Initializing an Application Using the CGE PEX 5.1 Extensions 3

Getting Started

In many ways, your application begins just like any other PEXlib application.

Initializing PEXlib

You initialize PEXlib the same way you initialize any PEXlib application.

Detecting a CGE PEX 5.1 Implementation

One of the first things you will want to do in your application is to find out if the PEX server has the CGE extensions. Then you can decide either to abort the application or configure the application to run without the extensions. You should query the server for the list of Escapes that the PEX extension supports using the PEXGetEnumTypeInfo request. If the server has the CGE extensions, then you will find the escape identifiers, starting at 0x9000 (PEXETEscapeChangePipelineContext), in the list. The following is a code sequence that performs the query:

```
{
    int status, i, found;
    int enum_types[1];
    unsigned long *enum_counts;
    PEXEnumTypeDesc *enum_values, *p_enum;

    enum_types[0] = PEXETEscape;

    status = PEXGetEnumTypeInfo(dpy,
                                drawable,
                                1,
                                enum_types,
                                PEXETCounts | PEXETIndex,
                                &enum_counts,
                                &enum_values);
    if (!status) {
        /* failure */
    }

    p_enum = enum_values;
    found = False;
    for (i=0; i<enum_counts[0]; i++) {
        if (((unsigned short) (p_enum++)->index) ==
        ((unsigned short) PEXETEscapeChangePipelineContext))
        found = True;
    }
}
```

Checking for Optional Features

One of the goals of CGE PEX 5.1 is to reduce the occurrence of *optional features* that application developers could not always count on being present in the PEX server implementation. While there is much more mandated functionality, there are still allowable differences in the degree that some of the features might be implemented across CGE PEX server implementations. For example, texture mapping is a required function, but only 2D texture maps are required to be implemented by all CGE PEX servers.

Once you have established that your application is connected to a CGE PEX server, you can then either count on the required functionality being there without further query, or you can issue additional PEXGetEnumTypeInfo requests to determine what additional functionality might be available. Continuing with the texture mapping case, you can now go ahead and use 2D

textures without issuing further queries. But if you also wanted to use 1D textures, you would have to query the PEXExtTMDomain enumerated type to make sure that the server supported 1D textures.

Many people design applications that are flexible in a few ways, depending on the level of this sort of optional support. The application usually queries the features for which it can adjust once during initialization and keeps the results around for easy access. It is usually a good idea to perform these queries only once, because round-trips to the server can be expensive and the data does not change during the session anyway.

Multi-Buffering

In accordance with the preferred industry-standard way of implementing double-buffering, CGE PEX 5.1 requires that platforms supporting CGE PEX 5.1 must also provide the X Multi-Buffering Extension (MBX). This means if you are using MBX 3.2 or later, then your applications are assured of portability on any CGE PEX 5.1 platform. This eliminates the need for you to invoke XListExtension or XQueryExtension functions. Use MBX in your applications any time double-buffering is desired. To use double-buffering via MBX, invoke:

```
XmbufCreateBuffers()
```

to create the image buffers. If the device only supports one MBX image buffer, then you could either *live* with single buffering, or find out if the server supports rendering to pixmaps. Issue PEXMatchRenderingTargets to do so. If rendering to pixmaps is supported, copy the pixmap to the window to simulate double-buffering.

For every window or pixmap drawable that is supported by a PEX implementation, a corresponding MBX buffer drawable, whether single-buffered or double-buffered, will be supported on any server supporting CGE PEX 5.1.

Fonts

All vendors who support CGE PEX 5.1 use the X-Windows Logical Font Description (XLFD) font naming format. Loading and using a font with a `CHARACTER_REGISTRY` and `CHARACTER_ENCODING` property of ISO8859-1

(*.*.*.*.*.*.*.*.*.*.*.*.ISO8859-1)

ensures you of at least one PEX-usable font on any CGE PEX 5.1 platform.

PEX Subsets

Do not use the workstation subset in your CGE PEX application. It is not required for a CGE PEX implementation, so you are not guaranteed that it will always be available. Once you are certain that the server supports CGE PEX 5.1, you also do not need to check for the renderer or structure subsets. They are required in CGE PEX 5.1.

Opening a Window, Selecting a Visual, Creating a Colormap, and Establishing Color Approximation

The following are the four basic steps to PEX Color support:

- Select a Visual in which to create the Window.
- Create a Colormap, or find one to share with other similar clients in that Visual.
- Load Colors into the Colormap.
- Create a Window in the selected Visual with the correct Colormap.

To help you complete these four basic steps, CGE PEX 5.1 offers utility extensions. These utilities are shipped as source code to give your application control over policy decisions regarding choice of visual and colormap organization. However, applications that support CGE PEX 5.1, should be able to use these functions without changes on any CGE PEX 5.1 supported platform.

As with other functions, call the PEXInitialize function prior to using these utilities.

Though there are a number of Visual and Colormap utilities, the main ones that will help you with the four steps listed above include:

- PEXUtSimpleWindowAndColormap which operates with “soft” criteria (discussed in “Visual Selection” on page 19) and creates a Window for PEX rendering.
- PEXUtMakeWindowAndColormap which is similar to the PEXUtSimpleWindowAndColormap function but offers you more explicit control over the criteria (see “Visual Selection” on page 19) and attributes used in selecting the Visual and creating the Window.
- PEXUtSelectVisual which is a lower-level utility that offers you control over the criteria used to select a Visual (see “Visual Selection” on page 19). You can then use other CGE-supplied utilities, or code of your own, to handle the Colormap and Color Approximation issues and the creation of a Drawable. Some of the CGE PEX utilities include: PEXUtMakeColormap, PEXUtGetStandardColormapInfo, and PEXUtCheckColorApproximation. See the CGE reference material for more details.

Visual Selection

If you have a simple application that does not have complicated requirements for the Visual, then you probably prefer to use the PEXUtSimpleWindowAndColormap utility. Otherwise, use the Visual selection criteria described in this section to provide input to the PEXUtMakeWindowAndColormap or PEXUtSelectVisual utility extensions.

You can supply one or more criterion for selection of a Visual. Criteria can be “hard” which indicates that if the results are not met by the Visual, then unacceptable results are produced. The utility returns a failure code or continues to the next criteria set if you have supplied more criteria. Criteria can also be “soft” which indicates that the characteristics are desirable but not required for a Visual to be acceptable.

There are two means of Visual Selection:

- Supply a single criteria set with some combination of “hard” and “soft” criteria. If the utility does not find a Visual that meets all of your “hard” criteria, then the utility returns an error code and your application can either quit or try another set of criteria. Failure to meet “soft” criteria is not considered a complete failure, though the utility informs you which ones were not met.

- Supply several sets of criteria (a “degradation” path) for Visual selection. Failure to meet the “hard” criteria in one set causes the utility to continue to the next set. If the utility reaches the last set that you supplied and cannot meet the requirements, then the utility returns an error code. Failure to meet “soft” criteria does not cause the utility to continue to the next set. Therefore, you probably want your degradation path to list “hard” criteria in the early sets and relax the requirements (for example, by listing “soft” criteria) in the later sets. When the utility finds a Visual that meets a set of criteria, the utility returns information on which set was successful.

Colormap Creation and Color Approximation

The Colormap and Color Approximation CGE utility extensions provide an interface to applications to ease the Colormap tasks. If you want to use these utilities in your application, then include the PEXUtCmap.h header file.

For PEX color support interoperability, use standard Colormaps. In general, with PEX 5.1, it is recommended that you use RGB_BEST_MAP and/or RGB_DEFAULT_MAP. You can use the PEXUtGetStandardColormapInfo utility to obtain a Colormap initialization and Color Approximation setup that PEX is likely to support in the specified Visual. If no interoperability property was used to obtain the Colormap description, the utility provides an *educated guess* about what color setup might be supported by PEX. The interoperable conventions include:

- As mentioned above, use standard Colormaps. In PEX it is recommended that you use RGB_BEST_MAP and/or RGB_DEFAULT_MAP. CGE PEX 5.1 servers are likely to define these properties and describe Colormaps that are supported by PEX via the PEXColorSpace approximation. With CGE PEX 5.1 utilities, this interoperable property is easy to meet by including the PEXUtStandardColormapProperty criterion as either *hard* or *soft* in each criteria set (see “Visual Selection” on page 19 for details on *hard* and *soft* criteria).
- Use the PEX 5.1 PEXEscapeQueryColorApprox escape or the CGE PEX 5.1 PEXExtQueryColorApprox escape to verify the setups available. An alternative setup is returned if the specified Colormap setup is not supported for the PEX ColorApproximation and your application then can choose either to adapt to the alternative configuration or to select another

Visual. The CGE PEX 5.1 utilities, PEXUtMakeWindowAndColormap and PEXUtSimpleWindowAndColormap, enable you to use the supplied alternative configuration.

To use the PEXUtMakeWindowAndColormap utility, for example, set one or two *hard* criteria to indicate the most essential characteristics required of a Visual or Window and any *soft* criteria. Ensure one of the *hard* or *soft* criteria is PEXUtStandardColormapProperty so that such properties are accessed. If two Visuals meet the *hard* criteria and an equal amount of *soft* criteria, the Visual with the most color resolution is selected. If neither Visual has more color resolution than the other, then the first Visual in the list provided by the server is selected.

If you create a PEX Color Approximation entry (for example, by calling the PEXUtSelectVisual or PEXUtGetStandardColormapInfo utility), you can verify that the PEX server supports the Color Approximation entry you want by issuing the CGE PEX 5.1 utility PEXUtCheckColorApproximation. Based on information from the server, the utility verifies that the Color Approximation setup is supported by PEX on the specified Visual. If the specified Color Approximation is not supported, then the utility attempts to supply an alternative Color Approximation.

Drawable Creation

Some of the utilities create the drawable for you and return it to you as a parameter. Others give you a Visual and Colormap that you would use on your own call to XCreateWindow.

3

Utilities

In addition to the CGE PEX 5.1 utility extensions, vendors who support CGE PEX 5.1 fully support the existing PEX 5.1 utility functions. PEX 5.1 utilities apply to PEX 5.1 functions only. You need to invoke other CGE utilities in order to obtain information applicable to CGE PEX 5.1 functions. For example, PEXGetSizeOCs returns values that apply to PEX 5.1 output commands but does not return information about CGE PEX 5.1 OCs, such as the output command PEXExtEllipse. Extended functions are provided to support the CGE PEX 5.1 extensions as described in the following table:

Table 3-1 Extended Functions for CGE PEX 5.1

PEX 5.1 Function	**CGE PEX 5.1**
PEXCountOCs	PEXExtCountOCs
PEXDecodeOCs	PEXExtDecodeOCs
PEXEncodeOCs	PEXExtEncodeOCs
PEXFreeOCData	PEXExtFreeOCData
PEXGetSizeOCs	PEXExtGetSizeOCs
PEXFetchElementsAndSend	PEXExtFetchElementsAndSend
PEXFreePCAttributes	PEXExtFreePCAttributes
PEXFreeRendererAttributes	PEXExtFreeRendererAttributes
PEXFreeTableEntries	PEXExtFreeTableEntries
None. Use bits to set value mask.	PEXExtRendererAttributeMask
None. Use bits to set value mask.	PEXExtRendererAttributeMaskAll
PEXSetPCAttributeMask	PEXExtSetPCAttributeMask
PEXSetPCAttributeMaskAll	PEXExtSetPCAttributeMaskAll

In addition to these utilities, CGE PEX 5.1 also provides Texture Mapping utilities:

Table 3-2 Texture Mapping Utilities for CGE PEX 5.1

PEX 5.1 Function	CGE PEX 5.1
None	PEXExtCreateFilteredTM
None	PEXExtCreateFilteredTMFromWindow
None	PEXExtFreeFilteredTM
None	PEXExtTMCoordFillAreaSetWithData
None	PEXExtTMCoordQuadrilateralMesh
None	PEXExtTMCoordSetOfFillAreaSets
None	PEXExtTMCoordTriangleStrip

3