Mali™ GPU Performance Analysis Tool

Version: 2.2

User Guide

ARM®

DUI 0502A (ID101409)

Mali GPU Performance Analysis Tool
User Guide

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
14 October 2009	A	Non-Confidential	First release for v2.2

Proprietary Notice

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.

Web Address

`http://www.arm.com`

Contents
Mali GPU Performance Analysis Tool User Guide

List of Tables
Mali GPU Performance Analysis Tool User Guide

List of Figures
Mali GPU Performance Analysis Tool User Guide

Preface

This preface introduces the *Mali GPU Performance Analysis Tool User Guide*. It contains the following sections:

About this book

This is for the *Mali GPU Performance Analysis Tool User Guide*. It provides guidelines for using the Performance Analysis Tool to assist in the development of applications for Mali *Graphics Processing Units* (GPU). This book is part of a suite belonging to the Mali Developer Tools.

Intended audience

This guide is written for software developers who are writing OpenGL ES or OpenVG applications using the Windows XP or Linux operating system for a system with a Mali GPU.

Using this book

This book is organized into the following chapters:

Chapter 1 ***Introduction***

Read this chapter for an introduction to the Performance Analysis Tool.

Chapter 2 ***Installing the Performance Analysis Tool***

Read this chapter for instructions on how to install the Performance Analysis Tool.

Chapter 3 ***The Performance Analysis Tool User Interface***

Read this chapter for a description of the Performance Analysis Tool user interface.

Chapter 4 ***Performance Analysis Tool Data Displays***

Read this chapter for a description of the performance information that the Performance Analysis Tool can display.

Chapter 5 ***Using the Performance Analysis Tool***

Read this chapter for a description of how to use the Performance Analysis Tool to get detailed information about applications running on an OpenGL ES Emulator or a Mali GPU.

Chapter 6 ***Customizing the Performance Analysis Tool***

Read this chapter for information about how to customise the Performance Analysis Tool displays, to show the exact information you require.

Glossary Read this for definitions of terms used in this book.

Conventions

Conventions that this book can use are described in:

- *Typographical*

Typographical

The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.
bold	Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.
`monospace`	Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.
`mono`space	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`monospace italic`	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
`monospace bold`	Denotes language keywords when used outside example code.
< and >	Enclose replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0 <Rd>, <CRn>, <CRm>, <Opcode_2>`

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, `http://infocenter.arm.com`, for access to ARM documentation.

ARM publications

This book contains information that is specific to this product. See the following documents for other relevant information:

- *Mali GPU Developer Tools Technical Overview* (ARM DUI 501)
- *Mali GPU Texture Compression Tool User Guide* (ARM DUI 0503)
- *Mali GPU Shader Developer Studio User Guide* (ARM DUI 0504)

- *Mali GPU Demo Engine User Guide* (ARM DUI 0505)
- *OpenGL ES 1.1 Emulator User Guide* (ARM DUI 0506)
- *Mali GPU Binary Asset Exporter User Guide* (ARM DUI 0507)
- *Mali GPU Shader Library User Guide* (ARM DUI 0510)
- *OpenGL ES 2.0 Emulator User Guide* (ARM DUI 0511)
- *Mali GPU Offline Shader Compiler User Guide* (ARM DUI 0513).

Other publications

This section lists relevant documents published by third parties:

- *OpenGL ES 1.1 Specification* at `http://www.khronos.org`.
- *OpenGL ES 2.0 Specification* at `http://www.khronos.org`.
- *OpenGL ES Shading Language Specification* at `http://www.khronos.org`.
- *OpenVG 1.0 Specification* at `http://www.khronos.org`.
- *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2* (5th Edition, 2005), Addison-Wesley Professional. ISBN 0-321-33573-2.
- *OpenGL Shading Language* (2nd Edition, 2006), Addison-Wesley Professional. ISBN 0-321-33489-2.

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product then contact `malidevelopers@arm.com` and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to `errata@arm.com`. Give:

- the title
- the number, DUI 0502A
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter describes the Mali GPU Performance Analysis Tool. It contains the following sections:

1.1 About the Mali GPU Performance Analysis Tool

The Mali GPU Performance Analysis Tool is an application that enables you to view and analyze performance data that has been generated by the Mali GPU when rendering an image on your platform. Performance data is captured directly from the Mali GPU hardware with instrumented drivers, or loaded from a performance data file pre-recorded by an instrumented version of the OpenGL ES 1.1 or OpenGL ES 2.0 drivers or an OpenGL ES 1.1 or OpenGL ES 2.0 Emulator. Data can also be recorded from the OpenGL ES and OpenVG software drivers on OpenGL ES and OpenVG call usage.

In the Performance Analysis Tool application you can display the values of individual performance counters in graphs and tables, as required. When you play a performance data file, you can see how the values change over time, enabling you to assess performance for every single frame. You can also view the contents of the framebuffer.

1.2 Example Performance Analysis Tool projects

Two example Performance Analysis Tool projects are included with the Performance Analysis Tool. The example projects are installed into the default installation directory.

On Microsoft Windows systems this is located at:

```
C:\Program Files\ARM\ Mali GPU Developer Tools\Performance Analysis Tool v2.2\examples\
```

On Linux systems this is located at in the following subdirectory, in the directory where you decompress the tar file:

```
ARM/Mali_Developer_Tools/Performance_Analysis_Tool_v2.2/examples
```

The Performance Analysis Tool projects directory includes two performance counters data files, `lightshow_multicore.ds2` and `dump.ds2`, with two project files, `lightshow_remake.pat` and `planet.pat`.

See *Performance Analysis Tool file types* on page 3-7 for more information about these file types. See *Using the Performance Analysis Tool in offline mode* on page 5-3 for information about loading an viewing these files.

Chapter 2
Installing the Performance Analysis Tool

This section describes how to install the Performance Analysis Tool. It contains the following sections.

2.1 Installing the Performance Analysis Tool on Microsoft Windows

This section describes how to install the Performance Analysis Tool on Microsoft Windows. It contains the following sections:

- *Installation requirements for the Performance Analysis Tool on Microsoft Windows*
- *Installation procedure for the Performance Analysis Tool on Microsoft Windows*

2.1.1 Installation requirements for the Performance Analysis Tool on Microsoft Windows

To install the Performance Analysis Tool on Microsoft Windows, you require:

- Microsoft Windows XP Professional Version 2002, service pack 3
- 53MB free hard disk space.

Note

The Performance Analysis Tool has been tested successfully on a 32-bit computer.

2.1.2 Installation procedure for the Performance Analysis Tool on Microsoft Windows

The procedure to install the Performance Analysis Tool on Microsoft Windows is:

1. Locate the Mali Developer Center website at:

   `http://www.malideveloper.com`

2. Select the Performance Analysis Tool package to download.

3. Run the file `Mali_GPU_Performance_Analysis_Tool_WinXP_v`*`m`*`.`*`n`*`.exe`.

   where:

   m identifies the major version

   n identifies the minor version.

4. Select the required installation options and then click **Finish** to complete the installation.

By default, the Performance Analysis Tool is installed in:

`C:\Program Files\ARM\Mali Developer Tools\Mali GPU Performance Analysis Tool v`*`m`*`.`*`n`*

The Performance Analysis Tool examples are installed in:

`C:\Program Files\ARM\Mali Developer Tools\Mali GPU Performance Analysis Tool v`*`m`*`.`*`n`*`\examples`

2.2 Installing the Performance Analysis Tool on Linux

This section describes how to install the Performance Analysis Tool on Linux. It contains the following sections:

- *Installation requirements for the Performance Analysis Tool on Linux*
- *Installation procedure for the Performance Analysis Tool on Linux*

2.2.1 Installation requirements for the Performance Analysis Tool on Linux

To install the Performance Analysis Tool on Linux, you require:

- Red Hat Enterprise Linux, Release 4
- `GNU tar` version 1.13 or higher
- 25MB free hard disk space.

Note

The Performance Analysis Tool has been tested successfully on a 32-bit computer.

2.2.2 Installation procedure for the Performance Analysis Tool on Linux

To install the Performance Analysis Tool on Linux:

1. Locate the Mali Developer Center website at:

   `http://www.malideveloper.com`

2. Download the following package:

   `Mali_GPU_Mali_Performance_Analysis_Tool_RHEL4_v`*`m`*`.`*`n`*`.tar.gz`

   where:

   m identifies the major version

   n identifies the minor version.

3. To decompress the file:

   - open a command terminal and navigate to the directory where you have downloaded the package
   - type the following command:

     `tar -zxvf Mali_GPU_Performance_Analysis_Tool_RHEL4_v`*`m`*`.`*`n`*`.tar.gz`

By default, the Performance Analysis Tool is installed in the following subdirectory, where you decompressed the tar file:

`ARM/Mali_Developer_Tools/Mali_GPU_Performance_Analysis_Tool_v`*`m`*`.`*`n`*

The Performance Analysis Tool examples are installed in:

ARM/Mali_Developer_Tools/Mali_GPU_Performance_Analysis_Tool_v*m.n*/examples

Chapter 3
The Performance Analysis Tool User Interface

This chapter describes the Performance Analysis Tool user interface. It contains the following sections:

3.1 About the Performance Analysis Tool user interface

The Performance Analysis Tool user interface is fully customizable and enables you to display information such as framebuffer contents and performance data, in graphical and tabular formats.

Figure 3-1 shows an example of the Performance Analysis Tool user interface using a data file created by an OpenGL ES Emulator. This window shows the result of importing data files and selecting values to be displayed in different panes.

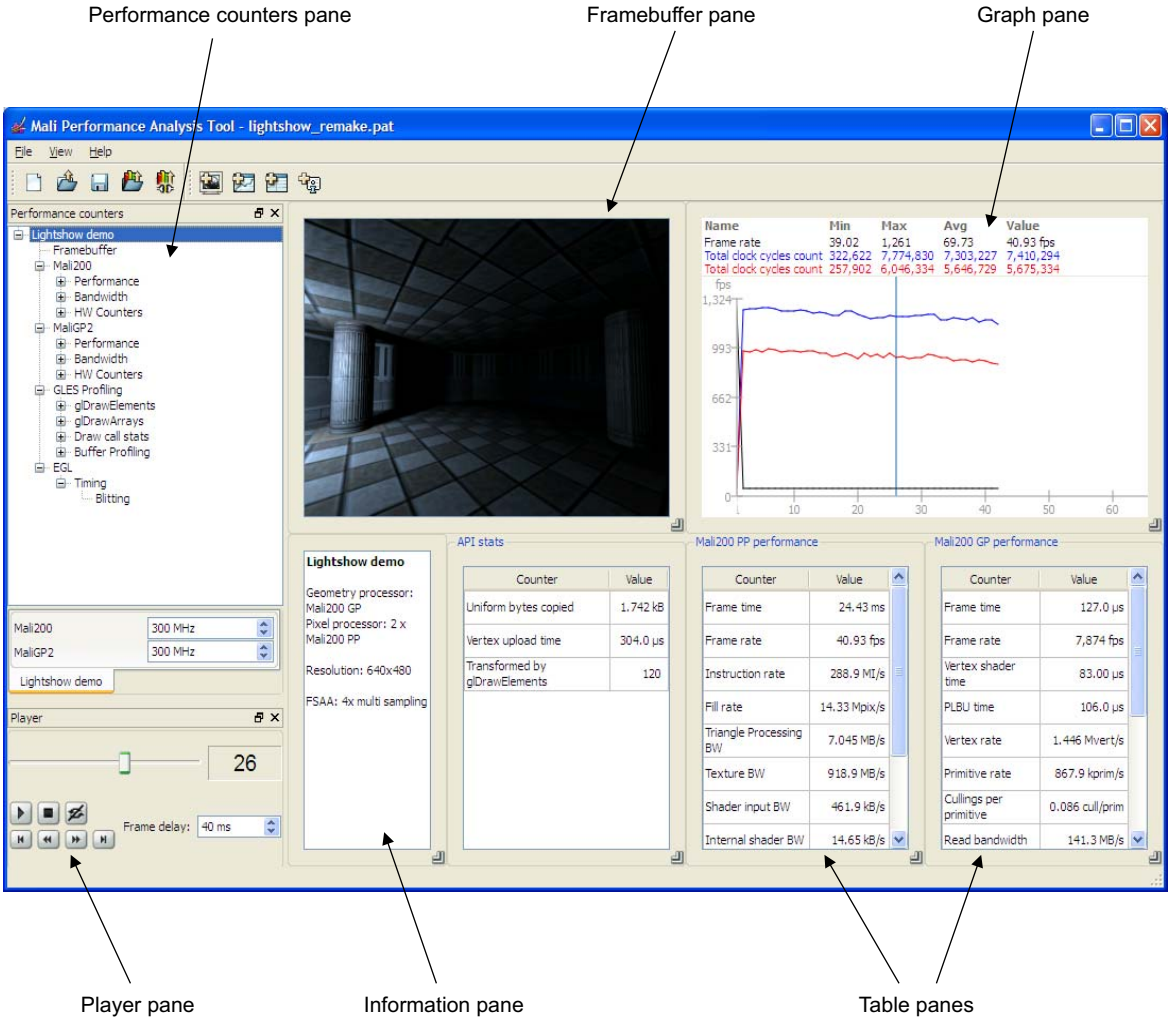

Figure 3-1 Example Performance Analysis Tool user interface

The Performance Analysis Tool user interface contains the following information:

- The title bar contains the name of the Performance Analysis Tool project currently displayed, which in this example is `lightshow.pat`.
- Buttons. Table 3-1 shows the Performance Analysis Tool buttons.

Table 3-1 Performance Analysis Tool buttons

Button	Use this button to...
	Create a new project.
	Open an existing project.
	Save a project.
	Import a performance data file.
	Connect to target
	Add a framebuffer pane.
	Add a graph.
	Add a table.
	Add an Info pane.

- The **Performance Counters** pane contains a list of available performance variables. The list includes:
  - Mali200 Performance, Mali200 Bandwidth, and Mali200 Counters are intermediate headers for groups of Mali200 performance variables
  - MaliGP2 Performance, MaliGP2 Bandwidth, and MaliGP2 Counters are intermediate headers for groups of MaliGP2 performance variables
  - frame time and frames per second are the names of performance variables that have values that can be displayed.

  - OpenGL ES API counters
  - EGL API timing counter.

- The **Frequency** pane displays the frequency (MHz) of the Mali system bus clock, this can be an assumed or simulated frequency.
- The **Player** pane enables you to control the playback of performance data files when you are using offline mode, or to start and stop the application when using online mode. The player controls operate much like the controls of consumer electronics players, such as CD players. The number of the current frame in the performance data file is shown prominently.

  When using online mode, you can also use the player to control how your application runs. For example, you can use the **pause** button to pause the application.
- The **Framebuffer** pane shows the current frame.
- The **Info display** pane shows information about the application being analyzed. You can display information about counters by dragging and dropping performance counters from the **Performance counters** pane.
- The remaining panes display performance data in graphical or tabular form, depending on the data you add to the Performance Analysis Tool window. When you play the performance data file, or start your application, if using online mode, you can see how the performance of the application changes over time.

  You can add table or graph panes to the Performance Analysis Tool main window. You can then select individual performance counters from the **Performance Counters** pane, and drag them to a particular table or graph, to display their values. This flexibility enables you to group related performance counters together. For example, you can create a table to display GLES counters and another one to display EGL counters.

  Figure 3-1 on page 3-2 shows:

  - OpenGL ES profiling pane containing graph and performance table
  - EGL timing information.

  After creating a particular set of tables and graphs, and associated performance data, you can save the settings that define the project, as a `.pat` project file. You can then use the `.pat` file to reload saved projects.

Note

The Hardware counters are only available when the Performance Analysis Tool is connected to a Mali GPU or when a performance data file has been previously dumped by a Performance Analysis Tool connected to a Mali GPU with instrumented drivers.

See Chapter 4 *Performance Analysis Tool Data Displays* for more information about the Performance Analysis Tool features.

See *Customizing graphs* on page 6-3 and *Customizing tables* on page 6-5 for more information about adding graphs and tables to the Performance Analysis Tool main window.

3.2 Modes of operation

The Performance Analysis Tool application operates in two different modes, the Performance Analysis Tool user interface is similar in these two modes.

Offline mode

In offline mode, you can use the Performance Analysis Tool to analyze performance data files locally. This enables you to view the properties of the OpenGL ES graphics without direct access to the target hardware platform.

You can load multiple performance data files at a time.

See *Using the Performance Analysis Tool in offline mode* on page 5-3 for more information about using offline mode.

Online mode

In online mode you can connect to a target hardware platform and analyze graphics applications running on the target. The Performance Analysis Tool uses a network connection to connect to the target platform.

You can connect to one online data source at a time.

Note

Online mode is only available when you connect to a hardware platform and not when using with an OpenGL ES 1.1 or 2.0 Emulator.

See *Using the Performance Analysis Tool in online mode* on page 5-7 for more information about using online mode.

You can connect to one online data source and load multiple performance data files simultaneously. This assists you in application optimization because it enables you compare multiple runs of an application.

3.3 Performance Analysis Tool file types

The Performance Analysis Tool uses the following file types:

`.ds2` files `.ds2` files are performance data files, created by the instrumented drivers or the OpenGL ES 2.0 emulator, that contain a description of the performance counters, together with the performance counter data in the form of a series of values and images. You can display one or more `.ds2` files in the Performance Analysis Tool **Performance counters** pane. See *Performance data files and counters* on page 4-3 for more information about performance data files.

`.pat` files `.pat` files contain user settings that define the tables and graphs that are displayed within a particular custom layout, or project, and also the performance counters the graphs and tables include. The `.pat` file also references the `.ds2` files that are used within a project. You use `.pat` files to reload saved projects. That is, when you load a `.pat` file, the previous layout of graphs and tables is restored.

See *Creating performance data files* on page 5-3 and *Creating project files* on page 5-4 for more information.

Chapter 4
Performance Analysis Tool Data Displays

This chapter describes the performance information that the Performance Analysis Tool displays:

4.1 About performance data

You can configure graphs and tables in the Performance Analysis Tool to display individual performance counters or groups of performance counters. The information contained in the tables and graphs at any instant corresponds to the frame currently displayed in the **Framebuffer output** pane. Appropriate units are displayed for all performance counters.

The Performance Analysis Tool is highly flexible, enabling you to add many graphs and tables that you can use to compare and analyze data.

For example, you can run your application three times with different anti-aliasing settings each time, and record performance data files for each run using the instrumented drivers. You can then play the recorded files together, looking at the same sequence number in each of the files. This enables you to study how using different levels of anti-aliasing affects your application performance.

You can also scale the data shown in your graphs and tables to a given frequency. This enables you to see roughly what the performance counter results might be, if your application used hardware running at different frequencies. For example, when you record your performance data file, you might be using a *Field-Programmable Gate Array* (FPGA) implementation of the GPU running at 20 MHz, or an ASIC implementation that runs at 400 MHz. By adjusting the value displayed on the **Frequency** pane in the Performance Analysis Tool, you can scale the performance counters to simulate the performance counter values obtained using alternative frequencies.

Note

If you scale data shown in graphs and tables, the frequencies are scaled linearly for all components in your system, not just the GPU. In a real system the relationship between components is not linear, so you must therefore use scaled values with caution.

The frequency that was used to create a performance data file is stored within the file itself. This is the default frequency that is displayed in the **Frequency** pane when you first load the performance data file.

4.2 Performance data files and counters

The **Performance counters** pane displays groups of performance counters that are contained within performance data files. The software and hardware performance counters contained within the files are shown in the form of a directory structure.

To view additional information about a performance counter, place the cursor over the counter, the extra information is shown in a tooltip. The same information can be viewed by dragging the counter into an Info display pane.

Use the instrumented drivers to record performance counters. You can enable or disable performance data collection, and configure how performance data is collected. See Activating performance data collection on page 10-5 for more information about controlling how the instrumented drivers collect performance data.

Figure 4-1 shows an example **Performance counters** pane from the Performance Analysis Tool main window.

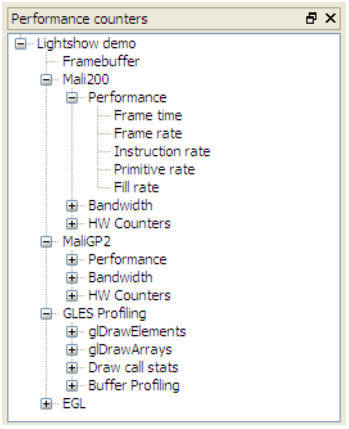

Figure 4-1 Example performance counters pane

Note

The Mali GPU counters are disabled in the Emulator dump files. These counters are Mali Hardware specific and only can be enabled and viewed when the Performance Analysis Tool is connected to a Mali GPU or when a performance data file has been previously dumped by a Performance Analysis Tool connected to a Mali GPU with instrumented drivers.

In this example, the **Performance counters** pane contains the following information:

<Planet demo>

This is the name of the `.ds2` performance data file. The Performance Analysis Tool can work with more than one performance data file.

The ability to work with more than one performance data file makes it possible to compare values from, for example, different hardware configurations. If you load more than one `.ds2` file, each file is displayed on a different tab within the **Performance counters** pane. See *Playing performance data files* on page 5-3 for more information about loading and playing performance data files.

Counters The performance counters shown are:

- **glDrawElements**
- **glDrawArrays**
- **Draw call stats**
- **Geometry statistics**
- **EGL timing**.

The counters are displayed in graphical and tabulated format.

Mali200 Performance

This contains performance variables that define the overall performance of the hardware. You can display the variables in a graph or table.

A value is calculated for each performance variable for each frame. The name of the variable identifies a stream of values, known as a time series. For example:

- Frame time. The time, in milliseconds, required to produce one frame, that is one framebuffer image, in the animated display.
- Frames per second. For frames that are equally complex, this is the number of frames generated each second.

See *Customizing tables* on page 6-5 for more information about displaying and customizing tables. See *Customizing graphs* on page 6-3 for more information about displaying and customizing graphs.

Mali200 Bandwidth

This contains variables that define the data traffic between the Mali-200 pixel processor and the main system memory. The data traffic is measured in units that change dynamically as the values change.

MaliGP2 Performance

This has counters similar to **Mali200 Performance**, however these are for the MaliGP2 geometry processor.

Mali200 and MaliGP2 Counters

This contains values obtained directly from the GPU hardware performance counter registers. The counters indicate the number of times that a particular event has occurred, such as bus reads or clock cycles.

Other counters

There are counters that are specific to OpenGL ES or OpenVG, such as the number of Draw calls.

Note

For all **Bandwidth** and **Performance** counters, unit prefixes might change depending on traffic volume. For example, a sudden reduction in data traffic might cause a measure of MB per second to change to KB per second.

Table 4-1 lists key Mali-200 GPU performance counters and provides descriptions and typical values.

Table 4-1 Mali-200 GPU performance counters

Counter	Description	Units	Typical value range
Frame time	Execution time required for one frame, assuming Mali is running at 200MHz.	ms	1.3-12
Frame rate	Number of frames produced per second of execution time, assuming Mali is running at 200MHz.	fps	0-750
Primitives read rate	Number of primitives read per second. The primitives are usually triangles. If a triangle touches more than one tile, it is read once for each tile it touches.	Thousand triangles per second (ktri/s)	0-2000
Fill rate	Number of pixels processed per second.	Megapixel/s	2-100

Note

Turning on multi-sampling does not affect the fill rate, turning on super-sampling does.

4.3 Framebuffer output

You can add a framebuffer output pane to the Performance Analysis Tool, to display the current contents of the framebuffer. You can also create custom graphs and tables that display real-time information relating to the scene displayed in the framebuffer. See *Graphs* on page 4-8 and *Tables* on page 4-9 for more information.

Figure 4-2 shows an example framebuffer output pane.

Figure 4-2 Example framebuffer output

A context menu appears when you right click on the framebuffer display. This enables you to zoom in and out of the of the image, display it at the original size, and fit the image to the pane. You can also zoom in and out of the display using the mouse wheel while the mouse pointer is over the framebuffer display.

See *Using the Performance Analysis Tool in offline mode* on page 5-3 for more information about the framebuffer output pane.

4.4 The Frequency pane

The **Frequency** pane displays the frequency of the system bus clock. Initially, these are set to the actual frequency in effect when the data was recorded. In many cases, performance data recording is performed on an FPGA prototyping system, running at a lower frequency than the main system.

Use the **Frequency** pane to set the frequency to that of your final product. This scales the performance variables according to your particular requirements. Figure 4-3 shows the **Frequency** pane.

Note

When you change the frequency from the default one used to record the performance data file, the Performance Analysis Tool uses linear interpolation to obtain the simulated results. In practice, the relationship between different hardware frequencies and associated performance results might not be linear. This feature is intended to provide approximate results only.

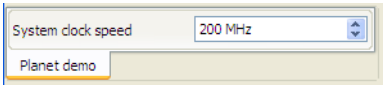

Figure 4-3 Frequency pane

4.5 Graphs

The Performance Analysis Tool enables you to monitor the performance of particular areas of the GPU, using graphs. Graphs and corresponding legends have the same color. The graph scrolls to the left to show new values for new frames. You can configure graphs to display the type of information you require. For example, you can create graphs that show the frame rate, measured in frames per second.

You can also display multiple performance data files simultaneously. For example, to view and compare the effects of different memory latencies, you can load one performance data file recorded on hardware with low latency, and another recorded on hardware with high latency. You can display performance counters from each file in a single graph pane, or you can use two separate panes.

Figure 4-4 shows an example frame rate graph, with minimum and maximum frame rates shown. The horizontal axis represents the frame sequence number. The value displayed is the value at the cursor location, or if no cursor is displayed, the value at the location of the vertical marker.

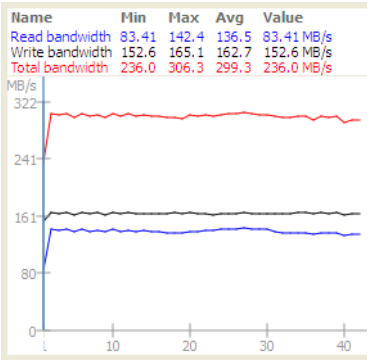

Figure 4-4 Example graph

See *Customizing graphs* on page 6-3 for more information about displaying and customizing graphs.

4.6 Tables

You can display performance counters using tables. As you play a file, the **Value** column changes to reflect the values for the current frame.

As with graphs, you can load multiple performance data files and display the performance counters side by side in tables for comparison. You can name tables to suit your requirements.

Figure 4-5 shows an example table containing a selection of performance counters from a performance data file.

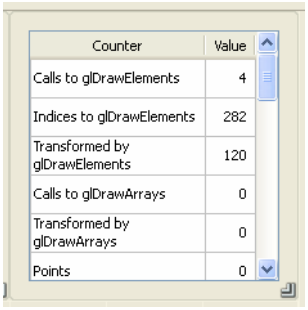

Figure 4-5 Example table

See *Customizing tables* on page 6-5 for more information about displaying and customizing tables.

Chapter 5
Using the Performance Analysis Tool

This chapter describes how you use the Performance Analysis Tool to view and analyze performance data obtained by running a graphics application on a Mali GPU or an instrumented version of one of the OpenGL ES 1.1 or OpenGL ES 2.0 Emulators. It contains the following sections:

5.1 Starting the Performance Analysis Tool

This section describes how to start the Performance Analysis Tool. It contains the following sections:

5.1.1 Starting the Performance Analysis Tool from Microsoft Windows

To start the Performance Analysis Tool from Microsoft Windows, select:

All Programs → ARM → Mali GPU Developer Tools → Performance Analysis Tool v2.2

5.1.2 Starting the Performance Analysis Tool from Linux

To start the Performance Analysis Tool from Linux:

1. Open a command terminal.
2. Navigate to:
   `ARM/Mali_Developer_Tools/Mali_Performance_Analysis_Tool_2.2/bin`
3. Set the `LD_LIBRARY_PATH` to define the location of the QT shared object files.
   If you are using a Bourne shell `sh`, use the command:
   ```
   export LD_LIBRARY_PATH=`pwd`:`pwd`/imageformats:$LD_LIBRARY_PATH
   ```
   If you are using a C shell `csh`, use the command:
   ```
   stenv LD_LIBRARY_PATH `pwd`:`pwd`/imageformats${LD_LIBRARY_PATH}
   ```
4. Type the following command to run the executable:
   ```
   ./pat
   ```

5.2 Using the Performance Analysis Tool in offline mode

This section describes:

- *Creating performance data files*
- *Playing performance data files*
- *Creating project files* on page 5-4
- *Using the player controls* on page 5-5
- *Unloading a performance data file* on page 5-6.

5.2.1 Creating performance data files

In offline mode Performance Analysis Tool uses performance data files. You can create performance data files, for analysis by running the GLES applications on the instrumented version of the OpenGL ES 2.0 Emulator. See *Mali GPU OpenGL ES Emulators User Guide* for more information.

Two performance dump data files, `lightshow_multicore.ds2` and `dump.ds2`, are included with the Performance Analysis Tool.

5.2.2 Playing performance data files

This section describes how you can view the contents of the framebuffer by playing a performance data file. In addition, you can customize the window by adding graphs and tables. See Chapter 6 *Customizing the Performance Analysis Tool* for more information about adding graphs and tables.

You can use the example performance data files included with the Performance Analysis Tool, or you can create your own files. See *Creating performance data files* for more information.

To play a performance data file and view the framebuffer contents of the file:

1. Select **Import Performance Data File** from the **File** menu, or type **CTRL+D**.
2. Navigate to the `.ds2` file that you want to open and click **Open**. The data appears in the **Performance counters** pane.

   Note

   You can also open a performance data file by dragging and dropping it into the **Performance counters** pane.

3. Select **Add Framebuffer** from the **View** menu, or type **CTRL+F** to add a framebuffer pane to the Performance Analysis Tool window.

4. Drag and drop the **Framebuffer** entry from the **Performance counters** pane to the framebuffer pane to add content to the pane, as Figure 5-1 shows.

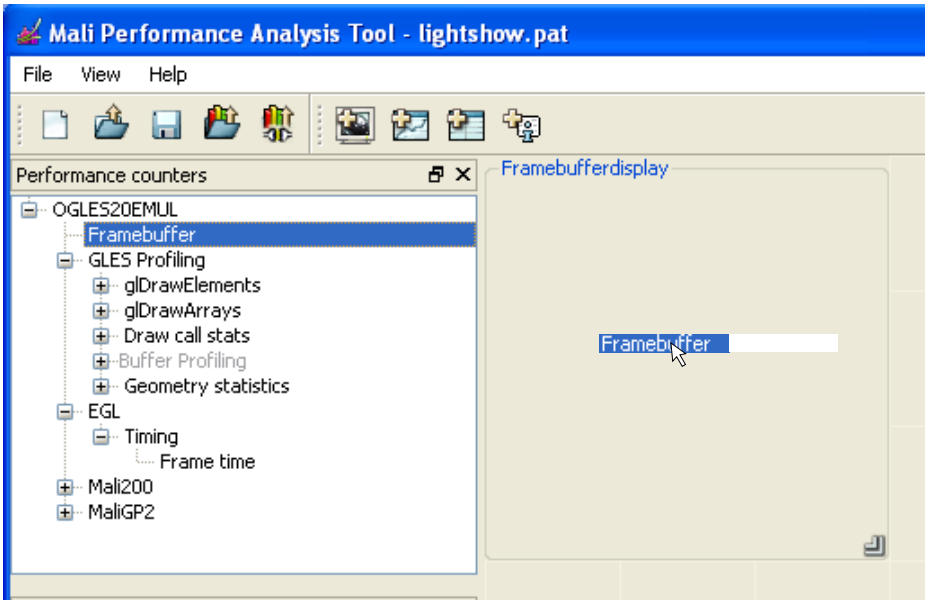

Figure 5-1 Adding content to the framebuffer pane

5. Click the **Play** button on the player to play the file. The contents of the framebuffer appear in the framebuffer pane.

5.2.3 Creating project files

At any time, you can save the current layout of the Performance Analysis Tool window as a `.pat` project file. This enables you to keep track of the performance counters that are displayed in specific tables and graphs when the project is opened. If there are several sets of performance data belonging to the same project then they are located in sub folders of the root directory holding the `.pat` file.

To create a new project file:

1. Create your required layout. See Chapter 6 *Customizing the Performance Analysis Tool* for more information about creating a custom layout.

2. Either select **Save project** from the **File** menu, type **CTRL+S**, or click on the **Save project** button.

3. Save the project as a `.pat` file in the root directory of the performance data files you are using. See *Creating performance data files* on page 5-3 for more information.

   For real-time performance data sources, the `.pat` file keeps track of the layout of panes and the Internet Protocol Address of the target platform. However, the Performance Analysis Tool does not write a copy of the performance data or frame buffers.

To open a previously-saved project:

1. Select **Open project** from the **File** menu, or type **CTRL+O**.
2. Navigate to the `.pat` file that you want to open and click **Open**.

Note

You can also open a saved project by dragging and dropping the `.pat` file into the **Performance counters** pane.

The project opens, displaying all previously-created graphs, tables, and performance counters.

5.2.4 Using the player controls

Use the **Play** and **Stop** buttons in the **Player** pane, as required, to play performance data files. To view counter values for an individual frame, or several localized frames, you can use the **Pause** button to pause playback and then use the **single step** buttons, marked with double arrows, to step through a sequence of frames. The sequence number of the current frame is also displayed.

Figure 5-2 shows the player controls.

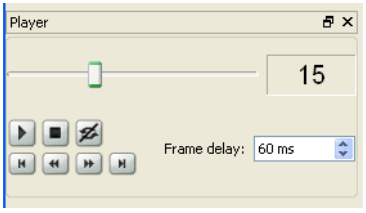

Figure 5-2 Player controls

The **Frame delay** box enables you to increase or decrease playback speed by adjusting the delay between successive frames. You can use the **Frame delay** up and down arrows to adjust the frame delay. Alternatively, type over the displayed value in the **Frame delay** box, then press **Enter**.

You can adjust the delay between 10ms and 300ms.

Note

If you are viewing a project with large framebuffers and many graphs, with a short frame delay specified, the actual delay might be greater than the one you specify.

5.2.5 Unloading a performance data file

To unload a performance data file from the Performance Analysis Tool window, right-click on the name of the file that you want to remove, and select **Unload performance counters**, as Figure 5-3 shows.

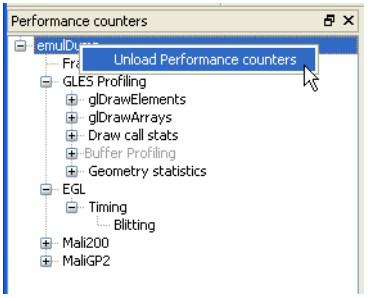

Figure 5-3 Unloading a performance data file

When you remove a performance data file, the information is not removed from the project until you save the project. If you close the project without saving it, the reference to the performance data file remains in the project.

Note

Unloading a performance data file only removes it from the project view. It does not delete the actual performance data file.

5.3 Using the Performance Analysis Tool in online mode

In online mode, the Performance Analysis Tool uses a network connection to display the performance data of a graphics application running on a development platform or other target computer. You can also read and display the data of one or more `.ds2` dump files. These dump files must have been written by earlier runs of the instrumented driver.

Note

To use the Performance Analysis Tool in online mode, you must have a system that includes Mali hardware and a standard network connection. In the early stages of development, this usually involves working with an ARM development board, with the Mali GPU implemented as either *Field-Programmable Gate Array* (FPGA) or as an *Application-Specific Integrated Circuit* (ASIC) test chip.

The host platform, typically a desktop computer running Windows or Linux, runs the Performance Analysis Tool application. The Performance Analysis Tool monitors your target platform continuously, and connects to the instrumented drivers when they start.

The target platform contains the GPU, and runs the graphics application linked with the Instrumented graphics drivers.

Communication between the target platform and the host is established using TCP/IP over an Ethernet connection.

Figure 5-4 on page 5-8 shows how you can use the Performance Analysis Tool to view performance data files in real time, and *Connecting to the target platform* on page 5-8 describes how to connect the target platform to your desktop computer.

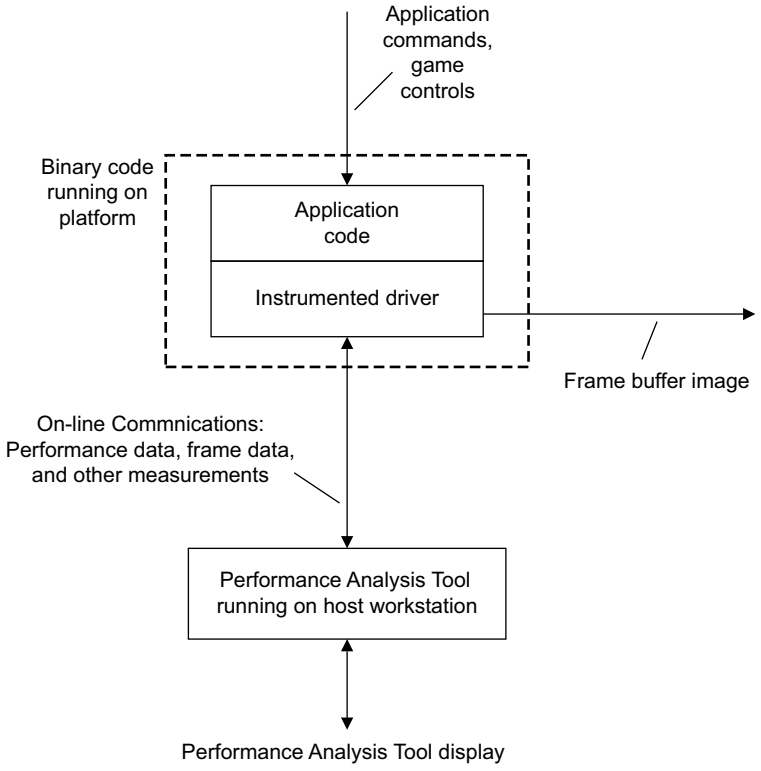

Figure 5-4 Viewing performance data in real time

5.3.1 Connecting to the target platform

Before you connect to the target platform, the GPU, the instrumented drivers, and the graphics application must be running.

Follow these steps to start real-time analysis:

1. Start the application with the appropriate instrumented driver.
2. In the Performance Analysis Tool, select **File** → **Connect to target**, or type **CTRL+SHIFT+C**, to connect to the target platform. If connection is successful, you are advised of this, and a copy of the image your application is analyzing appears in the Performance Analysis Tool framebuffer pane. The **Connect to Target** screen is also displayed.
3. From the list of network resources, select the name of a particular resource you want to connect to, and click **OK**.

4. You can add custom graphs and tables to the Performance Analysis Tool to view the parameters you want to observe, as described in Chapter 6 *Customizing the Performance Analysis Tool*.
5. You can now view the parameter values streaming in from the target system.
6. To view or save the performance data file that is created, access it in the location you specified when you set the name of the file, when you configured the instrumented drivers.

For information about the Instrumented drivers see the documentation for your platform.

Chapter 6
Customizing the Performance Analysis Tool

This chapter describes how to customize the Performance Analysis Tool. It contains the following sections:

6.1 About customizing the Performance Analysis Tool

You can customize the Performance Analysis Tool to display the performance information that you require.

You can add graphs, tables, and framebuffer output panes. For example, you can display several graphs, with each one displaying the output of several performance counters.

The procedures described in this section are examples only. You can use your own names for panes, and include the performance counters that you require.

You can use the Performance Analysis Tool to create project files that show one or more performance data files, the graphs and tables that you have defined, and the performance variables displayed in them. You can save a project as a `.pat` file, enabling you to store the layout you have created, and reload it at any time.

To save the project, you can either select **Save project** from the **File** menu, type **CTRL+S**, or click on the **Save Project** button.

Note

You can only add table, graph or info panes if there is space available within the screen area of the Performance Analysis Tool window. If there is not enough screen space an error message is generated.

6.2 Customizing graphs

This section describes how to customize the graphs you use to display performance data. You can add multiple graphs, and use them to display information from the performance counters contained in the **Performance counters** pane. It contains the following sections:

- *Adding a custom graph*
- *Viewing performance data in graphs* on page 6-4
- *Displaying a legend on a graph* on page 6-4
- *Changing the color of graph data* on page 6-4
- *Removing graphs* on page 6-4.

6.2.1 Adding a custom graph

To add a custom graph to the Performance Analysis Tool window, follow these steps:

1. You can select **Add Graph** from the **View** menu, type **CTRL+G**, or click on the **Add Graph** button. The new **Graph** pane is displayed.

2. To set a name for a graph pane, right-click anywhere in the graph, except on the graph data plot, and select **Set Title**, as Figure 6-1 shows. The **New Title** dialog box is displayed.

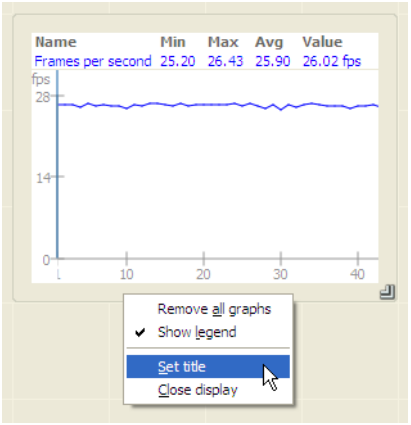

Figure 6-1 Setting a graph title

3. Type a new title for the graph and click **OK**.

4. To add performance counter information to the graph, drag and drop performance counters from the **Performance counters** pane, into the graph pane, as required.

6.2.2 Viewing performance data in graphs

When you pause or stop the playback of a performance data file, you can view the values associated with any point on the graph output. Follow these steps:

1. Play a performance data file. See *Playing performance data files* on page 5-3.
2. Press **Pause** or **Stop** on the player controls to stop playback.
3. Move the cursor anywhere on the graph to display values for that frame position. Figure 6-2 shows an example of this feature.

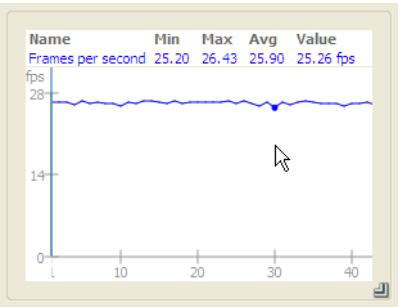

Figure 6-2 Viewing instantaneous values

6.2.3 Displaying a legend on a graph

You can display a legend on a graph, showing the highest and lowest values for the range of data contained in the performance data file. The values are displayed at the top of the graph, as all the graphs in this section show.

Right-click anywhere on the graph and select **Show legend** to display or remove the graph legend.

6.2.4 Changing the color of graph data

To change the color of data on a graph, right click on a data plot and select **Set graph color**. Select a new color from the color palette and click **OK**.

6.2.5 Removing graphs

To remove all graphs from your display, right-click on the title of any graph, and select **Remove all graphs**.

To remove a single data plot from a graph, right-click on the plot and select **Remove Graph**.

6.3 Customizing tables

This section describes how to customize the tables you use to inspect performance data files. You can add multiple tables, and use them to display information from the performance counters contained in the **Performance counters** pane.

You can also drag and drop, or copy and paste, selected rows from a table to other programs, similar to the functions found in spreadsheet programs.

6.3.1 Adding a custom table

To add a custom table to the Performance Analysis Tool window, follow these steps:

1. You can select **Add Table** from the **View** menu, type **CTRL+T** or click on the **Add Table** button. The new graph is displayed.
2. Right-click anywhere in the table and select **Set Title** to add a name to the table, as Figure 6-3 shows. The **New Title** dialog box is displayed.

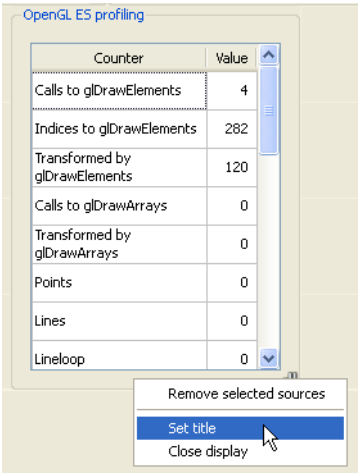

Figure 6-3 Setting a table title

3. Type a new title for the table and click **OK**.
4. Drag and drop performance counters from the **Performance counters** pane into the table pane to add performance counter information to the table.

6.4 The Info display pane

You can add an Info display pane to the Performance Analysis Tool window, enabling you to view information about performance counters, performance data files, or a framebuffer. For example, you can use the Information pane to display hardware names and descriptions.

You can add information to the Information pane, by dragging and dropping performance counters from the **Performance counters** pane.

To edit the contents of the Information pane, double-click inside the pane.

Note

Editing the contents of the Info display only modifies the text in the Info display, it does not modify descriptions anywhere else in the application.

Figure 6-4 shows an example Information pane.

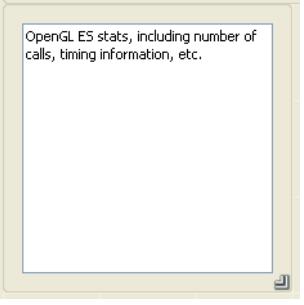

Figure 6-4 Example information pane

Glossary

This glossary describes the terms used in *Media Processing Division* (MPD) documents from ARM Limited.

AEL *See* ARM Embedded Linux.

ARM Embedded Linux (AEL)
A version of embedded Linux OS ported to the ARM architecture.

DCC *See* Digital Content Creation.

DDK *See* Mali DDK.

Digital Content Creation (DCC)
The creation and modification of custom content such as animations, graphics, textures, and images, before presentation in the final medium.

EGL *See* Embedded-System Graphics Library.

EGL driver *See* Native platform graphics interface.

Embedded-System Graphics Library (EGL)
A standardized set of functions that communicate between graphics software, such as OpenGL ES or OpenVG drivers, and the platform-specific windowing system that displays the image.

ESSL *See* OpenGL ES Shading Language.

ESSL compiler The compiler that translates shaders written in ESSL, into binary code for the shader units in the Mali GPU. There are two versions of the ESSL compiler:

- the on-target compiler
- the offline compiler.

Fixed-function pipeline
A process that uses standard functions to draw graphics on fixed-function graphics hardware, such as the Mali-55 GPU. The fixed-function pipeline is a requirement of OpenGL ES 1.1.

Fragment A fragment consists of all data, such as depth, stencil, texture, and color information, required to generate a pixel in the framebuffer. A pixel is usually composed of several fragments. A fragment can be multi-sampled and super-sampled.

Fragment processor A programmable component of the Mali GPU pixel processor, that runs fragment shaders.

Fragment shader A program running on the Mali GPU pixel processor that calculates the color and other characteristics of each fragment.

Geometry processor
A geometry processor is a component of a programmable-function Mali GPU. It executes vertex shaders with typical transform and lightning calculations, and generates lists of primitives for a pixel processor to draw.

GPU *See* Graphics Processor Unit.

Graphics application
A custom program that executes in the Mali graphics system and displays graphics content in a framebuffer for transfer to a display.

Graphics driver A software library implementing OpenGL ES or OpenVG, using graphics accelerator hardware.

See also OpenGL ES driver and OpenVG driver.

Graphics Processor Unit (GPU)
A hardware accelerator for graphics systems using OpenGL ES and OpenVG. The hardware accelerator comprises of an optional geometry processor and a pixel processor together with memory management. Mali programmable GPUs, such as the Mali-200 and Mali-400 MP GPUs, consist of a geometry processor and at least one pixel processor. Mali fixed-function GPUs, such as the Mali-55 GPU consist of a pixel processor only.

See also Geometry Processor, Pixel Processor, Mali MMU.

Instrumented drivers

Alternative graphics drivers that are used with the Mali GPU. The Instrumented drivers include additional functionality such as error logging and recording performance data files for use by the Performance Analysis Tool.

See also Mali DDK, Performance Analysis Tool.

Mali The name given to graphics software and hardware products developed by ARM, that accelerates 2D and 3D graphic applications.

Mali Binary Asset The proprietary file format that you can load onto the Mali Demo Engine and use on Mali GPUs. Mali Binary Asset is faster to parse, uses less memory, and is smaller in size than formats such as COLLADA XML. If a deployment format is not available, use the Mali Binary Assets Exporter to convert graphics assets into Mali Binary Asset.

Mali DDK A set of drivers, typically for AEL, that enable communication between AEL and the Mali GPU. These drivers are available as normal or instrumented versions.

See also Instrumented drivers.

Mali Demo Engine

The Mali Demo Engine is a component of the Mali Developer Tools. The Mali Demo Engine library enables you to develop 3D graphics applications more easily than using OpenGL ES alone.

Mali Demo Engine Library

A C++ class framework for developing OpenGL ES 2.0 applications for the Mali GPU.

Mali MMU A full-featured *Memory Management Unit* (MMU) that is present on Mali GPUs such as the Mali-200 GPU and the Mali-400 MP GPU.

Mali Remote Interface

The Mali Remote Interface is a component of the Instrumented driver. This interface enables the Performance Analysis Tool tool to access performance data dump files over a network interface.

See also Instrumented drivers, Performance data file.

Mali Surface The destination for Mali GPU output. It can potentially be for color, depth and stencil, however, when referring to EGL surfaces, it only refers to color output buffers.

Multi-sampling An anti-aliasing technique where each pixel in the framebuffer is split into multiple samples corresponding to different positions within the area covered by the pixel. The Mali GPU pixel processors support multi-sampling at four samples per pixel with negligible performance impact.

Native platform graphics interface (EGL) driver
A standardized set of functions that communicate between graphics software, such as OpenGL ES or OpenVG drivers, and the platform-specific window system that displays the image.

Offline Compiler A command line tool that translates vertex shaders and fragment shaders written in the ESSL into binary vertex shaders and binary fragment shaders that you can link and run on the Mali GPU.

On-target compiler A component of the Mali GPU OpenGL ES 2.0 driver that translates shader source files provided by the graphics application, into binary shader code, at runtime.

OpenGL ES driver The OpenGL ES driver is part of a driver stack that translates OpenGL ES API commands into data and instructions for the Mali GPU. Only the device driver controls the Mali GPU directly.

OpenGL ES Shading Language (ESSL)
A programming language used to create custom shader programs that can be used within a programmable pipeline, on the Mali GPU. You can also use pre-defined library shaders, written in ESSL.

OpenVG driver The OpenVG driver is part of a driver stack that translates OpenVG API commands into data and instructions for the Mali GPU. Only the device driver controls the Mali GPU directly.

Performance Analysis Tool
A fully-customizable GUI tool that displays and analyzes performance data files produced by the Instrumented drivers, together with framebuffer information.

See also Instrumented drivers, Performance data file.

Performance counter
Data produced by the Instrumented drivers and the Mali GPU, that can be displayed and analyzed as statistical information in the Performance Analysis Tool.

Performance data file
Files that contain a description of the performance counters, together with the performance counter data in the form of a series of values and images. Performance data files are saved in `.ds2` format and can be loaded directly into the Performance Analysis Tool.

Performance variable
Data produced by the Instrumented drivers and the Mali GPU, that can be displayed and analyzed as statistical information in the Performance Analysis Tool.

Pixel processor A pixel processor, such as the Mali-200 GPU pixel processor, performs rendering operations to produce a final image for display. The Mali-200 and Mali-400 MP GPU pixel processors receive completed vertex data from the Mali GPU geometry processor.

Primitive A basic element that is used, by the Mali GPU, with other primitives, to generate images. A primitive can be a point, a line, a triangle, or a quad. Properties of primitives are defined through the use of vertices. Each primitive is divided into fragments so that there is one fragment for each pixel covered by the primitive.

See also Vertex.

Programmable pipeline
A process that uses custom programs to draw graphics on programmable graphics hardware, such as the Mali-200 and Mali-400 MP GPUs. The programmable pipeline is a requirement of OpenGL ES 2.0.

Rasterization The process within the Mali GPU pixel processor that identifies the fragment of each triangle that is seen through each pixel on the display screen.

Rasterizer A unit or method to convert a geometrical description of primitives supported by the Mali GPU: point, line, triangle, or quad, to fragments. The rasterizer works on line equations generated in the triangle setup phase of Mali GPU pixel processors.

Render Rendering in the context of the Shader Debugger plug-in refers to the sending of shader effects to a remote Renderer, so that a preview image can be generated.

Renderer A device capable of rendering shader effects and returning an image of the rendered shader. Typically, this is a device with a Mali GPU, though it can also be a local or remote emulation.

Render Target *See* Mali Surface.

Shader A Shader or Shader Source file, refers to a single source file for a shader effect. This source file might be a vertex shader, a fragment shader or a non-specific utility shader that can be linked concatenated and compiled with other shaders.

See also Fragment shader, Vertex shader

Texture Compression Tool
A component of the Mali Developer Tools that you can use to compress textures and images, using the ETC algorithm.

Vertex A set of data defining the properties of one point of a primitive. For example, a point primitive, an endpoint of a line primitive, or a corner of a triangle primitive.

Vertex shader Program vertex shader data sent to the Mali GPU geometry processor. The geometry processor calculates the position and other characteristics, such as color and texture coordinates, for each vertex.

Vertex shader unit A programmable component of the Mali GPU geometry processor, that runs vertex shaders.

Vertices This is the plural form of the word vertex.

See also Vertex.