HD is now 8MP & HDR

Ian Williams – Manager, PSG Applied Engineering

NVIDIA

Agenda

- Intro – Quadro Solutions
- High Resolution & HDR Displays and Implications
- Stereo
- HDR
- Display Port
- Implications of Multiple display channels
- Addressing Multiple GPUs
- SLI Mosaic mode
- Combining Technologies

Quadro Solutions

Mobile

Desktop

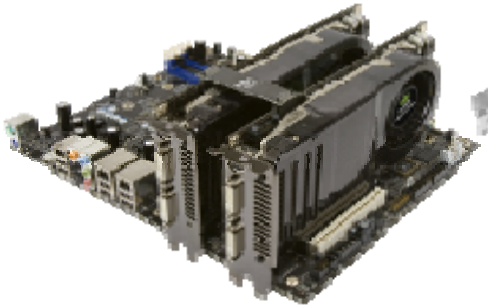
Power Desk Side

Remote Desktop Blades

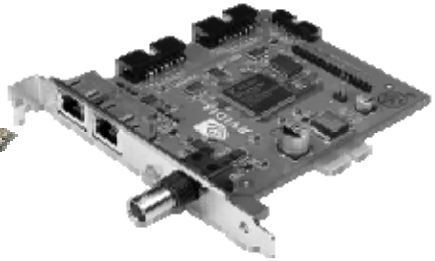
Remote Graphics Servers

Images courtesy of Softimage Co. and Avid Technology Inc., model provided by Acony Games.

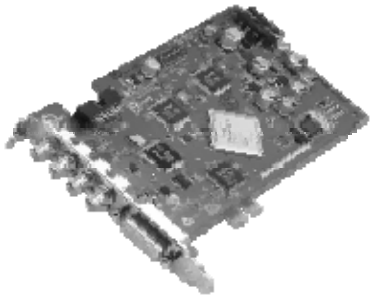
Enabling Industry Solutions



NVIDIA
SLI



NVDIA
G-Sync



NVIDIA
HD SDI

NVIDIA
Quadro Plex VCS

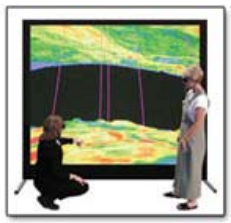
NVISION 08
THE WORLD OF VISUAL COMPUTING

Why use High Resolution & HDR Displays?

- Quality
- Detail
- Pixel real estate
- Collaboration
- Immersive experience
- Industry specific needs
- …….

High Resolution and HDR Display Technologies

- Panels
  - Industry focused – e.g. medical, video
- Projectors
- Multiple Panels
- Multiple Projectors



Images courtesy of HP, Sony, Barco, Mechdyne,

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

- OpenGL Quad Buffered Stereo
  - Application has explicit control of the stereo image
- Consumer Stereo Drivers
  - Infers stereo separation from single stream
- Active
- Passive

L, R, L, R, L, R, ...

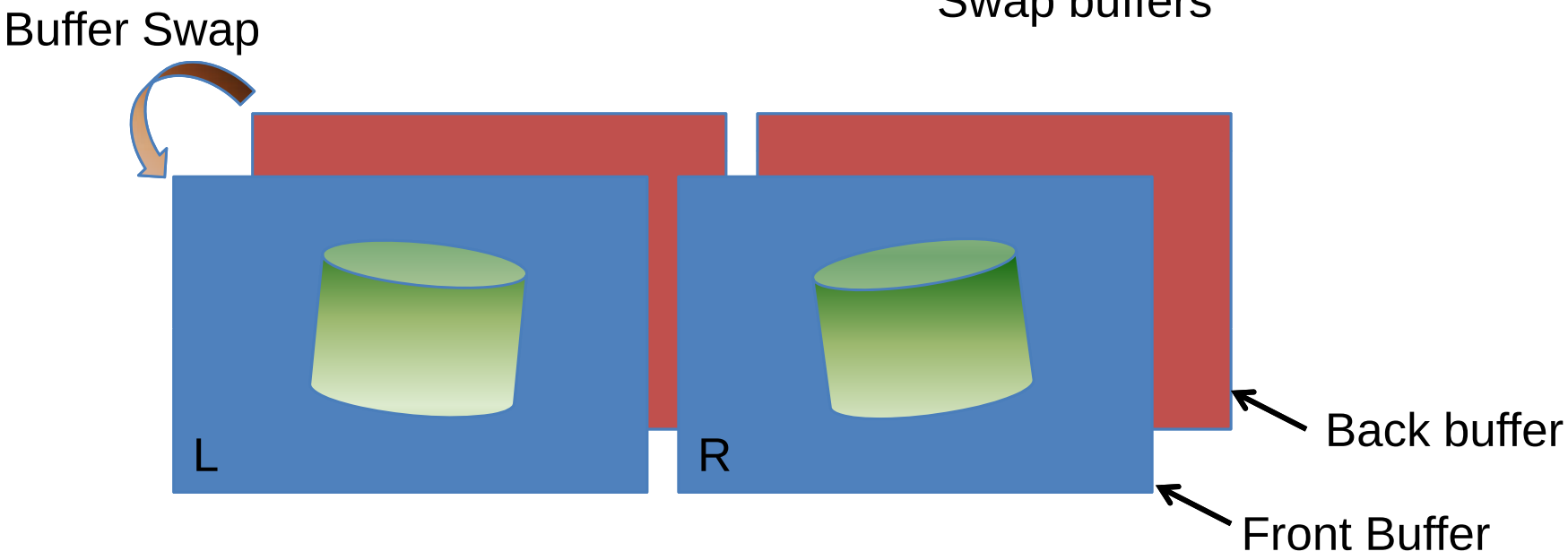
L, L, L, L, L, L, ...

R, R, R, R, R, R, ...

OpenGL QuadBuffered Stereo

Draw Loop:

- Set draw buffer to back left
- Load left eye view Matrix
- Draw Scene
- Set draw buffer to back right
- Load right eye view Matrix
- Draw Scene
- Swap buffers



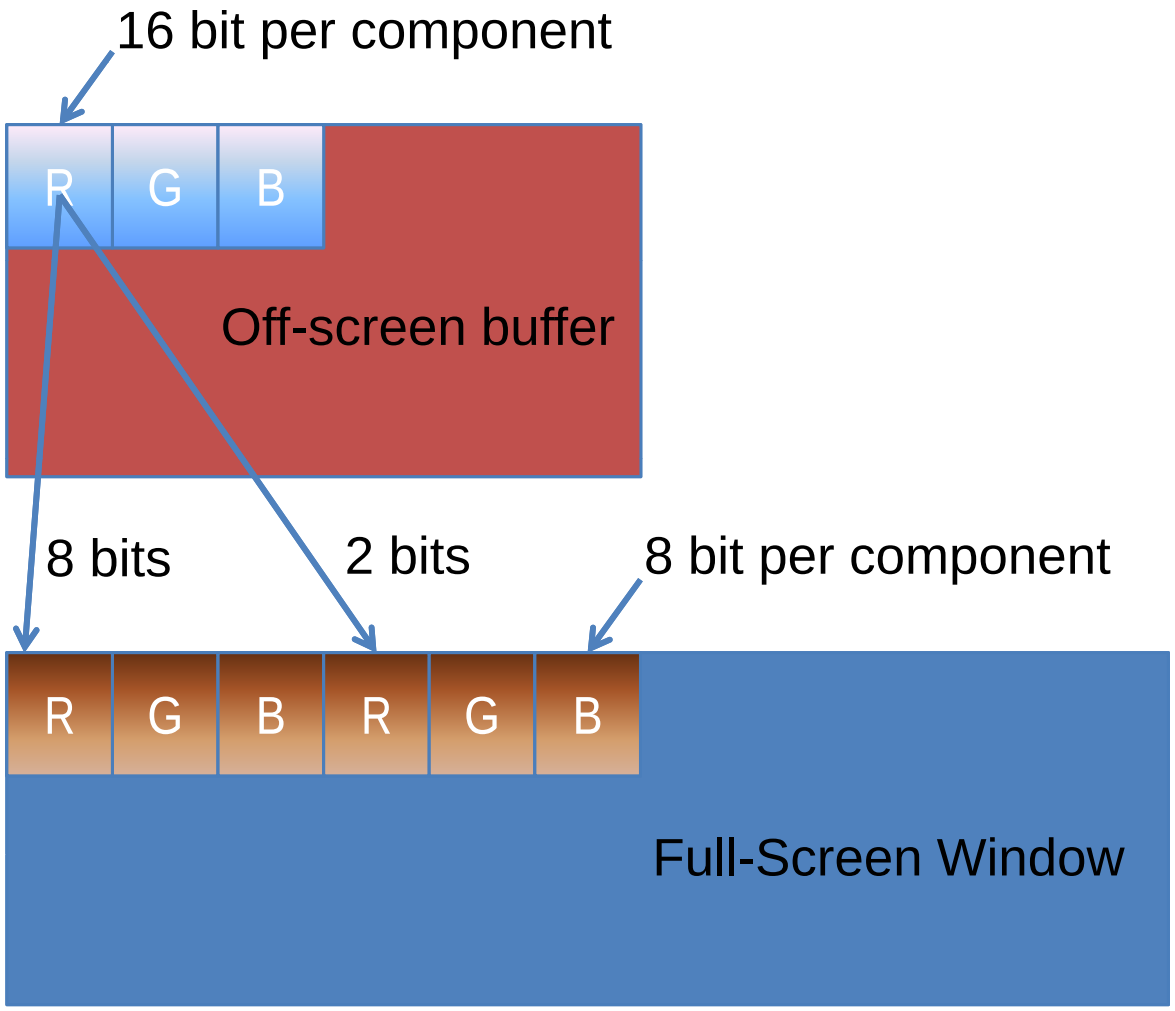
“Mechanics” of >8bit per component

- 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 Display Port

- Requires native Display Port GPU
- Desktop will be display correctly (in 8bit)
- Outline:
  - Open 10bit per component Pixel Format/Visual
  - Render

Display Port

- All lanes carry data (no dedicated clock lane)
- Link rate: either 2.7Gbps or 1.62Gbps per lane, based on link quality
- Flexible number of lanes, 1, 2, or 4, depending on device capability
- Link capacity for 4 lanes, 10.8 Gbit/sec (2.2x bandwidth of DVI for an equal number of wires)
- Freely trade pixel depth with resolution and frame rate

Display Port
- Cont.

- 1Mbit/sec bi-directional auxiliary channel
- Low power and low EMI
- Support for long cables (15m) and a latching connector
- Full support of a variety of audio formats as optional feature
- Robust content protection system as optional feature

Display Port Compared to DVI

** Reproduced from VESA Presentation	DisplayPort	DVI
# of high-speed pairs 1680x1050@18bpp 1600x1200@30bpp 2048x1536@36bpp	**(Dynamically changes)** **1 pair** **2 pairs** **4 pairs**	**(Dynamically Changes)** **3 Data + 1 Clk pair** **6 Data + 1 Clk pair** **N/A**
Bit rate, per pair	**2.7Gbits/sec, fixed rate (1.62Gbps option available)**	**Up to 1.65Gbps**
Total raw capacity per 4-differential pair single link	**10.8Gbits/sec**	**4.95Gbits/sec**
AC-coupled for process migration (65nm ~ 0.35um)	**Yes**	**No**
Color Depth Supported	**6/8/10/12/16 bpc**	**8 bpc only**
Aux. channels	**1Mbps AUX CH, 500us max. latency**	**DDC, No max. latency limit**
Channel Coding	**ANSI8B/10B (Open)**	**TMDS (Proprietary)**
HDCP/DPCD optional	**HDCP/DPCD Optional**	**HDCP optional**
Protocol	**Micro-Packet-based; extensible in future to add features.**	**Digitized and serialized analog video raster**
Internal connection	**Included in first release of spec**	**No TMDS-based standards**
EMI reduction method	**No clock channel** **Reduced number of pairs** **Data scrambling** **Spread spectrum**	**Transition minimized coding during display active period**

Multiple Display Channels

Why multiple display channels?

- Resolutions becoming larger than channel bandwidths
  - Sony 4K projector
  - Barco 56” panel
  - …….

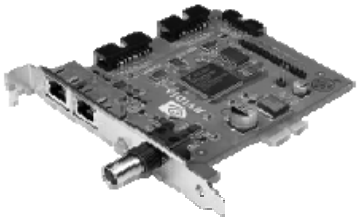
Implications of Multiple Display Channels

First a couple of questions:

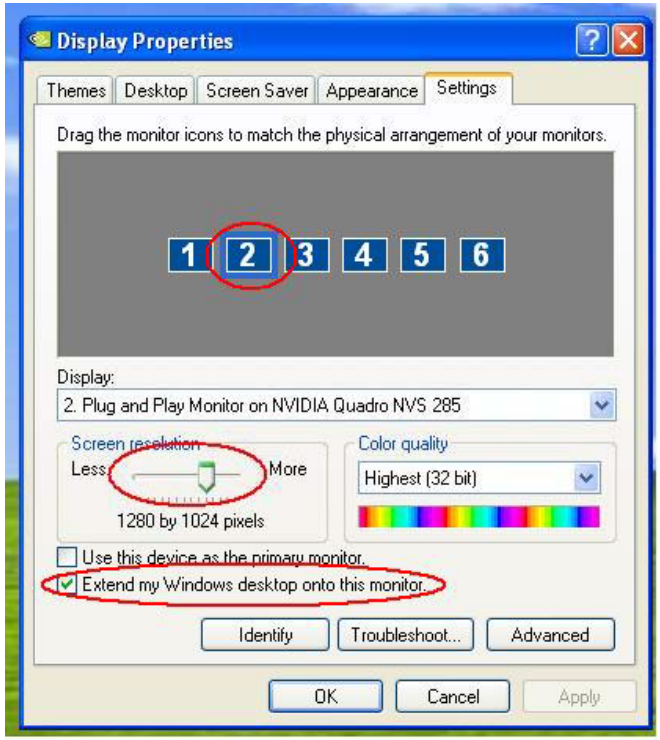
- Which OS - Windows or Linux?
- Level of application transparency:
  - Driver does everything?
  - Application willing to do some work?

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 Gsync card

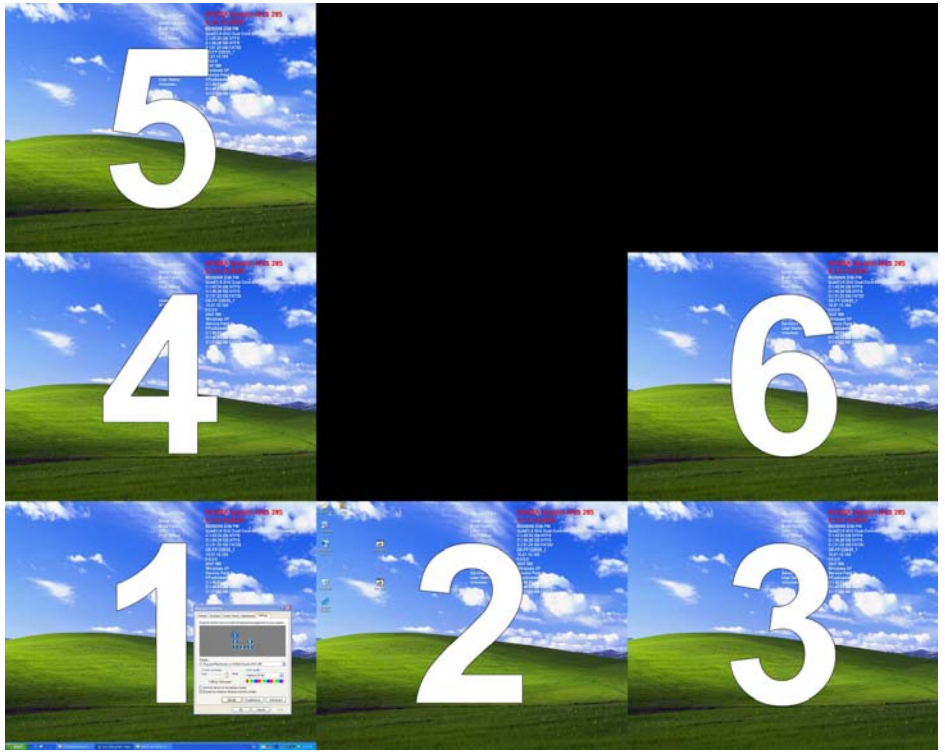


Multiple Displays - Windows



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
  - The 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 solves this

Multiple Displays - Windows

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

Verify first display exists and get display settings

```
if (EnumDisplayDevices(NULL, 0, &lDispDev, NULL)) {
      EnumDisplaySettings(lDispDev.DeviceName, ENUM_CURRENT_SETTINGS, &lDevMode);
}
```

Create Window on first display

```
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;
}
```

Verify second display exists and get display settings

```
if (EnumDisplayDevices(NULL, 1, &lDispDev, NULL)) {
       EnumDisplaySettings(lDispDev.DeviceName, ENUM_CURRENT_SETTINGS, &lDevMode);
}
```

Create Window on second display

```
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;
}
```

Multiple Displays - Linux

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

Multiple Displays - Linux

- Nvidia-settings provides full featured control panel for Linux
- Use nvidia-xconfig to create Xorg.conf the customize
- Drivers can capture EDID
  - Useful when display device hidden behind KVM or optical cable
- Synchronizing multiple displays requires gsync card

Addressing Multiple GPUs

GPU Affinity

- WGL extension, 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 `wlgMakeCurrent()`
  - Context will allow rendering only to GPU(s) in its affinity-mask

Addressing Multiple GPUs 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 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

```
#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>
```

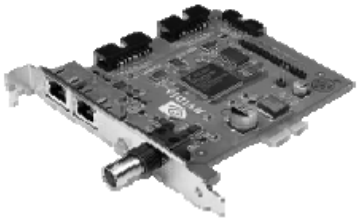
Create list of the first MAX_GPUs in the system

Create an affinity-DC associated with first GPU

Make the FBO current to render into it

Synchronizing Multiple Displays

- Requires Gsync
  - Syncrhonize 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.

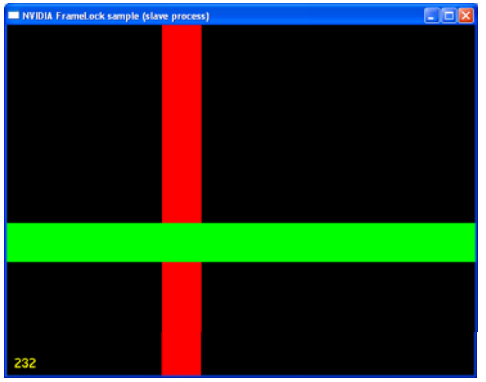
This extension also provides the capability to sychronize the buffer swaps of different swap groups, which may reside on distributed systems on a network. For this purpose swap groups can be bound to a swap barrier.

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

Issues

An implementation can not guarantee that the initialization of the swap groups or barriers will succeed because the state of the window system may restrict the usage of these features. Once a swap group or barrier has been sucessfully initialized, the implementation can only guarantee to sustain swap group functionality as long as the state of the window system does not restrict this. An example for a state that does typically not restrict swap group usage is the use of one fullscreen sized window per windows desktop.

```
BOOL wglJoinSwapGroupNV(HDC hDC,
                        GLuint group);

BOOL wglBindSwapBarrierNV(GLuint group,
                          GLuint barrier);

BOOL wglQuerySwapGroupNV(HDC hDC,
                         GLuint *group);
                         GLuint *barrier);

BOOL wglQueryMaxSwapGroupsNV(HDC hDC,
                             GLuint *maxGroups,
                             GLuint *maxBarriers);

BOOL wglQueryFrameCountNV(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.

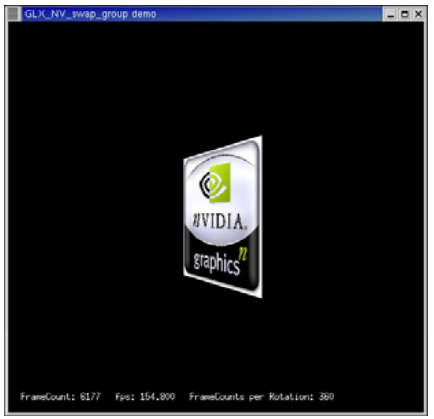
This extension also provides the capability to sychronize the buffer swaps of different swap groups, which may reside on distributed systems on a network. For this purpose swap groups can be bound to a swap barrier.

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

Issues

An implementation can not guarantee that the initialization of the swap groups or barriers will succeed because the state of the window system may restrict the usage of these features. Once a swap group or barrier has been sucessfully initialized, the implementation can only guarantee to sustain swap group functionality as long as the state of the window system does not restrict this. An example for a state that does typically not restrict swap group usage is the use of one fullscreen sized window per desktop.

```
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,
                             GLuint screen, GLuint *maxGroups,
                             GLuint *maxBarriers);

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

Bool glxResetFrameCountNV(Display *dpy);
```



Swap Group Usage Example

```
static void toggleSwapGroup()
{
    if (hasSwapGroupNV)
    {
        swapGroup = (swapGroup == 0) ? 1 :0;
        wglJoinSwapGroupNV(hDC, swapGroup);
        glutPostRedisplay();
    }
}

static void toggleSwapBarrier()
{
    if (hasSwapGroupNV && swapGroup > 0)
    {
        if (swapBarrier > 0) {
            swapBarrier = 0;
            wglBindSwapBarrierNV(swapGroup, swapBarrier);
        } else {
            hasSwapBarrierNV = initSwapBarrierNV();
        }
        glutPostRedisplay();
    }
}
```

```
display(void)
{
    .............
    /* refresh the current frame counter */
    frameCount = updateFrameCounter();
    .............

    /* adjust the width of the bars determine the current position
       of the bars depending on the frame counter */

    horiz_width = (windowWidth / 12) + 1;
    vert_width  = (windowHeight / 9) + 1;
    horiz_left = frameCount % (windowWidth - horiz_width);
    vert_top   = frameCount % (windowHeight - vert_width);

    glClear(GL_COLOR_BUFFER_BIT);

    /* draw the vertical and horizontal bars as rectangles */
    glColor3f(1, 0, 0);
    glRecti(horiz_left, windowHeight, horiz_left + horiz_width, 0);
    glColor3f(0, 1, 0);
    glRecti(0, vert_top, windowWidth, vert_top + vert_width);

    /* display frame counter if enabled */
    if (displayCounter) {
        char counterString[16] = ""; sprintf(counterString, "%d", frameCount);
        glColor3f(1, 1, 0); printStr(10, 10, counterString);
    }

    /* swap buffers to display current frame */
    glutSwapBuffers();
}
```

Using Gsync

Recommendations:

- Control Panel will cause regular CPU contention
  - Polls hardware status
- Currently implementing interrupt API
  - Return upon events
- Synchronize clients periodically in addition to swapbarrier

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

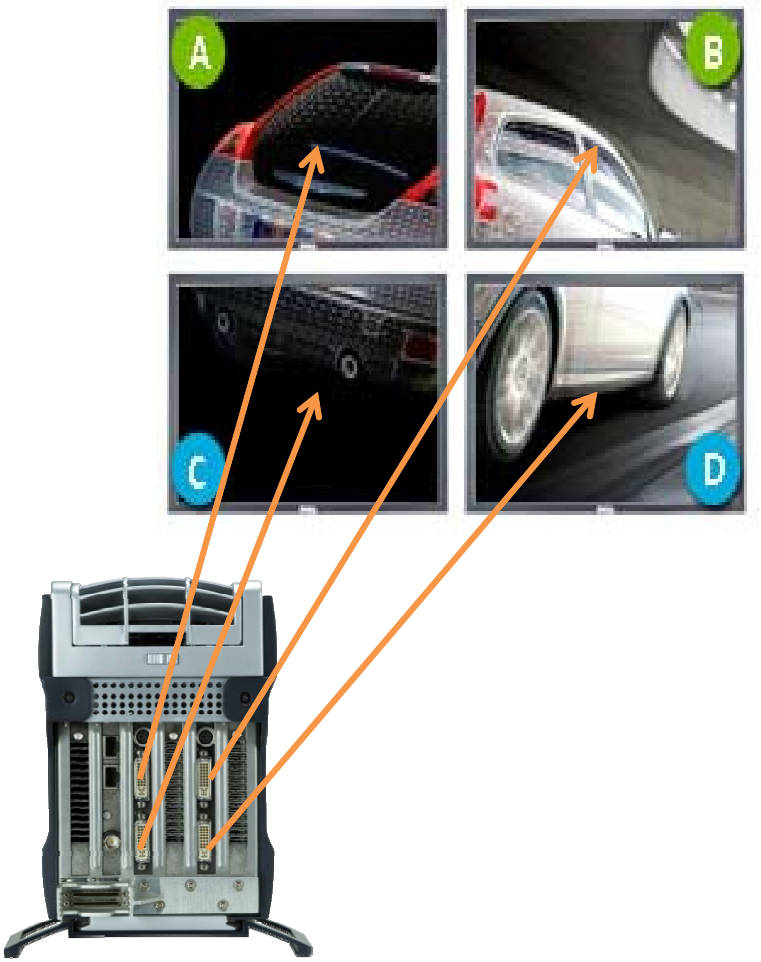
SLI Mosaic Mode

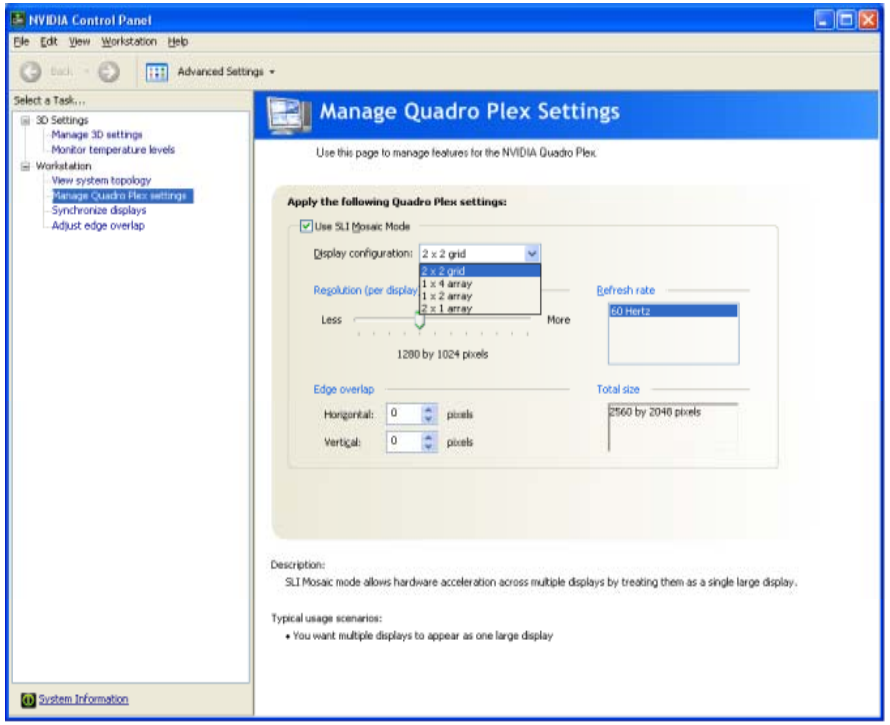
Details

- Quadro Plex only
- Operating System support
  - Windows XP, Linux, 32bit and 64bit
  - Vista (in future)
- Maximum desktop size = 8k X 8k
  - Pay attention to some FSAA modes
- Compatible with G-sync
  - Clustering tiled displays
- Phase 2
  - Stereo
  - More configs

SLI Mosaic Mode

Configurations





NVIDIA Quadro Plex – Roadmap

Series	1000	2000
Maximum GPU Performance	Quadro Plex Model IV 2 G80 GPUs 1.5GB/GPU 4 Dual Link DVI G-Sync II PCI-E Gen1	Quadro Plex 2200 D2 **2 G100 GPUs** **Double Precision** **4GB/GPU** **4 Dual Link DVI + 2 DP** **G-Sync II** **PCI-E Gen2**
Max. # GPU and/or Channels	Quadro Plex Model II 4 G71 GPUs 512MB/GPU 8 Dual Link DVI G-Sync PCI-E Gen1	Quadro Plex 2100 D4 4 G92 GPUs 1GB/GPU 8 Dual Link DVI G-Sync II PCI-E Gen2

Combining Technologies – Stereo and Multiple Displays

- Gsync necessary to synchronize GPUs
- Certain configurations work
- Other configurations don’t work
  - Ambiguity in configurations
  - Mainly due to control panel configuration
- SLI Mosaic Phase 2 will include stereo

Combining Technologies – HDR and Multiple Displays

- Mechanics of >8bit per component can works with multiple channels
  - With same limitations
  - And Mosaic mode

Combining Technologies – SLI Mosaic Mode and GPUs

- Currently Mosaic will assume/consume all GPUs
  - Phase 2 aims to address this
- Being driven by quality and performance demands of large venue display
  - Quality + Performance
- More capabilities will appear over time
  - Nvscale + Mosaic

Summary

- High Resolution Displays are becoming de-facto for collaborative and large venue installations
- Bandwidth requirements of current displays will still require multiple channels, even with Display Port
- Leading edge high resolution displays are HDR capable

Summary - cont.

- Demand for High Resolution & HDR technologies are being driven by economics
  - E.g. Digital Prototypes significantly less expensive than physical prototypes
- Current solutions can take advantage of High Resolution & HDR
  - Opportunity for high-value, high-margin applications

Thank You!

- Feedback & Questions.