The NVIDIA Windows PC OpenGL ES 2.0 and Khronos API Emulator Support Pack

Version 100

Contents

Introduction

This support pack includes a sort of "virtual platform"; a PC-based set of "wrapper" libraries that allow applications written to Khronos APIs (OpenKODE, OpenGL ES, EGL) to run (or at least link, depending on the Khronos API) with no source code modifications on both Tegra devkits and on a Windows-based PC. The Windows PC wrapper can make it easier for developers to share devkits by allowing application development for Tegra-targetted applications to be started on a Windows-based PC.

The pack includes Khronos headers and link libraries, as well as Windows PC DLLs that implement emulators for key Khronos media APIs. The pack is laid out in exactly the same format as any of the Windows CE 6.0 platform support packs, to ensure that minimal changes are required to move between the emulator's "virtual platform" and the Tegra devkit platform.

System Requirements

The minimum specifications for a PC to run emulator-based applications (details are given in the installation sections of this documentation) are:

Hardware

- Windows-compatible PC
    - ~2.0GHz CPU
    - 1-2GB RAM or better
- NVIDIA GPU, ~GeForce 6200 or newer (NV43), GeForce 8xxx series or newer recommended.

Tested/supported GPU hardware includes:

- NVIDIA Quadro NVS 110M
- NVIDIA GeForce 8400 series
- NVIDIA GeForce 8600 GTS
- NVIDIA Quadro FX 4000 (Quadro and PerfKit drivers)
- And most NVIDIA GeForce/Quadro products newer than the aforementioned GPUs

Tested/supported hardware with some reservations includes:

- NVIDIA GeForce 6200 TurboCache (no support for Luma-only FBOs)
- NVIDIA GeForce 6600GT (no support for Luma-only FBOs)
- NVIDIA GeForceGo 6600 TE (no support for Luma-only FBOs)
- NVIDIA Quadro NVS 135M (no support for advanced OpenGL ES 2.0 particles on some drivers)

Software

- Microsoft Windows XP or newer operating system installed on the host PC
    - XP has been extensively tested
    - Vista and Vista 64 has been given basic testing (although note that Vista 64 may have issues with other Platform Support packs, such as the Windows CE 6.0 devkit support packs. See the documentation for specific Platform Support packs for details)
    - Windows 7 has been given basic testing
- Microsoft Visual Studio:
    - Microsoft Visual Studio 2005 or 2008

Installing the Support Pack

Uninstalling the Previous SDK

If you have NVIDIA's NVAP SDK 0.1.x through 0.4.x installed on your host PC, you must first take the following actions to uninstall it and avoid conflicts:

If the previous SDK included an installer program, use it to uninstall the package. If the SDK was manually installed (some of the oldest SDKs were manual install), then take the following steps.

1) Remove any runtime DLL directories from previous NVAP SDKs from the system's `PATH` environment, such as the `platformlibs` paths added when installing a previous NVAP SDK.
2) Remove the old NVAPSDK environment variable as well.

In addition, previous versions of this emulator pack must be uninstalled before installing the latest version. ***Unlike some other platform support packs, only one emulator pack can be installed at any time, to avoid any DLL mismatches and/or path issues.***

To install the Support Pack, simply double click the installation file **win_x86_es2emu_<version>.msi**

Prerequisites Verification

The installer will detect and display the list of installed software required and/or recommended for development on the Windows PC emulator. Install any missing software before proceeding with installation. Please refer to the System Requirements documentation for details.

End-User License

Next you will be prompted with End-User License Agreement. Read the agreement and to accept it check the box "I accept the terms in the License Agreement". Click next to proceed.

Installation Type

There are 3 ways to install the Support Pack. Read the instructions on the screen and choose one of the three options.

"Typical" and "Complete" setup options will install the Support Pack in C:\Program Files\NVIDIA Corporation\win_x86_es2emu_<version>\ location by default. If you want to install the Support Pack to a different location then choose the "Custom" setup option, change the installation location and click Next. ***Do not move the Support Pack manually after install*** – the installer sets environment variables that point to the installed location of the Support Pack.

Following the selection of the installation type, file copying will begin. It can take several minutes to complete the installation. Once installation is complete, click "Finish". Since the installer adds and modifies environment variables including the PATH, you will be prompted to acknowledge that restarting the host PC is recommended.

Environment Variables

The installer adds or updates several environment variables to locate the Support Pack:

`NV_WINGL_X86_PLAT`: This environment variable is set to the path of the root of the Support Pack, and can be referenced in VCPROJ files to locate Khronos headers and libraries.

`PATH`: This standard environment variable is modified to append the path to the `bin` subdirectory of the Support Pack to the end of the search path. This path includes the Khronos API emulation DLLs, and allows applications linked against the emulator to run on the PC.

Layout of the Platform Support Pack

The platform support pack has the following hierarchy on the host PC once installed:

```
$(NV_WINGL_X86_PLAT)\
    bin\
        release\
```

Contain the Khronos API emulator implementation DLLs. This full path is added to the Windows PC's `$PATH` variable by the installer.

```
    include\
```

Contains standard, Khronos-mandated subdirectories for each Khronos API's headers (e.g. KD for OpenKODE Core). These headers match the ones used to build the emulator included in the Platform Support pack's bin directory.

```
    lib\
        release\
```

Contain Windows-compatible link libraries for the Khronos API emulators. These libraries match the ones used to build the emulator included in the Platform Support pack's bin directory.

The Khronos APIs

To maximize cross-platform compatibility, applications should avoid including and using headers and functions from the standard C libraries (`stdio.h`, `stdlib.h`, `memory.h`, etc). Instead, applications should limit their use of platform-specific system headers and instead use the Khronos OpenGL ES 2.0, OpenMAX IL 1.1, OpenVG 1.0, EGL 1.3, and OpenKODE Core 1.0 libraries. If possible, do not include Windows headers, standard platform library headers, etc.

This Platform Support pack ships with the following Khronos libraries supported in emulation:

- **OpenKODE Core 1.0**: `KD/kd.h`, `libKD.lib`: POSIX-like functionality for files, I/O, etc, along with basic window-system and input-handling functionality. See the Khronos documentation for details.
  http://www.khronos.org/registry/kode/specs/openkode.1.0.pdf
- **OpenGL ES 2.0**: `GLES2/gl2.h`, `GLES2/gl2ext.h`, `libGLES20.lib`: Shader-based 3D rendering. See the OpenGL ES 2.0 and GLSL-E 2.0 Shading Language documentation for details.
  http://www.khronos.org/files/opengles_spec_2_0.pdf,
  http://www.khronos.org/files/opengles_shading_language.pdf
- **EGL 1.3**: `EGL/egl.h`: Buffer and Context management, linking OpenKODE Core and OpenGL ES 2.0. See the Khronos documentation for details.
  http://www.khronos.org/registry/egl/specs/eglspec.1.3.pdf

The following libraries are also supplied, but only as "null" link libraries. None of the functions in these libraries do any work.

- **OpenGL ES 1.1**: `GLES/gl.h`, `GLES/glext.h`: Fixed-function-based 3D rendering. See the OpenGL ES 1.1 documentation for details. ***Link only. APIs not implemented.***

  http://www.khronos.org/registry/gles/specs/1.1/es_cm_spec_1.1.10.pdf
- **OpenMAX IL 1.1**: `openmax/il/*`: Support for hardware-accelerated video, audio and imaging. See the Khronos documentation for details. ***Link only. APIs not implemented.***

  http://www.khronos.org/files/openmax_il_spec_1_1_1.pdf
- **OpenVG 1.0**: `VG/*`: Support for vector graphics. See the Khronos documentation for details. ***Link only. APIs not implemented.***

  http://www.khronos.org/files/openvg_1_0_1.pdf

Building Code for the Khronos API PC Emulator

In order to build and link an application for the emulated Khronos platform on Windows, the paths to the emulator's Khronos headers and libraries must be added to the project:

1) Add the following paths to the "C/C++:General:Additional Include Directories" in all of the emulator configurations in the project:

   `$(NV_WINGL_X86_PLAT)/include`

   (for most Khronos APIs)

   `$(NV_WINGL_X86_PLAT)/include/OpenMax/il`

   (for OpenMAX IL)
2) Add the following paths to the "Linker:General:Additional Library Directories" in all of the emulator configurations in the project:

   `$(NV_WINGL_X86_PLAT)/lib/release`

The Platform Support pack installer will have already set the variable `$(NV_WINGL_X86_PLAT)` as required and added the path to the implementation DLLs to the system path. However, after installing the Platform Support pack, exit and restart any open MSVC++ instances, so that they can re-read the environment variables.

Link Libraries for Khronos Applications on the PC Emulator

The emulator platform includes both GLES2 and EGL in the same library/DLL. Thus, to link both GLES2 and EGL to an emulator app, link the library `libGLES20.lib` instead of

`libEGL.lib` and `libGLESv2.lib` as done on Tegra. This may be modified to match the Khronos standard in a coming release.

In order to successfully link an OpenKODE Core app on the emulator, both `libKD.lib` and `libnvkdmain.lib` must be added to the "additional libraries" line. Failure to include latter library will result in "undefined main" link errors.

Only release emulator libraries and DLLs are shipped with the platform support pack, so both release and debug configurations of applications should link with the release Khronos libraries.

Limitations/Differences Between Emulation and the Tegra

While the emulator is designed to match the Tegra where possible, there are numerous cases where the emulated behavior will not match the Tegra, See the following sections for details.

OpenGL ES 2.0

Please note the following differences between the Tegra platform's ES 2.0 support and that of the PC-based emulator:

Tegra Platform OpenGL ES 2.0 Driver	**SDK WinXP Emulation Wrapper**
Shader Support	
Source-code and Binary-precompiled shaders both supported	Binary-precompiled shaders not supported
Support for reading the framebuffer in a fragment shader via **`NV_shader_framebuffer_fetch`**	Only fixed-function alpha blending supported
Shader limitations exposed via GL state queries	Shader limitations should be read from the tables in the Tegra OpenGL ES 2.0 development guide; the wrapper is not guaranteed to return Tegra-correct values
Default precision specifiers supported	Default precision specifiers ignored.

Texturing Support	
Non-power of two textures do not support mipmapping or wrapping	Non-power of two textures can support mipmapping and wrapping
Supports ETC, LATC1 and LATC2 compressed textures	Does not support ETC or LATC textures
Maximum texture size 2048x2048 texels	Maximum texture size is implementation-dependent
Signed fixed-point and floating-point textures supported.	Signed shader support varies.
Buffer Formats	
Support for 16bpp depth, including optional use of a non-linear Z extension (`GL_NV_depth_nonlinear`).	Depth buffer formats differ by implementation and may include 24 or 32 bpp depth
Support for PBuffers in EGL (but FBOs are considered the best option in ES2.0)	No support for PBuffers
Performance	
Performance is, of course, representative of the device…	Performance is in no way linked to final Tegra device performance
Misc	
Support for vector-graphic rendering via the `GL_NV_draw_path` extension.	No vector-graphic support
Support for Coverage-sampled Antialiasing (CSAA).	No CSAA support – MSAA support may be enabled automatically on some drivers

GLSL-ES idioms not supported in GLSL

These are ignored, but will compile:

- Default precision specifiers

Shader Checking Issues

The shader-checking code is implemented outside of the native Windows GL shader compiler. As a result, the line numbers of shader errors will be incorrect with respect to the source shaders.

In some cases, there may be issues with shaders that are correct according to the GLSL-ES compiler for Tegra not compiling on the desktop wrapper. In addition, it can be confusing when developing shaders for the line numbers of errors to be offset.

In these (and other) cases, it may be possible to let these shaders "pass through" the wrapper by setting a hint that disables pre-processing of the shaders. The wrapper (but not on the actual Tegra development hardware) supports the following hint:

```
// to disable GLSL-ES shader-checking
glHint(GL_SHADER_COMPILER, 1);

// to enable GLSL-ES shader-checking (the default)
glHint(GL_SHADER_COMPILER, 0);
```

Emulator Programming Notes

OpenKODE Core File System

The root of the OpenKODE Core file system on the emulator is designed to match the WinCE-based Tegra driver as closely as possible. Thus, the root of the file system is based off of the location of the application's executable. Given a path to the executable `$(APP_PATH)`, the OpenKODE directories map as follows:

OpenKODE Core Directory	**Windows Path**
`/res`	`$(APP_PATH)\resources`
`/data`	`$(APP_PATH)\data`

Explicit use of removable storage is not recommended on the emulator, as it is not supported in the same manner on the WinCE Tegra OS images.

"Fullscreen" Window Size

OpenKODE "Fullscreen" windows are drawn as normal Win32 windows of a fixed size on the emulator. By default, all OpenKODE fullscreen windows are 800x480 pixels. Developers who want to change the "size" of the fullscreen windows on their system can do so by setting the following global environment variables on their development PC:

```
NVAP_SCREEN_WIDTH=

NVAP_SCREEN_HEIGHT=
```

To the desired height and width. After changing this value, restart Visual Studio to re-read the environment.

Window Handle Support on the Emulator

The emulator's EGL implementation can support both OpenKODE Core-created `EGLNativeWindowType` handles returned from `kdRealizeWindow(KDWindow*)` and also native Windows `HWND`s. This allows Win32 applications to cross over between the emulator and the Tegra devkit. However, a given application instance can only handle one of these two types of windows. Passing the first window handle to the emulator's EGL implementation causes that particular EGL "back end" to be installed for that application instance.

Cross-platform Windows HWND code that creates an 800x480 window on the Emulator and a fullscreen window on the Tegra devkit is as follows:

```
    HWND hwnd = CreateWindow(
        TEXT("EGLWndClass"),
        NULL,
        WS_OVERLAPPED|WS_SYSMENU|WS_THICKFRAME|WS_DISABLED,
        CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT, CW_USEDEFAULT,
        NULL,
        NULL,
        (HINSTANCE)GetModuleHandle(NULL),
        NULL);

    if (!hwnd)
        return -1;

    EnableWindow(hwnd, TRUE);

    RECT area;
    area.left = 0;
    area.top = 0;
#ifdef _WIN32_WCE
    area.right = GetSystemMetrics(SM_CXSCREEN);
    area.bottom = GetSystemMetrics(SM_CYSCREEN);

    SetWindowLong(hwnd, GWL_STYLE, WS_POPUP);

    SetWindowPos(hwnd, HWND_TOPMOST,
                 area.left, area.top,
                 area.right, area.bottom,
                 SWP_FRAMECHANGED);
#else
    area.right = 800;
    area.bottom = 480;

    SetWindowPos(hwnd, HWND_TOP,
                 area.left, area.top,
                 area.right, area.bottom,
                 SWP_NOMOVE);
#endif

    /* set as foreground window to give this app focus in case it doesn't
have it */
    SetForegroundWindow(hwnd);
    ShowWindow(hwnd, SW_SHOWNORMAL);
```

EGL and OpenGL ES 2.0 Conformance

Note that the goal of this platform support pack is to make it easier for developers to create OpenKODE, EGL and GLES 2.0 applications without deploying to a (possibly shared) devkit on every iteration. Thus, focus in development has been on functionality and basic feature set, rather than exact Khronos conformance. Please keep this in mind when developing applications. Differences between the (conformant) Tegra Khronos drivers and the emulator are listed in prior sections.

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