Depth-Presorted Triangle Lists

Ge Chen[1], Pedro Sander[1],

Diego Nehab[2],

Lei Yang[3],

Liang Hu[4]

1 香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

2 impa NATIONAL INSTITUTE FOR PURE AND APPLIED MATHEMATICS, BRAZIL

3 BOSCH Bosch Research North America

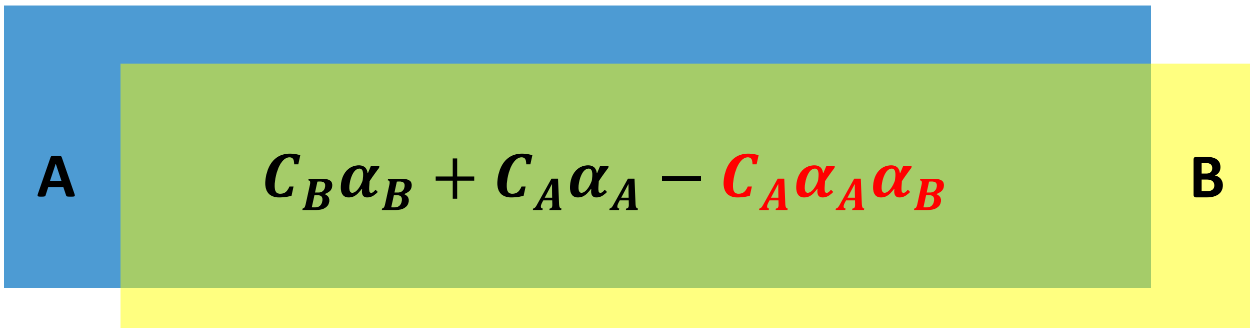
4 Google

Sponsored by ACM SIGGRAPH

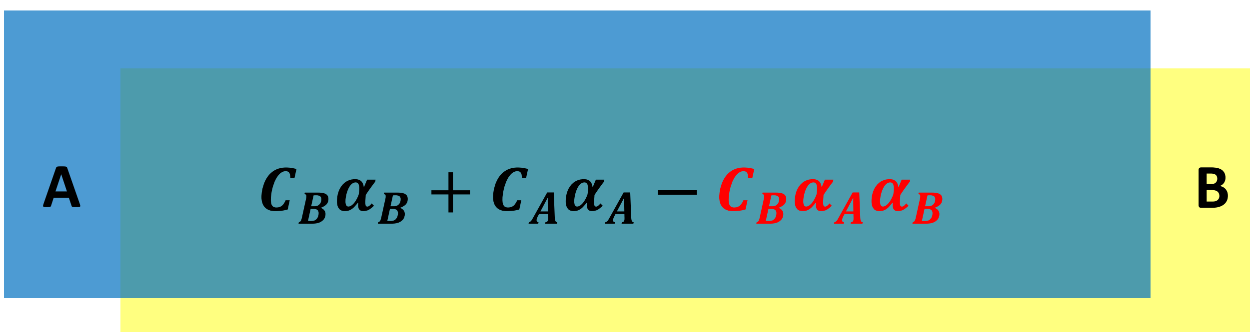
SIGGRAPH ASIA 2012

Alpha Blending

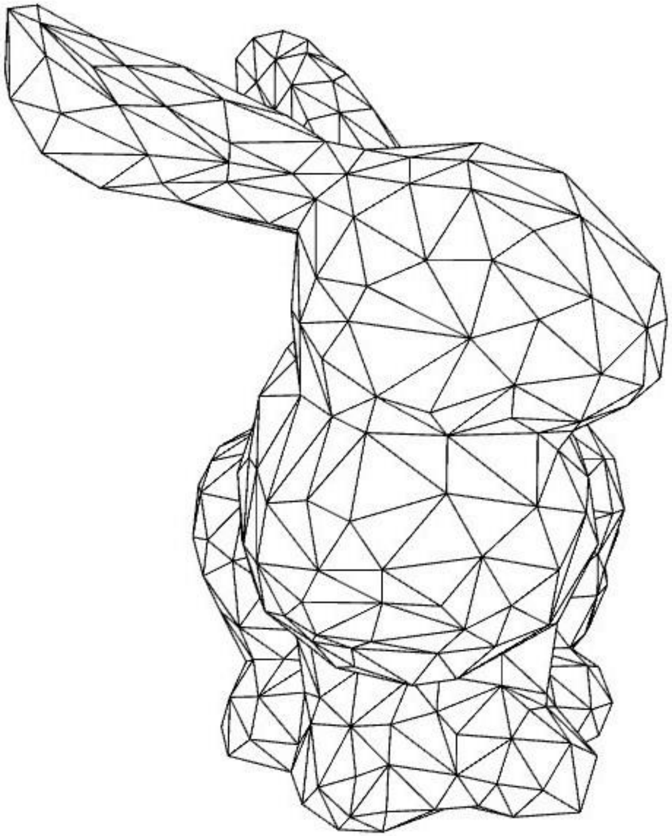
$$D_n = C_n\alpha_n + D_{n-1}(1-\alpha_n)$$



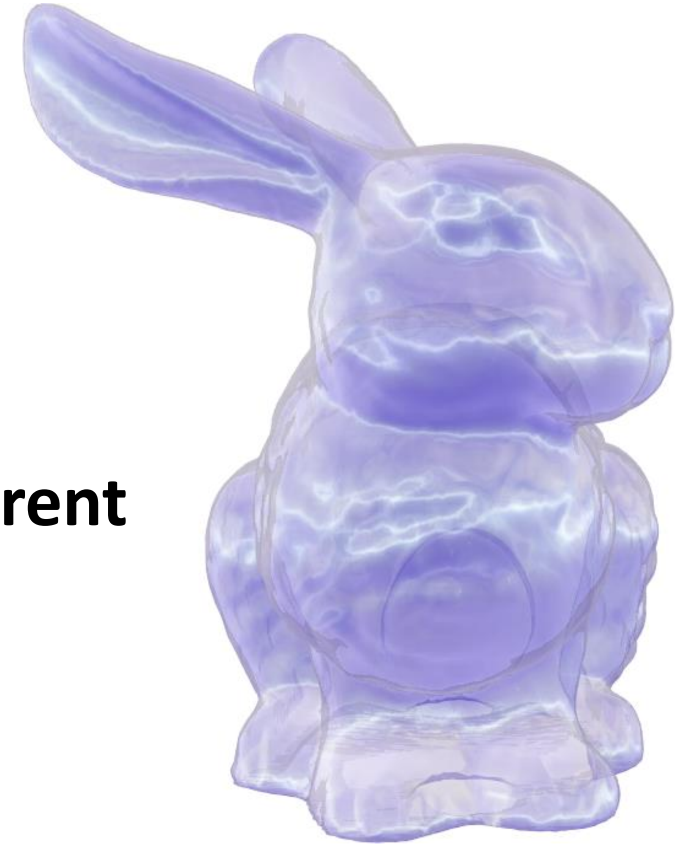
B over A



A over B



Semi-transparent



Issues:

- Fixed input order
- Large amount
- Sorting is slow
- Sort every time the scene or viewpoint is changed

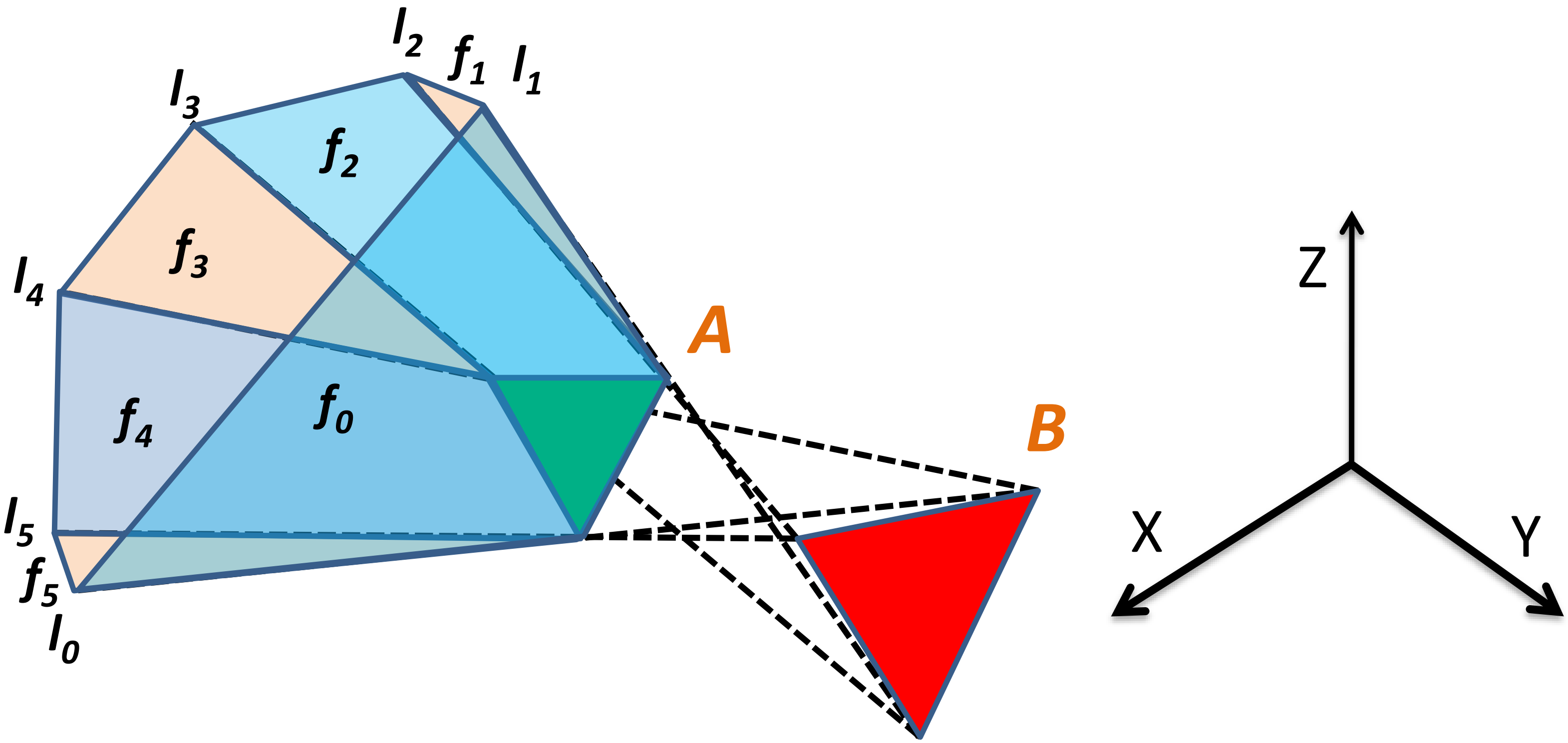
Solution:

Triangle order that applies to all view directions **without having to sort.**

Overview

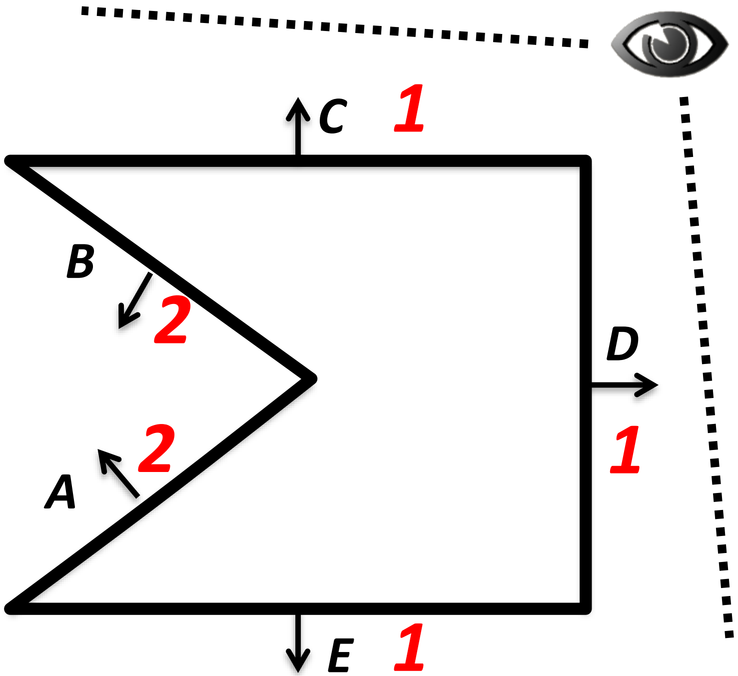
- Introduction
  - Occlusion region
  - Occlusion graph
- Depth-Presorted Triangle Lists
  - Motivations
  - Run-time Selection Algorithm
  - Preprocessing Algorithm
- Results
- Conclusions

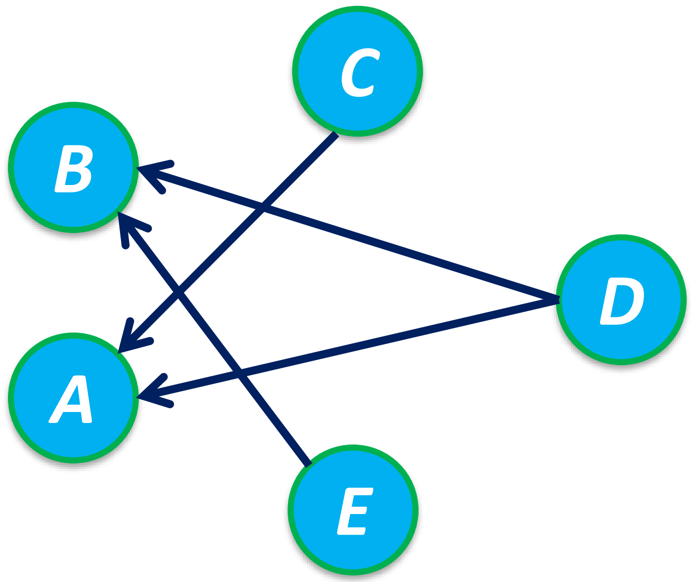
Occlusion Region



- *Occlusion region* $O_{A\to B}$ is bounded by up to six extruded planes, two triangle planes and viewpoint-space bounding planes
- Within this region, A occludes B

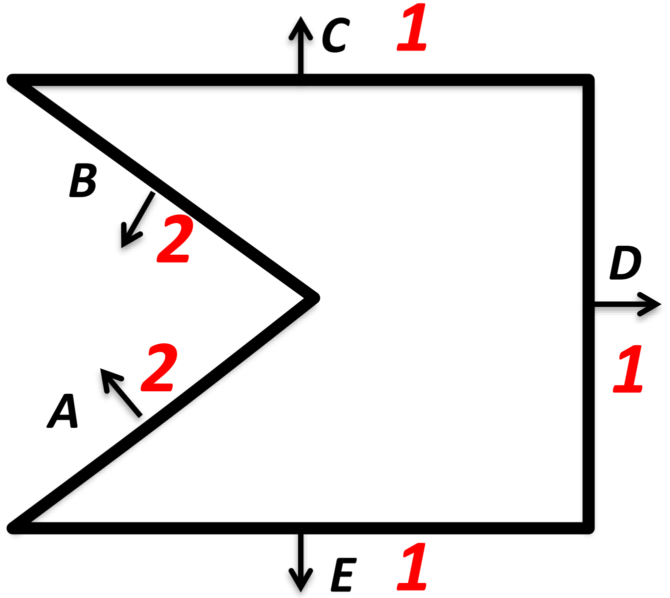
Occlusion Graph

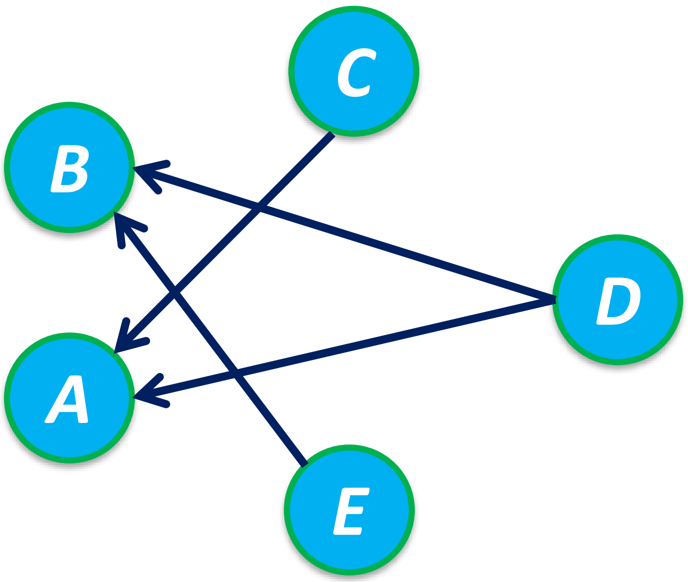




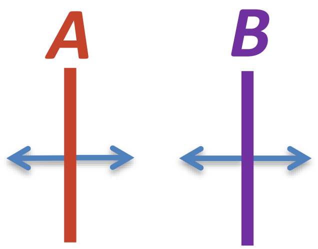
- Connects nodes if occlusion region exists
- If no cycles
  - Assign a number to each face by topological sort
  - The order of the assigned number is correct from any viewpoints

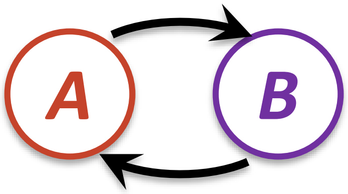
Occlusion Graph



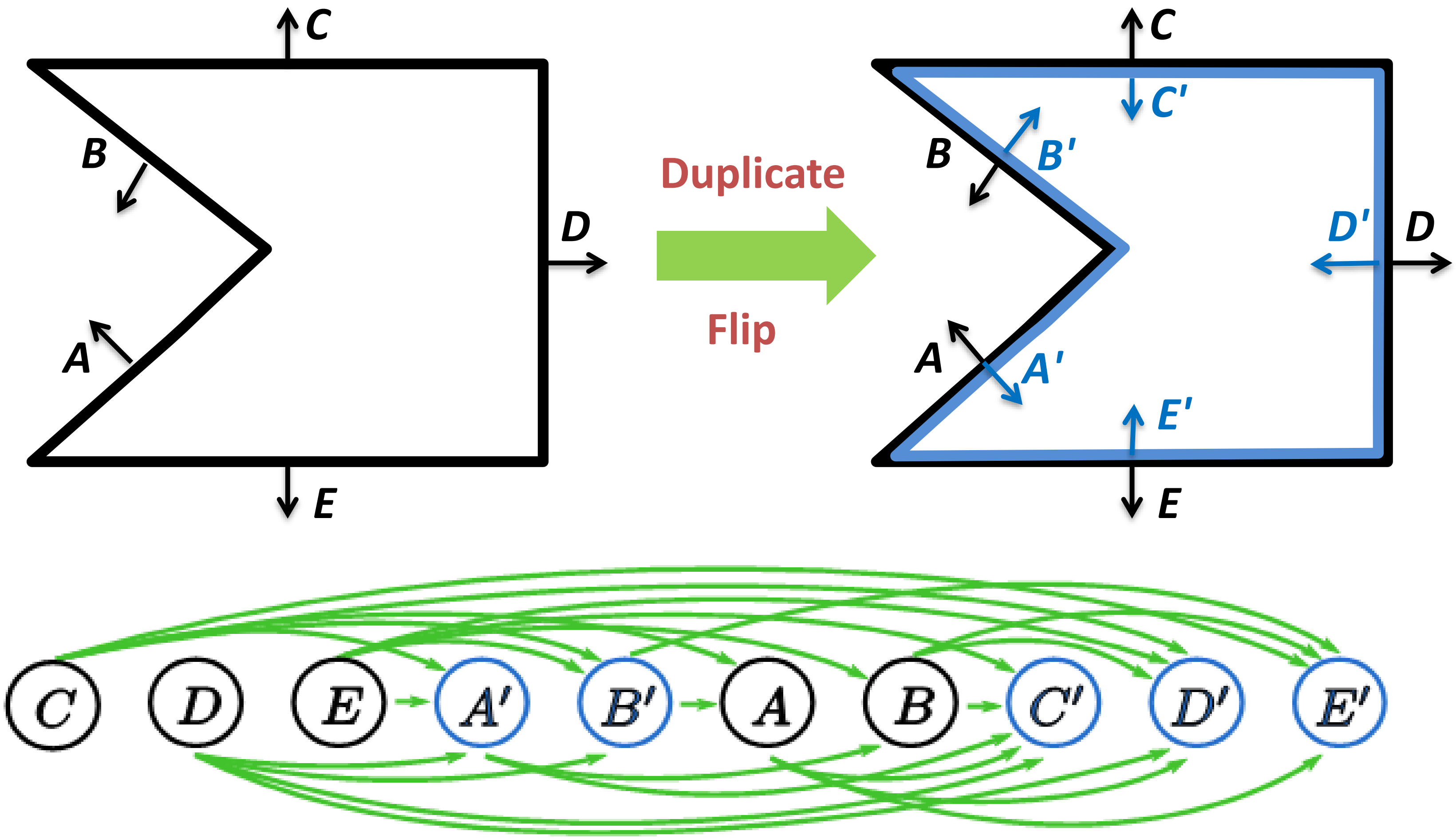


➢ Eliminating back-faces conflicts with transparency

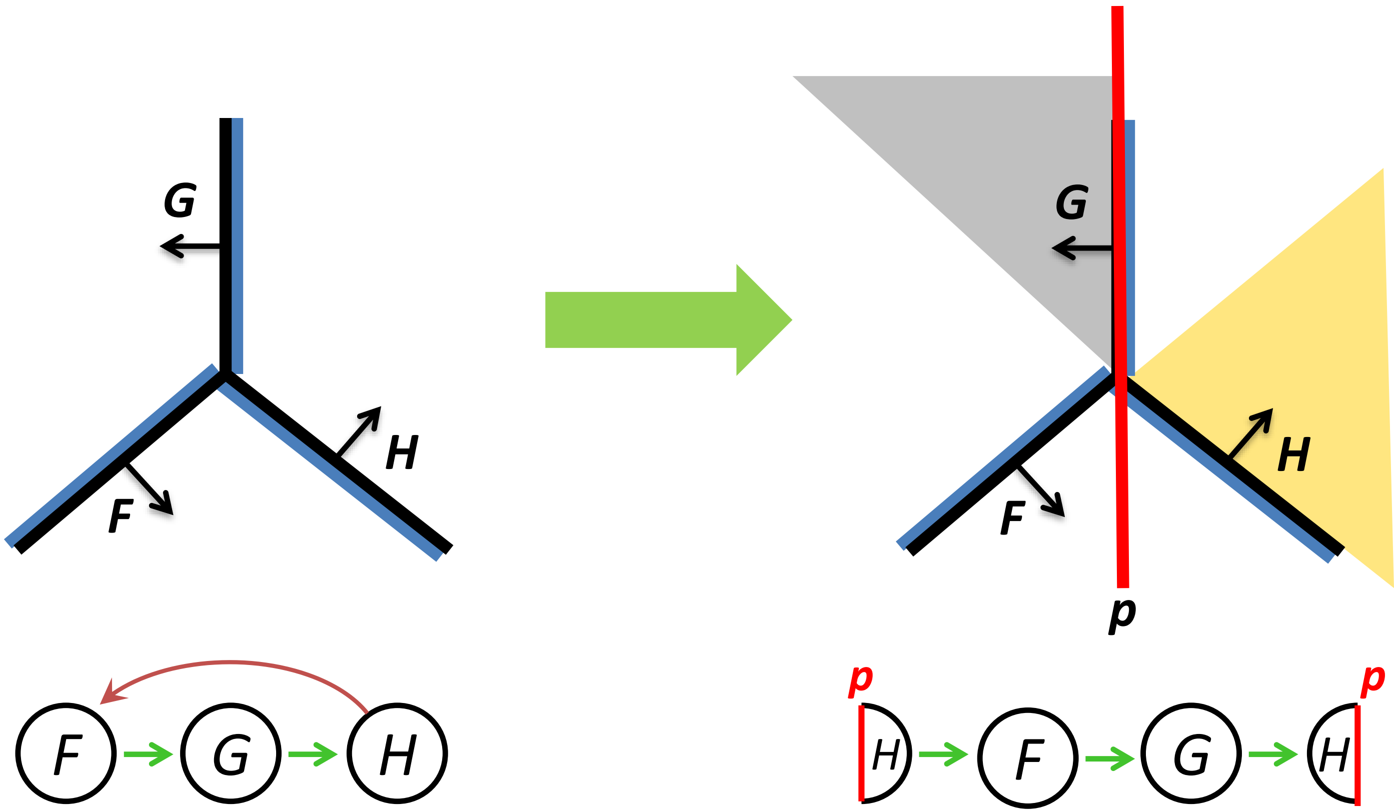




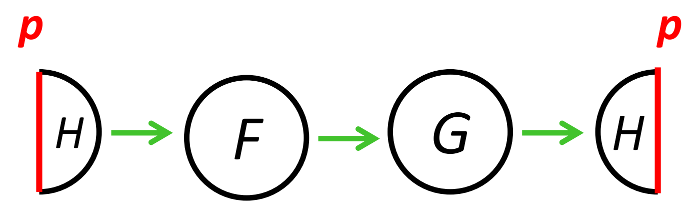
Motivation – Transparency



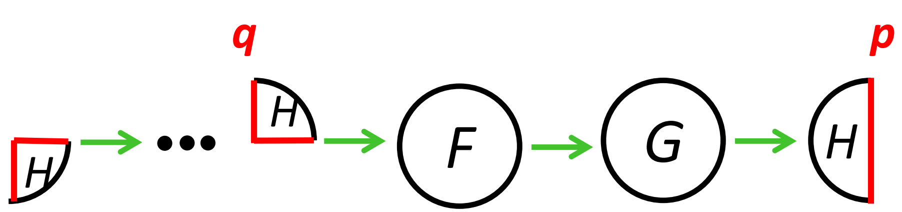
Motivation – Occlusion Cycle



Depth-Presorted Triangle Lists



- Requirements
  - ✓One draw call / triangle list
  - ✓Triangles may have multiple instances
  - ✓Associate one test plane to each triangle instance
  - ✓Accept only the first copy of all the duplicates
    - ✓Culling by Z-buffer ***less*** test
  - ✓Binary partition the rendering region for each duplicates



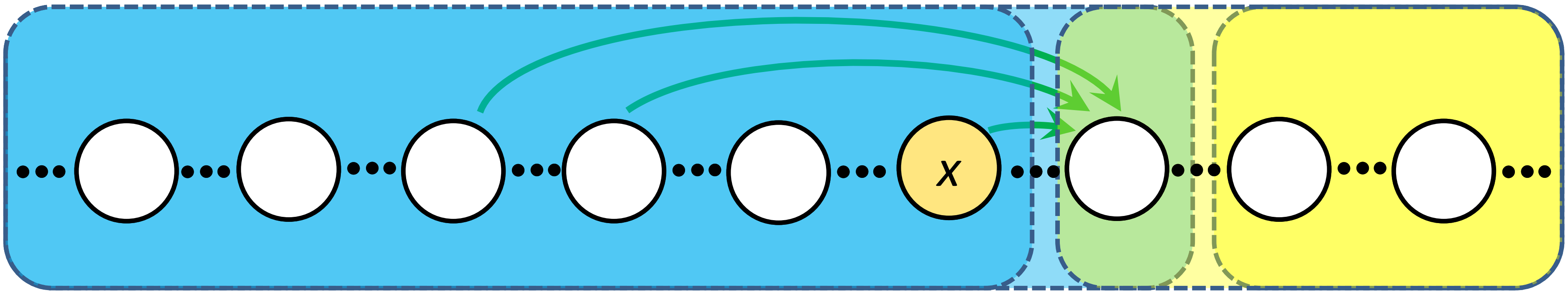
Run-time Selection Algorithm

- Each triangle is annotated by a test plane p_t
- If no associate plane, $p_t = [0,0,0 - 1]$
- At run-time, simply test $Dot(p_t, [Eye_{xyz}, -1]) > 0$
- Turn on depth test to guarantee that exactly one of the duplicates is rendered
- Plane test can be implemented in fragment shader, vertex shader or geometry shader

Preprocessing Algorithm - Outline

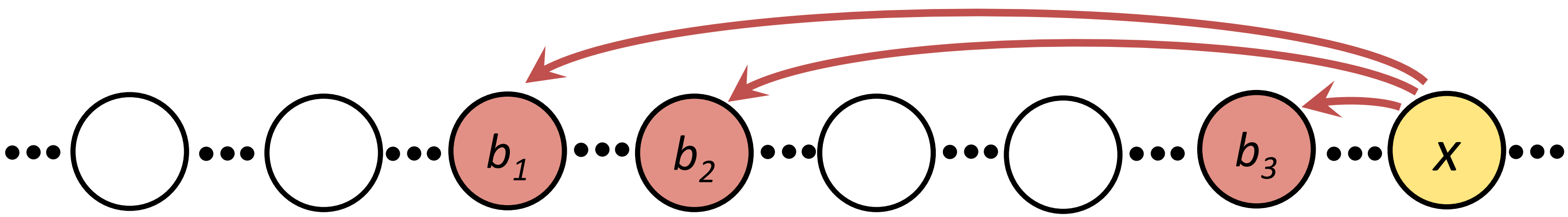
1. Create back-facing duplicates
2. Compute occlusion graph and generate a preliminary order
   - If no cycles, a topological sort is enough [Skiena 2008]
   - Otherwise, minimize num of back-edges
     - Minimum Feedback Arcset problem
3. Scan the ordering one by one
   - Operations: Keep, move, or duplicate

Preprocessing Algorithm – Keep



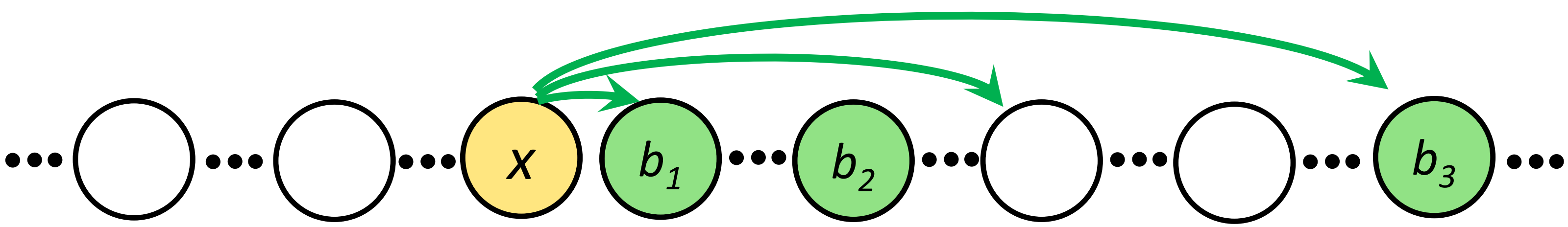
- From right to left
- Nodes (triangles) in the yellow regions are processed nodes (no longer need to consider)
- x is the current processing node
- f_* are forward-edge nodes (safe)
- If no back-edges, just keep and proceed to next node

Preprocessing Algorithm – Move

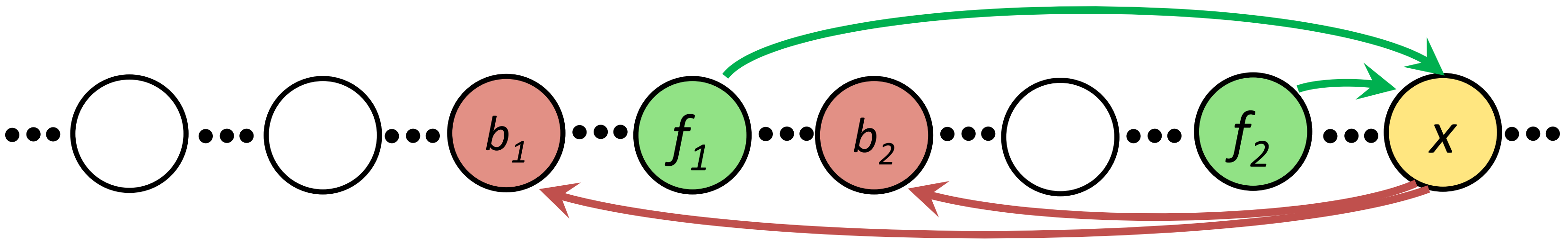


b_* are back-edge nodes (bad)

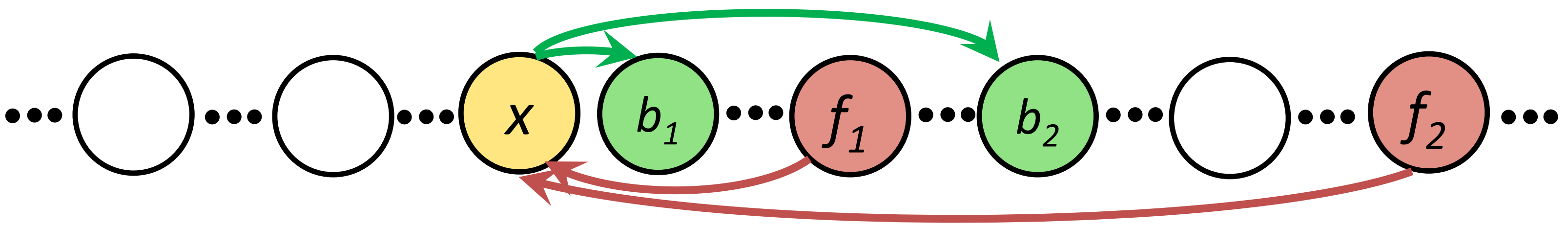
Move x directly in front of b_1



Preprocessing Algorithm – Duplicate

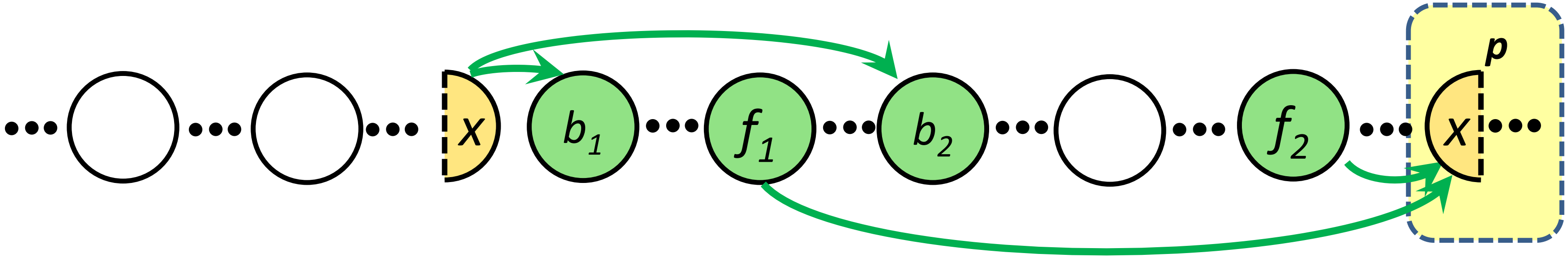


Move x directly in front of b_1 gives rise to **two new back-edges**

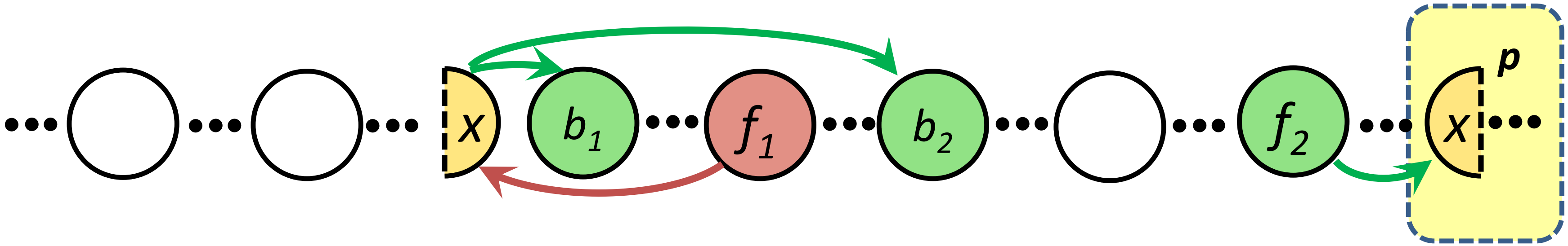


Preprocessing Algorithm – Duplicate

Find p that completely separates viewpoints associate to b_* from those to f_*



Otherwise, find p that separates as many forward-edges as possible, postpone handling new back-edges

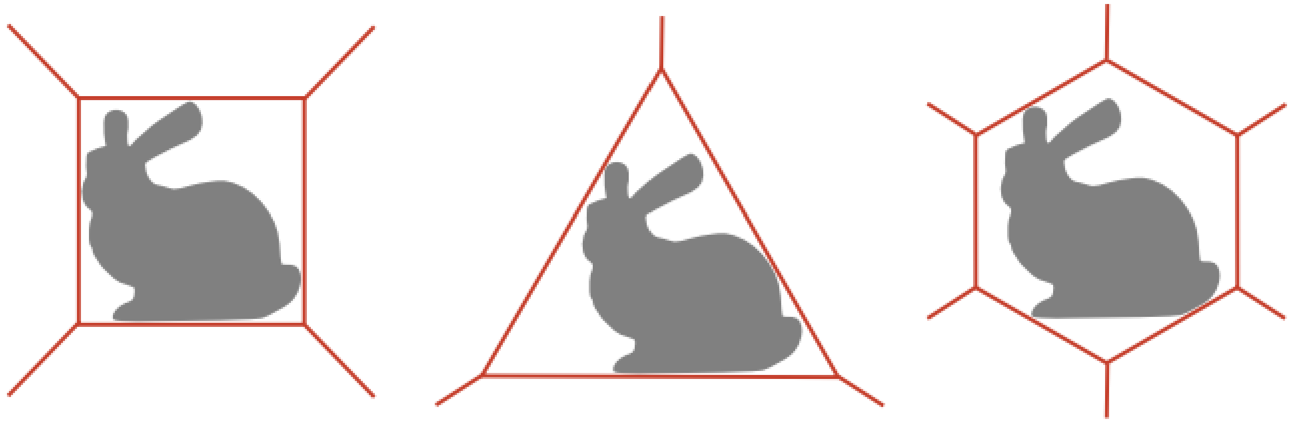


Preprocessing Algorithm

- Greedy algorithm
  - As long as we manage to separate at least one of the edges between f_* and x from ***at least one*** of edges between x and b_*, we have made progress
- How well the algorithm works depends on the choice of cutting plane p
  - Try to find a p that solves as many forward-edges as possible
- See paper for details on
  - Handling problematic cases
  - Computing p

Viewpoint-Space Partitioning

- A single depth-presorted triangle list requires far too many duplicates
- Divide viewpoint-space into several parts
  - Enclose the model in a bounding polyhedron with a given number of faces (4, 6, 16, 64)

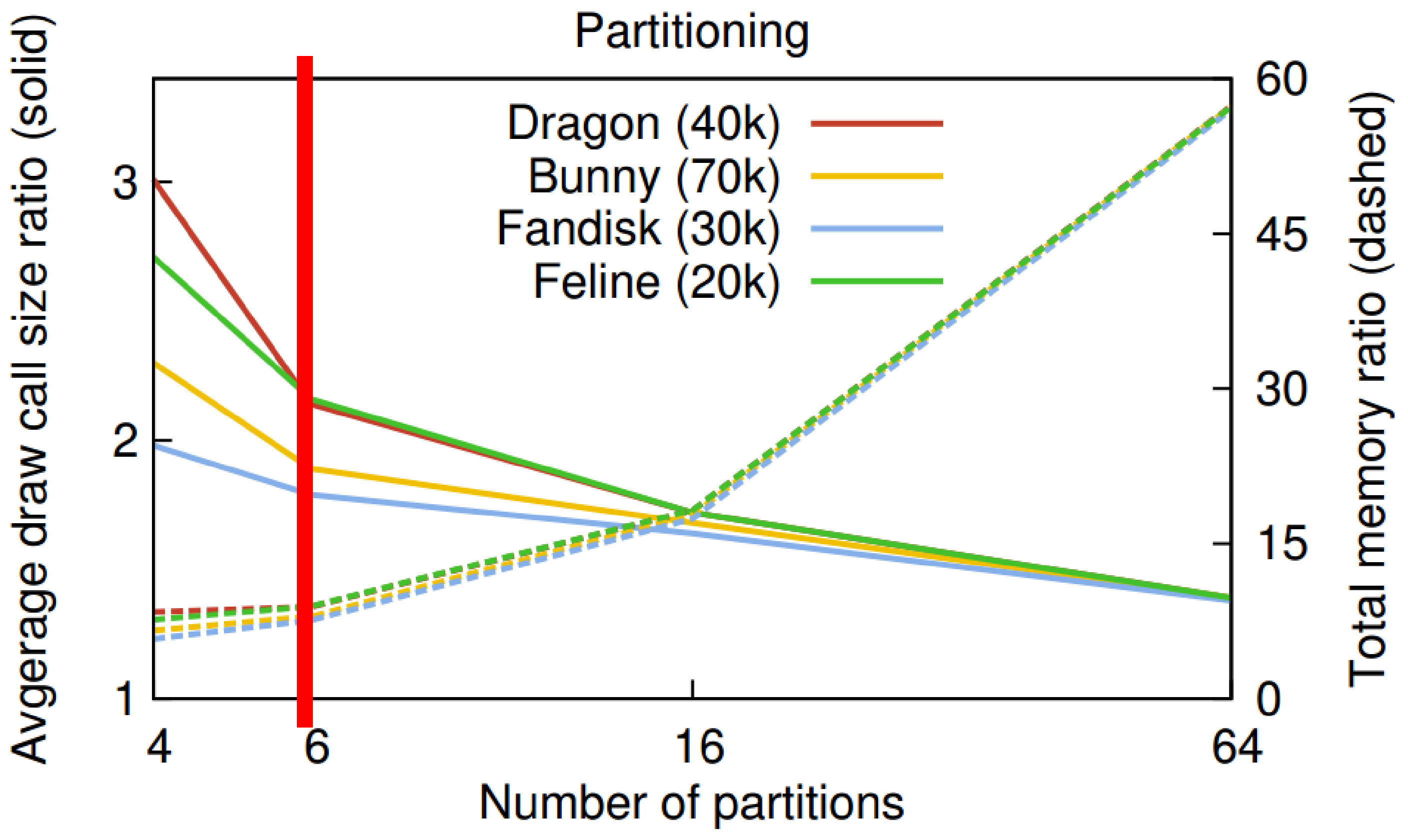


Viewpoint-Space Partitioning

- A single depth-presorted triangle list requires far too many duplicates
- Divide viewpoint-space into several parts
  - Enclose the model in a bounding polyhedron with a given number of faces (4, 6, 16, 64)
  - Restricts view-point outside bounding region
  - Further reduce the complexity of occlusion graph
  - Still a single draw call is used to render the appropriate index buffer segment

Viewpoint-Space Partitioning

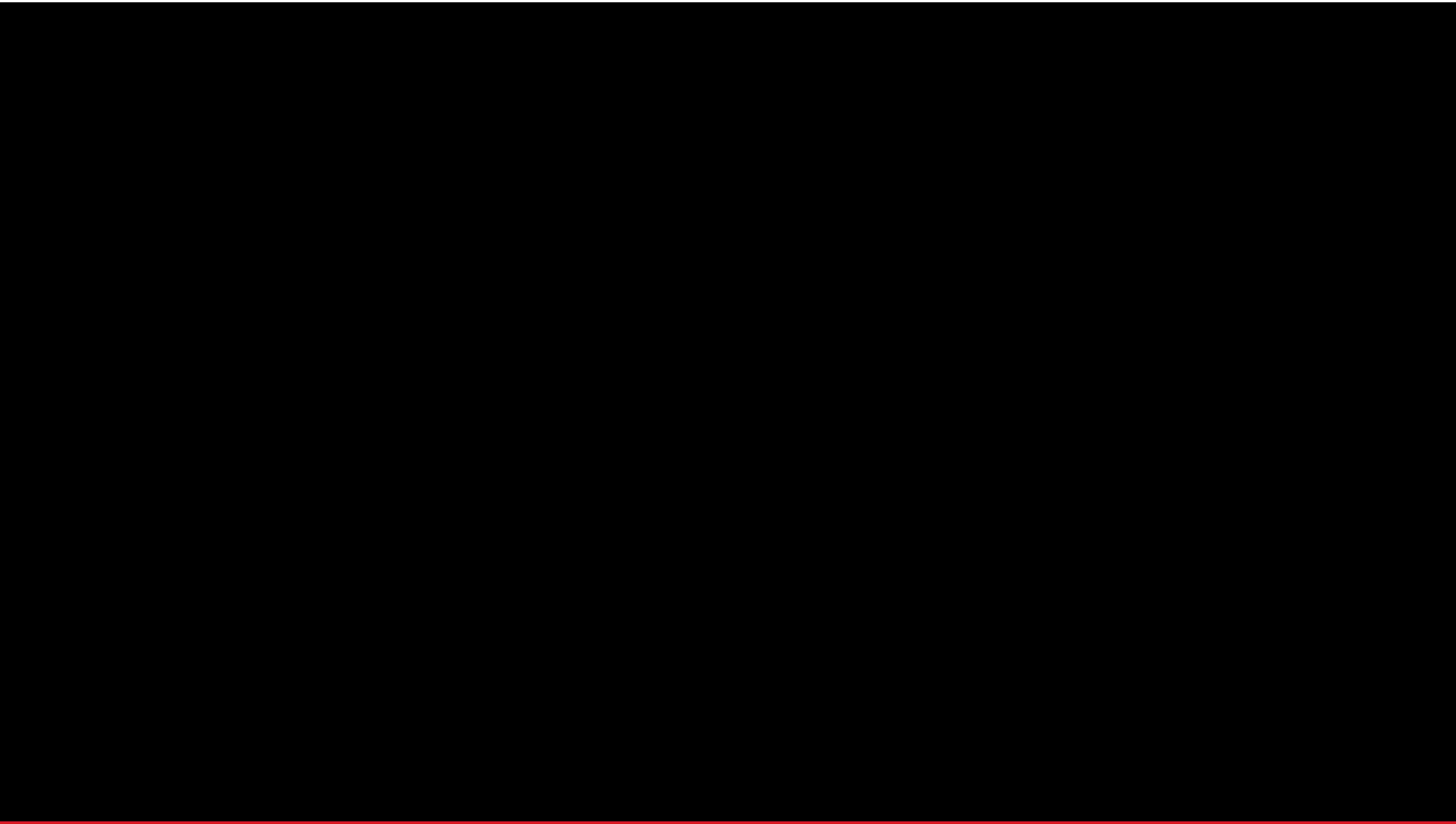
- 6 viewpoint partitions provides a good trade-off



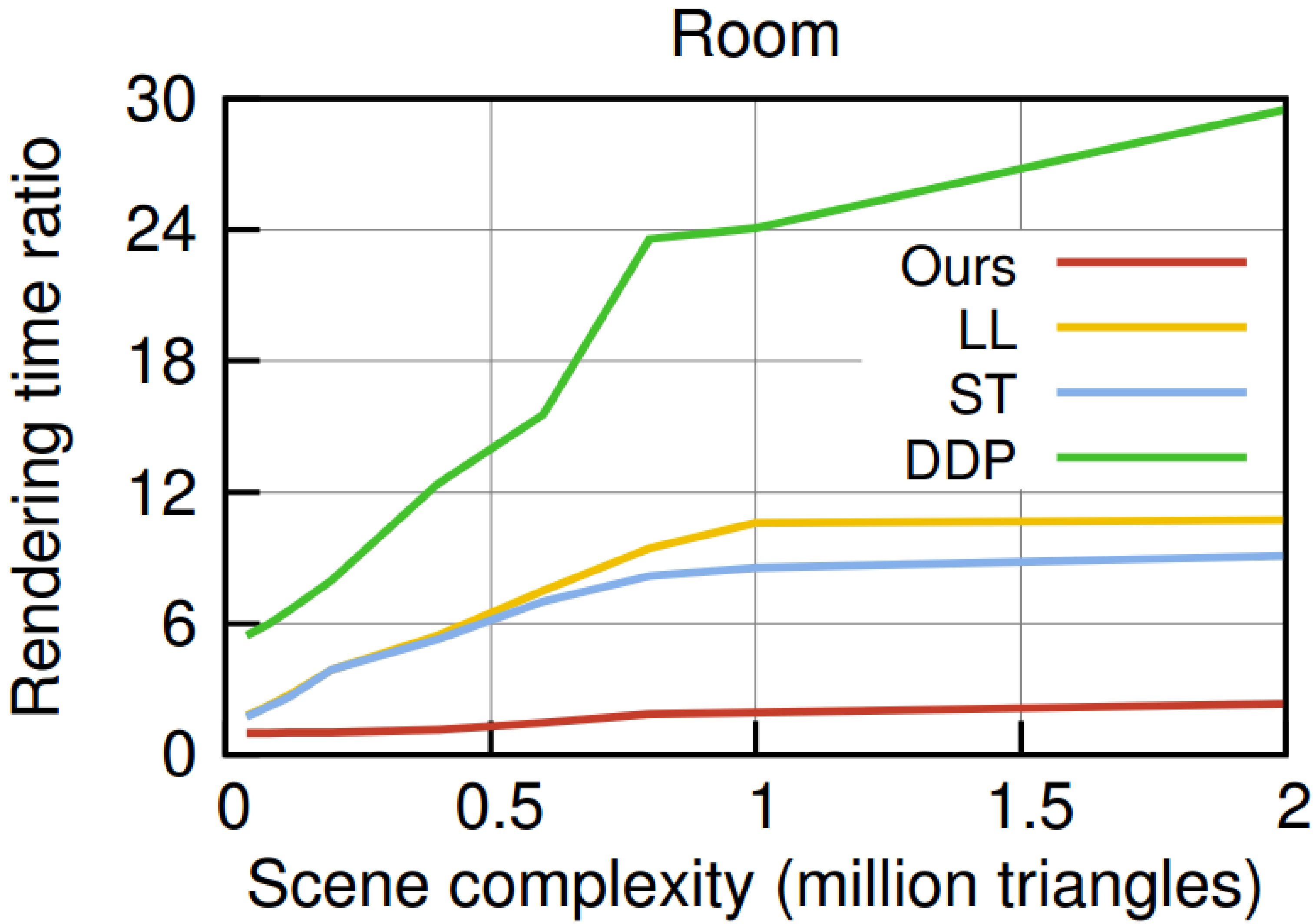
Results

- Compare with
  - LL: Per-pixel dynamic linked list
    [Yang et al. 2010]
  - DDP: Dual depth peeling
    [Bavoil and Myers 2008]
  - ST: Stochastic transparency
    [Enderton et al. 2010]
- Screen resolution:
  1280 x 720
- 4x MSAA

Results



Results


Room
Rendering time ratio
30
24
18
12
6
0
0
0.5
1
1.5
2
Scene complexity (million triangles)
Ours
LL
ST
DDP

Conclusions

- Limitations
  - Static model
  - Long time preprocessing
  - Outside of bounding polyhedron
- Advantages
  - Significantly fast in run-time
  - Simple run-time component
  - One single draw-call
  - A novel ***selection*** based scheme
- Future Work
  - Deformable objects with limited range
  - Reduce the number of duplicates
  - Speed up the preprocessing time

Thanks

- Acknowledgement:
  - HK RGC GRF grant #619509
  - INST grant from FAPERJ
  - All the students of VSGraph Lab at HKUST