

A comparison of virtual environments

Andreas Engberg, Howard Durand, Gilroy Menezes, Brent Lehman
at-on-ucsb@create.ucsb.edu

Department of Music, University of California, Santa Barbara, USA

I. BACKGROUND

Virtual Environments (VE) is a technology that has been around for several years, since the earliest flight-simulators in the early sixties until today's complete simulations of war and 3D game-techniques. The major difference between the earlier and the VE-applications of today, is that nowadays simulation takes place in real-time and allows the user to participate in the virtual world without any specialized systems or hardware. There exist many tools and devices that can increase realism in the virtual world. In this report, we try to find an off-the-shelf VE-system that fulfills, at least partially, the requirements set by the ATON project [1]. Two comparisons were done to find the most suitable system. The first comparison simply tests to see whether or not a system has some of the features we require. In the second test, we do more 'in-depth' testing to see the capabilities of the system along with some analysis of how easy the system is to use.

Note: Most of the previous work at CREATE has been aimed at the DIVE [16] system [14]. This means that the basics of DIVE are well-known to us but we wanted to test it on the same basis as the other systems.

II. INITIAL FINDINGS

The goal with this first test is to find only a couple of VE-systems that have all the basic features that ATON needs. These features are:

- Real-time rendering (30Hz)
- Multi-platform (different hardware systems)
- Support for different file-formats (at least VRML)
- Multiple users
- Usable API or open source
- Support for I/O devices

First of all, the Internet was searched to find different VE-systems. Since the ATON-project requires a large variety of features, some of the programs could easily be rejected. We also tried to avoid systems that had already been tested and described by other people [3]. All programs in following sections are VE-systems that at least could work as real-time renderers.

A. EON Studio

EON Studio [6] is a complete framework for creating virtual environments. The price is \$3,795 for the educational version. Since EON is built on OpenGL and claims to have the best rendering in the market, it seems to fulfill our requirements of the real-time rendering. EON seems to be built to handle large and small scale projects. Their newest product is an EON server that allows 3D objects to

be stored in a central location and distributed as needed. Several file formats can be read by EON; VRML 2.0, 3ds, dxf just to mention a few. Unfortunately, the EON Studio supports only Windows NT, only one user at a time and no open source. EON supports lots of hardware-devices but a definitive list could not be found.

Summary

EON is worth further investigation. Specially since it scales well and the promise about the best available rendering. The major disadvantages are the lack of non-Windows support, the lack of support for multiple users and the high price tag.

B. Vega

For a \$2000 educational price Vega [12] offers a scalable, real-time simulation environment. Vega is built to handle huge scenes, on the scale of cities. Versions for Irix and Windows NT are supported and multi-processor versions exist. VRML and OpenFlite are the supported file formats on Windows and anything that can be read by the SGI Performer libraries is supported on Irix. Vega is not an open source program but a flexible API is included. There is documented native support for the Flock of Birds, Daraglove and many other hardware devices. On NT it supports VRML and openflite. It is unclear if or how Vega supports multiple users.

Summary

Vega certainly makes the first cut. The inclusion of SGI support is strong motivation. MultiGen-Paradigm's educational sales representative has been very helpful in setting up a 30-60 day evaluation for us for free.

C. Meme

Meme [9] is a mixture between an operating system, a development system and a programming language. It has a high performance rendering system and is said to be system independent. Meme is intended to be used for VR applications on the internet and has support for multiple users and distributed objects. Meme is free, open-source and has a support for different I/O units.

Summary

The first impression is that this program is perfect for ATON. Unfortunately even if Meme claims to be system independent, it is available for Windows only (9x-series, not NT). After some discussion with the author of Meme (Marc De Groot), we learned that an OpenGL-version is

on its way but will take a couple of weeks to be released.

D. NPSNET-V

VE has been an interesting solution for war simulations. NPSNET [13][5] falls under this category and is highly suited for large networks. The system is based on Java, which allows it to run on several different hardware platforms. The real advantage of NPSNET is the network. The behaviour of the network protocol depends on the load of the network and other environment changes. NPSNET has an API that will fulfill our needs and the source code is available. One problem though is the lack of hardware support for I/O devices.

Summary

NPSNET is not very interesting, mainly because it is focused on war simulations. The webpage does not contain much information about its capabilities, but since NPSNET has such nice network features, it can be a nice alternative if no system is more appropriate.

E. VRJuggler

VRJuggler [4][10] is an environment for system-independent VR applications. The system is available for several platforms (Irix, Linux, Win32 to mention a few) and the source code is free. VRJuggler uses available hardware for rendering (OpenGL, Performer) and has support for different input-devices. The support for multiple users is well documented and virtual worlds can easily be distributed. One major disadvantage is that VRJuggler does not have any built-in support for file-formats; only direct rendering in OpenGL/Performer is available.

Summary

VRJuggler is probably one of the better free VE systems. Their webpage is up-to-date and new features are constantly being added. The problem with open-source software is usually the lack of documentation, but this program is an exception. It has great support for hardware devices and network support. The only problem is that VRJuggler does not support 3-d objects (creating or loading)- not even a scene-graph. This means that everything that includes graphics must be written in raw OpenGL, which is nice but very time-consuming.

F. DIVE

DIVE is a VE that has been around since the early nineties and is available for free for non-commercial usage. The system runs on several platforms and is aimed to applications with multiple users which communicate or interact in real-time. DIVE has support for VRML and other file-formats and has some support for I/O devices. The source is free, but it is not available for the latest version, and the API has some limitations which make it somewhat difficult to create vehicles or other user-controlled objects within the world. DIVE supports live audio- and video-streaming along with other useful features like collision detection and

on-the-fly object manipulation.

Summary

DIVE has been around for a while which is both good and bad. The good thing is that this program is well known and it seems to work alright, while the bad part is the lack of documentation for the latest release. This is still a good system which definitely is worth further testing.

G. Massive

Massive [8] is part of an on-going research at the University of Nottingham [2]. It is designed as a library and runs on Windows and SGI. Its API is based on MERL's Spline [11] facility, which supports real-time rendering and is easy to scale. Massive has only support for up to ten users and the support for hardware is very limited. An interesting feature in Massive is the support for audio transmissions in real-time along with its support for avatars.

Summary

This looks like a good system; the only problem is to know its current status. It seems impossible to get a copy of the system and the maintainers have not updated the webpage for a while. It is not worth the effort to find out whether the project is still running or if it is completed and no follow-up has been announced.

H. VRUT

The VRUT [15] package is an environment designed for building and designing virtual worlds. VRUT can import VRML worlds and attach a behaviour to objects in order to make them interact with a user. The description of a world is created by a Python script or via an interpreter. This makes it easy to hide all the complexity of the world in different layers. VRUT uses OpenGL as rendering engine and has support for different hardware units such as HMDs and tracking devices. The latest version of VRUT is only available for Windows but an older version is still available for SGI. VRUT has some basic network utilities, just enough to communicate between computers.

Summary

After a brief introduction and demonstration of VRUT by the author (Andy Beall), it was an easy choice to further test this system.

I. Conclusion of initial findings

Although several VE-systems claim to have support for a lot of tools and features, most of these are not fully operational or do not even exist. The decision was made to test only four systems further, but it has been really difficult to come up with four possible candidates that would be worth testing.

We chose to test Vega, DIVE, VRUT and Meme. Unfortunately, Meme turned out to be really difficult to work with (probably lack of Windows NT support) so after we had considered the rest of the systems and their availability, the choice would be to either select EON or VRJuggler.

Since EON would cost almost \$4000 just to test, we decided to try VRJuggler.

III. IN-DEPTH TESTING

In order to test the different platforms, several features were identified as important to ATON. These features includes both crucial aswell as desired ones. Since all the tested programs already passed most of the attributes described in section II; only a few of those attributes are described in this section.

The crucial issues are :

- Easy to use:
 - API
 - Object manipulation during run-time
- Quality of documentation
- Controlling a vehicle (with different I/O devices)
- Loading VRML-files:
 - UCSD campus [7]
 - Vehicles
- Generating objects on-the-fly
- Tying two eyes together (for HMD stereo effect)

The items listed above are required by ATON while the items listed below would wnhance realism.

The desired features are:

- Collision detection
- Using a video-feed as a texture (for live-conferancing)

The in-depth tests were taken place on different computers (SGI, SUN and PC) with different operating systems (Irix, Solaris and Windows). This means that some issues will be compared a bit different on each machine but the idea is to rate the system as a whole.

A. VRUT

The tested version of VRUT was for Windows only. No problem occured during installation and setting the system up. The documentation contained tutorials of both VRUT and Python, a reference to all functions supported by VRUT, and several different sample worlds. In addition to this, the author of VRUT (Andy Beall) has offered to help as much as he can. The machine on which the tests were taken place did not have any hardware support for rendering (VRUT uses OpenGL). Despite these conditions, the rendering went quite smoothly providing a constant framerate at 10 fps when using the UCSD world.

It is really intuitive to create objects, load VRML files and manipulate objects, and all the basic manoeuvres are explained in the tutorials or in the examples. VRUT has support for different I/O units, which makes it easy to control a vehicle with a glove or render the world in stereo. The only problem is that the mouse is already assigned to move the observer around in the world. VRUT does not support collision detection nor video feed but there exists a simple function that can display a batch of pictures in a sequence on an object or surface. These pictures are basically just textures that will be altered over time.

The network support is really limited in VRUT but it allows simple communication to take place between computers. The major problem is that all objects are only stored

locally, e.g. if Computer A creates an object it will not be visible to Computer B.

B. Vega

A SGI Octane was used to test a 30 day evaluation version of Vega. The interface of Vega is fairly complicated but this is obviously the result of a lot of flexibility. There are many parameters that can be manipulated and they are all set to intelligent defaults. The documentation and API are very good. There is a good combination of documentation of all the features mixed in with examples of how to use them. Loading the UCSD file was a bit difficult at first but once it was loaded it ran very smoothly. Vega support both VRML 1.0 and 2.0, both seems to work fine. The support for collision detection seems to work fine. It was really easy to create an avatar that drove along the surface of the world. It is very flexible to create and control a vehicle. The API lets the user define keyboard and mouse controllers and it includes built-in support for the Flock of Birds and many other common hardware devices. There is plenty of support for making HMDs work, including all the lens calibration facilities. Multiple users can be created on the same node (machine) but it is unclear whether these can be located on different machines.

Overall, this is an impressive package. We have experienced a little bit of difficulty figuring out how to use the features, but once they were figured out, they acted predictably and as documented. The documentation is thorough and so far accurate. Along with the Vega API the functionality of the Iris Performer Libraries is also obtained. The lack of obvious support for multiple users and video feeds are the only two drawbacks (other than the price).

C. DIVE

DIVE offers a considerable function set and appears to cover everything we might need. It is fairly easy to install—just unzip it and set a couple of environmental parameters. DIVE was tested on a Sun without any specialized hardware. It took several seconds to load the UCSD file. Although the file resulted in many warning messages, no apparent rendering errors occured. The rendering varied from five to fifteen frames per second. It is easy to control vehicles with DIVE via the Tcl interface. Tcl has support for input devices such as keyboard, mouse and Flock of Birds. DIVE has no problem running two different users on different platforms, and it provides a couple of helpful functions to make autonomous agents. On-the-fly objects can be created either statically by using DIVE scripts with nested Tcl code, or dynamically by DIVE scripts or Tcl scripts. It is possible to alter attributes of objects. In order to use a HMD, more in-depth coding to link DIVE either with the drivers or the software layer just above the drivers is necesarry e also need to link rendering processes either with each other or with DIVE in such a way that vehicle input results in coupled movements of both eyes.

The documentation is fairly well written, although it can be hard to follow. The Tcl and C commands are well-

covered but the syntax in DIVE is sometimes different from what is given in the documentation and the descriptive text is sometimes confusing. Interfacing with DIVE via C is barely covered which will be a problem when writing vehicles or talking to peripherals. Even if DIVE is supposed to be open-source, no source code is available, which would have come in handy when since the documentation is not up-to-date. A SDK is available for the Windows NT but it is provided without documentation.

Unfortunately, DIVE also gives the impression of being a bit buggy. Perhaps these bugs are actually features; the documentation is bad enough that it is hard to tell in some cases. If the source code were available, the documentation would suffice. Considering that this is freeware, the source code does not seem to be too much to ask for.

D. VRJuggler

Installing VRJuggler is really easy; simply extracting files from an archive. Setting up the system is much harder, at least in a Windows environment. VRJuggler runs under Cygwin and it has to be installed, which is not too hard even if it takes some time to get it right. The problem is that all examples requires Microsoft Visual C++ (VC++) to run and all the makefiles assumes that VC++ is installed and linked into Cygwin. This is frustrating since there exist no documentation about how this linking can be done. How come that VRJuggler is free but requires VC++ to run, at least to run all the examples? Once the system is set up, it works fine. There were no problems to compile and test the examples.

Since no support for importing files exist, the user have to write all the rendering at a low-level (OpenGL or Performer). This is fine if you got a lot of time, but since we already have created a world in VRML it would have been nice to use it without converting it to pure OpenGL/Performer-code.

The framerate is decent, although it is really difficult to say because the examples only contains of a few objects. VRJuggler has built in support for different I/O-devices and can simulate generic units such as wands, gloves etc. We were unable to test our hardware devices by some unknown reason but the simulation worked fine, even if it was difficult to control sometimes. VRJuggler will start a new window for each I/O unit that will be used in the session. There exist several different documents about VRJuggler describing everything from its internal architecture to programming guides and references. The source code is also available along with some well described and simple examples. The webpage is well maintained and it really feels like this is a system that is used by others.

VRJuggler is, unfortunately, a bit to complicated and it takes a lot of effort to get used to. The fact that it has no configuration tools makes it a bit difficult to work with, although this can really be useful when writing applications that requires lots of control and optimizations. VRJuggler is a good system but since it takes long time to learn and to program in, it will probably not be suitable for the ATON project.

IV. CONCLUSION

Most of the VE systems that were tested are working well but only in certain areas. It has become more and more clear to us that it is impossible to find a 'perfect' VE system. One must identify and prioritize features that are crucial to the project and try to minimize the penalty of the trade-offs with that particular system. In our case, we wanted a system that was easy to use, have real-time rendering, support for multiple users, good API to navigate vehicles in a world and was able to read VRML-files. Vega or VRUT are those systems that we have found suitable for the ATON project but they are far from perfect. Vega easy to configure and has a nice API along with built-in support for different I/O units but the trade-offs are the prize (\$3000 per copy) and that Vega may not support multiple users located on different machines. VRUT is easy to use, has a well working and simple API, support for I/O devices, free and since the creator of VRUT is working at UCSB, we get good support and lots of information about how VRUT can be used. The trade-offs are that the latest version of VRUT is only available for Windows (older versions for SGIs exist) and that only very native network is supported.

The next step in our research will be to further test Vega and VRUT and try to decide which one to choose. We will also set up a lab that we can use for testing of different I/O-devices (currently a Flock of Birds, a 5DT glove and a Kaiser HMD). The team that are setting up the lab will also write drivers for the different devices so that, regardless of VE system, a middleware can be added and connect the VE system and the I/O-device.

REFERENCES

- [1] Aton project. Technical report, CREATE. Department of Music. University of California. Santa Barbara, 2000. www.create.ucsb.edu/aton.
- [2] Steve Benford, Dave Snowdon, and Chris Greenhalgh. Hivek : the hive project. distributed vr runtime kernel. Technical report, School of Computer Science and Information Technology. University of Nottingham, February 1999. www.crg.cs.nott.ac.uk/research/systems/MASSIVE-3/docs/hivek-intro.pdf.
- [3] Allen Bierbaum and Christopher Just. Software tools for virtual reality application development. *SIGGRAPH98 Course 14- Applied Virtual Reality*, 1998.
- [4] Allen Douglas Bierbaum. Vr juggler: A virtual platform for virtual reality application development. Master's thesis, Iowa State University, 2000.
- [5] Micheal Capps, Don McGregor, Don Brutzman, and Michael Zyda. Npsnet-v: A new beginning for virtual environments. *IEEE Computer Graphics and Applications*, 2000.
- [6] EON Reality Inc. *EON Studio*, 2000. www.eonreality.com.
- [7] Stephen Travis Pope et al. Create aton world-building tasks. Technical report, CREATE. Department of Music. University of California. Santa Barbara, November 2000. www.create.ucsb.edu/aton/0010/UCSD.Model.html.
- [8] Chris Greenhalgh, Dave Snowdon, Dave Roberts, Milena Radenkovic, and Jim Purbrick. Massive-3. Technical report, School of Computer Science and Information Technology. University of Nottingham, 1999. www.crg.cs.nott.ac.uk/research/systems/MASSIVE-3/.
- [9] Immersive Systems Inc. *Meme*, 2000. www.immersive.com.
- [10] The Juggler Development Team. *VR Juggler*, 2000. www.vrjuggler.org.
- [11] Mitsubishi Electric Research Laboratory. *Scalable Platform for*

- Large Interactive Networked Environments (SPLINE)*, 1997. www.merl.com/projects/spline/index.html.
- [12] MultiGen-Paradigm Inc. *Vega*, 2000. www.multigen.com.
- [13] NPSNET Research Group. *NPSNET-V*, 2000. www.npsnet.org.
- [14] Stephen Travis Pope, Ken Fields, Alexandre Kouznetsov, and Paul Mobbs. *Drive*. Technical report, CREATE. Department of Music. University of California. Santa Barbara, 2000. www.create.ucsb.edu/drive/phase2/report.html.
- [15] Research Center for Virtual Environments and Behavior. University of California. Santa Barbara. *VRUT Manual*, August 1999. www.recveb.ucsb.edu/www/index.html.
- [16] Swedish Institute of Computer Science. *DIVE*, 2000. www.sics.se/dive.