OpenGL for Mac OS X

Matt Collins
GPU Software

Introduction

You’re wondering where we’re at?

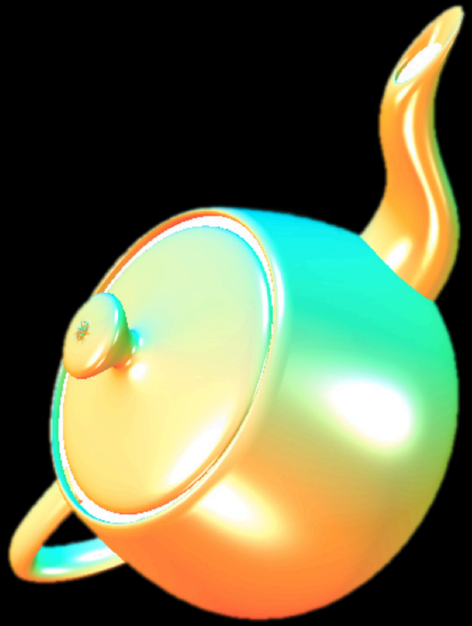
- What can I target?
- What’s new in 10.6.3 and 10.6.4?
- What does this mean for your app?

Your App Looks Great … Now You Want SPEED

- Performance tips and tricks to make your app shine

I’ve Heard About This Super Inferred Phong Shading Thing

- Cool techniques with 10.6.3+
- Rendering techniques and examples



OpenGL

- Lowest-level access to graphics hardware
- Most other graphics frameworks live on top of OpenGL
  - Core Image
  - Core Animation
  - Quartz Composer

Last Time at WWDC…

- Buffer objects
  - Vertex buffers
  - Index buffers
- Frame buffer objects
- Fixed function pipeline
  - Multitexture
- Programmable pipeline—shaders
  - Vertex/Geometry/Fragment shader

Where Are We Now?

- Better access to hardware functionality!
- 10.6.3 and above only!

First, Some Advice…

- Use Generic Vertex Attributes
    - Shaders are native
    - Fixed Function is emulated in the drivers
    - Easier to share with OpenGL ES 2.0

Now Available!

Now Available

- Making life easier
  - `EXT_provoking_vertex`
  - `EXT_vertex_array_bgra`
  - `ARB_depth_buffer_float`
- Empowering your app
  - `ARB_framebuffer_object`
  - `ARB_texture_array`
  - `ARB_instanced_arrays`

Now Available!

Now Available

- Performance and memory
  - `NV_conditional_render`
  - `ARB_texture_rg`
    - `EXT_texture_compression_rgtc`
  - `EXT_packed_float`
  - `EXT_texture_shared_exponent`

Learning!

- What does it do?
- What's my motivation, director?
- Demo of cool stuff!

Flexibility

Provoking Vertex Selection

EXT_provoking_vertex

- Control which vertex supplies attributes during Flat shading
- New entry points:
  - `glProvokingVertexEXT`(GLenum mode)
    - `GL_FIRST_VERTEX_CONVENTION`
    - `GL_LAST_VERTEX_CONVENTION`
- What about Quads?
  - Hardware dependent behavior
  - `GL_QUADS_FOLLOW_PROVOKING_VERTEX`

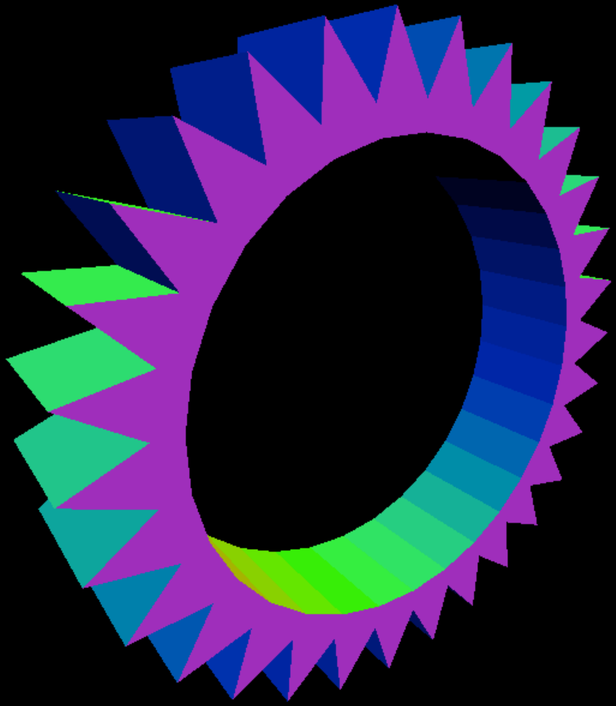
Provoking Vertex Selection

EXT_provoking_vertex

- Motivation?
  - Better flexibility
  - Allows the app to pick where to pull color/attributes from without modifying art assets

Provoking Vertex Selection

EXT_provoking_vertex



BGRA Ordering

EXT_vertex_array_bgra

- Specify colors in BGRA order
- `GL_BGRA` is the `SIZE` parameter:
  - `glColorPointer`
  - `glSecondaryColorPointer`
  - `glVertexAttribPointer`
- Size implied to be four
- Unsigned bytes only!

BGRA Ordering

EXT_vertex_array_bgra

```
glBindBuffer(GL_ARRAY_BUFFER, colorVBOName);
glVertexAttribPointer(index, GL_BGRA, GL_UNSIGNED_BYTE,
                      GL_FALSE, 0, NULL);
```

Floating Point Depth Buffers

ARB_depth_buffer_float

- Floating point depth buffer
- New formats
  - `GL_DEPTH_COMPONENT32F`
  - `GL_DEPTH32F_STENCIL8`
- New type
  - `GL_FLOAT_32_UNSIGNED_INT_24_8_REV`

Floating Point Depth Buffers

ARB_depth_buffer_float

- Motivation
  - Very deep scenes
  - Very small scenes
- Note:
  - Precision is greater closer to the near plane
  - Precision is greater closer to 0.0

Empowerment

Array Textures

EXT_texture_array

- Array of 1D or 2D textures
  - Each layer is a distinct image
  - No filtering between layers
  - Distinct mipmaps per level
- Programmable pipeline only!
- New texture target
  - `GL_TEXTURE_2D_ARRAY_EXT`, `GL_TEXTURE_1D_ARRAY_EXT`
- New samplers
  - `sampler2DArray, sampler1DArray`

Array Textures

EXT_texture_array

- Why?
  - Store unique data slices
- Why not use 3D textures?
  - Can’t mipmap each level

Demo

Array height maps

Instancing

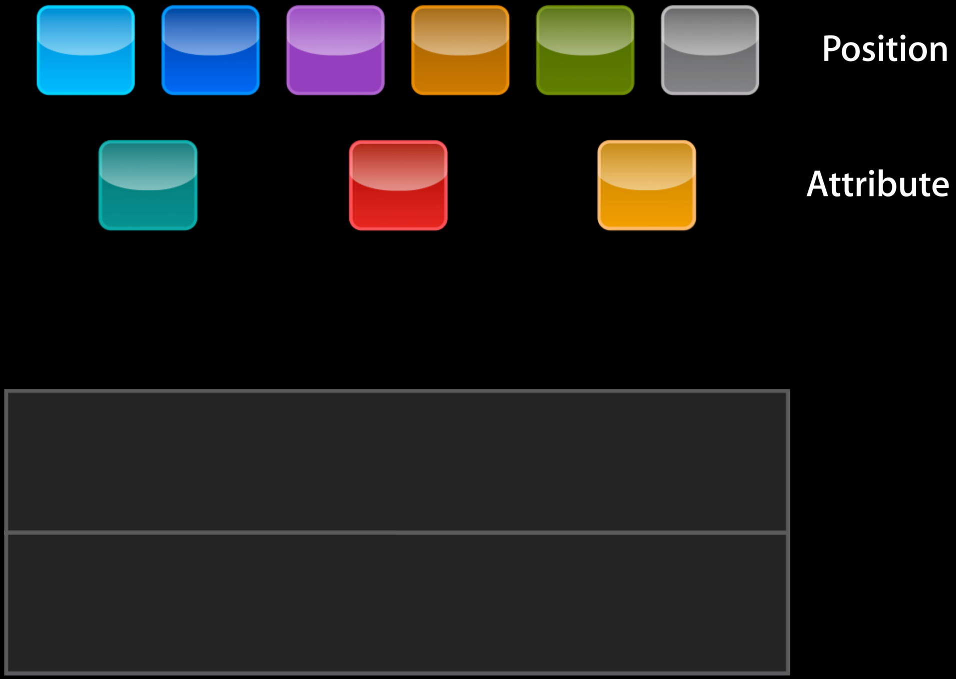
ARB_instanced_arrays

- Reuse primitives within a single draw
- Programmable Pipeline only!
- Requires Generic Vertex Attributes
- Vertex attributes at different rates
    - `glVertexAttribDivisorARB`

Instancing

ARB_instanced_arrays

- Saves overhead
- Many different techniques
  - Stream instancing
    - Source vertex attributes at different rates
    - Position/orientation matrices, for example



Demo

Instanced gears

Frame Buffer Objects

ARB_framebuffer_object

- Generalized offscreen render targets!
- Different dimensions
- Different formats

Frame Buffer Objects

ARB_framebuffer_object

- FBOs themselves are not new
- But ARB_`fbo` allows new techniques
  - Reuse Z-buffer
  - Render various data types
- More on this later!

Performance and Memory

RG Textures

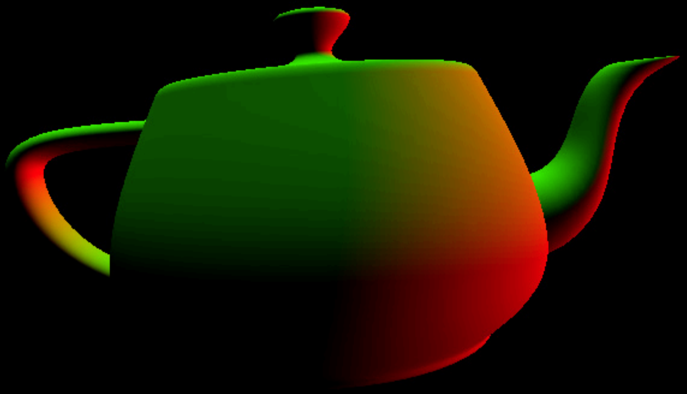
ARB_texture_rg

- One and two channel textures
- Can be R or RG
    - Many formats, including:
        - 8/16/32 unsigned ints
        - 16/32 floating point
- Can be a render target!

RG Textures

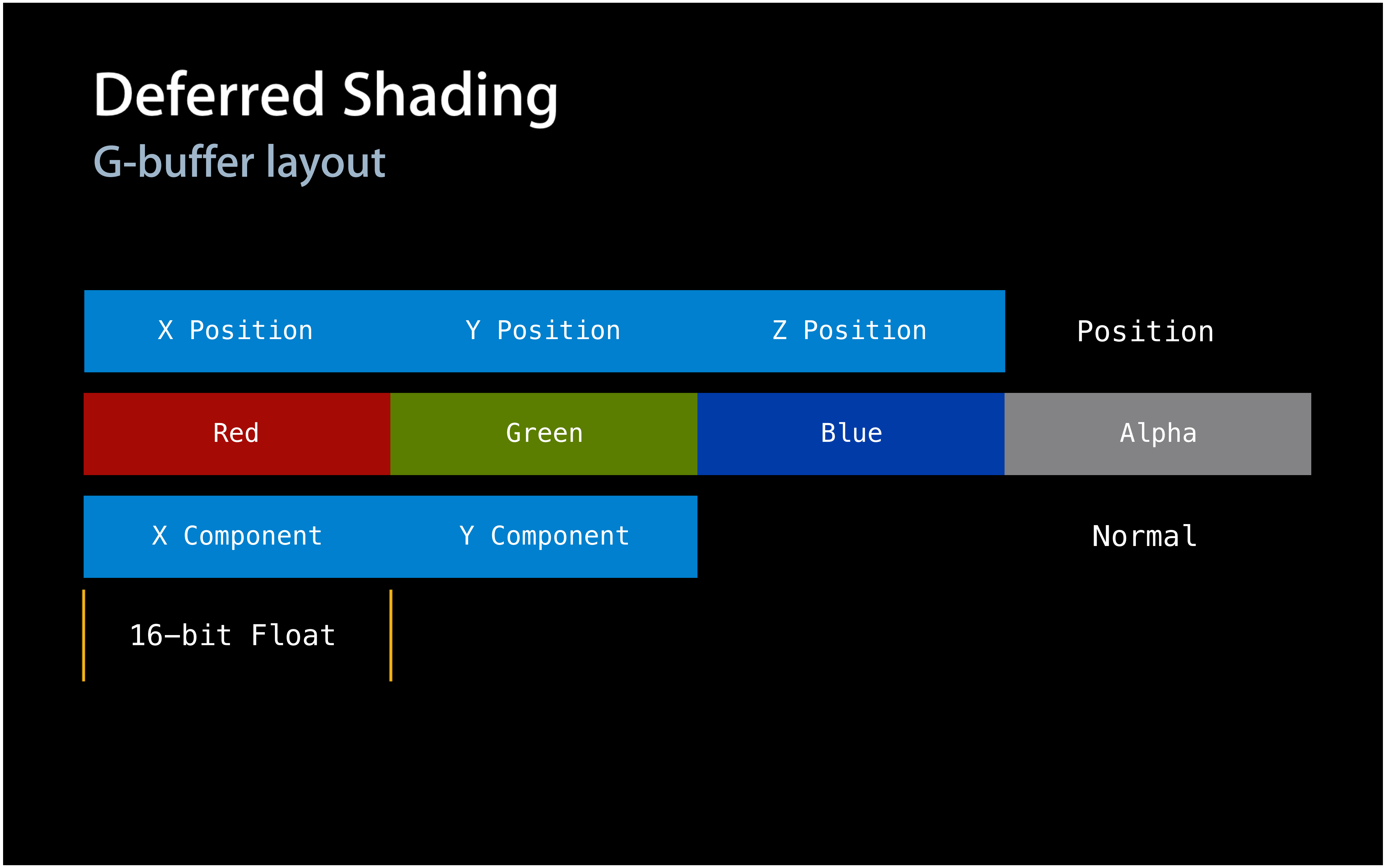
ARB_texture_rg

- But why?
  - Combines with ARB_fbo
  - Rendering data to a texture
  - Luminance isn’t renderable
- You may not need four components
  - Screen space motion blur
  - Deferred shading



Deferred Shading

- Transform geometry as usual
- Render lighting attributes for each visible pixel to G-buffer
- Render fullscreen quad
- Read from G-buffer to perform lighting calculations in screen-space



Deferred Shading

G-buffer layout

Deferred Shading

Storing out attributes

```
varying vec3 position, normal;
varying vec4 color;

void main() {

  gl_FragData[0].xyz = position.xyz;

  gl_FragData[1] = color;

  gl_FragData[2].xy = n.xy;

}
```

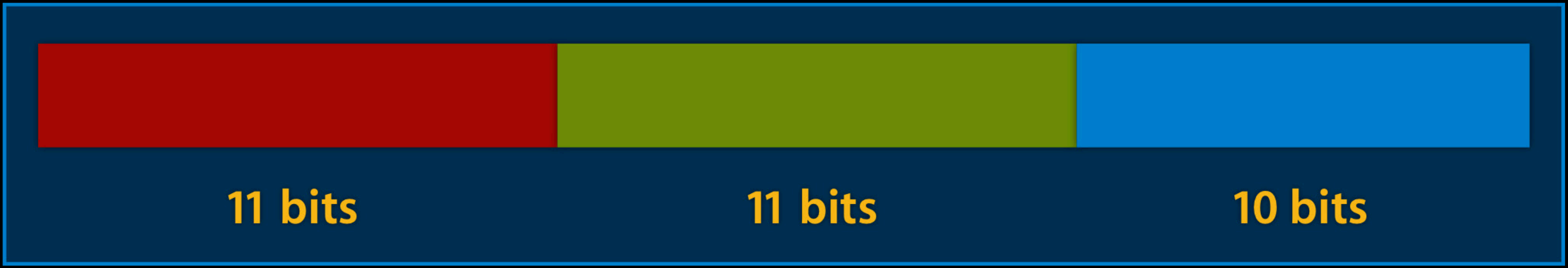
Demo

Deferred shading

Packed Floats

EXT_packed_float

- Pack three floats into 32 bits
- Format
  - `<internalformat> GL_R11F_G11F_B10F_EXT`
  - `<type> UNSIGNED_INT_10F_11F_11F_REV_EXT`



Packed Floats

EXT_packed_float

- Why?
  - High dynamic range
  - Sun is 1000x brighter than shadow
  - 8 bits are not enough to express this!

Conditional Rendering

NV_conditional_render

- Rendering based on occlusion queries
  - Removes roundtrip from GPU
- `glBeginConditionalRenderNV(GLuint id, GLenum mode)`
  - `id`—An occlusion query
  - `mode`—Wait or not
- `glEndConditionalRenderNV()`

Conditional Rendering

NV_conditional_render

```
// Setup by turning off writes
glColorMask(GL_FALSE, GL_FALSE, GL_FALSE, GL_FALSE);
glDepthMask(GL_FALSE);
```

```
glBeginQuery(GL_SAMPLES_PASSED_ARB, query);
// Draw your coarse bounding volume
glDrawElements(...);
glEndQuery(GL_SAMPLES_PASSED_ARB);
```

```
glColorMask(GL_TRUE, GL_TRUE, GL_TRUE, GL_TRUE);
glDepthMask(GL_TRUE);
```

```
// Conditionally render based on query result
glBeginConditionalRenderNV(query, GL_QUERY_WAIT_NV);
glDrawElements(...);
glEndConditionalRenderNV();
```

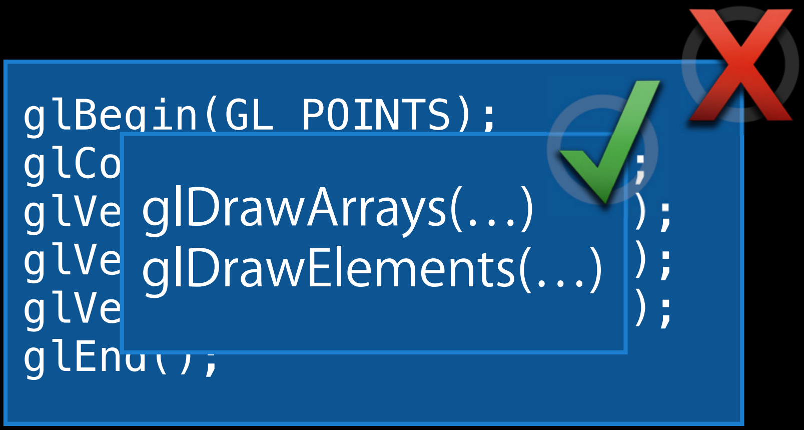
Demo

Conditional rendering

Performance Tips

Some stuff to avoid…

- Immediate mode is costly!
  - Specifying every point individually is SLOOOOOW
  - All data sent over the bus
  - You have VRAM, use it!



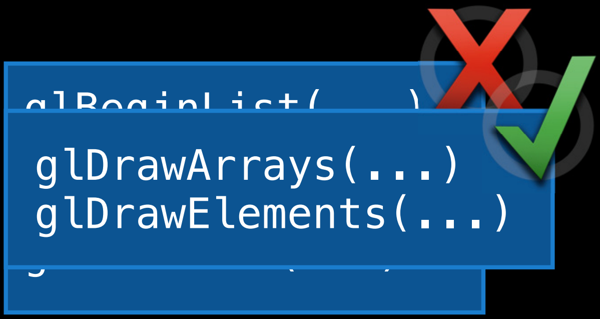
```
glBegin(GL_POINTS);
glCo                                  ;
glVe                                  );
glVe                                  );
glVe                                  );
glEnd();
```

```
glDrawArrays(…)
glDrawElements(…)
```

Performance Tips

Some stuff to avoid…

- Display Lists don't really help…
  - Not a performance boost
  - Sure you are caching commands, but Display Lists inherit state
  - So, we can't cache state—which is what hurts you



```
glDrawArrays(...)
glDrawElements(...)
```

Performance Tips

- Batch your state!
  - Important way to improve performance
  - All state changes require driver validation
    - Also sends a state vector to hardware
  - This is expensive!
    - Avoid by drawing similar objects
- Check Shark to see where time is being spent
- Also use Driver Monitor

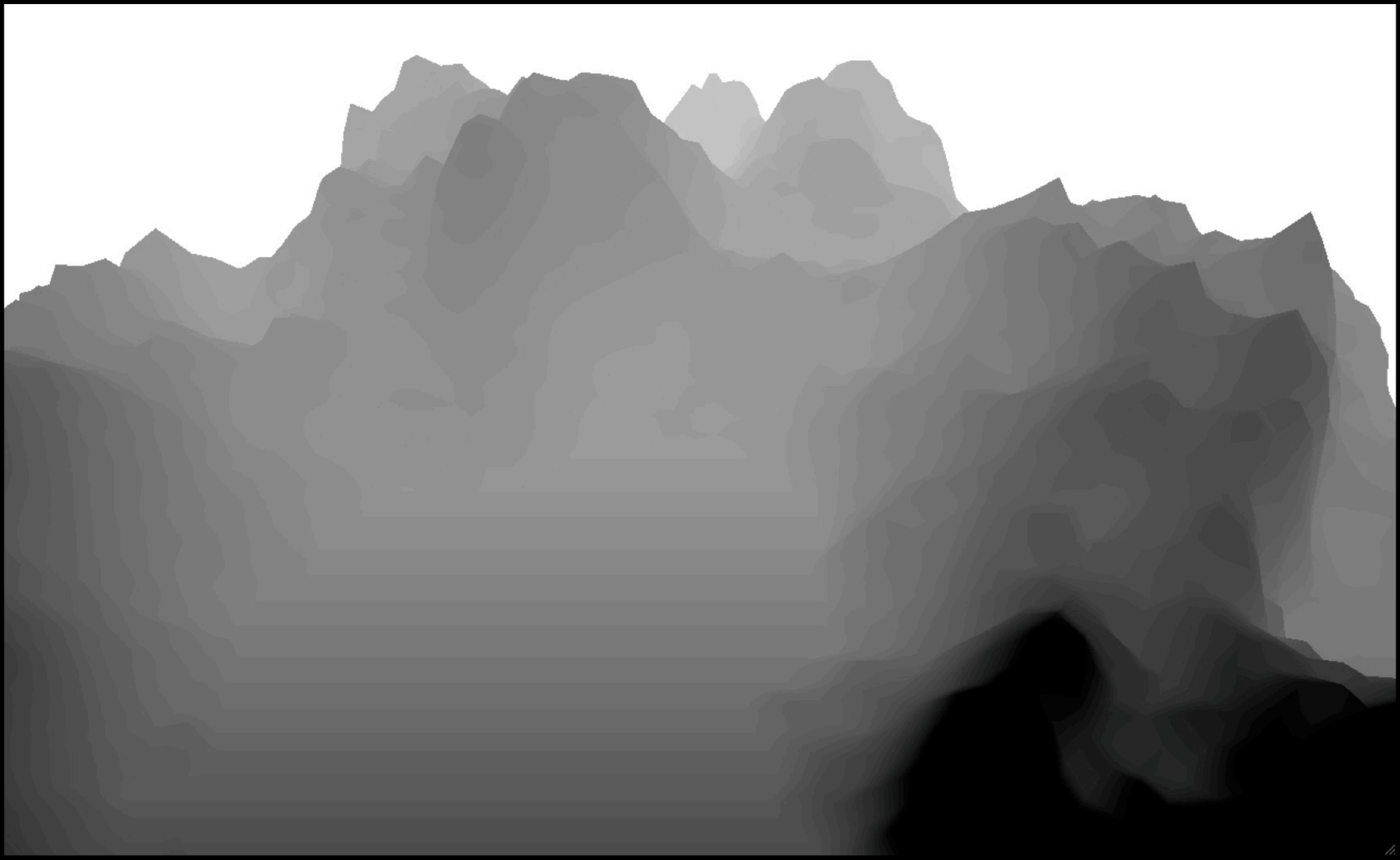
Performance Tips

- Hoist heavy calculations up the pipeline
    - Vertex shader may run 10,000 times a frame
    - Fragment Shader will run 1600 x 1200 = 1.92 million times!
        - Much more with overdraw!
- Keep an eye out for fallback!
    - `CGLGetParameter (ctx, kCGLCPGPUVertexProcessing, &vtx);`
    - `CGLGetParameter (ctx, kCGLCPGPUFragmentProcessing, &frag);`

Performance Tips

- Z-prepass with color writes turned off
  - Depth writes 2x as fast
  - Z-test AFTER the Fragment stage
  - Premade Z-buffer allows early-Z optimizations
    - Helps complex shaders and lots of overdraw
  - Certain techniques need incoming-Z
    - Crytek-style Screen Space Ambient Occlusion

Performance Tips



Performance Tips

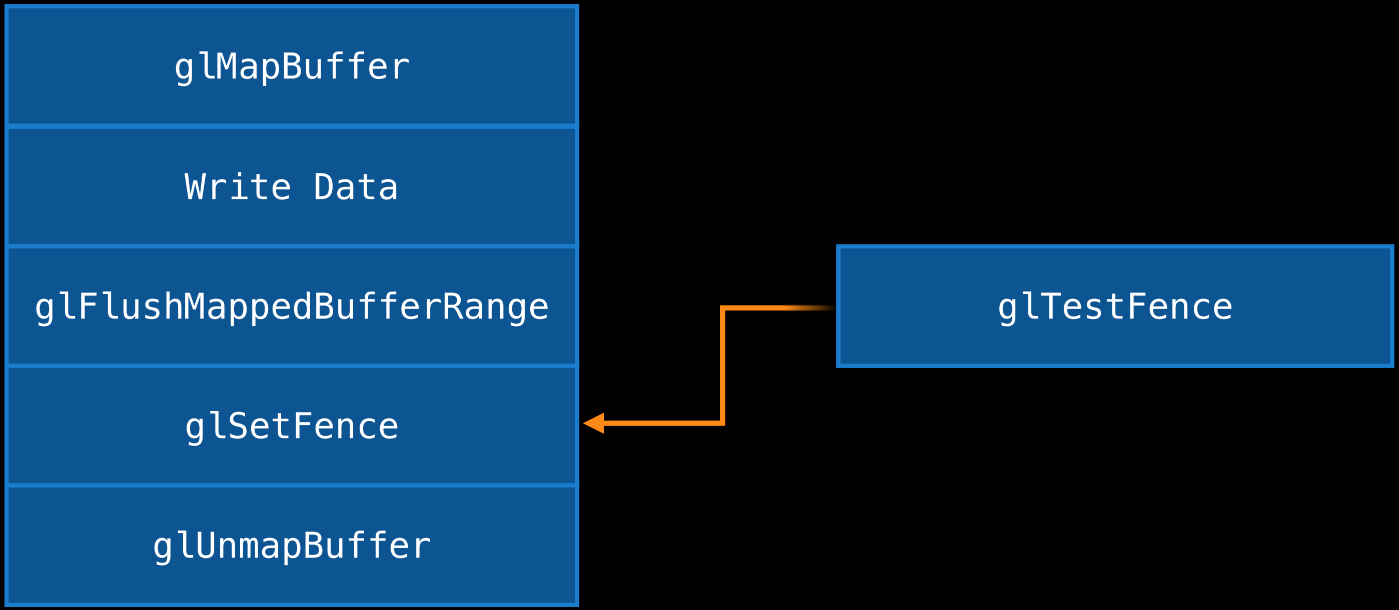
- `glFlushMappedBufferRange`
  - Map/Unmap will DMA the entire buffer
  - Asynchronous modification of the buffer object
  - Minimizes data copied back to system memory

```
glBufferParameteriAPPLE(GL_ARRAY_BUFFER,
       GL_BUFFER_FLUSHING_UNMAP_APPLE,
       GL_FALSE);
… //do unrelated work
GLvoid *data = glMapBuffer(GL_ARRAY_BUFFER, GL_WRITE_ONLY);
… //update data buffer
glFlushMappedBufferRangeAPPLE(GL_ARRAY_BUFFER, offset, bytes);
success = glUnmapBuffer(GL_ARRAY_BUFFER);
```

Performance Tips

- `glFence`
  - Test when a command is done
    - `glFlushMappedBufferRange` completed?
  - Needed with Multithreaded Engine and `glMapBuffer`!
  - Multithreaded synchronization
    - Texture upload on background context

Performance Tips



```
GLuint fence;
glGenFencesAPPLE(1, &fence);
... // do work here
glSetFenceAPPLE(fence);
... // do unrelated stuff
glTestFenceAPPLE(GL_FENCE_APPLE, fence);
```

Can We Put This All Together?

- An example leveraging these different techniques at once?
  - Instancing—lots of objects
  - `Texture_rg`—some type of deferred shading
  - Array Textures—terrain

Sources

- Hargreaves, Shawn and Harris, Mike. 2004. "Deferred Shading." Presentation. http://download.nvidia.com/developer/presentations/2004/6800_Leagues/6800_Leagues_Deferred_Shading.pdf
- Mitting, M. 2007. "Finding Next Gen—CryEngine 2." SIGGRAPH 2007.
- Shishkovtsov, Oles. 2006. "Deferred Shading in S.T.A.L.K.E.R." In *GPU Gems 2*, Addison-Wesley. pp 143-166
- Valiant, M. 2007. "Deferred Rendering in Killzone 2." http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf

More Information

Allan Schaffer
Graphics and Game Technologies Evangelist
aschaffer@apple.com

Documentation
OpenGL Dev Center
http://developer.apple.com/opengl

Apple Developer Forums
http://devforums.apple.com

Related Sessions

OpenGL Essential Design Practices	Pacific Heights Wednesday 11:30AM
OpenGL ES Overview for the iPhone OS	Presidio Wednesday 2:00PM
OpenGL ES Shading and Advanced Rendering	Presidio Wednesday 3:15PM
OpenGL ES Tuning and Optimization	Presidio Wednesday 4:30PM
Taking Advantage of Multiple GPUs	Nob Hill Thursday 10:15PM

Labs

OpenGL for Mac OS X Lab	Graphics and Media Lab C Thursday 2:00PM
OpenGL ES Lab	Graphics and Media Lab A Thursday 9:00AM

WWDC10