

Mali™ GPU Asset Conditioning Tool

Version: 1.0

User Guide

Non-Confidential - Draft - Beta



Mali GPU Asset Conditioning Tool

User Guide

Copyright © 2010 ARM. All rights reserved.

Release Information

The following changes have been made to this book.

Change history			
Date	Issue	Confidentiality	Change
4 January 2010	A-01a	Confidential - Draft	First draft for version 1.0
10 June 2010	A-02a	Non-Confidential - Draft	Second draft for version 1.0 - Beta

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Where the term ARM is used it means “ARM or any of its subsidiaries as appropriate”.

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.

Product Status

The information in this document is for a Beta product, that is a product under development.

Web Address

<http://www.arm.com>

Contents

Mali GPU Asset Conditioning Tool User Guide

- Preface**
 - About this book vii
 - Feedback ix

- Chapter 1**
 - Introduction**
 - 1.1 About the Asset Conditioning Tool 1-2
 - 1.2 Content conversion 1-3

- Chapter 2**
 - Installing the Asset Conditioning Tool**
 - 2.1 About the Asset Conditioning Tool installation bundle 2-2
 - 2.2 Installing the Asset Conditioning Tool on Microsoft Windows 2-3

- Chapter 3**
 - Using the Asset Conditioning Tool**
 - 3.1 Command line options 3-2
 - 3.2 Viewing converted files with third-party COLLADA viewers 3-6

- Glossary**

List of Tables

Mali GPU Asset Conditioning Tool User Guide

	Change history	ii
Table 3-1	Commands	3-2
Table 3-2	Conditioning steps	3-3
Table 3-3	Target names	3-4
Table 3-4	Conditioning step options	3-4

List of Figures

Mali GPU Asset Conditioning Tool User Guide

Figure 1-1	Content creation and conversion flowchart	1-2
Figure 1-2	Conversion of polygon to triangles	1-4
Figure 1-3	Conversion of a tristrip to triangles	1-4
Figure 1-4	Conversion of a trifan to triangles	1-5
Figure 1-5	Vertex reorganization to improve cache performance	1-6

Preface

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

- *About this book* on page vii
- *Feedback* on page ix.

About this book

This is the *Mali GPU Asset Conditioning Tool User Guide*. It provides guidelines for using the Mali GPU Asset Conversion Tool to optimize digital content for use on a system containing a Mali GPU.

Intended audience

This guide is written for system integrators and software developers who are writing or using digital content in the COLLADA file format.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the *Asset Conditioning Tool*.

Chapter 2 *Installing the Asset Conditioning Tool*

Read this chapter for information on how to install the Asset Conditioning Tool.

Chapter 3 *Using the Asset Conditioning Tool*

Read this for information on how to use the Asset Conditioning Tool.

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

Conventions

The typographical conventions are:

<i>italic</i>	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.
<u>monospace</u>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<i>monospace italic</i>	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 guide contains information that is specific to the Mali GPU Developer Tools. See the following documents for other relevant information:

- *Mali GPU Asset Conditioning Tool Release Notes*
- *Mali GPU Binary Asset Exporter User Guide* (ARM DUI 0507)
- *Mali GPU Developer Tools Technical Overview* (ARM DUI 501)
- *Mali GPU OpenGL ES Application Developer Guide* (ARM DUI 0363)
- *Mali GPU OpenVG Application Developer Guide* (ARM DUI 0380)
- *Mali GPU Offline Shader Compiler User Guide* (ARM DUI 0513)
- *Mali GPU Performance Analysis Tool User Guide* (ARM DUI 0502)
- *Mali GPU Shader Development Studio User Guide* (ARM DUI 0504)
- *Mali GPU Shader Library User Guide* (ARM DUI 0510)
- *Mali GPU Texture Compression Tool User Guide* (ARM DUI 0503)
- *Mali GPU User Interface Engine User Guide* (ARM DUI 0505)
- *OpenGL ES 1.1 Emulator User Guide* (ARM DUI 0506)
- *OpenGL ES 2.0 Emulator User Guide* (ARM DUI 0511).

Other publications

This section lists relevant documents published by third parties:

- *COLLADA 1.4.1 Specification* at <http://collada.org>
- *OpenGL ES 1.1 Specification* at <http://www.khronos.org>
- *OpenGL ES 2.0 Specification* at <http://www.khronos.org>.

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, ARM DUI 0528A-02a
- 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 provides information about the Mali Asset Conditioning Tool, and describes how to start using it in your workflow. It contains the following sections:

- *About the Asset Conditioning Tool* on page 1-2
- *Content conversion* on page 1-3.

1.1 About the Asset Conditioning Tool

The Asset Conditioning Tool optimizes content to improve its performance on the target hardware. The optimization stages are:

- transforming content from unsupported primitives in the input file to supported primitives in the output file
- compressing coordinate data
- optimizing mesh data to improve the rendering order and cache utilization.

Both the input and output files use the COLLADA interchange format. For more information on this format see the Khronos standard, www.khronos.org.

The Asset Conditioning Tool uses a command-line interface. This simplifies integrating the tool into your build system.

Figure 1-1 shows how the Asset Conditioning Tool creates a new content file that can be either:

- re-edited by a third-party content creation tool such as Blender
- passed to an application-specific loader and converter.

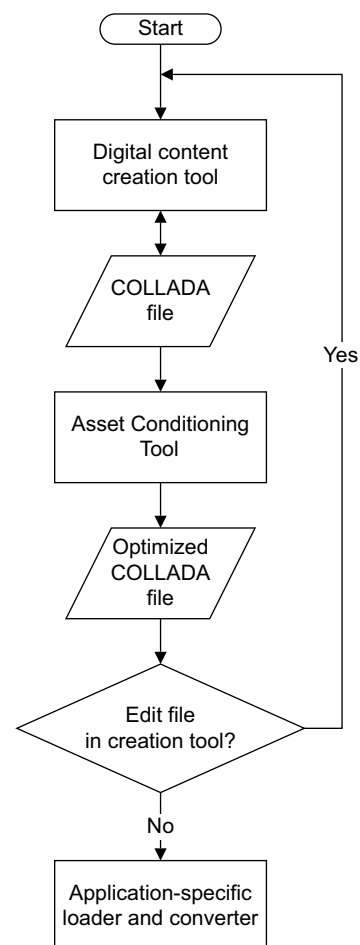


Figure 1-1 Content creation and conversion flowchart

1.2 Content conversion

The Asset Conditioning Tool performs the following optimizations:

- *Data optimization*
- *Primitive conversion*
- *Mesh optimization* on page 1-6.

1.2.1 Data optimization

Compression consists of using 16-bit or 8-bit numbers instead of floating-point numbers to identify coordinates.

A simplified version of the algorithm is:

1. Normalize all floating-point numbers for the coordinates to the range 0.0f to 1.0f.
2. Multiply the coordinates by 65535 (or 255 for 8-bit compression) and round the result to the nearest integer.
3. Offset the range by subtracting 32768 (or 128 for 8-bit compression).
4. The new coordinate specifiers are in the range -32768 to 32767 (or -128 to 127) and are therefore compatible with OpenGL ES.

Note

- By default, 16-bit compression is performed on all meshes. 8-bit compression is only performed if 16-bit compression is excluded.
 - Then a warning is output stating that lossy compression was performed and how many new duplicates were created.
 - If compression is successful, an inverse transformation matrix must be added to the node that holds this geometry. This transformation matrix can then be applied to transform the geometry to the state it was in before compression.
-

1.2.2 Primitive conversion

Primitive conversion is replacing a primitive shape with triangles. The following sections describe the conversions:

- *Conversion of polygons to triangles*
- *Conversion of tristrips and trifans to triangles* on page 1-4
- *Conversion of trifans to triangles* on page 1-5.

Conversion of polygons to triangles

Polygon conversion consists of:

1. identifying all <polygons> in the mesh and converting them into <triangles>.
2. identifying all <polylists> in the mesh and:
 - a. splitting each of the lists into individual <polygons>
 - b. converting each of the individual polygons into <triangles>.
3. replacing the <polygons> and <polylist> elements by <triangles> elements.

Figure 1-2 on page 1-4 shows conversion of a polygon to triangles:

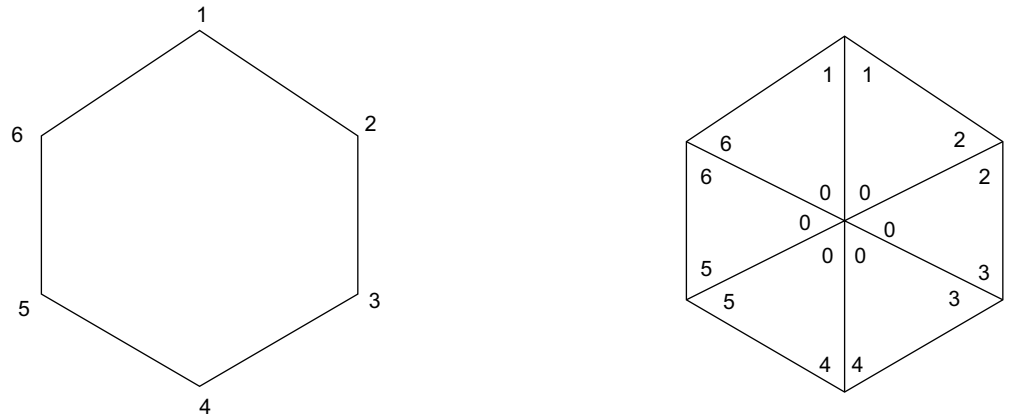


Figure 1-2 Conversion of polygon to triangles

Conversion of tristrips and trifans to triangles

To optimize cache performance, <tristrips> are converted to <triangles>.

A <tristrips> element is a list of triangles where the next triangle shares the last vertex of the current triangle. Figure 1-3 shows conversion of a tristrip to triangles:

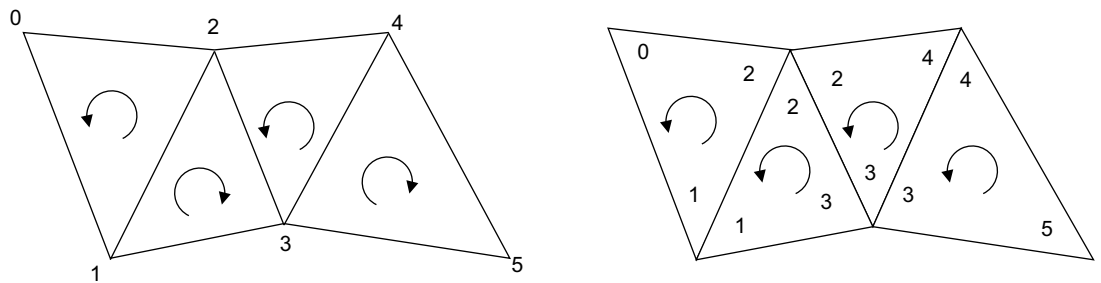


Figure 1-3 Conversion of a tristrip to triangles

Example 1-1 shows the coding for the original tristrip. The indices alternate between vertex indices (in *italics*) and normal indices:

Example 1-1 Tristrip with four triangles

```
<tristrips count="1" material="#Bricks">
  <input semantic="VERTEX" source="#verts" offset="0"/>
  <input semantic="NORMAL" source="#normals" offset="1"/>
  <p> 0 0 1 3 2 1 3 2 4 5 5 4 </p>
</tristrips>
```

The original <tristrip> is converted to the <triangles> element shown in Example 1-2:

Example 1-2 Conversion of tristrip to a <triangles> element

```
<triangles count="4" material="#Bricks">
  <input semantic="VERTEX" source="#verts" offset="0"/>
  <input semantic="NORMAL" source="#normals" offset="1"/>
  <p> 0 0 1 3 2 1
```

```

      2 1 1 2 3 3
      2 1 3 2 4 5
      4 5 3 4 5 2
    </p>
  </triangles>

```

Note

Because all triangles in a triangle list must be counter-clockwise, every even triangle in the tristrip is reversed to become a counter-clockwise triangle.

Conversion of trifans to triangles

Figure 1-4 shows conversion of a trifan to triangles:

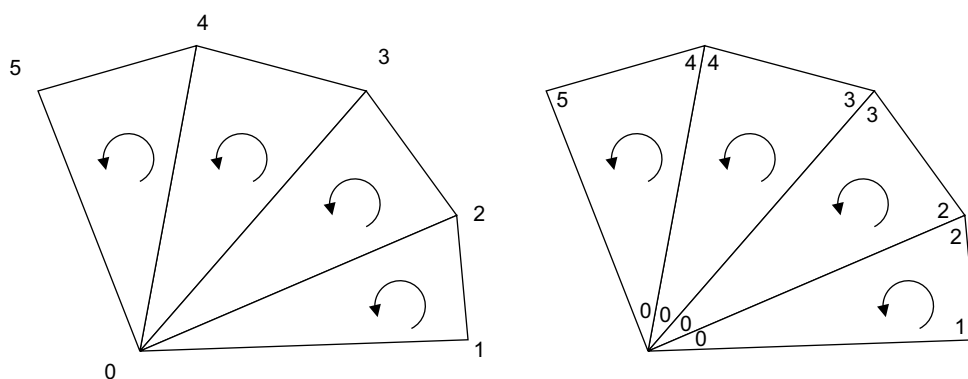


Figure 1-4 Conversion of a trifan to triangles

Example 1-3 shows the coding for the original trifan. The indices alternate between vertex indices (in italics) and normal indices:

Example 1-3 Trifan with four triangles

```

<trifans count="1" material="#Bricks">
  <input semantic="VERTEX" source="#verts" offset="0"/>
  <input semantic="NORMAL" source="#normal" offset="1"/>
  <p> 0 0 1 3 2 1 3 2 4 5 5 4 </p>
</trifans>

```

The original `<trifans>` is converted to the `<triangles>` element shown in Example 1-4:

Example 1-4 Conversion of trifan to a `<triangles>` element

```

<triangles count="4" material="#Bricks">
  <input semantic="VERTEX" source="#verts" offset="0"/>
  <input semantic="NORMAL" source="#normals" offset="1"/>
  <p> 0 0 1 3 2 1
      0 0 2 1 3 2
      0 0 3 2 4 5

```

```

0 0 4 5 5 4
</p>
</triangles>

```

1.2.3 Mesh optimization

A mesh might be inefficient if the cache must be frequently flushed and reloaded.

Figure 1-5 shows how vertex renumbering can order the triangles more efficiently:

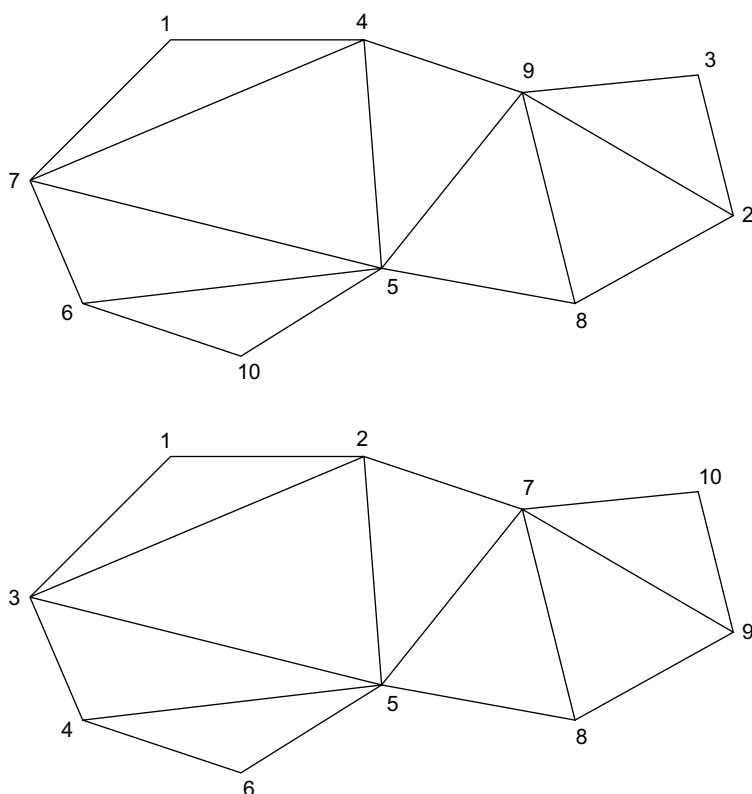


Figure 1-5 Vertex reorganization to improve cache performance

The algorithm for processing the vertices is complex, but a simplified version is:

1. Locate all <mesh> and <convex mesh> elements in a <geometry> that contain <triangles>.
2. Reduce, if possible, the number of vertices. This might require splitting meshes so that they are usable by `glDrawElements`.
3. Renumber the mesh to minimize the maximum vertex number.

Chapter 2

Installing the Asset Conditioning Tool

This section describes how to install the Asset Conditioning Tool. It contains the following sections.

- *About the Asset Conditioning Tool installation bundle* on page 2-2
- *Installing the Asset Conditioning Tool on Microsoft Windows* on page 2-3.

2.1 About the Asset Conditioning Tool installation bundle

The Asset Conditioning Tool bundle contains the Asset Conditioning Tool Windows binary in .msi format.

For more information, see the *Asset Conditioning Tool Release Note*.

2.2 Installing the Asset Conditioning Tool on Microsoft Windows

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

- *Installation requirements*
- *Installation procedure*

2.2.1 Installation requirements

To install the Asset Conditioning Tool on Microsoft Windows, you require:

- Microsoft Windows XP professional Version 2002, with service pack 2.
- A minimum of 172KB disk space for the Asset Conditioning Tool binary.
- The Visual C++ 2005 Redistributable Package.

The Redistributable Package is required to run the Asset Conditioning Tool, but it is not included in the installation bundle. Download the package from the Microsoft web site.

2.2.2 Installation procedure

The procedure to install the Asset Conditioning Tool on Microsoft Windows is:

1. Locate the Mali Developer Download Portal at:
<http://www.malideveloper.com>
2. Download the Asset Conditioning Tool package.
3. Run the file `Mali_GPU_Asset_Conditioning_Tool_WinXP_m.n.o.p.msi` by double clicking.
where:
 m = major version
 n = minor version
 $o.p$ = build version.
4. Select the required installation options and then click **Finish** to complete the installation.

By default, the Asset Conditioning Tool is installed in:

`C:\Program Files\ARM\Mali Developer Tools\Mali GPU Asset Conditioning Tool v $m.n$`

The following subdirectories are created:

- | | |
|-----------------------|---|
| <code>3rdparty</code> | This directory contains DLLs from third-party developers. |
| <code>bin</code> | This directory contains the <code>malict</code> executable. |

Chapter 3

Using the Asset Conditioning Tool

This chapter describes how to use the Asset Conditioning Tool. It contains the following sections:

- *Command line options* on page 3-2
- *Viewing converted files with third-party COLLADA viewers* on page 3-6.

3.1 Command line options

The Mali Asset Conditioning Tool syntax is:

```
malict command_list COLLADA_input_filename.dae
```

where:

command_list is one or more of the commands listed in Table 3-2 on page 3-3

COLLADA_input_filename.dae

is the name of the input file.

Table 3-1 Commands

Name	Abbreviation	Arguments	Description
help	h	None	Outputs help text on command line options and exits. The help information uses the dash (-) syntax to display the permitted options.
list	l	A comma separated list of conditioning steps	Outputs the COLLADA identifiers that are relevant for the specified given list of conditioning steps and exits. Example: /list triangulate,splitement
exclude	e	A comma separated list of conditioning steps followed by a colon and one of: <ul style="list-style-type: none"> a comma separated list of COLLADA identifiers a * to indicate all COLLADA identifiers 	Excludes the listed COLLADA identifiers from the specified conditioning steps. If you use an asterisk (*), all COLLADA identifiers are excluded for that conditioning step, and the step is effectively disabled. This option can be specified multiple times. Examples: /exclude unpack,splitarray:teapot,ball /exclude reorganize:*
include	i	A comma separated list of conditioning steps followed by a colon and one of: <ul style="list-style-type: none"> a comma separated list of COLLADA identifiers a '*' to indicate all COLLADA identifiers 	Includes the listed COLLADA identifiers from the specified conditioning steps If you use an asterisk (*), all COLLADA identifiers are excluded for that conditioning step, and the step is effectively disabled. By default the majority of conditioning steps include all COLLADA identifiers, however some conditioning steps (compress8) do not. You can use this on the command line to override the script options. You can specify this option multiple times. Example: /include compress8:cube
script	s	Script file name	Reads options from the specified file, and processes the command line options.
option	o	A conditioning step followed by a colon and a comma separated list of options for that conditioning step	Selects various options specific to the specified conditioning step. This option can be specified multiple times Example: /option all:verbose

Table 3-1 Commands (continued)

Name	Abbreviation	Arguments	Description
target	t	The target platform as either Mali200 or Mali400	Specifies the target platform for the conditioning steps. Default is Mali200
result	r	The COLLADA file name	Specifies the output file for the results of the processing. Default is <COLLADA Input File>_mct.dae
scriptout	so	The script output file name	Specifies the output file name for the script that will contain all options used in processing the COLLADA Input file. Default is none. No script file is output
version	v	None	Output the tool version and exit.

Under windows the commands can be pre-fixed by a forward slash (/), but you can use a dash (-) instead. Other Mali tools use dashes by default.

The application flags all unknown options, conditioning steps, conditioning step options, and target platforms as errors and exits.

3.1.1 Conditioning step names

Table 3-2 lists the names of the conditioning steps that can be used on the command line:

Table 3-2 Conditioning steps

Name	Short-hand	Conditioning step
triangulate	t	Specify conversion of unsupported primitives. This is also known as triangle tessellation.
unpack	u	Specify conversion of triangle strips and fans. This unpacks triangles from triangle strips and fans.
splitarray	sa	Split meshes for glDrawArrays.
splitelement	se	Split meshes for glDrawElements.
reorganize	r	Re-arrange indices. This reorganizes the triangles to be more cache friendly.

Table 3-2 Conditioning steps (continued)

Name	Short-hand	Conditioning step
compress16	c16	Specify that coordinate conversion uses 16-bit precision. This compresses meshes into a 16-bit integer co-ordinate space. ———— Note ———— This step can be overridden by the compress8 step to preform the maximum level of compression on a mesh.
compress8	c8	Specify that coordinate conversion uses 8-bit precision. This compresses meshes into a 8-bit integer co-ordinate space. ———— Note ———— <ul style="list-style-type: none"> This is step off by default, and all meshes are excluded. Use the include option to turn compress8 on. If compress8 is on, it takes precedence over the compress16 step running on a mesh
all		Use all of the conditioning steps. This is typically used with the exclude or the verbose options.

3.1.2 Target platform names

Table 3-3 lists the target names:

Table 3-3 Target names

Name	Short-hand	Description
mali55	m55	Mali 55
mali200	m200	Mali 200 with GP 2

3.1.3 Conditioning step options

Table 3-4 lists the conditioning step options:

Table 3-4 Conditioning step options

Name	Short-hand	Description
verbose	v	Output verbose information on each conditioning step.

By default the tool reports either:

- failure to process any elements
- the total number of elements that were processed either:
 - successfully
 - unsuccessfully
 - skipped (due to exclude list)

Each step has a verbose option that can report:

- which COLLADA element it is processing. The output information can include:
 - element type (for example <triangles>)
 - id
 - name.
- whether the element was skipped over or processed
- whether the processing (if done) was successful or failed
- if the processing was attempted but failed, the reason for the failure.

3.2 Viewing converted files with third-party COLLADA viewers

This section describes third-party viewers for COLLADA files.

———— **Note** —————

Not all viewers work with all COLLADA files.

ARM recommends the following viewers:

Blender with COLLADA plug-in

Download the files from the blender website, www.blender.org.

———— **Note** —————

Install Python before installing Blender.

COLLADA RT VIEWER

Download the files from the COLLADA website,
www.collada.org/mediawiki/index.php/Main_Page.

———— **Note** —————

This viewer uses COLLADA DOM 2.0.

Collview

Download the files from the Pinecoast website,
<http://www.pinecoast.com/collview.htm>.

———— **Note** —————

This viewer does not work with compressed integer data.

Glossary

This glossary describes some of the terms used in ARM manuals. Where terms can have several meanings, the meaning presented here is intended.

AEL *See* ARM Embedded Linux.

API *See* Application Programming Interface.

Application Programming Interface (API)

A specification for a set of procedures, functions, data structures, and constants that are used to interface two or more software components together. For example, an API between an operating system and the application programs that use it might specify exactly how to read data from a file.

ARM Embedded Linux (AEL)

A version of Linux ported to the ARM architecture.

COLLABorative Design Activity (COLLADA) files

Open-standard XML files that describe digital assets. COLLADA files are compatible with many graphics applications.

COLLADA *See* COLLABorative Design Activity (COLLADA) files.

Fps Frames per second. The number of individual frames that are rendered every second, to create an animation. Typical games and GUI applications run at 20-30 fps.

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

GPU *See* Graphics Processor Unit.

Graphics Processor Unit (GPU)

A hardware accelerator for graphics systems. Mali programmable GPUs, such as the Mali-200 GPU, consist of a geometry processor and a pixel processor. Mali fixed-function GPUs, such as the Mali-55 GPU consist of a pixel processor only.

Mali

A name given to graphics software and hardware products from ARM that aid 2D and 3D acceleration.

Mali Asset Conditioning Tool

The Mali Asset Conditioning Tool is a component of the Mali GPU Developer Tools. The Mali Asset Conditioning Tool enables you to optimize COLLADA files.

Mali User Interface Engine

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

Mali Binary Assets

The Mali Binary Assets file format provides an optimized binary representation of a 3D scene for loading into the Mali Demo Engine Library.

Mali GPU Developer Tools

A set of development programs that enables software developers to create graphic applications.

Mali Demo Engine Library

A C++ class framework for developing OpenGL ES 2.0 applications for the Mali GPU. The MDE Library is a component of the Mali SDK and Mali GPU Developer Tools.

Mesh

In graphics, a 3-dimensional arrangement of vertices and triangles, representing an object.

Performance Analysis Tool

A fully-customizable GUI tool that displays and analyzes performance data files produced by the instrumented drivers, together with framebuffer information. The PAT also features an online mode that enables application analysis in real-time.

Pixel

A pixel is a discrete element that forms part of an image on a display. The word pixel is derived from the term Picture Element.

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 PAT.

Red Hat Enterprise Linux (RHEL)

A version of desktop Linux operating systems

RHEL

See Red Hat Enterprise Linux (RHEL)

Performance variable

Data produced by the instrumented OpenGL ES 2.0 Emulator, that can be displayed and analyzed as statistical information in the PAT.

Shader

A program, usually an application program, running on the GPU, that calculates some aspect of the graphical output. See fragment shader and vertex shader.

Shader Library

A set of shader examples, tutorials, and other information, designed to assist with developing shader programs for the Mali GPU. The Shader Library is a component of the Mali SDK and Mali GPU Developer Tools.

Vertex shader

A program running on the geometry processor, that calculates the position and other characteristics, such as color and texture coordinates, for each vertex.