Unity3D Workflow - Integration & Environment

This guide provides a step-by-step workflow for developing and integrating Unity3D into the mobile application Xamera. It covers installation, environment setup, Unity-Android communication, and exporting the Unity project as an AAR (Android Archive) library for integration into the Xamera app

Table of Contents

[**Phase 0: Installation and Creation 2**](#_heading=h.lo2f4hw8qbgm)

[Step 0.1: Download and Install Unity 2](#_heading=h.ulknx05ihtls)

[Step 0.2: Access the Shared Project 2](#_heading=h.regua45wo8o9)

[Step 0.3: Optional - Create a new project 2](#_heading=h.vtnjyzeds5bt)

[**Phase 1: Designing your Environment 2**](#_heading=h.5gp6uaxprzj)

[Step 1.1: Create and Configure 3D Objects 2](#_heading=h.6f8jvei6s5d2)

[Step 1.2: Set Up Cameras 2](#_heading=h.41w2f08hxp9v)

[Step 1.3: Implement VR (Optional) 2](#_heading=h.wo7zb9fic2mt)

[Step 1.4: Add Maps and Textures 2](#_heading=h.s8id23zbo2ok)

[Step 1.5: Use the Unity Asset Store 2](#_heading=h.1xsbjwz5e3bh)

[**Phase 2: Unity-Android Communication Setup 3**](#_heading=h.9mpp7lf1gb8y)

[Step 2.1: Set Up Camera Cross-Communication 3](#_heading=h.1me42c7uyvjx)

[Step 2.2: Test Communication 3](#_heading=h.dp2qwxalrff3)

[**Phase 3: Exporting the Unity Project as a Gradle Build 3**](#_heading=h.lkvf5mfn10c5)

[Step 3.1: Configure Build Settings 3](#_heading=h.21kesmmgleni)

[Step 3.2: Export as a Gradle Project 3](#_heading=h.8i85catb6ndz)

[**Phase 4: Building the AAR File in Android Studio 3**](#_heading=h.rvytns19lfi1)

[Step 4.1: Open the Exported Gradle Project 3](#_heading=h.rjhvzjrclo3o)

[Step 4.2: Make Necessary Adjustments 3](#_heading=h.p41vqhaau8w0)

[Step 4.3: Build the AAR File 3](#_heading=h.fahy93o70qto)

[**Phase 5: Integrating Unity with Xamera 4**](#_heading=h.805u1d8500kl)

[Step 5.1: Import the Unity AAR File 4](#_heading=h.ield26emwxbr)

[Step 5.2: Implement Unity Viewspace in Xamera 4](#_heading=h.im4forjcqodp)

[**Phase 6: Coding in Xamera with Unity 4**](#_heading=h.2prnbxgpv222)

# Phase 0: Installation and Creation

## Step 0.1: Download and Install Unity

1. Download Unity Hub from the official website: <https://unity.com/download>
2. Install Unity Hub (Version 3.11.0).
3. Using Unity Hub, install the Unity Editor (Version 6000.0.34f1).
   * Ensure the Android Build Support module is installed.

## Step 0.2: Access the Shared Project

1. Navigate to the shared Adaptive-HCI-Unity3D project.
   1. If you can’t find the project contact me (Soham)
2. Download the project files to your local machine.
3. Open the project in Unity Hub.

## Step 0.3: Optional - Create a new project

1. New Project
2. Templates > Universal 3D

# Phase 1: Designing your Environment

## Step 1.1: Create and Configure 3D Objects

* Add 3D objects to your scene (e.g., models, primitives).
* Adjust their position, rotation, and scale.

## Step 1.2: Set Up Cameras

* Add and configure cameras for your scene.
* Adjust camera properties (e.g., field of view, clipping planes).

## Step 1.3: Implement VR (Optional)

* Enable VR support in Unity:
* Go to Edit > Project Settings > XR Plug-in Management.
* Install the required XR plugins (e.g., Oculus, OpenXR).
* Configure VR-specific settings.

## Step 1.4: Add Maps and Textures

* Import maps and textures into the project.
* Apply textures to 3D objects for realistic rendering.

## Step 1.5: Use the Unity Asset Store

* Browse the Unity Asset Store for pre-built assets, scripts, and plugins.
* Import assets into your project as needed.

# Phase 2: Unity-Android Communication Setup

## Step 2.1: Set Up Camera Cross-Communication

1. Implement a Live Listener to enable real-time data exchange between Unity and the Android app.
2. Use Unity's AndroidJavaClass and AndroidJavaObject to communicate with Android native code.

## Step 2.2: Test Communication

1. Send test data from Unity to Android and vice versa.
2. Ensure the communication is stable and reliable.

# Phase 3: Exporting the Unity Project as a Gradle Build

## Step 3.1: Configure Build Settings

1. Go to File > Build Settings.
2. Select Android as the target platform.
3. Configure player settings:
4. Set the Minimum API Level to match Xamera's requirements.
5. Enable ARM64 architecture for better performance.

## Step 3.2: Export as a Gradle Project

1. In the Build Settings window, check the Export Project option.
2. Click Build and Run.
3. Choose a directory to export the Gradle project.
4. Unity will generate a folder containing the Gradle project files.

# Phase 4: Building the AAR File in Android Studio

## Step 4.1: Open the Exported Gradle Project

1. Open Android Studio.
2. Select Open an Existing Project and navigate to the exported Gradle project folder.
3. Open the project.

## Step 4.2: Make Necessary Adjustments

1. Review the project structure and ensure all dependencies are correctly configured.
2. Make any required code tweaks or adjustments.

## Step 4.3: Build the AAR File

1. In Android Studio, go to Build > Make Project.
2. The AAR file will be generated in the unityLibrary/build/outputs/aar/ directory.
3. The file will typically be named unityLibrary-release.aar.

# Phase 5: Integrating Unity with Xamera

## Step 5.1: Import the Unity AAR File

1. Locate the AAR file generated in Phase 4.3.
2. Copy the AAR file to the libs directory in the Xamera Android project.
3. Add the AAR file as a dependency in the build.gradle file:

| dependencies {  implementation files('libs/unityLibrary-release.aar')  } |
| --- |

## Step 5.2: Implement Unity Viewspace in Xamera

1. Create a Unity view in Xamera's layout XML:

| <com.unity3d.player.UnityPlayerView  android:id="@+id/unityView"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent" /> |
| --- |

1. Initialize the Unity view in the activity:

| UnityPlayer unityPlayer = new UnityPlayer(this);  View unityView = unityPlayer.getView();  setContentView(unityView); |
| --- |

# Phase 6: Coding in Xamera with Unity