How to test and benchmark OpenGLES

Contents

Stable: 13.11.2020 - 16:53 / Revision: 13.11.2020 - 14:45

Stable: 13.11.2020 - 10:30 / Revision: 13.11.2020 - 14:48

A quality version of this page, approved on *13 November 2020*, was based off this revision.

Contents

1 Tests

1.1 Conformance tests

- How to launch Khronos OpenGLES conformance tests

2 Benchmarks

2.1 Linux Benchmarks

- How to launch glmark2 benchmark

Stable: 03.10.2019 - 13:22 / Revision: 26.09.2019 - 12:54

A quality version of this page, approved on *3 October 2019*, was based off this revision.

Contents

1 Introduction

The purpose of this article is to describe how to build and execute the **Khronos OpenGL ES 2.0 Conformance Tests**.

These tests are provided by the Khronos Group [1]. They are available as open source from the **Khronos CTS GitHub source repo** [2].

For a detailed description of the Khronos OpenGL ES2.0 CTS, please refer to the **openglcts README.md** [3] file.

2 How to install khronos-cts

2.1 Building khronos-cts

Execute the following command in the OpenSTLinux build environment:

```
PC $> bitbake khronos-cts
```

2.2 Installing kronos-cts on the target board

Execute the following command in the OpenSTLinux build environment:

```
PC $> scp tmp*/deploy/deb/*neon*/khronos-cts_opengl-cts-<CTS Release>.deb root@<IPBOARD>:
<SomewhereInTheBoard>
```

Information

But default dpkg will extract within /home/root/.

This debian package is rather huge. For instance opengl-cts-4.6.0 is around 115MB. If rootfs space is at sake this is better to unpack the debian somewhere else by appending instdir=<SomewhereElse> to the above command

3 How to execute khronos-cts

3.1 Running test suite

Execute the following commands on the target board:

```
Board $> cd /home/root
Board $> ./cts-runner --type=es2 1>results.txt
[     1] EGL: enable default configs for conformance test
...
```

Information

The test execution can take several hours.

Warning

The *free* system RAM amount required to complete this test suite in around 500MB. Under out of memory occurs.

3.2 Running a single/group of test(s)

```
Board $> cd /home/root
Board $> glcts.exe --deqp-case=dEQP-EGL.functional.*
```

4 Testing verdict example

```
Board $>
...
Test run totals:
  Passed:         13786/13957 (98.8%)
  Failed:         11/13957 (0.1%)
  Not supported: 131/13957 (0.9%)
  Warnings:       29/13957 (0.2%)
219/220 sessions passed, conformance test FAILED
```

In order to interpret result, please look at "Understanding the result"[4]

Even more there are several tools available to process those test logs[5]

5 References

- https://www.khronos.org/
- https://github.com/KhronosGroup/VK-GL-CTS
- https://github.com/KhronosGroup/VK-GL-CTS/blob/master/external/openglcts/README.md#introduction
- https://github.com/KhronosGroup/VK-GL-CTS/tree/master/external/openglcts#understanding-the-results
- https://github.com/KhronosGroup/VK-GL-CTS/tree/master/external/openglcts#test-logs

Open Graphics Library (See http://www.opengl.org/ for more details)

Compatibility Test Suite (Android specific) or Clear to send (in UART context)

Khronos Native Platform Graphics Interface (See http://www.khronos.org/egl/ for more details)

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

Stable: 26.10.2020 - 12:47 / Revision: 23.10.2020 - 12:51

A quality version of this page, approved on *26 October 2020*, was based off this revision.

Contents

1 Introduction

glmark2 *is an OpenGL 2.0 and ES 2.0 benchmark, developed by Alexandros Frantzis and Jesse Barker. It is based on the original glmark benchmark by Ben Smith.* extracted from ***The glmark2 official web site*** [1]

2 Using glmark2

Building glmark2 generates 2 binaries:

- **glmark2-es2-wayland**, to launch glmark2 benchmark as a wayland client.
- **glmark2-es2-drm**, to launch glmark2 benchmark as a native DRM/GBM openGLES application.

2.1 glmark2-es2-wayland

- Start Weston (if not already started)

```
Board $> systemctl start weston@root.service
```

- Launch glmark2

```
Board $> glmark2-es2-wayland
```

2.2 glmark2-es2-drm

- Stop Weston

```
Board $> systemctl stop weston@root.service
```

- The DRM display mode may need to be set and the DRM master token released so that glmark2 can use the DRM interfaces. This depends on the glmark2 version. For instance:

```
Board $> modetest -s 27:720x1280 -d &
```

Note: The connector and the mode are given as an example.

- Launch glmark2

```
Board $> glmark2-es2-drm
```

3 Source code location

- Official source code: https://github.com/glmark2/glmark2

4 To go further

You can find the full documentation of glmark2 in the related Ubuntu man page http://manpages.ubuntu.com/manpages/cosmic/man1/glmark2.1.html.

5 References

- https://github.com/glmark2/glmark2

Open Graphics Library (See http://www.opengl.org/ for more details)

Direct Rendering Manager (kernel module that gives direct hardware access to DRI clients, find more information on official DRI web site http://dri.freedesktop.org/wiki/DRM)