Appendix G

OpenGL Invariance

OpenGL is not a pixel-exact specification. It therefore doesn't guarantee an exact match between images produced by different OpenGL implementations. However, OpenGL does specify exact matches, in some cases, for images produced by the same implementation. This appendix describes the invariance rules that define these cases.

The obvious and most fundamental case is repeatability. A conforming OpenGL implementation generates the same results each time a specific sequence of commands is issued from the same initial conditions. Although such repeatability is useful for testing and verification, it's often not useful to application programmers, because it's difficult to arrange for equivalent initial conditions. For example, rendering a scene twice, the second time after swapping the front and back buffers, doesn't meet this requirement. Therefore, repeatability can't be used to guarantee a stable, double-buffered image.

A simple and useful algorithm that counts on invariant execution is erasing a line by redrawing it in the background color. This algorithm works only if rasterizing the line results in the same fragment *(x, y)* pairs being generated in both the foreground and background color cases. OpenGL requires that the coordinates of the fragments generated by rasterization be invariant with respect to framebuffer contents; which color buffers are enabled for drawing; the values of matrices other than those on the tops of the matrix stacks; the scissor parameters; all writemasks; all clear values; the current color, index, normal, texture coordinates, and edge-flag values; the current raster color, raster index, and raster texture coordinates; and the material properties. It is further required that exactly the same fragments be generated, including the fragment color values, when framebuffer contents, color buffer enables, matrices other than those on the tops of the matrix stacks, the scissor parameters, writemasks, or clear values differ.

OpenGL further suggests, but doesn't require, that fragment generation be invariant with respect to the matrix mode, the depths of the matrix stacks, the alpha test parameters (other than alpha test enable), the stencil parameters (other than stencil enable), the depth test parameters (other than depth test enable), the blending parameters (other than enable), the logical operation (but not logical operation enable), and the pixel-storage and pixel-transfer parameters. Because invariance with respect to several enables isn't recommended, you should use other parameters to disable functions when invariant rendering is required. For example, to render invariantly with blending enabled and disabled, set the blending parameters to GL_ONE and GL_ZERO to disable blending, rather than call **glDisable**(GL_BLEND). Alpha testing, stencil testing, depth testing, and the logical operation all can be disabled in this manner.

Finally, OpenGL requires that per-fragment arithmetic, such as blending and the depth test, is invariant to all OpenGL states except the state that directly defines it. For example, the only OpenGL parameters that affect how the arithmetic of blending is performed are the source and destination blend parameters and the blend enable parameter. Blending is invariant to

all other state changes. This invariance holds for the scissor test, the alpha test, the stencil test, the depth test, blending, dithering, logical operations, and buffer writemasking.

As a result of all these invariance requirements, OpenGL can guarantee that images rendered into different color buffers, either simultaneously or separately using the same command sequence, are pixel-identical. This holds for all the color buffers in the framebuffer or all the color buffers in an off-screen buffer, but it isn't guaranteed between the framebuffer and off-screen buffers.