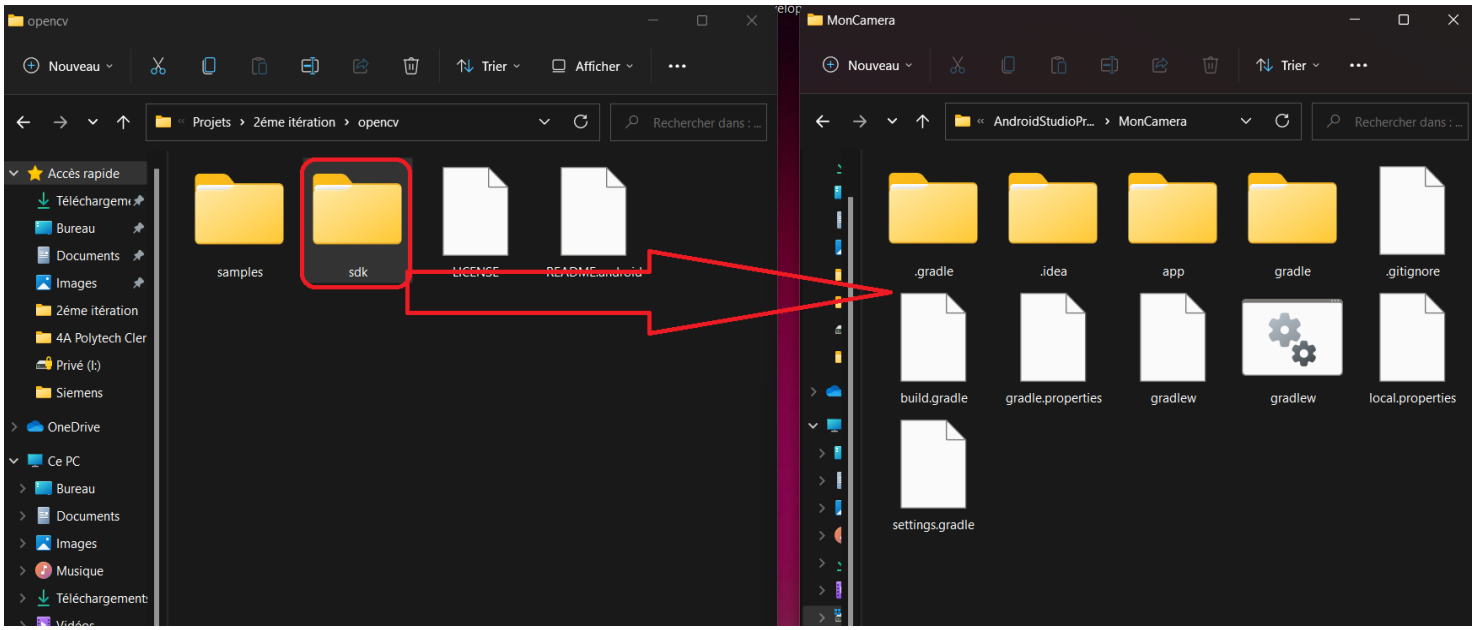


Ensuite, on ouvre le fichier du openCV et on copie le sous fichier '**sdk**'

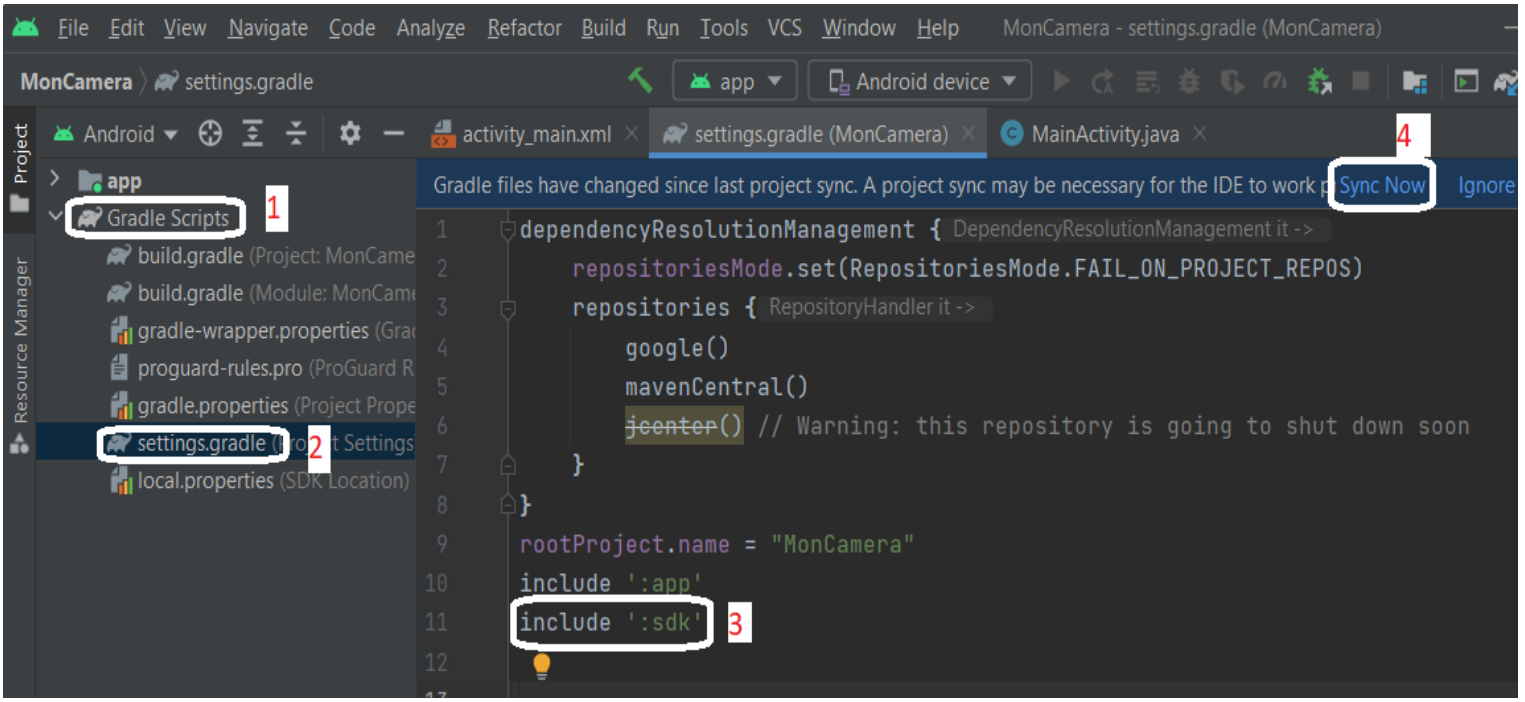
Après, on cherche sur le PC le fichier :



On ouvre le fichier de notre projet et on colle 'sdk'



Sur **Android Studio>Gradle Scripts>settings.gradle** , on ajoute la commande <**include ':sdk'**> et on clique sur **Sync Now.**

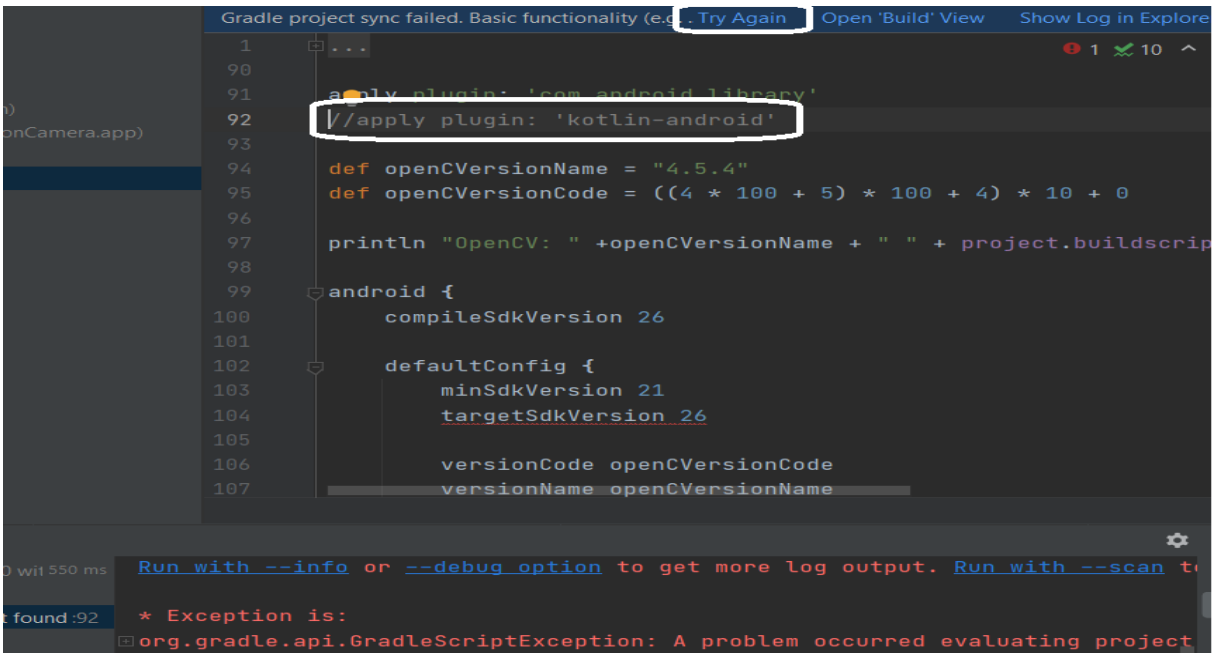


Si une erreur s'est produite : un nouvel angle s'ouvre (build.gradle)

On élimine la deuxième ligne dans cet angle par '**//**'

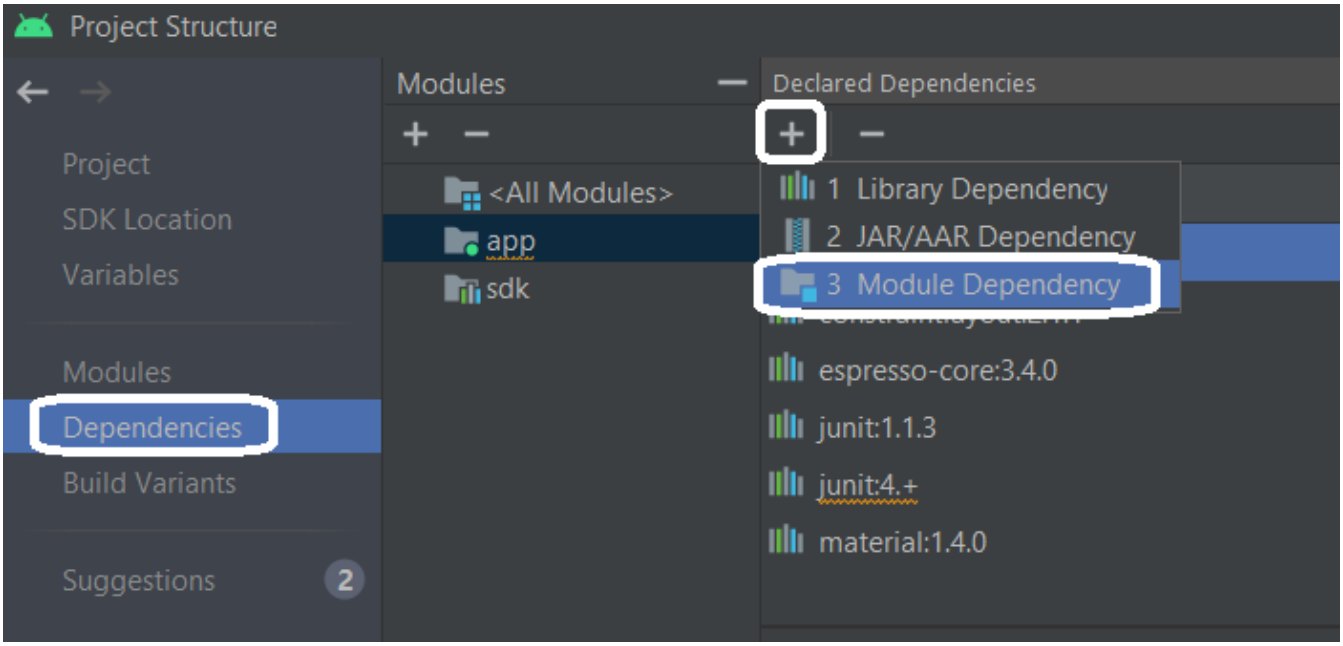
➔ (//apply plugin: 'kotlin-android')

Et on clique sur **Try Again**



Après, **File> Project Structure> Dependencies > + > Module Dependency** et on coche **sdk**

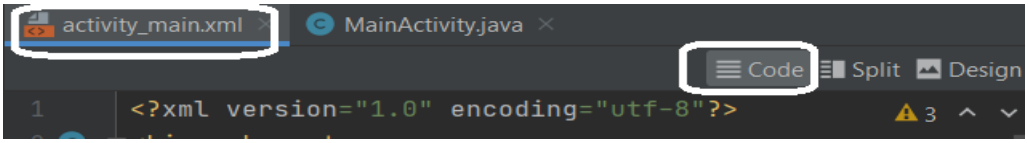
En clique sur **Finish**



- **2éme étape** : On va construire la première interface de l'application qui contient un bouton sert à lancer la caméra.

  L'angle " **activity_main.xml** " est celle qui va construire cette interface.
  (Si elle n'est pas ouverte, on peut la trouver dans **app>res>layout**)

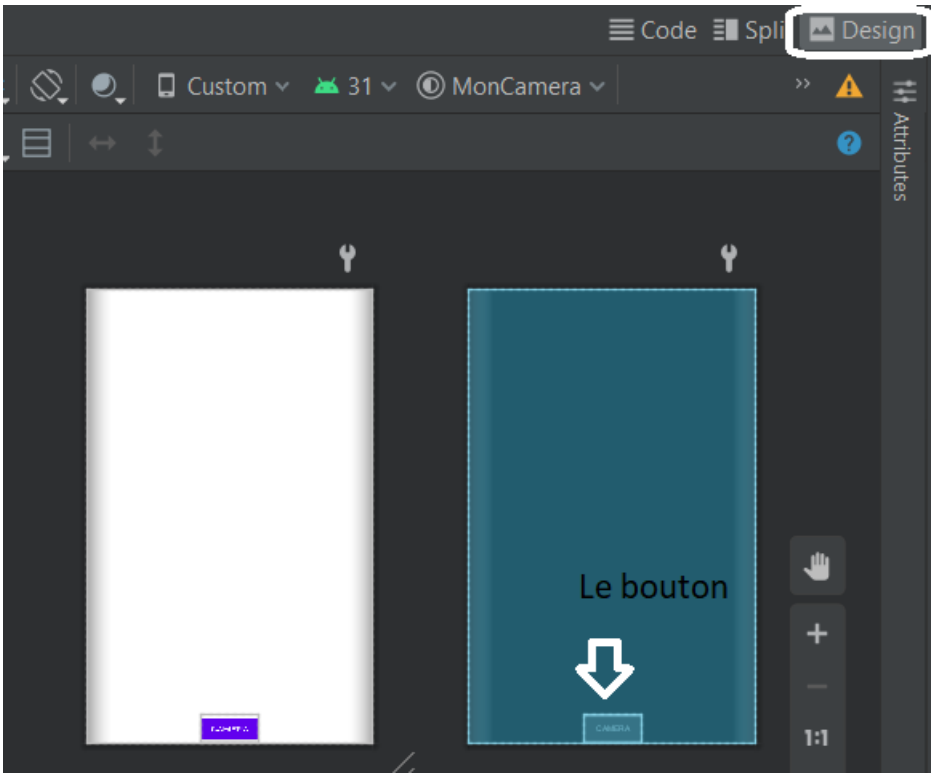
  Pour cela, on écrit le code suivant dans cet angle :




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity"
   android:gravity="center|bottom">

   <Button
      android:id="@+id/open_camera"
      android:layout_width="wrap_content"
      android:layout_height="50dp"
      android:text="Camera"></Button>
</LinearLayout>
```

On peut voir aussi cette interface si on tape sur **Design**



<u>3éme étape</u> : Corps de l'application

On premier temps, on va ajouter une nouvelle activité que l'on va utiliser après.

File>New>Activity>Empty Activity et on nomme cette nouvelle activité '' opencvcamera ''

Dans '' **MainActivity.java** '' on écrit le code suivant :

```
package com.example.MonCamera;   // Ne pas modifer cette package sur votre code: on fait, MonCamera est le nom de mon application


import androidx.appcompat.app.AppCompatActivity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;


public class MainActivity extends AppCompatActivity {

   private static String TAG = "MainActivity";

   private Button open_camera;                    //definition du bouton qui sert à ouvrir la caméra


   @Override

   protected void onCreate(Bundle savedInstanceState) {

      super.onCreate(savedInstanceState);

      setContentView(R.layout.activity_main);

      open_camera = findViewById(R.id.open_camera);          // Relie le bouton par son adresse

      // On va définir ici la fonction lorsqu'on tape sur le bouton qui ouvre la caméra dans une nouvelle interface

      open_camera.setOnClickListener(new View.OnClickListener() {

         @Override

         public void onClick(View view) {

            startActivity(new Intent(MainActivity.this, opencvcamera.class).addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK | Intent.FLAG_ACTIVITY_CLEAR_TOP));

         }

      });

   }
```

L'angle '' **activity_opencvcamera.xml** '' est l'interface où la caméra va s'ouvrir.

Pour cela, on écrit dans cette angle le code suivant :

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <org.opencv.android.JavaCameraView
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:id="@+id/camera_surface"/>
</FrameLayout>
```

Maintenant, l'angle '' **opencvcamera.java** '' est la fonction qui va ouvrir la caméra du smartphone et va rendre la vision en gray.

Avant de coder cette fonction, il faut ajouter pour l'application des permissions qui demande de l'utilisateur s'il va autoriser cette application à utiliser la caméra du smartphone.

Pour cela, dans **app>manifests>AndroidManifest.xml** on ajoute les codes de permissions suivantes **(avant <application !!!!)** :

```
<uses-permission android:name="android.permission.CAMERA"/>
   <uses-feature android:name="android.hardware.camera" android:required="false"/>
   <uses-feature android:name="android.hardware.camera.front" android:required="false"/>
   <uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>
   <uses-feature android:name="android.hardware.camera.autofocus" android:required="false"/>
```

Ensuite, dans l'angle '' **opencvcamera.java** '' on écrit le corps de la fonction :

```
package com.example.MonCamera;  //Ne modifier pas cette instruction dans votre programme : MonCamera est le nom de mon application


import androidx.annotation.NonNull;
import androidx.core.app.ActivityCompat;
import android.Manifest;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.util.Log;
import android.view.SurfaceView;
import org.opencv.android.BaseLoaderCallback;
import org.opencv.android.CameraBridgeViewBase;
import org.opencv.android.LoaderCallbackInterface;
import org.opencv.android.OpenCVLoader;
import org.opencv.core.CvType;
import org.opencv.core.Mat;
import org.opencv.imgproc.Imgproc;


public class opencvcamera extends Activity implements CameraBridgeViewBase.CvCameraViewListener2{
   Mat mRGBA,mRGBAT;          // declaration de deux matrices qui vont retourner la vision en gray ou vision normale
   private static final String TAG="MainActivity";
   CameraBridgeViewBase cameraBridgeViewBase;
   BaseLoaderCallback baseLoaderCallback=new BaseLoaderCallback(this) {
      @Override
//onManagerConnected sert à active la vue si openCV est chargé correctement
      public void onManagerConnected(int status) {
         switch (status) {
```

```
case LoaderCallbackInterface.SUCCESS: {
                Log.i(TAG, "onManagerConnected:opencv loader");
                cameraBridgeViewBase.enableView();
            }
            default: {
                super.onManagerConnected(status);
            }
            break;
        }
    }
  };
  @Override
//onCreate sert à lorsque on clique sur le bouton dans la première interface, elle nous ouvre une deuxième interface pour la caméra
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    ActivityCompat.requestPermissions(opencvcamera.this,new String[]{Manifest.permission.CAMERA},1);
    setContentView(R.layout.activity_opencvcamera);
    cameraBridgeViewBase=(CameraBridgeViewBase) findViewById(R.id.camera_surface);
    cameraBridgeViewBase.setVisibility(SurfaceView.VISIBLE);
    cameraBridgeViewBase.setCvCameraViewListener(this);
  }
  @Override
//onRequestPermissionsResult est la fonction qui va demander de l'utilisateur s'il autorise l'application à utiliser la caméra
  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    switch (requestCode){
      case 1:{
        if(grantResults.length>0 && grantResults[0]== PackageManager.PERMISSION_GRANTED){
          cameraBridgeViewBase.setCameraPermissionGranted();
        }
        else{
        }
```

```
}
    @Override
    protected void onResume() {
        super.onResume();
        if(OpenCVLoader.initDebug()){
            Log.d(TAG, "onResume :opencv initialized");
            baseLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }
        else{
            Log.d(TAG, "Error");
            OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION,this, baseLoaderCallback);
        }
    }
    @Override
//onPause sert à lorsque on sort de l'application, la caméra va s'arrêter
    protected void onPause() {
        super.onPause();
        if(cameraBridgeViewBase !=null){
            cameraBridgeViewBase.disableView();
        }
    }
    @Override
//onDestroy sert à lorsque on ferme l'application, la caméra va s'arrêter
    protected void onDestroy() {
        super.onDestroy();
        if(cameraBridgeViewBase !=null){
            cameraBridgeViewBase.disableView();
        }
    }
```

```
@Override
    public void onCameraViewStopped() {
        mRGBA.release();

    }

    @Override
//onCameraViewStarted sert à importer la vision de la caméra
    public void onCameraViewStarted(int width, int height) {
        mRGBA=new Mat(height, width, CvType.CV_8UC4);
        mRGBAT=new Mat(height, width, CvType.CV_8UC4);

    }

    @Override
//onCameraFrame est la fonction qui va modifier la vision de la caméra
    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame inputFrame) {
        mRGBA=inputFrame.rgba();
        mRGBAT=inputFrame.gray();
        Imgproc.cvtColor(mRGBA,mRGBA,Imgproc.COLOR_BGR2GRAY); // ce code va convertir le couleur en gray
        return mRGBA; // on retourne mRGBA qui est le nouveau couleur en gray
    }
}
```

Si on veut de ne pas modifier la vision en gray, on efface dans la fonction **onCameraFrame** la commande suivante :

```
Imgproc.cvtColor(mRGBA,mRGBA,Imgproc.COLOR_BGR2GRAY);
```

4éme étape : Importation de l'application sur le smartphone

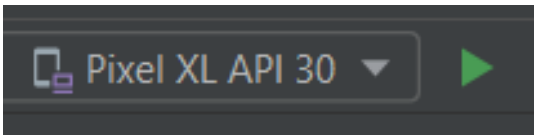
Solution 1 :

Android Studio nous permet de ne pas installer à chaque l'application sur le smartphone pour la tester. Pour cela, Android Studio nous donne la possibilité de crée un appareil virtuel.

On clique sur **No devices>AVD Manager**

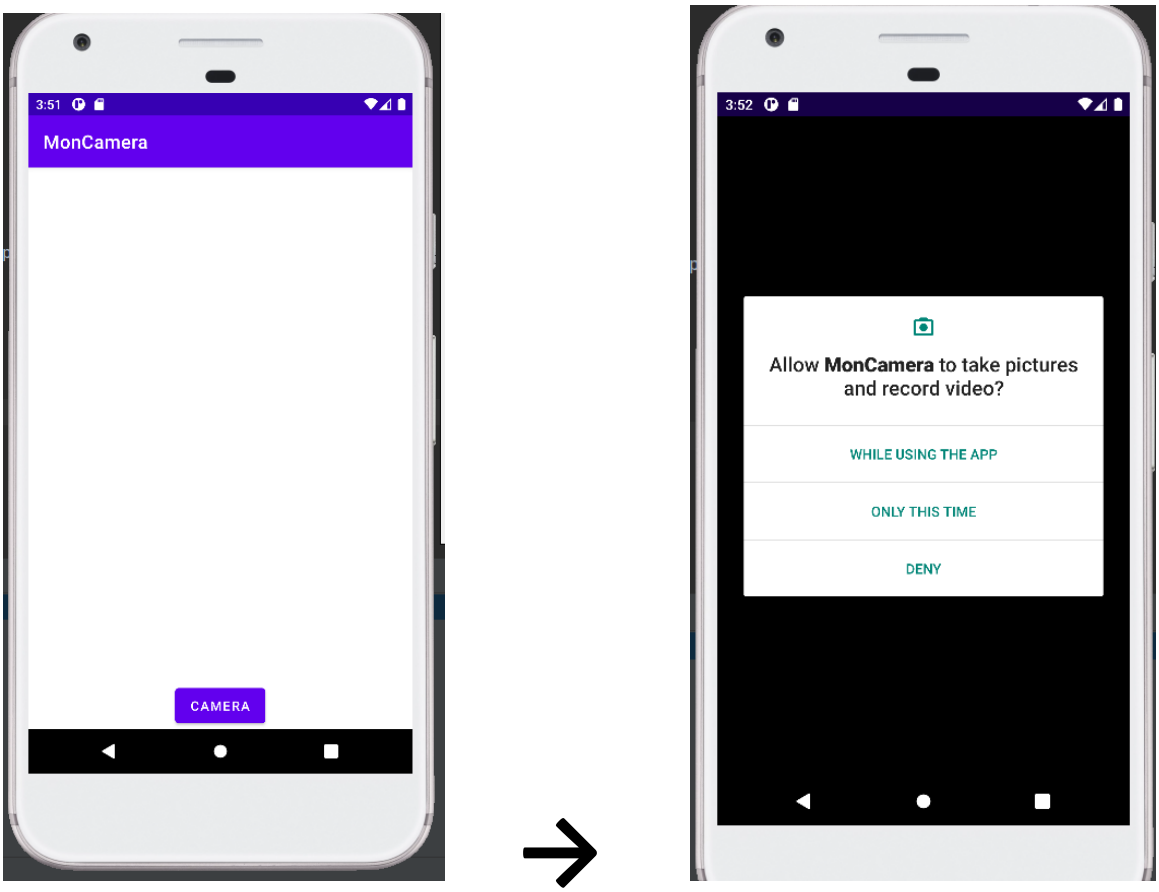
On choisit l'appareil désirer et on installe la '' **release name R ''** et on clique sur **Finish**

Ensuite, on clique sur **RUN** situé près de la nouvelle appreille configuré



On atteint un peu pour que l'application se construit. Après l'appareil virtuel s'ouvre dont notre application est ouverte et on clique sur le bouton **CAMERA.**

Pour la première fois de l'ouverture de l'application seulement, elle va demander l'autorisation de l'utilisation de la caméra.

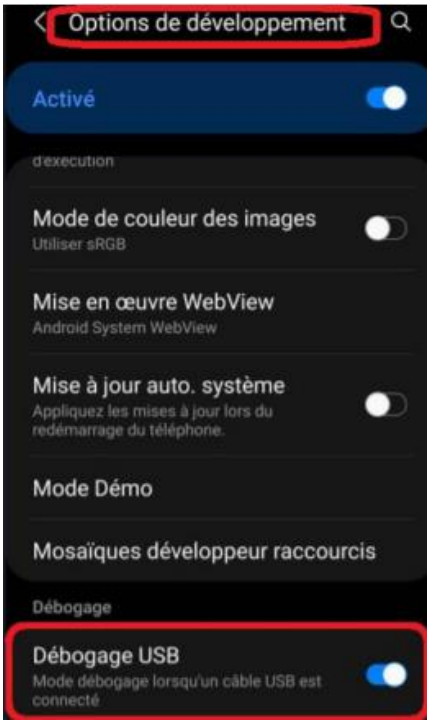


Solution 2 :

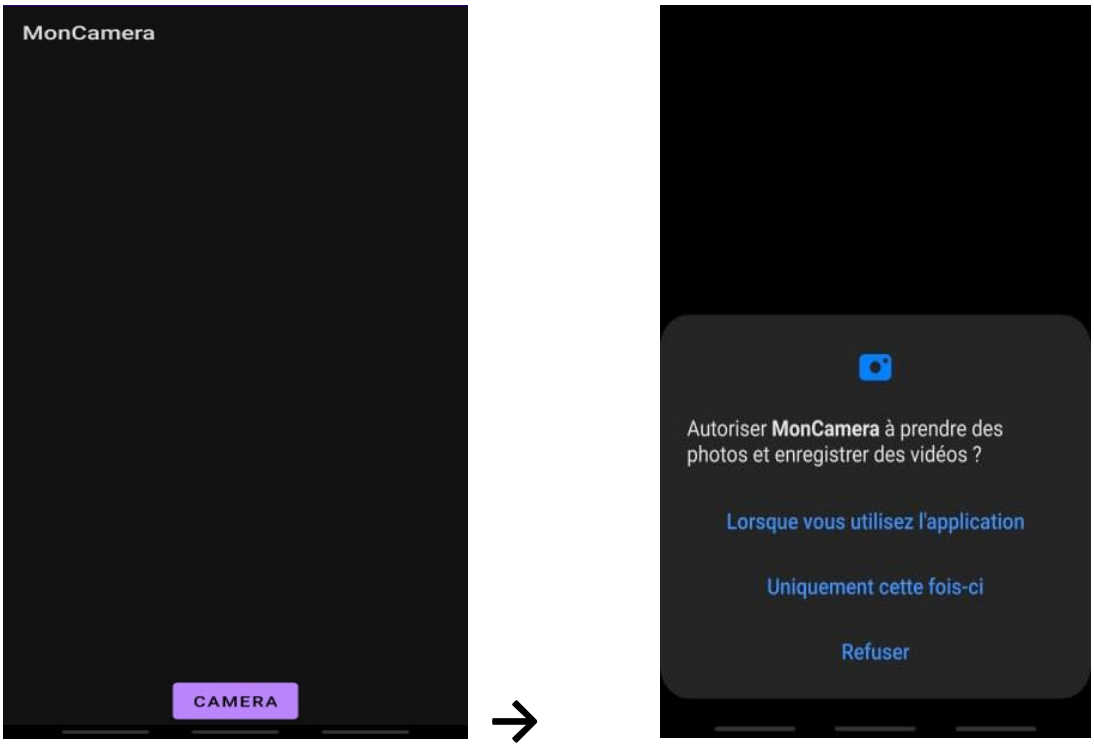
Si on désire d'installer l'application sur le smartphone :

Avant d'installer l'application sur le smartphone, il faut activer l'option ''Débogage USB'' En fait, Le Débogage USB est une option cachée d'Android permettant d'accéder à des options supplémentaires lorsque l'appareil est connecté à un ordinateur. Android dispose d'une option, Débogage USB, qui est utile l'indique pour les développeurs afin de déboguer une application. Cette option permet également de débloquer de nouvelles fonctions pour les bidouilleurs, et notamment l'accès aux outils ADB. Pour activer cette option, aller aux paramètres du téléphone > A propos du téléphone > Informations sur le logiciel et tapez 7 fois sur Numéro de version.

Une ‘’options de développement’’ s’ajoute. On l’ouvre et on active l’option ‘Débogage USB’



Ensuite, on connecte le téléphone avec le PC via son câble de connexion. On lance le programme en cliquant sur **RUN** et en atteint l’application installe sur le téléphone. Elle s’ouvre automatiquement dès que l’installation est terminée.



Par : Mohamed Amine KHARRAT

Et voila 😊