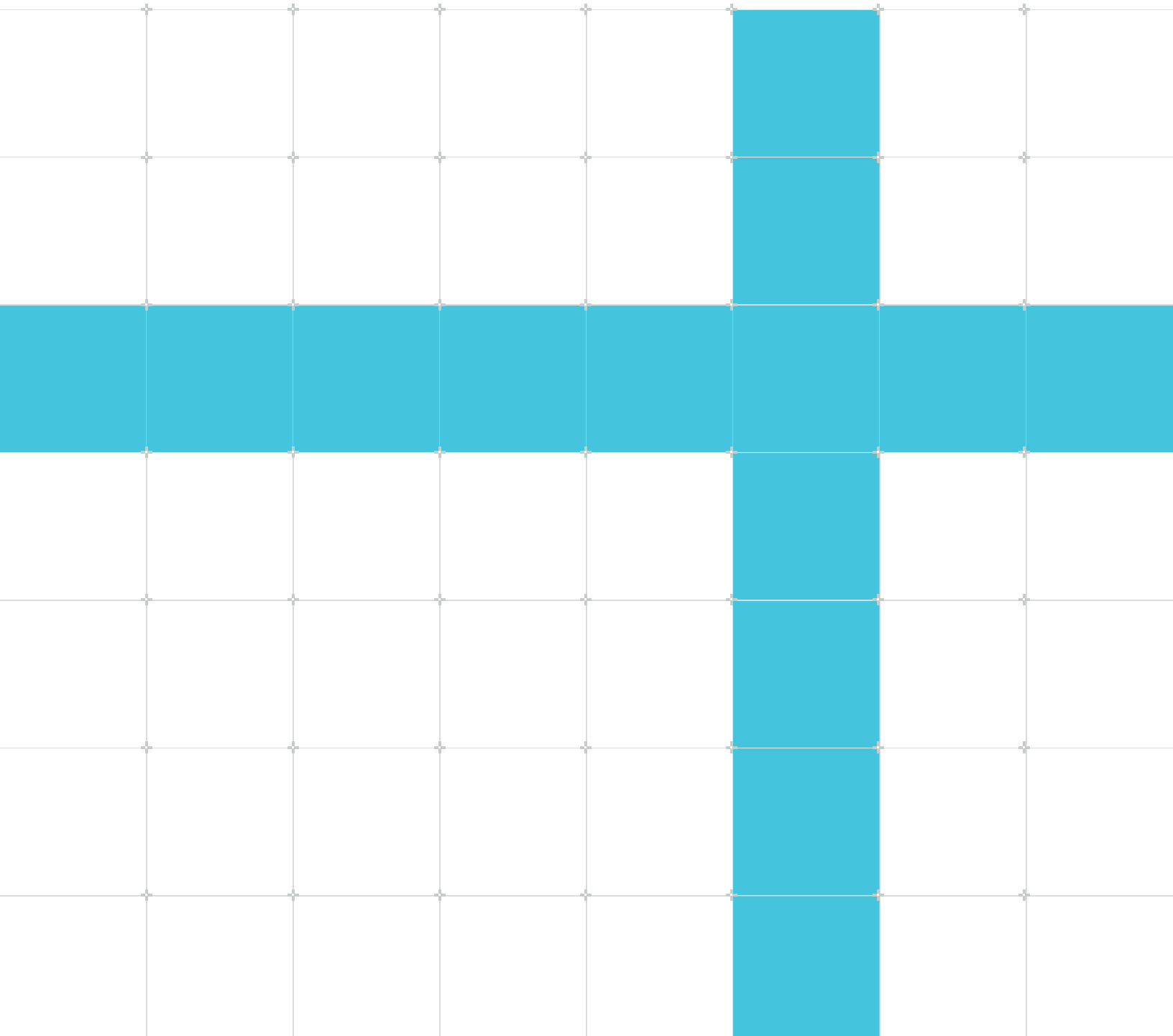
Integrate Arm Mobile Studio with Unreal Engine

Version 1.0

Issue 01
102717_0100_01_en



Integrate Arm Mobile Studio with Unreal Engine

Release information

Document history

Issue	Date	Confidentiality	Change
0100-01	13 October 2021	Non-Confidential	First release

Proprietary Notice

or indirectly, in violation of such export laws. Use of the word "partner" in reference to Arm's customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349|version 21.0)

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Feedback

Arm® welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

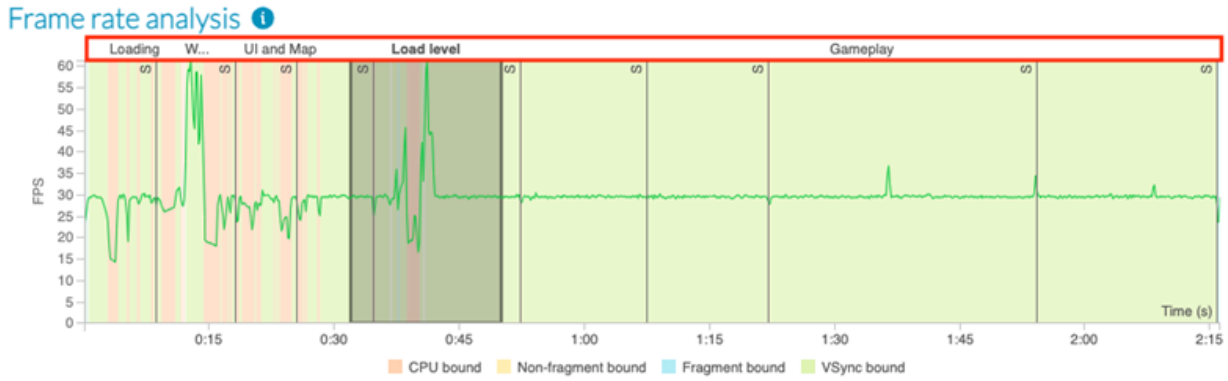
We believe that this document contains no offensive language. To report offensive language in this document, email terms@arm.com.

Contents

1. Overview

You can add instrumentation to your Unreal Engine game, so that you can analyze different regions separately in Arm Mobile Studio. For example, when you generate a Performance Advisor report, the region names can be seen in the Frame rate analysis chart, and the report contains dedicated analysis sections for each region.

Figure 1-1: Region names in Performance Advisor



This tutorial describes how to:

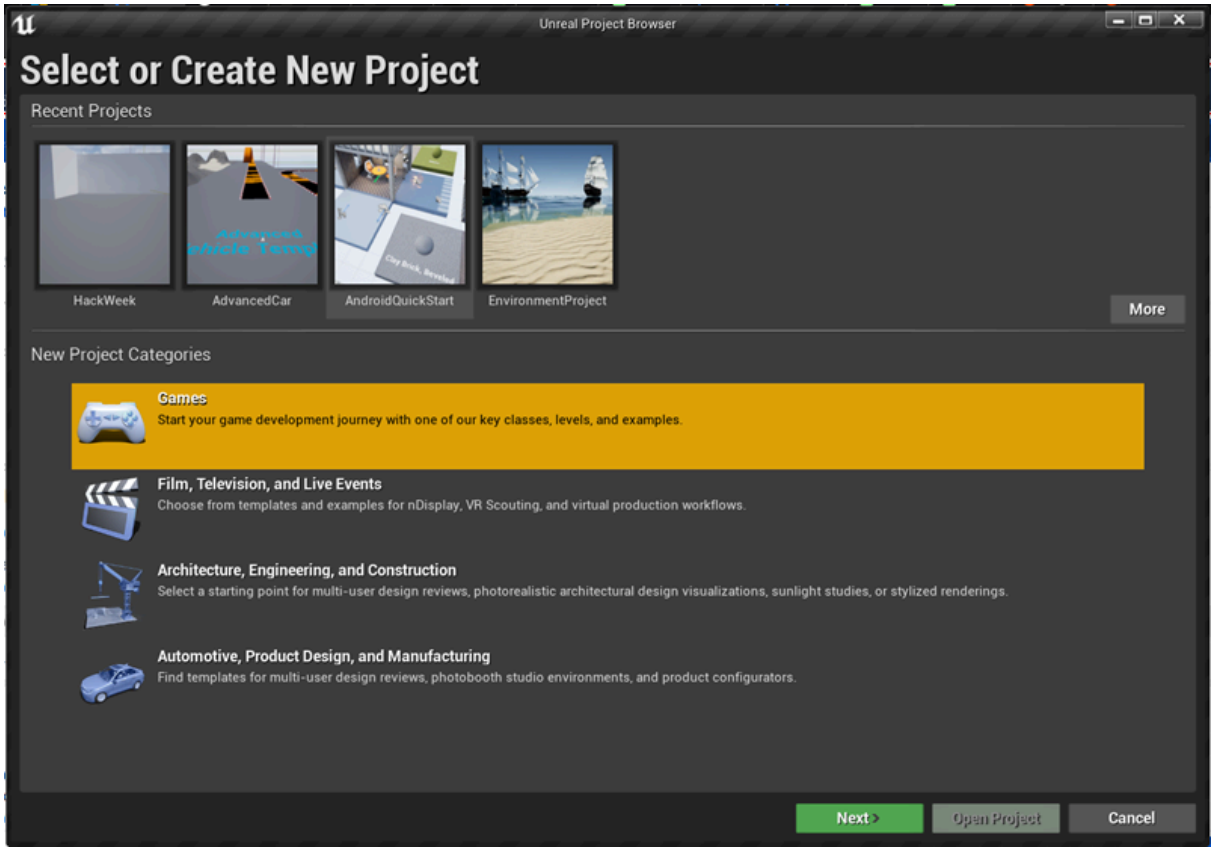
- Create a basic project within Unreal Engine
- Integrate Streamline annotations with the code in your project
- Structure the Streamline annotations to create region markers in a Performance Advisor report
- Build your project for Android devices.

2. Create an Unreal Engine project

Follow these steps to create a new Unreal Engine project for mobile:

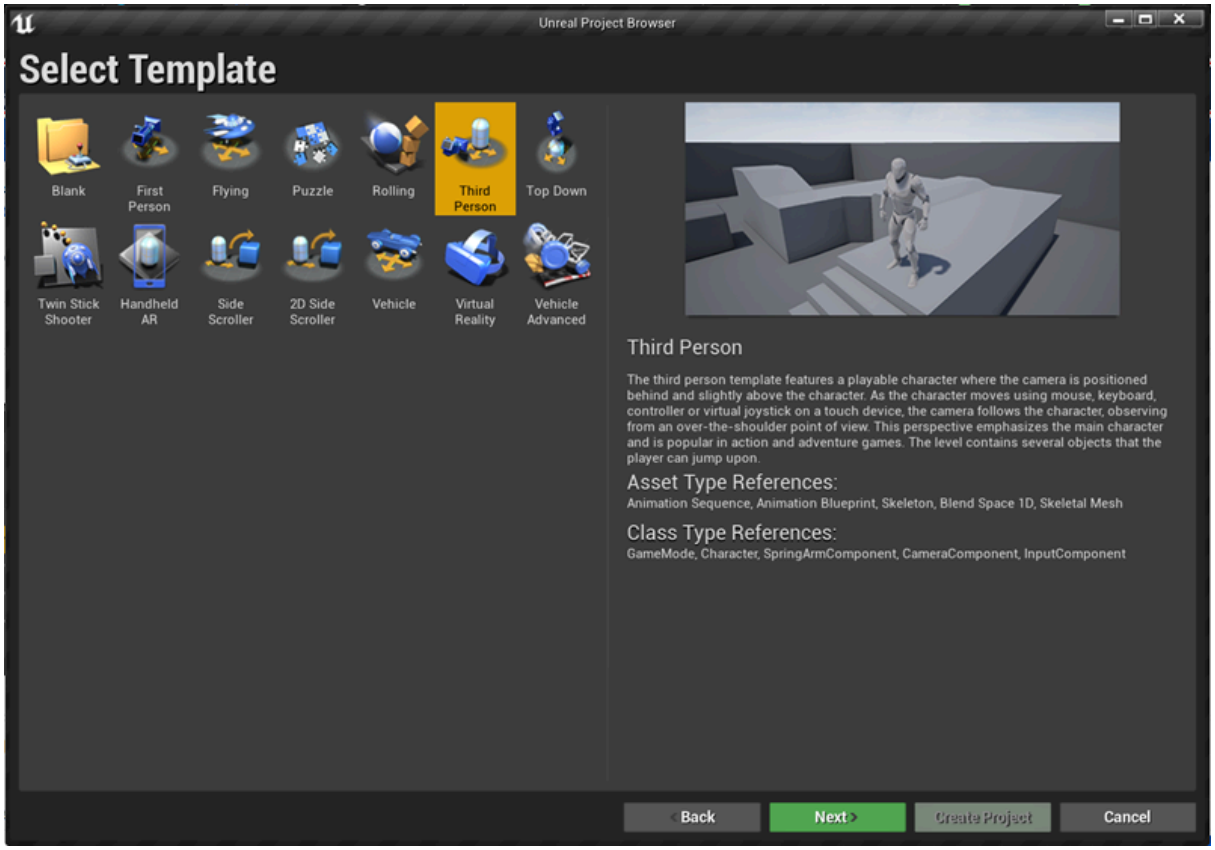
1. Launch Unreal Engine, select Games and click Next.

Figure 2-1: Create a new games project in Unreal Engine



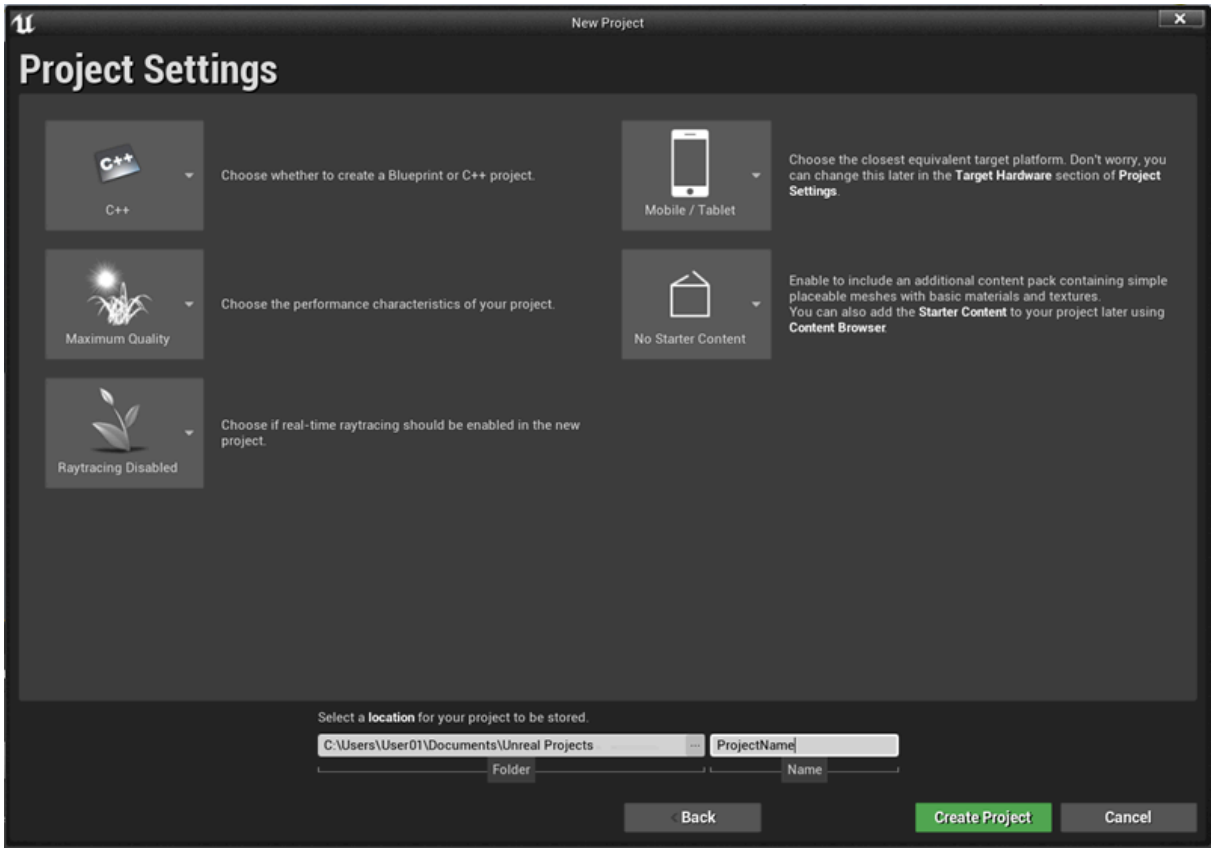
2. Unreal Engine provides some templates that contain already-configured controls for mobile devices. Or you can create your own by selecting Blank. Choose an option and click Next.

Figure 2-2: Selecting a template in Unreal Engine



3. Choose your required graphics settings for the project. For the purposes of this tutorial, here are some recommended settings:
    a. Create a C++ project
    b. Choose Mobile/Tablet as the target platform
    c. Choose Maximum Quality
    d. Choose No Starter Content to significantly reduce the size of the APK
    e. Set Raytracing Disabled as this is often too demanding for mobile GPUs
    f. Give your project a name and location.

Figure 2-3: Project settings in Unreal Engine



4. Click Create Project.

3. Add the Streamline annotation files to your project

To add annotations, you need to include 2 files to your Unreal Engine project. These files are located in your Arm Mobile Studio installation directory:

- `<install_location>/streamline/gator/annotate/streamline_annotate.c`
- `<install_location>/streamline/gator/annotate/streamline_annotate.h`

There are 2 ways to add these files to your project:

- Copy the files into your project, under `<unreal_project/Source/<project_name>` and then add the following line into any source file where you want to create annotations:

```
#include "streamline_annotate.h"
```

- Alternatively, build a `libstreamline_annotate` Makefile:

```
cd <install_location>/streamline/gator/annotate
make
```

Then, copy the `libstreamline_annotate` library into your project, under `<unreal_project/Source/<project_name>` and then add the following line into any source file where you want to create annotations:

```
#include "libstreamline_annotate"
```

Some of the libraries that are required to compile the given code aren't included by default with Windows or Microsoft Visual Studio. You might need to add them manually:

- In Microsoft Visual Studio, right-click on your project name within the Solution Explorer and select Manage NuGet Packages for <project_name>...
- Click Browse, then search for and select the package called pthreads.
- Select all of the options and click Install.

4. Create region markers in your application

To enable annotation for your application, you must add this line to one of your C++ classes:

```
ANNOTATE_SETUP;
```

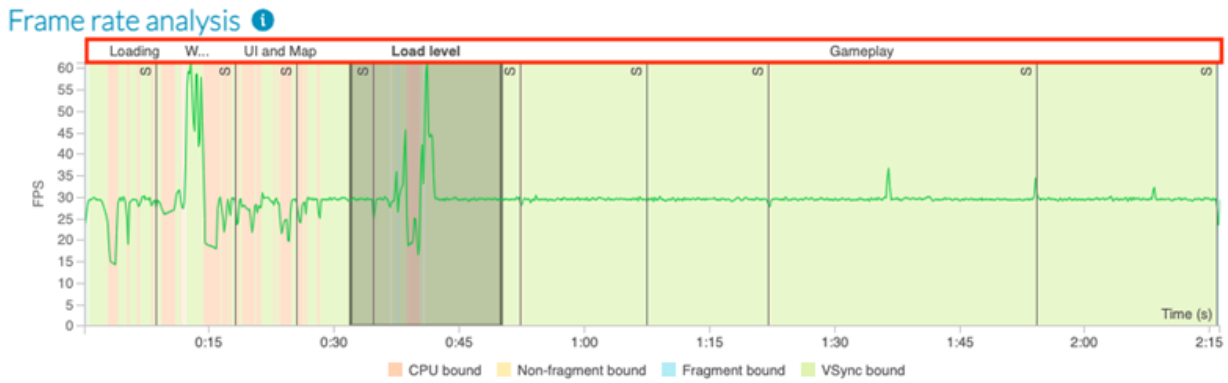
Then, to denote sections of your code as a region, enclose it with `ANNOTATE_MARKER_STR` statements as follows, replacing `<region_name>` with a unique name for that section, to be used in the Performance Advisor report.

```
ANNOTATE_MARKER_STR("Region Start <region_name>");
// some code in your application...
ANNOTATE_MARKER_STR("Region End <region_name>");
```

Do not use the same region name multiple times.

When you generate a Performance Advisor report, the region names can be seen in the Frame rate analysis chart, and the report contains dedicated analysis sections for each region.

Figure 4-1: Region names in Performance Advisor



5. Configure your project and build the application

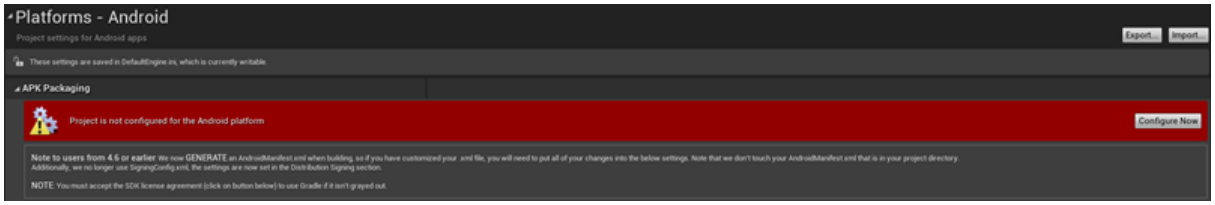
To build an APK from your Unreal Engine project that can be tested on a mobile device with Arm Mobile Studio tools, follow these steps:

1. Go to Edit > Project Settings > Platforms > Android SDK and check that the paths to your Android SDK, NDK and Java JDK are set correctly. These are required in order to compile the program.

Figure 5-1: Check SDK, NDK and JDK paths in Unreal Engine

2. Next, go to Platforms > Android and select Configure Now.

Figure 5-2: Configure for Android in Unreal Engine

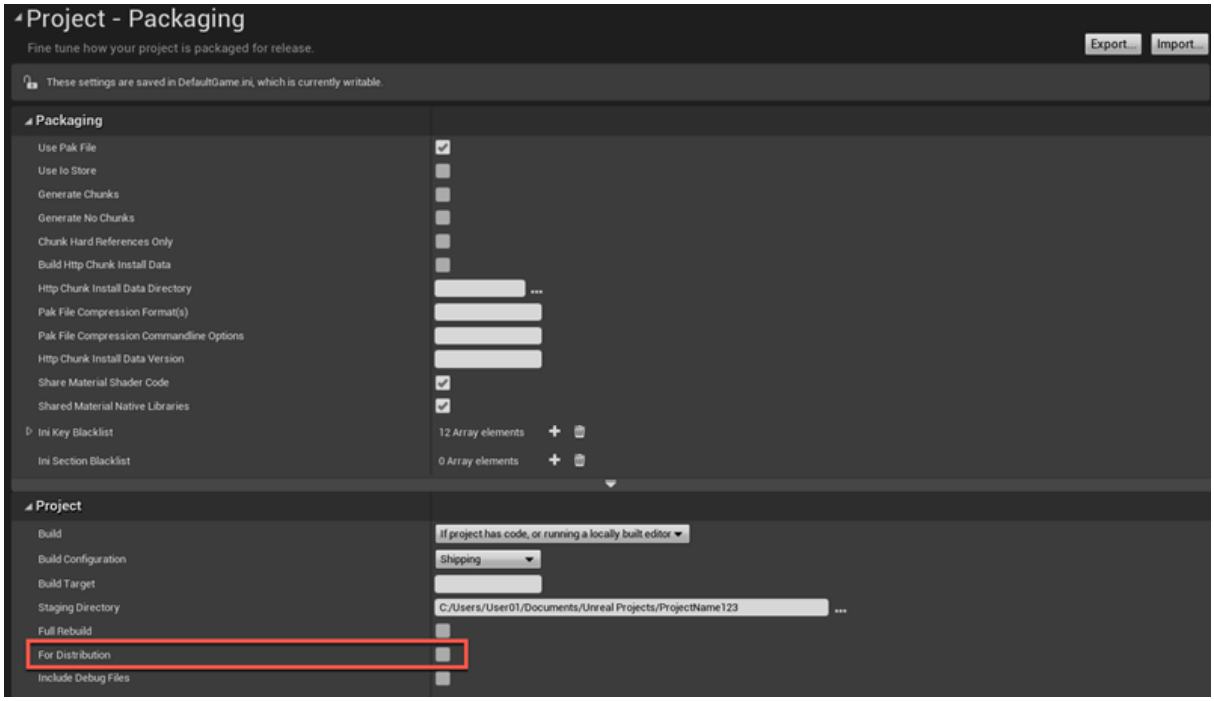


3. Consider setting the following options:
   a. Android Package Name - change this to something that will identify this application, for example, com.mycompany.mygame. You will need to look for this name when capturing data from the device with Arm Mobile Studio tools.
   b. Package game data inside .apk? - Choose this setting to combine the OBB data in to the same file instead of building a separate OBB file.This makes the APK easier to upload to a device when using Android Debug Bridge (`adb`).
   c. Support armv7/arm64 - Choose whether to support 32-bit (armv7) and or 64-bit (arm64) architectures. Selecting both significantly increases the size of the APK.
   d. Support OpenGL/Vulkan - Choose whether you want to build an OpenGL or Vulkan compatible application, or both.

Be aware of the following limitation in Performance Advisor when generating reports for Vulkan applications - Slow capture with lwi_me.py on Vulkan applications.

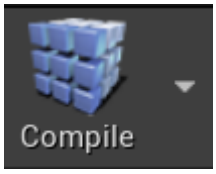
4. In Project Settings > Project > Packaging > Project, ensure that the For Distribution checkbox is not set. We want the APK to be debuggable, so that Arm Mobile Studio tools can collect data from it, this can not be done with a production build.

Figure 5-3: Check For Distribution is not set in Unreal Engine



5. Before packaging your project into an APK, use the Compile button on the Unreal Editor toolbar to test that your code compiles correctly. This reports any compilation errors in your code, and provides an output log at the bottom of the editor. Resolve any errors you find before packaging your project.

Figure 5-4: Compile button in Unreal Editor



6. To package your project into an APK, select File > Package Project > Android > Android (Multi:ASTC, DXT, ETC2) and specify an output directory for the file. Be aware that packaging might take some time, depending upon the size of your application.

Your APK can now be profiled with Arm Mobile Studio tools, and your markers will be displayed in Streamline and any Performance Advisor reports you generate. Refer to Get started with Performance Advisor for instructions on how to profile your application. In addition to the Performance Advisor report, you can also explore the Streamline capture, to see in detail how your game uses the CPU and GPU resources in the device.

6. Next steps

Now that you have a debuggable APK of your game, you can use Arm Mobile Studio tools to profile it, and your markers will be displayed in Streamline and any Performance Advisor reports you generate.

- Refer to Get started with Performance Advisor for instructions on how to profile your application.
- In addition to the Performance Advisor report, you can also explore the Streamline capture, to see in detail how your game uses the CPU and GPU resources in the device.
- To see all the graphics API calls your application makes, and investigate how objects are drawn to the screen, try Graphics Analyzer.
- To see how individual shader programs would perform on any of the available Mali GPUs, use Mali Offline Compiler.