An Improved Z-Buffer CSG Rendering Algorithm

Nigel Stewart, Geoff Leach and Sabu John
RMIT University, Australia

1998 EG/SIGGRAPH Workshop on Graphics Hardware

Goldfeather Methodology

Constructive Solid Geometry (CSG)
Z-Buffer Surface Parity
Tree Normalisation
Z-Buffer Surface Clipping
Optimisation - Geometric Pruning, Caching

Solid Modelling

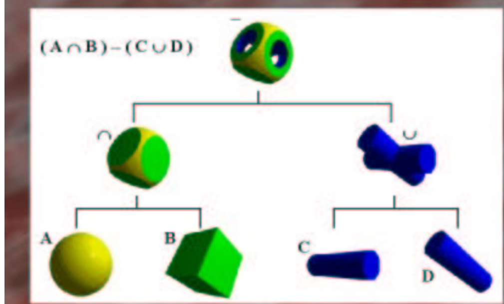
Analytic Surface Intersection
Boundary Representation (B-REP)

- Complex Mathematics
- Complex Algorithms
- Robustness issues

Image Space
- Simple Implementation
- Interactivity, Flexibility
- Slow

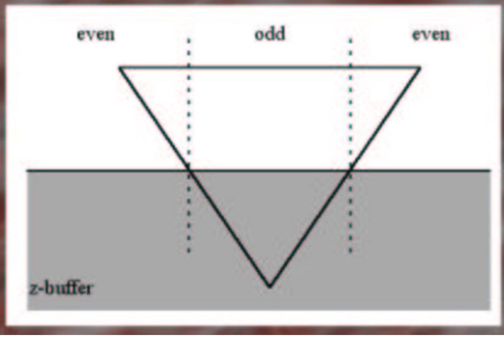
Constructive Solid Geometry (CSG)

Volumetric Objects at Leaf Nodes
Volumetric Operations at Parent Nodes
Boolean operations -
Union, Intersection, Difference



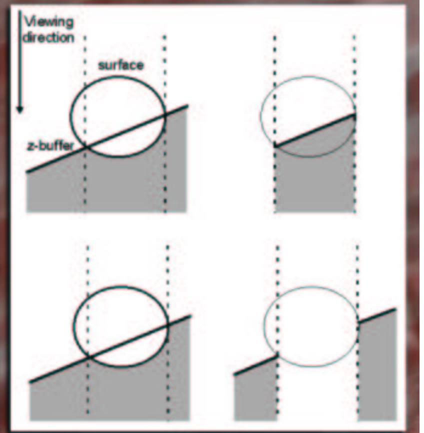
Z-Buffer Surface Parity

Counting the number of surfaces in front of z-buffer.
Detect pixels intersecting a given volume.



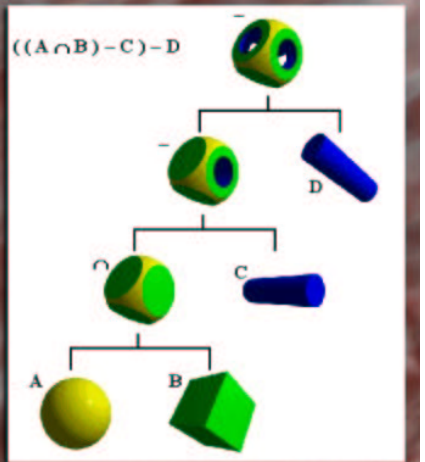
Z-Buffer Surface Clipping

Clip surface in Z-Buffer
Intersection
Difference



Tree Normalisation

Sum-Of-Products
Clipping problem



Goldfeather Algorithm

For each Surface -
- Draw into Z-Buffer
- Clip against others
- Merge into result

Two Z-buffers Required

n passes required (convex)

Goldfeather Optimisations

Geometric Pruning
- Collapse trivial cases such as A.A
- Use bounding boxes to avoid clipping

Caching
- Pre-compute invarient subtrees via b-rep

Comparitive Analysis

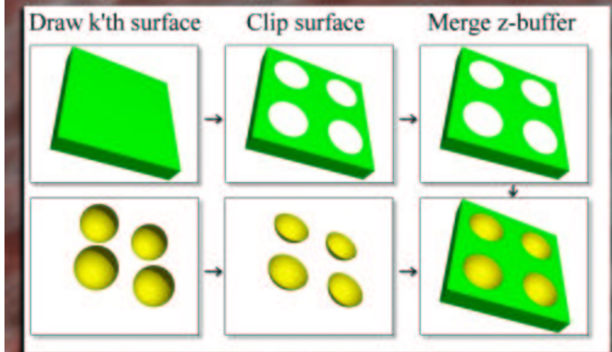
Goldfeather: n(an + b)
Clip layers: k(2an+b)

n is number of primitives
k is depth complexity
a is cost of rasterisation
b is cost of z-buffer copy

since b dominates performance,
speedup factor of n/k

A New Optimisation

Take advantage of depth complexity
Clip convex layers rather than primitives



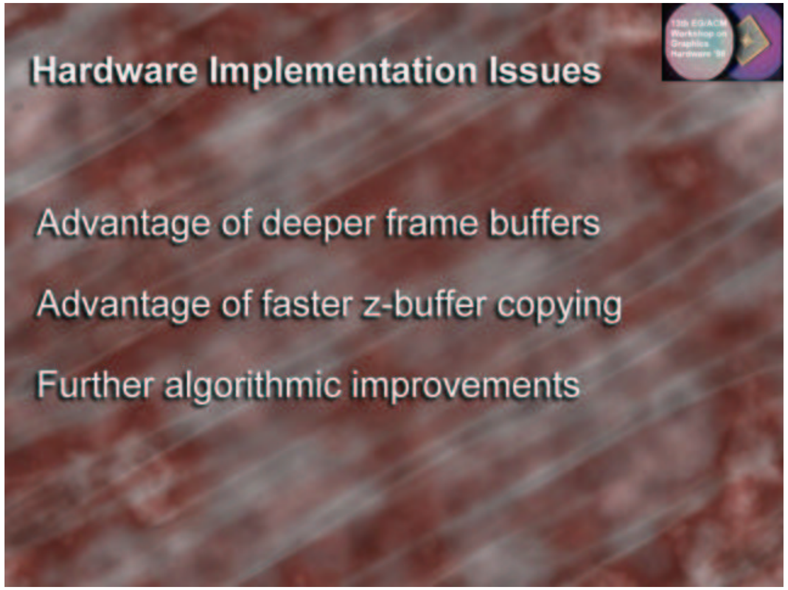
Depth Complexity

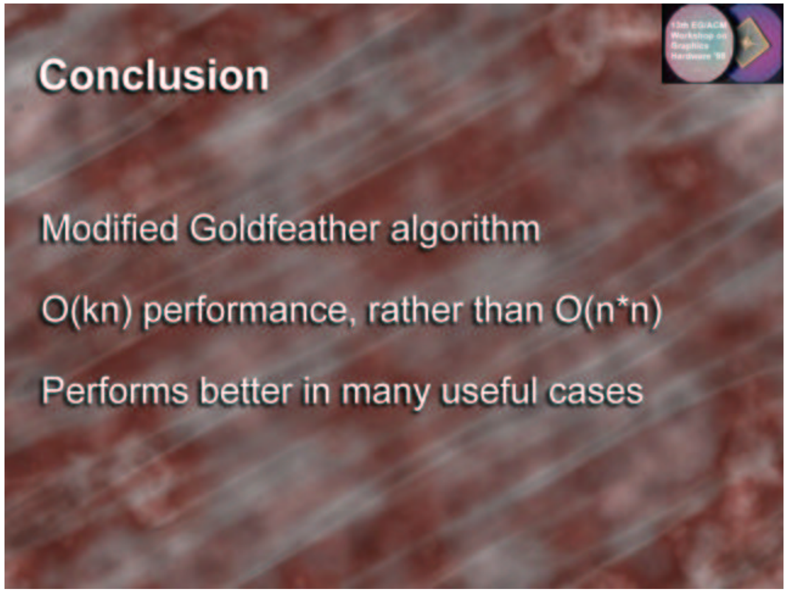
k(n) dependent on domain

k(n)<n observed
- for an NC verification application
- in nature

k(n)=n observed
- in rare (degenerate?) situations

Topic of research in own right


Hardware Implementation Issues
Advantage of deeper frame buffers
Advantage of faster z-buffer copying
Further algorithmic improvements


Conclusion
Modified Goldfeather algorithm
O(kn) performance, rather than O(n*n)
Performs better in many useful cases