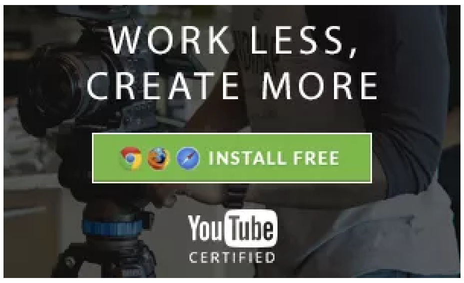


API

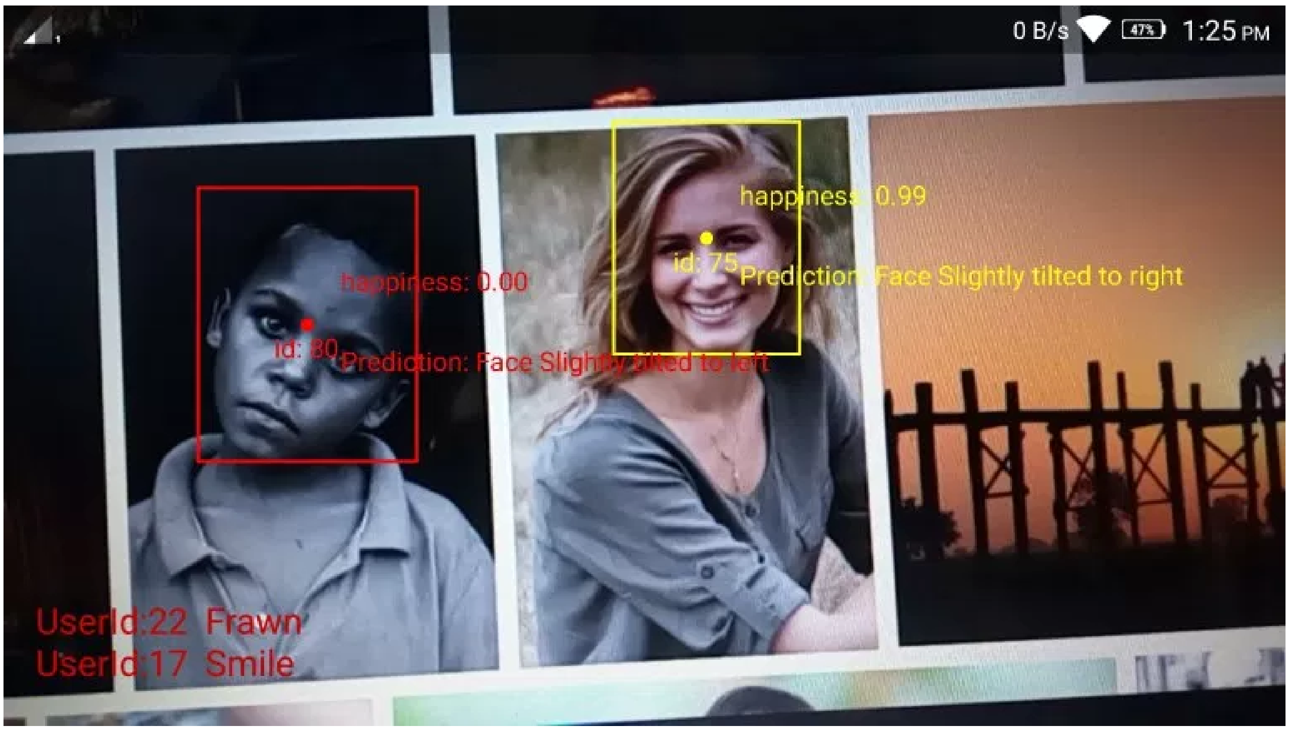
8 BY MYTRENDIN.COM

tutorial, we will do an example of Mobile Vision using Android Studio.If you are not familiar with Mobile Vision in Android Development, then you can check out our previous blog on Read Text

e Reader using Mobile Vision .In this example, we will make an app which will
ımera and at the same time make the prediction about the detected faces.
letection of faces. This implementation requires the use of Google Mobile
ervices library. This example can work offline and require the Android phone

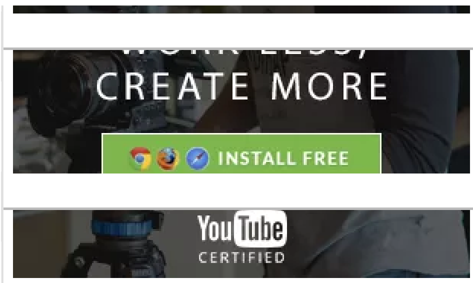
code

Want to Learn Advanced Android Application development from scratch- Beyond Basics



demo

Android Real Time Face Detection and Prediction using Google Mobile Vision API



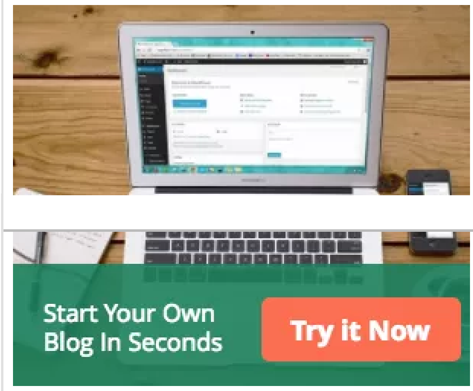
Java

```
s:play-services-vision:11.0.1'
```

Java

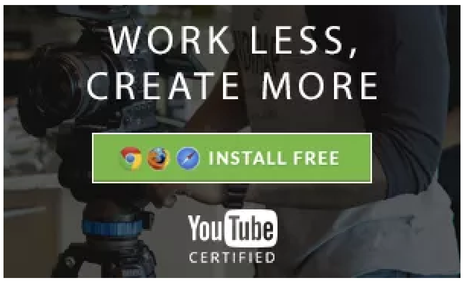
```
esign:25.3.1'
```

ary version should be same app compact version number.



Java

```
"utf-8"?>

.android.com/apk/res/android"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:keepScreenOn="true">

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:id="@+id/topLayout"

android:orientation="vertical"

android:layout_width="match_parent"

android:layout_height="match_parent">

<com.mytrendin.facetracking.CameraSourcePreview

android:id="@+id/preview"

android:layout_width="match_parent"

android:layout_height="match_parent">

<com.mytrendin.facetracking.GraphicOverlay

android:id="@+id/faceOverlay"

android:layout_width="match_parent"

android:layout_height="match_parent" />

```



```
                20dp"

                arent"


                ="true"

android:layout_alignParentBottom="true">
```

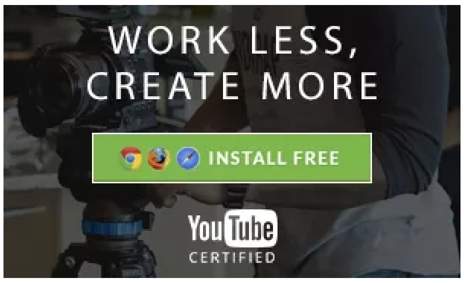
```
                olor/holo_red_dark"

                oid:textAppearanceMedium"

                arent"

                parent"

android:text="Hello" />

</ScrollView>

</RelativeLayout>
```

Here we are creating **CameraSourePreview** as a view group for displaying camera video images and **GraphicOverlay** as View for graphic content.we have created ScrollView for displaying updates that we get through prediction and displaying that here with user Id that is assigned by the tracker. We have used KeepScreenOn property to keep screen On. Rest content of the XML file is self-explanatory but in a case of any queries feel free to comment below.

Creating Landscape layout for above layout

Here, we have changed the orientation and rest are same as above.

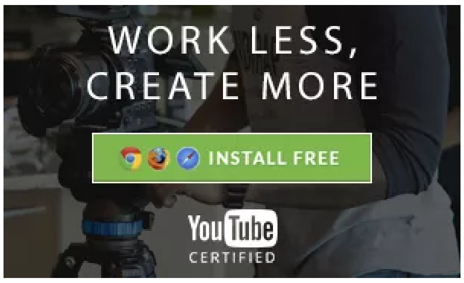
Java



```
arent"

parent"



ameraSourcePreview


android:layout_width="match_parent"
```

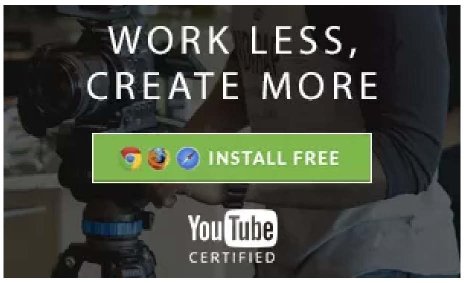
```
parent">

raphicOverlay


arent"
parent" />

CameraSourcePreview>
```

```
<ScrollView

android:id="@+id/scrollView"

android:paddingLeft="16dp"

android:paddingRight="16dp"

android:layout_marginBottom="20dp"

android:layout_width="match_parent"

android:layout_height="40dp"

android:layout_alignParentEnd="true"

android:layout_alignParentBottom="true">

<TextView

android:id="@+id/faceUpdates"

android:textColor="@android:color/holo_red_dark"

android:textAppearance="?android:textAppearanceMedium"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:text="Hello" />

</ScrollView>

</RelativeLayout>
```



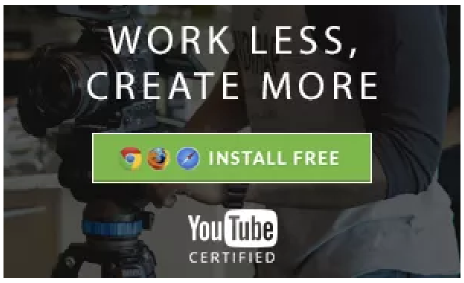
```
og;

xt;
gInterface;
import android.content.pm.PackageManager;
```

```
n.widget.Snackbar;
p.ActivityCompat;
p.AppCompatActivity;

ew;

import com.google.android.gms.common.ConnectionResult;

import com.google.android.gms.common.GoogleApiAvailability;

import com.google.android.gms.vision.CameraSource;

import com.google.android.gms.vision.MultiProcessor;

import com.google.android.gms.vision.Tracker;

import com.google.android.gms.vision.face.Face;

import com.google.android.gms.vision.face.FaceDetector;

import java.io.IOException;

public final class FaceTrackerActivity extends AppCompatActivity {

private static final String TAG = "FaceTracker";

private CameraSource mCameraSource = null;

private CameraSourcePreview mPreview;

private GraphicOverlay mGraphicOverlay;

private TextView mUpdates;

private static final int RC_HANDLE_GMS = 9001;

private static final int RC_HANDLE_CAMERA_PERM = 2;


@Override
```



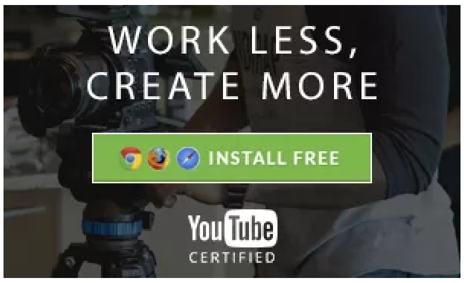
```
ewById(R.id.faceUpdates);
kSelfPermission(this, Manifest.permission.CAMERA);
MISSION_GRANTED) {
```

```
rmission() {
n is not granted. Requesting permission");
 new String[]{Manifest.permission.CAMERA};
owRequestPermissionRationale(this,
Manifest.permission.CAMERA)) {

ActivityCompat.requestPermissions(this, permissions, RC_HANDLE_CAMERA_PERM);

return;

}

final Activity thisActivity = this;

View.OnClickListener listener = new View.OnClickListener() {

@Override

public void onClick(View view) {

ActivityCompat.requestPermissions(thisActivity, permissions,

RC_HANDLE_CAMERA_PERM);

}

};

Snackbar.make(mGraphicOverlay, R.string.permission_camera_rationale,

Snackbar.LENGTH_INDEFINITE)

.setAction(R.string.ok, listener)

.show();

}
```



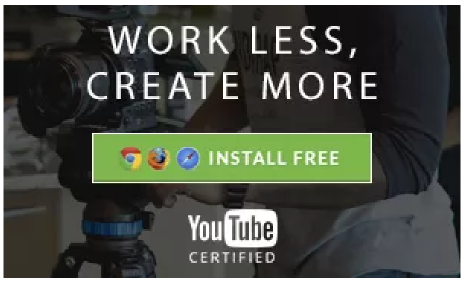
```
(new GraphicFaceTrackerFactory())
)) {
)
.setMessage("Face detector dependencies are not yet available.")
```

```
pendencies are not yet available.");
urce.Builder(context, detector)
4, 720)
.setFacing(CameraSource.CAMERA_FACING_BACK)
.setRequestedFps(30.0f)
.setAutoFocusEnabled(true)
.build();
}
/**
* Restarts the camera.
*/
@Override
protected void onResume() {
super.onResume();
startCameraSource();
}
/**
* Stops the camera.
*/
@Override
protected void onPause() {
```



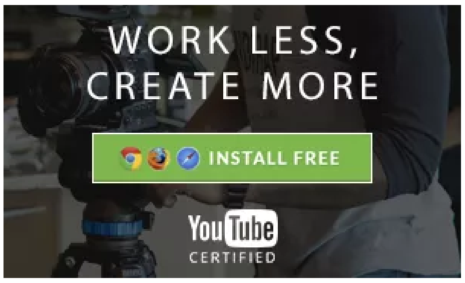
```
eline.
```

```
ionsResult(int requestCode, String[] permissions, int[] grantR
_CAMERA_PERM) {
Log.d(TAG, "Got unexpected permission result: " + requestCode);
super.onRequestPermissionsResult(requestCode, permissions, grantResults);
return;
}
if (grantResults.length != 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
Log.d(TAG, "Camera permission granted - initialize the camera source");
// we have permission, so create the camerasource
createCameraSource();
return;
}
Log.e(TAG, "Permission not granted: results len = " + grantResults.length +
" Result code = " + (grantResults.length > 0 ? grantResults[0] : "(empty)"));
DialogInterface.OnClickListener listener = new DialogInterface.OnClickListener() {
public void onClick(DialogInterface dialog, int id) {
finish();
}
};
```



```
ew
ce() {
 play services available.
lity.getInstance().isGooglePlayServicesAvailable(
getApplicationContext());
```

```
.SUCCESS) {
tance().getErrorDialog(this, code, RC_HANDLE_GMS);
try {

mPreview.start(mCameraSource, mGraphicOverlay);

} catch (IOException e) {

Log.e(TAG, "Unable to start camera source.", e);

mCameraSource.release();

mCameraSource = null;

}

}

}

//Graphic Face Tracker

private class GraphicFaceTrackerFactory implements MultiProcessor.Factory<Face> {

@Override

public Tracker<Face> create(Face face) {

return new GraphicFaceTracker(mGraphicOverlay,FaceTrackerActivity.this);

}

}

private class GraphicFaceTracker extends Tracker<Face> {


private GraphicOverlay mOverlay;
```

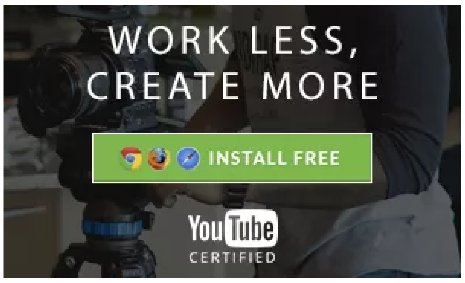


```
ceId, Face item) {


ector.Detections<Face> detectionResults, Face face) {

);


tector.Detections<Face> detectionResults) {

);

}

@Override

public void onDone() {

mOverlay.remove(mFaceGraphic);

}

}

}
```

First, we set our layout by writing

```
setContentView(R.layout.activity_face_tracker);
```

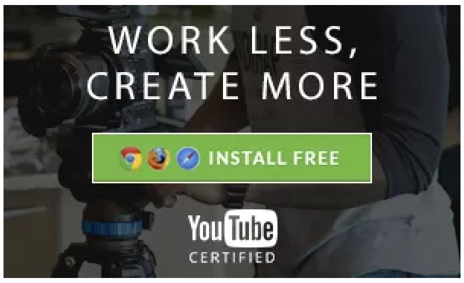
After setting the layout, before we could start the activity we need to check for camera permission.If camera Permission is granted by the user then we can start creating a camera and start to display on the screen.If not we need to request the permission and then start the camera.



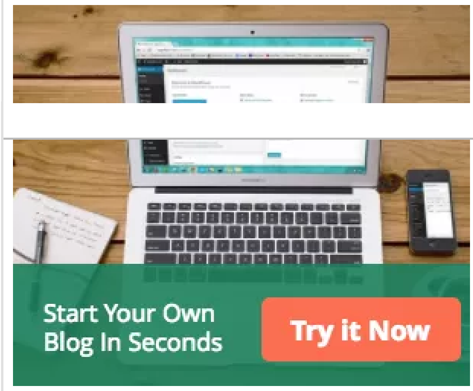
task of creating and initializing the **FaceDetector** object and providing the
amera Source to fetch the photo from the camera and feed it to the detector
rd this detected faces to the processor which process the faces and find the
iling, etc.

```java
ce() {
ionContext();
aceDetector.Builder(context)
tector.ALL_CLASSIFICATIONS)
(new GraphicFaceTrackerFactory())

.build());

if (!detector.isOperational()) {

new AlertDialog.Builder(this)

.setMessage("Face detector dependencies are not yet available.")

.show();

Log.w(TAG, "Face detector dependencies are not yet available.");

return;

}

mCameraSource = new CameraSource.Builder(context, detector)

.setRequestedPreviewSize(1024, 720)

.setFacing(CameraSource.CAMERA_FACING_BACK)

.setRequestedFps(30.0f)

.setAutoFocusEnabled(true)

.build();

}
```

GraphicFaceTrackerFactory creates a face tracker and then link it to the newly created face. So each face is assigned a tracker throughout detection process.

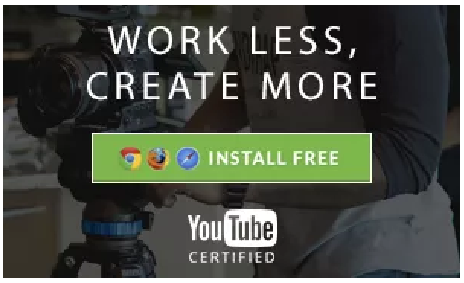


a face graphic with associated face overlay. when a new face is created it roughout Android Real-Time Face Detection. Whenever a face position is and maintains the information. When tracker misses the face then it hides the tected. When the face is assumed to be gone then remove the graphics annotation from the overlay.



Java

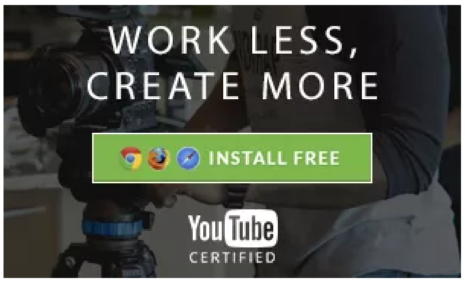
```
ker extends Tracker<Face> {

ay;

hic;

rlay overlay,Context context) {

mOverlay = overlay;

mFaceGraphic = new FaceGraphic(overlay,context);

}

/**

* Start tracking the detected face instance within the face overlay.

*/

@Override

public void onNewItem(int faceId, Face item) {

mFaceGraphic.setId(faceId);

}

/**

* Update the position/characteristics of the face within the overlay.

*/

@Override

public void onUpdate(FaceDetector.Detections<Face> detectionResults, Face face) {

mOverlay.add(mFaceGraphic);

mFaceGraphic.updateFace(face);

}

```



```
ector.Detections<Face> detectionResults) {
;
 * Called when the face is assumed to be gone for good. Remove the graphic annotation from
;
}
```

After this, It starts the camera Source to see the graphics drawn over the detected face on video images fetched from the Camera API.It starts or restarts the camera source if it exists. If camera source doesn't exist then it will be called after the camera source is created. It first checks whether play services are available or not. If not available it will show the error else it will proceed to camera source to display video image.If an exception is raised then camera resource is released.

Java



```
ance().getErrorDialog(this, code, RC_HANDLE_GMS);

try {
```

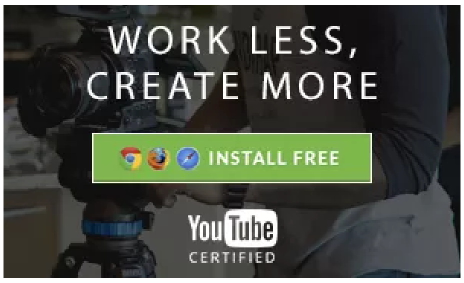
```
 mGraphicOverlay);

amera source.", e);

}

}
```

Related:
Detecting Swipe Gestures Android Tutorial
Firebase Authentication using facebook login in android
Create Key Hash for Facebook app using Android Studio
Encryption using Java Android Cryptography API
Image Recognition Using Google’s API
Read Text using Mobile Vision Text Recognization API in android

CameraSourcePreview.java

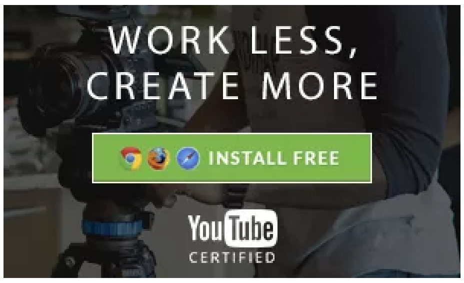
This file creates view group that could be used to handle different GraphicOverlay view. Before it set any overlay, it checks whether camera source is created or not and also whether surface Holder callback is available or not and when both are available then it starts the camera and set the overlay. It also checks the camera preview size so that proper view is selected for displaying graphics.

Java
```java
package com.mytrendin.facetracking;

import android.content.Context;

import android.content.res.Configuration;

```



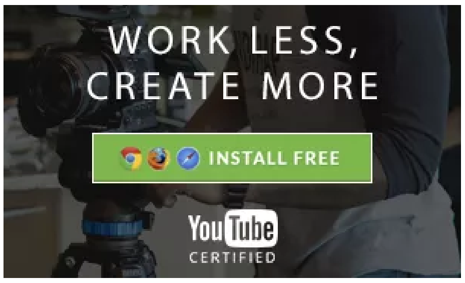
```
s.common.images.Size;

s.vision.CameraSource;


view extends ViewGroup {

TAG = "CameraSourcePreview";

private Context mContext;
```

```
View;

ted;

lable;

Source;

lay;

ontext context, AttributeSet attrs) {

super(context, attrs);

mContext = context;

mStartRequested = false;

mSurfaceAvailable = false;

mSurfaceView = new SurfaceView(context);

mSurfaceView.getHolder().addCallback(new SurfaceCallback());

addView(mSurfaceView);

}

public void start(CameraSource cameraSource) throws IOException {

if (cameraSource == null) {

stop();

}

mCameraSource = cameraSource;

if (mCameraSource != null) {

mStartRequested = true;

startIfReady();

}

}
```



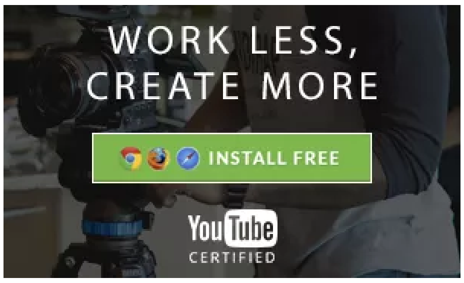
```
public void release() {
```

```
throws IOException {
aceAvailable) {
mCameraSource.start(mSurfaceView.getHolder());
if (mOverlay != null) {
Size size = mCameraSource.getPreviewSize();
int min = Math.min(size.getWidth(), size.getHeight());
int max = Math.max(size.getWidth(), size.getHeight());
if (isPortraitMode()) {
// Swap width and height sizes when in portrait, since it will be rotated by
// 90 degrees
mOverlay.setCameraInfo(min, max, mCameraSource.getCameraFacing());
} else {
mOverlay.setCameraInfo(max, min, mCameraSource.getCameraFacing());
}
mOverlay.clear();
}
mStartRequested = false;
}
}
```



```
camera source.", e);

@Override
(SurfaceHolder surface) {
```

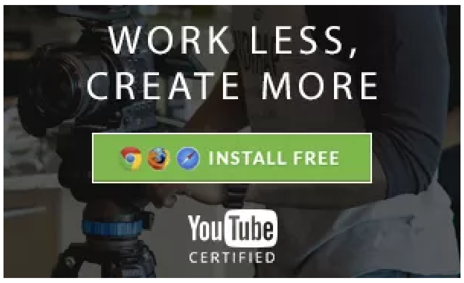
```
urfaceHolder holder, int format, int width, int height) {

}

@Override

protected void onLayout(boolean changed, int left, int top, int right, int bottom) {

int width = 320;

int height = 240;

if (mCameraSource != null) {

Size size = mCameraSource.getPreviewSize();

if (size != null) {

width = size.getWidth();

height = size.getHeight();

}

}

// Swap width and height sizes when in portrait, since it will be rotated 90 degrees

if (isPortraitMode()) {

int tmp = width;

width = height;

height = tmp;

}
```



```
ng fit width, does fit height instead.
ht) {

layoutHeight / (float) height) * width);
}

Count(); ++i) {
hildWidth, childHeight);
```

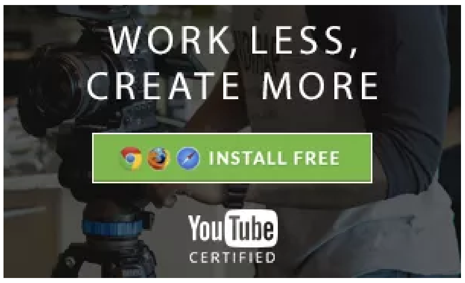
```
camera source.", e);

}

}

private boolean isPortraitMode() {

int orientation = mContext.getResources().getConfiguration().orientation;

if (orientation == Configuration.ORIENTATION_LANDSCAPE) {

return false;

}

if (orientation == Configuration.ORIENTATION_PORTRAIT) {

return true;

}

Log.d(TAG, "isPortraitMode returning false by default");

return false;

}

}
```

GraphicOverlay.java



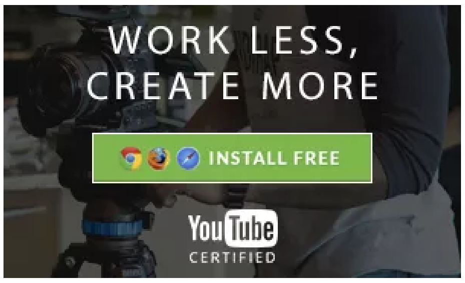
```
as;

eSet;


s.vision.CameraSource;



import java.util.Set;
```

```
extends View {

 new Object();


tor = 1.0f;


ctor = 1.0f;

private int mFacing = CameraSource.CAMERA_FACING_BACK;

private Set<Graphic> mGraphics = new HashSet<>();

public static abstract class Graphic {

private GraphicOverlay mOverlay;

public Graphic(GraphicOverlay overlay) {

mOverlay = overlay;

}

public abstract void draw(Canvas canvas);

/**

* Adjusts a horizontal value of the supplied value from the preview scale to the view

* scale.

*/

public float scaleX(float horizontal) {

return horizontal * mOverlay.mWidthScaleFactor;

}

/**

* Adjusts a vertical value of the supplied value from the preview scale to the view scale.

*/
```



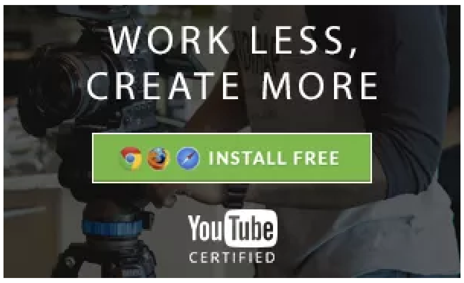
```
t x) {
raSource.CAMERA_FACING_FRONT) {
 scaleX(x);
```

```
rom the preview's coordinate system to the view coordinate
*/
public float translateY(float y) {
return scaleY(y);
}
public void postInvalidate() {
mOverlay.postInvalidate();
}
}
public GraphicOverlay(Context context, AttributeSet attrs) {
super(context, attrs);
}
/**
* Removes all graphics from the overlay.
*/
public void clear() {
synchronized (mLock) {
mGraphics.clear();
```



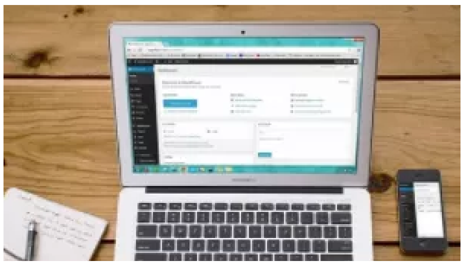
```
hic) {
postInvalidate();
```

```
overlay.
raphic) {
mGraphics.remove(graphic);

}

postInvalidate();

}

public void setCameraInfo(int previewWidth, int previewHeight, int facing) {

synchronized (mLock) {

mPreviewWidth = previewWidth;

mPreviewHeight = previewHeight;

mFacing = facing;

}

postInvalidate();

}

/**

* Draws the overlay with its associated graphic objects.

*/

@Override

protected void onDraw(Canvas canvas) {

super.onDraw(canvas);
```



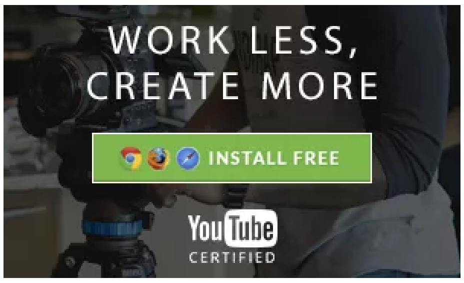
```
hics) {
```

```
}
```



Java

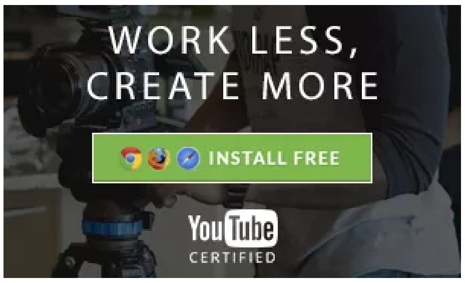
```
acking;

import android.content.Context;

import android.graphics.Canvas;

import android.graphics.Color;

import android.graphics.Paint;

import android.widget.ScrollView;

import android.widget.TextView;

import com.google.android.gms.vision.face.Face;

/**

* Graphic instance for rendering face position, orientation, and landmarks within an assoc

* graphic overlay view.

*/

class FaceGraphic extends GraphicOverlay.Graphic {

private static final float FACE_POSITION_RADIUS = 10.0f;

private static final float ID_TEXT_SIZE = 40.0f;

private static final float ID_Y_OFFSET = 50.0f;

private static final float ID_X_OFFSET = -50.0f;

private static final float BOX_STROKE_WIDTH = 5.0f;

```



```
};
```

```
olorIndex = 0;

aint;
```

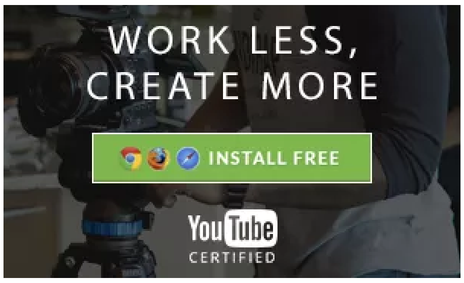
```
FaceGraphic(GraphicOverlay overlay, Context context) {

super(overlay);

mContext=context;

mCurrentColorIndex = (mCurrentColorIndex + 1) % COLOR_CHOICES.length;

final int selectedColor = COLOR_CHOICES[mCurrentColorIndex];

mFacePositionPaint = new Paint();

mFacePositionPaint.setColor(selectedColor);

mIdPaint = new Paint();

mIdPaint.setColor(selectedColor);

mIdPaint.setTextSize(ID_TEXT_SIZE);

mBoxPaint = new Paint();

mBoxPaint.setColor(selectedColor);

mBoxPaint.setStyle(Paint.Style.STROKE);

mBoxPaint.setStrokeWidth(BOX_STROKE_WIDTH);

}

void setId(int id) {

mFaceId = id;
```



```
for position on the supplied canvas.
```

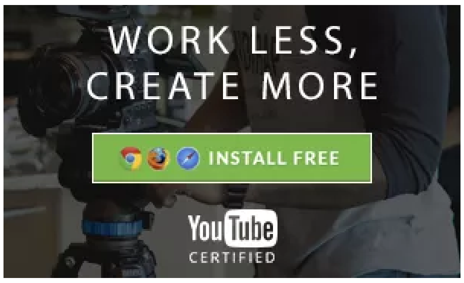
```
as) {
```

```
}

// Draws a circle at the position of the detected face, with the face's track id below.

float x = translateX(face.getPosition().x + face.getWidth() / 2);

float y = translateY(face.getPosition().y + face.getHeight() / 2);

canvas.drawCircle(x, y, FACE_POSITION_RADIUS, mFacePositionPaint);

canvas.drawText("id: " + mFaceId, x + ID_X_OFFSET, y + ID_Y_OFFSET, mIdPaint);

canvas.drawText("happiness: " + String.format("%.2f", face.getIsSmilingProbability()), x -

String prediction = getPrediction(face.getEulerY(),face.getEulerZ());

canvas.drawText("Prediction: "+prediction,x-ID_X_OFFSET,y-ID_Y_OFFSET+3*ID_TEXT_SIZE,mIdPa

// Draws a bounding box around the face.

float xOffset = scaleX(face.getWidth() / 2.0f);

float yOffset = scaleY(face.getHeight() / 2.0f);

float left = x - xOffset;

float top = y - yOffset;

float right = x + xOffset;

float bottom = y + yOffset;

canvas.drawRect(left, top, right, bottom, mBoxPaint);

```



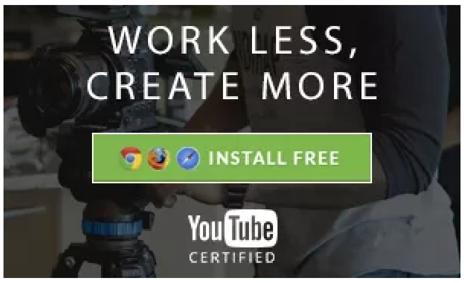
```
ubstring(len-30,len);
e)){
line);
}
```

```
line);
=(ScrollView)((Activity)mContext).findViewById(R.id.scrollView
Runnable() {
public void run() {

mScrollView.fullScroll(ScrollView.FOCUS_DOWN);

}

}, 600);

}

private String getPrediction(float eulerY, float eulerZ) {

String feature="";

if(eulerZ<5f && eulerZ >=0f){

if(eulerY>0f && eulerY<60f){

feature="Facing straight right";

}else{

feature="no tilt";

}

}else if(eulerZ>5f && eulerZ<45f){

if(eulerY>0f && eulerY<=60f){

feature="facing slightly right up";

}else {


feature="Face Slightly tilted to right";
```



```
t";

 >-5f){

{
feature="Facing right";
```

```
Z>-45f){

{

feature="Face Slightly tilted to left";

}

}else{

if(eulerY>-6f && eulerY!=0){

feature="Facing Left up";

}else{

feature="Face tilted to left";

}

}

return feature;

}

private String getUpdates(){

String update;

boolean smiling = mFace.getIsSmilingProbability() > SMILING_PROB_THRESHOLD;

boolean leftEyeClosed = mFace.getIsLeftEyeOpenProbability() < EYE_OPEN_PROB_THRESHOLD;

boolean rightEyeClosed = mFace.getIsRightEyeOpenProbability() < EYE_OPEN_PROB_THRESHOLD;

if(smiling) {

if(leftEyeClosed && rightEyeClosed) {
```



```
if (leftEyeClosed && !rightEyeClosed) {
```

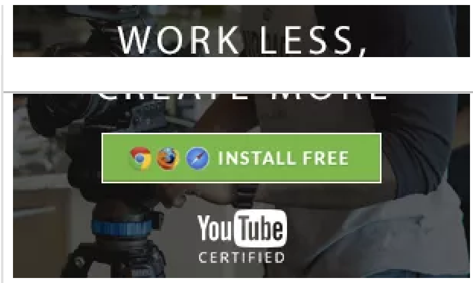
```
!leftEyeClosed){
update = "Frawn";

}

}

return update;

}

}
```

The face graphics class extends the **GraphicOverlay** class and implements all its method. This class renders the face position, orientation, and landmarks with associated graphic overlay view. In this class, we are creating the set of colors that could be used for assigning colors to the bounding box and text color for individual faces. The face graphics constructor takes overlay and context as parameter input.

In this constructor, we create Paint Object and assign some properties like color, text size, stroke. For each face, their Id is stored for further processing. Trigger post invalidate method to redraw the detected new face along with other. Draw method which was abstract in the parent class is now implemented. Draw method calculates the detected face's x and y coordinate and using that coordinate it calculates coordinate for drawing items on the canvas.Here we get the happiness parameter from the getIsSmilingProbability() method of Face Object.To make a prediction whether a person is smiling or frowning we use a get updates method which uses the computed values of smiling, left eye closed, right eye closed by using Android Real Time Face Detection.

Java

get the answer



Java

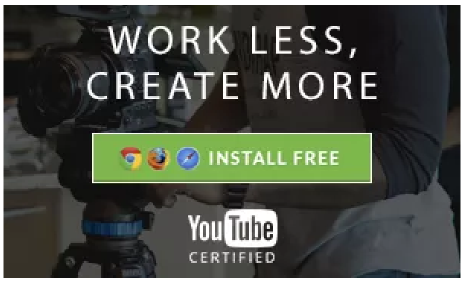
```java
eClosed) {

} else if(rightEyeClosed && !leftEyeClosed){
```

```java
if (leftEyeClosed && !rightEyeClosed) {

update = "Left Wink Frawn";

} else if(rightEyeClosed && !leftEyeClosed){

update = "Right Wink Frawn";

} else if (leftEyeClosed){

update = "Closed Eye Frawn";

} else {

update = "Frawn";

}

}
```

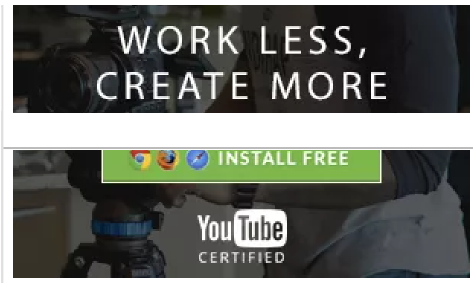
To make a prediction where person's face is like whether he is looking in left, right or in the forward direction to the camera. For this, we implement getPrediction method. It uses EulerY, EulerZ parameter from the face and using this two parameter nine different possibilities occur. But here we will not implement all the nine possibilities.

Java

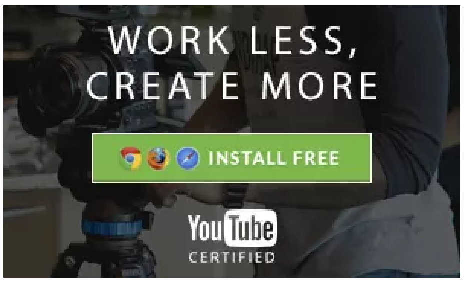
```java
private String getPrediction(float eulerY, float eulerZ) {

String feature="";

if(eulerZ<5f && eulerZ >=0f){

```



```
45f){

t up";

 to right";

}

";

}else if(eulerZ<0f && eulerZ >-5f){

if(eulerY>-60f && eulerY!=0){

feature="Facing right";

}else{

feature="no tilt";

}

}else if(eulerZ<-5f && eulerZ>-45f){

if(eulerY>-60f && eulerY!=0){

feature="Facing Left up";

}else{

feature="Face Slightly tilted to left";

}

}else{

if(eulerY>-6f && eulerY!=0){

feature="Facing Left up";

}else{

feature="Face tilted to left";

}
```


WORK LESS,
CREATE MORE
INSTALL FREE
YouTube
CERTIFIED

Start Your Own
Blog In Seconds
Try it Now



```
cher"

me"

_launcher_round"

android:supportsRtl="true"
```

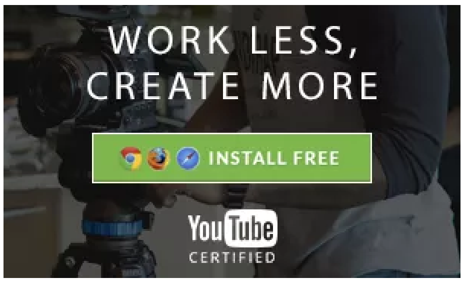
```
true"

ppCompat">

google.android.gms.version"

e_play_services_version"/>


oid.gms.vision.DEPENDENCIES"
android:value="face" />

<activity

android:screenOrientation="fullSensor"

android:theme="@style/Theme.AppCompat.NoActionBar.FullScreen"

android:name=".FaceTrackerActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>
```

we require camera permission and we have set hardwareAccelerated to true. We have used meta-data tag to mention the play service version and dependencies as the face.

Style.xml

Java

```
tentOverlay">@null</item>
```



ection), you saw how to use vision API to track multiple faces and make the
emoji over the detected face and that emoji will be based on the detected
, left wink, etc. It could also be possible to track single face instead multiple
tion process faster. All other things in this projecareself-explanatory. In the
. keep following for more amazing blogs. If you are a Beginner Learn Android

ibe to our YouTube Channel for videos related to this article.Please find us

Free Download is locked

We have something special for you, use one of the buttons below to free unlock!

like

+1 us

Start a blog & Earn Money Online!

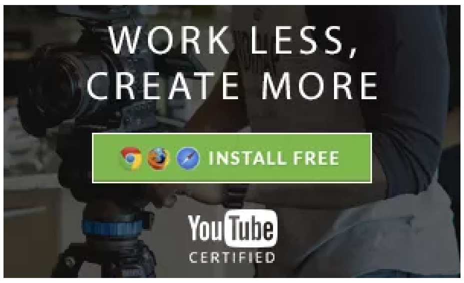
A blog is your digital portal to share your current passion online. It's text without ink stains.

GET STARTED NOW

Share this:

WhatsApp Facebook Reddit Twitter Tumblr Pinterest LinkedIn

Google Email Print



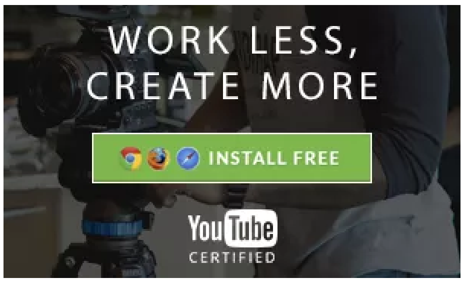
EMAIL

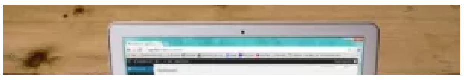
WEBSITE

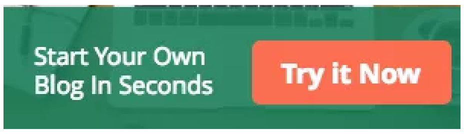
POST COMMENT

☐ NOTIFY ME OF FOLLOW-UP COMMENTS BY EMAIL.

☐ NOTIFY ME OF NEW POSTS BY EMAIL.


WORK LESS,
CREATE MORE
INSTALL FREE
YouTube
CERTIFIED




Start Your Own
Blog In Seconds
Try it Now