Распознавание поз: Камасутра с CameraX

Денис Неклюдов aka **life_in_tech** @ tiktok

Goal for today is
to make a camera app
that can detect faces
and human bodies

Agenda

1. Create empty project
2. Request camera permission
3. Show camera preview
4. Add camera switch
5. Add face detection and preview scale
6. Add pose detection

Initialize the project

```
git clone git@github.com:nekdenis/camera_workshop.git

git checkout step_01_empty_project
```

Adding compose support into build.gradle of your module

```
implementation("androidx.compose.ui:ui:${rootProject.extra["compose_version"]}")
implementation("androidx.compose.material:material:${rootProject.extra["compose_version"]}")
implementation("androidx.compose.ui:ui-tooling:${rootProject.extra["compose_version"]}")
implementation("androidx.lifecycle:lifecycle-runtime-ktx:${rootProject.extra["lifecycle_version"]}")
implementation("androidx.activity:activity-compose:${rootProject.extra["activity_compose_version"]}")

buildFeatures {
    compose = true
}
composeOptions {
    kotlinCompilerExtensionVersion = rootProject.extra["compose_version"] as String
}
```

Adding CameraX support into build.gradle of your module

```
implementation("androidx.camera:camera-core:${rootProject.extra["camerax_version"]}")
implementation("androidx.camera:camera-camera2:${rootProject.extra["camerax_version"]}")
implementation("androidx.camera:camera-lifecycle:${rootProject.extra["camerax_version"]}")
implementation("androidx.camera:camera-view:${rootProject.extra["cameraview_version"]}")
```

Compose

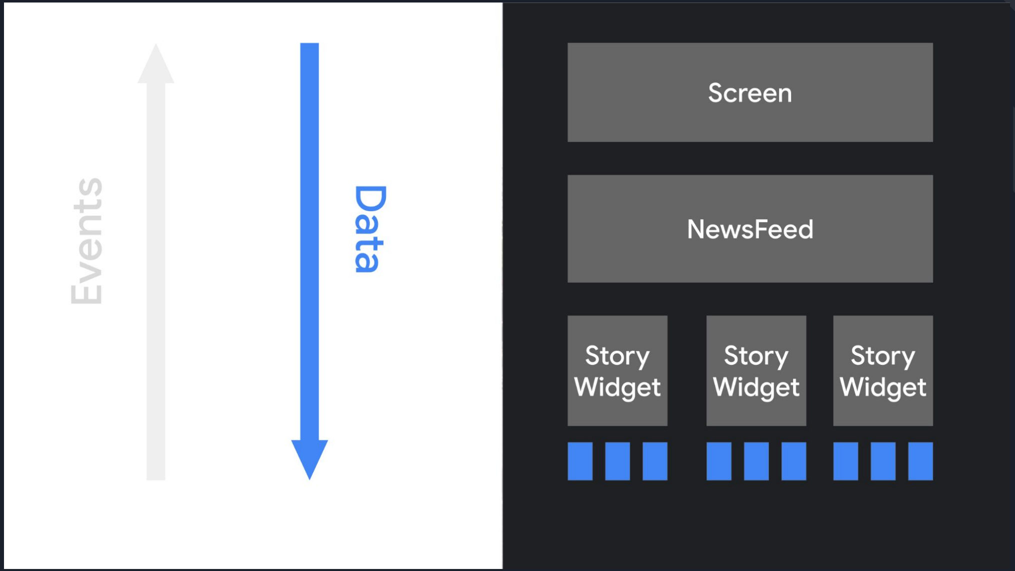
Everything is a function

State is a king

We don’t control occurrence of drawing

Unidirectional data flow

Unidirectional



Simple

```kotlin
@Composable
fun SwitchCamera(
    currentCamera: CameraState,
    switchCamera: () -> Unit
) {
    Button(onClick = switchCamera) {
        Text(currentCamera.name)
    }
}
```

Creating basic composable

```kotlin
fun setContent {
   Greeting("Android")
}

@Composable
fun Greeting(name: String) {
   Text(text = "Hello $name!")
}
```

Requesting camera permission

```
git clone git@github.com:nekdenis/camera_workshop.git

git checkout step_02_camera_permission
```

Adding permission to Manifest.xml

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

Checking permission

```
private fun permissionGranted() =
   ContextCompat.checkSelfPermission(this, Manifest.permission.CAMERA) ==
           PackageManager.PERMISSION_GRANTED
```

Requesting permission

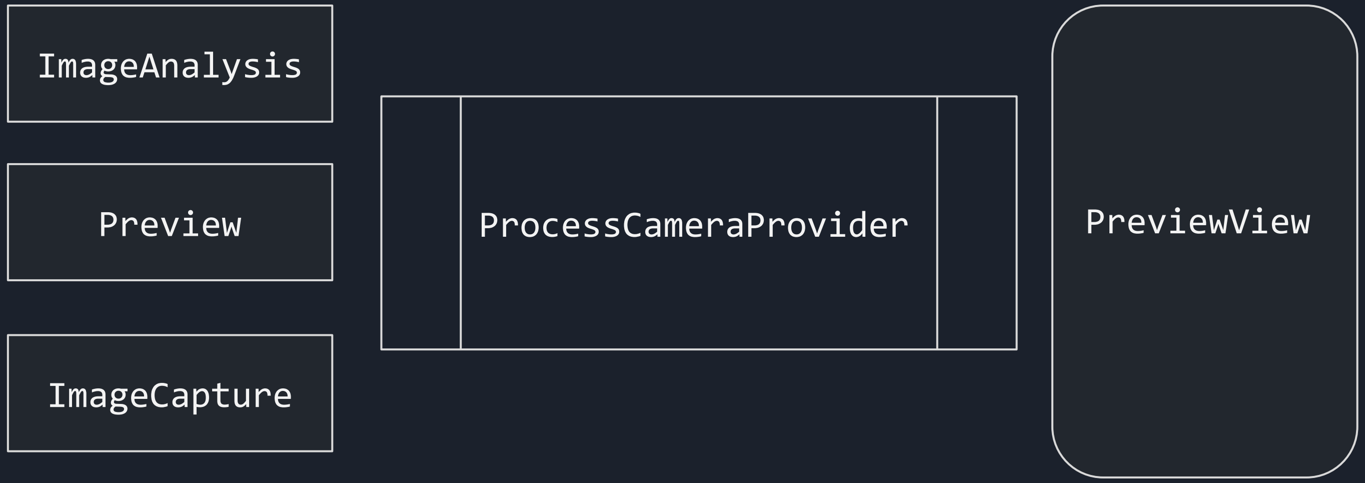
```kotlin
private fun requestPermission() {
   ActivityCompat.requestPermissions(
       this, arrayOf(Manifest.permission.CAMERA), 0
   )
}

override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<String?>,
grantResults: IntArray) {
   super.onRequestPermissionsResult(requestCode, permissions, grantResults)
   if (requestCode == 0) {
       if (grantResults[0] == PackageManager.PERMISSION_GRANTED) {
           initView()
       } else {
           Toast.makeText(this, "camera permission denied", Toast.LENGTH_LONG).show()
       }
   }
}
```

CameraX

Wrapper around Camera2 Android APIs

Simplifying it a lot

CameraX architecture



Compose remember(){}

A value computed by *remember* is stored in the Composition during initial composition, and the stored value is returned during recomposition.

Compose remember(key){}

When key changes lamba will be called again to compute new value

Showing camera preview

```
git clone git@github.com:nekdenis/camera_workshop.git

git checkout step_03_camera_preview
```

Camera composable

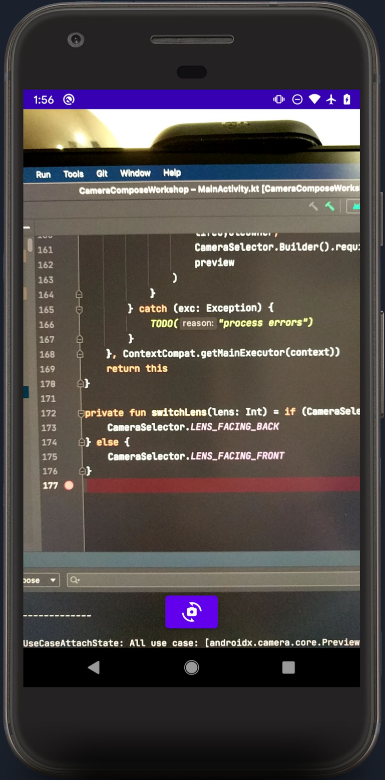
```kotlin
@Composable
fun CameraPreview() {

   val previewView = remember { PreviewView(context) }

   val cameraProviderFuture = remember {
       ProcessCameraProvider.getInstance(context)
           .configureCamera(...)
   }

    AndroidView(previewView)
}
```

Configure camera

```
private fun ListenableFuture<ProcessCameraProvider>.configureCamera(...) {

    addListener({

       val preview = androidx.camera.core.Preview.Builder()
            .build()
            .apply {
                setSurfaceProvider(previewView.surfaceProvider)
            }

        get().apply {
            unbindAll()
            bindToLifecycle(lifecycleOwner, cameraSelector, preview)
        }
    })
}
```

Adding switch button

```
git clone git@github.com:nekdenis/camera_workshop.git

git checkout step_04_camera_lens_switch
```



```
                                CameraSelector.Builder().requ
                                preview
                            )
                        }
                    } catch (exc: Exception) {
                        TODO( reason: "process errors")
                    }
                }, ContextCompat.getMainExecutor(context))
                return this
            }

            private fun switchLens(lens: Int) = if (CameraSele
                CameraSelector.LENS_FACING_BACK
            } else {
                CameraSelector.LENS_FACING_FRONT
            }
```

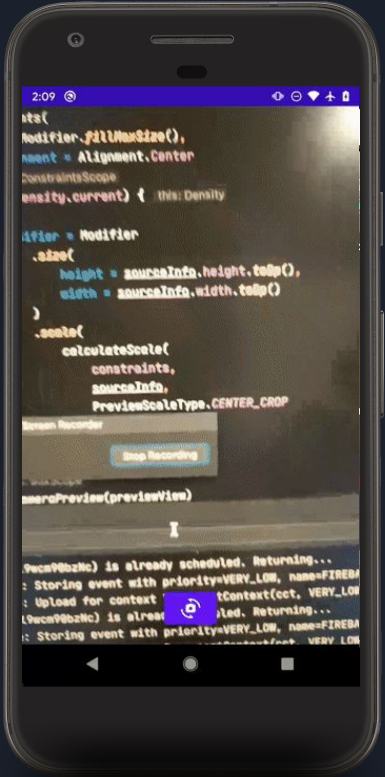
```
UseCaseAttachState: All use case: [androidx.camera.core.Preview
```

Configure camera

```
@Composable
fun Controls(
    onLensChange: () -> Unit
) {
    Box(
        modifier = Modifier.fillMaxSize(),
        contentAlignment = Alignment.BottomCenter,
    ) {
        Button(
            onClick = onLensChange
        ) { Icon(Icons.Filled.Cameraswitch, contentDescription = "Switch camera") }
    }
}
```

Adding face detection

```
git clone git@github.com:nekdenis/camera_workshop.git

git checkout step_05_face_detection
```



```
nts(
Modifier.fillMaxSize(),
nment = Alignment.Center
ConstraintsScope
ensity.current) {  this: Density

ifier = Modifier
 .size(
     height = sourceInfo.height.toDp(),
     width = sourceInfo.width.toDp()
 )
 .scale(
     calculateScale(
         constraints,
         sourceInfo,
         PreviewScaleType.CENTER_CROP

ameraPreview(previewView)
```

```
9wcm90bzNc) is already scheduled. Returning...
: Storing event with priority=VERY_LOW, name=FIREBA
: Upload for context              tContext(cct, VERY_LOW
l9wcm90bzNc) is alrea          led. Returning...
: Storing event with priority=VERY_LOW, name=FIREBA
```

Google MLKit

Simplest way to apply machine learning on mobile devices

Various use cases, such as Face Detection, Pose Detection, Smart Replies, Selfie Segmentation and more

Does not require a connection to the internet

Adding MLKit dependencies

```
implementation("com.google.mlkit:face-detection:16.0.6")

implementation("com.google.android.gms:play-services-mlkit-face-detection:16.1.5")
```

Wrapping library’s processor

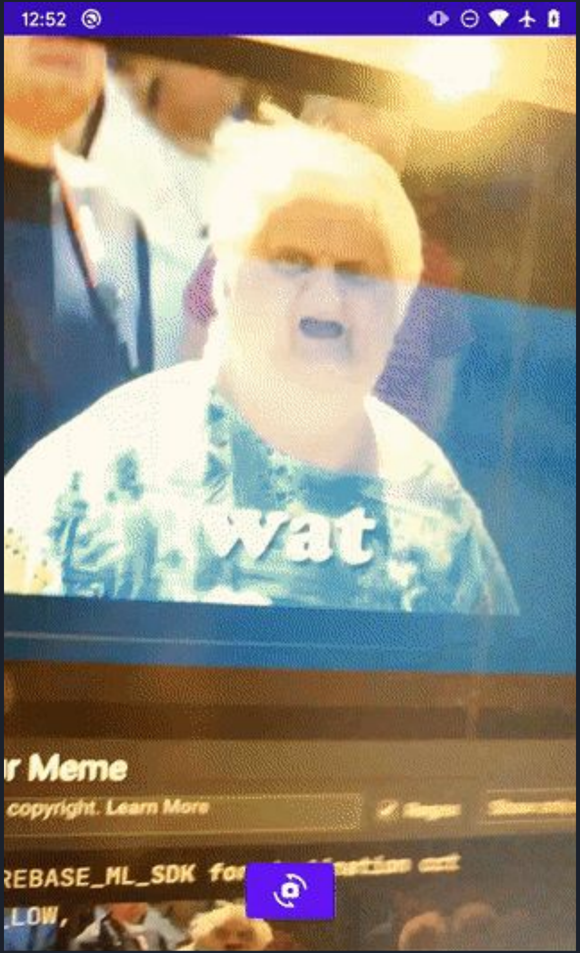
```
val detector = FaceDetection.getClient(faceDetectorOptions)

detector.process(InputImage.fromMediaImage(image.image!!, image.imageInfo.rotationDegrees))
    .addOnSuccessListener(executor) { results: List<Face> ->
      onDetectionFinished(results)
    }
    .addOnFailureListener(executor) { e: Exception ->
        Log.e("Camera", "Error detecting face", e)
    }
    .addOnCompleteListener { image.close() }
```

Binding analysis use case to camera

```
val imageProcessor = FaceDetectorProcessor()

val builder = ImageAnalysis.Builder()
val analysisUseCase = builder.build()

analysisUseCase.setAnalyzer(TaskExecutors.MAIN_THREAD, { imageProxy: ImageProxy ->
      imageProcessor.processImageProxy(imageProxy, onFacesDetected)
   }
)

bindToLifecycle(lifecycleOwner, cameraSelector, analysisUseCase)
```

Drawing detected face rectangles

```kotlin
@Composable
fun DetectedFaces(
   faces: List<Face>,
   sourceInfo: SourceInfo
) {
   Canvas(modifier = Modifier.fillMaxSize()) {
      for (face in faces) {

          drawRect(
              Color.Gray, style = Stroke(2.dp.toPx()),
              topLeft = Offset(face.boundingBox.left, face.boundingBox.top),
              size = Size(face.boundingBox.width(), face.boundingBox.height())
          )
      }
   }
}
```

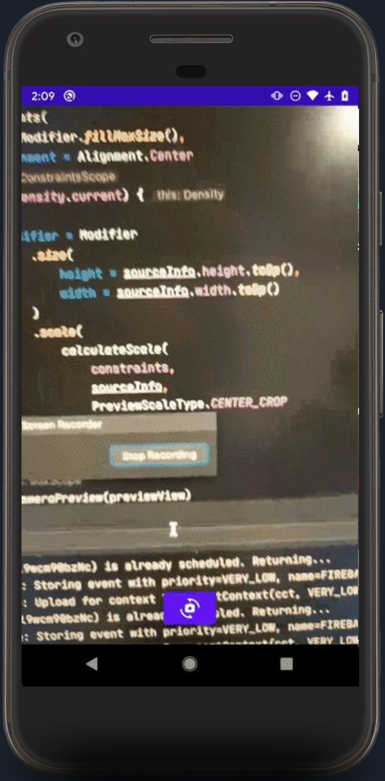

12:52
wat
r Meme
copyright. Learn More
REBASE_ML_SDK fo
LOW,

Placing preview and face into the same coordinates

```
BoxWithConstraints(
   modifier = Modifier.fillMaxSize(),
   contentAlignment = Alignment.Center
){
      Box()
           .size(
               height = sourceInfo.height,
               width = sourceInfo.width
           )
           .scale(calculateScale(...))
       {
           CameraPreview(previewView)
           DetectedFaces(faces = detectedFaces, sourceInfo = sourceInfo)
       }
   }
}
```

Scale calculation is simple

```
private fun calculateScale(
   constraints: Constraints,
   sourceInfo: SourceInfo,
   scaleType: PreviewScaleType
): Float {
   val heightRatio = constraints.maxHeight.toFloat() / sourceInfo.height
   val widthRatio = constraints.maxWidth.toFloat() / sourceInfo.width
   return when (scaleType) {
       PreviewScaleType.FIT_CENTER -> kotlin.math.min(heightRatio, widthRatio)
       PreviewScaleType.CENTER_CROP -> kotlin.math.max(heightRatio, widthRatio)
   }
}
```



```
nts(
Modifier.fillMaxSize(),
nment = Alignment.Center
ConstraintsScope
ensity.current) {  this: Density

ifier = Modifier
  .size(
      height = sourceInfo.height.toDp(),
      width = sourceInfo.width.toDp()
  )
  .scale(
      calculateScale(
          constraints,
          sourceInfo,
          PreviewScaleType.CENTER_CROP

ameraPreview(previewView)
```

```
9wcm90bzNc) is already scheduled. Returning...
: Storing event with priority=VERY_LOW, name=FIREBA
: Upload for context [illegible]tContext(cct, VERY_LOW
l9wcm90bzNc) is alrea[illegible]led. Returning...
: Storing event with priority=VERY_LOW, name=FIREBA
```

Adding pose detection

git clone git@github.com:nekdenis/camera_workshop.git

```
git checkout step_06_pose_detection
```

Thanks for your energy!