Interactive Visibility Ordering and Transparency Computations among Geometric Primitives in Complex Environments

Naga K. Govindaraju
Michael Henson

Presented by
Rajesh Venigalla
25\textsuperscript{th} Feb 2010
Objective

• The problem of computing visibility ordering is important in many Interactive 3D graphics applications.

• So this paper aims at computing the Visibility Ordering of Complex 3D Objects in graphics at interactive rates.
Existing Visibility Ordering Algorithms

• Object Level Ordering Algorithms
  – These algorithms perform ordering of primitives in object space

• Image space Algorithms
  – These algorithms perform ordering of fragment of primitives in the viewport.
Limitations of Object Level Ordering

- Expects objects to be simple or primitive
- Needs preprocessing
Limitations of Image Space Ordering

- Computes ordering among resulting pixels
- As a result these algorithms cannot be applied to some of the real-time problems like collision detection.
Key observations

• Following are the key observations in Interactive Graphics applications
  – Nearly sorted sequences.
  – Frame to Frame coherence.
Nearly sorted sequence

- A sequence is said to be nearly sorted if only few elements in the sequence are in disorder while most of the elements are already ordered.
  - If size of input sequence \( I \) is ‘N’ and the number of elements not in order is ‘Y’ then the input sequence \( I \) is said to be nearly sorted if 
    \[ Y << N \]
Frame to Frame coherence in ICG

• There is a high degree of coherence between consecutive frames in interactive applications.
• This means that the order in which objects are placed in a frame is same in the next frame to most extent and only the order of few objects might change.
Proposed sorting

- The proposed algorithm clearly exploits the above observations.
- The algorithm in each iteration separates the list of sorted sequence and adds it to a sorted list. Then it operates on the unsorted list which is supposed to be small.
- The algorithm iterates constant number of time before it sorts completely.
- Thus algorithm sorts in linear time
Proposed Sorting

• The sorting operation is done by constructing an occlusion graph of the objects in the screen space.

• A directed edge is drawn only between the overlapping objects in the screen space.
Implementation

• N-Body collision culling
  – Algorithm is modified to return two lists ‘S’ and ‘C’
  – ‘S’ represents sorted list of non-overlapping objects and ‘C’ represents objects that are overlapping or forming a cycle.
  – Algorithm is applied repeatedly by changing the viewing direction to get minimum ‘C’ value.
  – Then the exact collision detection algorithm is applied to this potentially colliding objects.
Implementation

• Order Independent Transparency
  – The algorithm is used to generate transparency effects in CAD.
  – During each frame the algorithm has to calculate the order of objects in the screen.
  – The algorithm exploits the frame-to-frame coherence by using the order of objects in the frame as an input sequence of objects to the current frame.
Limitations

• Algorithm assumes that there is sorting order among the input sequence.
• Input objects are assumed to be non-overlapping objects.
• Can not resolve cycles. Need to use some existing cycle resolving algorithms.
Other Application Areas

• Volume rendering of unstructured grids.
• Special effects generation including motion blur and depth of field generation.
• Occlusion culling.
Thank You