MINT/VXF - A High-Performance Computing Framework for Interactive Multimedia

Rama C. Hoetzlein
Media Arts & Technology Program
University of California Santa Barbara
Santa Barbara, CA 93106-5110
rch@umail.ucsb.edu

Dennis Adderton
Media Arts & Technology Program
University of California Santa Barbara
Santa Barbara, CA 93106-5110
dennis@mat.ucsb.edu

ABSTRACT

The development of a cross-disciplinary system for immersive, real-time multimedia requires the solution to several challenges simultaneously. Systems for interactive tiled displays range from low-level solutions via ChromiumGL [5] to distributed scene graphs with OpenSG and CGLX [1]. Libraries such as DII and SuperCollider are designed for input device abstraction and audio respectively[6]. However, integrated multimedia systems covering all of these needs while providing built-in, dynamic primitives for interdisciplinary research do not yet exist. We present MINT/VXF, a framework for high-performance visual computing and research in interactive multimedia. Built using the MINT core event system, GameX graphics engine [3], and NVIDIA's CUDA for hardware-based computing, MINT/VXF introduces a novel scene graph architecture which abstracts GPU-based node evaluation to selectively leverage GPU resources in distributed multimedia systems.

1. INTRODUCTION

MINT/VXF is an open source multimedia framework designed to enable new media art and scientific research by providing a consistent structure as a broadly designed meta-library for C++ which integrates specialized libraries from different disciplines. MINT, the core event system, was developed as an NSF Interactive Multimedia IGERT project (2006-2008) and focused on basic integration of subsystems in multimedia systems via event message passing. Specific goals of MINT/VXF include support for common tasks in both science and the arts, such as automatic cluster rendering configuration on arbitrary displays, seamless video and device integration, precise timing of sound and graphic events, and integrated tools for scientific visualization.

While frameworks in a particular discipline become more encompassing of their fields, they expand in power yet typically remain dedicated to their individual domains. MINT was conceived from the bottom up as a meta-library for integrating the tools of other disciplines. Only a light-weight wrapper with event passing is needed to incorporate new modules. MINT is therefore conceived as a design tool where researchers can work collaboratively on dedicated software projects.

While we imagine MINT/VXF being used broadly by scientific and artistic communities, practical efforts have focused on supporting high-performance computing with the Allosphere, a 30 foot immersive display environment in the Media Arts and Technology Program at UCSB, as a target test space [4]. To that end one of the key development areas of MINT is automatic detection and configuration of multimedia networks. Unlike other tools in this area, MINT/VXF uses a platform independent, client-server window manager to automatically scan, connect and create network connections and windows on available render servers running either DirectX or OpenGL.

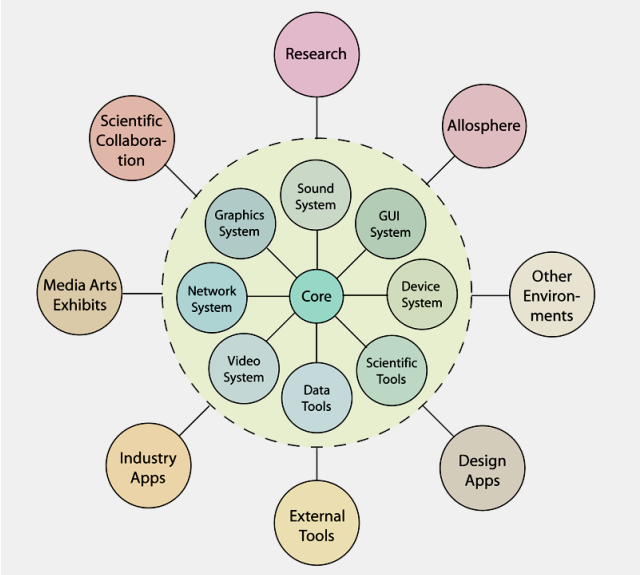

Figure 1: Overall design of MINT/VXF with internal sub-systems enabling external applications.

2. DEVELOPMENT

MINT/VXF was designed from the onset with integration in mind. The core event system is used by all sub-systems for basic message passing with serialization and timing control. The networking system transmits events over computer networks defined by task area. The graph system provides built-in dynamic primitives for all media tasks while allowing selective acceleration on the GPU. Finally, multimedia subsystems for audio, video, and graphics each interact with the scene graph to perform hardware specific input and output tasks. These layers are integrated to allow user applications to work with different multimedia domains.

The networking system of MINT/VXF is one of the cen-

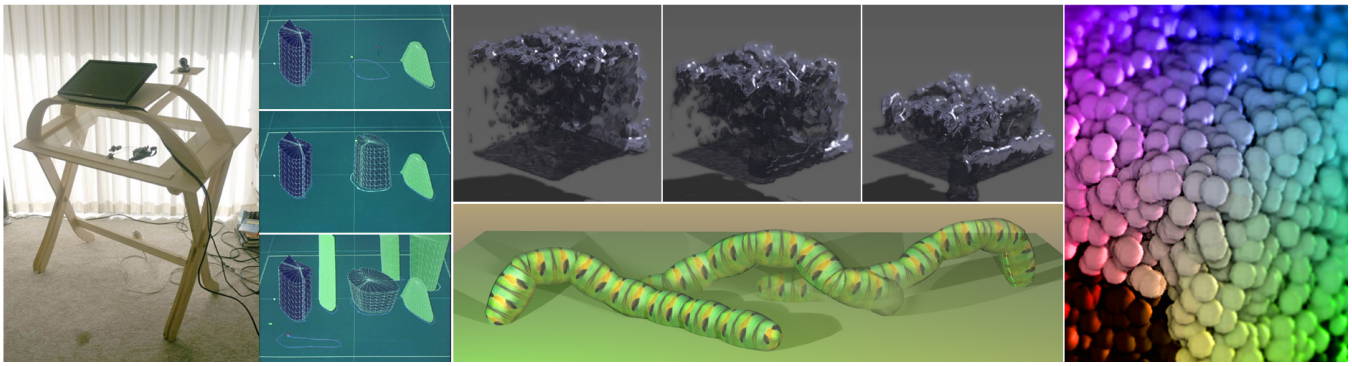

Figure 2: Several applications with dynamic geometry developed using MINT/VXF including a) 3D Multitouch Desktop, b) Fluid simulation running on either CPU or GPU, c) Kinematic animation of a jointed, articulated "inchworm", and d) Rendering of a point data set with shadow maps, screen-space ambient occlusion and depth of field.

tral aspects of the current design. MINT/VXF is similar to Open Scene Control in that it builds TCP and UDP connections for event-based message passing over a network. [8] We extend this with MINT/VXF to allow for virtual client-server networks for each domain subsystem. This greatly simplifies overall design by allowing a network of machines for audio, video, graphics, and input to overlap and coexist while keeping systems logically independent. Any set of computers may be designated as an input, audio output, or graphics output clients, or any combination of these. In this way, multiple machines may be dedicated to various tasks by domain. MINT/VXF may also act as both client and server over all subsystems, so applications can run on a single computer as one process.

The object graph system provides built-in nodes for images, sounds, mesh geometry, visualization objects, and point clouds, among others, to provide procedurally generated objects utilizing both the CPU and GPU with NVIDIA's CUDA Architecture. Based on advanced scene graphs found in modeling packages, such as Maya's dependency graph [2], the VXF object graph maintains functional relations in addition to geometric hierarchy. This allows nodes to notify one another as data changes and to rebuild themselves as needed at run time. To allow for rendering independence, each node has a proxy node within the rendering subsystem which maintains vertex buffers, geometry, and texture data on the GPU. Proxy nodes can communicate with one another to allow efficient GPU-to-GPU processes execution. While many CUDA-based applications are currently domain specific, MINT/VXF therefore allows multiple, distinct objects to take advantage of GPU parallelism in the same application.

Unlike other scene graph frameworks, each scene node retains both a user state and a dynamic state which may be updated by any MINT/VXF sub-system. This allows each output client to selectively evalute nodes downstream. An example of rebuilding dynamic state is the distributed rendering of volumetric surfaces. In this example, each client recieves volumetric data as an input file over the network. The input data uniformly updates the volume node on each client, but the downstream visualization node (e.g. marching cubes) can selectively evaluate the volume only as needed to reconstruct the visible surface on that client display. More generally, dynamic rebuilding allows partial node evaluation to be localized to each client while maintaining uniformity of the user application.

To support cluster rendering on tiled displays MINT/VXF uses a replicated scene graph approach. Similar to other systems, MINT/VXF collects input messages from input clients and forwards them to output clients running the user application, which updates the scene graph on each client machine [1]. Network traffic is minimized since most input events are from basic hardware devices. Many existing distributed multimedia systems define input as device-generated, and restrict network traffic to input messages. However, for applications where the results of intensive computational tasks must be broadcast to multiple downstream clients this becomes too restrictive. Input is thus defined in MINT/VXF as any information which cannot be deterministically evaluated by the client scene graph. Input messages may therefore include geometry, video, and other multimedia as needed.

3. RESULTS

MINT/VXF currently builds under Windows, Cygwin and Linux with a rendering module for OpenGL. Several prototype applications were developed to simultaneously test and extend the capabilities of MINT/VXF. The first of these, a 3D Multitouch Tabletop, uses two cameras to detect the 3D positions of colored LEDs which allow users to sketch curves on a two dimensional glass surface and then extrude those curves into space using gestures. Due to the nature of the interaction there are no mode changes. Users simply draw on the surface, lift their fingers, then pick the curve they wish to loft. Currently limited to extruded shapes, planned extensions to MINT/VXF may allow for more complex modeling tasks.

The second system developed in conjunction with MINT/VXF is a fluid simluation of free surface flows using smoothed particle hydrodynamics. The fluid system is a node which derives from a point set, which itself derives from a basic geometry class. Thus, due to integration with MINT/VXF,

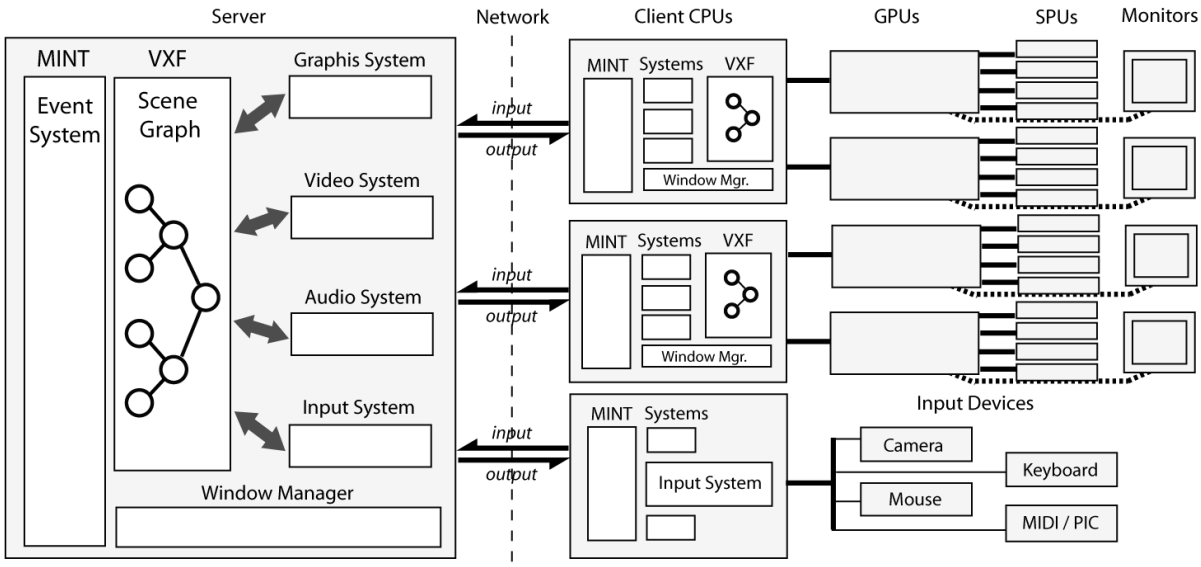

Figure 3: Architecture of MINT/VXF integrating multiple subsystems, tiled displays, and scene graph nodes capable of utilizing CPU or GPU resources. Each subsystem is able to interact with the scene graph.

the fluid class automatically benefits from point-based rendering, and volumetric rendering of points as metablobs. The fluid simulator can selectively switch bewteen CPU and GPU execution using CUDA. While rendering presently supports screen-space ambient occlusion, shadow maps, and depth of field, this application is currently being tested in the multi-display configuration.

A third application, currently in progress, is being developed to visualize the Schrödinger equations in the context of the Allosphere. Our goal is to use the GPU capabilities of MINT/VXF to allow scientists to visualize and interact with simulation parameters in quantum wave propagation. Other applications successfully tested in MINT/VXF include importing and smoothing detailed geometric meshes, and kinematic simulation of an articulated, segmented caterpillar; both running on networked tiled displays.

Although MINT/VXF does not currently optimize hardware rendering state like high-performance system such as IRIS Performer or OpenSceneGraph [7], this may be added. Instead our performance contributions focus on general evaluation of scene graph nodes for selective GPU acceleration, and GPU-based hardware rendering with buffers.

Overall, MINT/VXF presents novel solutions in several areas of real-time multimedia including virtual networks, GPU-enabled scene graphs, autonomous window management, input type abstraction, and proxy nodes for rendering. Due to its scope MINT/VXF is an on-going project. None the less, several prototype applications have been developed and new applications are currently underway. We intend to continue to develop MINT/VXF with increasingly sophisticated projects as the system matures. Future goals include providing a wider variety of geometric nodes, a more substantial audio subsystem, and additional rendering modules to support raytracing.

This research was funded in part by an NSF IGERT grant on Interactive Digital Multimedia (DGE- 0221713), and with the support of JoAnn Kuchera-Morin (Media Arts & Technology, Director) and Tobias Hollerer (Computer Science).

4. REFERENCES

[1] K.-U. Doerr and F. Kuester. CGLX: A cross-platform cluster graphics library, 2008.

[2] D. Gould. *Complete Maya Programming: An Extensive Guide to MEL and C++ API*. Morgan Kaufmann, San Francisco, CA, 2003.

[3] R. Hoetzlein and D. Schwartz. GameX: a platform for incremental instruction in computer graphics and game design. In *ACM SIGGRAPH 2005 Educators program*, number 36, New York, NY, USA, 2005. ACM Press.

[4] T. Hollerer, J. Kuchera-Morin, and X. Amatriain. The Allosphere: A large-scale immersive surround-view instrument. In *Proceedings of the 2007 Emerging Display Technologies Workshop at SIGGRAPH*, 2007.

[5] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. *ACM Transactions on Graphics (TOG)*, 21(3), 2002.

[6] J. McCartney. SuperCollider: A new real time synthesis langauge. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 257–258, 1996.

[7] J. Rohlf and J. Helman. Iris performer: a high performance multiprocessing toolkit for real-time 3d graphics. In *Proceedings of the 21st ACM SIGGRAPH Annual Conference on Computer Graphics*, New York, NY, 1994. ACM Press.

[8] M. Wright and A. Freed. Open sound control: A new protocol for communicating with sound synthesizers. In *Proceedings of the International Computer Music Conference*, Thessaloniki, Greece, 1997.