n-Dimensional Display Interface

Charles D. Estes
University of North Carolina at Chapel Hill
Brooks Computer Science Building, CB 3175
Chapel Hill, NC 27599-3175 USA
cdestes@cs.unc.edu

ABSTRACT

The n-Dimensional Display Interface is a display abstraction conceived from first principles with the intention of replacing the framebuffer as a new "narrow waist" for modern display pipelines.

Keywords

Display interface, framebuffer, scalable display

1. INTRODUCTION

Modern display use cases are dramatically pushing the limits of data transmission requiring higher channel capacity to meet their needs.[1] The growing trend of televisions with UltraHD spatial resolutions brings steep requirements. HDMI 1.4 [2] allows for 4096x2160 at 30 Hz with 48 bits/pixel requiring an uncompressed channel capacity of 12.15 Gbps. Emerging 8K use cases at higher 60 Hz refresh rates will then increase that by nearly an order of magnitude. Refresh rates likely will not stop at 60 Hz either, as many common LCD display panels can already refresh at 240 Hz. HDMI 1.4a [3] additionally handles 3D video at current HD standards while doubling the video output due to the need of producing two stereoscopic images for every frame.

The digital display standards are still keeping pace with these more traditional use cases, but the signaling requirements for the cables and connectors is getting increasingly restrictive. Furthermore, they are all intended for scenarios where the rendering device is connected to the display within the same room. Both remote displays such as VNC [4] and small wireless devices do not have the luxury of such high channel capacities and of a guaranteed quality of service. WirelessHD is addressing UltraHD video at 240 Hz with 48 bits/pixel, but only over special wireless video area networks (WVAN).[5] The Wi-Fi Alliance released a display specification over Wi-Fi along with its Miracast device specification that allow the streaming of high definition content of Wi-Fi networks.[6] Less traditional use cases such as remote displays and massive display walls often require novel solutions beyond the current standards.[7][8][9][10][11]

MM '16, October 15-19, 2016, Amsterdam, Netherlands

DOI: http://dx.doi.org/10.1145/2964284.2971476

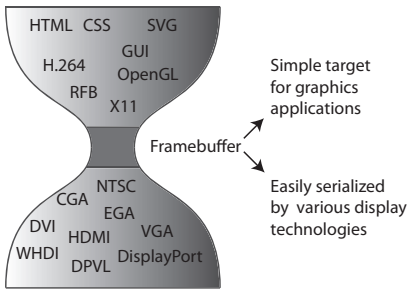

Figure 1: The framebuffer has served as a narrow waist for the display pipeline.

For the earliest computer display systems, the display interface was carefully designed based both on display and the attached host system. Such systems required a discrete *display generator* to drive the electronics of the display and a *display processor* to translate input form the host system for the display generator. Early work to optimize these display processors made tradeoffs between memory, channel capacity, and physical connections to the display. These custom solutions offered little re-use as their designs were just trips around the *wheel of reincarnation.*[12] With the emergence of CRT displays, a new more obvious display interface abstraction took shape, figuratively stopping the wheel.[13] The framebuffer served as a "narrow waist" for the display pipeline as computer graphics on the host system rapidly advanced independently of the signaling standards on the display side (figure 1). By all accounts, the framebuffer has been a great success. Modern displays have digital transmission standards that send compressed data, but nearly all of them still scan the contents of a framebuffer at a fixed refresh rate.

My research seeks to define a new, more flexible narrow waist, just above the framebuffer in terms of complexity. The concept was first introduced at NOSSDAV and later refined in a full conference paper at MMSYS.[14][15] The second phase of research published at ACMMM involved extensions for blending that allowed me to use a novel approach to video rendering on an nDDI display.[16] I am entering into the third phase of my research. With several promising directions available, I hope to use the Doctoral Symposium to collaborate and narrow my focus.

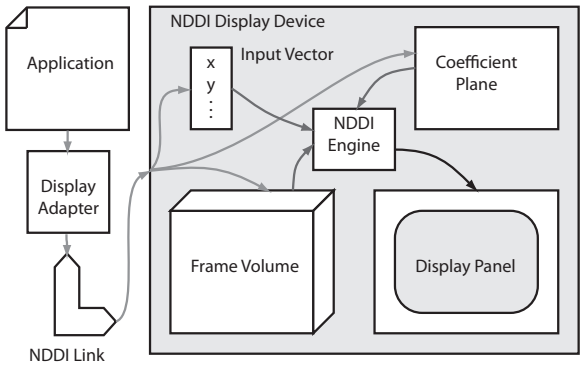

Figure 2: NDDI concept diagram.

2. N-DIMENSIONAL DISPLAY INTERFACE

The *n-Dimensional Display Interface* is a new abstraction from first principles. My goal for the research is to define a new, future-proof abstraction that is:

- **Framebuffer Compatible** for simple applications
- **Data-Driven** avoiding any use of stateful logic
- **Progressively Beneficial** per application needs
- **Highly Parallel** allowing for ideal scalability
- **Asynchronous** avoiding timing constraints

nDDI defines an abstraction just above the framebuffer in terms of complexity making it trivially compatible framebuffer. The primary strength of the interface is a highly flexible addressing scheme; reducing the transmission cost of both pixel data and the addressing information. This is accomplished by creating a large pixel store on the display itself, called the *frame volume* (figure 2). The frame volume can be configured by the driving application to any dimensionality using n-tuples to address pixels, pixel strips, planes, and n-volumes. This allows for efficient fill commands and quick remapping of display pixel values to newly arrived pixels or existing values in the frame volume. The highly parallel mapping operations occur within the *nddi engine* at each pixel site and are controlled by *coefficient matrices* at each display location within the *coefficient plane*. The entire operation is driven by the *input vector*. The length of the vector is configured by the application. The minimal length is two, where the x and y values are driven by the nDDI engine and any additional values are driven by the application. The mapping operation consists of a simple matrix multiplication which multiplies the input vector by the coefficient matrix for each location (x, y) where the x and y values of the input vector are inherently set to the same location (figure 3).

For basic framebuffer compatibility, an application can configure the frame volume in two dimensions and set the input vector to a length of two. Configuring those parameters will determine the size of each coefficient matrix, 2×2 in this case. The application would initialize those coefficient matrices to an identity matrix and then simply write to the frame volume as if it were a framebuffer at a refresh rate of 60 Hz. Configuring an nDDI display in more sophisticated ways can achieve a reduction in channel capacity.

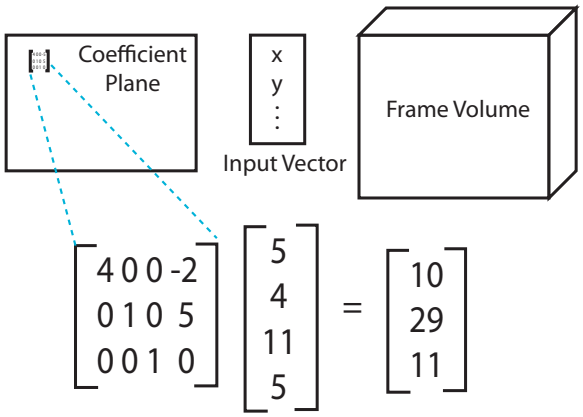

Figure 3: Example matrix multiplication for pixel mapping.

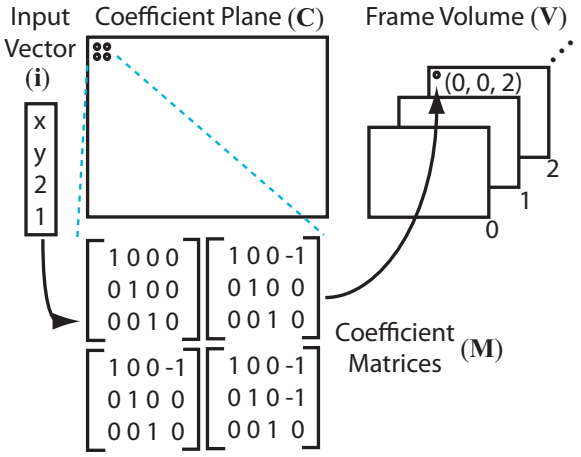

Figure 4: Example NDDI Configuration for simple video player with 4x scaling.

In the case of a video player, the application can configure the frame volume in three dimensions with x and y dimensions matching the video source and then some number of frames in the z dimension for buffering (figure 4). Frames are buffered into the frame volume, and the frames are advanced using the z value in the input vector. The physical display can be larger that the video source, and so the coefficient matrices are configured as simple affine transforms that map a source pixel from the video to multiple pixels on the display.

3. DRIVING AN NDDI DISPLAY

My initial experiments sought to configure an nDDI display in novel ways to provide transmission savings while still not utilizing any application level semantics. These *Pixel Bridge* experiments simply looked at each frame of an input video source and drove the display according to one of three different modes.

- **Framebuffer** - Configured as a 2D framebuffer
- **Flat Tiled** - Configured as a 2D, tiled framebuffer
- **Cached Tiled** - Configured as a cache of tiles

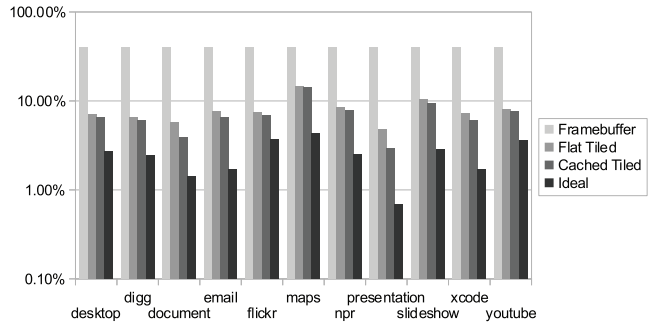

Figure 5: Results for large recordings.

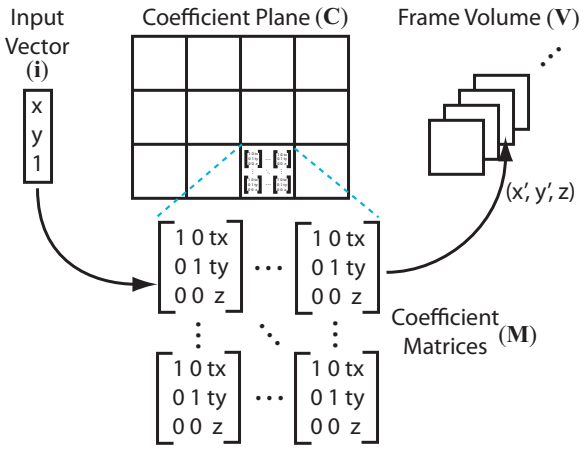

Figure 6: Cached tiler nDDI configuration.

The source videos were recorded video sessions and full motion video clips. The results for the recorded compute sessions are shown (figure 5) as a percentage ratio of the pixels sent for each mode compared to that of 60 Hz signal. Framebuffer Mode produced a basic 40% rate because it is only updating the display at 24 fps vs. 60 Hz. Flat Tiled produced a significant reduction at well under 10%. Cached Tiled mode configured the frame volume as a deep cache of tiles (figure 6) and produced a moderate improvement. All of the results were compared against a Perfect mode which was achieved simply by calculating the data needed to update changed pixels without considering any addressing information.

3.1 Video

For full motion video, my previous Pixel Bridge tiling modes did not fare well. For video, I introduced a DCT tiled mode that leveraged the semantics of video codecs to render the output video as a weighted blending of pre-rendered 8×8 DCT basis functions.[17] To achieve this blending, I experimented with three different blending modes and finally settled on coefficient plane based blending. I extended the single coefficient plane to 64 coefficient planes and added a 3-channel scaler alongside each of the coefficient matrices. Blending in this manner works by performing all 64 mappings for a pixel location, multiplying each pixel channel

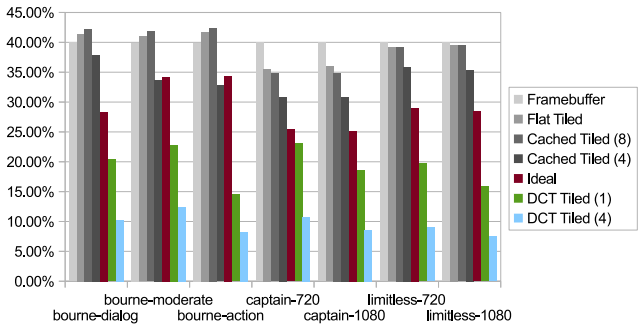

Figure 7: Revised video results with new DCT tiler.

by the associated scaler, summing each channel, and then dividing by a configured max scaler value.

For the DCT tiled mode, the 64 pre-rendered 8×8 basis functions were rendered into the frame volume. Then for each frame, only the significant DC and AC coefficients were transmitted. The results show a significant savings (figure 7). The previous cached tiler did not perform well even when using a lossy matching scheme (Cached Tiled 4). DCT tiled outperformed perfect with a very strong PSNR ($\sim$ 40) at the highest quality levels (DCT Tiled 1,4).

4. THE NEXT PHASE

I have conducted some preliminary investigations into the possible areas of research below. I strongly hope to use this forum of the Doctoral Symposium to collaborate with multimedia experts and researchers to narrow my focus and pave the final path to my dissertation.

- **Advanced nDDI Video** - I introduced a trivial prediction-residual decoder for the DCT tiler. I feel better rate control and a new approach to caching predictions will deliver even strong results.

- **Mixed Multimedia Blending** - Coefficient plane blending is well suited for simple blending use cases. I plan to add support for a per-pixel alpha channel in the frame volume. Extending coefficient plane blending in this way remains highly parallel and provides additional flexibility in terms of use cases supported: masks, compositing, anti-aliasing, font smoothing, etc.

- **Display Wall Simulation** - Display walls are a strong motivation behind the new nDDI abstraction, and I intend to simulate a massive display wall using an nDDI Link that will support multiple clients streaming content to the single nDDI display.

- **Memory Modeling and Simulation** - Extending the architecture to support multiple coefficient planes dramatically increased the memory requirements for the nDDI display, however the access patterns are all very regular and the memory can be localized to each pixel. On the other hand, the frame volume access patterns are much more dynamic. I hope to model them further and to work out a practical and novel caching scheme.

5. REFERENCES

[1] Benjamin Watson and David Luebke. The Ultimate Display: Where Will All the Pixels Come From? *Computer*, 38(8):54–61, aug 2005.

[2] High-definition multimedia interface specification version 1.4. HDMI Founders, June 2009.

[3] High-definition multimedia interface specification version 1.4a extraction of 3d portion. HDMI Founders, March 2010.

[4] T Richardson, Q Stafford-Fraser, K R Wood, and A Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, 1998.

[5] Wirelesshd specification version 1.1 overview, May 2010.

[6] Wi-Fi Alliance. Best Practices Document for Wi-Fi CERTIFIED Miracast ł Devices version 1.0, 2014.

[7] Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D. Kirchner, and James T. Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques - SIGGRAPH '02*, SIGGRAPH '02, page 693, New York, New York, USA, 2002. ACM Press.

[8] Nirnimesh, P. Harish, and P.J. Narayanan. Garuda: A Scalable Tiled Display Wall Using Commodity PCs. *Visualization and Computer Graphics, IEEE Transactions on*, 13(5):864 – 877, 2007.

[9] R. Singh, L. Renambot, A. Johnson, and J. Leigh. TeraVision: a distributed, scalable, high resolution graphics streaming system. In *IEEE International Conference on Cluster Computing (IEEE Cat. No.04EX935)*, pages 391–400. IEEE, 2004.

[10] Byungil Jeong, Luc Renambot, Ratko Jagodic, Rajvikram Singh, Julieta Aguilera, Andrew Johnson, and Jason Leigh. High-Performance Dynamic Graphics Streaming for Scalable Adaptive Graphics Environment. In *ACM/IEEE SC 2006 Conference (SC'06)*, pages 24–24. IEEE, nov 2006.

[11] Jianli Luo, Kaihuai Qin, Yanxia Zhou, Miao Mao, and Ruirui Li. GPU rendering for tiled multi-projector autostereoscopic display based on chromium. In *Visual Computer*, volume 26, pages 457–465, 2010.

[12] T. H. Myer and I. E. Sutherland. On the design of display processors, 1968.

[13] Robert L. Myers. *Display Interfaces: Fundamentals and Standards*. Series in Display Technology. Wiley, 2003.

[14] Charles D Estes and Ketan Mayer-Patel. Moving beyond the framebuffer. *Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video - NOSSDAV '11*, page 93, 2011.

[15] Charles D Estes and Ketan Mayer-Patel. The n-Dimensional Display Interface A More Elastic Narrow Waist for the Display Pipeline. In *MMSYS*, 2012.

[16] Charles D Estes and Ketan Mayer-Patel. Video Killed The Data Store Extending the n-Dimensional Display Interface for Full Screen Video. In *ACM Multimedia*, Brisbane, Australia, 2015.

[17] McMillan, L. (Sun Microsystems Inc., Research Triangle Park, NC, USA) and L. Westover. A forward-mapping realization of the inverse discrete cosine transform. In *Data Compression Conference*, pages 219 – 228, Snowbird, UT, USA, 1992.

[18] Vesa digital packet video link standard: Version 1. Video Electronics Standards Association, April 2004.

[19] Mitsubishi electric diamond vision is dallas cowboys' choice for new stadium. Press Release, April 2008. http://www.businesswire.com/news/home/20080416005327/en.

[20] Ibm introduces world's highest-resolution computer monitor. Press Release, June 2001. http://www-03.ibm.com/press/us/en/pressrelease/1180.wss.

[21] The insane hardware driving the world's biggest led billboard. website. http://gizmodo.com/#!5096475/the-insane-hardware-driving-the-worlds-biggest-led-billboard.

[22] New displayport 1.4 standard can drive 8k monitors over a usb type-c cable. website. http://arstechnica.com/gadgets/2016/03/new-displayport-1-4-standard-can-drive-8k-monitors-over-a-usb-type-c-cable/.

[23] Robert W Scheifler and Jim Gettys. The x window system. *ACM Trans Graph*, 5(2):79–109, 1986.

[24] Tristan Richardson, Frazer Bennett, Glenford Mapp, and Andy Hopper. A ubiquitous, personalized computing environment for all: Teleporting in an X Window System Environment. *IEEE Personal Communications*, 1(3):6–12, 1994.

[25] Kenneth (IBM T.J. Watson Research Center Ocheltree, Steven (IBM T.J. Watson Research Center Millman, David (Teradici Corporation) Hobbs, Jason (Columbia University) Nieh, and Ricardo (Columbia University) Baratto. Net2Display: A Proposed VESA Standard for Remoting Displays and I/O Devices over Networks. In *ADEAC*, 2006.

[26] F. Jutand, Z.J. Mou, and N. Demassieux. DCT architectures for HDTV. *1991., IEEE International Sympoisum on Circuits and Systems*, pages 196–199, 1991.

[27] Henry Fuchs. Distributing a Visible Surface Algorithm Over Multiple Processors. In *Proceedings of the 1977 annual conference on - ACM '77*, pages 449–451, 1977.

[28] Henry Fuchs, Jack Goldfeather, Jeff P Hultquist, Susan Spach, John D Austin, Frederick P Brooks Jr., John G Eyles, and John Poulton. Fast Spheres, Shadows, Textures, Transparencies, and Image Enhancements in Pixel-planes. *SIGGRAPH Comput. Graph.*, 19(3):111–120, July 1985.

[29] Steven Molnar, John Eyles, and John Poulton. PixelFlow: high-speed rendering using image composition. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques - SIGGRAPH '92*, SIGGRAPH '92, pages 231–240, New York, New York, USA, 1992. ACM Press.

[30] Turner Whitted, Jim Kajiya, Erik Ruf, and Ray Bittner. Embedded Function Composition. In *Proceedings of the Conference on High Performance Graphics 2009*, HPG '09, pages 47–50, New York, NY, USA, 2009. ACM.