Moving to OpenGL

Jason Mitchell
Dan Ginsburg
Rich Geldreich
Peter Lohrmann

Outline

- OpenGL Strategy - Jason
- Shipping shaders - Dan
- New debugging tools – Rich & Peter

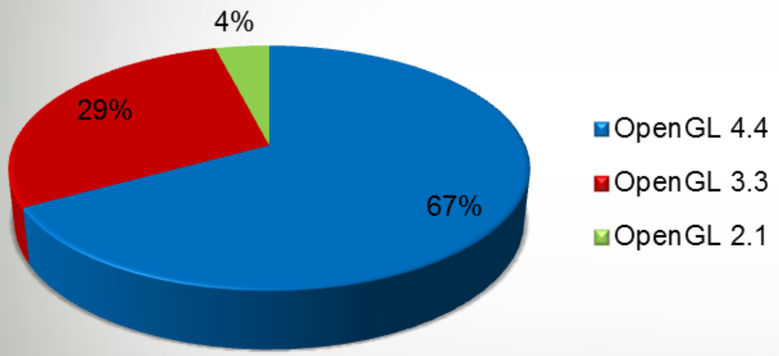
You are going to use OpenGL

OpenGL is Everywhere

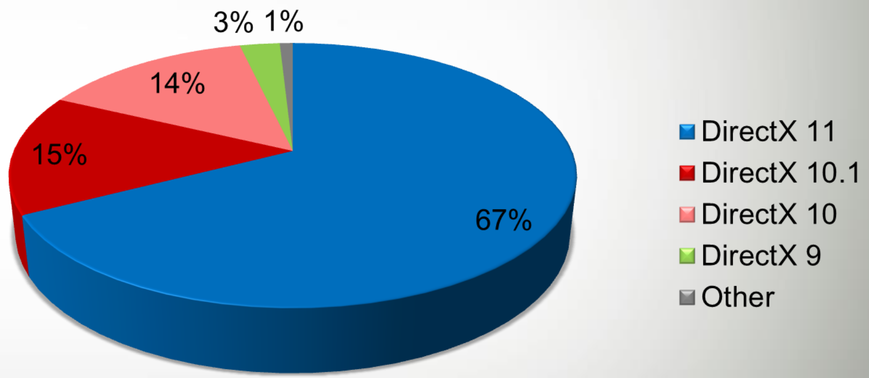
- **SteamOS**
- **Desktop** Linux, OS X & Windows
    - China overwhelmingly XP but fairly modern hardware
- **Mobile** OpenGL ES is ubiquitous
    - Even “Big OpenGL” arriving
- **WebGL**

Steam Graphics Hardware

OpenGL



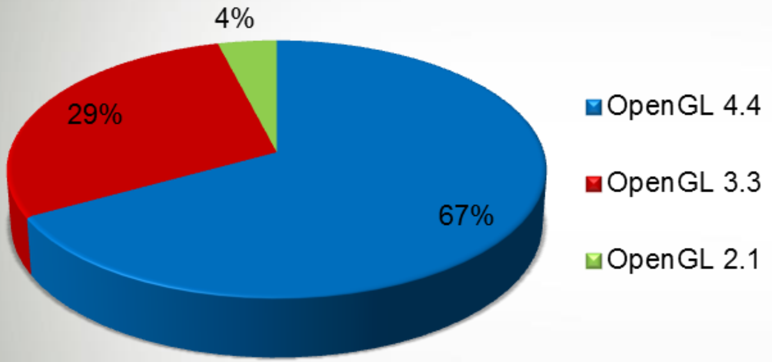
Direct3D



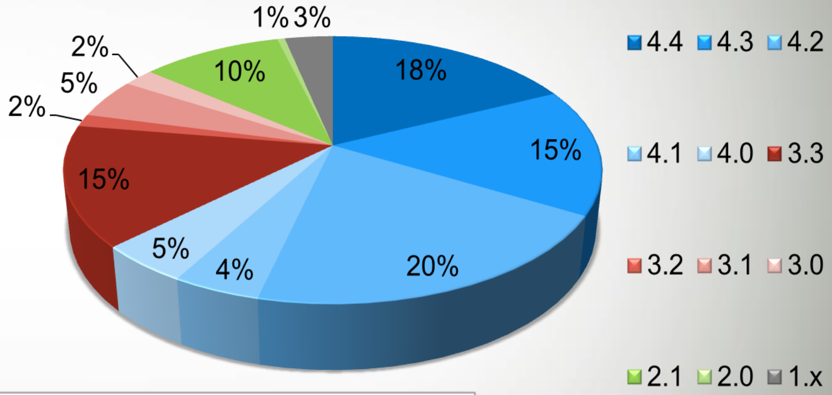
Steam Hardware Survey, Dec 2013

Steam OpenGL Drivers

Hardware Capability



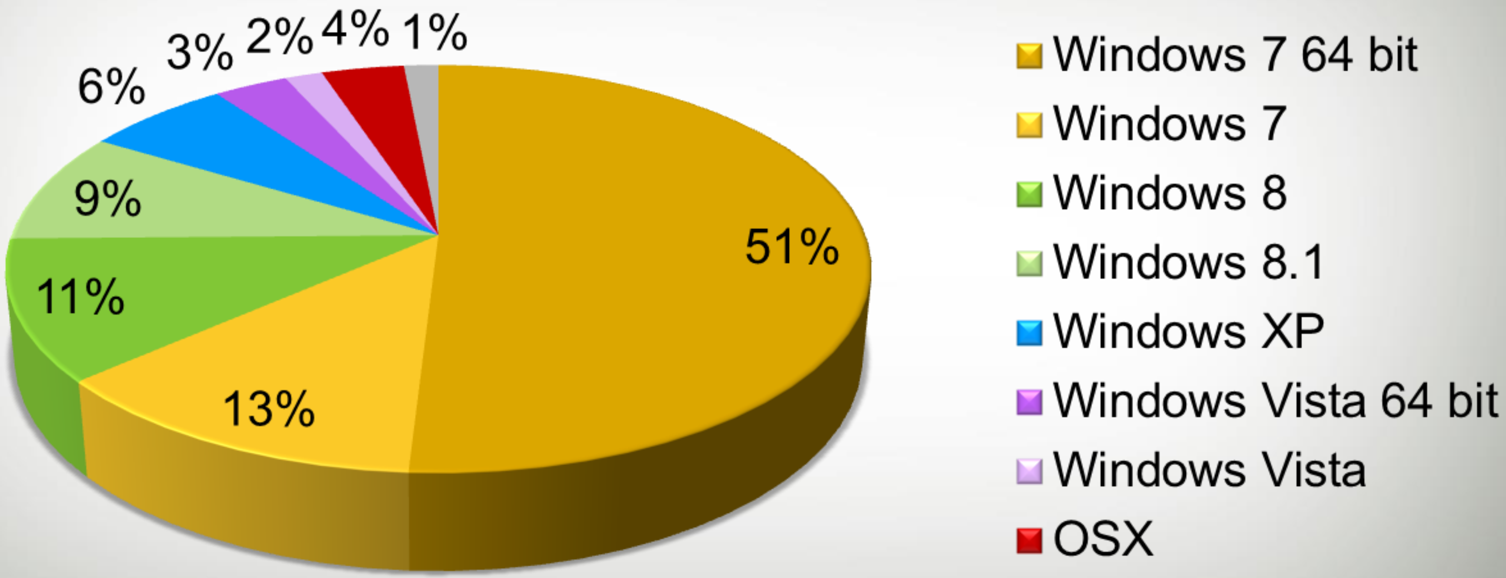
Installed Drivers



- Over time, we want the chart on the right to look more like the chart on the left
- Some challenges:
    - Apple currently on 4.1
    - Vendors have varying XP support

Steam Hardware Survey, Dec 2013

Steam Operating Systems



Steam Hardware Survey, Dec 2013

DirectX and Total Available Market

	GPUs	**Systems** **(Windows Vista, 7, 8)**
DirectX 11	67% →	62%
DirectX 10.x	96% →	86%
DirectX 9	100%	100%

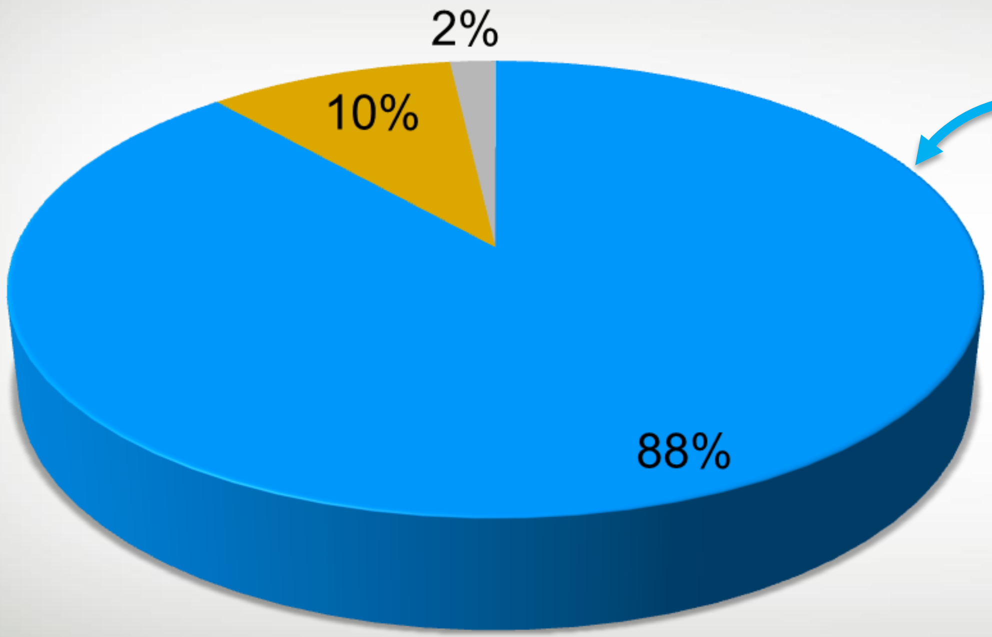
OpenGL and Total Available Market

	GPUs	Systems
OpenGL 4.x	67% →	67%
OpenGL 3.3	96% →	96%
OpenGL 2.1	100%	100%

Emerging Markets

- Valve is expanding beyond its traditional borders
- The most recent example is Dota in China
- Windows XP is extremely prevalent in China

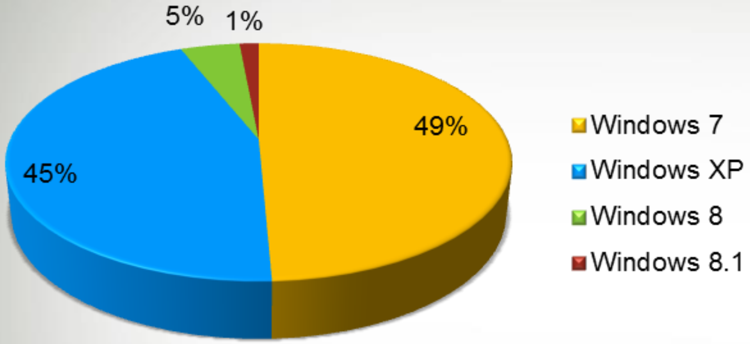
Chinese Cyber Cafe OS Versions

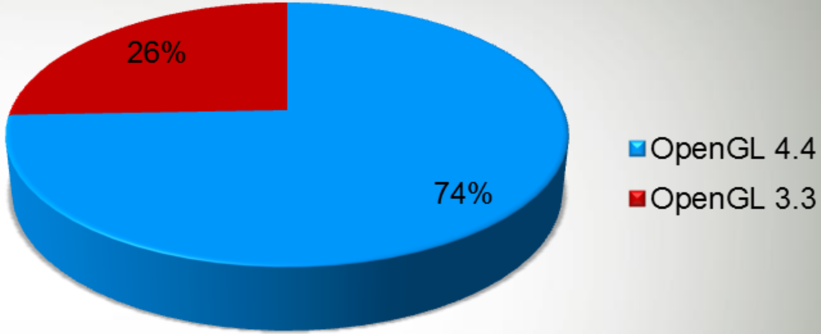


No DirectX10 or DirectX11 games for these customers

Data from the Yi You cyber cafe platform

Dota Users in China





- Windows XP very popular
  - We think this is a lower bound on XP in China
- Hardware is modern!
- Use OpenGL to access that hardware!

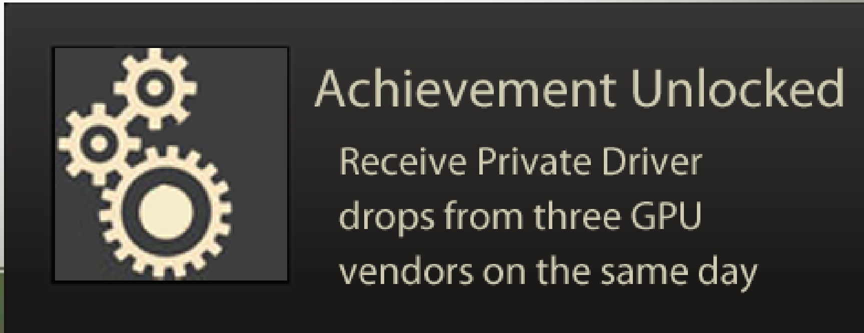
Dota users in China
January 2014

OpenGL Strategy

- Source 2 has multiple rendering backends
- OpenGL backend is a peer to others
- Currently Direct3D-centric
    - HLSL gets translated to GLSL
    - Separate Shader Objects etc
- Would like to drop the Direct3D backends and go OpenGL-exclusive

Working Closely With Desktop Vendors

- AMD
- NVIDIA
- Intel – Two separate teams!
  - Binary drivers on Windows
  - Open Source drivers on Linux
- Apple



Achievement Unlocked

Receive Private Driver drops from three GPU vendors on the same day

Our biggest near term challenges

- Shipping Shaders
  - Validation
  - Efficient shipping representation
- Graphics Debugging
  - Vendor tools are improving, especially NSIGHT
- Capturing repro scenarios
  - apitrace – Open source tool developed externally
  - VOGL – New open source tools from Valve

Dan

Rich & Peter

Overview

Shipping Shaders

- Translation
- Validation
- Shipping Representation

Overview

Shipping Shaders

- **Translation**
- Validation
- Shipping Representation

HLSL -> GLSL

Source 1:

- DX9ASM -> GLSL

Works, but some downsides:

- Debugging hard
- Loss of information
- Not extensible

HLSL -> GLSL

Source 2:

- Translate at the source level

Reasoning:

- Easier to debug
- Easier to use GLSL features
- D3D10/11 bytecode not as well documented as DX9

Translation Options

hlsl2glslfork

- Not DX10/11-compatible

MojoShader

- Shader Model 3.0 only

HLSLCrossCompiler, fxdis-d3d1x

- DX10/11 ASM

Translation Approach

Valve already had ANTLR-based HLSL parser:

- Used to extract semantics, constant buffers, annotations
- Only minimally understands HLSL, accepts everything inside of “{” “}”

Translation Approach

Use macros for HLSL/GLSL differences

Write GLSL-compatible HLSL

Extend our ANTLR-based parser:

- Strip HLSL-specific constructs
- Generate GLSL-specific constructs
- Generate GLSL main() wrapper

Zero run-time shader reflection

HLSL-> GLSL Wrappers

Macros for common types:

- `#define float4 vec4`

Macros for texture definitions and access:

- `#define CreateTexture2D( name ) uniform sampler2D name`
- `#define Tex2D( name, uv ) texture( name, ( uv ).xy )`

Wrappers for missing built-in functions:

- `float saturate( float f ) { return clamp( f, 0.0, 1.0 ); }`

HLSL -> GLSL Semantics

```
struct VS_INPUT {
   float3 vPositionOs : POSITION    ;
   float4 vNormalOs   : NORMAL      ;
   float2 vUv0        : TEXCOORD0   ;
};

struct PS_INPUT {
   float4 vOutPos    : SV_Position ;
   float3 vNormalWs  : TEXCOORD1   ;
   float2 vUv0       : TEXCOORD0   ;
};

layout(location = 0) in float3   VS_INPUT_gl_vPositionOs;
layout(location = 1) in float4   VS_INPUT_gl_vNormalOs;
layout(location = 2) in float2   VS_INPUT_gl_vUv0;

layout(location = 0) out float3 PS_INPUT_gl_vNormalWs;
layout(location = 1) out float2 PS_INPUT_gl_vUv0;
```

HLSL ->GLSL main() wrapper

```
void main() {
  VS_INPUT mainIn;
  PS_INPUT mainOut;

  mainIn.vPositionOs = VS_INPUT_gl_vPositionOs;
  mainIn.vNormalOs = VS_INPUT_gl_vNormalOs;
  mainIn.vUv0 = VS_INPUT_gl_vUv0;

  mainOut = MainVs( mainIn );

  gl_Position = mainOut.vOutPos;
  PS_INPUT_gl_vNormalWs = mainOut.vNormalWs;
  PS_INPUT_gl_vUv0 = mainOut.vUv0;
}
```

GLSL-Compatible HLSL

No implicit conversions:

- o.vColor.rgb = 1.0 - flRoughness; // BAD
- o.vColor.rgb = float3(1.0, 1.0, 1.0) - flRoughness.xxx; // GOOD

No C-style casts:

- int nLoopCount = (int) FILTER_N
- int nLoopCount = int (FILTER_NUMTAPS); // GOOD

ARB_shading_language_420pack

No non-boolean conditionals:

- #define S_NORMAL_MAP 1
- if (S_NORMAL_MAP) // BAD
- if (S_NORMAL_MAP != 0) // GOOD

No static local variables

Further GLSL Compatibility

- Use std140 uniform buffers to match D3D
- Use ARB_separate_shader_objects
- Use ARB_shading_language_420pack

Shader Reparser

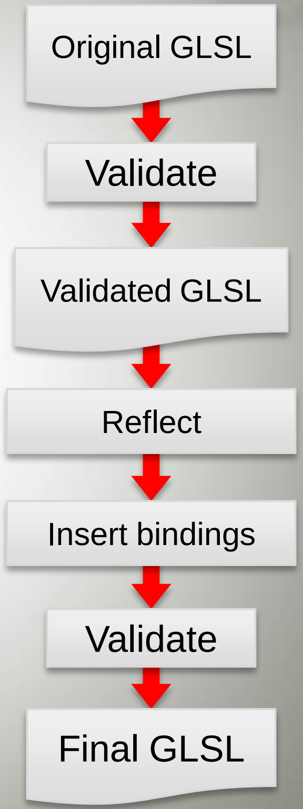
Zero run-time shader reflection

Set uniform block bindings:

```
layout( std140, row_major) uniform PerViewConstantBuffer_t
                          , binding=0
{
        float4x4 g_matWorldToProjection ;
        // …
};
```

Set sampler bindings:

```
uniform sampler2D g_tColor;
layout(binding=2)
```



Overview

Shipping Shaders

- Translation
- **Validation**
- Shipping Representation

Shader Validation

- Problem: how to determine GLSL is valid?
    - D3D has D3DX-like offline tool
    - Every OpenGL driver has a different compiler
    - Compilation tied to HW/driver in system

Reference Compilers Considered

- Compile on all GL drivers
  - Considered this option seriously, very painful
- cgc (NVIDIA)
  - End-of-life
- Mesa (used by glsl-optimizer project)
  - Good option, but was missing features we needed

OpenGL Community Problem

- Realized we should not solve this problem ourselves
- OpenGL needs a reference compiler
- Discussed with other ISVs and Khronos
- Khronos came through:
    - glslang selected as reference compiler

glslang Introduction

- Open source
- C and C++ API
- Command-line tool
- Linux/Windows

Valve-funded glslang Enhancements

- Extend GLSL feature support
    - GLSL v4.20
    - Shader Model 4/5 (GS/TCS/TES)
    - ARB_shading_language_420pack
    - ARB_gpu_shader5 (partial)
- Reflection API
    - Active uniforms, uniform buffers

How We Use glslang

- Every shader validated/reflected with glslang
- Used for distributed compilation
- Found many issues in our shaders we would not have found until testing:
    - AMD/NV/Intel accepting invalid GLSL
    - AMD/NV/Intel not accepting correct GLSL
    - Led us to file bugs against IHV’s

glslang

Where to get it:

http://www.khronos.org/opengles/sdk/tools/Reference-Compiler/

Overview

Shipping Shaders

- Translation
- Validation
- **Shipping Representation**

Shipping Shaders

Current options:

- GLSL source
- Program binaries (ARB_get_program_binary)

GLSL Source

Issues:

- Slow shader compiles compared to D3D bytecode
    - However, subsequent compiles are comparable to D3D if driver has a shader cache
- IP Leakage

Program Binaries

Issues:

- Extremely fragile to driver/HW changes
- Still requires GLSL to be available (at least at install time)

Shader Compilation Performance

	GLSL	Optimized GLSL (cgc)
Driver A	763 ms	132 ms
Driver B	229 ms	111 ms
Driver A Shader Cache	16 ms	14 ms

Intermediate Representation (IR)

Solves many problems at once:

- Faster compile times (comparable to D3D IL)
- No IP leakage
- Single reference compiler

Active area of work:

- OpenCL SPIR 1.2 exists
- Valve advocating for IR in Khronos

Summary

- Translation
- Validation
- Shipping Representation

VOGL
OpenGL Tracing and Debugging

Rich Geldreich, Peter Lohrmann

Why a New Debugger?

- The OpenGL debugging situation is, well, almost nonexistent (but improving).
- We’ve spent *a lot* of time debugging GL/D3D apps.
- We’ve been let down by the available debugging tools.

VOGL High Level Goals

- Open Source
- Steam Integration
- Vendor / Driver version neutral
- No special app builds needed
- Frame capturing, full stream tracing, trace trimming
- Optimized replayer
- OpenGL usage validation
- Regression testing, benchmarking
- Robust API support: GL v3/4.x, core or compatibility contexts
- UI to edit captures, inspect state, diff snapshots, control tracing

Key Concepts

- **Trace File** (Binary or JSON)
  - Binary trace: Header, GL trace packets, zip64 archive at end
  - JSON trace: 1 JSON file per frame + loose files or .zip archive
  - Archive contains: state snapshot, frame directory (offsets, JPEG’s), backtrace map, etc.
- **State Snapshot**
  - Restorable object containing all GL state: contexts, buffers, shaders, programs, etc.
  - Serialized as JSON+loose files, JSON diff’able using common tools

Key Concepts

- **Full-Stream Trace**
  - Contains **all** GL calls made by app
  - Optional: State snapshot keyframes for fast seeking
- **Single/Multi-Frame Trace**
  - State snapshot followed by next X frame(s) of GL calls
- **Trimming**
  - Take 2+ frame trace and extract 1+ frame(s) into a new trace file

Demos

- Driver/GPU torture test
- DVR-style replay mode
- vogleditor

Current App Compatibility

- Valve:
  - All GoldSrc engine titles: Half-Life, Counterstrike, TFC, etc.
  - All Source engine titles: Portal, DotA 2,TF2, L4D2, Half-Life 2, etc.
  - Steam: 2ft UI, Steam Overlay
- 3rd-party:
  - 10,000,000, Air Conflicts: Pacific Carriers, BIT.TRIP Runner2, Bastion, Brutal Legend, Cubemen 2, Darwinia, Dynamite Jack, Extreme TuxRacer, Galcon Fusion, Metro Last Light, Multiwinia, Natural Selection 2, No More Room in Hell, Not the Robots!!!, Oil Rush, Overgrowth, Penumbra (series), Postal 2 (Unreal Engine), Serious Sam 3, Solar 2, Starbound, Steel Storm, Strike Suit Zero, The 39 Steps, The Cave, Trine 2, Wargame: European Escalation, World of Goo, X3 (series)
- Various samples/test suites: OpenGL SuperBible 3rd and 4th editions, G-Truc GL 3.x samples
- Still working on:
  - Remaining Steam Linux titles
  - Piglit driver testing framework
  - G-Truc 4.x Samples
  - SuperBible 5th/6th edition samples

Common GL Issues We've Seen

- Incomplete textures (not setting **GL_TEXTURE_MAX_LEVEL)**
- Calling GL without an active context, unintentional leaks
- Bogus handles
- FBO completeness
- Shipping with GL errors – sometimes many per-frame
- Debug context warnings
- Perf: Not using trivial DSA (Direct State Access) equivalents
- Perf: Redundant state setting
- Odd patterns: glBindAttribLocation() called **after** linking the program (and never linking the program again), or calling glIsTexture() repeatedly vs. glGen'ing

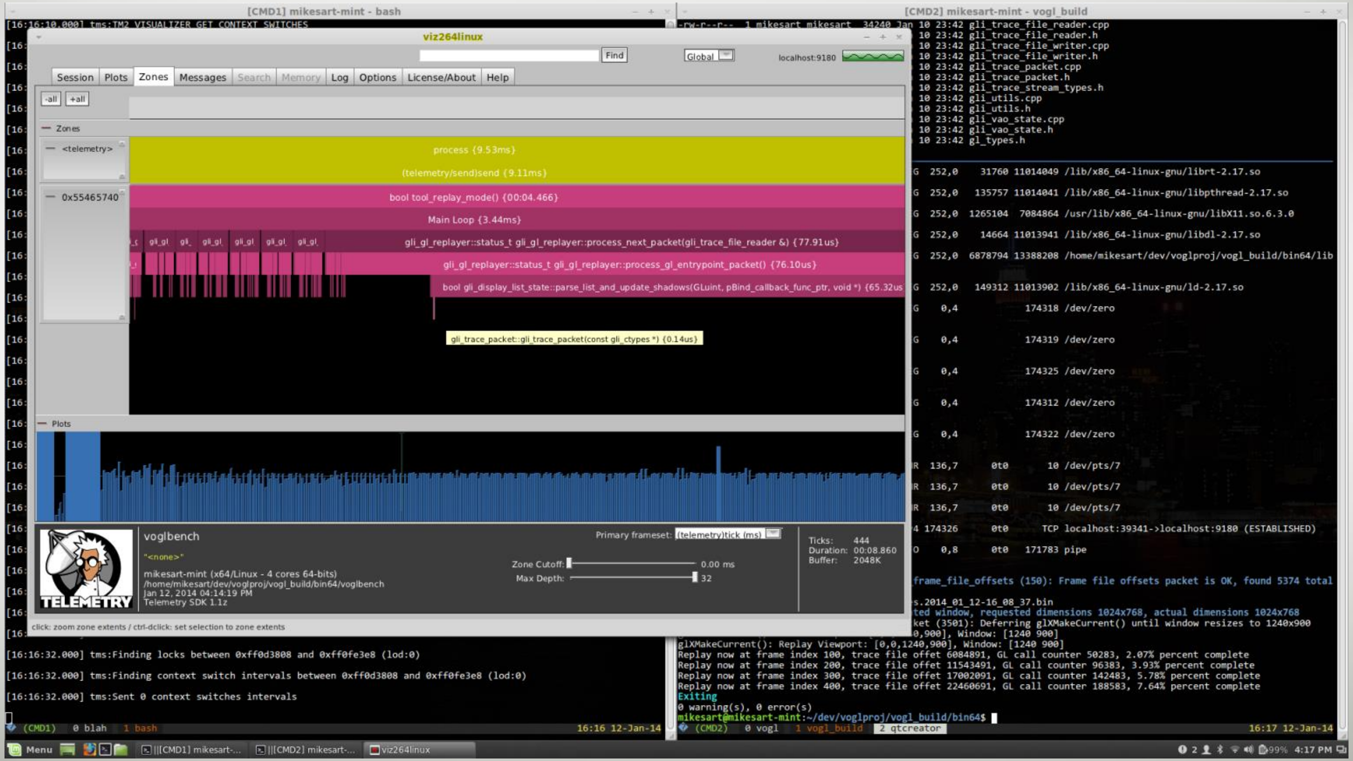
Core Tools 1/2

- **libvogltrace.so**: Tracer, loadable like libgl.so
- **voglreplay**: Command line trace processing tool which handles:
  - **Conversion**
    - Binary<->JSON traces. Conversion to/from JSON is guaranteed lossless.
  - **Playback**
    - Binary or JSON traces
  - **Trimming**
    - To 1-X frames, multi-generation trimming
  - **Dumping**
    - Dump state as JSON or FBO/backbuffer to PNG’s
  - **Finding**
    - Regex searching through API calls
  - **Statistics**

Core Tools 2/2

- **vogleditor**: Qt UI for debugging and editing trace files
- **voglbench**: Perf. and regression testing
  - Current plan is to distribute this tool to vendors and users
- **voglserver**: Run on remote box, launches apps with tracing (via Steam or directly) and controls the tracer SO
- Command line tools for remotely controlling a voglserver instance

RAD Telemetry Integration



```
process {9.53ms}
(telemetry/send)send {9.11ms}
bool tool_replay_mode() {00:04.466}
Main Loop {3.44ms}
gli_gl_replayer::status_t gli_gl_replayer::process_next_packet(gli_trace_file_reader &) {77.91us}
gli_gl_replayer::status_t gli_gl_replayer::process_gl_entrypoint_packet() {76.10us}
bool gli_display_list_state::parse_list_and_update_shadows(GLuint, pBind_callback_func_ptr, void *) {65.32us
gli_trace_packet::gli_trace_packet(const gli_ctypes *) {0.14us}
```

```
voglbench
"<none>"
mikesart-mint (x64/Linux - 4 cores 64-bits)
/home/mikesart/dev/voglproj/vogl_build/bin64/voglbench
Jan 12, 2014 04:14:19 PM
Telemetry SDK 1.1z

Zone Cutoff: 0.00 ms
Max Depth: 32

Ticks:     444
Duration:  00:08.860
Buffer:    2048K

click: zoom zone extents / ctrl-dclick: set selection to zone extents
```

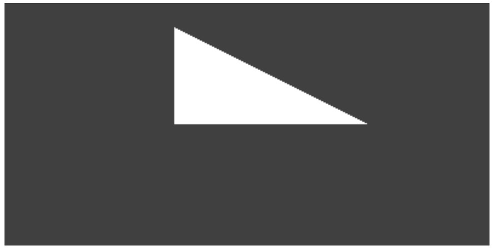
```
[16:16:10.000] tms:TM2_VISUALIZER_GET_CONTEXT_SWITCHES
[16:16:32.000] tms:Finding locks between 0xff0d3808 and 0xff0fe3e8 (lod:0)

[16:16:32.000] tms:Finding context switch intervals between 0xff0d3808 and 0xff0fe3e8 (lod:0)

[16:16:32.000] tms:Sent 0 context switches intervals

 (CMD1)  0 blah  1 bash                                              16:16 12-Jan-14
```

```
-rw-r--r--  1 mikesart mikesart  34240 Jan 10 23:42 gli_trace_file_reader.cpp
                                           10 23:42 gli_trace_file_reader.h
                                           10 23:42 gli_trace_file_writer.cpp
                                           10 23:42 gli_trace_file_writer.h
                                           10 23:42 gli_trace_packet.cpp
                                           10 23:42 gli_trace_packet.h
                                           10 23:42 gli_trace_stream_types.h
                                           10 23:42 gli_utils.cpp
                                           10 23:42 gli_utils.h
                                           10 23:42 gli_vao_state.cpp
                                           10 23:42 gli_vao_state.h
                                           10 23:42 gl_types.h

G  252,0    31760 11014049 /lib/x86_64-linux-gnu/librt-2.17.so

G  252,0   135757 11014041 /lib/x86_64-linux-gnu/libpthread-2.17.so

G  252,0  1265104  7084864 /usr/lib/x86_64-linux-gnu/libX11.so.6.3.0

G  252,0    14664 11013941 /lib/x86_64-linux-gnu/libdl-2.17.so

G  252,0  6878794 13388208 /home/mikesart/dev/voglproj/vogl_build/bin64/lib

G  252,0   149312 11013902 /lib/x86_64-linux-gnu/ld-2.17.so
G    0,4            174318 /dev/zero

G    0,4            174319 /dev/zero

G    0,4            174325 /dev/zero

G    0,4            174312 /dev/zero

G    0,4            174322 /dev/zero

R  136,7      0t0       10 /dev/pts/7
R  136,7      0t0       10 /dev/pts/7
R  136,7      0t0       10 /dev/pts/7
4 174326      0t0      TCP localhost:39341->localhost:9180 (ESTABLISHED)
O    0,8      0t0   171783 pipe

frame_file_offsets (150): Frame file offsets packet is OK, found 5374 total

s.2014_01_12-16_08_37.bin
ted window, requested dimensions 1024x768, actual dimensions 1024x768
ket (3501): Deferring glXMakeCurrent() until window resizes to 1240x900
0,900], Window: [1240 900]
glXMakeCurrent(): Replay Viewport: [0,0,1240,900], Window: [1240 900]
Replay now at frame index 100, trace file offet 6084891, GL call counter 50283, 2.07% percent complete
Replay now at frame index 200, trace file offet 11543491, GL call counter 96383, 3.93% percent complete
Replay now at frame index 300, trace file offet 17002091, GL call counter 142483, 5.78% percent complete
Replay now at frame index 400, trace file offet 22460691, GL call counter 188583, 7.64% percent complete
Exiting
0 warning(s), 0 error(s)
mikesart@mikesart-mint:~/dev/voglproj/vogl_build/bin64$
 (CMD2)  0 vogl  1 vogl_build  2 qtcreator                           16:17 12-Jan-14
```

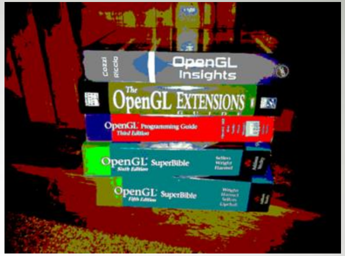
2 99% 4:17 PM

Simple JSON Trace File

```
// draw_triangle.json - Draws 1 white triangle on a gray background
// Replays with: voglreplay -endless draw_triangle.json
{
   "meta" : { "cur_frame" : 0, "eof" : true }, "sof" : { "pointer_sizes" : 4 },

   "packets" : [
       { "func" : "glXCreateContext", "context" : "0x0", "params" : { "dpy" : "0x1", "vis" : "0x1", "shareList" : "0x0", "direct" : true  },   "return" : "0x1" },
       { "func" : "glXMakeCurrent",    "context" : "0x0", "params" : { "dpy" : "0x1", "drawable" : "0x1", "context" : "0x1" }, "return" : true },

       { "func" : "glViewport", "params" : { "x" : 0, "y" : 0, "width" : 400, "height" : 200 } },

       { "func" : "glClearColor", "params" : { "red" : 0.25, "green" : .25, "blue" : .25, "alpha" : 1. } },
       { "func" : "glClear", "params" : { "mask" : "0x4000" } },

       { "func" : "glMatrixMode", "params" : { "mode" : "GL_PROJECTION" }, },

       { "func" : "glLoadIdentity" },

       { "func" : "glMatrixMode", "params" : { "mode" : "GL_MODELVIEW" } },
       { "func" : "glLoadIdentity" },

       { "func" : "glColor3f", "params" : { "red" : 1., "green" : 1., "blue" : 1. }, },
       { "func" : "glScalef", "params" : { "x" : 0.2, "y" : 0.2, "z" : 1. } },
       { "func" : "glTranslatef", "params" : { "x" : -1.5, "y" : 0., "z" : 0. } },

       { "func" : "glBegin", "params" : { "mode" : "GL_TRIANGLES" } },
          { "func" : "glVertex2f", "params" : { "x" : 0., "y" : 4. } },
          { "func" : "glVertex2f", "params" : { "x" : 4., "y" : 0. }, },
          { "func" : "glVertex2f", "params" : { "x" : 0., "y" : 0. } },
       { "func" : "glEnd" },

       { "func" : "glXSwapBuffers", "params" : {"dpy" : "0x1", "drawable" : "0x1" } }
   ]
}
```



References



- Bonus slides/info on Rich's blog: http://richg42.blogspot.com/
- John McDonald's gfxtrace - Experimental tracer for TF2 GL (Windows only):
    - https://github.com/nvMcJohn/gfxtrace
- apitrace - Full-stream tracer, replayer:
    - https://github.com/apitrace/apitrace
- apitrace's glapi.py - High quality GL API definition, contains key parameter namespace and array size information:
    - https://github.com/apitrace/apitrace/blob/master/specs/glapi.py
- Old Khronos GL/GLX .spec files - No longer updated, has many bugs/missing parameters:
    - https://cvs.khronos.org/svn/repos/ogl/trunk/doc/registry/public/oldspecs/
- Official Khronos XML spec files - Latest spec:
    - https://cvs.khronos.org/svn/repos/ogl/trunk/doc/registry/public/api/
- Alexandre Fournier's "gl-spec-parser" Python script - scrapes the Khronos reference, extension, and enumerates pages to XML:
    - https://github.com/AlexandreFournier/gl-spec-parser
- Piglit driver testing framework:
    - http://people.freedesktop.org/~nh/piglit/
- Universal Binary JSON (UBJ) format:
    - http://ubjson.org/

OpenGL is a Conversation

- OpenGL is extensible and constantly evolving
- This requires some of your attention

- The alternative, OS vendor control, is terrible
    - Stifles hardware innovation
    - Infrequent updates
    - APIs and tools tied to OS versions

VALVE
EPIC GAMES
EA
SONY make.believe
unity
FUTUREMARK
Oculus VR
TRANSGAMING

OS X Mavericks - 10.9

- Free update
- OpenGL 4.1
- We will require 10.9.x for future titles
  - A lot more reasonable since it’s free
  - 10.9.0 not quite there for us
- Test early since Apple’s latency is high

Apple

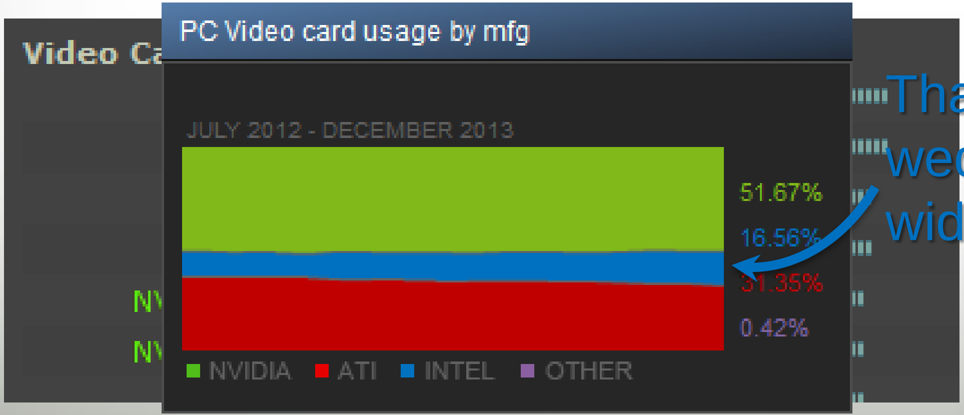
- Determine what you need, file radars and ping your contacts there
- We need some extensions that Mavericks lacks:

Radar	Extension
14495282	ARB_shading_language_420pack
14495583	EXT_clip_control
14495565	ARB_debug_output

Intel

- No excuses. You have a Haswell computer:

- Your customers have Intel GPUs:



That blue wedge is widening

Adding in OS X

Call To Action

- Front load your move to OpenGL
  - This is not a port. This is your 3D API.
  - Set up the hardware vendors to succeed
  - IHVs have internal Direct3D resource bias but we can change that with numbers
- Manage risk
  - If you have one, keep your D3D path alive for now
  - Useful as a basis for comparison anyway
- Join the conversation

Summary

- OpenGL Strategy
- Shipping shaders
- VOGL – Bang on it when it’s released!

Questions?

Linux / OpenGL Breakout Session

- 5pm in this room
  - Demo of game recording

References

- glslang: https://cvs.khronos.org/svn/repos/ogl/trunk/ecosystem/public/sdk/tools/glslang
- hlsl2glslfork: https://github.com/aras-p/hlsl2glslfork
- glsl-optimizer: https://github.com/aras-p/glsl-optimizer
- MojoShader: https://icculus.org/mojoshader/
- HLSLCrossCompiler: https://github.com/James-Jones/HLSLCrossCompiler
- fxdis-d3d1x: https://code.google.com/p/fxdis-d3d1x/
- cgc: http://http.developer.nvidia.com/Cg/cgc.html
- OpenCL SPIR: http://www.khronos.org/files/opencl-spir-12-provisional.pdf
- Porting Source to Linux: Valve’s Lessons Learned: http://www.gdcvault.com/play/1017850/