Infraestructura de Realidad Virtual Multiplataforma

Daniel Mejia
Universidad de los Andes, Ingeniería de Sistemas y Computación,
Bogotá, Colombia
dan-meji@uniandes.edu.co

and

Pablo A. Figueroa
Universidad de los Andes, Ingeniería de Sistemas y Computación,
Bogotá, Colombia
pfiguero@uniandes.edu.co

and

J. T. Hernández
Universidad de los Andes, Ingeniería de Sistemas y Computación,
Bogotá, Colombia
jhernand@uniandes.edu.co

Abstract

We present a software infrastructure for the development of multiplatform, virtual reality applications. We use open–source toolkits and the previously published InTml architectural framework in order to provide an environment where developers can modify in a predefined way devices, interaction techniques, and content quality. Our main contributions are the uniform execution environment for portable virtual reality applications over a multi–framework setup, the separation of responsibilities during development, and the analysis of important variation points in such type of applications. We show two simple applications as a proof of concept of this infrastructure

Keywords: Virtual Reality, Open–Source, InTml, Development Environments for VR, Portable VR Applications.

Resumen

Presentamos una infraestructura software para el desarrollo de aplicaciones de realidad virtual multiplataforma. Usando herramientas de distribución libre y la arquitectura de software InTml propuesta previamente, permitimos al desarrollador crear aplicaciones en las que hay una forma planeada de cómo variar los dispositivos de entrada–salida, las técnicas de interacción utilizadas, y la calidad del contenido. Las contribuciones más importantes de este trabajo son la definición de un ambiente de ejecución uniforme para aplicaciones de realidad virtual portables, la división de responsabilidades en el desarrollo y el análisis de los cambios importantes en aplicaciones de este tipo. Al final mostramos dos aplicaciones que hemos desarrollado bajo esta plataforma

Palabras claves: Realidad virtual, Software libre, InTml, Ambientes de desarrollo para RV, aplicaciones de RV portables

1 INTRODUCCIÓN

Una aplicación de realidad virtual (RV) permite a su usuario interactuar con información mediante novedosos dispositivos y técnicas de interacción, generalmente en un espacio tridimensional (3D). A pesar de un desarrollo de más de treinta años, son pocas las aplicaciones exitosas, principalmente por las expectativas tan altas que se tenían al principio y el consecuente desinterés. Sin embargo, aplicaciones en áreas como

petróleos, diseño de automóviles y entretenimiento han demostrado lo importante de esta tecnología y sus ventajas. Una aplicación de RV típica combina eventos de varios dispositivos de entrada no convencionales como trackers y cámaras, cambia el estado de un modelo con representación 3D y lo muestra mediante imágenes o sonido tridimensional. El desarrollo de este tipo de aplicaciones puede facilitarse actualmente con el uso de algún ambiente de desarrollo, bien sea comercial o académico. Sin embargo, dichos ambientes son aún complicados de usar, están dirigidos a cierto tipo de ambiente computacional o dispositivos de entrada salida y limitan el tipo de cambios en el software, necesarios para probar distintas opciones en un análisis de factibilidad de uso.

Este trabajo muestra nuestro análisis de las variaciones importantes en una aplicación de RV, y una infraestructura que soporta estas variaciones. El análisis de variaciones destaca qué elementos deben ser fácilmente cambiables en este tipo de aplicaciones, para asegurar que el desarrollador pueda probar diferentes alternativas de hardware y software y que la aplicación puede evolucionar organizadamente.

El artículo está dividido en las siguientes partes: descripción del trabajo previo, requerimientos, infraestructura, ejemplos y por último conclusiones y trabajo futuro.

2 TRABAJOS PREVIOS

Existen varios ambientes de desarrollo para aplicaciones de RV, con diversas capacidades y limitaciones. Muchos de ellos se valen de archivos de configuración, en los cuales se pueden dar valores a parámetros que pueden variar de una instalación a otra. Algunos cuentan con herramientas para editar dichos archivos [3], mientras que en otros debe usarse un simple editor de texto [14]. La tabla 1 describe las principales características de diversas alternativas.

Toolkit	Capacidades principales
MRToolkit [14]	Abstracción dispositivos Archivos con parámetros
CAVELib [18]	Abstracción dispositivos Distribución
Performer [13]	Grafo para la escena Capacidades computacionales
WTK [12]	Grafo para la escena Abstracción dispositivos Ambiente de desarrollo rápido
Alice [17]	Ambiente de desarrollo rápido
VRJuggler [3]	Abstracción dispositivos Editor de archivos de configuración Abstracción de librería gráfica Integración con otras herramientas
X3D [19]	Grafo para la escena Abstracción del ambiente de ejecución
VRPN [11]	Abstracción dispositivos Distribución
InTml [5]	Arquitectura de software para RV Integración con otras herramientas

Table 1: Características principales en algunas herramientas de desarrollo para aplicaciones de RV.

Las fortalezas más comunes en herramientas para desarrollo de RV están relacionadas con la abstracción en software de dispositivos de entrada y la organización de los elementos gráficos mediante un grafo. Sin embargo, una aplicación de RV completa consta de otros elementos, tales como técnicas de interacción, que no están directamente representados en elementos de los sistemas existentes [1]

3 VARIABILIDAD EN APLICACIONES DE RV

Nuestro laboratorio ha desarrollado varias aplicaciones de RV en el pasado, que han demostrado ciertos conceptos en el área, pero que también nos han mostrado limitaciones de la tecnología y de las herramientas

[1] Java3D [8] define el comportamiento dentro del mundo virtual como un elemento principal, pero no existen aún muchos ejemplos de dichos elementos y su integración con el resto del framework.

de base que hemos utilizado. Fruto de esta experiencia previa hemos identificado el siguiente conjunto de requerimientos, que deseamos satisfacer en un nuevo ambiente de desarrollo:

- El conjunto de dispositivos de entrada y de salida de una aplicación debe ser configurable. Debe ser posible utilizar dispositivos remotos, simular un dispositivo mediante otros, o cambiar totalmente la interfaz hardware de una aplicación.
- Se debe permitir el cambio de técnicas de interacción, como un mecanismo para probar diversas alternativas en el diálogo con el usuario y como forma de adaptar de manera óptima nuevos dispositivos de interacción.
- Debe poderse elegir la calidad de las gráficas mostradas dependiendo de las capacidades computacionales de la máquina donde corra la aplicación.
- Se debe poder contar con una librería de acceso a diversos dispositivos de entrada salida, diversos tipos de contenido y diversas técnicas de interacción.

La infraestructura que planteamos a continuación trata de satisfacer dichos requerimientos, mediante la integración de herramientas previamente nombradas.

4 INFRAESTRUCTURA

Nuestra infraestructura está compuesta del conjunto de dispositivos disponibles en nuestro laboratorio de informática gráfica, la plataforma de software disponible en éstos equipos y los procedimientos diseñados para el desarrollo de aplicaciones en dicha plataforma. A continuación se analizan estos elementos en detalle.

4.1 Hardware

Actualmente el laboratorio cuenta con diversos dispositivos que combinados en una aplicación pueden generar varios ambientes de RV. Estos dispositivos varían desde el soporte más básico en un PC convencional, hasta los necesarios dentro de un ambiente de proyección.

Un primer ambiente de RV básico es logrado a través de un PC convencional, en donde la interacción se realiza con el mouse y el teclado, siendo útil para aplicaciones de visualización y rendering, como por ejemplo procesamiento de imágenes.

El segundo ambiente es un "fish tank desktop" [2] con visualización estéreo, logrado por medio de un PC con buenas capacidades gráficas (tarjeta Quadro FX 3000 [9] y 2GB en RAM), un monitor de alto refresco vertical (120Hz) y trackers Flock of Birds para la interacción [16]. Este ambiente ha sido utilizado para la reconstrucción de imágenes médicas. El tercer ambiente disponible se basa en un PC convencional que despliegua la imagen en un i-glasses HMD (Head Mounted Display) de baja resolución.

Estos ambientes pueden utilizar diversos dispositivos de interacción, disponibles en otra máquina de manera remota, como guantes (izquierdo y derecho) [15], joystick con retroalimentación de fuerza [4] y un gamepad [7].

Finalmente el laboratorio se encuentra en este momento en proceso de adquisición de un ambiente proyectivo que le permita generar un sistema CAVE, para lograr así una inmersión completa del usuario en mundos generados.

4.2 Software

Nuestro sistema tiene como núcleo de integración VrJuggler ya que este ofrece interacción con otro tipo de tecnologías como Performer o VTK [6] y adicionalmente permite interactuar con diversidad de dispositivos.

Sobre esta infraestructura de software corre un framework InTml [5] adaptado para este ambiente. Este framework consta de un sistema configurable, que permite acceder a los distintos recursos con archivos de configuración, permitiendo así que las aplicaciones puedan cambiar el ambiente de RV de acuerdo a las necesidades. Para el desarrollador que usa este framework, una aplicación consiste en un conjunto de filtros (componentes) que representan dispositivos, técnicas de interacción y contenido; y que pueden conectarse entre sí en el desarrollo de una aplicación.

Adicionalmente a esto la infraestructura esta montado sobre una red TCP/IP para poder independizar los dispositivos del ambiente de desarrollo. Los dispositivos se encuentran configurados en un arquitectura cliente - servidor para lo cual utilizamos VRPN como servidor de dispositivos. Mediante esta librería es posible acceder a los trackers, joystick, gamepad y demás dispositivos desde cualquier máquina de nuestro laboratorio.

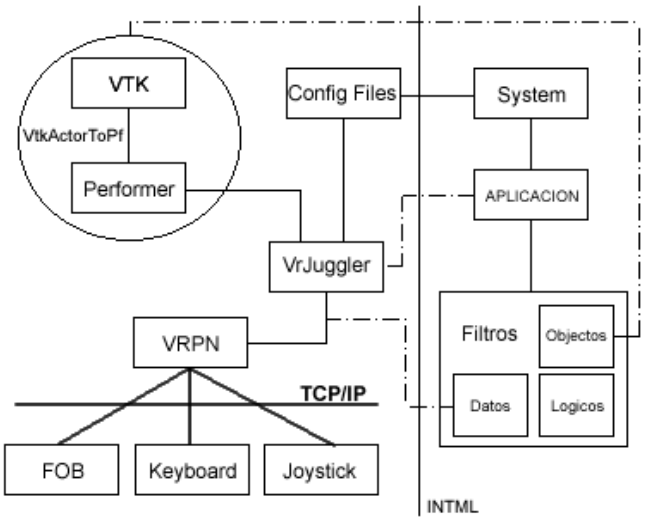

Figure 1: Descripción de la arquitectura de software

La integración de VRPN, VRJuggler e InTml crea una estructura particular de software, debido a que se debe modularizar la aplicación para correr sobre un sistema InTml. Estos se hace con la creación de filtros que nos permiten interactuar con todas las tecnologías. El diseño de toda la arquitectura de software utilizada se puede ver en el grafica anterior(imágen. 1), donde se muestra la división entre la plataforma e Intml, así como el acceso a cada una de las estas, representada por las lineas punteadas que salen de los filtros.

Estos filtros necesitan obtener datos de los dispositivos y acceder a objetos Performer. Para la obtención de datos aprovechamos la facilidad de VrJuggler para interactuar con VRPN por medio de sus archivos de configuración; y para acceder a los objetos lo que se hace es obtener un nodo Performer que contenga objetos y transformaciones. En el caso de un objeto VTK la transformación a un objeto Performer se realiza por medio de VTKActorToPf [10].

El desarrollo actual se centra en dos partes, la primera construir filtros InTml y la segunda usar estos filtros para desarrollar nuevas aplicaciones. Una vez se tiene los filtros, la creación de la aplicación se reduce a la unión de los mismos, simplificando así el desarrollo, ya que crear la aplicación se reduce a unir los filtros que se han desarrollado sin profundizar en el detalle técnico de cada uno de ellos. La creación es sencilla: los dispositivo se conectan a las técnicas de interacción y éstos a su vez se conectan con los objetos para modificarlos. Más adelante se mostrarán ejemplos de aplicaciones que corren bajo este esquema, donde se ilustrará claramente la unión de los filtros.

Por último, cabe destacar la división de trabajo que este ambiente de desarrollo permite: Hay un trabajo inicial de los desarrolladores para poner a punto un conjunto de filtros en InTml, los cuales pueden ser reutilizados más adelante en otras aplicaciones. De esta manera, el desarrollador final no tiene que entrar en todos los detalles intrínsicos en una aplicacion de RV, ya que éstos se encapsulan dentro de los filtros.

4.3 Ambiente Integrador

Así como la creación de la aplicación es sencilla, también se desea que esta sea flexible en su configuración de acuerdo a las necesidades del usuario final. Actualmente se han seleccionado ciertas variables de configuración que se consideran relevantes para el usuario y estas se pueden modificar por medio de parámetros en línea de comandos.

Se esta trabajando para que la sistema soporte no solo líneas de comando sino archivos de configuración ant [1], por medio de los cuales el usuario puede modificar las variables relevantes de una manera más estructurada. Los parámetros que actualmente consideramos son:

- Simulado (true o false). Permite cambiar los dispositivos que la aplicación usa por un ambiente simulado en el desktop. Se usa para pruebas principalmente
- 3d (true o false). Cambia la visualización de la aplicación a modo 3D o 2D. Util principalmente en el ambiente "fish tank".

- Calidad imagen (baja, media, alta). Permite variar la calidad de la producción de la imagen, para acelerar su despliegue [2].

4.4 Procedimientos

El primer proceso importante en nuestro laboratorio es el proceso de calibración de dispositivos, especialmente para los trackers. Este proceso se desarrolla antes de la ejecución de una aplicación y periódicamente para corregir cambios en el ambiente. En el caso de los trackers, se construyo una mesa con marcas de calibración cada 10cm, y que puede variar de altura, para tomar datos del tracker en una grilla 3D de 70cm de lado. Estos datos se pasan a un archivo de configuración que es leido por VRJuggler para calibrar los datos que arroja el dispositivo [3]

A un nivel de software se puede modificar los dispositivos de entrada y salida los cuales están representados como filtros. Hacer un cambio de dispositivos es hacer un cambio de filtros, buscar el filtro del dispositivo que se quiere, utilizarlo y conectarlo de una manera adecuada a los demás filtros en la aplicación. Al igual que con los dispositivos hacer un cambio en las técnicas de interacción se logra con un cambio de filtro. Entre mayor sea la cantidad de filtros desarrollados es mayor la cantidad de elementos disponibles para el desarrollo de aplicaciones.

Como ya vimos, cambios de rendimiento y opciones de alto nivel son seleccionadas mediante parámetros en la línea de comandos, dentro de los que se encuentra la calidad de la imagen y la visualización en estéreo, logrando así satisfacer los requerimientos de variabilidad planteados. Los parámetros a alto nivel se reflejan en el tratamiento interno de los filtros que representan objetos gráficos. En la implementación actual de filtros que representan objetos se maneja de dos maneras completamente distintas: en Performer se mantiene un estándar de nombres y en VTK se mantiene un parámetro dentro del filtro que se crea al empezar la aplicación.

El proceso de configuración del hardware es un proceso de creación de archivos en las diferentes tecnologías, principalmente en VRPN, donde se registra el dispositivo, dándole las propiedades necesarias. Posteriormente se crean los archivos VrJuggler necesarios para leer los dispositivos desde VRPN, dandoles un nombre significativo que les permita ser registrados en el sistema. Por último se configura un filtro en InTml, que representa el dispositivo y con esto crear los archivos necesarios para cada uno de los ambientes con los que se quiere trabajar.

Para crear un nuevo filtro se debe crear una clase en C++, que hereda de una clase genérica, de acuerdo a lo que se desea (un objeto, un dispositivo o un filtro). La nueva clase representa el comportamiento esperado del nuevo filtro, junto con las salidas y entradas esperadas.

Por último, consideramos un procedimiento para la ejecución de la aplicación. Actualmente se corre como una aplicación convencional que recibe los parámetros por línea de comandos, estos modifican la aplicación de acuerdo a sus valores. La idea final es correr la aplicación por medio de ant, lo que dividiría el proceso en dos, el primero un pequeño programa que lee parámetros en línea y cambia el archivo de propiedades, después se correría ant con el archivo creado.

5 APLICACIONES DE EJEMPLO

5.1 Un Visualizador Genérico

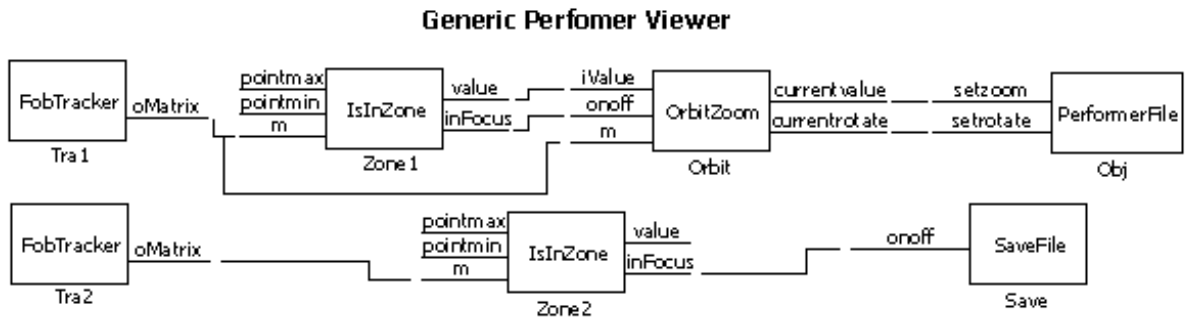

Figure 2: Visualizador Genérico

[2] Actualmente implementado para objetos gráficos de VTK, y en desarrollo para objetos de Performer.
[3] Estamos trabajando en un proceso que utiliza únicamente VRPN, para utilizar los datos de calibración a un nivel más bajo.

El objetivo de esta aplicación es dar las opciones típicas de un visualizador (zoom y rotación), para cualquier tipo de objeto Performer, la gráfica anterior (imágen. 2) muestra el diseño de la aplicación basándose en filtros, a continuación se describen cada uno de estos.

- Fobtracker: representa un tracker Flock of Birds, el cual retorna la matriz de transformación del punto, representada como un matrix44f de gmtl (librería de conceptos matemáticos, parte de VRJuggler).
- IsInZone: filtro de interacción que calcula si el punto que llega al puerto m, se encuentra dentro de la región delimitada entre pointmin y pointmax, retornando un valor verdadero (inFocus) y la correspondiente posición normalizada entre 0 y 1 en el espacio (value).
- OrbitZoom: filtro de interacción que calcula el zoom y la rotación . El valor de la translación con base en el valor recibido en iValue y la rotación con base en la matriz, estos valores son calculados si se encuentra activado el filtro (onoff) y envia los correspondientes valores a los salidas currentvalue y currentrotate.
- PerformerFile: Carga un archivo de un modelo Performer y crea el nodo correspondiente de translación y rotación, adicionalmente ofrece cambiar estos valores por medio de setrotate y setzoom.
- SaveFile: Al estar activo guarda el estado de las variables del usuario en un archivo, para poder recuperar posteriormente el mismo estado.

La aplicación une estos filtros, y obtiene el visualizador, donde se interactúa con un tracker, el primero, para zoom - rotate y el segundo para salvar el estado. Desde el punto de vista de ejecución, un usuario debe correr el programa mediante los parámetros previamente nombrados, mas el directorio donde se encuentran los archivos del objeto Performer.

5.2 Un Visualizador de Imágenes de Scanner

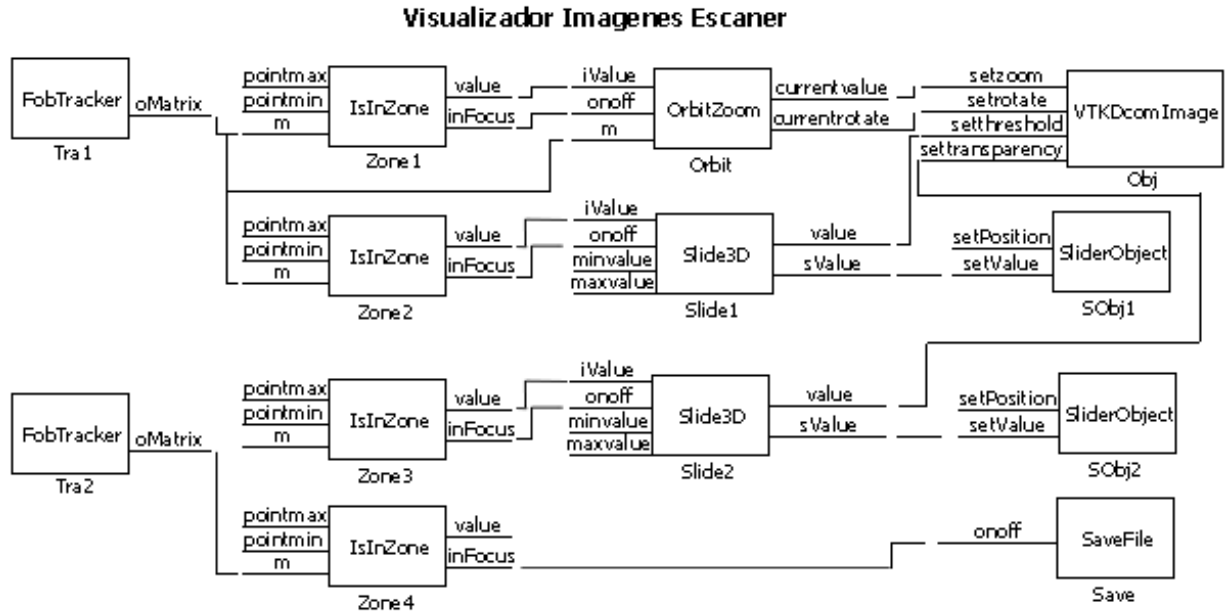

Figure 3: Visualizador de imgenes Escaner

Esta aplicación es menos genérica que la anterior, pero permite más opciones, ya que no solo es un visualizador sino que permite modificar los parámetros de reconstrucción. Esta aplicación utiliza muchos de los filtros de la anterior aplicación (imágen. 3) y otros nuevos que se especificarán a continuación.

- Slide3D: Se encarga de la lógica de un slider, calculando su posición y enviándola de dos maneras, la primera constantemente y la segunda al salir del slider (value, sValue).
- SliderObject: Es la forma de un slider generado en VTK (vtkCubeSource), y transformado a Performer por vtkActorToPf.

- VTKDComImage: Este filtro reconstruye dos superficies de acuerdo a dos umbrales establecidos, una de estas es fija y la otra con posibilidad de cambiar su umbral de reconstrucción y su transparencia. Toda la reconstrucción se hace por medio de VTK y su respectiva transformación en vtkActorToPf.

6 CONCLUSIONES Y TRABAJO FUTURO

Con toda esta infraestructura se logra obtener un banco de trabajo muy completo, que permite el desarrollo de aplicaciones de RV muy fácilmente, ya que se centra en el diseño de la aplicación y no en los aspectos técnicos que hay por debajo (división de responsabilidades) y además permite que la reutilización de las aplicaciones de RV sea confiable y efectiva.

La facilidad en los cambios de dispositivos, técnicas de interacción y calidad de los modelos permite no solo que una aplicación de RV, sea transportable a cualquiera de nuestros ambientes de RV, sino que además le da la posibilidad al desarrollador de probar diferentes configuraciones en el momento del desarrollo.

El sistema montado es efectivo, pero al ser basado en los filtros son muy pocas las aplicaciones que se pueden lograr, dado la poca cantidad de ellos actualmente. Pensamos desarrollar nuevos filtros con técnicas de interacción, dispositivos, y objetos, que nos permitan contar con una librería más completa para el desarrollo de aplicaciones.

Estamos trabajando también en el desarrollo de un "browser de InTml", aplicación que nos permita leer el archivo fuente de InTml (escrito en XML) y cargar dinámicamente el código de los filtros. Esta utilidad ya existia en un prototipo anterior en Java.

Por último, deseamos contar con una aplicación grafica que permita crear una nueva aplicación InTml en una forma más amigable para el usuario final, por medio de un lenguaje visual de programación.

References

[1] Apache. The apache ant project. http://ant.apache.org/, 2004.

[2] Kevin W. Arthur, Kellogg S. Booth, and Colin Ware. Evaluating 3d task performance for fish tank virtual worlds. *ACM Trans. Inf. Syst.*, 11(3):239–265, 1993.

[3] Allen Bierbaum, Christopher Just, Patrick Hartling, Kevin Meinert, Albert Baker, and Carolina Cruz-Neira. VR Juggler: A Virtual Platform for Virtual Reality Application Development. In *Proceedings of IEEE Virtual Reality*, pages 89–96, 2001.

[4] Microsoft Co. Sidewinder force feedback. http://www.microsoft.com/hardware/sidewinder/FFB2.asp, 2003.

[5] Pablo Figueroa, Mark Green, and H. James Hoover. InTml: A Description Language for VR Applications. In *Web3D 2002 Symposium Proceedings*, pages 53–58, 2002.

[6] Kitware. The visualization toolkit. http://public.kitware.com/VTK/, 2004.

[7] Logitech. Logitech dual action gamepad. http://www.logitech.com/index.cfm/products/details/US/EN,CRID=11,C
2004.

[8] Sun Microsystems. Java 3D Home Page. http://java.sun.com/products/ java-media/3D/index.html, 1997.

[9] NVIDIA. Nvidia quadro fx. http://www.nvidia.com/page/quadrofx_family.html, 2004.

[10] Paul Rajlich. vtkactortopf. http://brighton.ncsa.uiuc.edu/ prajlich/vtkActorToPF/, 2004.

[11] ll Russell M. Taylor, Thomas C. Hudson, Adam Seeger, Hans Weber, Jeffrey Juliano, and Aron T. Helser. VRPN: A device-independent, network-transparent VR peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 55–61. ACM Press, 2001.

[12] Sense8. Virtual reality development tools. The sense8 product line. http://www.sense8.com/products/index.html, 2000.

[13] SGI. Iris performer home page. http://www.sgi.com/software/performer, 2003.

[14] Chris Shaw, Jiandong Liang, Mark Green, and Yunqi Sun. The decoupled simulation model for virtual reality systems. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 321–328. ACM Press, 1992.

[15] 5DT Technologies. 5dt data glove 5. http://www.5dt.com/products/pdataglove5.html, 2002.

[16] Ascension Technologies. Flock of birds. http://www.ascension-tech.com/products/flockofbirds.php, 2004.

[17] Carnegie Mellon University and University of Virginia. Alice: Easy interactive 3D graphics. http://www.alice.org, 1999.

[18] VRCO. Cavelib library. http://www.vrco.com/products/cavelib/ cavelib.html, 2003.

[19] Web3D Consortium. Extensible 3D (X3D^{TM}) Graphics. Home Page. http://www.web3d.org/x3d.html, 2003.