An Interactive Dynamic Tiled Display System

Juliano Franz, Gelson Reinaldo, Anderson Maciel and Luciana Nedel
Instituto de Informática
Universidade Federal do Rio Grande do Sul
Porto Alegre, Brazil
{jmfranz@inf.ufrgs.br, gcreinal@ucs.br, amaciel@inf.ufrgs.br, nedel@inf.ufrgs.br}

Figure 1. Example of a composition of tablet PCs dynamically distributed to better visualize a route on Google Maps application.

Abstract—Data acquisition devices and algorithms are generating each day larger datasets. As displays are not evolving in the same velocity, the use of tiled displays systems is being seriously considered for the visualization of huge datasets. However, tiled-displays are expensive and large, requiring dedicated rooms for it. Therefore we propose a low cost and scalable tiled display using an array of movable tablet PCs. We also present a strategy to interact with applications running on this dynamic tiled display system, which can be operated by one or multiple users concurrently. Our solution is based on two principles: even if each tile is a separate computer, users should feel it as an unique application running on a single machine; interaction is provided by sketching gestures directly over the displays surfaces using the tablet stylus. Users may use the system in a natural way, as they are just taking notes in their own scrapbook. Preliminary results are presented and discussed.

Keywords-Tiled displays; gestures interaction; touch screen

I. INTRODUCTION

Nowadays it is common to have datasets and images which are too large to fit in a single display without compression or simplifications. While computer power has exponentially grown, display technology, on the other hand, has only doubled its resolution in the past decade [1]. Therefore there is a gap between data acquisition and sampling, and data display. A simple example of this is the regular use of Google Maps application. Normally, one who uses it to find an address or directions needs to zoom in and out many times, toggling between focus and context information.

Techniques for construction and configuration of tiled display systems have been focused by a number of research groups. Arrays of monitors or projectors in a fixed size matrix NxM can be managed by computer clusters to display a single image with large dimensions and high resolution, solving in a certain way the gap between datasets and displays size.

However, tiled displays systems are mounted on large fixed structures becoming a kind of display wall. Because of the great amount of displays and computers involved, it also requires a special electrical and refrigeration infrastructure.

The strategy used for interacting with on screen information is another concern on tiled display systems. When multiplying the resolution of a single regular screen, we cannot assume that mouse and keyboard continue being as efficient as they are with desktop setups. Firstly, because the area to be covered by the mouse pointer is much larger than with a single display. Secondly, because of the size of the display. In front of huge images, users prefer to stand up and move to interact with display walls.
In this paper we present a dynamic and portable tiled display assembled using tablet PCs with multi-user interaction in a turn-based scheme. One advantage of the use of tablet PCs is that we can rearrange the shape of the tiles (i.e. tablet PCs) to display the data in a more suitable way depending on the data we are interested on (see Figure 1).

The focus of this paper, however, is on the strategy used to interact with an application that runs in such system.

The remaining of this paper is organized as follows. In the next section we discuss some related works around the display wall problem. Later, in Section III, we overview our dynamic tiled display, and in Section IV the strategy used to interact with it is detailed. Finally, in Section V we present the preliminary results achieved and in Section VI our conclusions and future work.

II. RELATED WORKS

Izadi et al. [2] and Ni et al. [3] have discussed a number of solutions for tiled display systems. All of them are essentially hardware-based solutions around the large display paradigm. These hardware solutions were used as the start point to build software for interaction control, and data sharing as described below.

Electronic whiteboards [2] are a solution for one projector based systems and represent the starting point for the tiled display paradigm. Just like a standard whiteboard, the interaction device is a stylus. The main problem on these systems is that it is not scalable and only one person can interact at time.

To solve the scalability problem, it is common to use a set of LCD panels [3] arranged on a fixed structure in a special room. However, on most situations they have bezels that create a gap on the screen, and typically they cannot be rearranged to fit the data.

This gap problem is over passed in the projector array system [3]. This approach is very similar to the LCD displays. However they do not have the gap problems because there are no bezels between the tiles. The main issue, in this case, is the difference between the images generated by each projector. It can take up to days to calibrate all of them because of colors, brightness and contrast are never exactly the same even with projectors from the same manufacturer. Also, it is not easy to get them all perfectly aligned.

Concerning software, the data visualization should be supported by Chromium [4], which is an abstraction layer that replaces the OpenGL driver. With chromium it is possible to distribute any OpenGL application over the network to the display system. Chromium server intercepts all the API calls from the application and sends it to the other machines. But the server does not optimize the end-user application, and does not try to minimize the number of OpenGL calls. Because of that, it requires a fast Ethernet network.

The interaction on these types of tiled displays can be done using Synergy [5], that allows the use of a single mouse and keyboard over several machines. Simple to configure, it does not allow changing the displays relative position in run time. Also, it only allows the use of a single mouse and keyboard.

Xdmx [6] is a X11 extension that is used too extend the desktop to one or more computers over the network. The position of each computer is set before the initialization of the system. Therefore it is not possible to rearrange them on run time. Xdmx is a smart choice for 2D applications on fixed aspect tiled displays but it cannot handle 3D graphics.

III. DYNAMIC TILED DISPLAY

We propose a dynamic tiled display with multi-user interaction using low profile tablet PCs connected through a wireless network. The dynamic tiled display was built in the following way. A planar and transparent surface with fixed dimensions and position – as a glass table for instance – represents the virtual large screen, where the final image should be virtually drawn. Tablet PCs are then disposed on this surface. Each tablet has an LCD, which is used as a tile of the larger display to show the view of the corresponding synthetic camera updated based on the tablet position.

In such a way, moving a tablet causes the corresponding image on the LCD to be redrawn as if we were moving a window over the virtual large picture. Thus, each tile shows different parts of the overall scenario depending on their positions and orientations. Since tablets have LCD screens surrounded by frames, the tiled display formed by the LCDs will show the image with some parts missing, the ones occluded by the LCD frames (as can be seen in Figure 1). However, there is no distortion, just a natural discontinuity on the image.

An arrangement with a 2x2 tile array, for example, increases twice the dimension of the image that can be shown in relation to a single display; the resolution (pixels per inch) remains the same but there are four times more visible pixels. Figure 6 shows such an arrangement with five tablets with 1,280x800 pixels each.

The strategy used to track the tablet PCs on the surface (e.g. a glass table) is based on the use of marks placed under each tablet and that can be captured by a single camera put under the support surface (see Figure 2), pointing to it, and hosted on a server. When a user moves a tablet over the table, he/she changes its position on the virtual desktop. Since the camera detects the movement of a tablet, the server broadcasts a message over the network to change the tablet position on the virtual desktop.

IV. INTERACTION STRATEGY

A. Overview

Multiple users can interact with the system in turns using the tablet PCs stylus. When a user interacts with one tablet, the interaction event generated is sent to the server. The server then broadcasts the event to all the other tablets.
As mentioned before, our solution considers that the final application is replicated and runs independently on every tablet PC that composes the dynamic tiled display. However, for the final user, there is only one application that generates one big image – in fact exhibited on the set of tablet PC – and all interaction is done on this big image.

In order to allow the implementation of this idea, we need a virtual desktop big enough to fit the working surface (as shown in Figure 2) that is partially covered by the tablets. We are using VTWM (Virtual Tab Window Manager) [7], a window manager for the X11 system. Also, all data and interactions are replicated on every tablet using an event manager developed by us for this project. Messages are exchanged between tablets using a wireless network, and are managed by a server side application.

B. Apparatus

For this study we used HP/Compaq 2710p tablet PCs. The choice of that model was based on the long battery life, approximately 8 hours, and because they were provided by Hewlett-Packard to be used for studies in new interaction techniques. Because the machines have only 802.11g network adapters, we cannot rely on network speed.

Also, we developed an event manager for the tablet stylus, since VTWM did not have this feature. A server side application was also developed to manage the camera tracking information and to handle the events broadcast between the tablets. The interaction cycle, depicted in the Figure 3, follows the steps:

1) The user interacts with the image on a tablet using the stylus
2) As event is generated and broadcasted
3) The server side application catches this event and resends it to all other tablets
4) An interaction thread (running on each tablet) captures the event and calculates the resulting action on the portion of the image that is being exhibited on the tablet screen
5) VTWM refreshes the screen

C. The window management system

VTWM was first developed on the top of the TWM, a minimalist X Window manager. VTWM differs from TWM because it creates a virtual desktop where the user can arrange multiple applications. The virtual desktop size can be defined to any size and is independent on the resolution of the display. One advantage of the VTWM is that the desktop size does not affect RAM usage. In others words, bigger desktop size does not require more amount of the primary memory.

Stock VTWM does not support broadcasting mouse events and changing the position on the virtual desktop over the network. To solve this problem we have developed two threads to run with the window manager. One is responsible for tracking the mouse events and broadcasting them when the state has changed, as explained on the last section. The other one tracks the tablet position on the virtual desktop and manage the user interactions with the system. This last thread receives the message from the server and replicate the exact action that the user has done in the other tablets.

D. Event manager

The key for our interaction strategy is data and event replication. Therefore, it is very important to have all the tablets synchronized.

To recreate the interactions on the other tablets we created an event watcher and dispatcher. This manager runs as two threads together with the VTWM window manager. They are responsible for capturing the interaction events, packing, and...
sending them to the server application. Also, the manager
controls the user's turn. Multiple users can interact with
multiple tablets but only one at a time. So, if one user tries to
interact with one tablet while there is another user doing the
same thing, the manager blocks the first user interaction.

E. Communication

All the messages are sent to a dedicated server. A server-
side application is responsible for receiving those messages
and resend them to all of the other tablets connected with
it (see Figure 4). This application tags the begin and the
end of each interaction so the manager can control the
turns. There is no guarantee that we will be able to use
all the network bandwidth because it depends on the signal
strength. Also, we must ensure that all packages sent are also
received. We chose the Transmission Control Protocol (TCP)
as communication protocol since it provides reliable and
ordered delivery of packages. TCP is only reliable because
it has a huge header that controls package integrity and
delivery order. As TCP packages are relatively large, we
chose to send minimal data containing only the information
about mouse state, mouse position, and the tablet position
on the virtual desktop in order to avoid network overload.

F. The viewport idea

Just like a regular OpenGL application our solution works
with the concept of many viewports. Each tablet PC con-
ected to the system can see only a subset of the virtual
desktop. The size of the visible area is defined by the screen
resolution of the screen. Users can freely change the position
of the window in the world in order to reach different parts
of the image and interact with other applications.

The user can also use a mini-map on the corner of the
screen (Figure 5) to help them to know their location in the
virtual world.

V. Preliminary results

At the time this article was written we had a stable release
of the system. It allows the use of several tablet PCs or
other mobile computers to interact within a large and high-
resolution virtual desktop. It is also possible to physically
move these computers and track them to change the position
of the viewport on the desktop.

We have done preliminary experiments with users using
Google Maps (Figure 6) and GIMP (Figure 7) to enhance
the look and feel for further tests and check if the network
could handle the amount of data needed for the system run.
More complete and careful tests still have to be planned and
performed, but it is safe to say that the system is stable and
the principles beyond it are solid.

One issue with our solution was found with freehand
drawing on the GIMP application. The event sample rate that we use is considerably lower than the hand speed when drawing on the screen. Unfortunately, it is not possible to raise the sample rate because, given the current parameters, this would overload the wireless network. However, we hope to solve it with new generation hardware.

VI. CONCLUSION AND FUTURE WORK

With this study we intended to prove the concept of a low cost and low profile dynamic tiled display. Besides the free movement feature, this project has not required any gamma and geometry calibrations, which projector-based systems usually need. Also, at the current stage of development, the system is easily scalable because the initial setup is independent of the number of computers.

In further studies we intend to test the idea with more computers connected to the system. Also, we plan to merge this method with a static display arrangement to build a development framework that can run on any type of tiled display system using low network bandwidth.

ACKNOWLEDGMENT

The authors would like to thank Hewlett-Packard for the TabletPCs used on this project, as well as Microsoft, CNPq PIBIC program, and projects 309092/2008-6, 481762/2008-6, and 302679/2009-0 for the support to the researchers Juliano Franz, Anderson Maciel, and Luciana Nedel.

REFERENCES


Figure 7. Eiffel Tower photo being visualized in GIMP