Eurographics/ACM SIGGRAPH Symposium on Computer Animation (2004)
R. Boulic, D. K. Pai (Editors)

Exploiting Partial Visibility for Optimised Crowd Scene Rendering

D. Pearce, G. Ryder, R.J. Lapeer and A.M. Day

School of Computing Sciences
University of East Anglia
Norwich, NR4 7TJ

Abstract

This paper presents a new approach to optimising the rendering of large crowded scenes for use in virtual environment walk-through applications. We combine an existing impostor rendering technique with new approximated visibility techniques to display scenes populated with large numbers of virtual humans.

1. Introduction

We are concerned with the efficient display of large crowded environments. Aubel et al. [1] optimised the rendering using an image based rendering approach. However, computation effort scales with scene complexity with their approach. We adopt the method used by Tecchia et al. [5], with the addition of output sensitivity to alleviate scaling problems.

Output sensitive rendering has been widely applied to static scenes [2]. However, to the authors' knowledge, efficient output sensitive rendering is yet to be achieved in crowded scenes. Visibility determination has been an area of interest since the start of the computer graphics field, initially to solve the Hidden Surface Removal (HSR) problem. We focus on an approach which efficiently rejects invisible geometry before submission to the graphics pipeline, where HSR is performed by the z-buffer.

2. Rendering Virtual Humans

We adopt the technique presented in [5] for rendering, where images of the virtual human are captured and stored in memory for angles around the object for each frame in the animation. The number of images stored due to camera samples captured is a tradeoff between accuracy of impostor representation to actual model, and memory overheads. Through the use of multi-pass rendering with differing alpha test values, we blend colours into the textures at run-time to give the illusion of visual variety with a single model.

3. Partial Visibility Data Structures and Concepts

A partial occluder occupies a volume of space which restricts visibility through the volume by some amount, known as the solidity value (SV). This information is used during the visibility approximation algorithm to approximate a visibility value (VV) of each object or subspace in the scene. A shadow volume (or area in 2D) is computed for each partial occluder, given an observer position. The portion of the scene which lies inside a partial occluder's shadow volume is said to be partially occluded by the partial occluder. For a partially occluded object, or subspace of the scene, a visibility value (VV) is approximated. The VV of an object is the percentage of the object that is visible from a given observer position.

3.1. Data Structures and Management

Given a computer model, the data structures are initially constructed at a pre-processing phase, before any visibility computations are performed. The following sections discuss the underlying data structure construction phase and describe the run-time management required.

3.1.1. Occluder Synthesis

This phase analyses the scene and synthesises suitable partial occluders. A set of cells that have desirable visibility properties are identified. Our implementation of this phase uses a quad-tree data structure to recursively divide the scene until a desired cell size or number of objects per cell is achieved. The contents of the cell are then projected onto its boundary, and a SV is approximated for the cell as the percentage of the boundary that is occupied by these projections. As the virtual humans are free to move from cell to cell, cell containment information must be updated to identify virtual humans that have moved from one cell to another. The SV of each cell must also be updated.

3.1.2. Adjacency Graph

The scene is organised into a spatially subdividing data structure which is similar to the work by Teller [6] where spatial cells are connected to neighbouring cells by portals to create a structure known as an adjacency graph. In our structure we distinguish between two different spatial cells, a cell

is either a free cell (containing no useful partially occluding geometry), or a partially occluding cell (containing sufficient useful partially occluding geometry). An adjacency graph connects all neighbouring cells. This graph is computed once as a pre-process and dynamically updated only when the size of a partial occluder changes. This structure allows efficient traversal of the subspaces of the scene during run-time.

4. Visibility Computation

Our run-time visibility method approximates the visibility information for a given observer position. The visibility information is updated each time the observer's position is changed beyond a given threshold. We use a breadth first search of the adjacency graph data structure to encounter spatial cells in a front to back order. As a partially occluding cell is encountered, it is tested with the view frustum. If the partial occluder lies inside the view frustum, a new shadow volume is calculated, and the partial occluder is added to the back of a queue. We call this the Encountered Partial Occluder queue (EPO). As a new target cell is encountered the EPO set is used to approximate the degree of visibility of the target as follows: the visibility of a target cell is calculated using all shadowing volumes, considering the solidity of the shadow casting partial occluder. The percentage of the target cell that lies inside a shadow volume is multiplied by the SV of the partial occluder that casts the shadow, and the visibility of a target is calculated in this way considering all shadow volumes encountered on the traversal before reaching the target object.

5. Rendering Phase

In [3, 4] we described how a VV can be used to speed up the rendering of a natural environment. This involves reducing the number of rendering primitives sent to the graphics pipeline based on the visibility of each object. The objects that are completely visible are sent to the graphics hardware in full detail, while those that are less visible use a coarser representation, typically consisting of fewer rendering primitives. Additionally, objects which have a VV that is sufficiently low are culled from the rendering process completely. Heuristic threshold values are used to determine when LOD switches are performed and objects are culled from the rendering phase. Adjusting these heuristics allows a trade off between image quality and rendering performance. More specifically, before rendering the scene database, each object in the database is placed into a set that determines how it will be rendered based on the VV of the object from the observer's current location. The rendering of each object is determined by a number of heuristics: a number of LOD thresholds, and a cull threshold.

Figure 1 shows rendering times of scenes rendered using the method proposed in [1], combined with view frustum culling, (light line). Rendering times are also shown of the method proposed in this paper dark line.

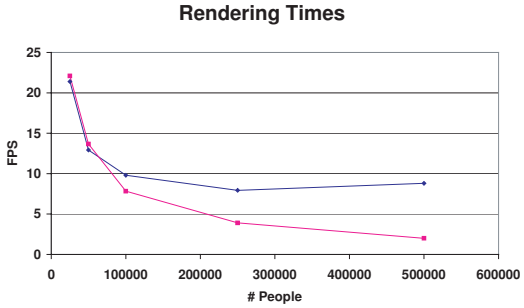

Figure 1: *Dark (upper) line - using the presented approach. Light lower line - our implementation of the approach presented in [1].*

6. Conclusions and Further Work

A new approach to approximating visibility calculations has been proposed that allows visibility information to be obtained at run-time with negligible CPU overhead. This technique has been applied to rendering dynamic crowd scenes to achieve output-sensitive rendering, allowing large scenes to be displayed at interactive frame-rates when viewed from ground level.

In this paper focus has been paid to the efficient rendering of large crowded environments using the proposed partial visibility techniques. There is still much work to be done to the area of efficient management of such large scenes as computation of human behaviour can become very demanding with large numbers of virtual humans.

References

[1] A. Aubel, R. Boulic, and D. Thalmann, *Real-time display of virtual humans: Levels of detail and impostors*, IEEE Transactions on Circuits and Systems for Video Technology (2000).

[2] C. Silva D. Cohen-Or, Y. Chrysanthou and F. Durand, *A survey of visibility for walkthrough applications*, IEE Transaction on Visualisation and Computer Graphics (2003).

[3] A. M. Day D. Pearce, *Exploiting partial visibility approximation in the real-time display of natural environments*, Theory and Practice of Computer Graphics (2003), p. 66–73.

[4] _____, *Partial visibility for virual reality applications*, WSCG 2004, Plzen, Czech Republic (2004), p. 205–212.

[5] C. Loscos F. Tecchia and Y. Chrysanthou, *Visualizing crowds in real-time*, Computer Graphics Forum **21** (2002), no. 4, p. 753–765.

[6] S. J. Teller, *Visibility computations in densely occluded polyhedral environments*, Ph.D. thesis, University of California at Berkeley, 1992.