



PEXlib5.2 emulation implementation
by Kai Wang

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in
Computer Science
Montana State University
© Copyright by Kai Wang (1995)

Abstract:

The PEXLib5.2 draft specification distributed by X consortium is under public review. In the specification, new function interfaces are defined. New primitives, such as cone and sphere are provided. Also some new rendering attributes, such as transparency and texture mapping, are supplied.

As my thesis project, I implemented the PEXlib5.2 OCC style functions defined in the PEXlib5.2 draft specification based on PEXlib5.1. With the emulation functions, user's application can run in the PEX5.1 environment just as in the PEX5.2 environment. In this paper, I introduce some background of PEX and PEXlib, analyze the features of PEXlib5.2, and then describe how to implement the PEXlib5.2 functions based on PEXlib5.1

PEXlib5.2 EMULATION IMPLEMENTATION

by
Kai Wang

A thesis submitted in partial fulfillment
of the requirements for the degree
of
Master of Science
in
Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana
November, 1995

N378
W1856

APPROVAL

of a thesis submitted by

Kai Wang

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

1/2/96
Date

J. Dunbig Stanley
Chairperson, Graduate Committee

Approved for the Major Department

1/2/96
Date

J. Dunbig Stanley
Head, Major Department

Approved for the College of Graduate Studies

1/7/96
Date

P. L. Brown
Graduate Dean

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Signature *Mary Kay*
Date 12/29/95

TABLE OF CONTENTS

	Page
APPROVAL PAGE	ii
STATEMENT OF PERMISSION TO USE	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vi
LIST OF FIGURES	vii
ABSTRACT	viii
INTRODUCTION	1
X Window System And PEX	1
PEXlib 5.1	3
PEXlib5.2 AND ITS FEATURES	8
Output Command Context(OCC) Interface	8
Flexible Data Format	9
Primitives	11
Rendering Attributes	14
PEXlib5.2 EMULATION IMPLEMENTATION	19
OC Context Functions	19

Output Primitive Function	19
Texture Mapping Utility Functions	32
CONCLUSION AND FUTURE DEVELOPMENT	35
REFERENCES	36

List of Tables

1	PEXlib5.1 Function Categories	5
2	Relationship between OCC and non-OCC Primitive Functions	12
3	New Primitive Functions Defined In PEXlib5.2	15
4	Transparency Methods	16
5	Attribute Fields of Mapping Table	18
6	OCC Mask Bits	21

List of Figures

1	The Architecture of X Window System	2
2	Relationship Between X And PEX	3
3	High Level View of PEX	4
4	Renderer And Structure Store Inside PEX	4
5	Stride data model.	11
6	Vertex Unpacked Data Model.	12
7	Different Arcs Drawn With PEXOCCArcs Function	23
8	Different Arcs Areas Drawn With PEXOCCArcAreas Function	25
9	The Selected Control Points For A Cone	27
10	A Cone Rendered With PEXOCCones Function	28
11	The Selected Control Points For A Sphere	29
12	A Sphere Rendered With PEXOCSpheres Function	30
13	Compute The Texture Map Coordinates	34

ABSTRACT

The PEXlib5.2 draft specification distributed by X consortium is under public review. In the specification, new function interfaces are defined. New primitives, such as cone and sphere are provided. Also some new rendering attributes, such as transparency and texture mapping, are supplied.

As my thesis project, I implemented the PEXlib5.2 OCC style functions defined in the PEXlib5.2 draft specification based on PEXlib5.1. With the emulation functions, user's application can run in the PEX5.1 environment just as in the PEX5.2 environment. In this paper, I introduce some background of PEX and PEXlib, analyze the features of PEXlib5.2, and then describe how to implement the PEXlib5.2 functions based on PEXlib5.1.

INTRODUCTION

X Window System And PEX

X Window System is a network-oriented windowing system. It provides a network-transparent and vendor-independent operating environment for distributed graphics applications.

X Window System uses a client-server architecture. X server is the piece of the program that controls the display resources, such as keyboard, mouse, and one or more display screens. X client is the user's application program. A client connects with X server to request services or responds to events through X network protocol. The network protocol interface is designed to work either within a single CPU or between CPUs. So the user's client can run locally or on a remote host that can connect with the server. To mask the detail of X network protocol, a C language function package known as Xlib is provided with the X Window System. Xlib is the application program interface to the X network protocol. Clients communicate with the server using functions in Xlib. The X Window System architecture is shown in Figure 1.

The X Window System only provides text and two-dimensional graphics. PEX is an extension to the X Window System for supporting three-dimensional graphics. PEX provides all the common features found in most modern 3D graphics system, but provides them in a way that is seamlessly integrated

with X. In fact, PEX is an extension to the X network protocol. As in the X protocol, a PEX application client communicates with the PEX server using the PEX protocol. The PEX server contains the PEX server extension, which receives and interprets the PEX protocol messages and executes the PEX requests. Just as Xlib is the application program interface to the X protocol, PEXlib subroutine package is provided as program interface to PEX protocol. The applications use functions in PEXlib to create and send PEX protocol messages. Figure 2 shows the relationship between X and PEX.

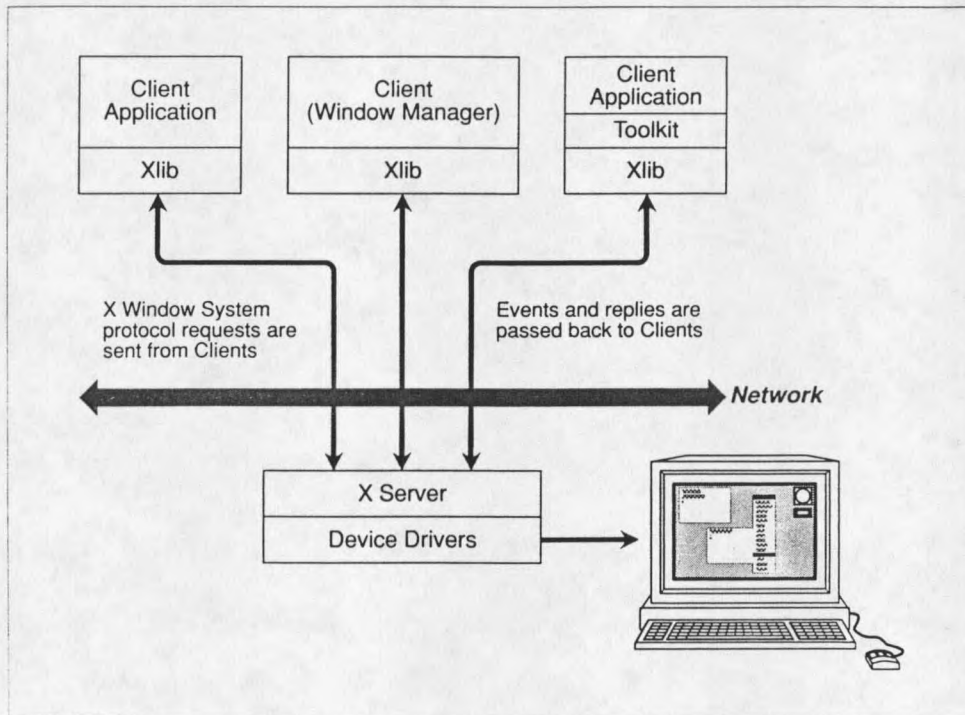


Figure 1: The Architecture of X Window System

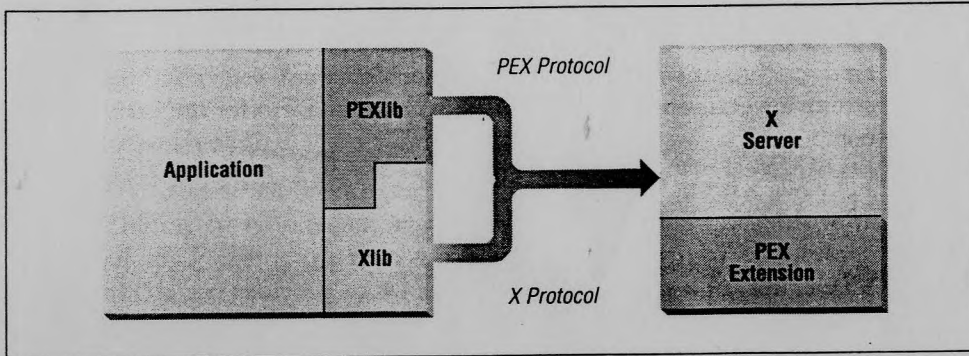


Figure 2: Relationship Between X And PEX

PEXlib 5.1

PEXlib5.1 is a C programming interface for PEX protocol. It is a high-level graphics library. In PEXlib a graphic image is composed of primitives and their attributes. Primitives are basic graphic objects like lines and polygons. Attributes control the appearance of primitives, such as color and orientation. An application sends output commands by using PEXlib functions to have PEX server render the primitives and their attributes. Figure 3 shows a high level view of PEX.

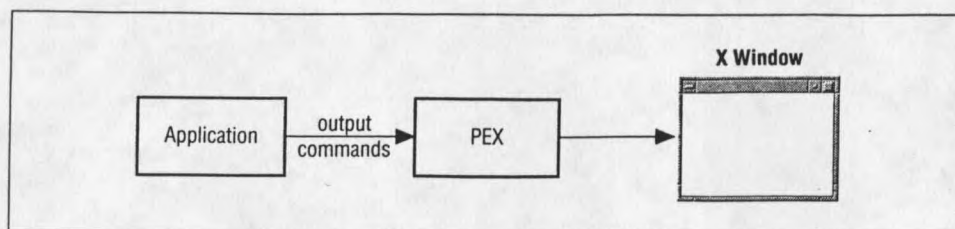


Figure 3: High Level View of PEX

There are two forms for rendering in PEX: immediate mode and structure mode. In immediate mode, the server renders primitives as soon as it receives them. The application must send the primitives and attributes that comprise the image each time the picture is generated. In structure mode, primitives, attributes and other information can be stored for later processing. The image can be regenerated simply by redrawing the contents of each structure. Figure 4 shows a renderer and a structure store inside PEX.

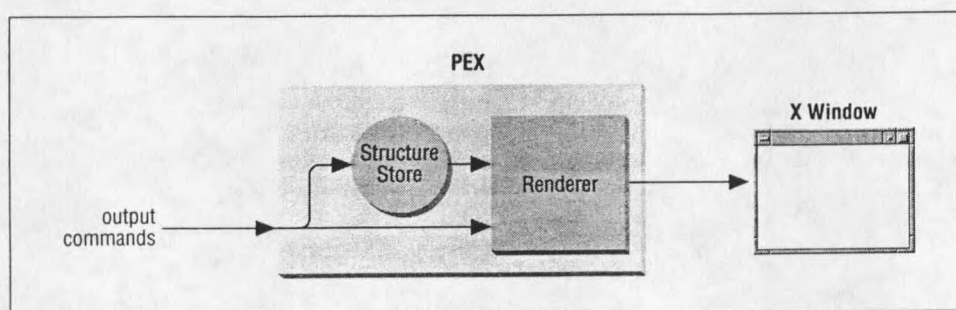


Figure 4: Renderer And Structure Store Inside PEX

PEXlib 5.1 is PEXlib fourth revision. It contains over 200 functions. Table 1 list the catégories of PEXlib 5.1 functions.

Table 1: PEXlib5.1 Function Categories

Category	Description
Startup and features	Initialize PEX, determine the capabilities of PEX servers.
Renderers	Create renderer, operate on renderer's attributes, draw primitives without storing them internally.
Structures	Create structure, operate on structures and store primitives in structure.
Line primitives and attributes	Generate polyline and polyline set output primitives and set their attributes.
Area primitives and attributes	Generate polygons and other multifaceted output primitives and set their attributes.
B-splines and attributes	B-spline curve and surface primitives and their attributes.
Texts and attributes	Draw texts and set their attributes.
Markers and attributes	Draw Marks and set their attributes.

Table 1: PEXlib5.1 Function Categories(continue)

Category	Description
Color	Define colormaps, set color approximation, and other color-related functions.
Lighting and shading	define light sources, the reflective properties of primitives, and specify shading smoothness.
Modeling and transformations	Use geometric transformations and hierarchy to model objects and collections of objects.
Viewing	Look at models from different perspective.
Lookup table	operations on PEX lookup table resource.
Pipeline context	Functions on PEX pipeline context resource.
Name sets	Functions on PEX name set resource.
Picking	select primitives from those displayed on the screen.

Table 1:PEXlib5.1 Function Categories(continue)

Category	Description
Spatial searches	Search a structure network for primitives that lie within a given region of world coordinates.
Font	Select different fonts for text.
Miscellaneous utilities	Other utility functions provided by PEX.

PEXlib5.2 AND ITS FEATURES

X consortium distributed a draft specification of PEXlib5.2 for public review. There are many new features of PEXlib 5.2 provided in the draft specification. A more convenient programming interface called Output Command Context is used for output primitives. Graphic data can be supplied in flexible data formats. In the specification, new primitives are added, such as arc, sphere and cone. New rendering attributes are provided, such as texture mapping, alpha blending and transparency.

Output Command Context(OCC) Interface

In PEXlib 5.1, the interface for the output command functions is defined as the explicit interface. The explicit interface uses the same first three arguments: `display`, which specifies the display connection; `resource_id`, which specifies the resource identifier for the targeted renderer or structure; and `req_type`, which specifies whether the application renders the output commands immediately, or stores the output commands in a structure. The explicit interface requires that you specify all arguments on every output command function you invoke.

In the PEXlib5.2 specification, a new interface for the output command functions, Output Command Context, is defined. In the Output Command Context interface, a data structure called Output Command Context (OC Context) is used to replace the first three arguments in PEXlib5.1 output command functions. The OC Context is an opaque structure that contains many of the arguments which are commonly found in output command functions. The user's application can use a set of special OC manipulation functions to modify the fields in the opaque OC Context, and then use the OC context in subsequent invocations of output command functions.

The Output Command Context interface has far fewer arguments than the PEXlib5.1 explicit interface, making coding easier and improving performance. In order to guarantee backwards compatibility, the explicit interface defined for PEXlib5.1 is included in the PEXlib5.2 specification. Output command functions defined in PEXlib5.2 have two forms: the original PEXlib5.1 format and OC Context format. But all the new primitive functions defined in the PEXlib5.2 specification only have the OCC form.

Flexible Data Format

In the PEXlib5.2 specification, data format flexibility is provided for primitives that have facet and/or vertex data parameters. We can supply the graphical data (coordinates, vertex attributes, facet attributes, and floating-point data) in three different forms. The three different forms are packed, stride, and unpacked.

• Packed Data Form

The packed form requires user to format the data into packed data structures defined in PEXlib. This is the only form supported in PEXlib5.1. When passing the facet or vertex data structure to OCC style functions, we need to set the `surface_attributes`, `surface_vertex_attributes`, `line_vertex_attributes`, `color_type`, and `data_model` fields in OC Context with correct values.

For PEXlib5.2, sometimes we need to use facet or vertex data that include floating point data. But PEXlib does not provide a set of structure type definition for the data form. Hence, we need to design our own data structure to pass the facet or vertex parameters. There are required orders for the structure definition:

For facet data, the required order is:

PEXColor *	one of the PEXlib color types, if provided.
PEXVector	normal, if provided.
float[n]	floating point data, if provided.

For vertex data, the required order is:

PEXCoord	Center.
PEXColor *	one of the PEXlib color types, if provided.
PEXVector	normal, if provided.
unsigned int	edges, if provided.
float[n]	floating point data.

◦ Stride Data Form

The stride data form is more flexible. It allows the user to supply facet or vertex data formatted in application-defined structured arrays without the need to copy the data into the PEXlib-defined structures when invoking the PEXlib functions. Each structure in the structured array corresponds to data for a single facet or vertex along with other application specific data.

When we want to use the stride data interface, we set `data_model` in OC context to `PEXDataStride`, and set the OC context values we have to set for packed. In addition, we need to set `facet_stride` or `vertex_stride` in stride structure to the size of each array element. and set the offset members to the offsets of the corresponding facet or vertex data items within the array element. Figure 5 shows the vertex stride data model.

◦ Unpacked Data Form

The unpacked data interface allows user to supply the facet or vertex data in separate lists for each data type. When using this data interface,

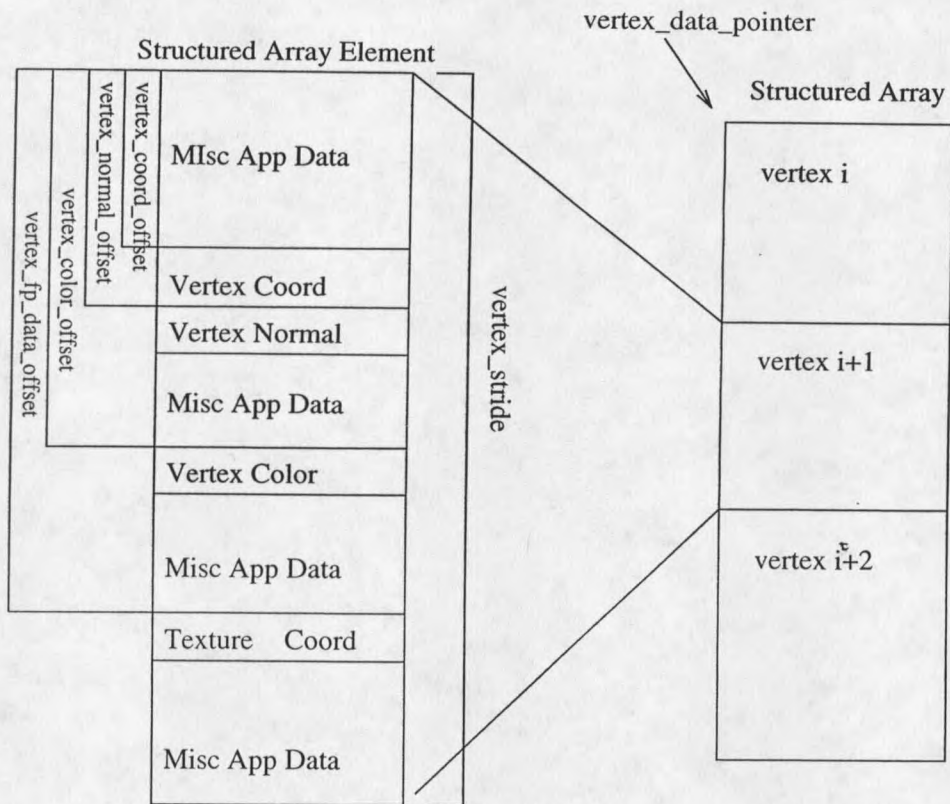


Figure 5: Stride data model.

we set the data_model OCC member to PEXDataUnpacked and set the corresponding size fields in the OCC. In addition, we need to place the pointers to each of the data arrays in PEXUnpackedFacetData or PEXUnpackedVertexData structure and pass the address of the structure. Figure 6 shows the vertex unpacked data model.

Primitives

In PEXlib5.2, primitive functions defined in PEXlib5.1 have two forms: the original PEXlib5.1 format and the corresponding OC Context style function. Some primitives simply have their OCC version of the functions. Some primitives are reorganized into new OCC functions so that one OCC function can serve the role of several non-OCC functions. Table 2 lists the relationship between OCC and non-OCC primitive functions.

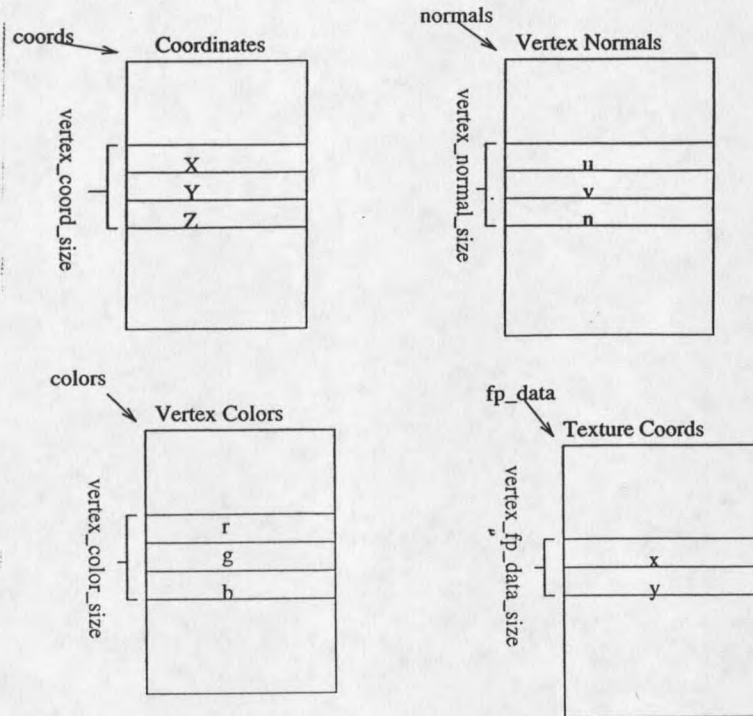


Figure 6: Vertex Unpacked Data Model.

Table 2: Relationship between OCC and non-OCC Primitive Functions

OCC Form	non-OCC Form
PEXOCCAnnotationText	PEXAnnotationText
PEXOCCAnnotationText2D	PEXAnnotationText2
PEXOCCArcAreas	not available
PEXOCCArcs	not available
PEXOCCCellArray	PEXCellArray
PEXOCCCellArray2D	PEXCellArray2D
PEXOCCCones	not available
PEXOCCEncodedAnnoText	PEXEncodedAnnoText
PEXOCCEncodedAnnoText2D	PEXEncodedAnnoText2D
PEXOCCEncodedText	PEXEncodedText

Table 2(continue)

OCC Form	non-OCC Form
PEXOCCEncodedText2D	PEXEncodedText2D
PEXOCCExtendedCellArray	PEXExtendedCellArray
PEXOCCFillArea	PEXFillArea
PEXOCCFillArea	PEXFillArea2D
PEXOCCFillArea	PEXFillAreaWithData
PEXOCCFillAreaSet	PEXFillAreaSet
PEXOCCFillAreaSet	PEXFillAreaSet2D
PEXOCCFillAreaSet	PEXFillAreaSetWithData
PEXOCCGDP	PEXGDP
PEXOCCGDP	PEXGDP2D
PEXOCCIndexedFillAreaSets	PEXSetOfFillAreaSets
PEXOCCIndexedTriangles	not available
PEXOCCMarkers	PEXMarkers
PEXOCCMarkers	PEXMarkers2D
PEXOCCMarkers	PEXMarkersWithData
PEXOCCNURBCurve	PEXNURBCurve
PEXOCCNURBCurve	PEXNURBCurveWithColor
PEXOCCNURBSurface	PEXNURBSurface
PEXOCCNURBSurface	PEXNURBSurfaceWithData

Table 2(continue)

OCC Form	non-OCC Form
PEXOCCPolyline	PEXPolyline
PEXOCCPolyline	PEXPolyline2D
PEXOCCPolylines	PEXPolylineSetWithData
PEXOCCPolyTriangle	not available
PEXOCCQuadrilateralMesh	PEXQuadrilateralMesh
PEXOCCSpheres	not available
PEXOCCText	PEXText
PEXOCCText2D	PEXText2D
PEXOCCTriangleFan	not available
PEXOCCTriangles	not available
PEXOCCTriangleStrip	PEXTriangleStrip

For PEX5.2, several new primitives are added. PEXlib5.2 defines primitive functions for these new primitives. All of the new primitive functions are only in OCC form. Table 3 lists the new primitive functions defined in PEXlib5.2.

Rendering Attributes

PEX5.2 provides many new rendering attributes. The important ones of these new attributes include Transparency, Alpha-blend, Text mapping.

• Transparency

PEX5.2 provides rendering attribute that applies transparency when rendering primitives. For transparency computation, the alpha component of the primitive's color is used. It is a value between 0.0 and 1.0. A pixel is transparent if its alpha value is 0.0; it is opaque if its

Table 3: New Primitive Functions Defined In PEXlib5.2

Function	Description
PEXOCCArcAreas	drawing circular or elliptical arc area.
PEXOCCArcs	drawing circular or elliptical arcs.
PEXOCCCones	drawing cones.
PEXOCCIndexedTriangle	drawing a set of triangle defined with the connectivity indices.
PEXOCCPolyTriangle	drawing a set of triangle whose vertices form any combination of triangle subprimitives.
PEXOCCSpheres	drawing spheres.
PEXOCCTriangleFan	drawing a set of triangles, their vertices are arranged in fan form.
PEXOCCTriangles	drawing a set of triangles. Every three vertices in the vertex list define one independent triangle.

alpha value is 1.0. We can define the alpha value by using `PEXOCC-SetAlphaBlend` output attribute function. The default alpha value for surface primitives is 1.0 - transmission coefficient.

PEXlib5.2 defines five different transparency methods. These methods are listed in table 3.

Table 4: Transparency Methods

Method	Description
<code>PEXTransparencyNone</code>	no transparency.
<code>PEXTransparencyScreenDoor</code>	screen-door transparency.
<code>PEXTransparencyAlphaBlend</code>	using alpha blending.
<code>PEXTransparencyTwoPass</code>	In the first pass, render only those primitives or portions of primitives with an alpha value of 1.0; render all other primitives in the second pass.
<code>PEXTransparencyMultiPass</code>	require multipass rendering.

We indicate method used to apply transparency by setting renderer's transparency method attribute.

• Alpha Blending

Alpha blending is one of PEX5.2 renderer's transparency methods. Alpha blending modulates the color coming out of the depth cueing and the previously rendered pixel according to the alpha blend function in the renderer's pipeline state. We can get different alpha blend functions by selecting the values of blend factors `src_blend_func` and `dest_blend_func`. The blend factors are set by using the following attribute output function:

```

void PEXOCCSetAlphaBlend(
    PEXOCC context,
    int src_blend_func,
    int dest_blend_func
)

```

Alpha blending is used if we set the renderer's transparency method to `PEXTransparencyAlphaBlend`.

• Texture Mapping

PEX5.2 supplies Texture Mapping mechanism that simulates material properties for surface primitives by adding visual detail with color. Texture mapping is enabled by setting the interior style to `PEXInteriorStyleTexture`.

In PEX5.2 only color mipmap texture maps are supported. A color mipmap is a pre-computed area sampling mechanism which reduces the cost of approximating the average color over a large number of texels. We create a texture map with `PEXCreateMipmap` or `PEXCreateMipmapTMFromResources` functions. A texture map resource is put into the texture mapping lookup table. We can enable the entries of the mapping table with the `PEXOCCSetSurfaceActiveTextures` function which modifies the pipeline state's active textures. Each entry of mapping table represents a text map and an associated set of attributes. User controls how to apply texture mapping by setting the attribute values. Table 4 lists all the attributes.

Table 5: Attribute Fields of Mapping Table

Field	Description
rendering control	control where in the rendering pipeline the text map is applied.
parameterization method	define how texture map coordinates are derived.
composition method	specify how the texture map is blended with the primitive's existing data.
min and mag	specify how texels are sampled from the texture map and mapped to primitive pixels when the area covered by different number of texel and pixel.
boundary condition	specify how to apply the texture to primitive when the texture coordinates is beyond the boundary of the texture map.
clamping method	specify whether to discontinue texturing or to use an explicit color when boundary condition Clamp-ExplicitMethod is selected.

PEXLIB5.2 EMULATION IMPLEMENTATION

The X consortium's draft specification for PEXlib5.2 is under public review. As my thesis, I implemented part of the OCC style functions which are defined in the PEXlib5.2 specification based on PEXlib5.1.

OC Context Functions

In PEXlib5.2 specification PEXOCC is defined as a pointer that points to a opaque structure. In my implementation, I define PEXOCC as a pointer which points to PEXOCCValues data structure.

PEXOCCValues * PEXOCC

PEXOCCValues is a data structure defined in PEXlib5.2.

I implemented PEXCreateOCC function to create an OC Context. PEXCreateOCC function return a PEXOCC initialized with default values. We can change OC Context default values using using PEXSetOCCValueMask function to set bits in the bitmask that correspond to the fields in the OCC that we want to modify. Table 5 lists the OCC mask bits.

I also implemented the set of PEXSetOCC functions that allow user to modify one value each time.

Output Primitive Function

For those OCC style output primitive functions that have their corresponding non-OCC functions in PEXlib5.1, the implementation is trivial. What I needed to do is using a OCC interface and calling the corresponding non-OCC function with the corresponding values in OCC.

```

typedef struct {
    Display          *display;
    PEXRenderer      renderer;
    PEXStructure     structure;
    PEXOCRequestType req_type;
    int              shape_hint;
    int              ignore_edges;
    int              contour_hint;
    int              contours_all_one;
    unsigned int     facet_attributes;
    unsigned int     line_vertex_attributes;
    unsigned int     marker_vertex_attributes;
    unsigned int     surface_vertex_attributes;
    unsigned int     edge_attributes;
    unsigned int     facet_fp_data_count;
    unsigned int     line_vertex_fp_data_count;
    unsigned int     marker_vertex_fp_data_count;
    unsigned int     surface_vertex_fp_data_count;
    int              color_type;
    char             *encoding_state;
    int              data_model;
    union {
        PEXOCCStrideData stride;
        PEXOCCUnpackedDataunpacked;
    } data_model_specs;
} PEXOCCValues;

```

/* DEFAULT Values */
 /* undefined */
 /* undefined */
 /* undefined */
 /* PEXOCRender */
 /* PEXShapeUnknown */
 /* False */
 /* PEXContourUnknown */
 /* False */
 /* PEXGANone */
 /* PEXGANone */
 /* PEXGANone */
 /* PEXGANone */
 /* PEXGANone */
 /* 0 */
 /* 0 */
 /* 0 */
 /* 0 */
 /* PEXColorTypeRGB */
 /* NULL Pointer */
 /* PEXDataPacked */

Table 6: OCC Mask Bits

The bits for the mask	PEXOCValues member
PEXOCMDisplay	display
PEXOCMRenderer	renderer
PEXOCMStructure	structure
PEXOCMReqType	req_type
PEXOCMShapeHint	shape_hint
PEXOCMIgnoreEdges	ignore_edges
PEXOCMContourHint	contour_hint
PEXOCMContoursAllOne	contours_all_one
PEXOCMFacetAttributes	facet_attributes
PEXOCMLineVertexAttributes	line_vertex_attributes
PEXOCMMarkerVertexAttributes	marker_vertex_attributes
PEXOCMSurfaceVertexAttributes	surface_vertex_attributes
PEXOCMEdgeAttributes	edge_attributes
PEXOCMFacetFPDataCount	facet_fp_data_count
PEXOCMLineVertexFPDataCount	line_vertex_fp_data_count
PEXOCMMarkerVertexFPDataCount	marker_vertex_fp_data_count
PEXOCMSurfaceVertexFPDataCount	surface_vertex_fp_data_count
PEXOCMColorType	color_type
PEXOCMEncodingState	encoding_state
PEXOCMDataModel	data_model
PEXOCMDataModelSpecs	data_model_specs
PEXOCMAll5.2	all members defined for PEXlib 5.2

PEX5.2 defines eight new primitives (see Table 3). The following describes how I implemented all these new output primitive functions in PEXlib5.2.

◦ PEXOCCArcs Function

```
void PEXOCCArcs(
    PEXOCC context,
    PEXArcType arc_type,
    unsigned int count,
    PEXPointer arcs
)
```

PEXOCCArcs output primitive function allows user to render a set of arcs. PEXlib5.2 defines eight types of arcs: PEXArcCircle, PEXArcCircularArc, PEXArcCircularClosedChordArc, PEXArcCircularClosedPieArc, PEXArcEllipse, PEXArcEllipticalArc, PEXArcEllipticalClosedChordArc and PEXArcEllipticalClosedPieArc. But we can only define one type of arc per function call.

I implemented the PEXOCCArcs function with B-spline curve function PEXNURBCurve. The control points are chosen so that the neighbouring points are close enough to calculate a smooth arc. For different radii, the density of the control points always keeps the same. When the radius is very big, we still get a smooth arc. Figure 7 shows the different arcs drawn with the PEXOCCArcs function.

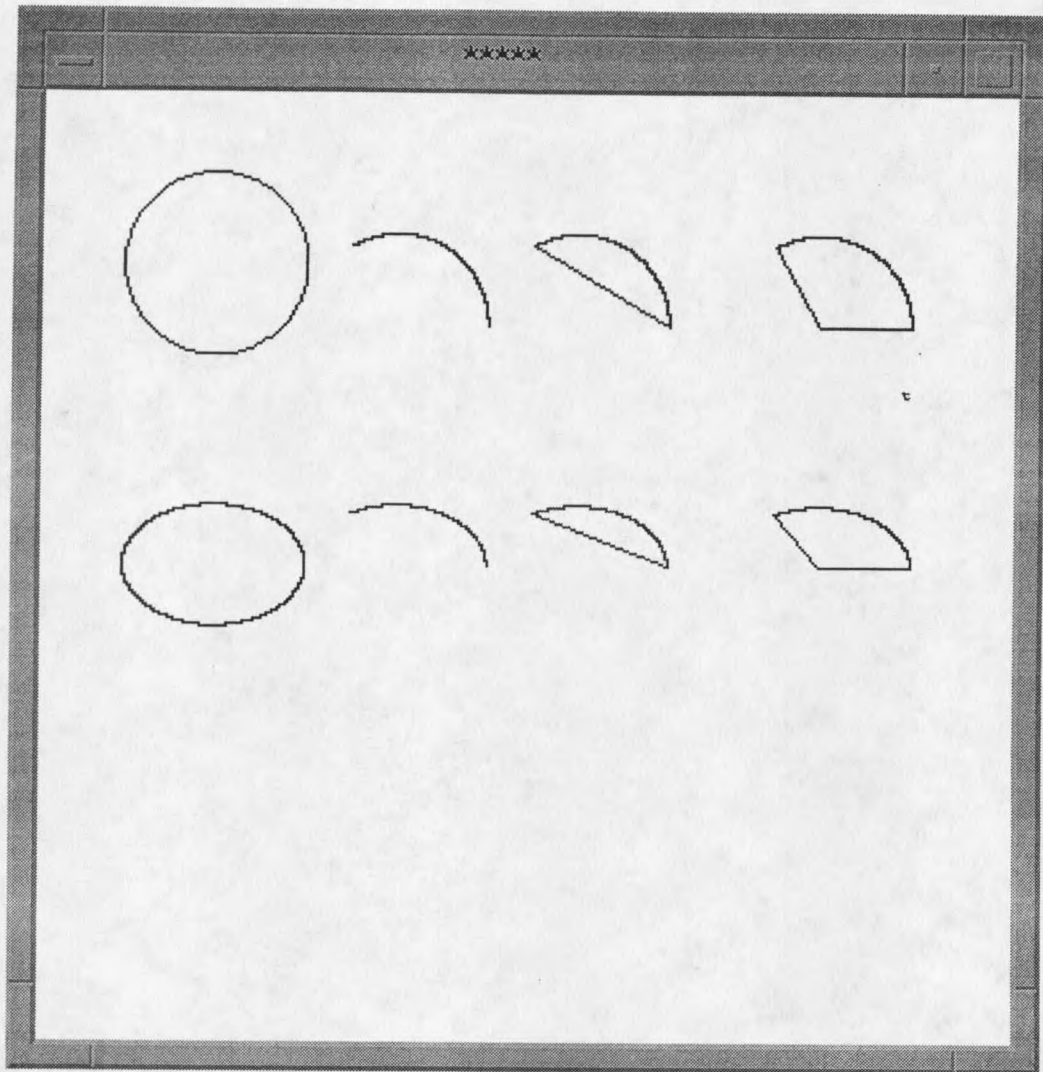


Figure 7: Different Arcs Drawn With PEXOCCArcs Function

◦ PEXOCCArcAreas Function

```
void PEXOCCArcAreas(
    PEXOCC context,
    PEXArcAreaType arc_type,
    unsigned int count,
    PEXPointer arcs
)
```

The PEXOCCArcAreas output primitive function allows user to render a set of areas defined by the arcs. PEXlib5.2 defines six different arc areas: PEXArcAreaCircle, PEXArcAreaCircularClosedChordArc, PEXArcAreaCircularClosedPieArc, PEXArcAreaEllipse, PEXArcAreaEllipticalClosedChordArc, or PEXArcAreaEllipticalClosedPieArc. We can only define one type of arc area per function call.

I implemented the PEXOCCArcAreas function with the PEXFillArea function. In order to get smooth edges for the areas, the vertices for calculating the areas are chosen so that two neighbouring vertices are close enough and with different radii, the density of vertices stays the same.

Figure 8 shows the different arc areas drawn with PEXOCCArcAreas function.

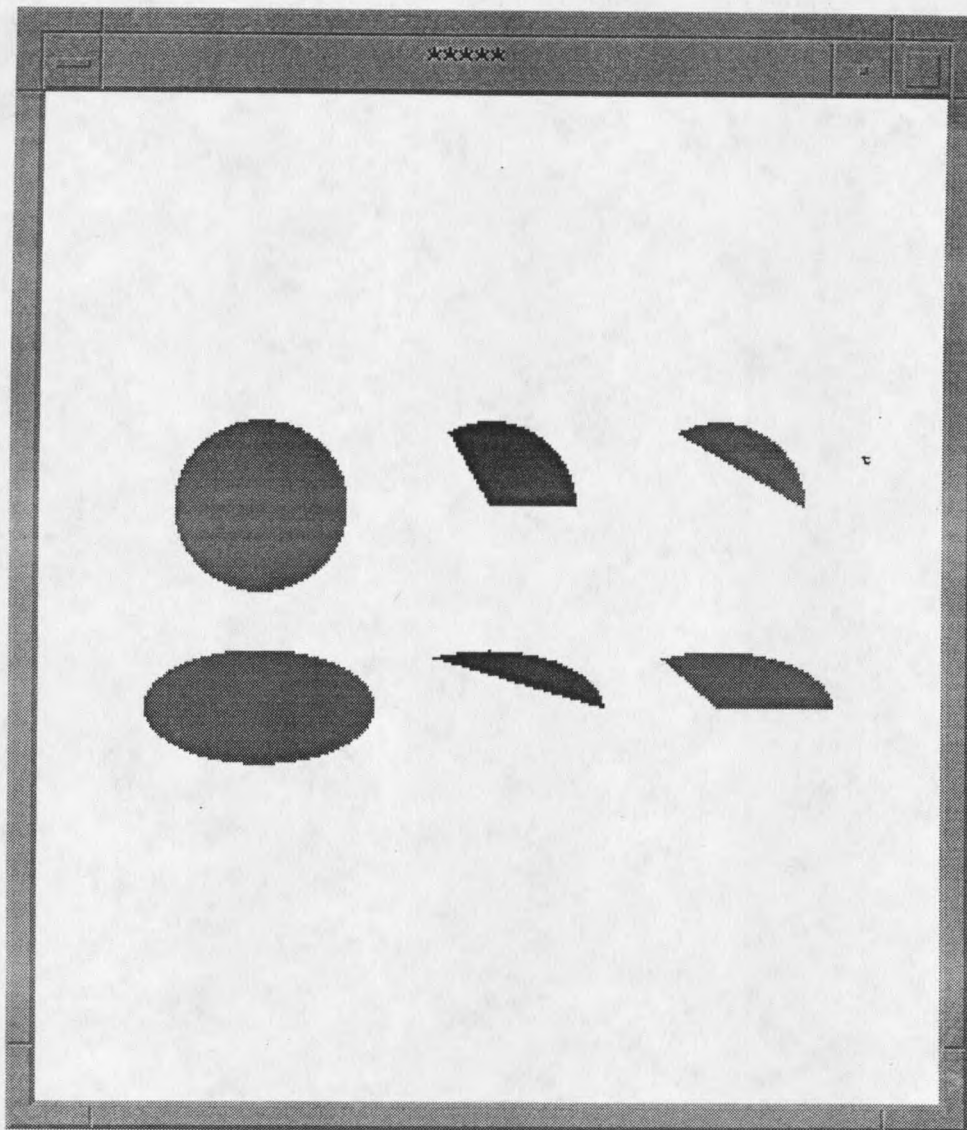


Figure 8: Different Arcs Areas Drawn With PEXOCCArcAreas Function

◦ PEXOCCCones Function

```
void PEXOCCCones(
    PEXOCC context,
    double lower_radius,
    double upper_radius,
    int lower_cap,
    int upper_cap,
    unsigned int num_cones,
    PEXPointer cones
)
```

PEXOCCCones function allows user to render a set of cones. A cone is defined as a three-dimension surface with upper and lower edges defined by two circles. We specify each cone with two vertices and two radii, we can supply a cap to the upper or lower circle by setting the corresponding cap flag.

In my implementation, I render a cone by creating a rational B-Spline surface with PEXNURBSurface function. Figure 9 shows how to select the control points. Figure 10 is a cone rendered with PEXOCCCones function.

In my implementation, a cone is always rendered in MC first. Its lower vertex is the origin of MC, and its upper vertex at +Y axis. Then the cone is transformed to the specified position in WC.

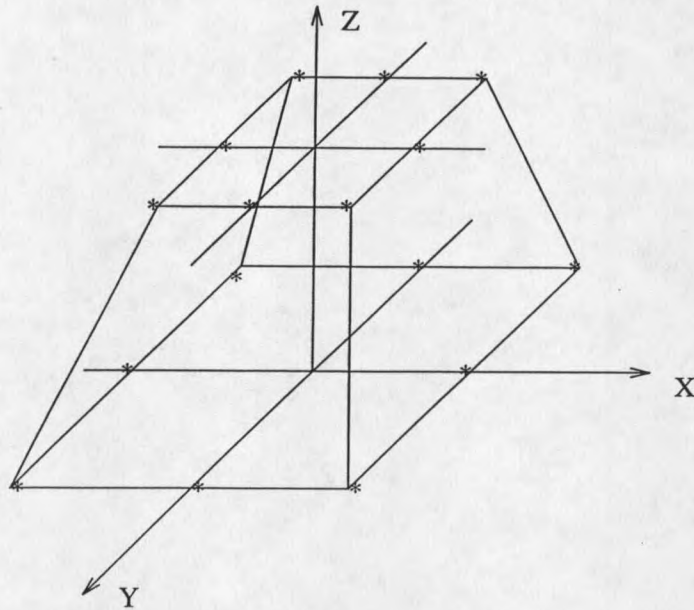


Figure 9: The Selected Control Points For A Cone

- **PEXOCCSpheres Function**

```
void PEXOCCSpheres(
    PEXOCC context,
    double radius,
    unsigned int count,
    PEXPointer spheres
)
```

PEXOCCSpheres allows user to render a set of spheres. Each sphere is specified with a center and a radius.

In fact, we can consider a sphere surface as the result of a half circle in X-Z plane sweeping along another circle in X-Y plane. In my implementation, a sphere is created as a rational B-Spline surface with PEXNURBSurface function.

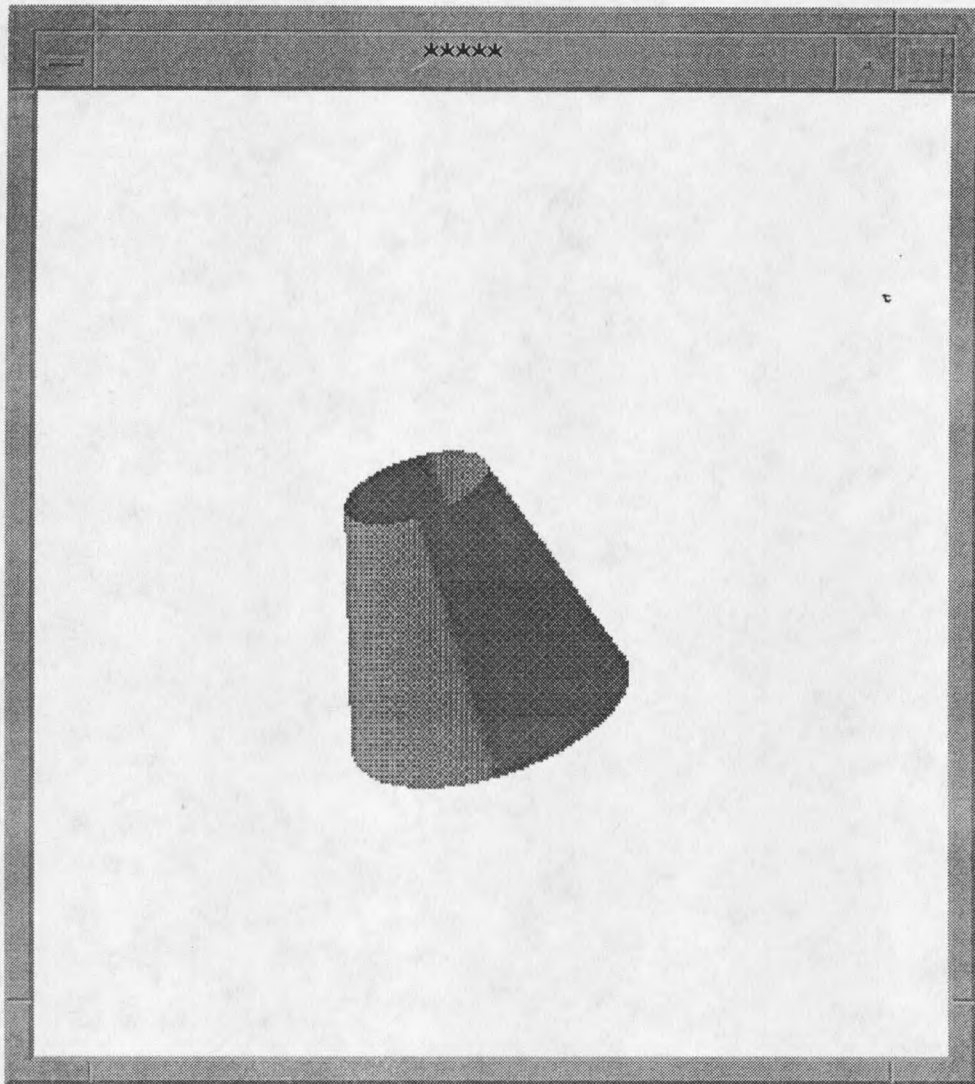


Figure 10: A Cone Rendered With PEXOCCones Function

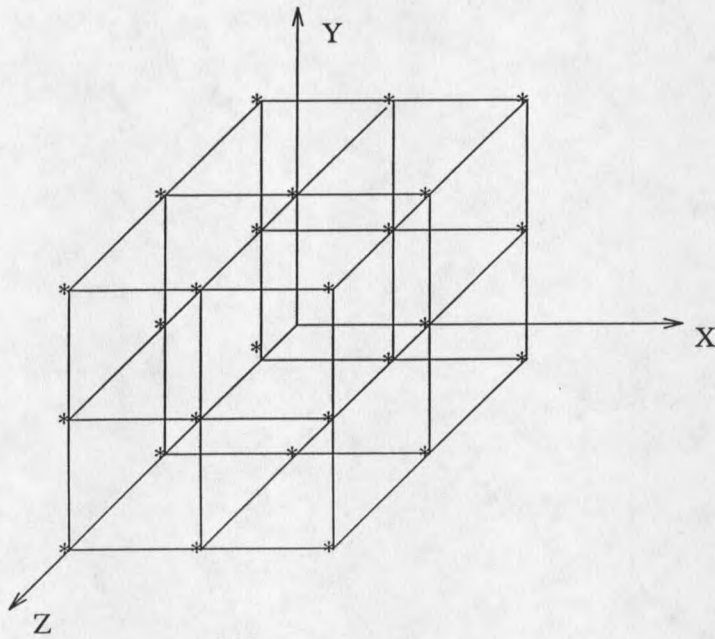


Figure 11: The Selected Control Points For A Sphere

Figure 11 shows how to select the control points. Figure 12 is a sphere rendered with PEXOCCSpheres function.

Each sphere is rendered in MC first with its center being the origin of MC. Then it is transformed to the specified position in WC.

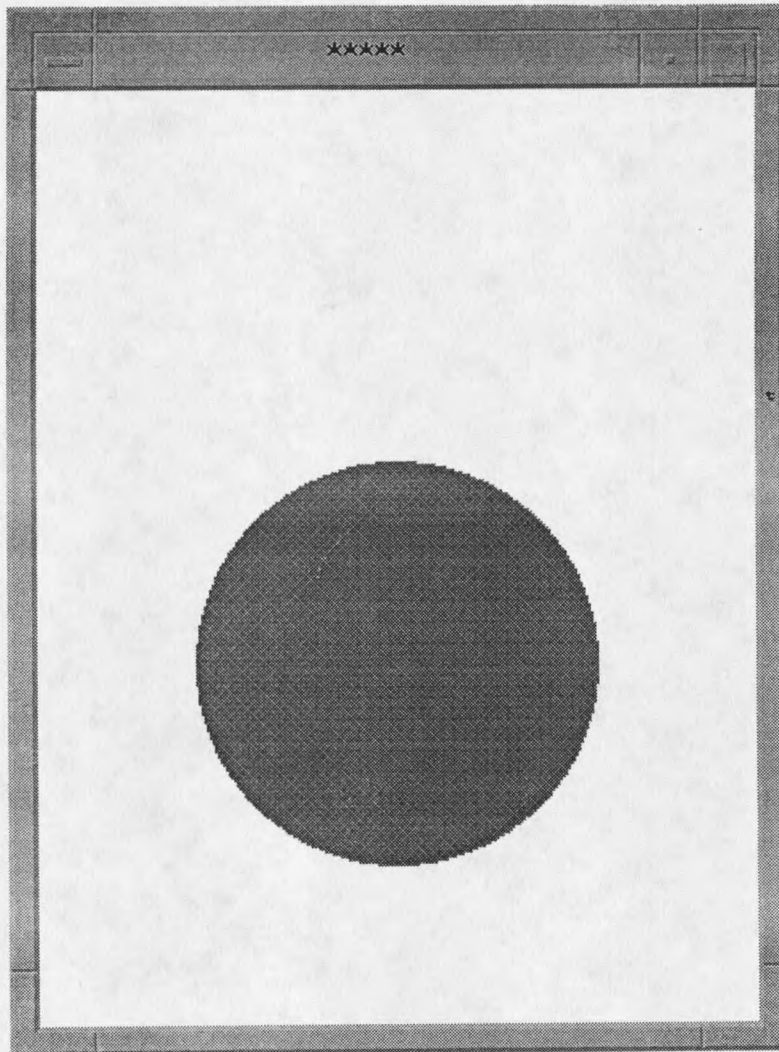


Figure 12: A Sphere Rendered With PEXOCSpheres Function

• Functions Rendering A Set of Triangles

PEXlib5.2 defines four output primitive functions that render a set of triangles according to the arrangement of their vertices:

```
void PEXOCCIndexedTriangles(
    PEXOCC context,
    unsigned int all_independent,
    unsigned int num_facets,
    PEXPointer facet_data,
    unsigned int num_verts,
    PEXPointer vertices,
    unsigned int index_count,
    PEXSwitch *per_edge_data,
    unsigned short *connectivity
)
```

```
void PEXOCCTriangleFan(
    PEXOCC context,
    PEXPointer facet_data,
    unsigned int count,
    PEXPointer vertices
)
```

```
void PEXOCCTriangles(
    PEXOCC context,
    unsigned int num_facets,
    PEXPointer facet_data,
    PEXPointer vertices
)
```

```

void PEXOCCPolyTriangle(
    PEXOCC context,
    unsigned int num_facets,
    PEXPointer facet_data,
    unsigned int num_verts, PEXPointer vertices
)

```

I implemented all these functions with PEXSetofFillAreaSet. Each triangle is considered as a facet, there is only one contour in the facet which is composed of three vertices(represented by three indices).

Texture Mapping Utility Functions

In PEXlib5.2, the eight OCC style utility functions for Texture mapping:

- PEXOCCTMCoordFillAreaSet
- PEXOCCTMCoordIndexedFillAreaSets
- PEXOCCTMCoordIndexedTriangles
- PEXOCCTMCoordPolyTriangle
- PEXOCCTMCoordQuadrilateralMesh
- PEXOCCTMCoordTriangleFan
- PEXOCCTMCoordTriangles
- PEXOCCTMCoordTriangleStrip

All these functions compute a projection of texture coordinates onto a fill area or triangle and store them in the floating-point data areas of each vertex. Three different projections are defined: sphere, cylinder and linear.

The projection attributes are stored in a PEXTMCoordData structure and passed onto the functions.

To compute the projection, a unit direction vector(s_0, s_1, s_2) is determined. The following are the formulas for computing the projections:

for sphere projection

$$t_0 = (s_0 == 0.0) ? ((s_2 > 0) ? 0.75 : 0.25) :$$

$$(-\arctan(s_2/s_0)/(2 * PI) +$$

$$((s_0 < 0) ? 0.5 : (s_0 > 0 \&\& s_2 > 0) ? 1 : 0))$$

$$t_1 = \arcsin(s_1)/PI + 0.5$$

$$-PI/2 \leq \arctan(r) \leq PI/2$$

$$-PI/2 \leq \arcsin(r) \leq PI/2$$

$$\text{Where}(s_0 * s_0 + s_1 * s_1 + s_2 * s_2) = 1$$

for cylinder projection

$$t_0 = (s_0 == 0.0) ? ((s_2 > 0) ? 0.75 : 0.25) :$$

$$(-\arctan(s_2/s_0)/(2 * PI) +$$

$$((s_0 < 0) ? 0.5 : (s_0 > 0 \&\& s_2 > 0) ? 1 : 0))$$

if (coord_source == normal)

$$t_1 = \arcsin(s_1)/PI + 0.5$$

else

$$t_1 = y_coord$$

$$-PI/2 \leq \arctan(r) \leq PI/2$$

$$-PI/2 \leq \arcsin(r) \leq PI/2$$

$$\text{Where}(s_0 * s_0 + s_1 * s_1 + s_2 * s_2) = 1$$

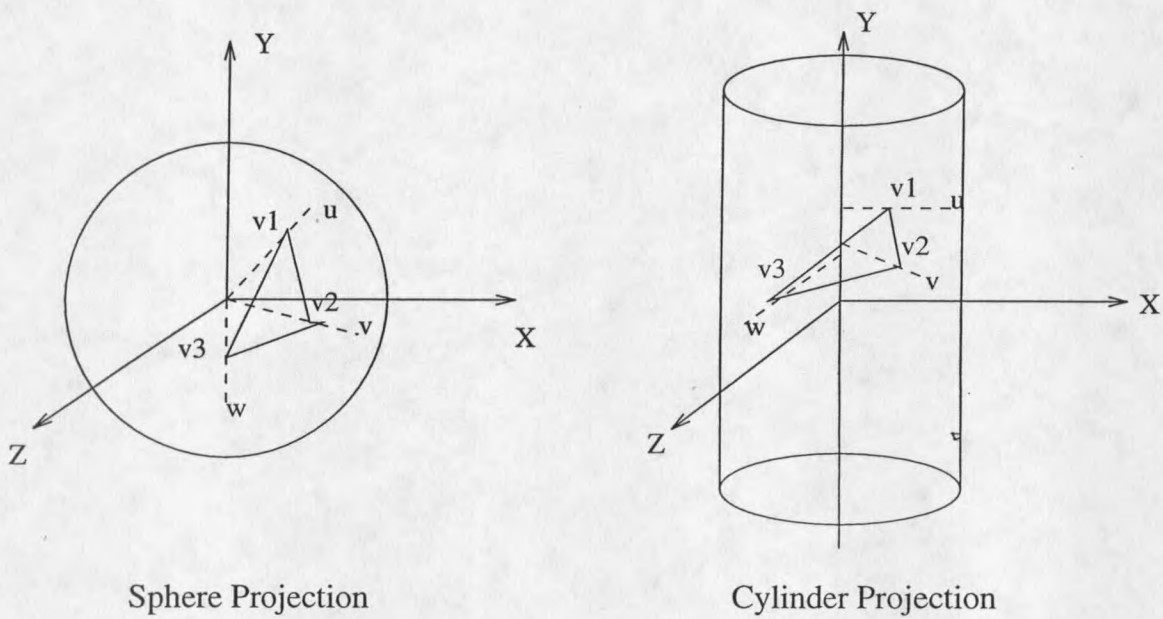


Figure 13: Compute The Texture Map Coordinates

for linear projection

$$t_0 = s_0 * p_0[0] + s_1 * p_0[1] + s_2 * p_0[2] + 1 * p_0[3]$$

$$t_1 = s_0 * p_1[0] + s_1 * p_1[1] + s_2 * p_1[2] + 1 * p_1[3]$$

Figure 13 shows how to compute a sphere projection and cylinder projection.

CONCLUSION AND FUTURE DEVELOPMENT

As my thesis project, I attempted to implement PEXlib5.2 library functions based on PEXlib5.1. In my implementation, I defined the data structures as the same as defined in the PEXlib5.2 specification. All the new output primitive functions were implemented. Although most of the primitive, attributes and utility functions defined in PEXlib5.2 specification can be implemented as emulation functions with PEXlib5.1 functions, some of them can not be implemented on client side, such as the functions about alpha blending and texture mapping. These attributes are invoked in the server's rendering pipeline. If we need to supply those attributes, we have to modify the PEX5.1 server.

In the future, I will improve the emulation functions according to the feedback of user's application. Also I will try to develop other PEXlib5.2 functions.

References

- [1] X consortium, *PEXlib5.2 Specification*, X Consortium Homepage.
- [2] James D. Foley, Andries Van Dam, Steven K. Feiner, John F. Hughes, Richard L. Philips, *Introduction to Computer Graphics*, Addison-Wesley Publishing Company, 1994.
- [3] Tom Grakins, *PEXlib Programming Manual*, O'Reilly Associates, Inc., 1992.
- [4] Paula Womack, *PEXlib: A Tutorial*, Prentice-Hall, Inc., 1994.
- [5] Steve Tallbott, *PEXlib Reference Manual*, O'Reilly Associates, Inc., 1992.
- [6] Oliver Jones, *Introduction to the X Window System*, Prentice-Hall, Inc., 1989.

MONTANA STATE UNIVERSITY LIBRARIES



3 1762 10224485 0