ANGLE and Cross-Platform WebGL Support

Shannon Woods
shannonwoods@google.com

ANGLE's users

WebGL Support

Chrome, Firefox

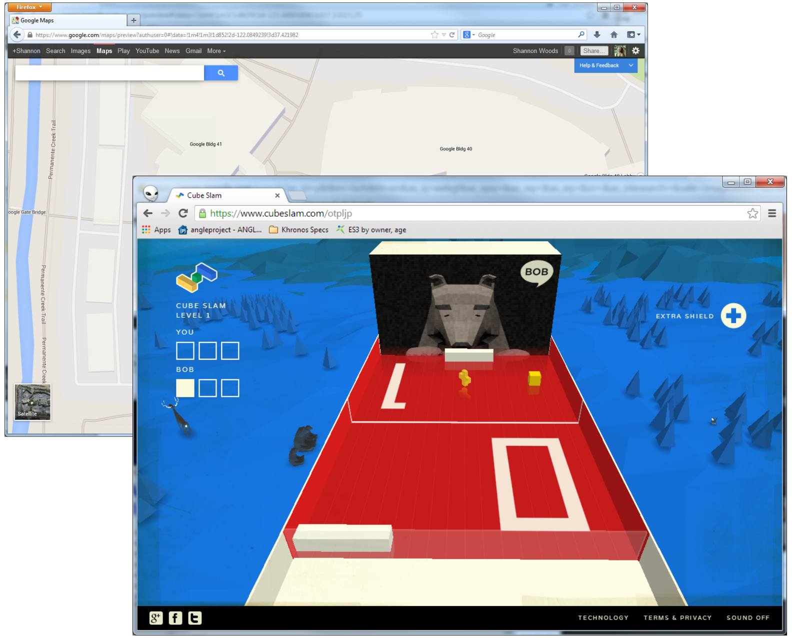
Shading Language Validation

Chrome, Firefox, Safari

Windows, MacOS, Linux, mobile

Portability

Qt, other projects



ANGLE: OpenGL ES via Direct3D

Why is ANGLE needed?

- WebGL's potential global userbase
- OpenGL ES drivers: Present? Conformant? Robust?
- Direct3D installed with Windows

What does ANGLE provide?

- OpenGL ES 2.0 over Direct3D 9, certified Nov. 2011
- OpenGL ES 2.0 over Direct3D 11, completed
- OpenGL ES 3.0 over Direct3D 11, in progress

Special Considerations for Performance

ANGLE's best practices not always the same as native drivers'

D3D 9 and D3D 11

- Use GL_LINE_STRIP or GL_LINES instead of GL_LINE_LOOP
- Use immutable textures when possible; if not, create new textures rather than redefining old
- Perform clears full-screen and unmasked
- Use GL_UNSIGNED_SHORT or GL_UNSIGNED_INT indices, not GL_UNSIGNED_BYTE
- Prefer RGBA8 to RGB8, RGBA16F to RGB16F in vertex buffers

Special Considerations for Performance

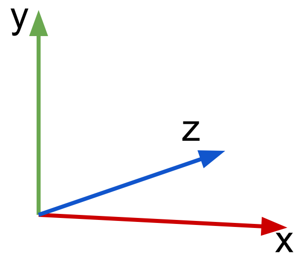
D3D9 Only

- Use static buffers when possible, flag buffers correctly
- Use BGRA_EXT/UNSIGNED_BYTE texture format

D3D11 Only

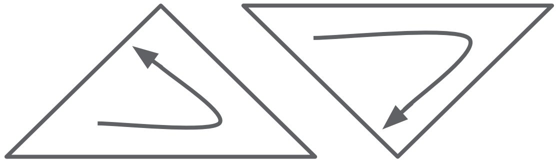
- Use GL_TRIANGLE_STRIP or GL_TRIANGLES instead of GL_TRIANGLE_FAN
- Use GL_RED for single-channel textures instead of GL_LUMINANCE

Challenges of the API Language Barrier



Differing coordinate systems

- OpenGL and D3D do not differ in handedness.
- D3D window origin is top left, y inverted during viewport transform
- ANGLE compensates with y-flip in vertex shader & on present
  - Handles render-to-screen/render-to-texture y-axis difference
  - Causes winding order to be reversed



Other considerations:

- D3D 9 pixel centers at integral locations

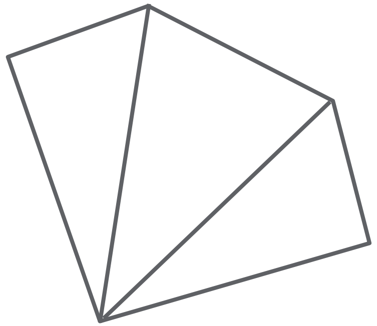
Challenges of the API Language Barrier

Vertex & Index Buffers

- D3D 9: Vertex vs. index declared at creation - Not so for OpenGL
- D3D 9: Supports fewer data types in buffers than OpenGL
- D3D 11: Increases type support, same buffers for vertex/index
  - Unnormalized integer data still requires conversion

Primitive Types

- D3D 9: does not have line loops
- D3D 11: eliminates triangle fans, large points



Challenges of the API Language Barrier

Textures

- D3D 9 has limited format support; textures converted on load
  - D3D 11 increases format support
- D3D 9 & D3D 11 require dimensions & format known at creation
  - GL has concept of completeness
  - GL’s immutable textures more like D3D, may be less overhead even in native drivers

Challenges of the API Language Barrier

Framebuffers

- Masked clear implemented via draw, resets state
    - Performance becomes more critical with EXT_draw_buffers
- Blit provided by ANGLE-specific extensions
    - Limitations due to D3D 9’s StretchRect
    - D3D 11 less limited, but extension remains the same
- Depth & stencil buffers are not separable in Direct3D
    - ANGLE will return GL_FRAMEBUFFER_UNSUPPORTED for separately defined depth & stencil buffers.
    - OES_packed_depth_stencil provides unified buffer

Challenges of the API Language Barrier

Shading Language

- Row-major vs. column-major matrices - no noticeable performance impact
- Complex flow control & large loops
  - Improved on Direct3D 10 and later compilers
- Short circuiting behavior - GL short-circuits, D3D evaluates both

```
if (a = func1() || a = func2())
```

- Ternary selection operator evaluation - again, D3D evaluates both

```
(condition ? func1() : func2())
```

The Future of ANGLE: OpenGL ES 3.0

OpenGL ES 3.0 implementation currently in progress
Supported only via Direct3D 11 backend
New support for:

- Integer, half float, packed vertex attributes
- Integer, sRGB, additional packed texture formats
- Expanded multisample & blit support
- VAOs, PBOs, UBOs, sampler objects
- Transform Feedback
- glMapBuffer
- Into core: instancing, sync support, query objects, multiple draw buffers

OpenGL ES 3.0: New API, New Caveats

Integer cube map sampling

- Will be emulated in the shader with 2D texture array

Texture Swizzle

- No D3D equivalent capability

Pixel Buffer Objects

- Some texture formats don’t have equivalent buffer SRV format
  - e.g.: RGB16, RGB8, RGBA4, RGB5A1, sRGB formats for unpack buffers

For More Information

ANGLE project

Home: http://code.google.com/p/angleproject/

Discussions: http://groups.google.com/d/forum/angleproject

Questions?