Mali™ OpenGL ES SDK for Linux on ARM
Version: 2.0.0
User Guide
Mali OpenGL ES SDK for Linux on ARM
User Guide

Copyright © 2011-2012 ARM. All rights reserved.

Release Information

The following changes have been made to this book.

<table>
<thead>
<tr>
<th>Date</th>
<th>Issue</th>
<th>Confidential</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>14 November 2011</td>
<td>A</td>
<td>Non-Confidential</td>
<td>First release.</td>
</tr>
<tr>
<td>10 April 2012</td>
<td>B</td>
<td>Non-Confidential</td>
<td>Updated with installation changes.</td>
</tr>
<tr>
<td>20 June 2012</td>
<td>C</td>
<td>Non-Confidential</td>
<td>Updated for version 1.2.</td>
</tr>
<tr>
<td>20 July 2012</td>
<td>D</td>
<td>Non-Confidential</td>
<td>Updated for version 2.0.0</td>
</tr>
</tbody>
</table>

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 final, that is for a developed product.

Web Address

http://www.arm.com
Contents

Mali OpenGL ES SDK for Linux on ARM User Guide

Preface
About this book ............................................................................................................ v
Feedback ................................................................................................................... vii

Chapter 1

Introduction
1.1 About the Mali SDK ................................................................................................. 1-2

Chapter 2

Installing the Mali OpenGL ES SDK for Linux on ARM
2.1 Mali SDK contents ................................................................................................... 2-2
2.2 Installing the Mali SDK on Windows ........................................................................ 2-3

Chapter 3

Building and Running the Samples
3.1 Building and running a sample from the command line ........................................ 3-2
3.2 Building and running the samples from Microsoft Visual Studio .............................. 3-3
3.3 Using the Template to write a sample application for Linux on ARM ...................... 3-4
Preface

This preface introduces the Mali OpenGL ES SDK for Linux on ARM User Guide. It contains the following sections:

• About this book on page v
• Feedback on page vii.
About this book

This is the *Mali OpenGL ES SDK for Linux on ARM User Guide*. It provides guidelines for using the *Mali OpenGL ES SDK for Linux on ARM* (Mali SDK) libraries and samples to develop graphics applications that run on a Linux platform that has an ARM processor.

Intended audience

This guide is written for system integrators and software developers creating OpenGL ES 2.0 and OpenGL ES 3.0 applications that are targeted to run on an embedded platform.

Using this book

This book is organized into the following chapters:

**Chapter 1 Introduction**

Read this for an introduction to the Mali SDK.

**Chapter 2 Installing the Mali OpenGL ES SDK for Linux on ARM**

Read this for a description on how to install and configure the Mali SDK on Windows.

**Chapter 3 Building and Running the Samples**

Read this for a description on how to build the Mali SDK samples on Windows.

Glossary

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


Typographical Conventions

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.

- **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>, <Rm>, <Opcode_2>`
Additional reading

This section lists publications by ARM and by third parties.


ARM publications

This guide contains information that is specific to the Mali Developer Tools. See the following documents for other relevant information:

- Mali GPU Application Optimization Guide (ARM DUI 0505)
- Mali GPU Performance Analysis Tool User Guide (ARM DUI 0502)
- Mali GPU Texture Compression Tool User Guide (ARM DUI 0503)
- Mali GPU Shader Developer Studio User Guide (ARM DUI 0504)
- OpenGL ES Emulator User Guide (ARM DUI 0511 for version 1.2)
- Mali GPU User Interface Engine User Guide (ARM DUI 0505)
- Mali GPU Mali Binary Asset Exporter User Guide (ARM DUI 0507)
- Mali GPU Shader Library User Guide (ARM DUI 0510)

Other publications

This section lists relevant documents published by third parties:

Feedback

ARM welcomes feedback on this product and its documentation.

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier 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 0607D
- 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 OpenGL ES SDK for Linux on ARM (Mali SDK), and describes how to start using it in your workflow. It contains the following section:

• About the Mali SDK on page 1-2.
1.1 About the Mali SDK

The Mali OpenGL ES SDK for Linux on ARM (Mali SDK) is a collection of resources to help you build OpenGL ES 2.0 and OpenGL ES 3.0 applications for a platform with a Mali GPU and an ARM processor. You can use it for creating new applications, training, and exploration of implementation possibilities.

You can use the Mali SDK to produce applications that will run on the following platforms:

- C++ applications that run under an embedded Linux operating system on an ARM processor
- C++ native applications that can run on an emulator on a Windows workstation.

_____ Note _______

- Hardware devices with a Mali-T600 series GPU supports OpenGL ES 3.0, OpenGL ES 2.0, and OpenGL ES 1.1.
- Hardware devices with a Mali-400 or Mali-200 GPU support OpenGL ES 2.0 and OpenGL ES 1.1.
- The OpenGL ES Emulator (version 1.0) supports OpenGL ES 3.0 and OpenGL ES 2.0.
- The OpenGL ES Emulator (version 1.2) supports OpenGL ES 2.0 and OpenGL ES 1.1.

1.1.1 Sample applications

The SDK includes sample applications for OpenGL ES 2.0 and OpenGL ES 3.0 APIs:

- Samples for OpenGL ES 3.0
- Samples for OpenGL ES 2.0 on page 1-3.

Samples for OpenGL ES 3.0

The following C++ native applications can be used on Linux on an ARM processor or on the OpenGL ES Emulator (version 1).

**Boids**

This shows how to use transform feedback to do calculations on the GPU using shaders. Movement calculations are done on the GPU. Data is not transferred back to the CPU, thus reducing memory transfers.

The scene is made up of 30 boids whose positions and velocities are calculated in a shader prior to rendering.

**Instancing**

This shows how to use instancing in a vertex shader to render multiple copies of the same geometry with different properties. A single copy of the geometry is uploaded to the GPU to reduce memory transfers.

The display shows ten spinning cubes in different positions and with different rotation speeds. All cubes are rendered from the same geometry memory.

**Instanced Tesselation**

This shows how to use instancing to create high complexity shapes from low-complexity geometry data. Patch data is combined with control points that are uploaded into buffers to create a complex model of a torus.

**ETC2TextureDemo**

This shows how to use each of the ten ETC2 texture compression formats that are supported in OpenGL ES 3.0.
IntegerLogic
This shows how to use integer formats in shaders. The sample demonstrates Rule 30 Cellular Automata.

Note
More information on individual sample applications might be supplied in a docs directory for that sample.

Samples for OpenGL ES 2.0
The following C++ native applications can be used on Linux on an ARM processor, the OpenGL ES Emulator (version 1 and version 1.2).

Antialias
This shows how to select anti-aliasing levels and the effect of different levels of anti-aliasing:
- a shader renders a simple triangle
- some text is written to the screen
- the FPS count is output on the terminal window.

Note
- With only approximately 2% performance drop, 4x anti-aliasing is nearly free on Mali hardware and is adequate for most applications.
- Significantly higher quality results from 16x anti-aliasing, but it has over 50% drop in performance.
- Because of the benefit with almost no cost, ARM recommends using 4x anti-aliasing rather than the default of no anti-aliasing.

Cube
This displays a spinning cube on the screen.
The sample application uses matrix functions, renders fonts, and writes the FPS value to the terminal.

EGLPreserve
This shows the change in behavior of eglSwapBuffers is caused by changing the EGL_SWAP_BEHAVIOR attribute to EGL_BUFFER_PRESERVED.

ETCAAtlasAlpha
This uses an alpha channel that was converted to a visible greyscale image. The alpha image is concatenated onto the original texture.

ETCCompressedAlpha
This uses an alpha channel that is delivered as a second packed texture.

ETCUncompressedAlpha
This uses an alpha channel that is provided as a raw 8-bit single channel image. Uncompressed alpha takes up more space than compressed alpha, but is more flexible and enables alpha and color information to be mixed.

ETCMipmap
This shows how to load and display ETC format textures with Mipmaps.
FrameBufferObject
This shows the render-to-texture feature of OpenGL ES 2.0. A colored spinning cube is rendered to a frame buffer, which is then attached as a texture on the faces of another spinning cube.

ListEGLConfigs
This shows:
• how to list the available EGLConfig
• how to select the correct EGLConfig to create a surface.

Rotozoom
This shows how to implement a classic demo effect by rotating and zooming a texture. It shows how to increase the speed of the effect by moving the work from the fragment shader to the vertex shader and CPU.

Template
This is an empty template that you can use to start developing a new application. The code is structured to contain everything that is required to compile and run, but nothing is rendered.

Triangle
This shows how to draw a simple colored triangle on the screen using a programmable shader.

--- Note ---
More information on individual sample applications might be supplied in a docs directory for that sample.

1.1.2 Simple framework library
The simple framework is a library you can use to simplify development of OpenGL ES 2.0 or OpenGL ES 3.0 applications that target ARM Mali platforms.

To view the simple-framework doxygen documentation, open the index.html file in the install_directory\docs\html folder.
Chapter 2
Installing the Mali OpenGL ES SDK for Linux on ARM

This section describes how to install the Mali SDL. It contains the following sections.

• Mali SDK contents on page 2-2
• Installing the Mali SDK on Windows on page 2-3.
2.1 Mali SDK contents

The Mali SDK bundle contains the C++ source code for the simple framework and samples. For more information see the Mali OpenGL ES SDK for Linux on ARM Release Note.

After installation on Windows, the files and directories shown in Figure 2-1 are placed in your chosen directory:

```
| Installation directory |   |   |
|------------------------+---+---|
|                        | arm-linux.cmake |   |
|                        | build-arm-linux.bat |   |
|                        | build-x86-win32.bat |   |
|                        | CMakeLists.txt |   |
|                        | docs |   |
|                        | EULA.rtf |   |
|                        | bin |   |
|                        | inc |   |
|                        | lib |   |
|                        | samples |   |
|                        | simple-framework |   |
|                        | utilities |   |
|                        | visual-studio |   |
```

Figure 2-1 Mali SDK files on Windows
2.2 Installing the Mali SDK on Windows

This section describes how to install the Mali SDK on Microsoft Windows. It contains the following sections:
- Installation requirements for the Mali SDK on Microsoft Windows
- Installation procedure for the Mali SDK on Microsoft Windows.

2.2.1 Installation requirements for the Mali SDK on Microsoft Windows

To install the Mali SDK on Microsoft Windows, you require:

- Microsoft Windows 7.
  - **Note**
    - The Mali SDK has been tested successfully on 32-bit and 64-bit versions of Microsoft Windows 7.
  - A minimum of 40MB disk space to install the Mali SDK library and applications.
  - CMake 2.8.0 or above (CMake 2.8.5 is included in the installation package).
  - Visual Studio 2008 or above.
  - The Khronos OpenGL ES and EGL libraries (these are included in the installation package).

2.2.2 Installation procedure for the Mali SDK on Microsoft Windows

To install the Mali SDK on Microsoft Windows:

2. Go to the Mali Developer Center web site at: http://www.malideveloper.com
3. Download the Mali OpenGL ES SDK for Linux on ARM package.
4. Run the file Mali_OpenGL_ES_SDK_for_Linux_on_ARM_vmn.op.Win32.msi by double clicking.
   - where: 
     - \(m\) identifies the major version 
     - \(n.o.p\) identifies the minor version.
5. Select the required installation options and then click **Finish** to complete the installation.
   - By default, the Mali SDK is installed in the documents folder of the current user. The following sub-folder is created:
   - **Note**
     - The installer updates the CMake present on your computer to CMake 2.8.5.
Chapter 3
Building and Running the Samples

This chapter describes how to build the samples provided with the Mali OpenGL ES SDK for Linux on ARM (Mali SDK). It contains the following sections:

• Building and running a sample from the command line on page 3-2
• Building and running the samples from Microsoft Visual Studio on page 3-3
• Using the Template to write a sample application for Linux on ARM on page 3-4
3.1 Building and running a sample from the command line

All of the samples can be built and run from the command line.

To use a terminal to build and run a sample on Microsoft Windows for the Mali OpenGL ES Emulator:

1. Open a Visual Studio command prompt.
2. Change to the Mali SDK installation directory.
3. To list the samples, enter:
   ```bash
   build-x86-win32.bat --help
   ```
4. Run the batch file to build the samples:
   - To build all of the OpenGL ES 2.0 and OpenGL ES 3.0 samples, enter:
     ```bash
     build-x86-win32.bat
     ```
   - To build a single sample named `sample_name`, enter:
     ```bash
     build-x86-win32.bat sample_name
     ```
5. The binary for the sample is created in `build\x86\`:
   - Run the binary to execute the OpenGL ES 3.0 code under emulation on your desktop:
     ```bash
     cd build\x86\OpenGL-ES-3.0\sample_name\sample_name
     ```
   - Run the binary to execute the OpenGL ES 2.0 code under emulation on your desktop:
     ```bash
     cd build\x86\OpenGL-ES-2.0\sample_name\sample_name
     ```

To use a terminal to build a sample on Microsoft Windows to run under Linux on an ARM device:

1. Open a command prompt.
2. Change to the Mali SDK root directory.
3. To list the samples, enter:
   ```bash
   build-arm-linux.bat --help
   ```
4. Run the batch file:
   - To build all of the samples, enter:
     ```bash
     build-arm-linux.bat
     ```
   - To build a single sample named `sample_name`, enter:
     ```bash
     build-arm-linux.bat sample_name
     ```
5. The binary for the sample is created in:
   - `build\arm\OpenGL-ES-3.0\sample_name` for OpenGL ES 3.0
   - `build\arm\OpenGL-ES-2.0\sample_name` for OpenGL ES 2.0.
6. The sample folder contains the binary file and the assets. Copy the folder to the computer with the ARM processor and run it there.
3.2 Building and running the samples from Microsoft Visual Studio

Project files for Microsoft Visual C++ 2008 are released in the visual-studio directory.

--- Note ---

- The assets folder contains all the resources for a sample and is required in its working directory.

---

To build all the OpenGL ES 3.0 samples with Microsoft Visual C++ 2008:

1. Open the solution file OpenGL ES 3.0 Samples.sln.
2. Select **Build Solution** from the **Build** menu or press **F7**.
3. If built with the debug option, all projects are placed into the following directory:
   
   `installation_directory\visual-studio\Debug\`

To build all the OpenGL ES 2.0 samples with Microsoft Visual C++ 2008:

1. Open the solution file OpenGL ES 2.0 Samples.sln.
2. Select **Build Solution** from the **Build** menu or press **F7**.
3. If built with the debug option, all projects are placed into the following directory:
   
   `installation_directory\visual-studio\Debug\`

To run one sample:

1. Set it as Start Up project by right-clicking on the corresponding project in the list and choose **Set as StartUp Project**.
2. Click **Start Debugging** from the **Debug** menu, or press **F5**, to run the sample.
3.3 Using the Template to write a sample application for Linux on ARM

Using the Template sample as a base you can easily get started writing OpenGL ES 2.0 applications.

To add code to Template:

1. Add setup code that will be run only at the start in the setupGraphics() method. This method performs startup actions such as loading shaders, enabling OpenGL ES states, and loading textures.

2. Place the code that will draw each frame in the renderFrame() method.

3.3.1 Building the sample

The sample can either be compiled and run on the desktop using the supplied emulator or compiled to a Linux on ARM binary and deployed to a device.

Building on Windows

To use Windows to build the sample to run on the desktop with the emulator:

1. Go to the root directory of the Mali SDK and run:
   build-x86-win32.bat Template

2. The binary is created in the build\x86\OpenGL-ES-2.0\Template directory.

3. Run the binary to start the OpenGL ES 2.0 code running under emulation on your desktop.

To build the sample to run on a device running Linux on ARM:

1. Go to the root directory of the Mali SDK and run:
   build-arm-linux.bat Template

2. The binary is created in the build\arm\OpenGL-ES-2.0\Template directory.

3. Deploy the binary and run it on a device running Linux on ARM.