[7] Funkhouser, T. A.; Sequin, C. H.; Teller, S. J.: Management of Large Amounts of Data in Interactive Building Walkthroughs, ACM SIGGRAPH Special Issue 1992 Symposium on Interactive 3D Graphics, 1992

[8] Greene, N.; Kass, M.; Miller G.: Hierarchical Z-Buffer Visibility, Proceedings of SIGGRAPH '93, in *ACM Computer Graphics*, 1993

[9] Maciel, P. W. C.; Shirley, P.: Visual navigation of large environments using textures clusters, Proceedings 1995 Symposium on Interactive 3D Graphics, June 1995

[10] Rohlf, J.; Helman, J.: IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics, Proceedings of SIGGRAPH '94, in *ACM Computer Graphics*, 1994

[11] Schaufler, G.: Exploiting Frame-to-Frame Coherence in a Virtual Reality System, Proceedings of IEEE VRAIS '96, 1996

[12] Shade, J.; Lishinski, D.; Salesin, D. H.; DeRose, T.; Snyder, J.: Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments, Proceedings of SIGGRAPH '96, in *ACM Computer Graphics*, 1996

[13] Teller, S. J.: Visibility Computations in Densley Occluded Polyhedral Environments, Ph.D. thesis, Computer Sience Division (EFCS), University of California, Berkeley, 1992

[14] Zyda, M. J.; Pratt, D. R.; Monahan, J. G.; Wilson, K. P.: NPSNET: Constructing A 3D Virtual World, ACM SIGGRAPH Special Issue 1992 Symposium on Interactive 3D Graphics, 1992

.

Figure 11

.

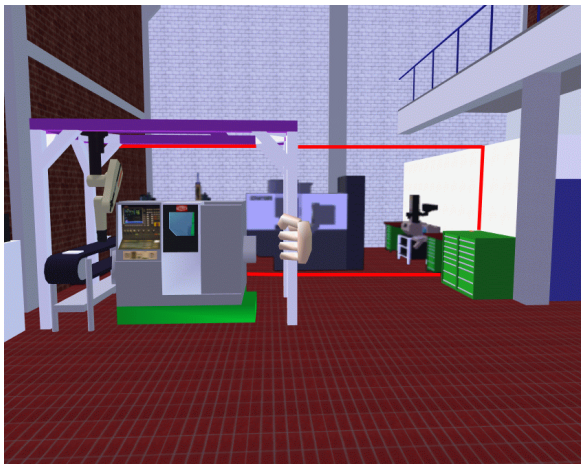

Figure 12

.

Figure 13

.

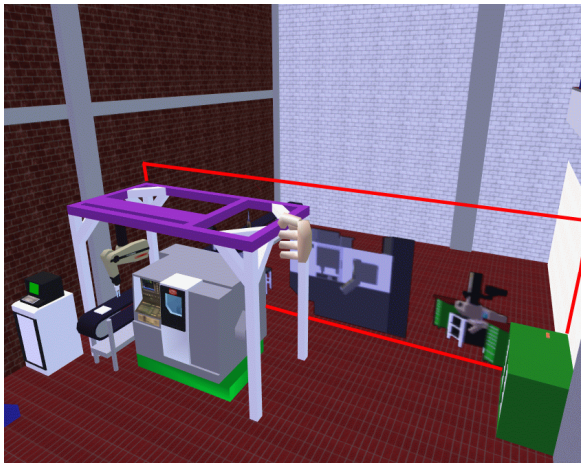

Figure 14

cally working (models containing one million and more polygons take several minutes to load from hard disk into main memory).

The 3D scene editor has subsequently been used to insert four virtual walls into the scene. Figure 8 and Figure 9 show a simplified top view of our test scene. For each textured virtual wall six texture maps are needed. This corresponds to an angle of error of 10 degrees. Our first experiments have shown that the angle of error should not exceed 15 degrees in order to get acceptable results. Further experiments have to be carried out to get more exact values. For two interior virtual walls textures must be generated for front and back faces since these virtual walls may be viewed from both sides. Main occluding objects like “real“ walls, floors etc. are always rendered as geometry and not displayed by textured virtual walls. These regions are set to black during the generation of the texture map. Subsequently these regions are set to 100% transparency.

A total of 36 texture maps has been precomputed for the test scene. During runtime not all of them have to be stored in texture memory. Since the cell-to-cell visibility paradigm is used, only those texture maps must be stored in texture memory which are potentially needed for the following frames. A simple texture paging algorithm has been implemented to exchange texture maps between memory and hard disk.

Figures 11 and 12 show frames from a walkthrough of the test scene. Figure 11 shows a walkthrough without textured virtual walls while Figure 12 shows a walkthrough with textured virtual walls enabled. The virtual wall is outlined. The hand icon which can be seen in the center of the images is needed for navigation and interaction when we use fully immersive VR display techniques (Head Mounted Displays, Datagloves). Figures 13 and 14 show a bird’s eye view of the situation shown in Figure 11 resp. 12. Figure 13 shows geometry rendering while Figure 14 shows textured virtual walls enabled. The virtual wall is outlined again. Note that Figure 14 is used for clarification purposes only. It is not possible for the user to experience such a situation during a normal walkthrough. Using textured virtual walls significant speedups of the frame rate have been measured. First experiments with the system yielded estimated speedup values between 100% and 400%. Speedup values depend very much on how much geometry is culled for the current frame. Currently further extensive tests are being carried out to rate the performance of the system.

8 Conclusions and future work

We have presented a new method which uses texture-based simplification for accelerating frame rates during interactive walkthroughs of complex indoor environments. We have introduced textured virtual walls which enable us to take advantage of the combination of two paradigms: texture-based simplification and cell-to-cell visibility. First results show significant frame rate speedups with little or no loss in image quality.

Currently we are working on a method which automatically generates virtual walls. This has been found to be a difficult task since classical partitioning methods like BSPs and octrees do not seem to be applicable for the generation of virtual walls. Moreover we are checking whether dynamic, run-time based generation of texture maps has advantages for solving our problem. Run-time based generation of texture maps does not require much disk space to store the textures. The large amount of disk space which is needed to store the textures is the major drawback of precomputed textures. Another problem which has to be addressed when using texture-based simplification is the visual artifacts caused by lighting. Since we use static lighting in our models visual artifacts due to incorrect lighting are currently not noticeable. But solutions are needed if other types of lighting are used.

References

[1] Aliaga, D. G.: Visualization of Complex Models Using Dynamic Texture-based Simplification, Proceedings of IEEE Visualization ‘96, 1996

[2] Airey, J. M.; Rohlf, J. H.; Brooks, F. P.: Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments, In Computer Graphics, Volume 24, Number 2, March 1990

[3] Clark, J. H.: Hierarchical Geometric Models for Visible Surface Algorithms, Communications of the ACM, 19, 10 (October 1976)

[4] Cruz-Neira, C.; Sandin, D. J.; DeFanti, T. A.: Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE, Proceedings of SIGGRAPH ‘93, in *ACM Computer Graphics*, 1993

[5] Foley, J. D.; van Dam, A.; Feiner, S. K.; Hughes, J. F.; Phillips, R. L.: *Computer Graphics Principles and Practice*. Addison–Wesley, 1990.

[6] Funkhouser, T. A.; Sequin, C. H.: Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments, Proceedings of SIGGRAPH ‘93, in *ACM Computer Graphics*, 1993

texture map which is valid for zone D1. These objects are highlighted in Figure 9. Objects which intersect a virtual

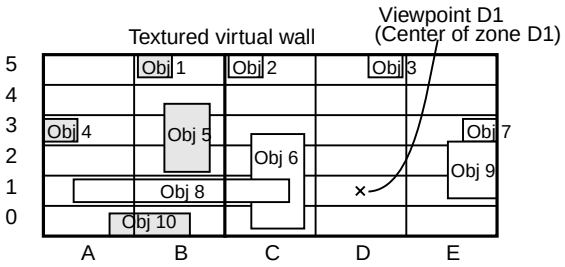

Figure 9 Objects to be rendered on a texture map valid for zone D1

wall are not displayed on the corresponding texture. They are rendered as geometry during the runtime phase because parts of these objects are in front of the virtual wall. If they were added to the texture the error metric would partially be violated. The object selection for texture generation is done by a scene graph traversal. Since all objects which are located behind a virtual wall are “invisible“ during the runtime phase, visibility information for particular cells and objects can be obtained by applying the cell-to-cell visibility paradigm proposed by Teller [13]. Although this method was originally designed for densely occluded indoor environments the ideas presented there can easily be extended to solve our problem. As in Teller’s method we use a lookup table to store visibility information of cells and objects. For each cell the lookup table contains information on which cells are “invisible“. It is referenced for retrieving information about which objects must be rendered for a texture map during preprocessing.

6 Runtime Phase

Geometry information of the virtual walls as well as the textures to be mapped onto them are stored in the scene graph hierarchy. As the viewer moves through the scene the scene graph is traversed for each fame. During the traversal the algorithm determines which virtual walls have to be displayed as well as which textures must be mapped on it and thus which objects can be culled for the zone the viewer is currently located in. This is done by referencing the lookup-table generated during the preprocessing stage. Near geometry (i. e. objects in front of a textured virtual wall) is rendered “as it is“. Classical frustum culling and level-of-detail modeling are used to further reduce the number of polygons which must be rendered. Furthermore our approach uses a blending mechanism to achieve smooth transitions between texture and geometry, as well as between two different textures. By crossing a border between two cells the viewer passes a blend zone. Moving towards a virtual wall within a blend zone will decrease

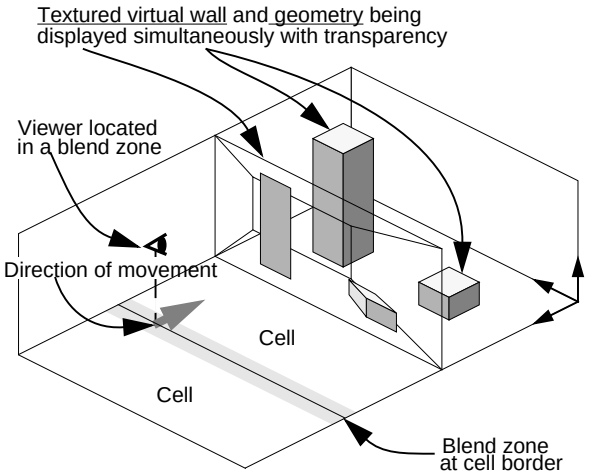

Figure 10 Blend zones used during transition between texture and geometry

the transparency value of the texture map and increase the transparency value of the geometry. Consequently the virtual wall and geometry are displayed simultaneously while a blend zone is passed (Figure 10). Note that only the geometry of the next cell is rendered. Further distant geometry is represented by the next textured virtual wall. A similar blending mechanism is used if textures must be exchanged on a virtual wall. Transparency-blending has been successfully applied to level-of-detail transitions. In our system the image quality is correspondingly much better in comparison to a simple switching. Due to the absence of “popping effects“ transitions between two representations are hardly noticeable

7 Results

We have implemented the algorithm on a SGI Onyx RE2 using OpenGL and IRIS Performer [10]. The indoor scene used for the first tests is a model of a medium sized laboratory for production engineering at our institute. The models of building parts, machine tools, robots and accessory have been created using mechanical CAD systems. These models have been imported into a 3D scene editor which has been used to apply realistic materials and detail texture maps. Furthermore this tool has been used to compose the scene and to modify the scene graph for efficient visibility culling. The whole scene consists of 14,500 polygons. We have chosen a relatively small scene for handling reasons since we first had to check whether our approach is basi-

The angle of error α is:.

$$\alpha = \text{atan}\left(\frac{f}{x_P - x_T}\right) = \text{atan}\left(\frac{\Delta b}{x_P}\right) \qquad (2)$$

Note that the perspective projection is independent from the viewing direction. In other words, the perspective projection does not change if the viewer rotates the head [5]. Therefore it is not necessary to distinguish different viewing directions for the calculation of the projection error.

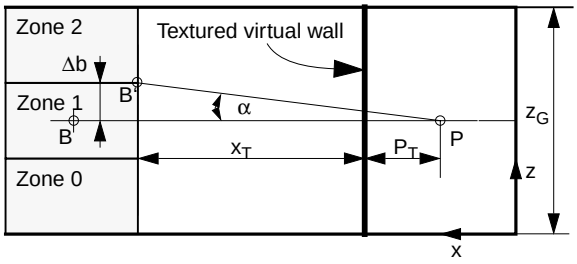

Figure 7 Error metric for textured virtual walls

A simple error metric can be defined as follows: Consider the zones introduced before in relation to the angle of error α (Figure 7). Without proof one can see that α reaches its maximum value under the following conditions:

- The current viewpoint B' in relation to the virtual wall is located in the closest possible corner of a zone.
- The point to be projected on the texture is located as close as possible behind the virtual wall (P_T=0).

Given a user defined value for the angel of error the minimum distance allowed between the edge of a zone and the virtual wall is:

$$x_{T_{min}} = \frac{\Delta b}{\tan\alpha} \qquad (3)$$

Since point B is the geometric center of the zone, Δb equals half the width of the zone. Therefore the width of the indoor scene must be:

$$z_G = 2\Delta b \cdot n_T \qquad (4)$$

where n_T equals the number of zones respectively the number of textures needed for a single virtual wall. Since the dimensions of the indoor scene (length and width) are known either the number of textures per wall necessary for a given amount of virtual walls or the number of virtual walls necessary for a given amount of textures per wall can be calculated. These computations depend on the angle of error the user of the system is prepared to accept.

The discussion of our error metric has been restricted to an indoor environment with a rectangular ground plan. Furthermore the error discussion has been limited to 2D assuming a constant height above the floor for the eye point. This has been done to clarify the basic ideas of textured virtual walls. Our approach is neither restricted to rectangular ground plans nor to a limited number of viewpoints (i. e. those with a constant height above the floor). These extensions can easily be made at some additional cost.

5 Preprocessing Phase

The input to our preprocessor is a 3D model of a complex indoor environment with no major occluders like "real" walls. In a first step we manually insert a number of virtual walls using a 3D editor. Virtual walls divide a large interior room into a set of smaller rooms. We call the rooms separated by virtual walls cells. Subsequently the algorithm divides each cell in zones. The number of zones generated for one cell depends on the distance between the virtual walls and the maximum angle of error the user accepts (see section 4). .

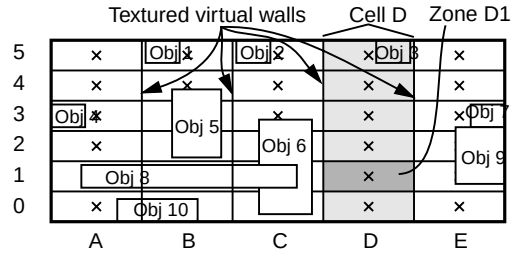

Figure 8 Partitioning an indoor scene using virtual walls

Furthermore the texture sampling point for each zone is computed. The diagram in Figure 8 presents an example. A top view of an indoor environment with several geometric complex objects is shown. Cells separated by virtual walls are named from "A" to "E". Zones are numbered, in this case from 0 to 5. A particular zone is addressed by its cell identifier and zone number. For clarification zone D1 and cell D are exemplary highlighted in Figure 8. The texture sample points are marked by an "x"During the next preprocessing step the specific textures are sampled. For example the objects #1, #4, #5, and #10 are rendered for a

added all geometry represented by it can be culled. The diagram in Figure 3 shows the result of the process. Note that the projection on the textured virtual wall is only correct for the texture sampling point and approximately correct for viewpoints which are located in a region near the texture sampling point.

4 Projection Errors

The central problem caused by texture-based simplification is that these texture-maps are only accurate for a single viewpoint: the texture sampling point. Accurate means that the perspective projection is only correct in this single case. The basic idea behind texture based simplification is that these errors can be accepted as long as the current viewpoint is not too far away from the texture sampling point and the texture-map representing distant geometry is considerably far away. Under these circumstances the change in the perspective projection is enough low that it is hardly noticeable. If a certain threshold is exceeded some kind of correction is necessary.

Our approach solves the problem by exchanging the texture mapped onto the virtual wall according to the current viewpoint. We define zones where a particular texture is "valid". Consider the diagram in Figure 4. As long as the

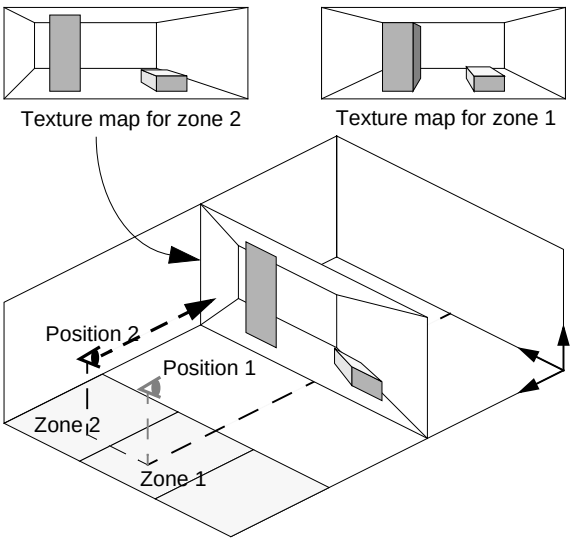

Figure 4 Introducing multiple textures and zones

viewer is located in zone 1 texture 1 is displayed on the virtual wall. If the viewer enters zone 2 texture 1 is replaced by texture 2. If the viewer leaves the shaded zones and moves toward the virtual wall the texture is changed to geometry. All textures necessary are precomputed. The geometric centerpoint of a zone is used as the sampling point for the particular texture which belongs to this zone. For most of the textures an off-axis-projection is needed. The top view diagram of the previously used scene (Figure 5) shows the reason for this necessity. We

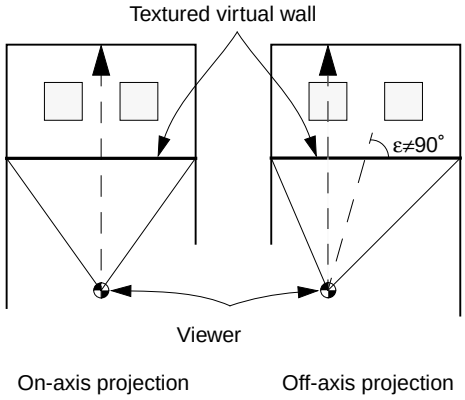

Figure 5 On-axis and off-axis perspective projection

use a projection-paradigm similar to that applied by Cruz-Neira et al. [4] in the CAVE system. Textured virtual walls can in fact be viewed as something like a "virtual CAVE-system"

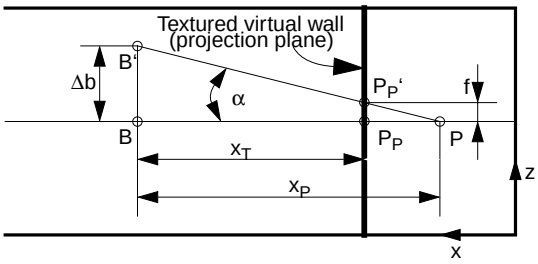

Figure 6 Projection error

The question to be answered is: What projection error is tolerable? In order to answer this question it is first of all necessary to define the projection error. Consider the top view diagram of an indoor scene shown in Figure 6. Assume the texture displayed on the virtual wall has been sampled at point B. Let P be a point of geometry represented by the texture. P_P is the point on the projection plane (i. e. the virtual wall) which corresponds to P. If the current viewpoint equals B the perspective projection is correct. Now assume the viewer moves parallel to the virtual wall to point B'. Now point P_P' corresponds to P. If the texture sampled at B is used as approximation at B', the perspective error is:.

$$f = \Delta b \cdot \frac{x_P - x_T}{x_P} \qquad (1)$$

mapped quadrilaterals, several of them will occlude each other. This increasing depth complexity of textured polygons is very costly to the pixel-processing of a z-buffer graphic hardware and may cause pixel fill limitations which as a result will decrease the frame rate.

Aliaga [1] proposed a texture-based simplification method which uses precomputed texture maps to represent distant geometry of complex indoor environments. Similar to our approach near geometry is rendered normally. As the viewer approaches texture-based simplifications these are changed to geometry. The approach eludes the projection error problem by morphing the geometry surrounding the static texture map. Depending on the current viewpoint the geometry is morphed in a matter that ensures it always matches the projection of the texture. The approach accepts to render geometry surrounding the texture inaccurately. Geometry morphing is only applied to objects located behind the projection plane of the texture. Near geometry is not morphed. This is acceptable as long as the current viewpoint is not far away from the texture sampling point. If the viewer gets distant from the sampling point by moving parallel to the texture (i. e. the distance to the texture is not changing) the projection error between near geometry on one side and morphed geometry and texture on the other side will be noticeable after some time.

Our approach does not need to solve this discontinuity problem because using virtual walls implies that there is no surrounding geometry in locations that are more distant from the viewpoint than the projection plane of the texture. In our system distant geometry is “invisible“ and therefore culled. The projection error problem with near geometry (i. e. geometry in front of the texture map) is solved by applying a simple error metric which chooses an appropriate texture map from a set of precomputed textures. It is not necessary to store all precomputed textures in the texture mapping hardware. Using the cell-to-cell visibility concept enables us to select the required subset of texture maps on time as the viewer moves through the scene and to page them from hard disk to texture memory when needed. Furthermore textured virtual walls ensure that there will be no pixel filling problem due to several overlapping texture maps.

3 Textured Virtual Walls

Textured virtual walls are simplified representations of distant geometric objects. Consider the diagram in Figure 1. t is a bird’s-eye view of a very simple indoor scene. A viewer is looking at two distant objects. From his viewpoint the scene looks like as shown in Figure 2. Now a textured virtual wall is added to the scene. It is a simple quadrilateral placed in front of the distant objects. his polygons now used as the “projection screen“. Distant geometry Iis rendered Tonto it and the resulting image is

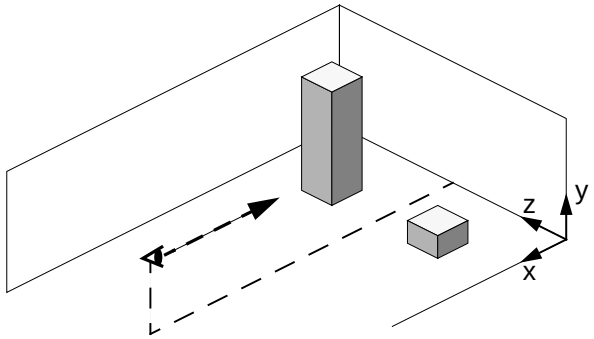

Figure 1 *Simple indoor scene*

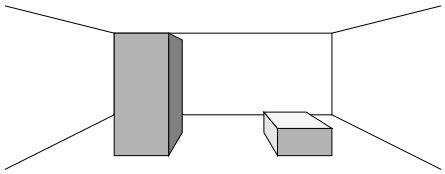

Figure 2 *View of the simple indoor scene seen from the viewpoint marked in Figure 1*

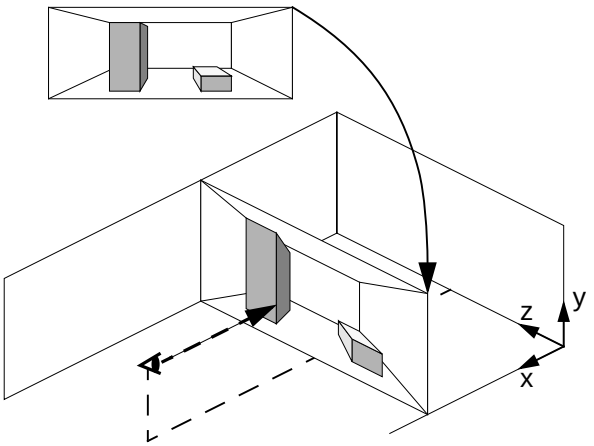

Figure 3 *Adding a textured virtual wall to the scene and culling geometry*

stored as a texture map. The viewing direction during the texture sampling isialways orthogonal to the textured virtual wall. The height of the texture sampling point above the floor is fixed to a value of the eye level of an average walking viewer. After the textured virtual wall has been

room into a number of separated smaller rooms. We call these separated rooms cells. The virtual walls virtually occlude all objects which are behind them. The sparsely occluded environment has changed to a virtually densely occluded environment. Teller [13] has proposed a method for visibility computations in densely occluded environments which introduces a concept called cell-to-cell visibility. We use a simplification of this concept to determine which cells are “virtually“ invisible from a given cell due to the virtual walls. During the walkthrough all geometry inside the “invisible“ cells is culled. Since the culled geometry is not really invisible it has to be represented in a suitable manner. This is done by textures mapped onto the virtual walls. These texture maps represent the culled geometry. Using texture maps this way in an interactive system is an approximation. The error introduced can be neglected if the current viewpoint is located nearby the texture sampling point. Crossing a certain error threshold will result in visual artifacts because of the incorrect perspective projection of the image in relation to the geometry which has not been culled. We use a new error metric to solve this problem. Controlled by the application of the error metric, the texture currently mapped on the virtual wall is exchanged, if the viewer moves along a path parallel to a virtual wall. The algorithm chooses an image from a set of precomputed textures to minimize the projection error. Transitions between two textures are made smoothly using transparency blending (alpha channel). If the user passes the error threshold when moving towards a virtual wall the texture map is replaced with geometry. Smooth transitions are enabled by adding blend zones near the error threshold where virtual walls and geometry are displayed simultaneously with changing transparency values.

The main contribution of our method is the extension of the powerful cell-to-cell visibility paradigm from densely occluded indoor environments to sparsely occluded indoor environments with the help of texture-based simplification. A further contribution is a new approach to minimize projection errors which occur if dynamic texture-based simplification of geometry is used in interactive systems.

In section 2, we discuss previous texture-based simplification algorithms. The remainder of the paper presents our approach in detail. In section 3, we describe the paradigm of textured virtual walls. The error metric used to control projection errors caused by textured virtual walls is presented in section 4. Sections 5 and 6 describe the preprocessing stage and the runtime phase for textured virtual walls. In section 7 we report on first results of our approach for a walkthrough of a manufacturing test scene. Section 8 concludes with a summary and future work.

2 Previous Work

Using texture-maps to replace parts of a visual database to reduce geometric complexity is a relatively new direction of research in interactive systems. To our knowledge the first system which uses this approach has been proposed by Maciel and Shirley [9]. They describe a system which is mainly based on the classical level-of-detail concept. Geometry of single objects is stored in the leaves of the scene graph hierarchy. Interior nodes of the hierarchy represent several single objects grouped together into an entity called a cluster. In general a cluster is a bounding box of a group of objects with precomputed textures mapped on each of its sides representing the geometry inside. Depending on the distance of the viewpoint either geometry or clustered objects - also called “imposters“ - are drawn. The problem of view dependency of textured imposters has been addressed by a benefit heuristic which seems to be less accurate than discussing the projection errors. Furthermore the algorithm has been applied to an outdoor scene which seems to be especially well suited for the bounding box concept. The method is not well suited for complex indoor scenes.

Schaufler [11] and Shade et al. [12] proposed methods which are very similar to each other. Both use texture-mapped quadrilaterals to represent single objects instead of groups of objects. The texture maps are not precomputed but generated from previously cached images. The view dependency problem is solved by an error metric based on the calculation of projection errors. If the current viewpoint differs too much from the texture sampling point, a better representation is displayed. The approach of Shade et. al [12] has been applied to a geometric complex but simple structured outdoor scene (island with more than 1117 trees each consisting of 36,230 triangles). This kind of scene is fairly tolerant of projection errors in contrast to complex indoor scenes of plants with many “prismatic“ objects. Within indoor scenes there are a lot of changing vanishing lines noticeable, and the viewer will probably recognize projection errors much earlier than in outdoor scenes.

After running for some time, image caching systems will cache images of images (i. e. texture maps) resulting in a constantly decreasing quality of the image displayed. Therefore after some time it will be necessary to rerender the whole geometry. Using precomputed textures eliminates this problem. Another problem of the methods discussed in [11] and [12] as well as in [9] is that these systems may suffer from pixel fill limitations. If there are hundreds of distant objects each represented by texture

Textured Virtual Walls - Achieving Interactive Frame Rates During Walkthroughs of Complex Indoor Environments

Peter Ebbesmeyer
Heinz Nixdorf Institut, University of Paderborn, Paderborn, Germany
pe@hni.uni-paderborn.de, http://wwwhni.uni-paderborn.de/vr

Abstract

We present a new approach for using texture mapped quadrilaterals as approximative representations for objects that are far away from the viewpoint. The method is suited for interactive visualization of complex indoor environments such as CAD models of large plants. In a preprocessing stage the 3D model is partitioned by virtual walls. These virtual walls are simple quadrilaterals which divide a large room into a set of separated cells. During the walkthrough phase the system only renders the geometry of cells surrounding the current viewpoint. All distant geometry is culled and replaced by "textured virtual walls" representing the same part of the model as the culled geometry. A description of techniques for minimizing visual artifacts and for controlling the transitions between textures and geometry if the viewpoint moves towards a virtual wall is given. The approach makes extensive use of texture-mapping hardware. It considerably reduces the number of polygons rendered by the 3D graphics pipeline and therefore contributes to achieve interactive frame rates.

1 Introduction

Interactive walkthroughs of models with very high geometric complexity are becoming increasingly important. Many rendering algorithms have been developed so far which reduce the number of polygons rendered in each frame without affecting the image quality. There are two simple basic ideas on which most of the methods proposed are based:

- Do not render objects which are invisible in the current frame.
- Do not render objects with a resolution that is higher than the capabilities of the display system or the perceptional capabilities of the viewer.

Approaches based on the first idea are called visibility culling algorithms. Pioneering work in this area has been done by Clark [3] about two decades ago. Several other visibility culling algorithms have been proposed since then [2] [7] [8] [14]. Methods which take advantage of the second idea are called Level-of-detail (LOD) algorithms. Clark [3] again was the first who proposed this technique. It has been constantly refined [6] [9] since then. Methods based on visibility culling or LOD modeling require modifying the scene graph hierarchy in a certain way. Despite the necessary preprocessing steps both techniques are widely in use because they have proven to be very effective in speeding up the frame rate of interactive systems.

Unfortunately there is an increasing number of visual databases which cannot be displayed with interactive frame rates even if visibility culling and LOD modeling are excessively used. This is especially true for very complex indoor environments that originate from plant design CAD systems. Examples for such environments are the interiors of power plants, oil rigs and large manufacturing plants.

With the advent of texture-mapping hardware a new category of simplification methods has emerged in recent years: The dynamic representation of geometric complex parts of a visual database using texture maps. We present a new approach to this direction for research in interactive systems. Our method is especially suited for complex indoor environments which are commonplace in plant design. The characteristic feature of these environments is vast interiors including a large amount of geometric complex objects (e. g. pumps, turbines, pipe-assemblies, machine tools, robots, etc.) which sparsely occlude each other.

Our method consists of two stages: A preprocessing stage and a walkthrough phase. During the preprocessing step we insert a set of "virtual walls" in the model. These virtual walls are simple quadrilaterals which divide a large