New Features

This section provides general information on new features in the XGL 3.2 release. For detailed information on changes to the XGL library at this release, see Appendix A, "Changes to the XGL 3.2 Library".

Vertex Texture Mapping

This release of the XGL 3.2 library includes vertex-level texture mapping, a method of texture mapping that does not require dedicated graphics hardware. Traditional pixel-level texture mapping computes texture color at every pixel. Because of this, a software implementation for texture mapping is usually slow. If texture mapping is not supported in hardware, the software implementation is the only way an application can use texture mapping. Vertex-level texture mapping computes texture color only per vertex of the primitive, and then the primitive's interior is Gouraud shaded. With this method of texture mapping, hardware accelerators that support fast triangle rendering can be utilized to perform the shading, thereby increasing the overall performance for texture-mapped primitives.

Vertex-level texture mapping is available through the Texture Map object `XGL_3D_CTX_SURF_TMAP_PERSP_CORRECTION` attribute, which defines the method used to compute texture coordinate values for surface interiors. For more information on vertex-level texture mapping, see Chapter 18, "Texture Mapping".

Environment Texture Mapping

This release of XGL also includes environment texture mapping. Environment texture mapping textures a surface as if it were reflected from a shiny object such as a sphere. The viewer position and the position of the surface determine how objects are reflected. Environment texture mapping is available through the `XGL_TMAP_PARAM_TYPE` attribute, which defines the parameterization method for mapping a texture to a 3D surface. For more information on environment texture mapping, see Chapter 18, "Texture Mapping".

Raster Text

With this release, applications can render text using bitmap fonts. Bitmap fonts are available with the X window system, or the application can define its own bitmap fonts. To render bitmap fonts, XGL uses the existing raster operators `xgl_context_copy_buffer()` for 2D and `xgl_image()` for 3D.

Raster text can be rendered as stencils or in block format. Stencil text sets the color of the foreground pixels of the character only, leaving the background pixels set to their previous color. This enables applications to render text on top of an existing image. Stencil text is enabled by setting the raster fill style attribute `XGL_CTX_RASTER_FILL_STYLE` to `XGL_RAS_FILL_STENCIL`. For raster text in block format, XGL renders the foreground color of the character pixels and sets the background color of the character bitmap to the current background color. For more information on raster text and stencils, see Chapter 12, "Text".

XGL and the Kodak Color Management System

At this release, an XGL application can use the Kodak Color Management System (KCMS) to correct colors between different devices, such as color monitors and printers. Because different technologies are used to produce colors on monitors and printers, the same set of RGB values may look different on a monitor than on the printed page. KCMS provides a set of library functions that correct for the differences in color display between devices. KCMS also includes a C application interface that an XGL application can use. For more information on KCMS, see Chapter 6, "Color".

Triangle-List Gcache

The Gcache object provides a facility to break down complex primitives into simpler forms, resulting in increased rendering performance if the underlying accelerator is optimized for the resulting decomposed primitive. The Gcache object includes a set of operators that decompose primitives into simpler geometries. At this release, the operator `xgl_gcache_triangle_list()` has been added to the set of operators provided by the Gcache object. This operator stores a 3D triangle list in the specified Gcache. For more information, see Chapter 17, "Caching Geometry".

DGA Transparent Overlay

Transparent overlay windows enable applications to superimpose images on graphics in an underlying display window. The XGL library provides support for transparent overlay windows, using the window system interface provided in the OpenWindows environment. At this release, XGL also provides support for accelerated rendering into overlay windows. If the hardware device graphics handler implements overlay support, XGL coordinates with the server to provide accelerated rendering into transparent overlay windows using DGA.