OpenGL and OpenGL ES

Dr. Thomas J Olson
OpenGL ES Working Group Chair
Director, Graphics Research, ARM MPD
The OpenGL Family: 3D Everywhere!

Cross-Platform 3D Graphics on the Desktop

Light-weight OpenGL subset for mobile, TV, automotive...

OpenGL ES 2.0 capability in a web browser
What is new in OpenGL 4.2?

• **GPU Computing in OpenGL**
  - Atomic counters
  - Shader I/O to memory

• **Does not replace OpenCL**
  - Does allow tightly coupled graphics and computing

• **Other New Features**
  - Immutable texture objects
  - Instanced transform feedback
  - BPTC texture compression
  - Shading language packing
OpenGL Compute: Example

• How to render transparent objects?
  - Correct result depends on drawing order
  - But sorting geometry on CPU is expensive!
Order-independent transparency

Approaches
  - Depth peeling
  - K-buffer

A new approach using OpenGL Compute:
  - Collect a list of fragments at each pixel
  - Sort and composite as a post-processing step
Collecting Fragments

- Integer texture stores count of fragments at each pixel
- Texture array stores color and depth for each fragment
Collecting Fragments

- Integer texture stores count of fragments at each pixel
- Texture array stores color and depth for each fragment
- Update count using atomic increment
Collecting Fragments

- Integer texture stores count of fragments at each pixel
- Texture array stores color and depth for each fragment
- Update using atomic increment
Collecting Fragments

- Integer texture stores count of fragments at each pixel
- Texture array stores color and depth for each fragment
- Update using atomic increment
- Result: unordered list of fragments
Post-Processing

- Draw a full-screen triangle
- Count contains number of valid fragments
- Sort in correct order and composite
Improvements and Issues

You can sort by depth while rendering
- Insertion sort is fine for a small number of layers

But what if you run out of layers?
- An interesting problem!
- Can reformulate the problem and approximate
OpenGL ES Today

The leading 3D API for mobile
- Standard on iOS, Android, Linux...
- Both 2D UI and games
- Both system and application

What’s new in 2012?
- Game engines: UE3, Unity, ...
- Advanced UI frameworks
- Robust market for high-end content

For us, this is success!
Working Group Activities

• **Next Generation OpenGL ES**
  - Working group’s main focus since mid-2009
  - Will be released when the market is ready

• **ARB / ES Convergence TSG**
  - Reorganizing to co-develop new technology
  - Example: immutable textures
  - Example: internalformat query
Portability in OpenGL ES 2.0

Dr. Thomas J Olson
Director, Graphics Research, ARM MPD
Chair, OpenGL ES Working Group
Writing Portable OpenGL ES 2.0

• Why is this a problem?
• What are the key portability issues in OpenGL ES 2.0?
• What are the best practices for managing them?
• Caveats
  - Just my opinion
  - Just a beginner’s guide
Portable Graphics: Problem Solved?

• OpenGL ES 2.0 is everywhere!
  - iOS
  - Android
  - WebOS
  - Symbian
  - etc

• So, everything is OK now, isn’t it?
  - Sadly, no.
  - And, it isn’t an accident
Why isn’t this a solved problem?

- Graphics APIs must trade off *portability* against *freedom to innovate*
Portability Issues in OpenGL ES 2.0

- Performance characteristics

- Implementation options
  - Implementation-defined limits
  - Shader engine arithmetic precision
  - Shading Language restrictions
  - Error Behavior

- Extensions
  - Texture compression
  - “Silent” extensions
Performance – Driver Issues

• Deferred execution
  - Rendering
  - Texture Upload
  - Compilation

• Compilation Speed

• State Changes
  - State dependency triggered compiles

• Draw Call Overhead
Performance – Other Issues

• **GPU Hardware**
  - Lots of variation even within a single GPU architecture / family
  - Each architecture has different strengths and weaknesses
  - Optimizing for one may hurt you on another

• **Best Practices:**
  - Focus on generic optimizations first
  - Get to know vendor DevRel teams and documents, and test on all your target GPUs early in the development cycle
  - Design your engine to adapt to device performance and capabilities
Implementation-defined limits

• OpenGL ES 2.0 allows implementations to differ
  - Example, “how many vertex attributes can I use”?
  - Application can query the implementation limit
  - Spec mandates minimum value
    - See specification Table(s) 6.17-6.20

• Best Practices
  - Stay within the Chapter 6 minimum-maximum values if you can
  - Or, query and adapt
Shader Engine Arithmetic Precision

• GLSL ES allows you to apply precision qualifiers to variables
  - Lowp (10-bit), mediump (16-bit), highp (24-bit)

• Qualifiers specify the minimum precision to provide
  - Implementations can provide more than you ask for
  - You can query via `GetShaderPrecisionFormat()`

• Support for highp is *optional* in the fragment shader
Best Practices - Arithmetic Precision

- **Watch out for**
  - large texture wrap factors
  - sensitive reflection vectors

- **Declare** `#precision float mediump` in your fragment shader
  - If you use `highp`, provide a fallback for implementations that don’t support it

- **Test on implementations that don’t provide highp**
Shading Language Restrictions

• GLSL ES 1.0 allows implementations to provide less-than-full support for the core shading language
  - No virtualization
  - Limited dynamic control flow
  - Limited array or vector indexing
    - May require indices to be compile-time constants
  - See GLSL ES 1.0 specification, Appendix A

• Many vendors relax some of these restrictions
  - ...but there is no query
Best Practices - Language Restrictions

• Stay within Appendix A guidelines if you can

• Or, be prepared to fall back

• Test using vendor off-line shader compilers
  - Integrate into your build system or shader authoring pipeline
Error Behavior

• Invalid pointers
  - Attribute buffers

• Shading Language Errors
  - Divide by zero
  - NaN generation / propagation

• API restrictions
  - TexImage2D type / format / internalformat restrictions
Extensions

• **Implementations can add new features**
  - Defined as additions to the core spec
  - Example, OES_texture_npot
  - Query using GetString[v]()

• **Categories**
  - OES: Ratified by Khronos, conformance tested
  - EXT: Supported by multiple vendors
  - Vendor: Supported by a single vendor, possibly proprietary

• **Best Practices**
  - AVOID if you can
  - If you use, prefer OES to EXT to Vendor
  - Always query and adapt
Texture Compression

• Texture compression is great! Use it!
  - Formats: ETC1, PVRTC, DXTn, ...

• The bad news:
  - No universally supported formats
  - To target both iOS and Android, you will need at least two art paths

• Best Practices
  - Use ETC1 (OES extension) if you can
  - On iOS: Use PVRTC (vendor extension)
  - Query for other vendor extensions
Using ETC1 textures

• **ETC1 is RGB-only. What if I need RGBA?**
  - Use two channels of ETC1
  - Or, use ETC1 for RGB and pack A into another texture

• **What about normals?**
  - Use two channels of ETC1 and compute third channel

• **In general, ETC1 has excellent resolution in luminance**
  - Works well when R/G/B are highly correlated
“Silent” Extensions

- OpenGL ES 2.0 restricts some features of OpenGL

- Example
  - `glTexImage2D(T, L, internalformat, W, H, B, format, type, d);`
  - In ES, `format` must match `internalformat`

- Rules are not well tested by the ES conformance test
  - Some implementations allow features that they shouldn’t
Best Practices - “Silent” Extensions

• Test on multiple implementations

• Know the rules

• Yell at ES 2.0 implementors who break them

• (I’ll help)
Summary

- **Writing portable OpenGL ES 2.0 is challenging!**
  - It’s the price we pay for rapid evolution of the technology

- **To meet the challenge:**
  - Know the specification
  - Design for the portable core of the API
  - Debug your code
  - Avoid implementation-specific features if you can
  - Test on all of the major implementations

- **Good luck!**
Acknowledgements

• Thanks to:
  - Maurice Ribble (Qualcomm)
  - Daniel Koch (Transgaming)
  - Acorn Pooley (NVidia)
  - Anders Lassen (Outtracks)
  - Jan-Harald Fredricksen (ARM)
  - Jakub Lamik (ARM)
  - Ed Plowman (ARM)
  - Rob Simpson (Qualcomm)

• Feedback and Suggestions
  - http://www.khronos.org/bugzilla/