XGL™ Accelerator Guide for Reference Frame Buffers

SunSoft, Inc.
A Sun Microsystems, Inc. Business
2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

THE NETWORK IS THE COMPUTER™

Contents

Preface. ix

1. General Acceleration Information 1

Batching. 1

Color Type and Visual 2

Tessellating Data 4

XGL Data Types 4

Using `xgl_inquire()`. 4

Use ASTI Deferral Mode 5

Disable Error Checking 5

Use of Backing Store 5

Faster Rendering Into Memory Rasters and Unaccelerated Frame Buffers 5

Faster Bounding Box Checking 6

2. GX and GXplus Accelerators. 7

Device-Dependent Issues 7

Accelerated Rendering Using Screen Coordinates (2D Only) 7

Deferral Mode 8
Backing Store 8
Picking. 8
Line Join 8
Hardware Double Buffering. 8
X Visuals 9
Attributes That Affect Acceleration. 9
Reference Table of Primitives and Attributes. 13
3. Acceleration Across a Network With the Xpex Pipeline 19
Rendering Across a Network 19
General Acceleration Notes 20
Depth Cueing 21
Line Widths 21
Use Gcache to Improve Performance 21
Line Patterns. 21
Markers 21
Lights. 21
Supported Point Types 22
Attributes That Affect Acceleration. 22
Reference Table of Primitives and Attributes. 28

Tables

Table 1-1 X Visual Class and XGL Color Type. 2

Table 2-1 Primitives and Related Attributes for the GX 14

Table 3-1 Point Types Supported in the Xpex Pipeline. 22

Table 3-2 Primitives and Related Attributes for Xpex. 29

Preface

The *XGL Accelerator Guide for Reference Frame Buffers* provides performance hints that you can use to write programs that use SunSoft hardware graphics accelerators. It lists attributes that are accelerated and shows how to make the most efficient use of XGL™ primitives.

Who Should Use This Book

This manual is intended for application programmers developing XGL applications.

How This Book Is Organized

This manual is organized as follow:

- **Chapter 1, "General Acceleration Information,"** provides information on general techniques to improve performance on all XGL platforms.
- **Chapter 2, "GX and GXplus Accelerators,"** gives specific hints and optimizations for the GX and GXplus accelerators.
- **Chapter 3, "Acceleration Across a Network With the Xpex Pipeline,"** provides information on accelerating an application across a network.

Related Books

- *XGL Programmer's Guide*
- *XGL Reference Manual*

Ordering Sun Documents

The SunDocs℠ program provides more than 250 manuals form Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals using this program.

For a list of documents and how to order them, see the catalog section of SunExpress™ On The Internet at `http://www.sun.com/sunexpress.`

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

Table P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`
`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`** `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

General Acceleration Information 1

This chapter provides general acceleration information that will enable an XGL program to run in the most efficient way regardless of the targeted platform. For specific information on the GX and GXplus devices, see Chapter 2, “GX and GXplus Accelerators”. For information on accelerating XGL applications across a network, see Chapter 3, “Acceleration Across a Network With the Xpex Pipeline”.

Note – Memory rasters are not accelerated on any device.

Batching

Due to the overhead involved in every XGL call, applications that use an XGL primitive to draw a single object per call do not achieve maximum performance. Applications run significantly faster through effective use of the *multi* primitives; drawing about 30 objects per call results in nearly peak performance.

Note – Avoid using the primitive `xgl_polygon()` unless absolutely necessary, since it draws only one polygon per call.

Color Type and Visual

For good performance, you should understand the color model of the targeted device and program accordingly. A *visual* is an Xlib concept that describes the way pixels are translated into colors. XGL supports these visual classes: PseudoColor, TrueColor, and DirectColor. Take care when selecting the visual because the visual class affects the XGL hardware color type, which in turn affects accelerator performance.

Although XGL allows the mixing of visuals and color types, for best results PseudoColor visuals should be used with a raster color type of `XGL_COLOR_INDEX`, and TrueColor and DirectColor visuals should be used with `XGL_COLOR_RGB`. Table 1-1 shows these combinations.

Table 1-1 X Visual Class and XGL Color Type

Visual Class	XGL Color Type
PseudoColor	`XGL_COLOR_INDEX`
DirectColor TrueColor	`XGL_COLOR_RGB`

The following code segment shows the proper way to select the visual that gives the best performance on any frame buffer (8- or 24-bit).

```
void
appli_get_accelerated_x_visual(dpy, scr, depth, vis)
Display       *dpy;      /* IN: Xlib display structure */
Screen        scr;       /* IN: X screen */
int           *depth;    /* OUT: depth of frame buffer */
Visual        **vis;     /* OUT: visual class for window */
{
  Visual         *appli_get_x_visual();

  *vis = NULL;
  if ((*vis = appli_get_x_visual(dpy, scr, 24, TrueColor)))
    *depth = 24;
  else
    /* we didn't get TrueColor, so try for DirectColor */
    if ((*vis = appli_get_x_visual(dpy, scr, 24,
                                   DirectColor)))
    *depth = 24;
  else
    /*
```

```
    * we didn’t get TrueColor or DirectColor, try PseudoColor
    */
  if ((*vis = appli_get_x_visual(dpy, scr, 8, PseudoColor)))
    *depth = 8;
  else
          /* we didn’t get a visual, return error */
    *depth = 0;
}

Visual         *
appli_get_x_visual(dpy, scr, depth, visual_class)
Display        *dpy;            /* IN: Xlib display structure */
Screen          scr;            /* IN: X screen */
int            *depth;          /* IN: depth of frame buffer */
int             visual_class;   /* IN: TrueColor, DirectColor
                                 * or PseudoColor
                                 */
{
  XVisualInfo     template;     /* X visual info structure */
  XVisualInfo    *visuals,
                 *v;            /* temporary structures */
  int             nvisuals;     /* number of visuals */
  int             i;

  template.screen = scr;
  template.depth = depth;

  visuals = XGetVisualInfo(dpy,
                         VisualScreenMask | VisualDepthMask,
                         &template, &nvisuals);

  for (v - visuals, i = 0; i < nvisuals; v++, i++)
    if (v -> class == visual_class)
      return (v -> visual);

  return ((Visual *) NULL);
}
```

Note – Some graphics devices have 24-bit X visuals that are linear. This means that they display colors as if the monitor had a linear intensity response. This usually requires color correction on the part of the device through a process called gamma correction. XGL works best when it renders to a window that was created with a linear visual. For details on how to find a linear visual, refer to the *Solaris X Window System Developer’s Guide*.

Tessellating Data

Tessellating (breaking down a complex polygon into triangles, for example) results in faster rendering on the GX accelerator. This performance optimization should be used only if the data can be easily tessellated into triangles, quadrilaterals, or rectangles. If the data cannot be easily tessellated, use XGL Geometry Caches to break down complex polygons and store them for later rendering. This is essentially a device-dependent performance optimization. Future graphics accelerators (from Sun or other vendors) may handle untessellated data more efficiently.

XGL Data Types

You should avoid copying geometric data into an XGL data structure and then calling an XGL primitive, since this costs both memory and time. If the format of the application's data is compatible with an XGL data type, you can set up pointers to the data and call XGL primitives directly with that data.

In addition, providing normals with your 3D data whenever necessary enables XGL to bypass calculating normals. If an object is to be rendered several times (such as in an animation sequence), this can result in a significant increase in speed.

Not sending down enough information can degrade performance, as can sending down too much data. For instance, sending down facet color data (or, worse yet, vertex data) when all the facets (vertices) have the same color results in slower rendering. Instead, the context color should be updated and only the geometry (x, y, z) information given to XGL.

Using `xgl_inquire()`

The API function `xgl_inquire()` enables an application program to get specific information about the frame buffer being used (such as the frame buffer name, the color type supported, whether hardware double buffering is available, and so on). The application program can use this information to render as efficiently as possible on that frame buffer. For instance, if the frame buffer supports hardware double-buffering, there is no need to set up Color Map double-buffering.

Use ASTI Deferral Mode

The default value of XGL_CTX_DEFERRAL_MODE (XGL_DEFER_ASTI) should be used whenever possible. Deferral mode XGL_DEFER_ASAP is slower and should be avoided.

Disable Error Checking

XGL error checking (enabled by setting XGL_SYS_ST_ERROR_DETECTION to TRUE) should be turned on while you are developing your application and turned off when you compile your application for distribution. Setting XGL_SYS_ST_ERROR_DETECTION to TRUE interferes with certain internal XGL optimizations. Error checking is disabled by default.

Use of Backing Store

If XGL_WIN_RAS_BACKING_STORE is set to TRUE, rendering will be slower whenever the window is partially or fully obscured, and slower still if anything is drawn into the obscured portion of the window (since such drawing is into a memory raster, which is not accelerated).

Faster Rendering Into Memory Rasters and Unaccelerated Frame Buffers

XGL releases subsequent to XGL 3.0 are significantly faster than XGL 3.0 at rendering certain primitives into memory rasters, or into simple frame buffers such as the CG3 (indexed color) and CG8 (TrueColor). Faster primitives include 2D vectors, 3D unshaded vectors (Z-buffered or not Z-buffered), and 3D polygons (including shaded and/or Z-buffered polygons) using either true or indexed color. The increase in speed for polygons currently applies only to 8-bit deep rasters and frame buffers.

This enhancement also applies to the GX and GXplus accelerators when rendering is unaccelerated for Z-buffered 3D polylines and Z-buffered and/or shaded polygons.

Faster Bounding Box Checking

XGL 3.0.1 and subsequent releases are faster than previous releases at performing bounding box checks on individual primitives. Complex primitives that are likely to be fully clipped away (during pan and zoom operations, for example) can gain in performance through the use of bounding boxes.

GX and GXplus Accelerators 2

The GX and GXplus graphics accelerators accelerate 2D and 3D solid-color polygons, solid and patterned vectors, and 2D and 3D transformations. The GX and GXplus do not have hardware Z-buffers or perform color interpolation. Z-buffered, depth cued, and Gouraud-shaded objects are not accelerated.

The hardware draws triangles and most quadrilaterals directly. More complex polygons must first be subdivided (tessellated) into triangles or quadrilaterals, either by the user or internally by XGL.

Device-Dependent Issues

Accelerated Rendering Using Screen Coordinates (2D Only)

The use of integer screen coordinates on the GX or GXplus results in increased performance, provided all attributes are set to allow acceleration and the following conditions are met:

- `XGL_CTX_VDC_MAP` must be set to `XGL_VDC_MAP_DEVICE` (the default).
- `XGL_CTX_VDC_ORIENTATION` must be set to `XGL_Y_DOWN_Z_AWAY` (the default).
- `XGL_CTX_LOCAL_MODEL_TRANS` and `XGL_CTX_GLOBAL_MODEL_TRANS` must be set to identity (the default).

- When using screen coordinates, the `xgl_multipolyline()` primitive will render substantially faster with these point types: *Xgl_pt_i2d*, *Xgl_pt_color_i2d*, *Xgl_pt_flag_i2d*.

Deferral Mode

Rendering with the GX and GXplus is substantially faster using `XGL_DEFER_ASTI` mode. Setting the deferral mode to `XGL_DEFER_ASAP` disallows certain optimizations and forces XGL to do an internal context post after every primitive.

Backing Store

When using a backing store, rendering is generally slower if the window is partially or fully obscured; it will be *much* slower if something is actually drawn into an obscured portion of the window (since that requires rendering into a memory raster). An optimization for multipolylines allows them to be drawn at highest speed into a partially obscured window if nothing is actually drawn into the obscured portion.

Picking

The GX uses the geometry semantic for picking. (See the man page for a complete description of this semantic.)

Line Join

The `XGL_JOIN_DEVICE` value of the attribute `XGL_CTX_LINE_JOIN` is equivalent to `XGL_JOIN_BEVEL`.

Hardware Double Buffering

The GXplus supports hardware double buffering. Double buffering can also be done on the GX using color map double buffering (see the *XGL Programmer's Guide*), but because the GX has an 8-bit deep frame buffer, the number of displayable colors is limited to 16.

X Visuals

The following Xlib visuals are supported:

- 8-bit StaticGray
- 8-bit GrayScale
- 8-bit StaticColor
- 8-bit PseudoColor (default)

Attributes That Affect Acceleration

This section provides an alphabetical list of the attributes affect the rendering performance of the GX and GXplus accelerators. Check the attribute list to ensure that your application program is not setting attributes that will degrade performance.

In the description of each attribute, the attribute name appears on the left, and the attribute's default, if any, appears to the right. Below the attribute name is a brief description of the attribute with a description of how particular settings enable the accelerator to fully accelerate a primitive or prevent the accelerator from fully accelerating a primitive. Below the description is the list of XGL primitives the attribute affects. For a more complete description of each attribute, consult the man page online or see the *XGL Reference Manual*.

XGL_CTX_CLIP_PLANES NULL (none enabled)

This attribute defines which of the sizes possible for view-clip planes are enabled. There is no acceleration if one, but not both, of the planes XGL_CLIP_ZMIN and XGL_CLIP_ZMAX is enabled. Therefore, it is recommended that the ZMIN and ZMAX planes be either both enabled or both disabled.

Affects all primitives.

XGL_3D_CTX_DEPTH_CUE_MODE XGL_DEPTH_CUE_OFF

This attribute turns on or off depth cuing for 3D Contexts. There is no acceleration if any value other than XGL_DEPTH_CUE_OFF is used.

Affects all primitives.

`XGL_3D_CTX_HLHSR_MODE` `XGL_HLHSR_NONE`

This attribute defines the Hidden Line and Hidden Surface Removal method to be used by the application. There is no acceleration if any value other than `XGL_HLHSR_NONE` is used.

Affects all primitives.

`XGL_3D_CTX_LINE_COLOR_INTERP` `FALSE`

This attribute enables or disables interpolation of color between vertices of a polyline. If any value other than the default is used, there is no acceleration.

Affects the following primitive:

```
xgl_multipolyline()
xgl_nurbs_curve()
```

`XGL_3D_CTX_SURF_FACE_CULL` `XGL_CULL_OFF`

This attribute specifies the facet-culling mode XGL applies to surfaces. If any value other than the default is used, these primitives are not fully accelerated.

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_polygon()
xgl_quadrilateral_mesh
xgl_triangle_strip()
xgl_triangle_list()
xgl_nurbs_surface()
```

`XGL_3D_CTX_SURF_FACE_DISTINGUISH` `FALSE`

This attribute determines whether back-facing attributes apply to back-facing surfaces. If any value other than the default is used, these primitives are only partially accelerated.

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()
xgl_triangle_list()
xgl_nurbs_surface()
```

XGL_3D_CTX_SURF_BACK_FPAT
XGL_CTX_SURF_FRONT_FPAT NULL

These attributes specify the raster containing the fill pattern for stipple-filled surfaces. There is no acceleration for primitives using XGL_CTX_SURF_FRONT_FILL_STYLE of XGL_SURF_FILL_STIPPLE or XGL_SURF_FILL_OPAQUE_STIPPLE.

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()
xgl_triangle_list()
xgl_nurbs_surface()
```

XGL_3D_CTX_SURF_FRONT_ILLUMINATION
XGL_3D_CTX_SURF_BACK_ILLUMINATION XGL_ILLUM_NONE

These attributes specify the type of illumination applied to front- and back-facing surfaces respectively. If either attribute is set to XGL_ILLUM_VERTEX or XGL_ILLUM_NONE_INTERP_COLOR, there is no acceleration.

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()
xgl_triangle_list()
xgl_nurbs_surface()
```

XGL_CTX_LINE_STYLE XGL_LINE_SOLID

This attribute specifies the line style (solid, patterned, alt-patterned) for rendering lines and curves. If any value other than the default is used, lines will be slower although still accelerated.

Affects the following primitive:

```
xgl_multipolyline()
xgl_nurbs_curve()
```

`XGL_CTX_LINE_WIDTH_SCALE_FACTOR` 0.0

This attribute specifies the line width of lines being rendered in Device Coordinates. The value is produced by multiplying the specified scale factor by the nominal line width of the device (one pixel).

- 2D lines are accelerated if the value is less than 1.0.
- 3D lines are accelerated if the value is less than or equal to 1.0.
- If a value from 1 to less than 2 is given, 2D and 3D lines are not accelerated.
- If a value of 2 or greater is given, 3D lines are partially accelerated, but 2D lines are not accelerated.

Affects the following primitives:

```
xgl_multipolyline()
xgl_nurbs_curve()
```

`XGL_CTX_EDGE_STYLE` `XGL_LINE_SOLID`

This attribute specifies the line style for surface edges. If `XGL_CTX_SURF_EDGE_FLAG` is set to `FALSE`, surface edges are accelerated regardless of the value of `XGL_CTX_EDGE_STYLE`. If `XGL_CTX_SURF_EDGE_FLAG` is set to `TRUE`, edges are partially accelerated if `XGL_CTX_EDGE_STYLE` is set to `XGL_LINE_SOLID` (the default), but not accelerated if the edge style is set to any value other than solid edges.

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()
xgl_triangle_list()
xgl_nurbs_surface()
```

`XGL_CTX_EDGE_WIDTH_SCALE_FACTOR` 0.0

This attribute specifies the line width of edges being rendered in Device Coordinates. The value is produced by multiplying the specified scale factor by the nominal line width of the device (one pixel). If a value greater than the default is given, and XGL_CTX_SURF_EDGE_FLAG is other than FALSE (the default), primitives drawing polygons with edges are not fully accelerated.

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()
xgl_triangle_list()
xgl_nurbs_surface()
```

Reference Table of Primitives and Attributes

Table 2-1 on page 14 lists the primitives that can be tuned to render with increased performance. Note also the following:

- The xgl_multi_simple_polygon() primitive is not accelerated for quadrilaterals that are bowtie or V-shaped. There is reduced acceleration if one or more of these facet flags (polygon classification flags) are not set:
  - XGL_FACET_FLAG_SIDES_ARE_3
  - XGL_FACET_FLAG_SIDES_ARE_4
  - XGL_FACET_FLAG_SHAPE_CONVEX
- The xgl_polygon() primitive is not accelerated for quadrilaterals that are *bowtie* or *V*-shaped, or for polygons with more than one boundary.
- If a specified transform maps circles to ellipses, lines, or points, circles drawn by xgl_multicircle() are not fully accelerated.
- No primitive is accelerated if XGL_3D_CTX_HLHSR_MODE is set to any value other than XGL_HLHSR_NONE.
- No primitive is accelerated if XGL_3D_CTX_DEPTH_CUE_MODE is set to any value other than XGL_DEPTH_CUE_OFF.

Table 2-1 Primitives and Related Attributes for the GX

Primitive	Attributes that Affect Acceleration
xgl_annotation_text()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_multi_simple_polygon()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_SURF_FACE_CULL XGL_3D_CTX_SURF_FACE_DISTINGUISH XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_3D_CTX_SURF_FRONT_ILLUMINATION XGL_3D_CTX_SURF_BACK_ILLUMINATION XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_multiarc()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_multicircle()	XGL_CTX_CLIP_PLANES XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_multi_elliptical_arc()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE

Table 2-1 Primitives and Related Attributes for the GX *(Continued)*

Primitive	Attributes that Affect Acceleration
xgl_multimarker()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_multipolyline()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_LINE_COLOR_INTERP XGL_CTX_LINE_STYLE XGL_CTX_LINE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_multirectangle()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_nurbs_curve()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_LINE_COLOR_INTERP XGL_CTX_LINE_STYLE XGL_CTX_LINE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE Note that NURBS curves are tessellated into polylines and rendered with the polyline attributes. The NURBS curve attributes also affect performance in that they affect the decomposition of curves into polylines.

Table 2-1 Primitives and Related Attributes for the GX *(Continued)*

Primitive	Attributes that Affect Acceleration
xgl_nurbs_surface()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_SURF_FACE_CULL XGL_3D_CTX_SURF_FACE_DISTINGUISH XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_3D_CTX_SURF_FRONT_ILLUMINATION XGL_3D_CTX_SURF_BACK_ILLUMINATION XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE Note that NURBS surfaces are tessellated into triangle strips and rendered with the triangle strip attributes. The NURBS surface attributes also affect performance in that they affect the decomposition of surfaces into triangle strips.
xgl_polygon()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_SURF_FACE_CULL XGL_3D_CTX_SURF_FACE_DISTINGUISH XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_3D_CTX_SURF_FRONT_ILLUMINATION XGL_3D_CTX_SURF_BACK_ILLUMINATION XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_quadrilateral_mesh()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_SURF_FACE_CULL XGL_3D_CTX_SURF_FACE_DISTINGUISH XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_3D_CTX_SURF_FRONT_ILLUMINATION XGL_3D_CTX_SURF_BACK_ILLUMINATION XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE

Table 2-1 Primitives and Related Attributes for the GX *(Continued)*

Primitive	Attributes that Affect Acceleration
xgl_stroke_text()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE
xgl_triangle_list()	Currently not accelerated.
xgl_triangle_strip()	XGL_CTX_CLIP_PLANES XGL_3D_CTX_SURF_FACE_CULL XGL_3D_CTX_SURF_FACE_DISTINGUISH XGL_3D_CTX_SURF_BACK_FPAT XGL_CTX_SURF_FRONT_FPAT XGL_3D_CTX_SURF_FRONT_ILLUMINATION XGL_3D_CTX_SURF_BACK_ILLUMINATION XGL_CTX_EDGE_STYLE XGL_CTX_EDGE_WIDTH_SCALE_FACTOR XGL_3D_CTX_HLHSR_MODE XGL_3D_CTX_DEPTH_CUE_MODE

Acceleration Across a Network With the Xpex Pipeline 3

An XGL application can run remotely and display on a user's local workstation if the two workstations are part of a network. XGL handles remote rendering automatically by sending X11 or PEX protocol messages to the server.

Rendering Across a Network

When an XGL client program is running remotely, XGL uses the Xpex pipeline to emit PEXlib or Xlib calls. XGL emits PEXlib calls if the server includes the PEX extension and XGL has access to its PEX loadable library. If PEX is not available, XGL emits Xlib calls. PEXlib supports 3D rendering; for 2D rendering, XGL emits Xlib calls.

The Xpex pipeline automatically detects whether it can use PEX to render a primitive. This is determined by the attributes in effect when the application renders the primitive or Gcache. For attribute settings not supported by the PEXlib, XGL emits Xlib calls. To improve performance for 3D rendering over the network, the application should set attributes as noted in the remainder of this chapter.

Note – The application can choose to render through the Xpex pipeline for local rendering by requesting the PEX protocol in a Window Raster Device creation call. In the device creation call, the application OR's in the name of the PEX protocol `XGL_WIN_X_PROTO_PEX` with the constant `XGL_WIN_X`. XGL will try to use the PEX protocol to communicate with the graphics device. If the PEX server or the PEX loadable library is not available, an error is issued, and the Window Raster is not created. See the *XGL Programmer's Guide* or the *XGL Reference Manual* for more information on requesting the PEX protocol at Device creation.

General Acceleration Notes

For acceleration with remote rendering, the application should avoid the following situations whenever possible. The Xpex pipeline emits Xlib calls in these cases:

- When the application asks for double buffering, but the PEX server does not support the X Multibuffering extension (MBX).
- When the application is attempting to draw to both the front and back buffers simultaneously in double-buffered rendering.
- When hidden surface elimination is enabled.
- When using a 2D Context.
- When the drawing destination is the accumulation buffer.
- When the number of model clip planes requested by the application is greater than the number supported by the PEX server.
- When all four XGL X-Y clipping flags are not specified as (`XGL_CLIP_XMIN | XGL_CLIP_XMAX | XGL_CLIP_YMIN | XGL_CLIP_YMAX`). The PEX protocol provides only one flag for all X-Y clipping, not two for each of X and Y, as XGL does.
- When picking is enabled.

Depth Cueing

The Xpex pipeline assumes that depth cueing is supported on the server when the application asks for it. PEXlib provides no way to ascertain whether this is true.

Line Widths

The application cannot determine whether fractional line widths can be realized with PEX, so XGL assumes that PEX servers can render fractional widths.

Use Gcache to Improve Performance

Using Gcaches can significantly improve performance with the Xpex pipeline. The information in the Gcache is stored by XGL on the remote PEX server, and therefore does not need to be sent over the network every time the primitive is rendered. This greatly reduces both network traffic and the time it takes to render a primitive.

Line Patterns

The predefined XGL line patterns xgl_lpat_dotted, xgl_lpat_dashed, and xgl_lpat_dash_dot are accelerated.

Markers

The predefined XGL markers xgl_marker_plus, xgl_marker_circle, xgl_marker_cross, and xgl_marker_asterisk are accelerated if the PEX server has corresponding markers defined.

Lights

The application should limit the number of lights to the number of lights supported by the PEX implementation. To determine the maximum number of lights, inquire the PEX server.

Supported Point Types

The pipeline uses the PEX protocol when the point type of a primitive is as shown in Table 3-1. Otherwise, the pipeline uses Xlib functions.

Table 3-1 Point Types Supported in the Xpex Pipeline

Primitive	Point Type
xgl_multipolyline() xgl_multi_simple_polygon()	XGL_PT_F3D XGL_PT_NORMAL_F3D XGL_PT_FLAG_F3D XGL_PT_NORMAL_FLAG_F3D XGL_PT_COLOR_F3D XGL_PT_COLOR_NORMAL_F3D XGL_PT_COLOR_FLAG_F3D XGL_PT_COLOR_NORMAL_FLAG_F3D
xgl_multimarker()	XGL_PT_F3D XGL_PT_COLOR_F3D
xgl_quadrilateral_mesh() xgl_triangle_strip()	XGL_PT_F3D XGL_PT_COLOR_F3D XGL_PT_NORMAL_F3D XGL_PT_COLOR_NORMAL_F3D

Attributes That Affect Acceleration

This section provides an alphabetical list of the attributes that affect PEX performance. In the description of each attribute, the attribute name appears on the left, and the attribute's default, if any, appears to the right. Below the attribute name is a brief description of the attribute with a description of how particular settings enable the accelerator to fully accelerate a primitive or prevent the accelerator from fully accelerating a primitive.

XGL_3D_CTX_JITTER_OFFSET 0.0

This attribute specifies the amount by which geometry should be offset in DC before drawing. There is no acceleration in the Xpex pipeline when XGL_CTX_JITTER_OFFSET is not equal to (0,0).

Affects all primitives.

XGL_CTX_ROP XGL_ROP_COPY

This attribute defines the raster operation. When XGL_CTX_ROP is set to any value other than XGL_ROP_COPY, no acceleration occurs.

Affects all primitives.

XGL_CTX_PLANE_MASK All bits set to 1.

This attribute defines the pixel plane mask. When XGL_PLANE_MASK is not set to all 1's (draw to all planes), there is no acceleration.

Affects all primitives.

XGL_CTX_RENDER_BUFFER XGL_RENDER_DRAW_BUFFER

This attribute controls which buffer is used for rendering. When the application attempts to draw to the front and back buffers simultaneously, as in XGL_CTX_RENDER_BUFFER = XGL_RENDER_DISPLAY_BUFFER | XGL_RENDER_DRAW_BUFFER, there is no acceleration.

Affects all primitives.

XGL_3D_CTX_DEPTH_CUE_INTERP TRUE

This attribute defines the depth cue interpolation mode in a Context. When XGL_3D_CTX_DEPTH_CUE_INTERP is not TRUE, there is no acceleration.

Affects all primitives.

XGL_3D_CTX_HLHSR_MODE XGL_HLHSR_NONE

This attribute defines the hidden line and hidden surface removal method to be used by the application. There is no acceleration if any value other than XGL_HLHSR_NONE is used.

Affects all primitives.

XGL_3D_CTX_VIEW_CLIP_PLUS_W_ONLY TRUE

This attribute controls clipping in homogeneous space. There is no acceleration if XGL_3D_CTX_VIEW_CLIP_PLUS_W_ONLY is FALSE.

Affects all primitives.

XGL_CTX_SURF_FRONT_FILL_STYLE
XGL_3D_CTX_SURF_BACK_FILL_STYLE

The surface fill attributes define how a surface is filled. If either attribute is set to XGL_SURF_FILL_SOLID, XGL_SURF_FILL_HOLLOW, or XGL_SURF_FILL_EMPTY, primitives are accelerated.

Affects the following primitives:
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()

XGL_3D_CTX_SURF_SILHOUETTE_EDGE_FLAG FALSE
This attribute controls whether silhouette edges are drawn around a surface. If this attribute is set to TRUE, no acceleration occurs.

Affects the following primitives:
xgl_multi_simple_polygon()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()

XGL_3D_CTX_SURF_TRANSP_METHOD
This attribute controls how transparent surfaces are rendered. If this attribute is set to either XGL_TRANSP_NONE or XGL_TRANSP_SCREEN_DOOR, the primitive is accelerated.

Affects the following primitives:
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()

XGL_3D_CTX_SURF_DC_OFFSET FALSE
This attribute offsets in z the DC coordinates of 3D polygons. If any value other than the default FALSE is used, no acceleration occurs.

Affects the following primitives:
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()

XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT
XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT

These attributes define which components of the illumination equation are used. The component flags are XGL_LIGHT_ENABLE_COMP_AMBIENT, XGL_LIGHT_ENABLE_COMP_DIFFUSE, and XGL_LIGHT_ENABLE_COMP_SPECULAR. Acceleration occurs when all flags are OR'ed together, when ambient and diffuse flags are used, and when only the XGL_LIGHT_ENABLE_COMP_AMBIENT is used. Otherwise, no acceleration occurs.

Affects the following primitives:
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()

XGL_3D_CTX_SURF_GEOM_NORMAL XGL_GEOM_NORMAL_FIRST_POINTS

This attribute controls how surface normals are calculated when they are not provided as part of the application data. The default value XGL_GEOM_NORMAL_FIRST_POINTS is accelerated.

Affects the following primitives:
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()

xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()

XGL_3D_CTX_SURF_NORMAL_FLIP FALSE

This attribute specifies whether vertex and facet normals are flipped. If this attribute is set to TRUE, then no acceleration occurs.

Affects the following primitives:

xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()

XGL_CTX_LINE_COLOR_SELECTOR XGL_LINE_COLOR_VERTEX

This attribute selects the source of a line's color. If a line's vertex has color and the color is not XGL_LINE_COLOR_VERTEX, there is no acceleration.

Affects the following primitives:

xgl_multipolyline()
xgl_nurbs_curve()

XGL_CTX_LINE_PATTERN

This attribute is used to set the handle of a line pattern object. The predefined XGL line patterns xgl_lpat_dotted, xgl_lpat_dashed, and xgl_lpat_dash_dot are accelerated.

Affects the following primitives:

xgl_multipolyline()
xgl_nurbs_curve()

XGL_CTX_LINE_JOIN XGL_JOIN_DEVICE

This attribute defines the shape of joins between line segments or curves. Only XGL_JOIN_DEVICE is accelerated for lines with XGL_LINE_WIDTH_SCALE_FASTER set to a value greater than 1.0

Affects the following primitives:

xgl_multipolyline()
xgl_nurbs_curve()

XGL_CTX_LINE_CAP XGL_CAP_BUTT

This attribute defines the shape of the end points of line segments and curves. Only XGL_CAP_BUTT is accelerated for lines with XGL_LINE_WIDTH_SCALE_FASTER set to a value greater than 1.0.

Affects the following primitives:

```
xgl_multipolyline()
xgl_nurbs_curve()
```

XGL_CTX_EDGE_PATTERN

This attribute is used to set the handle of a edge pattern object. The predefined XGL patterns xgl_lpat_dotted, xgl_lpat_dashed, and xgl_lpat_dash_dot are accelerated.

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()
```

XGL_CTX_EDGE_JOIN XGL_JOIN_DEVICE

This attribute defines the shape of joins between edge segments or curves. Only XGL_JOIN_DEVICE is accelerated for edges with XGL_LINE_WIDTH_SCALE_FASTER set to a value greater than 1.0

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()
```

`XGL_CTX_EDGE_CAP` `XGL_CAP_BUTT`

This attribute defines the shape of the end points of edge segments and curves. Only `XGL_CAP_BUTT` is accelerated for edges with `XGL_LINE_WIDTH_SCALE_FASTER` set to a value greater than 1.0.

Affects the following primitives:

```
xgl_multi_simple_polygon()
xgl_multiarc()
xgl_multicircle()
xgl_multi_elliptical_arc()
xgl_multirectangle()
xgl_polygon()
xgl_quadrilateral_mesh()
xgl_triangle_strip()
```

Reference Table of Primitives and Attributes

Table 3-2 on page 29 lists the primitives that can be tuned to render with increased performance. The following attributes affect all primitives:

- `XGL_3D_CTX_JITTER_OFFSET`
- `XGL_CTX_ROP`
- `XGL_CTX_PLANE_MASK`
- `XGL_CTX_RENDER_BUFFER`
- `XGL_3D_CTX_HLHSR_MODE`
- `XGL_3D_CTX_DEPTH_CUE_INTERP`
- `XGL_3D_CTX_VIEW_CLIP_PLUS_W_ONLY`

Table 3-2 Primitives and Related Attributes for Xpex

Primitives	Attributes that Affect Acceleration
xgl_annotation_text()	Currently not accelerated.
xgl_multi_simple_polygon()	XGL_3D_CTX_SURF_FRONT_FILL_STYLE XGL_3D_CTX_SURF_BACK_FILL_STYLE XGL_3D_CTX_SURF_SILHOUETTE_EDGE_FLAG XGL_3D_CTX_SURF_TRANSP_METHOD XGL_3D_CTX_SURF_DC_OFFSET XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT XGL_3D_CTX_SURF_GEOM_NORMAL XGL_3D_CTX_SURF_NORMAL_FLIP XGL_CTX_EDGE_PATTERN XGL_CTX_EDGE_JOIN XGL_CTX_EDGE_CAP
xgl_multiarc()	XGL_3D_CTX_SURF_FRONT_FILL_STYLE XGL_3D_CTX_SURF_BACK_FILL_STYLE XGL_3D_CTX_SURF_TRANSP_METHOD XGL_3D_CTX_SURF_DC_OFFSET XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT XGL_3D_CTX_SURF_GEOM_NORMAL XGL_3D_CTX_SURF_NORMAL_FLIP XGL_CTX_EDGE_PATTERN XGL_CTX_EDGE_JOIN XGL_CTX_EDGE_CAP
xgl_multicircle()	XGL_3D_CTX_SURF_FRONT_FILL_STYLE XGL_3D_CTX_SURF_BACK_FILL_STYLE XGL_3D_CTX_SURF_TRANSP_METHOD XGL_3D_CTX_SURF_DC_OFFSET XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT XGL_3D_CTX_SURF_GEOM_NORMAL XGL_3D_CTX_SURF_NORMAL_FLIP XGL_CTX_EDGE_PATTERN XGL_CTX_EDGE_JOIN XGL_CTX_EDGE_CAP

Table 3-2 Primitives and Related Attributes for Xpex

Primitives	Attributes that Affect Acceleration
xgl_multi_elliptical_arc()	XGL_3D_CTX_SURF_FRONT_FILL_STYLE XGL_3D_CTX_SURF_BACK_FILL_STYLE XGL_3D_CTX_SURF_TRANSP_METHOD XGL_3D_CTX_SURF_DC_OFFSET XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT XGL_3D_CTX_SURF_GEOM_NORMAL XGL_3D_CTX_SURF_NORMAL_FLIP XGL_CTX_EDGE_PATTERN XGL_CTX_EDGE_JOIN XGL_CTX_EDGE_CAP
xgl_multimarker()	Accelerated with certain markers types if the server has corresponding markers defined.
xgl_multipolyline()	XGL_3D_CTX_LINE_COLOR_SELECTOR XGL_CTX_LINE_PATTERN XGL_CTX_LINE_JOIN XGL_CTX_LINE_CAP
xgl_multirectangle()	XGL_3D_CTX_SURF_FRONT_FILL_STYLE XGL_3D_CTX_SURF_BACK_FILL_STYLE XGL_3D_CTX_SURF_TRANSP_METHOD XGL_3D_CTX_SURF_DC_OFFSET XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT XGL_3D_CTX_SURF_GEOM_NORMAL XGL_3D_CTX_SURF_NORMAL_FLIP XGL_CTX_EDGE_PATTERN XGL_CTX_EDGE_JOIN XGL_CTX_EDGE_CAP
xgl_nurbs_curve()	XGL_3D_CTX_LINE_COLOR_SELECTOR XGL_CTX_LINE_PATTERN XGL_CTX_LINE_JOIN XGL_CTX_LINE_CAP
xgl_nurbs_surface()	Currently not accelerated.

Table 3-2 Primitives and Related Attributes for Xpex

Primitives	Attributes that Affect Acceleration
`xgl_polygon()`	XGL_3D_CTX_SURF_FRONT_FILL_STYLE XGL_3D_CTX_SURF_BACK_FILL_STYLE XGL_3D_CTX_SURF_SILHOUETTE_EDGE_FLAG XGL_3D_CTX_SURF_TRANSP_METHOD XGL_3D_CTX_SURF_DC_OFFSET XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT XGL_3D_CTX_SURF_GEOM_NORMAL XGL_3D_CTX_SURF_NORMAL_FLIP XGL_CTX_EDGE_PATTERN XGL_CTX_EDGE_JOIN XGL_CTX_EDGE_CAP
`xgl_quadrilateral_mesh()`	XGL_3D_CTX_SURF_FRONT_FILL_STYLE XGL_3D_CTX_SURF_BACK_FILL_STYLE XGL_3D_CTX_SURF_SILHOUETTE_EDGE_FLAG XGL_3D_CTX_SURF_TRANSP_METHOD XGL_3D_CTX_SURF_DC_OFFSET XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT XGL_3D_CTX_SURF_GEOM_NORMAL XGL_3D_CTX_SURF_NORMAL_FLIP XGL_CTX_EDGE_PATTERN XGL_CTX_EDGE_JOIN XGL_CTX_EDGE_CAP
`xgl_stroke_text()`	Currently not accelerated.
`xgl_triangle_list()`	Currently not accelerated.
`xgl_triangle_strip()`	XGL_3D_CTX_SURF_FRONT_FILL_STYLE XGL_3D_CTX_SURF_BACK_FILL_STYLE XGL_3D_CTX_SURF_SILHOUETTE_EDGE_FLAG XGL_3D_CTX_SURF_TRANSP_METHOD XGL_3D_CTX_SURF_DC_OFFSET XGL_3D_CTX_SURF_FRONT_LIGHT_COMPONENT XGL_3D_CTX_SURF_BACK_LIGHT_COMPONENT XGL_3D_CTX_SURF_GEOM_NORMAL XGL_3D_CTX_SURF_NORMAL_FLIP XGL_CTX_EDGE_PATTERN XGL_CTX_EDGE_JOIN XGL_CTX_EDGE_CAP