Magic Input: A Multi-user Interaction System for SAGE Based Large Tiled-display Environment

Yihua Lou, Wenjun Wu, Hui Zhang
State Key Laboratory of Software Development Environment
Beihang University
Beijing, China
{louyh, wwj, hzhang}@nlsde.buaa.edu.cn

Abstract— In a large tiled-display environment, it is important to support multi-user interaction with juxtaposed applications for collaborative work. Although a few multi-user interaction systems have been developed by researchers, they often require modifications to desktop applications to become simultaneously accessible for multiple users. In this paper, we propose a novel multi-user interaction system for large tiled-display environments which needs no customizations to applications. To enable multiple users to cooperate within one application, three basic interaction strategies are introduced for different situations. Our experiment results show that this system not only enables users to interact with applications smoothly just like what they feel on their desktops but also presents them with more collaborative and immersive experience in a large tiled-display environment.

Keywords- SAGE, Multi-user interaction, large tiled-display environment, collaborative workspace

I. INTRODUCTION

Nowadays advanced scientific experiments and simulations are increasingly generating massive amount scientific data. As such data may be visualized in more than ten million pixels, using large tiled-display environments that can provide virtual desktops with tens of millions of pixels is becoming the trend for collaborative scientific researches. With such a virtual desktop, researchers can easily view and manipulate juxtaposed applications at the same time without any context switch such as cascading and resizing windows which are common practices in a window based desktop environment. For example, such displays are used in [1] as visualization instruments, enabling scientists to explore their problems more deeply. Moreover, it also creates a collaborative workspace for multiple researchers to analyze their data and have discussions simultaneously.

These environments are usually built of projector arrays or tiled LCD panels driven by a display cluster at the backend and there are a handful of software architectures to support such a configuration, which can be categorized into three types: distributing graphics primitives, running applications distributed and distributing pixels. WireGL [2] and its successor Chromium [3], take the approach of distributing graphics primitives. CGLX [4] follows the approach of running application distributed. Scalable Adaptive Graphics Environment (SAGE) [5] uses the approach of distributing pixels. As SAGE can achieve great flexibility and adaptability, we choose it as the software infrastructure for our own tiled-display environment and create a seamless collaborative environment.

To enable multi-user interaction in a SAGE driven environment, we propose a multi-user interaction system based on SAGE in this paper, enabling multiple users to interact with the target applications through keyboard, mouse and Wii Remote without customizing them.

In Section 2, we introduce the background on the design of SAGE. In Section 3, we review some related works and literature. Section 4 describes the design aspects of our system and Section 5 provides a detailed description of the system. In Section 6, we discuss the interaction strategy used in the system and in Section 7 demonstrate our experimental results. Section 8 briefly introduces some usages of the system, and Section 9 is the conclusion.

II. SAGE BACKGROUND

SAGE is a middleware that was developed by EVL-UIC. It is used as the software infrastructure to support stream pixels to any configuration and combination of displays that are driven by a computer cluster. With SAGE, the displays are converted to a single virtual desktop that contains windows, where each window is actually a view of visual contents of an application running remotely (referred as “target application” later in this paper). As described in [5], SAGE mainly includes four components: the Free Space Manager as a controller for all the operations on windows, the SAGE Application Interface Library (SAIL) that exposes a set of APIs for applications to communicate with Free Space Manager and stream pixels, SAGE Receiver that receives pixel streams and display them, and the SAGE User Interface that provides a simple graphics user interface for users to manipulate the windows freely, such as moving, resizing and rotating, as if they were local windows.

III. RELATED WORKS

As the advance of large tiled-display technology, delivering the content is not the only research focus any more. Instead, recent research efforts focus more on how to support collaborative user interaction in such an environment. We can categorize these researches into three groups based on their separate approaches.
The first research group mainly focuses on developing novel input technologies, such as the approaches described in [6], [7]. The second research group focuses on remote input redirection in a multi-user groupware or tiled-display environment, such as Mighty Mouse [8], PointRight [9] and IMPROMPTU [10]. Mighty Mouse is a cross-platform single display groupware tool which is proposed to support multi-user collaboration through input redirection based on VNC. It supports flexible movement, floor control and enhanced customization features, which makes collaboration smooth and seamless. PointRight is another input redirection system working in multi-machine, multi-user environments. It is designed with general and flexible input redirection system architecture and is used in some other systems such as Team Space [11]. IMPROMPTU is a collaboration tool which supports multi-user cooperation with the off-the-shelf applications in a Multiple Display Environment. The third research group focuses on extending the framework of existing tiled-display software to support multi-user interaction. Some researches such as Elvin [12] and iStuff [13] mainly deal with generic event notification services, while others are more complete as a multi-user interaction framework. An approach for ultra-high-resolution collaborative workspaces based on CGLX is proposed in [14], using multi-touch portable devices as the human interface device. The content of an application is streamed to the portable devices on which users can remotely manipulate the application, just as if it was running on the device. Multiple devices can be connected to the system to allow multiple users to manipulate simultaneously.

As SAGE itself focuses on how to distribute pixels effectively and only provides a simple user interface for basic window manipulation, a multi-user interaction system is presented in [15] to make better multi-user interaction experience. It consists of two parts: Direct Interaction Manager (DIM) and SAGE Widget Framework. The DIM is used to receive various device-specified events generated by input devices, convert them into pre-defined general events such as click, pan or zoom, and then send them to applications through SAGE’s Free Space Manager. The SAGE Widget Framework contains a set of common widgets which allows developers to write applications that can directly interact with users in a SAGE tiled-display environment. Although this interaction system provides a good solution, there are still several problems:

First, target applications must be refactored to use SAGE Widgets and connect their input handlers with DIM. Though these customizations are easy for some open-source or custom developed applications, it is not true for most commercial software packages.

Secondly, DIM converts the device specified events to general events, which may lose some important features of the device-dependent events and may disable some complex interaction operations. To address the issue, it can be more comfortable to use native input schemes built in operating systems that can support all device events than to implement a limited version that can only handle the general events.

Finally, both DIM and SAGE Widget Framework rely upon SAGE’s Free Space Manager for passing messages.

With the increase of coming input events, they bring heavier workload on Free Space Manager, which leads to slow response of the system. To reduce the overhead on Free Space Manager and improve the communication efficiency, it is appropriate to create a direct-through message channel between users and target applications bypassing Free Space Manager.

Our paper tries to solve the problems of the existing multi-user interaction systems and present a new multi-user interaction system named Magic Input for SAGE based tiled-display environments without requiring any source-code level modifications to target applications.

IV. DESIGN ASPECTS

To provide a general-purpose interaction system in a large tiled-display environment, we should make the system display-resolution neutral and capable of multi-user interaction over regular applications.

A. Adaptive

As the size of tiled-display environments varies, it is imperative for the interaction system to automatically adapt to different tiled-display sizes and different application window sizes, so that users can have a nearly consistent experience when using different tiled-display environments and different application window sizes.

B. Modification Free

Currently most proposed multi-user interaction systems are only designed for some particular applications, which means many non-customizable applications, especially the large amount of commercial software that do not offer either source codes or appropriate licenses for refactoring, cannot take advantage of such systems. Therefore, it is necessary for us to consider a new way to adapt these applications to support multi-user interaction in tiled-display environments while keeping them unmodified.

C. Multi-user

Usually there are many applications juxtaposed in the tiled-display for multiple users to operate simultaneously. Therefore, supporting simultaneous multi-user interaction is highly necessary for such interaction systems. Moreover, when multiple users are operating within the same application’s window, it is the system’s responsibility to coordinate these operations in order.

V. ARCHITECTURE OF MAGIC INPUT

Magic Input consists of three major parts: Input Manager, Input Client and Application Layer. Fig. 1 shows the architecture of Magic Input within the SAGE tiled-display environment.

A. Input Manager

Input Manager is the central controller of Magic Input. It receives input events from Input Clients, processes them and then forwards them to appropriate target applications. Due to the fact that one computer can support limited number of input devices, in order to enable multi-user interaction and
make the system more flexible, Input Manager is designed to use network interface for receiving and sending input events. Through the network, Input Manager can communicate with different Input Clients that capture device events, so that individual users can use their own input devices attached at their computers separately. To provide a flexible and robust service, Input Manager is designed to support dynamically adding and removing Input Clients, allowing users to join or leave freely at any time. Additionally, Input Manager is responsible for controlling the cursor positions on the tiled-display with each cursor representing an Input Client. It uses SAGE’s built-in pointer object to present cursors and controls the cursors via SAGE messages.

To successfully process multiple input events from the same Input Client, Input Manager assigns a First-In-First-Out (FIFO) input event queue for each Input Client. When an input event arrives at the Input Manager, it is first queued for processing. Then, after retrieved from the input event queue, it will be handled in the following steps:

First, if movement values are contained in the event, they are scaled according to the size of the tiled-display to update the current cursor position of the corresponding Input Client. In order to use mouse or other pointing devices in a large tiled-display environment while keeping a good user-experience, the mapping of movement to pixels should be recalculated so that a user will feel that the speed of the cursor on the tiled-display is the same as that on a desktop display. Moreover, the scale factor can be adjusted when a user moves the cursor among different application windows, which makes it possible for the user to select any detailed visual component on the tiled-display.

Second, after the cursor position is updated, if the cursor is located in the range of an application window, the corresponding Input Client can be captured by this application if the window is the top-most one. However, if an interaction strategy other than the Free strategy is employed for this application, the Input Client is captured only if the conditions described in Section 6 are satisfied. The Input Client is released from the captured application when its cursor moves out of its window.

Finally, if the event belongs to a captured Input Client, this event is forwarded to the application whose window is associated with the Input Client. If the forwarded event is generated by a pointing device, the event will contain normalized coordinates calculated according to the window size. Therefore, the size of the window on the tiled-display will not affect the calculation of cursor position at the application side.

Besides the above processing steps, when more than one Input Clients are captured by the same application, the timestamps of the input events from these Input Clients will be updated to reflect their arrival order before they are forwarded. By doing this, all the input events will be serialized so that the cursor position on the application side will always reflect the effect of the last operation.

To match the process capability of target applications, Input Manager may have to drop some input events if they exceed the rate limit.

B. Input Client

Input Client is running at the computer with input devices. It receives events generated by input devices, sends them to the Input Manager, and receives feedbacks from Input Manager. To support as many types of devices as possible while limiting the bandwidth consumption, Input Client should use an appropriate mapping table for device events. Currently, we choose the Microsoft DirectInput’s mapping table, which contains pre-defined values for keyboard and mouse events and can represent an input event in one byte for most input devices. When an input event is generated from the device, Input Client attaches a timestamp on the event, so that the event can be handled more precisely by the target application. For pointing devices, Input Client uses relative values to represent movements, that is, every such event contains the differences between the current position and last position of the device. Using relative values can reduce the process steps because these values can be applied directly without being normalized at Input Manager.

C. Application Layer

Application Layer is an I/O substrate designed as a loadable module for target applications. Currently we have implemented it for Microsoft Windows. When Application Layer is loaded into a target application, it takes control of the inputs and outputs of the application, so that the rendering view of the application will be streamed to tiled-display before being displayed locally and the input events from Input Manager will be delivered to the application as if they were generated locally. In this way, Application Layer can work with any non-customizable
target applications without any modification to them towards the tiled-display environment.

When an input event is received from Input Manager, all normalized coordinates embedded in the event are updated, so that the local cursor and the cursor on the tiled-display will point at the same position of the application window. Since all the input events are delivered to the application according to their timestamps, time-sensitive operations can be executed correctly.

VI. INTERACTION STRATEGY

To correctly support multi-user interaction in the tiled-display environment, especially when multiple users are operating in a same application view, some interactive strategies should be introduced. In this paper, we introduce the following three basic strategies: Free strategy, FIFO strategy and Priority strategy.

Free strategy is the default strategy used by Input Manager. It allows multiple Input Clients to be captured into one application with no additional control, which means all input events generated by these Input Clients will be forwarded to the target application.

Oppositely, the other two strategies can guarantee exclusive access to one application for a single user. Input Manager will create a per-application waiting queue for each application and Input Clients that can be captured into one application are queued. Subsequently, Input Manager makes the Input Client that is at the head of the queue be captured by the application. The only difference between the FIFO strategy and the Priority strategy is that Input Manager uses a FIFO queue for the FIFO strategy and a priority queue for the Priority strategy respectively. In our first implementation, each Input Client will have a pre-assigned integer priority value that ranges from 1 to 100, with higher value representing higher priority. For example, in a collaborative scenario with one coordinator, five presenters and twenty audiences, the coordinator will be assigned the priority value of 100, and each presenter will have a priority value of 50, while the priority value of each audience will be 10. Such an assignment makes the coordinator have the highest control privilege, and a presenter can determine whether the audiences can interact with his application when he is presenting.

These strategies are applied in the application’s scope so that Input Manager can support different strategies in one tiled-display environment.

VII. EXPERIMENTS AND ANALYSES

A. Experiment Environment

The experiment environment demonstrates our ultra-high-resolution tiled-display environment, as shown in Fig. 2. It has a tiled-display wall of 7 columns by 4 rows, consisting of twenty-eight 46-inch LCD screens, with the total resolution of 9520x3072 pixels, nearly 30 Megapixels. The system is driven by a SAGE display cluster of seven display nodes plus one control node, with each column driven by one display node. Each display node has one Quad-core Intel Xeon E5506 CPU, 4GB main memory, two NVidia Quadro FX 3800 GPUs with 1GB video memory. All the nodes are directly connected by a Gigabit Ethernet.

B. Performance

In order to evaluate the performance of the system, we choose the input latency as the performance metric, which is defined as the time interval between the event generation time and event reception time. Due to network transmission and process procedures, it may take more time for an input event to take effect through Magic Input than in a normal desktop environment. Therefore it is important to keep the input latency as low as possible for a good user-experience.

We choose Microsoft Word as the target application and deploy Input Client at the same computer where the target application runs to eliminate the influence of clock offset between different machines. Then we modify the Input Client to read inputs from a file instead of receiving inputs from devices directly. Each input event in the file consists of a generation time that guides Input Client when to send the input event, a type value that represents the device that generates the event and a key code. Input Manager is running at a different computer.

We first evaluate the input latency of single Input Client. Five test cases with each containing 500 different operations are used and each test case runs five times for calculating the average result. The final results are shown in Fig. 3. This experiment shows that despite the variation of maximal latency from 5 ms to 15 ms, the average latency stays steadily at about 0.8 ms.

Then we evaluate the input latency with more than one Input Clients. As there is only one target application, we
choose the Free strategy for it to support multiple Input Clients simultaneously. Another five test cases with each containing 500 operations per Input Client are used and each test case also runs five times to calculate the average result. Again, this experiment shows that in spite of various amounts of Input Clients, the maximal latency keeps stable at about 13 ms to 15 ms and the average latency is still about 0.8 ms. All these results are much better than the input latency given in [14], which needs a maximal time interval of 32 ms to reflect the user inputs.

C. Bandwidth Overhead

In Magic Input, the main overhead is caused by the transmission of input events and SAGE messages. In the following experiment, we choose Google Earth as the target application and design five test cases with each containing 500 navigation operations in different screen resolutions. All network traffics generated by Input Manager are recorded to measure the average bandwidth consumption, which is shown in Table I. It indicates that the bandwidth generated by Magic Input is independent of screen resolutions and the average bandwidth consumption is no more than 13 Kbps.

VIII. Usage Scenarios

We are now running Magic Input for daily use, including our weekly meetings, discussion seminars and technical demonstrations. Fig. 4 (a) shows the usage of Magic Input in our weekly meeting, and Fig. 4 (b) shows the usage in a collaborative demonstration. In both figures, red circles indicate the pointers on the tiled-display and green boxes indicate the input devices controlled by users, with one input device controlling one pointer.

<table>
<thead>
<tr>
<th>Screen resolution</th>
<th>Average bandwidth consumption (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1440x900</td>
<td>12.98</td>
</tr>
<tr>
<td>1360x768</td>
<td>8.89</td>
</tr>
<tr>
<td>1280x1024</td>
<td>12.99</td>
</tr>
<tr>
<td>1280x800</td>
<td>12.47</td>
</tr>
<tr>
<td>1024x768</td>
<td>11.22</td>
</tr>
</tbody>
</table>

IX. Conclusion and Future Work

We have presented a novel multi-user interaction system in this paper, which focuses on porting non-customizable applications to tiled-display environments for collaborative interaction. We have deployed a working large tiled-display collaborative workspace and tested the system with a couple of applications. By enabling users to use input devices freely in front of the tiled-display and directly interact with the applications shown on the tiled-display without watching their desktop screens, we successfully facilitate them to interact with applications smoothly just like on their desktops, thus presenting them more collaborative and immersive experiences. Additionally, our experiment results show that our system has low interaction latency and low bandwidth overhead. We are now working on using Kinect to provide gesture input, and we believe this will greatly improve user experience in a tiled-display environment.

ACKNOWLEDGMENT

This work was supported by National High Technology Research and Development Program of China (863 Program) No.2009AA043303, National Important Research Plan of Infrastructure Software Grant No.2010ZX01042-002-001-00 and the State Key Laboratory of Software Development Environment Funding No.SKLSDE-2009ZX-12.

References


