OpenGL ES 2.0 : Start Developing Now

Dan Ginsburg
Advanced Micro Devices, Inc.

Agenda

- OpenGL ES 2.0
  - Brief Overview
- Tools
  - OpenGL ES 2.0 Emulator
  - RenderMonkey w/ OES 2.0 Support
- OpenGL ES 2.0 3D Engine Case Study

What is OpenGL ES 2.0?

- OpenGL for Embedded Systems
- OpenGL ES 2.0
  - Fully shader-based
  - Based on ES Shading Language
  - Draft spec released at SIGGRAPH 05
  - Spec ratified and released at GDC 07

OpenGL ES 2.0 – Widespread Industry Support

- OpenGL ES 2.0 support announced from many companies:
  - AMD
  - NVIDIA
  - Imagination Technologies
  - ARM
  - ...and more…
- OpenGL ES 2.0 will become ubiquitous

ES 2.0 – The Problem for Game Developers

- Developers need to develop their game engines in advance of new hardware
- No hardware available today
- OpenGL ES 2.0 may require handheld developers to change their engines significantly
  - Shader-based API moves more burden to the application
  - Enables more flexibility through programmability

ES 2.0 – A Development Solution

- OpenGL ES 2.0 Emulator
  - OpenGL ES 2.0 implementation for Win32
  - Allows developers to write their engines in advance of hardware
- OpenGL ES 2.0 RenderMonkey
  - Develop OpenGL ES 2.0 shaders and effects

OpenGL ES 2.0 Emulator

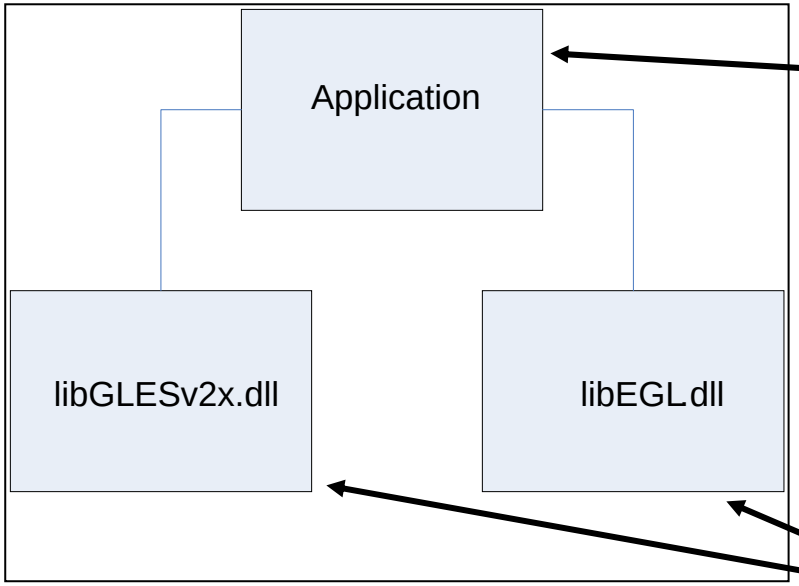
OpenGL ES 2.0 Emulator Goals

- Provide an OpenGL ES 2.0 development environment on the PC
  - Minimize porting effort once hardware is available
- Leverage features/performance of desktop hardware

OpenGL ES 2.0 Emulator – What is it?

- OpenGL ES 2.0 – libGLESv2x.dll + lib
- EGL 1.3 – libEGL.dll + lib
- Khronos standard header files
- Example programs
- Utilizes desktop hardware for rendering
  - Requires desktop OpenGL 2.0 hardware

OpenGL ES 2.0 Emulator – Usage Overview



Win32 Application Includes:
- GLES2/gl2.h
- EGL/egl.h

Links Against:
- libGLESv2x.lib
- libEGL.lib

Emulator:
- Implements ES 2.0 API plus extensions
- Implements EGL 1.3 API

OpenGL ES 2.0 Emulator - Features

- OpenGL ES 2.0 Core API
  - Full OpenGL ES 2.0 Implementation
- Optional Extensions:
  - 10.10.10.2 Vertex/Texture Data
  - FP16 Vertices and Textures
  - 3D and Non-Power-2 Textures
  - Compressed Texture Formats
    - ETC1, ETC3, ETC5, ATI_TC
  - Occlusion and Conditional Queries
  - Depth Textures

OpenGL ES 2.0 – Demo

OpenGL ES 2.0 Emulator – Enables Developers

- More than just a prototyping tool
  - Graphics code should move over easily from emulator to real hardware
- Mirrors top tier handheld developer approaches
  - Prototype on the PC
  - Move to handheld device as a final step

OpenGL ES 2.0 Emulator

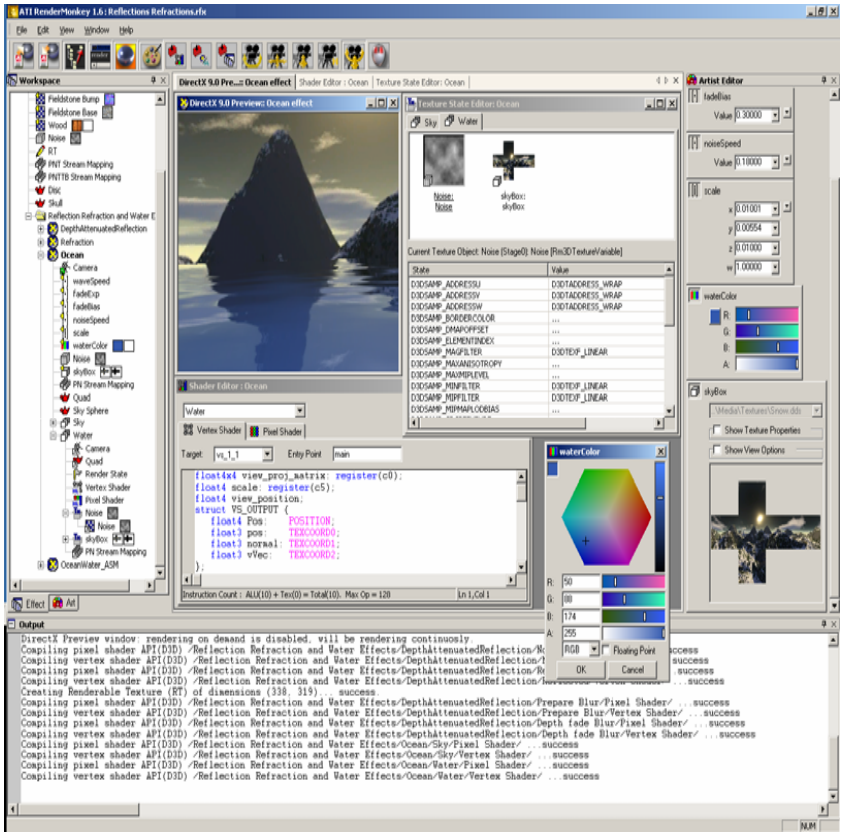
- Contact devrel@amd.com for more information
- PowerVR also provides an emulator and SDK:

  http://www.powervrinsider.com

Render Monkey – OpenGL ES 2.0 Support

What is RenderMonkey?

- Shader Development Environment
  - Rapid Prototyping of Shader Effects
- Multiple Shading Languages
  - **OpenGL ES Shading Language**
  - OpenGL Shading Language
  - DirectX HLSL
  - DirectX Assembler



RenderMonkey – Why use it?

- Full IDE for shader effect development
  - Programmer and artist view for rapid iteration
- Easy integration into game pipeline
  - Plug-in SDK for custom import/export
    - Effects, models, textures, variables, etc.
  - Support for many standard formats
    - DDS, BMP, TGA, X, OBJ, 3DS, FX
- Encompasses all effect resources

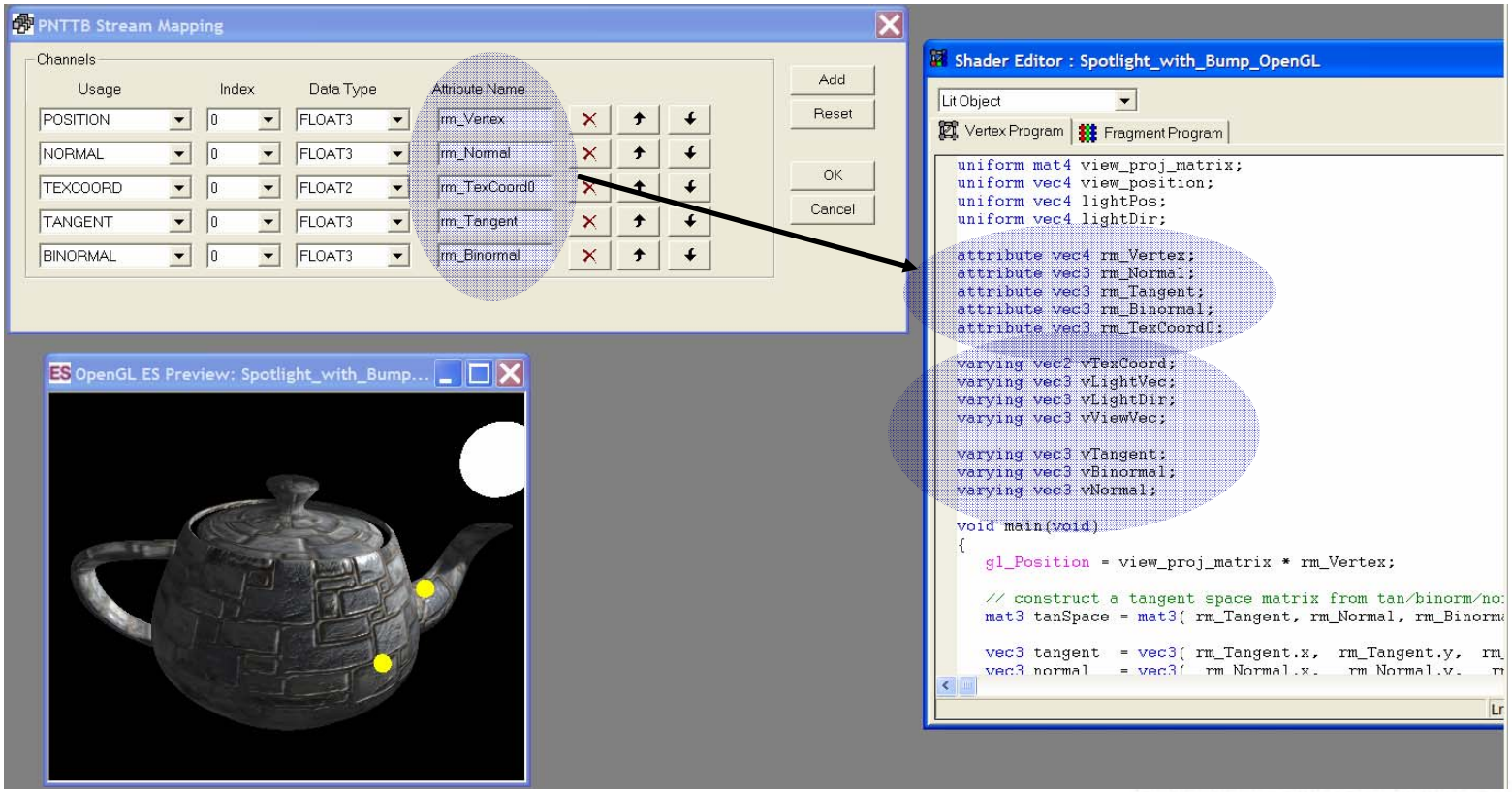
  Render state, texture state, variables, render targets, textures, models, etc…

RenderMonkey – What's new?

- Support for OpenGL ES 2.0
  - ES Shading Language v1.00
  - ES syntax highlighting
  - ES render/sampler states
  - Large suite of ES examples
  - User editable vertex attribute names

RenderMonkey – What is Different with ES Shaders?

- Generic vertex attributes
- User varyings



RenderMonkey – What is Different with ES Shaders?

- Most built-in uniforms removed
  - e.g. gl_ModelViewMatrix
  - RenderMonkey provides equivalent user named uniforms
- Default precision qualifier required for FS
- Various limitations:
  - Loop constructs
  - Relative addressing
- Extension enabling with #extension:
  - 3D Textures, derivatives

RenderMonkey – What is Different with ES Effects?

- Reduced render state
  - Alpha test must be done with *discard*
  - No polygon fill mode
  - No fixed-function state: fog, point size, etc.
- Reduced sampler state
  - Less texture wrap modes
  - No fixed-function LOD bias
  - No texture border color

RenderMonkey - Demo

ATI RenderMonkey 1.70 : MultiTone Car Paint.rfx

File Edit View Window Help

Workspace

- microflakeNMap
- normalMap
- showroomMap
- standard mapping
- Car
- Car Paint Effect Group_DX9
  - MultiTone Car Paint w/out Env. Ma
  - Environment Map Only Effect
  - MultiTone Car Paint
  - One Tone Microflake Effect
  - Full Car Paint Effect
- Car Paint EffectGroup_OpenGL ES
  - ES MultiTone Car Paint w/out Env. Ma
  - ES Environment Map Only Effect
  - ES MultiTone Car Paint
  - ES One Tone Microflake Effect
  - ES Full Car Paint Effect
    - Camera
    - Full Car Paint Effect
      - Camera
      - flakeSingleLayerColor
      - Car
      - Vertex Shader ES SL
      - Fragment Shader ES SL
      - normalMap
      - microflakeNMap
      - showroomMap
      - standard mapping

Effect | Art

OpenGL ES Previ...t Effect effect | Shader Editor :...ar Paint Effect

OpenGL ES Preview: Full Car ...

Shader Editor : Full Car Paint Effect

Full Car Paint Effect

Vertex Shader ES SL | Fragment Shader ES SL

```
#ifdef GL_FRAGMENT_PRECISION_HIGH
   // Default precision
   precision highp float;
#else
   precision mediump float;
#endif

uniform float microflakePerturbation;
uniform float brightnessFactor;
uniform float glossLevel;
uniform vec4 paintColorMid;
uniform vec4 paintColor2;
uniform vec4 flakeLayerColor;
uniform float normalPerturbation;
uniform float microflakePerturbationA;
uniform vec4 paintColor0;
uniform sampler2D normalMap;
uniform sampler2D microflakeNMap;
uniform samplerCube showroomMap;
//-----------------------------------------------------------------
// Phenomenological car paint visual effect: Simulates build up o
// on the metallic surface of a car or any other object. This vis
// emulates the suspended layers of microflakes in the paint coat
// blends between several colors of paint based on the viewing an
//
// Author: Natalya Tatarchuk
//       (based on the original assembly shader written by John Is
//
// Used in ATI's Car demo for Radeon 9700 launch, found here:
```

Artist Editor

paintColor0

R: G: B: A:

normalMap

Car Normal Map.tga | Browse

Show Texture Properties

Show View Options

showroomMap

Showroom.dds | Browse

Show Texture Properties

Show View Options

Output

```
Cannot compile shaders in 'Full Car Paint Effect' effect because the OpenGL ES preview window is closed.
Please open the OpenGL ES preview window to compile shaders.
OpenGL ES Preview Window: Compiling fragment shader API(OpenGL ES) /Car Paint EffectGroup_OpenGL ES/Full Car Paint Effect/Full Car Paint Effect/Fragment Shader ES SL/ ... success
OpenGL ES Preview Window: Compiling vertex shader API(OpenGL ES) /Car Paint EffectGroup_OpenGL ES/Full Car Paint Effect/Full Car Paint Effect/Vertex Shader ES SL/ ... success
OpenGL ES Preview Window: Linking program ... success
```

OpenGL ES 2.0 – 3D Engine Case Study

Sushi Demo Engine

- AMD's Demo Engine
- Support for:
  - DX9
  - DX10
  - OpenGL
  - OpenGL ES 2.0

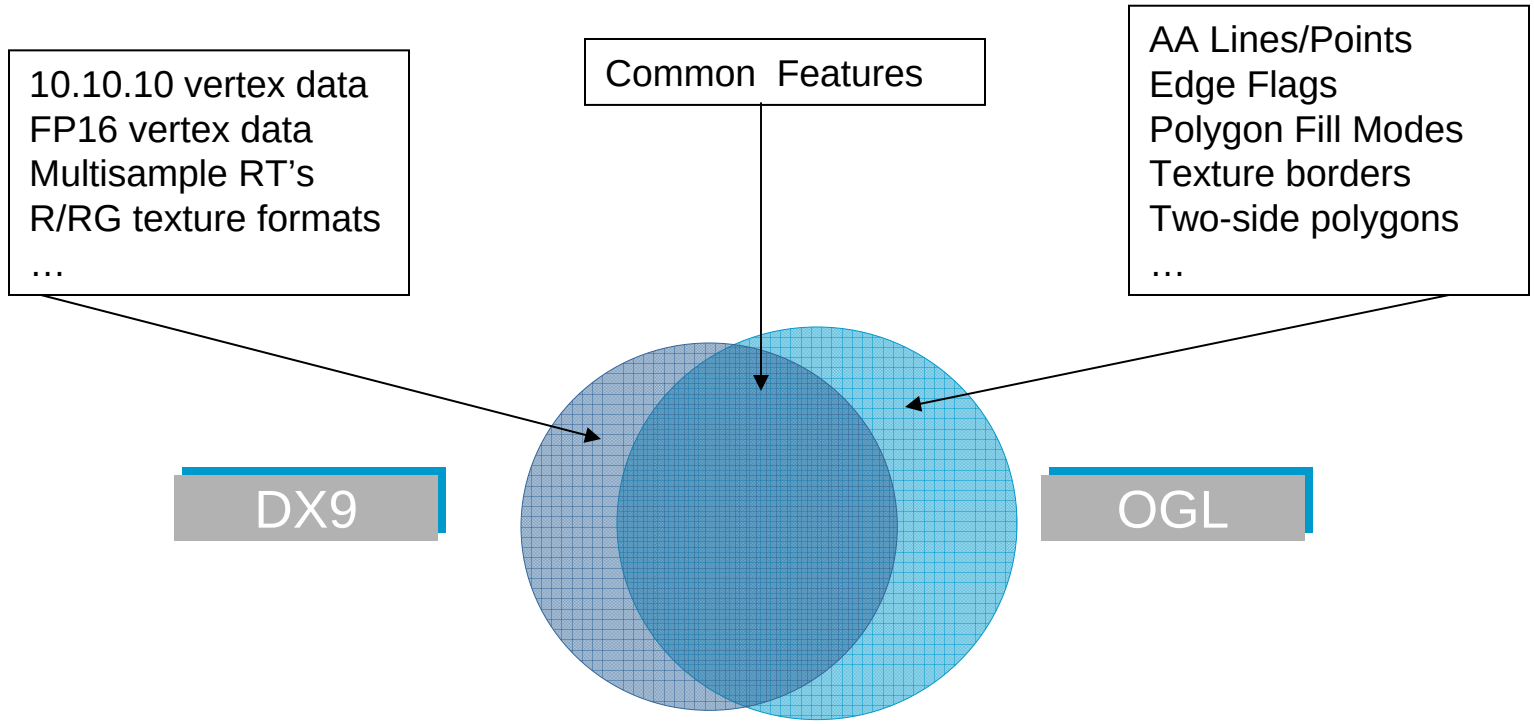
Key Challenges

- Designing an engine to target multiple APIs with different feature sets
- Designing a shader-based engine
- Platform compatibility
  - Large variance in handheld platform capabilities
  - Limitations make portability a challenge

Abstracting the Graphics API

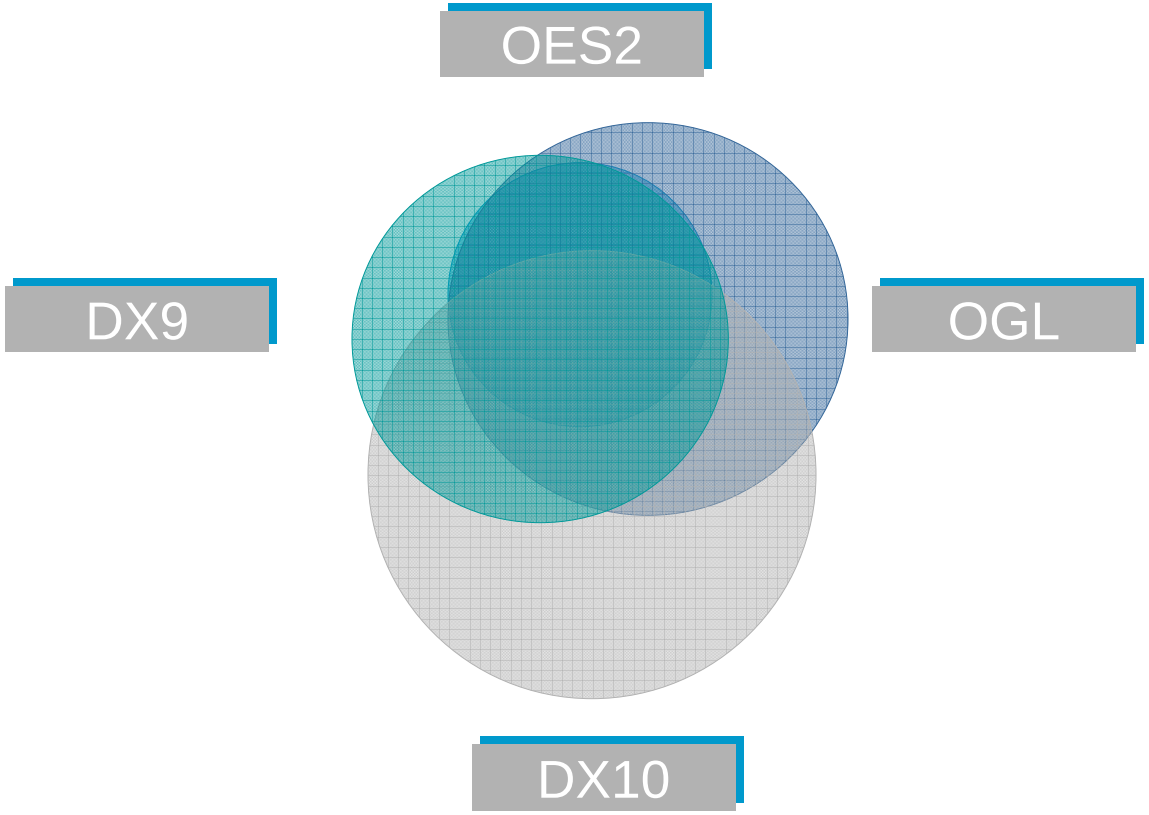
- Challenge: what level to abstract the 3D API?
  - Support all features of all APIs?
  - Support common set of features?
  - How to handle different shading languages?

State of the APIs - 2005



- In 2005, we abstracted the DX9 feature set.
- We used extensions to support missing features in OpenGL.

State of the APIs - 2007



- The choice is no longer so easy.
- Especially if you add game consoles to the mix…

Abstracting the API – How We Decided

- Driven by requirements:
  - Demos must use the latest features of all APIs
  - Exposing the lowest-common denominator not an option
- Running the same demo on each API not a requirement
- Let content drive the feature set rather than the API abstraction

Abstracting the API – What We Did

- Our API abstraction looks a lot like DX10
  - Resources
  - Views
  - Geometry Shaders
  - Stream Out
  - All the latest and greatest features…
- Each API implementation supports a subset of these features

API Abstraction – Fallback Paths

- Demo Engine is based off a scripting system using Lua
- Lua script provides fallback rendering paths.
- Trade off: High end features vs. Content portability
  - For Sushi, this was a fair tradeoff to make
  - It might not be for you…

Sushi - Effect System

- Encapsulate essential information about rendering techniques
- Essential part of shader-based engine
  - Develop our own?
  - Use someone else's?
    - Microsoft .FX
    - COLLADA FX
    - CgFX
- At the time, no existing solution fully fit our requirements

Sushi – Effect System Goals

- Multiple API / Shading Language Support
  - HLSL, GLSL, ES SL
- Flexible support for advanced rendering techniques
  - The effect system is the foundation that all the demos are built on

Sushi Effects – Cross-API Effect System

- Expresses the following data:
  - Shaders
  - Render State
  - Passes
  - Techniques
  - Variable Bindings
- Similar to Microsoft .FX, but multiple API support

Shader Authoring

- Many of our shaders authored in HLSL
- Needed a way to convert to:
  - OpenGL Shading Language
  - OpenGL ES Shading Lanauge
- Wrote a tool for this purpose:
  - HLSL2GLSL

HLSL2GLSL

- Command-line tool and library
- Converts SM 3.0 HLSL shaders to:
  - GLSL v1.10.59 shaders
  - ES SL v1.00 shaders
- Open-source:
  - http://sourceforge.net/projects/hlsl2glsl
  - Very flexible BSD license

Sushi – Platform Portability

- Handheld platforms have many constraints:

  Examples:

  - No Standard Template Library
  - No C++ Exceptions
  - Manual Cleanup Stack
  - Incomplete Standard Libraries
  - Limited Memory Footprint
  - No Floating Point Unit

Sushi - Portability

- Standard abstraction layers
  - Math, I/O, Memory, Window, etc.
- Custom template classes
  - Lists, vectors, maps, etc.
- Constrained use of C++
  - No exceptions
  - No STL

Summary

- Tools
  - OpenGL ES 2.0 Emulator
  - RenderMonkey w/ OES 2.0 Support
- OpenGL ES 2.0 3D Engine Case Study
  - Graphics API Abstraction
  - Effects System
  - Portability

Questions?

dan.ginsburg@amd.com