A Comparison of Spatial Subdivision Algorithms for Sort-First Rendering*

Tahsin M. Kurç, Hüseyin Kutluca, Cevdet Aykanat, and Bülent Özgüç
Dept. of Computer Engineering and Information Sci.
Bilkent University, 06533 Ankara, TURKEY

Abstract. This paper describes and experimentally evaluates three adaptive spatial subdivision heuristics for sort-first parallel graphics rendering on distributed-memory supercomputers. In sort-first rendering, image-space, or screen, is divided into regions. Each processor is assigned one or multiple regions to render. Primitives in the scene are redistributed among the processors according to region assignments. The total number of primitives in the system may increase due to duplication of primitives crossing multiple regions assigned to different processors. The common characteristic of the algorithms discussed in this paper is that a 2-dimensional mesh is superimposed on the screen and screen is subdivided using the primitive distribution on this mesh. Each processor is assigned a single region to render. The first algorithm uses summed area table for subdividing the screen. The second algorithm uses quadtrees for subdivision. The third algorithm utilizes a graph partitioning approach. The second and the third algorithms are new approaches to adaptive screen subdivision in sort-first rendering. We compare the algorithms in terms of load balancing performance, execution time of the algorithms, and percent increase in the total number of primitives after redistribution step. In this paper, we do not target a specific rendering method such as polygon rendering or volume rendering. Our load balancing criteria is the number of primitives in each processor. We present experimental results on a Parsytec's CC parallel system.

1 Introduction

Computer graphics rendering requires excessive amount of computation time and large memory space as the complexity of scenes to be rendered increases and more computation intensive algorithms are used for higher quality images. In addition, visualization of scientific data has become very popular in recent years. In such applications data sets are generally large and rendering algorithms require long execution times on single processor architectures. Distributed-memory supercomputers can provide necessary computing power and memory space for such applications. Molnar et al. [9] classify and evaluate parallel rendering approaches in three groups as sort-first, sort-middle, and sort-last. This classification is based on where in the rendering pipeline the redistribution of data is

* This work is partially supported by the Commission of the European Communities, Directorate General for Industry under contract ITDC 204-82166, and The Scientific and Technical Research Council of Turkey (TÜBİTAK) under grant EEEAG-160.
carried out. Here, we briefly describe sort-first parallel rendering. Initially each processor is assigned a subset of primitives in the scene. A pre-transformation step is applied on the primitives in each processor to find their positions on the screen. This pre-transformation step typically produces screen-space bounding boxes of the primitives. Screen is subdivided into regions and each processor is assigned one or multiple regions of the screen to perform rendering operations. In this paper, a region is referred to as a subset of pixels on the screen. This subset of pixels may form connected or disconnected subregions on the screen. The primitives are then redistributed among the processors using the screen-space bounding boxes so that each processor has the primitives crossing the region assigned to it. After the redistribution step, each processor performs geometry processing and rasterization steps of rendering pipeline on its region independent of the other processors. Primitives crossing more than one region, referred to here as shared primitives, are duplicated in the processors assigned those regions. For this reason, the total number of primitives in the system may increase after redistribution. Efficient sort-first parallel rendering on distributed-memory multicomputers requires, first of all, even subdivision of data and computations among processors. Subdivision of the screen can be done adaptively using primitive distribution over the screen for better load balancing. In addition, number of shared primitives should be minimized since such primitives result in inefficient utilization of computing power and overall memory space in the multicomputer due to duplicated computations and duplicated storage.

There are numerous approaches for sort-middle [17, 2, 4], sort-last [1, 14, 7, 8], and for sort-first [10, 16] parallel rendering. Mueller [10] investigates the sort-first rendering approach for interactive applications in polygon rendering. He uses an adaptive subdivision algorithm, referred to as mesh-based adaptive hierarchical decomposition (MAHD) algorithm, based on summed area table [3] generated from 2-dimensional mesh superimposed on the screen. In our earlier work [16], spatial subdivision is investigated for direct volume rendering. In that paper, 1-dimensional data structures, in particular arrays for each dimension of the screen, were used to find load distribution and to subdivide the screen space.

In this work, we present and experimentally evaluate three adaptive spatial subdivision heuristics for sort-first rendering on distributed-memory multicomputers. The common characteristic of these algorithms is that a 2-dimensional mesh is superimposed on the screen and screen is subdivided using the primitive distribution on this mesh. Each processor is assigned a single region to render. The first algorithm is the MAHD and uses summed area table for subdividing the screen. The second algorithm uses quadtree and orthogonal recursive bisection with medians-of-medians (ORB-MM) for subdivision. The third algorithm utilizes a graph partitioning approach. The second and the third algorithms are new approaches to adaptive screen subdivision for sort-first rendering. We compare the algorithms in terms of load balancing performance, execution time of the algorithms, and percent increase in the total number of primitives after redistribution step. In this paper, we do not target a specific rendering method such as polygon rendering or volume rendering. Our load balancing criteria is