

VGP352 – Week 9ish

⇒ Agenda:

- Discuss quiz
- Multiple render targets
- Floating-point textures
- Deferred shading



10-June-2009

© Copyright Ian D. Romanick 2009

MRT

- ⇒ Multiple color outputs from the fragment shader
 - For practical purposes, requires the use of framebuffer objects
 - Slightly changes GLSL syntax



10-June-2009

© Copyright Ian D. Romanick 2009

Framebuffer Objects

- Attach one or more renderable objects to it
 - 1D, 2D, and 3D versions exist

```
void glFramebufferTexture2DEXT (GLenum target,  
    GLenum attachment, GLenum textarget,  
    GLuint texture, GLint level);
```

```
void glFramebufferRenderbufferEXT(  
    GLenum target, GLenum attachment,  
    GLenum renderbuffertarget,  
    GLuint renderbuffer);
```

Selects how the buffer is used:

- Color buffer: `GL_COLOR_ATTACHMENT0`
- Depth buffer: `GL_DEPTH_ATTACHMENT`
- Stencil buffer: `GL_STENCIL_ATTACHMENT`



10-June-2009

© Copyright Ian D. Romanick 2009

MRT – FBO Usage

- ⇒ Use additional color attachments
 - e.g. `GL_COLOR_ATTACHMENT1`
 - Maximum number of attachments queryable with `GL_MAX_COLOR_ATTACHMENTS`
 - `EXT_fbo` requires that all color attachments have the same internal format
 - `ARB_fbo` / OpenGL 3.0 allow drivers to relax this restriction
 - The driver can still reject a particular combination
 - Most hardware can handle combinations with the same size internal formats
 - e.g. `GL_RGBA8` with `GL_RGBA_10_10_10_2`



10-June-2009

© Copyright Ian D. Romanick 2009

MRT – Setting Draw Buffers

⇒ Connect attachments with shader outputs:

```
void glDrawBuffers(GLsizei n,  
                  const GLenum *bufs);
```

- `bufs` gives a list of attachments points to connect, in the specified order, with shader outputs
 - Shader output 0 gets the first listed attachment, output 1 gets the second, etc.
- Maximum number of outputs queryable with `GL_MAX_DRAW_BUFFERS`



10-June-2009

© Copyright Ian D. Romanick 2009

MRT – GLSL Usage

- ⇒ `gl_FragColor` is but one output. What to do?
 - Replace with a new output that is declared as an array:

```
vec4 gl_FragData[ ];
```

- Each element in `gl_FragData` corresponds to one of the outputs set by `glDrawBuffers`



10-June-2009

© Copyright Ian D. Romanick 2009

Floating-point Textures

- Floating-point texture support added by OpenGL 3.0 and `GL_ARB_texture_float`
 - Adds several floating-point texture formats:
 - `GL_RGBA32F`
 - `GL_RGB32F`
 - `GL_RGBA16F`
 - `GL_RGB16F`
 - `GL_ARB_texture_rg` adds several more:
 - `GL_RG32F`
 - `GL_R32F`
 - `GL_RG16F`
 - `GL_R16F`



10-June-2009

Floating-point Textures

- ⇒ Floating-point formats used as the internal format of textures and renderbuffers
 - For texturing, they're just like any other internal format
 - Older hardware may not support all filtering points on FP textures



10-June-2009

© Copyright Ian D. Romanick 2009

Floating-point Rendering

➤ Additional clamping control is needed

- By default, all color outputs are clamped to [0, 1]
- Can be relaxed:

```
void glClampColor(GLenum target,  
                 GLenum clamp);
```

- target is GL_CLAMP_VERTEX_COLOR, GL_CLAMP_FRAGMENT_COLOR, or GL_CLAMP_READ_COLOR
- clamp is GL_FIXED_ONLY, GL_TRUE, or GL_FALSE
- Requires OpenGL 2.1 or GL_ARB_color_buffer_float



10-June-2009

© Copyright Ian D. Romanick 2009

Floating-point Rendering

- Default clamping mode for fragment outputs is `GL_FIXED_ONLY`
 - This means that clamping only happens when the output is fixed-point (i.e., not FP)
- One last warning:
 - Older hardware *cannot* blend into FP render targets



10-June-2009

© Copyright Ian D. Romanick 2009

References

Jones, Rob. "OpenGL Frame Buffer Object 201." GameDev.net. December 14th, 2006. Accessed on June 10th, 2009.
<http://www.gamedev.net/reference/articles/article2333.asp>



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading

- Scenes with high depth complexity or many lights suffer from several problems:
 - Many passes to implement the lights
 - Lots of wasted fragment processing
 - Difficulty with per-batch storage for shadow maps
 - Difficulty with stencil shadows from multiple lights
 - etc.
 - End result: poor performance



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading

- ⇒ What if we could easily:
 - Light each pixel (not fragment) exactly once
 - Only apply lights to the fragments they affect
 - Reduce per-light cost in scenes with many lights



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading

⇒ General idea:

- Render scene information needed for shading to an off-screen geometry buffer (G-buffer)
- Draw per-light geometry to screen sampling from G-buffer to calculate shading



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading – G-Buffer

- ⇒ All per-fragment data required for shading:
 - Normal
 - Position
 - Diffuse / specular color
 - etc.
- ⇒ Emit this during per-object rendering
 - Output this data instead of performing lighting calculations
 - Use MRT!



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading – G-Buffer

⇒ Example G-buffer layout:

- 2 RGBA16F outputs:

Diffuse (red)	Diffuse (green)	Diffuse (blue)	m
Normal (X)	Normal (Y)	Normal (Z)	n

- m is the Cook-Torrance roughness
- n is the index of refraction



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading – G-Buffer

⇒ Tough choices:

- Explicitly store position or derive from screen X/Y and depth value?
- Explicitly store the normals Z or derive from its X and Y?
- One of the most important parts of designing a deferred shading engine is selecting the parameters and the packing



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading – Lighting

- For each light, draw simplified bounding geometry
 - Perform lighting for each fragment drawn
 - Only light the areas of the scene that need lighting
 - Read from G-buffer at the screen X/Y position
 - Add calculated lighting to existing values
 - Examples:
 - Directional light: box
 - Point light: sphere
 - Spot light: cone



10-June-2009

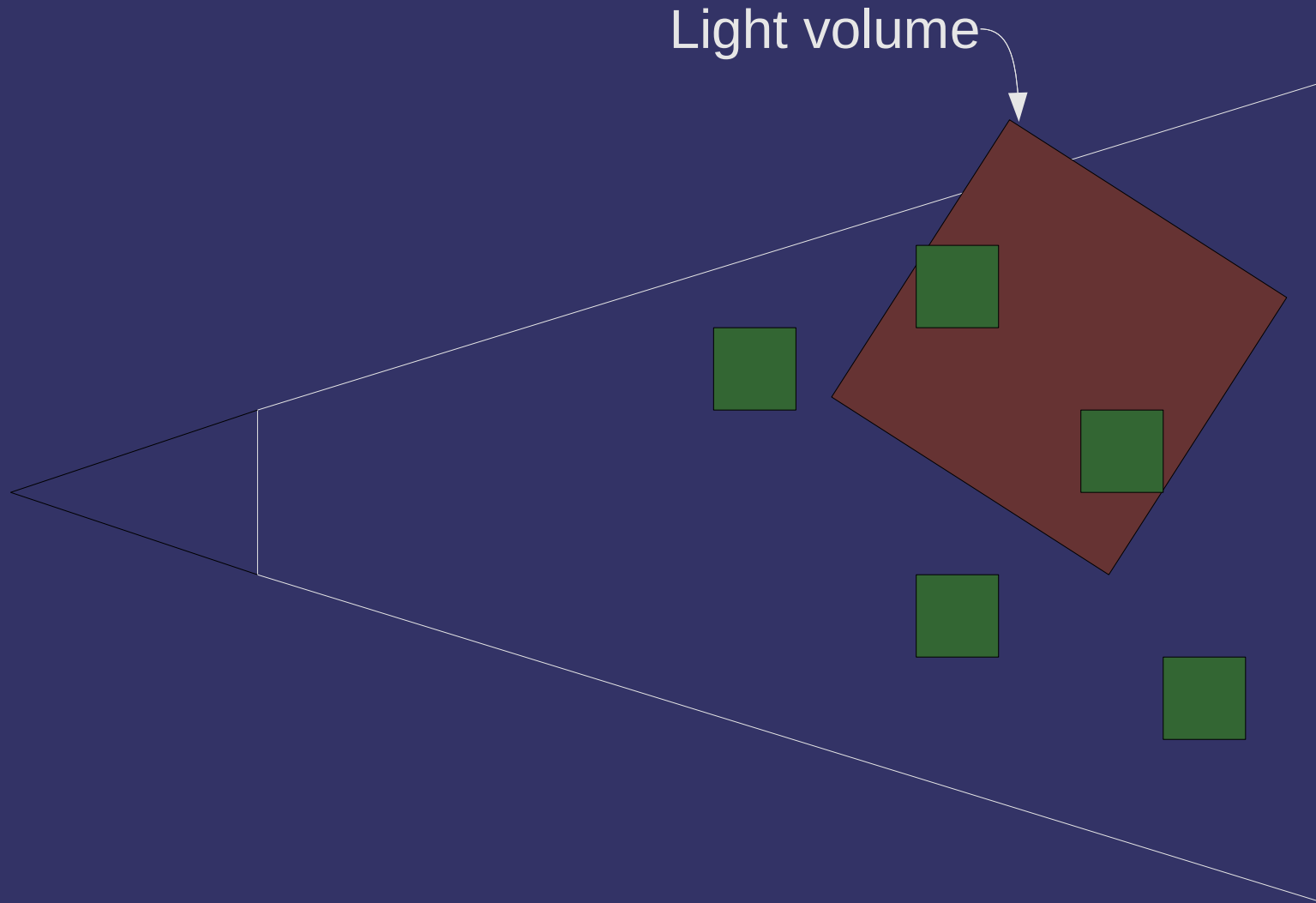
© Copyright Ian D. Romanick 2009

Deferred Shading – Lighting

- Optimize by letting the early stencil test discard many fragments
 - Draw the light volume once:
 - Disable color writes
 - Set depth function to `GL_LESS` and stencil function to `GL_ALWAYS`
 - Set Z-fail stencil operation to `GL_REPLACE` and all others to `GL_KEEP`
 - Draw the light volume again:
 - Enable color writes
 - Set depth function to `GL_LEQUAL` and stencil function to `GL_EQUAL`
 - Set all stencil operations `GL_KEEP`



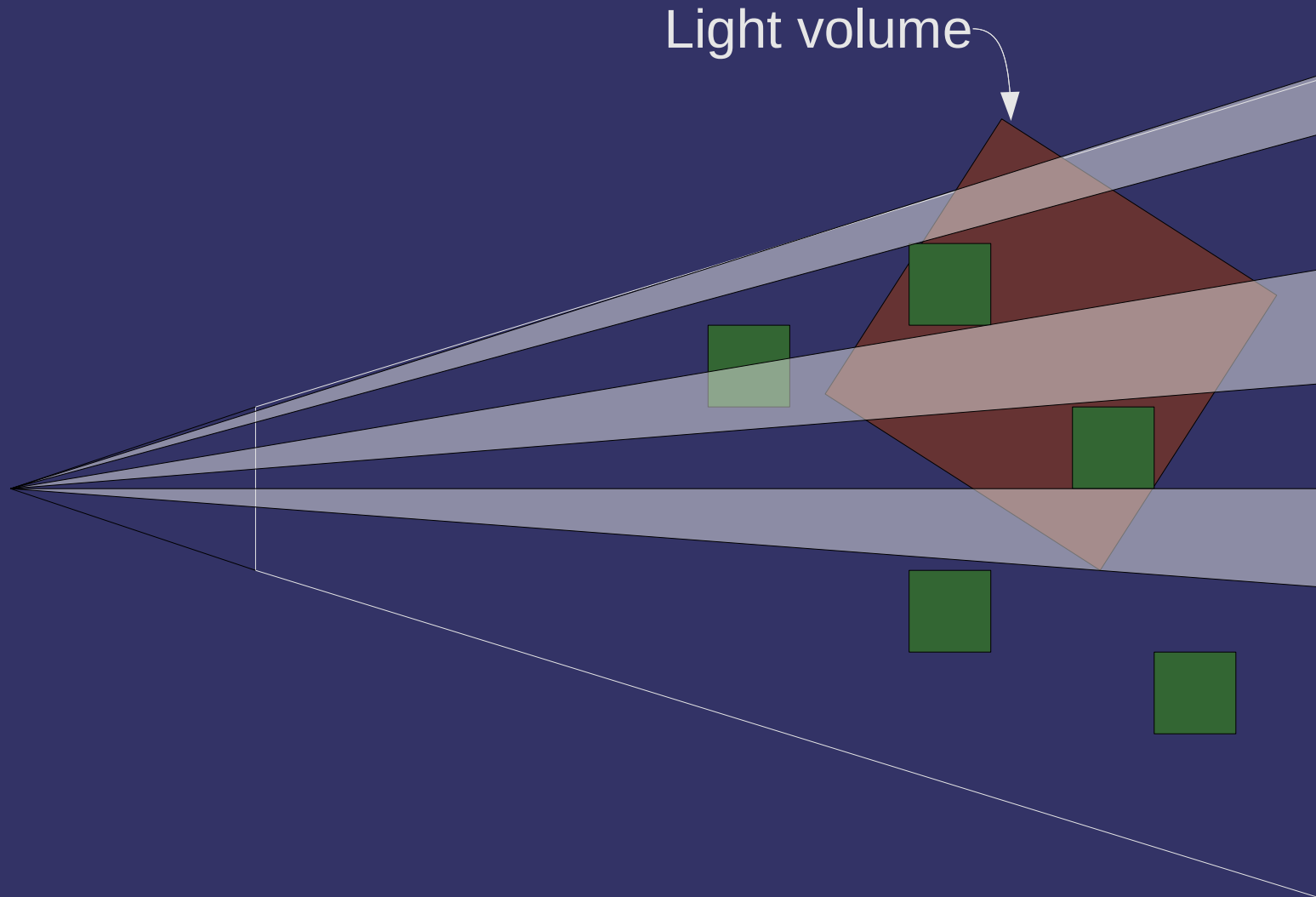
Deferred Shading – Lighting



10-June-2009

© Copyright Ian D. Romanick 2009

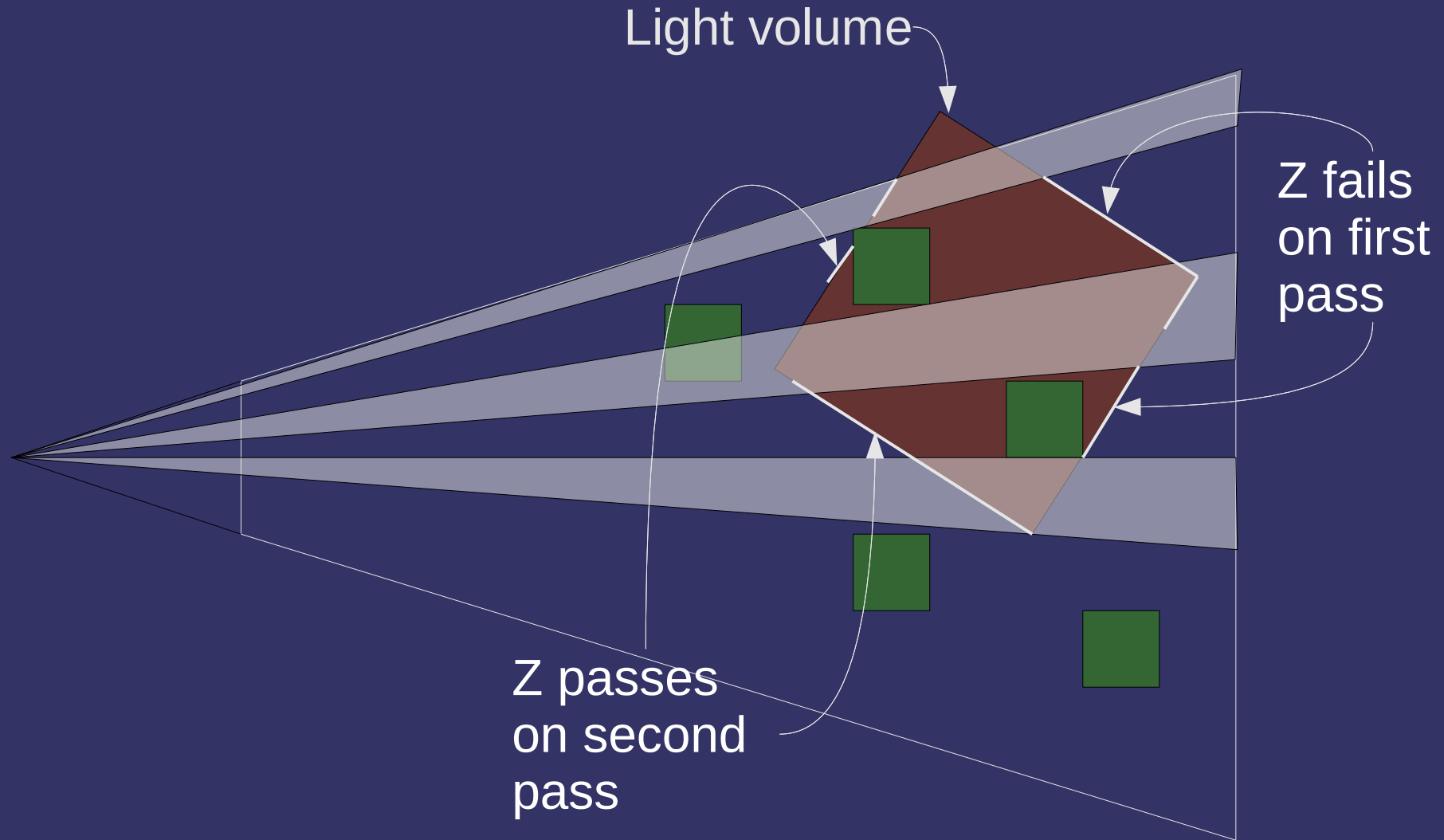
Deferred Shading – Lighting



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading – Lighting



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading – Drawbacks

⇒ What could go wrong?



10-June-2009

© Copyright Ian D. Romanick 2009

Deferred Shading – Drawbacks

⇒ What could go wrong?

- Transparency effects won't work
- Traditional anti-aliasing (multisampling) has problems



10-June-2009

© Copyright Ian D. Romanick 2009

References

Hargreaves, S., Harris, M. “Deferred Shading.” Nvidia 6800 Leagues Under the Sea. June 2004.

http://developer.nvidia.com/object/6800_leagues_deferred_shading.html

Fabio Policarpo, Francisco Fonseca, *Deferred shading tutorial*. Pontifical Catholic University of Rio de Janeiro. 2005.

http://www710.univ-lyon1.fr/~jciehl/Public/educ/GAMA/2007/Deferred_Shading_Tutorial_SBGAMES2005.pdf



10-June-2009

© Copyright Ian D. Romanick 2009

Next week...

- ⇒ And by “next week” I mean tomorrow...
 - Curves and curved surfaces
 - Discuss final



10-June-2009

© Copyright Ian D. Romanick 2009

Legal Statement

This work represents the view of the authors and does not necessarily represent the view of Intel or the Art Institute of Portland.

OpenGL is a trademark of Silicon Graphics, Inc. in the United States, other countries, or both.

Khronos and OpenGL ES are trademarks of the Khronos Group.

Other company, product, and service names may be trademarks or service marks of others.



10-June-2009

© Copyright Ian D. Romanick 2009