



INSTITUTO POLITÉCNICO NACIONAL



ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA: APPLICATION DEVELOPMENT FOR MOBILE DEVICES

PROFESOR: CIFUENTES ALVAREZ ALEJANDRO SIGFRIDO

PRESENTA:

RAMIREZ BENITEZ BRAYAN

GRUPO: 3CM17

“TERCER EXAMEN – APLICACIÓN DE REALIDAD AUMENTADA”

CIUDAD DE MEXICO A 14 DE JUNIO DE 2022

Índice

1. Introducción	3
2. Objetivo	3
3. Conceptos.....	3
4. Desarrollo.....	3
Paso 1: Preparación del Hardware.	4
Comprobación de requisitos.....	4
Paso 2: Crear un nuevo proyecto	5
Paso 3: Obtener un modelo 3D para la aplicación.	8
Paso 4: Descarga y configuración de SceneForm 1.16.0.....	12
Paso 5: Corrección de errores.....	17
Paso 6: Trabajar con el archivo activity_main.xml	22
Paso 7: Trabajar con el archivo MainActivity.java	26
5. Código.....	28
6. Capturas de la ejecución en el dispositivo.....	33
7. Conclusiones	35
8. Bibliografía	35

1. Introducción

En este proyecto se presenta una aplicación que permite observar un modelo en realidad aumentada. Comenzando con conceptos sobre realidad aumentada, herramientas y aplicaciones, para posteriormente, paso a paso, mostrar cómo incorporar un modelo 3D propio dentro de una aplicación de realidad aumentada, que permite integrar el modelo en el mundo real, empleando el teléfono móvil. Con el programa Android Studio, la plataforma de Google ARCore y código programado en Java.

2. Objetivo

Diseñar una aplicación de Realidad Aumentada AR que permita su ejecución en un dispositivo móvil Android.

3. Conceptos

A continuación, se muestran conceptos básicos para trabajar con realidad aumentada en dispositivos Android.

ARCore: es una plataforma de realidad aumentada. ARCore realmente ayuda al teléfono a detectar su entorno e interactuar con el mundo. ARCore utiliza principalmente 3 principios clave: seguimiento del movimiento, comprensión del entorno y estimación de la luz.

Lista de dispositivos compatibles con ARCore:

<https://developers.google.com/ar/devices>

Sceneform: es un marco 3D que ayuda a los desarrolladores a crear aplicaciones ARCore sin saber mucho sobre OpenGL. Sceneform viene con muchas características como verificar el permiso de la cámara, manipular activos 3D y mucho más.

4. Desarrollo

Para poder ejecutar la aplicación en nuestro dispositivo necesitaremos comprobar ciertos requisitos y cambiar algunos aspectos de este. Además, necesitaremos descargar en nuestra computadora algunas aplicaciones como la que será el entorno de desarrollo.

Paso 1: Preparación del Hardware.

Comprobación de requisitos

El dispositivo que utilizaremos debe tener ciertos requisitos mínimos para poder ser utilizado ya que tiene que soportar el programa ARCore el que es indispensable para el uso de la futura aplicación. Para esto necesitaremos un smartphone real, ya que ARCore no soporta emuladores de Android. Por lo que el dispositivo en cuestión debe de tener una versión de Android 7.0 o superior. En este caso se utilizará el dispositivo “MOTOROLA ONE MACRO” que tiene Android 10.

Para confirmar si el dispositivo es compatible con ARCore se puede visitar el siguiente enlace en el que aparece un listado con los modelos compatibles:

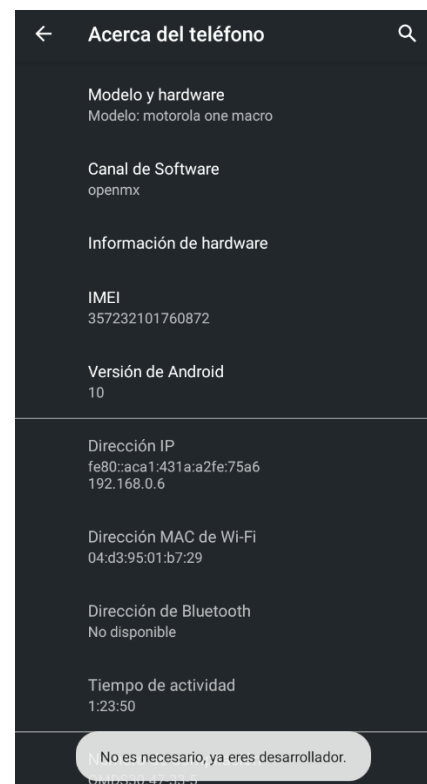
<https://developers.google.com/ar/devices>

Motorola	motorola one hyper	Admite varias resoluciones de textura de GPU: 1080p, 720p, 480p
Motorola	motorola one macro	
Motorola	motorola one power	
Motorola	motorola one vision	
Motorola	motorola one zoom	
Motorola	motorola Razr(2020)	Admite API de Depth
Motorola	moto x ⁴	Requiere Android 8.0 o una versión posterior.

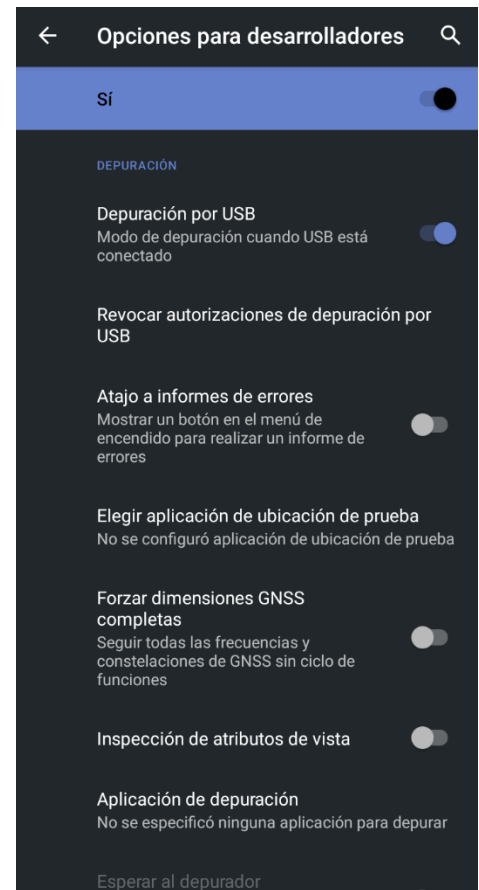
Una vez comprobado que nuestro dispositivo es compatible con la aplicación, debemos pasar a hacer ciertos ajustes en el panel de control. El primero es activar las opciones de desarrolladores del móvil en cuestión. En cada dispositivo, según la marca y modelo, cambia la forma de hacerlo. Pero generalmente todas se realizan de la siguiente forma:

Ajustes > Acerca del teléfono / Sobre el teléfono >

(El dispositivo ya tenía activadas las opciones de desarrollador por lo que ya no es necesario)

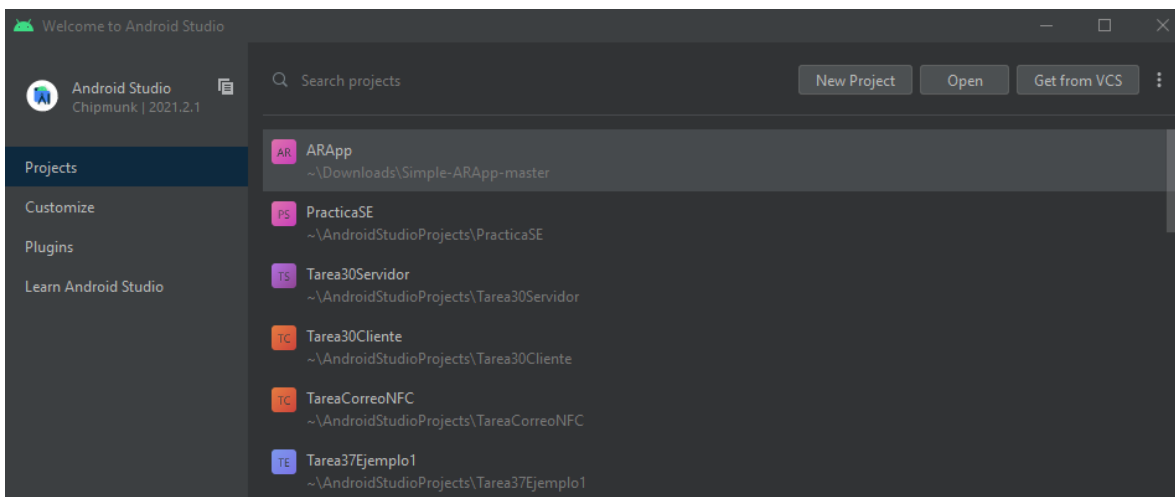


A continuación, hay que activar la Depuración USB. En este apartado cada modelo de dispositivo es diferente, pero la mayoría suelen hacer aparecer una notificación de este apartado cuando se conecta desde el ordenador por Android Studio.

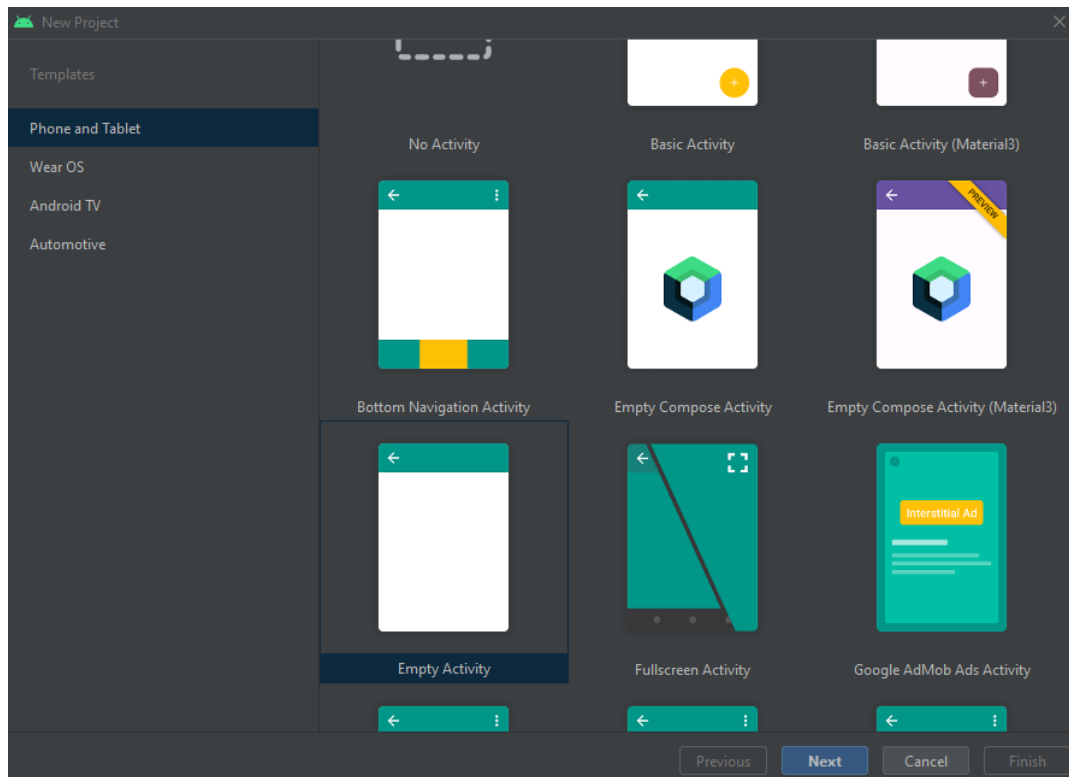


Paso 2: Crear un nuevo proyecto

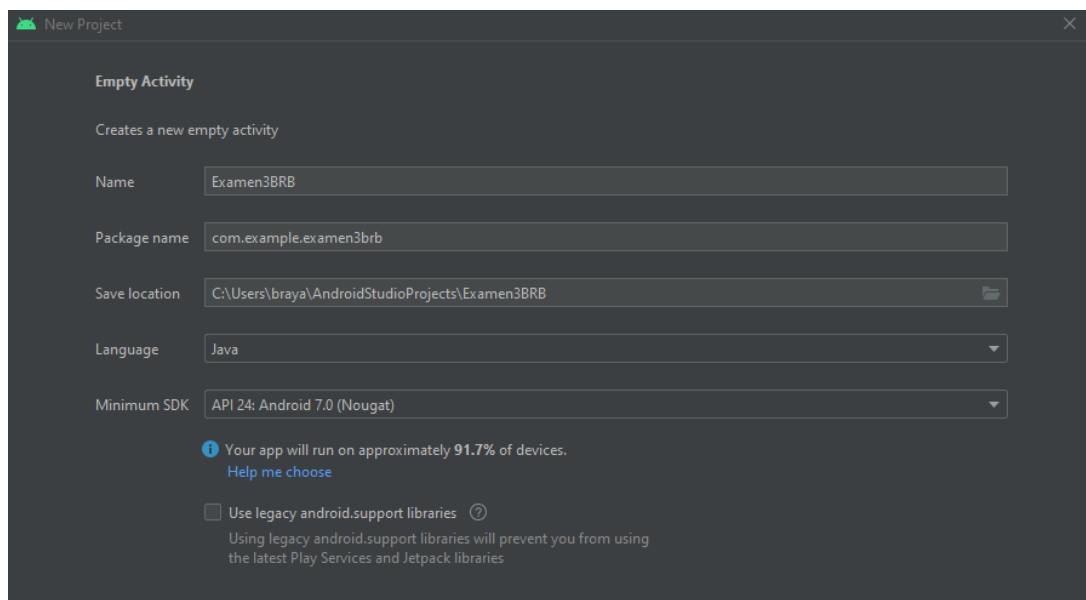
Seleccione la opción para 'New Project' de Android



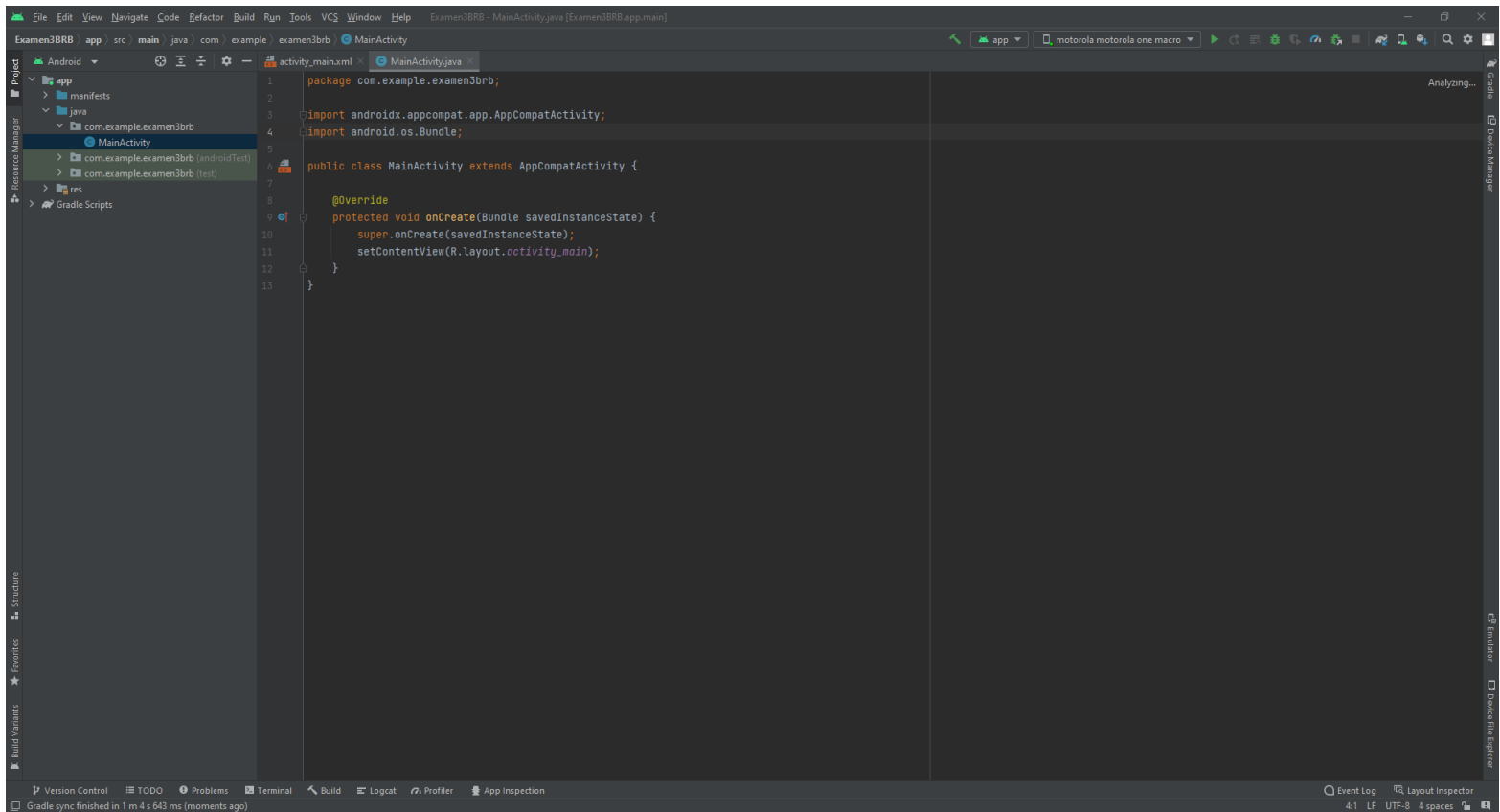
El siguiente paso es elegir la Actividad para móvil. La actividad en Android se refiere a una sola pantalla con una interfaz de usuario, se recomienda "Empty Activity".



Luego, asignamos un nombre 'Name', en Language escogemos Java y seleccionamos el SDK mínimo para seleccionar el sistema operativo que debe tener la versión mínima para ejecutar la aplicación, aquí "API 24 Android 7.0 (Nougat)" será nuestro SDK mínimo, y los teléfonos y tabletas con versiones anteriores a este sistema operativo no podrán ejecutar sus aplicaciones. Hacemos clic en "FINISH".



Finalmente, se crea una aplicación predeterminada con todos los archivos predeterminados. Y ahora podemos comenzar a escribir el código de la aplicación.

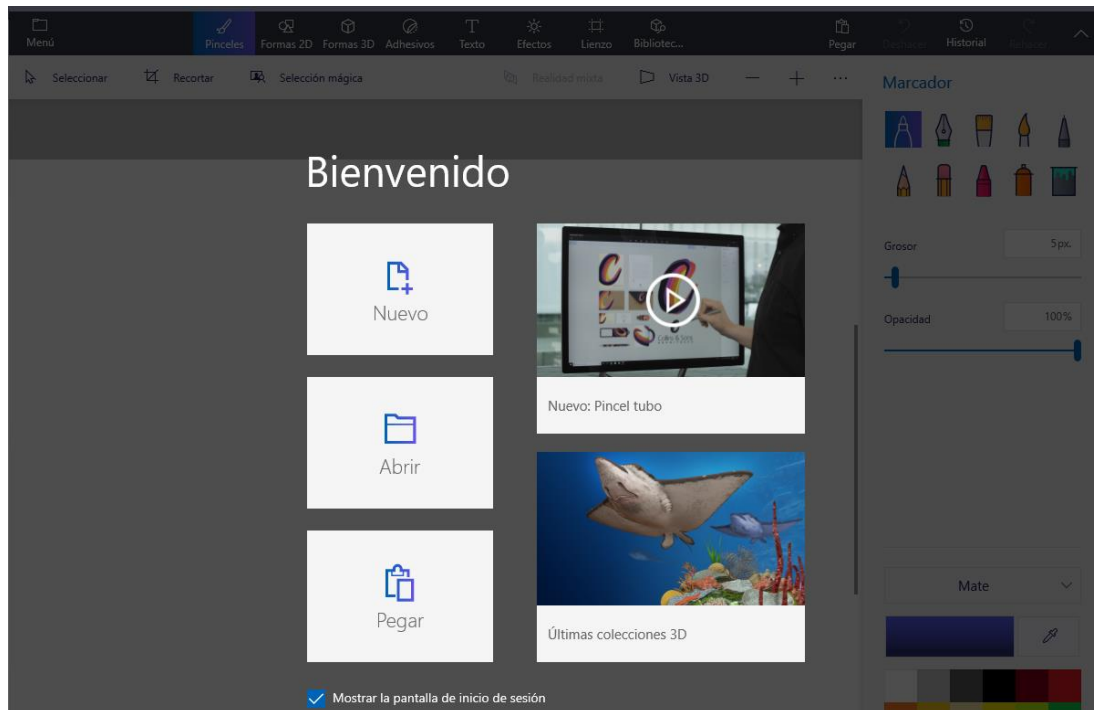


Paso 3: Obtener un modelo 3D para la aplicación.

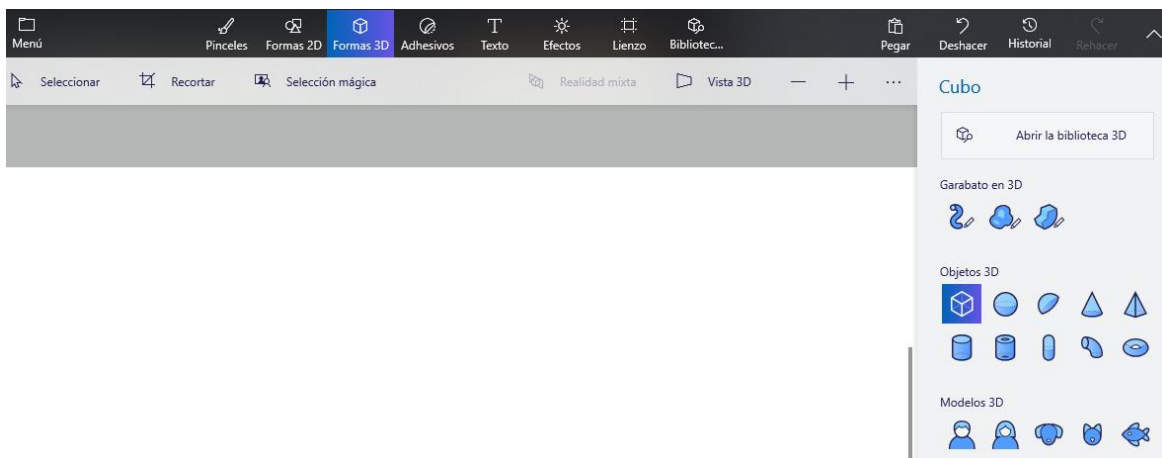
Sceneform 1.16.0 solo admite archivos glTF. glTF significa formato de transmisión GL. Ahora los archivos .glb son una versión binaria del formato de transmisión GL. Estos tipos de archivos de modelos 3D se utilizan en VR, AR porque admiten movimiento y animación.

Para el modelo 3D, debemos obtener un archivo .glb. Para esto existen dos formas, podemos tomar un modelo 3D, descargarlo de la web o hacer nosotros mismos. Para este proyecto se creó un modelo mediante la aplicación Paint 3D.

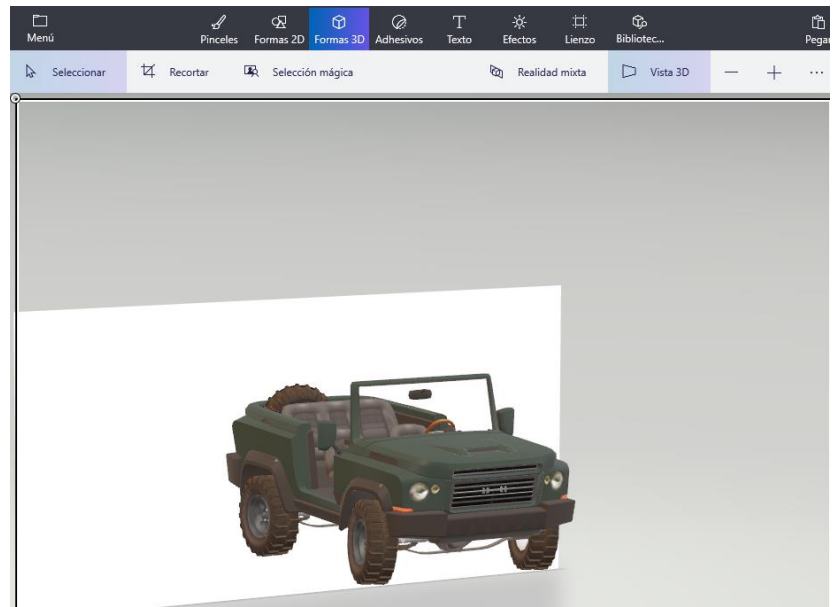
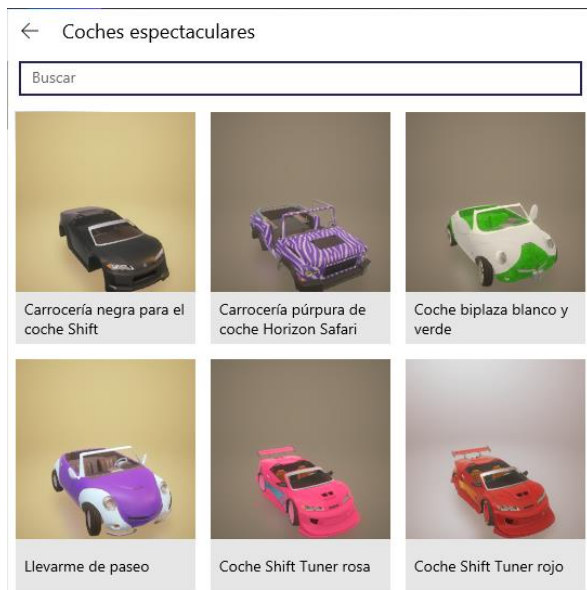
Primero abrimos Paint 3D y creamos un nuevo proyecto.



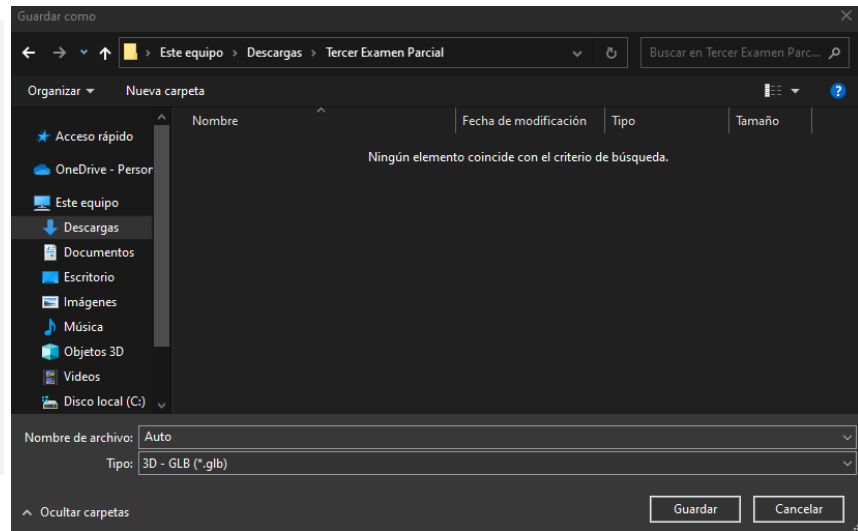
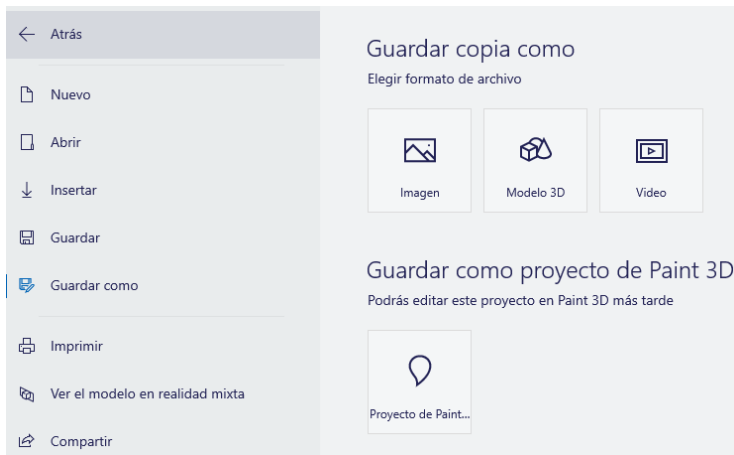
Seleccionamos 'Formas 3D', luego, 'Abrir la biblioteca 3D'



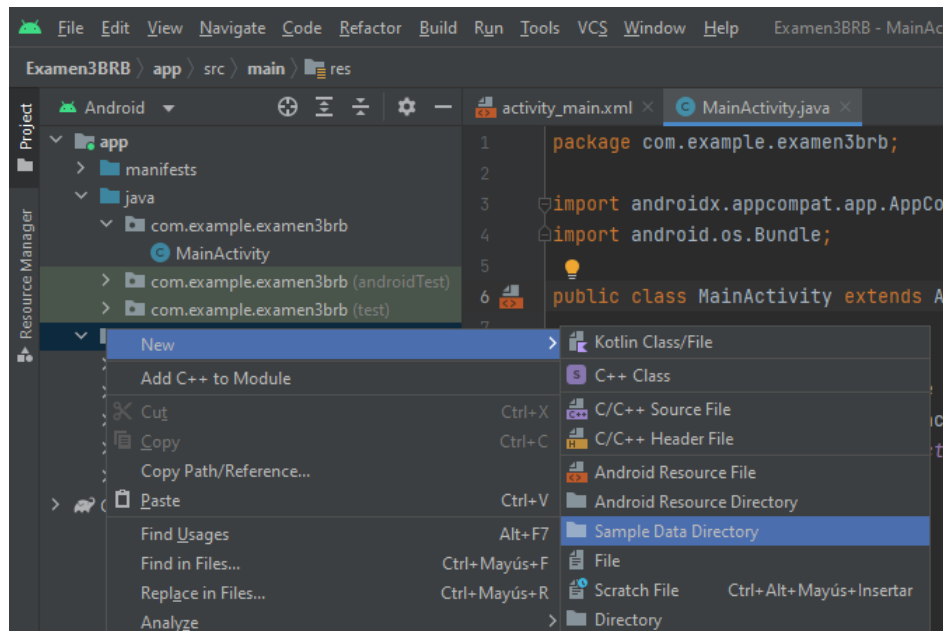
Seleccionamos un modelo 3D



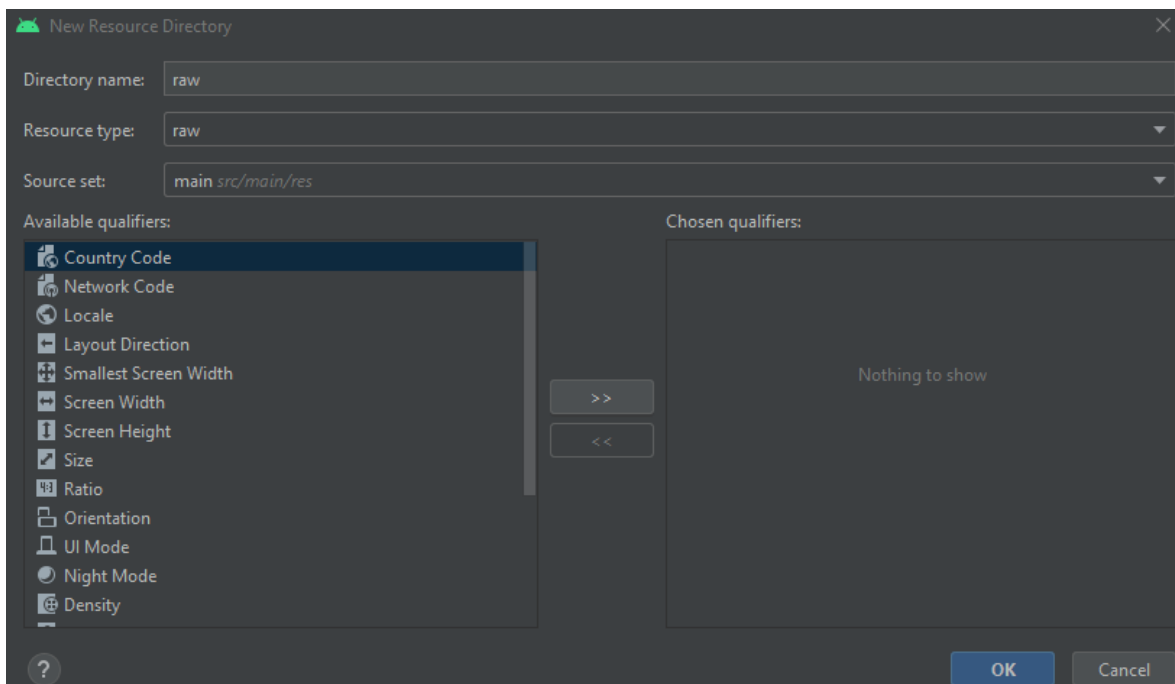
Luego guardamos el modelo como 'Modelo 3D' con extensión .glb y le asignamos un nombre, finalmente hacemos clic en guardar.



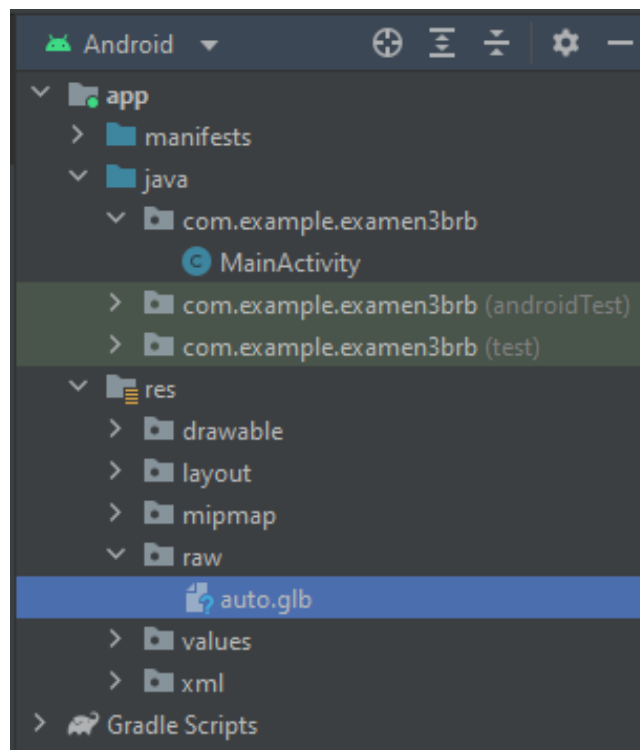
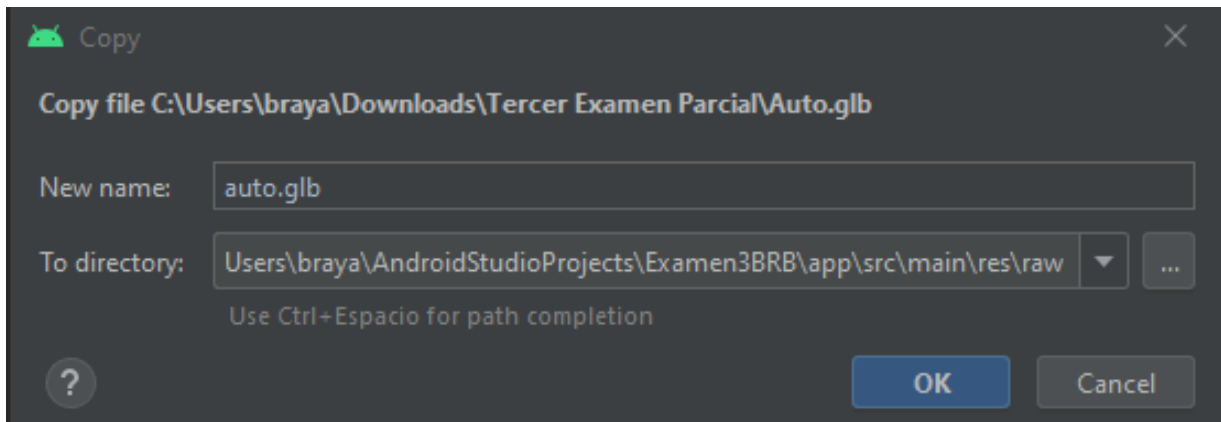
Regresamos a Android Studio y en el panel izquierdo, haga clic derecho en el directorio 'res'. Vaya a New > Android Resource Directory. Una ventana aparecerá.



Cambiamos el tipo de recurso: a Raw. Hacemos clic en Aceptar. Se genera una carpeta sin formato, en el directorio res.



Para terminar copiamos el . glb de ese directorio donde lo guardamos y lo pegamos en la carpeta sin procesar.



Paso 4: Descarga y configuración de SceneForm 1.16.0

Para la realidad aumentada necesitamos Sceneform SDK. Así que usaremos Sceneform 1.16.0 SDK y lo configuraremos manualmente.

Entramos al siguiente enlace de GitHub.

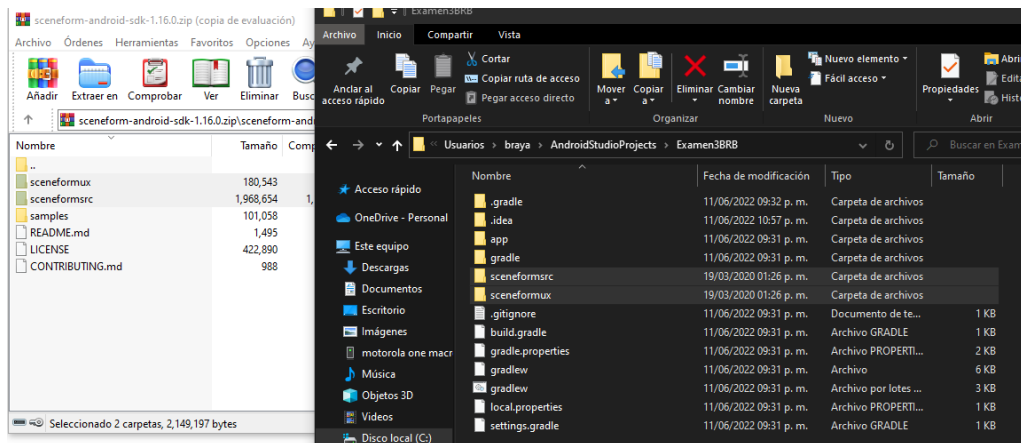
<https://github.com/google-ar/sceneform-android-sdk/releases/tag/v1.16.0>

Descargamos el archivo "sceneform-android-sdk-1.16.0.zip".

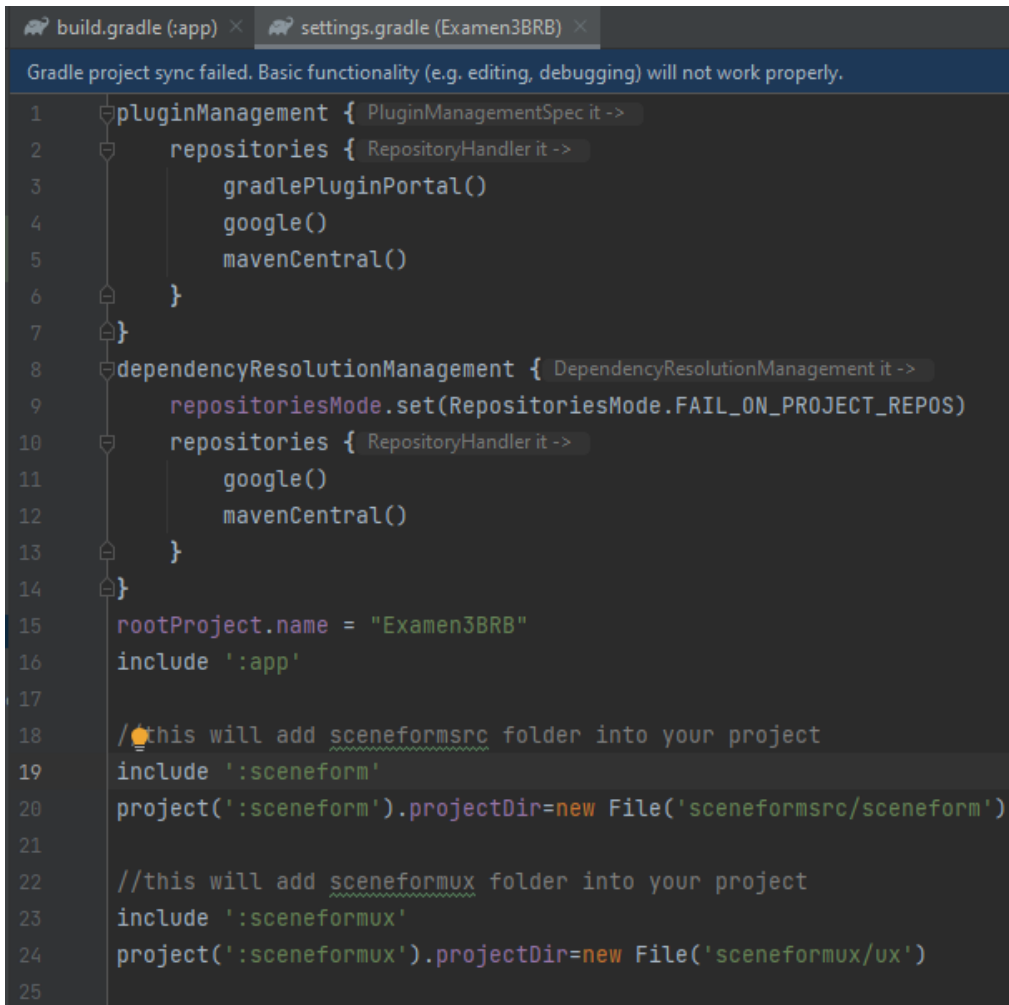
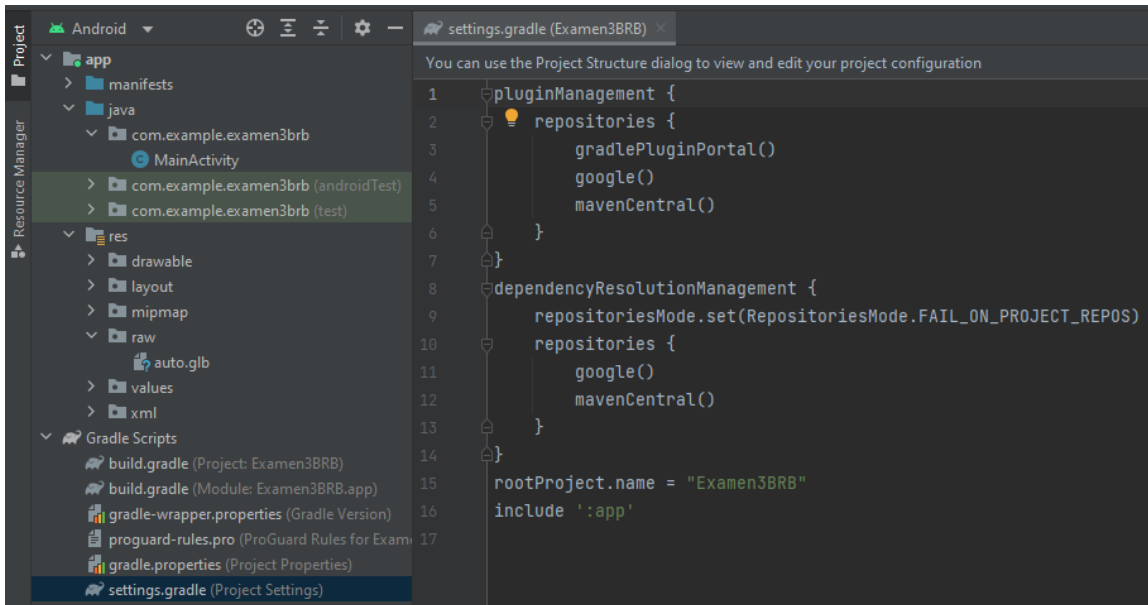
The screenshot shows the GitHub release page for 'Sceneform SDK for Android v1.16.0'. The page is in dark mode. At the top, it says 'Latest' and 'noelvictor1 released this 19 Mar 2020'. Below this, there are statistics: '5 commits to master since this release', 'v1.16.0', and '4d7b660'. The main section is titled 'New APIs and capabilities' and lists four bullet points: 'Adds source code access for Sceneform.', 'Uses the Filament maven release for Sceneform's Filament dependency', 'Adds a sample using Filament's gltfio loader including fast runtime glTF loading and glTF animation.', and 'Updated Filament to v1.4.5.'. Below this is a 'Deprecations' section with five bullet points: 'Removes all legacy Sceneform samples which are built on the closed-source and legacy Sceneform model codebase.', 'Removes the Sceneform assets library (replaced with gltfio loader from Filament).', 'Removes the Sceneform animation library (replaced with gltfio loader from Filament).', 'Deprecates support for SFB & the SFB Sceneform Android Studio plugin (gltf can be used instead).', and 'Sceneform API reference documentation for the previous version (1.15.0) has moved to developers.google.com/sceneform.'. At the bottom, there is an 'Assets' section with three items: 'sceneform-android-sdk-1.16.0.zip' (1.33 MB, 19 Mar 2020), 'Source code (zip)' (26 Mar 2020), and 'Source code (tar.gz)' (26 Mar 2020).

Asset	Size	Released
sceneform-android-sdk-1.16.0.zip	1.33 MB	19 Mar 2020
Source code (zip)		26 Mar 2020
Source code (tar.gz)		26 Mar 2020

Extraemos las carpetas 'sceneformsrc' y 'scenemux', donde se creamos nuestro proyecto. ("C:\Users\braya\AndroidStudioProjects\Examen3BRB" para mí)

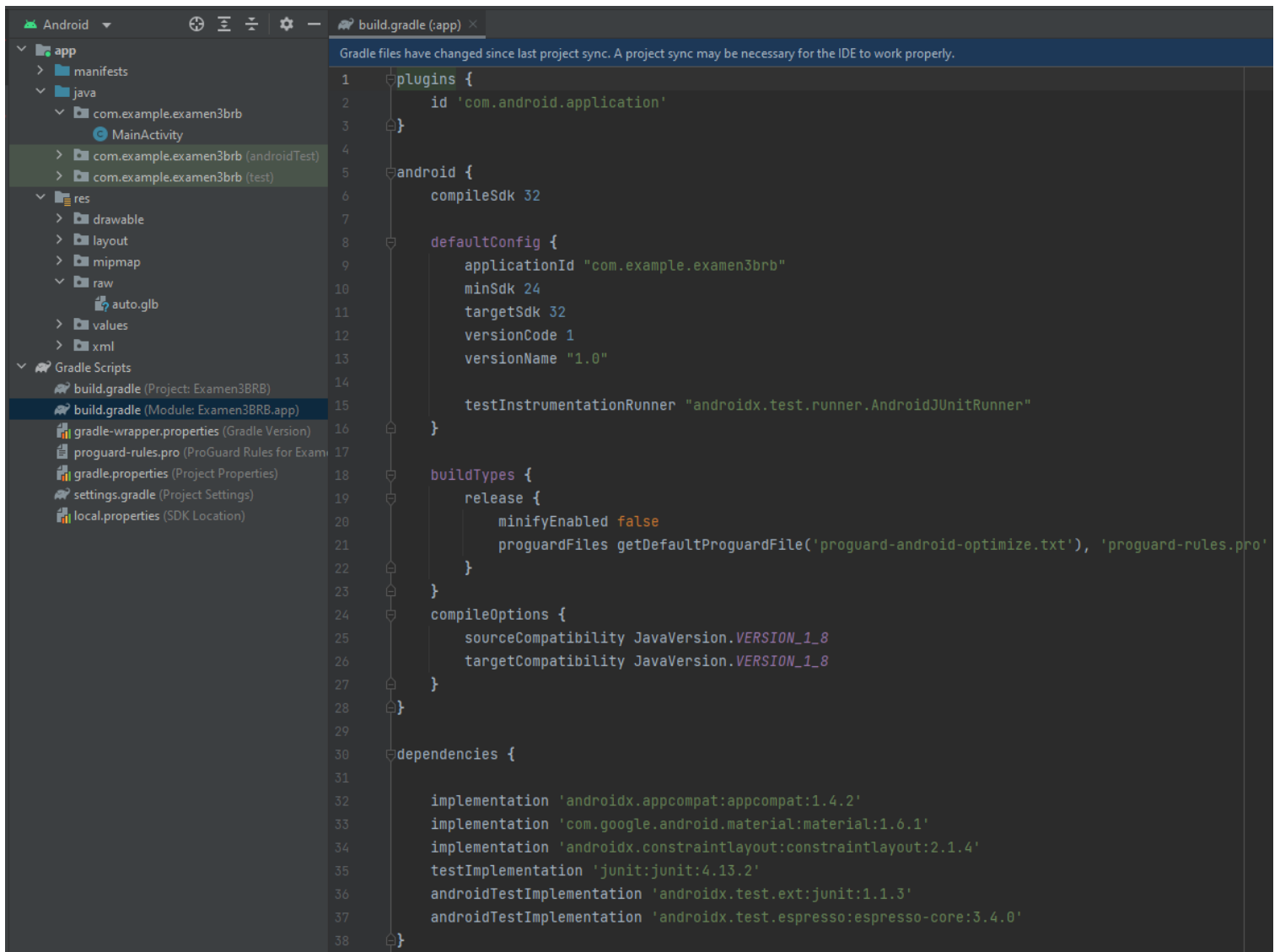


Regresamos a Android Studio y en Gradle Scripts> settings.gradle (Configuración del proyecto) agregamos estas líneas:

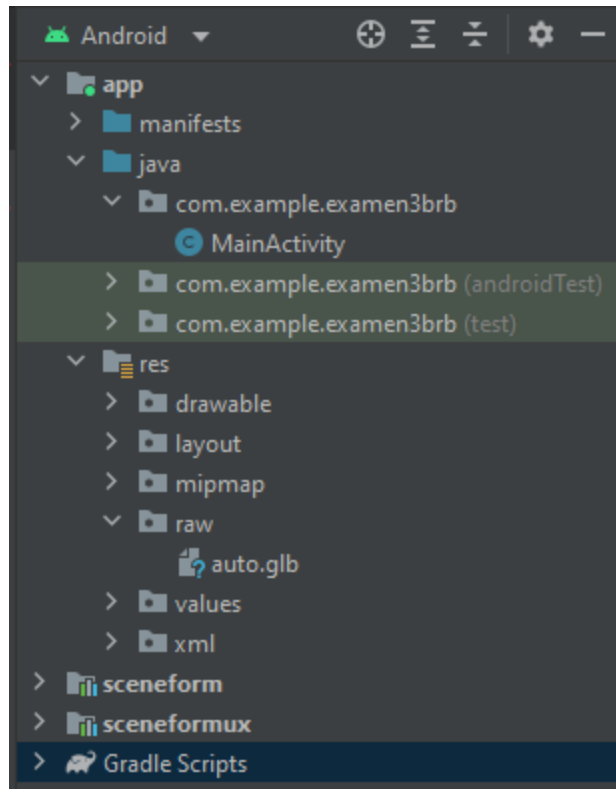


Luego, en el mismo archivo dentro del bloque "android" y justo después del bloque "buildTypes", agregue estas líneas (si aún no está allí):

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



Después de todo, estos cambios hacen clic en 'Sincronizar ahora' en la ventana emergente de arriba. Ahora la estructura de archivos de Android se verá así.



Luego abrimos app > manifest> AndroidManifest.xml

Agregamos estas líneas antes del bloque " aplicación ":

```
<!--This helps to permit the user to access Camera-->
<uses-permission android:name="android.permission.CAMERA" />
<!--This helps to check a specific feature in the phone's hardware,
    here it is OpenGL ES version 3-->
<uses-feature
    android:glEsVersion="0x00030000"
    android:required="true" />
<!--Here it is checking for AR feature in phone camera-->
<uses-feature
    android:name="android.hardware.camera.ar"
    android:required="true" />
```

Después de eso, agregamos esta línea antes del bloque " actividad ".

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

A continuación, se muestra el código completo para el archivo AndroidManifest.xml.

```
AndroidManifest.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      package="com.example.examen3brb">
5      <!--This helps to permit the user to access Camera-->
6      <uses-permission android:name="android.permission.CAMERA" />
7      <!--This helps to check a specific feature in the phone's hardware,
8          here it is OpenGL ES version 3-->
9      <uses-feature
10         android:glEsVersion="0x00030000"
11         android:required="true" />
12      <!--Here it is checking for AR feature in phone camera-->
13      <uses-feature
14         android:name="android.hardware.camera.ar"
15         android:required="true" />
16
17      <application
18         android:allowBackup="true"
19         android:dataExtractionRules="@xml/data_extraction_rules"
20         android:fullBackupContent="@xml/backup_rules"
21         android:icon="@mipmap/ic_launcher"
22         android:label="Examen3BRB"
23         android:roundIcon="@mipmap/ic_launcher_round"
24         android:supportsRtl="true"
25         android:theme="@style/Theme.Examen3BRB"
26         tools:targetApi="31">
27
28         <meta-data android:name="com.google.ar.core" android:value="required" />
29
30         <activity
31             android:name=".MainActivity"
32             android:exported="true">
33             <intent-filter>
34                 <action android:name="android.intent.action.MAIN" />
35
36                 <category android:name="android.intent.category.LAUNCHER" />
37             </intent-filter>
38         </activity>
39     </application>
```


Paso 5: Corrección de errores

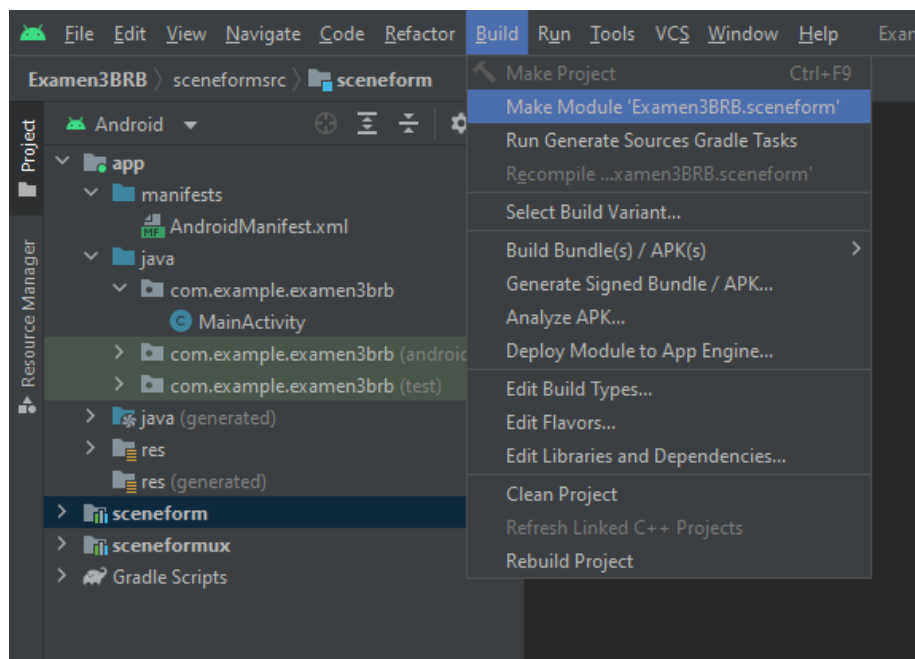
Ahora las carpetas descargadas sceneformsrc y sceneformux debemos migrarlas para el proyecto al nuevo Androidx. Podemos encontrar una manera de migrar todo el proyecto a Androidx o podemos cambiar las importaciones manualmente una por una.

Vamos a Build > Rebuild project

Encontraremos muchos errores. Entonces, en la sección 'Construir', hacemos doble clic en el error de importación del paquete. Se abrirá un código con la sección de error resaltada.

Necesitamos cambiar solo tres tipos de ruta de importación, que se indican a continuación, siempre que veamos el primero lo cambiamos por el segundo:


- ❖ ***android.support.annotation. -> androidx.annotation.***
- ❖ ***androidx.core.app -> androidx.fragment.app.***
- ❖ ***android.support.v7.widget. -> androidx.appcompat.widget.***



Hacemos esto hasta que no haya más errores. (Imagen con errores)

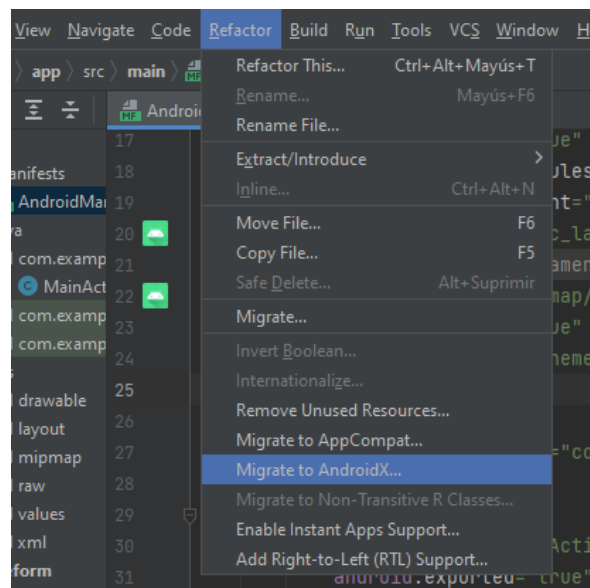
```
Camera.java x
1 package com.google.ar.sceneform;
2
3 import android.support.annotation.Nullable;
4 import android.support.annotation.VisibleForTesting;
5 import android.view.MotionEvent;
6 import com.google.ar.core.Pose;
7
```

(Imagen sin errores)

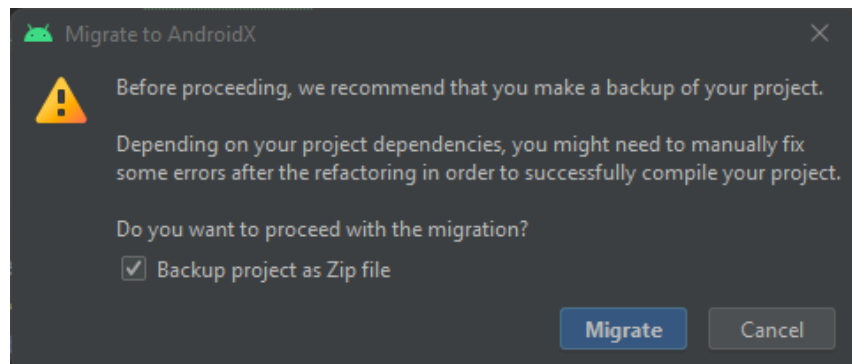


```
1 package com.google.ar.sceneform;  
2  
3 import androidx.annotation.Nullable;  
4 import androidx.annotation.VisibleForTesting;  
5 import android.view.MotionEvent;  
6 import com.google.ar.core.Pose;
```

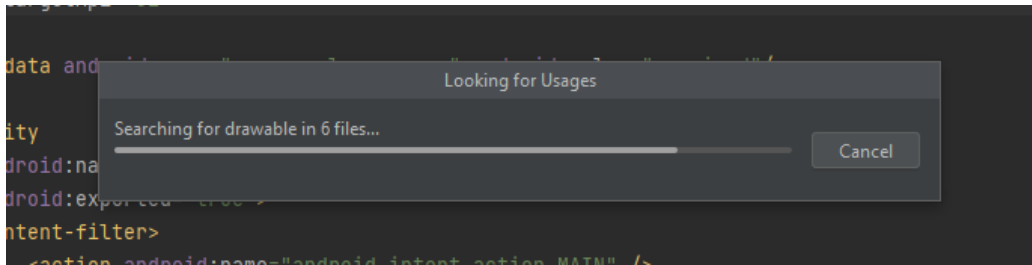
Podemos cambiar todas las rutas de importación de manera rápida haciendo clic en Refactor > Migrate to AndroidX



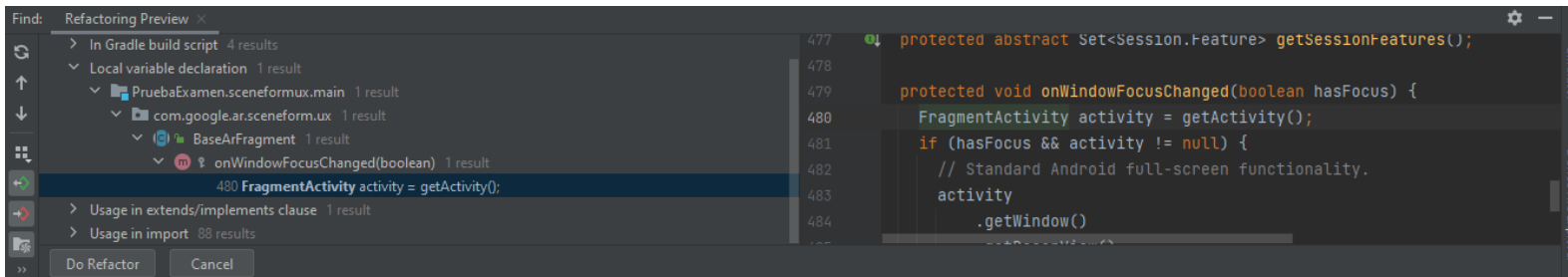
Nos mostrara la siguiente ventana en donde seleccionamos la casilla si queremos guardar un respaldo de nuestro proyecto antes de realizar los cambios, hacemos clic en Migrate.



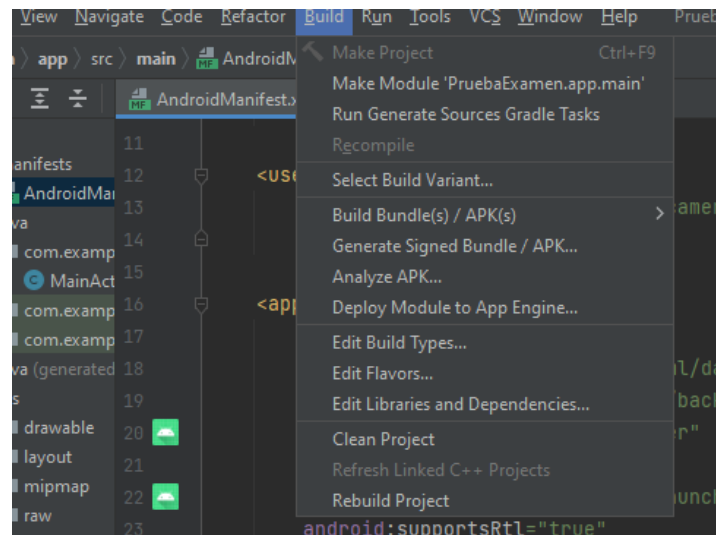
Luego esperamos



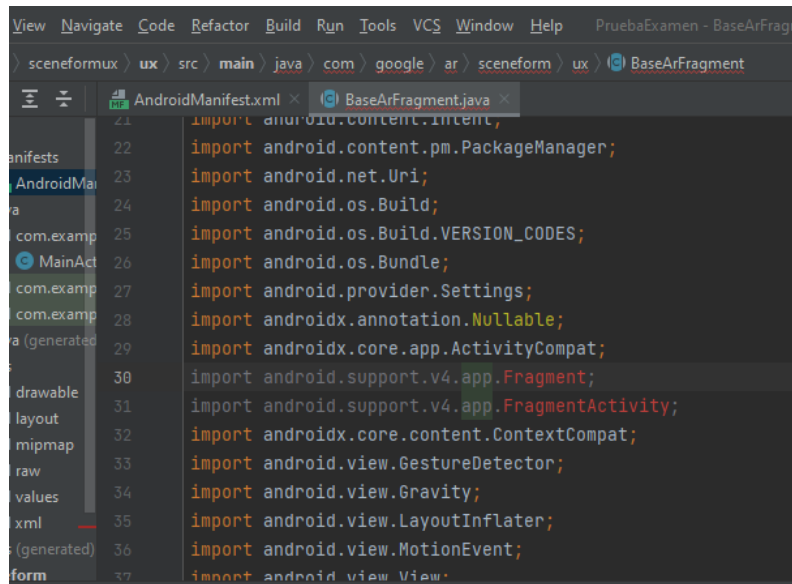
Nos aparecerá lo siguiente, hacemos clic en 'Do Refactor' y esperamos



Una vez concluido lo anterior hacemos clic en Build > Rebuild Project

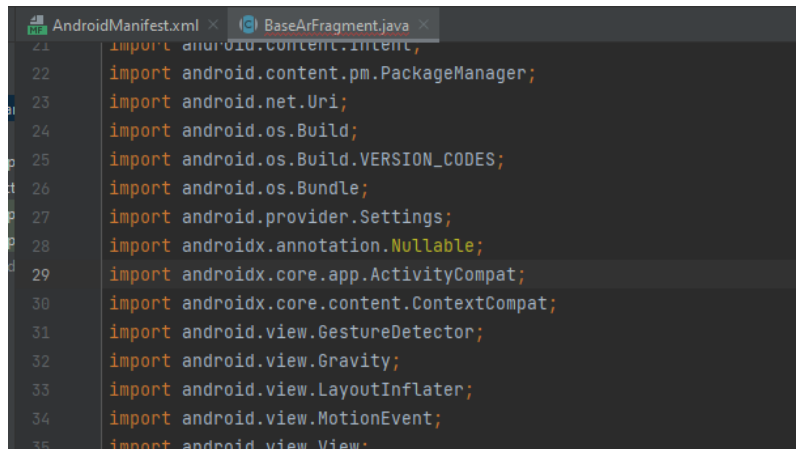


Nos aparecerán algunos errores como los siguientes en BaseArFragment.java



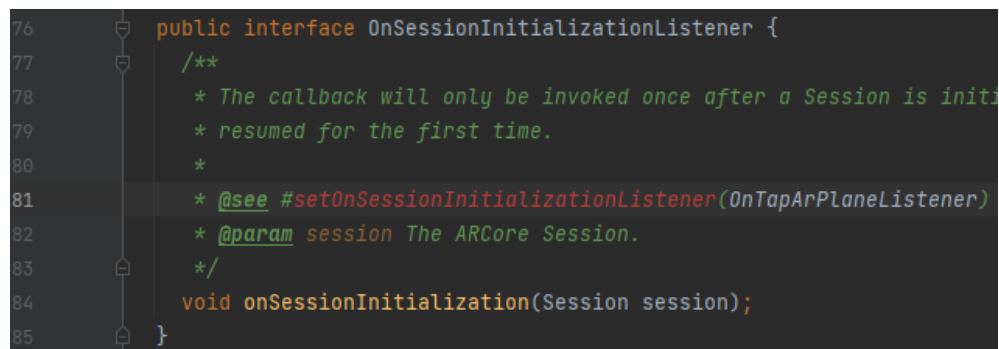
```
21 import android.content.Intent;
22 import android.content.pm.PackageManager;
23 import android.net.Uri;
24 import android.os.Build;
25 import android.os.Build.VERSION_CODES;
26 import android.os.Bundle;
27 import android.provider.Settings;
28 import androidx.annotation.Nullable;
29 import androidx.core.app.ActivityCompat;
30 import androidx.core.app.Fragment;
31 import androidx.core.app.FragmentActivity;
32 import androidx.core.content.ContextCompat;
33 import android.view.GestureDetector;
34 import android.view.Gravity;
35 import android.view.LayoutInflater;
36 import android.view.MotionEvent;
37 import android.view.View;
```

Eliminamos las dos importaciones que contienen errores y la siguiente línea dentro de BaseArFragment.java



```
21 import android.content.Intent;
22 import android.content.pm.PackageManager;
23 import android.net.Uri;
24 import android.os.Build;
25 import android.os.Build.VERSION_CODES;
26 import android.os.Bundle;
27 import android.provider.Settings;
28 import androidx.annotation.Nullable;
29 import androidx.core.app.ActivityCompat;
30 import androidx.core.content.ContextCompat;
31 import android.view.GestureDetector;
32 import android.view.Gravity;
33 import android.view.LayoutInflater;
34 import android.view.MotionEvent;
35 import android.view.View;
```

Eliminamos la línea 81 antes mencionada



```
76 public interface OnSessionInitializationListener {
77     /**
78      * The callback will only be invoked once after a Session is initiated
79      * resumed for the first time.
80      *
81      * @see #setOnSessionInitializationListener(OnTapArPlaneListener)
82      * @param session The ARCore Session.
83      */
84     void onSessionInitialization(Session session);
85 }
```

Después de eliminarla no marcara errores esta parte del código.

```
76 public interface OnSessionInitializationListener {
77     /**
78      * The callback will only be invoked once after a Ses
79      * resumed for the first time.
80      *
81      * @param session The ARCore Session.
82      */
83     void onSessionInitialization(Session session);
84 }
```

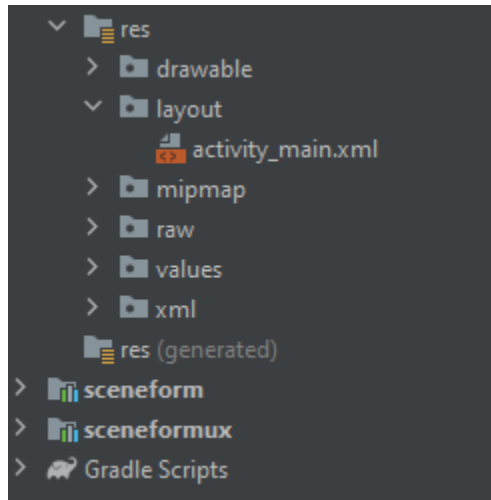
Luego agregamos las dos siguientes librerías en BaseArFragment.java (Línea de código 32 y 33)

```
27 import android.provider.Settings;
28 import androidx.annotation.Nullable;
29 import androidx.core.app.ActivityCompat;
30 import androidx.core.content.ContextCompat;
31
32 import androidx.fragment.app.Fragment;
33 import androidx.fragment.app.FragmentActivity;
34
35 import android.view.GestureDetector;
36 import android.view.Gravity;
37 import android.view.LayoutInflater;
```

Finalmente, los errores se habrán solucionado

Paso 6: Trabajar con el archivo activity_main.xml

Vamos al archivo res> layout> activity_main.xml.



Aquí está el código por default del archivo XML:

```
activity_main.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity">
9
10     <TextView
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="Hello World!"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintEnd_toEndOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19 </androidx.constraintlayout.widget.ConstraintLayout>
```

Cambiamos el código por default de activity_main.xml por el siguiente código

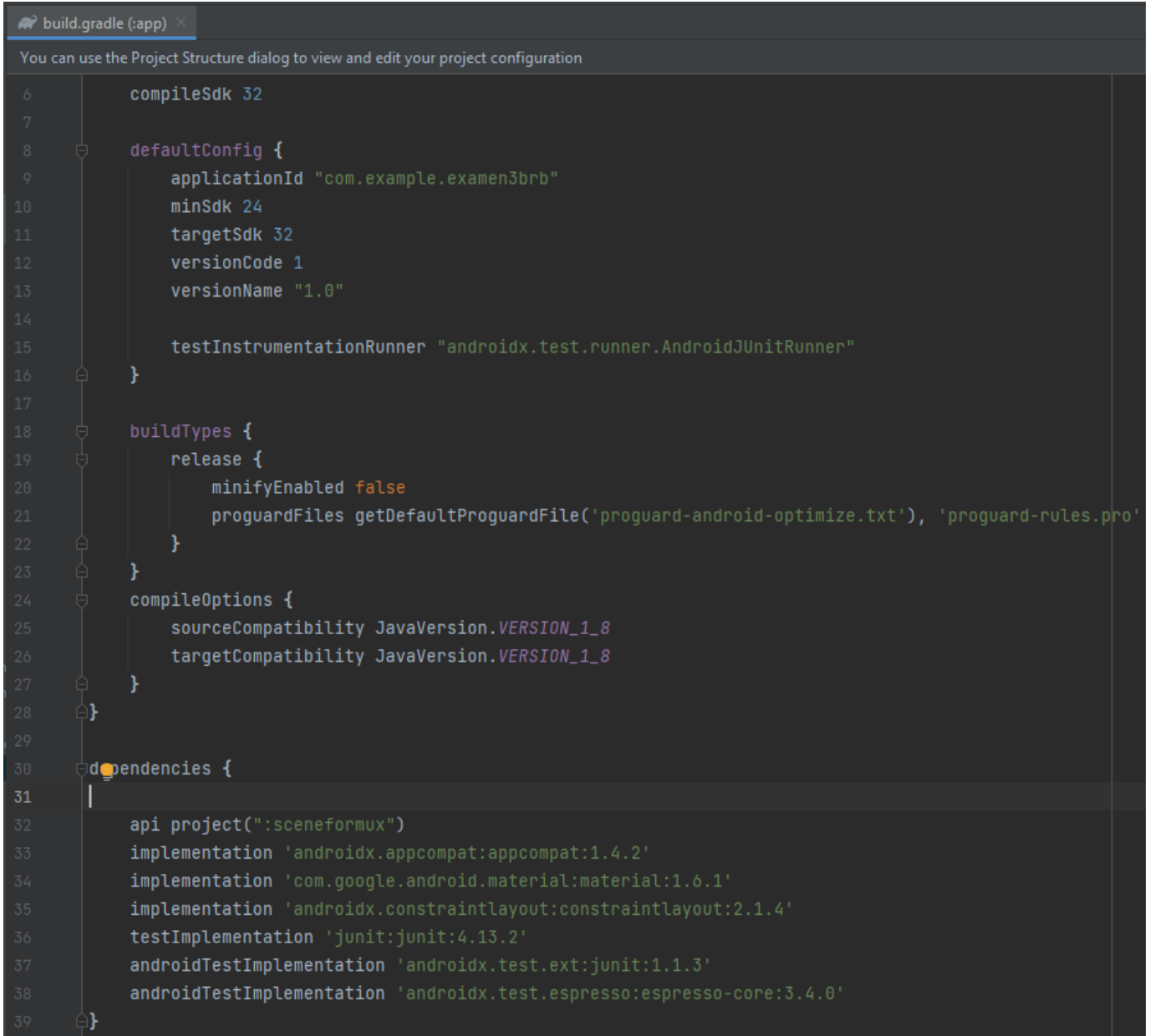
```
activity_main.xml x MainActivity.java x build.gradle (:app) x
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10    <!--This is the fragment that will be used as AR camera-->
11    <fragment
12        android:id="@+id/arCameraArea"
13        android:name="com.google.ar.sceneform.ux.ArFragment"
14        android:layout_width="match_parent"
15        android:layout_height="match_parent"
16        app:layout_constraintBottom_toBottomOf="parent"
17        app:layout_constraintEnd_toEndOf="parent"
18        app:layout_constraintStart_toStartOf="parent"
19        app:layout_constraintTop_toTopOf="parent" />
20
21 </androidx.constraintlayout.widget.ConstraintLayout>
```

En gradle.properties agregamos el siguiente código

```
build.gradle (:app) x gradle.properties x
1 # Project-wide Gradle settings.
2 # IDE (e.g. Android Studio) users:
3 # Gradle settings configured through the IDE *will override*
4 # any settings specified in this file.
5 # For more details on how to configure your build environment visit
6 # http://www.gradle.org/docs/current/userguide/build\_environment.html
7 # Specifies the JVM arguments used for the daemon process.
8 # The setting is particularly useful for tweaking memory settings.
9 org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
10 # When configured, Gradle will run in incubating parallel mode.
11 # This option should only be used with decoupled projects. More details, visit
12 # http://www.gradle.org/docs/current/userguide/multi\_project\_builds.html#sec:decoupled\_projects
13 # org.gradle.parallel=true
14 # AndroidX package structure to make it clearer which packages are bundled with the
15 # Android operating system, and which are packaged with your app's APK
16 # https://developer.android.com/topic/libraries/support-library/androidx-rn
17 android.useAndroidX=true
18 # Enables namespacing of each library's R class so that its R class includes only the
19 # resources declared in the library itself and none from the library's dependencies,
20 # thereby reducing the size of the R class for that library
21 android.nonTransitiveRClass=true
22 android.enableJetifier=true
```

Y en build.gradle (:app) agregamos en dependencias lo siguiente

```
api project(':sceneformux')
```

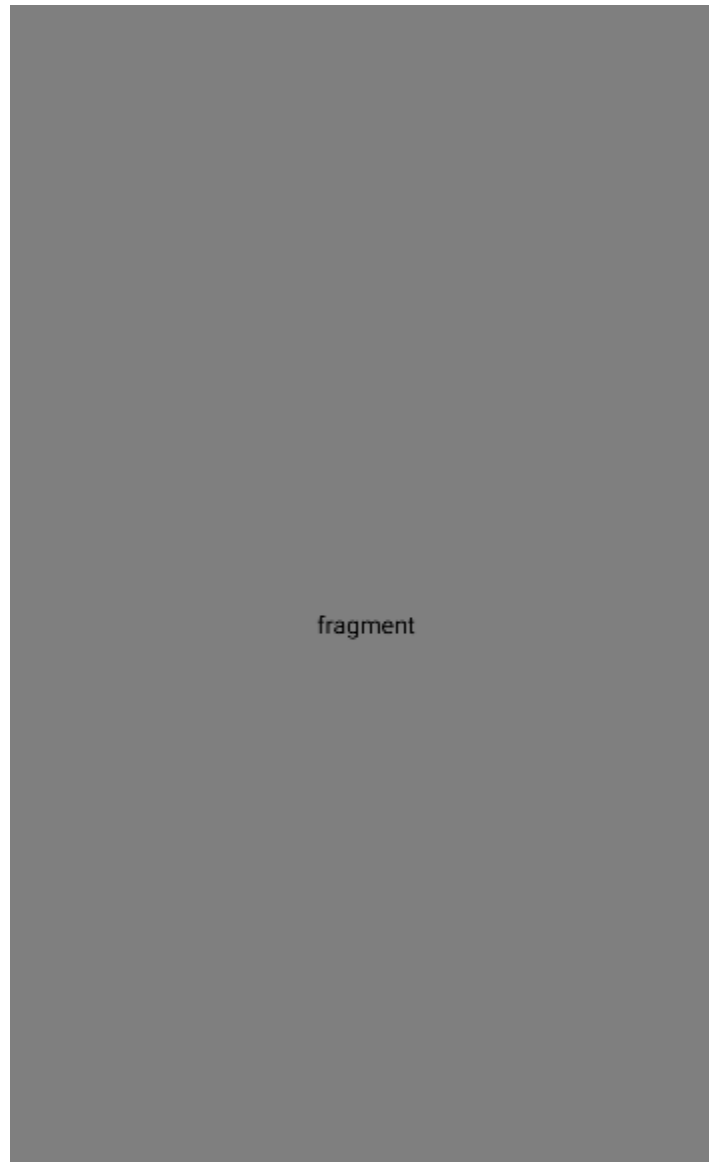


```
build.gradle (:app) x
You can use the Project Structure dialog to view and edit your project configuration

6      compileSdk 32
7
8      defaultConfig {
9          applicationId "com.example.examen3brb"
10         minSdk 24
11         targetSdk 32
12         versionCode 1
13         versionName "1.0"
14
15         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
16     }
17
18     buildTypes {
19         release {
20             minifyEnabled false
21             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
22         }
23     }
24     compileOptions {
25         sourceCompatibility JavaVersion.VERSION_1_8
26         targetCompatibility JavaVersion.VERSION_1_8
27     }
28 }
29
30 dependencies {
31     |
32     api project(":sceneformux")
33     implementation 'androidx.appcompat:appcompat:1.4.2'
34     implementation 'com.google.android.material:material:1.6.1'
35     implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
36     testImplementation 'junit:junit:4.13.2'
37     androidTestImplementation 'androidx.test.ext:junit:1.1.3'
38     androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
39 }
```


ArFragment contiene muchas características en sí mismo, como pedirle que descargue ARCore si aún no está instalado en su teléfono o como si solicita el permiso de la cámara si aún no lo ha otorgado. Entonces, ArFragment es lo mejor para usar aquí.

Después de escribir este código, la interfaz de usuario de la aplicación se verá así:



Paso 7: Trabajar con el archivo MainActivity.java

Vamos a `java> com.example.examen3brb` (el suyo puede diferir) `> MainActivity.java`

En la clase MainActivity, primero, tenemos que hacer un objeto de ArFragment.

```
private ArFragment arCam; //object of ArFragment Class
```

Ahora, creemos una función de verificación de hardware fuera de la función `onCreate()`. Esta función verificará si el hardware de su teléfono cumple con todos los requisitos sistémicos para ejecutar esta aplicación AR. Va a comprobar:

Es la versión API de Android `> = 24` en ejecución, lo que significa Android Nougat 7.0

Es la versión de OpenGL `> = 3.0`

Tener estos es obligatorio para ejecutar aplicaciones AR usando ARCore y Sceneform. Aquí está el código de esa función:

```
public static boolean checkSystemSupport(Activity activity) {  
    //checking whether the API version of the running Android >= 24 that means Android Nougat 7.0  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {  
        String openGLVersion = ((ActivityManager) Objects.requireNonNull(activity.getSystemService(Context.ACTIVITY_SERVICE))).getDeviceConfigurationInfo().getGLVersion();  
        //checking whether the OpenGL version >= 3.0  
        if (Double.parseDouble(openGLVersion) >= 3.0) {  
            return true;  
        } else {  
            Toast.makeText(activity, "App needs OpenGL Version 3.0 or later", Toast.LENGTH_SHORT).show();  
            activity.finish();  
            return false;  
        }  
    } else {  
        Toast.makeText(activity, "App does not support required Build Version", Toast.LENGTH_SHORT).show();  
        activity.finish();  
        return false;  
    }  
}
```

Dentro de la función `onCreate()` primero, necesitamos verificar el hardware del teléfono. Si devuelve verdadero, se ejecutará el resto de la función.

Ahora el ArFragment está vinculado con su ID respectivo utilizado en `activity_main.xml`.

```
arCam = (ArFragment) getSupportFragmentManager().findFragmentById(R.id.arCameraArea);  
//(ArFragment) getSupportFragmentManager().findFragmentById(R.id.arCameraArea);
```

Se llama a un `onTapListener`, para mostrar el modelo 3d, cuando tocamos en la pantalla.

Dentro de `setOnTapArPlaneListener`, se crea un objeto `Anchor`. `Anchor` realmente ayuda a traer objetos virtuales a la pantalla y hacer que permanezcan en la misma posición y orientación en el espacio.

Ahora se usa una clase `ModelRenderable` con un montón de funciones. Esta clase se utiliza para renderizar el modelo 3D descargado o creado adjuntándolo a un `AnchorNode`.

La función `setSource()` ayuda a obtener la fuente del modelo 3D.

La función `setIsFilamentGltf()` comprueba si es un archivo glb.

La función `build()` renderiza el modelo.

Se llama a una función dentro de la función `thenAccept()` para recibir el modelo adjuntando un `AnchorNode` con `ModelRenderable`.

La función `exception()` lanza una excepción si algo sale mal mientras se construye el modelo.

```
arCam.setOnTapArPlaneListener((hitResult, plane, motionEvent) -> {
    clickNo++;
    //the 3d model comes to the scene only when clickNo is one that means once
    if (clickNo == 1) {
        Anchor anchor = hitResult.createAnchor();
        ModelRenderable.builder() ModelRenderable.Builder
            .setSource(context: this, R.raw.auto)
            .setIsFilamentGltf(true)
            .build() CompletableFuture<ModelRenderable>
            .thenAccept(modelRenderable -> addModel(anchor, modelRenderable)) CompletableFuture<Void>
            .exceptionally(throwable -> {
                AlertDialog.Builder builder = new AlertDialog.Builder(context: this);
                builder.setMessage("Something is not right" + throwable.getMessage()).show();
                return null;
            });
    }
});
```

Ahora, veamos qué hay en la función `addModel()`. Se necesitan dos parámetros, el primero es `Anchor` y el otro es `ModelRenderable`. Se crea un objeto `AnchorNode`. Es el nodo raíz de la escena. `AnchorNode` se posiciona automáticamente en el mundo, según el `Anchor`. `TransformableNode` ayuda al usuario a interactuar con el modelo 3D, como cambiar de posición, redimensionar, rotar, etc.

```
private void addModel(Anchor anchor, ModelRenderable modelRenderable) {
    AnchorNode anchorNode = new AnchorNode(anchor);
    // Creating a AnchorNode with a specific anchor
    anchorNode.setParent(arCam.getArSceneView().getScene());
    //attaching the anchorNode with the ArFragment
    TransformableNode model = new TransformableNode(arCam.getTransformationSystem());
    model.setParent(anchorNode);
    //attaching the anchorNode with the TransformableNode
    model.setRenderable(modelRenderable);
    //attaching the 3d model with the TransformableNode that is already attached with the node
    model.select();
}
```

Finalmente agregamos las librerías necesarias en el MainActivity.java

```
import androidx.appcompat.app.AppCompatActivity;
import android.app.Activity;
import android.app.ActivityManager;
import android.app.AlertDialog;
import android.content.Context;
import android.os.Build;
import android.os.Bundle;
import android.widget.Toast;
import com.google.ar.core.Anchor;
import com.google.ar.sceneform.AnchorNode;
import com.google.ar.sceneform.rendering.ModelRenderable;
import com.google.ar.sceneform.ux.ArFragment;
import com.google.ar.sceneform.ux.TransformableNode;
import java.util.Objects;
```

5. Código

A continuación, se muestra el código para el proyecto.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.examen3brb">
    <!--This helps to permit the user to access Camera-->
    <uses-permission android:name="android.permission.CAMERA" />
    <!--This helps to check a specific feature in the phone's hardware,
        here it is OpenGL ES version 3-->
    <uses-feature
        android:glEsVersion="0x00030000"
        android:required="true" />
    <!--Here it is checking for AR feature in phone camera-->
    <uses-feature
        android:name="android.hardware.camera.ar"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
```

```

        android:theme="@style/Theme.Examen3BRB"
        tools:targetApi="31">

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

        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

            </intent-filter>
        </activity>
    </application>
</manifest>

```

MainActivity.java

```

import androidx.appcompat.app.AppCompatActivity;
import android.app.Activity;
import android.app.ActivityManager;
import android.app.AlertDialog;
import android.content.Context;
import android.os.Build;
import android.os.Bundle;
import android.widget.Toast;
import com.google.ar.core.Anchor;
import com.google.ar.sceneform.AnchorNode;
import com.google.ar.sceneform.rendering.ModelRenderable;
import com.google.ar.sceneform.ux.ArFragment;
import com.google.ar.sceneform.ux.TransformableNode;
import java.util.Objects;

public class MainActivity extends AppCompatActivity {

    private ArFragment arCam; //object of ArFragment Class

    private int clickNo = 0; //helps to render the 3d model only once
    when we tap the screen

    public static boolean checkSystemSupport(Activity activity) {
        //checking whether the API version of the running Android >= 24
        that means Android Nougat 7.0
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            String openGlVersion = ((ActivityManager)
Objects.requireNonNull(activity.getSystemService(Context.ACTIVITY_SERVICE
))).getDeviceConfigurationInfo().getGlEsVersion();
            //checking whether the OpenGL version >= 3.0
            if (Double.parseDouble(openGlVersion) >= 3.0) {
                return true;
            } else {
                Toast.makeText(activity, "App needs OpenGL Version 3.0 or
later", Toast.LENGTH_SHORT).show();
            }
        }
    }
}

```

```

        activity.finish();
        return false;
    }
} else {
    Toast.makeText(activity, "App does not support required Build
Version", Toast.LENGTH_SHORT).show();
    activity.finish();
    return false;
}
}

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

    if (checkSystemSupport(this)) {
        arCam = (ArFragment)
getSupportFragmentManager().findFragmentById(R.id.arCameraArea);
        // (ArFragment)
getSupportFragmentManager().findFragmentById(R.id.arCameraArea);
        arCam.setOnTapArPlaneListener((hitResult, plane, motionEvent)
-> {
            clickNo++;
            //the 3d model comes to the scene only when clickNo is
one that means once
            if (clickNo == 1) {
                Anchor anchor = hitResult.createAnchor();
                ModelRendererable.builder()
                    .setSource(this, R.raw.auto)
                    .setIsFilamentGltf(true)
                    .build()
                    .thenAccept(modelRenderable ->
addModel(anchor, modelRenderable))
                    .exceptionally(throwable -> {
                        AlertDialog.Builder builder = new
AlertDialog.Builder(this);
                        builder.setMessage("Something is not
right" + throwable.getMessage()).show();
                        return null;
                    });
            }
        });
    } else {
        return;
    }
}

private void addModel(Anchor anchor, ModelRendererable modelRenderable)
{
    AnchorNode anchorNode = new AnchorNode(anchor);
    // Creating a AnchorNode with a specific anchor
    anchorNode.setParent(arCam.getArSceneView().getScene());
    //attaching the anchorNode with the ArFragment
    TransformableNode model = new
TransformableNode(arCam.getTransformationSystem());
    model.setParent(anchorNode);
}

```

```

        //attaching the anchorNode with the TransformableNode
        model.setRenderable(modelRenderable);
        //attaching the 3d model with the TransformableNode that is
        already attached with the node
        model.select();
    }
}

```

activity_main.xml

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <!--This is the fragment that will be used as AR camera-->
    <fragment
        android:id="@+id/arCameraArea"
        android:name="com.google.ar.sceneform.ux.ArFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

```

Gradle:app

```

plugins {
    id 'com.android.application'
}

android {
    compileSdk 32

    defaultConfig {
        applicationId "com.example.examen3brb"
        minSdk 24
        targetSdk 32
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner
        "androidx.test.runner.AndroidJUnitRunner"
    }
}

```

```

        buildTypes {
            release {
                minifyEnabled false
                proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
            }
        }
        compileOptions {
            sourceCompatibility JavaVersion.VERSION_1_8
            targetCompatibility JavaVersion.VERSION_1_8
        }
    }
}

dependencies {

    api project(":sceneformux")
    implementation 'androidx.appcompat:appcompat:1.4.2'
    implementation 'com.google.android.material:material:1.6.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}

```

Settings.gradle

```

pluginManagement {
    repositories {
        gradlePluginPortal()
        google()
        mavenCentral()
    }
}

dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
    }
}

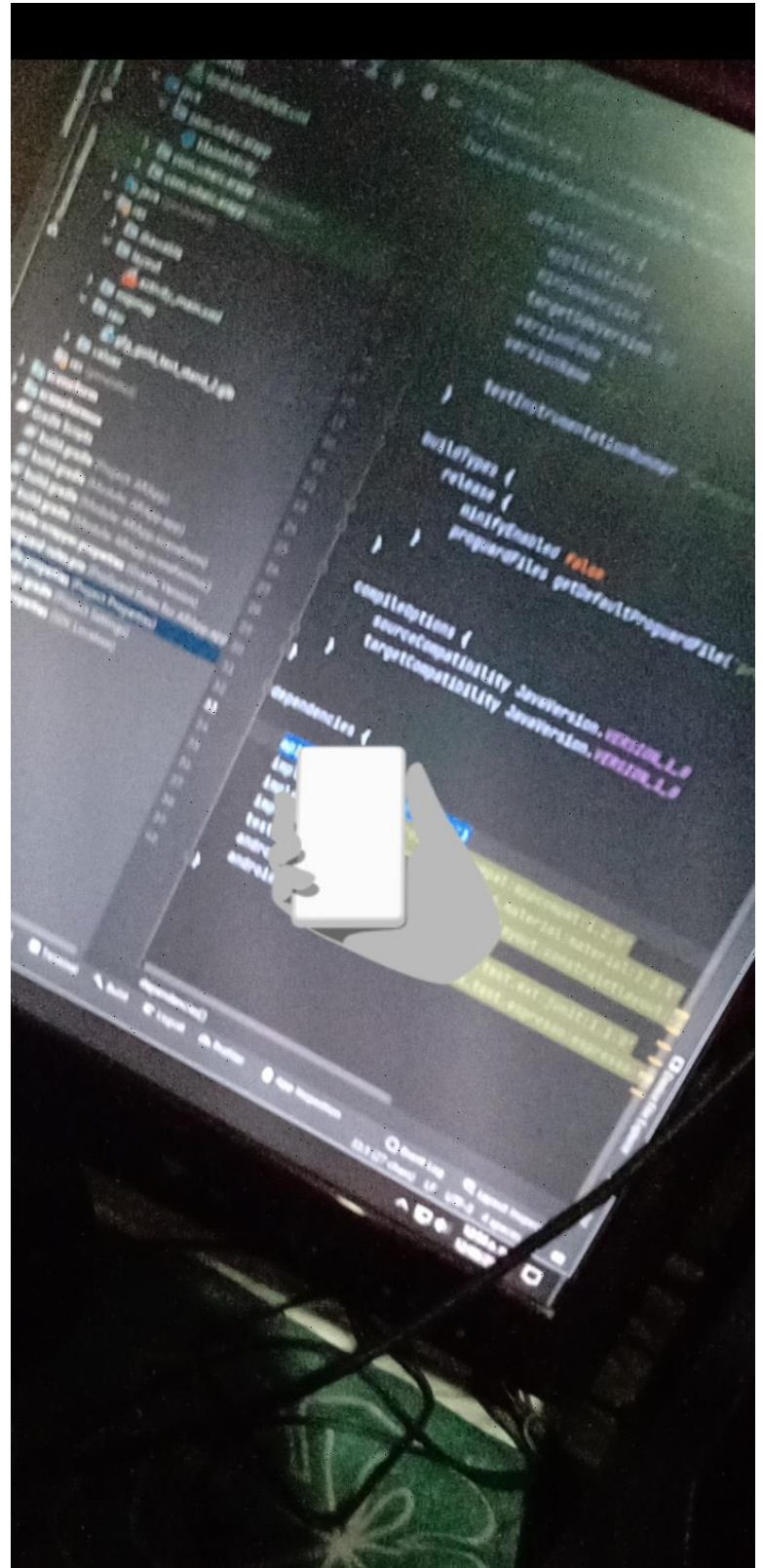
rootProject.name = "Examen3BRB"
include ':app'

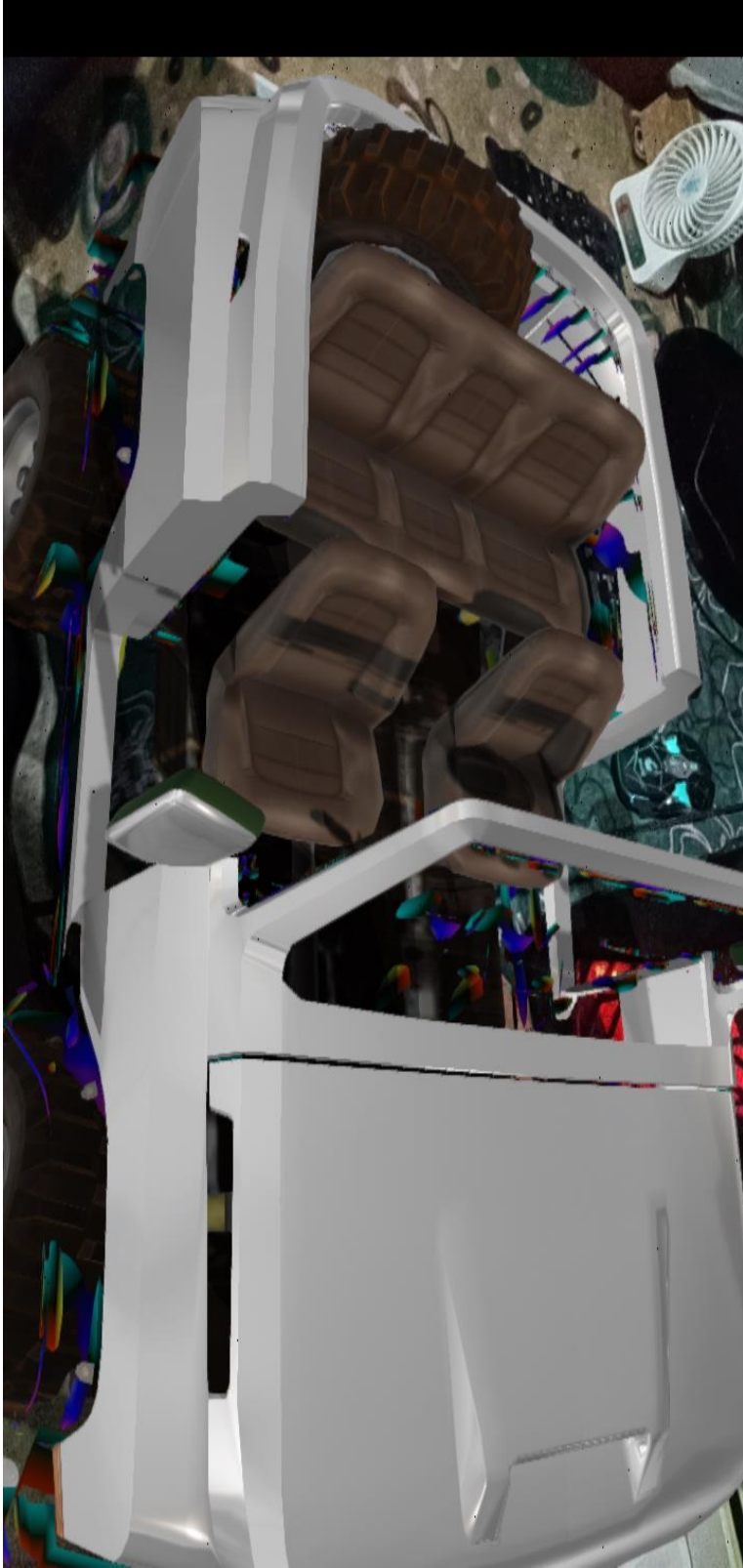
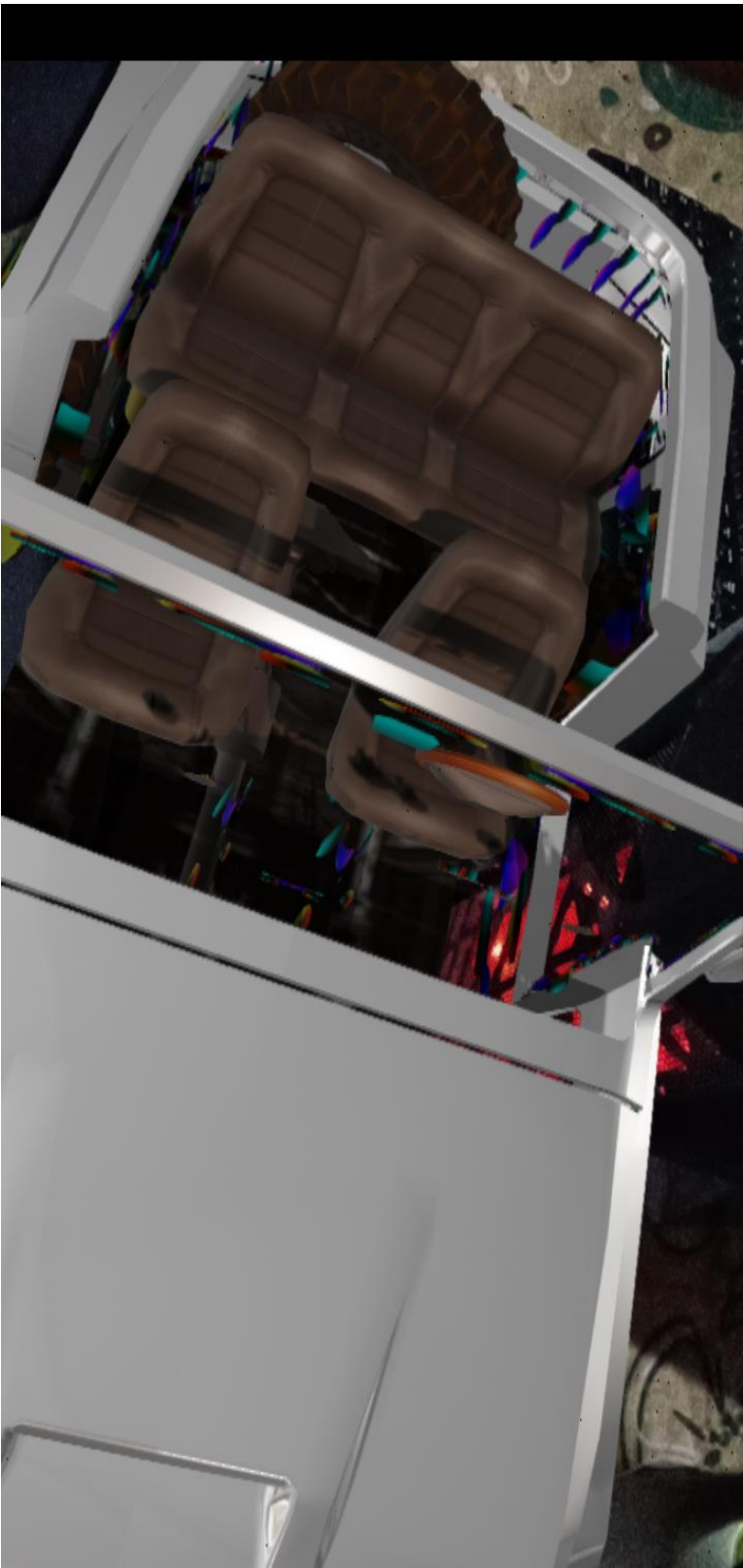
//this will add sceneformsrc folder into your project
include ':sceneform'
project(':sceneform').projectDir=new File('sceneformsrc/sceneform')

//this will add sceneformux folder into your project
include ':sceneformux'
project(':sceneformux').projectDir=new File('sceneformux/ux')

```


A screenshot of an Android phone screen. At the top, the status bar shows the time 12:54 and icons for alarm, Wi-Fi, cellular signal, and battery. Below the status bar, the app name 'Examen3BRB' is displayed in a dark header. The main area of the screen is black. A grey dialog box is centered on the screen. Inside the dialog, there is a camera icon at the top, followed by the text '¿Permitir que Examen3BRB tome fotos y grabe videos?'. Below the text are two buttons: 'Permitir' and 'Denegar'. At the very bottom of the screen, the Android navigation bar is visible with icons for back, home, and recent apps.





7. Conclusiones

En este trabajo se ha tratado la realidad aumentada, en este punto aprendemos lo que es la realidad aumentada, a definirla, cómo de importante es y va a llegar a ser, entendiendo su potencial, y que a través de la experimentación va a lograr hacerse un hueco en nuestro día a día. Además, vemos el entono de desarrollo integrado con el que vamos a trabajar, Android Studio, y la plataforma de realidad aumentada que vamos a utilizar en nuestra aplicación, ARCore. Con esta información podemos probar ya una aplicación demostración que ofrece Android, aprendemos los conceptos más básicos del lenguaje de programación a usar, Java, y explicamos el código de la demostración para así, tener la base para poder hacer las modificaciones necesarias en este y crear nuestras propias aplicaciones. En lo general y particular se ha logrado el objetivo del trabajo. Además, he aprendido no solo la creación de una aplicación en Android para la visualización de modelos 3D en realidad aumentada, si no también mejorado, entendido y profundizado en este mundo y esta tecnología de futuro.

8. Bibliografía

Arias, J. A. (2020, enero). Programación de una aplicación Android para la visualización de modelos 3D en realidad aumentada - PDF Descargar libre. PDF. Recuperado 12 de junio de 2022, de <https://docplayer.es/200782187-Programacion-de-una-aplicacion-android-para-la-visualizacion-de-modelos-3d-en-realidad-aumentada.html>

Getting started with Sceneform | Sceneform (1.15.0) |. (s. f.). Google Developers. Recuperado 12 de junio de 2022, de <https://developers.google.com/sceneform/develop/getting-started>

Lima, A. (s. f.). ¿Cómo crear una aplicación de Android de realidad aumentada simple? – Acervo Lima. Acervo Lima. Recuperado 12 de junio de 2022, de <https://es.acervolima.com/como-crear-una-aplicacion-de-android-de-realidad-aumentada-simple/>

Oldpro, M. (s. f.). azure-docs.es-es/tutorial-new-android-app.md at master · MicrosoftDocs/azure-docs.es-es. GitHub. Recuperado 12 de junio de 2022, de <https://github.com/MicrosoftDocs/azure-docs.es-es/blob/master/articles/spatial-anchors/tutorials/tutorial-new-android-app.md>