Joint Virtual Reality Conference of EGVE - ICAT - EuroVR (2009)
M. Hirose, D. Schmalstieg, C. A. Wingrave, and K. Nishimura (Editors)

Short Paper: Desktop Integration in Graphics Environments

Torsten Ullrich[1,2] , Volker Settgast[1,2], Christian Ofenböck[1], Dieter W. Fellner[1,2,3]

[1] Institut für ComputerGraphik und WissensVisualisierung Technische Universität Graz, Austria
[2] Visual Computing Division, Fraunhofer Austria Research, Graz, Austria
[3] Fraunhofer Institute for Computer Research and Technical University of Darmstadt, Germany

Abstract
In this paper, we present the usage of the Remote Desktop Protocol to integrate arbitrary, legacy applications in various environments. This approach accesses a desktop on a real computer or within a virtual machine. The result is not one image of the whole desktop, but a sequence of images of all desktop components (windows, dialogs, etc.). These components are rendered into textures and fed into a rendering framework (OpenSG). There the functional hierarchy is represented by a scene graph. In this way the desktop components can be rearranged freely and painted according to circumstances of the graphical environment supporting a wide range of display settings – from immersive environments via high-resolution tiled displays to mobile devices.

Categories and Subject Descriptors (according to ACM CCS): H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical User Interfaces, Windowing Systems I.3.2 [Computer Graphics]: Graphics Systems—Distributed/Network Graphics

1. Introduction

In the use of personal computers (PC) – especially in the field of human-computer-interaction – a shift of paradigms takes place. The "Windows, Icons, Menus, Pointer" paradigm of user interface design started in the 1970s. Major developments and improvements have been realized (in alphabetic order) by Apple, Digital Research, Microsoft, Motif, NeXT Computer, Sun Microsystems, Xerox Palo Alto Research Center, and many more. This user interaction is based on a single physical input device (a pointer) controlled by a single user. Information and data are presented by icons and windows. Available commands are compiled together in menus, which can be started via the pointing device. This graphical user interface paradigm is easy to use for both novice and power users.

With the usage of tabletop environments, tablet PCs, immersive environments etc. this single-pointer, single-user paradigm started to change. In 2007 Chia Shen et al. [SFWV] authored an open letter to operating system (OS) designers in which they outlined the major requirements of the new user interface paradigm. Concerning these requirements current operating systems have deficiencies which are currently compensated on application level.

In this paper we present a technique to cope with the graphical requirements; moreover it solves the integration problem of legacy applications in immersive environments. We use the remote desktop protocol to access a desktop, disassemble it into its components and rearrange/repaint it according to the setup of the environment, in which the desktop is embedded.

2. Related Work

Previous research has analyzed the influence of display setups to a user's performance in everydays work [BB09]. An overview of large high-resolution screen setups and cluster rendering software has been presented by Ni et al. [NSS*06].

The approach presented in this paper is similar to other projects: In 1993 Dykstra introduced the idea of mapping X11 windows as textures onto polygons in a virtual 3D world [Dyk93]. As X11 – the network-enabled windowing protocol for Unix systems [GKM86] – plays a minor role for many end-users following projects concentrated on Microsoft Windows. For example, Regenbracht et al [RBW01] integrated 2D desktops and 3D data sets into a tangible, augemented reality (AR) desktop environment. Besides computer vision and AR techniques they use a modified Virtual Network Computing (VNC) to display 2d desktops on

DOI: 10.2312/EGVE/JVRC09/109-112

3d geometry. Also Bues et al. [BBH03] and Nakashima et al. [NMKT05] presented a VNC-based solution.

Depending on the setting in which our approach is used, we utilize various input devices. As this paper concentrates on the network and computer graphics part we only summarize the main input methods briefly. We support multi-touch interaction (based on [KBBC05]), 3D-tracked pointing devices, gaming devices and standard input methods (mouse, keyboard). An overview on multi-touch techniques is presented online by B. Buxton [Bux09]. Implementation details of various multi-touch enabling libraries are described by P. Dietz and D. Leigh [DL01], S. Jordà et al. [JKGA06] and B. Ullmer and H. Ishii [UI00].

To access a desktop various network protocols have been designed. On Unix systems the most common solution is the X-server protocol. Enhancements for reduced bandwidth [Pin03], multi-pointer and multi-focuses support [Hut06, HT07] have been made and may be included into future X standards. On non Unix desktop systems the VNC protocol [RSFWH98] and the RDP [Mic09] are predominant and both are available (server and client) for Mac OSX^{TM}and Microsoft Windows^{TM}.

The main difference between RDP and VNC is the abstraction layer. All information within the VNC protocol are bitmap based, whereas RDP spezifies more abstract information such as font and text handling, vector graphics and bitmaps as fallback. Consequently, with RDP it is possible to identify window design elements and to adjust them to the destination environment. Furthermore RDP needs less network bandwidth than VNC. In [NYN03] Nieh et al. benchmarked RDP and VNC: "Overall, VNC [... is ...] faster at higher network bandwidths, whereas [...] RDP performed better at lower network bandwidths. This suggests that the more complex optimizations and higher-level encoding primitives used by [...] RDP are beneficial at lower network bandwidths when reducing the amount of data transferred significantly reduces network latency."

Having accessed a desktop's components we store them into texture images. Then the rendering can easily be done using any graphics framework such as OpenGL, DirectX etc. Our rendering solution uses OpenSG [RVB02] – a portable scene graph system to create real-time graphics programs based on OpenGL. Its extensibility, multi-thread safety and clustering capabilities allow customizing the graphics rendering to various graphics environments (from mobile devices to virtual environments and high-resolution projection walls).

3. Implementation

The Remote Desktop Protocol was introduced with Windows NT^{TM}. Once a remote connection is established the client forwards user input to the server, while the server sends drawing commands of elements of the screen to the client. If not disabled explicitly, the server sends updates of all changed graphical elements. Even changes, which are not visible e.g. due to occulsion, are sent to the client by default. The protocol uses bitmap compression and color palettes to keep the amount of data transmitted small. Caching of bitmaps, fonts (glyphs) and desktop screenshots is another way to reduce network traffic within RDP. The current version of RDP (6.1) supports seamless windows. In order to support earlier RDP versions we use a simple alternative developed by Cendio (http://www.cendio.com/seamlessrdp/). They provide a server side application (shell) that enables a seamless mode. Furthermore there are patches for *rdesktop* (http://www.rdesktop.org/) and *seamlessrdp* available from Fontis (http://www.fontis.com.au/rdesktop) allowing the use of one single connection to launch multiple programs.

Based on these techniques and applications – especially rdesktop – we created an object-oriented framework, which wraps the pure C applications in a modular manner. The network, compression and authentication parts are now integrated into an RDP module. This module uses an abstract window factory and abstract graphics/user interactions to work with. These interfaces can be implemented, for example, by classes which map RDP events to a graphical user-interface (GUI) library. In this case the result is a "normal" RDP client (used for debugging purposes). Though the abstract graphics and user interfaces can also be implemented in a different way. We provide a texture-based approach which is integrated into OpenSG. In this manner, we can integrate a Windows^{TM}desktop into any environment which is supported by OpenSG.

The RDP screens are drawn into the data fields of OpenSG texture objects which are used in material objects. A shader program applied to the material can create advanced visual effects. The material can be referenced by any geometric object in the scene graph – even more than once. For the geometric representation of a window we normally use a quadrangular plane shape. The geometry is added to the scene graph as child of a transformation node.

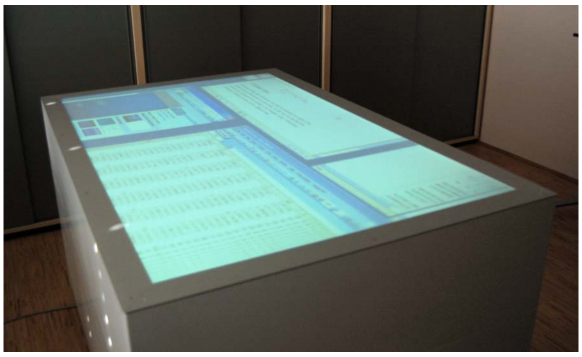

Figure 1: *At a tabletop display there is no fixed definition of top and bottom as people can move around the table and use it from any side. Furthermore all users can interact simultaneously.*

The rendering is decoupled from the RDP connection to the hosting system. Therefore, the texture content may be updated independently from rendering frame rates. The overall performance is sufficient for web browsers, word processor programs, spread sheet applications, presentation software, etc.. Fast frame updates for large display areas, for example a video player running in full screen mode, do not perform very good. Also areas of the desktop which are drawn in overlay/direct mode are not accessable for RDP applications.

User inputs have to be sent back to the RDP server. They can be divided into two classes: on the one hand there are user inputs concerning the arrangements of window elements (normally handled by a window manager); on the other hand there are inputs, which are passed directly to the corresponding application to which the window/dialog/etc. belongs. Both input categories now have to be handled in 3D. Instead of 2D screen coordinates an intersection point P_i has to be found, e.g. by shooting a ray along the input device into 3D object space. The intersection point belongs to some geometry in the scene graph and its texture coordinate system maps the intersection point P_i to screen coordinates. OpenSG has built-in methods to test the geometry in a scene graph for ray intersection. This method returns the hit object and triangle as well as the intersection point P_i.

The user input actions can be modified before they are sent to the RDP server. A double click event for example can be simulated with other kinds of actions depending on the input device. On the multi-touch table (see Figure 1) we are currently sending mouse clicks for each touch event that happens inside the remote window area. Obviously it is not possible to distinguish between a left click and a right click in this way. There are many ways of how to handle this restriction. One idea is to use gesture recognition. Another way would be to define different clicks depending on the time span between press and release events.

Having all the single desktop components the render system can modify the visual appearance of the desktop. In computer graphics the texture rendering is altered by shaders. It is possible to add special effects not only as an eye catcher, but also for practical reasons. We experimented with a shader for distorting a high resolution screen to fit in an area of lower resolution without clipping.

4. Applications

In this section we demonstrate a few examples in which our approach is used.

Desktop in XXL: In a tiled display setting with 4096×2048 pixels the desktop to render runs in a virtual machine. A virtual machine allows configuring the graphics settings independently from any hardware restrictions. Therefore it is possible to set the virtual desktop resolution to values up to 4096^2 (software limit of Windows XP$^{\text{TM}}$). Figure 2 shows such a high-resolution display with one single desktop rendered by three PCs and displayed by six projectors.

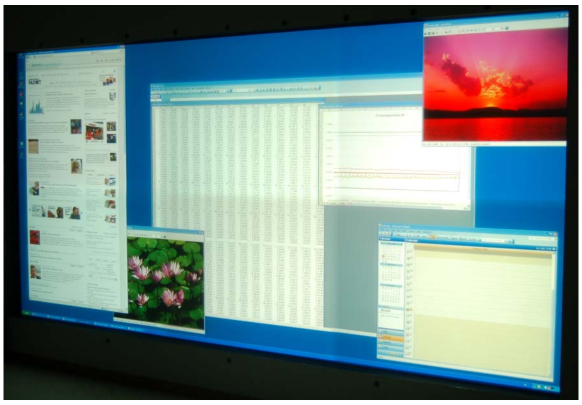

Figure 2: *Our tiled display with 3×2 projectors has about 4000×2000 pixels. The rendering is done on 3 PCs which are controlled by a master. Each render slave is connected to two projectors.*

Presentations on Arbitrary Displays: The previous example tries to make the most of the display technology. Another relevant example is a low-resolution one. Using the RDP technology it is possible to access any Windows PC and Mac (RDP server is not installed by default). All applications, which do not use direct overlay techniques – such as video players – or direct graphics hardware access, can be shown on the presentation wall. In this way it is possible to show, e.g., PowerPoint$^{\text{TM}}$presentations, which are running on a simple laptop.

Desktop in 3D: Our approach accesses the single components of a desktop separately. A functional hierarchy (parent window $\longrightarrow$ child dialog) is mapped to a scene graph hierarchy in OpenSG. As the default object space of OpenSG is three-dimensional the 2D desktop can be embedded into 3D. Figure 3 shows such a setting. The desktop is integrated into a CAVE-like virtual environment. The 2D structure is dissolved and merged with virtual reality. A tracked pointing device with buttons allows clicking into the 2D desktop. Windows can be arranged freely in the 3D space by clicking on the edges of the container geometry.

Desktop Virtualization: A virtual environment does not need to be three dimensional. Using a full-featured rendering framework – such as OpenSG – our approach can visualize a virtual desktop of higher resolution than the display resolution. This concept of virtual screens can be used, e.g., on a multi-touch table. There we render a virtual desktop, which has three times the resolution the multi-touch system has. Each screen is a little bit smaller than the system resolution so that if one screen is displayed 1 : 1 some space is left. The free space is used to display down-scaled version of the remaining screen of the desktop (to the left or to the right as appropriate).

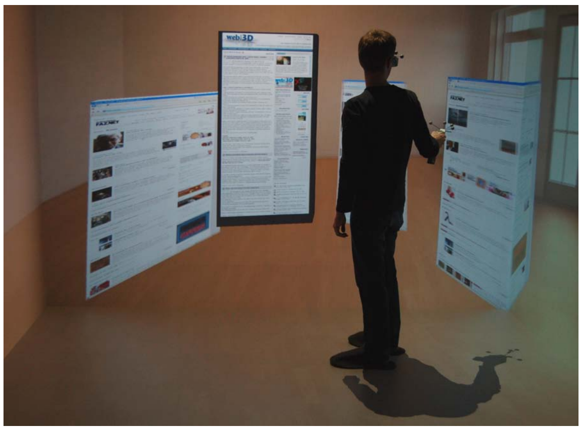

Figure 3: *2D windows are embedded into a 3D scene using a CAVE environment. The container geometry can be arranged freely in the 3D space.*

5. Contribution & Benefit

We have presented a system for rendering legacy Windows software directly in various graphics environments like tiled displays, tabletop surfaces and CAVEs. It uses an implementation of the Remote Desktop Protocol to fetch a desktop or single applications. The RDP server can run on a separate PC or as a virtual machine. Applications can be used without modification. We showed examples running on a tiled display with a high resolution, in an immersive environment and on a multi-touch table. With the possibility to freely transform application windows we can solve the orientation problem of tabletop screens. The system can also be used to manipulate the visual appearance of a desktop. User inputs can be mapped from various input devices to the common keyboard and mouse input events that the legacy software can process. The general restrictions of common desktop applications are not solved. To allow multiple users to work together in one application, substantial modifications on a lower level are necessary. The applications and also the operating system would need to allow multiple inputs simultaneously and multiple applications being in focus at the same time.

In the future the system will be integrated into a multi-touch table framework. Its availability and the terms of license will be discussed as soon as the integration is finalized.

References

[BB09] BI X., BALAKRISHNAN R.: Comparing Usage of a Large High-Resolution Display to Single or Dual Desktop Displays for Daily Work. *Conference on Human Factors in Computing Systems 27* (2009), 1005–1014.

[BBH03] BUES M., BLACH R., HASELBERGER F.: Sensing surfaces: bringing the desktop into virtual environments. *Proceedings of the workshop on Virtual Environments (EGVE) 39* (2003), 303–304.

[Bux09] BUXTON B.: Multi-Touch Systems that I Have Known and Loved. online: http://www.billbuxton.com/multitouchOverview.html, 2009.

[DL01] DIETZ P., LEIGH D.: DiamondTouch: a multi-user touch technology. *Proceedings of the 14th annual ACM symposium on User interface software and technology 14* (2001), 219 – 226.

[Dyk93] DYKSTRA P.: X11 in virtual environments. *Proceedings of IEEE 1993 Symposium on Research Frontiers in Virtual Reality 9* (1993), 118–119.

[GKM86] GETTYS J., KARLTON P. L., MCGREGOR S.: The X Window System. *ACM Transactions on Graphics 5* (1986), 79–109.

[HT07] HUTTERER P., THOMAS B. H.: Groupware Support in the Windowing System. *Proceedings of the eight Australasian conference on User interface 64* (2007), 39–46.

[Hut06] HUTTERER P.: The Multi-Pointer X Server. online: http://wearables.unisa.edu.au/mpx, 2006.

[JKGA06] JORDÀ S., KALTENBRUNNER M., GEIGER G., ALONSO M.: The reacTable – A Tangible Tabletop Musical Instrument and Collaborative Workbench. *International Conference on Computer Graphics and Interactive Techniques archive ACM SIGGRAPH 2006 Sketches 25* (2006), 91.

[KBBC05] KALTENBRUNNER M., BOVERMANN T., BENCINA R., CONSTANZA E.: TUIO: A Protocol for Table-Top Tangible User Interfaces. *Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation 6* (2005), 1–5.

[Mic09] MICROSOFT CORPORATION: *Remote Desktop Protocol: Basic Connectivity and Graphics Remote Specification (Version 9.0).* Microsoft Corporation, 2009.

[NMKT05] NAKASHIMA K., MACHIDA T., KIYOKAWA K., TAKEMURA H.: A 2D-3D integrated environment for cooperative work. *Proceedings of the ACM symposium on Virtual Reality Software and Technology 12* (2005), 16–22.

[NSS*06] NI T., SCHMIDT G. S., STAADT O. G., LIVINGSTON M. A., BALL R., MAY R.: A Survey of Large High-Resolution Display Technologies, Techniques, and Applications. *Proceedings of the IEEE Conference on Virtual Reality 14* (2006), 223–236.

[NYN03] NIEH J., YANG J. S., NOVIK N.: Measuring Thin-Client Performance Using Slow-Motion Benchmarking. *ACM Transactions on Computer Systems 21*, 1 (2003), 87 – 115.

[Pin03] PINZARI G. F.: Introduction to NX technology. *NoMachine Technical Report 309* (2003), 1–7.

[RBW01] REGENBRECHT H., BARATOFF G., WAGNER M.: A tangible AR desktop environment. *Computers & Graphics 25*, 5 (2001), 755–763.

[RSFWH98] RICHARDSON T., STAFFORD-FRASER Q., WOOD K. R., HOPPER A.: Virtual network computing. *IEEE Internet Computing 2* (1998), 33–38.

[RVB02] REINERS D., VOSS G., BEHR J.: OpenSG: Basic concepts. *Proceedings of OpenSG Symposium 2002 1* (2002), 1–7.

[SFWV] SHEN C., FORLINES C., WIGDOR D., VERNIER F.: Open Letter to OS Designers from the Tabletop Research Community. online: http://www.diamondspace.merl.com/papers/2007_open_letter_to_OS_developers.pdf.

[UI00] ULLMER B., ISHII H.: Emerging frameworks for tangible user interfaces. *IBM Systems Journal 39* (2000), 915–931.