# The OmegaDesk: Towards A Hybrid 2D & 3D Work Desk

Alessandro Febretti, Victor A. Mateevitsi, Dennis Chau, Arthur Nishimoto, Brad McGinnis, Jakub Misterka, Andrew Johnson, Jason Leigh

Electronic Visualization laboratory, University of Illinois at Chicago

**Abstract.** OmegaDesk is a device that allows for seamless interaction between 2D and 3D content. In order to develop this hybrid device, a new form of Operating System is needed to manage and display heterogeneous content. In this paper we address the hardware and software requirements for such a system, as well as challenges. A set of heterogeneous applications has been successfully developed on OmegaDesk. They allowed us to develop a set of guidelines to drive future investigations into 2D/3D hybridized viewing and interaction.

## 1 Introduction

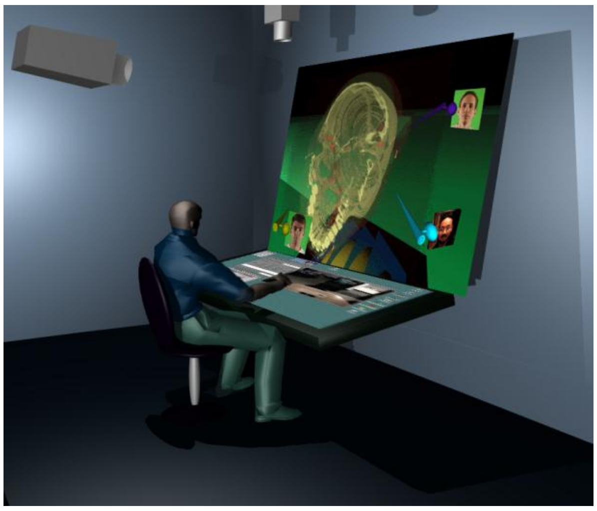

**Fig. 1.** This figure illustrates the initial concept of OmegaDesk as envisioned in 1999.

Historically, Virtual Reality (VR) systems have been thought of entirely for the purposes of supporting virtual world interactions. In 1999 the Electronic Visualization Laboratory (EVL) conceived of a new type of work desk that would blend 2D and 3D display and interaction capabilities to enable users to work seamlessly with 2D content (such as text documents and web browsers), as well as 3D content (such as 3D geometry and volume visualizations). We believed that for VR to emerge out of a small niche community, it had to become a seamless part of the computing continuum. At the time, the state of the art in hardware did not make such a conceived system practical. However today minimally encumbering and reliable stereoscopic displays and tetherless tracking systems are becoming highly affordable. Also, numerous vendors are emerging to provide multi-touch overlays that are easy to incorporate into existing display systems.

It is therefore possible now to develop our hybrid 2D/3D work desk, which we call OmegaDesk. What is still missing however is a new form of Operating System that enables the effortless and intuitive manipulation of both 2D content (such as spreadsheets, word processing documents, web browsers) and 3D content (such as CAD or scientific visualizations). In this paper we report on our first steps toward addressing this problem which resulted in the development of an API and exemplary applications for examining issues relating to 2D/3D hybridized viewing and interaction.

## 1.1 Vision

The effectiveness of presenting data in different modalities has been the subject of previous research. 2D views have been found to be better when used to establish precise relationships between data, and for visual search [1] and [2], while 3D is very effective for approximate 3D manipulation and navigation, especially with the use of appropriate cues, like shadows. In [3] it is suggested that combining both views leads to good or better analysis and navigation performance than using 2D or 3D alone. These findings are confirmed in [4], where in an air traffic control simulation 2D displays proved to be better for checking aircraft speed and altitudes while 3D was best used to perform collision avoidance.

Our vision for OmegaDesk is of an *integrated hardware and software system* that allows for rapid development and deployment of tools that make use of this hybrid visualization capability. Also, we envision OmegaDesk not specifically as a VR device or a Workstation, but a *Work Desk* - i.e. computer-enhanced furniture. Applicative scenarios range from scientific visualization of complex scientific datasets, ([5], [6] ), interaction with dynamic geospatial information (e.g. air traffic control, [4]), analysis of medical data for research or surgery planning ( [7] , [8]), and in general scenarios where a 3D, qualitative display of information can be enriched by a separate or overlayed 2D, quantitative view of the same information.

We will first describe the implementation of the OmegaDesk, and the middleware to drive it. Along the way we will describe some of the challenging issues we have encountered in building the system. Then we will describe the applications that we have built to test the system, and the lessons learned. Lastly we will conclude with an evaluation of developed case studies and our plans for future investigation and development of the system.

## 2 Related Work

Considered as a purely hardware system, the OmegaDesk structure is comparable to other designs**.** The sliceWIM system presented in [5] offers two separate views of the data, with interaction done exclusively through a touch interface. While effective, the system has been designed around a very specific task (exploration of volume datasets) and while it supports an overview and detail view of the data, it is not really designed to support the overlapping of 3D and 2D information. The IQ-Station [9] is a low cost immersive system based on a 3D display and a set of OptiTrack motion capture cameras. Although there are some technical similarities between the IQ-Station and the OmegaDesk, the former focuses less on the hybrid 2D and 3D aspect that is central in our design.

In the introduction we also stated how OmegaDesk needed an operating system or middleware that would enable the development of applications on a hybrid 2D/3D system. This middleware would allow for both high performance scientific visualization and interaction with higher level, rapid development toolsets. This gives application programmers the ability to rapidly develop on platforms such as Unity3D and Processing[10]. Additionally it was also important that there was a layer of abstraction between input devices and the developer.

A variety of libraries (as trackD[11] and Vrpn[12]) offer an abstraction layer to handle virtual reality input devices. Others, like freeVR[13] and the Vrui[14] toolkit take this a step further, integrating display management for 3D rendering. Products like getReal3D[15] allow users to design virtual reality environments using of high level toolsets (Unity in this case).

## 3 OmegaDesk Hardware

The OmegaDesk concept is illustrated in Fig. 2. OmegaDesk consists of two stereo displays, one positioned horizontally in a 45-degree angle and another positioned vertically in front of the user. The PC that drives the displays is a Windows 7 64bit machine on an Intel Core2 2.93GHZ with 4GB of RAM and two NVIDIA GeForce GTX 480 GPU cards. For OmegaDesk two Panasonic Viera TC-P65VT25 have been used.

The use of commercially available displays allows the flexibility of using any high-resolution 3D consumer display system and enables the low cost construction of such work-desks. While the cost of high-resolution 3D displays has dropped significantly in the past 5 years, it is our belief that it will drop further, making it affordable to build future OmegaDesk-like work desks.

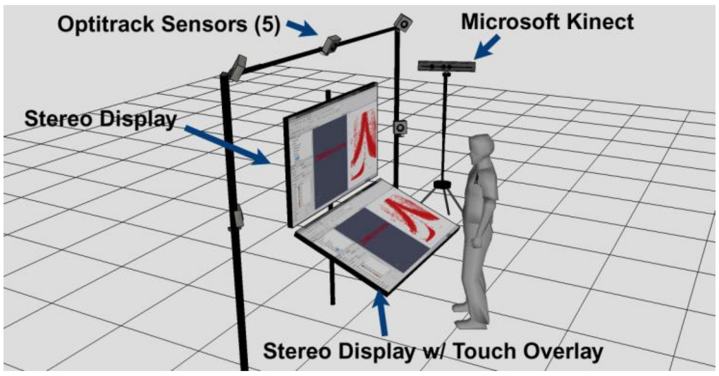

**Fig. 2.** This figure shows the various commercial technologies that make up OmegaDesk.

**Table 1.** Operational modes of OmegaDesk

| Operational Mode | Potential Application Usage |
|---|---|
| Top 3D, Bottom 3D | Fully immersive mode. Ideal for applications that require navigation thru a virtual space or bringing 3D objects close-up for manipulations, etc. |
| Top 3D, Bottom 2D | 3D Viewer mode. The vertical display is used to visualize 3D objects and worlds, while the horizontal display can be used to control aspects of the visualization. |
| Top 2D, Bottom 3D | 'Bathtub' mode. The horizontal display is used to look at 3D data bottom-down, like looking at a fish tank from top and the vertical display is used to look at 2D projections or slices of the data. |
| Top 2D, Bottom 2D | Touch augmented desktop / cubicle mode. The vertical display is the wall of the cubicle while the horizontal display is like a giant iPad where document editing and manipulation can be performed. |

## 3.1 Input Interfaces

For manipulation of objects in 2D the bottom display is overlayed with the Multi-Touch G$^3$ Plus overlay manufactured by PQLabs that can detect simultaneously up to 32 touches. For head tracking and 3D object manipulation OmegaDesk can use either the five OptiTrack FLEX:V100R2-FS positioned around OmegaDesk or a Microsoft Kinect. Kinect user tracking is performed through the OpenNI library[16]. While Kinect can perform tether-less multi-body tracking, it lacks the accuracy of OptiTrack and does not provide orientation for all the tracked body parts. On the other hand the coverage area of OptiTrack is reduced in comparison with the Kinect's (Fig. 3).

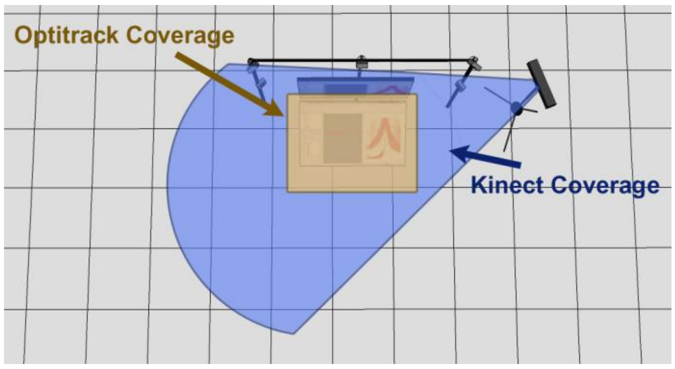

**Fig. 3.** This diagram shows the area of coverage of both the Optitrack and the Kinect.

Immersive navigation is accomplished with the use of game controllers. With the wide adoption of game consoles like the Wii, Xbox 360 and PlayStation 3 users are accustomed to navigate worlds using a game console. Both the PlayStation 3 and Xbox 360 wireless controllers can be used as props when developing applications for OmegaDesk.

## 4 Omegalib

The final software development objective for OmegaDesk would be the creation of a 2D-3D-aware Operating System. A first step towards that objective is the implementation of a middleware system that would ease the development of applications on hybrid work desks, and increase their portability across hardware changes or device configurations. We explained how none of the existing libraries was covering our full set of requirements in an easy, out-of-the-box way. This led us to build our own software development kit, called *Omegalib.*

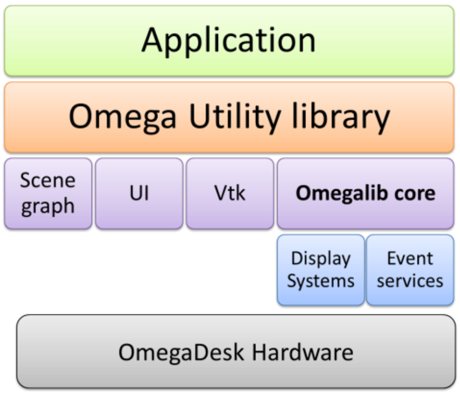

**Fig. 4.** This diagram shows the overall outline of the Omegalib architecture.

### 4.1 Hardware abstraction

Inside Omegalib, hardware abstraction is implemented through two concepts: *display system abstraction* and *input system* abstraction.

**Display system abstraction.** Omegalib manages rendering using the concept of display systems: A display system takes care of setting up the graphical hardware system, creating windows and viewports, setting up transformations and rendering pipelines and calling the appropriate application-level rendering functions. Currently, two display systems have been implemented: a simple GLUT based display system used mainly for debug purposes, and an Equalizer based display system.

Equalizer is a toolkit for scalable parallel rendering based on OpenGL. It allows users to develop scalable graphics applications for a wide range of systems ranging from large distributed visualization clusters and multi-processor multipipe graphics systems to single-processor single-pipe desktop machines [17]. In the near future, we are considering the introduction of a new display system to support autostereoscopic displays based on active parallax barriers, like the Dynallax [18].

The separation between rendering management and the actual application rendering code allowed us to support the concept of *rendering layers***.** Layers represent conceptually separate sets of graphical primitives (for instance a 3D scene and a 2D interface) that can be enabled or disabled for specific output channels of the display system. In this way, it is very easy to implement separate 3D views for the same application, or create a management window running on a secondary display, showing an administration UI or a debug-mode scene rendering.

It is also possible to perform rendering of layers on separate threads, and compose them in the target channel frame buffer: this can be used to make the rendering performance of 2D elements of the application independent from the complexity of the 3D scene, in order to maintain a good frame rate and responsiveness on the UI as the visualized scene grows in complexity.

**Input device abstraction.** Omegalib gives applications access to input devices through the concept of *event services*: an event service manages one physical or logical event source in the system. For instance it can:

- offer access to events from a real input device, like a touch display or a motion capture system;
- receive events from a remote source through a network connection;
- generate input from a logical source, like a user interface button or slider;
- process events from other sources to act as a *background utility service*. For example, a service can get position data for the user head from a tracking or motion capture service, update the observer head matrices for a scene and send the application updates on the user tracking status).

Event services allow for a great deal of flexibility. They abstract the physical input devices available to the system. Also, they allow to modularize several common components of a virtual reality application (like user tracking or network message passing), so that they can easily be reused in applications.

Omegalib also supports the streaming of events to external applications, acting as a display-less *input server*. This simplifies the development of OmegaDesk applications using differents toolsets (as Unity or Processing) and streamlines the integration of input support into legacy applications that treat the device displays as normal screens, but want to use the motion capture, tracking or multitouch capabilities of OmegaDesk.

**Configuration.** Similar to other VR libraries, Omegalib allows applications to be reconfigured using system description files: display system, event service and application parameters are all stored in configuration files: the same application can run on OmegaDesk with head and hand tracking, on a multitouch tiled display without stereo support, or on a developer laptop using just mouse and keyboard interaction.

### 4.2 Interaction

Through use of tracker based mocap, Kinect user tracking and touch screens OmegaDesk offers a wide range of possibilities in terms of user interaction. Different applications may request subsets of the available input devices and implement an interaction scheme that works best for the specific application scenario: in some instances, the motion capture system may be used just for head tracking, while interaction with the application 3D objects can be realized through the touch screen. In other scenarios we may need a full mocap-based interaction scheme, with direct hand manipulation of the 3D objects.

We think a certain, predefined number of interaction metaphors would satisfy most of the interaction needs of final applications. In this case, it makes sense to modularize them and make them available to application developers as packaged interaction schemes that can be easily turned on, off or switched inside an application, allowing for both consistency and reuse of interaction schemes, and fast prototyping of applications using different interaction techniques. To implement this, omegalib offers support for a simple scene graph system based on Ogre[19] that can be controlled through *interaction objects*. These objects implement interaction policies, by getting input from the event services and controlling nodes and objects in the scene graph.

### 4.3 Integration with Scientific Visualization tools.

One of the purposes of OmegaDesk is to be used as a scientific visualization tool: it is therefore necessary to integrate it with standard tools and libraries, like the Visualization Toolkit (VTK) [20]**.** Through Omegalib, Omegadesk is able to load VTK pipelines as python scripts, render them through the omegalib display system and interact with VTK actors and 3D models using the interaction schemes presented in the previous section. VTK python scripts can also create user interface widgets that modify the visualization pipeline, and can be controlled through the touch screen. It is also possible to create VTK programs for OmegaDesk natively, using the C++ VTK API directly. This makes it extremely easy to build VTK programs for OmegaDesk or port legacy pipelines to the system.

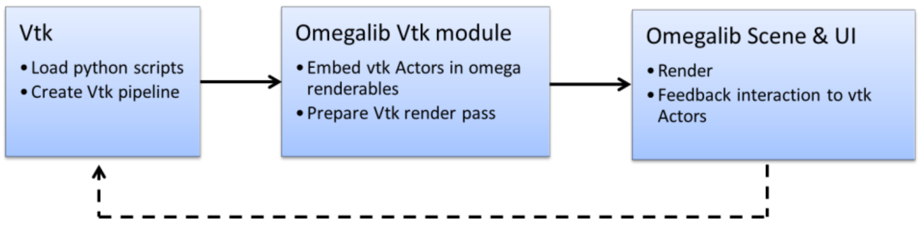
Vtk
• Load python scripts
• Create Vtk pipeline
Omegalib Vtk module
• Embed vtk Actors in omega renderables
• Prepare Vtk render pass
Omegalib Scene & UI
• Render
• Feedback interaction to vtk Actors

**Fig. 5.** The integration of VTK pipelines inside Omegalib is done through a support module that performs VTK actor encapsulation and feeds back user actions to the pipeline.

## 5 Application Case Studies

A set of heterogeneous application has been developed on OmegaDesk so far. Some are built to test the interaction and display capabilities of the system while others are designed to solve domain-specific problems in areas as different as rehabilitation therapy, histology or fluid dynamics.

### 5.1 Mesh Viewer / VTK Viewer

The mesh viewer application has been developed to test 3D object manipulation via hand gestures. It allows the user to drop one or more objects inside a 3D scene by selecting them through the touch display. Interaction takes place using both hands to intuitively perform rotation, scaling and moving. Head and hand tracking can be provided by the Optitrack system or the Kinect alone.

The VTK viewer application takes the mesh viewer concept a step further: it supports loading of VTK pipelines through python scripts and rendering of multiple VTK actors. These actors can then be manipulated using the same interaction techniques offered by the mesh viewer. Additionally, selected parameters of the VTK pipeline can be configured at runtime though a touch interface created dynamically on the bottom display.

### 5.2 Physical Therapy Simulation

The Physical Therapy Simulation is a rehabilitation exercise created using Unity3D and Omegalib through a collaborative effort with the Kinesiology department at UIC. It is used to test the efficacy of physical therapy through the use of VR. The scene consists of a simple room where a virtual ball is tossed to the patient. This has the effect of strengthening feedforward postural control in the user/patient which allows for maintaining a quality of balance during daily movements.

This application will help determine if visual stereoscopy will provide enough visual cues to the brain to enhance current physical therapy methods. It utilizes Omegalib's data streaming capability from an OptiTrack motion capture system and Kinect.

### 5.3 Histology Viewer

With the development of powerful microscope optics and the latest advances in image sensors that deliver high resolution imaging capabilities, the scientists are able to dwell into the micro and nano scale to explore sightings unseen under normal conditions by the naked eye. In particular, in the medical lasers research field, physicians study 1cm by 0.5cm blocks of laser damaged skin. Using specialized hardware the block is sliced in 4 microns thick slices and digitized by the use of a powerful microscope equipped with a medical imaging device. Typically the physicians use a standard image viewer to browse through the histology images and identify the damaged parts.

To leverage the OmegaDesk capabilities, a prototype Histology Viewer was developed. The skin block is reconstructed by stacking the slices and using ray-casting algorithms to generate a data volume. The top display visualizes the 3D reconstruction and gives physicians the ability to look at the data with an high level of detail. The bottom multi-touch display controls the visualization and is used to select what slices of the block will be shown. The physicians can browse back and forth through the data by touching and sliding and also select slices of interest to investigate further. Zooming and rotating are also supported by the pinching and rotating gestures.

### 5.4 Flow Visualization

FlowViz is a generic 3D flow visualization for Omega Desk, The application has been built using Processing, and has been designed to be easily portable to devices offering a subset of the capabilities of OmegaDesk The goal of the project was to create a tool that would enable the viewer to better understand the complex nature of flow data. It is thought that viewing the complex 3D flow in a native 3D environment will allow the viewer to better understand its behavior. Also, by utilizing the multi-touch interface the viewer is allowed to interact with the simulation in an intuitive way: users can touch a 2D representation of the 3D view, causing a stream source to be spawned from the point touched. This source can either be a dynamic particle generator or a static streamline. Particles will flow through the vector field, exposing its behavior. In addition the user may spawn multiple plot windows showing different representations of the model. Users can brush over and select portions which outline corresponding regions of the 3D data.

## 6 Evaluation and Future Work

This paper presented OmegaDesk, a prototype 2D and 3D work desk. We described the requirements for such a system to be effective, and how we addressed them at the software and hardware level. The development of several heterogeneous applications on the system allowed us to assess its efficacy in very different domains.

The presented applications made use of different device modalities. The mesh viewer used both displays as 3D viewports to create a more immersive experience, overlaying a 2D user interface on the touch-enabled screen, and used hand gestures to interact with the data. The histology and flow visualizations treated the bottom screen as a 2D data presentation display, with the entire interaction driven by the touch surface (no hand gestures). Finally, the physical therapy simulation made use of the top 3D screen only. In this case the interaction was based on hand and head tracking, without the need for touch support. Even the current set of applications does not cover all of the possible OmegaDesk configurations, it allowed us to develop an initial set of considerations and guidelines for future development on this platform.

It is clear how 3D hand gestures can be used for approximate object manipulation, or for applications that don't need precise control. In these instances they can be a very effective and intuitive way of interacting with the system. When more control is needed though, the power and precise control offered by a touch screen and a 2D or 2.5D interface is unmatched. In this case, it is very important to link the information displayed on the 2D and 3D portions of the application, so that changes in one view of the data influence all the others. These changes should be propagated as quickly as possible and, most importantly, each view of the data should be able to update itself on the displays, without depending on the refresh speed of other views. This is similar in concept to the separation of processes running in an operating system: even if they can exchange data with each other, none of them should be allowed to slow down the entire system.

Our future work will involve not only building new applications leveraging OmegaDesk capabilities, but also continuing the development of omegalib, to make it a complete, Operating System – like middleware supporting complex multimodal development on our evolving hardware system.

## Pictures

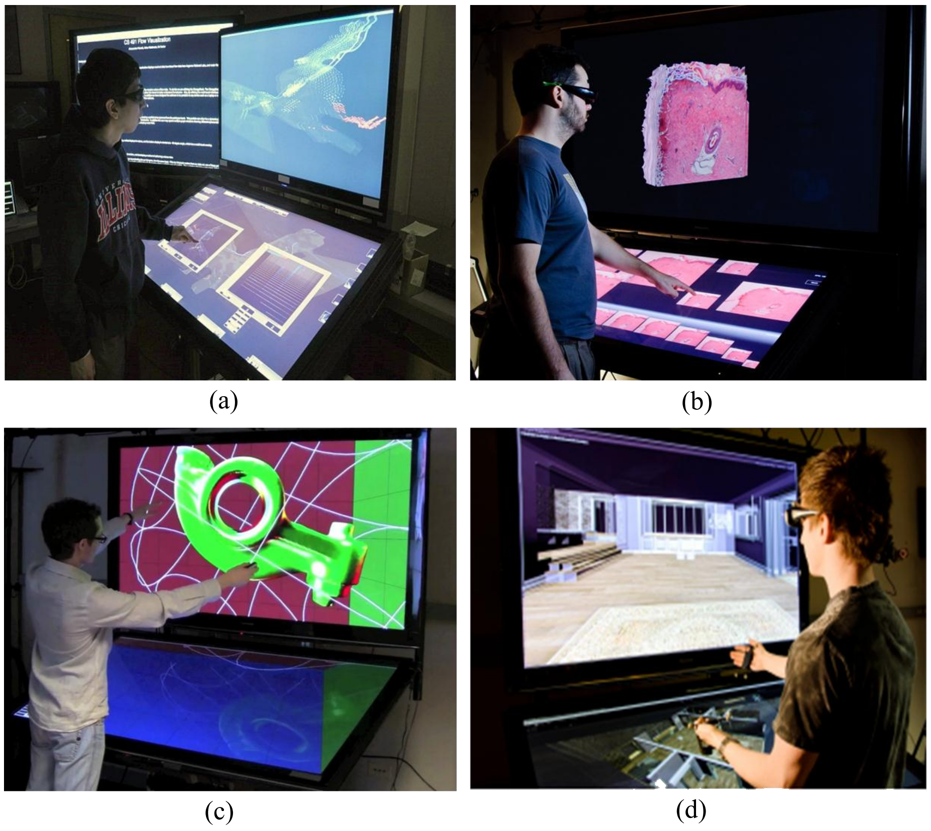

**Fig. 6.** Photos of several applications running on OmegaDesk. (a) User interaction with 2D graphs of water flow in a specific area in Corpus Christi Bay as he compares them to the vector field of the surrounding areas. (b) Reviewing 2D histology slides while comparing them to the 3D volume rendering. (c) A user rotating and translation an object within the mesh viewer. (d) A user using OmegaDesk to simulate catching a ball as part of physical therapy.

## Acknowledgements

This publication is based on work supported in part by Award Nos. FA7014-09-2-0003, FA7014-09-2-0002, made by the US Air Force, and Award CNS-0935919, made by the National Science Foundation.

## References

1. Smallman, H.S., St John, M., Oonk, H.M., Cowen, M.B.: Information availability in 2D and 3D displays. Computer Graphics and Applications, IEEE. 21, 51–57 (2001).
2. Springmeyer, R.R., Blattner, M.M., Max, N.L.: A characterization of the scientific data analysis process. Visualization, 1992. Visualization '92, Proceedings., IEEE Conference on. pp. 235–242 (1992).
3. Tory, M., Kirkpatrick, A., Atkins, M., Moller, T.: Visualization task performance with 2D, 3D, and combination displays. IEEE transactions on visualization and computer graphics. 12, 2–13 (2006).
4. Van Orden, K., Broyles, J.: Visuospatial task performance as a function of two- and three-dimensional display presentation techniques. Displays. 21, 17–24 (2000).
5. Coffey, D., Malbraaten, N., Le, T., Borazjani, I., Sotiropoulos, F., Keefe, D.F.: Slice WIM: a multi-surface, multi-touch interface for overview+detail exploration of volume datasets in virtual reality. I3D '11: Symposium on Interactive 3D Graphics and Games. (2011).
6. Kreylos, O., Bethel, E.W., Ligocki, T.J., Hamann, B.: Virtual-Reality Based Interactive Exploration of Multiresolution Data. Presented at the.
7. Hemminger, B.M., Molina, P.L., Egan, T.M., Detterbeck, F.C., Muller, K.E., Coffey, C.S., Lee, J.K.T.: Assessment of real-time 3D visualization for cardiothoracic diagnostic evaluation and surgery planning. J Digit Imaging. 18, 145–153 (2005).
8. Pechlivanis, I., Schmieder, K., Scholz, M., König, M.: 3-Dimensional computed tomographic angiography for use of surgery planning in patients with intracranial aneurysms. Acta …. (2005).
9. Sherman, W.R., O'Leary, P., Whiting, E.T., Grover, S., Wernert, E.A.: IQ-Station: a low cost portable immersive environment. ISVC'10: Proceedings of the 6th international conference on Advances in visual computing. Springer-Verlag (2010).
10. Processing, http://processing.org/, (2011).
11. trackd, Mechdyne Corporation, http://www.mechdyne.com/trackd.aspx, (2011).
12. Russell M Taylor Ii, T.C.H.A.S.H.W.J.J.A.T.H.: VRPN: A Device-Independent, Network-Transparent VR Peripheral System. (2001).
13. FreeVR, http://www.freevr.org/, (2011).
14. Vrui VR Toolkit, http://idav.ucdavis.edu/~okreylos/ResDev/Vrui/index.html, (2011).
15. getReal3D, Mechdyne Corporation, http://www.mechdyne.com/getreal3d.aspx, (2011).
16. OpenNI, http://www.openni.org/, (2011).
17. Eilemann, S., Makhinya, M., Pajarola, R.: Equalizer: A Scalable Parallel Rendering Framework. Visualization and Computer Graphics, IEEE Transactions on. 15, 436–452 (2009).
18. Peterka, T., Kooima, R., Sandin, D., Johnson, A., Leigh, J., DeFanti, T.: Ad-

vances in the Dynallax Solid-State Dynamic Parallax Barrier Autostereoscopic Visualization Display System. IEEE transactions on visualization and computer graphics. 14, 487–499 (2008).
19. OGRE, http://www.ogre3d.org/, (2011).
20. Schroeder, W.J., Avila, L.S., Hoffman, W.: Visualizing with VTK: A Tutorial. IEEE. 1–8 (2000).