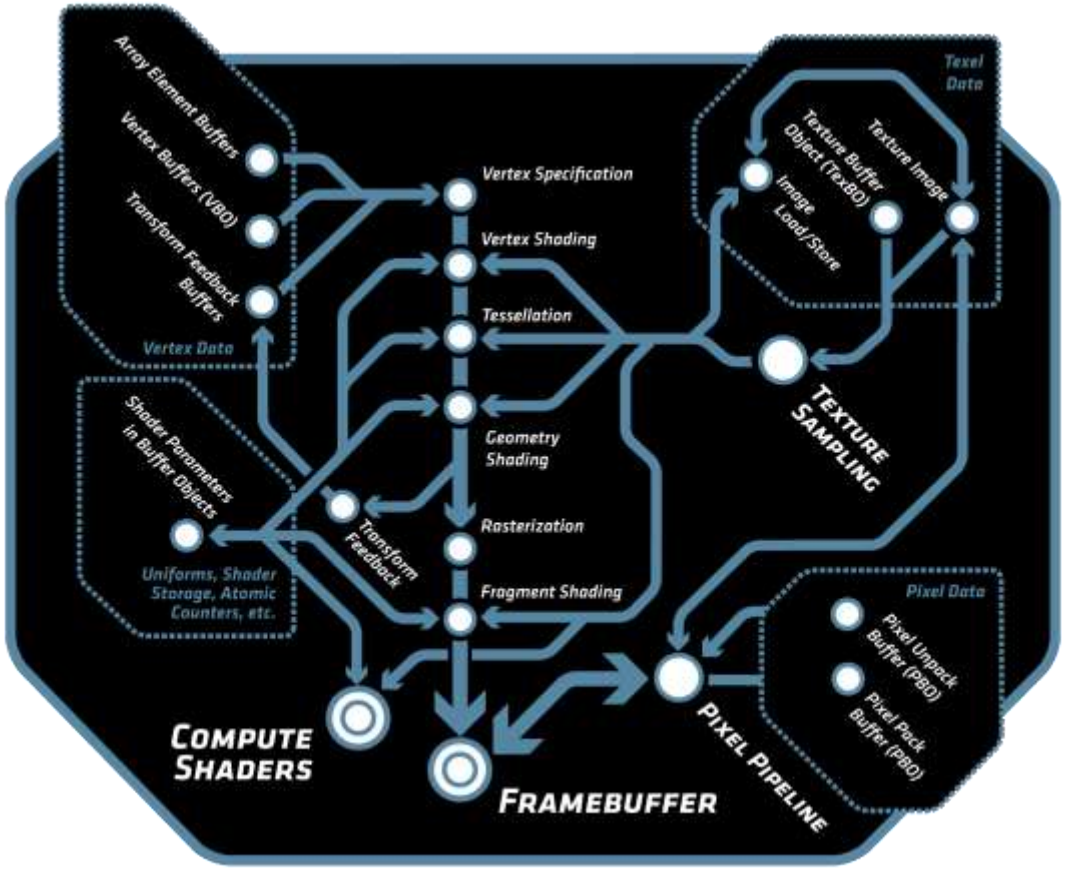


KHRONOS GROUP

OpenGL

BOF

Siggraph 2013

Barthold Lichtenbelt
OpenGL ARB chair

KHRONOS GROUP

Announcing OpenGL 4.4

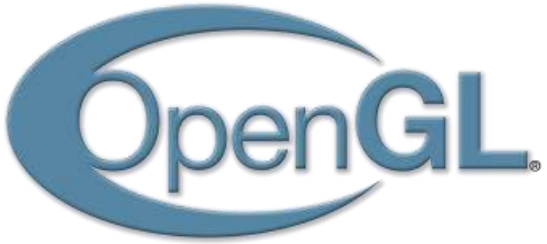
OpenGL BOF Agenda

- **OpenGL 4.4, news and updates**
  - Barthold Lichtenbelt, NVIDIA

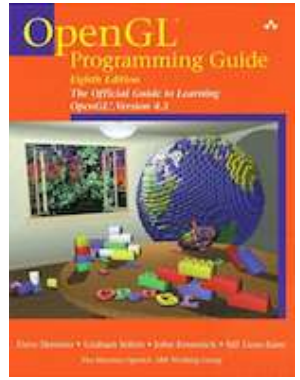
- **nvFX: A New Shader-Effect Framework for OpenGL and OpenGL Compute and other APIs**
  - Tristan Lorach – Senior Devtech Engineer – NVIDIA

- **From DirectX to OpenGL in 2013**
  - Karl Hillesland – demo/research team – AMD

- **OpenGL Batching for the masses**
  - Samuel Gateau – Senior Devtech Engineer - NVIDIA

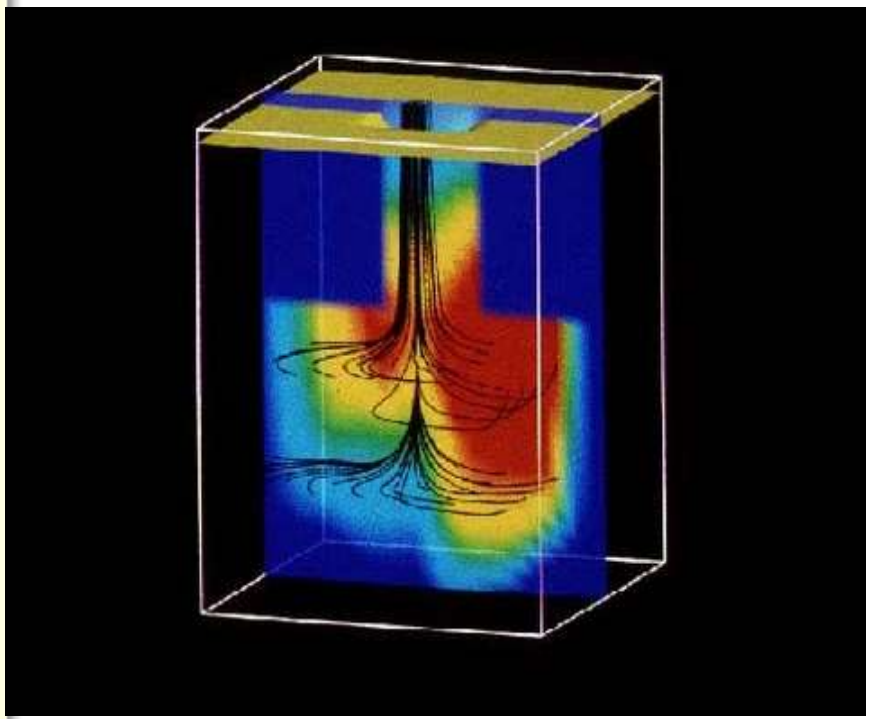


OpenGL Ecosystem News



- **Books**
  - New **Red Book** (OpenGL Programming Guide) - March 2013
  - New **OpenGL Super Bible** – today!
- **Tools and projects**
  - **GLEW** - updated to OpenGL 4.4 today!
  - **Regal** - updated to 4.3
  - **GLintercept** - updated to 4.3
  - **Apitrace** – http://apitrace.github.io/
  - **G-truc** - http://www.g-truc.net/
  - **GLSL Debugger** source released - https://github.com/XenonofArcticus/GLSL-Debugger
  - **http://open.gl** - OpenGL tutorial using WebGL to illustrate its content
  - **KTX Library** updated to 2.0 - http://www.khronos.org/opengles/sdk/tools/KTX

Coming soon – SPECviewperf 12



- SUPPORTS OpenGL and that other API
- MAKES IT SIMPLE to add new viewsets
- PROVIDES portability to OSs beyond Windows
- ENABLES dynamic viewsets
- DELIVERS all-new updated viewsets

XML API Registry - http://www.opengl.org/registry

- **Replaces old SGI ".spec" files with a structured XML format**
  - Covers GL, GLES 1/2/3, EGL, GLX, and WGL APIs
- **Includes Relax NG schema for validation**
- **Python scripts to transform the registry into other forms**
  - including traditional glext.h / glcorearb.h headers
- **All available from Khronos public Subversion server**
- **Much easier to maintain than .spec files were**
  - We've been able to close almost all public .spec / header bugs as a result
- **Better represents relationships between commands/enums/types, extensions, profiles, and versions**
- **Use this for generating language bindings, wrapper libraries, etc**
  - Submit patches, instead of parsing extension text specifications

KHRONOS GROUP

OpenGL 4.4 reference pages

Huge thanks to Graham Sellers!!!

OpenGL Conformance Test Suite released!

- **Conformance submissions are required for GL 4.4 implementations**
  - encouraged for earlier driver versions
- **Shared codebase with OpenGL ES 3.0 CTS**
  - additional desktop-specific tests
- **Core profile functionality**
- **Enhancements underway to add more coverage**

KHRONOS GROUP

OpenGL

PRESS

- **Over 180 articles world-wide!**

"... DirectX appears to be standing still."

- **http://www.brightsideofnews.com/news/2013/7/23/khronos-moves-foward-with-opencl-20-and-opengl-44.aspx**

"Finally, in a move that should have developers everywhere jumping with joy, OpenGL finally has official and up to date conformance tests. "

- **http://www.anandtech.com/show/7161/khronos-siggraph-2013-opengl-44-opencl-20-opencl-12-spir-announced**

KHRONOS GROUP

What is new in OpenGL 4.4?

- **ARB_buffer_storage**
  - Immutable storage for buffer objects
  - Explicit control over buffer placement; vidmem vs sysmem and cache behavior
  - Allows a mapped buffer to be used by the GPU
- **ARB_enhanced_layouts (GLSL)**
  - Allows compile-time constants in qualifiers
  - More control for placing shader interface variables
  - Pack vectors more efficiently with scalar types
  - More control of variable layout inside uniform blocks and shader storage blocks
  - In shader control of transform feedback variables.
- **ARB_query_buffer_object**
  - Allows a buffer object to be target of a query
  - Avoids CPU getting involved, no pipeline stall

What is new in OpenGL 4.4?

- **ARB_clear_texture**
  - Clear texture values to a specific value
- **ARB_texture_mirror_clamp_to_edge**
  - allows the texture to be mirrored in the negative s, t, and r directions.
- **ARB_texture_stencil8**
  - Create and sample stencil only textures
- **ARB_vertex_type_10f_11f_11f_rev**
  - Packs 3 components into 32 bit value
- **ARB_multi_bind**
  - One call to perform multiple bindings
  - Reduces driver CPU overhead

New ARB only extensions

- **ARB_bindless_texture**
  - Allow referencing textures by handle in a shader
- **ARB_sparse_texture**
  - Support texture sizes beyond physical memory
  - Choose which parts of a texture are resident
- **ARB_seamless_cubemap_per_texture**
  - Control the "seamless" switch for cubemaps per texture
- **ARB_indirect_parameters**
  - "count" parameter of a multi-draw-indirect call can now come from a buffer object

New ARB only extensions

- **ARB_compute_variable_group_size**
  - Allow compute shader dispatch to set size of the workgroup
- **ARB_shader_draw_parameters**
  - `gl_BaseInstance`, `gl_BaseVertex` and `gl_DrawID` as new GLSL builtins
- **ARB_shader_group_vote**
  - compute the composite of a set of boolean conditions across a group of shader invocations

ARB_Buffer_Storage

- **Immutable storage for buffer objects**

```
void BufferStorage(enum target,
                   sizeiptr size,
                   const void * data,
                   bitfield flags);
```

```
DYNAMIC_STORAGE_BIT - If not set, allocation will be GPU accessible
MAP_READ/WRITE_BIT  - Controls CPU caching policies
MAP_COHERENT_BIT    - Shared access by client and server will be coherent (*)
MAP_PERSISTENT_BIT  - Can use buffer while mapped
CLIENT_STORAGE_BIT  - This is a hint. Memory location will favor client access
```

```
If you access a buffer without the right bit set, Bad Things will happen.
```

```
(*) but read spec carefully!
```

Enhanced Layouts in GLSL

- **Shader based Transform Feedback Layout**
  - Specify buffers, strides, offsets
  - No TransformFeedbackVaryings() command needed

```
layout (xfb_buffer = 0, xfb_stride = 32) out b {
    layout (xfb_offset = 0)  vec2 a;  // a goes to byte offset 0 of buffer 0
                             vec4 b;  // b is not captured, no xfb_offset
    layout (xfb_offset = 16) vec4 c;  // c goes to offset 16 of buffer 0
};                                    // there is a hole at bytes 8 through 15
```

- **Compile-Time constants, in any integer layout**

```
const int start = 6;
    layout(location = start + 2) int vec4 v; // Sets location to 8
```

Enhanced Layouts in GLSL

- **Explicit byte-offset layout of uniform blocks**

```
uniform layout(std140) Block {
        layout(offset = 0) vec4 batman;       // gets byte offset 0
        layout(offset = 64) vec4 robin;       // gets byte offset 64
    };
```

- **Locations on Input and Output blocks**

```
 layout(location = 4) in block {
         vec4 batman;                          // gets location 4
         vec4 robin;                           // gets location 5
         layout(location = 7) vec4 joker;      // gets location 7
         vec4 riddler;                         // gets location 8
     };
```

Enhanced Layouts in GLSL

- **Component-level slot utilization**

Old way

```
// consume 5 slots
in vec3 batman[4];
in float robin;
```

New way

```
// consume X/Y/Z components of 4 slots
layout(location = 0, component = 0) in vec3 batman[4];

// consumes W component of first slot
layout(location = 0, component = 3) in float robin;
```

Bindless Textures

- **Problem statement**
  - Binding to different texture objects takes validation time in driver
  - Applications are limited to small palette of bound textures

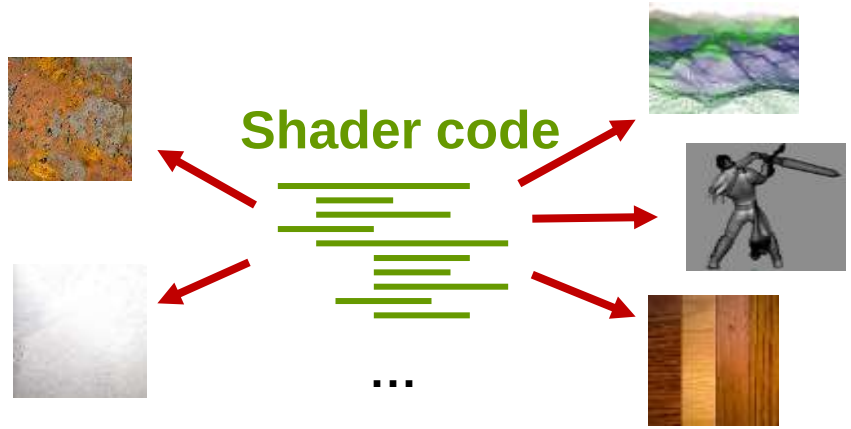
- **Traditional OpenGL**
  - GPU memory reads are "indirected" through bindings
  - Limited number of texture units

- **Solution : Exposes textures as handles**
  - Let shaders access textures directly

Bindless Textures

- Increase number of unique textures available to shaders at run-time
- More different materials and richer texture detail in a scene

Existing texture binding model

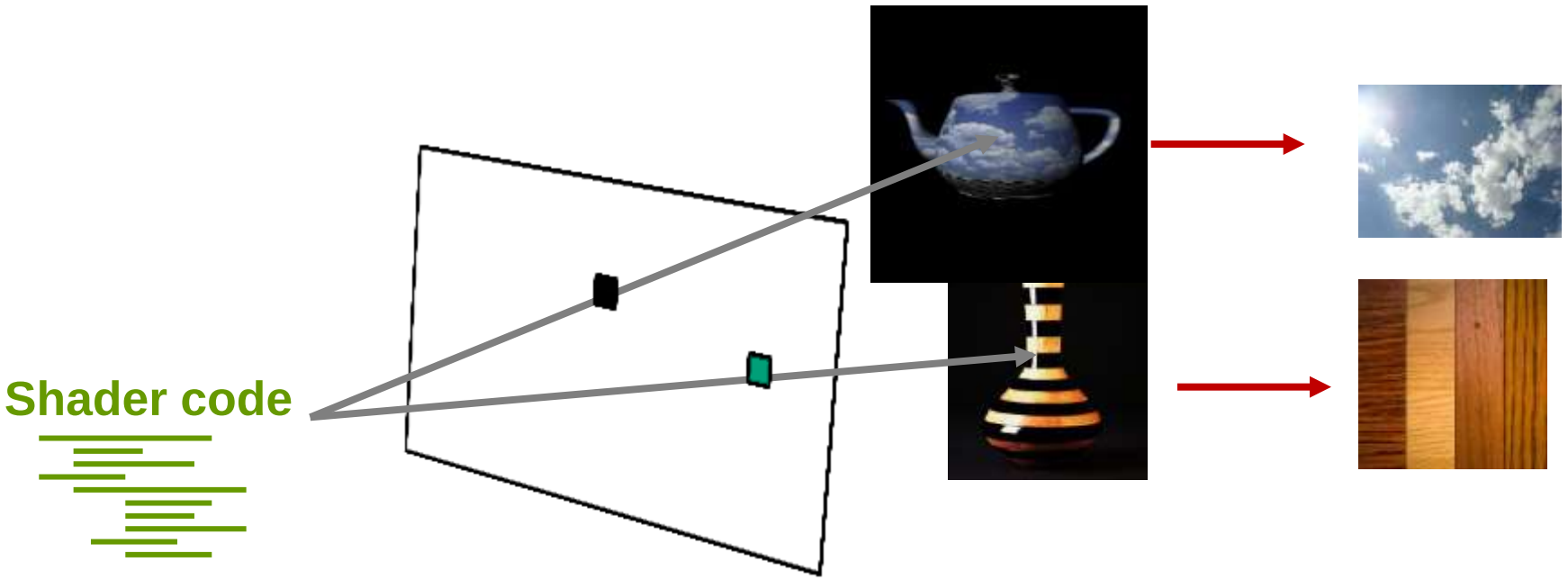


bindless textures

over 1 million unique textures

Bindless Textures

- **Apropos for ray-tracing and advanced rendering where textures cannot be "bound" in advance**



Bindless Textures

Existing texture binding model

CPU

Load texture A
Load texture B
Load texture C
Bind texture A to slot I
Bind texture B to slot J
Draw()

GPU

Read from texture at slot I
Read from texture at slot J

CPU

Bind texture C to slot K
Draw()

GPU

Read from texture at slot K

bindless textures

CPU

Load textures A, B, C
Draw()

GPU

Read from texture A
Read from texture B
Read from texture C

Bindless model reduces CPU overhead and improves GPU access efficiency