Large Scale Visualization
Ian Williams & Steve Nash, PSG Applied Engineering
Agenda

• Intro - Quadro Solutions
• High Resolution & HDR Displays and Implications
• Stereo
• HDR
• Implications of Multiple display channels
• Addressing Multiple GPUs
• SLI Mosaic mode
• Combining Technologies
Quadro Visual Computing Platform

- SceniX Scene Graph
- C CUDA OpenCL
- SLI Mosaic Mode
- SLI Multi OS
- 30-bit Color
- mental ray
- reality server
- PhysX
- CompleX Multi-GPU
- OptiX Interactive Ray Tracing

NVIDIA SLI
NVIDIA CUDA
NVIDIA G-Sync
NVIDIA HD SDI
NVIDIA Quadro Plex VCS
# Quadro FX Family

<table>
<thead>
<tr>
<th>Product Segment</th>
<th>Target Audience</th>
<th>Key Additional Features</th>
<th>Quadro Solution</th>
<th>Estimated Street Price</th>
</tr>
</thead>
</table>
| Ultra High-End  | 4D Seismic Analysis  
4D Medical Imaging | + 4GB GPU Memory 
+ 240 CUDA Parallel Cores | Quadro FX 5800 | $ 3,299 |
| High-End        | Digital Special Effects  
Product Styling | + G-Sync  
+ SLI Frame Rendering | Quadro FX 4800 | $ 1,799 |
| High-End        | High End MCAD  
Digital Effects  
Broadcast | + SDI  
+ Stereo  
+ SLI Multi-OS | Quadro FX 3800 | $ 899 |
| Mid-Range       | Midrange CAD  
Midrange DCC | +25% better Perf than FX 580 | Quadro FX 1800 | $ 599 |
| Entry           | Volume CAD  
Volume DCC | +30% better performance than FX 380  
+ 30-bit Color | Quadro FX 580 | $ 149 |
| Entry           | Volume CAD  
Volume DCC  
Productivity Apps | +50% Better Performance than FX 370 | Quadro FX 380 | $ 99 |
# Quadro Systems

<table>
<thead>
<tr>
<th>Product Segment</th>
<th>Target Audience</th>
<th>Key Additional Features</th>
<th>Quadro Solution</th>
<th>Estimated Street Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>1U Rackmount</td>
<td>Offline &amp; Remote Rendering</td>
<td>Four GPUs 16 GB total GPU Memory</td>
<td>Tesla S1070</td>
<td>$ 9,000</td>
</tr>
<tr>
<td>Deskside or 3U Rackable</td>
<td>Seismic Analysis Product Styling Scalable Graphics</td>
<td>2 GPUs SLI Mosaic Mode - Easy 4K CompleX OptiX</td>
<td>QuadroPlex 2200 D2</td>
<td>$ 10,750</td>
</tr>
</tbody>
</table>
AXE - Engine Relationships

SceniX scene management engine

CgFX API
QB Stereo API
30-bit & SDI API

OptiX ray tracing engine
Complex scene scaling engine
Non-Graphic Applications
Custom Applications
Open SceneGraph

AXE Center
AXE Flexibility
AXE Reach
Application Acceleration Engines - Overview

**SceniX** – scene management engine
- High performance OpenGL scene graph built around CgFX for maximum interactive quality
- Provides ready access to new GPU capabilities & engines

**CompleX** – scene scaling engine
- Distributed GPU rendering for keeping complex scenes interactive as they exceed frame buffer limits
- Direct support for SceniX, OpenSceneGraph, and soon more

**OptiX** – ray tracing engine
- Programmable GPU ray tracing pipeline that greatly accelerates general ray tracing tasks
- Supports programmable surfaces and custom ray data
Why use Large Scale Visualization?

- Quality
- Detail
- Pixel real estate
- Stereo
- Immersive experience
- Industry specific needs
- .......
Display Technologies

• Panels
  • Industry focused - e.g. medical, video

• Projectors

• Multiple Panels

• Multiple Projectors

Images courtesy of HP, Sony, Barco, Mechdyne,
Large Scale Visualization

Beyond 8 DVI Dual Link Requires Clustered PCs with Quadro G-Sync to synchronize displays and Multi GPU aware software.

Applications written to run on a single display just work across larger display formats.
Implications of High Resolution and HDR

• Performance
• Stereo
• “Mechanics” of >8bit per component
• Multiple display channels
  • OS impact
  • Synchronization
  • Clustering
Performance Implications of High resolutions & HDR

• GPU memory
  • 3840x2160 desktop at 16x FSAA ~400MB of framebuffer.

• Performance
  • Fill-rate
  • Window system implications

• Texture size & depth
  • 16 bit per component

• .........
Stereo

• Consumer Stereo Drivers (3DVision)
  • Stereo separation from single stream
• OpenGL Quad Buffered Stereo
  • Application has explicit control of the stereo image
• Active
• Passive
“Mechanics” of >8bit per component

• Possible using both DVI or Display Port
  • Display Port much easier

• Textures etc. need to be >8bit per component
  • FP16, I16 (G8x GPUs and beyond)
  • RGBA, LA, L
Implementing HDR over DVI

• Full screen only
  • Desktop, GUI, etc will not be correctly displayed
• Format specific to display device
• Outline:
  • Configure double-wide desktop
    • Significantly easier if exported by the EDID
  • Create full-screen window
  • Render to off-screen context
    • E.g. OpenGL FBO
  • Draw a textured quad
    • Use fragment program to pack pixels - display device specific
    • G8x GPUs (and beyond) implement bitwise fragment operations as well as fixed point textures
Implementing HDR over DVI - cont

Specific packing format is Display Specific
Implementing HDR over DVI
- cont

Specific packing format is Display Specific

- Off-screen buffer
  - 16 bit per component
  - 8 bits
  - 2 bits

- Full-Screen Window
  - 8 bit per component
HDR and Display Port

- Requires native Display Port GPU

- Desktop will be display correctly (in 8bit)

- Outline:
  - Open 10bit per component Pixel Format/Visual
  - Render
Multiple Display Channels

Why multiple display channels?

- Resolutions becoming larger than channel bandwidths
  - Sony, JVC 4K projectors
  - Barco and Mitsubishi panels
  - 

Implications of Multiple Display Channels

First a couple of questions:

• Which OS - Windows or Linux?
• Level of application transparency:
  • Driver does everything?
  • Application willing to do some work?
Multiple Displays - Windows

- Attach Multiple Monitors using Display Properties
- Extend the Desktop to each GPU
- Ensure ordering is correct for desired layout
- Adjust Resolutions and Refresh Rates
- Displays using Refresh Rates <48Hz can be problematic
- Synchronizing displays requires G-sync card
Multiple Displays - Windows
Multiple Displays - Windows

Things you don’t intend are also possible
Multiple Displays - Windows

Things to note:

• Windows can be opened anywhere on (and off) the complete desktop
• Windows can span display boundaries
• However maximizing will lock to one display
  • Where the window centroid is located
• Likewise full screen windows
• WGL Desktop size is considered outer rectangle spanning all displays
• Driver will typically send data to all GPUs (in case window is moved, etc.)
  • GPU Affinity OpenGL extension solves this
Multiple Displays - Windows

```c
DISPLAY_DEVICE lDispDev;
DEVMODE lDevMode;
lDispDev.cb = sizeof(DISPLAY_DEVICE);

if (EnumDisplayDevices(NULL, 0, &lDispDev, NULL)) {
    EnumDisplaySettings(lDispDev.DeviceName, ENUM_CURRENT_SETTINGS, &lDevMode);
}
g_hWnd1 = createWindow(hInstance, lDevMode.dmPosition.x, lDevMode.dmPosition.y, X0, Y0);

if (!g_hWnd1) {
    MessageBox(NULL, "Unable to create first window(s).", "Error", MB_OK); return E_FAIL;
}

if (EnumDisplayDevices(NULL, 1, &lDispDev, NULL)) {
    EnumDisplaySettings(lDispDev.DeviceName, ENUM_CURRENT_SETTINGS, &lDevMode);
}
g_hWnd2 = createWindow(hInstance, lDevMode.dmPosition.x, lDevMode.dmPosition.y, X1, Y1);
if (!g_hWnd2) {
    MessageBox(NULL, "Unable to create second window(s).", "Error", MB_OK); return E_FAIL;
}
```

Verify first display exists and get display settings

Create Window on first display

Verify second display exists and get display settings

Create Window on second display
Addressing Multiple GPUs on Windows

GPU Affinity

- WGL extension (WGL_NV_gpu_affinity), core OpenGL not touched
  - GLX definition in the works

- Application creates affinity-DC
  - HDC wglCreateAffinityDCNV(const HGPUNV *phGpuList);
  - Special DC that contain list of valid GPUs -> affinity mask
  - Affinity mask is immutable

- Application creates affinity context from affinity-DC
  - As usual with RC = wglCreateContext(affinityDC);
  - Context inherits affinity-mask from affinity-DC

- Application makes affinity context current
  - As usual using wglMakeCurrent()
  - Context will allow rendering only to GPU(s) in its affinity-mask
Addressing Multiple GPUs on Windows cont.

GPU Affinity

- Affinity context can be made current to:
  - Affinity DC
    - Affinity mask in DC and context have to be the same
    - There is no window associated with affinity-DC. Therefore:
      - Render to pBuffer
      - Render to FBO
  - DC obtained from window (regular DC)
    - Rendering only happens to the sub-rectangle(s) of the window that overlap the parts of the desktop that are displayed by the GPU(s) in the affinity mask of the context.

- Sharing OpenGL objects across affinity contexts only allowed if affinity mask is the same
  - Otherwise wglShareLists will fail
Addressing Multiple GPUs on Windows cont.

GPU Affinity

• Enumerate all GPUs in a system
  • BOOL wglEnumGpusNV(int iGpuIndex, HGPUNV *phGpu);
  • Loop until function returns false

• Enumerate all display devices attached to a GPU
  • BOOL wglEnumGpuDevicesNV(HGPUNV hGpu, int iDeviceIndex, PGPU_DEVICE lpGpuDevice);
  • Returns information like location in virtual screen space
  • Loop until function returns false

• Query list of GPUs in an affinity-mask
  • BOOL wglEnumGpusFromAffinityDCNV(HDC hAffinityDC, int iGpuIndex, HGPUNV *hGpu);
  • Loop until function returns false

• Delete an affinity-DC
  • BOOL wglDeleteDCNV(HDC hdc);
Addressing Multiple GPUs cont.

GPU Affinity – Render to Off-screen FBO

```c
#define MAX_GPU 4
int gpuIndex = 0;
HGPUNV hGPU[MAX_GPU];
HGPUNV GpuMask[MAX_GPU];
HDC affDC;
HGLRC affRC;
while ((gpuIndex < MAX_GPU) && wglEnumGpusNV(gpuIndex, &hGPU[gpuIndex])) {
    gpuIndex++;
}
GpuMask[0] = hGPU[0];
GpuMask[1] = NULL;
affDC = wglCreateAffinityDCNV(GpuMask);
<Set pixelformat on affDC>
affRC = wglCreateContext(affDC);
wglMakeCurrent(affDC, affRC);
<Create a FBO>
glBindFramebufferEXT(GL_FRAMEBUFFER_EXT, b);
<now render>
```

Create list of the first MAX_GPUs in the system

Create an affinity-DC associated with first GPU

Make the FBO current to render into it
Multiple Displays - Linux

• Two traditional approaches depending on desired level of application transparency or behavior:
  • Separate X screens
    • 3D Windows can’t span X screen boundaries
    • Location of context on GPU allows driver to send data to only that GPU
  • Xinerama
    • One large virtual desktop
    • 3D Windows can span X screen boundaries
    • Will typically result in driver sending all data to all GPUs (in case window moves)
Multiple Displays - Linux

• Use nvidia-xconfig to create customized xorg.conf

• nvidia-settings provides full featured control panel for Linux

• Drivers can capture EDID
  • Useful when display device hidden behind KVM or optical cable

• Synchronizing multiple displays requires G-sync card
Synchronizing Multiple Displays

• Requires G-sync
  • Synchronize vertical retrace
  • Synchronize stereo field
  • Enables swap barrier

• OpenGL Extensions
  • Windows: WGL_NV_Swap_Group
  • Linux: GLX_NV_Swap_Group
WGL_NV_Swap_Group

Name

NV_swap_group

Dependencies

WGL_EXT_swap_control affects the definition of this extension.
WGL_EXT_swap_frame_lock affects the definition of this extension.

Overview

This extension provides the capability to synchronize the buffer swaps of a group of OpenGL windows. A swap group is created, and windows are added as members to the swap group. Buffer swaps to members of the swap group will then take place concurrently.

This extension also provides the capability to synchronize the buffer swaps of different swap groups, which may reside on distributed systems on a network. For this purpose swap groups can be bound to a swap barrier.

This extension extends the set of conditions that must be met before a buffer swap can take place.

BOOL wglJoinSwapGroupNV(HDC hDC, GLuint group);
BOOL wglBindSwapBarrierNV(GLuint group, GLuint barrier);
BOOL wglQuerySwapGroupNV(HDC hDC, GLuint *group);
        GLuint *maxGroups, 
        GLuint *maxBarriers);
BOOL wglQueryMaxSwapGroupsNV(HDC hDC, 
        GLuint *count);
BOOL wglResetFrameCountNV(HDC hDC);
GLX_NV_swap_group

Name

NV_swap_group

Overview

This extension provides the capability to synchronize the buffer swaps of a group of OpenGL windows. A swap group is created, and windows are added as members to the swap group. Buffer swaps to members of the swap group will then take place concurrently.

This extension also provides the capability to synchronize the buffer swaps of different swap groups, which may reside on distributed systems on a network. For this purpose swap groups can be bound to a swap barrier.

This extension extends the set of conditions that must be met before a buffer swap can take place.

Bool glxJoinSwapGroupNV(Display *dpy, GLXDrawable drawable, GLuint group);

Bool glxBindSwapBarrierNV(Display *dpy, GLuint group, GLuint barrier);

Bool glxQuerySwapGroupNV(Display *dpy, GLXDrawable drawable, GLuint *group);

GLuint *barrier);

Bool glxQueryMaxSwapGroupsNV(Display *dpy, GLXDrawable drawable, GLuint *group);

GLuint *maxBarriers);

Bool glxQueryFrameCountNV(Display *dpy, GLuint screen, GLuint *count);

Bool glxResetFrameCountNV(Display *dpy);
Using G-sync

Recommendations:

• Control Panel will cause regular CPU contention
  • Polls hardware status

• Use additional synchronization mechanisms in addition to swapbarrier
  – Broadcast frame count
SLI Mosaic Mode

Multiple Displays made easy!

• Enables transparent use of multiple GPUs on multiple displays
  • Enables a Quadro Plex (multiple GPUs) to be seen as one logical GPU by the operating system
  • Applications ‘just work’ across multi GPUs and multi displays
  • Works with OGL, DX, GDI etc

• Zero or minimal performance impact for 2D and 3D applications compared with a single GPU per single display

• Doesn’t support multiple View Frustums
SLI Mosaic Mode

Details

• Quadro Plex only

• Operating System support
  • Windows XP, Linux, 32bit and 64bit
  • Vista/Win 7 soon

• Maximum desktop size = 8k X 8k
  • FSAA may exacerbate desktop size

• Compatible with G-sync
  • Clustering tiled displays

• Supports Stereo
SLI Mosaic Mode

Configurations
Performance Hit for Multiple Displays

Viewperf 10.0

- 1 screen
- 4 screens
- 8 screens

- 3dsmax-04
- catia-02
- ensight-03
- maya-02
- proe-04
- sw-01
- tcvis-01
- ugnx-01
Programmatically controlling Mosaic Mode

- NvAPI provides direct access to NVIDIA GPUs and drivers on Windows platforms
- Nvidia Control Panel GUI shows tested configurations
- More advanced configuration possible through NvAPI
Programatically controlling Mosaic Mode (cont’d)

```
NV_MOSAIC_TOPOLOGY topo;  // struct defines rowCount, colcount & gpuLayout
NV_MOSAIC_SUPPORTED_TOPOLOGIES supportedTopoInfo;  // list of topologies

// Get List of Supported Topologies and display resolutions
nvStatus = NvAPI_Mosaic_GetSupportedTopoInfo(&supportedTopoInfo, type);

// Set Mosaic Mode for a given topology
nvStatus = NvAPI_SetCurrentMosaicTopology(&topo);

// To Disable Mosaic Mode
nvStatus = NvAPI_EnableCurrentMosaicTopology(0);
```
Beyond SLI Mosaic Mode

- Can combine Mosaic for partial set of all GPUs
- Use CUDA or GPU Affinity for non-display GPUs
- Requires “Manual” Configuration

- Combine Mosaic with CompleX Application Acceleration Engine
Summary

• Demand for Large Scale Viz & HDR technologies are being driven by economics
  • E.g. Digital Prototypes significantly less expensive than physical prototypes however demand high quality and realism

• Very large resolutions are de-facto standard for collaborative and large venue installations

• Pixel bandwidth requirements still require multiple channels, even with Display Port

• Some large venue displays are HDR capable
Summary - cont.

• Be aware of performance implications when using multiple GPUs
  – Use affinity/Separate Xscreens
• Solutions like SLI Mosaic Mode extends the reach of Large Scale Visualization
• Combining solutions enables unprecedented realism and interactivity
  – CompleX + Mosaic = interactive massive datasets
  – OptiX + Mosaic = unprecedented realism on a large scale
  – Compute + viz cluster = flexible utilization with massive compute power
Thank You!

• Feedback & Questions.