ODYSSEE – EXPLORATIONS IN MIXED REALITY THEATRE USING DIRECTX9

A. Eliëns
Intelligent Multimedia Group,
Department of Computer Science
Vrije Universiteit
De Boelelaan 1081, 1081 HV Amsterdam,
Netherlands
E-mail: eliens@cs.vu.nl

KEYWORDS
mixed reality theatre, multimedia applications, DirectX9.

ABSTRACT

In this paper we will discuss our experiences in developing a mixed reality application for a theatre production of the Odyssee. The Odyssee is a wellknown account of the travels of Ulysse leaving Troje, in 24 episodes ending in his return to Ithaca and his reunion with Penelope. The actual theatre production, which is performed in temporarily empty office buildings, takes 12 parts which are played in 12 successive rooms through which the audience, subdivided in small groups, is guided one room after another for about five minutes per room. The initial idea was to have a large number of see-through goggles and augment the actual performance with additional information using text and images. In the course of the project, however, we had to scale down our ambitions, and we ended up using simple LCD-projection goggles with a low-resolution camera, for which we developed a mixed reality application, on the DirectX platform, using video capture projection in 3D with text and images. What we will describe here covers our final application, the criteria and guidelines we used in our production, as well as what may in retrospect be characterized as our explorations of DirectX.

INTRODUCTION

In june 2003, our group was asked to advise on the use of VR in a theatre production of the Odyssee. Lacking experience in this field, we accepted the invitation to participate with some reluctance, since at the time we didn't have any clue what the VR for the theatre production should look like. Nevertheless, we took the invitation as a challenge and started looking for appropriate hardware, bothering collegues for information on mixed reality art productions, and downloading code to explore software technologies. Many hurdles ware to be taken. We had to deal with organizational issues, such as finding the money for financing the actual production (which proved to be quite a hurdle), finding the right people (students, in our case) to select material and contribute to the code; aesthetic issues, in particular to determine which approach to take to reach at an effective solution; and not in the least technical issues, to realize the production on a sufficiently efficient low-cost platform.

In this short paper, we will first briefly describe the Odyssee theatre production. Then we will report on how we arrived at our present mixed reality solution. And, after a brief characterization of our platform of choice, we will look at our mixed reality solution in somewhat more (technical) detail. We finish with recapitulating the lessons we learned from our explorations in mixed reality theatre, and a brief discussion of the further development and implementation of our system.

BACKGROUND – THE ODYSSEE THEATRE PRODUCTION

The Odyssee. theatre production was initiated by Ground Control (www.ground-control.org), as a successor of previously succesful theatrical spectacles, including an open air performance of Faust. In effect, two performances of the Odyssee were planned, an out-door (external) version, involving real ships at the shore of a lake, and an in-door (internal) version, to be played in temporarily empty office buildings. The in-door version is meant to give a more psychological rendering of the Odyssee, see (Entanaclaz, 2003), where the travels of Ulysses are experienced by the audience as a confrontation with themselves. Our contribution was asked for the in-door version, to enhance the experience of the audience with additional VR.

The Odyssee is a wellknown account of the travels of Ulysses leaving Troje, in 24 episodes ending in his return to Ithaca and his reunion with Penelope. The actual theatre production takes 12 parts which are played in 12 successive rooms through which the audience, subdivided in small groups, is guided one room after another for about five minutes per room. Our initial idea was to add information in the form of text and images, to direct the interpretation of the audience towards a particular perspective. In that beginning stage, somewhat optimistically, we planned to offer multiple perspectives to each participant, in an individualized manner, dependent on the actual focus of attention of the individual participant.

INITIAL IDEAS – VR AND AUGMENTED REALITY

Our first problem was to find suitable hardware, that is see-through goggles. Searching the Internet gave us the name of a relatively nearby company, Cyber Mind NL (www.cybermind.nl) , that specialized in entertainment VR solutions. Both price-wise and in terms of functionality semi-transparent see-through glasses appeared to be no option, so instead we chose for simple LCD-projection goggles with a (head-mounted) low-resolution camera. This solution also meant that we did not need expensive head orientation tracking equipment, since we could, in principle, determine focus using captured image analysis solutions such as provided by the AR Toolkit (www.hitl.washington.edu/artoolkit). Moreover, captured video feed ensured the continuity and reactiveness needed for a true (first-person perspective) VR experience.

Augmented or mixed reality is an interesting area of research with many potential applications, see (Grau, 2003). However, in the course of the project we dropped our ambition to develop personalized presentations using image analysis, since we felt that the technology for doing this in a mixed reality theatre setting was simply not ripe, and instead we concentrated on using the captured video feed as the driver for text and image presentation. In addition, we developed image manipulation techniques to transform the (projection of the) captured video, to obtain more implicit effects, as to avoid the explicit semantic overload resulting from the exclusive use of text and images.

TECHNOLOGICAL CONSTRAINTS – THE DIRECTX9 PLATFORM

After a few experiments with the AR Toolkit, it soon appeared that the frame rate would not be sufficient, on the type of machines our budget would allow for. Moreover, reading the AR Toolkit mailing list, marker tracking in a theatrical context seemed to be more or less unfeasible. So, we shifted focus to the DirectX SDK 9, both for video capture and projection in 3D. The DirectX9 toolkit is a surprisingly functional, and very rich technology for multimedia applications, supporting streamed video, including live capture, 3D object rendering and precise synchronisation between multimedia content-related events, Adams (2003). At that time, and still at the time of writing, our own *intelligent multimedia technology* was no option, since it does not allow for using live video capture and is also lacking in down-to-the-millisecond synchronisation.

After exploring texture mapping images copied from the incoming captured video stream, we decided to use the VMR-9 video mixing renderer introduced in DirectX 9, that allows for allocating 3D objects as its rendering surface, thus avoiding the overhead of explicit copies taken from a video processing stream running in a separate thread. Although flexible and efficient, DirectX is a low-level toolkit, which means that we had to create our own facilities for processing a scenegraph, world and viewpoint transformations, and, even more importantly, structuring our mixed reality presentations in time.

STRUCTURING TIME – MAINTAINING ‘SEE THROUGH’ AESTHETICS

One of the problems we encountered in discussing what we conveniently may call the VR with the producer of the Odyssee theatre performance was the high expectancy people have of VR, no doubt inspired by movies as the Matrix and the like. In mixed reality applications, manipulating persons, warps in space, and basically any intensive image analysis or image manipulation is simply not possible in real time. Moreover, there is a disturbing tendency with the layman to strive for semantic overload by overlaying the scene with multiple images and lines of text, thus obscuring the reality captured by the camera and literally blocking the participants view and awareness of the scene. Basically, as a guideline, we tend to strive for 70% visibility of the scene, 20% image or projection transformations and only 10% of information in the form of text and images.

The total duration of our presentation is only 2 minutes, or 118 seconds to be precise. We made a subdivision in 4 scenes, with transitions inbetween, hierarchically ordered in a tree-like structure. Initially, we abstracted from the actual duration, by taking only the fraction of the time passed (in relation to the total duration) as an indication for which scene to display. However, when the development reached its final stages, we introduced actual durations that allowed us to time the sequence of scenes to the tenth of a second. In addition, we used multiple layers of presentation, roughly subdivided in background captured image, the transformed captured image projected on 3D objects, and, finally, pictures and text. These layers are rendered on top of eachother, triggered in a time-based fashion, semi-independent of one another. The frame rate varies between 20 and 30, dependent on the number of images simultaneously used for texturing. Our final mixed reality theatre application may be considered a prototype, awaiting to be put to the test by a larger audience.

LESSONS LEARNED – OUR EXPLORATIONS REVISITED

Altogether, the development of the mixed reality theatre application has been quite an experience, in multiple ways. Not in the least it has been (and still is) a challenge to explain the possibilities of mixed reality applications to the layman, that do not take the abstractions we use in our daily academic life for granted.

Reinventing the wheel is not as simple as it seems. Nevertheless, developing scenegraph processing facilities and the appropriate timing mechanisms for controlling the mixed reality presentation was, apart from being a rekindling of basic skills, a learnful experience.

In our project, the major obstacle became the hardware, since our approach required one PC and goggle set per visitor. Also, providing multiple visitors with a goggle became a problem, due to the wiring.

TECHNICAL ISSUES – PROGRAMMING THE PRESENTATION SYSTEM

In the course of time, I continued working on the system and it has been used for parties as well as for enlivening my lectures. It actually does include many of the features of a VJ system, and is currently named ViP (www.virtualpoetry.tv). The major challenge, when I started development, was to find an effective way to map live video from a low/medium resolution camera as textures onto 3D geometry. I started with looking at the ARToolkit but I was at the time not satisfied with its frame rate. Then, after some first explorations, I discovered that mapping video on 3D was a new (to some extent still experimental) built-in feature of the DirectX 9 SDK, in the form of the VMR9 (video mixing renderer) filter.

The Video Mixing Renderer filter

The VMR filter is a compound class that handles connections, mixing, compositing, as well as synchronization and presentation in an integrated fashion, Pesce (2003). Before discussing the VMR9 in more detail, let's look first at how a single media stream is processed by the filter graph. Basically, the process consists of the phases of parsing, decoding and rendering. For each of these phases, dependent on respectively the source, format and display requirements, a different filter may be used. Synchronization can be either determined by the renderer, by pulling new frames in, or by the parser, as in the case of live capture, by pushing data on the stream, possibly causing the loss of data when decoding cannot keep up with the incoming stream.
The VMR was originally introduced to allow for mixing multiple video streams, and allowed for user-defined compositor and allocator/presenter components. Before the VMR9, images could be obtained from the video stream by intercepting this stream and copying frames to a texture surface. The VMR9, however, renders the frames directly on Direct3D surfaces, with (obviously) less overhead. Although the VMR9 supports multiple video pins, for combining multiple video streams, it does not allow for independent search or access to these streams. To do this you must deploy multiple video mixing renderers that are connected to a common allocator/presenter component.
When using the VMR9 with Direct3D, the rendering of 3D scenes is driven by the rate at which the video frames are processed.

The ViP system

In developing the ViP system, I proceeded from the requirement to project live video capture in 3D space. As noted previously, this means that incoming video drives the rendering of 3D scenes and that, hence, capture speed determines the rendering frame rate.
I started with adapting the simple allocator/presenter example from the DirectX 9 SDK, and developed a scene management system that could handle incoming textures from the video stream. Inherited by all classes is the scene class interface, which allows for (one-time) initialization, time-dependent compositing, restoring device settings and rendering textures. The scene graph itself was constructed as a tree, using both arrays of (sub) scenes as well as a dictionary for named scenes, which is traversed each time a video texture comes in.
Later on, I adapted the *GamePlayer* which uses multiple video mixing renderers, and then the need arose to use a different way of indexing and accessing the textures from the video stream(s).
Adopting the scene class as the unifying interface for all 3D objects and compound scenes proved to be a convenient way to control the complexity of the ViP application. However, for manipulating the textures and allocating shader effects to scenes, I needed a global data structure (dictionaries) to access these items by name, whenever needed. As a final remark, which is actually more concerned with the software engineering of such systems than its functionality per se, to be able to deal with the multiple variant libraries that existed in the various releases of DirectX 9, it was needed to develop the ViP library and its components as a collection of DLLs, to avoid the name and linking clashes that would otherwise occur.

CONCLUSIONS

We have described, in a somewhat anecdotical fashion, our experiences in developing a mixed reality application for the Odyssee theatre production, to enhance the participants experience of the performance. Our explorations involved, among others, to deal with expectancies of VR, aesthetic issues, not to mention production schedules, cooperation, financial issues, but above all it meant setting the first steps in developing technology for mixed reality theatre. Most important, however, is that our explorations show the richness of the DirectX toolkit, not only for games but also for realtime multimedia presentations.

ACKNOWLEDGEMENTS

We thank Bart Gloudemans and Rutger van Dijk (both students at the Vrije Universiteit) for their practical work on the project, as well as their general contribution to the final contents of the work. Furthermore, we are grateful to Johan Hoorn and Bert Barten for engaging us in the Odyssee theatre project.

REFERENCES

Adams J. 2003. “Advanced Animation with DirectX”. Premier Press.
Entabaclaz A.. 2003. “Les metamorphoses d'Ulysse -- reecritures de l'Odyssee”. Editions Flammarion, Paris.
Grau O.. 2003. “Virtual Art -- From Illusion to Immersion”. MIT Press.
Pesce M. 2003. “Programming Microsoft DirectShow for digital video and television”. Microsoft Press.

AUTHOR BIOGRAPHY

ANTON ELIENS studied art, psychology, philosophy, and computer science. He is lecturer at the Vrije Universiteit Amsterdam, where he teaches multimedia courses. He is also coordinator of the Master Multimedia for Computer Science. He has written books on distributed logic programming and object oriented software engineering.