WebVR: an interactive web browser for virtual environments

Barsoum, Emad, Kuester, Falko

Emad Barsoum, Falko Kuester, "WebVR: an interactive web browser for virtual environments," Proc. SPIE 5664, Stereoscopic Displays and Virtual Reality Systems XII, (22 March 2005); doi: 10.1117/12.582624

Event: Electronic Imaging 2005, 2005, San Jose, California, United States

WebVR: An Interactive Web Browser for Virtual Environments

Emad Barsoum and Falko Kuester

Cal-(IT)^2 Center of Gravity, University of California, Irvine, CA 92697

ABSTRACT

The pervasive nature of web-based content has lead to the development of applications and user interfaces that port between a broad range of operating systems and databases, while providing intuitive access to static and time-varying information. However, the integration of this vast resource into virtual environments has remained elusive. In this paper we present an implementation of a 3D Web Browser (WebVR) that enables the user to search the internet for arbitrary information and to seamlessly augment this information into virtual environments. WebVR provides access to the standard data input and query mechanisms offered by conventional web browsers, with the difference that it generates active texture-skins of the web contents that can be mapped onto arbitrary surfaces within the environment. Once mapped, the corresponding texture functions as a fully integrated web-browser that will respond to traditional events such as the selection of links or text input. As a result, any surface within the environment can be turned into a web-enabled resource that provides access to user-definable data. In order to leverage from the continuous advancement of browser technology and to support both static as well as streamed content, WebVR uses ActiveX controls to extract the desired texture skin from industry strength browsers, providing a unique mechanism for data fusion and extensibility.

Keywords: Virtual Reality, 3D Web Browser, WWW, Navigation, 2D/3D User Interface

1. INTRODUCTION

Texture mapping has become a predominant technique when it comes to adding content and realism to 3D environments, while keeping the polygonal budget low. However, commonly these textures are statically mapped and therefore fail to capture time varying information. At the same time this approach provides only one-directional flow of information and therefore prohibits direct interaction with the texture content. In particular, virtual environments are now heavily relying on texture-based techniques to further increase the realism of the visualized model or scene. The emerging need for pervasive data integration and the support for the visualization and analysis of complex time varying information in virtual environments requires support for dynamic information exchange in a bi-directional fashion. In other words, users should be able to freely operate on the presented data, reference other data source and correlate them, regardless of where the information exists. The pervasive nature of web-based content has lead to the development of applications and user interfaces that port between a broad range of operating systems and databases, while providing intuitive access to static and time-varying information. However, the integration of this vast resource into virtual environments has remained elusive.

Leveraging from the capabilities of commodity browsers, the most current advancements in Internet browser technology and its power to interact with different types of information such as text, video, images, sound,... etc., can be combined. In addition, with the continuous advances of the HTML and DHTML standards, many conventional applications can be written as web pages using JavaScript. Alternatively, Java applets and ActiveX components can be embedded into HTML pages to provide access to existing applications. Within the web context we can the easily write applications that interact with each other and provide a means for users to jointly work in shared virtual spaces, regardless of this space being inside or outside the actual VR system. Interestingly, many web have the potential to be embedded into other applications, i.e. Internet Explorer can be embedded using ActiveX controls, Mozilla using XPCOM, and for many other open source browsers such as KHTML, FireFox and Mozilla, the available source code can be directly used.

In this paper we present a 3D web browser (WebVR) for virtual environments that enables the user to browse the web for arbitrary information and to embed the queried information into the virtual environment without having to leave the virtual space.

2. RELATED WORK

A number of researchers have investigated different strategies for optimizing the use of 2D as well as 3D display technology to interact with multi-dimensional context and content rich information. In particular, research in the area of 3D graphical user interfaces has received growing attention.[1–6] Many of these research projects are based on simply adding depth to the otherwise flat user interface design or by embedding the 2D interface into an otherwise 3D environment. In contrast, Wasson[7] implemented 3D desktop using OpenGL for switching virtual desktops in a seamless 3-dimensional manner on Linux, by converting the entire desktop as a bitmap texture and map it on a 3D object, so each virtual desktop is mapped on different surface on a 3D object or on different 3D object and the current selected desktop is mapped into a fullscreen. Another example is MacOS-X[8] from Apple that pursues a similar approach. Provided with these types of work spaces the development of data fusion strategies that facilitate display of, and interaction with, multi-dimensional data in feature rich virtual environments is further stimulated for regular desktop environments as well as semi-immersive or fully immersive settings.

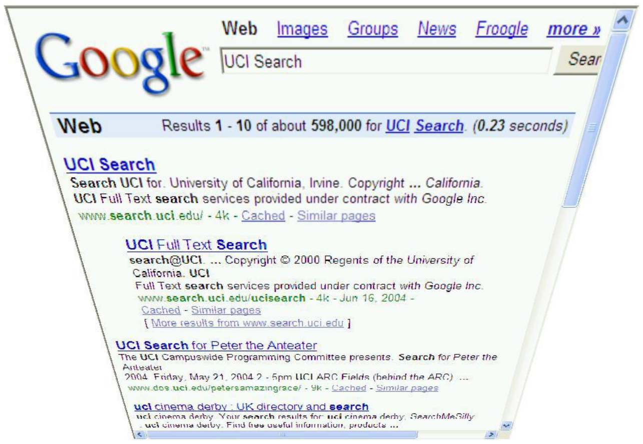
Hascoet[9] has provided advanced visualization and interaction paradigms to support web browsing through the intensive use of bookmarks and navigation history, his motivation was based on the fact that most studies show the importance of bookmarks and navigation history, while most web browser do a poor job to support them. Angus and Sowizral,[10] combined the open source code of the Mosaic Browser with the InterViews toolkit to provide web contents inside a virtual environment (VE). In the later case, the utilized graphical user interface toolkit was tasked with creating a two-dimensional visual interface within a three-dimensional space while supporting mouse input for interaction. To provide additional haptic feadback, Bowman et al.[11] describe a virtual notepad for VEs that enables the user to take notes and to augment information on top of the existing scene. This virtual notepad was developed for the DIVE system, which also has been used by Hagsand[12] to explore possible modes of multi-user interactions within VR systems contributing to the emerging question, if both the notion of 3D user interfaces and data representation can be combined to provide more natural access to multi-dimensional data in virtual environments.

3. IMPLEMENTATION

WebVR is build on two main components, consisting of (i) a web browser engine and that functions as a plug-in for (ii) a VR engine. The web engine manages the conversion of web contents to active texture maps consisting of an OpenGL compatible bitmap and a texture context that handles all events (input and output requests) associated with the texture. The VR engine itself manages the scene graph and the display loop in combination with a device loop that manages system wide input and output events. The design is based on the Model-View Controller model (MVC).[13] MVC was selected due to its modularity and flexibility, which allowed the concurrent exploration of different algorithms in combination with visual representations of the scene.

The underlying idea is that the web browser engine can create visual snapshots (images) of the currently displayed page and pass this image data to the scene graph of the VR system and subsequently to OpenGL, such that it can be used as a texture skin on any 3D object. For this texture skin to be useful the user must be able to interact with it analogously to a normal webpage. In order to facilitate this a dynamic rather than a static texture is required that can be continuously streamed to the render system, while user input can be passed back to the browser.

Rather than embedding a task specific browser implementation into our framework the presented approach leverages ActiveX controls to extract the desired texture skins from a commodity browser, providing a unique mechanism for data fusion and extensibility. A strength of this approach, termed WebVR, is that it provides access to the standard data input and query mechanisms offered by conventional web browsers with the difference that the browser is tasked with the creation of active texture-skins which are subsequently passed to the virtual reality system for integration with the existing scene. Once the texture-skin is mapped mapped onto an arbitrary surface, the corresponding texture functions as a fully integrated web-browser that will respond to traditional events such as the selection of links or text input. As a result, any surface within the environment can be turned into a web-enabled resource that provides access to user-definable data. Using this technique WebVR extends the possible application for web-centric data and automatically inherits advanced feature sets as browser capabilities grow.



Web Results **1** - **10** of about **598,000** for UCI Search. *(0.23 seconds)*

UCI Search
Search UCI for. University of California, Irvine. Copyright ... *California.*
UCI Full Text **search** services provided under contract *with Google Inc.*
www.**search.uci**.edu/ - 4k - Cached - Similar pages

UCI Full Text **Search**
search@UCI. ... Copyright © 2000 Regents *of the University of*
California. UCI
Full Text **search** services provided under contract *with Google Inc.*
www.**search.uci**.edu/**ucisearch** - 4k - Jun 16, 2004 -
Cached - Similar pages
[More results from www.search.uci.edu]

UCI **Search** for Peter the Anteater
The UCI Campuswide Programming Committee presents. ***Search*** *for Peter the* Anteater
2004. Friday, May 21, 2004 2 - 5pm UCI ARC Fields *(behind the ARC)* ...
www.dos.uci.edu/petersamazingrace/ - 9k - Cached - Similar pages

uci cinema derby : UK directory and **search**
uci cinema derby. Your **search** results for: **uci** *cinema derby. SearchMeSilly*
. **uci** cinema derby. Find free useful information, products ...

Figure 1. Web search query mapped onto an arbitrary polygon.

3.1. Texture Skin Creation

For the texture skins to function properly, it has to be possible to generate textures for arbitrary web contents, including regular text, forms, images, videos, animation or combination thereof. At the same time it must be possible to pass input back to the active page if it supports more than passive viewing.

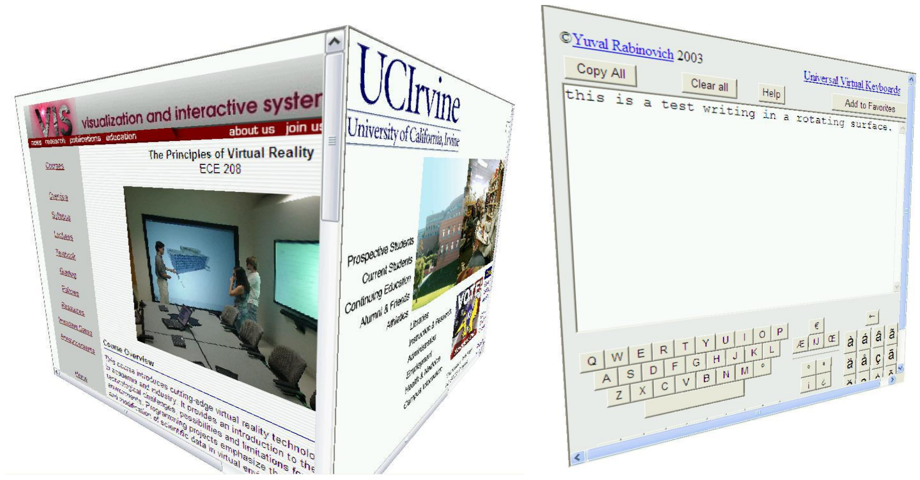
Each active texture consists of a texture and I/O context pair that is associate with its own instance of the web browser engine as well as an active reference address. Active textures inherit all of OpenGL's texture mapping functionality while adding a texture specific reference frame that is used to map input events to the proper image coordinates that can be passed to the browser for further image specific queries.

The creation of an active texture is broken up into multiple steps, (1) the user creates an instance of an active texture and provides a reference web address, (2) the web engine retrieves the corresponding document and renders its contents into an off-screen buffer, (3) the acquired image is converted into a bitmap and an OpenGL compatible texture is returned, including information about the image size and format. Subsequently, (4) this texture is mapped onto arbitrary geometry within the scene (Figure 1).

The web browser engine utilizes off-screen render capabilities to remain hidden from the user, as the reference textures are being generated from web documents. This is achieved by assigning an individual instance of the web browser engine, in this case an Internet Explorer (IE) ActiveX Control, to each active texture object. Once created the instance of the web browser stays active as long as the active texture exists. Once an active texture is mapped, two different events can trigger updates to the texture. The first is an update event initiated when a remote server is pushing new web contents, as can expected for updates to animated images, video streams, applets, etc. The second is a user triggered input event that may correspond to the desire to follow a web link embedded in a page, provide keyboard input for a form (Figure 2), or to navigate an interactive application.

3.2. Interactive Updates

Server side push events are supported via a user tunable active textures loop. During each pass of this loop, all browser instances verify if their corresponding web contents was updated. If an update occurred, a new texture and texture context are automatically generated and updated within the scene. This approach provides for seamless integration of time-varying, web-based data into the virtual environment. The selected refresh rate is critical decision, since animations and movies will appear the smoothest if the observe the refresh rate provided within the browser. However, image acquisition at these rates will introduce CPU and memory overhead as well as possible delays while updating the information within the scene graph and OpenGL context respectively. The default for the user selectable refresh rate was set to seven frames per second for streamed images.



this is a test writing in a rotating surface.

Figure 2. Interactive text input using a standard applet.

User events are processed in the context of the current viewing position and viewing direction as well a the interaction tool and mode to determine the active texture that the user is currently interacting with as well as the intended operation. Once identified, the texture context is used to convert screen coordinates or device coordinates (data glove position, etc.) to the proper image coordinates which are then passed to the web browser to query for further mode specific information, i.e. new image data. Since active textures can be mapped onto arbitrary surfaces within the scene, it is possible to create active wallpaper, floors, ceilings or objects that can fuse real-time web-based contents into the virtual environment. Figure 3 shows a simple example for an active texture space in which all textured surfaces are active, regardless of the type of data being displayed.

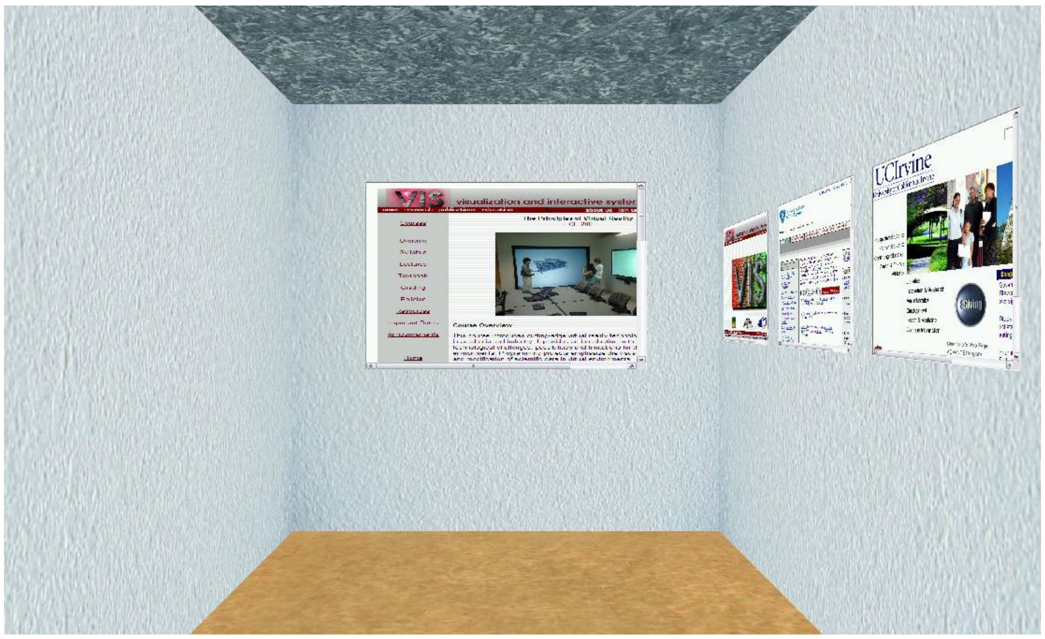


Figure 3. Room environment based on active textures associated with each polygon.

3.3. WebVR Architecture

Figure 4 provides an overview of the WebVR system architecture. At execution time, WebVR loads the model, creates the scene graph and creates the desired number of active textures. For each active texture, an individual instance of the IE ActiveX Control is created, the initial references queried, converted to a bitmap and mapped as a texture skin onto the target geometry. More specifically, after each instance finishes loading its appropriate web page is renderer off-screen, captured an converted to a bitmap. The rendering stage that maps from the web page to the final bitmap always creates a bitmap whose depth corresponds to the depth supported (enabled) by the available hardware. To compensate for hardware differences, the depth resolution of the generated bitmap is verified and if necessary converted to 24 bit or use alpha channel.

To be able to see animation in any web page that contains flash animation or video clips, we need to update all bitmaps from each instance of IE ActiveX periodically. If we choose high refresh rate all animation will run smoothly, but at the expense of CPU and memory usage, because each update involves a bitmap copy and if we choose low refresh rate any animation will be flickering and will be seen as slide show. We choose refresh rate of 4 frames per second which work smoothly for low frame rate animation like flash and doesn't consume much CPU and memory especially when a lot of surfaces is loaded. Also to reduce the overhead on the CPU and memory, we only update the surfaces that are visible to the user.

When the user interact with any 3D object in the scene, he interact in 3D coordinate system, which need to be translated to 2D coordinate system relative to the web browser surface that the user select, before send it as window message to the web browser. To do that we used OpenGL API "gluUnProject" and "glReadPixels" to get the 3D (x,y,z) position of the mouse relative to the 3D scene before any affine (translation and rotation) transformation in the scene. By knowing the initial position of each surface that contains web browser and which surface the user has select, we can map directly the 3D mouse position before affine transformation to 2D position relative to the web browser surface.

Throughout this process the instances of the browser will remain invisible to the user.

To provide development support on regular desktop systems, WebVR offers three modes of operations including (1) web interactive, (2) 3D Interactive and (3) VR. While in web interactive mode the user can interact with the active texture using a keyboard and mouse interface as if it is a regular webpage. The 3D interactive mode provides access to the same conventional interface, but in this case allows free scene manipulation while preventing input to the web engine. State changes can be easily toggled through a hot key. In VR mode, navigation and interaction are decoupled via the input device, i.e. a spatially tracked data glove with mode specific gestures.

3.4. Bitmap Creation

Different strategies are available to perform a screen capture on a running application. One approach is to capture the window handle of the application, obtain its device context and subsequently to use this device context to create the bitmap. Unfortunately, for this to work, the application must be visible to the user. This causes undesired problem since the browser window will appear in front of the running VR application whenever a new image is being generated. Furthermore, this approach becomes problematic when many active textures have to be supported simultaneously. WebVR therefore takes advantage of ActiveX controls to embed the browser capabilities into our framework. In this case a window handle as well as the device context for ActiveX to user as the host application have to be specified. A bitmap can be created as the device context and appear as a regular device context to the ActiveX control, just as a screen or printer would.

Figure 5 shows this process in more detail. A hidden window was created to host the ActiveX control, and a device context from a bitmap which is compatible with the screen in respect to its size, resolution and depth. Two different interfaces have to be implemented for all visible ActiveX controls, consisting of the *IViewObject* and *IOleObject*. The *IOleObject* has information about the size of the ActiveX control, which can be used to obtain the precise rectangle dimensional in pixel units (HMETRIC unit converted to pixel unit) that the ActiveX Control occupies. *IViewObject* is the interface responsible for the drawing of the object inside the host application and therefore requires a valid device context which in this case is the newly created bitmap compatible device context. To obtain the *IViewObject* and *IOleObject* interface pointers from the ActiveX control, a two step

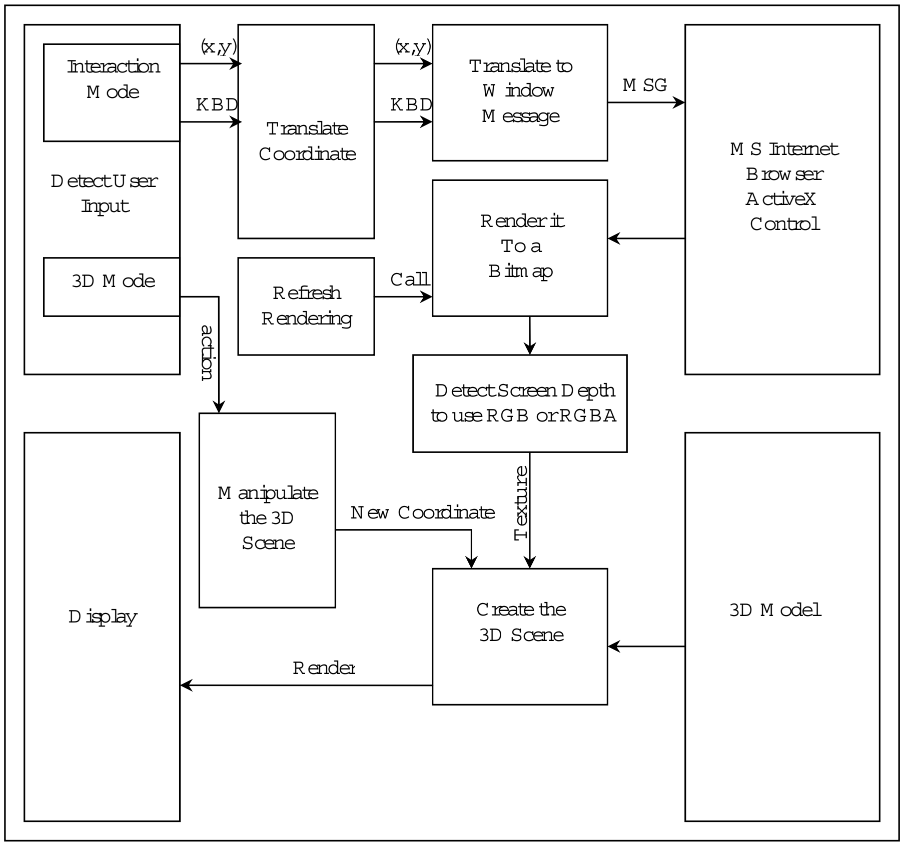


Figure 4. WebVR flow chart.

approach can be employed that first gets the *IUnknown* interface and subsequently uses the *IUnknown* interface to query the pointers for *IViewObject* and *IOleObject* via the *QueryInterface* method.

To be able to capture a bitmap from the ActiveX control, the ActiveX itself needs to be visible, however, the host window does not. This allows the bitmap creation to completely take place off-screen. While WebVR currently uses the IE ActiveX control, the chosen approach does work any ActiveX control, providing flexibility for the fusion of other data sources.

4. RESULTS

In this paper we have implemented a web browser that can be used inside 3D environment and can be mapped to any 3D surface. We tested WebVR in various web sites and compared its speed to MS Internet Explorer outside the 3D scene, specially sites that contain animation, 4 frames per second refresh rate work smoothly for most animations, except animations that require higher frame rate like video.

The delay in web interaction depends on the complexity of the scene, on how many surfaces contain a web browser and on the hardware used specially the graphic card, processor and memory. For a simple 2D square surface the interaction with the web like write a text or click on a link is almost the same as Internet Explorer, but for Cube shape which have web browser on all its surfaces, it lag 2 to 4 second behind IE depend on the action. The hardware used in this test is Pentium IV 1.7 GHz, 256 MB RAM and Getforce 16 MB graphic card.

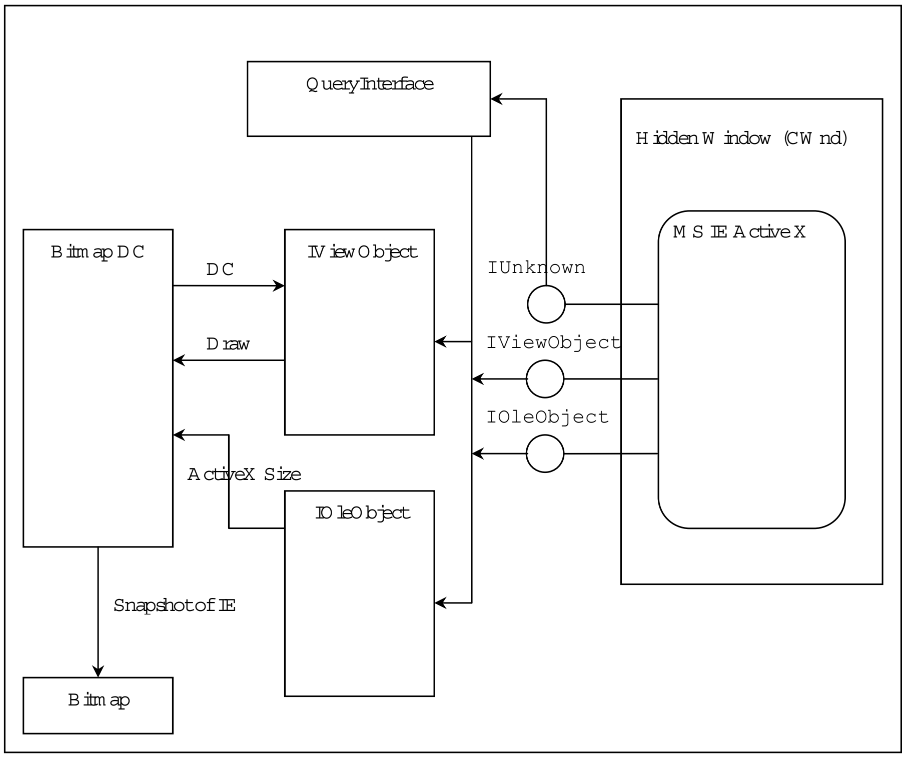


Figure 5. Capturing MS Internet Explorer as Bitmap Image.

5. CONCLUSIONS

The pervasive nature of web-based content has lead to the development of applications and user interfaces that port between a broad range of operating systems and databases, while providing intuitive access to static and time-varying information. We have presented a method that allows the fusion of web content into virtual environments and described the implementation strategy. The resulting framework, titled WebVR provides access to the standard data input and query mechanisms offered by conventional web browsers, with the difference that it generates active texture-skins of the web contents that can be mapped onto arbitrary surfaces within the environment. Once mapped, the corresponding texture functions as a fully integrated web-browser that will respond to traditional events such as the selection of links or text input. As a result, any surface within the environment can be turned into a web-enabled resource that provides access to user-definable data. The presented approach has demonstrated that web contents can be merged into virtual envWebVR shows good performance results and web-based video streams can be merged into the running framework at 7Hz and we anticipate that real-time streaming of high-bandwidth videos will be feasible under PCI-e 16x. Since WebVR is based on commodity browser technology, and does provide direct input and output translation, the generated active textures allows bi-directional data exchange between the WebVR enabled application and external web contents. This allows multiple users to form collaborative workspaces via existing web services, without the need to develop a system specific protocol. In other words, arbitrary applications that provide access to the WebVR plugin, can communicate with each other.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation under Grant Number EIA-0203528 and the California Institute for Telecommunications and Information Technology (Cal-(IT $)^2$). The above support is greatly appreciated.

REFERENCES

1. D. Hamar, "http://www.hamar.sk/sphere/, as of 10/25/2004."
2. Microsoft, "The TechTask Gallery," tech. rep., Microsoft Research, http://research.microsoft.com/adapt/TaskGallery, April 2004.
3. Sun, "http://www.sun.com/software/looking_glass/, as of 10/25/2004."
4. 3dwm, "http://www.3dwm.org, as of 10/25/2003."
5. O. Chapuis, "http://insitu.lri.fr/~chapuis/metisse/index.html, as of 10/25/2004."
6. D. Snowdon, "http://www.crg.cs.nott.ac.uk/~dns/vr/www3d/webnet96-final.html, as of 10/25/2004."
7. B. Wasson, "http://desk3d.sourceforge.net, as of 10/25/2004."
8. Apple, "http://www.apple.com/macosx/features/fastuserswitching/, as of 12/20/2004."
9. M. Hascoet, "Interaction and visualization supporting web browsing patterns," in *Proceedings of the Fifth International Conference on Information Visualisation (IV'01)*, pp. 413–418, IEEE Computer Society, July 2001.
10. I. G. Angus and H. A. Sowizral, "Vrmosaic: web access from within a virtual environment," *IEEE Computer Graphics and Applications* **16**, pp. 6–10, May 1996.
11. D. A. Bowman, E. Kruijff, J. Joseph J. LaViola, and I. Poupyrev, "An introduction to 3-d user interface design," *PRESENCE* **10**, pp. 96–108, February 2001.
12. O. Hagsand, "Interactive multiuser VEs in the DIVE system," *IEEE MultiMedia* **3**, pp. 30–39, Spring 1996.
13. K. G. and P. S., "A cookbook for using the model view controller user interface paradigm in smaltalk 80," *Journal of Object Oriented Programming* **1**(3), pp. 26–49, 1988.