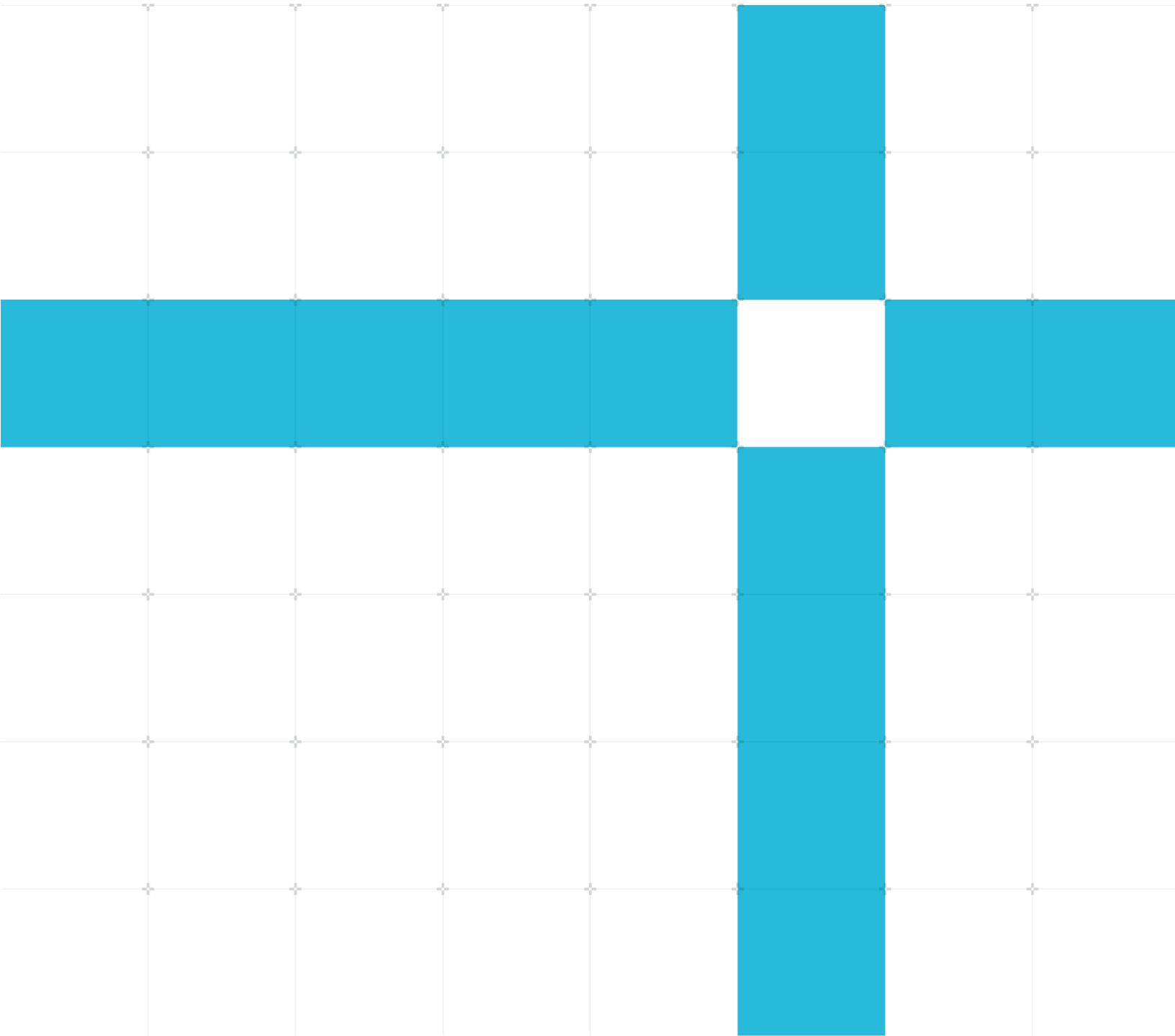
Arm® Performance Studio 2024.6

Product revision: r24p6-00rel0

Release Note

Non-Confidential

Issue 00
DSHVE-DC-06003



Arm Performance Studio 2024.6

Release Note

Non-Confidential Proprietary Notice

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 Performance 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

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

1.1 Product description

The Arm® Performance Studio tool suite enables application developers to detect performance bottlenecks in their Arm CPU software and Arm Immortalis™ and Arm Mali™ GPU rendering. Profiling is provided through analysis of performance counters from the hardware, and the graphics API usage of the target application.

This release of Arm Performance Studio includes:

- Streamline, for profiling software and graphics rendering performance. Streamline integrates Performance Advisor, a reporting tool used for automating graphics performance analysis and reporting in continuous integration deployments.
- Frame Advisor, for profiling rendering efficiency and usage of graphics APIs.
- Mali Offline Compiler, for static analysis of shader programs and compute kernels.
- RenderDoc for Arm GPUs, for debugging and inspecting usage of graphics APIs.

1.1.1 Component versions

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

- Streamline 9.4
- Frame Advisor 1.5
- Mali Offline Compiler 8.6
- RenderDoc for Arm GPUs 2024.6

1.1.2 About RenderDoc for Arm GPUs

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

Arm is contributing changes to the upstream project, if they are willing to accept them, but inevitably some Arm-specific features will only be available on the Arm fork.

1.2 Release status

This is the REL quality release of the Arm Performance Studio 2024.6 (r24p6-00rel0) software.

1.3 Feedback

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 performancestudio@arm.com.

1.4 Changes in this release

This release of Arm Performance Studio contains the following changes.

1.4.1 Performance Studio

The Performance Studio bundle has the following changes:

- No changes.

1.4.2 Streamline

Streamline has the following changes:

- **Improvement:** Download images from target configuration option has been moved from the Live mode tab to the Capture settings dialog.
- **Fix:** Immortalis-G720 and Immortalis-G725 calculation for the Texture utilization and Texture CPI metrics has been corrected.
- **Fix:** streamline_me.py will fall back to extracting tar files without using a data filter on older Python versions where data filtering is not available.

1.4.3 Frame Advisor

Frame Advisor has the following changes:

- **Improvement:** Application name shown in macOS launcher has been pretty-printed.
- **Improvement:** Android application selection now supports selecting non-launchable activities and specifying custom command line arguments.
- **Improvement:** Navigation breadcrumbs in the GUI have been merged into a single breadcrumb shown in the Frame hierarchy view.
- **Improvement:** Frame Hierarchy view now shows Transfer workloads. This is the first phase of support for Transfer workloads, so image-based transfers do not yet show as nodes in the Render Graph view or show output in the Framebuffer view. Phase two which adds this support is planned for our next release.

- **Improvement:** Render Graph view frame graph visualization has been redesigned with a cleaner look.
- **Improvement:** Render Graph view can now show Vulkan debug labels.
- **Improvement:** Content Metrics for Vertex Shading Efficiency (VSE) and Vertex Memory Efficiency (VME) now support color ramp highlighting to indicate the draws below a threshold score. Thresholds are configured in the Preferences dialog.
- **Improvement:** Detailed Metrics view for geometry analysis has been restructured to present the detailed breakdown in the context of the VSE and VME metrics.
- **Improvement:** Preferences user interface has been redesigned, with preferences pages split by functional area.
- **Improvement:** Many views give clearer error messages when the currently selected API call is not supported by the view.
- **Fix:** Framebuffer view HDR range selection slider now behaves more consistently when changing other settings and switching images.
- **Fix:** vkCmdClearAttachments() is correctly processed as a clear call by frame modelling.
- **Fix:** glColorMask() is correctly processed by frame modelling and no longer throws an error.

1.4.4 Mali Offline Compiler

Mali Offline Compiler has the following changes:

- **Improvement:** Compiler backend for Valhall CSF and later GPU architectures updated to the r53p0 DDK.
- **Improvement:** Allocated stack size is now reported, in addition to the existing metrics that shows the number of bytes stored to the stack dur to spilling.

1.4.5 RenderDoc for Arm GPUs

RenderDoc for Arm GPUs is based on upstream RenderDoc 1.36, and has the following changes since the previous Arm release:

- **Improvement:** RenderDoc baseline updated to 1.3.6 upstream version.
- **Improvement:** Mali Offline Compiler is now available as a SPIR-V cross-compile target in the shader viewer.

1.5 Known issues in this release

We are aware of several known issues impacting the tools in this Arm Performance Studio release. The tools are under active development, and we aim to resolve these in a future software release.

1.5.1 Streamline

Streamline has the following known issues:

- 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. This is fixed in the Mali DDK r47p0 release.
- 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. To re-enable display scaling support, you can set the environment variable `STREAMLINE_ENABLE_HIDPI` to 1 and restart the tool.

1.5.2 Frame Advisor

Frame Advisor has the following known issues:

- FRADV-6619: Unreal Engine can use multiple child processes to compile Vulkan shader programs, which cannot currently be intercepted. A workaround for this issue is to add the settings below to the DefaultEngine.ini and DefaultGame.ini configuration files:

```
[/Script/Engine.RendererSettings]
r.psoprecaching=0


[/Script/AndroidRuntimeSettings.AndroidRuntimeSettings]
Android.Vulkan.NumRemoteProgramCompileServices=0
```

- FRADV-865: Frame capture can take a long time and needs further performance optimization.
- FRADV-4841: API modelling does not handle indirect draws.
- FRADV-4841: API modelling does not handle base-vertex draws.
- FRADV-4978: API modelling does not fully handle multi-context OpenGL ES applications, although it should work for most content.
- FRADV-4972: API modelling does not handle OpenGL ES vertex array objects.
- FRADV-3557: API modelling does not handle 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. We have no plan to support this functionality, as doing so would be very invasive to application performance.
- 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-4951: Render Graph view does not currently reflect the effect of Vulkan resolve attachments.

- Modelling is not handling stencil-only surface attachments for OpenGL ES or Vulkan. Packed depth-stencil surfaces are supported.

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 Mali Offline Compiler

Technical support for Arm Performance Studio is provided through 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 Performance Studio tools.

- Streamline
- Performance Advisor
- Frame Advisor
- Mali Offline Compiler

To learn more about Arm Immortalis and 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:

Table 2-1: Host operating system version support

Operating system	CPU architecture	Version
Windows	x86-64	10 or later
macOS	x86-64	10.15 (Catalina) or later
Ubuntu Linux	x86-64	20.04 (Focal Fossa) or later
Ubuntu Linux	Arm AArch64	20.04 (Focal Fossa) or later

The following host operating system versions are now out of their upstream support window and are deprecated. Arm Performance Studio support will be dropped in a future release:

- Windows 10
- Ubuntu 20.04
- macOS 10, 11, and 12

The following host CPU architectures are deprecated. Arm Performance Studio support will be dropped in a future release and replaced with native Apple silicon support:

- macOS for x86-64

Table 2-2: Host operating system feature availability

Operating system	CPU architecture	Version
Windows	x86-64	Mali Offline Compiler does not support OpenCL kernels.

2.3 Target OS support

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

Table 2-3: Target operating system version support

Feature	Version
Streamline	Android 9 or later Ubuntu 20.04 (Focal Fossa) or later
Streamline Performance Advisor for OpenGL ES applications	Android 9 or later with manual annotation Android 10 or later with the lightweight interceptor
Streamline Performance Advisor for Vulkan applications	Android 9 or later
Frame Advisor for OpenGL ES applications	Android 10 or later
Frame Advisor for Vulkan applications	Android 9 or later
RenderDoc for Arm GPUs	Android 9 or later Ubuntu 20.04 (Focal Fossa) or later

2.4 Related projects

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

2.4.1 Performance Studio for Unity package

Current version: 1.5.0 (September 2022)

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

- C# bindings for the Streamline 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 way to add semantic markup a Streamline capture. It can be used to emit the markers that Performance Advisor uses to denote interesting gameplay regions in generated reports.

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/performance-studio-integration-for-unity/

2.4.2 ASTC Encoder texture compressor

Current version: 5.1.0 (November 2024)

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.

5.x series release changes:

- Added support for Arm Scalable Vector Extensions (SVE) with options for either a 128-bit and 256-bit compile-time selected backend.
- Added ability to disable use of native gather instructions on x86 AVX2 builds, as they are significantly slower than scalar fallbacks on some x86 microarchitectures.
- Added optimizations to improve performance, particularly helping SSE and NEON builds that lack native gather instructions.

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 libGPUCounters library

Current version: 2.3.0 (June 2024)

The libGPUCounters (formerly called 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.

2.3.0 release changes:

- Added support for Immortalis-G925, Mali-G725, and Mali-G625 GPUs.
- Updated counter names and derivations to match latest Streamline generator.

The source code is available on GitHub:

- https://github.com/ARM-software/libGPUCounters

2.4.4 libGPUInfo library

Current version: 1.2.0 (June 2024)

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.

1.2.0 release changes:

- Fixed remaining compilation warnings to avoid build failures when compiling with `-Werror` and `-Wpedantic` compiler feedback.

The source code is available on GitHub:

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

3 Installation

This section describes how to install and configure Performance Studio to run on 64-bit Windows, macOS®, and Linux.

Arm Performance Studio requires Android Debug Bridge (adb) and Python 3.8 (or later), 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 Performance 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 Performance Studio folder, and select the "Arm Streamline 2024.6" shortcut,
- Performance Advisor is a feature of the Streamline command-line application. 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.

  After 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 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, open the Windows Start menu, navigate to the Arm Performance Studio folder, and select the "Arm Frame Advisor 2024.6" shortcut.

3.2 Install on macOS

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

Arm recommends that you set the permissions for the installation directory to prevent other users from writing to it. This is typically achieved with the **`chmod`** command. For example, **`chmod go-w <dest_dir>`**.

Open the tools directly from the Arm Performance 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 feature of the Streamline command-line application. 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.

  After 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 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 **<installation_directory>/frame_advisor** directory, and double-click the **FrameAdvisor-gui.app** file.

3.3 Install on Linux

Arm Performance 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_Performance_Studio_2024.6_linux.tgz
```

Arm recommends that you set the permissions for the installation directory to prevent other users from writing to it. This is typically achieved with the **`chmod`** command. For example, **`chmod go-w <dest_dir>`**.

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 feature of the Streamline command-line 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.

  After 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 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 <installation_directory>/frame_advisor directory in a terminal, and run the **frame_advisor** file:

```
cd <installation_directory>/frame_advisor
./FrameAdvisor-gui
```

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
```