# Chapter 2

# EGL Operation

## 2.1 Native Window System and Rendering APIs

EGL is intended to be implementable on multiple operating systems (such as Sym-

### 2.1.2 Displays

Most EGL calls include an `EGLDisplay` parameter. This represents the abstract display on which graphics are drawn. In most environments a display corresponds to a single physical screen. The initialization routines described in section 3.2 include a method for querying a *default display*, and platform-specific EGL extensions may be defined to obtain other displays.

## 2.2 Rendering Contexts and Drawing Surfaces

The OpenGL ES specification is intentionally vague on how a *rendering context* (an abstract OpenGL ES state machine) is created. One of the purposes of EGL is to provide a means to create an OpenGL ES context and associate it with a surface.

EGL defines several types of drawing surfaces collectively referred to as `EGLSurface`s. These include *windows*, used for onscreen rendering; *pbuffers*

used when mixing native and OpenGL ES rendering is desirable, since there is no need to move data between the back buffer visible to OpenGL ES and the native pixmap visible to native rendering APIs. However,acesrendermay

rendering contexts to share such state rather than replicating it in each context.

EGL provides for sharing certain types of server state among contexts existing in a single address space. At present such state inclu2

# Chapter 3

tations should generate EGL_BAD_NATIVE_PIXMAP and EGL_BAD_NATIVE_WINDOW

Termination marks **all** EGL-specific resources associated with the specified display

maximum height. The value for EGL_MAX_PBUFFER_PIXELS is static and assumes that no other pbuffers or native resources are contending for the framebuffer memory. Thus it may not be possible to allocate a pbuffer of the size given by EGL_MAX_PBUFFER_

attribute will not be checked.

| Attribute |
| --- |

If no `EGLConfig` matching the attribute list exists, then the call succeeds, but *num_config*

### 3.4.3 Querying Configuration Attributes

To get the value of an `EGLConfig` attribute, use

`EGLBoolean` **eglGetConfigAttrib**

should be generated. If there is already an `EGLConfig` associated with *win* (as a result of a previous **eglCreateWindowSurface** call), then an `EGL_BAD_ALLOC` error is generated. Finally, if the implementation cannot allocate resources for the new EGL window, an `EGL`

**eglCreatePixmapSurface** call), then a `EGL_BAD_ALLOC` error is generated. Finally, if the implementation cannot allocate resources for the new EGL pixmap, an `EGL_BAD_ALLOC` error is generated.

### 3.5.4 Destroying Rendering Surfaces

An `EGLSurface`

window surface will eventually be resized by the implementation to match (as discussed in section 3.8.1). If there is a discrepancy because EGL has not yet resized the window surface, the size returned by **eglQuerySurface** will always be that of the EGL surface, not the corresponding native window.

For a pbuffer, they will be the actual allocated size of the pbuffer (which may be less than the requested size if EGL_LARGEST_PBUFFER is EGL_TRUE).

**eglQuerySurface** returns EGL_FALSE on failure and *value* is not updated. If

underlying either *draw* or *read* is no longer valid, an EGL_BAD_NATIVE_WINDOW error is generated. If *draw* and *read* cannot fit into graphics memory simultaneously, an EGL_BAD_MATCH error is generated. If the previous context of the calling thread has unflushed commands, and the previous surface is no longer valid, an EGL

Native rendering calls made with the specified marking *engine*, and which affect the surface associated with the calling thread's current context, are guaranteed to be executed before OpenGL ES rendering calls made after

### 3.8.4 Posting Errors

**eglSwapBuffers** and

# Chapter 5

# EGL Versions and Enumerants

# Chapter 6

# Glossary

**Address Space**

# Appendix A

# Index of EGL Commands