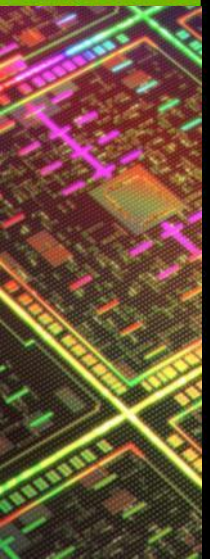
Warping & Blending for Multi-Display Systems

Shalini Venkataraman
Senior Applied Engineer, NVIDIA
shaliniv@nvidia.com



Agenda

- The problem – what do we mean by seamless?
- The way it has been done up till now
- Our Solution
- Programming for Multi-display configurations

The Problem

- Increases in pixel density and total pixels have not kept pace with increases in CPU and GPU power
- Different solutions for adding more pixels
  - LCDs: obtrusive bezels in the way

  - Nearly bezel-less

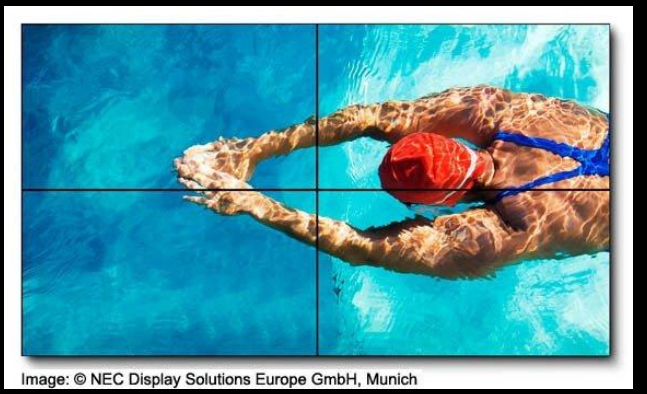
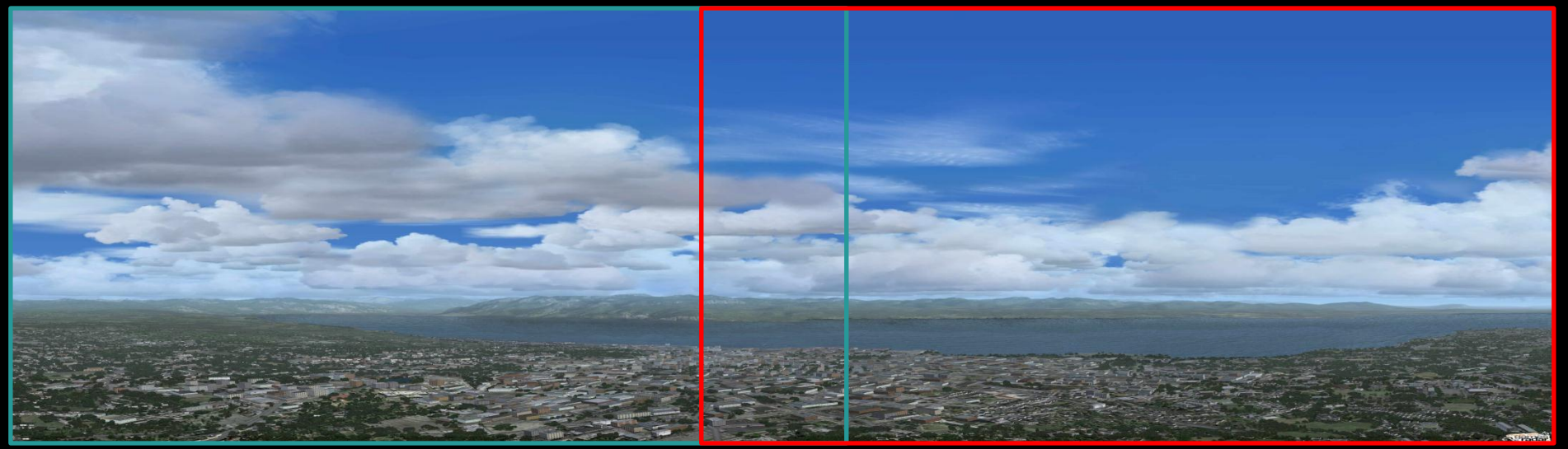
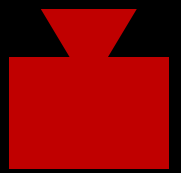


Image: © NEC Display Solutions Europe GmbH, Munich

The Problem (cont'd)

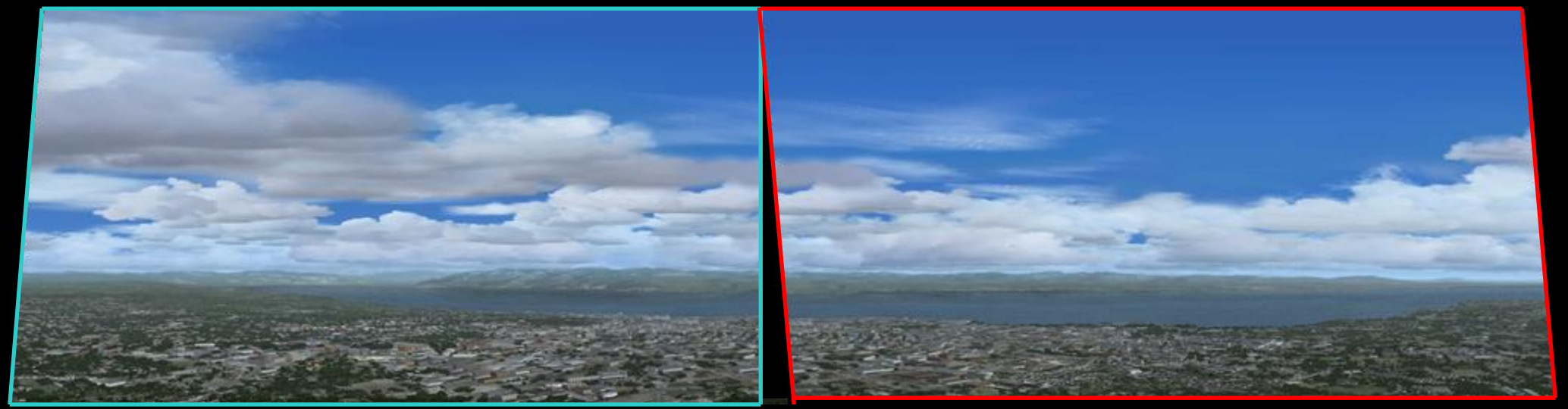
- Projectors: overlap the edges to hide the seam





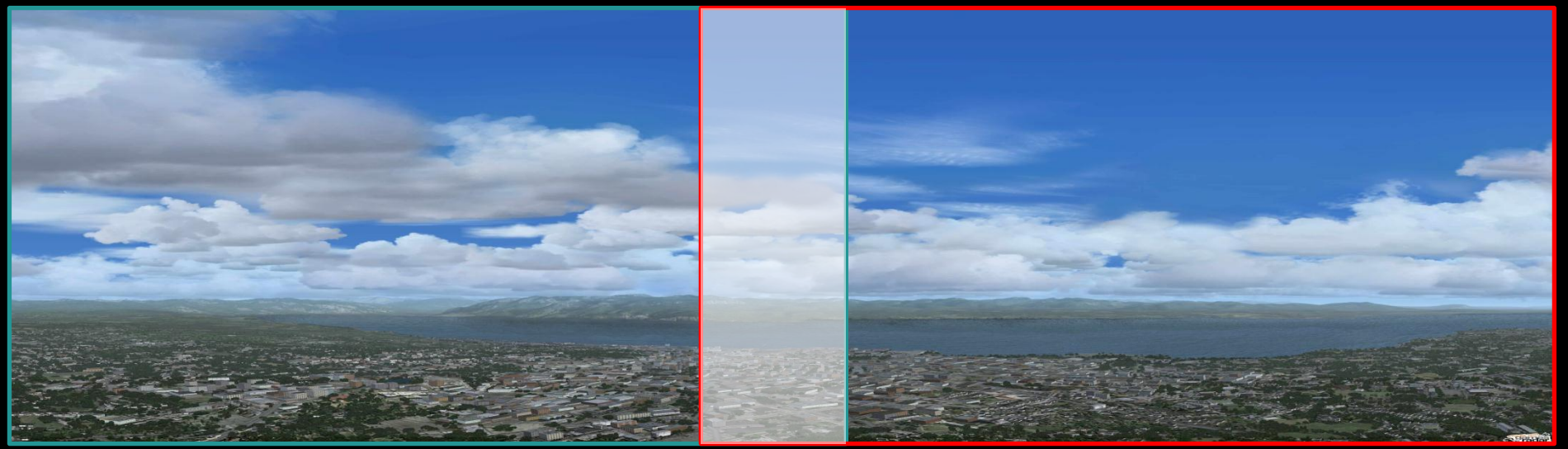
The Problem (cont'd)

- Projectors: optics (and screens) are never perfect



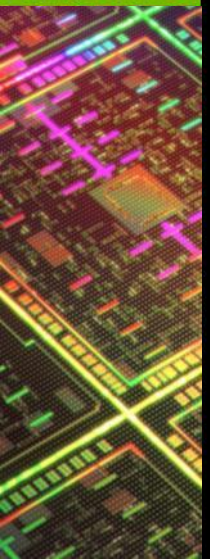
The Problem (cont'd)

- Just creating the overlap makes a hot spot since the overlap region gets twice the light



The solution

- Warp & Blend
  - Warp = Geometry Corrections
  - Blend = Intensity corrections
- Can do one or the other, or both



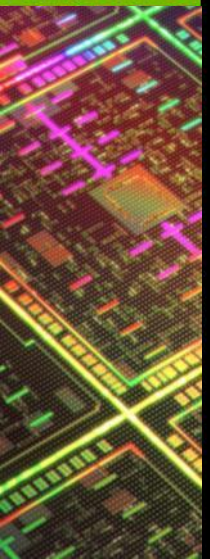
The way it's been done up until now

- Hardware appliance for warp and intensity adjustment
  - Expensive
  - Extra performance delay tax on the display pipeline
  - Additional complexity

- Software warp and intensity adjustment
  - Applications need to be written to manage
  - There has not been an easy way to implement this for any application, until now…

NVIDIA's Solution

- We can do this on the GPU!
  - GPUs are inherently parallel and already have the pixel information
    - Fast for image processing operations
  - GPUs are designed for imaging, texturing and raster operations (compared with external boxes using FPGAs)
  - Perform the transformation in the display pipeline before the pixels get scanned out
  - By doing this on the GPU, we have more flexibility: high quality filtering, integration with SLI Mosaic, etc.



NVIDIA's Solution

- Works on Quadro 5000, 6000, and Quadro Plex 7000
- Use it with G-sync to get synchronization between displays

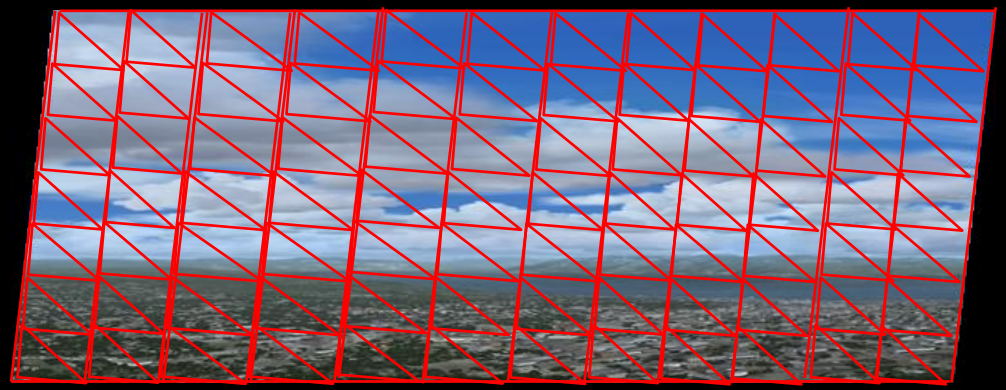
How's it Done: Warp & Blend Workflow

Define Distortion

Numerical

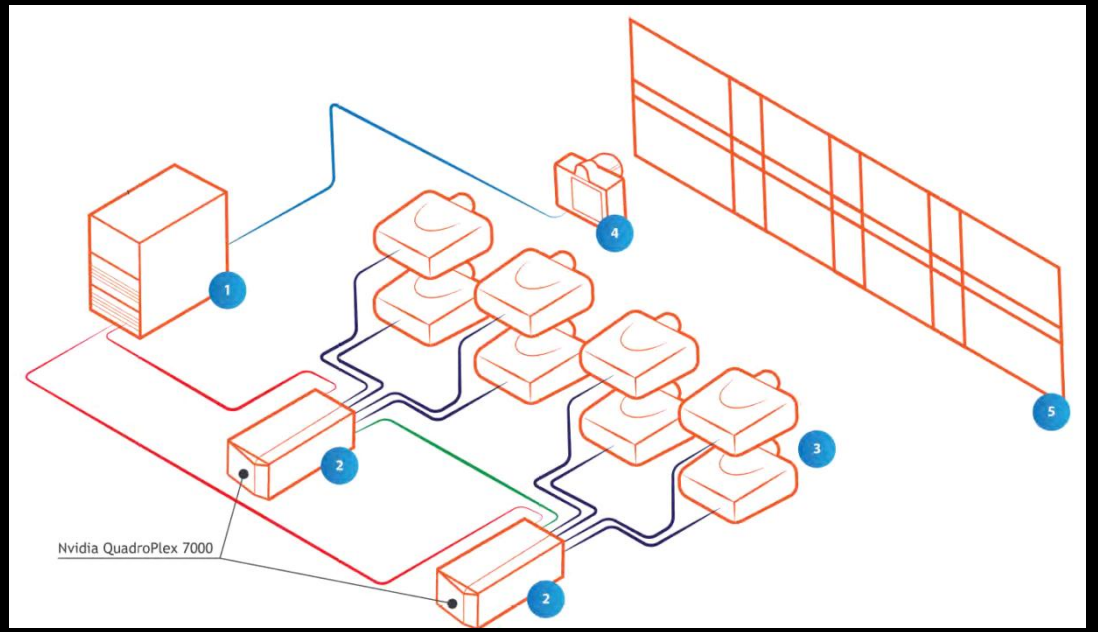
Optical

Create warping mesh and texture coordinates to implement distortion



Typical Warping Mesh contains 4-2M vertices

How's it Done: Overall Workflow

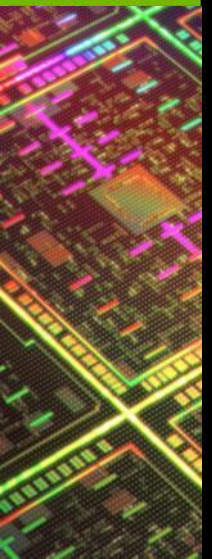


- Set Mosaic Topology
- Capture imagery for warp & blend calculation (optical)
- Compute and set overlap
- For each display
  Compute and set warp, intensity and black adjustment

Doug Traill, S0341 - See the Big Picture Scalable Visualization Solutions for System Integrators, GTC 2012 recordings

How's it Done Programmatically: NVAPI

- NVAPI is Nvidia's programmatic interface to configure and control the GPUs. http://developer.nvidia.com/nvapi
  - Query/Set GPU and display configurations, layouts etc
- New interfaces are added in the 295+ NDA version to allow warping and intensity adjustment before the final scanout
- R302 NDA version will add support for image offset to do black-level adjustment
- Works on single screen, multiple screens and multi-gpu configuration
- Win 7 only



Enumerating Displays

- Get number of grids

```
NvU32 gridCount;
NvAPI_Mosaic_EnumDisplayGrids(NULL, &gridCount)
```

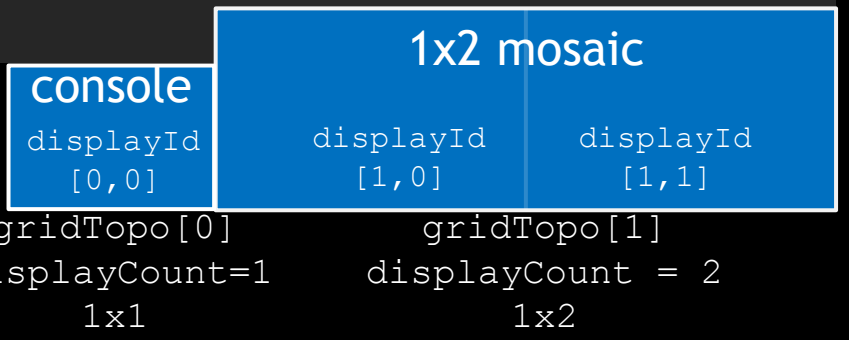
- Get display topology

```
gridTopo = new NV_MOSAIC_GRID_TOPO[gridCount];
NvAPI_Mosaic_EnumDisplayGrids(gridTopo, gridCount)
```

Getting Display Topology

- Iterate over all grids and displays and get properties

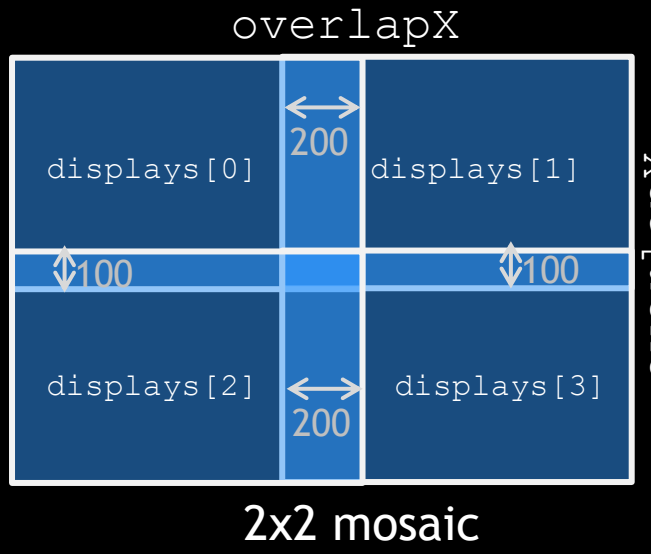
```
for (NvU32 iGrid = 0; iGrid < gridCount; iGrid++) {
    NvU32 numDisplays = gridTopo[iGrid].displayCount ; //No of displays in this grid
    NvU32 numRows = gridTopo[iGrid].rows; //No of rows in this grid
    NvU32 numCols = gridTopo[iGrid].columns; //No of columns in this grid
    NV_MOSAIC_DISPLAY_SETTING& ds = gridTopo[iGrid].displaySettings;
    ds.width; ds.height; ds.freq ; //Width, Heiht and Refresh Rate for all displays
    for (NvU32 iDisplay=0; iDisplay< gridTopo[iGrid].displayCount ;iDisplay++) {
        NV_MOSAIC_GRID_TOPO_DISPLAY& display = gridTopo[iGrid].displays[iDisplay];
        NvU32 displayId = display.displayId ;//unique identifier for this display, that
                                            // will be used for all subsequent functions
        display.overlapX; //horizontal overlap for this display, explained later
        display.overlapY; //vertical overlap for this display, explained later
    }
}
```



Programming overlap per grid

- Specifying overlapX and overlapY
  - the number of pixels of overlap between each display and the ***previous*** row or column
  - All displays in a column (row) should have same overlapX (overlapY)

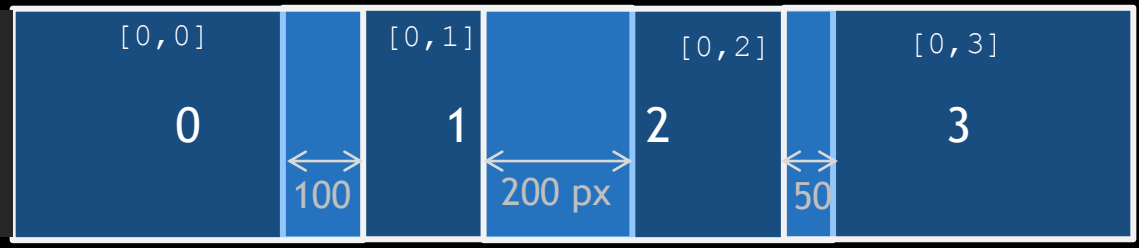
```
NV_MOSAIC_GRID_TOPO& grid = gridTopo[1];
//column 0: set overlapX =0
grid.display[0].overlapX = 0;
grid.display[2].overlapX = 0;
//row 0: set overlapY =0
grid.display[0].overlapY = 0;
grid.display[1].overlapY = 0;
//column 1: 200px overlap between column 0 & 1
grid.displays[1].overlapX = 100;
grid.displays[3].overlapX = 100;
//row 1: 100px overlap between row 0 & 1
grid.displays[2].overlapY = 100;
grid.displays[3].overlapY = 100;
```



Overlap cont'd

- Displays in different rows/columns can have different overlaps

```
grid.display[0].overlapX = 0;
grid.display[1].overlapX = 100;
grid.display[2].overlapX = 200;
grid.display[3].overlapX = 50;
```



1x4 mosaic

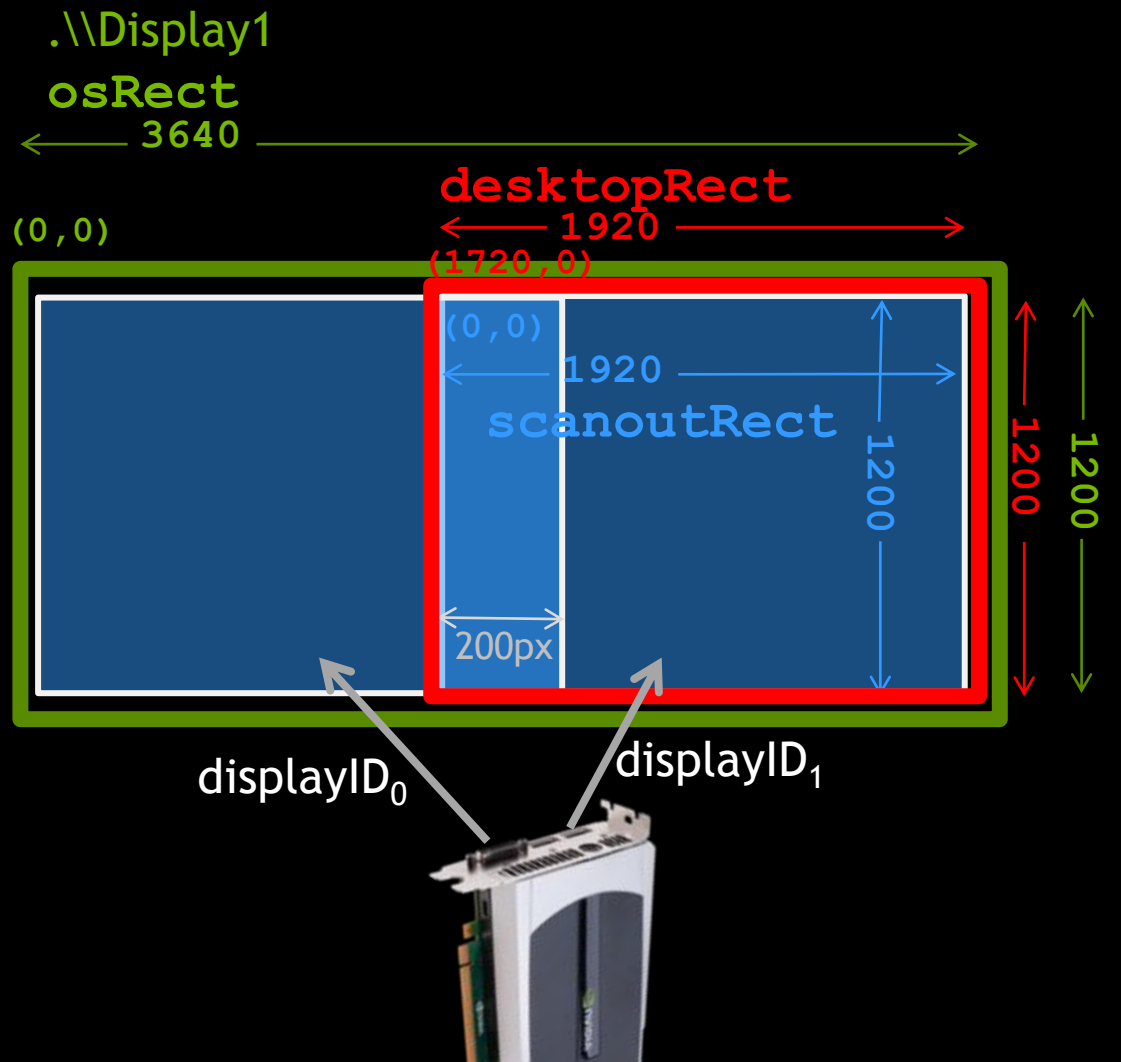
- Set for entire grid topology

```
NvAPI_Status ret = NvAPI_Mosaic_SetDisplayGrids(gridTopo, gridCount,
NV_MOSAIC_SETDISPLAYTOPO_FLAG_CURRENT_GPU_TOPOLOGY);
```

Check return value and handle errors properly!

Fun with display coordinate systems

- scanoutRect
  - Per display
- desktopRect
  - Subregion relative to desktop
  - Includes overlap
- osRect
  - Extent of OS-visible virtual desktop
    - eg .\\Display1
  - Includes overlap



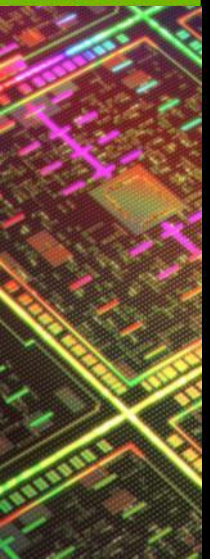
Getting display coords from NVAPI

- For each display, get its scanoutRect and desktopRect

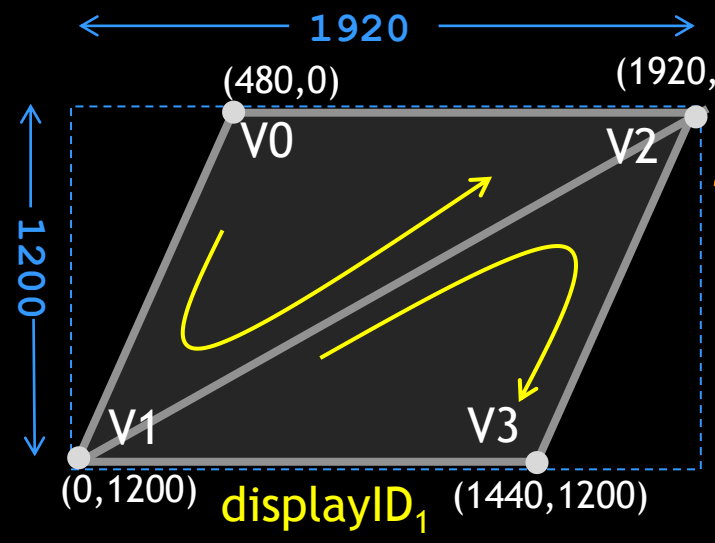
```
NvSBox desktopRect; //extent of this display wrt desktop
NvSBox scanoutRect; //extent per display
NvAPI_GPU_GetScanoutConfiguration(displayId, &desktopRect, &scanoutRect);
```

- For each display, get its osRect

```
NvSBox osRect; //os coordinates for this virtual display
DEVMODEA dm = { 0 };
dm.dmSize = sizeof(DEVMODEA);
if (EnumDisplaySettingsA(displayName, ENUM_CURRENT_SETTINGS, &dm)) {
    osRect.sX = dm.dmPosition.x;
    osRect.sY = dm.dmPosition.y;
    osRect.sWidth = dm.dmPelsWidth;
    osRect.sHeight = dm.dmPelsHeight;
}
```

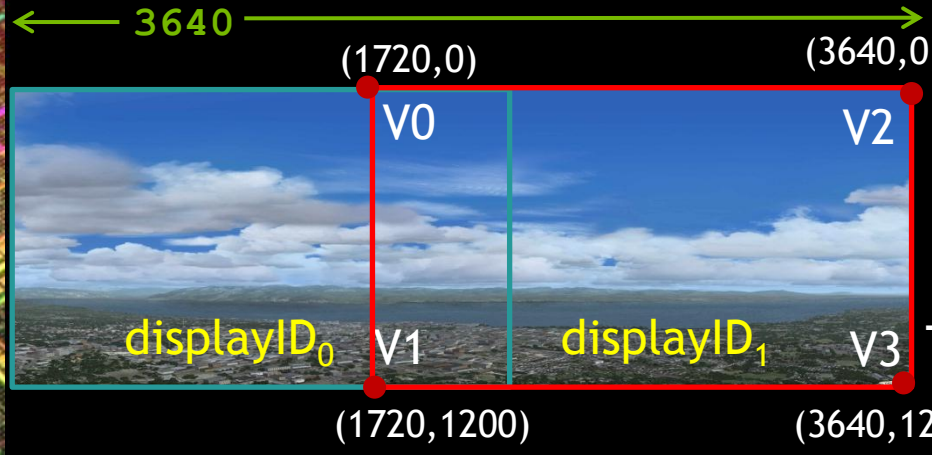


Warp example



Vertex positions specified in scanoutRect space

Positions

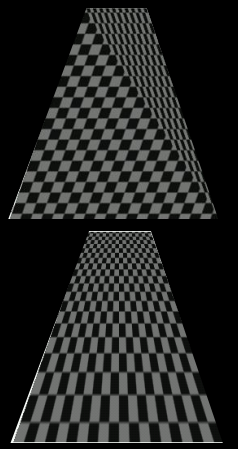


Texture Coords

Texture coords specified from desktopRect

Warping Data Structure

- **NV_SCANOUT_WARPING_DATA**
  - vertexFormat : strip or triangle list
    - NV_GPU_WARPING_VERTICE_FORMAT_TRIANGLESTRIP_XYUVRQ
    - NV_GPU_WARPING_VERTICE_FORMAT_TRIANGLES_XYUVRQ
  - vertices : array of 6 float vertex
    - x,y : mesh coordinates per-display rectangle
    - u,v : texture coordinates in desktop space
    - r,q : perspective mapping to simulate 3D warp
  - numVertices
  - textureRect
    - Pass in the osRect



Warping - Code

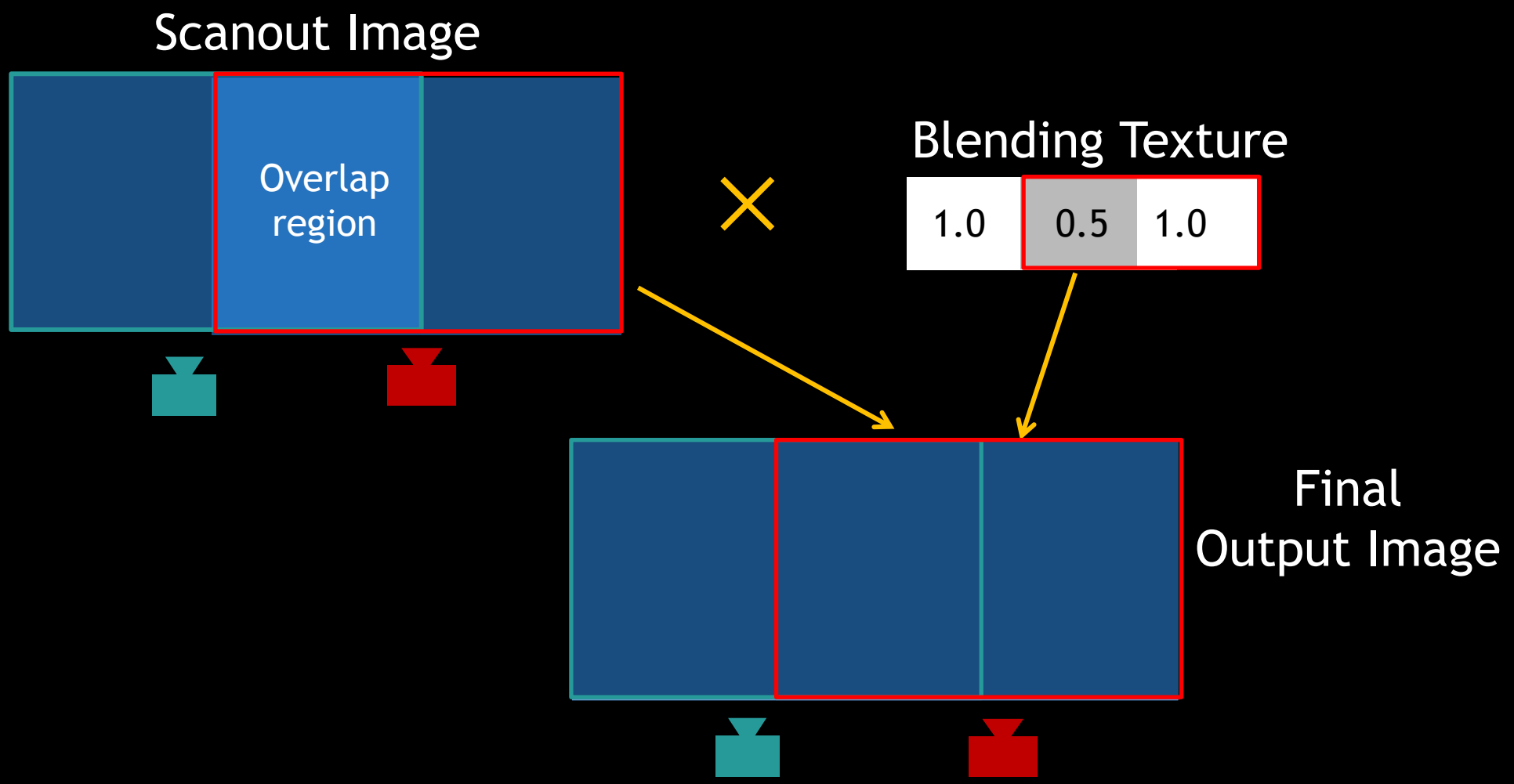
- Enable Warping

```
float vertices[numVerts*6] ={x0,y0,u0,v0,r,q, x1,y1,u1,v1,r,q, …};
NV_SCANOUT_WARPING_DATA warpingData;
warpingData.version = NV_SCANOUT_WARPING_DATA_VER;
warpingData.numVertices = numVerts;
warpingData.vertexFormat =
        NV_GPU_WARPING_VERTICE_FORMAT_TRIANGLESTRIP_XYUVRQ;
warpingData.vertices = vertices;
warpingData.textureRect = osRect;
int sticky = 0; // output - Reserved field for future use
int maxNumVertices = 0; // output - returns the #pixels at scanout
// This call does the  warp
NvAPI_Error error =  NvAPI_GPU_SetScanoutWarping(displayId, &warpingData,
&maxNumVertices, &sticky);
```

- Disable Warping

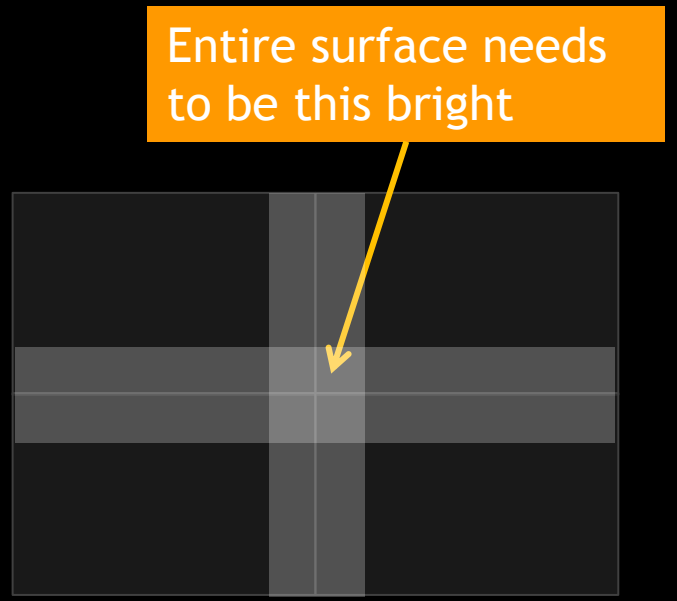
```
warpingData.numVertices = 0;
warpingData.vertices = NULL;
NvAPI_GPU_SetScanoutWarping(displayId,…);
```

Blend Example



Blend - with Offset Texture

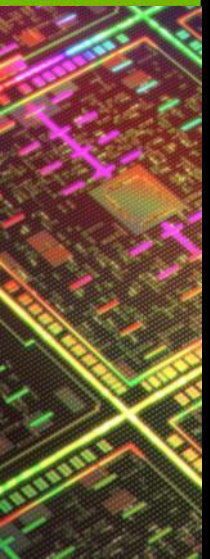
- New feature starting R302
- Separate offset texture
  - Inverse of black-level image
  - Can be 1 or 3 channel
  - Blended with already modulated image



$$Output = Input * blendTexture * (1 - offsetTexture) + offsetTexture$$

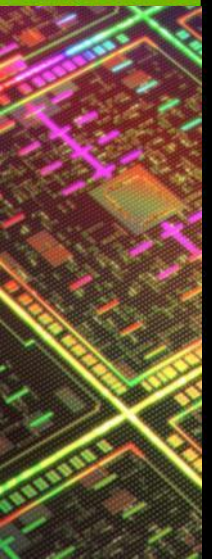
Blend/Intensity Adjustment

- `NV_SCANOUT_INTENSITY_DATA`
  - `width, height`
    - Dimensions of blending texture
    - Normally same dimensions as scanout rectangle
    - If larger than scanout size, driver dynamically downsamples using box filter
  - `blendingTexture`
    - `float[width*height*3]`, RGB with same storage layout as OpenGL
    - Set to NULL for no adjusments
  - `offsetTexture`
    - Same dimensions as blendingTexture
  - `offsetTexChannels`
    - Number of components in the offsetTexture, 1 or 3



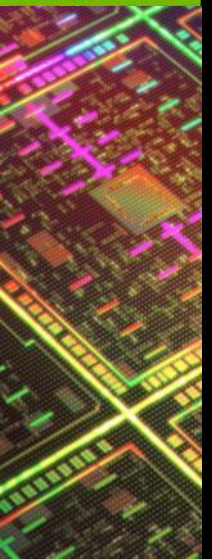
Blending - Code

```
NV_SCANOUT_INTENSITY_DATA intensityData;
// simple 1x2 config, overlap region is modulated by 0.5
float intensityTexture[6]         = {0.5f, 0.5f, 0.5f,
                1.0f, 1.0f, 1.0f} ;
// overlapped region doesn't require an offset
float offsetTexture[6]          = {0.0f, 0.0f, 0.0f,
                0.1f, 0.1f, 0.1f} ;

intensityData.version               = NV_SCANOUT_INTENSITY_DATA_VER;
intensityData.width                 = 2;
intensityData.height                = 1;
intensityData.blendingTexture       = intensityTexture;
intensityData.offsetTexture         = offsetTexture;
intensityData.offsetTexChannels     = 3;

int sticky = 0; // output - Reserved field for future use
// This call does the intensity map
NvAPI_Status error =  NvAPI_GPU_SetScanoutIntensity(displayId,
&intensityData, &sticky);
```



Pointers

- Disabling/enabling warp is expensive
  - Requires modeset, lag in projector environments
  - However, changing the warp mesh does not require modeset
    - Eg During calibration, use identity quad with warp call to simulate no warping
- Changing warp mesh is not deterministic
  - Warp should not be changed for continuous updates
    - Eg eye tracking at 60Hz, best to do that in the app
  - OK to change it infrequently
    - Eg during calibration



Feature Roadmap

- Filtering
  - Other options
- Offset Image addition
  - Various blending modes
- Persistence across reboot
  - making the settings consistent across reboots
- Linux support


Questions?
quadrosvs@nvidia.com