Technical Note TN2014

Insights on OpenGL

CONTENTS

Missing OpenGL Geometry

QuickTime Movie Output

"Native" Pixel Format

Enabling packed pixels

VBL syncing

Page flipping

Frame Tears

Using OpenGL with DrawSprocket

Using OpenGL with QuickDraw

Buffer Allocation

More to Come

References

Downloadables

This Technote describes a variety of issues regarding OpenGL and graphics on the Macintosh. It covers items such as integration with QuickTime and DrawSprocket, problems such as frame tearing and unwanted geometry clipping, as well as clarifications and answers to frequently asked questions.

This Note is directed at application developers who are familiar with OpenGL and 3D graphics in general.

Updated: [Mar 26 2001]

Missing OpenGL Geometry

There are many things that can keep portions of OpenGL geometry from drawing on the screen. However, two of the most common are as follows:

1. Your surfaces(s) are being clipped by one or more of the view frustum planes
2. Your camera is pointing at the back side(s) of the surface(s)
   - Normals are reversed
   - Camera coordinates or viewpoint coordinates are wrong

If your view is completely black, chances are your camera is pointed in the wrong direction entirely. Usually this is due to positioning the camera too far up or down the z-axis or having the camera looking at the wrong view coordinates. Note that the default camera position is (0.0, 0.0, 0.0) pointed down the negative Z-axis.

Back to top

QuickTime Movie Output

To use OpenGL as an output for a QuickTime movie, each frame of the QuickTime movie should be loaded into a `GWorld` then uploaded as a texture to an OpenGL polygon to display the surface. Sample code for this is available at the following URL:

http://developer.apple.com/samplecode/Sample_Code/Graphics_3D/OpenGL_Movie.htm

You have to extract each frame from the movie as a texture and then apply it to a polygon in OpenGL. Note that you'll have to invert your v-coordinate on the texture, or load it in reverse order, otherwise your movie will play upside down. See the sample called "OpenGL Movie" on the Apple Developer web site for more information.

Back to top

"Native" Pixel Format

The native pixel format is ARGB 8888 in 32-bit and in 16-bit, it is ARGB 1555. Packed pixels can be used to support these native formats through the OpenGL pipeline.

Back to top

Enabling packed pixels

Packed pixels is supported in OpenGL 1.2 or later on the Macintosh. This can be verified by checking the version string of the renderer. Renderers may also support packed pixels via the extension "GL_APPLE_PACKED_PIXELS" (note that it the actual string returned by the ATI drivers is "GL_APPLE_PACKED_PIXEL").

When a pixel format is chosen, you must use 'AGL_NO_RECOVERY' in order to ensure that the pixels are passed through OpenGL in the specified format.

The following are the optimal choices for pixel formats in OpenGL:

```
        16-Bit:
            Pixel Format    : GL_BGRA_EXT
            Pixel Type        : GL_UNSIGNED_SHORT_1_5_5_5_REV
        32-Bit:
            Pixel Format    : GL_BGRA_EXT
            Pixel Type        : GL_UNSIGNED_SHORT_1_5_5_5_REV
```

Back to top

VBL syncing

Synchronize the frame rate of my application to the monitor's refresh rate (VBL syncing) can be enabled with the following:

```
        GLint swapInt = 1;
        AGLSetInteger(aglContext, AGL_SWAP_INTERVAL, &swapInt);
```

Note that this will lock your frame rate to a divisible integer of your monitor's current refresh rate.

Back to top

Page flipping

ATI drivers currently enable page flipping on RAGE 128-based systems with a full-screen, non-occluded drawable. Window-mode page flipping is not supported.

Note that the benefits of page flipping have become less and less pronounced as hardware and software technology has advanced. The typical gain is only a few frames per second since VRAM to VRAM blitting is so fast on modern hardware.

Back to top

Frame Tears

Frame tears are a synchronization issue unrelated to swap speeds, contrary to popular belief. A tear occurs when the buffers are swapped while the electron beam is in the middle of painting on the screen. VBL synching should solve this problem.

Back to top

Using OpenGL with DrawSprocket

See the full screen sample at the following URL: http://developer.apple.com/samplecode/Sample_Code/Graphics_3D/

Back to top

Using OpenGL with QuickDraw

To draw to the front buffer with QuickDraw, you must synchronize your drawing with OpenGL, which entails stalling the OpenGL pipeline (not recommended for performance reasons).

This can be achieved by calling `glFinish()` immediately following a call to `aglSwapBuffers()`. QuickDraw can then be used to draw to the front buffer. Note that since the contents of the buffer will again be invalidated at the next call to `aglSwapBuffers()`, flickering may occur. The recommended solution is to use QuickDraw to draw into a GWorld then use the GWorld as a texture or draw the contents of the GWorld directly into the back buffer using `glDrawPixels()`.

Back to top

Buffer Allocation on Mac OS 9 and OS X

OS 9 allocates buffers on a per-context basis; each next context that is created has its own set of independent buffers that are exclusive to that context. Buffers can be shared among contexts on Mac OS 9 by specifying context to share with in `aglCreateContext()`.

Stencil, AUX (auxiliary) and accumulation buffers are not supported in hardware on Mac OS 9.

Mac OS X allocates buffers on a per-surface basis; a "surface" is the equivalent of a drawable on Mac OS X. You may create and attach multiple contexts to a single surface in Mac OS X. All contexts created on a given surface will share the same set of buffers.

Mac OS X also supports AUX (auxiliary) buffers and an 8-bit stencil buffer. Accumulation buffers are not supported in hardware on Mac OS X.

Back to top

More to Come

The following issues will be addressed in the future:

- Which of the following - NSGL, AGL, and CGL- should be used with Mac OS X
- Using OpenGL with CoreGraphics
- How does OpenGL on the Macintosh handle extensions? What extensions are supported?

Back to top

References

OpenGL ARB OpenGL Programming Guide, 3rd Edition Addision-Wesley

OpenGL ARB OpenGL Reference Manual, 3rd Edition Addision-Wesley

Foley, van Dam, Feiner, Hughes Computer Graphics - Principles and Practices, 2nd Edition in C Addision-Wesley

Back to top

Downloadables

Acrobat version of this Note (size in bytes, size in K) Download

Back to top