Easy Screens and Play: A Library for Information Visualization in Tiled Display Environments

Younghun Jung*, Geongi Gim†, Myeongjae Kim‡, Yejin Kim§, Kwangyun Wohn¶

Graduate School of Culture Technology, Korea Advanced Institute of Science and Technology

Abstract

In this work, we present a library, called *Easy Screens and Play (ESP)*, for information visualization in tiled display systems. We describe the design of the ESP library in the tiled display environment for novice users and its unique features as compared to other distributed display libraries. We also demonstrate the efficacy and effectiveness of the ESP library by using several examples of information visualization. Discussions on the extension of the library and future improvements are presented as well.

Keywords: Information visualization, client-server architecture, programming interface, tiled display system.

1 Introduction

A tiled display system is a platform to visualize information. In particular, when a large amount of information needs to be visualized at once, such a system offers more pixels for data than any single display [1]. It builds a large high-resolution display by combining multiple displays and computers. However, thus far, such a system has been used only in a few professional fields because of its price and complexity. Several libraries and software have been developed for tiled displays, but they are still difficult to use for information visualization by novice developers with little experience.

In this work, we have developed a library facilitating easy programming for non-experts to create information visualization on a tiled display system. Using this library, users do not need to consider the architecture of a distributed display system; they can create a visualization application on the server by assuming that they are dealing with one gigantic display canvas. The library also offers additional functions to create several common information visualizations so that users can visualize data effectively with minimal programming.

2 Previous Work

Several approaches to develop software for a tiled display system have been studied in the past. The *Scalable Adaptive Graphics Environment (SAGE)* [3] is a passive client system in that the server calculates each and every piece of pixel information and sends it to the appropriate client. *Cross Platform Cluster Graphics Library (CGLX)* [4] is an active client system in that the server sends only the visualization commands, and each client renders and visualizes the data for itself. *Massive Pixel Environment (MPE)* [5] is a well-known active client system and is similar to our ESP library in that it uses processing as the basic programming platform [2].

* email: youngj6169@kaist.ac.kr
† email: raynuzeek@kaist.ac.kr
‡ email: donijoya@kaist.ac.kr
§ email: cheshire@kaist.ac.kr
¶ email: wohn@kaist.ac.kr

3 Design

3.1 Design Approach

The goal of this research is to facilitate the use of tiled displays for information visualization. Hence, the library should be easy to use. A passive client system can easily create visualizations but is very expensive. In contrast, an active client system may not be easy to use but is relatively cheap to build. Hence, ESP is designed to facilitate the creation of visualization by programming only at the server of an active client system. Processing is selected as the programming language. It is easy and is widely used by the information visualization community. The ESP library provides basic visualization functions imitating those of Processing. Therefore, Processing users can easily create various types of visualizations on tiled displays without any additional knowledge. The library should also be able to run on tiled displays of various sizes. This is made possible by putting several size variables in the template and the setup file of each client when installing ESP.

3.2 Visualization Function

The names of visualization functions for tiled displays are very similar to those of the original functions of Processing. What users need to do is just replace the first letter of an original function with a capital letter and then add "**mServer.m**" before it. For example, when drawing a rectangle, the **rect( )** function is used on a Processing sketch window. For the corresponding task on tiled displays, the function should be **mServer.mRect( )**. Processing users do not need to study how to use this library.

3.3 Operation Process

The server runs Processing, the ESP library, and the template for programming. Clients have a setup file containing the client information and the program for visualization. The server sends the **frameStart** and **frameEnd** commands at the start and the end of every frame, respectively. When executing an imitated visualization function, its name and arguments are sent to clients. Clients store them in each queue until **frameEnd** is received. Once this has been done, the clients execute the original visualization functions of the current frame. Clients can visualize their area by translating to the origin of the integrated screen prior to the actual drawing because clients have the corresponding location and size information in the setup file.

3.4 Improvements

For improving performance, visualization functions have a provision for detecting the visualizing area so that any command that does not have to draw is not sent to the client. For people unfamiliar with programming, an additional library is developed. It provides functions for several basic visualizations such as bar graph, line graph, pie chart, and scatterplot matrices.

4 Implementation

The developed ESP library has been extensively tested on numerous examples ranging from a simple visualization such as a pie chart to a real-time interactive visualization on tiled displays.

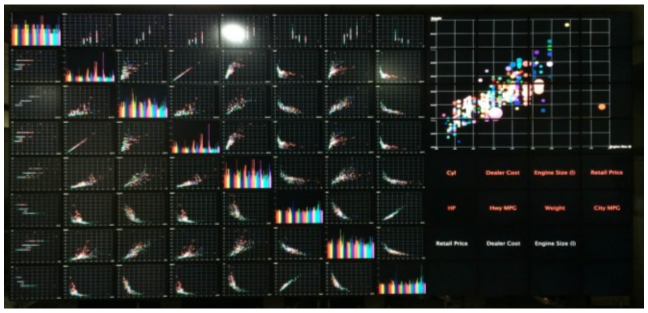

Figure 1: Multivariate visualization on tiled displays

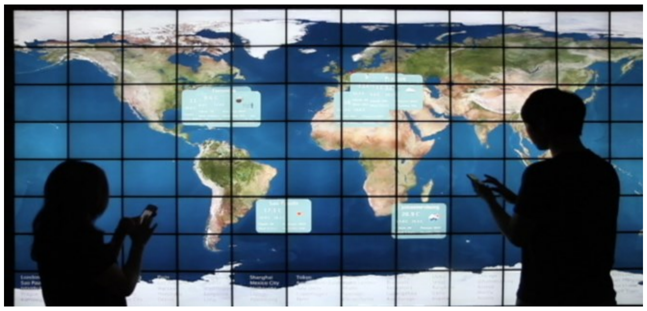

Figure 2: Visualization of city weather data in real time

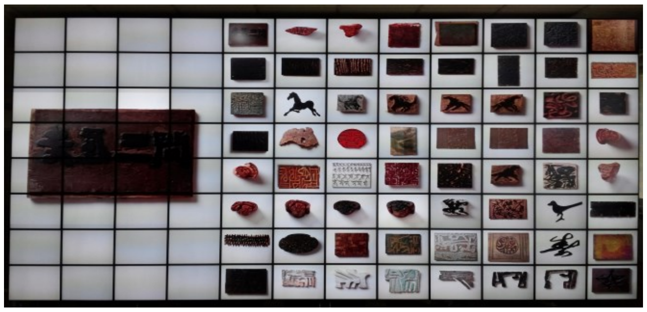

Figure 3: Artwork exhibition on tiled displays

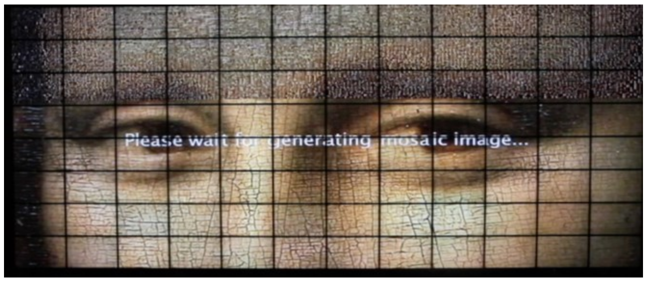

Figure 4: Image visualization using the photo mosaic technique

We use a dataset related to vehicles [6] and perform multivariate visualization by using a scatterplot matrix (see Figure 1). We can also export weather data from a weather API [7]. Users select a specific city on the web by using their mobile devices, and the corresponding weather data are visualized real time at the location of each city on the world map (see Figure 2). In the case of artworks exhibition, a user can select a specific artwork from the several artworks on the tiled display by using a mobile device (Figure 3). In an example of media art, a large image can be visualized by using several images combined with the photo mosaic process (Figure 4).

5 Results and Discussion

We have identified some problems and limitations through the above implementations. The current ESP library supports 2D visualization functions offered by Processing. From the perspective of a client system, the higher the performance and the frame rate of a visualization application, the less accurate is the synchronization. When developers use the ESP library to visualize contents on tiled displays, external libraries for visualization in Processing cannot be used for visualization on tiled displays. Meanwhile, the ESP library has unique features. Compared to MPE, users run the visualization application on the server and they do not need to work each client program. Compared to CGLX implemented by using C++, the ESP library is based on Processing for novice developers and users need to create a visualization application on the server only. In comparison with SAGE, the network load is less and there is no need to set up expensive high-performance facilities.

6 Conclusion

ESP is a visualization library for tiled display systems, targeted for users with little programming skill. As the ESP library is based on the Processing programming language, even a person without any knowledge of the architecture of tiled display systems can create non-trivial information visualization with minimal effort. This library is also easy to use for developers with some experience of the Processing language, as all they need to do is just add simple keywords to the existing visualization functions that Processing offers. Further, the use of the ESP library can be extended. We aim to forge an integrated display environment that includes not only the tiled display system (having a uniform configuration) but also a heterogeneous mixture of the displays of laptops, mobile devices, and other computing devices.

7 Acknowledgments

This research was partially supported by NRF and the BK21 Plus Framework.

References

[1] Johnson, G. P., Abram, G. D., Westing, B., Navr'til, P., & Gaither, K. (2012, September). Displaycluster: An interactive visualization environment for tiled displays. In *Cluster Computing (CLUSTER), 2012 IEEE International Conference on* (pp. 239-247). IEEE.

[2] Processing.org, http://processing.org.

[3] Jeong, B., Renambot, L., Jagodic, R., Singh, R., Aguilera, J., Johnson, A., & Leigh, J. (2006, November). High-performance dynamic graphics streaming for scalable adaptive graphics environment. In *SC 2006 Conference, Proceedings of the ACM/IEEE* (pp. 24-24). IEEE.

[4] Doerr, K. U., & Kuester, F. (2011). CGLX: a scalable, high-performance visualization framework for networked display environments. *Visualization and Computer Graphics, IEEE Transactions on*, *17*(3), 320-332.

[5] Westing, B. M., & Turknett, R. (2012). Extending the Processing Programming Environment to Tiled Displays. *Vis*, 2–3.

[6] Interactive Data Visualization, http://www.idvbook.com

[7] http://openweathermap.org