IPT & EGVE Workshop (2005)
R. Blach, E. Kjems (Editors)

Towards Better Quality in Virtual Environments

F. Mannuß and A. Hinkenjann

Computer Graphics Lab, Bonn-Rhein-Sieg University of Applied Sciences (BRSU), Germany

Abstract

This paper describes the work done at our Lab to improve visual and other quality of Virtual Environments. To be able to achieve better quality we built a new Virtual Environments framework called basho. basho *is a renderer independent VE framework. Although renderers are not limited to graphics renderers we first concentrated on improving visual quality. Independence is gained from designing* basho *to have a small kernel and several plug-ins.*

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

1. Introduction

Most VE frameworks today concentrate on offering quality user interfaces, collaboration support, device independence etc. Examples are AVANGO [Tra01] or Lightning [BLRS98]. Rarely, VE frameworks lay emphasis on enhancing visual quality by being renderer independent.

basho [HM04] is a rendering and scene description independent Virtual Environments framework currently under development at our lab. Rendering in this context is not bound to graphical rendering. Further examples are physics our sound rendering. The framework is kept small by using a kernel that loads all content, like renderers and scene descriptions at run time as plug-in components.

2. Better Quality

To be able to support more types of renderers, basho introduced its own layer of objects, called *virtual objects*. With these objects basho´s kernel stays independent from graphics (physics, audio, ...) APIs because the attributes needed by these APIs are not embedded into the virtual objects.

2.1. Virtual Objects as Attribute Containers

Objects of class virtualObject are the components of which a virtual scene is composed. Thinking of objects in the real world virtualObjects try to imitate them. Therefore a virtualObject is not simply a node in a scene graph. It stores all attributes that are needed to describe an object, like a table, in the virtual world.

virtualObjects by itself are empty. They only have a name and the ability to store attributes. All functionality is aggregated through attributes. Therefore, virtualObjects are containers for attributes.

All attributes that can be added to a virtualObject are stored in plug-ins that are loaded at run time. With attributes stored in plug-ins and the fact that all attributes encapsulate the underlying storage of the data, independence of the underlying data storage is achieved.

In order to stay independent of the loaded plug-in, attributes cannot be created through *new* or be directly accessed. They can be accessed by a set/get method pair only. Adding attributes to a virtualObject is done indirectly via the virtualObjectAttributeFactory. It is a *prototype factory* [GHJV95] that knows all loaded attributes. Grouping virtualObjects can be done using a virtualObjectContainer.

Virtual objects found the basis for being able to employ different (parallel) renderers.

2.2. Optimal Visual Quality

Optimal visual quality is not always achieved by using a very good photo-realistic renderer. Mostly, global illumination renderers tend to be slower than hardware accelerated local illumination renders. Therefore, optimal quality is gained from the trade-off between speed and quality. It

makes sense to employ different renderer types for different demands, even in a single scene. In a complex scene, like a finely modeled city, it might be desireable to use a quality renderer (like a ray tracer) for the foreground and a fast renderer (like a point based renderer) for the background.

basho supports this freedom of choice of renderers by separating the renderers from the kernel. The corresponding kernel manager is called *scene renderer*.

2.3. sceneRenderer

The sceneRenderer is closely coupled with the sceneManager as seen in figure 1, because the render plug-ins use or change the virtualObjects stored in the sceneManager. However, render plug-ins are not connected with each other. The only connection is the indirect connection through the virtualObjects all renderers can manipulate.

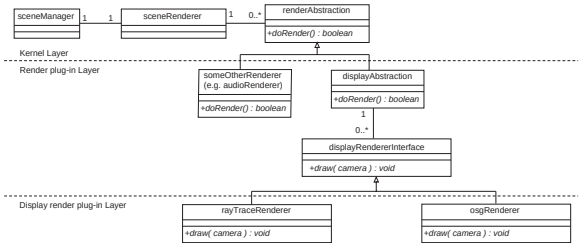

Figure 1: *Class hierarchy showing the sceneRenderer-Interface and the display renderer plug-in.*

All graphics renderers are not connected directly to the sceneRenderer. They are instead plug-ins of the *display abstraction* and the display abstraction is a render plug-in connected to the sceneRenderer. The display abstraction is responsible for creating the output window, managing the camera, calling all its display renderers and to merge their results finally. Merging renderer results is done by comparing the depth components of every pixel. The color value that corresponds to the nearest depth value is chosen. All renderers are implemented as threads to allow concurrency. With this, it is easy to distribute the different renderers on different computers utilising possibly different render-techniques. Every renderer itself can build a distributed system, too.

3. Examples

During the LabNight of the IEEE Virtual Reality Conference 2005 a technical demonstration of *basho* has been shown. In this demo three renderers, point based rendering, ray tracing and hardware rendering, where utilised to create an interactive scene as seen in figure 2. However, all objects have been statically assigned to the different renderers.

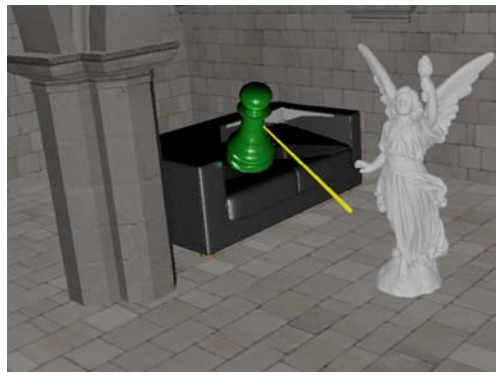

Figure 2: *Three different renderer types displaying a scene.*

Next to those three graphics renderers a sound and a physics renderer have been implemented as *basho* plug-ins.

4. Conclusion and Future Work

The ability to mix different rendering techniques increases the rendering quality where needed. At the moment objects are statically assigned to different renderers. A manager is needed to assign objects dynamically to different renderers. Other points to solve are antialiasing and transparency. When mixing global and local illumination techniques, shadows cannot be cast on locally illuminated objects as well as reflections of those objects on globally illuminated objects, e.g.. This issues are under current investigation.

This work was partially supported by the Ministry for Science and Research (MWF) of the federal state of Northrhine-Westfalia under grant 80081505.

References

[BLRS98] BLACH R., LANDAUER J., RÖSCH A., SIMON A.: A highly flexible virtual reality system. In *Future Generation Computer Systems Special Issue on Virtual Environments* (Elsevier Amsterdam, 1998).

[GHJV95] GAMMA E., HELM R., JOHNSON R., VLISSIDES J.: *Design Patterns*. Addison-Wesley Verlag, 1995.

[HM04] HINKENJANN A., MANNUSS F.: basho - A Virtual Environment Framework. In *SVR 2004* (Sao Paulo, Brazil, 2004), 7th Symposium on Virtual Reality.

[Tra01] TRAMBEREND H.: Avango: A Distributed Virtual Reality Framework. In *Proceedings of AFRIGRAPH 2001, 1st International Conference on Computer Graphics, Virtual Reality and Visualization in Africa* (November 2001), ACM.