ARCore by Google

Jacob Avalos, Lydia De Castilhos, & Yulissa Martinez
CIS 357 01

ARCore App Tutorial: Integrating Firebase for AR Experiences

1. Overview

This ARCore app demonstrates how to integrate Firebase for managing AR experiences. The app allows users to:

- Place and interact with 3D models in an AR environment.
- Capture and upload screenshots of the AR scene.
- Host and resolve cloud anchors using Firebase.

Key Features:

- **AR Screen**: Place and interact with 3D models.
- **Firebase Integration**: Authentication, Firestore, and Storage.
- **Cloud Anchors**: Host and resolve anchors for shared AR experiences.

2. Getting Started

Prerequisites:

- Install Android Studio.
- Set up Firebase Project.

Dependencies: Add these dependencies to your `build.gradle` file:

```
dependencies {
    implementation("org.jetbrains.kotlin:kotlin-stdlib:1.9.0")
    implementation("androidx.core:core-ktx:1.10.1")
    implementation("androidx.appcompat:appcompat:1.6.1")
    implementation("com.google.android.material:material:1.9.0")
    implementation("androidx.activity:activity-compose:1.6.1")
    implementation("androidx.constraintlayout:constraintlayout-compose:1.1.0")
    implementation("androidx.compose.ui:ui:1.5.1")
    implementation("androidx.compose.material3:material3:1.3.1")
    implementation("androidx.navigation:navigation-compose:2.6.0")
    implementation(platform("com.google.firebase:firebase-bom:33.7.0"))
    implementation("com.google.firebase:firebase-auth:21.3.0")
    implementation("com.google.firebase:firebase-firestore:24.4.0")
    implementation("com.google.firebase:firebase-storage:20.2.1")
    implementation(libs.ar.core)
    testImplementation("junit:junit:4.13.2")
    androidTestImplementation("androidx.test.ext:junit:1.1.5")
    androidTestImplementation("androidx.test.espresso:espresso-core:3.5.1")
    androidTestImplementation("androidx.compose.ui:ui-test-junit4:1.5.1")
    debugImplementation("androidx.compose.ui:ui-tooling:1.5.1")
    implementation("io.github.sceneview:arsceneview:0.10.0")
    implementation("com.google.android.gms:play-services-auth:16.0.1")
    implementation("com.squareup.picasso:picasso:2.71828")
    implementation("io.coil-kt:coil-compose:2.0.0")
}
```

Firebase Configuration:

- Add `google-services.json` to the `app/` directory.
- Enable Authentication and Firestore in the Firebase Console.

3. Firebase Setup

Step 1: Initialize Firebase

```
// Initialize Firebase
FirebaseApp.initializeApp(this)
firestore = FirebaseFirestore.getInstance()
```

Step 2: Log Out User on Start

```
// Log out user on start
FirebaseAuth.getInstance().signOut()
```

Step 3: Retrieve ARCore API Key

```
// Retrieve ARCore API Key from AndroidManifest.xml
val applicationInfo = packageManager.getApplicationInfo(packageName,
PackageManager.GET_META_DATA)
val arcoreApiKey = applicationInfo.metaData.getString("ARCORE_API_KEY")
```

4. Step-by-Step Coding Instructions

4.1 MainActivity

File: `MainActivity.kt`

Activity Initialization

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
```

- **Explanation**: Initializes the activity and sets the content view to `activity_main.xml`.

Log Out User on Start

```
// Log out user on start
 FirebaseAuth.getInstance().signOut()
```

- **Explanation**: Logs out the user when the activity starts to ensure a fresh authentication state.

Initialize Firebase

```
// Initialize Firebase
 FirebaseApp.initializeApp(this)
 firestore = FirebaseFirestore.getInstance()
```

- **Explanation**: Initializes Firebase and Firestore instances for database operations.

Initialize FirebaseAuth

```
// Initialize FirebaseAuth
 val auth = FirebaseAuth.getInstance()
```

- **Explanation**: Initializes Firebase Authentication to manage user authentication.

Retrieve ARCore API Key

```
// Retrieve ARCore API Key from AndroidManifest.xml
 val applicationInfo = packageManager.getApplicationInfo(packageName,
PackageManager.GET_META_DATA)
 val arcoreApiKey = applicationInfo.metaData.getString("ARCORE_API_KEY")
```

- **Explanation**: Retrieves the ARCore API key from the app's manifest file for AR functionalities.

Set Content with Compose

```
setContent {
    val navController = rememberNavController()
    MyApp {
        AppNavigator(navController, auth)
        }
    }
}
```

- **Explanation**: Sets the content view using Jetpack Compose, initializing the navigation controller and calling the `MyApp` composable function.

`MyApp` Composable Function

```
@Composable
fun MyApp(content: @Composable () -> Unit) {
    content()
}
```

- **Explanation**: A simple composable function that takes another composable as content and displays it.

`AppNavigator` Composable Function

```
@Composable
fun AppNavigator(navController: NavHostController, auth: FirebaseAuth) {
    // Check if the user is logged out
    val currentUser = auth.currentUser
    val startDestination = if (currentUser == null) "login" else "home"
```

- **Explanation**: Determines the start destination based on the user's authentication state.

Navigation Host

```
  NavHost(navController = navController, startDestination = startDestination) {
      composable("login") { LoginScreen(navController, auth) }
      composable("new_account") { NewAccountScreen(navController, auth) }
      composable("home") { HomePageScreen(navController) }
      composable("catalog") { CatalogScreen(navController) }
      composable("account") { AccountScreen(navController) }
```

- **Explanation**: Sets up the navigation host with different composable destinations for login, account creation, home, catalog, and account screens.

Navigate to ARScreen Activity

```
     // Navigate to ARScreen Activity with optional anchorId parameter
```

```
        composable("ar_screen?anchorId={anchorId}") { backStackEntry ->
            val anchorId = backStackEntry.arguments?.getString("anchorId")

            // Navigate to ARScreen activity and pass the anchorId
            val context = LocalContext.current
            LaunchedEffect(Unit) {
                val intent = Intent(context, ARScreen::class.java).apply {
                    putExtra("anchorId", anchorId)
                }
                context.startActivity(intent)
            }
        }
    }
}
```

- **Explanation**: Defines a composable for navigating to the `ARScreen` activity, passing an optional `anchorId` parameter.

4.2 ARScreen

File: `ARScreen.kt`

Activity Initialization ```kotlin override fun onCreate(savedInstanceState: Bundle?) { super.onCreate(savedInstanceState) setContentView(R.layout.activity_main) // Get the anchor ID from intent extras (if any) receivedAnchorId = intent.getStringExtra("anchorId") // Initialize the scene view sceneView = findViewById(R.id.sceneView).apply { this.lightEstimationMode = Config.LightEstimationMode.DISABLED } // Initialize modelNode early as a placeholder modelNode = ArModelNode(sceneView.engine, PlacementMode.INSTANT) sceneView.addChild(modelNode) // Ensure the AR session is initialized sceneView.onArSessionCreated = { arSession -> session = arSession session.configure(Config(session).apply { cloudAnchorMode = Config.CloudAnchorMode.ENABLED }) // If an anchor ID was provided, resolve it receivedAnchorId?.let { resolveCloudAnchorAsync(it) } } // Initialize media player for video node mediaPlayer = MediaPlayer.create(this, R.raw.ad) // Initialize UI elements placeButton = findViewById(R.id.place) resolveButton = findViewById(R.id.resolveButton) captureButton = findViewById(R.id.captureButton) hostAnchorButton = findViewById(R.id.hostAnchorButton)

```
// Place model without hosting the cloud anchor
placeButton.setOnClickListener {
    placeModel()
}

// Resolve Cloud Anchor when button clicked
```

```
resolveButton.setOnClickListener {
    val anchorId = receivedAnchorId
    anchorId?.let { resolveCloudAnchorAsync(it) }
}

// Add the new capture button functionality
captureButton.setOnClickListener {
    captureScreenshotAndUpload()
}

// Host Cloud Anchor when button clicked
hostAnchorButton.setOnClickListener {
    if (isModelPlaced && currentAnchor != null) {
        hostCloudAnchorAsync(currentAnchor!!)
        Toast.makeText(this, "Model is placed!", Toast.LENGTH_SHORT).show()
    } else {
        Toast.makeText(this, "Model is not placed yet!", Toast.LENGTH_SHORT).show()
    }
}

// Initialize video and model nodes
videoNode = VideoNode(
    sceneView.engine,
    scaleToUnits = 0.7f,
    centerOrigin
```

Further Discussion/Conclusions

This tutorial demonstrated how to build a collaborative AR application using ARCore, Cloud Anchors, and Firebase. While we used native ARCore features, developers might consider third-party libraries like Vuforia or Unity AR Foundation for cross-platform development.

Find the complete source code for this tutorial on our GitHub repository.

See Also

- ARCore Documentation
- Cloud Anchors Documentation