ARCore and Sceneform Basics

Sensor Based Mobile Applications

Patrick Ausderau, Ulla Sederlöf, Jarkko Vuori
Original author: Kari Salo

Helsinki Metropolia University of Applied Science

2022

Metropolia

Outline

Augmented Reality Intro

Sceneform

Filament

Basic AR App

3D Models

Lab

For more info:

`https://developers.google.com/ar`

Metropolia

Augmented Reality

- "Digital information is overlaid on the physical world" –typically visual info, could be also auditory info (audio augmented reality)
- Several tools for Android:
  - Wikitude, Vuforia, EasyAR, Maxst, DeepAR, etc.
  - ARCore by Google
    - Supported by devices on Android $>= 7.0$[1]

[1] https://developers.google.com/ar/devices

AR Terms

- World space
  - Coordinate space in which the camera and objects are positioned
  - Camera and object positions are updated in world space from frame to frame
- Pose
  - Represents an object's position and orientation in world space

AR Terms (Continued...)

- Anchor
  - Describes a fixed location and orientation in the real world.
  - Ensures that objects appear to stay at the same position and orientation in space
  - Can be attached to Trackable (pose relative to a Trackable) or ARCore session (stay at the same pose in world space)
- Trackable
  - Trackable is something (plane, augmented image or point) that ARCore can track and that Anchors can be attached to

`https://developers.google.com/ar/develop/fundamentals`

`https://www.andreasjakl.com/basics-of-ar-anchors-keypoints-feature-detection/`

`https://www.andreasjakl.com/basics-of-ar-slam-simultaneous-localization-and-mapping/`

Sceneform

- no need to learn OpenGL, unless you want
  - You can use OpenGL instead of Sceneform[2]
- high-level scene graph API
  - three ways to create virtual objects to be displayed on a scene
    - using standard Android widgets, like ImageView, Button,...
    - using basic shapes (like cube or cylinder) and materials to create a 3D model
    - using existing 3D models in glTF-format

`https://www.threekit.com/blog/gltf-everything-you-need-to-know`

`https://www.threekit.com/blog/when-should-you-use-fbx-3d-file-format`

`https://www.threekit.com/blog/obj-3d-file-format-when-should-you-use-it`

[2] check e.g.: `https://github.com/google-ar/arcore-android-sdk/tree/master/samples/hello_ar_java`

Sceneform (Continued...)

- high-level scene graph API
  - includes ArFragment which does a lot of AR related preparations:
    - checks that ARCore and camera permission are available
    - creates ArSceneView, which displays camera content
    - after ARCore has detected Planes highlights them

Sceneform Terms

- Scene
  - The ARSceneView has a Scene attached to it.
  - Scene is tree-like data structure that holds Nodes that contain the virtual objects to be rendered.
- Renderable
  - Android widget (ViewRenderable)
  - 3D model (ModelRenderable)

  that can be rendered on the screen by Sceneform.

Sceneform Terms (Continued...)

- Node
  - contains all the information Sceneform needs to render it (including its position, orientation, and renderable object) as well as interact with it (including its collision shape and event listeners).
- TransformableNode
  - Node that can be selected, translated, rotated, and scaled using gestures from TransformationSystem
- AnchorNode
  - Node that is automatically positioned in world space based on an ARCore Anchor

Filament

- Filament is a real-time physically-based renderer written in C++. It is mobile-first, but also multi-platform.

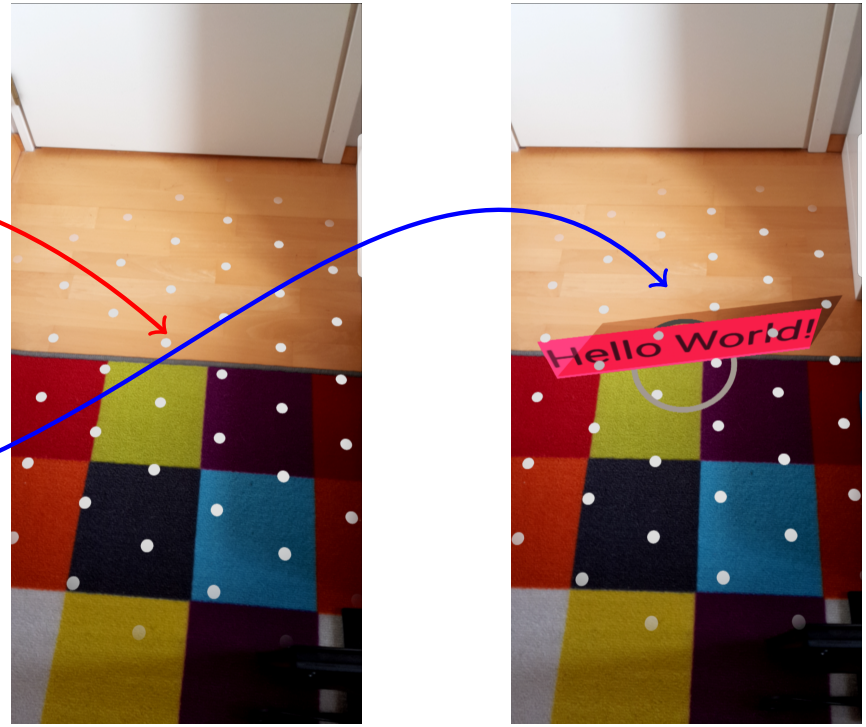
  `https://google.github.io/filament/`

  `https://medium.com/@philiprideout/getting-started-with-filament-on-android-d10b16f0ec67`

  `https://github.com/google/filament/tree/main/android/samples`
- Based on Romain Guy's comments it seems that Filament will replace Sceneform, so near future you will see ARCore + Filament integration
  - Sceneform SDK version >= 1.16.0 open sourced... `https://github.com/google-ar/sceneform-android-sdk`
  - ...archived at 1.17.1...
  - ...maintained by volunteers on a fork...`https://github.com/ThomasGorisse/sceneform-android-sdk`
  - ...and will be moved to SceneView (no more fragments, compose in progress,...) `https://github.com/SceneView/sceneview-android`

Basic AR App

- Let's create an app, which detects automatically planes (on flat surfaces) inside camera view (utilising ArFragment). When a user taps a plane then an Android widget (TextView) will be added into that plane.



`https://developers.google.com/ar/develop/java/quickstart`

Min SDK: 24

Dependencies

- In manifest, add permission, feature and metadata:

```
1  <uses-permission android:name="android.permission.CAMERA" />
2  <uses-feature android:name="android.hardware.camera.ar"
       android:required="true" />
3  <application><!-- ... -->
4    <meta-data android:name="com.google.ar.core"
         android:value="required" />
5  </application>
```

Dependencies (Continued...)

- In module build gradle:

```
1  //...
2  android {
3    //...
4    compileOptions {
5      sourceCompatibility JavaVersion.VERSION_1_8
6      targetCompatibility JavaVersion.VERSION_1_8
7    }
8  }
9  //...
10 dependencies {
11   //...
12   implementation
         "com.gorisse.thomas.sceneform:sceneform:1.21.0"
13 }
```

Activity Layout

```
1  <FrameLayout><!-- xmlns,... -->
2  <!-- FrameLayout: display view elements on top of each other
       -->
3
4    <androidx.fragment.app.FragmentContainerView
5      android:id="@+id/sceneform_fragment"
6      android:name="com.google.ar.sceneform.ux.ArFragment"
7      android:layout_width="match_parent"
8      android:layout_height="match_parent" />
9
10 </FrameLayout>
```

ViewRenderable Layout

- ViewRenderable needs it's own layout

```
1  <LinearLayout><!-- xmlns,... -->
2
3      <TextView
4          android:layout_width="wrap_content"
5          android:layout_height="30dp"
6          android:id="@+id/txt"
7          android:textSize="24sp"
8          android:background="@color/colorAccent"
9          android:text="@string/hello" />
10
11 </LinearLayout>
```

Activity

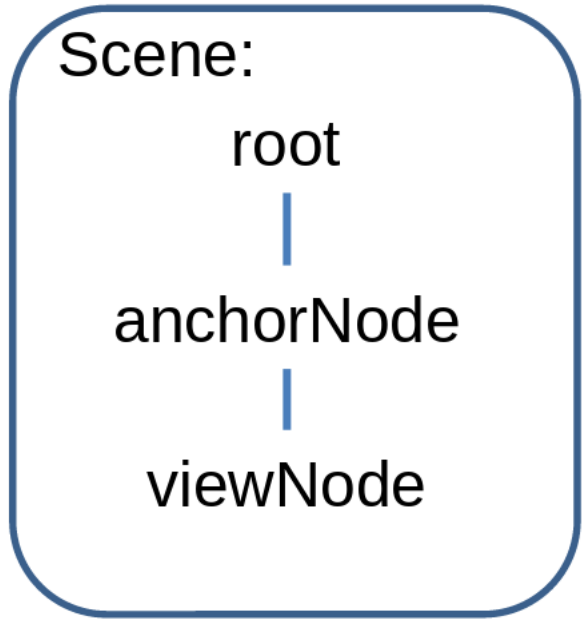
```
1  private lateinit var arFrag: ArFragment
2  private var viewRenderable: ViewRenderable? = null
3
4  //onCreate()...
5    arFrag = supportFragmentManager.findFragmentById(
         R.id.sceneform_fragment) as ArFragment
6    ViewRenderable.builder()
7      .setView(this, R.layout.rend_text)
8      .build()
9      .thenAccept{viewRenderable = it}
```

Builder will create renderable asynchronously (it will return `CompletableFuture<ViewRenderable>`). Using CompletableFuture's `thenAccept` callback method we will have access to the result, i.e. receive the renderable when it is created.

Activity (Continued...)

```
1  //onCreate()... continued... consider separated function...
2  arFrag.setOnTapArPlaneListener { hitResult: HitResult?, _, _ ->
3    viewRenderable ?: return@setOnTapArPlaneListener
4    //Creates a new anchor at the hit location
5    val anchor = hitResult!!.createAnchor()
6    //Creates a new anchorNode attaching it to anchor
7    val anchorNode = AnchorNode(anchor)
8    // Add anchorNode as root scene node's child
9    anchorNode.setParent(arFrag.arSceneView.scene)
10   // Can be selected, rotated...
11   val viewNode = TransformableNode(arFrag.transformationSystem)
12   viewNode.renderable = viewRenderable
13   // Add viewNode as anchorNode's child
14   viewNode.setParent(anchorNode)
15   // Sets this as the selected node in the TransformationSystem
16   viewNode.select()
17 }
```

Activity (Continued...)



Basic AR App

- Let's create an app, which detects automatically planes (on flat surfaces) inside camera view (utilising ArFragment). When a user clicks a button then a 3D model will be added in the screen center.

`https://developers.google.com/sceneform/develop/create-renderables`

Min SDK: 24

Dependencies

- Same build gradle as Basic AR App
- In manifest, add extra permission and feature:

```
1  <!-- optional, if 3d model stored remotely -->
2  <uses-permission android:name="android.permission.INTERNET" />
3  <!-- optional, make sure of minimal Graphics Library support
       -->
4  <uses-feature android:glEsVersion="0x00030000"
       android:required="true" />
5  <!-- permission camera, feature camera.ar, metadata ar.core -->
```

Activity

```
1  import android.graphics.Point
2
3  // class...
4  private lateinit var arFrag: ArFragment
5  private var modelRenderable: ModelRenderable? = null
6
7  private fun getScreenCenter(): Point {
8    //  find the root view of the activity
9    val vw = findViewById<View>(android.R.id.content)
10   // returns center of the screen as a Point object
11   return Point(vw.width / 2, vw.height / 2)
12 }
```

Activity (Continued...)

```
1  //onCreate...
2  findViewById<Button>(R.id.btn_add_tree).setOnClickListener {
       add3dObject() }
3  arFrag = supportFragmentManager.findFragmentById(
       R.id.sceneform_fragment) as ArFragment
4  // (CC BY 4.0) Donated by Cesium for glTF testing.
5  // https://github.com/KhronosGroup/glTF-Sample-Models/
6  // put gltf, bin, jpg in assets
7   ModelRenderable.builder()
8       .setSource(this, Uri.parse("CesiumMan.gltf"))
9       .setIsFilamentGltf(true)
10      .setAsyncLoadEnabled(true)
11      .setRegistryId("CesiumMan")
12      .build()
13      .thenAccept { modelRenderable = it }
14      .exceptionally {
15        Log.e(TAG, "something went wrong ${it.localizedMessage}")
16        null
17      }
```

Activity (Continued...)

```
1  private fun add3dObject() {
2    val frame = arFrag.arSceneView.arFrame
3    if (frame != null && modelRenderable != null) {
4      val pt = getScreenCenter()
5      // get list of HitResult of the given location in the
           camera view
6      val hits = frame.hitTest(pt.x.toFloat(), pt.y.toFloat())
7      for (hit in hits) {
8        val trackable = hit.trackable
9        if (trackable is Plane) {
10         val anchorNode = AnchorNode(hit!!.createAnchor())
11         anchorNode.parent = arFrag.arSceneView.scene
12         val mNode =
               TransformableNode(arFrag.transformationSystem)
13         mNode.renderable = modelRenderable
14         mNode.parent = anchorNode
15         mNode.select()
16         break
17       }
18     }
19   }
20 }
```

Metropolia

Activity (Continued...)

in case the model is too big (or too small) and/or don't let zoom in/out enough

```
1  //...
2  val mNode = TransformableNode(arFrag.transformationSystem)
3  mNode.renderable = modelRenderable
4  // Default min is 0.75, default max is 1.75.
5  mNode.scaleController.minScale = 0.05f
6  mNode.scaleController.maxScale = 2.0f
7  mNode.localScale = Vector3(0.2f, 0.2f, 0.2f) // scale at 20%
8  // scale must be set before setting parent
9  mNode.parent = anchorNode
10 //...
```

Lab_w4_d1 Basic 3D App

- Create a simple AR application, which detects planes and let user to attach 3D-model into scene by clicking a button
  - Ideally use your own 3d model
  - Otherwise, find some 3D-model without animation in .glTF (version2) format. Respect copyright.
- When user taps the model hide button
- If you feel adventurous, try SceneView (replacement of Sceneform)