Mali™ OpenGL ES 2.0 SDK for Android

Version: 1.0.0

User Guide

ARM®

ARM DUI 0587A (ID120411)

Mali OpenGL ES 2.0 SDK for Android
User Guide

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

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 Android User Guide

Preface

This preface introduces the *Mali OpenGL ES 2.0 SDK for Android User Guide*. It contains the following sections:

About this book

This is the *Mali OpenGL ES 2.0 SDK for Android User Guide*. It provides guidelines for using the libraries and samples in the *Mali OpenGL ES 2.0 SDK for Android* (Mali SDK) to develop graphics applications that run on an Android 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 Mali SDK.

Chapter 2 *Installing the Mali OpenGL ES 2.0 SDK for Android*

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

Chapter 3 *Building and Running the Samples*

Read this for a description on how to build the SDK 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:

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 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 0587A
- 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 Android* (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

Mali SDK is a collection of resources to help you build OpenGL ES 2.0 applications for a platform with a Mali GPU. You can use it for creating new applications, training, and exploration of implementation possibilities.

Mali SDK runs on the following platforms:

- Microsoft Windows XP Professional Version 2002, service pack 3
- Ubuntu Linux 10.4.

You can use the Mali SDK to produce the following types of Android applications:

Android NDK

These are C++ applications that have a Java wrapper. These run under the Android OS on an ARM processor.

Android SDK

These applications are written in Java and run under the Android OS on an ARM processor.

1.1.1 Sample applications

The following Android NDK applications can be used on Android.

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

FrameBufferObject

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

The following Android SDK Java applications can be used on Android.

RenderToTexture

This sample is similar to the FrameBufferObject NDK sample, except this sample is written in Java.

SpinningCube

This displays a spinning cube on the screen.

The textures are rendered by Java code.

Template This is an empty Java 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.

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 Android

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

2.1 Mali SDK contents

The Mali SDK bundle contains the software, C++ source code for the simple framework, and samples.

For more information see the *Mali OpenGL ES 2.0 SDK for Android Release Note*.

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

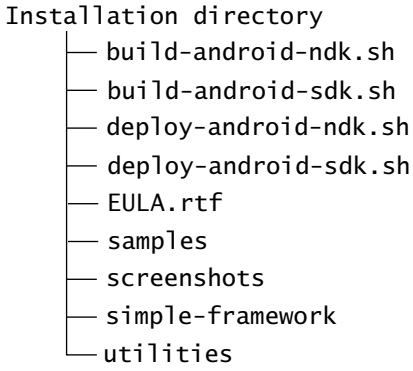

Figure 2-1 Mali SDK files and directories

2.2 Installing the Mali SDK on Microsoft Windows

This section describes how to install the Mali SDK on Microsoft Windows. It contains the following sections:

- *Installation requirements for Microsoft Windows*
- *Installation procedure for Microsoft Windows*.

Note

The Mali SDK has been tested successfully on a 32-bit version of Microsoft Windows XP Professional edition with version 6b of the Android NDK.

2.2.1 Installation requirements for Microsoft Windows

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

- Microsoft Windows XP Professional, service pack 3
- a minimum of 40MB disk space to install the Mali SDK library and applications.
- Cygwin from `http://cygwin.com/`.
- Ant (version 1.8 or above) from `http://ant.apache.org/`
- Android SDK from `http://developer.android.com/sdk/index.html`
- Android NDK from `http://developer.android.com/sdk/ndk/index.html`
- Java JDK from `http://www.oracle.com/technetwork/java/javase/overview/index.html`.

  Caution

  — The `jarsigner` tool supplied with Java SE 7 is not usable with the Mali SDK for manually signing non-debug releases.

  — Use the `jarsigner` file supplied with Java SE 6 to manually sign releases.

- the Khronos OpenGL ES and EGL libraries (these are included in the Android NDK)

Note

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

2.2.2 Installation procedure for Microsoft Windows

To install the Mali SDK for Android on Microsoft Windows:

1. If you wish to use the Eclipse build environment, download and install Eclipse from `http://www.eclipse.org/`. Eclipse version 3.5 (at least) is recommended.
2. If you have not already done so, download and install the Android SDK and related tools listed in *Installation requirements for Microsoft Windows*.
3. If you have not already done so, download and install the Java JDK.
4. Download the Mali SDK for Android package.
5. Run the SDK Manager.exe tool from the Android SDK and install the following packages:
   - Android SDK Platform-tools

- Android 2.2 (API 8 and 10).

6. Update the PATH environment variable to include the location of the Android SDK, Android NDK, and Ant:

```
set PATH=%PATH%;android_sdk_path\tools;android_sdk_path\platform-tools;android_ndk_path;ant_path\bin
```

where:

android_sdk_path
is the path to the location where you installed the Android SDK

android_ndk_path
is the path to the location where you installed the Android NDK

ant_path is the path to the location where you installed Ant.

7. Go to the Mali Developer Center web site at:

   `http://www.malideveloper.com`

8. Run the file `Mali_OpenGL_ES_2_0_SDK_for_Android_m.n.o.p_Win32.msi` by double clicking.

   where:

   m identifies the major version

   n.o.p identifies the minor version.

9. 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 Android m.n.o`

10. Change to the `utilities` folder in the installation folder and verify that your installation is properly setup:

```
bash check-android-sdk.sh
bash check-android-ndk.sh
```

2.3 Installing the Mali SDK on Linux

This section describes how to install the Mali SDK Linux. 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 with version 6b of the Android NDK.

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.
- Ant (version 1.8 or above) from `http://ant.apache.org/`
- Android SDK from `http://developer.android.com/sdk/index.html`
- Android NDK from `http://developer.android.com/sdk/ndk/index.html`
- Java JDK from `http://www.oracle.com/technetwork/java/javase/overview/index.html`.

  Caution

  — The `jarsigner` tool supplied with Java SE 7 is not usable with the Mali SDK for manually signing non-debug releases.

  — Use the `jarsigner` file supplied with Java SE 6 to manually sign releases.

- the Khronos OpenGL ES and EGL libraries (these are included in the Android NDK)
- You can use the Eclipse environment, but all of the samples can be build from the command line.

2.3.2 Installation procedure for Linux

To install the Mali OpenGL ES 2.0 SDK for Android on Linux:

1. If you wish to use the Eclipse build environment, download and install Eclipse from `http://www.eclipse.org/`. Eclipse version 3.5 (at least) is recommended.
2. If you have not already done so, download and install the Java JDK.
3. Download and install the Android SDK and related tools listed in *Installation requirements for Linux*.
4. Run the SDK Manager.exe tool from the Android SDK and install the following packages:
   - Android SDK Platform-tools
   - Android 2.2 (API 8 and API 10).
5. Update the path to include the location of the Android SDK, Android NDK, and Ant software:

```
export PATH=$PATH:android_sdk_path/tools:android_sdk_path/platform-tools:android_ndk_path:ant_path/bin
```

where:

android_sdk_path
is the path to the location where you installed the Android SDK

android_ndk_path
is the path to the location where you installed the Android NDK

ant_path is the path to the location where you installed Ant.

6. Go to the Mali Developer Center web site at:

   `http://www.malideveloper.com`

7. Download the tar file containing the Mali SDK for Android package:

   `Mali_OpenGL_ES_2_0_SDK_for_Android_`*`m.n.o.p`*`_linux.tgz`

   where:

   m identifies the major version

   n.o.p identifies the minor version.

8. 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_Android_`*`m.n.o.p`*`_linux`

      The Mali SDK is installed in the following sub-folder:

      `Mali_OpenGL_2_0_ES_SDK_for_Android_`*`m.n.o`*

9. Change to the `utilities` folder in the installation folder and verify that your installation is properly setup:

   ```
   bash check-android-sdk.sh
   bash check-android-ndk.sh
   ```

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 Android* (Mali SDK). It contains the following sections:

3.1 Building an Android sample from the command line

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

- *Building and running an Android NDK sample*
- *Building and running an Android SDK sample*.

3.1.1 Building and running an Android NDK sample

To use a terminal to build and deploy an Android NDK sample from Microsoft Windows or x86 Linux:

1. Ensure your path contains the Android NDK as described in Chapter 2 *Installing the Mali OpenGL ES 2.0 SDK for Android*.
2. Open a command prompt.
3. Change to the Mali SDK root directory.
4. Run the script to build the sample:
   - To build all of the Android NDK samples, run:
     `bash build-android-ndk.sh`
   - To build a single sample named *sample_name*, run:
     `bash build-android-ndk.sh sample_name`
5. Run the script to deploy the sample to a connected Android device:
   - To deploy all of the Android NDK samples, run:
     `bash deploy-android-ndk.sh`
   - To deploy a single sample named *sample_name*, run:
     `bash deploy-android-ndk.sh sample_name`
6. Go to the Android menu screen on the device and run the sample.

3.1.2 Building and running an Android SDK sample

To use a terminal to build and deploy an Android SDK sample from Microsoft Windows or x86 Linux:

1. Ensure your path contains the Android SDK as described in Chapter 2 *Installing the Mali OpenGL ES 2.0 SDK for Android*.
2. Open a command prompt.
3. Change to the Mali SDK for Android root directory.
4. Run the batch file to build the sample:
   - To build all of the Android SDK samples, run:
     `bash build-android-sdk.sh`
   - To build a single sample named *sample_name*, run:
     `bash build-android-sdk.sh sample_name`
5. Run the batch file to deploy the sample to a connected Android device:
   - To deploy all of the Android SDK samples, run:
     `bash deploy-android-sdk.sh`
   - To deploy a single sample named *sample_name*, run:
     `bash deploy-android-sdk.sh sample_name`

6. Go to the Android menu screen on the device and run the sample.

3.2 Writing an Android NDK sample

You can use the template sample as a base to get started writing OpenGL ES 2.0 applications.

This section describes using C++ OpenGL ES 2.0 and JNI.

To add code to Template:

1. Open `Template.cpp` in the `jni` folder
2. Add setup code which will be run once at the start in the `setupGraphics()` method. This performs startup action such as, for example, loading shaders, enabling OpenGL ES states, and loading textures.
3. Place the code which will draw each frame in the `renderFrame()` method

3.2.1 Testing the sample

To build and run the sample from Windows or Linux:

1. Go to the root directory of the Mali SDK and run:
   ```
   bash build-android-ndk.sh Template
   ```
2. The APK file (Android Package) is created in
   `samples/android-ndk/Template/bin.`
3. Deploy and run the sample on an Android device, run:
   ```
   bash deploy-android-ndk.sh Template
   ```

   Note

   The sample must be deployed to an Android device because the Android Emulator does not currently support OpenGL ES 2.0.

3.3 Writing an Android SDK sample

You can use the Mali SDK Template sample as a base for writing your own OpenGL ES 2.0 applications.

This section describes using OpenGL ES 2.0 directly from Java.

To add code to Mali SDK Template sample:

1. Open `GLES20Renderer.java` in the `src` folder
2. Add setup code which will be run once at the start in the `onSurfaceCreated()` method. This performs startup action such as, for example, loading shaders, enabling OpenGL ES states, and loading textures.
3. Place the code which will draw each frame in the `onDrawFrame()` method

3.3.1 Testing the sample

To build and run the sample for an Android device:

1. Go to the root directory of the Mali SDK and run:
   ```
   bash build-android-sdk.sh Template
   ```
2. The APK file (Android Package) is created in
   `samples/android-sdk/Template/bin.`
3. Deploy and run the sample on an Android device, run:
   ```
   bash deploy-android-sdk.sh Template
   ```

   Note

   The sample must be deployed to an Android device because the Android Emulator does not currently support OpenGL ES 2.0.

3.4 Building the samples from Eclipse

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

3.4.1 Building and running the Android SDK samples

To use Eclipse to build and run an Android SDK sample:

1. Ensure you have installed the Android and Ant software as described in Chapter 2 *Installing the Mali OpenGL ES 2.0 SDK for Android*.
2. Ensure you have installed the ADT plugin, see `http://developer.android.com/sdk/eclipse-adt.html`.
3. Start Eclipse.
4. Select **File → New → Android project**.
5. Select **Create project from existing source**.
6. Click **Browse...** and select the individual sample folder.
7. Click **Finish**.
8. Click **Run** and select the Android device as the target.
9. The sample executes on the Android device.

3.4.2 Building and running the Android NDK samples

To use Eclipse to build and run an Android NDK sample:

1. Ensure you have installed the Android and Ant software as described in Chapter 2 *Installing the Mali OpenGL ES 2.0 SDK for Android*.
2. Ensure you have installed the ADT plugin, see `http://developer.android.com/sdk/eclipse-adt.html`.
3. Start Eclipse.
4. Create the `Common` project from the source files:
   a. Select **File → New → Android project**.
   b. Select **Create project from existing source**.
   c. Click **Browse...** and select the `Common` folder inside the `samples/android-ndk` folder.
   d. Click **Finish**.
   e. Select the `Common` project.
   f. Select **Build Project** from the **Project** menu.
5. Select **File → New → Android project**.
6. Select **Create project from existing source**.
7. Click **Browse...** and select the individual sample folder.
8. Click **Finish**.
9. The C++ sections of the code will not be compiled by the standard Eclipse build command so an alternative is required. To compile the C++ code either:
   - Run the `ndk-build` command from a command line in the sample folder.

- Add a custom build command to the eclipse project to make it part of the Eclipse build process:
  1. Select **Properties** from the **Project** folder.
  2. Select **Builders**
  3. Click **New**
  4. Select **Program** for the configuration type
  5. Enter `NDK Build` for the name
  6. On Windows, set the location and argument fields:
     a. In **Location** enter the path to cygwin bash, for example:
        `C:\cygwin\bin\bash.exe`
     b. In **Arguments** enter:
        `--login -c "cd '${project_loc}'; ndk-build"`

     On Linux, set the location and directory fields:
     a. In **Location** enter the path to `ndk-build`:
        `path_to_android_ndk``/ndk-build`
        where *`path_to_android_ndk`* is the root of the Android NDK.
     b. In **Working Directory** enter:
        `${workspace_loc:/`*`name_of_project`*`}`
        where *`name_of_project`* is the name of the project such as, for example, `${workspace_loc:/Cube}`.
  7. Press **OK**
  8. Move the new **NDK Build** Builder to the top of the list of builders

10. Click **Run** and select the Android device as the target.
11. The sample executes on the Android device.