iTILE FRAMEWORK FOR CONSTRUCTING INTERACTIVE TILED DISPLAY APPLICATIONS

Seokhwan Kim
Department of Computer Science, College of Software, Sangmyung University
7 Hongji_dong, Jongno_gu, Seoul, Republic of Korea

Minyoung Kim, Yonjoo Cho
Department of Digital Media Technology, College of Software, Sangmyung University
7 Hongji_dong, Jongno_gu, Seoul, Republic of Korea

Kyoung Shin Park
Multimedia Engineering, Division of Computer, College of Engineering, Dankook University
San 29, Anseo_dong, Dongnam_gu, Cheonan, Choongnam, Republic of Korea

Keywords: Tiled Display Framework, Large-sized and high-resolution display, Interaction.

Abstract: We describe a new scalable display framework, called iTILE, designed to help ease the development of interactive graphics applications that run on a large tiled display. The framework supports the execution of multiple interactive scene-graph applications, synchronized rendering, distributed data sharing, and unified user interface mechanism on a cluster-driven tiled display. It enables the application window launching, moving and resizing on a tiled display that requires a dynamic reconfiguration of rendering nodes. It also provides the standardized method to support various input devices for the interactive contents. This paper presents the design and the implementation of iTILE architecture and a few applications built using this framework.

1 INTRODUCTION

Nowadays, there are many large public information displays available in public spaces, such as the airports, banks, and shopping centers. Such displays are also found in the subway stations or at the bus stops to provide information about the map of the near district and other public transportation schedules. However, current public information displays are dedicated for display only, and so they do not allow users to interactively acquire more detailed information.

Recently, large high-resolution wall displays have been used in some industry fields, such as the military situation room, the 1:1 scale design of mechanics, high-definition video streaming, and the visualization of three-dimensional massive dataset. (Ni et. al., 2006). The high-resolution display helps users improve the efficiency of their tasks since it allows users to see the fine detail view of the contents if they get close to the displays and the overview of the overall contents at a distance.

The scalable high-resolution displays can be built using multiple projectors or multi-screen tiled displays. Relatively large-sized and medium resolution display is constructed with a few beam projectors. It can generate a seamless display that looks like one large single display if the projectors are aligned perfectly well. The tiled display constructed using many LCD panels does not need recalibration. In addition, LCDs are relatively cheap, and the colour variations across multiple screens are low. The LCD tiled display is most widely adopted because of its cost effectiveness and scalability.

There are several frameworks developed to support scalable high-resolution tiled display. The examples include SAGE (Jeong et. al., 2006), WireGL/Chromium (Humphreys et. al., 2002),

Kim S., Kim M., Cho Y. and Park K.
iTILE FRAMEWORK FOR CONSTRUCTING INTERACTIVE TILED DISPLAY APPLICATIONS.
DOI: 10.5220/0001794603670372
In *Proceedings of the Fourth International Conference on Computer Graphics Theory and Applications (VISIGRAPP 2009)*, page
ISBN: 978-989-8111-67-8

Garuda (Nirnimesh and Narayanan, 2007), and Equalizer (http://www.equalizergraphics.com). However, these works mostly focused on scalability or distributed rendering of the tiled display and not much considered user interaction. With the advent of technology, we envision that the large high-resolution tiled displays will allow users to interactively view and use the multiple applications simultaneously at the same time.

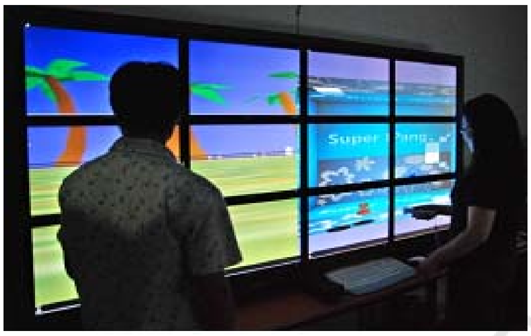

Figure 1: iTILE applications running on the tiled display system.

In this paper, we describe a new interactive tiled display framework, called iTILE, designed for supporting easy construction of interactive applications that can run on a scalable high-resolution tiled display. Figure 1 shows iTILE applications running on a 4x3 tiled display system that uses one master and six slave computers. On the left is the Field educational virtual reality application running on three slave nodes and on the right is the Super Pang interactive 3D game that runs on all six slave nodes.

The main features of iTILE framework are the execution of multiple three dimensional scene graph applications, synchronized rendering and distributed data sharing across the screens, and standardized way to support various input devices on a PC-cluster driven distributed tiled display system. It also supports the application window launching, moving and resizing on a tiled display that requires a dynamic reconfiguration of rendering node in real-time.

In following section, we will review some works related to our framework and then describe the design of iTILE system architecture and some details implementations. Next, we briefly describe some applications developed using iTILE framework and future research directions.

2 RELATED WORK

WireGL is a parallel rendering framework on a cluster computer system, designed to help render three dimensional graphics by maximizing their graphics power. It was not specifically designed for tiled display, and more aims at parallel rendering utilizing the distributed systems. It, however, could support a sufficiently large image rendered on the tiled display by dividing the large image into the small sized image fit to each monitor. In WireGL, when all node computers complete the rendering of their part, they send the image to the central server. Then, the server makes one large image and then divides it to the small pieces of images for each node. However, the performance is degraded when the image gets very large. For example, if the image resolution is 3200 X 2400 then the frame rate gets below 20. WireGL also supports the hardware implementation of tiled display rendering using Lightning-2. (Stoll et. Al., 2001). Chromium is the successor of WireGL; also designed for parallel rendering framework. However, its performance and mechanism for tiled display rendering is the same as WireGL.

Equalizer developed by University of Zurich is another parallel rendering framework similar to WireGL and Chromium. Its rendering engine is also based on OpenGL. Moreover, it sends the additional rendering factors (such as, the location and the orientation of virtual camera, and the viewport) to each node, and then each node renders to show its own part of the image. Equalizer provides the user interface class for event handling. However, the events are controlled by the GUI system rather than directly from the operating system. That is, the event processing routine for the input device should be embedded to the GUI system's loop. Thus, if a new input device driver does not support transferring inputs to the GUI system, the Equalizer application developers have to implement this feature.

Garuda developed by Deemed University is an Open Scene Graph (OSG) based three dimensional graphics application framework for the tiled display. Its performance is fairly good even under the 100 MB/s Ethernet and nVidia's 6600GT graphics chipset. The main purpose of this framework is to support OSG application without any modification, on the low-end PC cluster system. This framework synchronizes all the nodes using a message passing mechanism through a network. However, it does not support multiple windows on the tiled display and only support a mouse and keyboard as input devices.

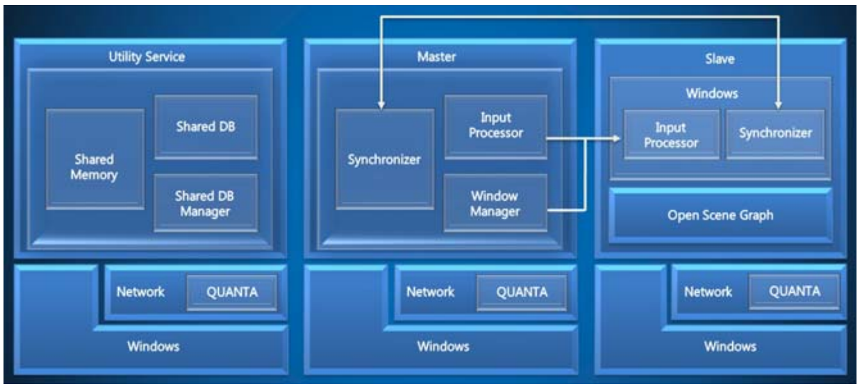

Figure 2: The Overall Architecture of iTILE Framework.

SAGE, developed by EVL, University of Illinois at Chicago, is a high-resolution video streaming middleware through an extremely fast network. SAGE uses "the virtual frame buffer." That is, the images (generated from one computer or cluster computers) are transferred to the virtual frame buffer, and then the display nodes read the frame buffer via a network and present the image. SAGE also supports adjusting the size or location of the image. Its virtual frame buffer is similar to WireGL's distributed rendering mechanism in which it gathers the image from the nodes and redistributes to them. SAGE uses an optical network to improve the performance by reducing network delay.

Most of existing tile display frameworks focused on scalability or distributed rendering, and hence they provide limited user interaction schemes. SAGE is designed to stream the screenshots of the various applications from other computers onto the tiled display system. SAGE supports multiple windows using scripts, and it also allows moving and resizing the application window. The primary goal of WireGL/Chromium and Equalizer is to distribute and balance the rendering loads to multiple distributed computers. Equalizer supports heading tracking of user to interact with the application on the tiled display. Garuda aims at running standalone Open Scene Graph applications on a low-cost tiled display system without modifying any source code. Garuda only supports keyboard and mouse interaction to interact with the application.

3 ITILE FRAMEWORK

The goal of iTILE framework is to support easy construction of multiple interactive graphics applications running on a tiled display system. It also provides dynamic application window management on the tiled display as well as user interaction mechanism for iTILE applications. Figure 2 shows the architecture of iTILE framework. The framework consists of iTILE master, slave, and framework utility service modules.

The master is in charge of dynamic application window management and input event processing to control the applications running on the slave nodes. The slaves render the scenes of tiled display applications. The framework utility services support the features needed by both master and slaves, such as shared database, shared database manager, and shared memory. The master, slave, and framework utility service are communicating each other via message passing over the network. The framework is written in C++ and Microsoft winsock2 and QUANTA networking library. (DeFanti et. al., 2003). The rendering modules of the slaves are used Open Scene Graph (OSG) graphics library. (http://www.openscenegrap.org).

The communication mechanism is implemented in the Network Transfer Module, using QUANTA networking library. We also use PGM Reliable Transport Protocol Specification (RFC 3208) for reliable multicast. PGM guarantees the transmission and the order of packet like TCP protocol does. However, it needs to resend the data packet to all computers in a multicast group even if only one host computers does not receive the packet. Consequently, the host which already gets the packet also receives the same packet twice when this retransmission occurs. iTILE gets around this retransmission problem by limiting the size of the data packet and putting packet number.

3.1 Utility Service

The framework utility service has a set of independent modules like shared DB, shared DB manager and shared memory.

The shared DB is used for sharing data among all slave nodes, and the shared DB manager makes sure to maintain data consistency and synchronization in shared DB. Shared memory is considered as a shared DB on a local machine. Shared memory is independent to the master but it runs on the same master computer. If the application writes data in shared memory, the framework reads the data and executes specified routine.

Shared DB is an implementation of distributed shared memory (under we will use the term DSM). When the users navigate in the virtual world that consists of static objects, the master's input processor is used to distribute the virtual camera information according to the viewing perspectives required in the slave nodes. However, when the world is more dynamic, DSM is used to share the dynamic object's state changes synchronously across all slaves. Shared DB is an independent process that can run on any slave machine, and the synchronized data in shared DB can be accessed by any process at any time.

Shared DB manager provides the synchronization mechanism for data in shared DBs. iTILE's shared DB manager enhances the weak consistency model. (Dubois et. al., 1986). In our implementation, the slave node first asks the shared DB manager to get a lock to the shared memory. Then, the slave can access the critical memory section if the shared DB manager allows it. After updating the data, the slave node needs to release the lock back to the shared DB manager.

Our implementation of shared memory service is a wrapper of shared memory served by operating system. Shared memory is used in shared DB to enable accessing data across multiple processes. It is also used for new input device. The master's input processor reads the specific range of shared memory for a new input device and then executes the interaction routine. The map between interaction routine and data needs to be specified in the program. Using the iTILE's shared memory, it is easy to create a routine that writes the input data into the shared memory.

3.2 Master Module

The master module is composed of window manager, input processor and synchronizer. Master's window manager processes the windowing events, such as moving or resizing application windows. Master's input processor handles the application events, such as user navigation and manipulation of virtual objects. Master's synchronizer works with slaves' counterparts to update the application's screen rendering simultaneously.

Master's window manager controls the information about the application windows for all slave nodes, such as window location and viewport. It enables dynamic creation and removal of multiple application windows on the tiled display system at run-time. It keeps monitoring whether the window events are happened or not. If so, the window manager re-configures the location and size of the window as well as recalculates the graphical viewport and view frustum to properly render three dimensional environments in each slave's application window. The window manager sends the window information via multicast protocol to each slave's input processor.

Master's input processor is another core module of iTILE framework. Input processor receives events based on the values from input devices. If it reads out the values from input devices, it generates a message (in pre-defined form) and sends it to the slave nodes. It also sends the virtual camera information such that slave nodes can render right view perspectives. As shown in Figure 2, the messages from both input processor and window manager are sent to slave's input processor.

When a new application window is launched from the master, new synchronizer threads get also created in both master and slaves. When each slave node renders a scene and is ready to swap the frame buffer, it sends a notification to the master's synchronizer. When the master's synchronizer receives the notifications from every rendering node, it sends a rendering message to all slaves so that they can actually draw on their screens.

3.3 Slave Module

The slave module consists of Open Scene Graph renderer and slave's window. The slave's window consists of input processor and synchronizer. We employ the Open Scene Graph open-source graphics APIs for graphics rendering. Slave's input processor identifies the appropriate message handler for the application windows running on the slave node. The slaves also have a synchronizer for each iTILE application window to make sure all slaves rendering updated simultaneously.

Slave's input processor manages both windowing events and application specific events.

When the input processor receives the messages from the master, the input processor's message dispatcher determines the appropriate module for those messages. If the message is from master's window manager, the slave's window updates its size and location changes. If the message is from the master's input processor, the slave's input processor analyzes the message and then executes the message handler routine.

Slave's synchronizer sends the rendering status (such as, ready to render new frame buffer) to the master's synchronizer. Then, the master sends the allowance message to the slaves when all slaves are ready to swap their frame buffer. The slave's display buffer swapping is blocked until slave's synchronizer receives the allowance message from the master.

4 APPLICATIONS

Figure 3 shows users interactive with three iTILE framework applications on the 4x3 tiled display system, which was constructed with 12 LCD panels and one master and six slave computers. In this section, we will briefly discuss some applications written using iTILE framework: the Field educational virtual environment, the Moyangsung virtual heritage environment, the Super Pang interactive 3D game, and the simple OSG model viewer controlled by a Nintendo Wii remote controller.

4.1 Field Virtual Environment

Field is an educational virtual reality application originally developed by Electronic Visualization Laboratory, University of Illinois at Chicago for helping elementary students to teach scientific inquiry learning skills. There are many kinds of flowers and plants in the virtual field that consists of grass, gravel, and sand area. In field, users can freely navigate the world and observe the natural phenomena for scientific inquiry. This application is originally developed by using SGI Performer graphics library and YG virtual reality scripting framework (http://www.evl.uic.edu/yg/). We have re-implemented Field written using iTILE framework and OSG library.

iTILE framework provides several derived modules for the OSG rendering and input processing classes. Hence, simple OSG applications containing static object models and navigation can easily be ported to the iTILE applications by simply replacing the viewer and the manipulator classes for the master or the slave nodes. The efforts required converting this kind of OSG applications to iTILE's are relatively low – In the Field application, for example, less than ten lines of the source code are changed.

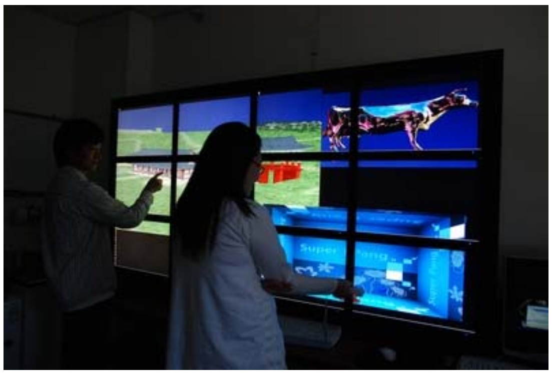
Figure 3: Users interact with three iTILE applications on the 4x3 iTILE tiled display system.

4.2 Moyangsung Virtual Environment

Moyangsung is a virtual reality cultural heritage environment designed for users to learn cultural background story of a Korean war-defensive castle. It is constructed on a hill as a fortress. There are four main gates and about twenty-two traditional buildings inside the castle, and bamboo and pine tree forests. Users can freely walk around the virtual Moyangsung to experience various cultural sites. This application is also written using SGI Performer graphics library and YG virtual reality scripting framework. Similar to Field, the Moyangsung application is simply re-written using OSG library and iTILE framework's master/slave viewer and manipulator classes.

4.3 Super Pang Interactive 3D Game

Our framework's distributed shared memory mechanism helps developers easily translate more dynamic VR applications, such as Super Pang, to the tiled display system. Super Pang is a simple OSG-based interactive 3D game developed by Sangmyung University. This game starts with a character agent and a balloon floating in the world; a player uses a keyboard or wii remote controller to move the character and shoot the laser beam to hit the balloon. The balloon would get split if the beam crosses it. Then, the player continues to shoot the balloons until all balloons get removed within a limited time.

Unlike Field, the Super Pang game world is more dynamic with moving objects and user interactions. We employed the iTILE's shared DB class to share the user input data and the game character's position, across the slave nodes to render the correct contents in the tiled display. The DSM is also used to update the state of all moving objects (i.e., balloons) in the 3D world across the slave computers.

4.4 3D Model Viewer

We extended a simple OSG application, osgViewer, to allow it to manipulate 3D object models with a Nintendo Wii remote controller (often called Wiimote). In this application, we developed a specialized input processor class, CMasterOSGTrackballManipulatorWithIO, for the master node. This specialized input device handling module simply reads Wiimote data and put them in the shared memory as specified in the master node's input processor. Then, users can use the new device for several user interactions, such as zooming in/out, moving the object model left/right/up/down, and rotating the model within the viewer application without modifying any source code of the program. This application demonstrates the convenience of our framework, which provides a mechanism to easily add new input device handling without changing any code in the application program.

5 CONCLUSIONS

Tiled displays offers scalability, large-format, and high-resolution used for various applications, such as high-resolution image, massive scientific visualization, video streaming, and design prototyping. However, the development of tiled display applications requires a lot of efforts. While several scalable high-resolution tiled display frameworks have been developed, they mostly focus on scalability or distributed rendering of computer graphics and not much considered user interaction.

This paper presented the design and implementation of iTILE framework. iTILE framework is designed to support easier and faster development of the interactive graphics applications for the scalable tiled display. This framework works on the PC-cluster driven tiled display system consisting of a master and multiple slave computers. It provides the window manager for executing multiple application windows, synchronized rendering, distributed data sharing using shared DB manager, and standardized mechanism to support various input devices.

We have also showed some examples where existing OSG applications can easily be ported to run on the tiled display using our framework. However, current iTILE framework is little tightly coupled to Open Scene Graph library and hence it only supports OSG-based applications. In our current framework, the interaction with the input device is needed to be specified in the application program. In the future, we will continue to improve our framework to decouple it from Open Scene Graph library and add the new standardized component structure for interaction scheme.

ACKNOWLEDGEMENTS

This research was supported by the Collaboration Research Project program, supervised by the Korea Research Council of Fundamental Science and Technology under the Korea Institute of Science and Technology Information.

REFERENCES

T, Ni., G, Schmidt., O, Staadt., M, Livingston., R, Bell., 1999. A Survey of Large High-Resolution Display Technologies, Techniques, and Applications. In *VR Conference 2006.* IEEE.

B, Jeong., L, Renambot., R, Jagodic., R, Singh., J, Aguilera., A, Johonson., J, Leight., 2006. High-Performance Dynmaic Graphics Streaming for Scalable Adaptive Graphics Environment. In *SC.* IEEE.

G, Humphreys., M, Houston., R, Ng., R, Frank., S, Ahen., P, D, Kirchner., J, T, Klosowski., 2002. Chromium: a stream-processing framework for interactive rendering on clusters. In *ACM Transactions on Graphics*. ACM.

H, P, Nirnimesh., P, J, Narayanan., 2007. Garuda: A Scalable Tiled Display Wall Using Commodity PCs. In *IEEE Transactions on Visualization and Computer Graphics.* IEEE.

G. Stoll., M. Eldridge., D, Patterson., A, Webb., S, Berman., R, Levy., C, Caywood., M, taveira., S, Hunt., P, Hanrahan., 2001. Lightning-2: a high-performance display system for PC clusters, In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques.* ACM.

E, He., J, Alimohideen., J, Eliason., N, Krishaprasad., O, Yu., Leigh., T, Defanti., J, 2003. Quanta: a toolkit for high performance data delivery over photonic networks. In *Future Generation Computer Systems*. ACM.

M, Dubois., C, Scherurich., F, A, Briggs., 1986. Memory Access Buffering in Multiprocessors. In *13th Annual International Symposium on Computer Architecture.* ACM.