debido a no poder realizar la actividad anterior con la camara se hace un nuevo intento, esta vez con camera x

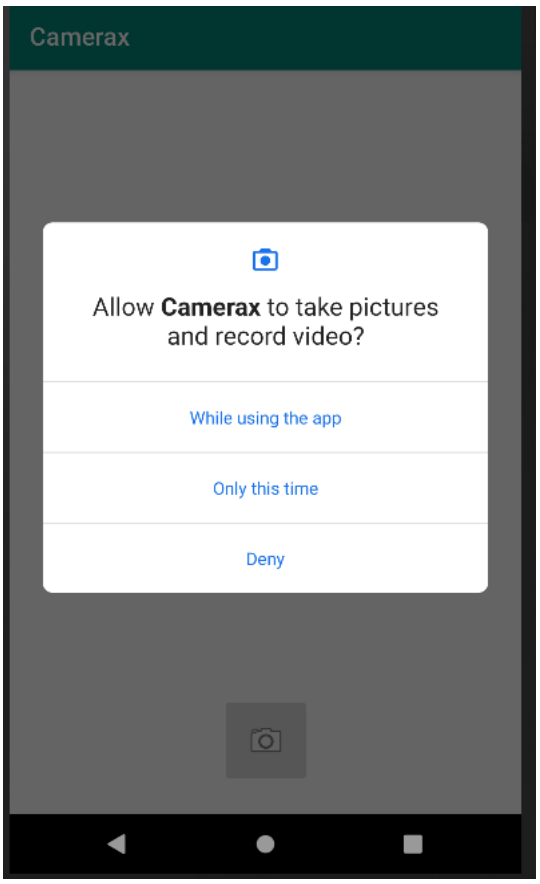
Main Activity

```
public class MainActivity extends AppCompatActivity {

   private int REQUEST_CODE_PERMISSIONS = 10; //arbitrary number, can be changed
accordingly
   private final String[] REQUIRED_PERMISSIONS = new
String[]{"android.permission.CAMERA","android.permission.WRITE_EXTERNAL_STORAG
E"}; //array w/ permissions from manifest
   TextureView txView;

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

      txView = findViewById(R.id.view_finder);

      if(allPermissionsGranted()){
         startCamera(); //start camera if permission has been granted by user
      } else{
         ActivityCompat.requestPermissions(this, REQUIRED_PERMISSIONS,
REQUEST_CODE_PERMISSIONS);
      }
   }

   private void startCamera() {
      //make sure there isn't another camera instance running before starting
      CameraX.unbindAll();

      /* start preview */
      int aspRatioW = txView.getWidth(); //get width of screen
      int aspRatioH = txView.getHeight(); //get height
      Rational asp = new Rational (aspRatioW, aspRatioH); //aspect ratio
      Size screen = new Size(aspRatioW, aspRatioH); //size of the screen

      //config obj for preview/viewfinder thingy.
      PreviewConfig pConfig = new
PreviewConfig.Builder().setTargetAspectRatio(asp).setTargetResolution(screen).build();
      Preview preview = new Preview(pConfig); //lets build it

      preview.setOnPreviewOutputUpdateListener(
            new Preview.OnPreviewOutputUpdateListener() {
               //to update the surface texture we have to destroy it first, then re-add it
               @Override
```

```
            public void onUpdated(Preview.PreviewOutput output){
                ViewGroup parent = (ViewGroup) txView.getParent();
                parent.removeView(txView);
                parent.addView(txView, 0);

                txView.setSurfaceTexture(output.getSurfaceTexture());
                updateTransform();
            }
        });

    /* image capture */

    //config obj, selected capture mode
    ImageCaptureConfig imgCapConfig = new
ImageCaptureConfig.Builder().setCaptureMode(ImageCapture.CaptureMode.MIN_LATENC
Y)
            .setTargetRotation(getWindowManager().getDefaultDisplay().getRotation()).build();
    final ImageCapture imgCap = new ImageCapture(imgCapConfig);

    findViewById(R.id.capture_button).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            File file = new File(Environment.getExternalStorageDirectory() + "/" +
System.currentTimeMillis() + ".jpg");
            imgCap.takePicture(file, new ImageCapture.OnImageSavedListener() {
                @Override
                public void onImageSaved(@NonNull File file) {
                    String msg = "Foto guardada " + file.getAbsolutePath();
                    Toast.makeText(getBaseContext(), msg,Toast.LENGTH_LONG).show();
                }

                @Override
                public void onError(@NonNull ImageCapture.UseCaseError useCaseError,
@NonNull String message, @Nullable Throwable cause) {
                    String msg = "Photo capture failed: " + message;
                    Toast.makeText(getBaseContext(), msg,Toast.LENGTH_LONG).show();
                    if(cause != null){
                        cause.printStackTrace();
                    }
                }
            });
        }
    });

    /* image analyser */
```

```
        ImageAnalysisConfig imgAConfig = new
ImageAnalysisConfig.Builder().setImageReaderMode(ImageAnalysis.ImageReaderMode.AC
QUIRE_LATEST_IMAGE).build();
        ImageAnalysis analysis = new ImageAnalysis(imgAConfig);

        analysis.setAnalyzer(
            new ImageAnalysis.Analyzer(){
                @Override
                public void analyze(ImageProxy image, int rotationDegrees){
                    //y'all can add code to analyse stuff here idek go wild.
                }
            });

        //bind to lifecycle:
        CameraX.bindToLifecycle((LifecycleOwner)this, analysis, imgCap, preview);
    }

    private void updateTransform(){
        /*
        * compensates the changes in orientation for the viewfinder, bc the rest of the layout
stays in portrait mode.
        * methinks :thonk:
        * imgCap does this already, this class can be commented out or be used to optimise the
preview
        */
        Matrix mx = new Matrix();
        float w = txView.getMeasuredWidth();
        float h = txView.getMeasuredHeight();

        float centreX = w / 2f; //calc centre of the viewfinder
        float centreY = h / 2f;

        int rotationDgr;
        int rotation = (int)txView.getRotation(); //cast to int bc switches don't like floats

        switch(rotation){ //correct output to account for display rotation
            case Surface.ROTATION_0:
                rotationDgr = 0;
                break;
            case Surface.ROTATION_90:
                rotationDgr = 90;
                break;
            case Surface.ROTATION_180:
                rotationDgr = 180;
                break;
            case Surface.ROTATION_270:
                rotationDgr = 270;
                break;
```

```
            default:
                return;
        }

        mx.postRotate((float)rotationDgr, centreX, centreY);
        txView.setTransform(mx); //apply transformations to textureview
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
        //start camera when permissions have been granted otherwise exit app
        if(requestCode == REQUEST_CODE_PERMISSIONS){
            if(allPermissionsGranted()){
                startCamera();
            } else{
                Toast.makeText(this, "Permissions not granted by the user.",
Toast.LENGTH_SHORT).show();
                finish();
            }
        }
    }

    private boolean allPermissionsGranted(){
        //check if req permissions have been granted
        for(String permission : REQUIRED_PERMISSIONS){
            if(ContextCompat.checkSelfPermission(this, permission) !=
PackageManager.PERMISSION_GRANTED){
                return false;
            }
        }
        return true;
    }
}
```


Camerax
Allow Camerax to take pictures and record video?
While using the app
Only this time
Deny


5:20
Camerax