# arm

## Arm<sup>®</sup> Mobile Studio 2023.5

Product revision: r23p5-00rel0

## **Release Note**

#### Non-Confidential

Issue 00

Copyright <sup>©</sup> 2023 Arm Limited (or its affiliates). DSHVE-DC-06003 All rights reserved.

### Arm Mobile Studio 2023.5 Release Note

Copyright © 2023 Arm Limited (or its affiliates). All rights reserved.

#### Non-Confidential Proprietary Notice

This document is protected by copyright and other related rights and the practice 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. 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 implementations infringe any third party patents.

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, has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

This document may include technical inaccuracies or typographical errors.

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.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly 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.

The Arm corporate logo and words marked with <sup>®</sup> or <sup>™</sup> are registered trademarks or trademarks of Arm Limited (or its affiliates) in the US and/or elsewhere. All rights reserved. Other brands and names mentioned in this document may be the trademarks of their respective owners. Please follow Arm's trademark usage guidelines at https://www.arm.com/company/policies/trademarks.

Copyright © 2023 Arm Limited (or its affiliates). All rights reserved.

Arm Limited. Company 02557590 registered in England. 110 Fulbourn Road, Cambridge, England CB1 9NJ. (LES-PRE-20349)

### 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.

#### Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on Arm Mobile Studio, 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.

To report offensive language in this document, email terms@arm.com.

# Contents

1	Release overview	5
1.1	Product description	5
1.1.1	Component versions	5
1.2	Release status	5
1.3	Introducing Frame Advisor	5
1.3.1	Feedback	6
1.3.2	The near-term roadmap	7
1.3.3	Get started with Frame Advisor	7
1.4	Changes in this release	9
1.4.1	Mobile Studio	9
1.4.2	Streamline	9
1.4.3	Frame Advisor	10
1.4.4	Mali Offline Compiler	10
1.4.5	Graphics Analyzer	10
1.5	Known issues in this release	10
1.5.1	Streamline	10
1.5.2	Frame Advisor	11
2	Support	13
2.1	How-to videos	
2.2	Host OS support	
2.3	Target OS support	
2.4	Related projects	
2.4.1		
2.4.2		
2.4.3		
2.4.4		
3	Installation	17
3.1	Install on Windows	17
3.2	Install on macOS	17
3.3	Install on Linux	18

# 1 Release overview

The following sections describe the product and its quality status at time of release.

## 1.1 Product description

Arm<sup>®</sup> Mobile Studio is a tool suite enabling Android application developers to detect performance bottlenecks in their Arm CPU software and Arm Immortalis<sup>™</sup> and Arm Mali<sup>™</sup> GPU rendering. Profiling is provided through analysis of performance counters from the hardware, and the target application's graphics API usage.

This release of Arm Mobile Studio includes:

- Streamline, for profiling application software and system rendering performance. Streamline integrates **Performance Advisor**, a reporting tool used for automating rendering performance analysis and reporting in continuous integration deployments.
- NEW! Frame Advisor, for profiling rendering efficiency and usage of graphics APIs.
- Mali Offline Compiler, for static analysis of shader programs and compute kernels.
- Graphics Analyzer, for debugging and inspecting usage of graphics APIs.

#### 1.1.1 Component versions

This release of Arm Mobile Studio includes the following tool versions:

- Streamline 8.9
- Frame Advisor 1.0
- Mali Offline Compiler 8.2
- Graphics Analyzer 5.12.2

## 1.2 Release status

This is the REL quality release of the Arm Mobile Studio 2023.5 (r23p5-00rel0) software.

## 1.3 Introducing Frame Advisor

This release introduces a new tool into the studio, an early access release of Frame Advisor, the next generation of our API-aware performance tooling. This tool is a graphics profiler, which captures the API calls and GPU output for selected application frames.

This tool is designed to focus on performance analysis and linting for Arm<sup>®</sup> GPU best practice recommendations, enabling you to see how your application is performing on Arm GPUs and uncover performance issues. You can use the analysis that Frame Advisor provides to

Copyright  $^{\odot}$  2023 Arm Limited (or its affiliates). All rights reserved. Non-Confidential

understand how your frame is built up using GPU workloads, the data flow between these GPU workloads, and the efficiency of the rendering commands within each render pass workload.

eme Hierarchy 🛛 🕄	Render Graph	80	API Calls							88	Detailed Metrics (Call 1791)
+			(Search function c		4						Frame 999 > Render pass 10 > Call 1791 Hide definitions
<ul> <li>Frame 999 12 passes, 208 draws, 111</li> <li>Render pass 0.2 draws, 0 prims</li> <li>Render pass 1.2 draws, 0 prims</li> <li>Render pass 2.2 draws, 2 prims</li> </ul>			TID Ct 11050 HT	(/D184C/_	al 1//8	Return value	UL_NU_ENKL	Function call glDisable(cap++ elStercilFuncSe	Duration (ms) 0.00	Usmodelled *	This analysis provides information about the size and efficiency of the me data used in a single draw call.
> Render pass 3 179 draws, 111397			11655 8		1780			giStercilOpSep	0.00		Mesh complexity
Render pass 4.2 draws, 2 prims Render pass 5.2 draws, 2 prims	Renders 200-100	10000	11655 01		1781			gisterciFuncs	0.00		Efficient meshes minimize the number of vertices and primitives, and the
Render pass 6 2 draws, 2 prims	Resolution 1170-140 Resolution 1170-140 Disc	avials 1	11655 8		1782			elStencilOuSec	0.00		of each vertex in memory.
Render pass 7 2 draws, 2 prims Render pass 8 3 draws, 2 prims	Resident States States	ulation 1170+540	11655 0		1783			giSterciMaski	0.00		Vertices shaded: 4     Primitives: 2
Render pass 9 3 draws, 2 prims	Coart house 114 Coart house 114	tarð Tastari 114 🔸 📃	11655 0		1784			giPushDebugG	0.00		Index range: 4
Render pass 10 7 draws, 68 prims glinvalidateFramebuffer Call 17	Single 12202     Resulting 1220-1000     Resulting 1220-1000	( <u> </u>	11655 0		1785			glUseProgram(	0.01		<ul> <li>Index size: 2 bytes</li> </ul>
glDrawElements Call 1725, 2 p			11655 01		1785			glUniform4tv0k	0.01		Vertex size: 24 bytes
glDrawElements Call 1752, 62; glDrawElements Call 1763, 2 p.			11655 0		1787				0.01		<ul> <li>Mesh bandwidth: 140 bytes</li> </ul>
giDrawElements Call 1791, 2 p								g/BufferSubDat			Mesh locality
glinvalidateFramebuffer Call 17 glinvalidateFramebuffer Call 17			11655 8:		1788			gVertexAttribl	0.00		Efficient indexed meshes reuse vertices multiple times, reducing the a
ender pass 11 2 draws, 1 prim			11655 8:		1789			gVertexAttribi	0.00		of redundant shading and data fetch that is required to process the mo indix rate shows the number of unique indices per primitive, giving a m
			11655 8:		1790			gVertexAttribi	0.00		of how vertices are shared across multiple primitives. Increasing the n
			11655 8	(7b1s4c7_	1791		GL_NO_ERR.	gDrawDemon	0.12	-	of times that each vertex is used will reduce this measure. Vertex efficiency shows the ratio of vertex shader invocations to the nur
	4 Famebulters		Content Metrics							80	vertex encoders in the index buffer. Values over 1 indicate that either un unique indices in the index buffer. Values over 1 indicate that either un vertices are being shaded, due to overshading, or that vertices are being processed multiple times, due to poor locality.
	Frame 799 - Render pass 10 - Call 1791	800 ·   60		bender pass		Name	Prins	Vert efficienc			Sparse efficiency shows the ratio of vertice indices shaded compared to member of unique indices in the index buffer. Any GPUs always shade in groups of 4 consecutive indices, even if those indices are not used model. Values over 1 indicate that the model is overhading due to an indices in these groups of 4. Minimize overhading by Tigthy packing to ensure all vertices between the the mini and max value are used.
			999	2		glDrawElem	29	2 1.00	21		Temporal locality shows the number of indices (mean and standard devi between neuse of an index value. Reducing this metric improves cache
			199			glDrawElem	29				efficiency of the post-transform cache during vertex shading.
			999	3		glDrawElem	50				Spatial locality shows the index difference (mean and standard deviati between neighbouring indices. Reducing this metric between neighbouring
			999			giDrawElem_	66				indices improves memory access and cache efficiency during vertex st • Index rate: 2.00
			999			giDrawElem.	15				Vertex efficiency: 1.00
	the start	0	999			gDrawElem.	280				Sparse efficiency: 1.00
			999			giDrawElem.	29				Temporal locality: 2.00 (StdDev 1.00)
		~	999				15				Spatial locality: 1.00 (StilDev 0.62)
		$\langle \rangle$		3		glDrawElem					Mesh redundancy
			999	3		glDrawElem	29				Efficient meshes minimize the amount of redundant data in their mod
			999	3		glDrawElem	25				Degenerate primitives shows the number of primitives that have zero a small number of degenerate primitives are expected for encoding spa
		-	999	3		glDrawElem	25				jumps in the mesh, but a significant percentage may indicate a mesh en
		A second	909	3	485	glDrawElem	29	1 0.99	31		issue.
	(185	54, 1078  (20, 32, 56, 255)	999	3	495	glDrawElem	63	9 1.00	31		Duplicate vertices shows the number of vertices that have identical dat another vertex in the model. Duplicated vertices are a waste of bandw
			999	3	497	glDrawElem	16	8 0.99	31		aim to make this metric as small as possible.
			909	3	533	glDrawElem	119	4 1.00	30		Vertex padding shows the number of bytes of unused padding in each v accounting for space between attributes or between vertices. Padding
	Color 0 2540 × 5080		999	3	540	giDrawElem.	28	8 1.00	33		same cache line as used data is still fetched from memory, so aim to ma metric as as small as possible.
	21		909			glDrawElem.	313	7 1.66	28		
											<ul> <li>Degenerate primitives: 0</li> </ul>

Using Frame Advisor, you can:

- See the structure of your frame in the Render Graph view, which shows how the GPU processes your render passes and how data flows between them.
- Inspect how your application constructs each render pass by stepping through draw calls in the Framebuffer view, including support for the popular overdraw visualization from Graphics Analyzer.
- Find expensive or inefficient draw calls in the tabular Metrics view, and then analyze problem cases in the Detailed Metrics view, which presents a comprehensive analysis of a single draw call.

Frame Advisor will focus on profiling use cases and performance analysis. For functional debugging we recommend using the open-source RenderDoc tool.

### 1.3.1 Feedback

Frame Advisor is a new tool, and this first early access release focuses on providing a small number of analysis features that we believe are most useful for mobile application developers.

This first release has several known issues, and we have a long roadmap of features and optimizations to implement, so this tool will continue to improve and evolve over the next few years. This version has been primarily developed on Windows and macOS, and we would highlight that there are some dependency issues on Ubuntu Linux that require users to manually install some of the missing libraries. This will be fixed in the next release. Please refer to the Known Issues in section 1.5.2 of this document for information about missing functionality and performance problems in this release.

We love to hear developer feedback, and prioritize things that developers ask for, so please let us know about any bugs you encounter, or feature requests for a future release.

You can send feedback using this form, or you can email us at mobilestudio@arm.com.

Copyright  $^{\odot}$  2023 Arm Limited (or its affiliates). All rights reserved. Non-Confidential

#### 1.3.2 The near-term roadmap

As a new tool, we would like to let you know where we are headed over the next few releases. Here is a look at what we are planning.

Important! This roadmap is subject to change as we get feedback from developers.

- Host support:
  - o Improved support for Linux host machines.
- Optimizations:
  - o Improve capture performance.
  - o Reduce host tool memory footprint.
  - o Reduce saved capture file size.
- API coverage
  - o Add support for missing draw types.
  - o Add support for transfer workloads.
  - o Add support for compute workloads.
  - o Add support for Vulkan 1.3 and dynamic rendering.
- Data visualizations
  - Add support for an interactive 3D mesh visualization.
  - o Add support for a tabular program/pipeline shader metrics visualization.

### 1.3.3 Get started with Frame Advisor

Here's how to get up and running with Frame Advisor.

#### Before you begin

- Ensure you have Android Debug Bridge (ADB) installed.
- Connect your device to your computer via USB.
- Your device needs to be in developer mode, with USB debugging enabled.
- The build of your Android application must be debuggable.

#### Procedure

- 1. Open Frame Advisor:
  - On Windows, from the Start menu, navigate to Arm MS <version> and select Arm MS Frame Advisor <version>.
  - On macOS, navigate to the *<install\_directory>/frame\_advisor* folder, and double-click the **Frame\_Advisor.app** file.

• On Linux, navigate to the <install\_directory>/frame\_advisor directory in a terminal, and run the frame\_advisor file:

cd <install\_directory>/frame\_advisor
./frame advisor

2. When Frame Advisor launches, select New trace.

Welcome to Frame Advisor		arm
¥	Your recent files steel_arms_vulkan.fac dark_arms.fac	
New trace		
Open file		

3. Select your device and the application you want to analyze.

Select devi	ice	(Filter search	€ Refresh
Status	Name	≜ Serial	
Connected	SM-A505FN	RZ8MC03VVEW	
elect appl	lication Show debugg	able only (Filter search	2 Refresh
Debug	Application	Activity	<b>^</b>
$\sim$	com.Arm.DarkArms	com.unity3d.player.UnityPlayerActivity	
$\checkmark$	com.arm.Kangaroo	com.epicgames.unreal.GameActivity	
$\checkmark$	com.arm.littleOverdraw	com.unity3d.player.UnityPlayerActivity	
$\checkmark$	com.arm.malideveloper.openglessdk.metaballs	.Metaballs	
$\checkmark$	com.arm.malideveloper.openglessdk.occlusionculling	.OcclusionCulling	-
Configure API settings OpenGL E Vulkan	Application settings		
		Next >	Cancel

- 4. If your application is using Vulkan change the API settings to Vulkan.
- 5. Click Next > . The application starts on the device and Frame Advisor shows the Capture screen.

Capture frames for analysis	Captured frames
	No frames captured.
Capture mode Frame count Color buffer All attachments Overdraw API: OpenGLES Connected(00:01:12) Status: Running	
Hit Space to pause	
	Analyze Cancel

- 6. On the **Capture** screen, you can optionally:
  - a. Increase the number of frames to capture (max 3).
  - b. Change the capture mode to capture all attachments, or overdraw only.
- 7. Click **Pause** just before you get to the problem scene in your application. Use the step button to focus in on the right place.
- 8. Click **Capture** to start capturing the frame(s).
- 9. When the capture completes, click **Analyze** to view the results. Frame Advisor processes the data and presents it in the Analysis screen. Now you can explore your frames to evaluate how efficiently they were rendered to the device.

See the *Frame Advisor user guide* for further details about how to analyze the data.

## 1.4 Changes in this release

This release of Arm Mobile Studio contains the following changes.

#### 1.4.1 Mobile Studio

Mobile Studio has the following changes:

• Frame Advisor is now included in the release bundle.

#### 1.4.2 Streamline

Streamline has the following changes:

- Android device connection now supports passing command line options.
- Android device connection now supports passing a non-launchable activity.

- Performance Advisor report generation is now much faster.
- **Fix:** Counter capture now works around a driver erratum that users have encountered on recent Immortalis-G715 and Mali-G715 series devices.
- Deprecation notice: Energy profiling using Arm Energy Probe, or an NI DAQ probe, is a deprecated feature. It will be removed in a future release.

#### 1.4.3 Frame Advisor

Frame Advisor has the following changes:

• This is the first release of Frame Advisor.

#### 1.4.4 Mali Offline Compiler

Mali Offline Compiler has the following changes:

- Compiler libraries updated to r45p0 DDK for Bifrost architecture, or newer.
- Variable shading rate static analysis added to vertex shader reports if emitting a perprimitive shading rate.

### 1.4.5 Graphics Analyzer

Graphics Analyzer has the following changes:

- No changes in this release.
- Deprecation notice: Graphics Analyzer is now deprecated. It will be removed in a future release, but only after we decide Frame Advisor is providing enough to be considered a viable replacement.

Graphics Analyzer has the following known issues:

• SDDAP-12605: Tool loading splash-screen renders upside down on macOS 14 (Sonoma).

## 1.5 Known issues in this release

This release of Arm Mobile Studio contains the following known issues.

### 1.5.1 Streamline

Streamline has the following known issues:

- **SDDAP-12605:** Application loading splash-screen renders upside down on macOS 14 (Sonoma).
- SDDAP-12653: Application can crash when toggling between OS light and dark themes on macOS 14 (Sonoma).
- **SDDAP-12290:** The Mali DDK can fail to emit the Perfetto data required for the scheduling timeline visualization. This can result in entries with unidentified processes and queues. It

can also result in time ranges which show as idle in the scheduler timeline when the GPU is clearly active in the counter data.

• SDDAP-11426: High DPI display scaling has been disabled by default on Linux hosts, due to persistent reliability issues across multiple distributions and graphics drivers. If desired, display scaling support can be re-enabled by setting the environment variable STREAMLINE\_ENABLE\_HIDPI to 1 and restarting the tool.

### 1.5.2 Frame Advisor

Frame Advisor has the following known issues:

• FRADV-5238: On Ubuntu 20.04, the host tool is missing a sufficiently new GLIBC library dependency. Users can manually install the missing dependency using:

```
sudo add-apt-repository -y ppa:ubuntu-toolchain-r/test
```

```
sudo apt update && apt upgrade -y
```

```
sudo apt install -y gcc-11 g++-11
```

FRADV-5236: On Ubuntu 22.04 or newer, the host tool is missing libicui18n.so.66 library dependency. Users can manually install the missing dependency using:
 wget http://security.ubuntu.com/ubuntu/pool/main/i/icu/libicu66\_66.1-2ubuntu2 amd64.deb

```
sudo apt update && sudo dpkg -i libicu66 66.1-2ubuntu2 amd64.deb
```

- FRADV-4806: Host tool does not display properly on macOS when using the dark mode system theme.
- FRADV-4931: Host tool needs further memory optimization during capture and postanalysis. It is recommended to use this version on a host machine with at least 16GB of memory. If memory allocation problems are encountered, try capturing fewer frames to reduce memory requirements.
- FRADV-4842: FAC files need further size optimization. If file storage capacity problems are encountered, try capturing fewer frames to reduce storage requirements.
- FRADV-865: Frame capture can take a long time and needs further performance optimization.
- FRADV-4841: API modelling is not yet handling indirect draws.
- FRADV-4841: API modelling is not yet handling base-vertex draws.
- **FRADV-4978:** API modelling is not yet fully handling multi-context OpenGL ES applications, although it should mostly work.
- FRADV-4979: API modelling is not yet handling resources that are uploaded and unmapped before the start of the captured frame burst.
- FRADV-4972: API modelling is not yet handling OpenGL ES vertex array objects.
- FRADV-4966: API modeling is not yet handling Vulkan descriptor set updates that use vkUpdateDescriptorSetsWithTemplate(). This can cause draw calls to be missing geometry statistics and links to be missing in the Render Graph view. This is known to cause missing geometry metrics in applications using the Unity game engine.

- FRADV-3557: API modelling is not yet handling Vulkan 1.3 or the dynamic rendering extensions.
- FRADV-4980: API modelling is not handling command buffers that are created before the captured frame burst. Note, currently we have no plan to support this functionality, as doing so would be very invasive to application performance.
- FRADV-3898: Actual and Ideal mesh memory bandwidth is not yet factoring in the position/non-position attribute use in the vertex shader; the only factor considered is the presence of padding bytes in the buffer memory layout.
- FRADV-3546: Transfer commands are not yet treated as workloads for the purposes of navigation or the Render Graph view.
- FRADV-4639: Compute dispatches are not yet treated as workloads for the purposes of navigation or the Render Graph view.
- FRADV-3558: Float image formats are not yet supported in the Framebuffer view.
- FRADV-4950: Render Graph view doesn't currently reflect the effect of Vulkan read-only attachments or optimized **storeOp** behaviors.
- FRADV-4951: Render Graph view doesn't currently reflect the effect of Vulkan resolve attachments.

# 2 Support

To help you get started we provide a number of quick start guides available online:

- Get started with Streamline
- Get started with Frame Advisor
- Get started with Performance Advisor
- Get started with Graphics Analyzer
- Get started with Mali Offline Compiler

Technical support for Arm Mobile Studio is provided via our developer forums:

• Developer forums on community.arm.com

## 2.1 How-to videos

Refer to the following videos to learn how to use Arm Mobile Studio tools.

- Streamline
- Performance Advisor
- Graphics Analyzer
- Mali Offline Compiler

To learn more about Mali GPUs and how to develop optimized graphics content for mobile devices, refer to the **Mali GPU Training Series**.

## 2.2 Host OS support

This release has been developed for the following host operating systems:

Operating system	Version
Windows	10 or newer
macOS	10.15 (Catalina) or newer
Ubuntu Linux	20.04 (Focal Fossa) or newer

## 2.3 Target OS support

This release has been developed for the following target operating systems:

#### Table 2-2: Target operating system used in developing this release

Feature	Version
Streamline	Android 8 or newer
Streamline Performance Advisor for OpenGL ES applications	Android 8 or newer with manual annotation Android 10 or newer with the Light-weight Interceptor
Streamline Performance Advisor for Vulkan applications	Android 9 or newer
Frame Advisor for OpenGL ES applications	Android 10 or newer
Frame Advisor for Vulkan applications	Android 9 or newer
Graphics Analyzer OpenGL ES	Android 8 or newer
Graphics Analyzer Vulkan	Android 9 or newer

## 2.4 Related projects

Arm provides several open-source projects that can be used by application developers as part of their application development.

#### 2.4.1 Mobile Studio for Unity package

Current version: 1.5.0 (September 2022)

The Mobile Studio for Unity package provides an open-source Unity game engine integration for Streamline and Performance Advisor. The package provides:

- C# bindings for Streamline's annotation API, allowing users to export custom software counters, and event annotations.
- Integration with the Unity profiler data source, exporting Unity object counts and memory allocations as custom software counters.

The annotation API provides a generic means to markup a Streamline capture. It can be used to emit the semantic tags that Performance Advisor reports use to denote interesting gameplay regions.

Recent changes:

• None.

The package is available on GitHub and can be imported directly into your Unity project using the Unity package manager. See the GitHub project documentation for more details.

• https://github.com/ARM-software/mobile-studio-integration-for-unity/

#### 2.4.2 ASTC Encoder texture compressor

Current version: 4.6.1 (November 2023)

The Arm ASTC Encoder (astcenc) is an open-source texture compressor for the Adaptive Scalable Texture Compression (ASTC) texture format. It supports all block sizes, all color profiles, as well as both 2D and volumetric 3D textures. The astcenc compressor can be built as

either a standalone command line application or a library that can be integrated into an existing asset creation pipeline.

4.6.1 release changes:

- Removed core codec use of **reinterpret\_cast** outside of the SIMD libraries, making the core compliant with strict-aliasing rules.
- Improved heuristic tuning and tuning limits for a small improvement to performance and small reduction in memory footprint.
- Fixed a memory leak that occurred when allocating a decompress-only context in a full compressor build.
- Improved performance on Windows systems with more than 64 cores.

The source code is available on GitHub, in addition to binary releases of the command line utility for Windows, macOS, and Linux.

• https://github.com/ARM-software/astc-encoder

#### 2.4.3 HWCPipe library

#### Current version: 2.2.0 (November 2023)

The Hardware Counter Pipe (HWCPipe) library is an open-source utility that allows applications to select and sample a set of Arm GPU performance counters. This library provides access to the same counter data that can be visualized in the Streamline tool, allowing integration of Arm GPU data into custom tooling.

2.2.0 release changes:

- Arm Midgard architecture GPUs now return an execution engine count instead of zero.
- Arm Mali-G78AE now returns a warp width instead of zero.
- Arm 5<sup>th</sup> Generation architecture GPUs now have improved counter names and expressions reflecting the semantic changes introduced by deferred vertex shading.
- A workaround has been implemented for a kernel interface version vs kernel interface implementation mismatch in some shipping Immortalis-G715 devices.

The source code is available on GitHub:

• https://github.com/ARM-software/HWCPipe

#### 2.4.4 libGPUInfo library

#### Current version: 1.0.0 (June 2023)

The libGPUInfo library is an open-source utility that can be integrated into an application to query the configuration of the Arm GPU present in the system, including the GPU model, shader core count, shader core performance characteristics, and cache size. This information can be used to adjust the application workload at runtime to match the capabilities of the device being used.

1.0.0 release changes:

- Added an option for emitting YAML output to the command-line support utility.
- Added dynamic IP configuration query support for the Mali-G310 and Mali-G510 GPUs, as the arithmetic and texturing performance of each shader core can be configured by the chipset manufacturer.

The source code is available on GitHub:

• https://github.com/ARM-software/libGPUInfo

# 3 Installation

This section describes how to install and configure Arm Mobile Studio to run on 64-bit Windows, macOS<sup>®</sup>, and Linux.

Mobile Studio requires **Android Debug Bridge (ADB)** and **Python 3.8** (or newer), to enable connection to your device. Make sure you have these tools installed and that you have configured your environment to use them.

## 3.1 Install on Windows

Arm Mobile Studio is provided with an installer executable. Double-click the **.exe** file and follow the instructions in the setup wizard.

- To open Streamline, open the Windows Start menu, navigate to the Arm Mobile Studio folder, and select the "Arm MS Streamline 2023.5" shortcut,
- Performance Advisor is a command-line tool that is part of the Streamline application. To use it to generate a performance report, you must first run the provided Python script to enable Streamline to collect frame data from the device. This process is described in detail in the **Get started with Performance Advisor tutorial**.

Once you have captured a profile with Streamline, run the Streamline-cli -pa command on the Streamline capture file. This command is added to your PATH environment variable during installation, so it can be used from anywhere. Streamline-cli.exe -pa <options> my capture.apc

- To open Graphics Analyzer, open the Windows Start menu, navigate to the Arm Mobile Studio folder, and select the "Arm MS Graphics Analyzer 2023.5" shortcut.
- To run Mali Offline Compiler, open a command terminal, navigate to your work directory, and run the malioc command on a shader program. The malioc command is added to your PATH environment variable during installation, so can be used from anywhere. malioc.exe <options> my\_shader.frag
- To open Frame Advisor, from the **Start** menu, navigate to **Arm MS <version>** and select **Arm MS Frame Advisor <version>**.

## 3.2 Install on macOS

Arm Mobile Studio is provided as a .dmg package. To mount it, double-click the .dmg package and follow the instructions. The Mobile Studio directory tree is copied to the Applications directory on your local file system for easy access.

Open the tools directly from the Arm Mobile Studio directory in your Applications directory.

• To open Streamline, go to the <installation\_directory>/streamline directory, and open the Streamline.app file.

• To run Performance Advisor, go to the <installation\_directory>/streamline directory, and double-click the Streamline-cli-launcher file. Your computer will ask you to allow Streamline to control the Terminal application. Allow this.

The Performance Advisor launcher opens the Terminal application and updates your PATH environment variable so you can run Performance Advisor from any directory.

Performance Advisor is a command-line tool that is part of the Streamline application. To use it to generate a performance report, you must first run the provided Python script to enable Streamline to collect frame data from the device. This process is described in detail in the **Get started with Performance Advisor tutorial**.

Once you have captured a profile with Streamline, run the **Streamline-cli -pa** command on the Streamline capture file to generate a performance report: Streamline-cli -pa <options> my\_capture.apc

- To open Graphics Analyzer, go to the <installation\_directory>/graphics\_analyzer/gui directory and open the Graphics Analyzer.app file.
- To run Mali Offline Compiler, go to the <installation\_directory>/mali\_offline\_compiler directory, and double-click the mali\_offline\_compiler\_launcher file.

The Mali Offline Compiler launcher opens the Terminal application and updates your PATH environment variable so you can run the **malioc** command from any directory.

To generate a shader analysis report, run the **malioc** command on a shader program: malioc <options> my\_shader.frag

On some versions of macOS, you might see a message that Mali Offline Compiler is not recognized as an application from an identified developer. To enable Mali Offline Compiler, cancel this message, then open System Preferences > Security and Privacy, and select Allow Anyway for the malioc application.

• To open Frame Advisor, navigate to the *<install\_directory>/frame\_advisor* folder, and doubleclick the **Frame\_Advisor.app** file.

## 3.3 Install on Linux

Arm Mobile Studio is provided as a gzipped tar archive. Extract this tar archive to your preferred location, using a recent version (1.13 or later) of GNU tar: tar xvzf Arm Mobile Studio 2023.5 linux.tgz

Open the tools directly from the location where you extracted the package.

• To open Streamline, go to the <installation\_directory>/streamline directory and run the Streamline file.

```
cd <installation_directory>/streamline
./Streamline
```

• Performance Advisor is a command-line tool that is part of the Streamline application. To use it to generate a performance report, you must first run the provided Python script to

enable Streamline to collect frame data from the device. This process is described in detail in the **Get started with Performance Advisor tutorial**.

```
Once you have captured a profile with Streamline, go to the <installation_directory>/streamline directory and run the Streamline-cli -pa command on the Streamline capture file to generate a performance report: cd <installation_directory>/performance_advisor ./Streamline-cli -pa <options> my_capture.apc
```

- To open Graphics Analyzer, go to the <installation\_directory>/graphics\_analyzer/gui directory and run the aga file.
   cd <installation\_directory>/graphics\_analyzer/gui ./aga
- To run Mali Offline Compiler, go to the <installation\_directory>/mali\_offline\_compiler directory and run the malioc command on a shader program.
   cd <installation\_directory>/mali\_offline\_compiler ./malioc <options> my shader.frag
- To open Frame Advisor, navigate to the <install\_directory>/frame\_advisor directory in a terminal, and run the frame\_advisor file:
   cd <install\_directory>/frame\_advisor
   ./frame\_advisor

You might find it useful to edit your PATH environment variable to add the paths to the **Streamline-cli** and **malioc** executables so that you can run them from any directory. Add the following commands to the **.bashrc** file in your home directory, so that they are set whenever you initialize a shell session:

PATH=\$PATH:/<installation\_directory>/streamline
PATH=\$PATH:/<installation\_directory>/mali\_offline\_compiler