Seamless Compute and OpenGL Graphics Development in NVIDIA® Nsight™ 3.1 Visual Studio Edition

Jeff Kiel - Manager, Graphics Tools
Agenda

- Computational Graphics and Visual Computing
- Maximus™
- Developer Challenges
- Debugging Computational Graphics with Nsight Visual Studio Edition
- Conclusion and Q&A
Deferred Shading
Ambient Occlusion
Simulation & Visualization
Medical imaging
Manufacturing
Compute Shader Based Deferred Shading

Graphics pipeline rasterizes gbuffers for opaque surfaces

Compute pipeline uses gbuffers
  - cull light sources
  - compute lighting
  - shading

Johann Anderson (DICE) - Compute Shader Based Deferred Shading
Multi-GPU Platform Overview

APP

Graphics API/CUDA

OS

GPU Driver

KMD

MEM

CPU

MEM

GPU

MEM

CE1

CE2

CE1

CE2

I/O HUB
Single-GPU Simulation and Visualization

- Mem. xfers
- Draw
- Kernels
- Mem. xfers

\[ F_{n-1} \]

\[ F_n \]
Multi-GPU Simulation and Visualization

\[ F_{n-1} \Rightarrow F_n \Rightarrow F_{n+1} \]

Kernels
Mem. xfers
Draw
Mem. xfers
Developer Challenges

- Debugging GPU Compute and Graphics with interop
- Context switching overhead
- Multi-Core/Multi-GPU race conditions
- Data transfers - host-to-device, device-to-host, P2P
- Asynchronous transfers
- Multi-core/Multi-threaded
- Multi-GPU - Graphics/Graphics, Graphics/Compute
- Concurrent kernel execution
- Driver models (TCC/WDDM)
NVIDIA® Nsight™ Visual Studio Edition

Visual Studio integrated development for GPU and CPU
Frame profiler - OpenGL 4.2, Direct3D 9/11
- Automatic GPU bottleneck determination
- Draw call and frame timings
- Grouping/sorting by NVTX/D3D Perf Markers and state

Frame debugger - OpenGL 4.2, Direct3D 9/11
- Draw call and state inspection
- Frame capture and playback (source code gen D3D9/11)
- Nsight HUD for draw call scrubbing and inspection

HLSL and GLSL Shader debugger
- Native GPU shader debugging and GPU memory views
- Complex condition breakpoints and Pixel History
- Local single GPU shader debugging

Application and system trace
- Inspect OpenGL and Direct3D workloads, CPU and GPU
- Correlate threads, call stack, API calls, WDDM kernel queues and resulting GPU workloads
- Concurrent draw call execution and memory transfer trace
NVIDIA Nsight for Compute Developers
CUDA 4.2 and 5.0 and Maximus™

CUDA debugger
- Debug CUDA kernels directly on GPU hardware
- Info page and Parallel warp watch view
- Use on-target conditional breakpoints to locate errors

CUDA memory checker
- Out of bounds memory access detection
- Enables precise error detection

Application and system trace
- Review CUDA workloads across CPU and GPU
- Activity correlation panel

CUDA profiler
- Source code correlation
- Deep kernel analysis to detect factors limiting maximum performance
- Unlimited experiments on live kernels
Debugging and Profiling

IslandsGL - GPU water simulation

Debugging

GI Voxels - Indirect illumination using voxel tracing

Application Tracing

Bindless GBuffer - Deferred rendering with bindless

Profiling
Demo: Setting up your application

Set up target application, working directory, arguments, and machine

Enable serialization of application and assets to target machine
Demo: Debugging IslandsGL

- Real time GPU performance graphs
- Batch Histogram with enable/disable feature
Demo: Debugging IslandsGL

Scrub through the scene draw call by draw call.

See details about the current call.

Preview of current render targets, depth, & stencil.
Demo: Debugging IslandsGL

Resource inspection on the HUD
Demo: Debugging IslandsGL

- Full Visual Studio integration with docking
- State visualization with API Inspector
- Pretransform Geometry
Demo: Debugging IslandsGL

Scrub through scene draw call by draw call

Visualize scene ranges using NV Tools Extension
Event List shows all API calls in the frame (with parameters)

Performance stats including CPU and GPU call duration

Filter based on API call text
Demo: Debugging IslandsGL

API Inspector shows API state for entire pipeline

<table>
<thead>
<tr>
<th>Current vertex attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Current generic attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Index</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

API Inspector shows API state for entire pipeline
Demo: Debugging IslandsGL

Textures page shows format, mips, etc.

Per unit texture parameters
Demo: Debugging IslandsGL

Shader pages show samplers

Active Uniforms
Demo: Debugging IslandsGL

Program page shows params/shaders for currently bound program

Active uniforms

Bound samplers & images
Demo: Debugging IslandsGL
Demo: Debugging IslandsGL

Resource window shows all used resources

Resource details like format, size, etc.

Thumbnails of resource revisions
Demo: Debugging IslandsGL

Use mouse wheel to zoom in and inspect bitmap resources

Text readout of color values
Demo: Debugging IslandsGL

Display binary buffers according to the underlying data type

<table>
<thead>
<tr>
<th>Offset</th>
<th>Address</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x00000000</td>
<td>255.99933215 -4000.00000000 255.00000000 1.00000000</td>
</tr>
<tr>
<td>0x0000010</td>
<td>0.42550000</td>
<td>0.85100001 2.00000000 255.00000000 1.00000000</td>
</tr>
<tr>
<td>0x0000020</td>
<td>0x00000000</td>
<td>256.00000000 1.00000000 0.42550000 2.00000000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

**Resource Info**

- **Type**: Buffer
- **Name**: Z
- **Revision**: 0
- **Size**: 5700
Demo: Debugging IslandsGL

- Zoom resource display
- Display list of contributing fragments
- Select pixel to gather history on
Demo: Debugging IslandsGL

Conditional breakpoint to inspect fragment’s shader instance

Debug Pixel
Demo: Debugging IslandsGL

Shader debugging - standard VS windows work as expected

Nsight focus picker gives context for the graphics threads
Demo: Debugging IslandsGL

Frame Timings gives quick performance feedback

FPC: Full Pipeline Cost
EPC: Empty Pipeline Cost
IDC: Incremental Draw Cost
Demo: Application Analysis GI Voxels

Select Trace Settings

Launch Application
Demo: Application Analysis GI Voxels

Filtering based on CPU core utilization
Demo: Application Analysis GI Voxels

Timeline shows details about what process/thread runs on what core.
Demo: Application Analysis GI Voxels

CPU/GPU frame latency
Demo: Application Analysis GIVoxels

Timeline shows CPU->GPU and GPU->CPU transfers
Demo: Application Analysis GLVoxels

Draw calls executing in parallel on GPU
Demo: Application Analysis GIVoxels

Capture complete stack trace for every API call and...
Demo: Application Analysis GLVoxels

```c
// set position
if (levUP == 8)
{
    GLuint vertEscVO = glGetAttribLocation(meshStruct::VA_POSITION);
    glEnableVertexAttribArray(vertEscVO);
    glVertexAttribPointer(vertEscVO, 3, GL_FLOAT, GL_FALSE, 0, 0);
}

// draw
void MeshesRenderer::draw(GLuint prog, uint curFrame)
{
    if (prog != lastProg)
    {
        locAttributes[MeshStruct::VA_POSITION] = glGetUniformLocation(prog, "vertexPosition");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(prog, "vertexTexCoord");
        locAttributes[MeshStruct::VA_NORMAL] = glGetUniformLocation(prog, "vertexNormal");
        locAttributes[MeshStruct::VA_TEXCOORD] = glGetUniformLocation(program, "d_curMaterialIndex");
        for (uint attY = 0; attY < MeshStruct::VA_MAX_MESH_VERTEX_ATTRIBS; attY++)
```

...link back to source code
Demo: Application Analysis GLVoxels

Similar data in tabular form
Demo: Performance Bindless GBuffer

Bindless Graphics app running at 17FPS before
Demo: Performance Bindless GBuffer

Shader, L2, and Memory bottlenecks
Demo: Performance Bindless GBuffer

Running on GeForce GTX660 which has 144.2 GB/s of memory bandwidth

54.7 ms total kernel time = 7.89 GB of memory bandwidth available, we used 5.95GB or 75%
Demo: Performance Bindless GBuffer

L1 hit rate of 14% 
\((9.4 - 8.1) / 9.4 = .138\)

L2 hit rate of \(~37\%\)  
\((8.1 - 5.1) / 8.1 = .370\)
Demo: Performance Bindless GBuffer

Additional performance graphs

Group draw calls by common state
Demo: Performance Bindless GBuffer

After optimizations running at ~23 FPS

Performance improvements:
1) Multiple tiled allocators
2) Localized memory access
3) Simplified spill algorithm
Demo: Performance Bindless GBuffer

Kernel times reduced from 54.7ms to 41.1ms

Shader, L2, and Memory bottlenecks reduced
Demo: Performance Bindless GBuffer

41.1 ms total kernel time = 5.93 GB of memory bandwidth available, we used 3.24 GB or 55%
Demo: Performance Bindless GBuffer

L1 hit rate of 26%
\[
\frac{(5.8 - 4.3)}{5.8} = 0.259
\]

L2 hit rate of ~42%
\[
\frac{(4.3 - 2.4)}{4.3} = 0.419
\]
NVIDIA® Nsight™ Visual Studio Edition

Download

http://www.nvidia.com/nsight

Developer Forums


Documentation

Start → All Programs → NVIDIA Nsight Visual Studio Edition 3.0 → User Guide

Instructional videos

Come and try Nsight at the Nsight Training Labs

Room 205A (Second Floor)

<table>
<thead>
<tr>
<th>Time</th>
<th>Tuesday, July 23</th>
<th>Wednesday, July 24</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:30am - 10:30am</td>
<td>Debugging</td>
<td>Profiling</td>
</tr>
<tr>
<td>11:00am - 12:00pm</td>
<td>Profiling</td>
<td>Debugging</td>
</tr>
<tr>
<td>1:30pm - 2:30pm</td>
<td>Debugging</td>
<td>Profiling</td>
</tr>
<tr>
<td>3:00pm - 4:00pm</td>
<td>Profiling</td>
<td>Debugging</td>
</tr>
<tr>
<td>4:30pm - 5:30pm</td>
<td>Debugging</td>
<td>Profiling</td>
</tr>
</tbody>
</table>
Backup from here on...
Bulleted content
Line Chart Sample

Series 1

Series 2

Series 3