

OpenGL ES 1.1 Emulator

Version: 1.0

User Guide

ARM[®]

OpenGL ES 1.1 Emulator

User Guide

Copyright © 2009 ARM. All rights reserved.

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
14 October 2009	A	Non-Confidential	First release for v1.0

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.

Unrestricted Access is an ARM internal classification.

Product Status

The information in this document is final, that is for a developed product.

Web Address

<http://www.arm.com>

Contents

OpenGL ES 1.1 Emulator User Guide

	Preface	
	About this book	x
	Feedback	xii
Chapter 1	Introduction	
	1.1 About the OpenGL ES 1.1 Emulator	1-2
Chapter 2	Installation and Configuration for Windows	
	2.1 Installing the OpenGL ES 1.1 Emulator on Windows	2-2
	2.2 Installing the Khronos header files	2-6
	2.3 Configuring the OpenGL ES 1.1 Emulator	2-8
	2.4 Building the OpenGL ES 1.1 examples for Windows	2-12
Chapter 3	Installation and Configuration for Linux	
	3.1 Installing the OpenGL ES 1.1 Emulator on Linux	3-2
	3.2 Installing the Khronos header files	3-5
	3.3 Configuring the OpenGL ES 1.1 Emulator	3-7
	3.4 Building the OpenGL ES 1.1 examples for Linux	3-10
Chapter 4	Implementation information	
	4.1 OpenGL ES 1.1 implementation information	4-2
	4.2 EGL implementation information for Windows	4-4

4.3 EGL implementation information for Linux 4-9

Glossary

List of Tables

OpenGL ES 1.1 Emulator User Guide

	Change history	ii
Table 2-1	OpenGL ES 1.1 Emulator directory file contents	2-4
Table 2-2	OpenGL ES 1.1 Emulator library structure	2-10
Table 3-1	OpenGL ES 1.1 Emulator directory file contents	3-4
Table 3-2	OpenGL ES 1.1 Emulator library structure	3-8

List of Figures

OpenGL ES 1.1 Emulator User Guide

Figure 2-1	OpenGL ES 1.1 Emulator directory structure	2-4
Figure 2-2	OpenGL ES 1.1 include directory	2-7
Figure 2-3	simpApp11 image	2-13
Figure 2-4	rotateTriangle11 image	2-14
Figure 3-1	OpenGL ES 1.1 Emulator directory structure	3-4
Figure 3-2	OpenGL ES 1.1 include directory	3-6
Figure 3-3	simpApp11 image	3-11
Figure 3-4	rotateTriangle11 image	3-12

Preface

This preface introduces the *OpenGL ES 1.1 Emulator User Guide*. It contains the following sections:

- *About this book* on page x
- *Feedback* on page xii.

About this book

This is the *OpenGL ES 1.1 Emulator User Guide*. It provides guidelines for using the OpenGL ES 1.1 Emulator to develop 2D and 3D graphics applications that are targeted to run on embedded platforms. This book is part of a suite belonging to the Mali Developer Tools.

Intended audience

This guide is written for system integrators and software developers using a PC to develop OpenGL ES 1.1 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 OpenGL ES 1.1 Emulator.

Chapter 2 *Installation and Configuration for Windows*

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

Chapter 3 *Installation and Configuration for Linux*

Read this for a description on how to install and configure the emulator on Linux.

Chapter 4 *Implementation information*

Read this for information about the implementation of the emulator.

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

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.

<code>monospace</code>	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
<code>monospace italic</code>	Denotes arguments to monospace text where the argument is to be replaced by a specific value.
<code>monospace bold</code>	Denotes language keywords when used outside example code.
<code>< and ></code>	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 Developer Tools Technical Overview* (ARM DUI 501)
- *Mali GPU Performance Analysis Tool User Guide* (ARM DUI 0502)
- *Mali GPU Texture Compression Tool User Guide* (ARM DUI 0503)
- *OpenGL ES 2.0 Emulator User Guide* (ARM DUI 0511).

Other publications

This section lists relevant documents published by third parties:

- *OpenGL ES 1.1 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.

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 0506A
- 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 OpenGL ES 1.1 Emulator and describes how to start using the tool in your particular workflow.

It contains the following section:

- *About the OpenGL ES 1.1 Emulator* on page 1-2.

1.1 About the OpenGL ES 1.1 Emulator

The OpenGL ES 1.1 Emulator is a library that maps OpenGL ES 1.1 API calls to the OpenGL 2.0 API.

By running on a standard PC, the emulator helps in development and testing of OpenGL ES applications because no embedded platform is required. The emulator requires a graphics card that supports at least OpenGL 2.0.

Chapter 2

Installation and Configuration for Windows

This chapter provides information about installing and configuring the OpenGL ES 1.1 Emulator on Windows. It contains the following sections:

- *Installing the OpenGL ES 1.1 Emulator on Windows* on page 2-2
- *Installing the Khronos header files* on page 2-6
- *Configuring the OpenGL ES 1.1 Emulator* on page 2-8
- *Building the OpenGL ES 1.1 examples for Windows* on page 2-12.

2.1 Installing the OpenGL ES 1.1 Emulator on Windows

The OpenGL ES 1.1 Emulator is provided for two *Operating Systems* (OS):

- Microsoft Windows
- Red Hat Enterprise Linux.

The installation procedure varies depending upon the OS being used. This chapter describes the installation on a Windows OS.

Note

The OpenGL ES 1.1 Emulator has been tested successfully on a 32-bit computer.

2.1.1 Supported Hardware and Software

The OpenGL ES 1.1 Emulator, Windows version, has been tested with the following hardware and software:

- The OpenGL ES 1.1 Emulator:
 - Windows XP Professional, version 2002, service pack 2
 - NVIDIA GeForce 8400 GS graphics card with driver version 6.14.11.7519
 - ATI Radeon X300/X550/X1050 Series graphics card with driver version 8.522.0.0.

Note

The graphics cards and driver versions are recommendations. The emulator typically also works with other graphics cards and driver versions, provided they support OpenGL 2.0 or above with appropriate extensions.

The minimum appropriate extensions are:

- WGL_ARB_extensions_string
- WGL_ARB_pixel_format
- WGL_ARB_pbuffer
- WGL_ARB_render_texture
- EXT_framebuffer_object.

Wherever possible, update your drivers to the latest version.

-
- The Windows versions of the OpenGL ES 1.1 Emulator was built with Microsoft Visual Studio 2005 and links with applications developed using Microsoft Visual Studio 2005.

Determining the driver version for the video card

To determine the NVIDIA driver version:

1. Click Right on the desktop to open the NVIDIA Control Panel.
2. In the NVIDIA Control Panel, under the Help menu, select **System Information**.
3. Select the **Display** tab.
4. The version number appears as ForceWare version.

To determine the ATI driver version:

1. Click Right on the desktop.
2. Select **Catalyst(TM) Control Center**.
3. In the left-hand menu, expand **Information Center**.
4. Under Information Center, select **Graphics Software**.
5. The version number appears as Catalyst(TM) version.

2.1.2 Disk requirements

The OpenGL ES 1.1 Emulator requires a minimum of 2MB disk space.

2.1.3 Installation procedure

This section describes the installation procedure, it contains the following sections:

- *Installing the OpenGL ES 1.1 Emulator on Microsoft Windows*
- *OpenGL ES 1.1 Emulator content* on page 2-4.

Installing the OpenGL ES 1.1 Emulator on Microsoft Windows

To install the OpenGL ES 1.1 Emulator on a Windows system:

1. Go to the Mali Developer Center website at:
<http://www.malideveloper.com>
2. Select the package to download:
OpenGL_ES_1_1_Emulator_WinXP_vm.n.exe
where:
m identifies the major version
n identifies the minor version.
3. Run the OpenGL_ES_1_1_Emulator_WinXP_vm.n.exe file by double clicking on it.
4. Select the installation options and then click **Finish** to complete the installation.

By default, the Mali OpenGL ES 1.1 Emulator is installed in:

C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator *vm.n*\

OpenGL ES 1.1 Emulator content

The downloaded package contains the OpenGL ES 1.1 Emulator Windows binaries along with two simple example applications based on OpenGL ES 1.1, which run on the OpenGL ES 1.1 Emulator.

Figure 2-1 shows the directory structure that is created at the path where you have installed the emulator, by default C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator *vm.n*\

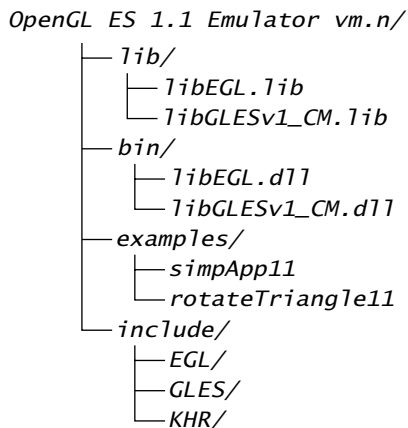


Figure 2-1 OpenGL ES 1.1 Emulator directory structure

Table 2-1 shows the OpenGL ES 1.1 Emulator directory file contents.

Table 2-1 OpenGL ES 1.1 Emulator directory file contents

Filename	Description
lib\libEGL.lib	Import library for EGL implementation
lib\libGLESv1_CM.lib	Import library for OpenGL ES 1.1 Emulator
bin\libEGL.dll	DLL for EGL implementation
bin\libGLESv1_CM.dll	DLL for OpenGL ES 1.1 Emulator
examples\simpApp11\simpApp11.c	Application source code for simpApp11 example

Table 2-1 OpenGL ES 1.1 Emulator directory file contents (continued)

Filename	Description
examples\simpApp11\Makefile	Makefile to build the simpApp11 example
examples\simpApp11\simpApp11.png	Reference image for simpApp11 example
examples\rotateTriangle11\Makefile	Makefile to build the rotateTriangle11 example
examples\rotateTriangle11\main.c	Application source code for rotateTriangle11 example
include\EGL\eglplatform.h	Header file for EGL native types for OpenGL ES 1.1 Emulator on Windows XP
include\GLES\	Directory for OpenGL ES 1.1 Khronos headers
include\KHR\KHRplatform.h	Header file for OpenGL ES native types for OpenGL ES 1.1 Emulator on Windows XP

2.2 Installing the Khronos header files

The Khronos header files are not included in this release for copyright reasons. The header files must be downloaded from the Mali Developer Center website.

———— **Note** ————

The Khronos headers must be in place to build the supplied examples or your own applications.

2.2.1 Khronos header files for OpenGL ES 1.1

The following sections describe the OpenGL ES 1.1 and EGL 1.3 Khronos headers required for this release of the OpenGL ES 1.1 Emulator.

OpenGL ES 1.1 header files

The following OpenGL ES 1.1 header files must be downloaded from the Mali Developer Center website:

<http://www.malideveloper.com>

- 1.1/gl.h
- 1.1/glplatform.h
- 1.1/gltext.h

Copy these headers into the directory:

C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator *vm.n*\include\GLES

EGL header files

The following EGL header files must be downloaded from the Mali Developer Center website:

<http://www.malideveloper.com>

- egl.h
- egltext.h

Copy these headers into the directory:

C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator *vm.n*\include\EGL

The original include directory is shown in Figure 2-1 on page 2-4. Figure 2-2 on page 2-7 shows the contents of the directory after the header files are copied to it.

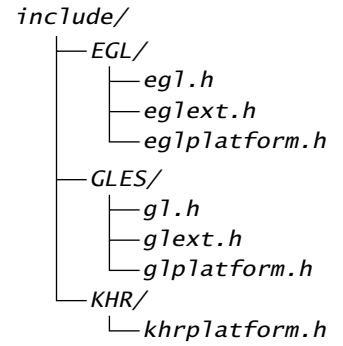


Figure 2-2 OpenGL ES 1.1 include directory

2.3 Configuring the OpenGL ES 1.1 Emulator

This section provides information for configuring your system to use the OpenGL ES 1.1 Emulator when the installation is complete. It contains following sections:

- *Using the OpenGL ES 1.1 Emulator*
- *OpenGL ES 1.1 Emulator integration on page 2-9.*

2.3.1 Using the OpenGL ES 1.1 Emulator

To run any OpenGL ES 1.1 application on the OpenGL ES 1.1 Emulator, you must link it to the OpenGL ES 1.1 Emulator libraries. These libraries, `libEGL.lib` and `libGLESv1_CM.lib` are provided as static libraries, and are linked while building an OpenGL ES 1.1 application. The dynamically linked libraries, `libEGL.dll` and `libGLESv1_CM.dll`, must be provided at run-time.

At run-time, the dlls are provided through the environment variable `Path`. The Windows installer creates an environment variable `EMUBIN11` which points to the directory where the emulator dlls are installed. If the emulator has been installed in the default location, `EMUBIN11` is set to:

```
C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator v $m.n$ \bin\
```

Therefore you must add the variable `EMUBIN11` to the system environment variable `Path` after the installation is complete on your Windows PC. You only have to do this once on a system. Subsequent installations or un-installations are taken care of by the system.

To add the environment variable `EMUBIN11` to system environment variable `Path`:

1. Open the System Properties dialog by selecting **Start** → **Control Panel** → **System**
2. Click on the **Advanced tab**
3. In the **Advanced tab**, click on the **Environment variables** button
4. Select **Path** in the system variables
5. Click on **Edit**
6. Go to end of the line and add:
 ;`%EMUBIN11%`
7. Click **OK** three times to close three levels of dialog:
 - Edit System Variable
 - Environment Variables
 - System Properties.

The system path now includes the OpenGL ES 1.1 binaries required for running OpenGL ES 1.1 applications.

———— **Caution** ————

There can be problems caused by running applications on the OpenGL ES 1.1 Emulator if the OpenGL ES 2.0 Emulator is also installed on the PC. To avoid problems, you must do the following before running an application on an emulator:

- If you want to run an application on the OpenGL ES 1.1 Emulator, ensure that the system environment variable Path contains the string %EMUBIN11% before the string %EMUBIN20%
- If you want to run an application on the OpenGL ES 2.0 Emulator, ensure that the system environment variable Path contains the string %EMUBIN20% before the string %EMUBIN11%.

This makes the application use the DLLs of the appropriate emulator at runtime.

For details on building the examples, see *Building the OpenGL ES 1.1 examples for Windows* on page 2-12.

2.3.2 OpenGL ES 1.1 Emulator integration

This section describes:

- *DLLs and libraries*
- *EGL configuration* on page 2-10
- *EGL context creation* on page 2-11.

DLLs and libraries

The OpenGL ES 1.1 Emulator consists of two DLLs, corresponding to the separate OpenGL ES 1.1 and EGL 1.3 APIs. For each of these DLLs, there is a corresponding import library for the OpenGL ES 1.1 application to statically link against. Applications must include both of the import libraries in builds to link against the OpenGL ES 1.1 and EGL 1.3 APIs. The DLLs use the `__stdcall` calling convention.

Table 2-2 shows the OpenGL ES 1.1 emulation library structure

Table 2-2 OpenGL ES 1.1 Emulator library structure

Filename	Description
bin\libGLESv1_CM.dll	DLL for OpenGL ES 1.1 Emulator
bin\libEGL.dll	DLL for EGL implementation
lib\libGLESv1_CM.lib	Import library for libGLESv1_CM.dll
lib\libEGL.lib	Import library for libEGL.dll

EGL configuration

The EGL library supplied with the OpenGL ES 1.1 Emulator supports OpenGL ES 1.1 configurations only. Ensure that in the OpenGL ES 1.1 application, the attribute list passed as a parameter to `eglChooseConfig` includes the attribute `EGL_RENDERABLE_TYPE` set to the value `EGL_OPENGL_ES_BIT`.

Example 2-1 shows a coded section.

Example 2-1

```

EGLDisplay Display;

EGLint Attributes[] = {
    EGL_RENDERABLE_TYPE, EGL_OPENGL_ES_BIT,
    EGL_RED_SIZE, 8,
    EGL_GREEN_SIZE, 8,
    EGL_BLUE_SIZE, 8,
    EGL_NONE
};

EGLConfig Configs[1];

EGLint NumConfigs;

...

eglChooseConfig(Display, Attributes, Configs, 1, &NumConfigs);

```

EGL context creation

The EGL library supplied with the OpenGL ES 1.1 Emulator supports OpenGL ES 1.1 contexts only. It is important to ensure that, in the OpenGL ES 1.1 application, the attribute list passed as a parameter to `eglCreateContext` includes the attribute `EGL_CONTEXT_CLIENT_VERSION` set to the value 1.

Example 2-2 shows a coded section.

Example 2-2

```
EGLDisplay Display;  
  
EGLConfig Configs[1];  
  
EGLint ContextAttributes[] = {  
    EGL_CONTEXT_CLIENT_VERSION, 1,  
    EGL_NONE  
};  
  
...  
  
Context = eglCreateContext(Display, Configs[0], EGL_NO_CONTEXT,  
                           ContextAttributes);
```

2.4 Building the OpenGL ES 1.1 examples for Windows

This section describes two examples:

- *simpApp11 example*
- *rotateTriangle11 example* on page 2-13.

———— Note ————

Throughout this section, it is assumed that the OpenGL ES 1.1 Emulator is installed at its default path:

C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator *vm.n*

where:

- m* identifies the major version
- n* identifies the minor version.

2.4.1 simpApp11 example

The simpApp11 example code for using the OpenGL ES 1.1 Emulator is included in the directory C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator *vm.n*\examples\simpApp11

To build and run this example:

1. Ensure that OpenGL ES 1.1 Emulator is installed.
2. Ensure that the OpenGL ES 1.1 and EGL header files have been added to the C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator *vm.n*\include directory. This is described in *Installing the Khronos header files* on page 2-6.
3. Ensure that the environment variable EMUBIN11 is added to the system environment variable Path. This is described in *Using the OpenGL ES 1.1 Emulator* on page 2-8.
4. The next steps involve using Microsoft Visual Studio 2005 to build and run the application:
 - a. Select **Start** → **All Programs** → **Microsoft Visual Studio 2005** → **Visual Studio Tools** → **Visual Studio 2005 Command Prompt**
 - b. Change directory to where the example is present, that is:

C:\Program Files\ARM\Mali Developer Tools\Mali OpenGL ES 1.1 Emulator *vm.n*\examples\simpApp11

- c. To build the example application, type the following command at the command prompt:

nmake

- d. To run the example application, type the following command at the command prompt:
simpApp11
- e. An additional window with a spinning, colored cube appears when the example application is running. Figure 2-3 shows an image.

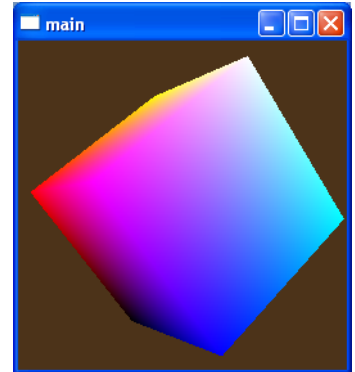


Figure 2-3 simpApp11 image

- f. To end the program, close this window and press **Ctrl+C**.

2.4.2 rotateTriangle11 example

The rotateTriangle11 example code for using the OpenGL ES 1.1 Emulator library is included in the directory C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator vm.n\examples\rotateTriangle11

To build and run this example:

1. Ensure that Mali OpenGL ES 1.1 Emulator is installed.
2. Ensure the OpenGL ES 1.1 and EGL header files have been added to the C:\Program Files\ARM\Mali Developer Tools\OpenGL ES 1.1 Emulator vm.n\include directory. This is described in *Installing the Khronos header files* on page 2-6.
3. Ensure the environment variable EMUBIN11 is added to the system environment variable Path as described in *Using the OpenGL ES 1.1 Emulator* on page 2-8.

4. The next steps involve using Microsoft Visual Studio 2005. This can be achieved by selecting the following application from the start menu to build and run the application:

- a. Select **Start** → **All Programs** → **Microsoft Visual Studio 2005** → **Visual Studio Tools** → **Visual Studio 2005 Command Prompt**

- b. Change directory to where the example is present, that is,:

C:\Program Files\ARM\Mali Developer Tools\Mali OpenGL ES 1.1 Emulator\vm.n\examples\rotateTriangle11

- c. To build the example application, type the following command at the command prompt:

nmake

- d. To run the built example application, type the following command at the command prompt:

main

- e. An additional window with a rotating, red colored triangle appears when the example application is running. Figure 2-4 shows the image.

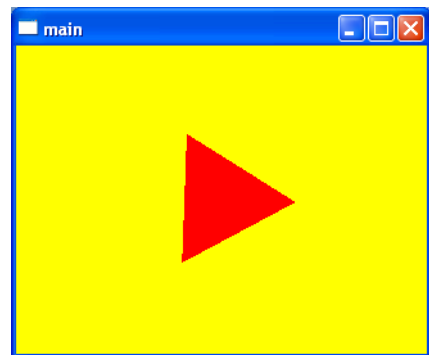


Figure 2-4 rotateTriangle11 image

- f. To end the program, close this window.

Chapter 3

Installation and Configuration for Linux

This chapter provides information about installing and configuring the OpenGL ES 1.1 Emulator on Linux. It contains the following sections:

- *Installing the OpenGL ES 1.1 Emulator on Linux* on page 3-2
- *Installing the Khronos header files* on page 3-5
- *Configuring the OpenGL ES 1.1 Emulator* on page 3-7
- *Building the OpenGL ES 1.1 examples for Linux* on page 3-10.

3.1 Installing the OpenGL ES 1.1 Emulator on Linux

The OpenGL ES 1.1 Emulator is provided for the following *Operating Systems* (OS):

- Microsoft Windows
- Red Hat Enterprise Linux.

The installation procedure varies depending upon the OS. This chapter describes the installation on Linux OS.

———— **Note** —————

The OpenGL ES 1.1 Emulator has been tested successfully on a 32-bit computer.

3.1.1 Supported Hardware and Software

The OpenGL ES 1.1 Emulator, Linux version, has been tested with the following hardware and software:

- Red Hat Enterprise Linux release 4
- NVIDIA GeForce 8400 GS graphics card with driver version 180.44.

———— **Note** —————

The graphics card and driver version are recommendations. The emulator typically also works with other graphics cards and driver versions, provided they support OpenGL 2.0 or above, with appropriate extensions. The platform must also support GLX 1.4. Wherever possible, update your drivers to the latest version.

NVIDIA driver version

To determine the NVIDIA driver version on a Linux machine:

1. Open the terminal, and type the following command:
`nvidia-settings`
2. A dialog box is opened with the card and driver details.

3.1.2 Disk requirements

The OpenGL ES 1.1 Emulator requires a minimum of 1.5MB disk space.

3.1.3 Installation procedure

This section describes the installation procedure, it contains the following sections:

- *Installing the OpenGL ES 1.1 Emulator on Linux*
- *OpenGL ES 1.1 Emulator content.*

Installing the OpenGL ES 1.1 Emulator on Linux

To install the OpenGL ES 1.1 Emulator on a Linux system:

1. Go to the Mali Developer Center website at:
`http://www.malideveloper.com`
2. Select the package to download:
`OpenGL_ES_1_1_Emulator_RHEL4_vm.n.tar.gz`

———— **Note** —————

where:

- m*** identifies the major version
n identifies the minor version.
-

3. To decompress the file, type the following command:
`tar -zxvf OpenGL_ES_1_1_Emulator_RHEL4_vm.n.tar.gz`

———— **Note** —————

You must use GNU tar version 1.16, or a later version, to untar the deliverables, because many versions of tar have problems dealing with very long path names. To find the version of tar being used type `tar --version`

After decompressing, the Mali Developer Tools are installed in:

`ARM/Mali_Developer_Tools`

By default, the OpenGL ES 1.1 Emulator is installed in:

`ARM/Mali_Developer_Tools/OpenGL_ES_1_1_Emulator_vm.n`

OpenGL ES 1.1 Emulator content

Figure 3-1 on page 3-4 shows the directory structure that is created at the path where you have installed the emulator.

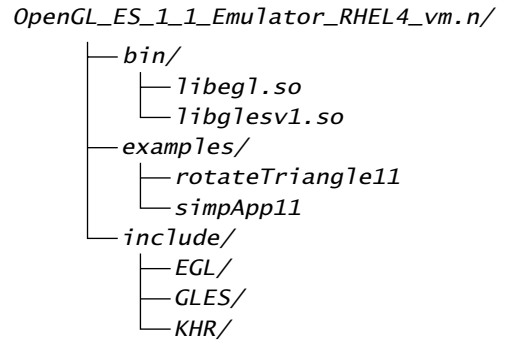


Figure 3-1 OpenGL ES 1.1 Emulator directory structure

Table 3-1 shows the OpenGL ES 1.1 Emulator directory file contents.

Table 3-1 OpenGL ES 1.1 Emulator directory file contents

Directory	Description
bin	OpenGL ES 1.1 Emulator shared libraries
examples/rotateTriangle11	Source code for a simple application that shows a rotating triangle
examples/simpApp11	Source code for a simple application that shows a rotating cube
include	Placeholders for Khronos OpenGL ES 1.1 headers <ul style="list-style-type: none"> • EGL • GLES • KHR

3.2 Installing the Khronos header files

The Khronos header files are not included in this release for copyright reasons. The header files must be downloaded from the Mali Developer Center website.

———— **Note** —————

The Khronos headers must be in place to build the supplied examples or your own applications.

3.2.1 Khronos header files for OpenGL ES 1.1

The following sections describe the OpenGL ES 1.1 and EGL 1.3 headers required for this release of the OpenGL ES 1.1 Emulator.

OpenGL ES 1.1 header files

The following OpenGL ES 1.1 header files must be downloaded from the Mali Developer Center website at <http://www.malideveloper.com>:

- 1.1/gl.h
- 1.1/glplatform.h
- 1.1/gltext.h

Copy these headers into the directory:

<installation root directory for OpenGL ES 1.1 Emulator>/include/GLES

EGL header files

The following EGL header files must be downloaded from the Mali Developer Center website at <http://www.malideveloper.com>:

- egl.h
- egltext.h

Copy these headers into the directory:

<installation root directory for OpenGL ES 1.1 Emulator>/include/EGL

The original include directory is shown in Figure 3-1 on page 3-4. Figure 3-2 on page 3-6 shows the contents of the directory after the header files have been copied to it.

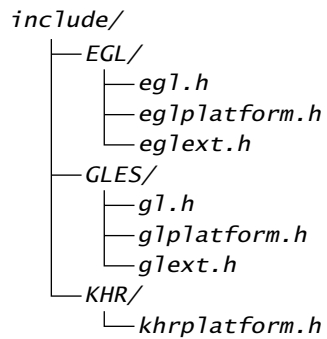


Figure 3-2 OpenGL ES 1.1 include directory

3.3 Configuring the OpenGL ES 1.1 Emulator

This section provides information to configure your system to use the OpenGL ES 1.1 Emulator when the installation is complete. It contains the following sections:

- *Using the OpenGL ES 1.1 Emulator*
- *OpenGL ES 1.1 Emulator integration.*

3.3.1 Using the OpenGL ES 1.1 Emulator

To run any OpenGL ES 1.1 application on the OpenGL ES 1.1 Emulator, you must link it to the OpenGL ES 1.1 Emulator libraries.

Before you can run the application, you must add the path of the Emulator libraries to the system environment variable `LD_LIBRARY_PATH`.

The command to do this using the bash Linux shell is:

```
export LD_LIBRARY_PATH=<installation root directory for OpenGL ES 1.1 Emulator>/bin
```

The command to do this using the tsch Linux shell is:

```
setenv LD_LIBRARY_PATH <installation root directory for OpenGL ES 1.1 Emulator>/bin
```

The system library search path now includes the OpenGL ES 1.1 Emulator libraries required to run OpenGL ES 1.1 applications.

For details on building the examples, see *Building the OpenGL ES 1.1 examples for Linux* on page 3-10.

3.3.2 OpenGL ES 1.1 Emulator integration

This section describes:

- *Libraries*
- *EGL configuration* on page 3-8
- *EGL context creation* on page 3-9.

Libraries

The OpenGL ES 1.1 Emulator contains two libraries corresponding to the separate OpenGL ES 1.1 and EGL 1.3 APIs.

Table 3-2 shows the OpenGL ES 1.1 emulation library structure.

Table 3-2 OpenGL ES 1.1 Emulator library structure

Filename	Description
bin\libglesv1.so	Library for OpenGL ES 1.1 API
bin\libegl.so	Library for EGL 1.3 API

EGL configuration

The EGL library supplied with the OpenGL ES 1.1 Emulator supports OpenGL ES 1.1 configurations only.

———— **Note** ————

Ensure that in the OpenGL ES 1.1 application, the attribute list passed as a parameter to `eglChooseConfig` includes the attribute `EGL_RENDERABLE_TYPE` set to the value `EGL_OPENGL_ES_BIT`.

Example 3-1 shows a coded section:

Example 3-1

```

EGLDisplay Display;

EGLint Attributes[] = {
    EGL_RENDERABLE_TYPE, EGL_OPENGL_ES_BIT,
    EGL_RED_SIZE, 8,
    EGL_GREEN_SIZE, 8,
    EGL_BLUE_SIZE, 8,
    EGL_NONE
};
EGLConfig Configs[1];

EGLint NumConfigs;

...

eglChooseConfig(Display, Attributes, Configs, 1, &NumConfigs);
    
```

EGL context creation

The EGL library supplied with the OpenGL ES 1.1 Emulator supports OpenGL ES 1.1 contexts only.

Ensure that, in the OpenGL ES 1.1 application, the attribute list passed as a parameter to `eglCreateContext` includes the attribute `EGL_CONTEXT_CLIENT_VERSION` set to the value 1.

Example 3-2 shows a coded section:

Example 3-2

```
EGLDisplay Display;

EGLConfig Configs[1];

EGLint ContextAttributes[] = {
    EGL_CONTEXT_CLIENT_VERSION, 1,
    EGL_NONE
};

...

Context = eglCreateContext(Display, Configs[0], EGL_NO_CONTEXT,
                          ContextAttributes);
```

3.4 Building the OpenGL ES 1.1 examples for Linux

The following sections describe the examples:

- *simpApp11 example*
- *rotateTriangle11 example* on page 3-11.

———— **Note** —————

The Khronos headers must be installed to build the examples.

3.4.1 simpApp11 example

The simpApp11 example code for using the OpenGL ES 1.1 Emulator is included in the directory:

```
<installation root directory for OpenGL ES 1.1 Emulator>/examples/simpApp11
```

To build and run this example:

1. Ensure that the OpenGL ES 1.1 and EGL header files have been added to the include directory. This is described in *Installing the Khronos header files* on page 3-5.
2. Ensure that the system environment variable LD_LIBRARY_PATH is set to the directory containing the OpenGL ES 1.1 libraries. This is described in *Using the OpenGL ES 1.1 Emulator* on page 3-7.

3. Navigate to the directory where the example is present:

```
cd <installation root directory for OpenGL ES 1.1 Emulator>/examples/simpApp11
```

4. Use the make command to build the simpApp11 application

```
make
```

5. To run the example application, type the following command:

```
./simpApp11
```

An additional window with a spinning colored cube appears when the example application is running. Figure 3-3 on page 3-11 shows an image.

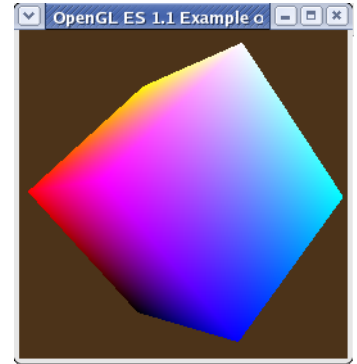


Figure 3-3 simpApp11 image

To end the program, close this window and press **Ctrl+C**

3.4.2 rotateTriangle11 example

The rotateTriangle11 example code for using the OpenGL ES 1.1 Emulator is included in the directory:

```
<installation root directory for OpenGL ES 1.1 Emulator>/examples/rotateTriangle11
```

To build and run this example:

1. Ensure the OpenGL ES 1.1 and EGL header files have been added to the include directory. This is described in *Installing the Khronos header files* on page 3-5.
2. Ensure that the system environment variable LD_LIBRARY_PATH is set to the directory containing the OpenGL ES 1.1 libraries. See *Using the OpenGL ES 1.1 Emulator* on page 3-7.
3. Navigate to the directory where the example is present:

```
cd <installation root directory for OpenGL ES 1.1 Emulator>/examples/rotateTriangle11
```
4. Use the make command to build the rotateTriangle11 application:

```
make
```
5. To run the example application, type the following command:

```
./main
```

An additional window with a rotating, red colored triangle appears when the example application is running as shown in Figure 3-4 on page 3-12.

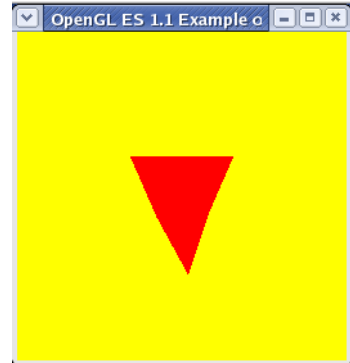


Figure 3-4 rotateTriangle11 image

To end the program, close this window and press **Ctrl+C**.

Chapter 4

Implementation information

This chapter provides implementation information about the OpenGL ES 1.1 and EGL APIs in the emulator. It contains the following sections:

- *OpenGL ES 1.1 implementation information* on page 4-2
- *EGL implementation information for Windows* on page 4-4
- *EGL implementation information for Linux* on page 4-9.

4.1 OpenGL ES 1.1 implementation information

The OpenGL ES 1.1 Emulator converts OpenGL ES 1.1 API calls to OpenGL 2.0 API calls. The OpenGL 2.0 calls are handled by the Windows graphics drivers.

The API conversion checks OpenGL ES 1.1 parameters and rejects invalid OpenGL ES 1.1 parameter values which would otherwise be accepted by OpenGL 2.0 because of the differences between OpenGL ES 1.1 and OpenGL 2.0 specifications.

The OpenGL ES 1.1 Emulator depends on the functionality of the OpenGL 2.0 implementation provided by the graphics card drivers. In some cases, this dependency can result in limitations in the OpenGL ES 1.1 implementation. This occurs when the behavior of the graphics card drivers differs from OpenGL ES 1.1 specification.

4.1.1 General limitations

The following sections describe the general limitations of the OpenGL ES 1.1 Emulator:

- *Implementation-specific behavior*
- *Fixed-point data gives reduced performance*
- *Compressed texture formats* on page 4-3
- *Vertex buffer objects performance* on page 4-3
- *Multiple threads and multiple contexts* on page 4-3.

Implementation-specific behavior

Wherever the OpenGL ES 1.1 specification permits implementation-specific behavior, that behavior is typically determined by the underlying driver. The behavior of the graphics card drivers can differ from the behavior of Mali drivers and hardware. This includes implementation-dependent limits such as, for example:

- texture sizes
- extensions
- mipmap level calculation
- framebuffers.

Fixed-point data gives reduced performance

OpenGL 2.0 does not provide support for fixed-point data, but this is required by the OpenGL ES 1.1 specification. The OpenGL ES 1.1 Emulator converts fixed-point data and passes it to OpenGL 2.0. This means, for the OpenGL ES 1.1 Emulator, fixed-point data gives lower performance when compared to using floating point data. The effect is

stronger if you use a client-side vertex array rather than a vertex buffer object. This is because the OpenGL ES 1.1 Emulator must convert a client-side vertex on each draw call, in case the client application modifies the data between draw calls.

Compressed texture formats

GL_OES_compressed_ETC1_RGB8_texture is supported, but it is treated internally as RGB. It is possible therefore to mix this format with uncompressed RGB texture data in ways that can cause an error or an incomplete texture when used with Mali GPU drivers.

Vertex buffer objects performance

Vertex buffer objects are supported by OpenGL ES 1.1 Emulator and are implemented in software. Vertex buffer objects therefore follow the behavior specified in the OpenGL ES 1.1 specification rather than the behavior of the underlying OpenGL 2.0 driver. This might affect performance.

Multiple threads and multiple contexts

Multiple threads and multiple contexts are not supported and might cause unpredictable behavior.

4.1.2 EGL limitations

EGL has been implemented on top of GLX 1.4. The EGL implementation supports OpenGL ES 1.1 only. It does not support OpenGL ES 2.0 or OpenVG. For more information see:

- *EGL implementation information for Windows* on page 4-4
- *EGL implementation information for Linux* on page 4-9.

4.2 EGL implementation information for Windows

This section describes:

- *EGL native types*
- *Display initialization*
- *Default display example*
- *Window example* on page 4-5
- *Creation of window surface* on page 4-5
- *Creation of pixmap surfaces* on page 4-5
- *Creation of Pbuffer surfaces* on page 4-5
- *Synchronization of pixmap surfaces* on page 4-5
- *EGL limitations* on page 4-6.

4.2.1 EGL native types

The EGL implementation defines the platform specific native types as follows:

```
typedef HDC NativeDisplayType;  
  
typedef HBITMAP NativePixmapType;  
  
typedef HWND NativeWindowType;
```

These are defined in the file `include\EGL\egl_native_types.h`

4.2.2 Display initialization

The `eglGetDisplay` call expects to be passed either the value `EGL_DEFAULT_DISPLAY` or the *Handle* of the *Device Context* (HDC) for the window created to display the rendered output from the OpenGL ES 1.1 Emulator in an OpenGL ES 1.1 application.

Default display example

Example 4-1 shows a default display:

Example 4-1

```
EGLDisplay sEGLDisplay;  
  
// EGL init.  
sEGLDisplay = eglGetDisplay((EGLNativeDisplayType) EGL_DEFAULT_DISPLAY);  
eglInitialize(sEGLDisplay, NULL, NULL);
```

Window example

Example 4-2 shows a default display:

Example 4-2

```
EGLDisplay sEGLDisplay;

// Window
...

// Create Window
Window = CreateWindowEx(...)

// EGL init.
sEGLDisplay = eglGetDisplay(GetDC(sWindow));
eglInitialize(sEGLDisplay, NULL, NULL);
```

4.2.3 Creation of window surface

For an example of the code to create a window surface on OpenGL ES 1.1 Emulator, see file `examples\simpApp11\simpApp11.c`.

4.2.4 Creation of pixmap surfaces

To access data bits of a Windows bitmap in the EGL API, you must pass the bitmap to EGL as a native pixmap. You must create this pixmap with the Windows API call `CreateDIBSection()`. This is to enable access to the bitmap data bits.

4.2.5 Creation of Pbuffer surfaces

A Pbuffer has no associated native structure and is created through the specification of attributes to `eglCreatePbufferSurface`. No platform specific code is required.

4.2.6 Synchronization of pixmap surfaces

Pixmap surfaces are supported through the use of graphics driver Pbuffers. To get the correct behavior of OpenGL ES 1.1 rendering on to the native pixmap, the appropriate EGL synchronization calls must be used. This corresponds to the expected use of these calls in the EGL 1.3 specification.

The call `eglWaitNative(EGL_CORE_NATIVE_ENGINE)` copies bitmap data from the native bitmap to the graphics driver Pbuffer before the OpenGL ES 1.1 API calls are made to render to the Pbuffer. The calls `eglWaitClient()`, `eglWaitGL()` and `glFinish()` copy data back from the graphics driver Pbuffer to the native pixmap after rendering by OpenGL ES 1.1.

———— **Note** —————

The native bitmap must not be selected into a device context. The application fails when calling `eglWaitNative` and the call fails.

4.2.7 EGL limitations

The EGL library intends to provide sufficient functionality for the OpenGL ES 1.1 Emulator to pass Khronos OpenGL ES 1.1 conformance tests and to provide a platform for running OpenGL ES 1.1 applications on a Windows PC. The EGL library is a limited implementation of the EGL 1.3 specification. This section provides additional information about these limitations:

- *Multiple threads and multiple contexts*
- *Window pixel format*
- *Limited bitmap support on page 4-7*
- *Limited results from surface queries on page 4-7*
- *No support for swap intervals on page 4-7*
- *Changing display modes on page 4-7*
- *Use of displays following `eglTerminate` on page 4-7*
- *`EGL_MATCH_NATIVE_PIXMAP` attribute not supported on page 4-7*
- *Resizing a native window on page 4-7*
- *Pbuffer lost events are not checked on page 4-7*
- *`EglChooseConfig` always sets `WGL_DOUBLE_BUFFER_ARB` true on page 4-8.*

Multiple threads and multiple contexts

Multiple threads and contexts are not supported and might lead to unpredictable behavior.

Window pixel format

You must set pixel format only through `eglCreateWindowSurface()`.

Limited bitmap support

Bitmap rendering only works correctly for uncompressed, bottom-up, 32-bit RGB bitmaps.

Limited results from surface queries

All parameters to `eglQuerySurface` are implemented, but those specific to OpenVG, and those that depend on the physical properties of the display, for example `EGL_HORIZONTAL_RESOLUTION`, return arbitrary values or `EGL_UNKNOWN`.

No support for swap intervals

The `eglSwapInterval` function has no effect and always succeeds. The swap interval depends on the OpenGL 2.0 driver.

Changing display modes

Changing display modes is not supported. For example, this can cause a `Pbuffer lost` event.

Use of displays following `eglTerminate`

Displays are destroyed in `eglTerminate` so following calls treat the display as invalid.

`EGL_MATCH_NATIVE_PIXMAP` attribute not supported

The attribute `EGL_MATCH_NATIVE_PIXMAP` is not supported by `eglChooseConfig`. This defect is documented in the *OpenGL ES 1.1 Emulator Errata*.

Resizing a native window

Resizing a native window does not result in the surface attributes being updated. This defect is documented in the *OpenGL ES 1.1 Emulator Errata*.

Pbuffer lost events are not checked

A change of display mode might result in loss of `Pbuffer` memory. This event is not checked for. This defect is documented in the *OpenGL ES 1.1 Emulator Errata*.

EglChooseConfig always sets WGL_DOUBLE_BUFFER_ARB true

The EGL attribute list is translated to an attribute list for WGL. This WGL attribute list always has the WGL_DOUBLE_BUFFER_ARB set to true. This means that all available matching WGL configurations might not be returned. This defect is documented in the *OpenGL ES 1.1 Emulator Errata*.

4.3 EGL implementation information for Linux

The EGL implementation intends to provide sufficient functionality for the OpenGL ES 1.1 Emulator to pass Khronos OpenGL ES1.1 conformance tests and to provide a platform for running OpenGL ES applications on a Linux PC. The EGL library is a limited implementation of the EGL 1.3 specification. This section provides additional information about these limitations:

- *Unimplemented functions*
- *Resizing a native window*
- *eglChooseConfig always selects configurations that use the back buffer*
- *Some EGLConfig attributes are not supported* on page 4-10
- *EGLConfigs not sorted* on page 4-10
- *Attributes for windows not supported* on page 4-10
- *Some Pbuffer attributes are not supported* on page 4-10
- *Attributes for pixmaps not supported* on page 4-11
- *Incorrect error code returned instead of EGL_BAD_MATCH* on page 4-11
- *Limited results from surface queries* on page 4-11
- *eglMakeCurrent succeeds with incompatible surface and context* on page 4-11.

4.3.1 Unimplemented functions

There are 34 functions in EGL 1.3 specification of which the following six functions are not implemented:

- `eglCreatePbufferFromClientBuffer`
- `eglSurfaceAttrib`
- `eglBindTexImage`
- `eglReleaseTexImage`
- `eglSwapInterval`
- `eglCopyBuffers`

4.3.2 Resizing a native window

Resizing a native window does not result in the surface attributes being updated. This defect is documented in the *OpenGL ES 1.1 Emulator Errata*.

4.3.3 eglChooseConfig always selects configurations that use the back buffer

The EGL specification enables you to specify whether to use the back buffer or not in the attribute list passed to `eglCreateWindowSurface`. But the GLX function for window surface creation does not permit this. The GLX function for choosing configurations permits you to specify whether you want to use a back buffer or not. So EGL uses this

function to select only those configurations which enable use of the back buffer. As a side effect of this, applications can never disable use of the back buffer. This defect is documented in *OpenGL ES 1.1 Emulator Errata*.

4.3.4 Some EGLConfig attributes are not supported

The following EGLConfig attributes are not supported:

- EGL_LUMINANCE_SIZE
- EGL_ALPHA_MASK_SIZE
- EGL_BIND_TO_TEXTURE_RGB
- EGL_BIND_TO_TEXTURE_RGBA
- EGL_COLOR_BUFFER_TYPE
- EGL_MAX_SWAP_INTERVAL
- EGL_MATCH_NATIVE_PIXMAP

———— **Note** —————

- `eglChooseConfig` returns an error if any of the above attributes is specified in the attribute list.
 - `eglGetConfigAttrib` returns an error if any of the above attributes is queried.
-

4.3.5 EGLConfigs not sorted

The list of configurations returned by `eglChooseConfig` is not sorted. This is because EGL and GLX have different sorting criteria.

4.3.6 Attributes for windows not supported

Attributes for windows are not supported. The attribute list passed to `eglCreateWindowSurface` must be NULL or empty.

4.3.7 Some Pbuffer attributes are not supported

`eglCreatePbufferSurface` returns an error if any of the following attributes are specified in the attribute list:

- EGL_TEXTURE_FORMAT
- EGL_TEXTURE_TARGET
- EGL_MIPMAP_TEXTURE
- EGL_VG_COLORSPACE
- EGL_VG_ALPHA_FORMAT

4.3.8 Attributes for pixmaps not supported

Attributes for pixmaps are not supported. The attribute list passed to `eglCreatePixmapSurface` must be NULL or empty.

4.3.9 Incorrect error code returned instead of EGL_BAD_MATCH

Sometimes, instead of EGL_BAD_MATCH, EGL returns an incorrect error code. This happens because GLX does not have an error code corresponding to EGL_BAD_MATCH and EGL is not always able to detect the real cause of the error.

4.3.10 Limited results from surface queries

`eglQuerySurface` returns an error if any of the following attributes are queried:

- EGL_VG_ALPHA_FORMAT
- EGL_VG_COLORSPACE
- EGL_HORIZONTAL_RESOLUTION
- EGL_MIPMAP_TEXTURE
- EGL_MIPMAP_LEVEL
- EGL_PIXEL_ASPECT_RATIO
- EGL_RENDER_BUFFER
- EGL_TEXTURE_FORMAT
- EGL_TEXTURE_TARGET
- EGL_VERTICAL_RESOLUTION

4.3.11 eglMakeCurrent succeeds with incompatible surface and context

On some platforms, `eglMakeCurrent` succeeds with incompatible surface and context. This might not happen on other platforms.

Glossary

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

- 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.
- Attribute aliasing** The processes whereby applications are permitted to bind more than one vertex shader attribute name to the same generic vertex attribute index.
- Debugger** A debugging system that includes a program, used to detect, locate, and correct software faults, together with custom hardware that supports software debugging.
- Display subsystem** The display that a final image is viewed on and the associated software that controls the operation of the display.
- Ericsson Texture Compression (ETC)**
A 4 *bit-per-pixel* (bpp) texture compression algorithm.
- ESSL** *See* OpenGL ES Shading Language.

ETC	<i>See</i> Ericsson Texture Compression.
Fragment shader	A program running on the pixel processor that calculates the color and other characteristics of each fragment.
Framebuffer	A memory buffer containing a complete frame of graphical data.
GPU	<i>See</i> Graphics Processor Unit.
Graphics application	A custom program that runs on the platform CPU and uses the Mali GPU to display graphics.
Graphics driver	A software library implementing OpenGL ES or OpenVG, using graphics accelerator hardware. <i>See also</i> OpenGL ES driver.
Graphics Processor Unit (GPU)	A hardware accelerator for graphics systems using OpenGL ES and OpenVG. The hardware accelerator comprises of an optional geometry processor and a pixel processor together with memory management. Mali programmable GPUs, such as the Mali-200 and Mali-400 MP GPUs, consist of a geometry processor and at least one pixel processor. Mali fixed-function GPUs, such as the Mali-55 GPU consist of a pixel processor only.
Mali	A name given to graphics software and hardware products from ARM that aid 2D and 3D acceleration.
Mali Developer Tools	A set of development programs that enables software developers to create graphic applications.
OpenGL ES driver	On graphics systems that use the OpenGL ES API, the OpenGL ES driver is a specialized driver that controls the graphics hardware.
OpenGL ES Shading Language (ESSL)	A programming language used to create custom shader programs that can be used within a programmable pipeline, on graphics hardware. You can also use pre-defined library shaders, written in ESSL.
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 counter	A register in the processor that is incremented when a certain state is detected in the processor. Performance counter data is represented as performance variables that are displayed in the Performance Analysis Tool.

Performance data file

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

Performance variable

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

Red Green Blue Alpha (RGBA)

An implementation of the RGB color model, that includes an Alpha value to indicate opacity. If a fragment has an Alpha value of 0%, it is fully transparent, making it invisible. An Alpha value of 100% results in a fully opaque pixel.

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 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.

