Tegra PerfHUD ES 1.1 Quickstart Guide

Version 1.1

Contents

Introduction

This quick start guide is designed to assist a new user in setting up and starting PerfHUD ES profiling of applications on a Tegra device.

PerfHUD ES enables experimentation and in-depth analysis of OpenGL ES applications. Through data provided by the OpenGL ES driver and the Tegra SoC, developers are provided live system status, including GPU performance and bottleneck information, in order to help target optimizations where they are most needed.

The debugging capabilities of PerfHUD ES give full access to the state of the OpenGL ES pipeline, related textures and shaders, and all rendering states to help find the causes for improper setup, rendering anomalies, and performance issues.

For the rest of the document, note that all downloads are available on the Tegra Devsite:

http://tegradeveloper.nvidia.com/tegra/downloads

Requirements

Host PC

- Ideally, a minimum 2GHz CPU with 2GB RAM or more
- An OpenGL 1.5 or better GPU (if you have an NVIDIA GeForce 6 series GPU or higher, that should suffice). The "PerfHUD ES User Guide" installed during this setup process covers specific GPU feature requirements in depth, as certain OpenGL extensions are necessary for the host PC to visualize the graphics rendered on the Tegra device.
- An available USB port, for flashing your Tegra device, or for Android ADB connection.
- OS Support:
    - Windows host: Windows XP, Windows Vista, and Windows 7, both 32-bit and 64-bit variants, have been tested and all seem to operate properly.
    - Mac host: Recent OS X 10.6 has been tested. Prior OS X versions may work, but have not been verified.
    - Linux host: Ubuntu 10.04 has been tested, but other Debian operating systems may also work.
- An Ethernet connection to the Tegra device from the host PC, specifically with an IP address on the Tegra device that is reachable from the host. Options include:
    - A crossover network cable for direct host-to-device connections (for host with extra Ethernet port, and device with wired Ethernet port)
    - Wired Ethernet hub/router with both on the same LAN (for host and/or target with wired Ethernet port)
    - Wireless Ethernet AP/router (for target and/or host as available).

  Further detailed network setup instructions are outside the scope of this document.

Tegra Device OS

- For Android: A recent Tegra Android OS image which has PerfHUD support.
- For Linux: A recent Tegra LDK or L4T Linux OS with PerfHUD ES support. We have tested both the 10.8.2 and 10.9.3 L4T releases (which include support in-pack), and an add-on support pack is available for LDK 10.9.3 (which may work with other releases).

PerfHUD ES Host Installation

Download the applicable PerfHUD ES Host installer from the Tegra Devsite for the host operating system you are using, then follow the related instructions below.

We assume you have already downloaded and installed a target Tegra device OS pack, have flashed the device, and have it booted and functioning. Note that we do not currently have OS flash packs for Mac OS X, so if that is your host OS, you'll need a different host PC for flashing your device initially.

We also assume you have a program that you want to analyze ready to run on the device.

Windows Host Setup

1) Download the Windows PerfHUD ES Host installer executable from the Tegra Devsite. The file should be something like "`NVIDIA_PerfHUD_ES_1.1.exe`" for Windows host.
2) Launch the installer.
3) Click "Next".
4) Decide to accept the EULA and then click "Next".
5) Decide on your installation location and then click "Next".
6) Click "Install" and wait for installation to complete.
7) Click "Finish".

We ***strongly*** recommend you read the "PerfHUD ES User Guide" document to learn how to use the PerfHUD ES host user interface to examine your GLES application. The installer adds a link to the doc in the the Start Menu under:

```
All Programs => NVIDIA Corporation => NVIDIA PerfHUD ES Tegra2
```

Linux Host Setup

1) Download the Linux PerfHUD ES Host installer executable from the Tegra Devsite. The file should be something like "`perfhudes_1.1_i386.deb`" for Linux host (note there should also be a "`perfhudes_1.1_amd64.deb`" installer file for 64-bit Linux hosts).
2) Install as you would any application normal.

Mac Host Setup

1) Download the Mac OS X PerfHUD ES Host installer executable from the Tegra Devsite. The file should be something like "`perfhudes_1.1.dmg`" for Mac OS X host.
2) Install as you would any application.

PerfHUD ES Target Device Support Setup

Android Device Setup

By default, android OS images do not have PerfHUD ES target support active/enabled. The OS installer does however include host-side scripts that allow PerfHUD ES target support to be enabled or disabled on demand, over a standard ADB connection.

> **All of these scripts automatically and immediately reboot the devkit after enabling or disabling PerfHUD support.** However, no data should be lost, as the device is *not* reflashed by this process.

Note that all the steps below assume you have connected your device to the host PC via ADB.

Enabling PerfHUD Target Support from a Windows Host

1) Open a command prompt in the install target (platform) directory. By default, this is:

   `Program Files\NVIDIA Corporation\tegra_froyo_20101105\perfhud_switch\`

2) Run the script: `enable_perfhud.bat`
3) On success the output should be something like:

```
\perfhud_switch>enable_perfhud.bat
\perfhud_switch>pushd "C:\Program Files\NVIDIA
Corporation\tegra_froyo_20101105\perfhud_switch\" \perfhud_switch>adb remount
remount succeeded
C:\Program Files\NVIDIA Corporation\tegra_froyo_20101105\perfhud_switch>adb push en_perf.txt
/system/lib/egl/egl.cfg
1 KB/s (0 bytes in 24.000s)
\perfhud_switch>adb push libs/libEGL_perfhud.so /system/lib/egl/libEGL_perfhud.so
419 KB/s (0 bytes in 13440.000s)
\perfhud_switch>adb push libs/libGLESv2_perfhud.so /system/lib/egl/libGLESv2_perfhud.so
1653 KB/s (0 bytes in 582092.000s)
\perfhud_switch>adb push libs/libGLESv1_CM_perfhud.so /system/lib/egl/libGLESv1_CM_perfhud.so
1182 KB/s (0 bytes in 113476.000s)
\perfhud_switch>adb reboot
```

Disabling PerfHUD Support from a Windows Host

1) Open a command prompt in the install target (platform) directory. By default, this is:

   `Program Files\NVIDIA Corporation\tegra_froyo_20101105\perfhud_switch\`

2) Run the script: `disable_perfhud.bat`
3) On success the output should be something like:

```
\perfhud_switch>disable_perfhud.bat
\perfhud_switch>pushd "C:\Program Files\NVIDIA
Corporation\tegra_froyo_20101105\perfhud_switch\"
```

```
\perfhud_switch>adb remount
remount succeeded
\perfhud_switch>adb push dis_perf.txt /system/lib/egl/egl.cfg
1 KB/s (0 bytes in 22.000s)
\perfhud_switch>adb reboot
\perfhud_switch>
```

Enabling PerfHUD Support from a Linux Host

1) Open a command prompt in the install target (platform) directory. By default, this is:

   `tegra_froyo_20101105/perfhud_switch`

2) Run the script: ./`enable_perfhud.sh`
3) On success the output should be something like:

```
me@ubuntu:~/20101105/tegra_froyo_20101105/perfhud_switch$ ./enable_perfhud.sh
~/20101105/tegra_froyo_20101105/perfhud_switch ~/20101105/tegra_froyo_20101105/perfhud_switch
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
remount succeeded
0 KB/s (24 bytes in 0.077s)
47 KB/s (13440 bytes in 0.278s)
121 KB/s (582092 bytes in 4.666s)
448 KB/s (113476 bytes in 0.247s)
~/20101105/tegra_froyo_20101105/perfhud_switch
```

Disabling PerfHUD Support from a Linux Host

1) Open a command prompt in the install target (platform) directory. By default, this is:

   `tegra_froyo_20101105/perfhud_switch`

2) Run the script: ./`enable_perfhud.sh`
3) On success the output should be something like:

```
me@ubuntu:~/20101105/tegra_froyo_20101105/perfhud_switch$ ./disable_perfhud.sh
~/20101105/tegra_froyo_20101105/perfhud_switch ~/20101105/tegra_froyo_20101105/perfhud_switch
remount succeeded
0 KB/s (22 bytes in 0.081s)
~/20101105/tegra_froyo_20101105/perfhud_switch
```

Linux Device Setup

Install PerfHUD ES Support Pack

> **Note:** **If you are using the L4T 10.8.2 pack or later, you can skip this step and go directly to the "Turn On/Off PerfHUD Target Support" section below.**

At this time, LDK customers will need the "`tegra_linux_perfhud_10.8_10.9.tgz`" support pack for this next step – if you don't already have it, please contact your support representative. It is easiest to copy the archive to the Tegra device (using you favorite method, either copying to

boot media from a host linux PC, or using remote file transfer like SCP), and then simply unpack it on-device using a remote shell, with:

```
tar -xzvf tegra_linux_perfhud_10.8_10.9.tgz
```

If you execute 'ls', you should see find now have 'perfhud.sh', as well as two PerfHUD libraries ('.so' files). The first time you run the perfhud.sh script, it will detect if the libraries are missing from the system, and attempt to install them properly (moving them to /usr/lib).

Enable/Disable PerfHUD Target Support

To enable PerfHUD support (and auto-install the support libs if not yet installed), open a command shell to the device, and from the default user home directory, execute:

```
./perfhud.sh on
```

If you later want to turn off PerfHUD support, for performance or compatibility, just execute:

```
./perfhud.sh off
```

Connecting to the Target Device

We assume you have enabled PerfHUD support on the target device, as specified in the relevant section above.

Start Target Application

Launch the application to be profiled on the Tegra device. You should see something like the following in the log output (remote shell on linux device, adb logcat on android device):

```
NVPerfHUD ES is active.
Communication layer init succeed, listen on port 7876.
```

That indicates the PerfHUD support code is running and waiting for the host app to connect.

Start PerfHUD ES Host

Run the PerfHUD ES Host program on your host PC. You will then be prompted with a dialog requesting the Target device's IP address.

Connect to Target over TCP/IP

If you are running just a single application, in a non-GLES-accelerated window manager:

1) Enter the IP address for your target into the PerfHUD ES Host dialog.
2) Click "Connect".

If you are running *multiple* GLES applications simultaneously on the target, or a window manager that also leverages OpenGL ES in addition to applications, you'll need to instead:

1) Enter the IP address for your target into the PerfHUD ES Host dialog.
2) Click "Advanced"
3) Click "Refresh graphics process list"
4) Select your application from the process list
5) Click "Connect"

Note that when there's only a single GLES application running, if you quit the application and then start it (or another GLES app) without exiting the PerfHUD ES Host, the Host will automatically re-establish a connection to the new app instance on the same target and port without user intervention. This is useful for repeated test runs as you find and fix issues.

Connect to Android Target using ADB

1) On your host PC, open a command shell and run the commandline "`adb logcat`", and note the port assigned to your program. This will show up in the log as a message like:

```
Communication layer init succeed, listen on port 7876.
```

2) On your host PC, you then must enable port forwarding for the given communications port, using the commandline:

```
adb forward tcp:<port> tcp:<port>
```

   Replace <port> in the above line by the previously noted port number – in the example output above it is '`7876`'.
3) Enter "localhost" as the IP address into the PerfHUD ES Host dialog.
4) Click "Advanced"
5) Click "Refresh graphics process list"
6) Select your application from the process and click "Connect"

Note that if you have problems with adb port forwarding, you may need to check that you aren't running a local firewall that is blocking the port.

Also note that if you are only (re)running one application on Android, you may find that the target port number isn't changing. In this case, you can shortcut steps 3-6 above by entering "`localhost:<port>`" as the IP address in the PerfHUD ES Host dialog box, and the host should directly connect to the running program without further steps.

Common Problems

The windows ding.wav plays over and over again.

There is an option to play a sound effect each time a buffer object is uploaded. To disable this, go to the View menu, and deselect "Audio Notifications".

The Speed Bar is not visible and the frame profiler is gray and not selectable.

Please ensure that your application is using the "egl_nv_perfmon" extension to acquire all application side timing information, as detailed in the "PerfHUD ES User Guide" document. The PerfHUD ES Host relies on application use of the timer functions in order to 'freeze time' and know the application will rerender the same exact frame repeatedly.

My application isn't animating when run with PerfHUD Host connected.

If you've implemented the EGL timing extension support, and the Speed Bar is up on the screen, make sure it's on the "Play" green triangle, not the "Stop" blue square, otherwise the application is being repeatedly fed the same timestamp and won't be animating, will seem 'frozen' at times.

My status bar says: "ERROR: target uses protocol x.x but host uses protocol x.x"

You have a version mismatch between the driver and the PerfHUD host. Always update your driver and PerfHUD host in tandem.

Known Issues

The Frame Profiler panels seem to have invalid data for utilization, bottlenecks, etc.

The current Tegra 3d driver isn't exposing the necessary low-level counters needed to obtain full profiling information. At this time, the only usable information is individual draw call durations, which can be seen by unchecking the three "Bucket Definition" checkboxes in the top-left of the profiling interface, and review the calls/times in the Draw Calls list at the top-right – if you switch to the Frame Debugger while a given draw call is selected, that call will be active in the debugger interface for easy identification. The driver work is under investigation.

Application hangs with target device log showing an error like "Output FIFO does not refill, context read is stuck".

There is no current workaround, please try restarting the target device and application. If the issue persists, please contact us. We are investigating the issue.

My application fails to run with GL_OUT_OF_MEMORY.

Please try restarting the target device and application. If the issue persists, please contact your support representative.

The Timing Graph varies wildly for a static scene.

There is no current workaround. We are investigating the issue.

The Frame Debugger takes a long time to complete or fails to complete.

Please try closing any other applications or increasing the memory available on your host PC.

The Frame Debugger locks when selecting a particular draw call or calls.

There is no current workaround. We are investigating the issue.

The Frame Debugger feature "Highlight Draw Call" shows seemingly incorrect visual differences between two draw calls/frames.

The Frame Debugger only sees the 'changes' to the framebuffer between each set of draw calls. So if the application renders a lot, but the actual visual/data difference in the framebuffer snapshot between the two calls is minimal (or the same), that affects the ability of the debugger to identify what pixels the draw call changed. Adding extra clears into debug-only builds of your application may improve the situation if you are attempting to debug visual issues, but of course note that extra clears certainly affect any performance gathering and should be disabled if performance is desired.

Framebuffer objects are displayed incorrectly when attached to a texture sampler.

There is no current workaround. We are investigating the issue.

The Texture Viewer tab does not appear or is empty for a draw call with textures.

Please try restarting the PerfHUD ES Host and reconnect to the target device. If the issue persists, please contact your support representative.

The shader viewer shows "Microcode disassembler is not available" for vertex and fragment microcode modes.

Vertex and fragment microcode disassembly is not supported by PerfHUD ES on Tegra. Please select a source mode instead. The unsupported modes will be removed in a future release.

The geometry viewer stutters on very complex geometry.

Please try closing other running applications on the host PC to ensure the host has all available CPU, memory, and GPU resources. Also, ensure that you have the latest drivers installed for your graphics card.

Vertical scroll bars appeared clipped along the vertical axis.

There is no current workaround. We are investigating the issue.

Portions of the Performance Dashboard overlay the Frame Debugger.

Please try restarting the PerfHUD ES Host to resolve this issue.

Programs using NVIDIA's GLES path extension fail to run with the PerfHUD target driver active, but run fine when it is disabled.

There is no current workaround. We are investigating the issue.

NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050
www.nvidia.com