A GPU Cluster Without the Clutter: A Drop-in Scalable Programmable-Pipeline with Several GPUs and Only One PC

Eric Penner* University of Calgary
Ryan Schmidt† University of Calgary
Sheelagh Carpendale‡ University of Calgary

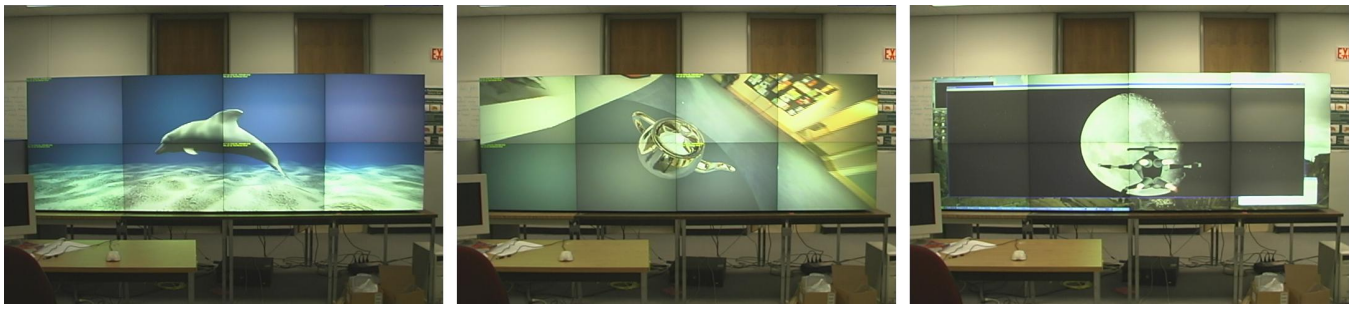

Figure 1: Unmodified Direct3D samples running on an 8 projector wall. Fullscreen and windowed modes and cubemap support is shown.

Abstract

An interactive multi-display cluster-style rendering system is presented which runs on a single PC rather than a cluster of PCs. Many Direct3D features, including support for a programmable pipeline, are supported via a drop-in library replacement for Direct3D. While the presented profiling results are bandwidth limited, PCI cards have been used. It is expected that the greater bandwidth of PCI express will produce results equivalent to those achieved with a rendering cluster system.

CR Categories: I.3.2 [Graphics Systems]: Graphics systems—Distributed/network graphics

Keywords: graphics cluster,distributed rendering, Direct3D, PCI express, SLI

1 Introduction

Traditionally, multi-screen interactive and immersive rendering environments have been driven either by graphics supercomputers, such as SGI's Onyx systems, or by cluster-rendering systems, such as Chromium [Humphreys et al. 2002]. Supercomputer-based systems provide excellent performance but come at a prohibitively high cost. Cluster-based rendering systems distribute the rendering load across multiple commodity PCs via high-speed networking, achieving comparable results at a much lower cost.

*e-mail: pennere@cpsc.ucalgary.ca
†e-mail:rms@cpsc.ucalgary.ca
‡e-mail:sheelagh@cpsc.ucalgary.ca

Our work questions the assumption that one PC lacks the power to support graphics-intensive applications which require multiple high-resolution displays, such as fully immersive environments and display wall environments. To this end, we have developed a cluster-style rendering system that utilizes a cluster of GPUs in a PC, rather than a cluster of independent PCs. We have tested the system with 10 displays and can theoretically support up to 20 displays at resolutions up to 38 megapixels, at interactive frame rates. We also address providing basic support for the programmable GPU pipeline, which is increasingly utilized in 3D visualization.

2 Related Work

During the last few years, feature rich single-display graphics interfaces with programmable pipelines have become available at a very low cost, which has prompted research into the use of clusters of commodity PCs to build one high-resolution interactive rendering system [Bierbaum 2000; Staadt et al. 2003; Humphreys et al. 2002; Tarault et al. 2005]. One of the primary challenges in building such a system is the efficient sharing of data amongst all the PCs in the cluster. Even with a gigabit network, this has proven to be a major bottleneck for such systems, and much research has been devoted to the efficient use of network bandwidth. Another major challenge for cluster rendering systems is providing a regular application programming interface (API) that makes existing applications easy to port. Chromium [Humphreys et al. 2002] tackles this challenge by providing a drop-in OpenGL library that hides the complexities of the cluster while providing a standard OpenGL interface. However, while it provides a regular interface to a fixed-function pipeline, Chromium lacks a lot of OpenGL's features due the problems inherent in running over a distributed network. It also provides little support for a programmable pipeline, which has become an important feature in recent years in creating visually realistic effects and interactive applications.

To utilize the new PCI express interface for commodity PCs, nVIDIA®SLI™makes use of several graphics cards in one PC. This technology is used to make displays faster rather than to support more of them, but it shows that using several graphics cards in one PC is feasible due to the dedicated bandwidth provided by

PCI express. While cluster rendering systems have become very effective for most high-end rendering environments, little research has questioned how much the cluster of PCs is actually needed with the availability PCI express graphics cards and graphics cards that support 2 or even 4 displays. The common reasoning is that one PC is simply not powerful enough to support high-end rendering tasks [Tarault et al. 2005].

3 Architecture

The new PCI Express (PCIe) bus can support several graphics boards with a dedicated bandwidth of up to 8000MB/s per card (PCIe *x16*). Most common PCs have one PCIe *x16* slot or two *x8* slots, and several more *x1* slots which run at 500MB/s. Even at *x1* speeds, this is 4 times the bandwidth of the gigabit networking commonly found in cluster-rendering systems, and the bandwidth is dedicated for each PCIe card. Essentially, PCIe provides a very high-bandwidth interconnect solution for cluster rendering, however the cluster is built from multiple GPUs in a single computer, rather than from multiple computers. We have developed a drop-in library replacement for Direct3D that supports hardware-accellerated rendering of existing applications across up to 20 displays. No modification of the original application is required. This is achieved by transparently rendering to independent Direct3D windows created for each display.

4 Implementation

Since our system currently only requires one PC, the management of multiple GPUs is heavily simplified. For example, drawing calls and state changes can be directly forwarded to each real interface (Direct3D device). However, some areas still needed special attention. The system is flexible in that it supports any combination of Direct3D-compatible video boards. We deal with the differences between interfaces by exposing the 'least common denominator' of features that are available on all the interfaces through our virtual interface.

4.1 Distributing Data

When a resource (eg. vertexbuffer, texture, shader, etc.) is locked for writing, we create a system memory scratch space that mimics a real resource. When the resource is unlocked, the data is copied to each real interface. The frame buffer is distributed such that each interface holds only the necessary area of the framebuffer. This is accomplished by intercepting the *SetTransform()* call and modifying the projection matrix for each interface. For windowed acceleration, a child window is dynamically positioned for each Direct3D interface to properly clip the rendering on each display.

4.2 The Programmable Pipeline

Since the programmable pipeline has no notion of a projection matrix, but rather only shader registers and code, a heuristic was developed to handle the projection matrix modification. Calls to *SetVertexShaderConstant()* are examined to determine if the content is being set to 16 floats (likely to be a matrix), and if the last float in the array is not equal to 1 (as is the case for perspective projection matrices). This heuristic is not guaranteed to be correct, however we found it sufficed for all Direct3D games and demos we tested. In the future we would like to examine the actual shader instructions and apply the projection matrix update to the output position register.

5 Results

After implementing just the basic functions and interfaces in the Direct3D library we were able to run most of the Direct3D demos on our 8-display wall using 4 dual-head GPUs. The frame rate was almost identical to that found when the application ran on just one interface. One major bottle-neck occurred when dynamic vertices were used and vertex buffers were updated each frame. If the application was bound by vertex streaming performance, then performance scaled inversely with the number of interfaces used. As noted, the PCI bus is very bandwidth limited. It is likely that the enhanced bandwidth of the PCIe bus will significantly reduce this bottleneck.

6 Conclusion

We have introduced a cluster-style scalable rendering system that supports a fully featured fixed and programmable pipeline and can support up to 20 displays. While our system cannot scale beyond the available PCIe slots available in a given PC, we believe that with 2-headed or 4-headed cards, the 5 PCIe slots commonly available is enough for many high-end rendering applications. By expanding on our system and providing extensions beyond Direct3D's functionality, it should be simple to use our library to support immersive environments with passive stereo.

References

BIERBAUM, A. 2000. *VR Juggler: A Virtual Platform for Virtual Reality Application Development*. Master's thesis, Iowa State University.

HUMPHREYS, G., HOUSTON, M., NG, Y., FRANK, R., AHERN, S., KIRCHNER, P., AND KLOSOWSKI, J., 2002. Chromium: A stream processing framework for interactive graphics on clusters.

STAADT, O. G., WALKER, J., NUBER, C., AND HAMANN, B. 2003. A survey and performance analysis of software platforms for interactive cluster-based multi-screen rendering. In *EGVE '03: Proceedings of the workshop on Virtual environments 2003*, ACM Press, New York, NY, USA, 261–270.

TARAULT, A., CONVARD, T., BOURDOT, P., AND VEZIEN, J.-M. 2005. Cluster-based solution for virtual and augmented reality applications. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, ACM Press, New York, NY, USA, 293–296.

VOSS, G., BEHR, J., REINERS, D., AND ROTH, M. 2002. A multi-thread safe foundation for scene graphs and its extension to clusters. In *EGPGV '02: Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 33–37.