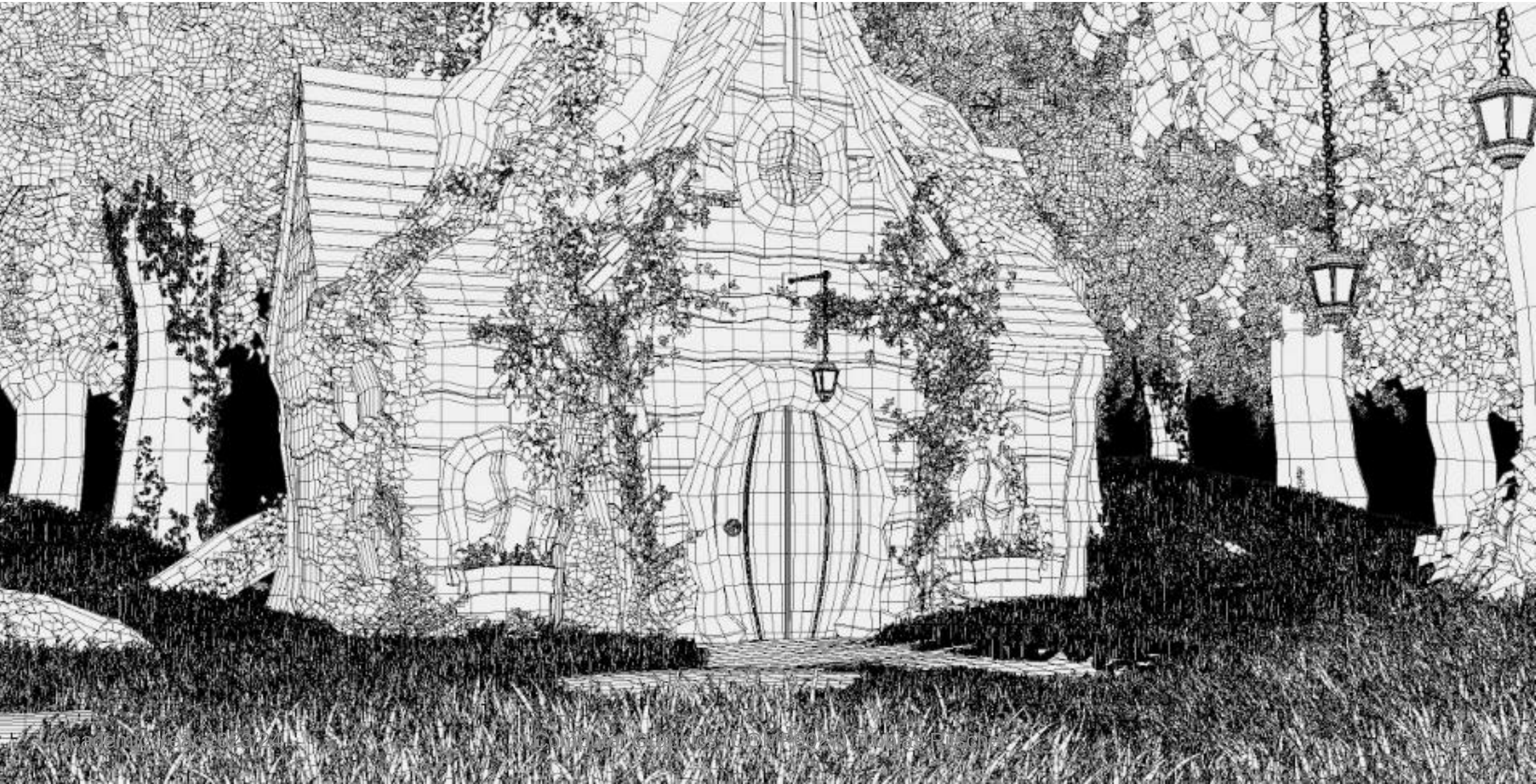
Computer Graphics (COMP0027) 2023/24

Bounding Volumes

Tobias Ritschel

Polygonal meshes



Trivial

- Simple
  - Loop over all rays
  - Loop over all polygons
  - Intersect
- Complexity
  - Quadratic in ray-poly
  - linear in polys
- Can we do better?

Acceleration

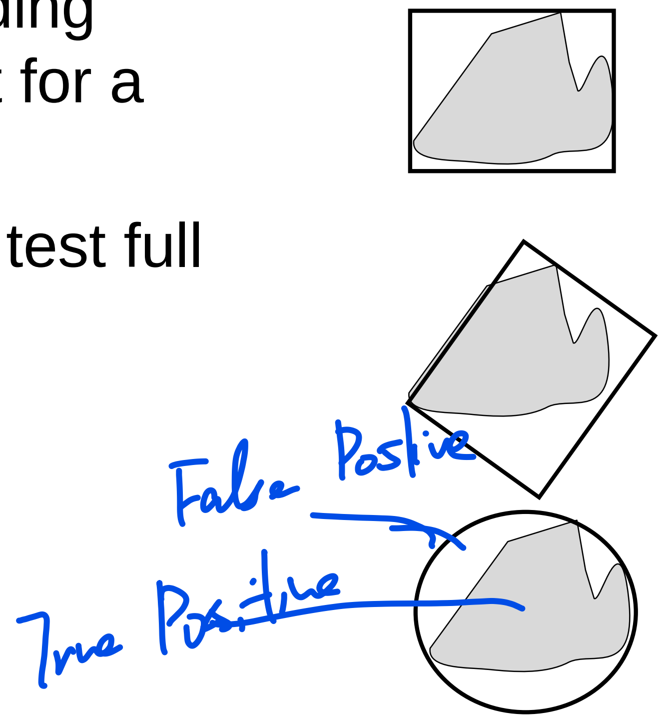
- In ray tracing 90% of cost is in ray-polygon intersections
- As described so far, $O(n)$ where n is number of polygons
- As in general algorithms in CS, optimization can be done by appropriate & efficient representations
- What can we exploit?

Overview

- Many potential data structures
- Choose three common ones
  - Bounding volumes
  - Hierarchical bounding volumes
  - kD-Tree
- There are others such as octrees and quad trees.
- Note, that this is related to issues of hashing & indexing in tables.

Bounding Volume

- Find a tight bounding volume and use it for a *reject test*
- If hit volume then test full object



Axis Aligned Bounding Box (AABB)

Bounding Box

OBB 还是 EBB

Bounding Sphere 未有

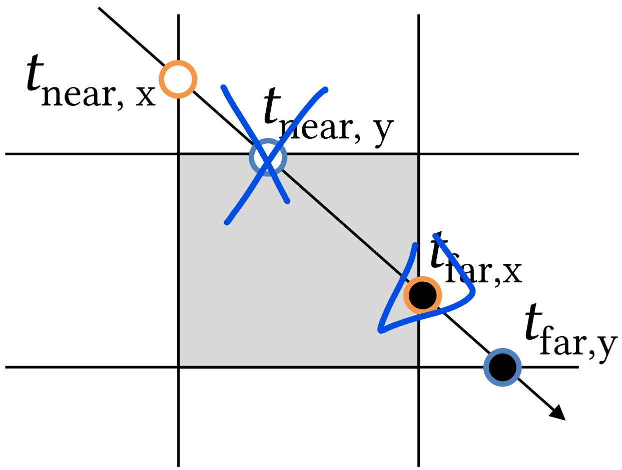
Fast BV Tests (AABB)

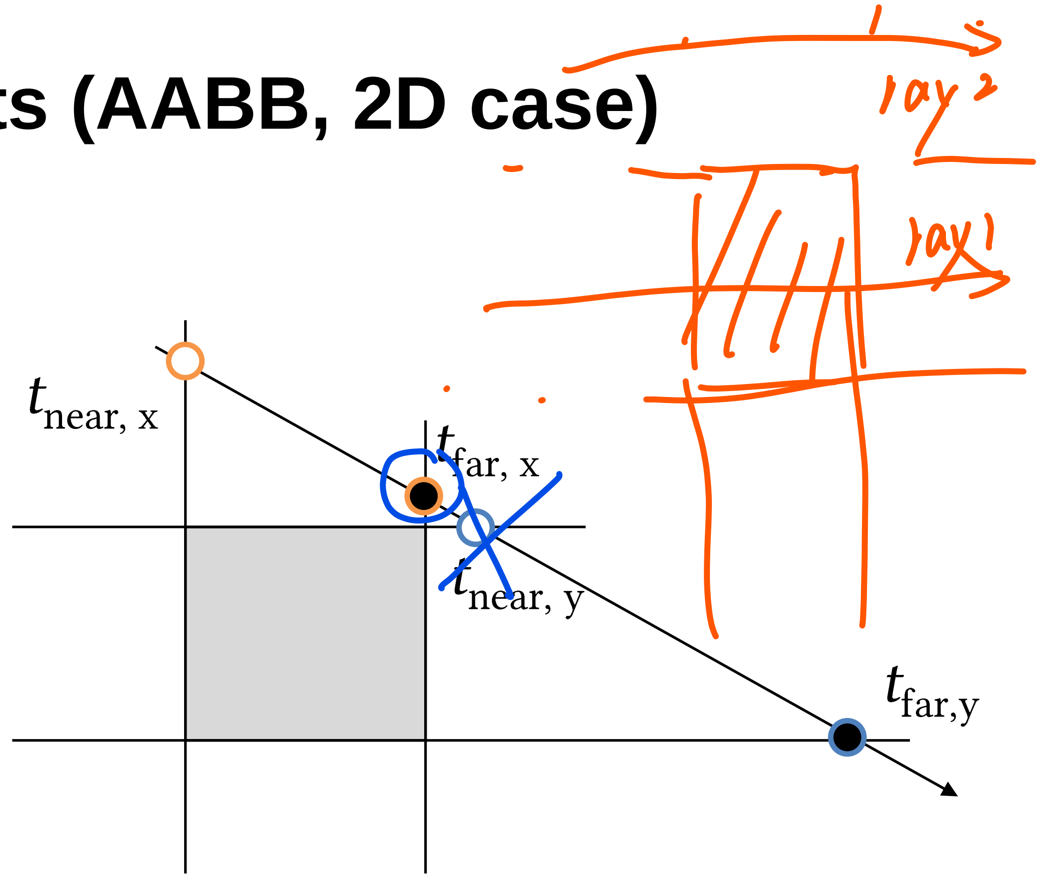
- Box-Ray test (when box planes parallel to axes)
  - A box is three sets of parallel planes, each set orthogonal to the other two
  - Ray defined by $\mathbf{r}(t) = \mathbf{p} + t\,\mathbf{d}$
    - Calculate t_{near} for each of the three axis
    - Find max of the three t_{near}
    - Calculate t_{far} for each of the three axis
    - Find min of the three t_{far}
  - If max t_{near} is greater that min t_{far}, then the box is not intersected

txnear tynear tznear

txfar tyfar tzfar

Fast BV Tests (AABB, 2D case)





Pros and Cons

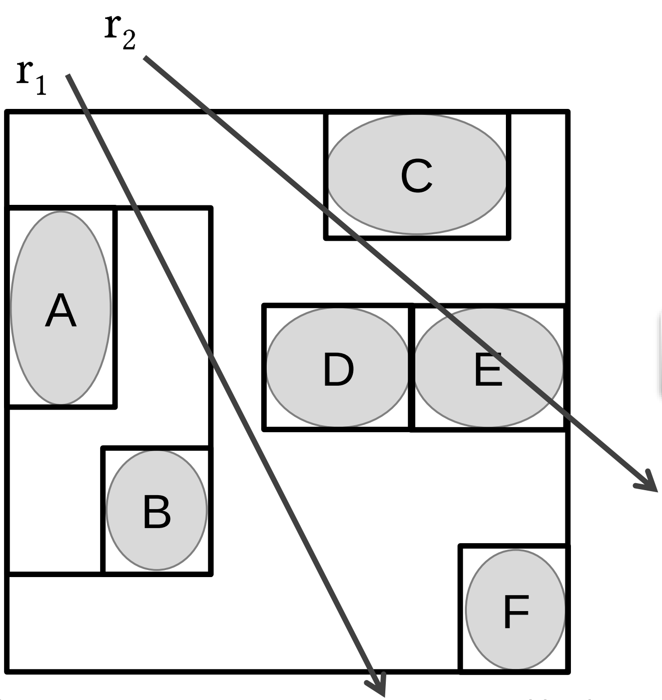
- Utility is a trade-off between the simplicity of the bounding volume and the “void” space
- Pros: a bounding volume can be extremely efficient when it is unlikely that the volume will be “hit” (for ray tracing, visibility, etc.)
- Cons: the “void space” may be very large, leading to many redundant expensive tests.

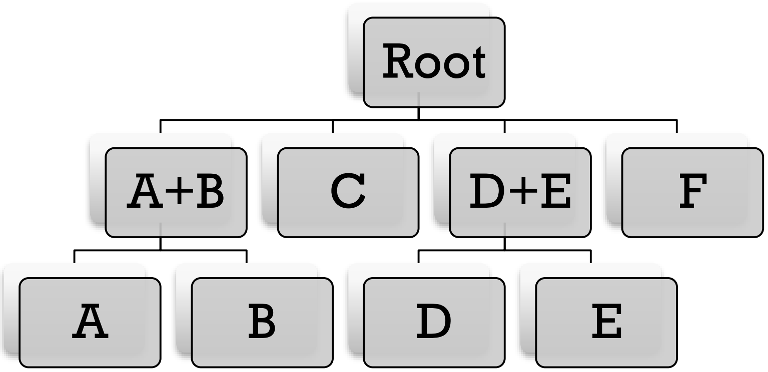
- Solution: better intermediate representations.

Bounding Volume Hierarchy

BVH

- Organise a hierarchy of bounding volumes
  - Bounding volumes of bounding volumes





r_1 tests with A+B, then A and B

r_2 tests with C and D+E, then D and E

Bounding Volume Hierarchy

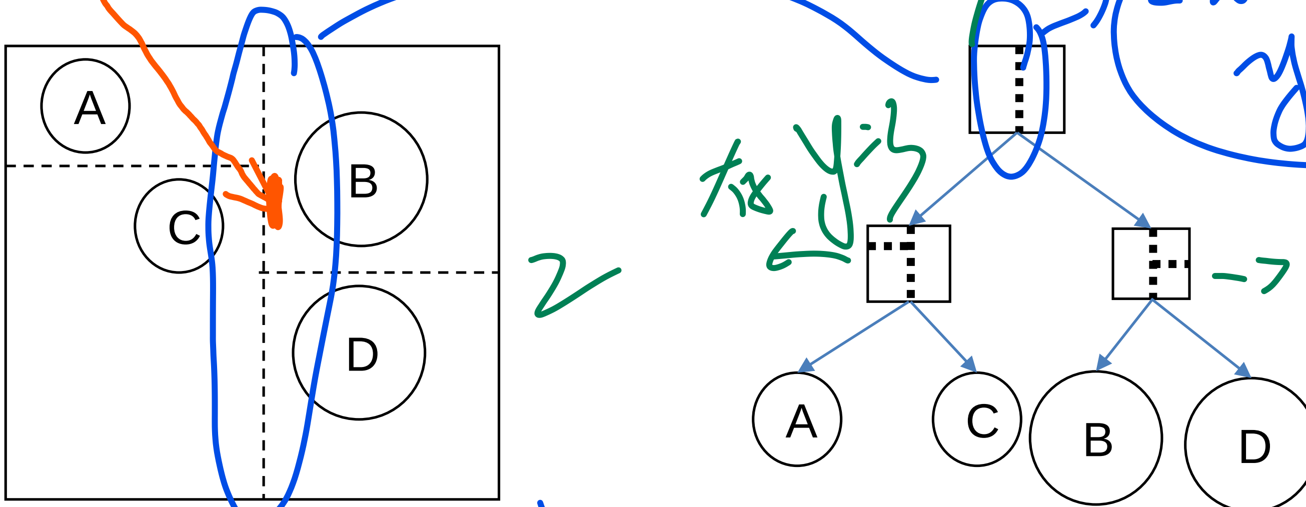
- Pros:
  - Very good adaptivity
  - Efficient traversal $O(\log n)$
- Cons
  - How to arrange BVs?

kD-Tree

→只存 x=4

- One of a class of spatial data structure that partition space
- Uses horizontal and vertical splits

是存 x=? 或 y=? 要可以



y

3,

2

4

存 y的

→存 y=2

Traversing a kD-tree

穿过

recursive

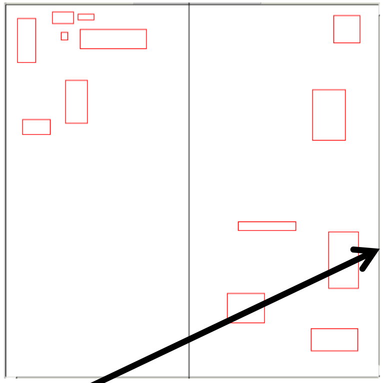
```
kdTree-RayIntersect(Ray, Node)
    If node is a leaf, intersect ray with objects
    else
        Clip ray to near side of the split (RayNear)
        kdTree-RayIntersect(RayNear, Node->near)
        If (no intersection)
            Clip ray to far side of the plane (RayFar)
            kdTree-RayIntersect(RayFar, Node->far)
```

碰到分割线就停…

返回左子树没找到 找右子树 (假设ray是从左到右)

Building good kD-trees

- What split do we really want?
  - Main Idea: The one that makes ray tracing cheap
  - Write down an expression of cost and minimize it
  - Cost Optimization
- What is the cost of tracing a ray through a cell?

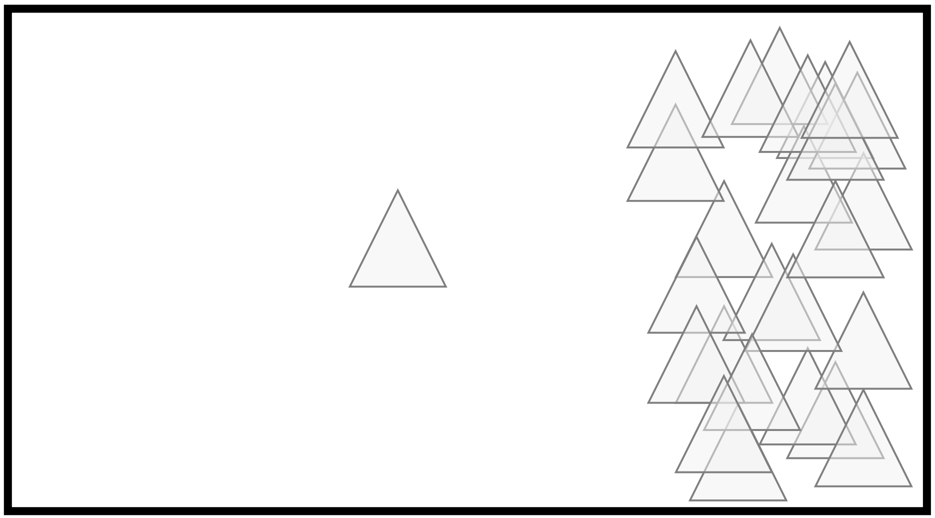


1(a) 撞到的概率

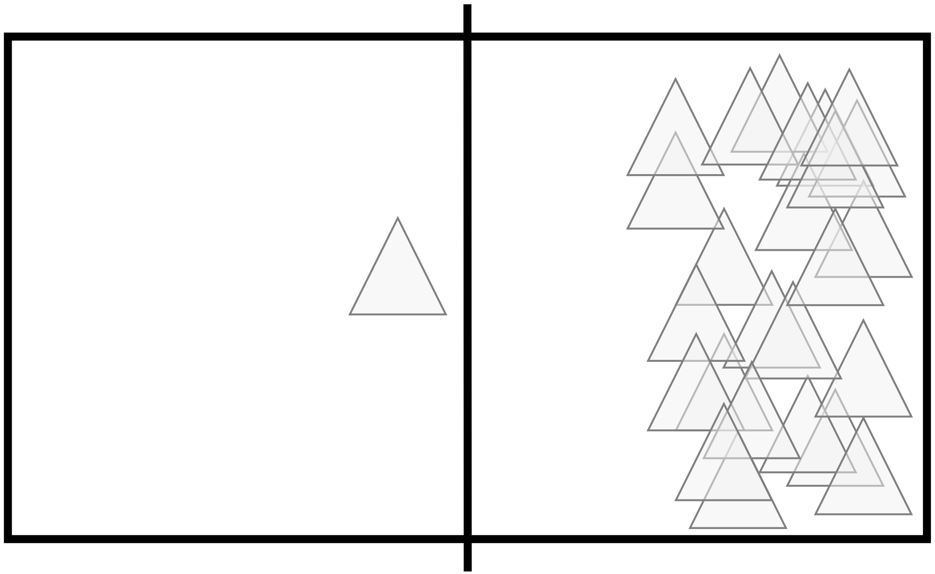
$$\text{cost(cell)} = \text{cost}_{\text{trav}} + \text{prob(hit L) cost(L)} + \text{prob(hit R) cost(R)}$$

撞到左区域后，遍历时间的cost

Splitting with Cost in Mind

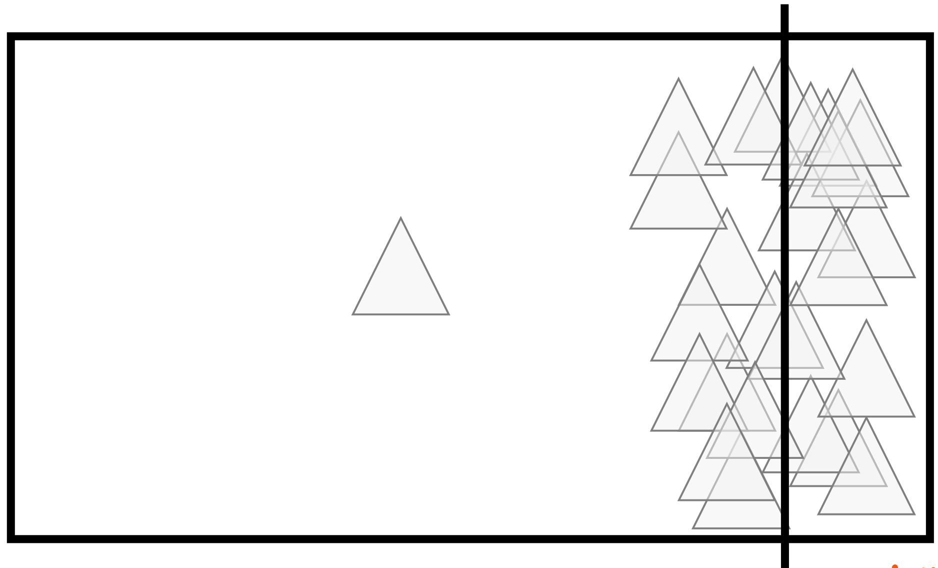


Split in the middle



- Makes the L & R probabilities equal
- Pays no attention to the L & R costs

Split at the Median



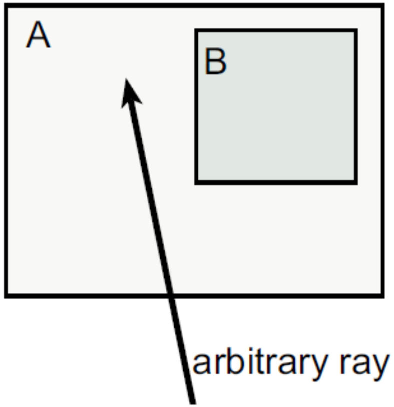
- Makes the L & R costs equal
- Pays no attention to the L & R probabilities

→ 所以分割线不一定要穿过中间

Surface Area and Rays

- Probability of a ray hitting an object that is completely inside a cell is:

$$\mathrm{prob}(\mathrm{hit}_B \mid \mathrm{hit}_A) = S_b / S_a$$

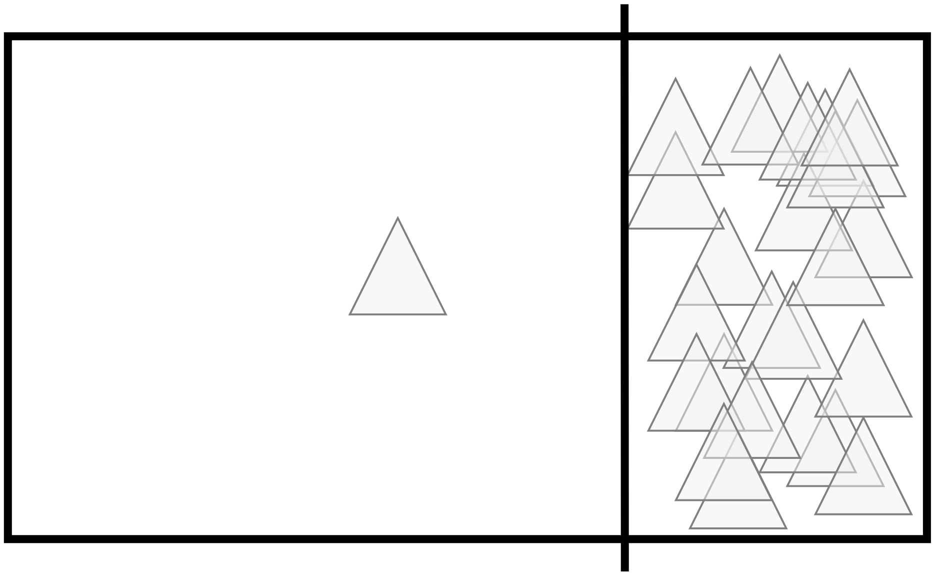


Surface Area Heuristic

- Need the probabilities
  - Turns out to be proportional to surface area
- Need the child cell costs
  - Simple triangle count works great (very rough approx.)

$$
\begin{aligned}
\mathrm{cost}(\mathrm{cell}) &= \mathrm{C}_{\mathrm{trav}} + \mathrm{prob}(\mathrm{hit}\ L)\ \mathrm{cost}(L) + \quad \mathrm{prob}(\mathrm{hit}\ R) \\
\mathrm{cost}(R) &= \mathrm{C}_{\mathrm{trav}} + \mathrm{SA}(L)\ \mathrm{triCount}(L) + \mathrm{SA}(R)\ \mathrm{triCount}(R)
\end{aligned}
$$

Cost-Optimized Split



- Automatically and rapidly isolates complexity
- Produces large chunks of empty space

Recap

- Several techniques can be applied for accelerating the ray intersection tests with the scene
- Bounding volumes are a very obvious acceleration and almost always a good idea
- Bounding volume hierarchies are useful, but geometry is often irregularly spaced around the environment, so BVH is inefficient in empty space
- kD-Trees are one of a class of spatial data structure that balance precision in implementation with general utility. Very commonly used in practice