Binary Space Partitioning Trees

A Computational Representation of Space

Bruce F. Naylor

Spatial Labs Inc.

Summit, NJ

naylor@spatial-labs.com

Virtual Worlds

Recreating the “real world”

- can use arbitrary amounts of computation
- founded on computational geometry

Computational Components

- graphics - modeling and rendering
- physics - collisions and dynamics
- movement - creature body motion
- behaviors - AI-based simluation
- networking - client/server multi-participant
- data management - distributed database

Virtual World Applications

Training

- military - combat and vehicle piloting
- commerical - vehicle piloting and maintenance/repair
- medical - surgery planning

Entertainment

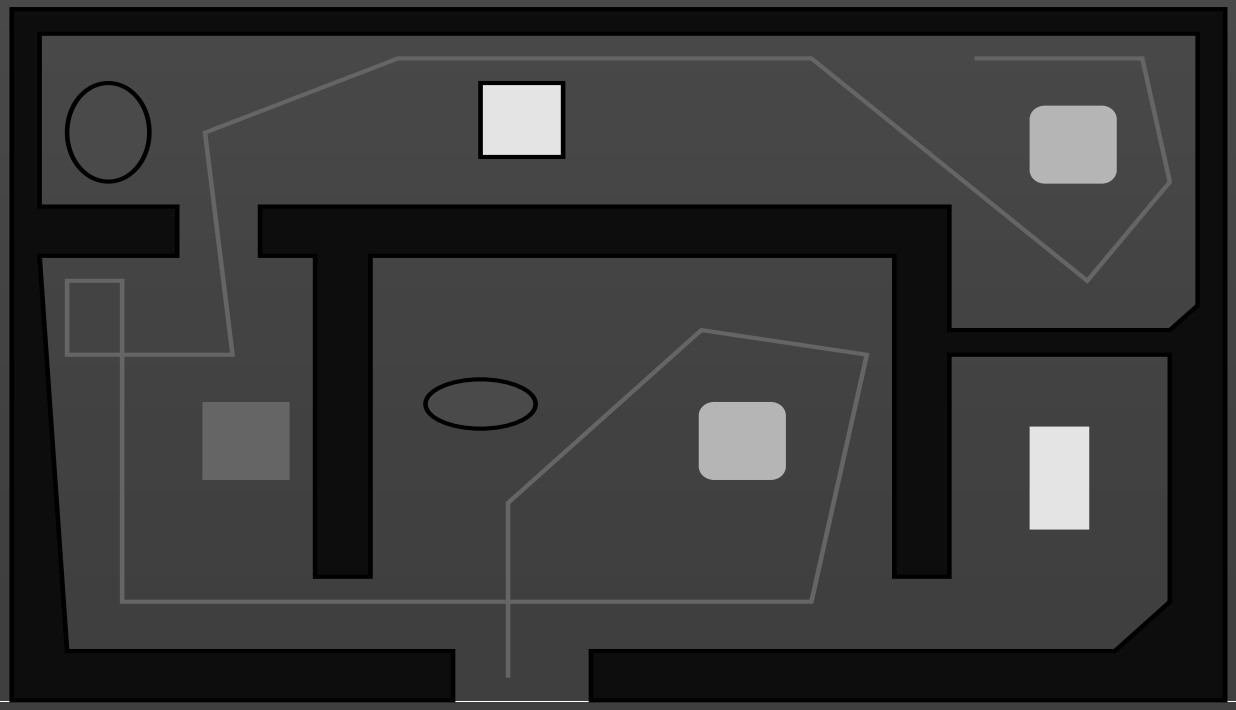
- games - 1st person shooters
- virtual communities - networked 3D chat
- animation - films, TV

Design

- CAD/CAM - assembly of large models
- architecture - evaluation and presentation

Interacting with Virtual Worlds

Navigation through virtual worlds requires determining what is visible and what collides for every frame



Properties of Virtual Worlds

Environments are dynamic

- continuous, real-time state changes at 15-60 frames per sec.

Large amounts of data

- 10K to 10M elements in geometric database

Only small subset needed for each frame

- limited field-of-view
- occlusion
- limited contact

Need fast *Geometric Database* search engine

Efficient Geometric Computation

Fast computation of *spatial relations*

- Determining visibility and intersection requires knowing spatial relations between polygons

  Naive algorithm would be $O(n^2)$

Can be accelerated by spatial search structures

- Reduceable from $O(n \log_2 n)$ to $O(1)$ by

  Octrees, BSP Trees, and other

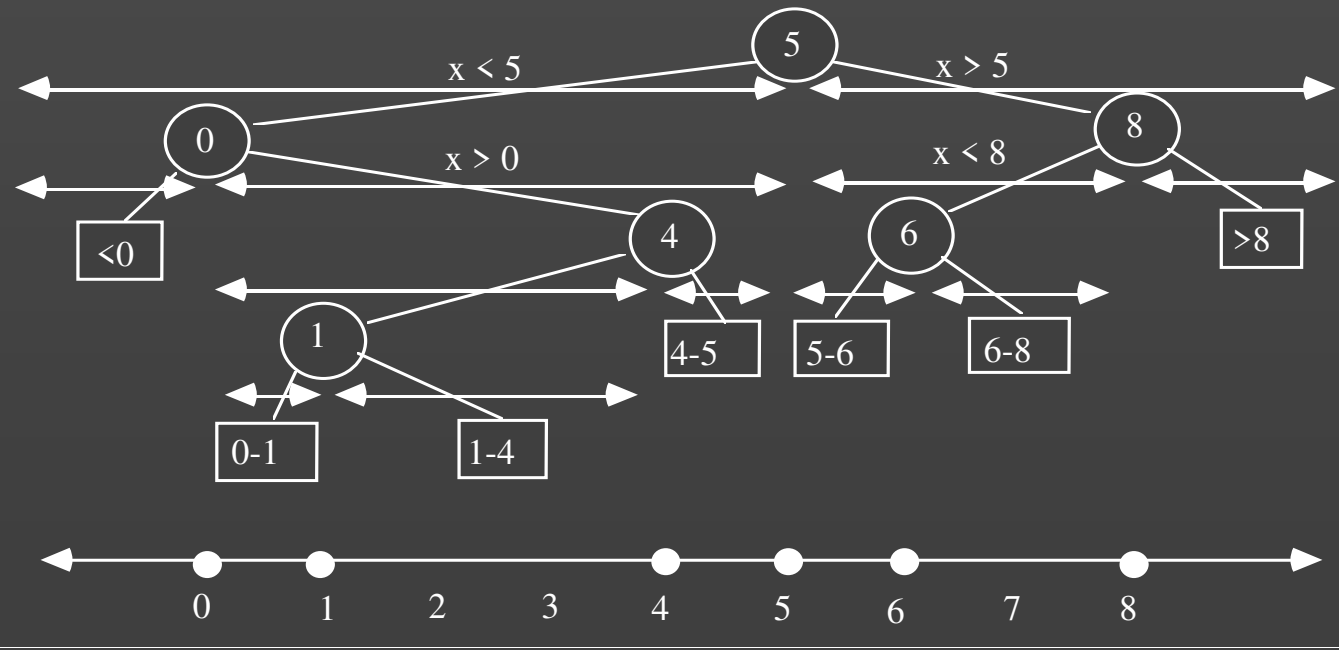
  Bounding-Volume Hierarchies

The larger the database, the greater the importance of search structures!

Symbolic Searching and Sorting

Binary Search Trees

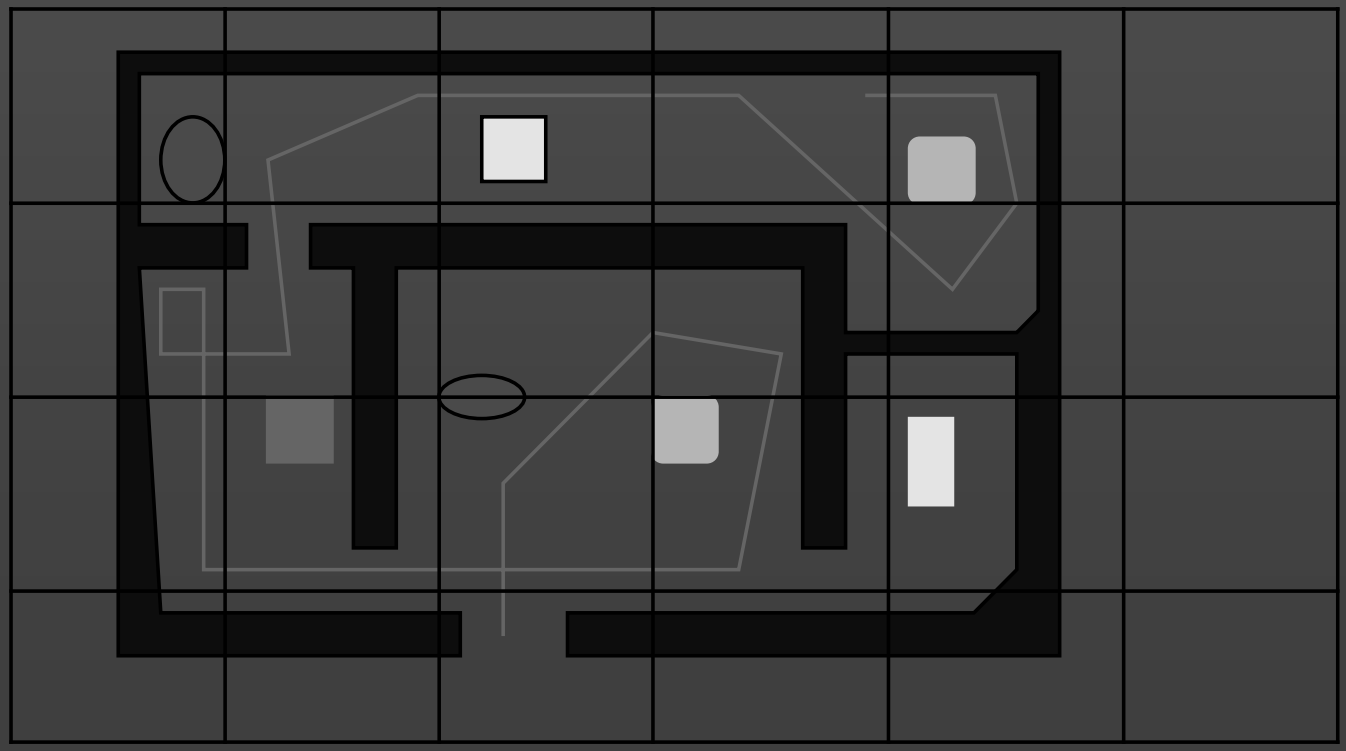
- Insertion, deletion and retrieval in $O(\log_2 n)$
- Tree represents hierarchy of intervals



Regular Grids (Arrays)

Optimal for uniformly distributed geometry

But geometry is almost never uniformly distributed!!!

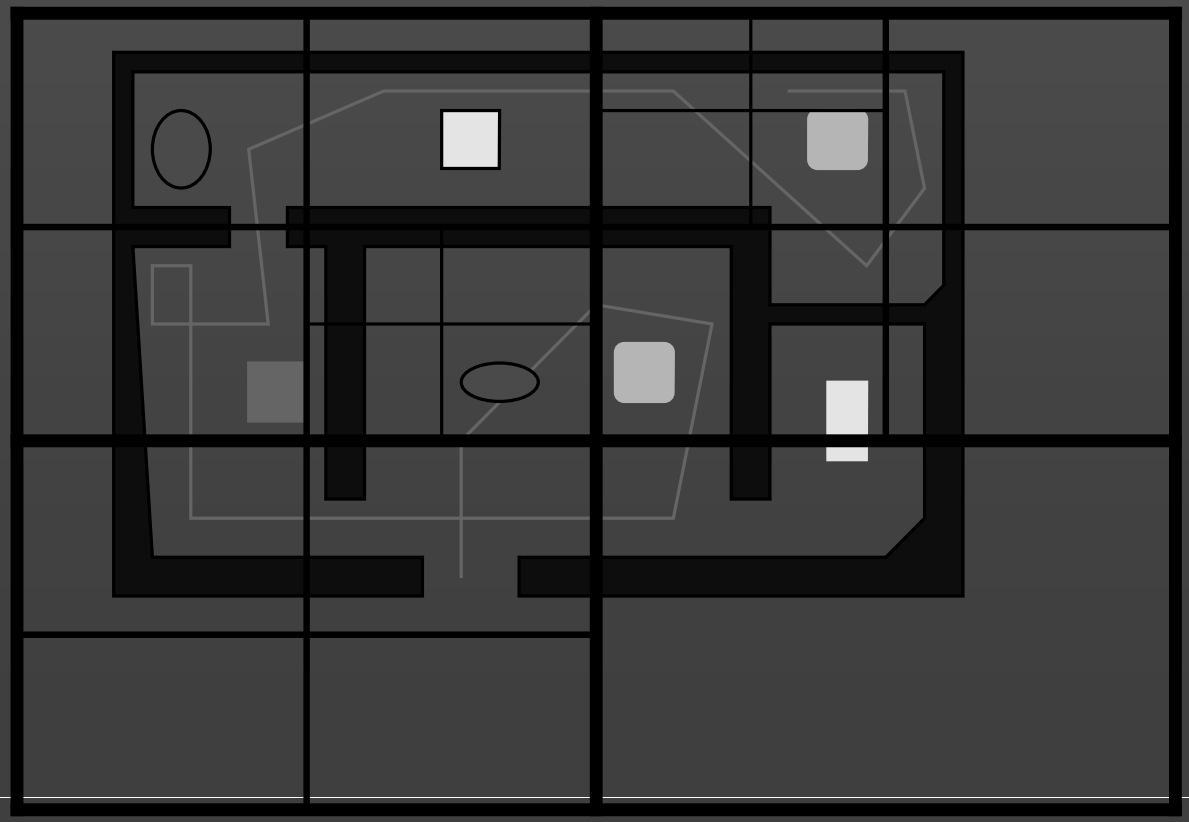


Octrees

Hierarchical, adaptive, regular subdivision

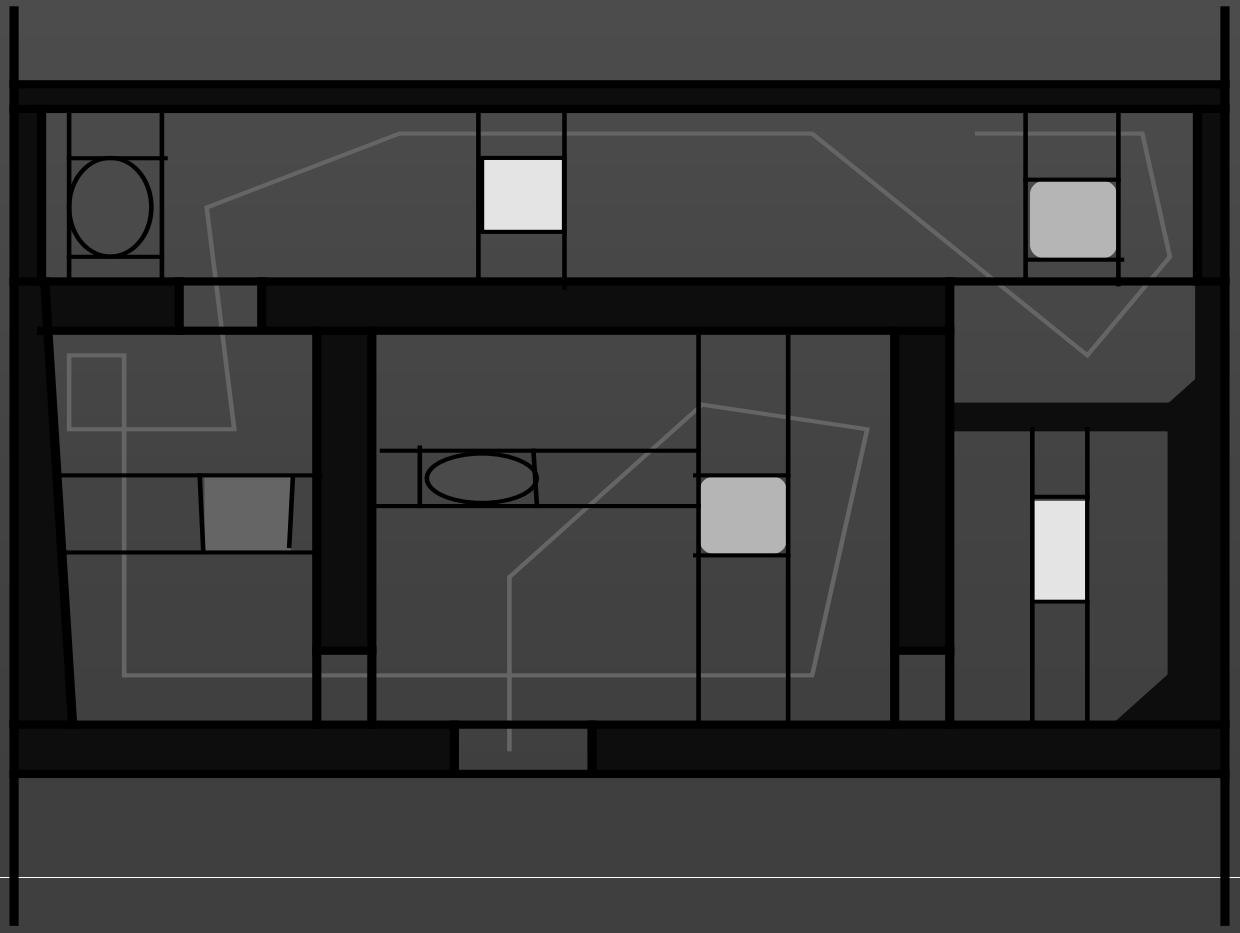
- must decide whether, but not where, to subdivide

Not transformable and does not fully solve visibility



Binary Space Partitioning Trees

Geometry induces an organization on space



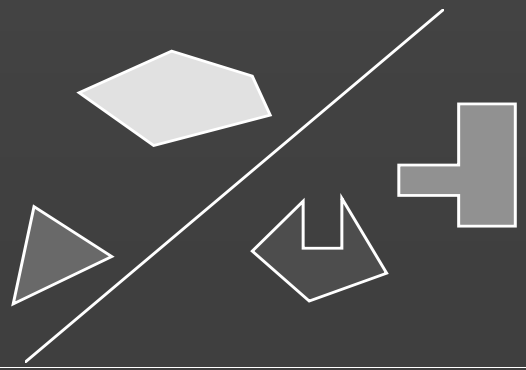
Plane Power

Exploit properties of planes

- To reduce number of intersection calculations
- To reduce number of visibility calculations

Intersections

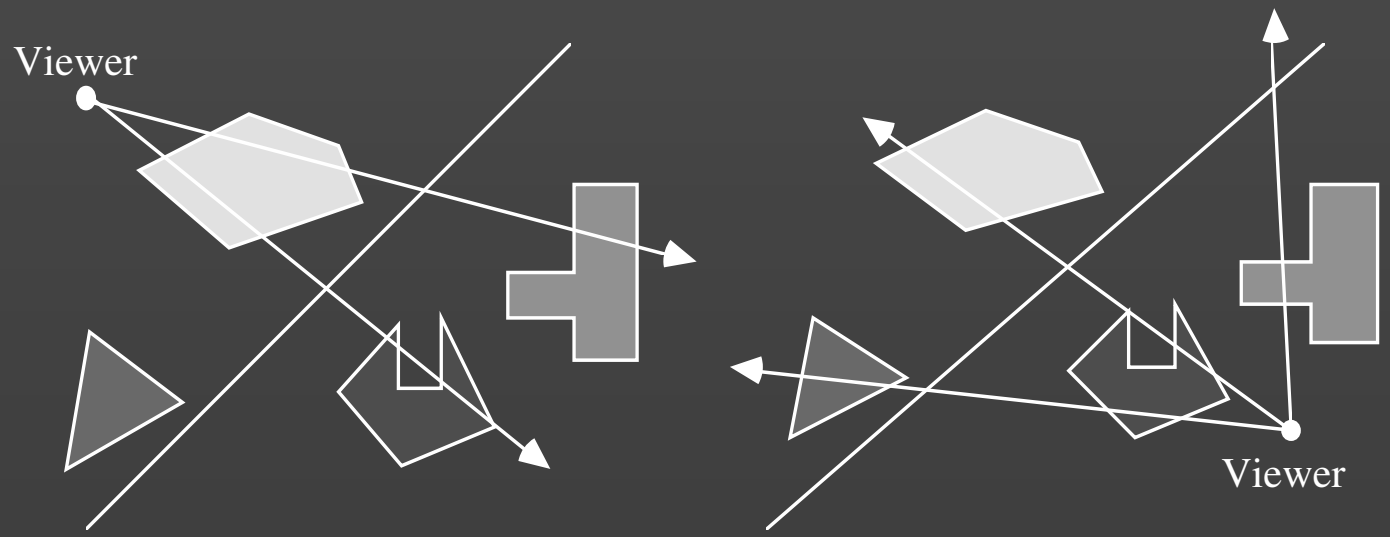
- Objects on opposite sides of a plane cannot intersect.



Plane Power

Visibility

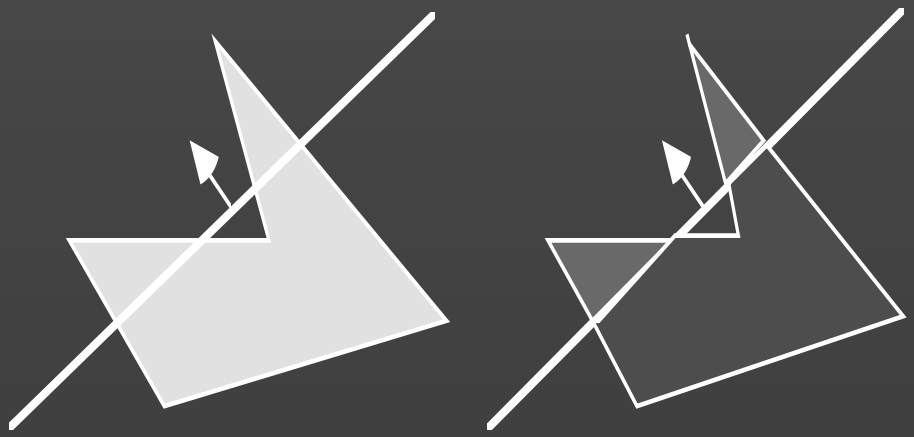
- objects on same side as viewer are "closer" than objects on opposite side
- use visibility order with “Painter’s Algorithm”



Plane Power

Partitioning

- geometry must allow partitioning by a plane (hyperplane)
- limit to 1st degree algebraic sets which are closed under intersection



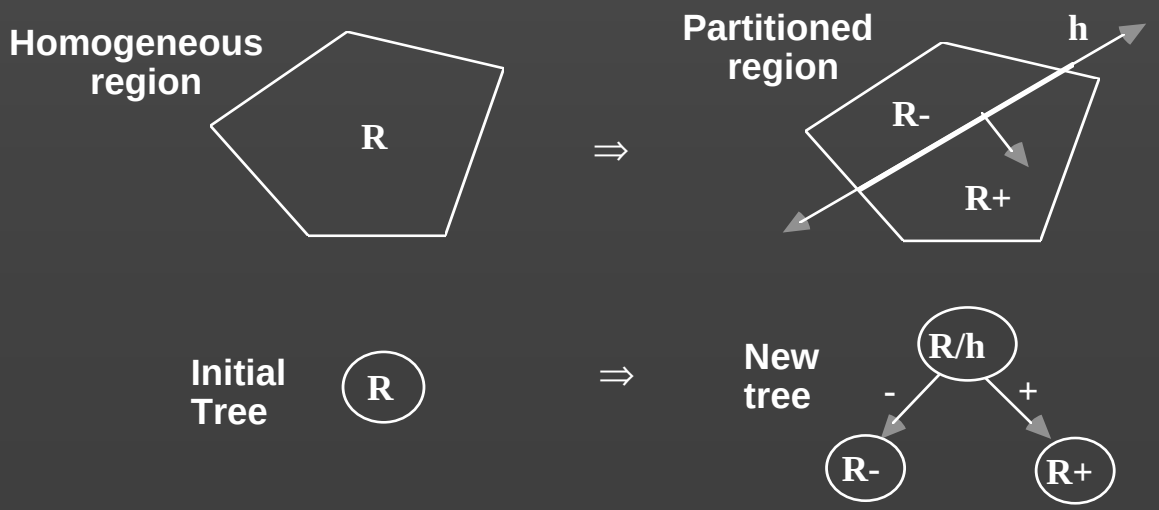
Binary Partitioning

Single geometric operation

- partition a convex region by a hyperplane

Single combinatorial operation

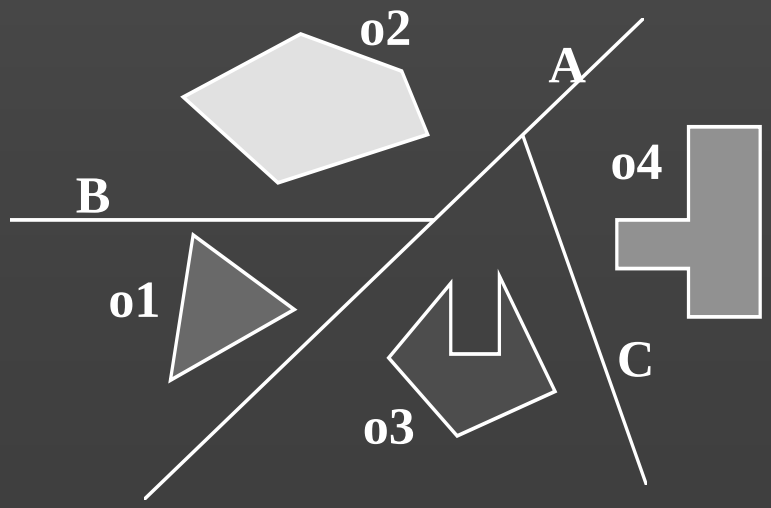
- two child nodes added as leaf nodes

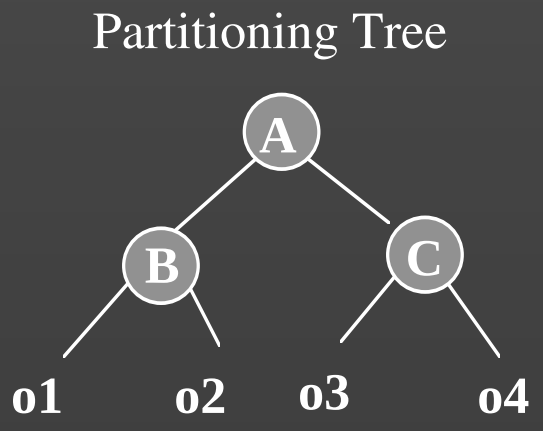


Spatial Search Structure

Inter-object Trees

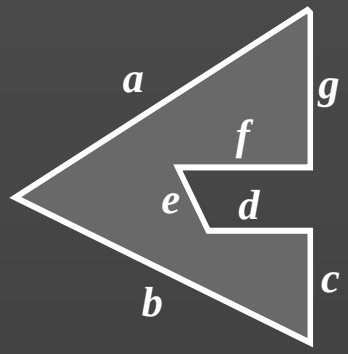
- generalizes binary search trees to dimensions > 1
- compare to hyerplanes instead of points



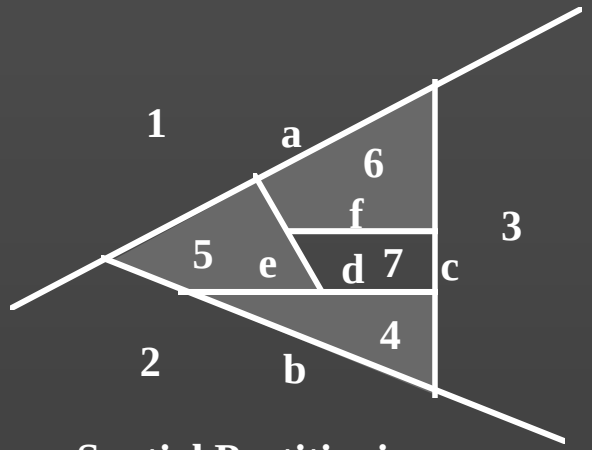


Solids as BSP Trees

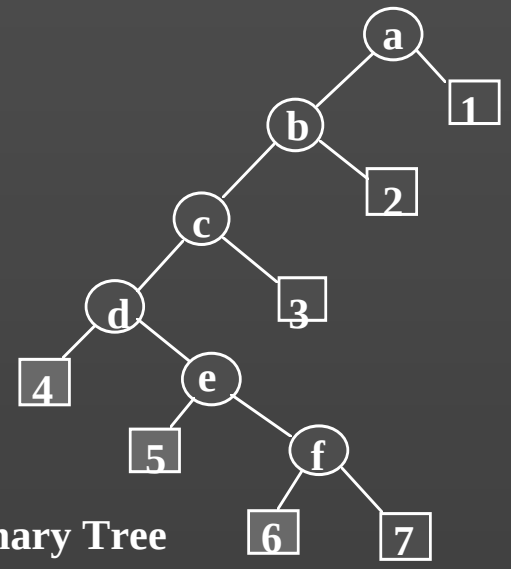
Intra-object Trees



Orignal B-rep



Spatial Partitioning



Binary Tree

New Representation of Polytopes

Foundations are *hyperplanes* instead of *points*

- *duals* in projective space
- encodes *membership* hierarchy vs. *topology*
- *volumetric* vs. *surfaces*, implicit vs. parametric

***All* polytopes can be represented by a *single* tree**

- hierarchical *convex* decomposition of space into **in** and **out** cells
- more general - multiple components, unbounded, non-manifold
- every tree is valid polytope - **syntax == semantics**

Computational Representation of Space

Space Partitioning

- combines search structure with representation
- objects *induce* structure on space, ala Relativity
- intrinsically computational - tree is a computation graph

Encodes Spatial-Relations

- pre-compute tree as off-line process
- benefits reaped over many frames : space-time trade-off
- **merge trees** in real-time to compute inter-object spatial relations on-the-fly

Functionality of BSP Trees

Intersections

View-volume culling

- eliminate geometry not in field-of-view

Constructive Solid Geometry

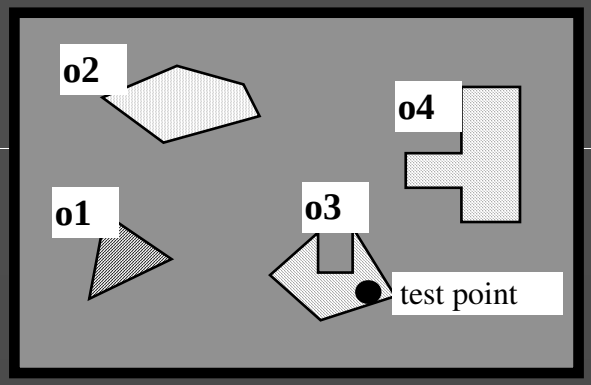
- union, intersection and difference of objects
- construct and modify objects

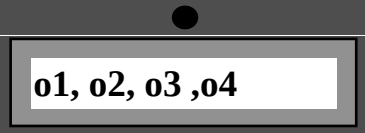
Collision Detection

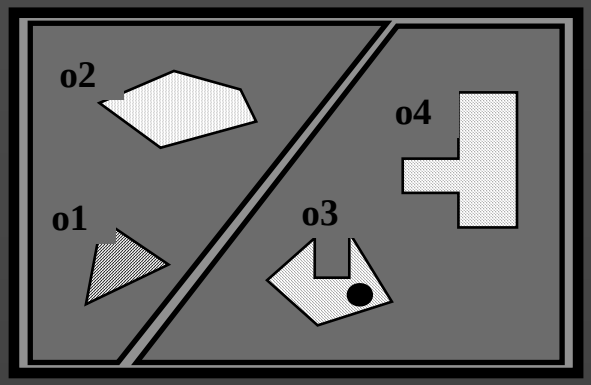
- between objects or between viewer-environment

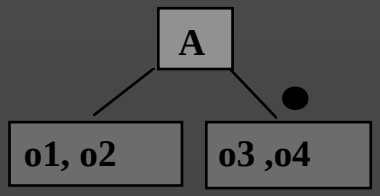
Classification

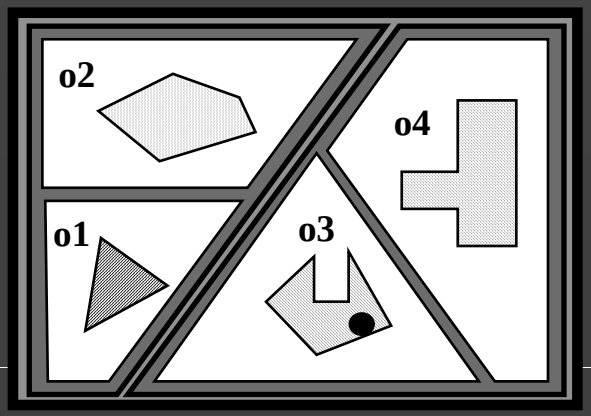
- point, ray, object (e.g. triggers, in shadow)

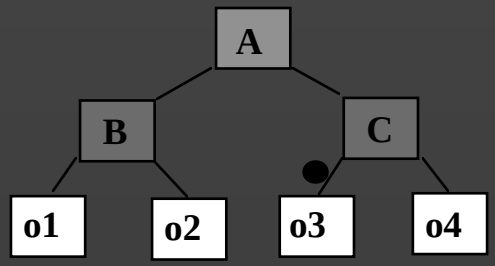

o2
o4
o1
o3
test point


o1, o2, o3 ,o4


o2
o4
o1
o3


A
o1, o2
o3 ,o4


o2
o4
o1
o3


A
B
C
o1
o2
o3
o4

Visibility Orderings

Rasterization

- visible surfaces - no numerical error, no over-write
- transparency - unlimited layers
- anti-aliasing - no sub-pixel buffers

Beam Tracing

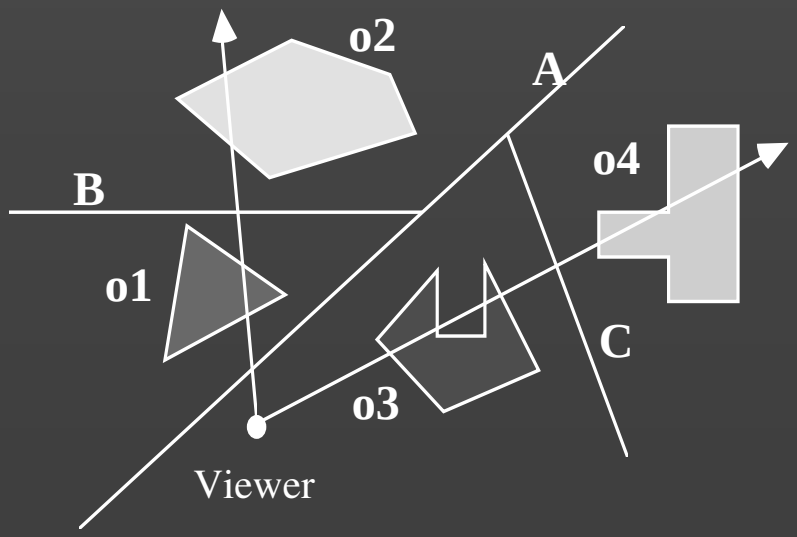
- occlusion culling - view dependent and dynamic
- shadows & spot lights - general volumetric
- reflections - planar mirrors

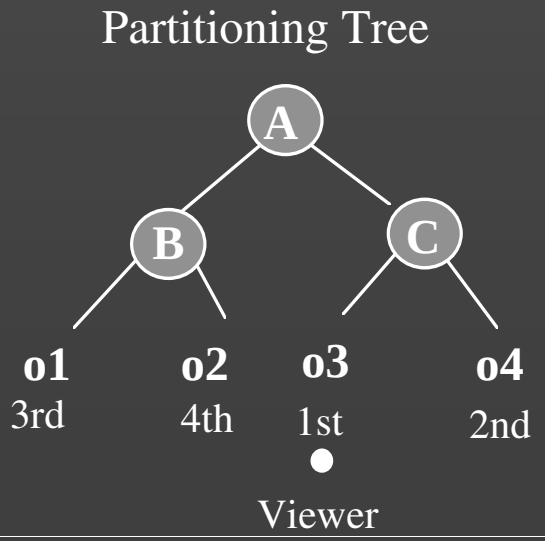
Ordering as Tree Traversal

Recursively order subtrees

- determine on which side of the root hyperplane the viewer lies

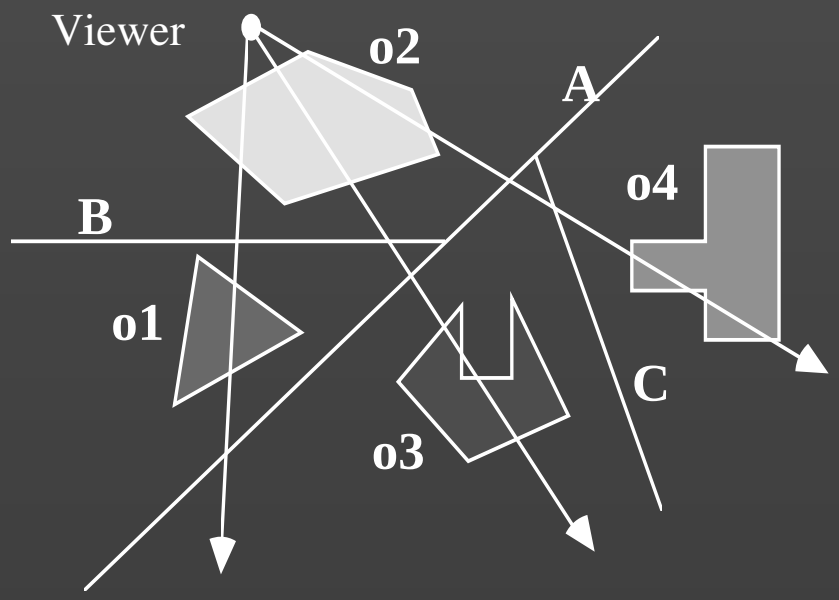
  near-subtree -> far-subtree

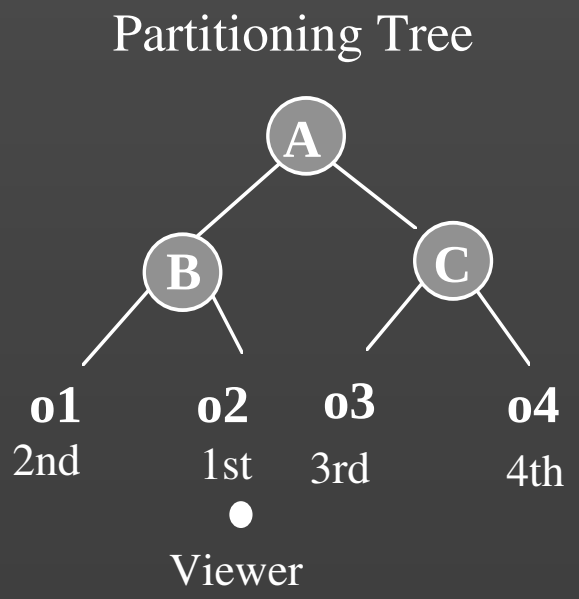




Ordering as Tree Traversal

Requires only O(T) operations





Subset Hierarchy

Visibility : Hierarchy of Separating Planes

- node v corresponds to a plane *separating* geometry in two subtrees rooted at v
- no comparisions between elements of each subtree
- O(n) sort

Intersections : Hierarchy of Bounding-Volumes

- node v corresponds to a convex region *containing* all geometry in the subtree rooted at v
- no intersection with bounding volume means no intersection with contents of volume -> O(log n)

Tree Merging

Modeling

- set operations for CSG
- interference detection, tolerancing

Interactive Inspection

- solid cut-aways, x-ray probe

Dynamics

- collision detection

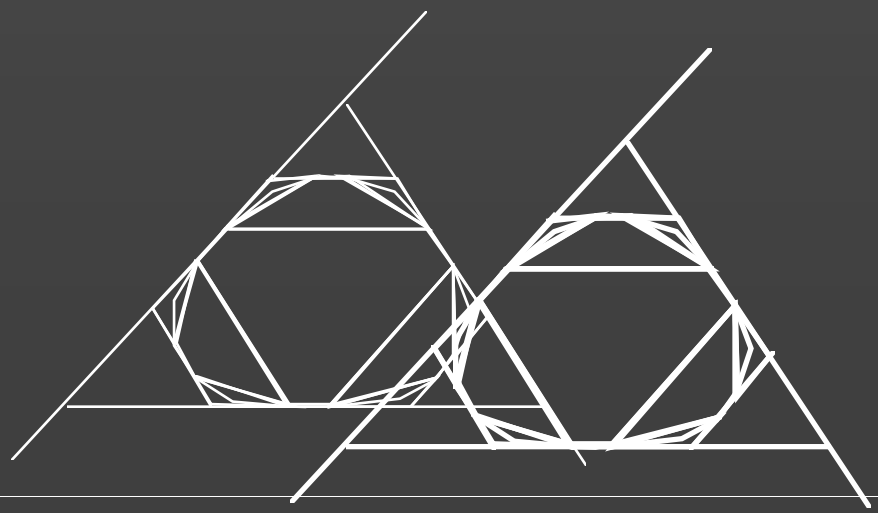
Rendering (merge object-trees into a model-tree)

- view-volume clipping/culling
- inter-object visibility orderings
- sorting for transparency
- computing and casting shadows

Tree Merging

Computes spatial relations between two objects

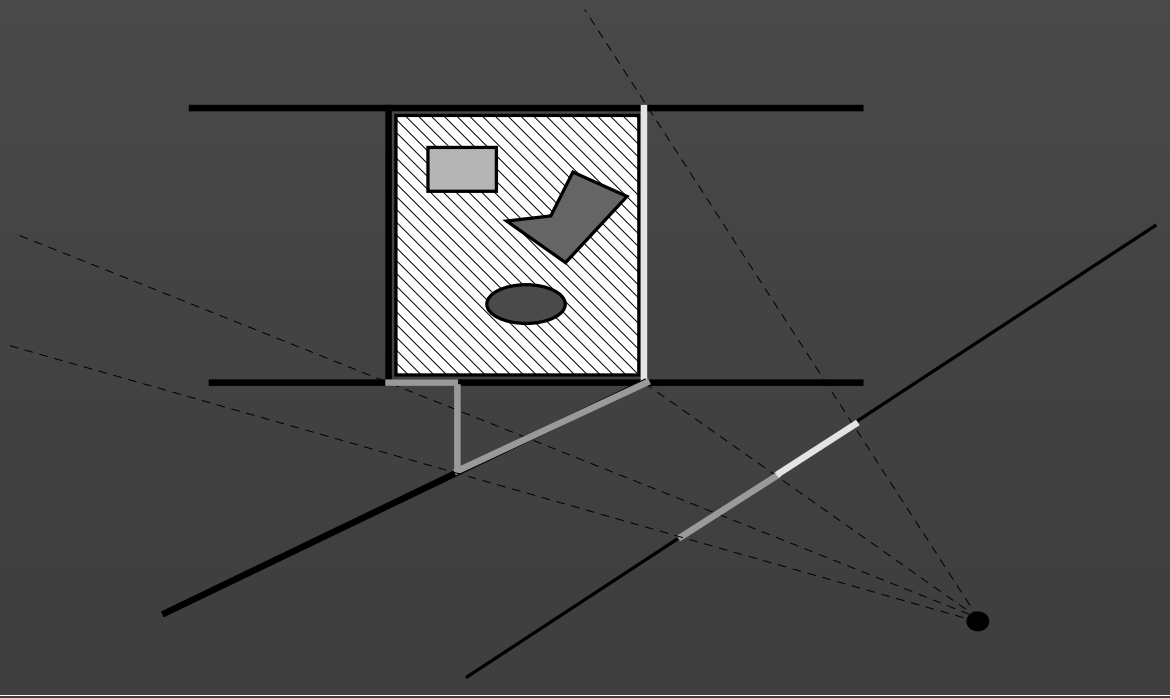
- new_tree = tree1 <op> tree2
- like merging sorted lists
- merges two bounding volumes hierarchies
- zooms in to regions of intersection



Occlusion Culling

Discard occluded subtrees without examining contents

- Project 3D model tree to 2D image tree
- Test silhouette of tree-region against 2D occlusion mask



Constructing Good Trees

Good Trees

Many trees represent the same set

- Partitioning Trees are a program for searching space
- There are many possible searches for the same set

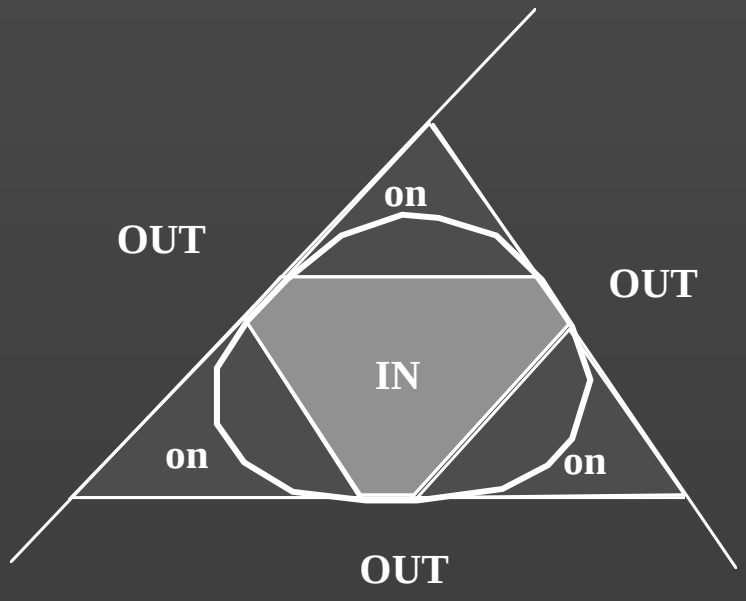
Which trees are the good ones?

- Quantitative answer: ones that minimize expected cost
- Qualitative answer: ones that represents set as a sequence of approximations

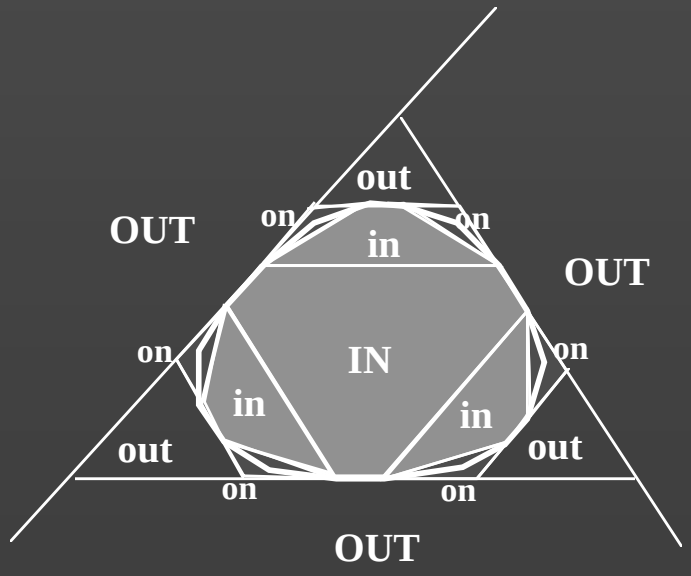
Sequence of Approximations

Exploit hierarchy of regions

- Regions decrease in size along any tree path
- Regions converge in the limit to the surface



1st level Approximation

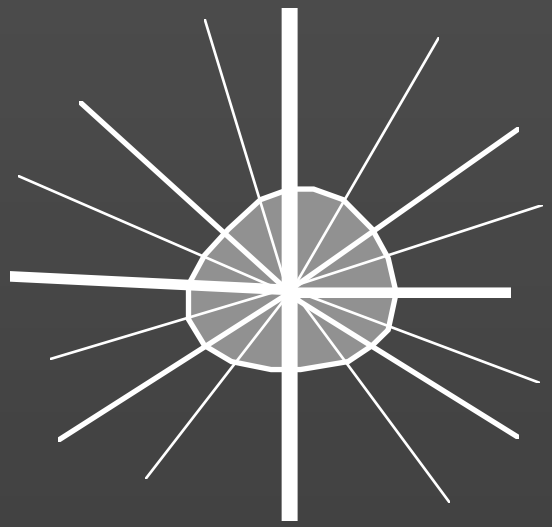


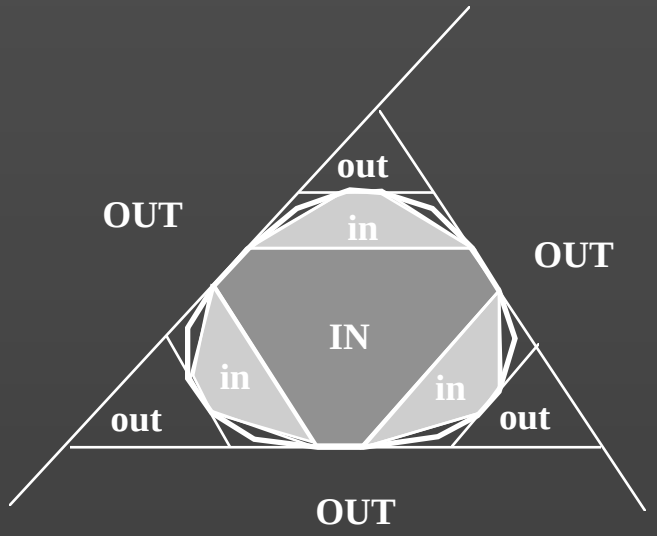
2nd level Approximation

Good vs. Bad Trees

Size and/or worst case not enough

- both trees are size O(n) and depth O(log n).





Expected case differentiates

Point	log n/log n	log n/ O(1+ E[n])
Line	O(n)/O(n)	log n / O(1+ E[n])

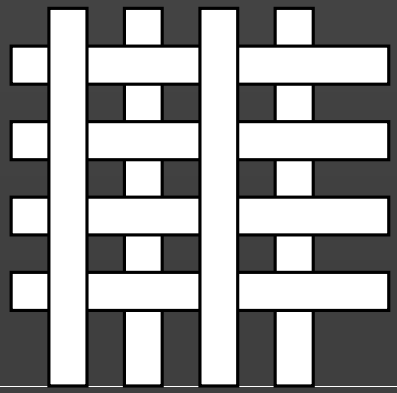
Worst Case Analysis

Basket weaving requires $O(n^2)$ size

- geometric versions of the Cartesian Product (checkerboard)
- true for any convex decomposition scheme

All examples violate the Principle of Locality

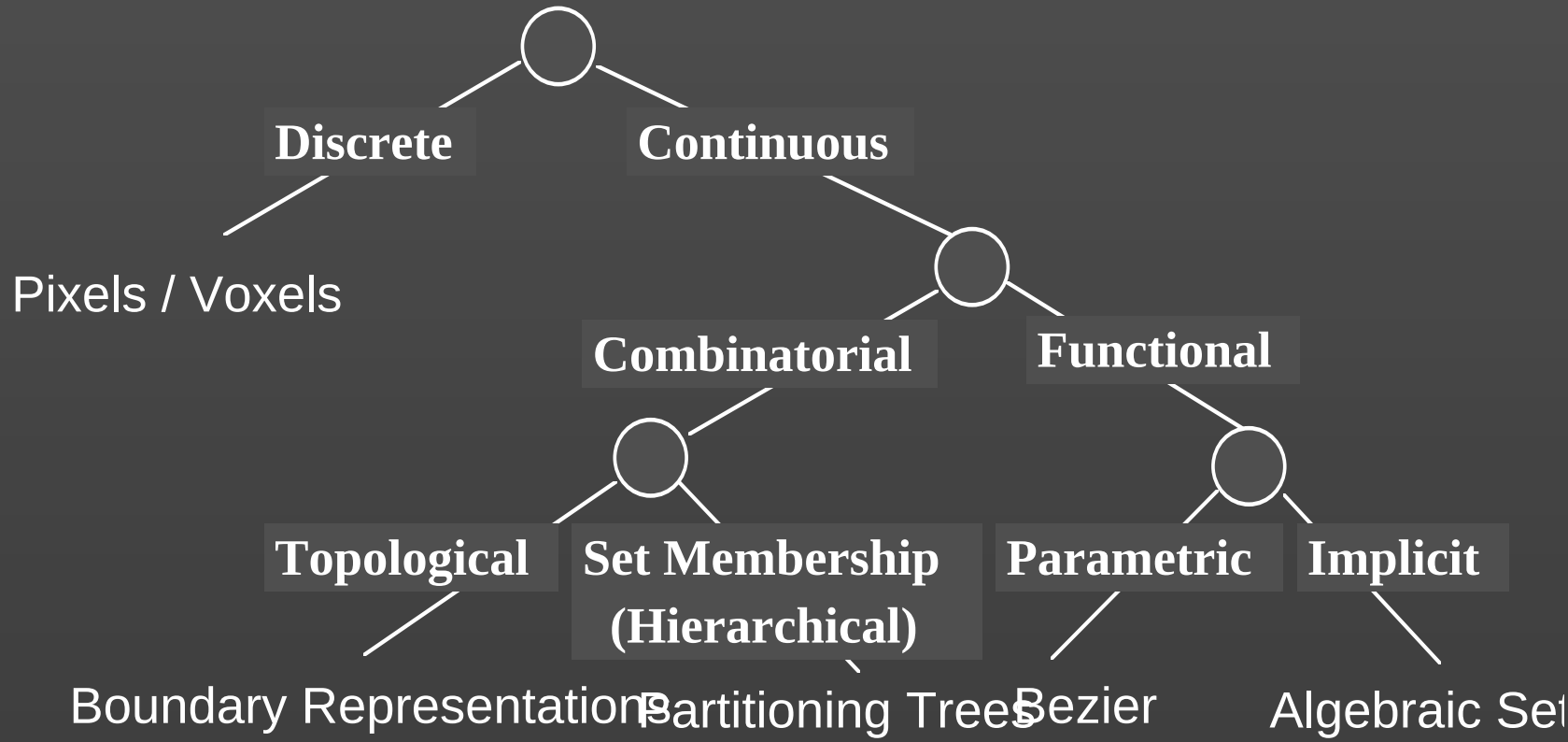
- typically, features are local not global
- spatial partitioning schemes accelerate computations only when features are local



Supporting Multiple Representations with Partitioning Trees

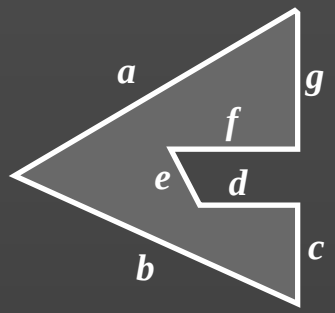
Conversions from other representations to Partitioning Trees

Taxonomy for Representations of Geometry

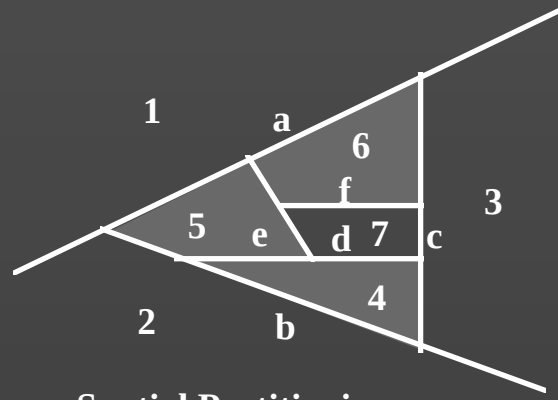


Converting B-reps to Trees

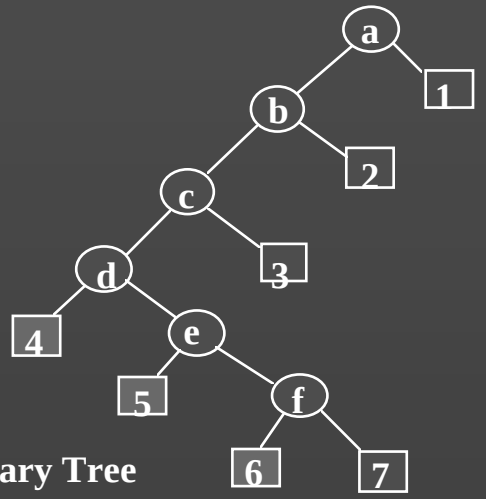
Intra-object Trees



Orignal B-rep



Spatial Partitioning



Binary Tree

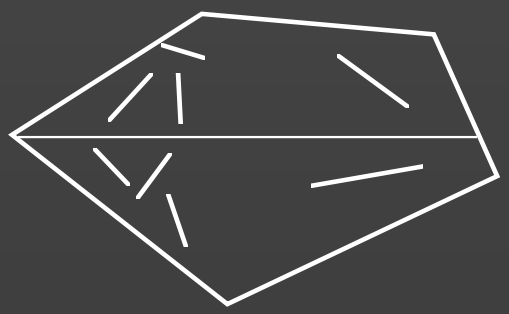
Good Trees ~ Huffman Codes

Probabilities ~= Size

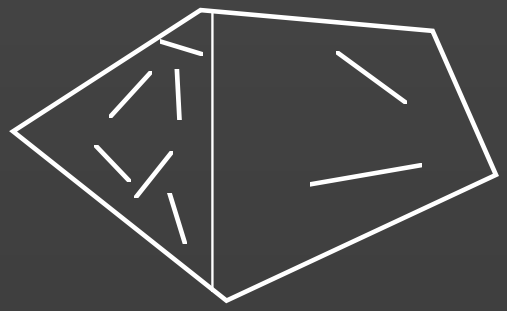
- We want short paths to high probability nodes, a.k.a. large cells; long paths to details

Tree reflects distribution of geometry

- Balanced optimal only for uniform distribution
- We want few pieces in large regions; many pieces in small regions



Balanced



Better

Computing Expected Cost

Minimizing expected cost == good trees

- Yields sequence of approximations rep.

Expected Cost Model

- $E_{cost}[T] = 1 + p^{-} * E_{cost}[T^{-}] + p^{+} * E_{cost}t[T^{+}]$
- branch probabilities depend upon region r, hyperplane h, and operation o.
- for point classification:

  $p+ = | r^{+} | / | r |$

  $p- = | r^{-} | / | r |$

Converting B-reps to Trees

Which hyperplanes?

- face hyperplanes
- edge hyperplanes
- approximating hyperplanes

Which order?

- minimize expected cost
- $E_{cost}[\ T\ ] = 1 + p^- * E_{cost}[\ T^-\ ] + p^+ * E_{cost}[\ T^+\ ]$
- can calculate p^+ and p^- exactly
- estimate $E_{cost}[\ T^+\ ] = |b^+|^n$, $E_{cost}[\ T^-\ ] = |b^-|^n$

Brep vs. Tree Sizes

Averaged results of converting b-reps to trees

- sizes ranged from 100's to 10,000's
- compression from co-planar faces

Type	tree/brep faces	tree/brep size
Buildings	1.25	.45
Man made	1.60	.75
Animate	2.00	1.0

Boundary Representations vs. Partitioning Trees

Competing or complementary ?

- express difference aspects of geometry: topology vs. hierarchical set membership
- b-reps great for specification of geometry and scan-conversion
- but for efficiency, must be paired with heirarchical structure and discrete algorithms

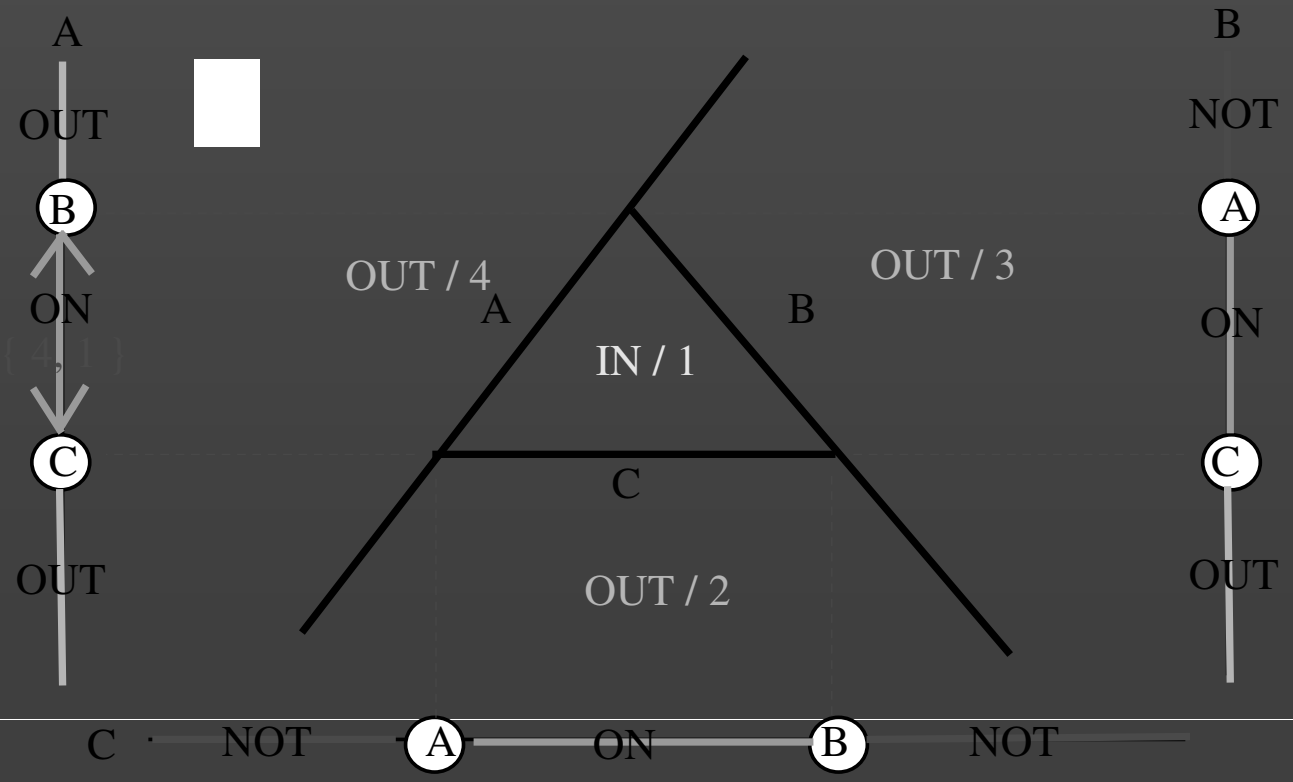
Partitioning Trees as glue

- provide strong coupling between topological/parametric and hierarchical implicits

Multi-Dimensional Partitioning Trees

Use lower dimensional trees for boundary elements

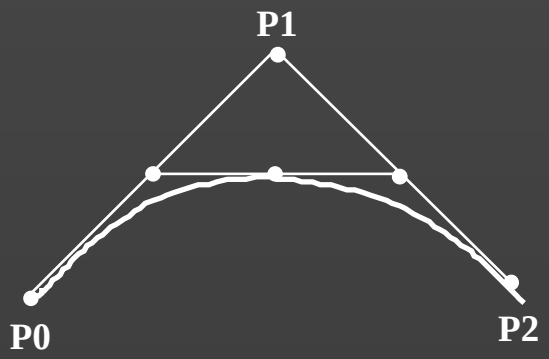
- recurse in dimension
- add incidence arcs between adjacent cells

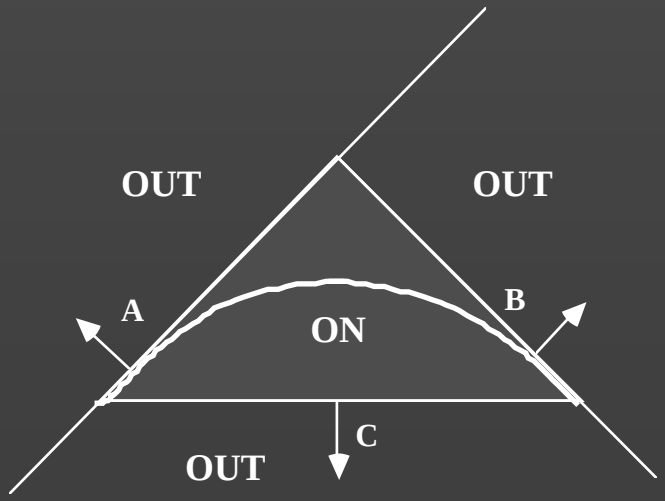


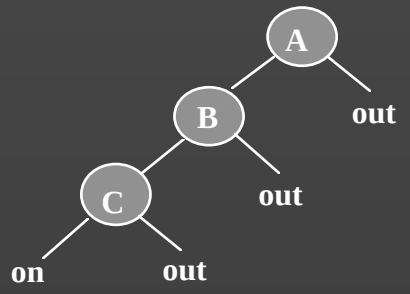
Bezier Curves as Partitioning Trees

Control polygon provides piecewise linear approximate

- use hierarchy of control polygons to build a tree
- use 3-valued logic: IN, OUT, ON





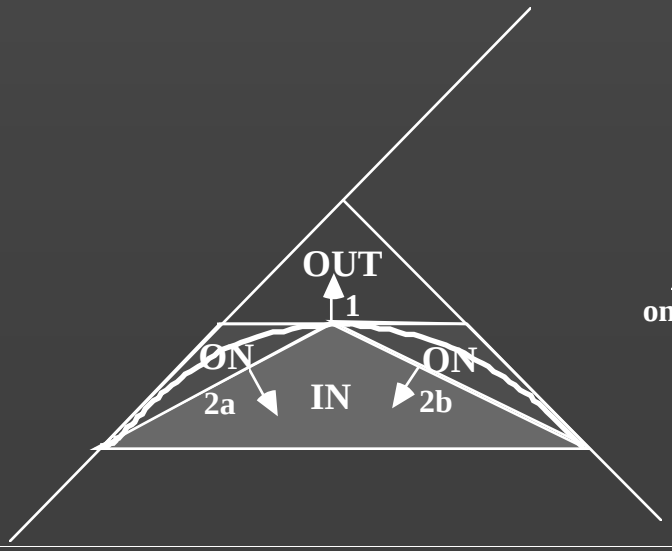


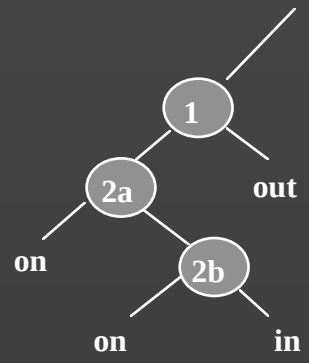
Bezier Segment Tree

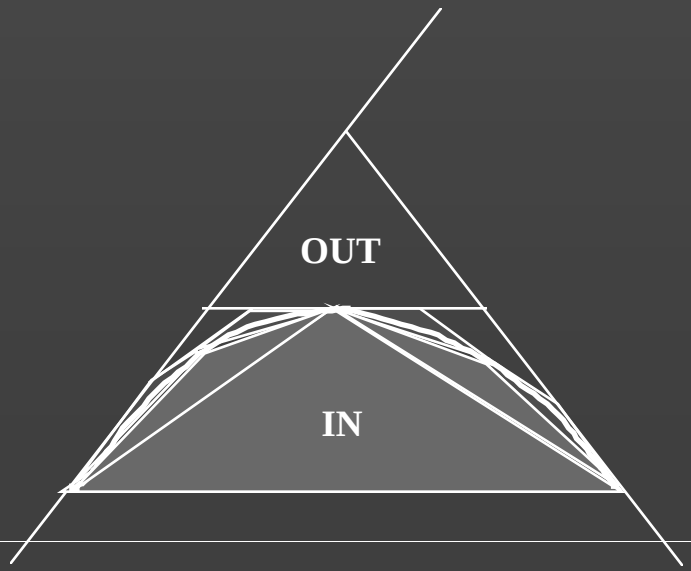
Remember mid-point subdivsion as Tree

- improves approximation of both interior and exterior at each step
- converges in the limit to the surface

ON area = error of approximation

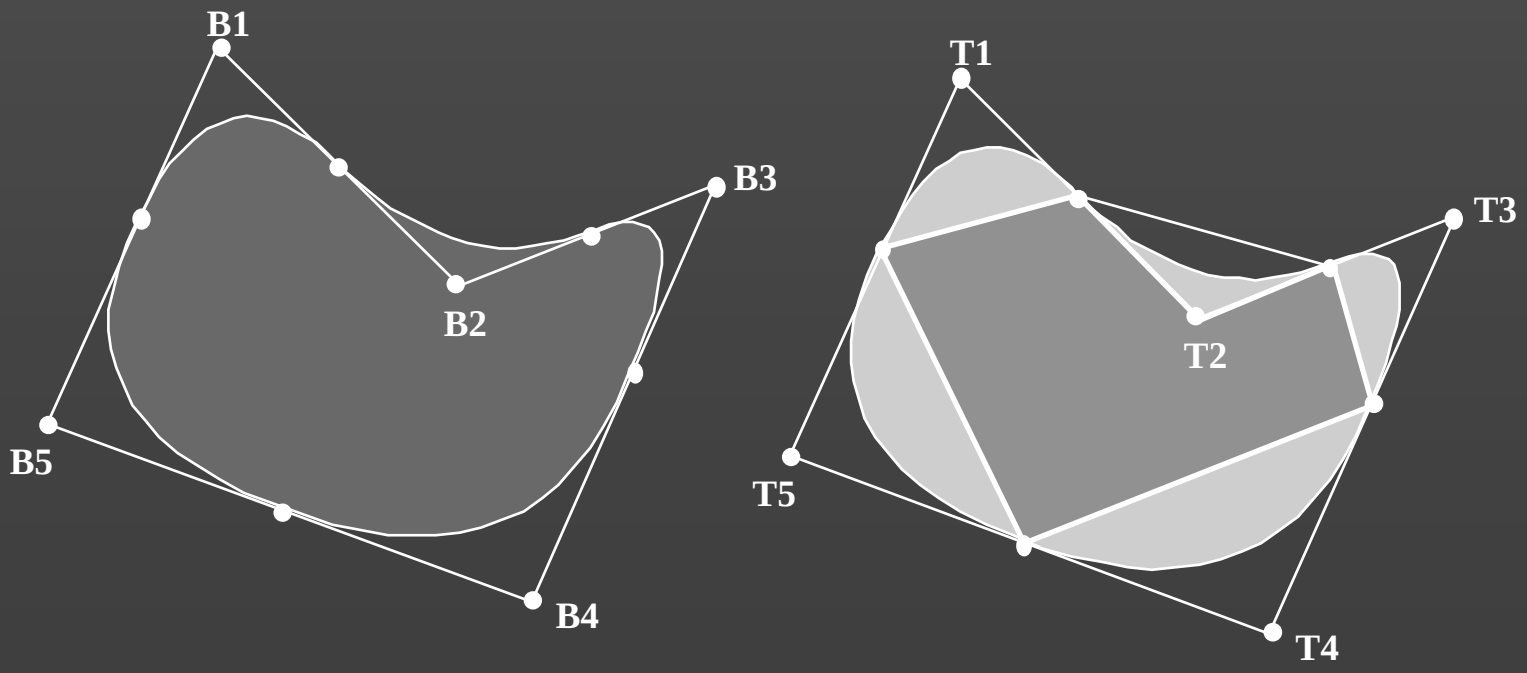






B-Splines as Trees

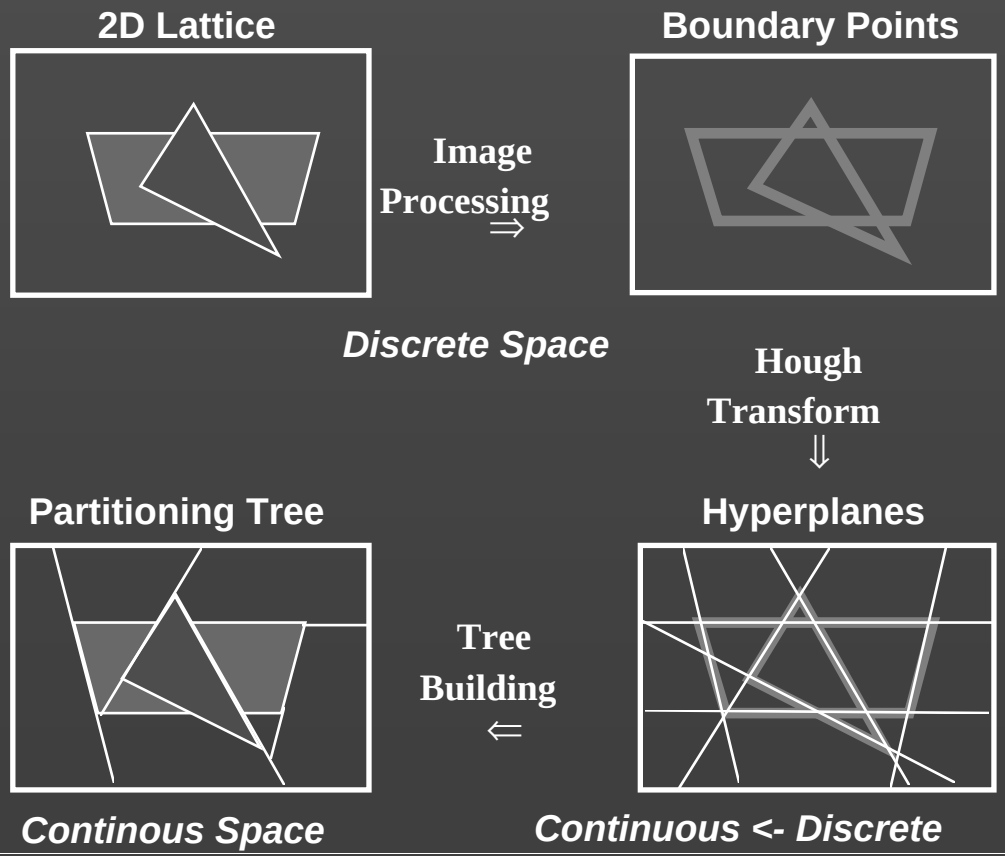
Merge segment trees to form b-spline bounded solid.



Trees from Pixels/Voxels

Segments image into convex regions

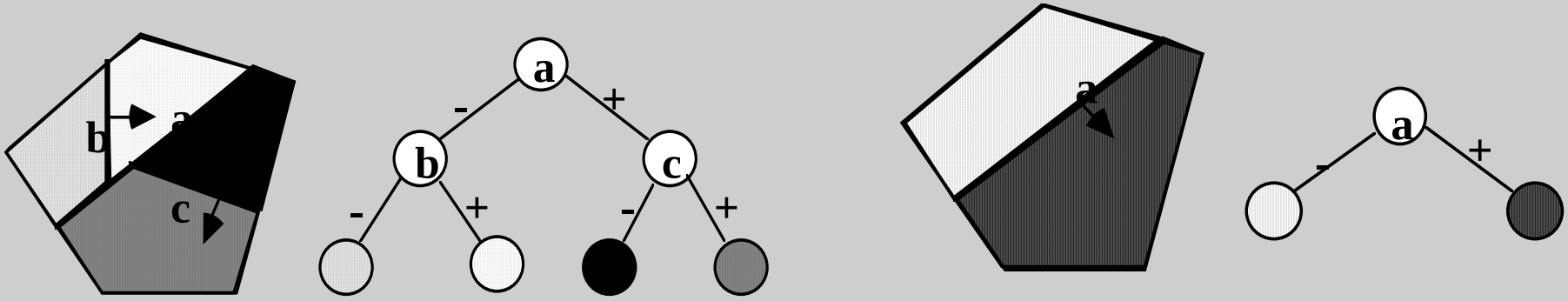
- applicable to visualization, computer vision and video compression



Multi-Resolution Trees

Representation of Function Spaces

- define a function piecewise by associating a continuous function at each cell
- at internal regions, use least-squares approximate
- truncating the tree produces an approximation, ala the Taylor series



Before pruning

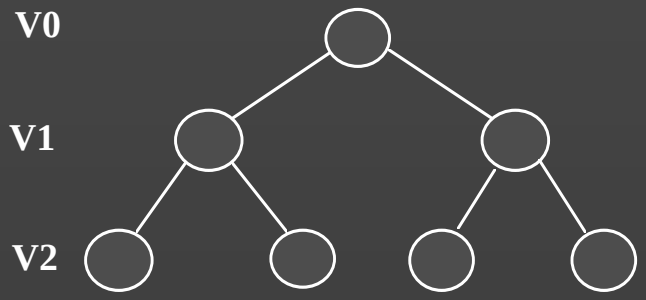
After pruning

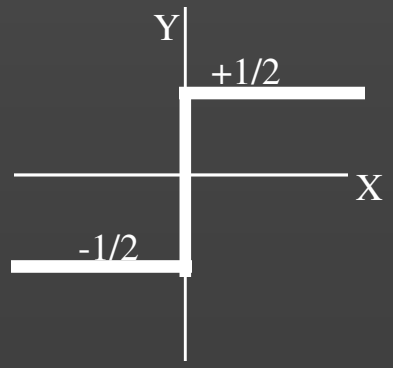
Partitioning Tree Wavelets

Wavelets (piecewise constant)

- nested sequence of function spaces

  each tree level is a function space
- orthogonality guarenteed by least-squares.
- hp specifies translation and rotation of wavelet
- region specifies compact support





Affine invariant but continuity sacrificed

Summary

Primary operations

- intersections : interpret tree as a hierarchy of bounding volumes
- visibility orderings : interpret tree as a hierarchy of separating planes

Secondary operations

- set operations, collision detection, view-volume clipping
- visible surfaces, transparency, visibility culling, shadows, ray-tracing, radiosity
- image segmentation for computer vision and compression.