ARCore & Sceneform Workshop

- Welcome! Before we get started...
- Clone the project
- Check that Android Studio is the correct version or higher
- Set up an ARCore supported device...
- ... or emulator

About ARCore

- Google released ARCore in February 2018.
- ARCore helps devs build apps that can understand the environment around a device and place objects and information in it.

About Sceneform

- Google followed up with Sceneform at I/O 2018.
- Sceneform helps devs render 3D scenes on Android without needing to learn OpenGL.

Add the dependency

In `app/build.gradle`

```
dependencies {
    // ...

    implementation "com.google.ar.sceneform.ux:sceneform-ux:1.4.0"
}
```

Configure the manifest

Setup camera in the `<manifest>` section

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-feature android:name="android.hardware.camera.ar" android:required="true" />
```

In the `<application>` section, add a Play Store filter for users on devices that are not supported by ARCore.

```
<meta-data android:name="com.google.ar.core" android:value="required" />
```

Add the ARFragment

In `content_main.xml`

```
<fragment
    android:id="@+id/arFragment"
    android:name="com.google.ar.sceneform.ux.ArFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```

Run it!

Emulator troubleshooting

- Check that your Android Emulator is updated to 27.2.9 or later.
- Follow the instructions linked in the README.

Motion Tracking

ARCore detects visually distinct **feature points** in each captured camera image and uses these points to compute a device's change in location over time.

Motion tracking

Visual feature point information is combined with measurements from the device's Inertial Measurement Unit (IMU).

Motion tracking

This combined data is used to estimate the **pose**, defined by **position** and **orientation**, of the device camera relative to the world over time.

Pose

Everything in an AR scene has a **pose** within a 3D world coordinate space.

Each **pose** is composed of:

- x-axis translation
- y-axis translation
- z-axis translation
- rotation

Pose

Sceneform aligns the pose of the **virtual camera** that renders your 3D content with the pose of the device's camera provided by ARCore.

Pose

Because the rendered virtual content is overlaid and aligned on top of the camera image, it appears as if your virtual content is part of the real world.

Feature points

feature points are visually distinct features that ARCore detects in each captured camera image (e.g. the corner of a table, a mark on a wall)

Plane detection

ARCore looks for clusters of feature points that appear to lie on common horizontal or vertical surfaces and provides this data to Sceneform as **planes**.

Plane detection

A **plane** is composed of:

- A center **pose**
- An **extent** along each axis
- A **polygon**, a collection of 2D vertices approximating the detected plane.

Plane detection

The Sceneform fragment renders a plane-grid to indicate via the UI where these planes exist.

Goal:

When a user taps a point on the screen that intersects with a plane, we want to place an object at that point.

Detecting taps

In `MainActivity.kt`, set up the `TapHelper.kt` gesture detector.

```
private lateinit var arSceneView
private val tapHelper = TapHelper(this)

override fun onCreate(savedInstanceState: Bundle?) {
    //...

    arSceneView = (arFragment as ArFragment).arSceneView
    arSceneView.setOnTouchListener(tapHelper)
}
```

Update loop

```
private lateinit var arSceneView
private val tapHelper = TapHelper(this)

override fun onCreate(savedInstanceState: Bundle?) {
    //...

    arSceneView = (arFragment as ArFragment).arSceneView
    arSceneView.setOnTouchListener(tapHelper)

    arSceneView.scene.addOnUpdateListener { frameTime ->
        // Handle one tap per frame.
        val motionEvent = tapHelper.poll()
    }
}
```

Hit test

We want to determine if a **ray** extending from the **x, y** coordinate of our tap intersects a plane.

If it does, we'll place an **anchor** at the **pose of the intersection**.

Hit test

In `MainActivity.kt`

```
private fun hitTest(x : Float, y : Float) : Boolean {
    val frame = arSceneView.arFrame
    val hits: List<HitResult>
    if (frame != null) {
        hits = frame.hitTest(x, y)
        for (hit in hits) {
            val trackable = hit.trackable
            if (trackable is Plane && trackable.isPoseInPolygon(hit.hitPose)) {
                return true
            }
        }
    }
    return false
}
```

Anchors

Run it!

Sceneform plugin

Sceneform plugin: installation

Import an asset

Supported formats

Sceneform assets

Sceneform assets: editing

Loading assets

Renderables

Nodes

Scene graph

Run it!

Now for the fun stuff...

Cloud anchors

Augmented Faces

Grazie!