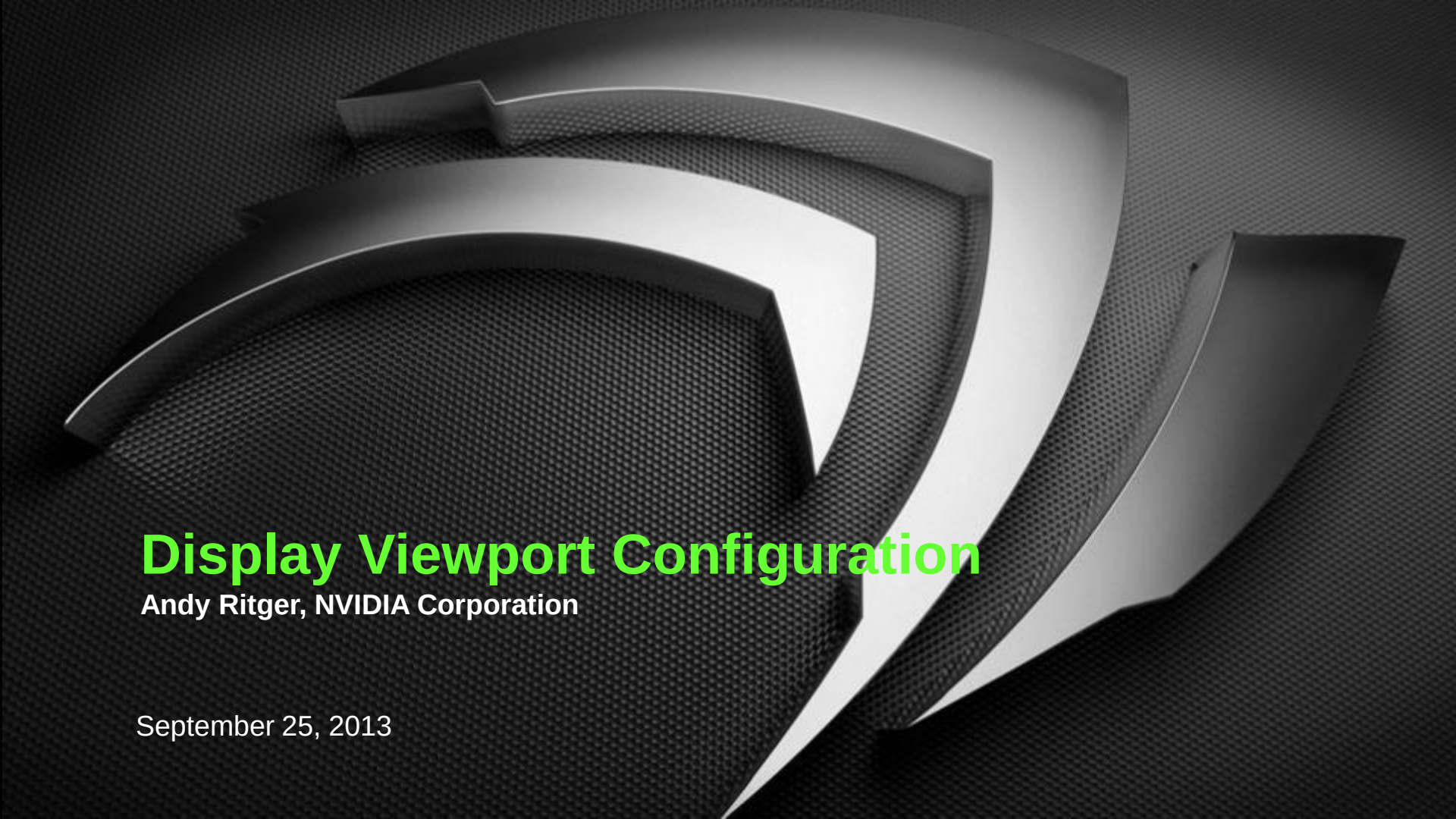


Display Viewport Configuration

Andy Ritger, NVIDIA Corporation

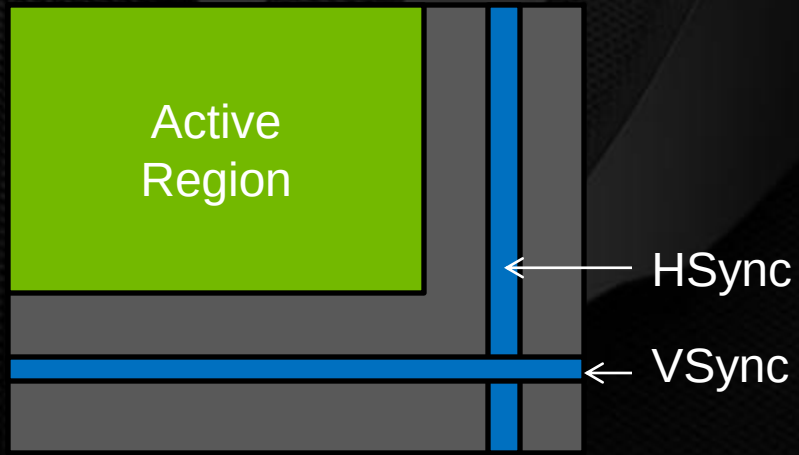
September 25, 2013

Introduction

- Discuss some challenges around configuration of:
  - Fixed-resolution displays.
  - Flatpanel scaling.
  - Overscan Compensation.
  - Aspect Ratio scaling.

CRTs

- In the old days, there were CRTs, driven by VGA.
- A resolution and refresh rate on CRT described by a "mode":
  - Active region
  - Blanking period
  - Sync pulse



Digital Displays

- Digital displays normally driven by protocols such as TMDS, LVDS, DisplayPort.
- Resolution and refresh rate still communicated by a "mode".
- Digital displays can generally only accept the modes advertised in their EDID.
- To achieve any other desktop resolution, the GPU has to scale the desktop resolution to one of the EDID-reported modes.

Special-Purpose Digital Display Scaling

- To overcome the limited modes in a digital display's EDID, GPUs added special-purpose scalers in the scanout pipeline for digital displays.
- These scalers allowed the GPU to fetch a region of one size and upscale to the resolution of the mode being sent to the display.

Configuration of Special-Purpose Digital Display Scaling

- To X users/clients, flat panel scaling looks like:
  - One mode was validated, but only its resolution was used.
  - A different mode (one of the EDID modes) is actually sent to the display.
- Causes confusion:
  - Onscreen display reports something different than X server.
  - Refresh rates may not agree.
  - How to choose which EDID mode to use?
    - A digital display may have multiple EDID modes that the user wants to target (e.g., 60Hz vs 120Hz for stereo)
    - Example user confusion: http://lists.x.org/archives/xorg/2012-August/054867.html

Overscan Compensation

- TVs (analog or digital) historically have a bezel around edge.
- For a lot of video content, the content at the edge is considered less important
- TVs "overscan", spilling content behind bezel.
- GPUs need to provide "overscan compensation", to optionally shrink the content, such that nothing is hidden by bezel.
  - Downscale content (e.g., from 1920x1080 to 1900x1060, with 10 pixel border): filtering artifacts.
  - Change content size (e.g., display 1900x1060 image centered within 1920x1080 mode).

Aspect Ratio

- The application might have content in an aspect ratio other than that of the monitor.
- The user or application might want a configuration such that the content is letterboxed (or pillarboxed, or windowboxed) within a mode on the monitor.
- E.g., 1920x1080 content on a 1920x1200 mode.

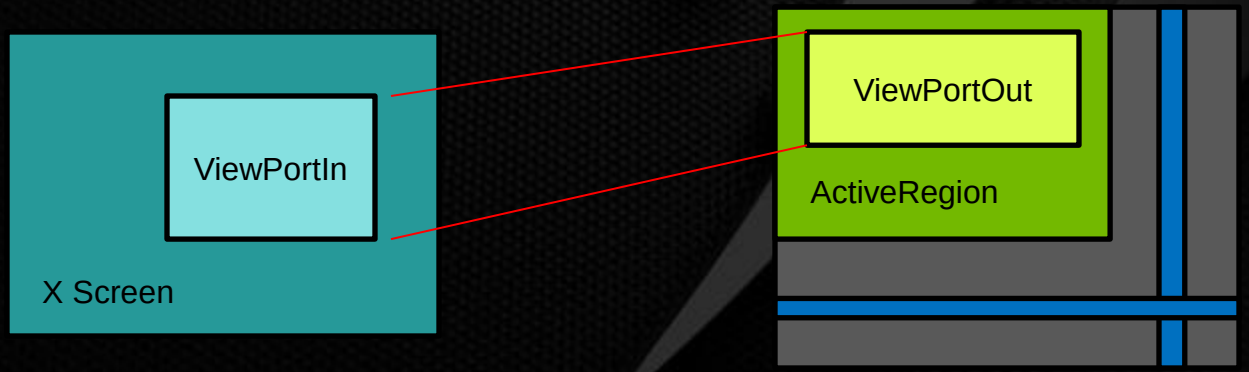
Scaling Capabilities Evolved

- With modern GPUs (NVIDIA at least, I assume others), scaling is much more generalized:
  - Scaling can be used independently of the output protocol.
  - Both up- and down-scaling are supported.

Scaling Capabilities Evolved (cont.)

- Four different resolutions:
  - Size of X screen.
  - ViewPortIn: region within X screen that GPU's head fetches.
  - ViewPortOut: region within the active region to which ViewPortIn should be scaled.
  - ActiveRegion: the pixels sent to the monitor.

- These hardware capabilities are the basis for display scaling, overscan compensation, and aspect ratio configuration.



How To Configure?

- It would be nice to give users the flexibility of the hardware.
- Traditional digital display scaling encodes some amount of policy:
  - The only scaling options are: none, full, aspect, center.
  - It would be nice to move that policy out of the X server.
- Solution:
  - Decouple scaling configuration from the mode.
  - Avoid having multiple knobs (viewport and flatpanel scaling) to control one setting.
  - Make ViewPortIn and ViewPortOut explicit "meta" configuration used in conjunction with the mode.

How To Configure? Defaults

- ViewPortOut defaults to size of ActiveRegion.
- ViewPortIn defaults to size of ViewPortOut.
- X screen defaults to size of ViewPortIn.

How To Configure? MetaMode Syntax

- In NVIDIA's MetaMode syntax, ViewPortIn and ViewPortOut can be specified explicitly:

```
"1920x1200 { ViewPortOut = 1920x1080+0+60, ViewPortIn = 1280x720 }"
```

How To Configure? RandR

- The "Border" Output property defines the ViewPortOut.
- The transformation matrix defines the scaling from ViewPortIn to ViewPortOut.
  - Drivers can identify when the transformation matrix can be fulfilled by display ViewPort.

```
xrandr --output DVI-I-1 --mode 1920x1200 \
  --set Border 0,60 --scale 0.6666667x0.6666667
```

How To Configure?

- The API building blocks are there today in RandR.
- Border and transformation matrix are pretty-low level for users; need to improve tools.
- How does explicit viewport configuration interact with the "scaling" output property?
  - For NVIDIA, we solved this by just removing the scaling property.
  - This caused user confusion, because the tools were not in place to make the migration seemless.
- Note that if you limit the modepool to EDID-only modes, this can be a small list.

Problem: Advertising Resolutions

- ViewPort configuration provides a lot of flexibility:
  - The possible resolutions to advertise are arbitrary.
  - How to choose which resolutions to advertise, e.g., in full-screen applications, GNOME/KDE control panels?
- It seems nice to not bake this policy into display servers or drivers.
- Many applications don't want the burden of having to choose.
- Encapsulate in a utility library?

Suggestions

- Drivers:
  - Honor the "Border“ RandR Output property.
  - Take advantage of display ViewPorts to satisfy applicable transformations.
- Define a utility library that:
  - Looks at the available (real) modes for a given monitor.
  - Uses a configuration file to decide what resolutions should be advertised.
  - Computes the ViewPort configuration needed to achieve each resolution.

Questions