Hybrid User Interfaces Using Seamless Tiled Displays

Dieter Schmalstieg and Gottfried Eibner
Vienna University of Technology
Austria

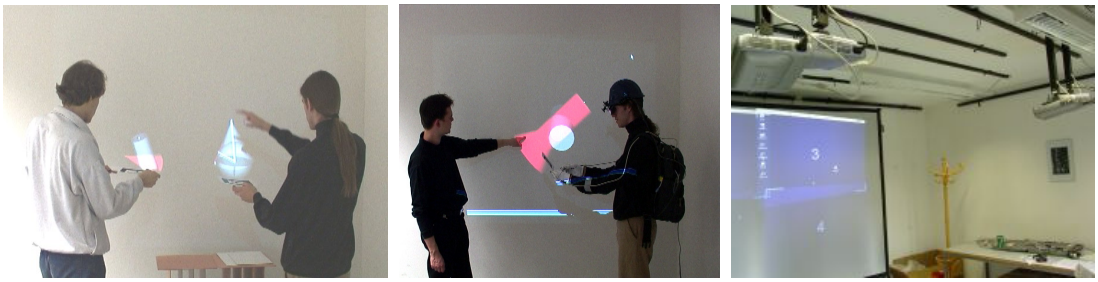

Fig. 1: (left) Hybrid user interface combining tangible objects with monoscopic projector-based graphics for a geometry education scenario; (middle) same application, but the right hand side user wears a stereo head mounted display; (right) a 2x2 seamless tiled display configuration

1 Introduction

In 1998, two influential projects made innovative use of immersive projection technology: The first, the *Office of the Future* developed at UNC [Raskar98] aims at making the use of immersive projection in conventional, potentially small and constrained rooms possible. The authors propose to use image based modeling based on structured light techniques to rapidly reconstruct irregular shaped projection surfaces, then use the obtained models for pre-warping imagery to be projected using multipass rendering. This approach together with software-based blending of multiple overlapping front projections represented a leap forward in building inexpensive seamless tiled displays. The initial results were followed by detailed examination of several aspects of the involved projection technology, most recently the notion of iLamps [Raskar03], self-configuring arrays of projector-camera systems. However, all work in the project has focused on techniques for projection technology, leaving the aspect of a graphics application programmer's interface out of scope.

The second influental project, *EMMIE,* developed at Columbia University [Butz99] also aimed at visual instrumentation of an office or conference environment, but with a focus on ubiquitous computing rather than spatially immersive displays. EMMIE is a distributed system for creating a hybrid user interface, which combines deskop, projection, and head-mounted augmented reality (AR) displays. The hybrid user interface envelopes a group of users with a virtual "ether", in which three-dimensional virtual objects can be placed. By spatially arranging the available displays in the environment, users can visualize the virtual objects. Unfortunately, this interesting approach was not further developed.

In this paper, we will make the attempt to bridge the gap between these two works. The Office of the Future did not explore how to use the ample display space for ubiquitous computing tasks, often involving collaboration of multiple users. It also does not consider the desire for using personal displays such as notebooks, PDAs, or AR displays. EMMIE demonstrates how to perform multi-user multi-display visualization, but relies on conventional projection for creating larger displays.

The work in this paper presents a hybrid user interface system, which can also use seamless tiled displays to provide truly scalable imagery. It builds on *Studierstube*, which independently of EMMIE evolved into a hybrid user interface system [Schmalstieg00]. Recent versions of Studierstube [Schmalstieg02] provided a distributed architecture for managing multiple displays and interacting with multiple users.

We will show that because of the similarity of hybrid user interfaces and seamless tiled displays, only a few key elements have to be added to a hybrid user interface framework to support seamless tiled displays, while many aspects of the system can be re-used. The result is an architecture for hybrid user interfaces that can use seamless tiled displays as an additional powerful display modality. As a consequence, 3D user interface design greatly benefits from the generous display space becoming available without loosing the diversity of the hybrid user interface approach (see Fig. 1).

2 Related Work

We have already discussed the Office of the Future and EMMIE. A common denominator of these projects is Weiser's idea of *ubiquitous computing* [Weiser91] as a future paradigm on interaction with computers. Computers are constantly available in our surrounding by embedding them into everyday items, or by making them wearable. Ubiquitous computing introduces two key ideas: (1) The environment (or the user) is instrumented so that the relevant display and interaction is transported to the task location, rather than being tied to a fixed computer workplace. Displays can include conventional displays, projection displays, and augmented reality. (2) All elements of the instrumented environment are networked so they can collaborate.

Therefore, ubiquitous computing implies the use of hybrid user interfaces, which make comprehensive use of the ubiquitous computing resources by mixing and matching interaction and display capabilities dependent on location and task. Projection-based displays are often a part of this mix, because they are physically encompassing the other components and the user.

Besides EMMIE, there are several other examples of such interfaces. Rekimoto has presented setups for multi-computer direct manipulation, such as in [Rekimoto97], where a stylus is used to drag and drop data across display boundaries. Later, this idea was picked up in Active Surfaces [Rekimoto99], where projections on tables and walls provide a shared space encompassing other devices such as notebook computers.

The Tangible Media Group at MIT has developed a number of heterogeneous user interfaces based on the theme of tangible user interfaces [Ishii97]. For example, the metaDESK [Ullmer97] is a back-projection table that presents information which can be manipulated with tracked props on its surface. It also offers a combination of viewing modalities, e. g., a separate hand-held display used to observe the scene in a different dimension (3D graphics), and with an independent viewpoint. The luminous room [Underkoffler99] is a similar framework where tangible objects are manipulated on a table surface to create complex application setups such as a simulated optical workbench.

In contrast, seamless tiled displays have been mostly examined as dedicated high-end visualization environments, designed in particular for the visual inspection and manipulation of very large data sets (such as medical or geodesic models), which can maximally benefit from the very large resolution. Examples include the Lawrence Livermore National Lab PowerWall [Walls00] or Sandia National Lab. The errection of the Princeton Display Wall [Li00] in a recreational area and its uses for several non-scientific purposes already hint at the versatility of scalable tiled displays for hybrid user interfaces.

3 Approach

3.1 System design rationale

Our hybrid user interface system already supports distributed rendering with soft real-time constraints. Multiple users can experience the sensation of a shared space, which is populated by virtual objects. This is done by distributing and synchronizing instances of the virtual objects to all hosts in the cluster. Each host is then responsible for rendering to its locally attached displays. For example, a tracked HMD allows a user to independently choose a viewpoint. The hybrid user interface allows to mix and match a range of displays from very small (PDA) to very large (projection screen) depending on application requirements, choose from monoscopic and stereoscopic display modes and so forth.

It is important that changes computed at one host are propagated to the other hosts immediately, so that the illusion of a shared virtual space is not disrupted. However, displays which are typically observed individually (i. e., users do not normally look at two such displays at the same time) allows to relax constraints on the degree of synchronization. For example, two computers with unequal graphics power may render the virtual scene with differences in frame rate, without adversely affecting the overall joint experience.

A seamless tiled display driven by a cluster of hosts has a similar architecture in that multiple hosts combinations share a common scene, which is rendered simultaneously. Every host is connected to a projector (or two projectors in case of the now-common dual headed graphics accelerators). We can therefore re-use the general distributed rendering architecture of the hybrid user interface platform if we add the necessary more strict synchronization.

Following the ideas of [Raskar98] that in the future every office should be equipped with inexpensive projection, we considered only commodity hardware. The most demanding level of synchronization, pixel-accurate genlocking was ruled out because by the time of equipment selection, it was limited to very expensive graphics solutions, and is not generally compatible with the inexpensive DLP projectors that we settled for, which introduce their own pixel timing. We therefore considered application level and display level (swapbuffer)

synchronization, which are discussed in sections 3.2 and 3.3, respectively.

Finally, a seamless tiled display in an office environment generally requires front projection from an oblique angle. Consequently, the images from multiple projectors will have non-rectangular shape and will overlap in an irregular way. We choose to restrict projections to planar surfaces, because we found that there are plenty of such surfaces in typical office environments. Such a setup is much easier to calibrate, and allows rendering with image warping in a single rendering pass. Multiple projections can be blended in software using alpha masks without loosing rendering speed. Details on the use of projectors are given in section 3.4.

The approach outlined above allows us to treat multiple individual displays of a hybrid user interface and multiple parts of a seamless tiled display in a uniform way, discriminated only by the configuration addressing the hardware specifics, which are part of the rendering engine and irrelevant to the application programmer. Seamless tiled displays are designed to be easily scalable to the maximum available area in an office environment. Therefore, our architecture is extremely versatile, allowing to arrange visualizations in a true ubiquitous computing style.

3.2 Application synchronization

A popular approach for development of distributed and parallel graphics application is to use a distributed shared scene graph. Distribution is performed implicitly through a mechanism that keeps multiple local replicas of a scene graph synchronized without exposing this process to the application programmer or user. Our own implementation of this concept, Distributed Open Inventor (DIV) [Hesina99] is based on the popular Open Inventor (OIV) toolkit [Strauss92] and propagates scene graph changes using reliable multicast.

Extensions in *Studierstube* are created through OIV subclassing, and can be loaded and registered with the system on the fly. Using this mechanism, we can take the scene graph based approach that avoids a dual database (graphical + application data) to its logical consequence by embedding applications as nodes in the scene graph. Applications in *Studierstube* are written as new application classes that derive from a base application node. Multiple instances of application objects can be present in the scene graph simultaneously for multitasking.

Moreover, a new application node will be added to all replicas of a scene graph, and will therefore be distributed. With the application node all data contained in attributes will be replicated – a scene subgraph of graphical objects, but also attributes that are not visible objects but represent other application data. Non-graphical attributes are simply added as additional "fields" of the application node that do not directly contribute to rendering. We have found this unified treatment of graphical and non-graphical data to drastically simplify application development.

Application specific computations, typically callbacks triggered by events created through user input, need not be repeated at every host. Instead, for every application instance, a master host is determined, which is responsible for performing all execution of application code. The updates to the application state resulting from these computations are then replicated in the slaves' replicas of the application instance. Using this scheme, application specific computation is distributed over the workgroup. At the same time, the master host can be determined for every application instance separately. Coarse grained parallelism is introduced by distributing the master responsibilities over the hosts.

Running multiple hosts that share a scene graph as a display cluster is only slightly different to the above approach. One host is a designated master and collects user data, like tracking or mouse motion events, and performs application-specific computation. The other hosts within the cluster are display hosts, and are running as slaves. They execute rendering code and display the scenery on the tiled display. With this approach, a copy of our Studierstube framework runs on each display host concurrently.

While DIV takes care of application level synchronization, this does not impose any restrictions regarding image update rate, and is therefore not acceptable for a seamless tiled display, which relies on all displays showing portions of the same image at the same time.

3.3 Display synchronization

We therefore extend the distributed shared scene graph approach with display synchronization based on a swap buffer barrier. Rendering halts just before a buffer switch should occur. When all display hosts are ready to switch their frame buffer content, they will be notified to continue code execution.

Since OIV traverses its scene graph to render the scene graph, it is the easiest way to do synchronization within the OIV traversal philosophy. Therefore, to synchronize frame buffer swapping, we create a *synchronization node* and add it as the last node in the traversal order of each display host's scene graph, so that

the host will pause and wait for the synchronization condition right after finishing the rendering.

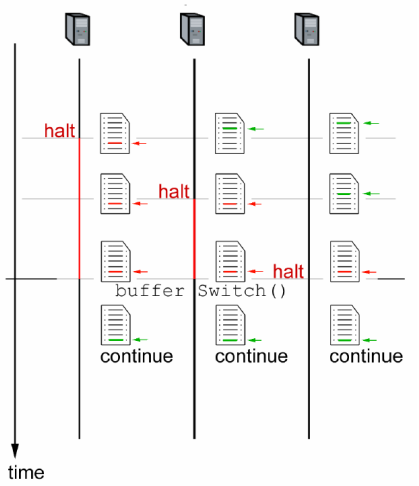

Fig. 2: A time diagram showing application level synchronization of multiple copies of an single-threaded application. After each buffer switch the code's execution path is synchronized again.

When one of these synchronization nodes is traversed during rendering, it notifies the master host that it has finished rendering. After all display hosts have sent their signals to the master, it is the master's turn to notify each node to proceed and swap buffers (Fig. 2).

Unfortunately, it is not sufficient to just perform best-effort data distribution and add a barrier for swap buffer locking, as user interaction, application behavior, update propagation to the display hosts, and rendering all happen in parallel and it is difficult to estimate variations in latency, responsiveness etc. in a soft real-time environment such as ours. Care has to be taken, that each display host's copy of the scene graph is in an equivalent state when it is used for rendering the contribution for a given image.

Consider the update events sent from the master to the display hosts an ordered sequence. Since we can not guarantee that events are delivered to every display hosts at exactly the same time, we have to divide the sequence of events that occur between two successive frames into two parts:

(1) All events that occurred before the buffer switch have to be distributed to all display hosts before the buffer switch takes place.

(2) All events occurred after the buffer switch compared to the master's execution path at that time must be retarded until the next buffer switch occurs.

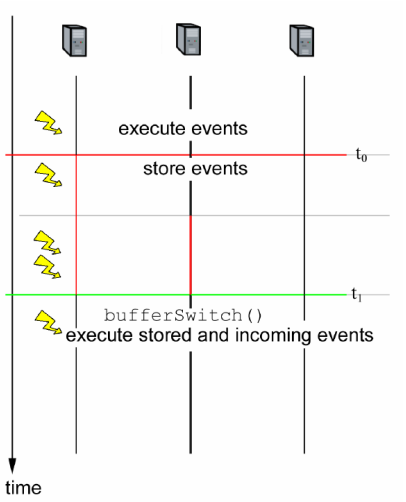

Fig. 3: A time diagram showing event retarding on each display host. The control host sends events via multicast (flashes), so that every event is distributed to all display hosts. After the first display host has finished rendering (at t_0), all subsequent events have to be stored on each display host until a buffer switch occurs (at t_1).

Retarding the events after finishing rendering can be done on the master side or the slave side. We chose to collect the events on the display side because it minimizes latency after rendering commences. Fig. 3 shows the scenario where events are collected on the slave side by the display hosts and are retarded until the next buffer switch occurs.

3.4 Projector calibration

Our calibration software is based on the work of Raskar et al. [Raksar02]. This method calibrates oblique projectors with image detection techniques, and computes homographies between projector space, camera space and display space. With these homographies it is easy to generate a projection matrix for each display that incorporates a pre-warping to compensate for the oblique projection.

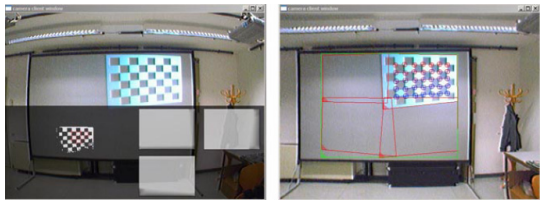

Fig. 4: (left) Detecting feature points of a projected chessboard pattern during calibration. (right) All feature points are detected and the convex hull of each calibrated projector is shown in red.

This calibration technique uses a video camera to observe checkerboard patterns presented by each projector to extract feature points and finds the corresponding collineation between camera and

projector pixels (Fig. 4). With this collineation it is easy to compute projector-to-projector and display-to-projector homographies. The projector-to-projector homographies are used to compute the needed blending masks for intensity weighting in overlapping regions.

During rendering the projector-to-display homographies are used for pre-warping in order to compensate for the oblique projection. Blending mask using alpha blending techniques are used as a post-rendering step to blend multiple pixels in overlapping regions. The scenery will be corrected, rendered, and blended in a single pass, making the overall approach very efficient.

4 Implementation

4.1 Runtime framework extensions

The following extensions had to be made to the OIV/Studierstube framework to support the new display cluster style of operation:

Synchronization node. As mentioned previously, the barrier for swap buffer synchronization was implemented as a new OIV node, which is placed last in the scene graph traversal. Consequently, the next rendering (but not the processing of application updates) is paused until all display hosts are ready to continue. The synchronization node spawns a concurrent thread to monitor the communication even if it is not given control by the rendering traversal.

The synchronization node is implemented as an abstract base class, which processes a synchronization protocol, but leaves the use of a concrete communication channel as a virtual function. For our system, we have implemented UDP based transmission over standard Ethernet. Each host is equipped with two 100-T Ethernet cards connected to separate dedicated switches, so that one network can be dedicated to synchronization, while the other network is used for application level synchronization. All network communication is done using the ACE library [Schmidt00], including multicast communication from the master to the slaves.

In our scene-graph based retained mode data distribution scheme, we found latency to be more limiting than throughput, and therefore did not consider upgrading to the now widely available Gigabit network interface cards. Yet our implementation is generic in terms of the used communication technology, so alternative networking technologies or classic serial/parallel lines such as used in other parallel rendering environments like SoftGenLock [Allard03] or Lightning-2 [Stoll01] is a straight forward extension.

Event retarding. As mentioned above, the DIV protocol for application level synchronization had to be extended to support event retarding. Event updates are timestamped and buffered at the receiver, it is therefore straight forward to assign frame numbers to events and require that the rendering hosts always render the scene graph in a specific state after applying the event updates up to a certain point in the sequence. However, we have not yet addressed the issue of concurrently updating and rendering the scene graph with a finer-grained multithreading structure.

Alpha blending node: A node for attenuating the brightness of the final image using an alpha mask was added to OIV's collection of nodes. It is used to seamless blend the output of multiple projectors. The alpha mask is prepared by the calibration software described below.

Camera node with collineation: Studierstube has a quite sophisticated camera mode for off-axis projection, which allows to independently assign tracking to the viewpoint and image plane. While the existing camera code handled all cases of observers looking at the image plane(s) from an oblique angle, we had to extend the camera model with another matrix attribute for collineation, which handles oblique projection. All these attributes can be configured independently, so one can easily combine different modes such as straight/oblique projection and on-axis/off-axis viewing.

4.2 Calibration software

Multi-projector systems are complex assemblies which require frequent and careful calibration, in particular in the ever-changing environment of instrumented office spaces. Therefore, ease of use and flexibility are key requirements for such systems. To address these concerns, we based the design of our calibration tool on a client-server model of communicating components running on our display cluster.

A server process handles all clients that registers at it, directing messages to the right clients. Each client represents a user, a projector, or a camera. The user client processes user inputs and invokes the different calibration steps on the user's demand. A projector client represents a display within the render cluster. The camera client handles the video stream from a camera and computes homographies and blending masks

The homographies are extracted using pattern recognition techniques of OpenCV [Intel04]. A simple chessboard pattern is projected for each

display and captured by the camera. The feature points of the pattern are extracted using the FindChessBoardCornerGuesses() function and the homographies are compute with the FindHomography() function.

These homographies are extended to 4x4 matrices to be usable in a common graphics pipeline. After the extension the matrices have to be corrected to approximate depth buffer values as described in [Raskar00] due to depth value distortion when the homographies are applied to common view frustum or projection matrices. The results from this calibration process, which is triggered by the user, but is otherwise designed to run fully automated, are written to configuration files used as input for the Studierstube display cluster.

The most common alteration of a display cluster, namely moving one of the projectors or camera(s) is thus easily handled by simply running the calibration software again. If the topology of the system is modified, such as adding or removing a projector, re-arranging the hosts or host-projector assignments or changing from monoscopic to stereoscopic (our system can also support passive stereo), a simple configuration file for the calibration software must be updated.

5 Results

To verify our hypothesis that a hybrid user interface system can be upgraded to a seamless tiled display, we let existing Studierstube applications run on the seamless tiled display. The application programmer's interface of Studierstube was not modified, so all demo applications worked well except for those cases where a specific multi-user network configuration was assumed for advanced collaboration functions.

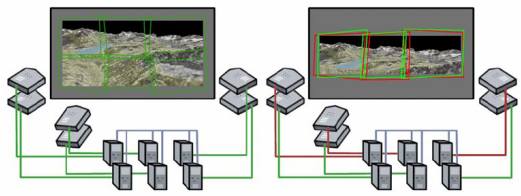

Fig. 5: A mono tiled display on the right and a stereo tiled display on the left. Note the overlap of two projectors at a time from the left eye group (red) and the right eye group (green).

The applications themselves did not have to be altered, only the configuration of the Studierstube runtime system for the hosts had to be matched to the new cluster system. Fig. 1 shows some examples of a hybrid user interface involving projection in action. Two users are collaborating in a geometry education session [Kaufmann03] involving tangible objects as input devices.

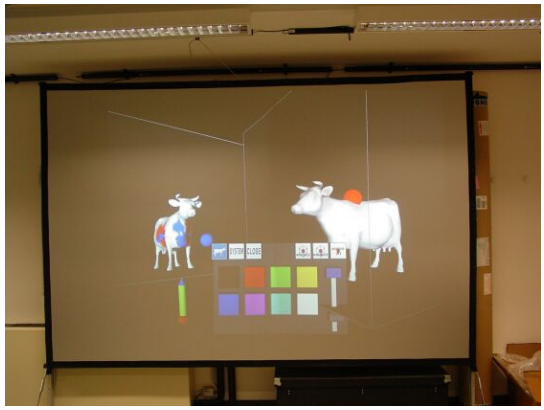

Fig. 6: In this 2x2 front projection setup at 1024x768 resolution, a casual observer cannot observe seams between individual projected images.

We have set up a 2x2 array of Sharp DLP projectors at 1024x768 resolution, which can also be re-arranged as a 2x1 passive stereo display, using polarization filters, glasses, and a polarization retaining silver screen (Fig. 5). Changing from the mono to the stereo configuration just requires re-adjusting the projectors in their mounts, attaching the filters, and re-running the calibration software, and can be done in under 10 minutes. Fig. 6 demonstrates the quality of registration and blending in 2x2 projectors monoscopic mode showing several visualization objects.

The oblique front projection from ceiling mounted projectors allows a user to approach the projection as near as ~1m without casting a shadow at the screen. All this is possible in a floor space of about 4x3m. The 2x2 display is driven by a cluster of 3 PCs (2.4GHz, Quadro4 graphics), one of which is designated as the master, the other two work as slaves each driving two projectors using the dual headed graphics card.

Initial examination of the performance confirmed our expectation that the system is reasonably scalable for an office-type environment. Remember that our system was not designed for highest scalability of very large visualization projects, but rather for bringing extensible display space to instrumented environments.

Naïve distribution of the overall scene graph to all nodes without any rendering optimizations naturally does only increase the display area, but not speed up rendering because a non-modified renderer does not make efficient use of the fact that it is only responsible for a tile.

We therefore developed a simple terrain renderer with view frustum culling and resolution-dependent level of detail (ROAM) rendering, and examined the frame rates in different cluster configurations.

Fig. 7: Schematic view of the two and three projector arrangements.

The first configuration uses a single host and a single projectors. The second configuration uses one host as the master, and two display hosts driving two projectors each (Fig. 7). The third configuration uses three host and three projectors, one host assuming a double role of display host and master. We rendered a terrain with about 500.000 polygons.

Display hosts	Projectors	Avg. fps	Polygons per host
1	1	12	25%
2	4	15	50%
3	3	20	25%

Table 1: Performance figures for the Terrain renderer in various cluster configurations

As can be seen from Table 1, we always achieve a speedup over the single host case even if one of the display hosts has to perform extra work as the master. However, these initial results only show the feasibility of our approach to cluster rendering, and are neither general (because of the small cluster) nor do they represent a carefully optimized result. They simply demonstrate that a scene graph based environment originally designed for hybrid user interface experiments with little focus on efficiency also performs reasonably when converted to a quasi parallel rendering mode.

6 Conclusions and Future Work

The system presented in this paper works well as a seamless tiled display, while simultaneously allowing the whole range of hybrid user interfaces to be deployed. Yet we are just starting to exploit the potential of our instrumented office environment to identify the potential advantages for novel user interface ideas. Future work will therefore primarily involve user interface and application experiments.

However, there are a number of technical limitations that we hope to address as well. One important aspect is the introduction of a more fine-grained multi-threading approach of scene graph traversal, in particular for better synchronization support. In addition, we would like to develop heuristics that can address rendering load balancing in the cluster in a more automated way.

Finally, while our calibration algorithm is scalable, the presented implementation only supports one camera. While this was not a limitation in practice for us, it restricts the physical scalability of the seamless tiled display. Therefore, we plan to extend our system to use multiple simultaneous cameras covering a larger display area.

Acknowledgements. This work was sponsored by the Austrian Science Foundation *FWF* under contract no. Y193, and Vienna University of Technology by *Forschungsinfrastrukturvorhaben TUWP16/2002*. We would like to thank Ramesh Raskar from MERL for the code sample for homography computation, and Alex Bornik from TU-Graz for the Linux version of the swapbuffer synchronization code.

7 References

[Allard03] J. Allard, V. Gouranton, G. Lamarque, E. Melin and B. Raffin: Softgenlock: Active Stereo and Genlock for PC Cluster. Proc. IPT/EGVE'03 Workshop, May 2003, Zurich, Switzerland

[Butz99] A. Butz, T. Höllerer, S. Feiner, B. MacIntyre, C. Beshers. Enveloping Computers and Users in a Collaborative 3D Augmented Reality, Proc. IWAR '99, pp. 1999.

[Hesina99] G. Hesina, D. Schmalstieg, A. Fuhrmann, W. Purgathofer: Distributed Open Inventor: A Practical Approach to Distributed 3D Graphics. Proceedings of ACM Virtual Reality Software & Technology '99 (VRST'99), pp. 74-81, London, December 20-22, 1999.

[Intel04] Intel Research: OpenCV web page, visited Jan. 2004. http://www.intel.com/research/mrl/research/ opencv/

[Ishii97] Ishii H., B. Ulmer. Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms, Proc. CHI '97, pp. 234-241, 1997.

[Kaufmann03] H. Kaufmann, D. Schmalstieg: Mathematics And Geometry Education With Collaborative Augmented Reality. computers & graphics, Vol. 27, No. 3, pp. 339-345, 2003.

[Li00] K. Li, H. Chen, Y. Chen, D. Clark, P. Cook, S. Damianakis, G. Essl, A. Finkelstein, T. Funkhouser, T. Housel, A. Klein, Z. Liu, E. Praun, R. Samanta, B. Shedd, J. Singh, G. Tzanetakis, J. Zheng: Building and Using A Scalable Display Wall System, IEEE Computer Graphics and Applications 25(4), 2000.

[Raskar98] R. Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin and Henry Fuchs. The Office of the Future : A Unified Approach to Image-Based Modeling and Spatially Immersive Displays, SIGGRAPH 1998.

[Raskar99] Ramesh Raskar, Oblique Projector Rendering on Planar Surfaces for a Tracked User, SIGGRAPH 1999 (1999), Sketches and applications.

[Raskar00] Ramesh Raskar, Immersive Planar Display using Roughly Aligned Projectors, In IEEE Virtual Reality (March 2000).

[Raskar02] Ramesh Raskar, Jeroen van Baar, Jin Xiang Chai A Low-Cost Projector Mosaic with Fast Registration, In Proceedings of Fifth Asian Conference on Computer Vision (January 2002).

[Raskar03] R. Raskar, J. van Baar, P. Beardsley, T. Willwacher, S. Rao, C. Forlines: iLamps: Geometrically Aware and Self-Configuring Projectors, SIGGRAPH 2003 (2003).

[Rekimoto97] J. Rekimoto. Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments, Proc. UIST '97, pp. 31-39, 1997.

[Rekimoto99] J. Rekimoto, M. Saitoh. Augmented Surfaces: A Spatially Continuous Workspace for Hybrid Computing Environments, Proceedings of CHI'99, pp.378-385, 1999.

[Schmalstieg00] D. Schmalstieg, A. Fuhrmann, G. Hesina. Bridging Multiple User Interface Dimensions with Augmented Reality. Proceedings of the 3rd International Symposium on Augmented Reality (ISAR 2000), pp. 20-30, Munich, Germany, Oct. 5-6, 2000.

[Schmalstieg02] D. Schmalstieg, A. Fuhrmann, G. Hesina, Zs. Szalavari, L. M. Encarnação, M. Gervautz, W. Purgathofer: The Studierstube Augmented Reality Project PRESENCE - Teleoperators and Virtual Environments, Vol. 11, No. 1, pp. 32-54, MIT Press.

[Schmidt00] D. Schmidt, M. Stal, H. Rohnert, F. Buschmann: Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects. Wiley & Sons, 2000.

[Stoll01]G. Stoll, M. Eldridge, D. Patterson, A. Webb, S. Berman, R. Levy, C. Caywood, M. Taveira, S. Hunt, and P. Hanrahan. Lightning-2: A High-Performance Display Subsystem for PC Clusters. Proceedings of SIGGRAPH '01.

[Strauss92] P. Strauss, R. Carey: An object oriented 3D graphics toolkit, Proceedings SIGGRAPH '92, pp. 341-347, 1992.

[Ullmer97] B. Ullmer, H. Ishii. The metaDESK: Models and Prototypes for Tangible User Interfaces. In Proceedings of ACM UIST'97, Banff, Alberta, Canada, pp. 223-232, 1997.

[Underkoffler99] J. Underkoffler, B. Ullmer, H. Ishii. Emancipated Pixels: Real-World Graphics in the Luminous Room. Proc. SIGGRAPH 1999, pp. 385-392, 1999.

[Walls00] D. Walls, R. Schikore, R. Fischer, R. Frank, R. Gaunt, J. Hobson, B. Whitlock: High-Resolution Multiprojector Display, IEEE Computer Graphics and Applications 25(4), 2000.

[Weiser91] M. Weiser. The Computer for the twenty-first century. Scientific American, pp. 94-104, 1991.