Google ARCore Software Development Kit for Android Apps

Dmytro Zubov, PhD
dmytro.zubov@ucentralasia.org

Naryn, Kyrgyzstan, October 26, 2020

… is here

Naryn, Kyrgyzstan, October 26, 2020

Augmented reality is here

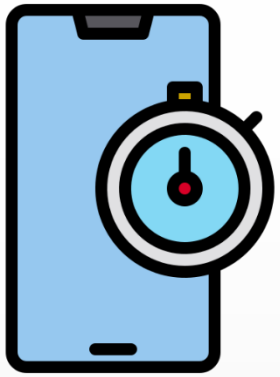
We gonna mix real and virtual worlds :)

Special thanks to Valeriya Nikiforova and Jawad Haider for sharing their photos

September 22, 2021

July 2018, river Almatinka and man-made lake Sayran, Kazakhstan

Lessons learnt last time



- Java maps
- Intro to Java concurrency
- Develop a web-browser mobile app in 5 min

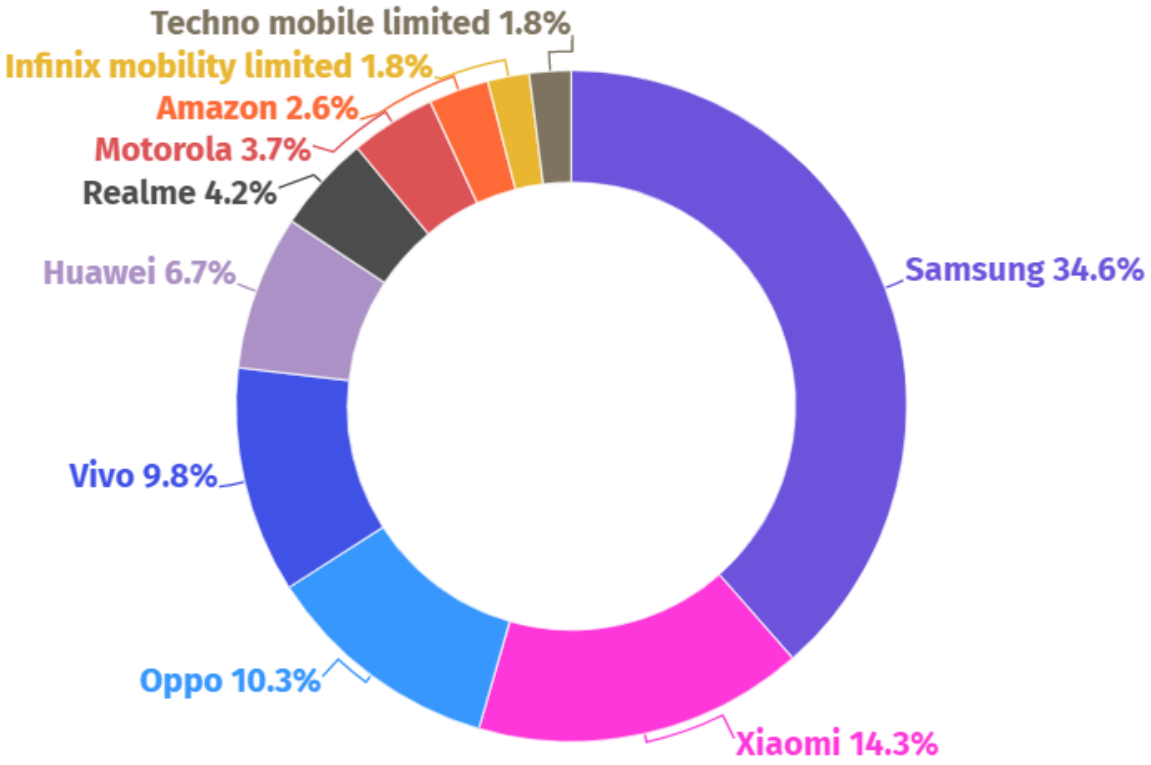
What we gonna discuss today?

- What is Augmented Reality (AR)?
- Difference between Virtual Reality, Augmented Reality and Mixed Reality
- Types of AR
- AR applications
- Intro to ARCore
- AR Java Android App in Android Studio
- AR Java Android App in Android Studio: A Local 3D model
- AR Java Android App in Android Studio: A Local 3D model (shape)

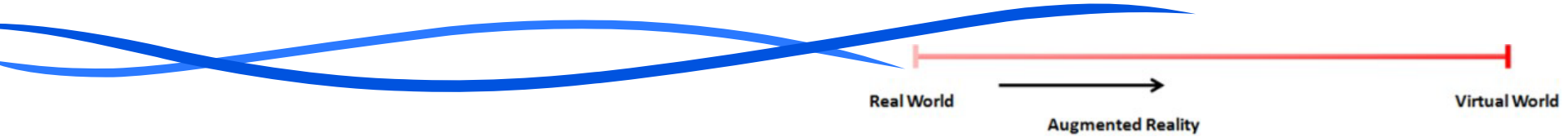
November 2022

Top Android phone manufacturers by market share, as of November 2022 (in percentage)

Source: AppBrain



What is Augmented Reality (AR)?



- *Definition* 1: AR can be defined as a system that fulfills three basic features – a combination of real and virtual worlds, real-time interaction, and accurate 3D registration of virtual and real objects (https://en.wikipedia.org/wiki/Augmented_reality)
- *Definition* 2: AR is a variation of Virtual Reality (VR). VR technologies completely immerse a user inside a synthetic environment. While immersed, the user cannot see the real world around him/her. In contrast, AR allows the user to see the real world, with virtual objects superimposed upon or composited with the real world (https://www.slideshare.net/zilk/making-augmented-reality-applications-with-android-ndk-28065038).

What is Augmented Reality (AR)?

- *Other definitions* - ?

What is Augmented Reality (AR)?

- Virtual Fixtures – first AR system, U.S. Air Force, Wright-Patterson Air Force Base (1992; https://en.wikipedia.org/wiki/Augmented_reality)

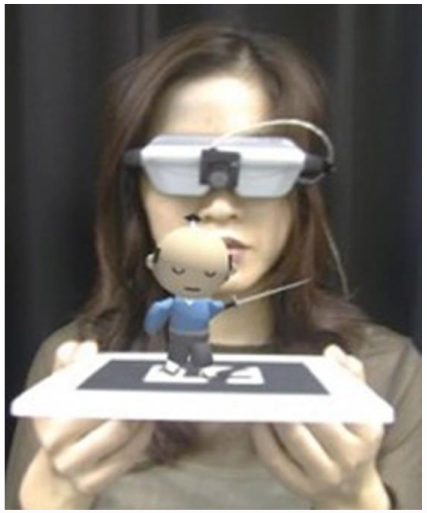
Difference between Virtual Reality, Augmented Reality and Mixed Reality

https://www.youtube.com/watch?v=3mr_S5mOtsw

Types of AR

- Tracking-based:
  - GPS + Compass + Gyro + Accelerometer
  - Marker (Fiduciary, frame, etc.)
  - NFT (Natural Feature Tracking; 2D images)
  - 3D (Pre-trained point cloud)
  - Live 3D (Simultaneous localization and mapping)
  - Face, fingers, body, and other objects that can be described as Haar-like features

Types of AR (cont.)



- Marker-based:
  - Marker-based AR uses a camera and a visual marker to determine the center, orientation, and range of its spherical coordinate system
  - Markers work by having software recognize a particular pattern, such as a barcode or symbol, when a camera points at it, and overlaying a digital image at that point on the screen

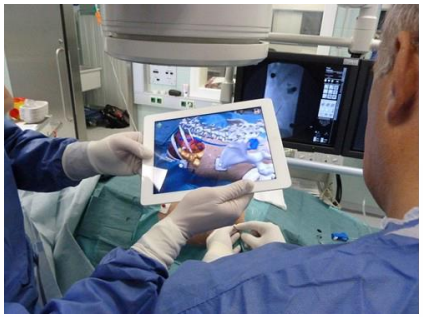
Types of AR (cont.)

- Image-based:
  - Image targets are images that the AR SDK can detect and track
  - Unlike traditional markers, data matrix codes and QR codes, image targets do not need special black and white regions or codes to be recognized
  - The AR SDK uses sophisticated algorithms to detect and track the features/key points that are naturally found in the image itself

Types of AR (cont.)

- Location-based:
  - ° GPS + Compass + Gyro + Accelerometer
  - ° Location-based applications use the ability of a particular device to record its position in the world and then offer data that is relevant to that location: finding the way around a city, remembering where the car was parked, naming the mountains around specific location, etc.

AR Applications



- Medical:
  - Computer scientists get output images from the computer tomography (CT) for the virtually produced image of the inner body. A modern spiral-CT makes several X-Ray photographs with diverse perspectives and then reconstructs their 3-dimensional perspective. A computer-aided tomogram is clearer than a normal X-ray photograph because it enables differentiation of the body's various types of tissue. **The computer scientist then superimposes the saved CT scans with a real photo of the patient on the operating table.** For surgeons, the impression produced is that of looking through the skin and throughout the various layers of the body in 3-dimensions and in color.

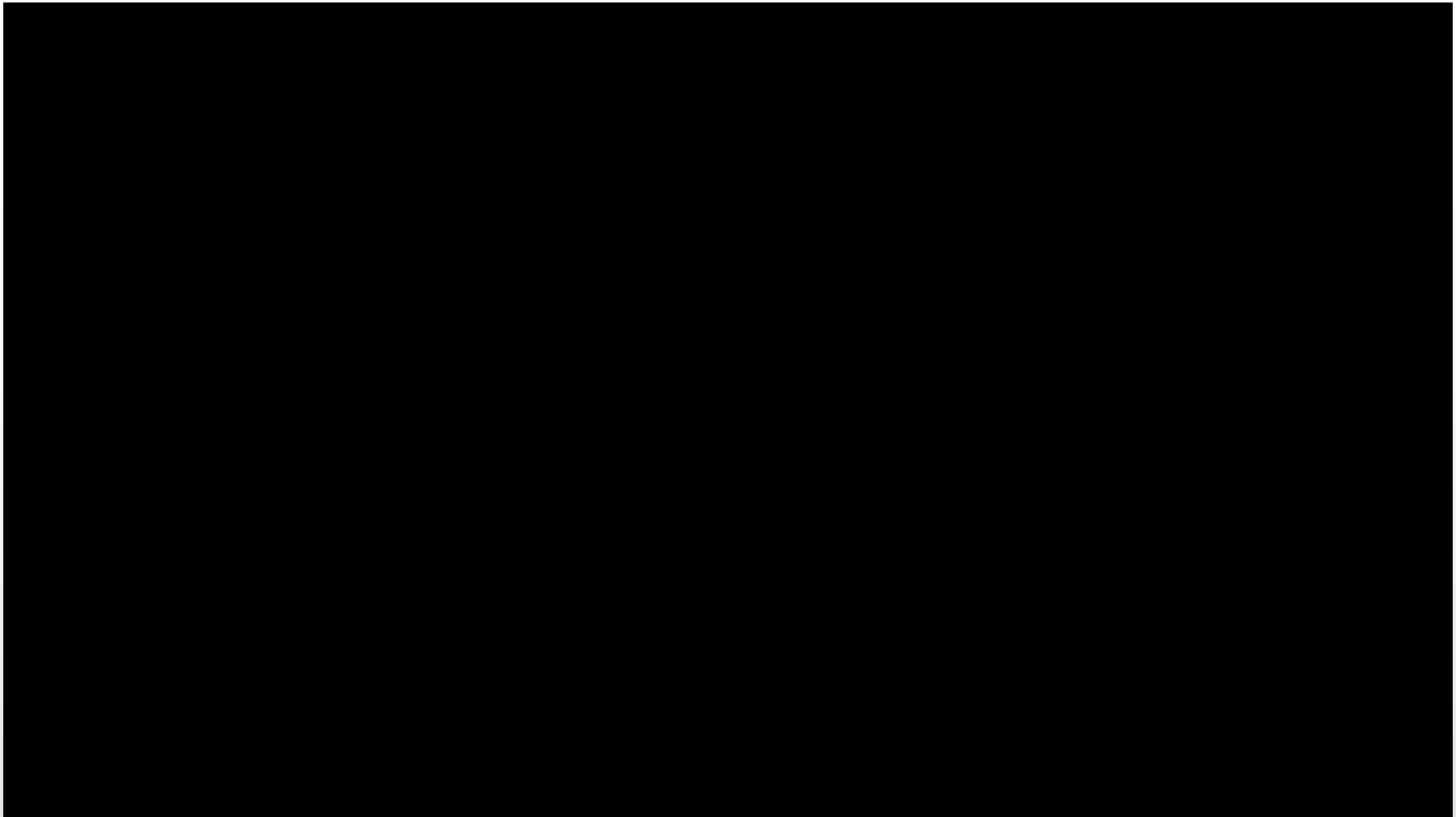
AR Applications (cont.)

- Entertainment:
  - An example: The “virtual watch” is created by AR technology that allows the consumer to interact with the design by twisting their wrist for a 360-degree view. Shoppers can “try on” different watches using the wristband (marker), e.g., http://ar-watches.com/marker/marker.pdf

AR Applications (cont.)

https://www.youtube.com/watch?v=rKOBR3Vztlo&ab_channel=ARLOOPAAugmented%2FVirtualReality

- Entertainment (cont.): Trying on virtual watches using AR app

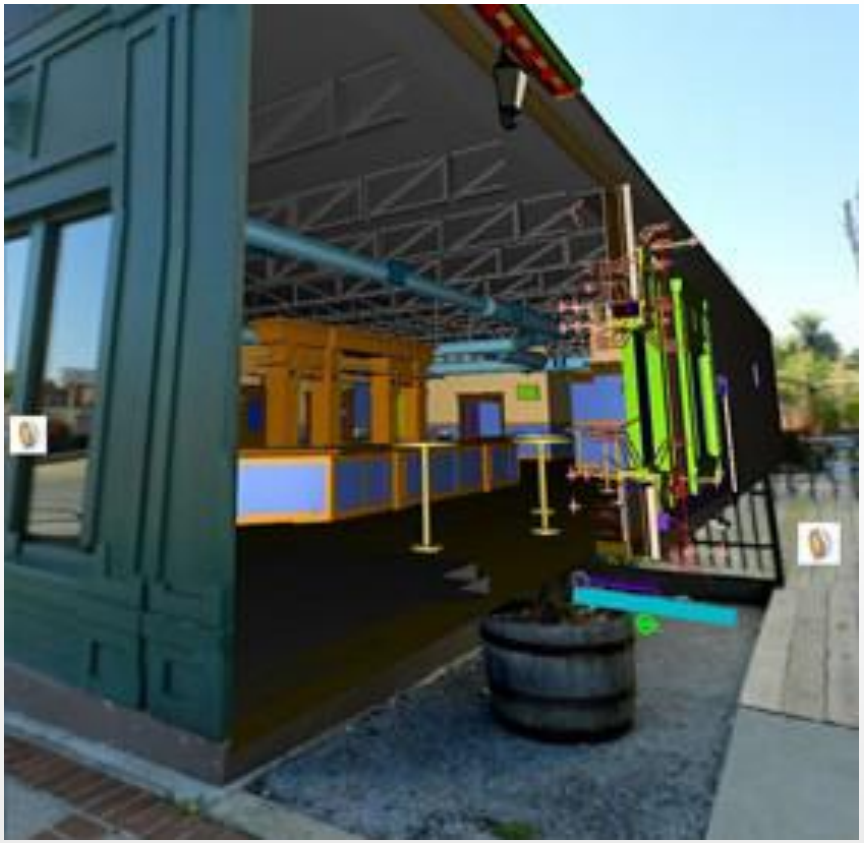


AR Applications (cont.)

- Military
  - The military has been using displays in cockpits that present information to the pilot on the windshield of the cockpit or the visor of the flight helmet. This is a form of AR display.

AR Applications (cont.)

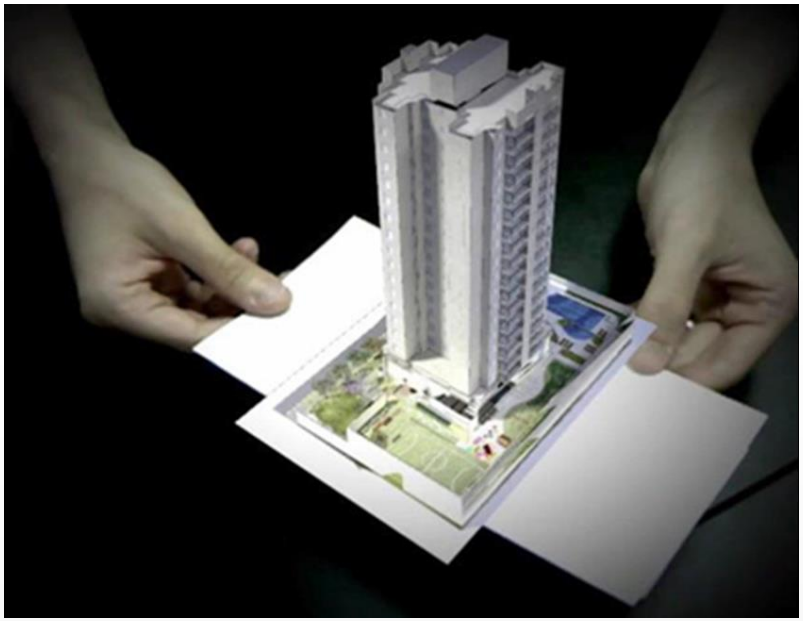
- Engineering design:
  - Integrating drawings and cutouts with real-world images provides context for an engineer



AR Applications (cont.)

- Tourism:
  - By viewing a physical environment whose elements are augmented by computer generated images, the viewer can experience a historic place or event as if he or she has traveled back in time

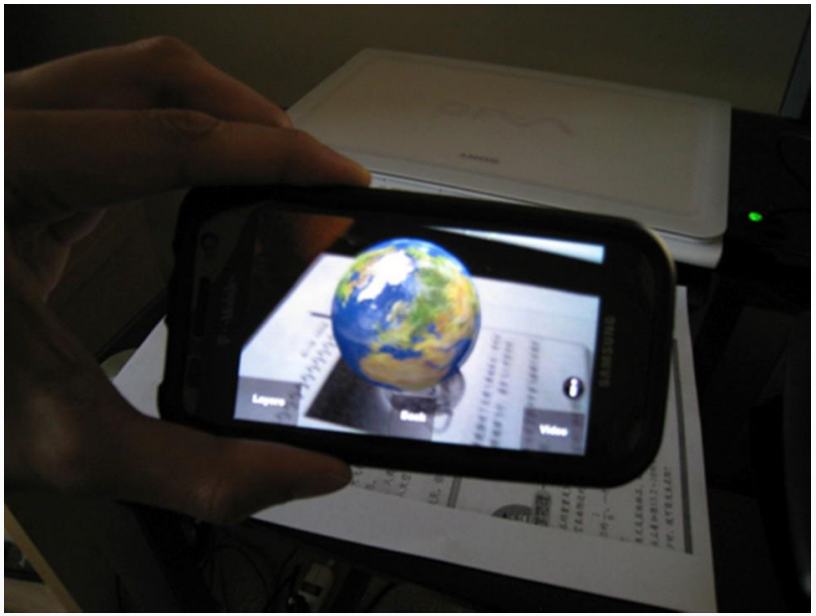
AR Applications (cont.)



- Architecture:

  ◦ AR can aid in visualizing building projects. Computer-generated images of a structure can be superimposed into a real-life local view of a property before the physical building is constructed there. AR can also be employed within an architect's workspace, rendering into their view animated 3D visualizations of their 2D drawings.

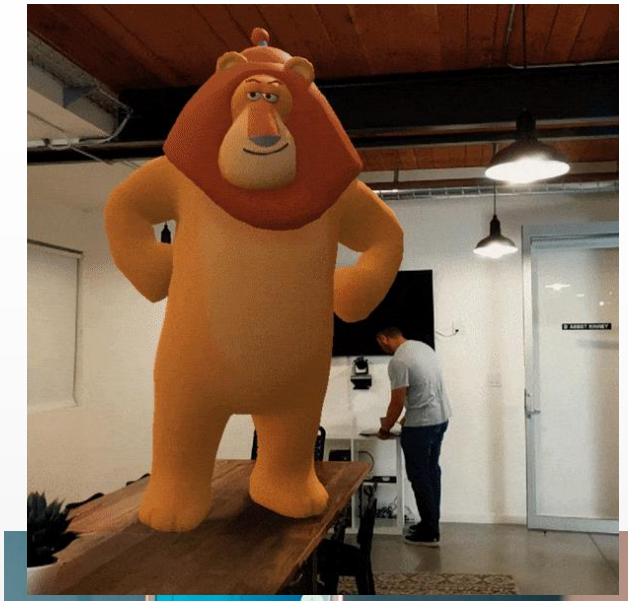
AR Applications (cont.)

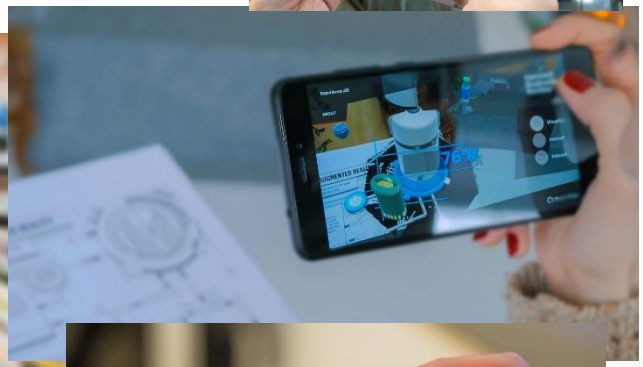


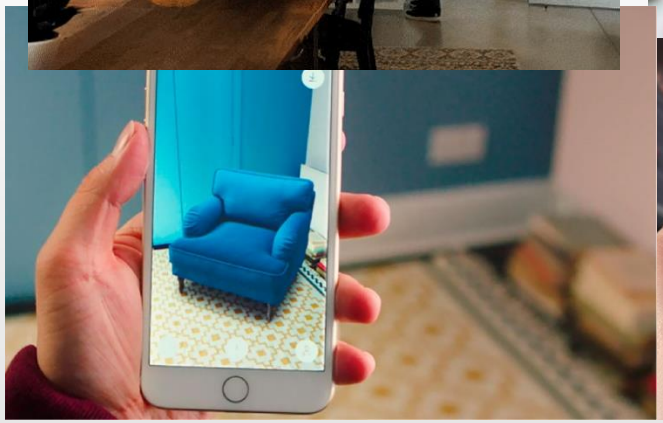
- Education:
  - AR technology has been successfully used in various educational institutes to act as add-ons to the textbook material or as a virtual, 3D textbook in itself. Normally done with head mounts the AR experience allows the students to ‘‘relive’’ events as they are known to have happened, while never leaving their class.

AR Applications (cont.)

- and many other apps …

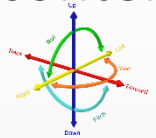






Intro to ARCore

- ARCore is a software development kit developed by Google that allows for augmented reality applications to be built
- ARCore uses three key technologies to integrate virtual content with the real world as seen through the phone's camera:



  - Six degrees of freedom allows the phone to understand and track its position relative to the world
  - Environmental understanding allows the phone to detect the size and location of flat horizontal surfaces like the ground or a coffee table
  - Light estimation allows the phone to estimate the environment's current lighting conditions

https://en.wikipedia.org/wiki/ARCore

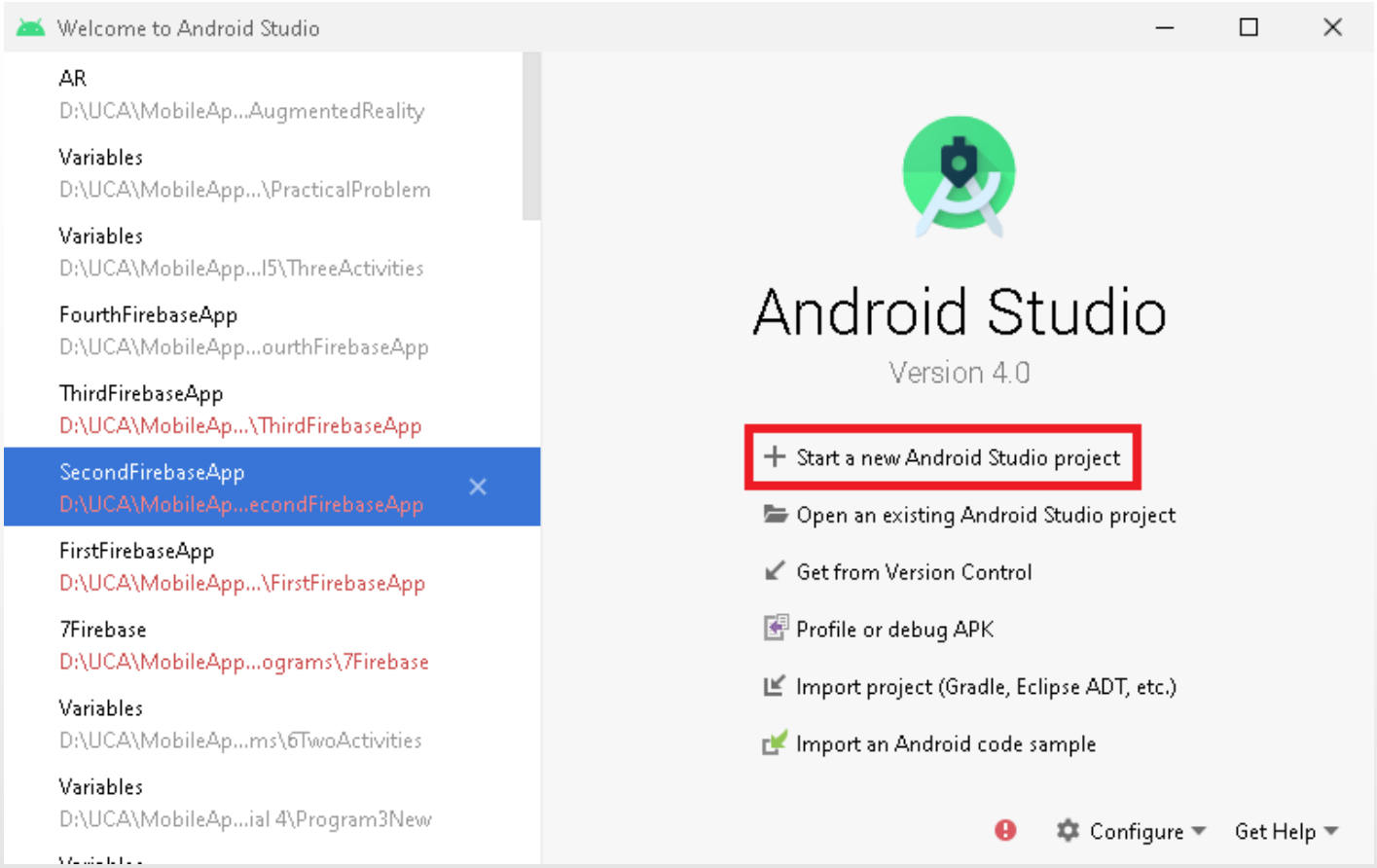
Intro to ARCore

- In 2018 Google released ARCore, the framework for creating AR apps. Although it's well optimized and really cool, it's supported on a limited number of devices. The list is growing, but it contains about 100 devices as of Nov 6, 2019. **It is currently the easiest way to create augmented reality apps - ?**

https://www.netguru.com/blog/augmented-reality-mobile-android

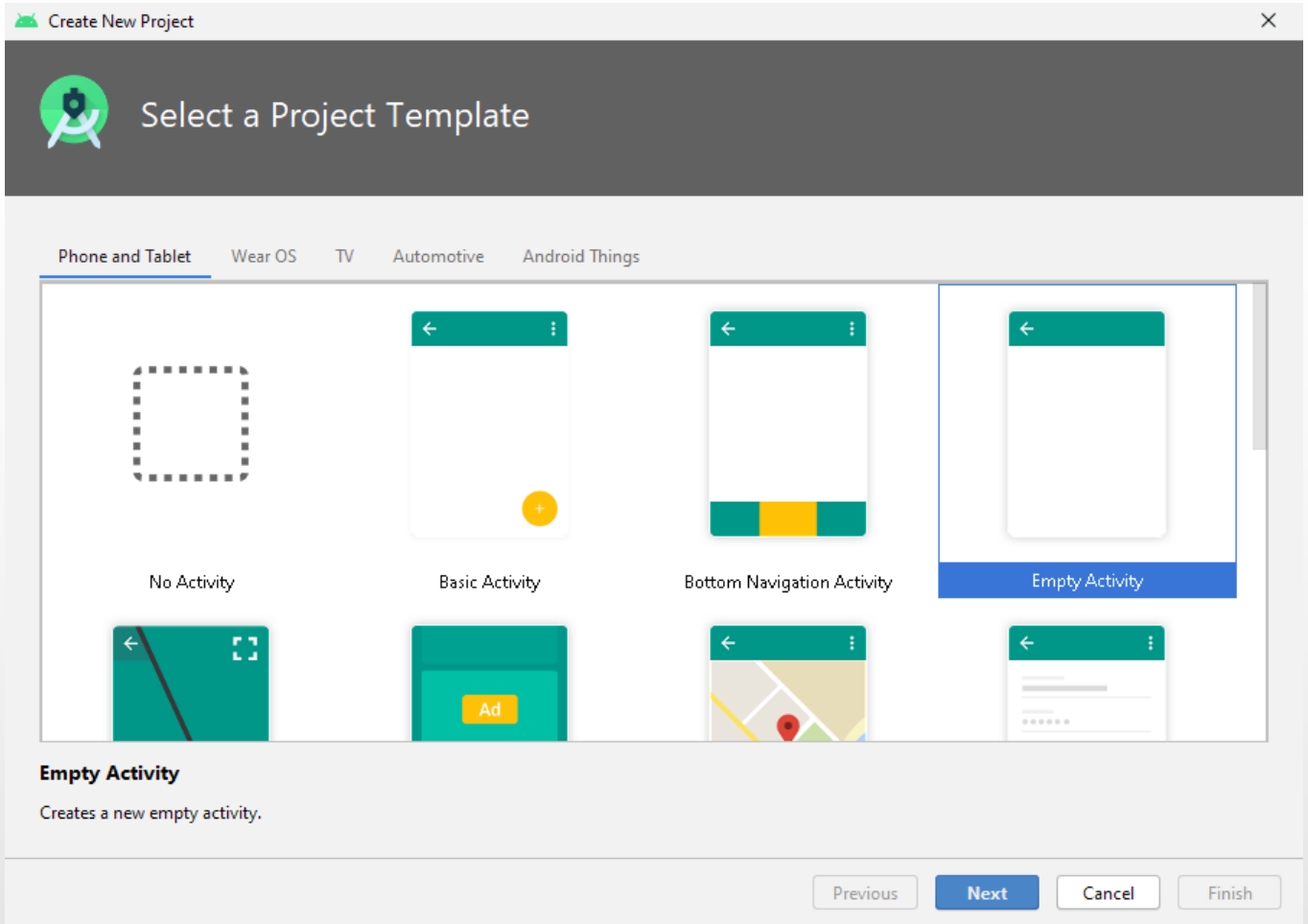
AR Java Android App in Android Studio

- Start a new Android Studio Project in Android Studio 3.1 and above (we use version 4.0):



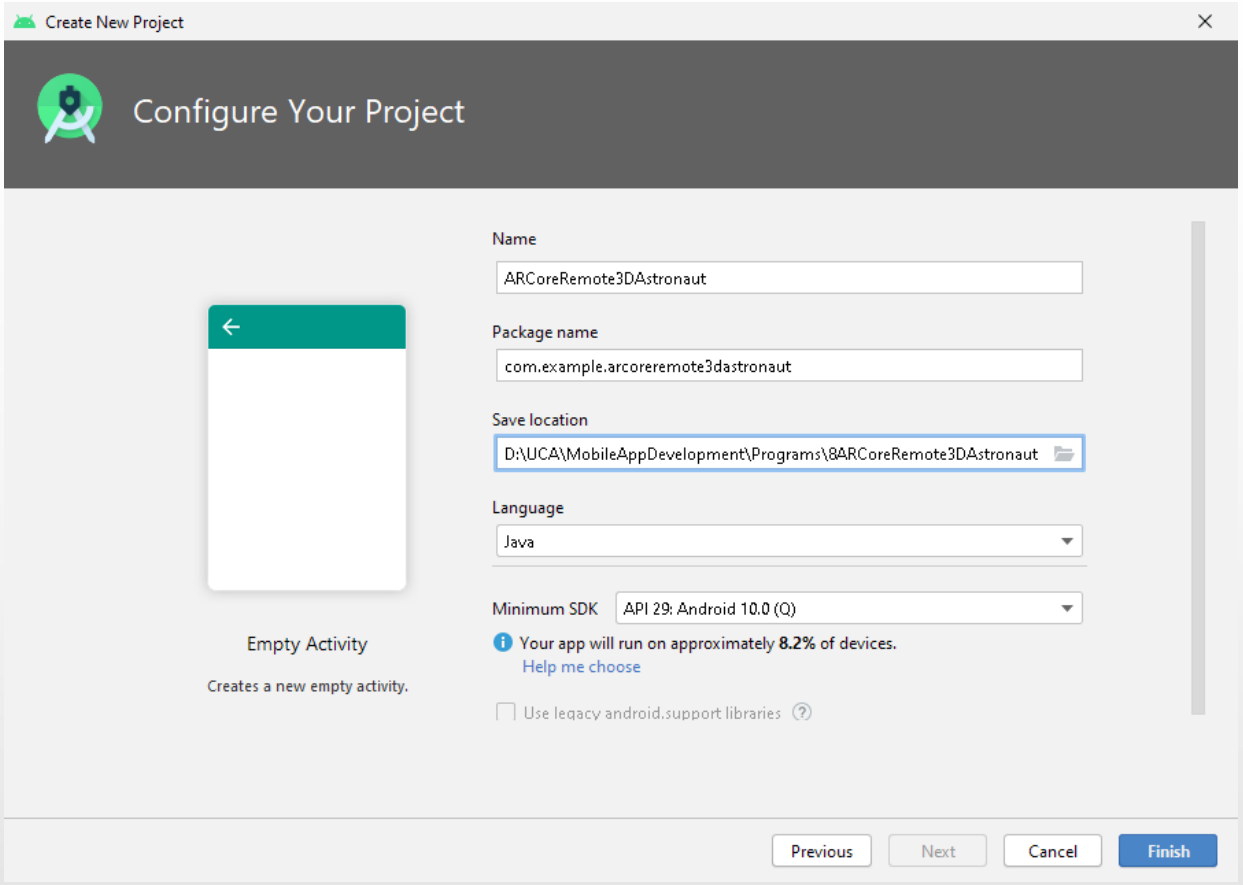
AR Java Android App in Android Studio (cont.)

- Select a project template “Empty Activity”:



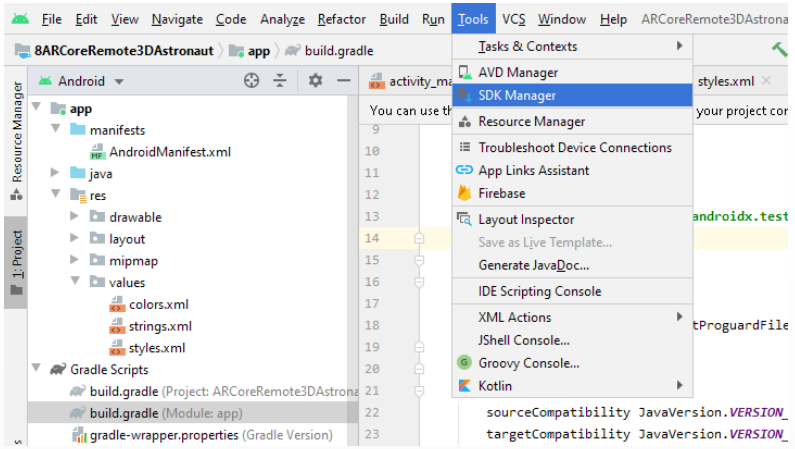
AR Java Android App in Android Studio (cont.)

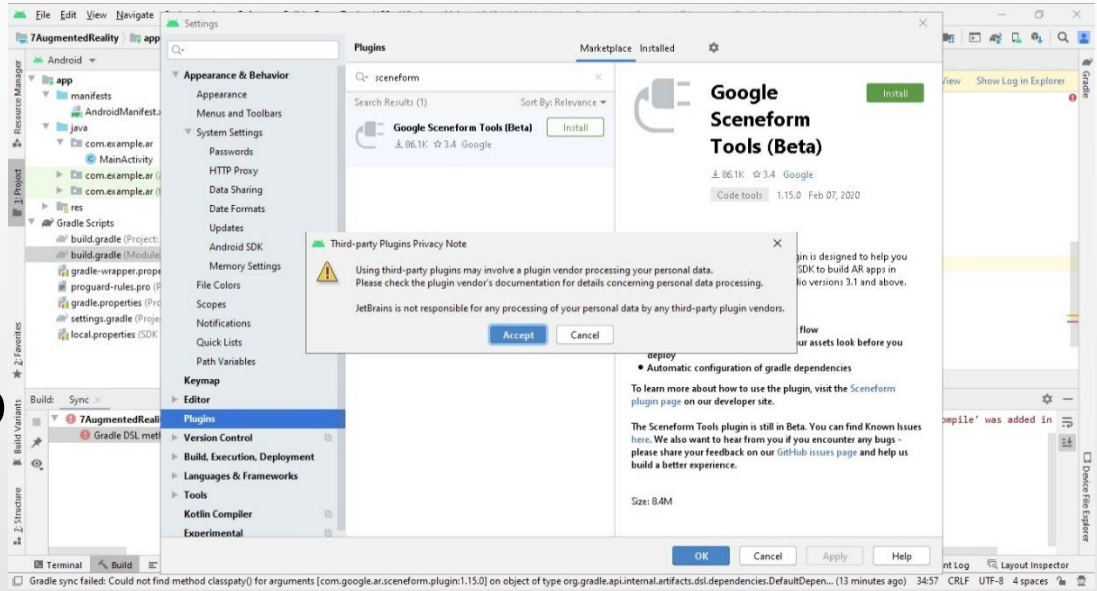
- Configure the project (minimum SDK must be API 24 or above for physical devices, 27 – for virtual devices; we use API 29):



AR Java Android App in Android Studio (cont.)

- Install the Google Sceneform Tools plugin if you don't have it in Android Studio:
  - Go to SDK Manager:
  - Go to Plugins and install the Google Sceneform Tools plugin if necessary:
  - Google Sceneform Tools (Beta) plugin is designed to help developers work with 3D assets and build AR apps in Android Studio





AR Java Android App in Android Studio (cont.)

- Device, smartphone or virtual device like AVD, MUST have the Android API level 24 (27) or newer and OpenGL ES version 3.0 or newer to work properly with Sceneform SDK. **My AVD was blocked (and frozen) to access the acceleration hardware :(**

Samsung Galaxy A7 (Oct 2018)

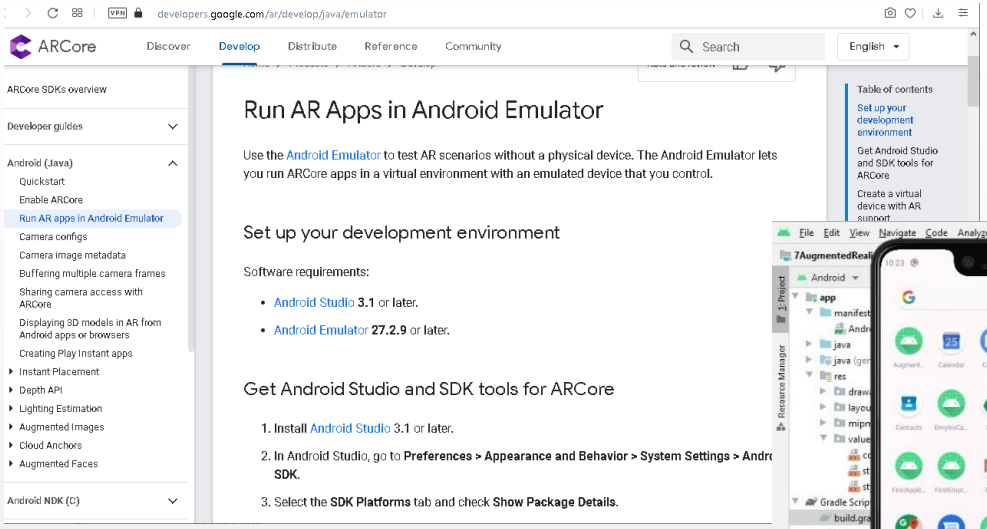
Android 10 OS

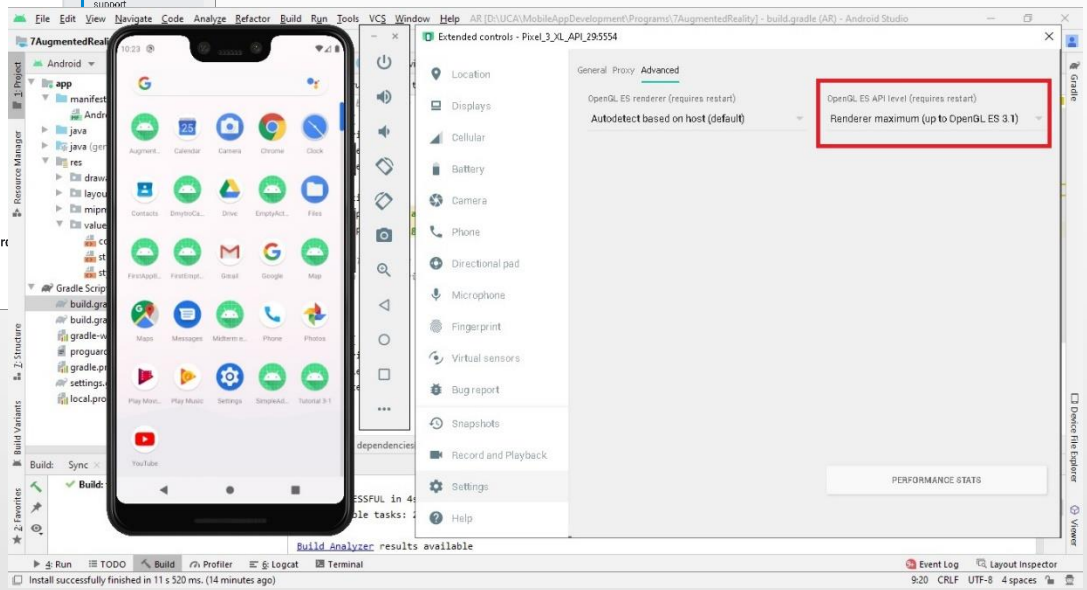
Samsung Galaxy M31 (Mar 2020)

Android 12 OS

AR Java Android App in Android Studio (cont.)

- Run AR apps in Android Emulator
  - Pls follow the instructions provided by Google:
    https://developers.google.com/ar/develop/java/emulator





AR Java Android App in Android Studio (cont.)

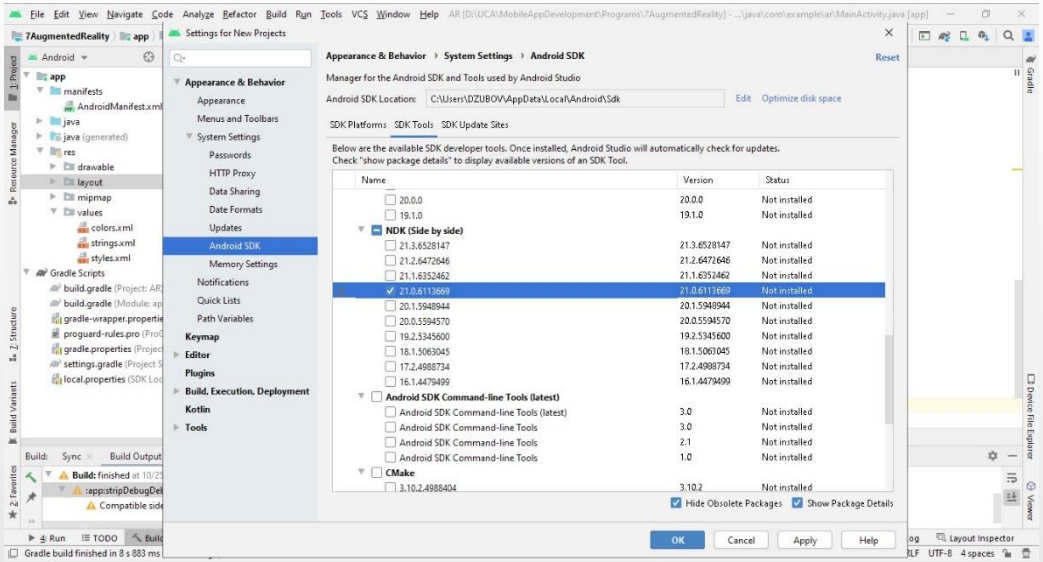
- Compatibility Check
  - This method checks whether our device can support Sceneform SDK or not. The SDK requires Android API level 27 or newer and OpenGL ES version 3.0 or newer. If a device does not support these two, the Scene would not be rendered, and our app will show a blank screen.

```
public static boolean checkIsSupportedDeviceOrFinish(final Activity activity) {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.N) {
        Log.e(TAG, "Sceneform requires Android N or later");
        Toast.makeText(activity, "Sceneform requires Android N or later", Toast.LENGTH_LONG).show();
        activity.finish();
        return false;
    }
    String openGlVersionString =
            ((ActivityManager) activity.getSystemService(Context.ACTIVITY_SERVICE))
                    .getDeviceConfigurationInfo()
                    .getGlEsVersion();
    if (Double.parseDouble(openGlVersionString) < MIN_OPENGL_VERSION) {
        Log.e(TAG, "Sceneform requires OpenGL ES 3.0 later");
        Toast.makeText(activity, "Sceneform requires OpenGL ES 3.0 or later", Toast.LENGTH_LONG)
                .show();
        activity.finish();
        return false;
    }
    return true;
}
```

https://medium.com/free-code-camp/how-to-build-an-augmented-reality-android-app-with-arcore-and-android-studio-43e4676cb36f

AR Java Android App in Android Studio (cont.)

- It is recommended to install the NDK (Native Development Kit) to avoid the error “Compatible side by side NDK version was not found.”
- To install a specific version of the NDK, do the following:
  - With a project open, click Tools > SDK Manager
  - Click the SDK Tools tab
  - Select the Show Package Details checkbox
  - Select the NDK (Side by side) checkbox and the checkboxes below it that correspond to the NDK versions you want to install. Android Studio installs all versions of the NDK in the android-sdk/ndk/ directory



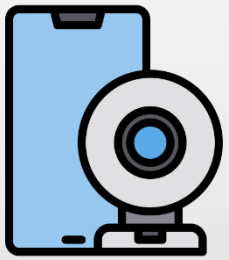
AR Java Android App in Android Studio (cont.)

- Modify the `AndroidManifest.xml` file to access camera and Internet, as well as to use AR with a camera:

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

- Modify the `AndroidManifest.xml` file to make the Google Play Services for AR available for our app:

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

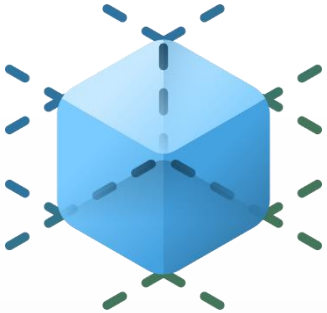


AR Java Android App in Android Studio (cont.)

- The `AndroidManifest.xml` file may look as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.arcoreremote3dastronaut">

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="ARCoreRemote3DAstronaut"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

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

</manifest>
```

AR Java Android App in Android Studio (cont.)



- Modify the `build.gradle (Module: app)` file:
  - Add the ARCore library as a dependency*:

```
implementation 'com.google.ar.sceneform.ux:sceneform-ux:1.15.0'
```

  - Add the ARCore assets library as a dependency to load 3D models directly:

```
implementation 'com.google.ar.sceneform:assets:1.15.0'
```

**The Sceneform SDK provides a high-level scene graph API, high-performance 3D renderer, and Android Studio tooling to make it easy for Android developers to work with 3D without having to know OpenGL.*

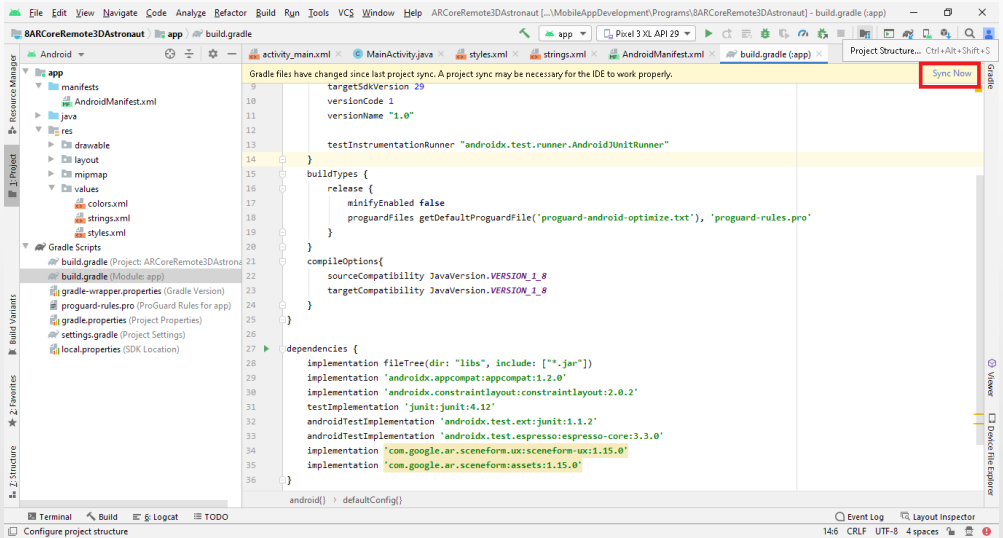
AR Java Android App in Android Studio (cont.)

- In the `build.gradle (Module: app)` file, specify the version 8 of Java that is required for the `Sceneform` library:

```
compileOptions{
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
}
```

- Click on “Sync Now”:

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 29

    defaultConfig {
        applicationId "com.example.arcoreremote3dastronaut"
        minSdkVersion 29
        targetSdkVersion 29
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
```



AR Java Android App in Android Studio (cont.)

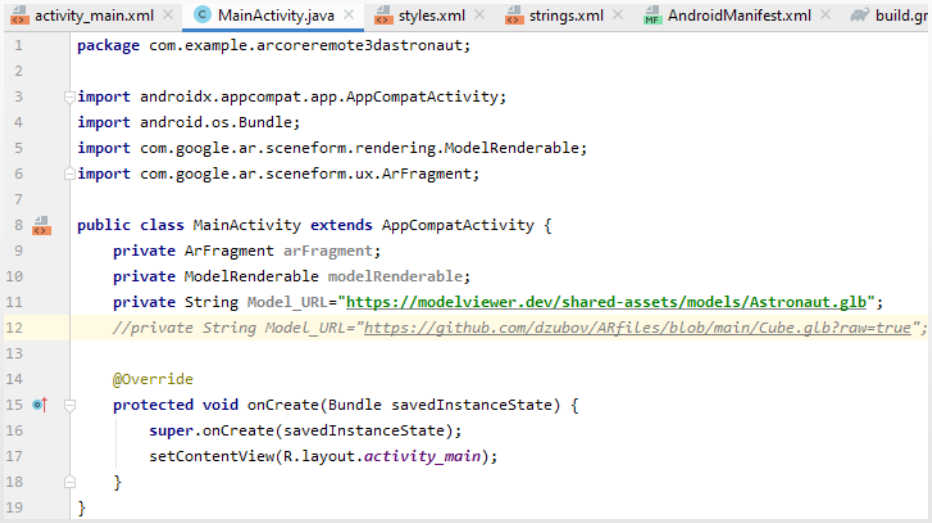
- Change the layout in `activity_main.xml` from `androidx.constraintlayout.widget.ConstraintLayout` to `RelativeLayout`, delete `TextView`, and add the element `fragment` as follows:

activity_main.xml | MainActivity.java | styles.xml | strings.xml | AndroidMa

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".MainActivity">
8
9      <fragment
10         android:id="@+id/fragment"
11         android:name="com.google.ar.sceneform.ux.ArFragment"
12         android:layout_width="match_parent"
13         android:layout_height="match_parent"/>
14
15 </RelativeLayout>
```

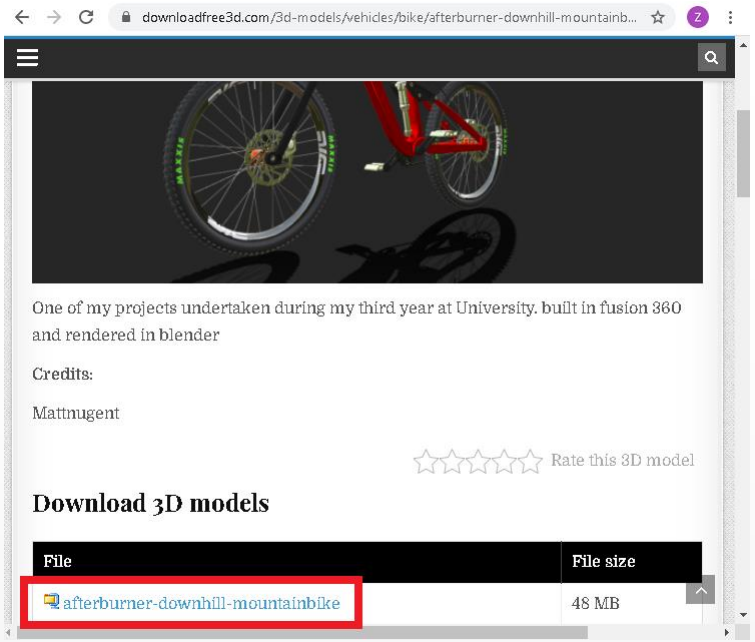
AR Java Android App in Android Studio (cont.)

- Declare a private field `arFragment` of type `ArFragment` (import a class `com.google.ar.sceneform.ux.ArFragment` for that) in `MainActivity.java`
- Declare a private field `modelRenderable` of type `ModelRenderable` (import a class `com.google.ar.sceneform.rendering.ModelRenderable` for that) in `MainActivity.java`
- Declare and initialize a private field `Model_URL` of type `String` with URL to 3D model in MainActivity.java



AR Java Android App in Android Studio (cont.)

- What is a GLB file?
  - GLB is the binary file format representation of 3D models saved in the GL Transmission Format (glTF). Information about 3D models such as node hierarchy, cameras, materials, animations and meshes is in binary format.
  - This binary format stores the glTF asset (JSON, .bin and images) in a binary blob. It also avoids the issue of increase in file size which happens in case of glTF. GLB file format results in compact file sizes, fast loading, complete 3D scene representation, and extensibility for further development.



https://downloadfree3d.com/file-format/glb/

https://grabcad.com/library?page=1&time=all_time&sort=recent&query=glb

https://www.hongkiat.com/blog/60-excellent-free-3d-model-websites/

https://docs.fileformat.com/3d/glb/#:~:text=GLB%20is%20the%20binary%20file,GL%20Transmission%20Format%20(glTF).&text=This%20binary%20format%20stores%20the,images)%20in%20a%20binary%20blob.

AR Java Android App in Android Studio (cont.)

- Where can we take GLB files?

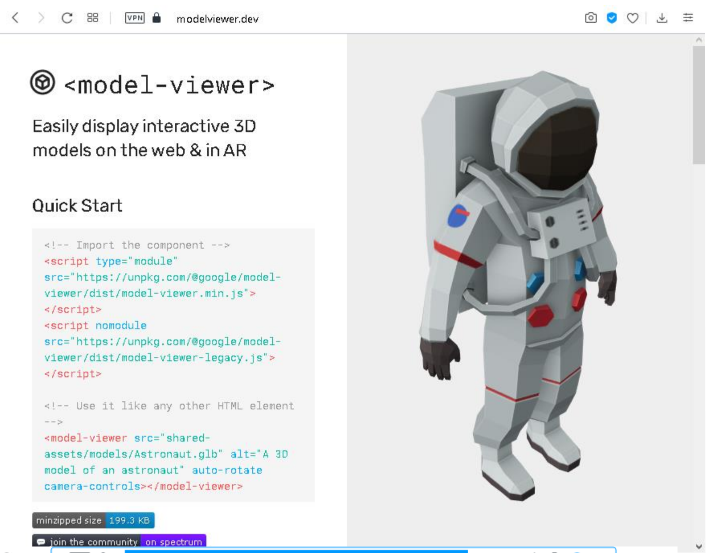
◦ Specialized websites like https://modelviewer.dev

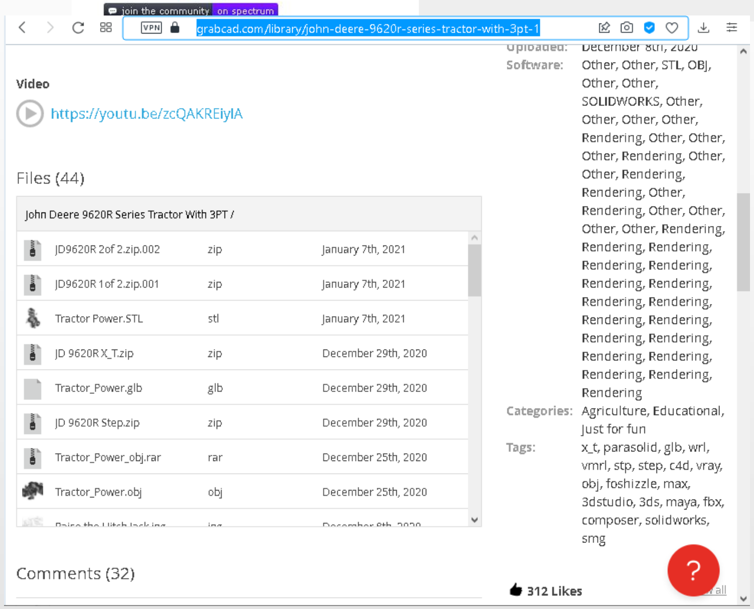
◦ General purpose websites to browse, distribute, and download 3D objects. The examples are as follows:

https://downloadfree3d.com/file-format/glb/

https://grabcad.com/library?page=1&time=all_time&sort=recent&query=glb

https://www.hongkiat.com/blog/60-excellent-free-3d-model-websites/





AR Java Android App in Android Studio (cont.)

- Inside the callback `onCreate`, assign the value to the field `arFragment`:

```
arFragment = (ArFragment) getSupportFragmentManager().findFragmentById(R.id.fragment);
```

- Create a method `setUpModel()` to load and setup 3D model:

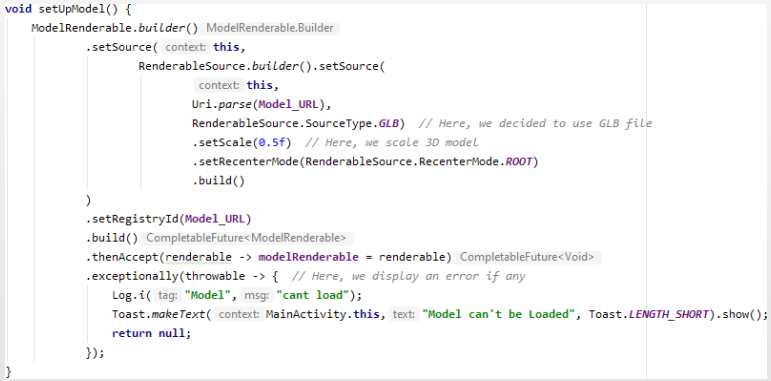
activity_main.xml | MainActivity.java | styles.xml | strings.xml | AndroidManifest.xml | build.gr

```
1  package com.example.arcoreremote3dastronaut;
2
3  import androidx.appcompat.app.AppCompatActivity;
4  import android.os.Bundle;
5  import com.google.ar.sceneform.rendering.ModelRenderable;
6  import com.google.ar.sceneform.ux.ArFragment;
7
8  public class MainActivity extends AppCompatActivity {
9      private ArFragment arFragment;
10     private ModelRenderable modelRenderable;
11     private String Model_URL="https://modelviewer.dev/shared-assets/models/Astronaut.glb";
12     //private String Model_URL="https://github.com/dzubov/ARfiles/blob/main/Cube.glb?raw=true";
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         arFragment = (ArFragment) getSupportFragmentManager().findFragmentById(R.id.fragment);
19         setUpModel(); // This method loads and setups 3D model
20     }
21
22     void setUpModel() {
23     }
24 }
```

AR Java Android App in Android Studio (cont.)

- Load and setup 3D model in method `setUpModel()`:

```
void setUpModel() {
    ModelRenderable.builder()
            .setSource(this,
                    RenderableSource.builder().setSource(
                            this,
                            Uri.parse(Model_URL),
                            RenderableSource.SourceType.GLB)  // Here, we decided to use GLB file
                            .setScale(0.5f)  // Here, we scale 3D model
                            .setRecenterMode(RenderableSource.RecenterMode.ROOT)
                            .build()
            )
            .setRegistryId(Model_URL)
            .build()
            .thenAccept(renderable -> modelRenderable = renderable)
            .exceptionally(throwable -> {  // Here, we display an error if any
                Log.i("Model","cant load");
                Toast.makeText(MainActivity.this,"Model can't be Loaded",
Toast.LENGTH_SHORT).show();
                return null;
            });
}
```



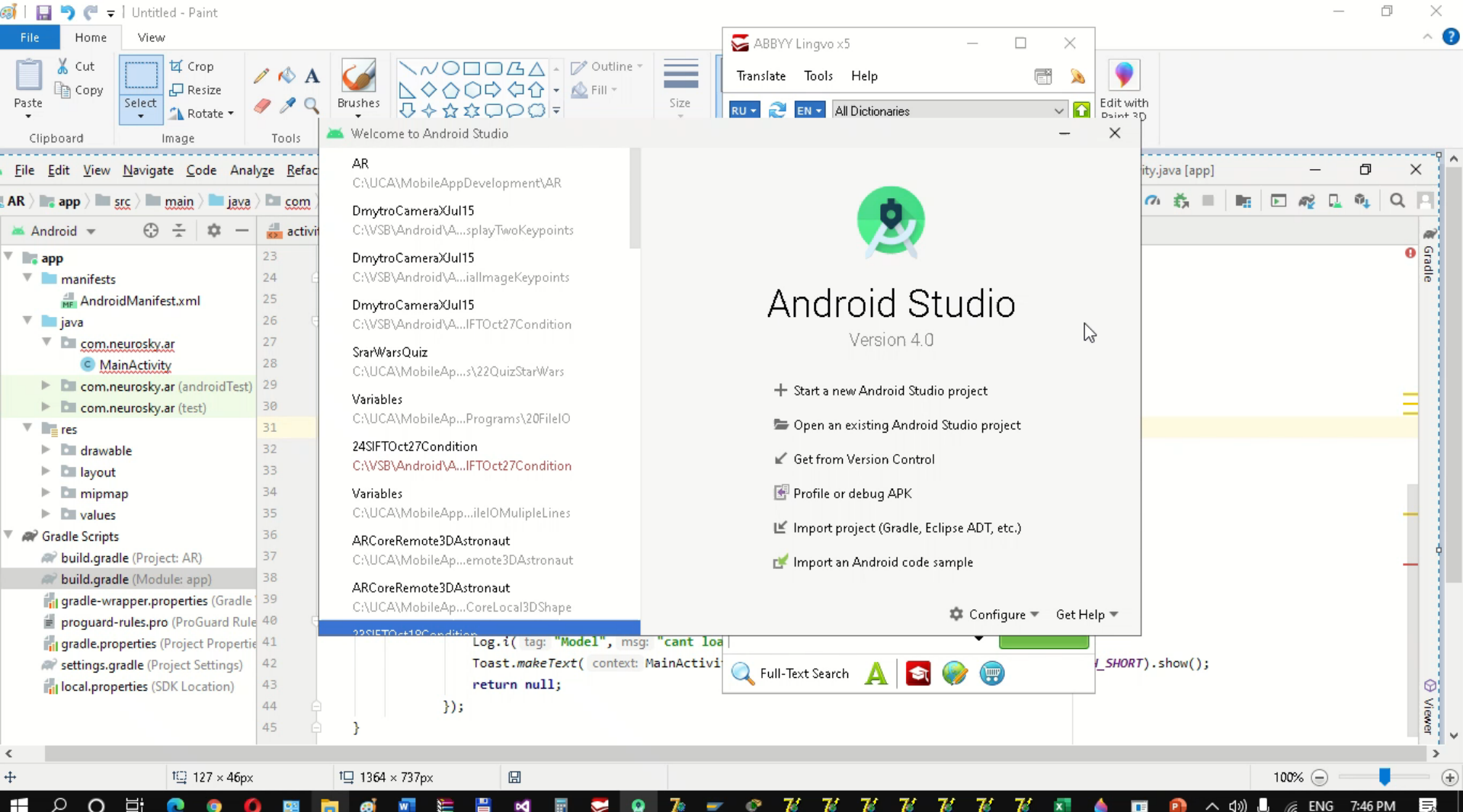
AR Java Android App in Android Studio (cont.)

- Setup the plane in the method `setUpPlane()`. Here, we also check if the user taps the screen; if yes, we have a specific position to place our 3D model with possibility to scale and drag it.

```
private void setUpPlane() {
// This method setups the plane. Here, we check if the user taps on the screen.
// If yes, we have a specific position where we place our 3D model
   arFragment.setOnTapArPlaneListener(new BaseArFragment.OnTapArPlaneListener() {
// Here, we check if the user taps on the screen
       @Override
       public void onTapPlane(HitResult hitResult, Plane plane, MotionEvent motionEvent) {
// If the user taps on the screen, we have a position on the screen
           Anchor anchor = hitResult.createAnchor();
           AnchorNode anchorNode = new AnchorNode(anchor);
           anchorNode.setParent(arFragment.getArSceneView().getScene());
           createModel(anchorNode);
// Here, we create 3D model on the smartphone's screen at the specific position
       }
   });
}

private void createModel(AnchorNode anchorNode){
// Here, we create 3D model on the smartphone's screen at the specific position
   TransformableNode node = new TransformableNode(arFragment.getTransformationSystem());
// TransformableNode allows us to scale and drag 3D model
   node.setParent(anchorNode);
   node.setRenderable(modelRenderable);
   node.select();
}
```

AR Java Android App in Android Studio

- Short video on AR:

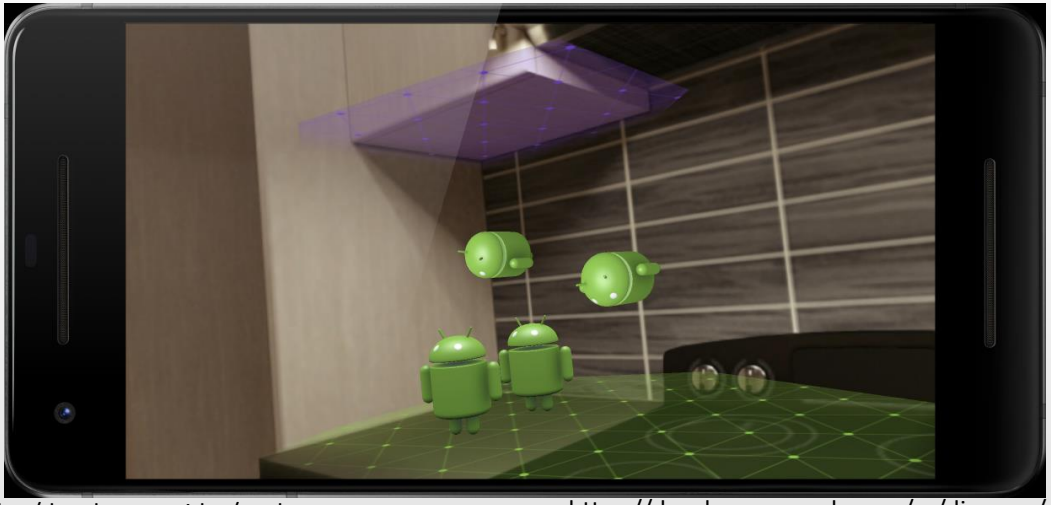


AR Java Android App in Android Studio (cont.)

- Terminology:
  - **Scene:** This is the place where all our 3D objects will be rendered. This scene is hosted by the AR Fragment which we included in the layout. An anchor node is attached to this screen which acts as the root of the tree and all the other objects are rendered as its objects.
  - **HitResult:** This is an imaginary line (or a ray) coming from infinity which gives the point of intersection of itself with a real-world object.
  - **Anchor:** An anchor is a fixed location and orientation in the real world. It can be understood as the (x, y, z) coordinate in the 3D space. We can get an anchor's post information from it. Pose is the position and orientation of the object in the scene. This is used to transform the object's local coordinate space into real-world coordinate space.
  - **AnchorNode:** This is the node that automatically positions itself in the world. This is the first node that gets set when the plane is detected.
  - **TransformableNode:** It is a node that can be interacted with. It can be moved around, scaled, rotated and much more.

AR Java Android App in Android Studio (cont.)

- What are the ARCore plane and anchors?
  - ***Anchors*** are used to make virtual objects appear to stay in place in an AR scene
  - ARCore looks for clusters of feature points that appear to lie on common horizontal or vertical surfaces, like tables or walls, and makes these surfaces available to the app as ***planes***. ARCore can also determine each plane's boundary and make that information available to the app. The app can use this info to place virtual objects resting on flat surfaces.

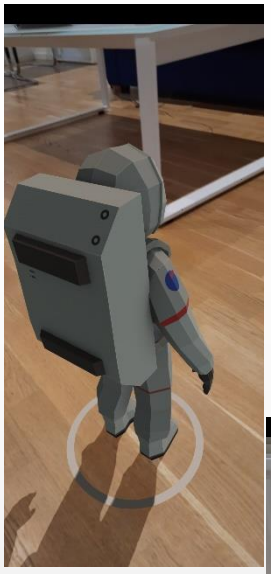


https://developers.google.com/ar/develop/developer-guides/anchors

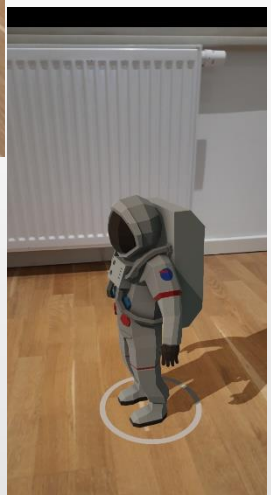
https://developers.google.com/ar/discover/concepts#environmental_understanding

AR Java Android App in Android Studio (cont.)

- Let’s start our app :)

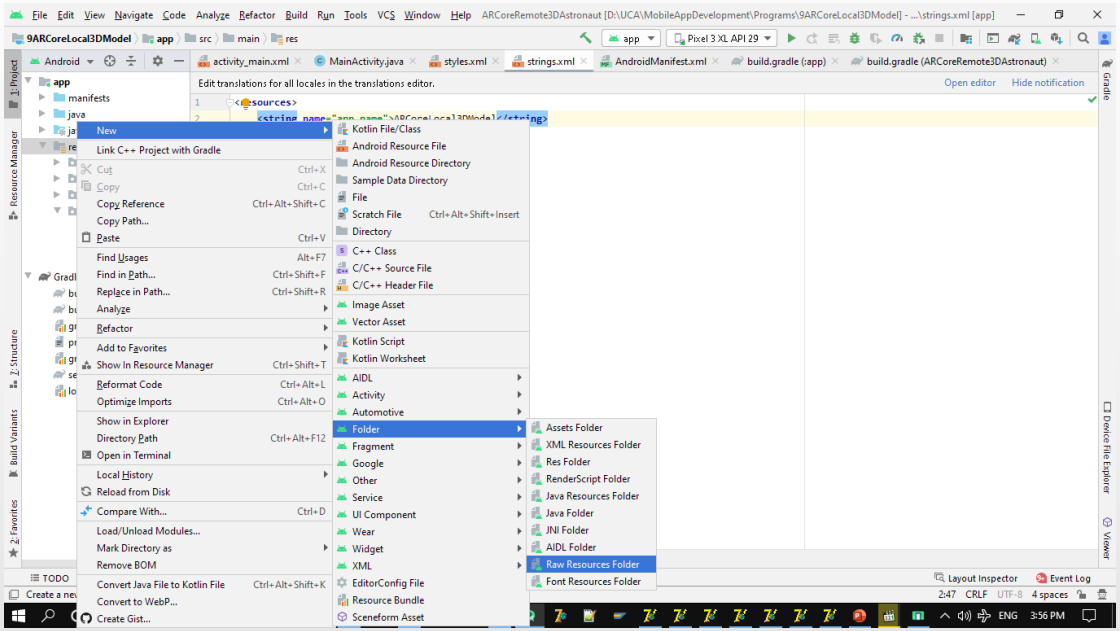



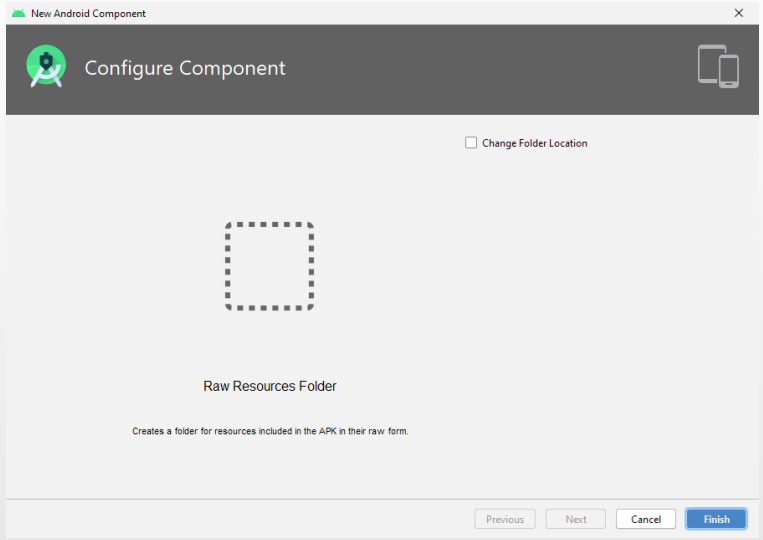



AR Java Android App in Android Studio: A Local 3D Model

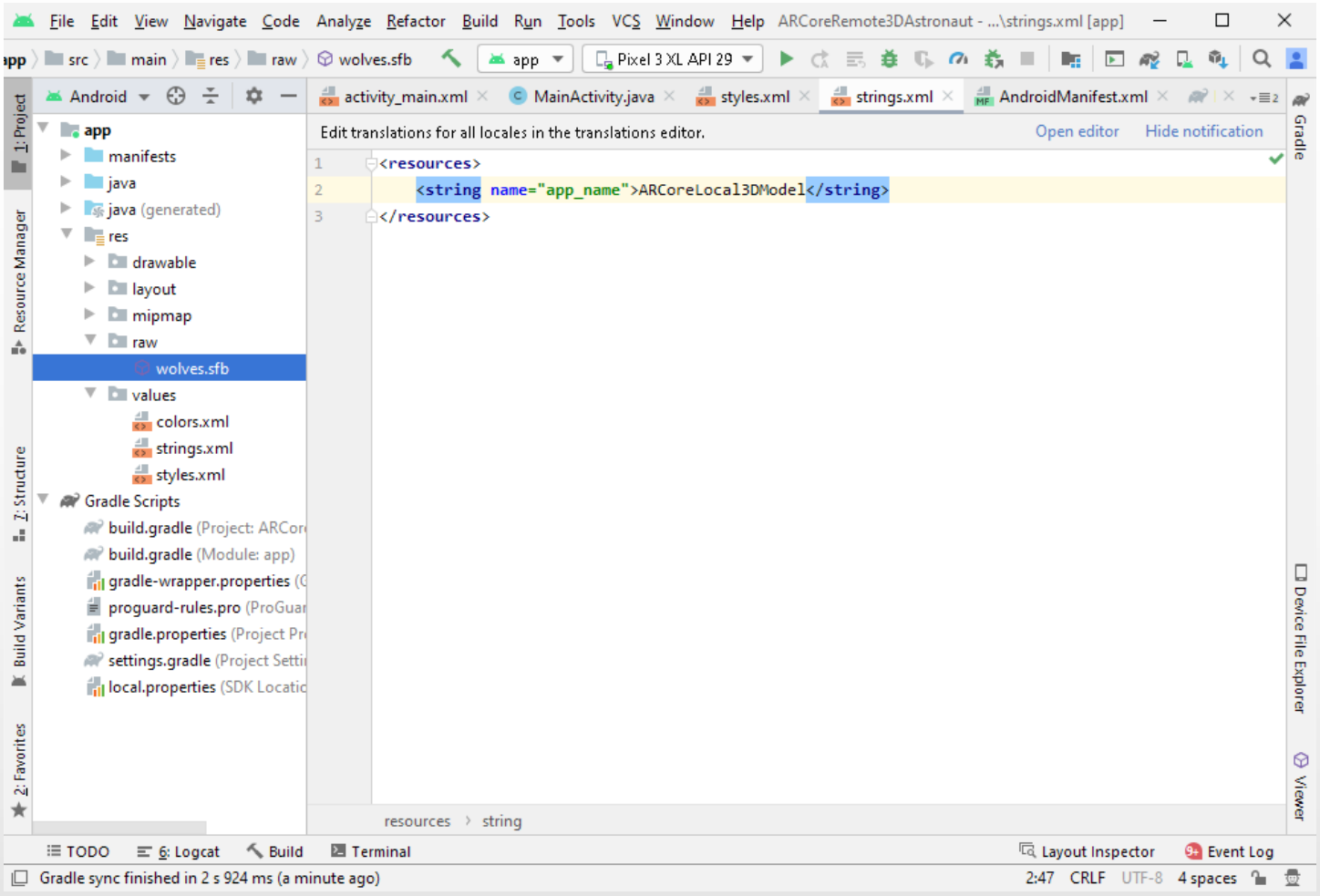
- In this example, we place a 3D model (SFB file, the Sceneform Binary asset) into the folder `raw` first. Then, we display this 3D model on the screen at the specific position.
  - Create a Raw Resources Folder





AR Java Android App in Android Studio: A Local 3D Model (cont.)

- Copy the SFB file to the `Raw Resources Folder`:



AR Java Android App in Android Studio: A Local 3D Model (cont.)

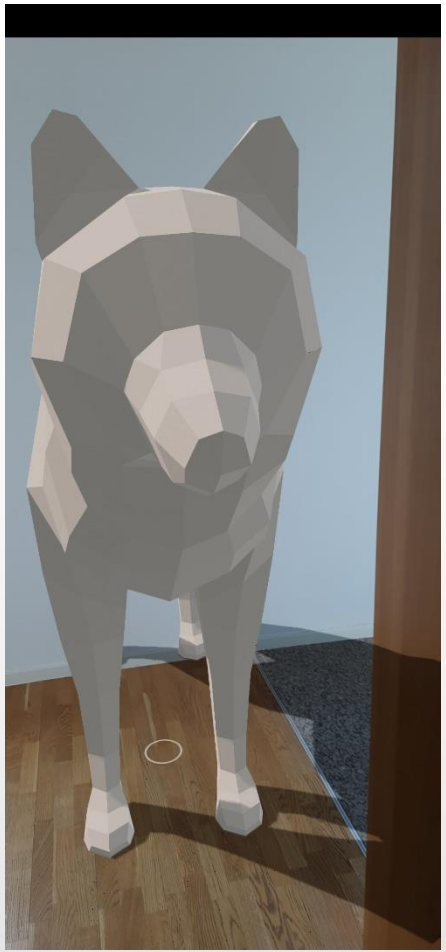
- Modify method `setUpModel()` as follows:

```
void setUpModel() {
    ModelRenderable.builder()
            .setSource(this, R.raw.wolves)
            .build()
            .thenAccept(renderable -> modelRenderable = renderable)
            .exceptionally(throwable -> {
                Toast.makeText(MainActivity.this,"Model   can't   be
Loaded", Toast.LENGTH_SHORT).show();
                return null;
            });
}
```

```
void setUpModel() {
    ModelRenderable.builder() ModelRenderable.Builder
            .setSource( context: this, R.raw.wolves)
            .build() CompletableFuture<ModelRenderable>
            .thenAccept(renderable -> modelRenderable = renderable) CompletableFuture<Void>
            .exceptionally(throwable -> {
                Toast.makeText( context: MainActivity.this, text: "Model can't be Loaded", Toast.LENGTH_SHORT).show();
                return null;
            });
}
```

AR Java Android App in Android Studio: A Local 3D Model (cont.)

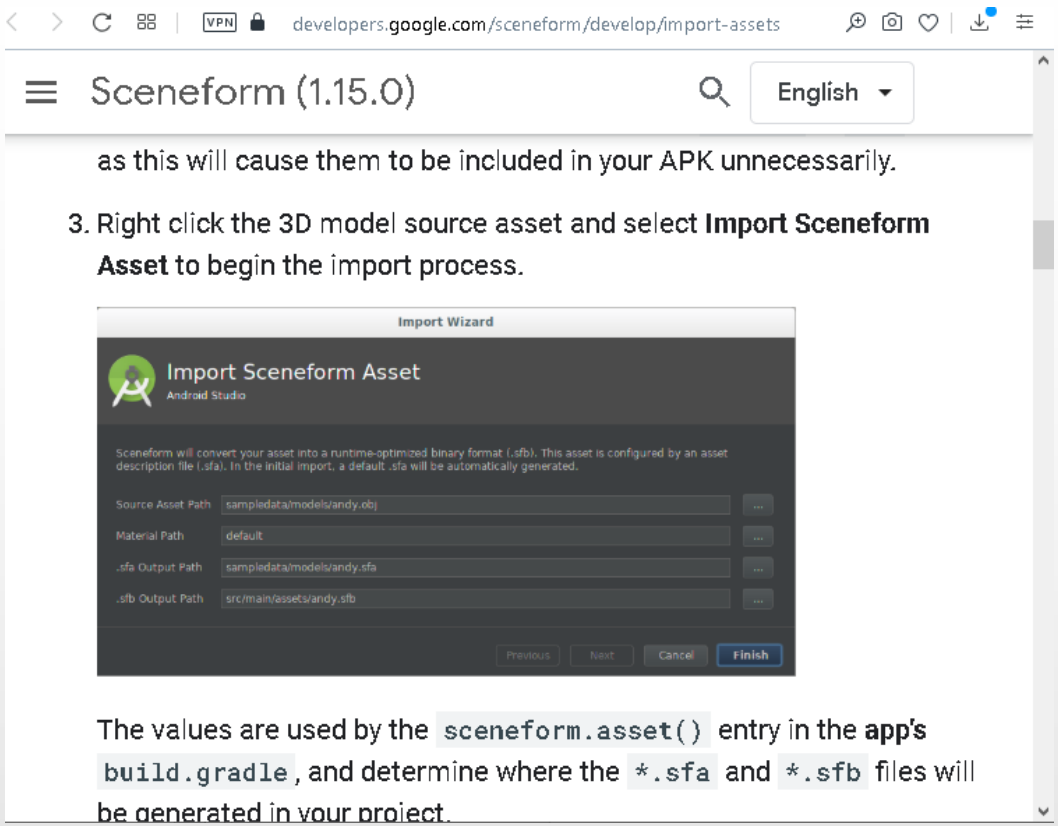
- Let’s start the app :)



AR Java Android App in Android Studio: A Local 3D Model (cont.)

- Where we can find Sceneform Binary assets, i.e., SFB files?

  - We can import them from OBJ, glTF, and FBX files (https://developers.google.com/sceneform/develop/import-assets). **Okay, this approach doesn't work for my Android Studio 4. So, what should we do? :)**



AR Java Android App in Android Studio: A Local 3D Model (shape)

- In this app, we use the red sphere as a 3D model
  - Modify method setUpModel() as follows:

```
void setUpModel() {
  MaterialFactory.makeOpaqueWithColor(this, new Color(android.graphics.Color.RED))
     .thenAccept(material -> {
        modelRenderable = ShapeFactory.makeSphere(0.2f, new Vector3(0, 0.2f, 0.2f), material);
     });
}
```

```
void setUpModel() {
    MaterialFactory.makeOpaqueWithColor( context: this, new Color(android.graphics.Color.RED))
            .thenAccept(material -> {
                modelRenderable = ShapeFactory.makeSphere( radius: 0.2f, new Vector3( v: 0,  v1: 0.2f,  v2: 0.2f), material);
            });
}
```

Do you have any questions or comments?

Thank you
for your attention !

In this presentation:

- Some icons were downloaded from flaticon.com and iconscout.com