Conference and Exhibition of the European Association of Virtual and Augmented Reality (2014)
G. Zachmann, J. Perret, and A. Amditis (Editors)

PolyVR - A Virtual Reality Authoring System

Dipl.-Phys. Victor Häfner

Institute for Information Management in Engineering
Karlsruhe Institute of Technology

Abstract

Virtual Reality applications are very complex. 3D modelling and scene authoring is very time consuming and requires expertise in computer sciences. A further problem are the advanced Virtual Reality hardware setups, which have to be planned, configured and maintained. PolyVR is a Virtual Reality authoring system that allows the user to dynamically create immersive and interactive virtual worlds. The created content is abstracted from the hardware setup, allowing a flexible deployment even on complex distributed visualization setups. The hardware setup configuration is dynamic, it can be changed while the virtual world is running. This includes viewport management, tracking systems and other Virtual Reality technologies such as haptic devices. Scene authoring is realised through manipulation and configuration of the scenegraph nodes as well as through Python scripting for dynamic and interactive content. Python bindings provide access to all built-in features of PolyVR. It allows for a powerful and highly flexible way of interfacing with external libraries or other resources, and is a very intuitive language with a low learning curve.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism —Virtual Reality

1. Introduction

The importance of Virtual Reality (VR) is growing, not only in research institutes, but also in gaming, education and industry. Novel VR devices and technologies targeted at the mass market started to appear in 2006 with the Wii Remote [Lee08]. This means affordable hardware for gamers, students and hobby-developers, but also for schools, universities and small and medium enterprises (SME). The major hardware developments are in visualization devices like TVs, which have bigger screen diagonals and passive stereo, and interaction devices like the Wii Remote, Microsoft Kinect [Zha12] and leap motion [WBRF13] with many more having been announced. Another important novelty are the omnidirectional treadmills, such as the Omni [Omn] or the Virtualizer [Vir], that recently appeared on the market. They are likely to greatly impact navigation and interaction paradigms in applications.

Simultaneously to the hardware becoming more affordable, reliable and efficient, the market for new devices grows in gaming, research and industry. This calls for more mature software packages to ease the development of applications making use of these VR technologies such as tracking, novel interaction devices and distributed visualizations. There are new markets to be explored in the field of SMEs, due to lower investment costs for VR technologies. A limiting factor that cripples the adoption of VR in SMEs is the complexity of VR hardware setups. Not only do they have to be designed and configured, but also maintained, which results in significant costs. A further factor is the content management. Real world data is difficult to readjust for a real time and interactive applications. Real time oriented 3D

Figure 1: *Child immersed in a virtual world*

modelling and scene authoring is very time consuming and requires broad expertise in computer science with a particular focus on computer graphics. Gathering this know-how would require excessive time investments, while purchasing it as a service is both a costly and an unsustainable solution.

Fostering the development of VR technologies for enterprises is thus an important topic. The maintenance of VR hardware and equipment has to be possible without much training. The enterprise needs to have a good perspective for its VR investments. A solution has to be extensible and sustainable. This means simple workflows to integrate any new data, robust interfaces and intuitive VR systems with flat learning curves.

VR in education is also an important and growing field. New didactic approaches have to be developed for virtual worlds. These would open many possibilities to teach complex concepts in a more visual, interactive and intuitive ways, especially for STEM subjects (fig.1). A further, more challenging task than using VR for teaching purposes would be giving the students a hands-on experience of working on VR projects and developing VR applications themselves. The time available for this is limited and the more of it the students need to learn how to use the software tools themselves, the less they have to work on the actual projects. Furthermore, establishing continuity in the development cycles of the said VR projects, allows the students to work on their projects in the following semesters or for next generations of students to work upon the bases built by their predecessors. To achieve the continuity it is important to refactor the projects in between semesters. The drawback is that refactoring student code is very time consuming.

Using VR in research comes with some additional requirements for the VR system. Extensibility with complex interfaces is very important, e.g. when dealing with numerical simulations. The produced sets of data are often very large and complex, just as the reuse of real-time optimizations for visualization and interaction is very difficult yet equally important.

This paper describes how the VR authoring system PolyVR addresses the following needs:

- Flexible and efficient VR hardware management
- Novel VR authoring tools with low learning curves, suitable for educational purposes
- Mechanisms to foster sustainability and efficient reuse of solutions

The system is described in combination with examples from research, industry and student projects within the context of virtual engineering.

2. Related Works

To create virtual worlds one needs to prepare the content, add the game logic and deploy the software on advanced VR hardware setups. There are different approaches to building such applications. A low-level approach is to use C++ libraries such as game engines, scenegraphs and VR device libraries. Famous examples for this are the open source projects VRJuggler [BJH*01], OpenSceneGraph [BO04] and OpenSG [Rei02]. VRJuggler is a virtual platform library, which allows one to hide the details of the VR hardware setup, such as clustering, VR devices and tracking. It has to be combined with scenegraph libraries, like OpenSceneGraph or Ogre3D [Ogr], to develop a complete application. OpenSG is also an example of a scenegraph library, but with a focus on clustering and threading as well as numerous advanced features used for VR applications. OpenSG by itself is enough for developing VR applications, with only small libraries for communicating with VR devices having to be added. PolyVR is heavily based on OpenSG. These libraries are very performant and flexible. The drawback is that using them directly is both very time consuming and requires a significant degree of expertise in software programming and VR technologies. An individual lacking this know-how would have to make a significant time investment to achieve even the basic goals such as setting up a C++ project. For student projects, this would also mean that the supervisor would have to put in a significant amount of effort into refactoring the project, for example for the next students that have to work on it. This makes reuse difficult.

There are also higher level solutions. Among them are scene authoring tools with GUI and VR capabilities. BlenderCAVE [PQTK13] and MiddleVR [Mid] are extensions for the well-known 3D modeller Blender and the 3D engine Unity. Both VR extensions are quite recent and have a somewhat limited amount of features. Furthermore, the performance is not as good as that of VRJuggler or OpenSG. The complexity is also quite high, because the extensions are built upon other sophisticated tools and are not very well integrated. There are also dedicated tools for VR application development like COVISE [Wie95] and 3DVia Studio Pro [Stu]. They are quite advanced, but have some important drawbacks. COVISE is already comparatively old and the rendering is outdated. 3DVia Studio Pro is very expensive and leads to a vendor lock-in due to the the inability to port the code written for it. Both tools are not open and thus do not present sustainable solutions, at least for the academic environment.

3. Concept Overview

Past experiences with the deployment of the aforementioned VR tools in the areas of research, projects, industry projects and teaching led us to the conclusion that an effective VR solution would have to meet the following requirements:

- VR content has to be abstracted from the hardware setup it runs on.
- The hardware configuration, monitoring and maintenance has to be very time-efficient.

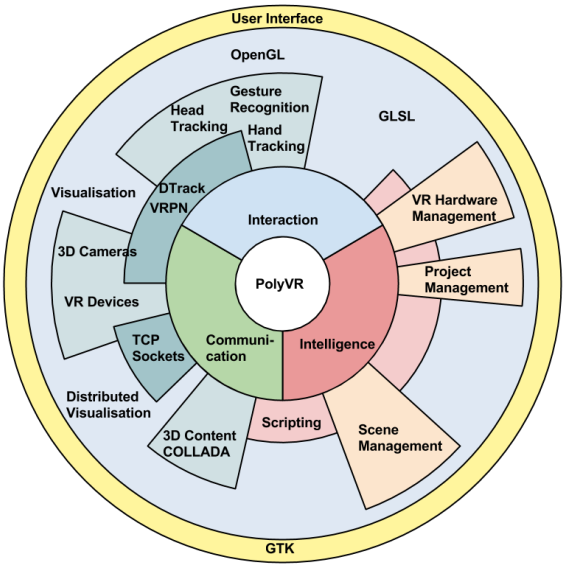

Figure 2: *PolyVR software architecture*

- The procedure for creating a new virtual world has to be quick and should not require any obscure project configuration on behalf of the user.
- The feature, logic and interaction implementation should be dynamic. No compilation and starting steps should be needed.

All of the above has been implemented in PolyVR. The software has a graphical user interface that allows the user to configure the hardware setup dynamically and does not require application restarts after each change. The VR content creation GUI is clearly separated from the hardware configuration. No project configuration needs to be performed by the person creating a virtual world as PolyVR comes with a wide range of functionality for immersive and interactive virtual worlds. The use of external libraries is possible through Python. PolyVR comes with an integrated Python interpreter and editor. These scripts allow to dynamically create and modify the world's logic, without the need for compiling and restarting the world for testing.

4. Solution and Software Architecture

An overview of the software architecture is depicted in fig.2.

4.1. Scenegraph

The backbone of any Virtual Reality authoring system is the scenegraph. Scenegraph libraries like OpenSceneGraph, Ogre3D and others are widely used in research and industry. They have different focuses depending from their origin. Most of them offer similar scenegraph functionality as well as other features, often used for handling content management and rRendering. PolyVR uses the OpenSG library. OpenSG focuses on clustering and threading, which makes it ideal for any Virtual Reality application in distributed visualization setups.

4.2. Scripting

Static content defines most of a virtual world. While it is computationally cheap, it is important to put a significant portion of projectâĂŹs resources into content creation. It is equally important to bring the world to life and to allow the user to interact with it. Dynamic content like animations and game logic, but also physics and other simulations, have to be developed and configured. A considerable amount of work results from fine tuning every parameter to get the desired results. Fig.3 shows a script that configures a complex mechanism from parameterized gear and thread primitives. An integral concept behind the creation of PolyVR is to simplify and accelerate this process significantly. Scripting is an important feature that allows us to dynamically create and configure logic, functionality within and interactions with the virtual world. Python is supported for accessing the scene content and integrating features, like animations, interaction and physics. GLSL, HTML and CSS are also supported. GLSL allows the user to write his own vertex, geometry and fragment shaders. HTML and CSS can be used to write web pages hosted by PolyVR. This is especially interesting for interacting with the virtual world from a mobile device or from a web browser. HTML and CSS are also used to design web sites and render them with WebKit to textures that can be placed in the virtual world itself. URI strings can be converted to QR code textures to allow an intuitive and simple way to connect to PolyVR.

4.3. Advanced Features

This section describes some of the more advanced built-in functionality of PolyVR. It can be accessed and configured using solely scripts written in Python. The physics engine Bullet [C*06] is well known for simulating collision detection, soft and rigid body dynamics. It is the basis for realistic behaviour of dynamic objects in a virtual world. An important concept is the use of paths and stroke objects that allow one to generate a mesh by extruding a profile along the paths. Paths are defined by 3D points that can be Bezier interpolated. They are also used for the camera flights or logistic simulation. PolyVR comes with a simple logistic simulation that can be used for virtual production lines. Another simulation in the engineering field is the mechanism simulation. With its help complex mechanisms build from gears, threads and chains can be brought to life. These are a part of the basic parametric primitives of PolyVR. CGal [FGK*98] has also been integrated in order to be able to construct even more complicated objects. CGal allows one to easily combine primitives using boolean operations. This is also called

constructive solid geometry and is a basic module of CAD modelling.

New features are added with each project. The integration method is clean and easily extensible. All of the features are written in C++ with Python bindings. This allows for a sustainable workflow, easy testing of functionality and thereforethus an efficient maintenance of old features.

4.4. Hardware Cconfiguration

A VR hardware setup has three types of components:, tracking systems, input devices and displays. PolyVR supports the DTrack [ART] and VRPN [TIHS*01] protocols for most VR interaction devices and tracking systems. OpenSG comes with the windows and viewports, which can be configured dynamically in PolyVR as seen on fig.4 and 6. The preferred way to allow mobile devices to control a PolyVR application is by accessing a website hosted by PolyVR. HTML, CSS and Javascript can be added to and edited as scripts within a project. Built-in examples are provided, such as a simple navigation or a keyboard for text entry in VR. A QR Code generator is integrated and allows to generate textures that can be placed anywhere in the virtual world. This is a comfortable way to encode URLs for the users, which allows them to easily connect to the application.

5. Applications

Applications developed with PolyVR have been deployed on various hardware setups. One advanced setup is for instance a CAVE environment with the dimensions of 5m x 2.6m x 2m and a combined total of 12 million pixels. The rendering cluster consists of seven nodes with Quadro 4600 cards. Tracking and interaction devices are from ART. Other setups consist of 3D TVs, with active or passive stereo, and HMDs. Interaction devices like the leap motion have been used with the VRPN protocol. Haptic devices have also been used with the API and devices from Haption [Hap].

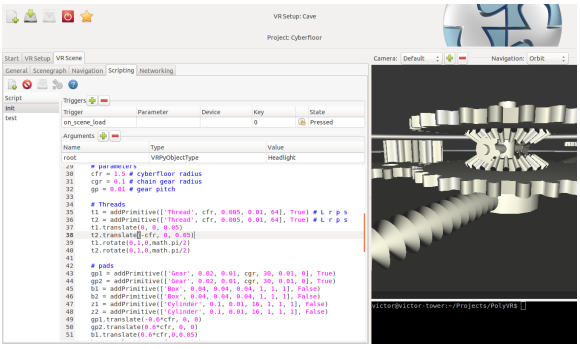

Figure 3: *Embedded Python editor with a script for a mechanism simulation*

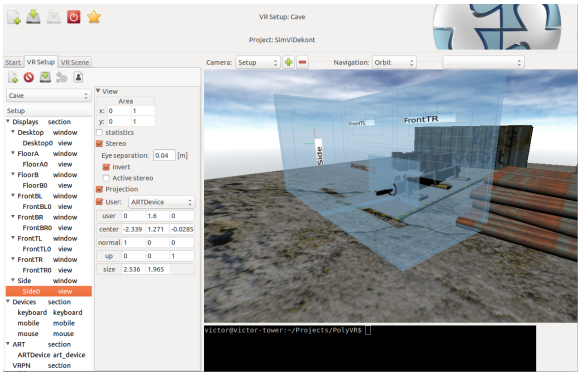

Figure 4: *Dynamic CAVE display configuration*

A project that makes use of the mechanics simulation is a virtual prototype of a cyber floor simulator (see fig.3). It is a device that allows natural walking without moving in space, similar to a treadmill but more flexible for two directions. The device essentially mimics a consistent floor, while keeping the user's center of mass in the middle of the device. What differentiates this prototype from other concepts is that the user is not sliding on a low-friction surface, as is the case with the Omni or the Virtualizer. A prototype will have to be constructed to validate the concept. Another project is a virtual factory (see fig. 6 and 5). Real CAD data of a production line is integrated with the logistic simulation, dynamic meta information and animations of the material flows.

PolyVR currently is used for a practical course [HHO13] that features two sustainable student projects, a driving simulator and the so-called energy experience lab. The driving simulator is a mixed reality setup (fig.7) with a real car interface, built into a Smart Fortwo. The vehicle is situated in front of a powerwall with ART tracking. The carâĂŹs interior has not been altered. The students have reverse engineered the CAN bus gaining access to most of the car inputs and indicators, such as the speedometer and the tachometer. The virtual world is a chunk-based generated world us-

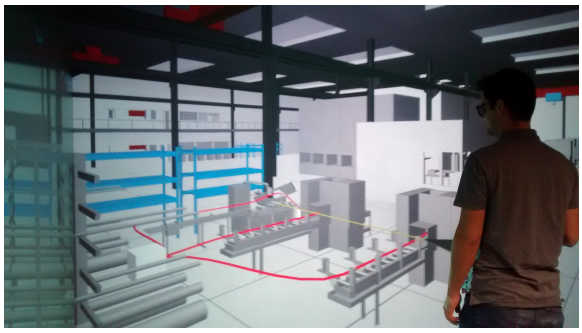

Figure 5: *Virtual factory with logistics simulation*

Figure 6: *Three sided CAVE*

Figure 7: *Student project driving simulator*

ing the OpenStreetMap system as seed data. This allows the user to drive through huge worlds that heavily correlate with the real one. The energy experience lab is a set of sensors and actuators for public buildings, accompanied by a 3D TV with a virtual representation of the building. The focus of the project is on data visualization, monitoring and control for energy efficiency in public buildings.

6. Future Works

PolyVR has reached a certain level of maturity with regard to basic scene authoring and VR features. A more advanced feature that will be implemented is an efficient multiplayer feature based on the OpenSG clustering capabilities. The goal is to enable the fusion of the content running on multiple instances of PolyVR from different locations and on different hardware setups. Another feature is using the GIT API as a version control system for the project. This will add the possibility to browse the project history, create and manage branches and thus greatly enhance the reusability and sustainability of the development work. Future long-term development will take place in the field of virtual engineering and be geared towards the vision of a comprehensive virtual factory modelling and simulation, including product development, production planning and monitoring.

References

[ART] Art. http://www.ar-tracking.com/. Accessed: 2014-08-28. 4

[BJH*01] BIERBAUM A., JUST C., HARTLING P., MEINERT K., BAKER A., CRUZ-NEIRA C.: Vr juggler: A virtual platform for virtual reality application development. In *Virtual Reality, 2001. Proceedings. IEEE* (2001), IEEE, pp. 89–96. 2

[BO04] BURNS D., OSFIELD R.: Open scene graph a: Introduction, b: Examples and applications. In *Virtual Reality Conference, IEEE* (2004), IEEE Computer Society, pp. 265–265. 2

[C*06] COUMANS E., ET AL.: Bullet physics library. *Open source: bulletphysics. org 4*, 6 (2006). 3

[FGK*98] FABRI A., GIEZEMAN G.-J., KETTNER L., SCHIRRA S., SCHÖNHERR S., ET AL.: On the design of cgal, the computational geometry algorithms library. 3

[Hap] HAPTION V.: 6d35-45 6-dof haptic interface. 4

[HHO13] HÄFNER P., HÄFNER V., OVTCHAROVA J.: Teaching methodology for virtual reality practical course in engineering education. *Procedia Computer Science 25* (2013), 251–260. 4

[Lee08] LEE J. C.: Hacking the nintendo wii remote. *Pervasive Computing, IEEE 7*, 3 (2008), 39–45. 1

[Mid] Middlevr. http://www.imin-vr.com/middlevr. Accessed: 2014-08-28. 2

[Ogr] Ogre3d. http://www.ogre3d.org. Accessed: 2014-08-28. 2

[Omn] Virtuix omni. http://www.virtuix.com/. Accessed: 2014-08-28. 1

[PQTK13] POIRIER-QUINOT D., TOURAINE D., KATZ B. F.: Blendercave: A multimodal scene graph editor for virtual reality. 2

[Rei02] REINERS D.: *OpenSG: A scene graph system for flexible and efficient realtime rendering for virtual and augmented reality applications*. PhD thesis, 2002. 2

[Stu] 3dvia studio pro. http://www.3ds.com/products-services/3dvia/3dvia-studio/. Accessed: 2014-08-28. 2

[TIHS*01] TAYLOR II R. M., HUDSON T. C., SEEGER A., WEBER H., JULIANO J., HELSER A. T.: Vrpn: a device-independent, network-transparent vr peripheral system. In *Proceedings of the ACM symposium on Virtual reality software and technology* (2001), ACM, pp. 55–61. 4

[Vir] Cyberith virtualizer. http://www.virtuix.com/. Accessed: 2014-08-28. 1

[WBRF13] WEICHERT F., BACHMANN D., RUDAK B., FISSELER D.: Analysis of the accuracy and robustness of the leap motion controller. *Sensors 13*, 5 (2013), 6380–6393. 1

[Wie95] WIERSE A.: Performance of the collaborative visualization environment (covise) visualization system under different conditions. In *IS&T/SPIE's Symposium on Electronic Imaging: Science & Technology* (1995), International Society for Optics and Photonics, pp. 218–229. 2

[Zha12] ZHANG Z.: Microsoft kinect sensor and its effect. *MultiMedia, IEEE 19*, 2 (2012), 4–10. 1