Quadro Kepler for Developers Theater @ SIGGRAPH 2012

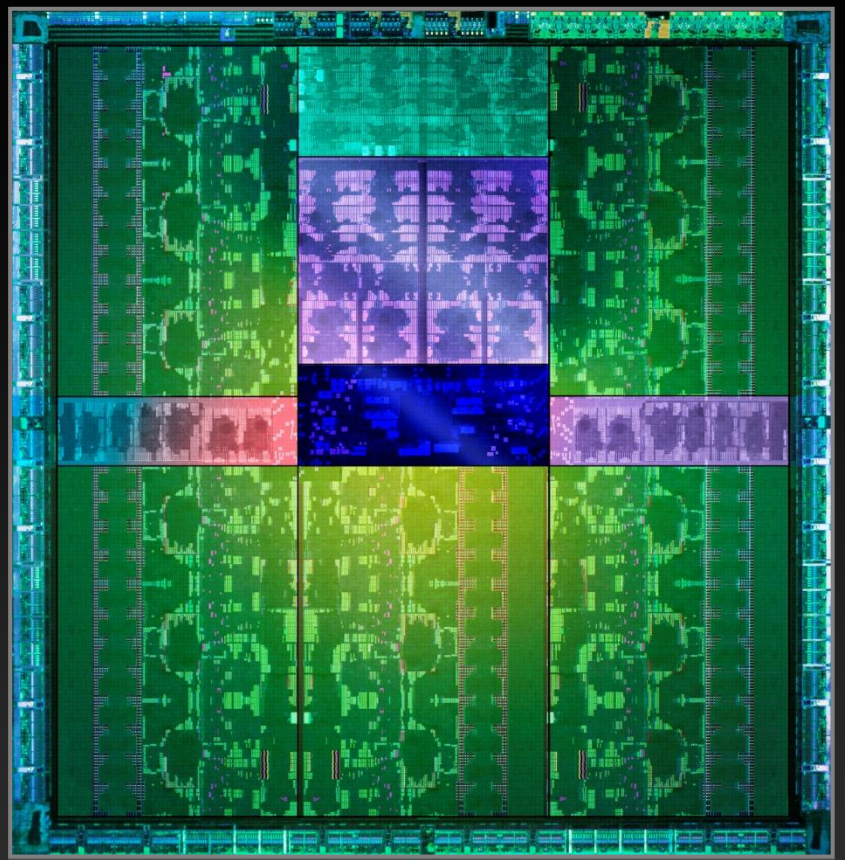
Ian Williams – Director Applied Engineer, NVIDIA

Shalini Venkataraman – Senior Applied Engineer, NVIDIA

Quadro Kepler for Developers

Topics:

- **Kepler Architecture**
- **Maximus**
- **OpenGL 4.3**
- **WMI**
- **Developer Tools**
- **Summary**

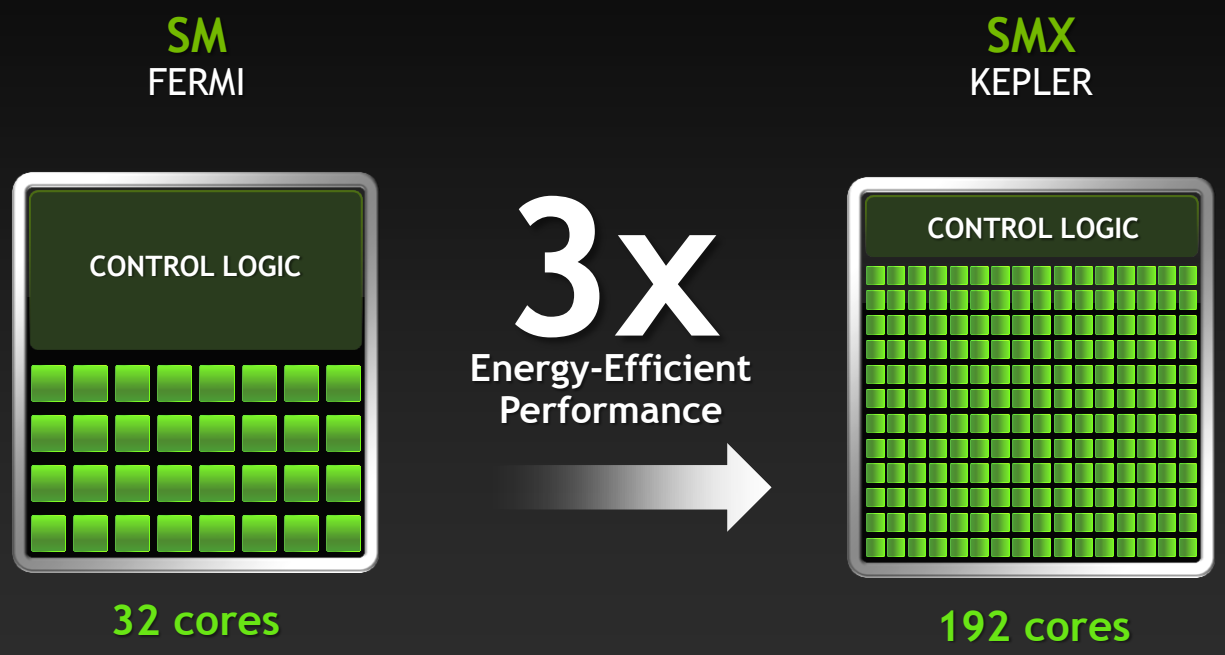


KEPLER
Architectural Features

- SMX
- PCIe Gen 3
- Texture improvements
- FXAA/TXAA
- Display Features
- Dynamic Parallelism

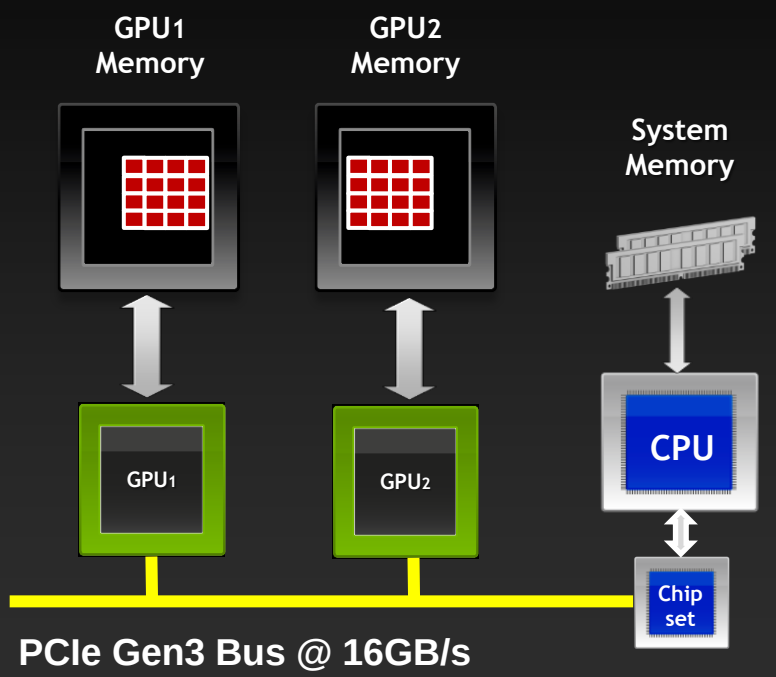
SMX

Delivers more processing performance and efficiency through an innovative streaming multiprocessor design that allows a greater percentage of space to be applied to processing cores versus control logic



PCIe Gen3

Next generation bus interconnect improves GPU to system to GPU data throughput for applications that require large data movement between system resources



2x Bandwidth

vs.

PCIe Gen2

Kepler Texture Improvements

- Increased Texture Units

Texture Slicing

Object-Order

CPU - Plane-box intersection
GPU – 3d texture lookup + blending

Raycasting (CUDA & GLSL)

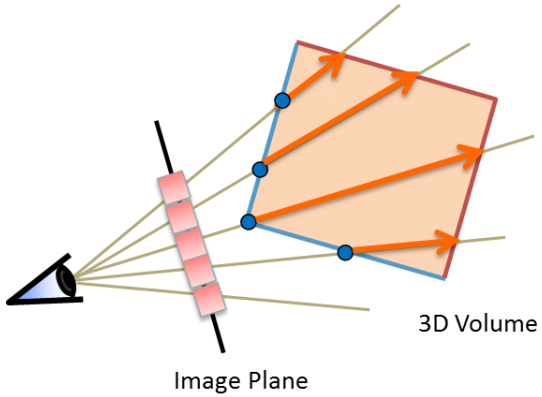
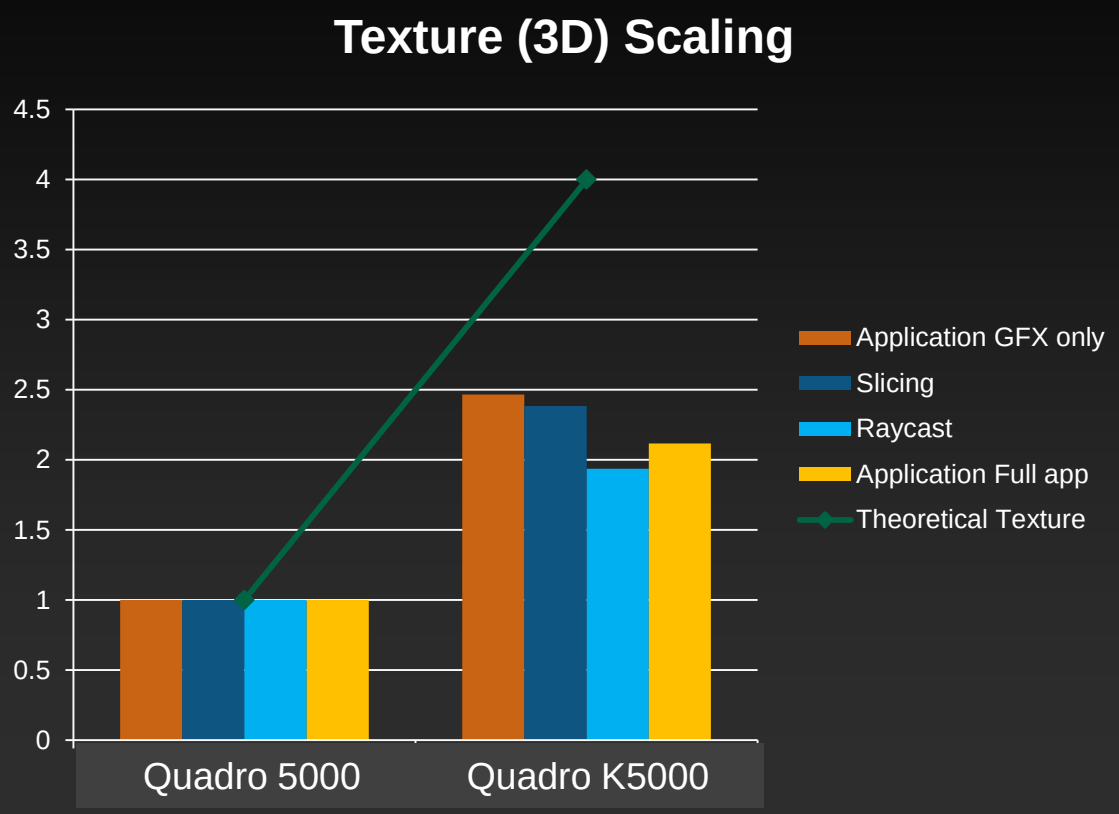


Image-Order

CPU – invokes fragment shader or cuda kernel
GPU – All the heavy lifting!
Raycasting + texture lookup + blending

Quadro K5000



Kepler Texture Improvements cont.

- Deep 3D Textures
  - Z >> X & Y
- Extension GL_NVX_deep_textures
- GL_MAX_3D_TEXTURE_SIZE will still be 4k.
- Introduce two or three new glGet() queries for:

  GL_MAX_DEEP_3D_TEXTURE_WIDTH = 2k
  GL_MAX_DEEP_3D_TEXTURE_HEIGHT = 2k
  GL_MAX_DEEP_3D_TEXTURE_DEPTH = 16k

Kepler Texture Improvements cont.

- **Increased Texture Sizes:**

Architecture	OGL/ GeForce	OGL/ Tesla	OGL/ Quadro	Cuda/ GeForce	Cuda/ Tesla	Cuda/ Quadro
Fermi	512^3	512^3	2k^3	2k^3	2k^3	2k^3
Kepler	2k^3	2k^3	4k^3, Deep	2k^3	4k^3, Deep	4k^3, Deep

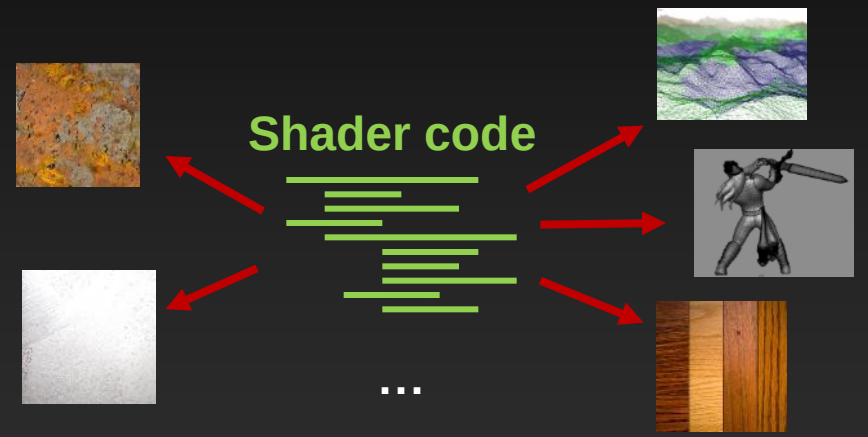
Bindless Graphics

- **Problem:** Binding to different objects (textures, buffers) takes a lot of validation time in driver
  - And applications are limited to a small palette of bound buffers and textures
  - Approach of OpenGL, but also Direct3D
- **Solution:** Exposes GPU virtual addresses
  - Let shaders and vertex puller access buffer and texture memory by its virtual address!

Kepler – Bindless Textures

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

Pre-Kepler texture binding model



Kepler bindless textures
over 1 million unique textures

Kepler - Bindless Textures

Pre-Kepler 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

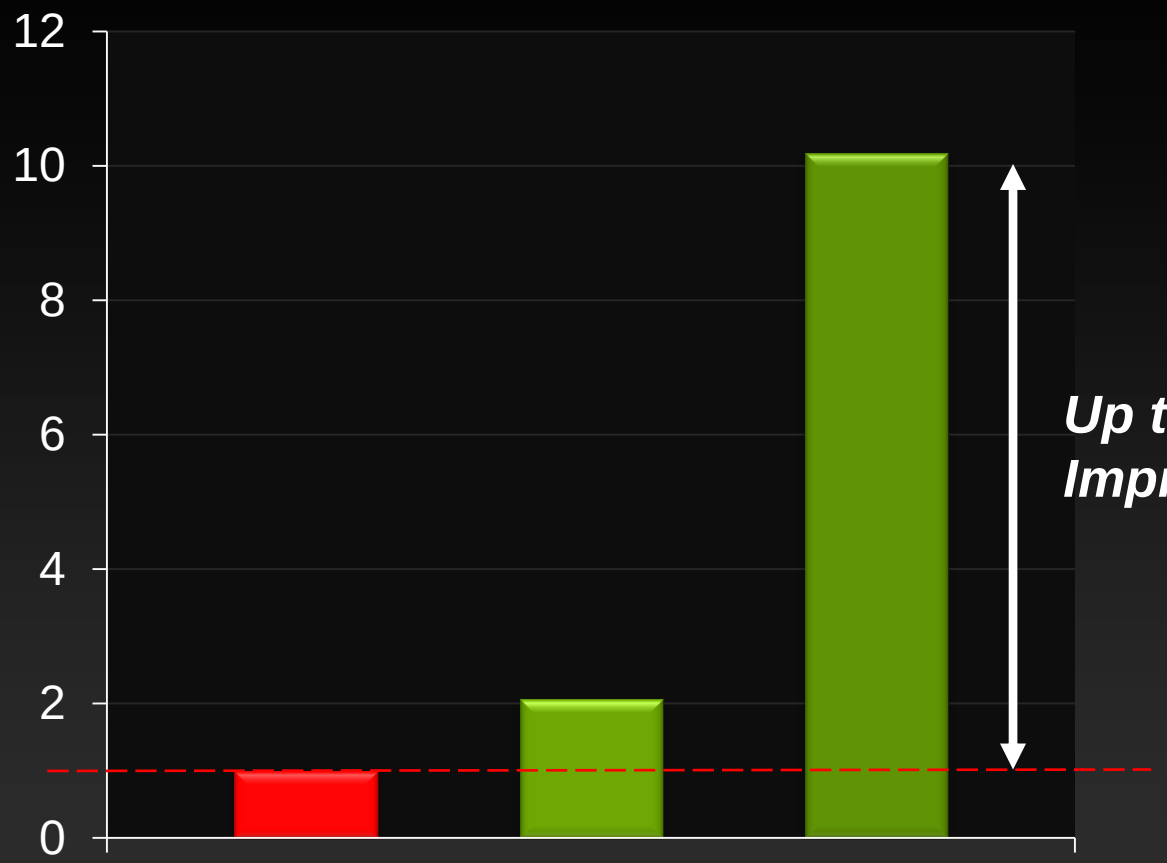
Kepler bindless textures

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

Bindless performance benefit



Up to 10x theoretical performance Improvement with bindless

Pre-Kepler texture binding
Bindless - heavy GPU load
Bindless - light GPU load

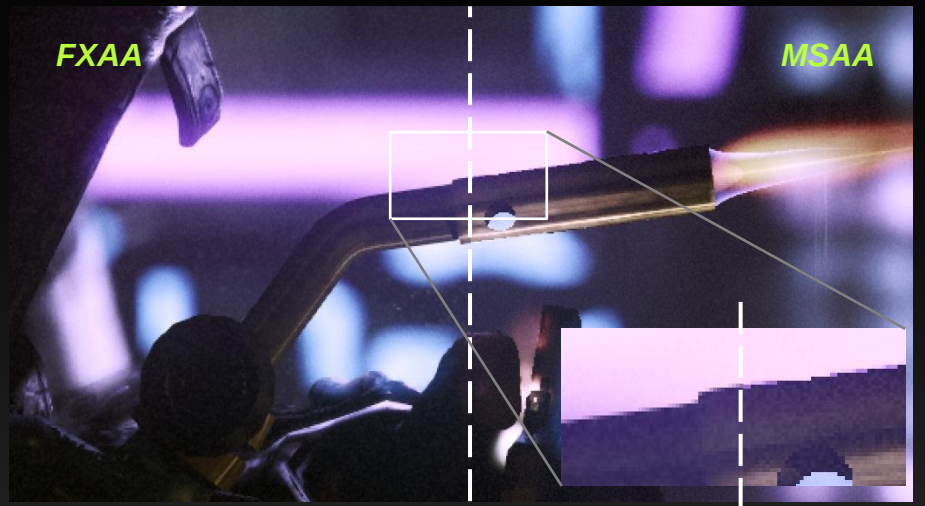
Numbers obtained with a directed test

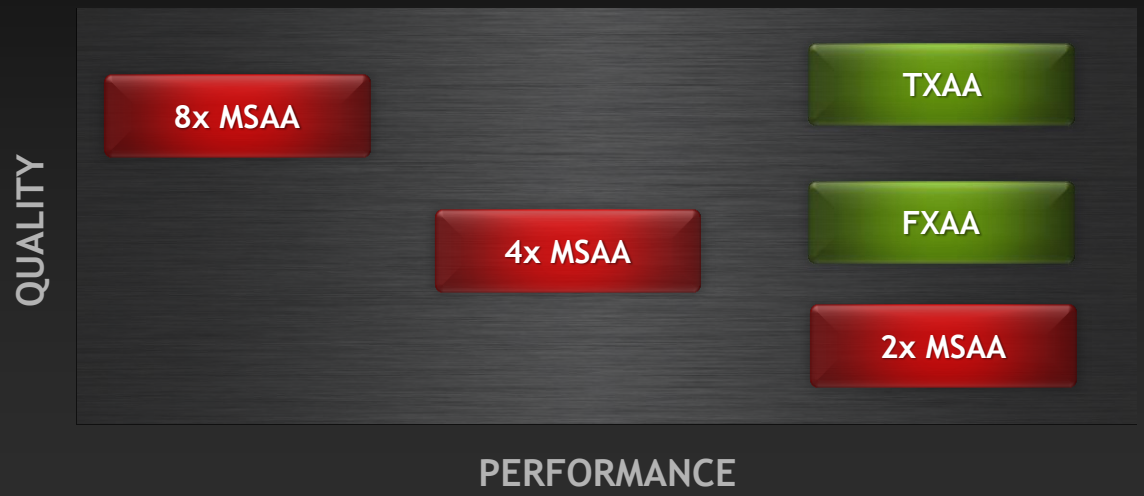
FXAA / TXAA

- FXAA user configurable
- Allows higher quality filtering without requiring sub-sample storage
- Higher quality for the lower memory footprint

- TXAA is a developer feature
- Currently DX
- Temporal Filtering in addition to spatial filtering
- Higher quality

FXAA / TXAA


FXAA
MSAA

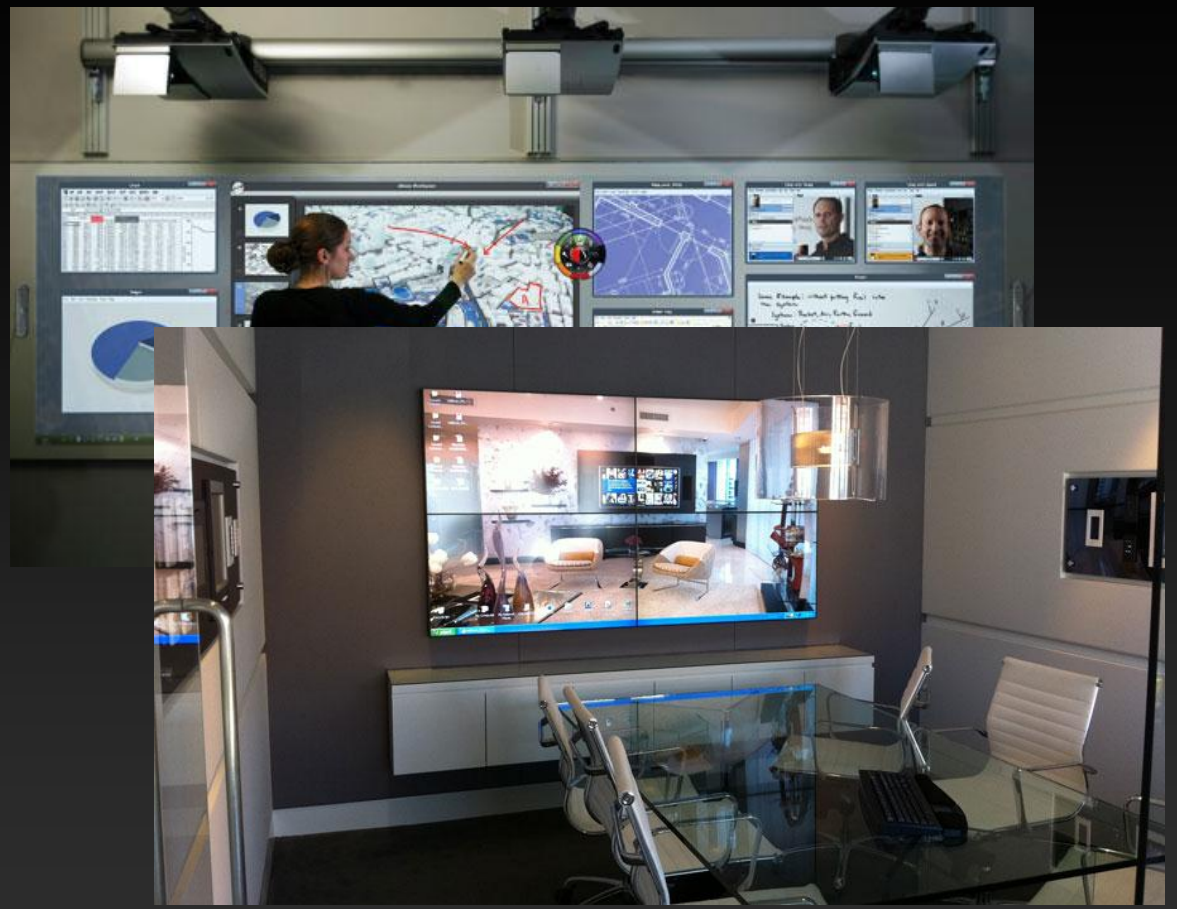

8x MSAA
TXAA
4x MSAA
FXAA
2x MSAA
QUALITY
PERFORMANCE

Kepler Display Capabilities

- 4 Displays per GPU*
- Combines with:
  - Premium Mosaic mode
  - Warp and Blend API
  - GPU Direct for video (low latency I/O)

* Win XP limited to 2 displays

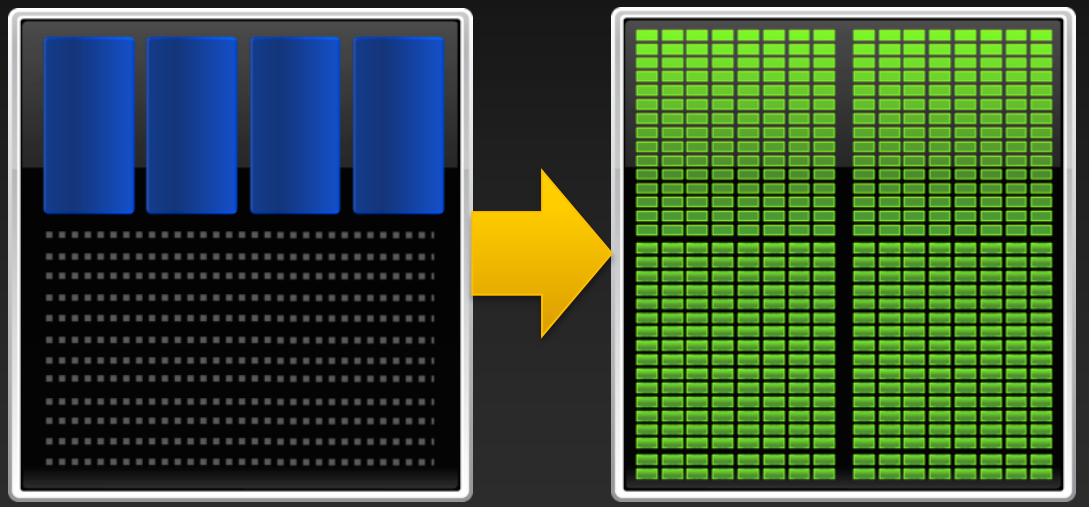
New Options for Collaborative Displays



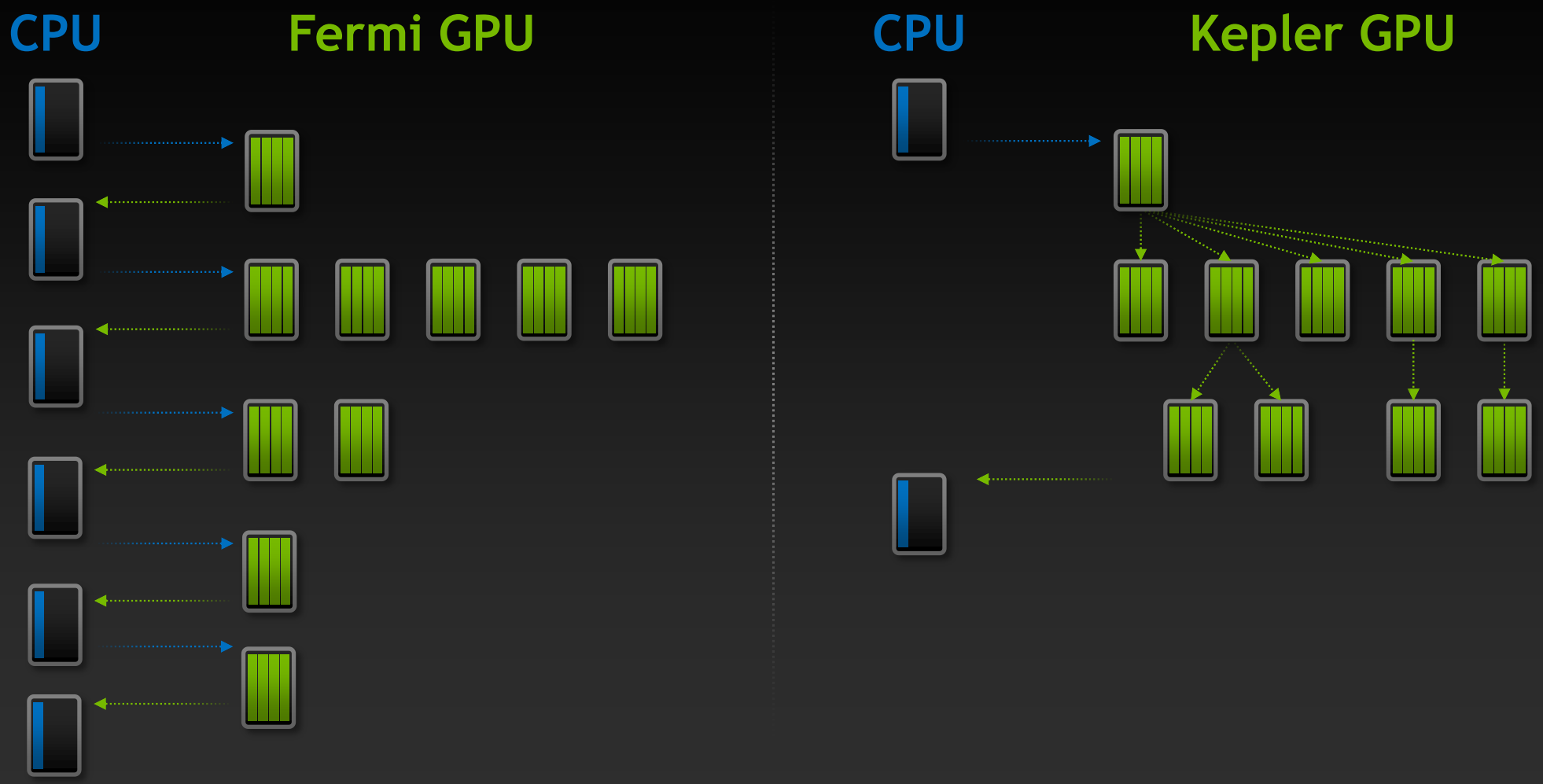
Including life Size Retina Displays

Development Trends…

- More non-graphics developers are moving code to the GPU
- More graphics and professional application developers are mixing Compute and OpenGL
  - Computational graphics
  - Finite element analysis
  - Volume rendering
  - Tessellation
  - Ray tracing
  - Simulation
  - …

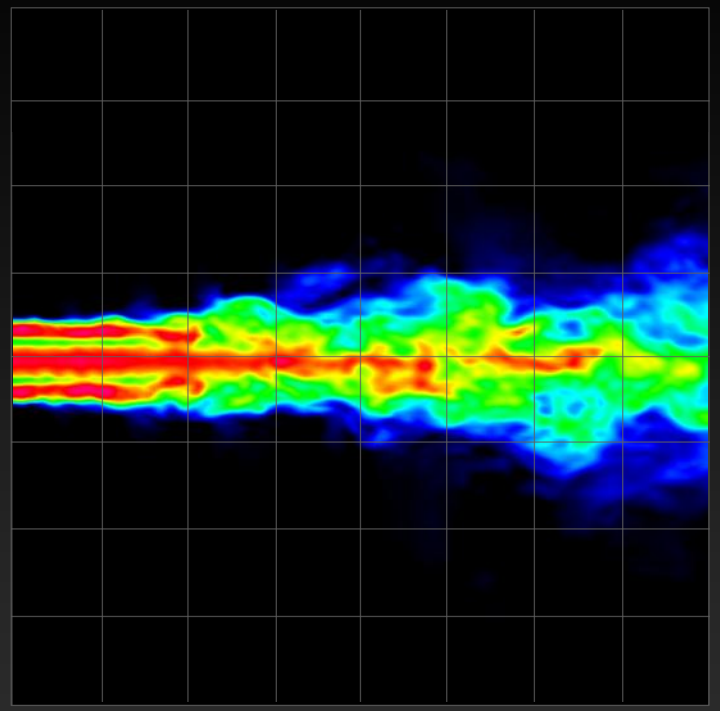


Dynamic Parallelism



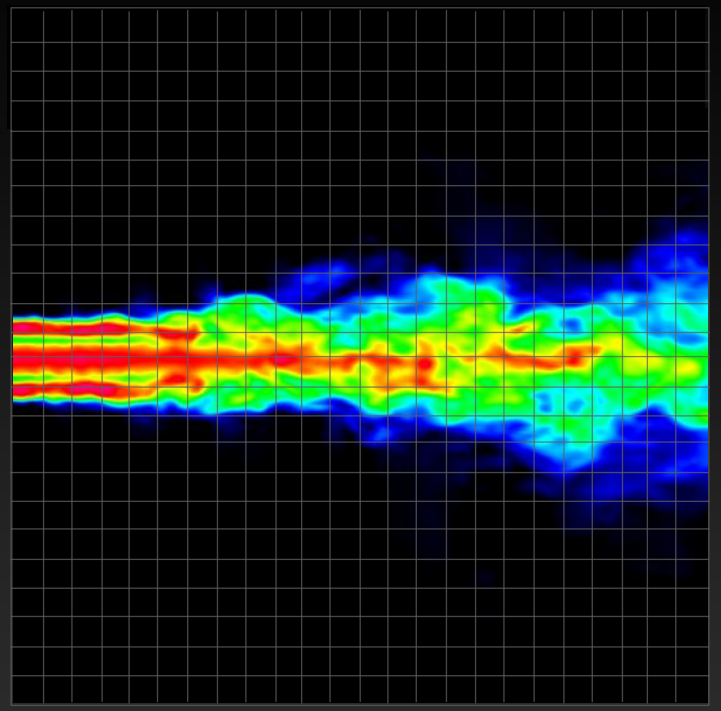
Dynamic Work Generation

Coarse grid



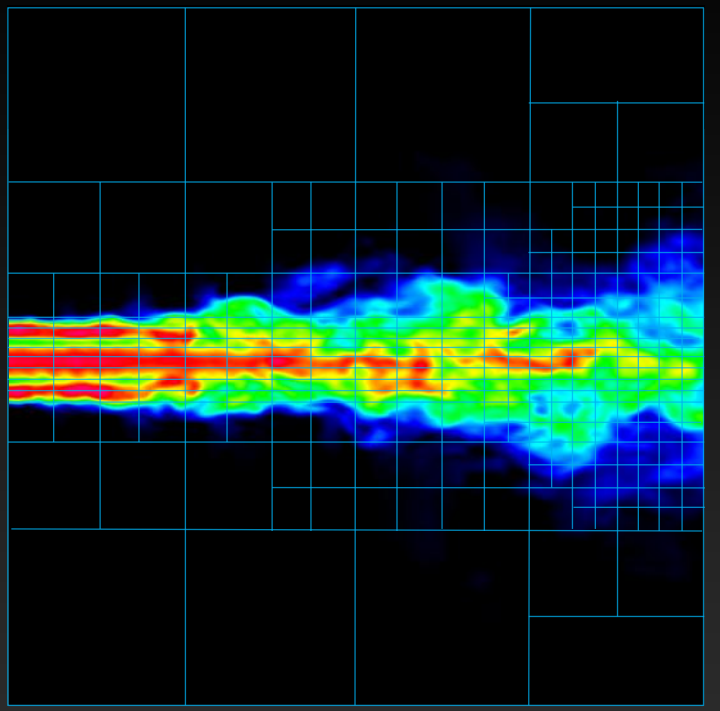
Higher Performance
Lower Accuracy

Fine grid



Lower Performance
Higher Accuracy

Dynamic grid



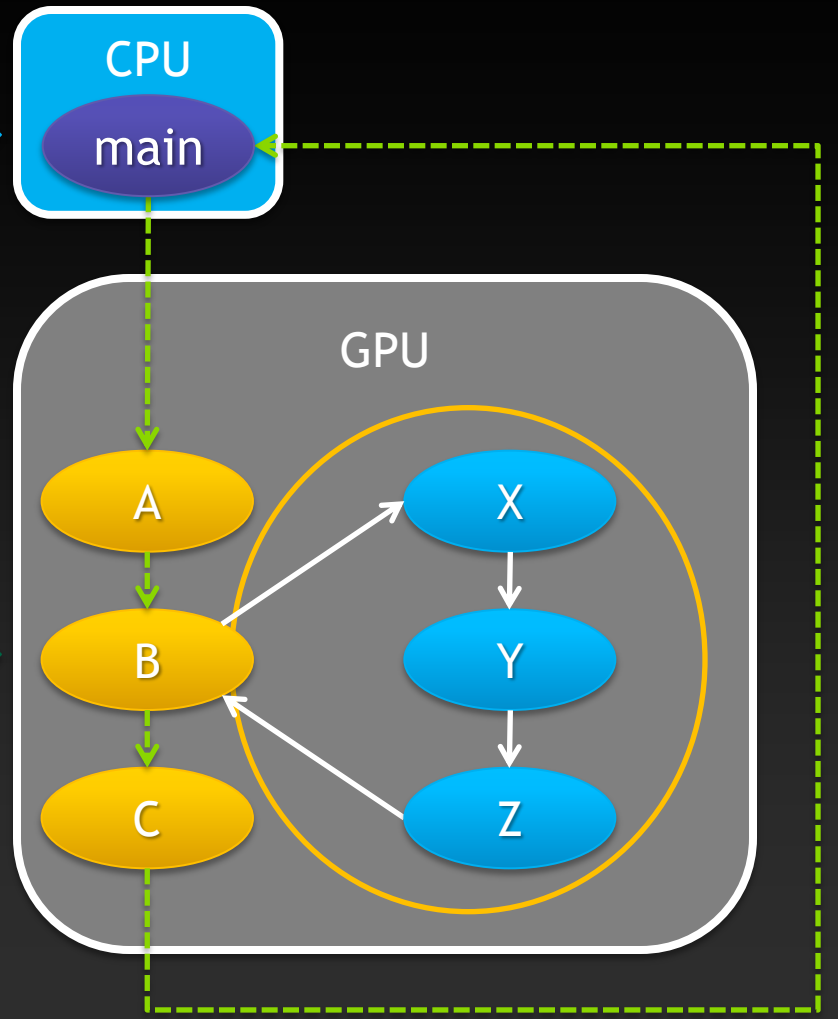
Target performance where accuracy is required

Supported on GK110 GPUs

Familiar Syntax and Programming Model

```
int main() {
    float *data;
    setup(data);

    A <<< ... >>> (data);
    B <<< ... >>> (data);
    C <<< ... >>> (data);

    cudaDeviceSynchronize();
    return 0;
}
```

```
__global__ void B(float *data)
{
    do_stuff(data);

    X <<< ... >>> (data);
    Y <<< ... >>> (data);
    Z <<< ... >>> (data);
    cudaDeviceSynchronize();

    do_more_stuff(data);
}
```



Maximus for Developer

- Visualization + Compute
  - Eg 4D time varying
- Best characterized by:
  - What features did you leave on the table because you didn’t have enough compute power?
  - What would you have implemented if you had a dedicated teraflop of compute power?

Programming for Compute and Graphics

- Check out the Tech Talk:

  9:00 AM - 10:30 AM GPU Programming for High-Performance Graphics Workstation Applications

  Alina Alt, Senior Applied Engineer, NVIDIA
  Wil Braithwaite, Senior Applied Engineer, NVIDIA
  Shalini Venkataraman, Senior Applied Engineer, NVIDIA

OpenGL 4.3

- OpenGL 4.3 is truly super set of DX 11
- Introduces the OpenGL GLSL Compute Language
- Check out the Tech Talk:

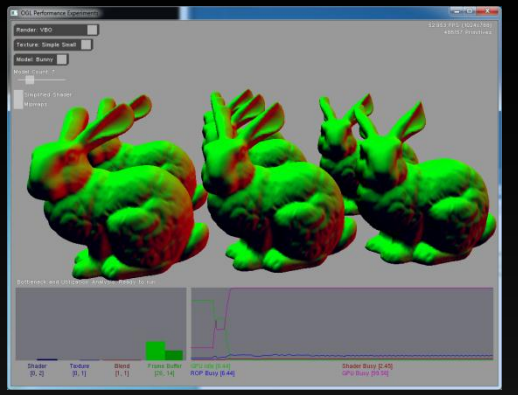
  11:50 AM - 12:50 PM NVIDIA OpenGL in 2012
  Mark Kilgard, Principal Software Engineer, NVIDIA

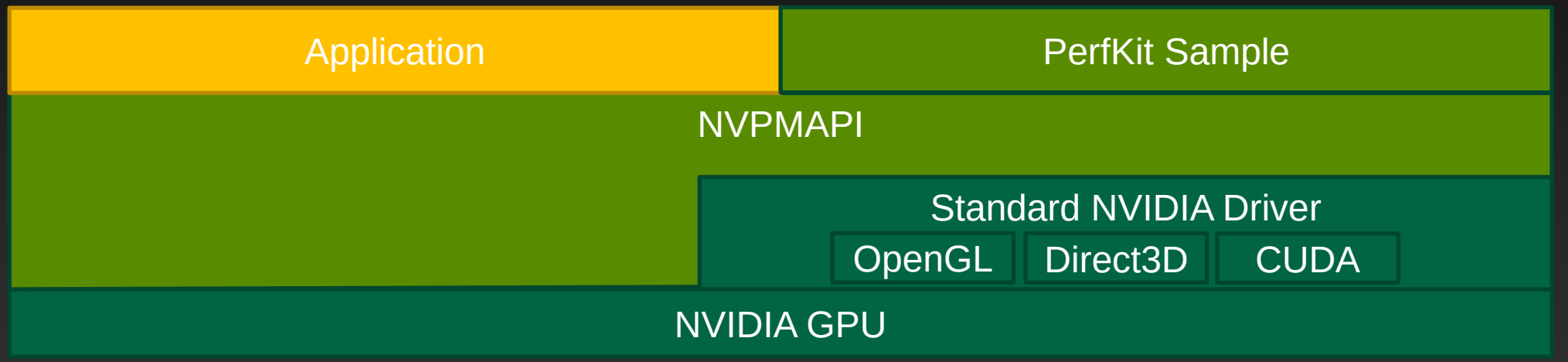
NVWMI

- NVIDIA Windows Management Instrumentation infrastructure
- Exposes GPU information and configuration to network control
- Installs with every NVIDIA driver on Quadro GPUs
  - Option during install

New NVIDIA® PerfKit

- GPU and Software Performance Counter API
  - Performance Monitoring
  - Automated bottleneck analysis
  - Graphics and Compute
- Support Kepler and Fermi Achitecture
- Available on Windows (Linux release very soon), no special driver needed

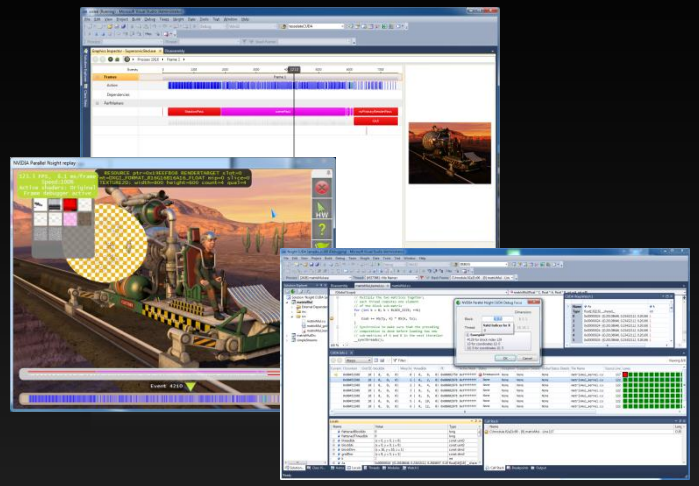




http://www.nvidia.com/object/nvperfkit_home.html

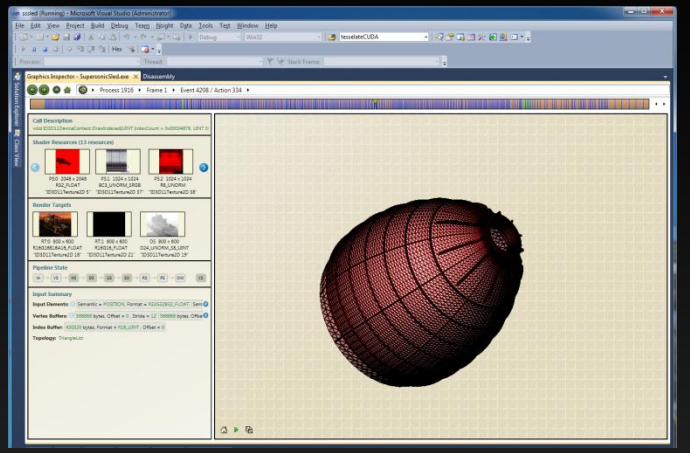
NVIDIA® Nsight™

Microsoft® Visual Studio®



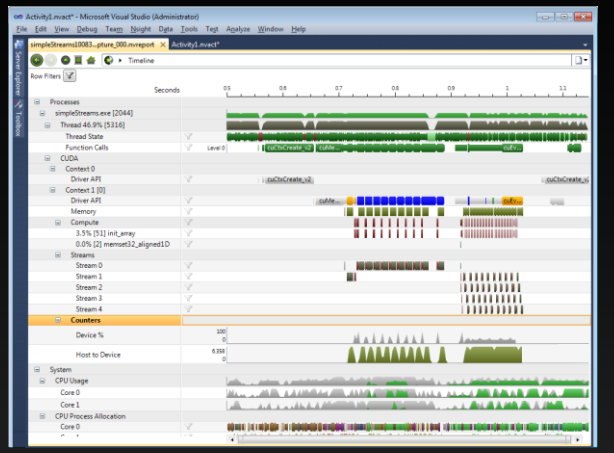
GPU Debugger

- GPU native Compute and Graphics debugging
- GPU breakpoints including complex conditionals
- GPU memory views and exception reporting
- Dynamic Shader Editing



Graphics Inspector

- Real-time inspection of 3D API calls
- Investigate GPU pipeline states
- See contributing fragments with Pixel History
- Profile frames to find GPU bottlenecks



System Analysis

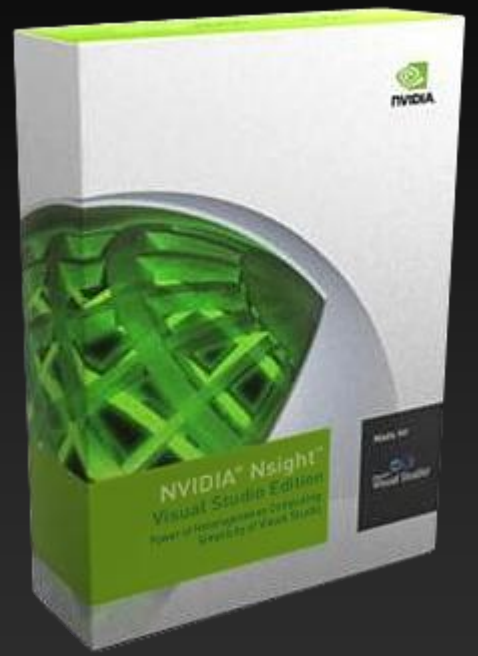
- View CPU & GPU events on a single timeline
- Examine workload dependencies, memory transfers
- CPU/OS, Compute and Graphics APIs Trace
- Capture call stack and jump to source

Nsight™ Visual Studio Edition

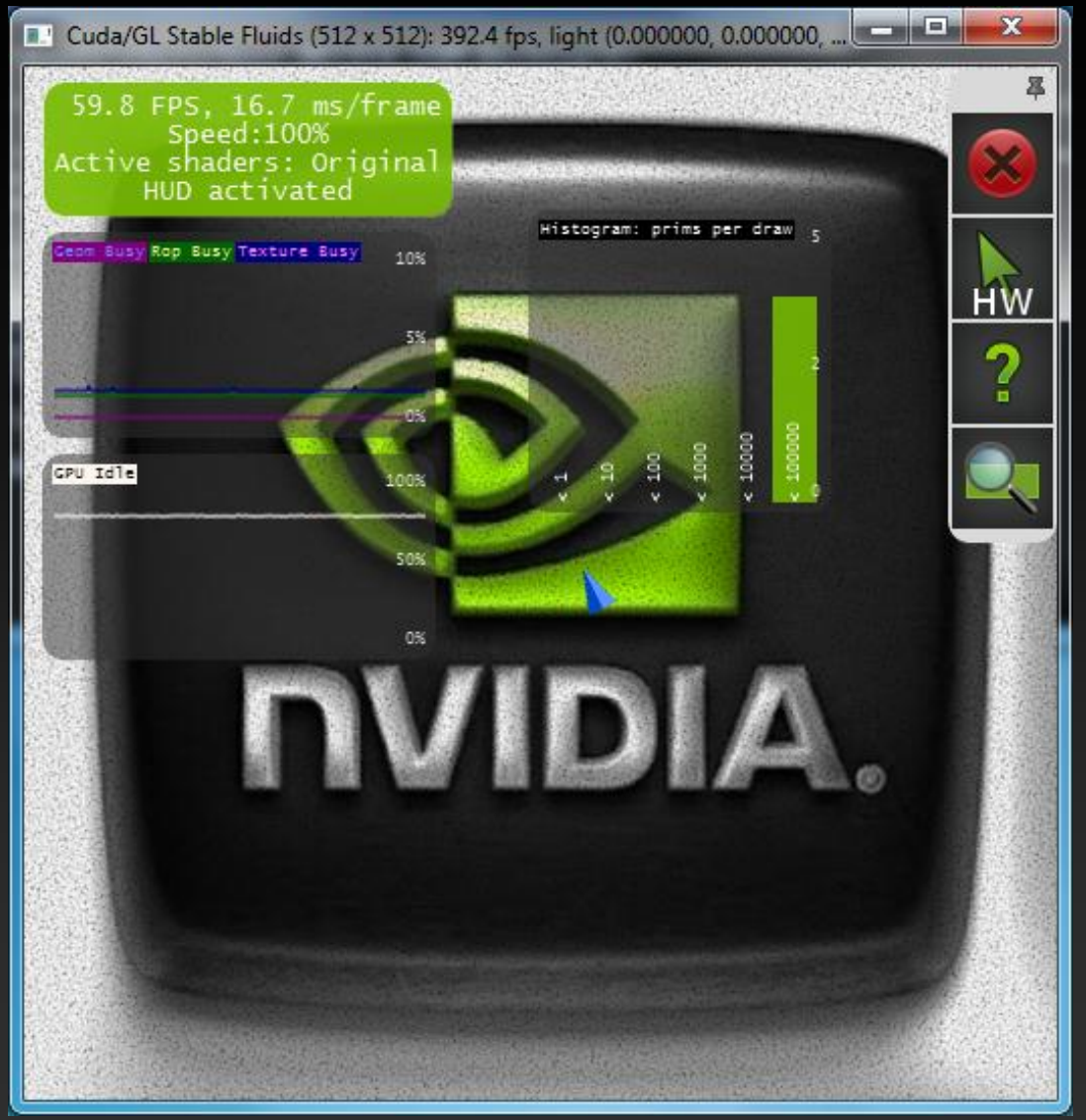
Release Codename "Hulk"

Perfect Maximus development companion

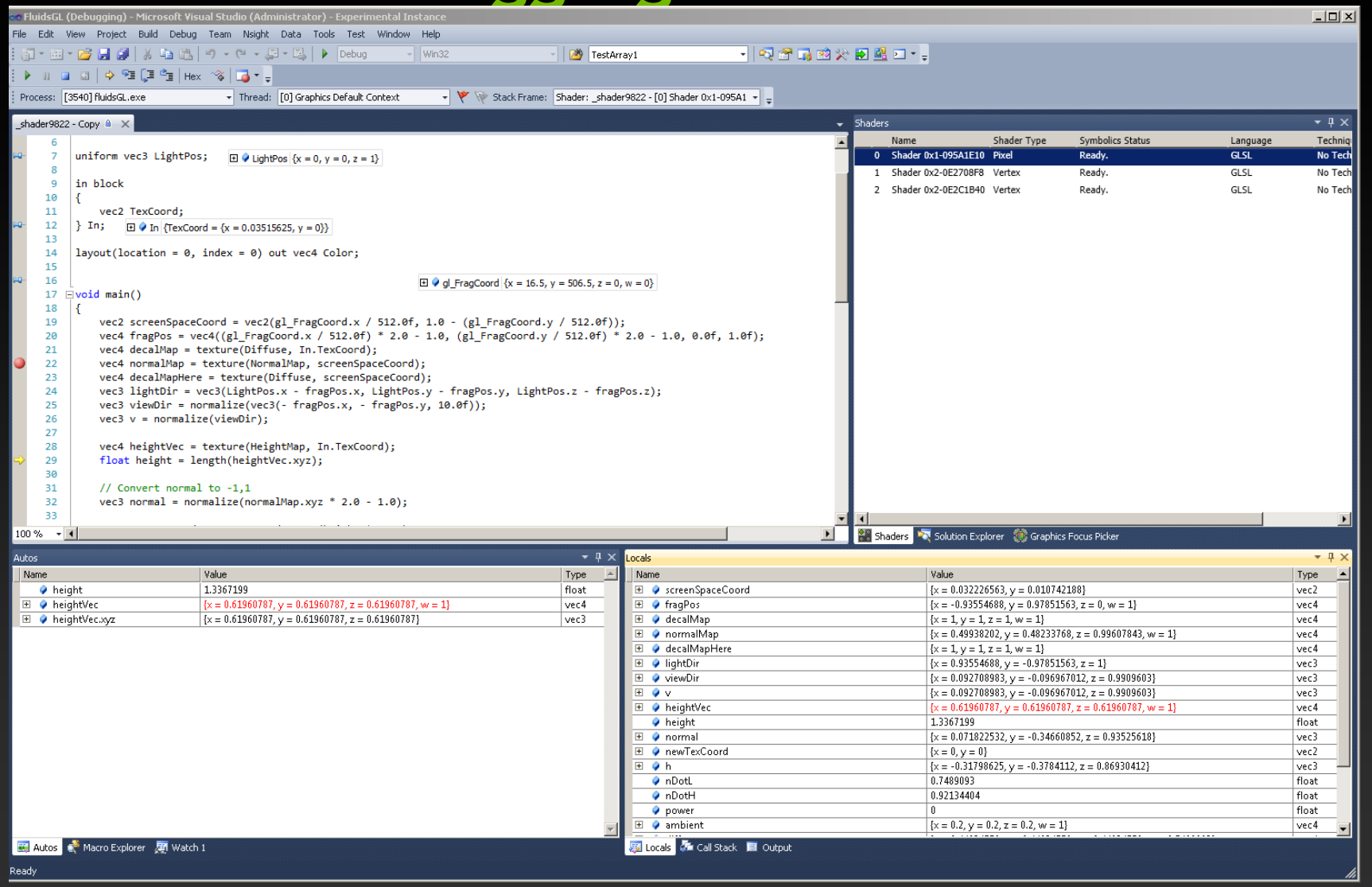
- Full support for CUDA® 5.0 and Kepler II (GK110)
- Full support for modern OpenGL (4.2 Core only for now)
- Debug CUDA and Graphics in the same session
- Single GPU Debugging for Graphics and Compute
- Advanced source-code-correlated CUDA® profiling



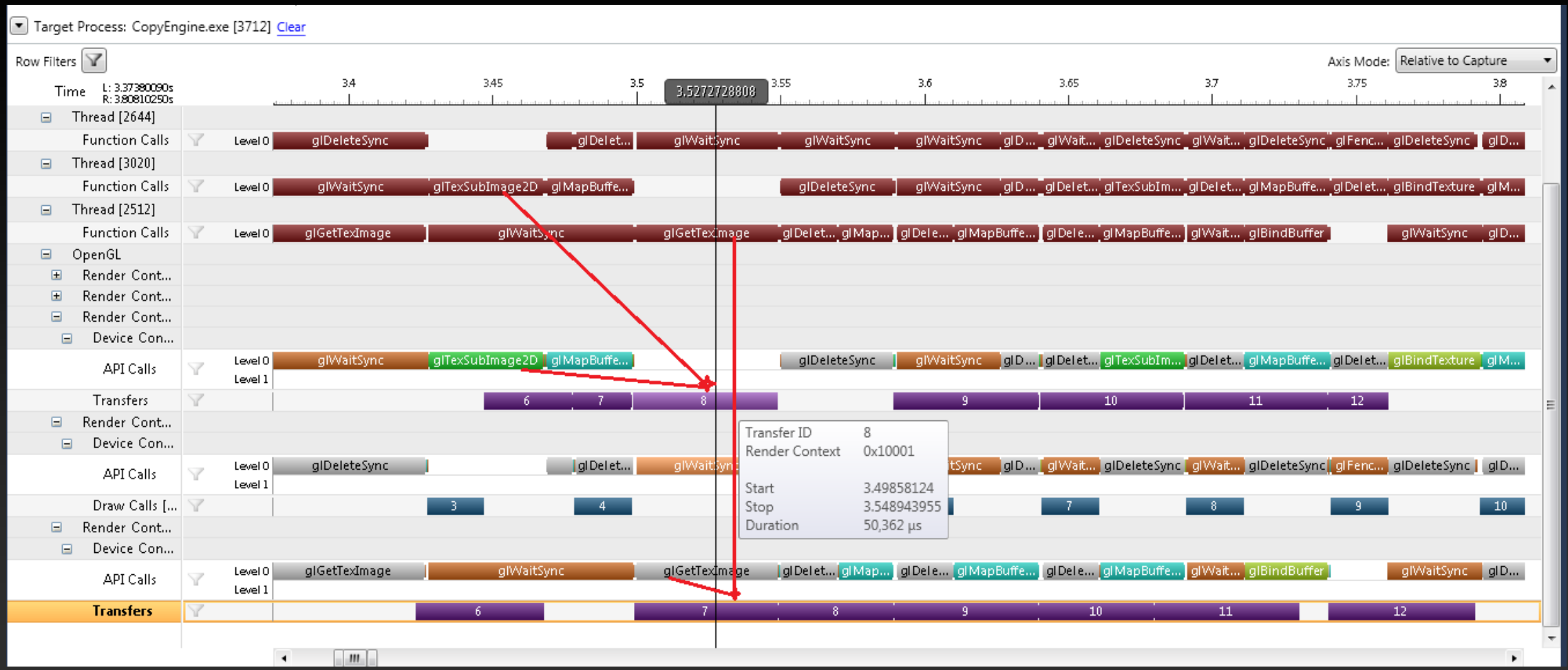
Nsight OpenGL HUD



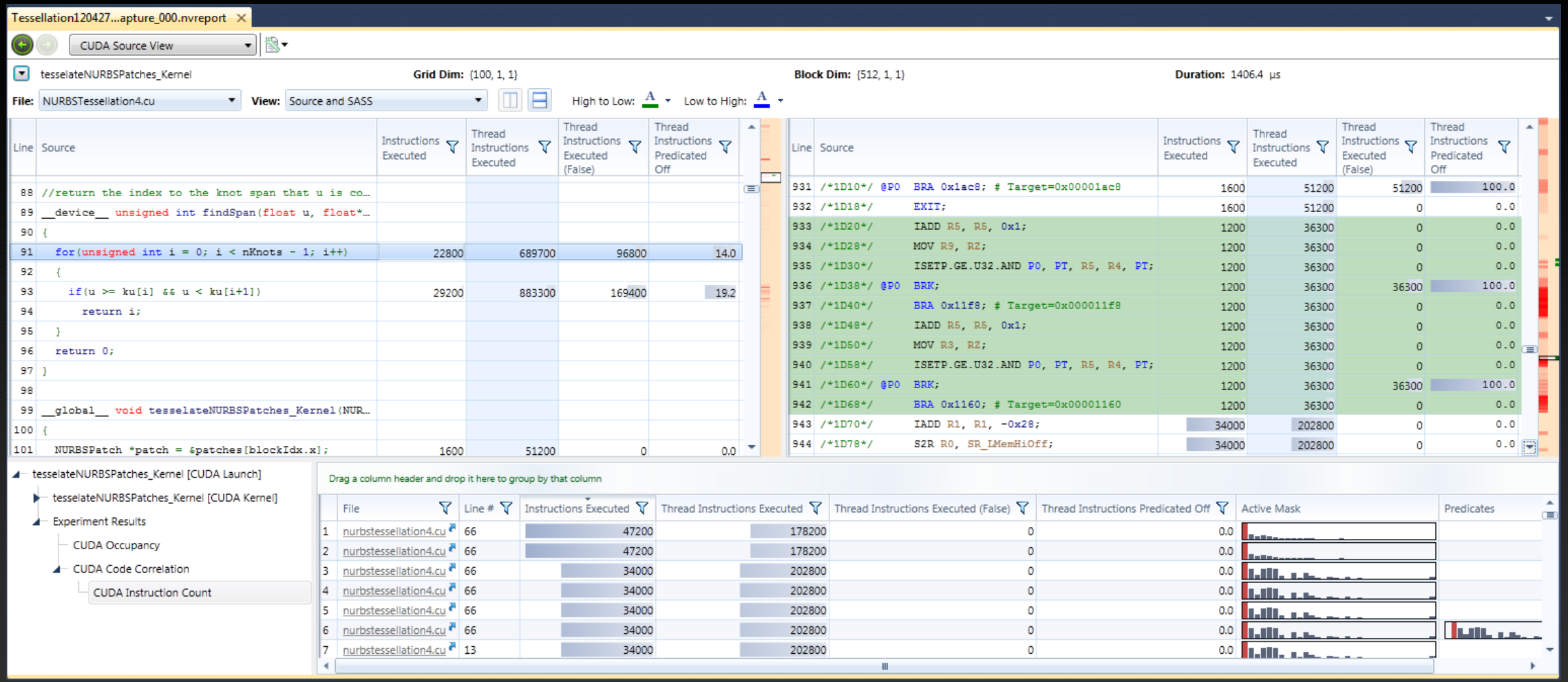
GLSL Shader Debugging



Trace OpenGL async memory transfers



Hulk Source Code Correlation (WIP)



OpenGL Insight

- Full debugging for OpenGL & Compute on a single GPU, including GLSL shaders
  - Shader break points!
- Check out the Tech Talk:

  10:40 AM - 11:40 AM Enabling the Next-Generation of Computational Graphics with Nsight Visual Studio Edition
  Jeff Kiel, Manager, Graphics Related Developer Tools, NVIDIA

Summary

- Kepler architecture offers some significantly improved architectural efficiency
- Many new features and capabilities for developers
- Graphics combined with compute becoming increasingly prevalent
- Powerful new developer tools to compliment Kepler architecture