OpenGL BOF
Siggraph 2011
OpenGL BOF Agenda

- **OpenGL 4 update** – Barthold Lichtenbelt, NVIDIA
- **OpenGL Shading Language Hints/Kinks** – Bill Licea-Kane, AMD
- **Ecosystem update** – Jon Leech, Khronos
- **Viewperf 12, a new beginning** – Ian Williams, NVIDIA
- **Brink preferred rendering with OpenGL** – Mikkel Gjoel, Splashdamage

- Trivia throughout
T-shirt and trivia prizes sponsors

- T-shirts and trivia prizes were made possibly through donations from

- **Individuals**
  - John Kessenich
  - Rob Barris
  - Vicki and Dave Shreiner
  - Mike Weiblen
  - Brian Paul
  - Ken Russell

- **Companies**
  - Intel, AMD, NVIDIA, Imagination Technologies, ARM, Out of Control

HUGE THANKS!
OpenGL 4 update

Barthold Lichtenbelt
OpenCL ARB chair
OpenGL Ecosystem – 3D Everywhere

OpenGL ES 2.0 on desktop as subset of OpenGL 4.2 for mobile content flexibility – including native support for WebGL

Leading-edge functionality developed first on desktop

WebGL driving new-generation security features into OpenGL family

Mobile functionality subset that is deployed on billions of devices

Pervasive OpenGL ES 2.0 availability enables Browser vendors to build 3D directly into HTML5
Visual Computing Ecosystem

- High performance compute and graphics interop – buffer and events
- Compute and mobile APIs interoperate through EGL
- JavaScript bindings to OpenCL
- Parallel computation in HTML

• OpenCL
• OpenGL
• OpenGL ES
• WebGL
• WebCL
OpenGL for Each Hardware Generation

1.0
- Fixed Function

2.0
- Vertex and Fragment Shaders

3.0
- Geometry Shaders

4.0
- Tessellation and Compute

'Shape realism'

'Surface realism'
Accelerating OpenGL Innovation

Bringing state-of-the-art functionality to cross-platform graphics

OpenGL 2.0  
2004

OpenGL 2.1  
2005

OpenGL 3.0  
2006

DirectX 9.0c  
2007

DirectX 10.0  
2008

DirectX 10.1  
2009

DirectX 11  
2010

OpenGL 4.1  
2011

OpenGL 3.3/4.0  
2011

OpenGL 3.2  
2011
Press Reactions

Many
Press Reactions

• "With OpenGL 4.2, the Khronos Group has yet again steamed ahead of Microsoft and its slow-and-steady DirectX framework."

http://www.extremetech.com/computing/92266-opengl-4-2-released-drives-directx-further-into-the-ground
OpenGL 4.2 new functionality (1/2)

- **ARB_shader_atomic_counters**
  - GLSL built-in functions to query and increment/decrement atomic counters
  - Counters stored in buffer objects

- **ARB_shader_image_load_store**
  - Allow shaders to load from, store to, and perform atomic read-modify-write operations to a single level of a texture

- **ARB_texture_storage**
  - Immutable textures

- **ARB_transform_feedback_instanced**
  - Draw multiple instances of the result of a transform feedback
  - Without query and sync needed for `DrawArraysInstanced()`

- **ARB_shading_language_420pack**
  - Collection of new GLSL features
OpenGL 4.2 new functionality (2/2)

• **ARB_texture_compression_BPTC**
  - ARB extension since January 2010, promoted to core now
  - Same formats as BC6H/BC7 in DirectX 11

• **ARB_base_instance**
  - Specify offset within buffer object for instanced drawing
  - Adds three new API calls

• **ARB_internalformat_query**
  - Find out supported number of samples per internal format
OpenGL 4.2 performance improvements

- **ARB_compressed_texture_pixel_storage**
  - Modify an arbitrary subset of compressed texture
  - Blizzard requested feature

- **ARB_shading_language_packing**
  - Allows GLSL shaders to read and write 16-bit floating-point encodings
  - Pack small data types into one larger one
  - Implementable on pre GL 4 hardware

- **Uniform sampler initialization, Uniform block binding initialization**
  - Use layout to set texture image unit, and UBO index
  - No need to call GetUniformLocation() or GetUniformBlockLocation()
  - Part of ARB_shading_language_420pack

- **ARB_map_buffer_alignment**
  - Pointer returned is aligned for SSE/AVX CPU instructions

- **ARB_conservative_depth**
  - Shader constructs that help enable depth testing before fragment processing
ARB_texture_storage

- Texture structure is immutable on creation
  - Format
  - Dimensions
  - Presence of mipmap levels

- Contents are mutable
- Always complete (!)

```c
void TexStorage{1D,2D,3D}(enum target, sizei levels,
enum internalformat, sizei width,
sizei height, sizei depth)
```
**Image_load_Store and Atomic Counters**

- **Single rendering pass Order Independent Transparency**
  - Fragment shader writes to layer in 2D image Array using ARB_image_load_store
  - Layer to write to is tracked with a 2D image updated with atomic adds
  - Result: Unordered list of fragments per pixel

- **Sort and compositing pass**
  - Full screen quad
  - Copy all fragments into an array
  - Bubble sort array
  - Combine front-to-back

- **Can be 1.5 – 2.0x faster**
  - Over k-buffer
  - Over bucket sort depth peeling

Transform Feedback Instanced example

• **Need**: Draw same geometry N time
  - Don’t waste tessellation GPU cycles

• **Solution**: Tessellate geometry only when LOD changes
  - Using tessellation added in OpenGL 4.0

• **Capture output with transform feedback**
  - xfb objects capture primitive counts

• **Render N times with**
  - `DrawTransformFeedbackInstanced()`
  - New in OpenGL 4.2

Image courtesy of http://bimuzer.com/
Compressed Pixel Texture Storage

Compressed texture

Application updates a subregion

Application memory

GPU memory

Compressed texture
That needs updating
Compressed Pixel Texture Storage

Application memory ➔ GPU memory

App has to update entire texture with `glCompressedTexImage2D`

App copies sub-region to another Memory location and use `glCompressedTexSubImage2D`

Most efficient!

With `compressed_texture_pixel_storage` data can get copied directly to GPU memory

© Copyright Khronos Group, 2010 - Page 18
ARB_shading_language_420pack

- **Initializer lists**
  - adds initializer lists as a method of initializing arrays and structures in GLSL

- **.length() and scalar swizzle**
  - adds support for the .length() method on all vector and matrix types
  - adds the ability to perform vector swizzle operations on scalar values

- **qualifiers**
  - `const` can now declare function variables with non-constant initializers
  - qualifiers can be in any order

- **“binding” layout qualifiers**
  - Use `layout` to set texture image unit and UBO index

- **Others**
  - Any character in a comment
  - `\` as a continuation character
Shading Language Packing

• Pack and convert
  - vec2 representing two FP16 values into a uint
  - vec2 and vec4 vectors to 8 and 16 bit int into a uint
  - And vice-versa

• Benefits
  - Can be implemented on pre GL4 hardware
  - Saves attributes to pass data among shaders
  - Saves transform-feedback bandwidth
  - Therefore performance increase due to less mem BW and storage used

```cpp
uint packUnorm2x16(vec2 v);
uint packSnorm2x16(vec2 v);
uint packUnorm4x8(vec4 v);
uint packSnorm4x8(vec4 v);
vec2 unpackUnorm2x16(uint v);
vec2 unpackSnorm2x16(uint v);
vec4 unpackUnorm4x8(uint v);
vec4 unpackSnorm4x8(uint v);
vec2 packHalf2x16(vec2 v);
vec2 unpackHalf2x16(uint v);
```