OpenGL on OS/2 Beta Documentation

IBM Power Personal Systems
Graphics Systems
Austin Texas

CHAPTER 1 OpenGL on OS/2

OpenGL is a highly functional 3D API available on many different platforms including many X Windows systems, and also Micrsoft's Windows NT. It supports real time 3D rendering of points/lines/polygons including support for lighting, texture mapping, anti-aliasing, fogging, motion blur, hidden surface removal, transparency, and double buffering of images. This beta release implements OpenGL in software, so special hardware acceleration is not needed. However, beacause it is a software implementation, patience might be required for complex rendering.

OpenGL on OS/2 is currently in beta testing.

Most of the samples are written to a simple toolkit (AUX or TK) which insulates the user from OS/2 Windowing , User input, and OpenGL PGL calls. AUX and TK demos are best for doing simple 3d programming, allowing the user to concentrate on playing with 3D functionality.

OpenGL on OS/2 is implemented as several DLL's on top of OS/2. It can bypass Presentation Manager when displaying OpenGL images by using a 'direct' OpenGL context. It can also render OpenGL images into a standard Presentation Manager bitmap by using an 'indirect' OpenGL context. Direct contexts allow faster OpenGL rendering, while Indirect contexts allow for integrated OpenGL and OS/2 PM drawing.

A portion of the OpenGL API deals with integrating OpenGL into whatever windowing system it is running on. In Presentation Manager, this portion of the API is called PGL . It provides the following functionality:

- Specification of Visual Configurations
- Context Creation
- Binding an OpenGL Context to a PM Window
- Swapping a window's buffers
- Integration of OpenGL and OS/2 drawing
- Utilizing OS/2 fonts in OpenGL

CHAPTER 2 Compiling Demos

Makefiles

Makefiles are for IBM C Set compiler, the compiler we use to compile OpenGL. OpenGL is a 3D API, makefiles for all different OS/2 compilers are not included in the beta program. Compiler and Linker flags are explained in the provided C Set makefile.

Other Compilers

These libraries have been tried using 3 compilers: IBM C Set, WATCOM 10.0, and gcc. They have not been tested with Borland. You should read the explanations in the Makefiles for the switches we use with the IBM C Set compiler. You should make sure you compile/link your demos with case exact turned on, meaning you want the compiler/linker to call glVertex3fv(), not call GLVERTEX3FV(). Make sure you specify a very big stack, if you look at our .def files (another C Set-ism), we are specifying a 1 meg stack. I know that you won't get something this big without specifying it. The OpenGL entry points pass the parameters on the stack, so your calling sample should do this (as opposed to the faster passing in registers). Read through the C Set compiler/ linker parameters.

IBM Developers Toolkit

Even if you are using another compiler, you will probably need the IBM Developer's Toolkit. For OpenGL, it gives you the needed header files to compile. For example, you need os2.h (and all the other header files that go with it), and in the case of gcc, you will need LINK386.EXE to link your programs together. For OS/2 2.1, you can purchase it seperately, for Warp, it is packaged with the OS/2 Developer's Connection, and I'm not sure if it is available seperately. Call 1-800-6DEVCON to subcribe.

CHAPTER 3 How Do I run demos?

Several changes must be made to your config.sys:

c:\opengl\lib must be added to the LIBPATH statement in your config.sys

DEVICE=c:\dev\smvdd.sys must be added to OS/2 2.1 config.sys.

DEVICE=c:\dev\ssmdd.sys must be added to OS/2 Warp config.sys

You must be running in 8 bit (256 color) or 24 bit mode.

Some demos need flags. Check the source code to see what flags you should pass:

wave -db -dr

-db for double buffered

-dr for direct rendering (-ir for slower indirect rendering)

CHAPTER 4 New for this Beta

Added Support for other compilers, device driver bug fixes.

Please read DEVCON documentation for receiving technical support for this product.

Our current beta release supports the following Visual Configurations on direct and indirect contexts:

Table 1: Supported Visual Configs

Visual Config	Single/ Double Buffer	RGBA/ Color Index	Depth Buffer	Alpha Buffer	Accum Buffer	Stencil Buffer
1	Double	RGBA(3,3,2,8)	24	8	16,16,16,16	8
2	Double	CI(256)	24	0	0	8
3	Double	RGB(3,3,2)	24	0	0	8
4	Double	RGBA (3,3,2,8)	24	8	0	8
5	Double	RGBA(8,8,8,8)	24	8	16,16,16,16	8

CHAPTER 5

PGL Function Specification

5.1 PVISUALCONFIG pglChooseConfig(hab, attriblist)

Description:

PVISUALCONFIG is required when creating an OpenGL context. This call will return a suitable PVISUALCONFIG based on the minimum requirements passed in through attriblist. Each Boolean attribute will be matched exactly, and each integer attribute will meet or exceed the specified value. All Boolean attributes default to False, except PGL_USE_GL, which defaults to True. Default attributes are superseded by the attributes listed in attriblist. Boolean attributes included in attriblist are understood to be true, integer attributes are followed immediately by the corresponding desired or minimum value. Attributes which are not specified The attriblist is terminated by None. A PVISUALCONFIG is needed when creating an OpenGL rendering context.

Parameters:

HAB	hab	Handle to Anchor Block
int	*attriblist	Specifies a list of Boolean attributes and integer attribute/value pairs. The last attribute in the list must be None.

Return Values:

PVISUALCONFIG	Pointer to a VisualConfig structure meeting the requirements of attriblist
NULL	No suitable VisualConfig was found

The interpretations of the various PGL visual attributes are as follows:

PGL_USE_GL Ignored. Only visual configs that can be rendered with PGL are considered.

PGL_BUFFER_SIZE Must be followed by a nonnegative integer that indicates the desired color index buffer size. The smallest index buffer of at least the specified size is preferred. Ignored if PGL_RGBA is asserted.

PGL_LEVEL Must be followed by an integer buffer-level specification. This specification is honored exactly. Buffer level zero corresponds to the default frame buffer of the display. Buffer level one is the first overlay frame buffer, level two the second overlay frame buffer, and so on. Negative buffer levels correspond to underlay frame buffers.

PGL_RGBA If present, only RGBA visual configs are considered.Otherwise, Color Index visual configs are considered.

PGL_DOUBLEBUFFER If present, only double bufferered visual configs are considered. Otherwise both single buffered and double buffered visual configs may be considered.

PGL_SINGLEBUFFER If present, only single buffered visual configs are considered. Otherwise both single buffered and double buffered visual config may be considered.

PGL_STEREO If present, only stereo visual configs are considered. Otherwise, both monoscopic and stereoscopic visual configs are considered.

PGL_AUX_BUFFERS Must be followed by a nonnegative integer that indicates the desired number of auxiliary buffers. Visual configs with the smallest number of auxiliary buffers that meets or exceeds the specified number are preferred.

PGL_RED_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, the smallest available red buffer is preferred. Otherwise, the largest available red buffer of at least the minimum size is preferred.

PGL_GREEN_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, the smallest available green buffer is preferred. Otherwise, the largest available green buffer of at least the minimum size is preferred.

PGL_BLUE_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, the smallest available blue buffer is preferred. Otherwise, the largest available blue buffer of at least the minimum size is preferred.

PGL_ALPHA_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, the smallest available alpha buffer is preferred. Otherwise, the largest available alpha buffer of at least the minimum size is preferred.

PGL_DEPTH_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, visual configs with no depth buffer are preferred. Otherwise, the largest available depth buffer of at least the minimum size is preferred.

PGL_STENCIL_SIZE Must be followed by a nonnegative integer that indicates the desired number of stencil bitplanes. The smallest stencil buffer of at least the specified size is preferred. If the desired value is zero, visual configs with no stencil buffer are preferred.

PGL_ACCUM_RED_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, visual configs with no red accumulation buffer are preferred. Otherwise, the largest possible red accumulation buffer of at least the minimum size is preferred.

PGL_ACCUM_GREEN_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, visual configs with no green accumulation buffer are preferred. Otherwise, the largest possible green accumulation buffer of at least the minimum size is preferred.

PGL_ACCUM_BLUE_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, visual configs with no blue accumulation buffer are preferred. Otherwise, the largest possible blue accumulation buffer of at least the minimum size is preferred.

PGL_ACCUM_ALPHA_SIZE Must be followed by a nonnegative minimum size specification. If this value is zero, visual configs with no alpha accumulation buffer are preferred. Otherwise, the largest possible alpha accumulation buffer of at least the minimum size is preferred.

Attriblist example:

```
attribList ={PGL_RGBA, PGL_RED_SIZE, 4, PGL_GREEN_SIZE, 4,PGL_BLUE_SIZE, 4,
        None};
```

Specifies a single-buffered RGB visual config in the normal frame buffer, not an overlay or underlay buffer. The returned visual config supports at least four bits each of red, green, and blue, and possibly no bits of alpha. It does not support color index mode, double-buffering, or stereo display. It may or may not have one or more auxiliary color buffers, a depth buffer, a stencil buffer, or an accumulation buffer.

The user should not modify the fields of the returned PVISUALCONFIG structure.

When the user is done with the PVISUALCONFIG returned, the memory can be freed by calling the C library routine free().

5.2 PVISUALCONFIG *pglQueryConfigs(hab)

Description:

This call will return a NULL terminated array of pointers to available visual config structures for an OpenGL application to choose from. A visual config defines what buffer configurations (depth,accum,alpha,stencil) are available. An application should choose the simplest visual config that will suit its needs (least number of buffers).Programmers can also use the helper routine pglChooseConfig which will choose a suitable visual based on minimum requirements. A PVISUALCONFIG is required to create an OpenGL context. The list of Visual Configs is not guaranteed to be in any certain order.

Parameters:

HAB	hab	Handle to anchor block

Return Values:

PVISUALCONFIG *	Pointer to null terminated array of pointers to VISUALCONFIG structures containing buffer configs

Notes:

The user should not modify the fields of the returned PVISUALCONFIG structures.

When the user is done with the returned PVISUALCONFIG list , that memory can be freed by calling the C library routine free().

5.3 HGC pglCreateContext(hab,pVisualConfig,ShareList,IsDirect)

Description:

This call will create an OpenGL context using the given visual config . A visual config specifies what framebuffer resources are available to the rendering context. If an OpenGL context was successfully created, a handle to it will be returned, else NULL will be returned.

Parameters:

HAB	hab	Handle to anchor block
PVISUALCONFIG	pVisualConfig	Pointer to VISUALCONFIG structure containing desired buffer config
HGC	ShareList	Handle of OpenGL context with which to share display lists within a process.
BOOL	IsDirect	TRUE: Bypass GPI, and render to a PM window. Generally faster. FALSE: This context will use Gpi for OpenGL rendering, using GpiBitBlt to blit OpenGL rendering to a PM window

Return Values:

HGC	Handle to successfully created context
NULL	Error occurred.

Configurations can be listed through pglQueryConfigs()

Direct context- bypasses PM when displaying OpenGL rendering . Not available on all OS/2 configurations. If IsDirect parameter is TRUE, but a direct context is not available, an indirect context will be created. pglIsIndirect can be called to see if an OpenGL context is direct or indirect.

Indirect context - uses GpiBitBlt to get image to screen . Allows access to a PM bitmap containing OpenGL rendered image. The PM Bitmap is not guaranteed to contain any OpenGL rendering until the OpenGL graphics pipeline has been flushed (glFlush , pglSwapBuffers or pglWaitGL).

If ShareList is not NULL, then all display list definitions are shared by ShareList and the newly created context. An arbitrary number of contexts can share display lists, but each context must be owned by the same process.

5.4 BOOL pglMakeCurrent(hab,hgc,hwnd)

Description:

This call will bind an OpenGL hgc (context) to a PM window . The pglMakeCurrent subroutine will replace the old current context (if there was one) with hgc. Before unbinding the old context, an implicit flush of the old context will take place. Thus subsequent OpenGL rendering commands will use hgc to modify hwnd. The first time hgc is made current to a window, its viewport will be initialized to the full size of hwnd. Subsequent calls to pglMakeCurrent with hgc will not affect its viewport. To unbind the current context without binding a new one, call pglMakeCurrent(hab,NULL,None). pglMakeCurrent returns True if successful, False otherwise.

Parameters:

HAB	hab	Handle anchor block
HGC	hgc	Handle to OpenGL context
HWND	hwnd	PM Window handle

Return Values:

True	Context was successfully bound to the window
False	Error occurred.

Notes:

Only one context can be bound to a window at a time.

The PM window must have been created with window client class CS_SIZEREDRAW and CS_MOVENOTIFY

The application cannot WinSubclassWindow(hwnd) while an OpenGL context is bound to hwnd.

5.5 Bool pglDestroyContext(hab,hgc)

Description:

This function will destroy hgc, and all the resources that belong to it. If it is currently bound to a hwnd, this call will be ignored.

Parameters:

HAB	hab	Handle to Anchor Block
HGC	hgc	Handle to OpenGL context

Return Values:

True	Context was successfully destroyed, or context was bound to a window
False	Error occured.

If hgc is currently bound to a window, pglDestroyContext will return without destroying hgc. A current context can be unbound from the currently bound window by calling pglMakeCurrent(hab,NULL,None);

5.6 Bool pglCopyContext(hab, hgc_src, hgc_dst, attrib_mask)

Description:

This call will copy some portion of state from hgc_src to hgc_dst. The attrib_mask parameter determines what group(s) of state variables will be copied. attrib_mask must contain the bitwise OR of the same symbolic names that can be passed to glPushAttrib. Set attrib_mask to GL_ALL_ATTRIB_BITS to copy the max amount of state. This copy can be done only if hgc_src and hgc_dst were created in the same process.

Parameters:

HAB	hab	Handle to Anchor Block
HGC	hgc_src	Source OpenGL context
HGC	hgc_dst	Destination OpenGL context
GLuint	attrib_mask	Specifies which portions of hgc_src are to be copied to hgc_dst

Return Values:

True	Context state was successfully copied from hgc_src to hgc_dst
False	Error occured.

If hgc_src is not current to the thread issuing the request, then the state of hgc_src is undefined.

Not all values of OpenGL state can be copied. For example, pixel pack and pixel unpack state, render mode state, select and feedback state are not copied. The state that can be copied by this command is exactly the same state that can be manipulated by the OpenGL command glPushAttrib.

5.7 LONG pglIsIndirect(hab,hgc)

Description:

This function will return what type of context hgc is. The hgc parameter must have been returned from pglCreateContext.

Parameters:

HAB	hab	Handle to Anchor Block
HGC	hgc	Handle to OpenGL context

Return Values:

NULL	This Context bypasses conventional PM blitting methods, and is faster .
<0	Error occured.
1	This Context uses conventional PM blitting methods, and while it is slower, allows an application to integrate OpenGL rendering and Gpi rendeing commands. See notes section.

Notes:

If pglIsIndirect returns One, it uses Gpi to blit OpenGL rendering . Integration of OpenGL rendering and Gpi renderingis only allowed on Indirect contexts, and is controlled through the use of pglWaitPM and pglWaitGL. Applications must call pglGrabFrontBitmap when they need access to the actual bitmap, and call pglReleaseFrontBitmap when they are done with it.

The PM Bitmap is not guaranteed to contain any OpenGL rendering until the OpenGL graphics pipeline has been flushed (glFlush, pglSwapBuffers, or pglWaitGL).

5.8 HGC pglGetCurrentContext(hab)

Description:

Look up the current OpenGL context for this process. A context is ‘current’ if it was the last context to be bound to a window by calling pglMakeCurrent.

Parameters:

HAB	hab	Handle to Anchor Block

Return Values:

HGC	Current Context
NULL	No Current context exists for this process

Notes:

Only One current context is allowed per process.

5.9 HWND pglGetCurrentWindow(hab)

Description:

This function returns the current hwnd that is bound to an OpenGL context. A window is current if it was the last window to be bound to an OpenGL context by calling pglMakeCurrent.

Parameters:

HAB	hab	Handle to Anchor Block

Return Values:

HWND	Current window that is bound to an OpenGL context
NULL	No window is currently bound to any OpenGL context for the calling process

Notes:

Only one current window is allowed per process.

5.10 HPS pglWaitGL(hab)

Description:

Ensure that all OpenGL rendering commands made prior to pglWaitGL are executed before and Gpi rendering calls made after pglWaitGL. This call is ignored if there is no current context or if the current context is a direct context which does not use Gpi for displaying OpenGL images. The return value for this function is HPS that has the bitmap set in it to be the bitmap containing OpenGL rendering.

Parameters:

HAB	hab	Handle to Anchor Block

Return Values:

HPS	hps	Handle to the PS which is used to blit the OpenGL image to the screen. This HPS can be used for Gpi drawing commands .
NULL		No Current Context exists, or current context is not using Gpi to display OpenGL images.

Notes:

Gpi drawing intermingled with OpenGL drawing is only allowed on Indirect contexts that use Gpi to blit their OpenGL rendering to the screen. Prior to doing any Gpi drawing, the user must call pglWaitGL in order to flush the OpenGL rendering stream.

pglWaitGL is ignored if there is no current context.

5.11 void pglWaitPM(hab)

Description:

Gpi calls made prior to pglWaitPM are guaranteed to be executed before OpenGL rendering calls made after pglWaitPM.

Parameters:

HAB	hab	Handle to Anchor Block

Return Values:

Notes:

Gpi drawing intermingled with OpenGL drawing is only allowed on Indirect contexts that use Gpi to blit their OpenGL rendering to the screen. Gpi drawing is not guaranteed to be visible until pglWaitPM has been called.

Gpi has no concept of front and back buffers. All Gpi drawing commands are therefore assumed to affect the OpenGL front buffer.

pglWaitPM is ignored if there is no current context

5.12 void pglSwapBuffers(hab,hwnd)

Description:

Swaps the front and back buffers of hwnd. This routine has no effect on windows attached to single buffered contexts. An implicit glFlush is done by pglSwapbuffers before it returns. Double buffered rendering is done when smooth animation between frames is desired. OpenGL commands issued after calling pglSwapBuffers are not issued until the buffer swap is complete.

Parameters:

HAB	hab	Handle to Anchor Block
HWND	hwnd	Handle to Window whose buffers are to be swapped.

Return Values:

Notes:

When a window's buffers are swapped, the back buffer becomes the front, and the front buffer will become the back. The programmer can control whch buffer is affected by OpenGL rendering calls through the use of glDrawBuffer.

Front and back buffers are not created for a window until it has been bound to an OpenGL context. This call has no effect on a PM window which has never been bound to an OpenGL context.

The window specified by hwnd does not CURRENTLY have to be bound to an OpenGL context, just needs to have been bound at some point.

5.13 BOOL pglUsePMBitmapFont(hab, id,first,count,listbase)

Description:

This function is not implemented in the current beta release.

This function will create count OpenGL display lists containing bitmaps of the named OS/2 logical font specified by id. Each bitmap will consist of a single glBitmap command. These display lists will be numbered listbase through listbase + count -1. The parameters to glBitmap for display list listbase+i are derived from bitmap first+i in the logical font. OpenGL might delay glBitmap creation until a font glyph is accessed.

Parameters:

HAB	hab	Handle to anchor block
LONG	id	Font id returned from GpiCreateLogFont
int	first	index of first glyph to be taken
int	count	number of glyphs to be taken
int	listbase	index of first display list to be created.

Return Values:

NULL	Error occured.
Non NULL	Bitmap display lists were successfully created.

Notes:

Empty display lists are created for all glyphs requested but not defined in the logical font specified by id.

Outline fonts are not valid fonts for this call.

pglUsePMBitmapFont is ignored if there is no current OpenGL context.

5.14 BOOL pglUsePMOutlineFont(hab, id,first,count,listbase)

Description:

This function is not implemented in the current release.

This function will create count OpenGL display lists containing verticies of the named OS/2 logical font specified by id. Each Display List will consist of a single glVertex2i command. These display lists will be numbered listbase through listbase + count -1.

Parameters:

HAB	hab	Handle to anchor block
LONG	id	Font id returned from GpiCreateLogFont
int	first	index of first character to be taken
int	count	number of characters to be taken
int	listbase	index of first display list to be created.

Return Values:

NULL	Error occured.
Non NULL	display lists were successfully created.

Notes:

Empty display lists are created for all characters requested but not defined in the logical font specified by id.

Bitmap fonts are not valid fonts for this call.

pglUsePMOutlineFont is ignored if there is no current OpenGL context.

5.15 LONG pglQueryCapability(hab)

Description:

Query OpenGL capability on this machine.

Parameters:

HAB	hab	Handle to Anchor Block

Return Values:

NULL	No OpenGL Support on this machine
1	OpenGL Support available through PM blitting only
2	Advanced OpenGL support available

Notes:

5.16 void pglQueryVersion(hab,major, minor)

Description:

Queries release of OpenGL libraries installed.

Parameters:

HAB	hab	Handle to Anchor Block
int	*major	Returns major release number of OpenGL libraries
int	*minor	Returns minor release number of OpenGL libraries

Return Values:

Notes:

5.17 BOOL pglGrabFrontBitmap(hab,phps, phbitmap)

Description:

This subroutine queries the HPS and HBITMAP which contain the bitmap representation of the front buffer of the current OpenGL window. While the bitmap is locked, the application will NOT receive WM_SIZE or WM_ADJUSTPOSITION messages in the current OpenGL hwnd, and the current window will NOT be sizeable. An implicit glFlush() occurs before this call completes. Applications must call pglReleaseFrontBitmapwhen they are done with the HBITMAP. The HBITMAP is only valid between pglGrabFrontBitmap and pglReleaseFrontBitmap, the HPS will be valid until the current window is unbound by calling pglMakeCurrent.

Parameters:

HAB	hab	Handle to Anchor Block
HPS	*phps	Pointer returning HPS with phbitmap set in it
HBITMAP	*phbitmap	Pointer returning HBITMAP containing OpenGL rendering

Return Values:

BOOL	True: Bitmap was grabbed
	False: No Current window exists for this process

Notes:

Applications should not destroy the HPS or HBITMAP. Applications should not use GpiSetPalette to modify the HPS's color palette. Applications should not disassociate HPS from it's DC, or unset the bitmap from HPS.Applications should not set any PS size, units and format using GpiSetPS. In other words, "Look, but don't touch!"

See pglWaitPM and pglWaitGL for integrating OpenGL andGpi drawing.

Applications should call pglReleaseFrontBitmap as soon as possible after locking the bitmap with this call.

5.18 BOOL pglReleaseFrontBitmap(hab)

Description:

This subroutine releases the HBITMAP which was grabbed using pglGrabFrontBitmap. The calling process will now begin recieving WM_SIZE and WM_ADJUSTPOSITION messages in its current OpenGL window again.

Parameters:

HAB	hab	Handle to Anchor Block

Return Values:

BOOL	True: Bitmap was released
	False: No bitmap needed to be released.

Notes:

See pglWaitPM and pglWaitGL for integrating OpenGL and PM drawing.

Applications should call pglReleaseFrontBitmap as soon as possible after locking the bitmap with pglGrabFrontBitmap.