

Mali™ OpenGL ES 2.0 SDK for Linux on ARM

Version: 1.0.0

User Guide

ARM®

Mali OpenGL ES 2.0 SDK for Linux on ARM

User Guide

Copyright © 2011 ARM. All rights reserved.

Release Information

The following changes have been made to this book.

Change history			
Date	Issue	Confidentiality	Change
14 November 2011	A	Non-Confidential	First release.

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 2.0 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 2.0 SDK for Linux on ARM	
	2.1 Mali SDK contents	2-2
	2.2 Installing the Mali SDK on Windows	2-3
	2.3 Installing the Mali SDK on Linux	2-4
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	3-4
	3.3 Building and running the samples from Eclipse	3-5
	3.4 Using the Template to write a sample application for Linux on ARM	3-7

Preface

This preface introduces the *Mali OpenGL ES 2.0 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 2.0 SDK for Linux on ARM User Guide*. It provides guidelines for using the *Mali OpenGL ES 2.0 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 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 2.0 SDK for Linux on ARM

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

Chapter 3 Building and Running the Samples

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

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.

The *ARM Glossary* is available on the ARM Infocenter at, <http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html>.

Typographical 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 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)
- *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)
- *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>.
- *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, 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 0607A
- 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 2.0 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 2.0 SDK for Linux on ARM* (Mali SDK) is a collection of resources to help you build OpenGL ES 2.0 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.

The Mali SDK runs on the following platforms:

- Microsoft Windows XP Professional, service pack 3
- Ubuntu Linux version 10.04.

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 Linux or Windows workstation.

1.1.1 Sample applications

The following C++ native applications can be used on Linux on an ARM processor.

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 example uses matrix functions, renders fonts, and writes the FPS value to the terminal.

EGLPreserve

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

ETCAtlasAlpha

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

ETCCompressedAlpha

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

ETCUncompressedAlpha

This example 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 example shows how to load and display ETC format textures with Mipmaps.

FramebufferObject

This example 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 example shows:

- how to list the available EGLConfig
- how to select the correct EGLConfig to create a surface.

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

Note

More information on individual samples might be supplied in the docs folder of the sample.

Chapter 2

Installing the Mali OpenGL ES 2.0 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](#)
- [Installing the Mali SDK on Linux on page 2-4.](#)

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 2.0 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-eclipse-projects.bat
├── build-msvc-projects.bat
├── build-x86-win32.bat
├── CMakeLists.txt
├── EULA.rtf
├── bin
├── inc
├── lib
├── samples
├── simple-framework
├── utilities
└── visual-studio-projects

```

Figure 2-1 Mali SDK files on Windows

After installation on Linux, the files and directories shown in [Figure 2-2](#) are placed in your chosen directory:

```

Installation directory
├── arm-linux.cmake
├── build-arm-linux.sh
├── build-eclipse-projects.sh
├── build-x86-win32.sh
├── CMakeLists.txt
├── EULA.rtf
├── bin
├── inc
├── lib
├── samples
├── simple-framework
└── utilities

```

Figure 2-2 Mali SDK files on Linux

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 XP Professional, service pack 3

———— **Note** —————

The Mali SDK has been tested successfully on a 32-bit version of Microsoft Windows XP Professional edition.

- a minimum of 40MB disk space to install the Mali SDK library and applications.
- the GNU EABI Toolchain for ARM Processors (Sourcery CodeBench Lite Edition for ARM) from Code Sourcery (download from <http://codesourcery.com>)
- CMake 2.8.5 (this is included in the installation package)
- the Khronos OpenGL ES and EGL libraries (these are included in the installation package)

———— **Note** —————

You can use the Eclipse environment, but all of the samples can be build from the command line. Use Eclipse version 3.5 or later.

2.2.2 Installation procedure for the Mali SDK on Microsoft Windows

To install the Mali SDK on Microsoft Windows:

1. If you prefer to use the Eclipse build environment, download and install Eclipse from <http://www.eclipse.org/>. Eclipse version 3.5 (at least) is recommended.
2. Download the GNU EABI Toolchain for ARM Processors (Sourcery CodeBench Lite Edition for ARM) from <http://codesourcery.com>.
3. Go to the Mali Developer Center web site at:
<http://www.malideveloper.com>
4. Download the Mali OpenGL ES 2.0 SDK for Linux on ARM package.
5. Run the file `Mali_OpenGL_ES_2_0_SDK_for_Linux_on_ARM_m.n.o.p_Win32.msi` by double clicking.
where:
m identifies the major version
n.o.p identifies the minor version.
6. 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:
`ARM\Mali OpenGL ES 2.0 SDK for Linux on ARM m.n.o`

2.3 Installing the Mali SDK on Linux

This section describes how to install the Mali SDK on a Linux workstation. It contains the following sections:

- [Installation requirements for Linux](#)
- [Installation procedure for Linux.](#)

Note

The Mali SDK has been tested successfully on a 32-bit version of Linux OS.

2.3.1 Installation requirements for Linux

The following are required for Linux platforms:

- Ubuntu Linux version 10.04
- a minimum of 40MB disk space to install the Mali SDK library and applications.
- CMake 2.8.5 (this is included in the installation package)
- the GNU EABI Toolchain for ARM Processors (Sourcery CodeBench Lite Edition for ARM) from Code Sourcery (download from <http://codesourcery.com>)
- the Khronos OpenGL ES and EGL libraries (these are included in the installation package)

Note

You can use the Eclipse environment, but all of the samples can be build from the command line. Use Eclipse version 3.5 or later.

2.3.2 Installation procedure for Linux

To install the Mali SDK on a Linux workstation:

1. If you prefer to use the Eclipse build environment, download and install Eclipse from <http://www.eclipse.org/>. Eclipse version 3.5 (at least) is recommended.
2. Download the GNU EABI Toolchain for ARM Processors (Sourcery CodeBench Lite Edition for ARM) from <http://codesourcery.com>.
3. Go to the Mali Developer Center web site at:
<http://www.malideveloper.com>
4. Download the following package:
Mali_OpenGL_ES_2_0_SDK_for_Linux_on_ARM_m.n.o.p_Linux.tgz
where:
m identifies the major version
n.o.p identifies the minor version.
5. Decompress the file:
 - a. open a command terminal and navigate to the directory where you have downloaded the package
 - b. type the following command:
tar -zxvf Mali_OpenGL_ES_2_0_SDK_for_Linux_on_ARM_m.n.o.p_Linux.tgz

6. The following directory is created which contains the Mali SDK:
Mali_OpenGL_ES_2_0_SDK_for_Linux_on_ARM_m.n.o

Chapter 3

Building and Running the Samples

This chapter describes how to build the samples provided with the *Mali OpenGL ES 2.0 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 3-4*
- *Building and running the samples from Eclipse on page 3-5.*
- *Using the Template to write a sample application for Linux on ARM on page 3-7*

3.1 Building and running a sample from the command line

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

- [Building and running a sample from a Windows workstation](#)
- [Building and running a sample from a Linux workstation](#).

3.1.1 Building and running a sample from a Windows workstation

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 2.0 Emulator:

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

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. To list the samples, enter:
`build-arm-linux.bat --help`
3. Change to the Mali SDK root directory.
4. Run the batch file:
 - To build all of the samples, enter:
`build-arm-linux.bat`
 - To build a single sample named *sample_name*, enter:
`build-arm-linux.bat sample_name`
5. The binary for the sample is created in `build\arm\sample_name`.
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.

3.1.2 Building and running a sample from a Linux workstation

To use a terminal to build and run a sample on x86 Linux for the Mali OpenGL ES2.0 Emulator:

1. Open a command prompt.
2. Change to the Mali SDK root directory.

3. To list the samples, enter:
`bash build-x86-linux.sh --help`
4. Run the batch file:
 - To build all of the samples, enter:
`bash build-x86-linux.sh`
 - To build a single sample named *sample_name*, enter:
`bash build-x86-linux.sh sample_name`
5. The binary for the sample is created in `build\x86\sample_name`.
6. Run the binary to execute the OpenGL ES 2.0 code under emulation on your desktop.

To use a terminal to build and run a sample on x86 Linux to run on 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.sh --help`
4. Run the batch file to build the samples:
 - To build all of the samples, enter:
`bash build-arm-linux.sh`
 - To build a single sample named *sample_name*, enter:
`bash build-arm-linux.sh sample_name`
5. The binary for the sample is created in `build\arm\sample_name`.
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-projects` directory.

Note

- Samples require the OpenGL ES 2.0 Emulator to run on Windows, see the *OpenGL ES 2.0 Emulator User Guide*.
 - The `assets` folder contains all the resources for a sample and is required in its working directory.
-

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

1. Open the solution file `Mali_SDK.sln`.
2. Select **Build Solution** from the **Build** menu or press **F7**.
3. All the projects are built and placed into the corresponding directory in: `visual-studio-projects/samples/linux`.

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. The individual Visual Studio project for the sample is placed in the corresponding directory in `visual-studio-projects/samples/linux`.

Released Visual Studio projects are configured with a post-build command to copy the assets files from the sources directory to the working directory of each sample.

3.2.1 Generating Visual Studio projects with CMake

Solution and project files for different versions of Visual Studio can be generated using CMake:

1. Create a new directory inside the installation directory of the Mali SDK.
2. Change to the directory you created.
3. Enter the following:

```
cmake ../ -DCMAKE_SUPPRESS_REGENERATION=1 -DCMAKE_USE_RELATIVE_PATHS=1
```
4. New compatible solution and project files for the Visual Studio version are created.

3.3 Building and running the samples from Eclipse

This section describes how to use Eclipse to build and run applications.

3.3.1 Building the samples and running them in the Mali OpenGL ES 2.0 Emulator

To build the samples:

1. Open a command window and change to the Mali SDK installation directory.
2. Run the batch file that builds the Eclipse projects:
 - From Windows enter:
build-eclipse-projects.bat
 - From Linux enter:
bash build-eclipse-projects.sh
3. Start Eclipse.
4. Select **File** → **Import** to open the Import dialog.
5. Click **General** and select **Existing Projects into Workspace**.
6. Click **Browse** and navigate to the directory where the projects were created.
7. Select the x86-build directory and click **OK**.
8. Click **Finish**.
9. Select the project in Eclipse **Project Explorer** pane.
10. Select **Build Project** from the **Project** menu.
11. Click **Run** from the **Run** menu to display the select application dialog.
12. Select the sample to run and click **OK**.
13. The sample executes on the Mali OpenGL ES 2.0 Emulator.

3.3.2 Building the samples and running them under Linux on an ARM device

To build the samples:

1. Open a command window and change to the Mali SDK installation directory.
2. Run the batch file that builds the Eclipse projects:
 - From Windows enter:
build-eclipse-projects.bat
 - From Linux enter:
bash build-eclipse-projects.sh
3. Start Eclipse.
4. Select **File** → **Import** to open the Import dialog.
5. Click **General** and select **Existing Projects into Workspace**.
6. Click **Browse** and navigate to the directory where the projects were created.
7. Select the arm-build directory and click **OK**.
8. Click **Finish**.

9. Select the project in Eclipse **Project Explorer** pane.
10. Select **Build Project** from the **Project** menu.
11. Copy the binary file and the assets folder to your target and run it there.

———— **Note** —————

The procedure for running the application depends on your target device.

3.4 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

3.4.1 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\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\Template` directory.
3. Deploy the binary and run it on a device running Linux on ARM.

Building on Linux

To use Linux 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-linux.sh Template`
2. The binary is created in the `build/x86/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.sh Template`
2. The binary is created in the `build/arm/Template` directory.
3. Deploy the binary and run it on a device running Linux on ARM.