Sharing Visual Information in Virtual Environments using Projective Texture Mapping

Yannick Francken, Johan Huysmans, Philippe Bekaert
Hasselt University - Expertise Centre for Digital Media
transnationale Universiteit Limburg
Wetenschapspark 2, 3590 Diepenbeek, Belgium
{yannick.francken, johan.huysmans, philippe.bekaert}@uhasselt.be

ABSTRACT

We present a method for sharing visual information in 3D virtual environments, using a projective texture mapping based method. Avatars can share information with other avatars by projecting relevant information into the environment. In addition to standard projective texture mapping, an important depth cue is added: projected light is attenuated in function of the light-travel distance. This is efficiently accomplished on a per vertex basis by adaptively super-sampling under-sampled polygons. This way, the projection quality is maximized while keeping a fixed frame rate. Our technique is implemented into the Quake III engine, extending its shading language with GLSL fragment and vertex shaders.

Categories and Subject Descriptors:
H.5.1 [INFORMATION INTERFACES AND PRESENTATION (I.7)]: Multimedia Information Systems, I.3.7 [COMPUTER GRAPHICS]: Three-Dimensional Graphics and Realism

General Terms: Algorithms, Performance.

Keywords: Projective Texture Mapping, Visual Information Sharing.

1. INTRODUCTION

Information sharing between clients in 3D virtual environments is a common problem, especially when a traditional setup of keyboard and/or mouse is being used. Typically client-client interaction is achieved by keyboard, where users can type messages to other users in the virtual environment. More recent approaches make the interaction multimodal, for example by using speech and video. In this paper, a new projection based interaction technique that allows for sharing visual information among clients, is presented.

Complementary to our work, interactive lens techniques were first introduced in the context of see-through interfaces [1] and later applied to 3D virtual environments [2, 10]. It allows the user to focus on certain parts of the environment by utilizing a virtual lens, which annotates the data. For example, a magnifying lens can be used to show hidden layers of the environment. Recently, lens techniques have been extended to support convex volumes and implementation on current graphics hardware [8].

Another see-through approach is cutting, where parts of the geometry are cut away in order to see through it, for example using CSG operations [3]. The region of interest can be defined by the user, using a lathe, whereas the volume is defined through an image-space selection technique, namely the first intersection of a ray perpendicular to the view plane.

In the augmented reality domain projection techniques are often used. Raskar et al. [7] tag objects in the real world using RFIG tags in order to make the world self-describing. Users can query the RFIG tags by using a projector to obtain knowledge about the world. These techniques can be used for inventory control as part of logistics.

Combining augmented reality with virtual interfaces, we propose a virtual projection based approach [9]. Visual information, for example images, movies or personal information, is projected on top of the geometry. The area of interest is defined by the user through the use of a projector, which size, shape and contents can be customized. As viewers in the virtual world can observe other users' projections, an information sharing method is achieved.

Practical applications of our approach are numerous. In learning environments, the projector can visualize additional information of the data being studied, for example labeling different parts of a car engine. Our implementation is part of a socio-cultural project, called Virtual Arts Centre of the Future [6], where communication of visual information in virtual environments plays a prominent role.

For our purpose, a prototype is implemented into the Quake III engine [5]. We have extended the Quake III shading language, Q3Radiant, with GLSL [4] fragment and vertex shader functionality. Using these more expressive shaders, not only the visual appearance of the game is considerably enhanced, but users can also easily share content by projecting customized shaders into the virtual world.

2. ALGORITHM

Our projective texture mapping based technique consists of multiple stages. First, all triangles that intersect the projection frustum are collected. Then, a homogeneous triangle mesh is generated to replace these triangles. For the resulting triangles, intensity attenuation values are computedper vertex in function of depth and the affected triangles are redrawn.

ACE'07, June 13–15, 2007, Salzburg, Austria.

(a) Projecting player (b) Observing player (c) Shader room

Figure 1: Extended Q3Radiant shaders. (a) player projecting on a wall (b) other player observes projection (c) bumpmapped room showing a shader projection together with other fixed shaders

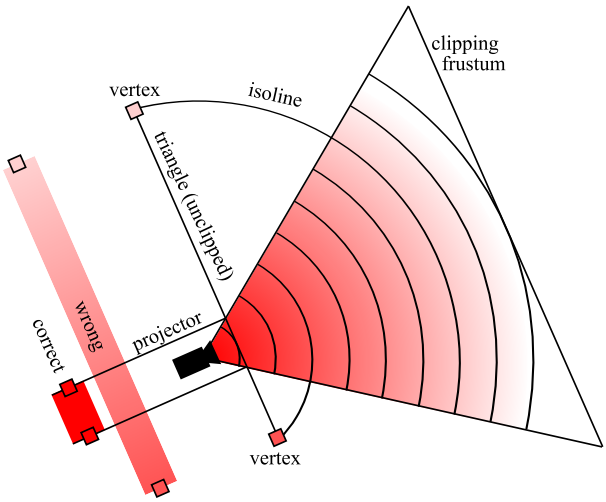

Figure 2: Projection on large triangle. Intensity attenuation by linearly interpolating assigned vertex intensities using isolines is not allowed.

When a user utilizes a projector, it defines a projection frustum containing a set of triangles in the world which can possibly be affected by the projector. As the intensity of the projection decreases with the distance from the projection origin, a certain distance exists where the received intensity is low enough that it becomes invisible to the observer. The drop-off rates are typically defined by a radial attenuation function. This maximum distance constrains the depth of the frustum. The determined frustum will be used further in the pipeline, during the triangle culling and clipping stages.

In order to get a realistic intensity value for every vertex, care must be taken when processing large triangles. In case of a typical radial attenuation function, increasing values propagate in a spherical pattern around the projection origin, based on the isolines (Figure 2). Thus, when processing large triangles, a simple linear interpolation of the vertex intensity values will not generate correct attenuation values. As a consequence, the triangles must be super-sampled in all dimensions to get a homegeneous triangle mesh.

The triangle splitting heuristic aims at generating a homogeneous triangle mesh, given a certain error tolerance. The error tolerance defines the density of the triangle mesh, i.e., a low tolerance will result in a very dense mesh where a larger tolerance will result in a less dense, faster generated mesh (Figure 3). In our implementation, a greedy splitting heuristic is used which adaptively super-samples all triangles in their largest dimension until the specific error tolerance is achieved. The tolerance is defined as the length of a triangle in its largest dimension, guaranteeing a homegeneous triangle mesh in every dimension. The error tolerance is adapted at runtime, striving for a stable frame rate and a maximal projection quality every frame.

In the next phase, every vertex is labeled based on the projected texture coordinate (s, t, q). Per vertex exactly one binary number is constructed, depending only on the signs of the components s, t and q, similar to the Cohen-Sutherland clipping algorithm. This way every single bit defines on which side of the frustum half-spaces the vertex is located. By applying fast binary operations, we cull the triangles lying fully outside the projector frustum. Afterwards, a more precise frustum clipping is applied (by OpenGL) on the remaining triangles and attenuation values are calculated per vertex. The spherical intensity isovalue pattern is approximated by piecewise linear interpolation of the vertices. Then, all triangles influenced by the projection are redrawn.

3. IMPLEMENTATION

We have extended Quake III with our projective texture mapping based algorithm. Players can use the projector as an ordinary, non-lethal weapon projecting the requested data onto the world. Other players in the world can observe the projection.

In order to get a general, dynamic system, we have extended the Q3Radiant shading language with GLSL fragment and vertex shaders. The original Q3Radiant language allows for creating animated textures using simple operations like texture blending, texture coordinate transformations, vertex deformations etc. We have added the possibility to insert GLSL vertex and fragment shaders into the different Q3Radiant shader stages, resulting in a easy-to-use and very powerful language. Created shaders can also be used to customize the projection content. A few different extended shaders are shown in Figure 1.

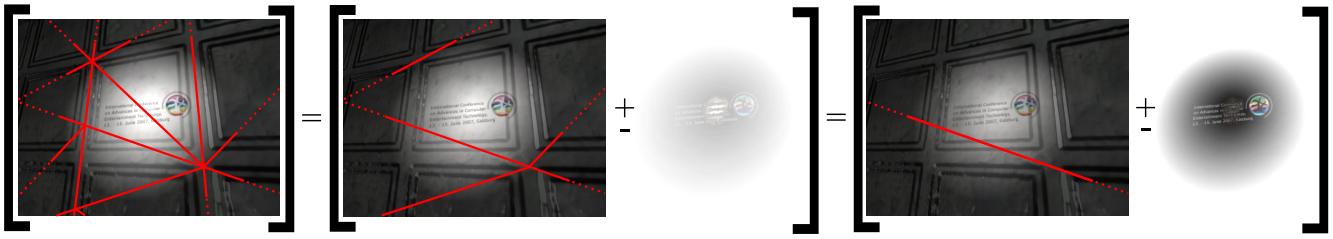

Figure 3: Attenuation error with respect to edge length. Red lines represent the triangle edges, only the solid pieces are lying in the projection frustum. Left: small edge length resulting in correct projection, Middle: medium edge length resulting in a small error, Right: long edge length resulting in a large error.

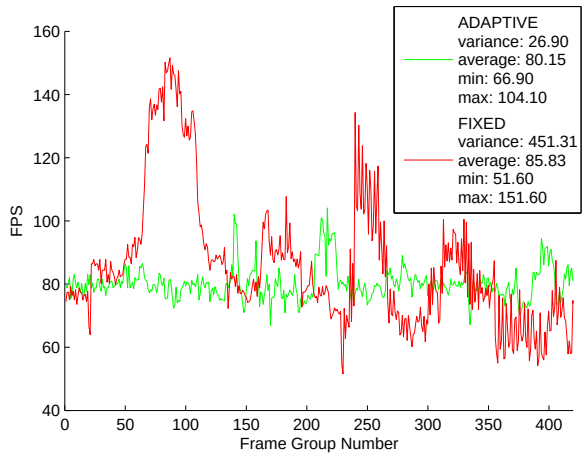

Figure 4: Comparison of frame rates: adaptive and fixed triangle super-sampling (groups of 10 frames). The *average* projection quality is the same for both graphs, although the quality is better spread in the adaptive case to keep a frame rate around 80 fps.

4. RESULTS

Figure 3 shows the effects of the homogeneousness of the triangle mesh. The left of the figure shows a projection with a dense enough triangle mesh which does not show approximation errors. The middle one is visualized with a coarser mesh, where a slight deviation in accuracy appears. The right figure shows an even coarser approximation, where the loss in image detail is more severe than in the previous images.

To test the results of our splitting heuristic, we have recorded a demo which is both played with and without our adaptive splitting heuristic. As can be seen in Figure 4, our adaptive technique accomplishes a quasi-fixed frame rate for the whole demo scene, while the non-adaptive heuristic results in greater frame rate variations. The overall frame rate for our adaptive splitting technique is only slightly below those of the fixed heuristic. In both approaches, the mean precision of the subdivisioning technique is the same. However, in our technique this precision is met almost for every frame while in the fixed heuristic, this precision is met in all frames.

Thus our approach results in an almost fixed processing time per frame, where the precision is controllable. This can be exploited to use the remaining CPU-time for other processing, like physics etc.

5. CONCLUSIONS

In this paper we have demonstrated a visual information sharing technique in 3D virtual environments, by using projective texture mapping. We have developed an adaptive splitting heuristic, which observes both the frame rate and splitting quality, to generate a homogeneous triangle mesh in order to compute correct intensity values for every vertex, resulting in a stable frame rate.

We have implemented our projective texture mapping approach in the Quake III engine, extending its shading language with GLSL fragment and vertex shaders. This extended shading language can be used by the user to customize his projections.

Possible future work includes an investigation of relevant attenuation functions, with their corresponding ideal splitting heuristics. We believe a more advanced splitting heuristic will increase the speed of our algorithm considerably. A migration of our algorithm from CPU to graphics hardware may also result in an overall performance speedup.

A practical extension is the integration of volumetric shadowing techniques to avoid projecting through the scene objects. While achieving more realistic visual quality, this might result in performance bottlenecks in cases where many clients are using a projector.

6. ACKNOWLEDGMENTS

The authors acknowledge financial support on a structural basis from the ERDF (European Regional Development Fund), the Flemish Government and the Flemish Interdisciplinary institute for BroadBand Technology (IBBT, Virtual Arts Centre of the Future project).

This paper is realized as a part of the Virtual Arts Centre of the Future (VACF) project, funded by IBBT. Projector concept by: Thomas Soetens, Kora Van den Bulcke. Research and development initiated and in collaboration with Workspace Unlimited: Thomas Soetens, Kora Van den Bulcke and Patrick Bergeron (`www.workspace-unlimited.org`). Furthermore we would like to thank our colleagues for their help and inspiration.

7. REFERENCES

[1] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: the see-through interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics*

and interactive techniques, pages 73–80, New York, NY, USA, 1993. ACM Press.

[2] P. Cignoni, C. Montani, and R. Scopigno. Magicsphere: an insight tool for 3d data visualization. *Computer Graphics Forum*, 13(3):317–328, 1994.

[3] C. Coffin and T. Hollerer. Interactive perspective cut-away views for general 3d scenes. In *VR '06: Proceedings of the IEEE Virtual Reality Conference (VR 2006)*, page 118, Washington, DC, USA, 2006. IEEE Computer Society.

[4] OpenGL Shading Language, `http://www.opengl.org/documentation/glsl/`.

[5] Quake 3, `http://www.idsoftware.com/`.

[6] Virtual Arts Centre of the Future, `https://projects.ibbt.be/vacf/`.

[7] R. Raskar, P. Beardsley, J. van Baar, Y. Wang, P. Dietz, J. Lee, D. Leigh, and T. Willwacher. Rfig lamps: interacting with a self-describing world via photosensing wireless tags and projectors. *ACM Trans. Graph.*, 23(3):406–415, 2004.

[8] T. Ropinski and K. Hinrichs. Real-Time Rendering of 3D Magic Lenses having arbitrary convex Shapes. In *Journal of the International Winter School of Computer Graphics (WSCG04)*, pages 379–386. UNION Agency - Science Press, 2004.

[9] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 249–252, New York, NY, USA, 1992. ACM Press.

[10] J. Viega, M. J. Conway, G. Williams, and R. Pausch. 3d magic lenses. In *UIST '96: Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 51–58, New York, NY, USA, 1996. ACM Press.