An Efficient Algorithm For Hidden Surface Removal

Ketan Mulmuley
The University of Chicago
1989

http://doi.acm.org/10.1145/74333.74372

Presentation by Steve Palmer

What is Hidden Surface Removal?

- Project a 3D image onto a view plane, displaying only the surfaces which would not be obstructed from view.

Introduction

- Distinguishing features of Mulmuley's algorithm.
  - Randomized surface processing and fragment removal.
  - The “deeper” the surface, the less time spent processing it.
  - Complexity is roughly proportional to the visible output times the log of the depth.

Hidden Surface Removal

- Partition the view plane and label each partition with an appropriate face name.
- Projection of all faces establishes junctions on the view plan.
- An efficient algorithm will spend little time on invisible junctions.

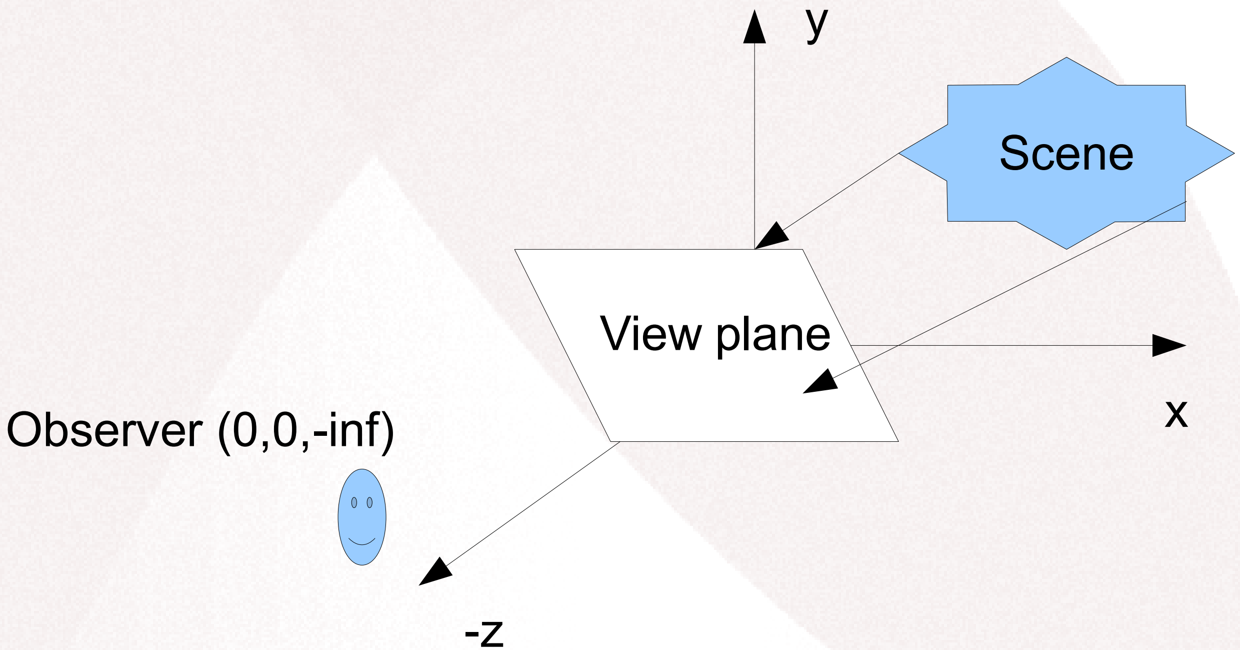
Background

- Θ Series -
  - Critical for Analysis of Mulmuley's algorithm.
  - The theta series of a lattice is the generating function for the number of vectors with norm n in the lattice.

  (http://mathworld.wolfram.com/ThetaSeries.html)

Associating Faces with a Theta Series

- Perspective projection of faces onto view plane with an observer at (0,0,-inf)



Establishing a Theta Series: Definitions

- A face, h, obstructs a junction, q, if the projection of h onto the view plane makes q invisible.
- Obstruction level, level(q) is the number of faces which obstruct q.
- V_l is the set of junctions at obstruction level l.
- V_1 is the set of visible faces.

Establishing a Theta Series

- For every real s >= 0

$$\theta(s) = \Sigma \frac{\nu(l)}{l^s}$$

  - (sum over l)

Algorithmic Implications from this Theta Series

- $\Theta(0)$ = Number of junctions in the view plane.
- Existing algorithms were linear in $\Theta(0)$.
- This paper's algorithm is linear in $\Theta(1)$.
- $\Theta(\infty)$ = Number of visible junctions
  - Open question – Can hidden surface removal be done in time that is linear in $\Theta(\infty)$ or even $\Theta(s)$?
  - Conjecture: NO

Randomization

- Hidden surface removal is a type of “sort” problem... quicksort suggests randomization strategy.
- quicksort “divide and conquer” does not translate to hidden surface removal.
- Probabalistic game theory analysis gave rise to this algorithm.

Limitations

- Since this is a general purpose algorithm, “Special situations” cannot be cheaply detected.
  - Car in front of a grass field
  - A box containing lots of items.
- Conventional heuristics should also be used
  - Clipping
  - Hierarchical comparisons

The Algorithm - Setup

- n – The number of faces
  - Non-intersecting faces assumed
  - Arbitrary shapes allowed
  - Preprocessing complete
  - Special face O is the background
- Establish Partition H_0 labeled with the background face, O.

The Algorithm – Processing

- Create partitions H_1, H_2, ..., H_n
  - Add one randomly selected face at a time.
  - Partition H_{k+1} = Partition H_k + Face f_{k+1}
- Label each region in each partition with the currently visible face.
- H_n is the final visibility partition
- Scan H_n from left to right and paint labelled faces.

The Algorithm - Partitioning

- Partition H_k is constructed from randomly chosen faces $f_0, \dots, f_k$.
- In general, edge e from a face will be partly visible, or not at all.
  - Disconnected visible parts of e are called fragments.
  - All fragments establish a partition, but shapes are complicated.
- Pass a vertical through each fragment's end points, stopping at another edge or a window boundary.
- All resultant regions are trapezoids.

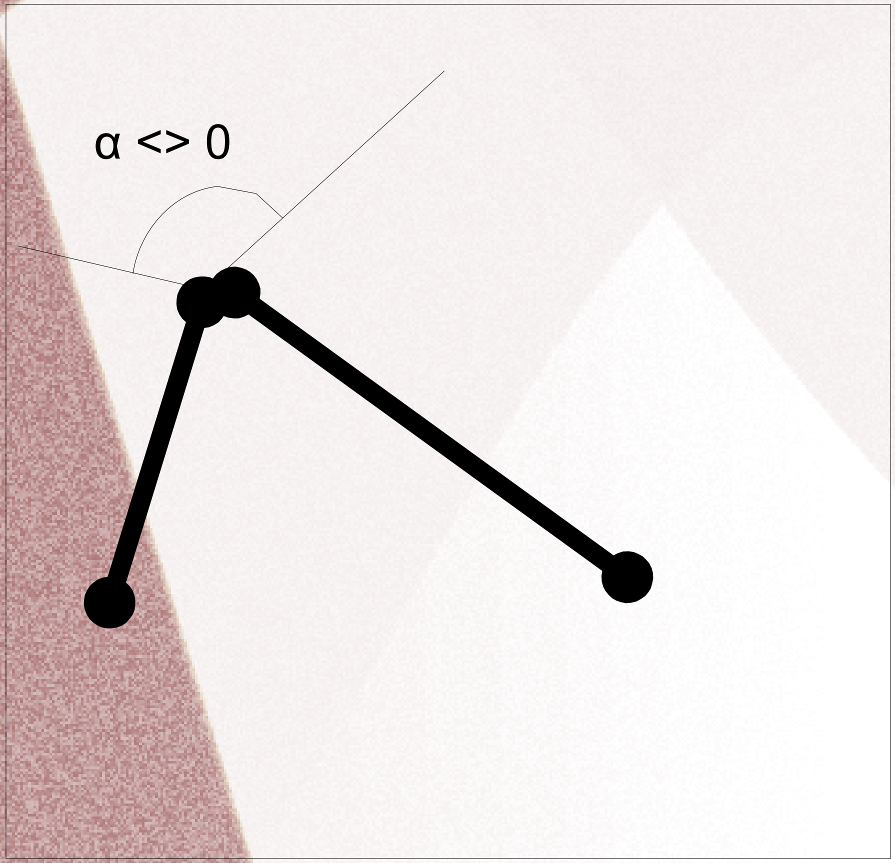
The Algorithm – One Partition

Representation

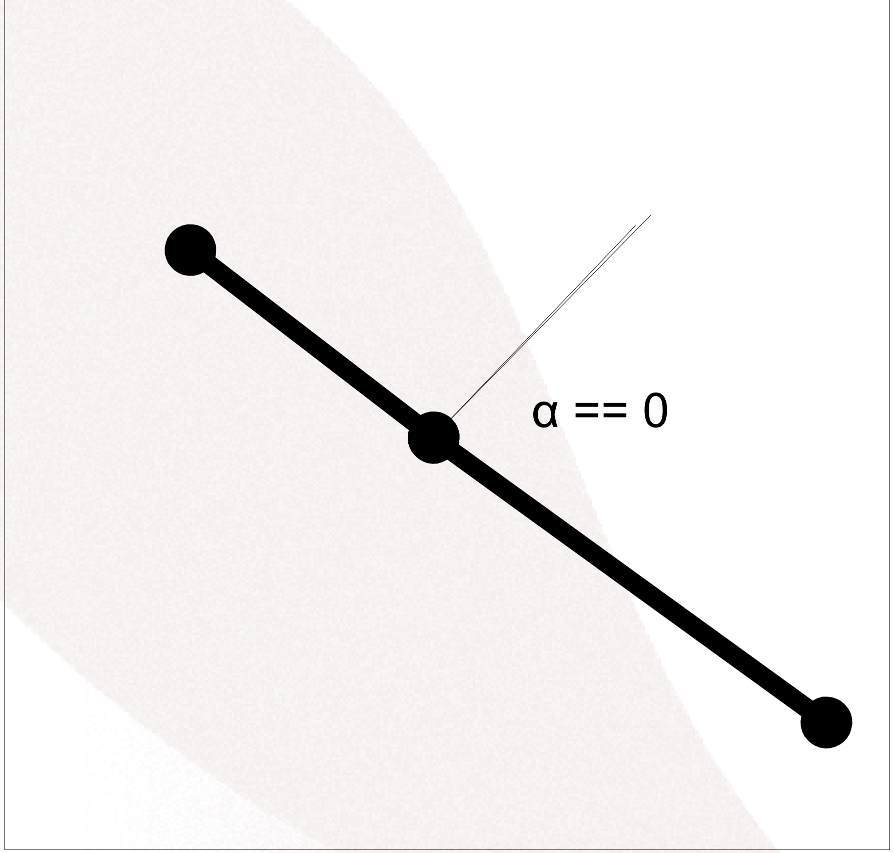
- Trapezoid:
  - Definition: A vertex v of the partition is said to be visible in the face of R if the boundary of R has a tangent discontinuity at R.
  - Each trapezoid is represented as a circular list of visible vertices
- Adjacency list:
  - List of adjacent regions in which the vertex is visible.

Tangent Discontinuity

- Tangent Discontinuous

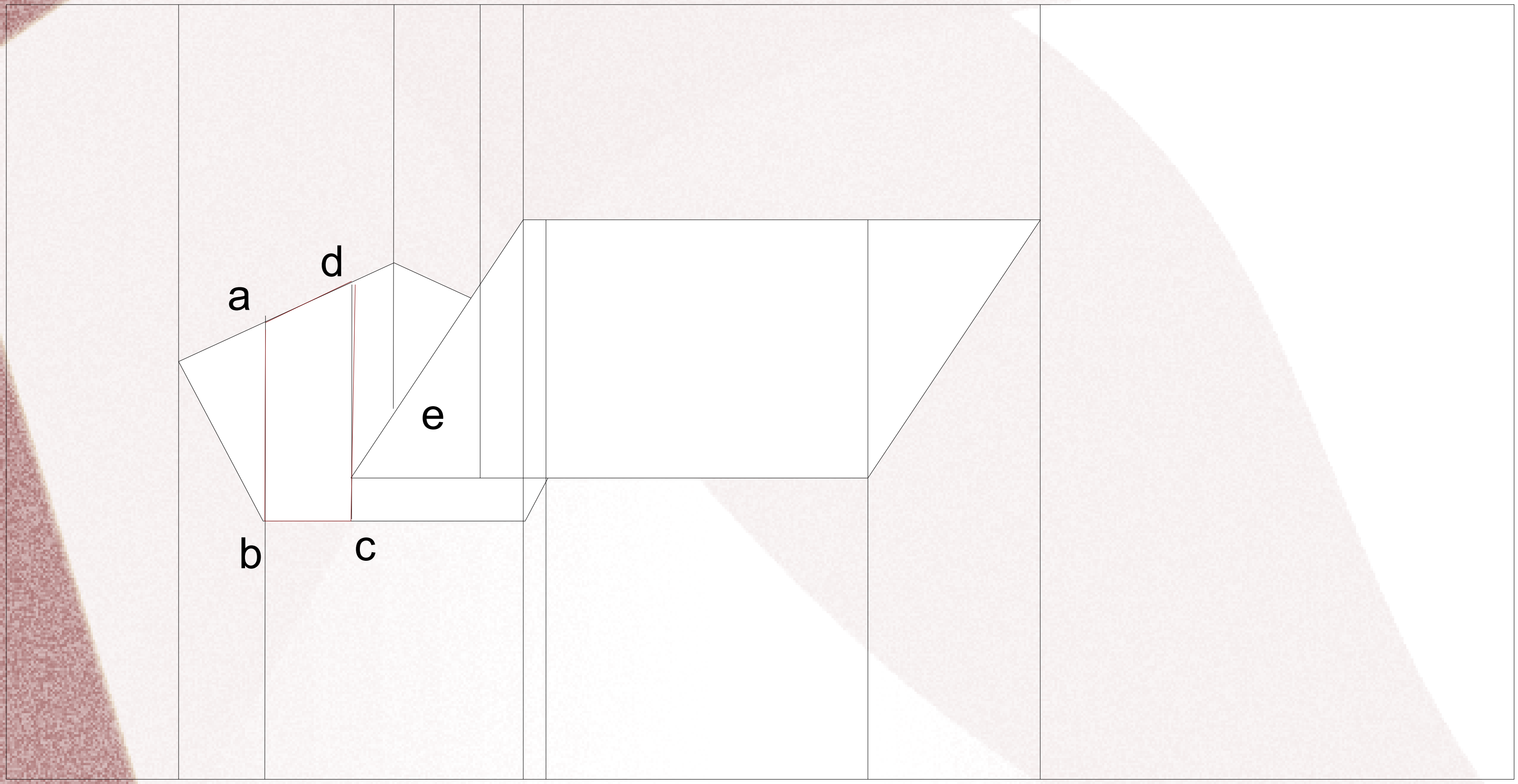


- Tangent continuous



Trapezoid Representation - Example

- Trapezoid (a,b,c,d) – a, b, c, d are visible



- Junction e is not visible in the triangle.

Additonal Information Needed

- Which regions of H_k will be impacted by adding random face f_{k+1}?
- Conflict Information
  - A face is in conflict with a region if
    - The face's projection intersects with the region
    - The region does not obscure the face
  - If a face is in conflict with a region, it will be visible (at least partially) in H_{k+1}.

Dealing with Conflict - Preliminary Updates

- Ignore regions with no conflict

- In conflicted regions, update along the boundary
  - Move counterclockwise around the face (projection)
  - Split conflicting edges.
  - Pass vertical through new conflicted junctions.
  - Update the visible face references.
  - Update conflict and adjacency information.

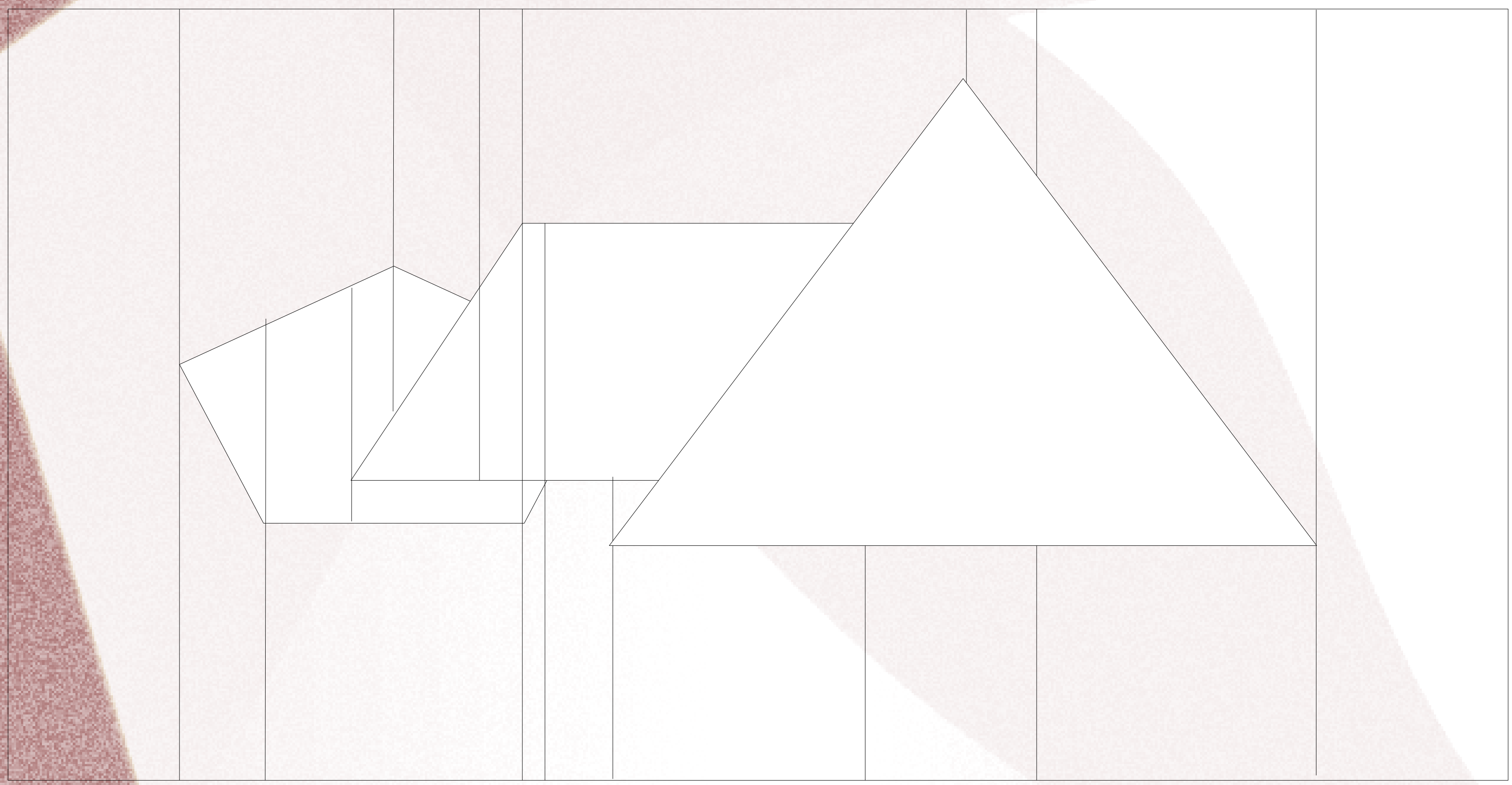
- Update the interior – trapezoids that conflict with the new face, but are not adjacent to it.
  - Change Trapezoid representations
  - Update adjacency lists

Trapezoid Conflict

Dealing with Conflict - Reconfiguration

- It's too expensive to repeat the line-drawing exercise.
- The projection of the new face is a trapezoidal decomposition with unnecessary trapezoids.
  - Randomly remove hidden fragments until they're all gone.

The Complete Partition



Update Conflict Information

- Merge & Sort the conflict information from before the line removal step.
  - Conflict information is stored in order by 'x' coordinate.
- When all faces have been randomly selected, the algorithm is complete.

Analysis

- Mulmuley's algorithm provides hidden surface removal where the time spent processing obstructed surfaces is inversely proportional to the depth of the surface.
  - Expected number of conflicts:

    $O(n \log(n) + \Theta(1))$

  - Conjectured lower bound:

    $\Omega(n \log(n) + \Theta(1))$

Questions

- Thank You!