Chapter 19
Interactive Rendering of Indoor and Urban Environments on Handheld Devices by Combining Visibility Algorithms with Spatial Data Structures

Wendel B. Silva
University of Utah, USA

Maria Andréia F. Rodrigues
Universidade de Fortaleza – UNIFOR, Brazil

ABSTRACT

This work presents a comparative study of various combinations of visibility algorithms (view-frustum culling, backface culling, and a simple yet fast algorithm called conservative backface culling) and different settings of standard spatial data structures (non-uniform Grids, BSP-Trees, Octrees, and Portal-Octrees) for enabling efficient graphics rendering of both indoor and urban 3D environments, especially suited for low-end handheld devices. Performance tests and analyses were conducted using two different mobile platforms and environments in the order of thousands of triangles. The authors demonstrate that navigation at interactive frame rates can be obtained using geometry rather than image-based rendering or point-based rendering on the cell phone Nokia n82.

INTRODUCTION

Interactive rendering of indoor and urban environments is a crucial problem and still a challenge for the field of computer graphics. Basically, the visualization pipeline describes the essential steps to generate an image from data stored in memory. The pipeline may be broken into four critical stages: retrieval from storage, processing in main memory, rendering, and display. It has to be computed for every frame to visualize an interactive motion (Akenine-Möller, Haines, &
Hoffman, 2008). The possible frame rate depends mainly on the 3D data. Several are the factors that are influential in this problem, such as: number of objects in an environment and its geometric complexity, object distribution, object dynamics, level of realism of the objects, number of pixels to be painted, etc. For example, the resulting rendering time usually grows with the number of triangles in the scene.

The display of 3D interactive graphics requires that each polygon is clipped against the viewing frustum (Cohen-Or, Chrysanthou, Silva, & Durand, 2002). In other words, rendering the objects that lie outside the field of view of the camera would be a waste of time since they are not visible on the screen. The fact that the users have control over their movements around the 3D environment means that rendering has to happen instantaneously in response to user interaction. As we know, real-time rendering relies on the graphics processors on the local execution platform to make all the rendering calculations. So, it can make some significant demands on the users’ software and hardware capabilities.

Visibility algorithms are used to speed up image generation by rejecting invisible geometry before actual hidden-surface removal is performed. Visibility is not a trivial problem since a small change in the viewpoint might cause large changes in the visibility (Cohen-Or, Chrysanthou, Silva, & Durand, 2002).

The challenge of optimizing visibility algorithms for interactive rendering of polygons becomes even more complex when the execution platform is a handheld device (non-interactive graphics can be simply rendered as simple bit-maps). Actually, implementing fully 3D graphics on handheld devices is particularly difficult because the majority of them, that form the largest part of the market share, still present important limitations when compared to traditional personal computers: low processing power; little storage memory; lack of dedicated software support; etc. Also, there is an order of magnitude difference between high and low-end mobile devices in graphics processing and computational capacity (Pulli, Aarnio, Miettinen, Roimela, & Vaarala, 2008; Capin, Pulli, & Akenine-Möller, 2008). Moreover, although some hardware characteristics are similar between our handheld devices of today and our personal computers from the nineties, there are some important and distinct features too, for example, the velocity of memory information allocation.

While some research has been done on the use of handheld computer technology, the full potential within the context of interactive rendering is still unknown. A visibility algorithm, for example, may run efficiently in one handheld device but be inefficient on another. Hence, there is an evident demand for proposals of optimization that propitiates the efficient use of the different technological resources available in each type of mobile device, in such a way as to ensure the generation of compact and simultaneously realistic implementations. This means that new studies and algorithms are required. The key question is to know what the best choice of visibility algorithm is in this context.

In our previous work (Silva & Rodrigues, 2009) we presented a detailed view of a 3D visualization and navigation system we have implemented on handheld devices, using the Open Graphics Library for Embedded Systems (OpenGL ES API) (KhronosGroup, 2008). Our prototype system initially presented only the traditional view frustum culling algorithm and two implementations of the backface culling algorithm: one based on the traditional algorithm (Clark, 1976) and another one available in the OpenGL ES API (Vincent, 2009), associated with Octrees for geometric rendering of an indoor environment containing 6199 polygons.

In this work, we have considerably extended our system for enabling efficient 3D graphics rendering of both indoor and urban environments on handheld devices. Firstly, we propose to examine several standard visibility algorithms we have implemented (view-frustum culling and