Contents

Chapter 2

EGL Operation

2.1 Native Window System and Rendering APIs

EGL is intended to be implementable on multiple operating systems (such as Symbian, embedded Linux, Unix, and Windows) and *native window systems*

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 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, pixmap surfaces may, for the same reason, have restricted capabilities and performance relative to window and pbuffer surfaces.

Native rendering will not be supported by pbuffer surfaces, since the color buffers of pbuffers are allocated internally by EGL and are not accessible through any other means.

Native rendering may be supported by window surfaces, but only if the native window system has a compatible rendering model allowing it to share the OpenGL ES back buffer.

When both native rendering APIs and OpenGL ES are drawing into the same

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 includes only *texture objects* and *vertex buffer objects*; additional types of state may be shared in future revisions of OpenGL ES where such types of state (for example, display lists) are defined and where such sharing makes sense.

2.4.1 Texture Objects

OpenGL ES texture state can be encapsulated in a named texture object. A texture object is created by binding an unused name to the texture target `GL_`

Chapter 3

EGL Functions and Errors

3.1 Errors

Where possible, when an EGL function fails it has no side effects.

EGL functions usually return an indicator of success or failure; either an `EGLBoolean`

EGL

To release resources associated with use of EGL and OpenGL ES on a display, call

```
EGLBoolean eglTerminate(EGLDisplay dpy);
```

Termination marks **all** EGL-specific resources associated with the specified display for deletion. If contexts or surfaces created with respect to *dpy* are *current* (see section

On failure, `NULL` is returned. An `EGL_NOT_INITIALIZED` error is generated if EGL is not initialized for *dpy*. An `EGL_BAD_PARAMETER` error is generated if *name*

EGL Token Name	Description
EGL_WINDOW_BIT	EGLConfig

a transparent pixel will be drawn when the red, green and blue values which

```
EGLBoolean eglGetConfigs(EGLDisplay dpy,
   EGLConfig *configs, EGLint config_size,
   EGLint *num_config);
```

to get the list of all `EGLConfig`s that are aA2 Td[(to)-196(get)-19(v)25(ailabl196(geton)-196(of)-196(asp

AtLeast Only `EGLConfig`s with an attribute value that meets or exceeds the specified value are selected.

Exact Only `EGLConfig`s whose attribute value equals the specified value are matched.

Mask Only `EGLConfig`s for which the bits set in the attribute value include all the bits that are set in the specified value are selected (additional bits might

If **eglGetConfigAttrib** succeeds then it returns `EGL_TRUE` and the value for the specified attribute is returned in *value*

3.5.2 Creating Off-Screen Rendering Surfaces

EGL supports off-screen rendering surfaces in pbuffers. Pbuffers differ from windows in the following ways:

1. Pbuffers are typically allocated in offscreen (non-visible) graphics memory

or EGL_TEXTURE_2D. The default value of EGL_TEXTURE_TARGET is EGL_NO_TEXTURE.

EGL_MIPMAP_TEXTURE indicates whether storage for mipmaps should be allocated. Space for mipmaps will be set aside if the attribute value is EGL_TRUE and EGL_TEXTURE_FORMAT is not EGL_NO

3.5.3 Creating Native Pixmap Rendering Surfaces

EGL also supports rendering surfaces whose color buffers are stored in native pixmaps. Pixmaps differ from windows in that they are typically allocated in off-screen (non-visible) graphics or CPU memory. Pixmaps differ from pbuffers in that they do have an associated native pixmap and native pixmap type, and it may be possible to render to pixmaps using APIs other than OpenGL ES and EGL.

```
EGLBoolean eglDestroySurface(EGLDisplay dpy,
   EGLSurface
```

Attribute

The specified color buffer is released back to the surface. The surface is made available for reading and writing when it no longer has any color buffers bound as textures.

The contents of the color buffer are undefined when it is first released. In particular, there is no guarantee that the texture image is still present. However, the contents of other color buffers are unaffected by this call. Also, the contents of the depth and stencil buffers are not affected by **eglBindTexImage** and **eglReleaseTexImage**.

If the specified color buffer is no longer bound to a texture (e.g., because the

If **eglCreateContext** succeeds, it initializes the rendering context to the initial

3.7.3 Binding Contexts and Drawables

To make a context current, call

```
EGLBoolean eglMakeCurrent(EGLDisplay dpy,
  EGLSurface draw, EGLSurface read,
  EGLContext ctx);
```

3.8 Synchronization Primitives

3.9 Posting the Color Buffer

After completing rendering, the contents of the color buffer can be made visible in a native window, or copied to a native pixmap.

3.9.1 Posting to a Window

To post the color buffer to a window, call

```
EGLBoolean eglSwapBuffers(EGLDisplay dpy,
  EGLSurface surface);
```

If *surface* is a window surface, then the color buffer is copied to the native window associated with that surface. If *surface* is a pixmap or pbuffer surface, **eglSwapBuffers** has no effect.

The color buffer of

eglGetProcAddress may be queried for all of the following functions:

Chapter 5

EGL Versions and Enumerants

Each version of EGL supports a specified OpenGL ES version, and all prior versions of OpenGL ES up to that version, including both Common and Common-Lite profiles. EGL 1.0 supports OpenGL ES 1.0, and EGL 1.1 supports OpenGL ES 1.1.

5.1 Compile-Time Version Detection

To allow code to be written portably against future EGL versions, the compile-time

Mark Callow, HI
Mark Tarlton, Motorola
Mike Olivarez, Motorola
Neil Trevett, 3Dlabs
Pvd[(Neil).grl, Motorola

Carlos Sarria, Imagination Technologies
Chris Tremblay, Motorola
Claude Knaus, Esmertec
Clay Montgomery, Nokia
Dan Petersen, Sun
Dan Rice, Sun
David Blythe, HI
David Yoder, Motorola
Doug Twilleager, Sun
Ed Plowman, ARM
Graham Connor, Imagination Technologies
Greg Stoner, Motorola
Hannu Napari, Hybrid
Harri Holopainen, Hybrid

Jnriaarala(,)-250(Nokia)]TJ 0 -13.549 Td[Jerray, Sun
, Imagination Technologies
ec(,)-250(ilicion)-250(Grphicss)]TJ 0 -13.549 Td[Karri, Nokia
, ymbiaun
udagv,n Te

alPloy, HI
T, Motorola
e, Motorola
Tro,
Timatos(,)-250N(viila)]TJ 0 -13.55 Td[(Perri)-250Ko Hybrid

,omordnbriAarnioNokia

Perri,omotorolOls0(Motorol15xais)-250Instrumentsn
rHybrid

Index of EGL Commands