Integrating Unity modules or games into native Android applications is a common practice for developers looking to leverage Unity's powerful game engine within a broader Android app. The process involves creating a Unity library that can be imported into your Android project. Here's an overview of the steps involved:

### 1. Export Unity Project as an Android Library

First, you need to export your Unity project as an Android Library (AAR) or as an Android project:

1. **Open your Unity project.**
2. **Go to File > Build Settings.**
3. **Select Android as the platform.**
4. **Click on Player Settings and navigate to the Publishing Settings.**
5. **Check the "Export Project" option to export it as an Android Studio project.** For an AAR library, you might need a custom exporter or third-party tool, as Unity doesn't directly support exporting as AAR.

### 2. Integrate the Unity Library into Your Android Project

After exporting your Unity project as an Android Studio project or AAR, you need to integrate it into your native Android app:

#### **If You Have an Android Studio Project:**

* **Import the Unity project as a module:**
  1. Open your Android Studio project.
  2. Go to File > New > Import Module.
  3. Select the exported Unity project folder.
  4. Follow the prompts to integrate the Unity module into your Android project.

#### **If You Have an AAR File:**

* **Add the AAR to your project:**
  1. Copy the AAR file into the **libs** folder of your Android app module.
  2. Open your app's **build.gradle** file.
  3. Add the following to your dependencies block:

gradleCopy code

implementation fileTree(dir: 'libs', include: ['\*.aar'])

### 3. Load the Unity Activity from Your Native Android App

After integrating the Unity module or AAR into your Android project, you can start the Unity Activity from your native code. For example:

javaCopy code

Intent intent = new Intent(this, UnityPlayerActivity.class); startActivity(intent);

Ensure that **UnityPlayerActivity** is declared in your project's **AndroidManifest.xml** file, which should have been automatically handled if you imported the Unity project correctly.

### Considerations

* **Communication between Unity and Android:** For advanced integration, such as calling Android methods from Unity or vice versa, you'll need to use Unity's **AndroidJavaClass** or **AndroidJavaObject** for invoking Android Java code from Unity C# scripts.
* **Unity as a Library (UaaL):** Unity offers the Unity as a Library feature, which allows you to integrate Unity more seamlessly into native mobile apps. This feature might require additional setup and configuration based on the Unity and Android versions.

### Troubleshooting

* **Compatibility Issues:** Ensure your Unity and Android project settings are compatible, particularly the Android API levels and dependencies.
* **Performance:** Integrating Unity can significantly increase the size of your app and impact performance. Optimize your Unity project for mobile deployment.

This overview provides a basic pathway for integrating a Unity module into a native Android application. Depending on your specific requirements, you might need to delve into more detailed documentation and community forums for advanced integration scenarios.

User

**You**

give me some sample links github links there its same thing is uses, give me some example links or code

ChatGPT

**ChatGPT**

Integrating Unity with native Android apps is a specialized topic, and finding exact GitHub repositories that match your needs can be challenging without a direct search. However, I can guide you on how to find relevant examples and mention some keywords and repository characteristics you should look for.

### Keywords for Searching

When searching on GitHub or other code-sharing platforms, use specific keywords that match your integration goals. Here are some suggested search terms:

* **Unity Android integration**
* **Unity Android Library AAR**
* **UnityPlayerActivity Android**
* **Unity as a Library (UaaL)**
* **Unity Android plugin**

### What to Look for in Repositories

1. **Recent Activity:** Look for repositories that have been updated recently to ensure compatibility with the latest versions of Unity and Android.
2. **Documentation:** Good repositories usually have clear instructions on how to use their code or integrate the library into your project.
3. **Issue Tracker:** Check the issues tab to see common problems users face and how active the maintainers are in addressing those issues.
4. **Stars and Forks:** Repositories with more stars and forks are typically more reliable and widely used, indicating community trust.

### Sample Repositories

Here are a couple of GitHub search links that could lead you to relevant repositories. Please note that the availability and quality of repositories can vary:

* GitHub search for Unity Android Integration
* GitHub search for Unity Android Library AAR

Due to the dynamic nature of GitHub and the varying specifics of each project, I can't provide direct links to specific repositories. However, these searches are a good starting point.

### Example Code Snippets

While specific GitHub repositories may provide detailed examples, here's a basic idea of how code snippets might look for integrating Unity with Android:

**Android Activity to Start Unity Activity**

javaCopy code

// Intent to start UnityPlayerActivity Intent intent = new Intent(this, UnityPlayerActivity.class); startActivity(intent);

**Unity C# Script Calling Android Function**

csharpCopy code

using UnityEngine; public class CallAndroidMethod : MonoBehaviour { void Start() { AndroidJavaClass unityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer"); AndroidJavaObject currentActivity = unityPlayer.GetStatic<AndroidJavaObject>("currentActivity"); currentActivity.Call("yourAndroidMethod", "parameter1"); } }

Remember, these snippets are basic and meant to give you an idea of how the integration could be implemented. The exact implementation details will depend on your project's requirements.

### Final Note

Since the integration process can be complex and specific to your project's needs, it's also beneficial to consult the official Unity documentation on integration and the Android developer documentation for the latest practices and APIs.

using UnityEngine;

public class CallAndroidMethod : MonoBehaviour

{

void Start()

{

AndroidJavaClass unityPlayer = new AndroidJavaClass("com.unity3d.player.UnityPlayer");

AndroidJavaObject currentActivity = unityPlayer.GetStatic<AndroidJavaObject>("currentActivity");

currentActivity.Call("yourAndroidMethod", "parameter1");

}

}

// Intent to start UnityPlayerActivity

Intent intent = new Intent(this, UnityPlayerActivity.class);

startActivity(intent);

**Keywords for Searching**

When searching on GitHub or other code-sharing platforms, use specific keywords that match your integration goals. Here are some suggested search terms:

* **Unity Android integration**
* **Unity Android Library AAR**
* **UnityPlayerActivity Android**
* **Unity as a Library (UaaL)**
* **Unity Android plugin**

**What to Look for in Repositories**

1. **Recent Activity:** Look for repositories that have been updated recently to ensure compatibility with the latest versions of Unity and Android.
2. **Documentation:** Good repositories usually have clear instructions on how to use their code or integrate the library into your project.
3. **Issue Tracker:** Check the issues tab to see common problems users face and how active the maintainers are in addressing those issues.
4. **Stars and Forks:** Repositories with more stars and forks are typically more reliable and widely used, indicating community trust.

**Sample Repositories**

Here are a couple of GitHub search links that could lead you to relevant repositories. Please note that the availability and quality of repositories can vary:

* GitHub search for Unity Android Integration
* GitHub search for Unity Android Library AAR