RenderDoc for Arm® GPUs

Version 2024.4

User Guide

Non-Confidential

Issue 00
109669_2024.4_00_en

RenderDoc for Arm® GPUs User Guide

This document (109669_2024.4_00_en) was issued on 2024-09-05. There might be a later issue at http://developer.arm.com/documentation/109669

The product version is 2024.4.

See also: Proprietary notice | Product and document information | Useful resources

Start reading

If you prefer, you can skip to the start of the content.

Intended audience

This document is intended for software developers who want to use RenderDoc for Arm GPUs for frame-based graphical analysis of Android and Linux applications.

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.

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.

Contents

1. Introduction to RenderDoc for Arm® GPUs

RenderDoc for Arm® GPUs enables RenderDoc users to capture, analyze, and debug graphics applications from Windows, Linux, and Android devices. This section introduces RenderDoc for Arm® GPUs and describes how it works with RenderDoc.

Overview of RenderDoc for Arm® GPUs

RenderDoc for Arm® GPUs is an Arm fork of the RenderDoc open-source graphics API debugger. The Arm release includes support for API features and extensions that are available on the latest Arm GPUs, but are not yet supported in upstream RenderDoc.

Arm contributes changes to the upstream project, but some Arm-specific or Android-specific features are only available in the Arm fork.

Arm-specific features

RenderDoc for Arm® GPUs is based on upstream RenderDoc 1.33 and has the following extensions:

- Improved support for capturing applications using Vulkan ray tracing.
- Improved support for capturing applications on Android devices.
- Improved compatibility with devices from a range of Android device manufacturers.
- Improved support for capturing applications on automotive Linux devices.
- Binary RenderDoc releases for developers using macOS host machines.

Support

The following resources provide additional help and information:

Resource	Link
RenderDoc documentation	https://renderdoc.org/docs/index.html
To ask a question directly, you can email the Arm® Performance Studio team	performancestudio@arm.com
RenderDoc for Arm® GPUs on Arm Community	Graphics, Gaming, and VR community forum

2. Get started with RenderDoc for Arm® GPUs on Android

This tutorial describes how to set up an Android target and capture frames, and we highlight some of the debug features that are available in RenderDoc for Arm® GPUs.

Learn how to:

- Setup your Android target to prepare your computer and Android target.
- Capture on Android with a mobile application running on your Android target.
- Analyze and debug your Android capture highlights some of the debug features that are available in RenderDoc for Arm® GPUs.

Additional learning resources

Additional tutorials and help articles are available as part of the RenderDoc documentation, including:

- *How do I use RenderDoc on Android?*
- *RenderDoc Quick Start*
- *How-to topics in the RenderDoc documentation*

2.1 Setup your Android target

Complete the required setup tasks before you use an Android target with RenderDoc for Arm® GPUs.

Before you begin

Before you can setup your Android target, you must complete the following tasks:

- Install *Android Debug Bridge*. `adb` is available with the Android SDK platform tools, which are installed as part of *Android Studio*. Alternatively, you can download them separately as part of the *Android SDK platform tools*.
- *Download Arm Performance Studio for free* and follow the installation instructions in the *Arm® Performance Studio Release Notes*.

Note

Your Android target must be running Android 9.0 or later.

Procedure

1. Connect your target to your computer through USB and ensure that the target is switched on.
2. Enable *Developer Mode* on your target.

3. On your target, go to **Settings > Developer Options** and enable **USB Debugging**. If your target asks you to authorize connection to your computer, confirm the connection.
   Test the connection by entering `adb devices` in a command-line utility. If successful, the command returns the target ID.

   ```
   adb devices
   List of devices attached
   ce12345abcdf1a1234       device
   ```

   If you see that the target is listed as unauthorized, try disabling and re-enabling **USB Debugging** on the target, and accept the authorization prompt to enable connection to the computer.
4. If you have Android Studio open, it interferes with RenderDoc debugging by attaching to the package itself. You can either close Android Studio, or disable adb integration in Android Studio using the **Tools > Android > Enable ADB integration** setting.

Next steps

After connecting and configuring your Android target, you can now perform an on-target capture using RenderDoc for Arm® GPUs.

- Capture on Android

2.2 Capture on Android

Set up and perform a capture on your Android target, ready for analysis using RenderDoc for Arm® GPUs.

Before you begin

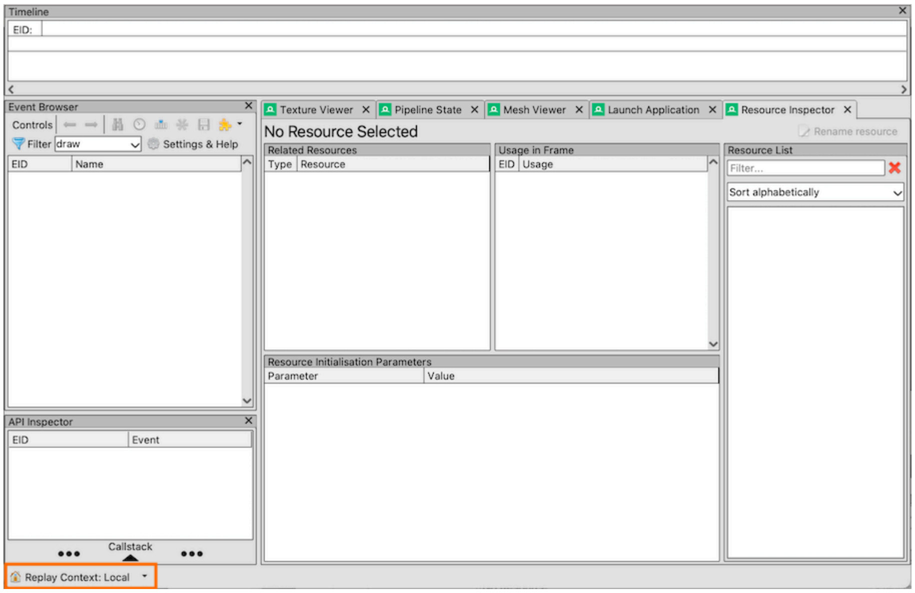
Before you begin this task, you must:

- Set up your Android target and system as described in Setup your Android target.
- Install a debuggable APK of your mobile application on your target.

Procedure

1. Connect your Android target to your host machine using USB.
2. Select your connected Android target from the **Replay Context** dropdown list at the bottom left of the RenderDoc UI.

Figure 2-1: Replay Context dropdown location in RenderDoc



If you do not see your target listed in the dropdown list, run `adb devices` in a command-line utility. The utility returns the ID and prompts RenderDoc to list the target. If it does not, see the setup tasks to check you have set up the target correctly.

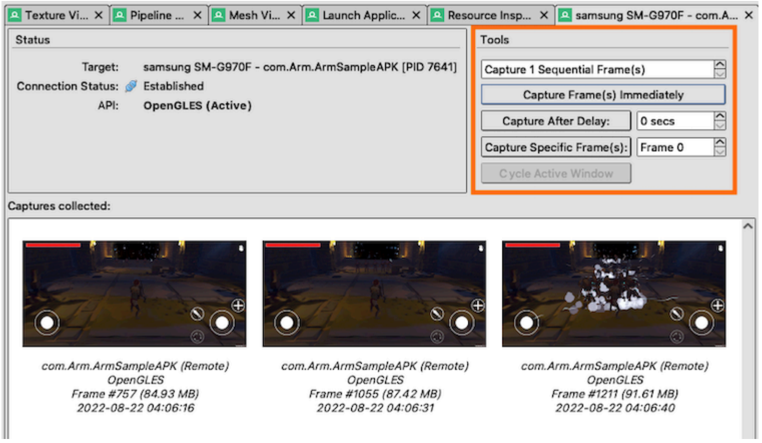
Note

A red cross next to your target indicates that you have not run RenderDoc on this target before. When you connect for the first time, RenderDoc installs its capture and replay application on the target. Now the target is shown as connected in the dropdown list.

After connecting, the RenderDoc app starts running on your target.

3. Navigate to the **Launch Application** tab in RenderDoc for Arm® GPUs. Here:
   a. Set the **Executable Path** to the application you want to debug. Click the **Browse** button to view all the installed application packages on the target. Choose the required application package folder and select the activity executable within it.
   b. Optionally, you can specify the **Working Directory**. If you do not specify this, the capture is temporarily saved in the same location as the executable.
   c. You can also specify additional **Capture Options** and specify a list of frames to capture, in the **Actions** section.
4. Click **Launch**, to start the application running on your target. After a successful launch, a new target-specific tab opens in the UI and from here you can capture your frames. You can:
   - Capture one or more frames immediately
   - Capture one or more frames after a delay
   - Capture one or more frames after a specific frame

Figure 2-2: Capture frame controls



Target: samsung SM-G970F - com.Arm.ArmSampleAPK [PID 7641]
Connection Status: Established
API: OpenGLES (Active)

Capture 1 Sequential Frame(s)
Capture After Delay: 0 secs
Capture Specific Frame(s): Frame 0

com.Arm.ArmSampleAPK (Remote)
OpenGLES
Frame #757 (84.93 MB)
2022-08-22 04:06:16

com.Arm.ArmSampleAPK (Remote)
OpenGLES
Frame #1055 (87.42 MB)
2022-08-22 04:06:31

com.Arm.ArmSampleAPK (Remote)
OpenGLES
Frame #1211 (91.61 MB)
2022-08-22 04:06:40

Note

Captured frames are stored temporarily on the target.

5. When you have finished capturing the frames of interest, keep the application running on the target while you save the frames you want to debug. Click **Save** and then choose a location on your machine. Or you can click **Open** to view them directly without saving.

Note

When you click **Save**, RenderDoc disconnects from the target.

6. To stop the mobile application running, use the Android swipe control to close the application. Keep the RenderDoc application running though, so that you can analyze and debug your captures.

Next steps

When you have finished capturing, you can then analyze, debug, and edit your frames using RenderDoc for Arm® GPUs.

- Analyze and debug your Android capture

2.3 Analyze and debug your Android capture

Use the debug features available in RenderDoc for Arm® GPUs to analyze and debug your Android capture.

The primary purpose of RenderDoc is to help you diagnose rendering problems that occur in your application. When you have captured a frame you can use the tool to interactively explore all of its API calls and rendering events. By stepping through the frame you can identify the problematic rendering events, and then review the configuration used by the event to root cause the problem.

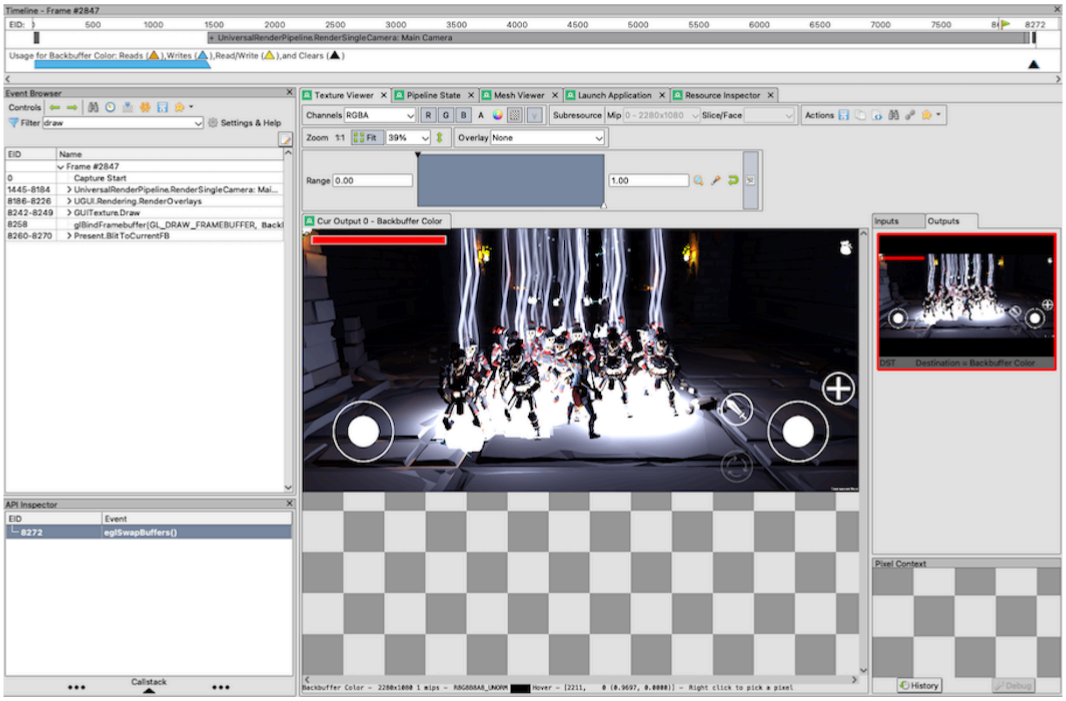
Load a saved capture

You can either load a frame capture for analysis directly after capture, or you can load a previously saved frame capture.

1. Ensure that your Android target is connected to your computer, and select your target from the **Replay Context** dropdown list.
2. If you have just taken a new frame capture, select the capture from the **Captures collected** window and click **Open**. Alternatively, you can load a previously saved frame capture from the **File > Open Capture** menu.

When the frame has loaded, it is displayed on the target and in the Texture Viewer tab, and the **Event Browser** is populated in RenderDoc for Arm® GPUs.

Figure 2-3: RenderDoc UI with a loaded frame



Filter: draw

Channels: RGBA

Mip: 0 - 2280x1080

Zoom: 39%

Overlay: None

Range: 0.00 – 1.00

EID	Name
	Frame #2847
0	Capture Start
1445-8184	UniversalRenderPipeline.RenderSingleCamera: Mai...
8186-8226	UGUI.Rendering.RenderOverlays
8242-8249	GUITexture.Draw
8258	glBindFramebuffer(GL_DRAW_FRAMEBUFFER, Backb
8260-8270	Present.BlitToCurrentFB

EID	Event
8272	eglSwapBuffers()

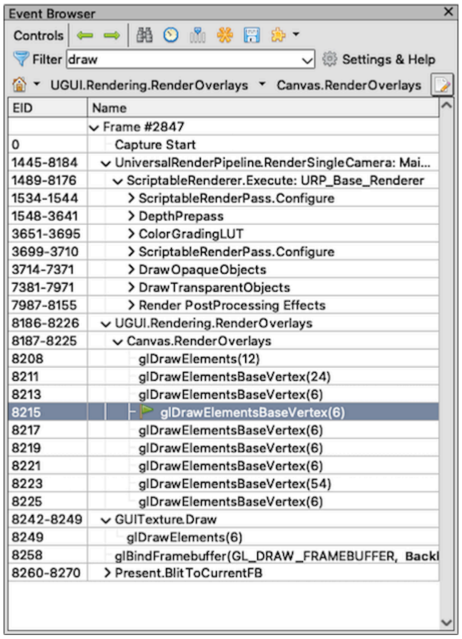
```
Backbuffer Color - 2280x1080 1 mips - R8G8B8A8_UNORM ■ Hover - [2211,    0 (0.9697, 0.0000)] - Right click to pick a pixel
```

Navigate the frame capture

Use the **Event Browser** to navigate through the frame capture. By default, the **Event Browser** shows all `action()` events, which include draws, copies, and clears. Enter a search term in the **Filter** dropdown to filter these events.

Filter expressions can be complex. For more details, see the RenderDoc documentation at *How do I filter visible events?*.

Figure 2-4: Event browser view in RenderDoc



Filter: draw

EID	Name
	Frame #2847
0	Capture Start
1445-8184	UniversalRenderPipeline.RenderSingleCamera: Mai...
1489-8176	ScriptableRenderer.Execute: URP_Base_Renderer
1534-1544	ScriptableRenderPass.Configure
1548-3641	DepthPrepass
3651-3695	ColorGradingLUT
3699-3710	ScriptableRenderPass.Configure
3714-7371	DrawOpaqueObjects
7381-7971	DrawTransparentObjects
7987-8155	Render PostProcessing Effects
8186-8226	UGUI.Rendering.RenderOverlays
8187-8225	Canvas.RenderOverlays
8208	glDrawElements(12)
8211	glDrawElementsBaseVertex(24)
8213	glDrawElementsBaseVertex(6)
8215	glDrawElementsBaseVertex(6)
8217	glDrawElementsBaseVertex(6)
8219	glDrawElementsBaseVertex(6)
8221	glDrawElementsBaseVertex(6)
8223	glDrawElementsBaseVertex(54)
8225	glDrawElementsBaseVertex(6)
8242-8249	GUITexture.Draw
8249	glDrawElements(6)
8258	glBindFramebuffer(GL_DRAW_FRAMEBUFFER, Backl
8260-8270	Present.BlitToCurrentFB

When you select an event it is highlighted with a green flag. All the other windows in the UI update to display information that is specific to the selected event. You can use this to view the render state and data resources that are used by the current event, and view the GPU output that resulted from it.

For more details, see the RenderDoc documentation at *Event Browser*

Debug a shader

Use the **Mesh Viewer** in RenderDoc to select an input vertex. Right-click anywhere in a row that is of interest, and select **Debug this Vertex** to open the vertex shader in the shader debugger.

Note

If the **Debug this Vertex** button is greyed out, this option might not be available for your device and API combination.

For more details, see the RenderDoc documentation at:

- *How do I debug a shader?*
- *How do I use shader debug information?*

Edit a shader

Shaders are one of the most important aspects of GPU processing, and errors in user shaders are a common cause of rendering problems. RenderDoc allows you to edit a shader used by an action event and replay it on the connected target. This feature allows you to quickly iterate changes without having to rebuild and deploy your entire application.

To launch the shader editor, click the **Pipeline State** tab then click the **Edit** button next to the shader. A text editor opens where you can make your edits to the code. To save and recompile the code, click **Apply changes**.

Figure 2-5: Edit shader button

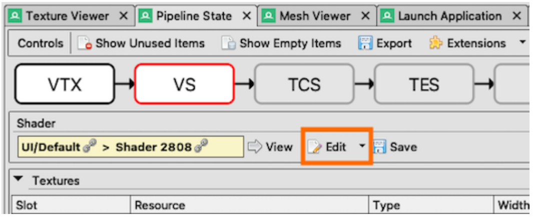
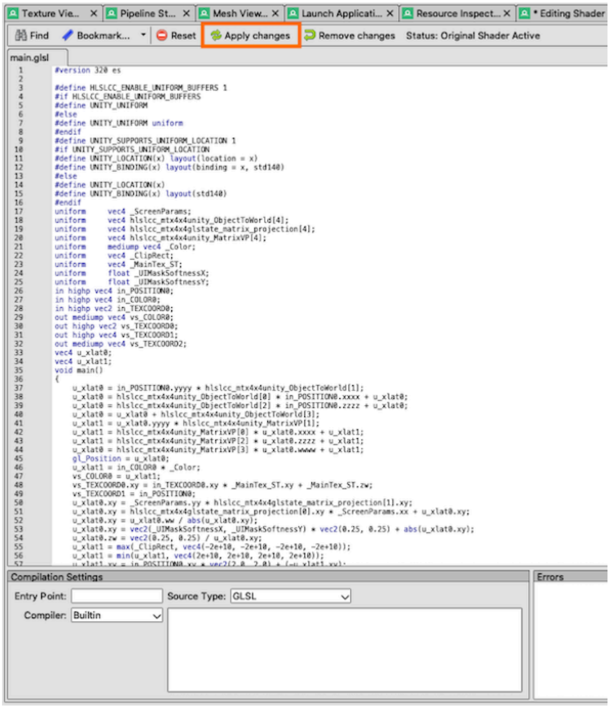


Figure 2-6: Editing the shader code



```
#version 320 es

#define HLSLCC_ENABLE_UNIFORM_BUFFERS 1
#if HLSLCC_ENABLE_UNIFORM_BUFFERS
#define UNITY_UNIFORM
#else
#define UNITY_UNIFORM uniform
#endif
#define UNITY_SUPPORTS_UNIFORM_LOCATION 1
#if UNITY_SUPPORTS_UNIFORM_LOCATION
#define UNITY_LOCATION(x) layout(location = x)
#define UNITY_BINDING(x) layout(binding = x, std140)
#else
#define UNITY_LOCATION(x)
#define UNITY_BINDING(x) layout(std140)
#endif
uniform     vec4 _ScreenParams;
uniform     vec4 hlslcc_mtx4x4unity_ObjectToWorld[4];
uniform     vec4 hlslcc_mtx4x4glstate_matrix_projection[4];
uniform     vec4 hlslcc_mtx4x4unity_MatrixVP[4];
uniform     mediump vec4 _Color;
uniform     vec4 _ClipRect;
uniform     vec4 _MainTex_ST;
uniform     float _UIMaskSoftnessX;
uniform     float _UIMaskSoftnessY;
in highp vec4 in_POSITION0;
in highp vec4 in_COLOR0;
in highp vec2 in_TEXCOORD0;
out mediump vec4 vs_COLOR0;
out highp vec2 vs_TEXCOORD0;
out highp vec4 vs_TEXCOORD1;
out mediump vec4 vs_TEXCOORD2;
vec4 u_xlat0;
vec4 u_xlat1;
void main()
{
    u_xlat0 = in_POSITION0.yyyy * hlslcc_mtx4x4unity_ObjectToWorld[1];
    u_xlat0 = hlslcc_mtx4x4unity_ObjectToWorld[0] * in_POSITION0.xxxx + u_xlat0;
    u_xlat0 = hlslcc_mtx4x4unity_ObjectToWorld[2] * in_POSITION0.zzzz + u_xlat0;
    u_xlat0 = u_xlat0 + hlslcc_mtx4x4unity_ObjectToWorld[3];
    u_xlat1 = u_xlat0.yyyy * hlslcc_mtx4x4unity_MatrixVP[1];
    u_xlat1 = hlslcc_mtx4x4unity_MatrixVP[0] * u_xlat0.xxxx + u_xlat1;
    u_xlat1 = hlslcc_mtx4x4unity_MatrixVP[2] * u_xlat0.zzzz + u_xlat1;
    u_xlat0 = hlslcc_mtx4x4unity_MatrixVP[3] * u_xlat0.wwww + u_xlat1;
    gl_Position = u_xlat0;
    u_xlat1 = in_COLOR0 * _Color;
    vs_COLOR0 = u_xlat1;
    vs_TEXCOORD0.xy = in_TEXCOORD0.xy * _MainTex_ST.xy + _MainTex_ST.zw;
    vs_TEXCOORD1 = in_POSITION0;
    u_xlat0.xy = _ScreenParams.yy * hlslcc_mtx4x4glstate_matrix_projection[1].xy;
    u_xlat0.xy = hlslcc_mtx4x4glstate_matrix_projection[0].xy * _ScreenParams.xx + u_xlat0.xy;
    u_xlat0.xy = u_xlat0.ww / abs(u_xlat0.xy);
    u_xlat0.xy = vec2(_UIMaskSoftnessX, _UIMaskSoftnessY) * vec2(0.25, 0.25) + abs(u_xlat0.xy);
    u_xlat0.zw = vec2(0.25, 0.25) / u_xlat0.xy;
    u_xlat1 = max(_ClipRect, vec4(-2e+10, -2e+10, -2e+10, -2e+10));
    u_xlat1 = min(u_xlat1, vec4(2e+10, 2e+10, 2e+10, 2e+10));
    u_xlat1.xy = in_POSITION0.xy * vec2(2.0, 2.0) + (-u_xlat1.xy);
```

Compilation Settings

Entry Point:

Source Type: GLSL

Compiler: Builtin

Errors

Note

Any changes to the shader will affect all action events that use this shader.

For more details, see the RenderDoc documentation at *How do I edit a shader?*

More things you can do with RenderDoc for Arm® GPUs

This section has described a few of the things you can do with RenderDoc for Arm® GPUs after you have captured a frame. See the *RenderDoc documentation* to explore the full list of features.

Proprietary Notice

This document is protected by copyright and other related rights and the use or implementation of the information contained in this document may be protected by one or more patents or pending patent applications. No part of this document may be reproduced in any form by any means without the express prior written permission of Arm Limited ("Arm"). No license, express or implied, by estoppel or otherwise to any intellectual property rights is granted by this document unless specifically stated.

Your access to the information in this document is conditional upon your acceptance that you will not use or permit others to use the information for the purposes of determining whether the subject matter of this document infringes any third party patents.

The content of this document is informational only. Any solutions presented herein are subject to changing conditions, information, scope, and data. This document was produced using reasonable efforts based on information available as of the date of issue of this document. The scope of information in this document may exceed that which Arm is required to provide, and such additional information is merely intended to further assist the recipient and does not represent Arm's view of the scope of its obligations. You acknowledge and agree that you possess the necessary expertise in system security and functional safety and that you shall be solely responsible for compliance with all legal, regulatory, safety and security related requirements concerning your products, notwithstanding any information or support that may be provided by Arm herein. In addition, you are responsible for any applications which are used in conjunction with any Arm technology described in this document, and to minimize risks, adequate design and operating safeguards should be provided for by you.

This document may include technical inaccuracies or typographical errors. THIS DOCUMENT IS PROVIDED "AS IS". ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, any patents, copyrights, trade secrets, trademarks, or other rights.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Reference by Arm to any third party's products or services within this document is not an express or implied approval or endorsement of the use thereof.

This document consists solely of commercial items. You shall be responsible for ensuring that any permitted use, duplication, or disclosure of this document complies fully with any relevant

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

PRE-1121-V1.0

Product and document information

Read the information in these sections to understand the release status of the product and documentation, and the conventions used in Arm documents.

Product status

All products and services provided by Arm require deliverables to be prepared and made available at different levels of completeness. The information in this document indicates the appropriate level of completeness for the associated deliverables.

Product completeness status

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

Revision history

These sections can help you understand how the document has changed over time.

Document release information

The Document history table gives the issue number and the released date for each released issue of this document.

Document history

Issue	Date	Confidentiality	Change
2024.4-00	5 September 2024	Non-Confidential	New document for v2024.4

Change history

For information about the technical changes to the RenderDoc for Arm® GPUs User Guide, see the *Arm® Performance Studio Release Notes*.

Conventions

The following subsections describe conventions used in Arm documents.

Glossary

The Arm Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Arm documentation uses typographical conventions to convey specific meaning.

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<and>`	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

Caution

We recommend the following. If you do not follow these recommendations your system might not work.

Warning

Your system requires the following. If you do not follow these requirements your system will not work.

Danger

You are at risk of causing permanent damage to your system or your equipment, or harming yourself.

Note

This information is important and needs your attention.

Tip

A useful tip that might make it easier, better or faster to perform a task.

Remember

A reminder of something important that relates to the information you are reading.

Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® Performance Studio Release Notes	107649	Non-Confidential
Download Arm Performance Studio for free	–	Non-Confidential

Non-Arm resources	Document ID	Organization
Android Debug Bridge	–	Android Developers
Android SDK platform tools	–	Android Developers
Android Studio	–	Android Developers
Developer Mode	–	Android Developers
Event Browser	–	RenderDoc
How do I debug a shader?	–	RenderDoc
How do I edit a shader?	–	RenderDoc
How do I filter visible events?	–	RenderDoc
How do I use RenderDoc on Android?	–	RenderDoc
How do I use shader debug information?	–	RenderDoc
How-to topics in the RenderDoc documentation	–	RenderDoc
RenderDoc documentation	–	RenderDoc
RenderDoc Quick Start	–	RenderDoc