CSCI 4560/6560 Computational Geometry

https://www.cs.rpi.edu/~cutler/classes/computationalgeometry/S22/

Lecture 21: Binary Space Partitions

Outline for Today

- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter's Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?

Outline for Today

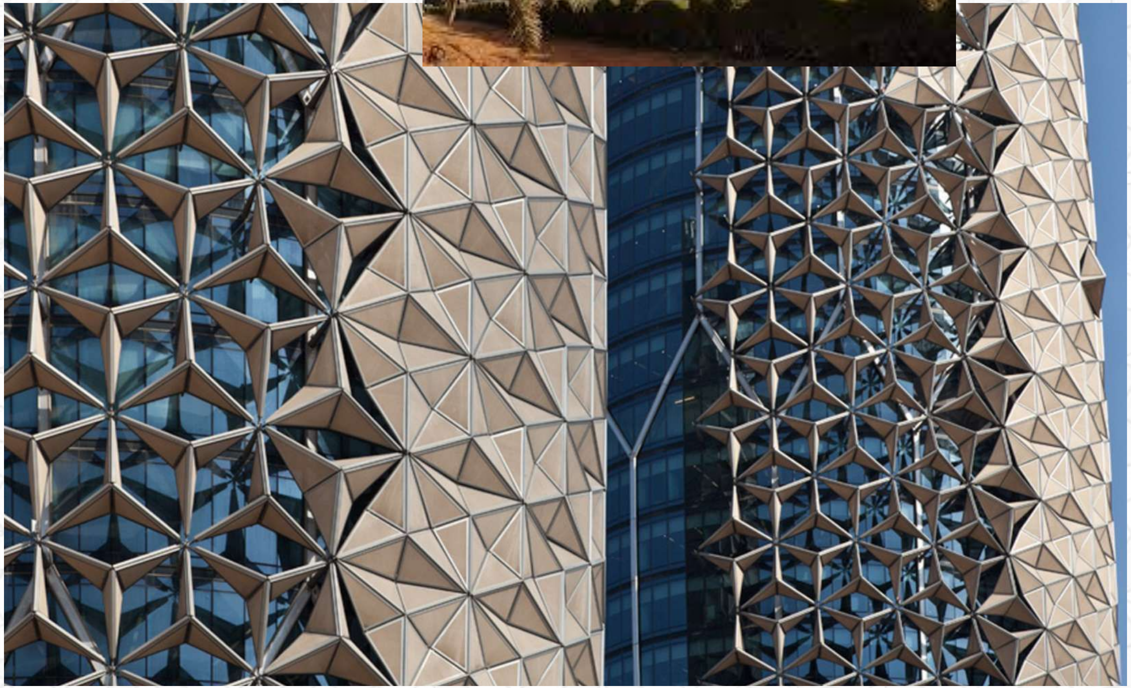
- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter's Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?

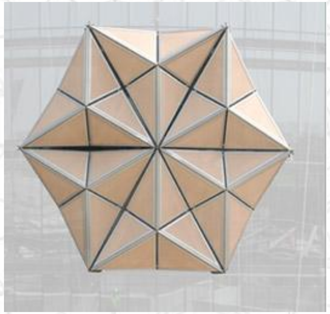
Zellij - Mosaic Tilework

https://en.wikipedia.org/wiki/Zellij

Kinetic Architecture

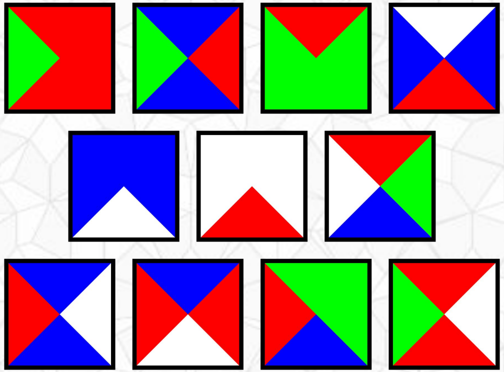
Al Bahar Towers, Abu Dhabi, UAE
Aedas UK, Diar Consult, Arup, 2012

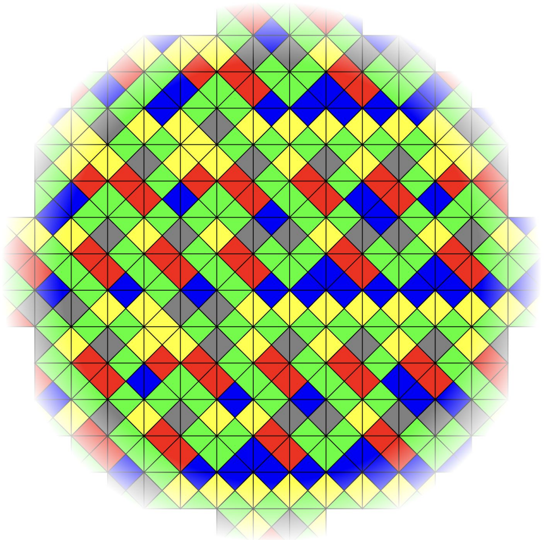




Wang Tiles / Wang Dominoes

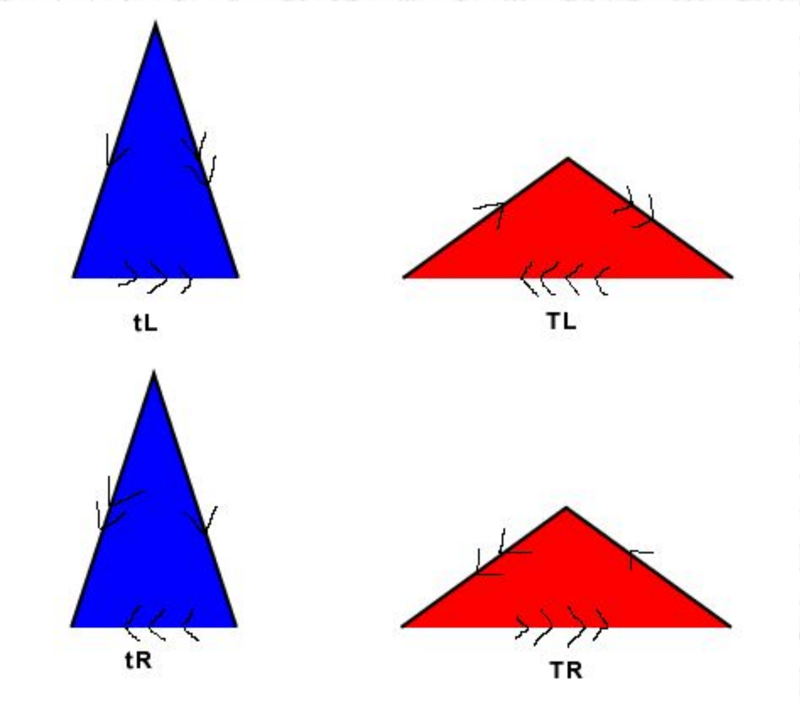
- Square tiles, edges labeled with colors, must be placed without rotation, with matching edges
- In 1961, Hao Wang conjectured that any finite set of tiles that could tile a plane infinitely, could be tiled periodically
- In 1966, Robert Berger proved that non-periodic Wang tile sets existed
- In 2015, Emmanuel Jeandel and Michael Rao proved that the smallest non-periodic Wang tile set was 11 tiles w/ 4 colors
- Applications: natural-looking, aperiodic synthesized texture, heightfields, & more

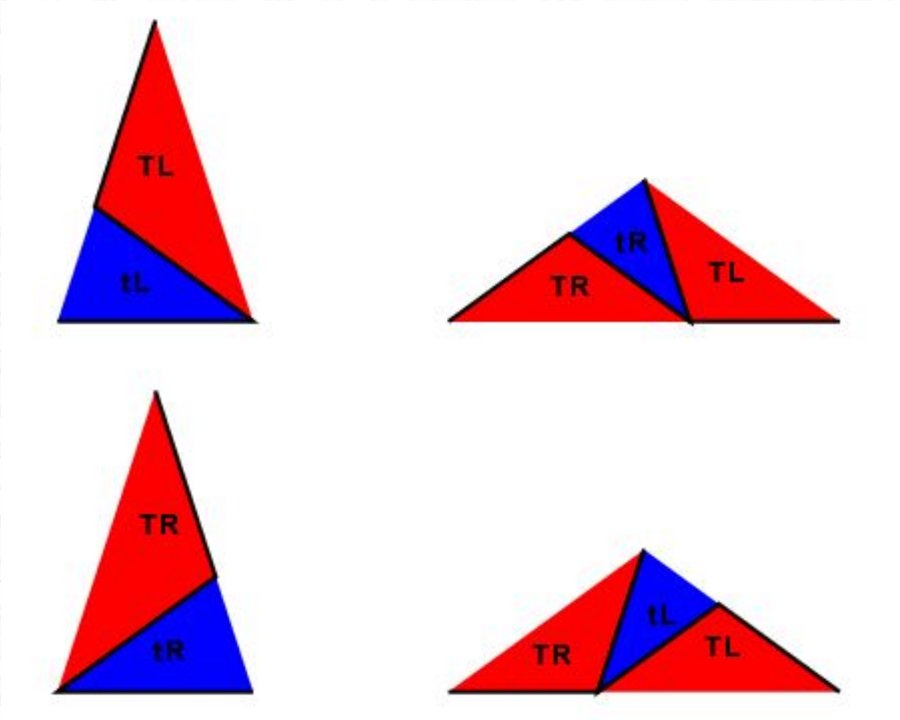




Penrose Tilings Can be Subdivided

- *And conversely, this is how they are proved to be aperiodic!*





https://personal.math.ubc.ca/~cass/courses/m308-02b/projects/schweber/penrose.html

Outline for Today

- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter's Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?

Motivation: Summer Vision Project

- "Summer Vision Project" 1966 10 undergraduate students at MIT were tasked with solving computer vision

It was a "BHAG":
Big Hairy Audacious Goal

Did they (professor/students) realize it at the time?)

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
PROJECT MAC

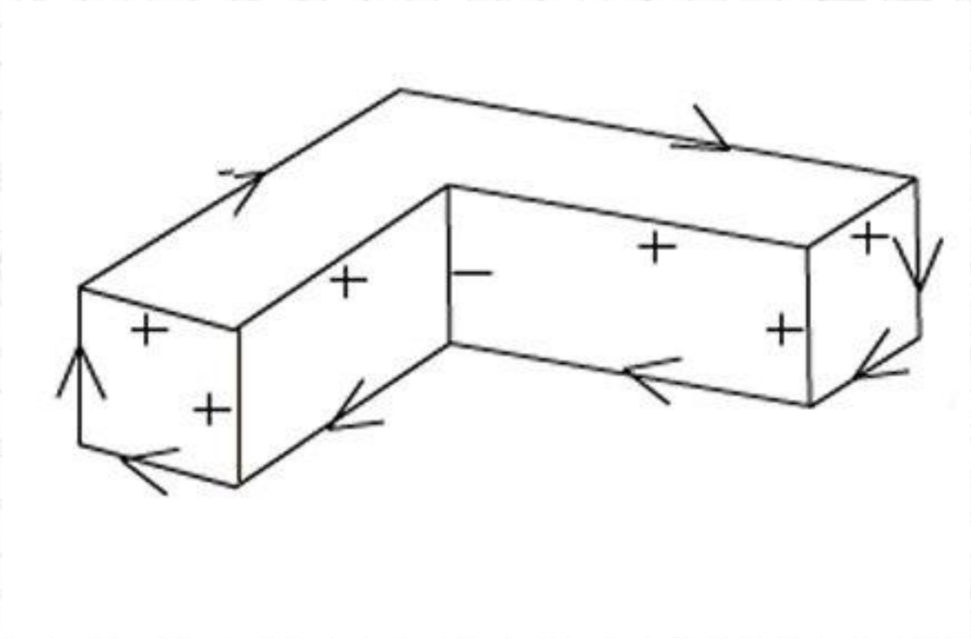
Artificial Intelligence Group July 7, 1966
Vision Memo. No. 100.

THE SUMMER VISION PROJECT

Seymour Papert

The summer vision project is an attempt to use our summer workers effectively in the construction of a significant part of a visual system. The particular task was chosen partly because it can be segmented into sub-problems which will allow individuals to work independently and yet participate in the construction of a system complex enough to be a real landmark in the development of "pattern recognition".

Motivation: Early AI & Early Computer Vision

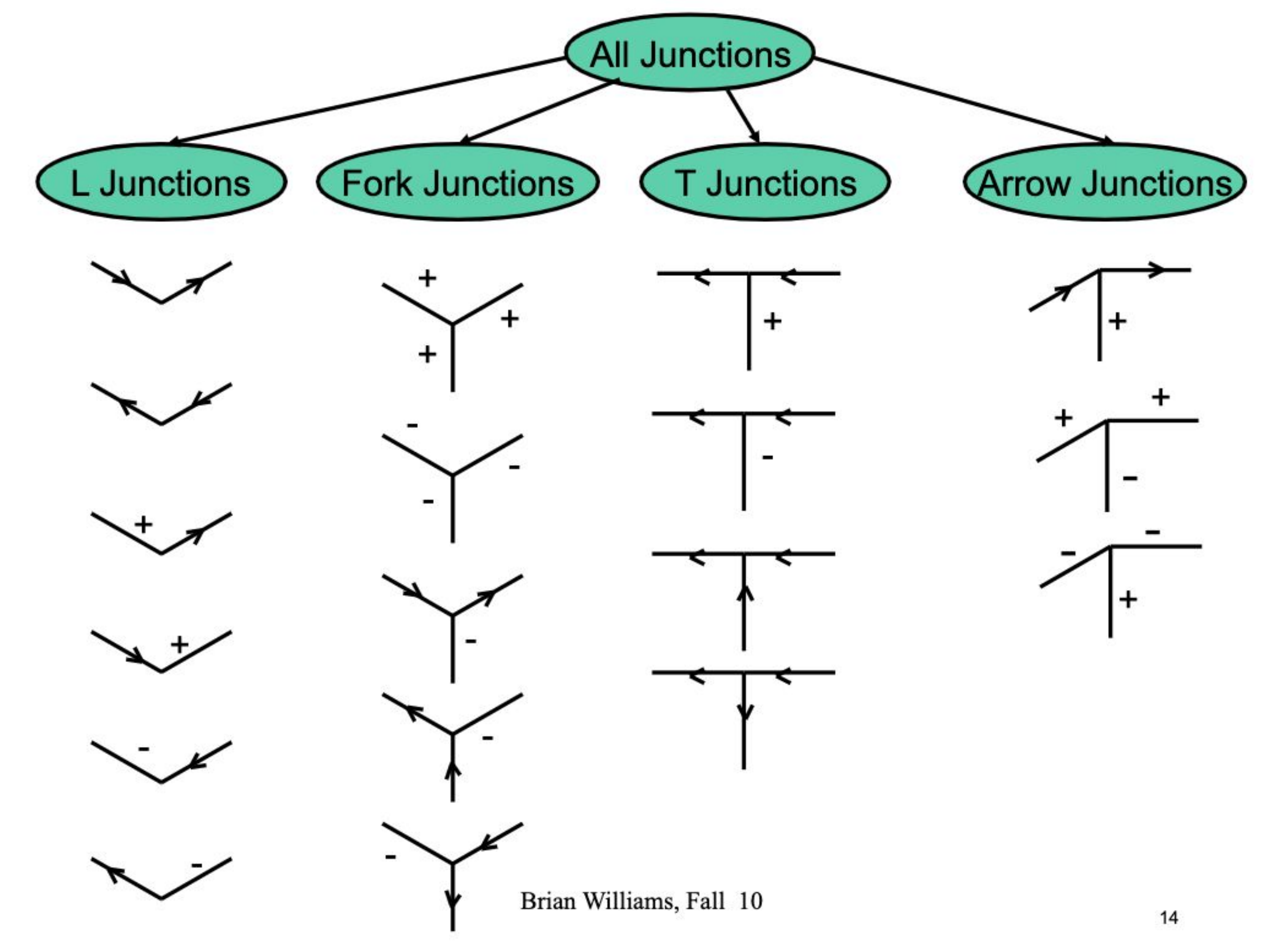


http://www-g.eng.cam.ac.uk/mmg/teaching/artificialintelligence/nonflash/constraint2.htm

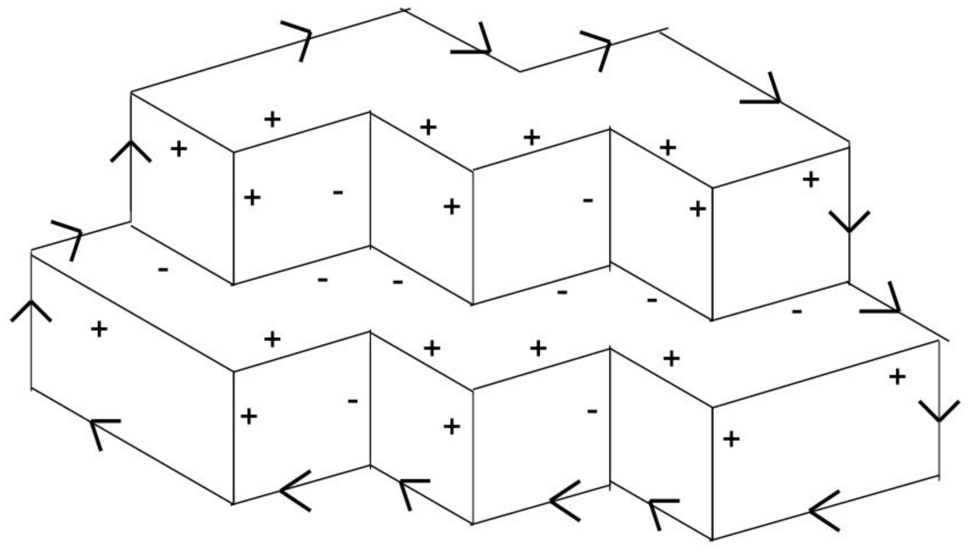
Line Labeling Constraint Propagation

"Interpretation of opaque, trihedral solids with no surface marks", Huffman & Clowes, 1971

"Compute Labeling through Local Propagation"
Waltz, 1972



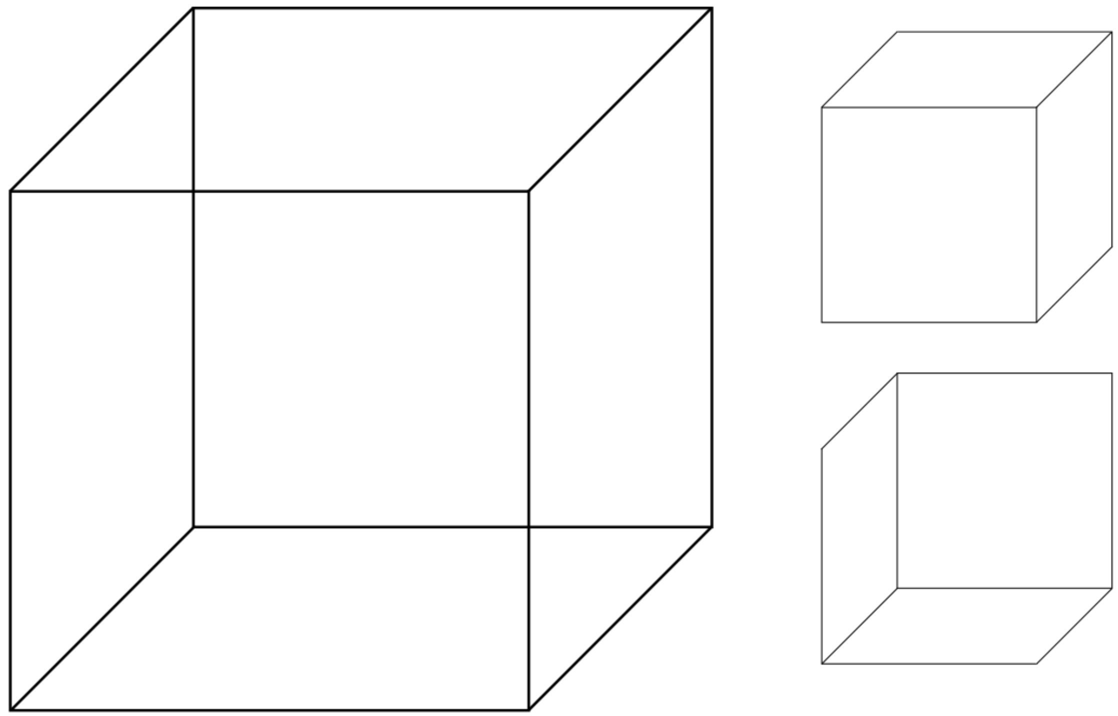
Motivation: Early AI & Early Computer Vision



MIT 6.034 Artificial Intelligence, Fall 2010
Open CourseWare
https://www.youtube.com/watch?v=l-tzjenXrvI

Necker Cube

- A two dimensional representation of a three dimensional wire frame cube
- Viewer’s perception can flips back and forth between equally possible perspectives

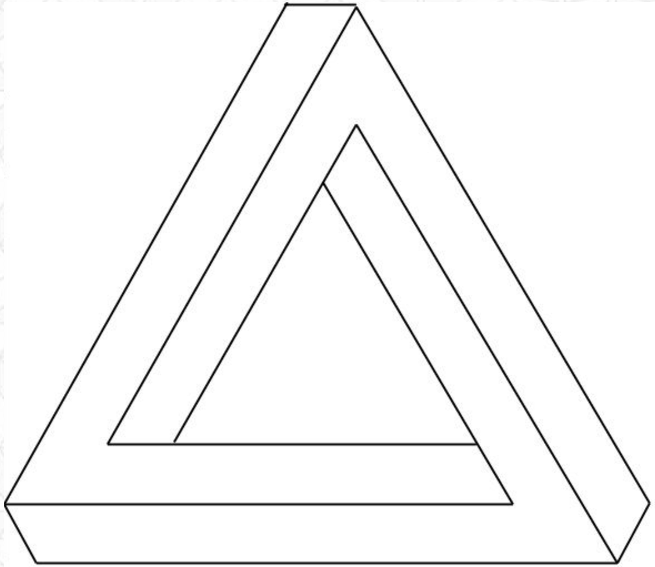


https://www.newworldencyclopedia.org/entry/necker_cube

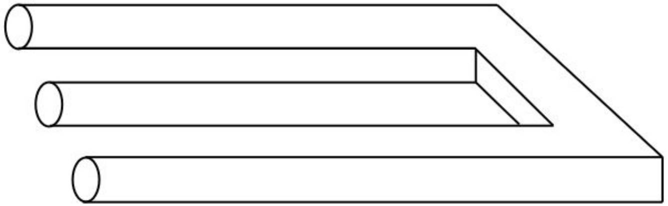
https://commons.wikimedia.org/wiki/File:Necker%27s_cube.svg

Impossible Objects

- Penrose triangle

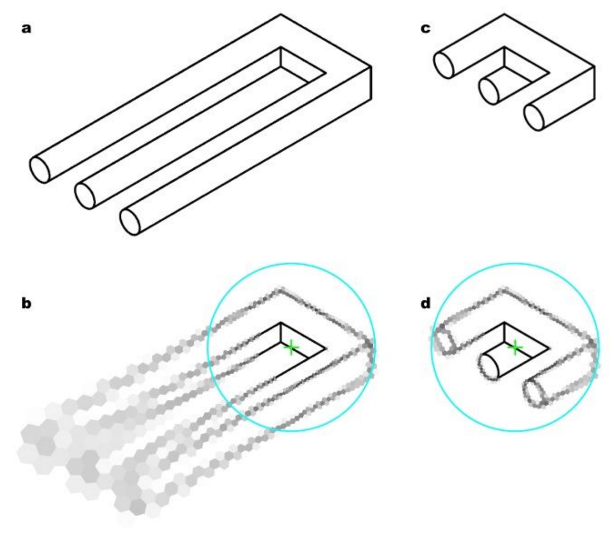


- Devil's tuning fork



https://simple.wikipedia.org/wiki/Impossible_object

Belvedere
M.C. Escher
1958



“Combining Deep Learning and Active Contours Opens The Way to Robust, Automated Analysis of Brain Cytoarchitectonics”, Thierbach et al, 2018

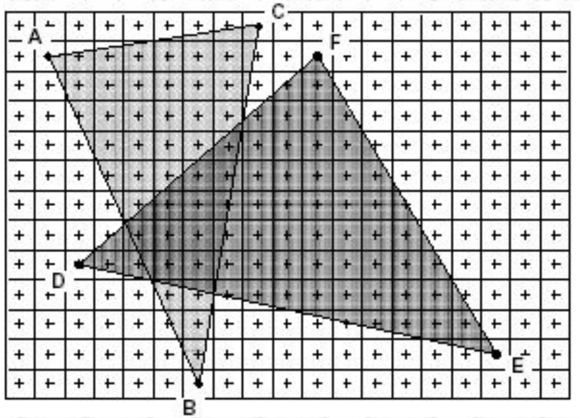
Outline for Today

- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter's Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?

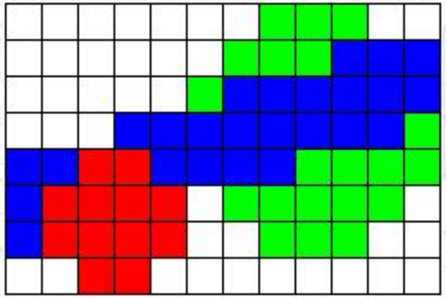
Hidden Line Drawing / Depth Buffer (z-Buffer)

- Given a primitive's vertices & the color / illumination at each vertex:
- Figure out which pixels to "turn on" to render the primitive
- Interpolate the color / illumination values to "fill in" the primitive
- At each pixel, keep track of the closest primitive (depth buffer / z-buffer)

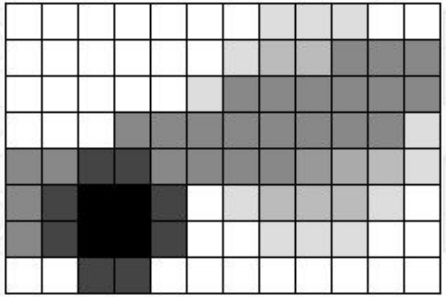
Triangles can be in any order!
A.k.a. "Polygon soup"




```
glBegin(GL_TRIANGLES)
glNormal3f(...)
glVertex3f(...)
glVertex3f(...)
glVertex3f(...)
glEnd();
```



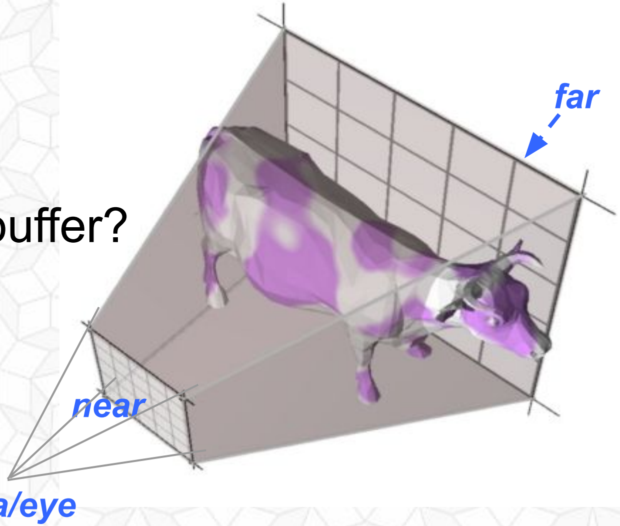
frame buffer

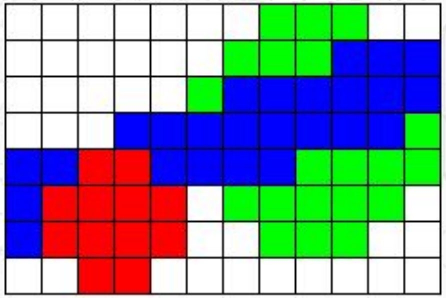


depth buffer

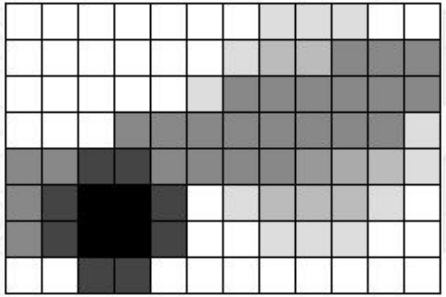
Scan Conversion / Rendering Pipeline

- Running time of depth buffer / z-buffer?
- Extra memory use for depth buffer / z-buffer?
- Flaws with depth buffer / z-buffer?





frame buffer



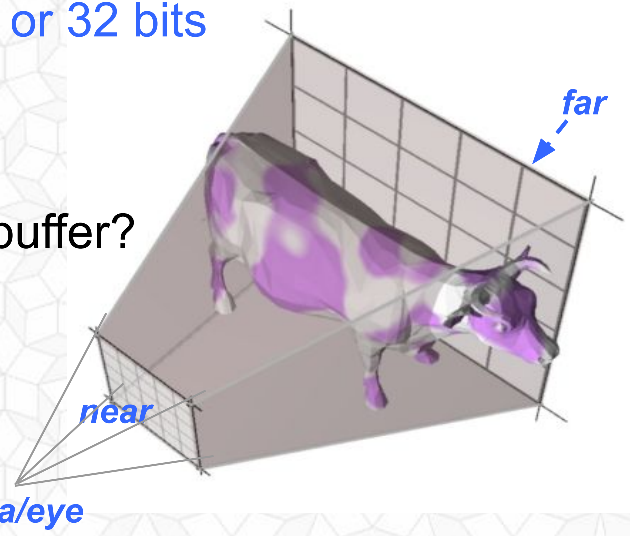
depth buffer

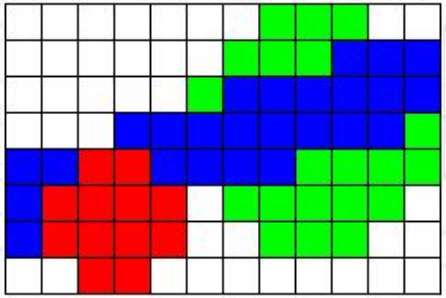
Scan Conversion / Rendering Pipeline

- Running time of depth buffer / z-buffer?
  → *O(n * w * h)* worst case large triangles
  → *O(n)* in practice

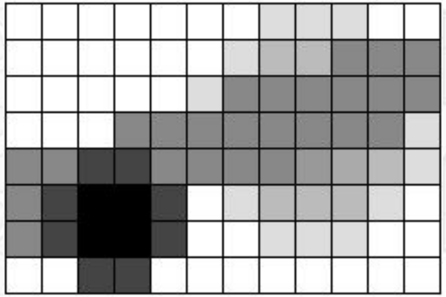
- Extra memory use for depth buffer / z-buffer?
  → *O(w*h)* * 8 bits or 24 bits or 32 bits
  In early graphics, this was too expensive to consider!

- Flaws with depth buffer / z-buffer?
  - Limited precision
  - Need to choose near & far plane carefully





frame buffer



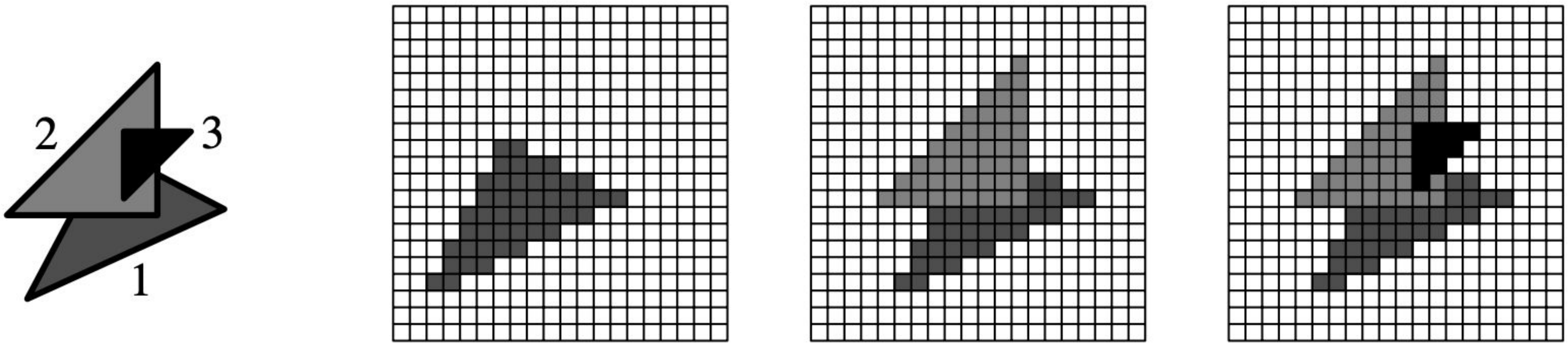
depth buffer

Outline for Today

- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter's Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?

Hidden Line Drawing: Painter’s Algorithm

- Let’s order the primitives by how close they are to the camera
- Draw the primitives from back to front
- Then we don’t need to keep track of the depth!
  Save memory!



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

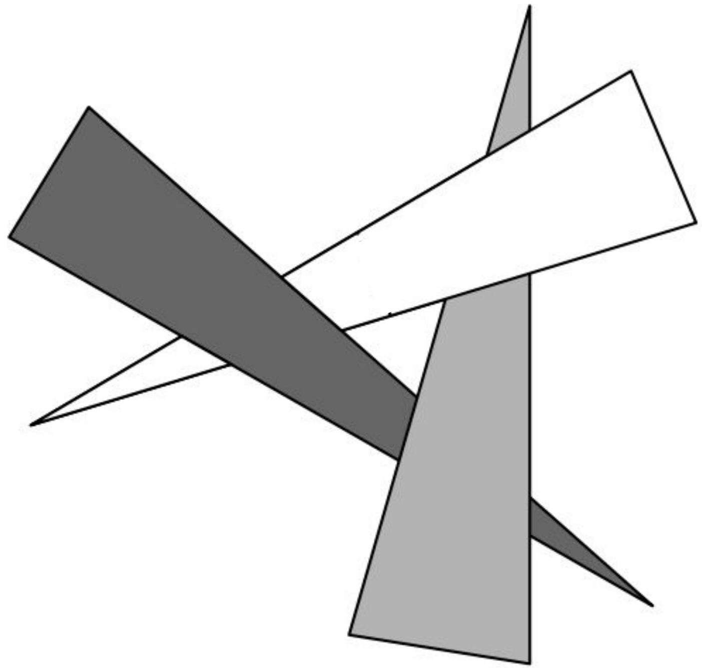


Bob Ross -
Peaceful Waters
Season 3
Episode 13

https://www.twoinchbrush.com/painting/peaceful-waters


Hidden Line Drawing: Painter's Algorithm

- Let's order the primitives by how close they are to the camera
- Draw the primitives from back to front

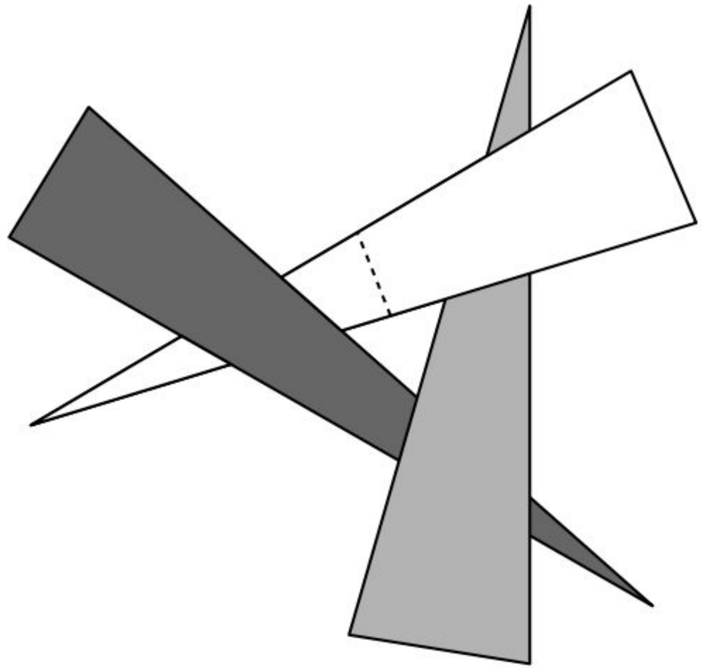


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Hidden Line Drawing: Painter's Algorithm

- Let's order the primitives by how close they are to the camera
- Draw the primitives from back to front

- Warning: Object layering may be complex and have cycles
  E.g., a > b, b > c, c > a
- *Solution: Split primitives as necessary to break cycles*



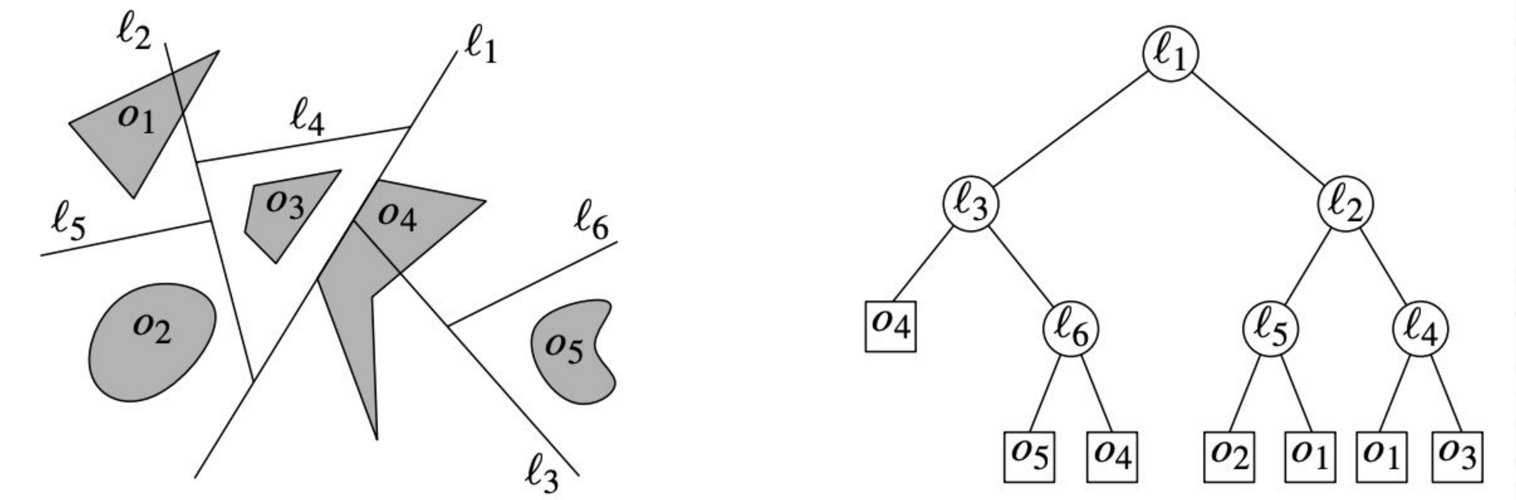
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Outline for Today

- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter's Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?

Definition: Binary Space Partition

- Place items in a binary tree, each node stores a half plane
- Primitives that are collinear with the half plane are stored in the node
- Items overlapping a half plane are copied/split into two primitives
- We recurse until exactly one item is left, it is stored in the leaf

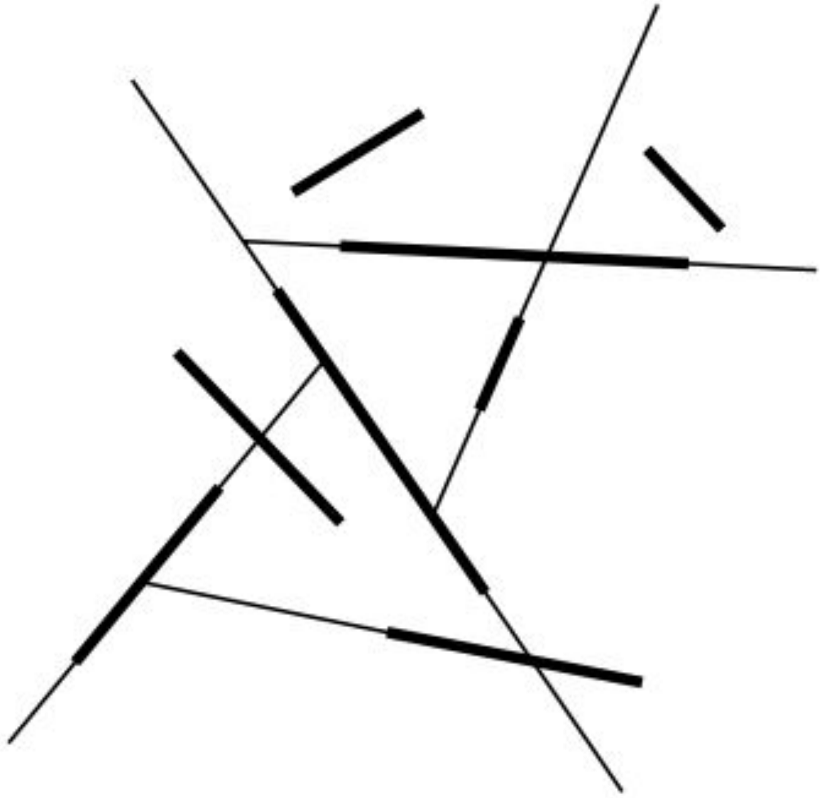


Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Auto-Partition

- In practice, it is common to use the primitives as half-planes
- If a BSP only uses half-planes derived from the input data, it is called an auto-partition

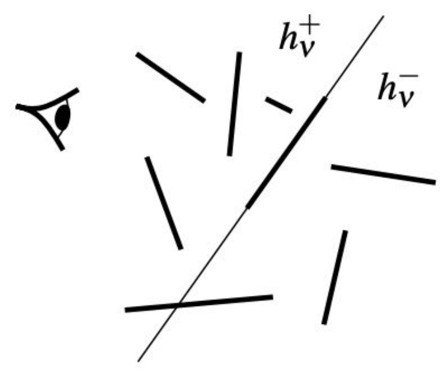
- Primitive is stored at the node (rather than pushed down to a leaf)
    - So it will probably be smaller…
    - But the optimal partitioning (minimal # of nodes) may require hyperplanes that are not derived from the input!

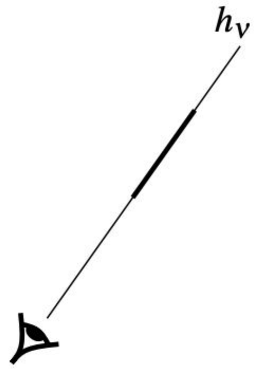


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Using a BSP to Render via Painter’s Algorithm

- If we’re at a leaf,
  - Render items in current node
- Else if camera to **left** of current node hyperplane
  - Recurse to right of current node
  - Render items in current node
  - Recurse to left of current node
- Else if camera is to **right** of current node hyperplane
  - Recurse to left of current node
  - Render items in current node
  - Recurse to right of current node
- Else we’re on the split plane
  (we can ignore items in current node)
  - Recurse to left of current node
  - Recurse to right of current node





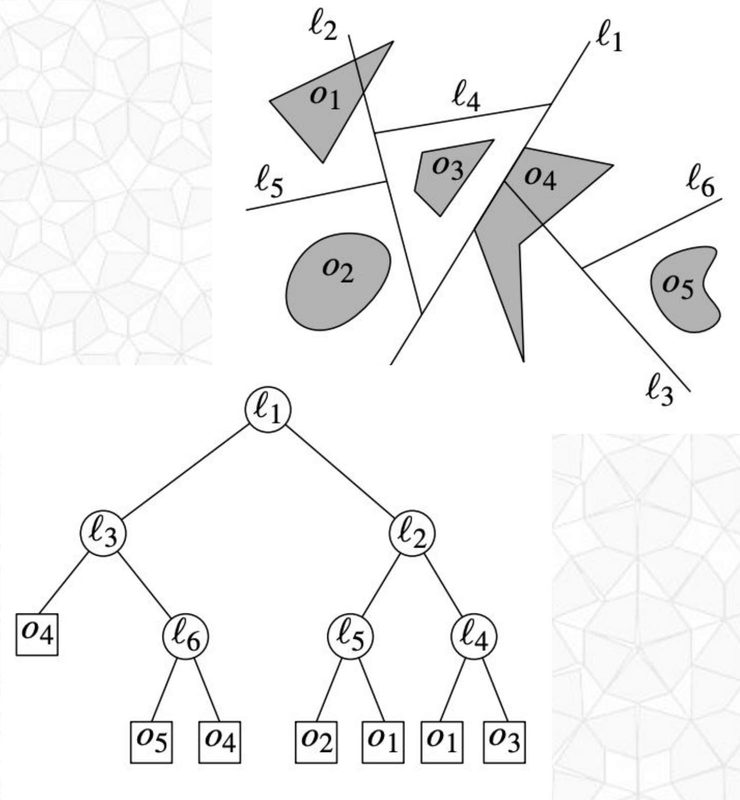
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Outline for Today

- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter’s Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?

Analysis: Using BSP for Painter's Algorithm

- For *n* non-intersecting primitives
- Best case:
- Worst case:
- Overall: Painter's algorithm

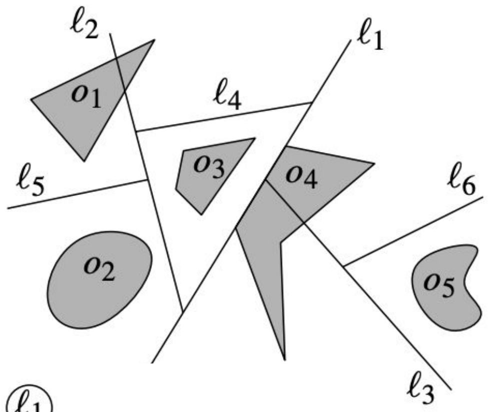


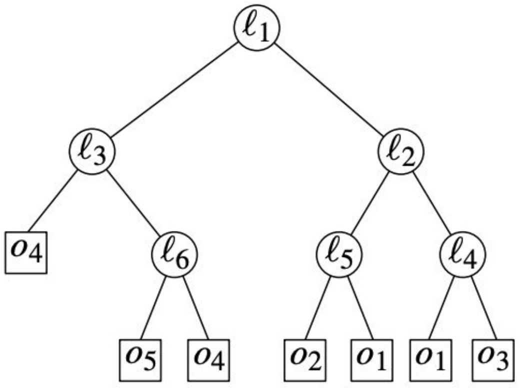
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Analysis: Using BSP for Painter's Algorithm

- For *n* non-intersecting primitives
- Best case:
  - No primitives are split
  - O(*n*) nodes in the tree
  - Tree is perfectly balanced, height = *O(log n)*
- Worst case:
  - Every primitive is split by every plane
  - $O(n^2)$ nodes in the tree
  - Tree is unbalanced, height = *O(n)*
- Overall: Painter's algorithm
  - O(# of nodes in the tree)
  - (height is irrelevant!)

- ***Can we do better than worst case??***



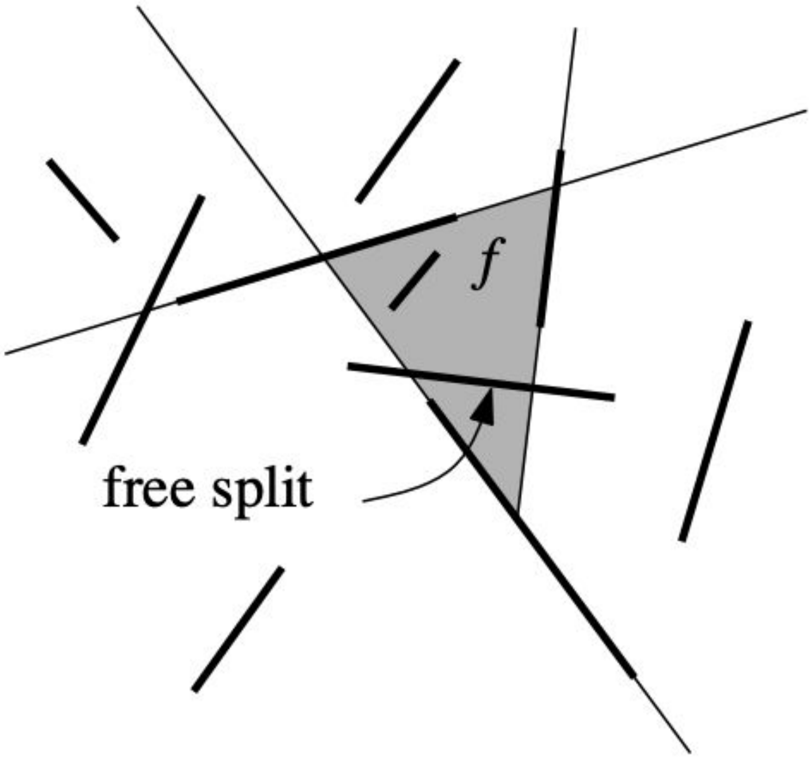


Computational Geometry Algorithms and Applications, de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Small Optimization: “Free Split”

- Our input primitives do not intersect

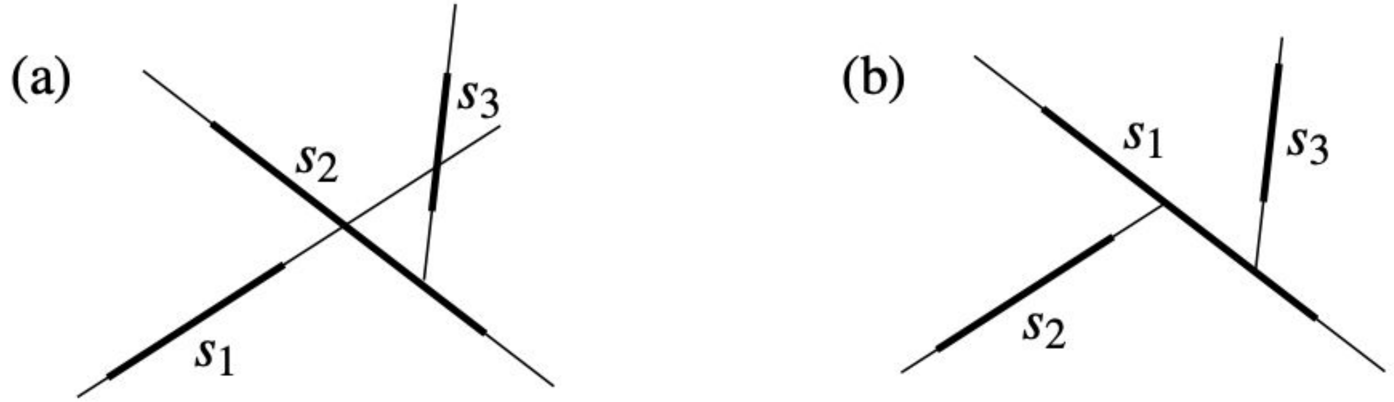
- If we can determine that both primitive endpoints are on the half plane boundaries of the current subtree
- Choosing that primitive as the next half plane node is guaranteed not to split any primitives



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Randomized Incremental Construction

- Note: Some orderings are better than others:
  (result in fewer split primitives)



- *Let's randomize the order!*

Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

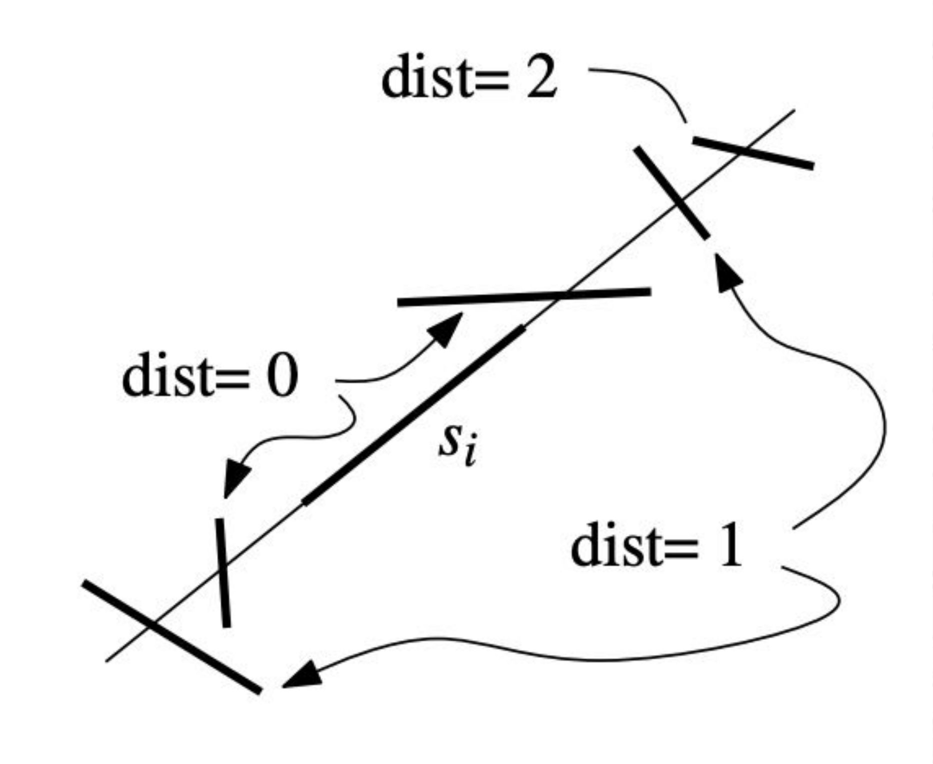
Randomized Incremental Construction

- Let's randomize the order!

  $s_0, s_1, s_2, ... s_i .... s_k ...$

- What's the chance that a primitive s_k will be split by the half plane derived from s_i?

- If there are many other segments between s_i and s_k there is a good chance one of them will shield s_k from being split by s_i



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Randomized Incremental Construction

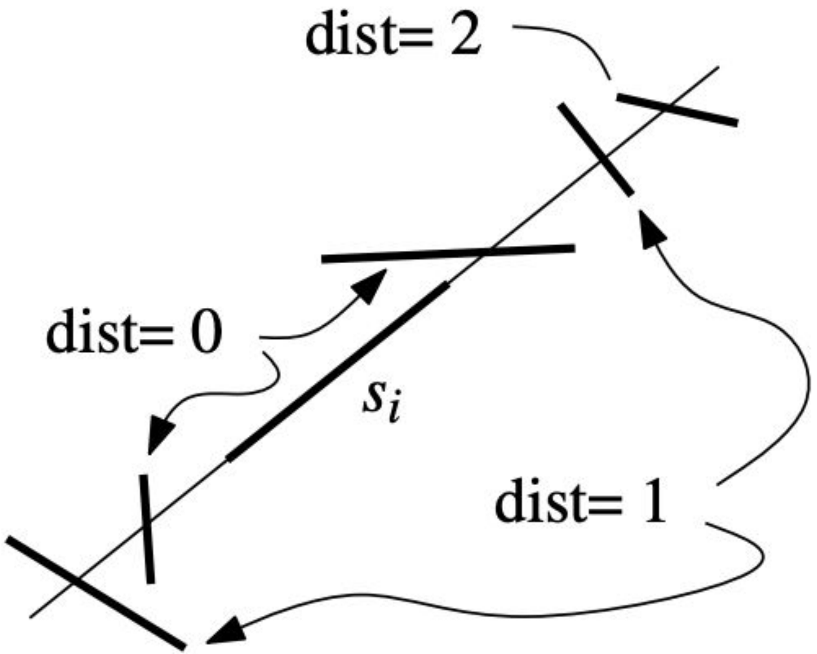
- Let's randomize the order!

  $s_0, s_1, s_2, \ldots . \, s_i \ldots . \, s_k \ldots$

- Randomized BSP
  can be shown to be
  have *O(n log n)* nodes

- And can be constructed
  in $O(n^2 \log n)$

- *Which is better than our worst case*
  But still doesn't seem great…



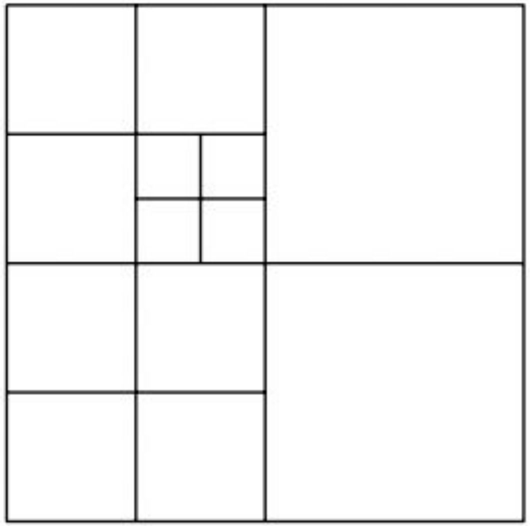
Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Outline for Today

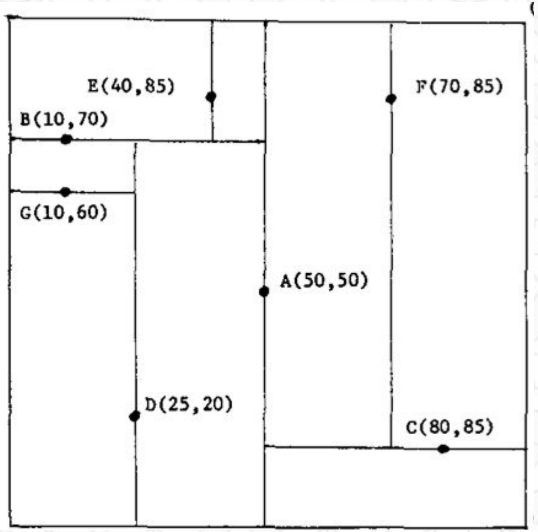
- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter’s Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?

Discussion - Quad Tree, kD Tree, BSP

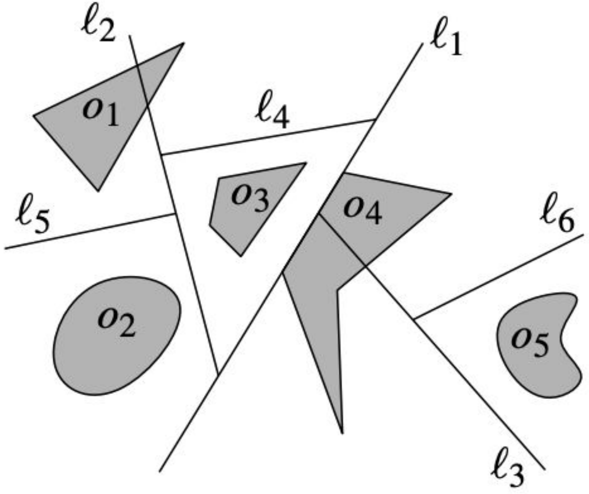
- k-D trees are a special case of BSP (where splits are always axis aligned)
- Quad trees are a special case of k-D trees
  (where splits are always at the midpoints)



Quad Tree



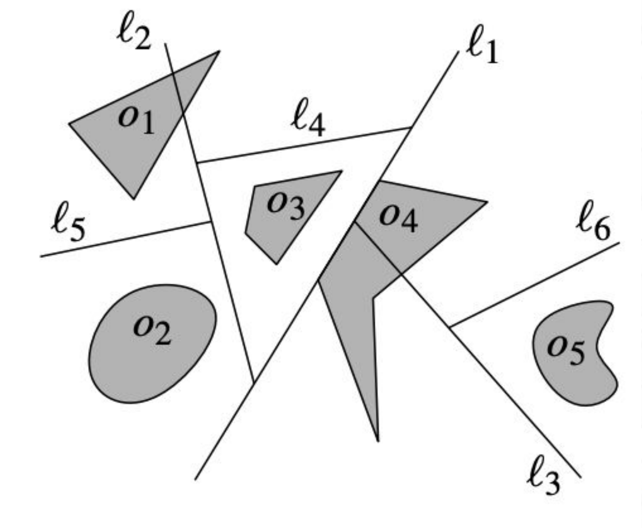
k-D Tree



BSP

Discussion - BSP & Low Density Scenes

- BSP are harder to visualize, and therefore perhaps harder to intuitively understand, debug, and analyze
- Usually the performance of a BSP is much better than the conclusion reached by randomized analysis.
- Why?
  - In practice most objects are relatively small
  - In practice density of objects in a scene is sparse
  - Therefore it is likely the objects can be separated by planes without requiring the expected worst case number of splits
- *For more details, see analysis in the book…*



Computational Geometry Algorithms and Applications,
de Berg, Cheong, van Kreveld and Overmars, Chapter 12

Outline for Today

- Homework 6 Posted
- Last Time: Periodic & Non-Periodic Tiling
- Line Drawings & Early Computer Vision / AI
- Hidden Line Drawing: z-Buffer
- Hidden Line Drawing: Painter's Algorithm
- Binary Space Partition
- Binary Space Partition Analysis
- Discussion & Comparison to Quad Tree & kD Tree
- Next Time: ?