Lecture 13:
Ray Tracing 2 (Acceleration)
Last Lecture

- Shadow mapping for rasterizers
- Introduction to ray tracing
- Ray generation
- Ray object intersection
  - Ray sphere intersection
  - Ray implicit surface intersection
  - Ray triangle intersection = Ray plane intersection + inside test
Accelerating Ray-Surface Intersection
Ray Tracing – Performance Challenges

Simple ray-scene intersection

- Exhaustively test ray-intersection with every triangle
- Find the closest hit (with minimum $t$)

Problem:

- Naive algorithm $= \#\text{pixels} \times \# \text{triangles} (\times \#\text{bounces})$
- Very slow!

For generality, we use the term objects instead of triangles later (but doesn’t necessarily mean entire objects)
Ray Tracing – Performance Challenges

San Miguel Scene, 10.7M triangles
Ray Tracing – Performance Challenges

Plant Ecosystem, 20M triangles
Bounding Volumes
Bounding Volumes

Quick way to avoid intersections: bound complex object with a simple volume

- Object is fully contained in the volume
- If it doesn’t hit the volume, it doesn’t hit the object
- So test bvol first, then test object if it hits
Ray-Intersection With Box

Could intersect with 6 faces individually

Better way: box is the intersection of 3 pairs of slabs
Ray Intersection with Axis-Aligned Box

2D example; 3D is the same! Compute intersections with slabs and take intersection of $t_{\min}/t_{\max}$ intervals.

Intersections with $x$ planes
How do we know when the ray misses the box?

Intersections with $y$ planes

Final intersection result

Note: $t_{\min} < 0$
Ray Intersection with Axis-Aligned Box

- Recall: a box (3D) = three pairs of infinitely large slabs

- Key ideas
  - The ray enters the box **only when** it enters all pairs of slabs
  - The ray exits the box **as long as** it exits any pair of slabs

- For each pair, calculate the $t_{\text{min}}$ and $t_{\text{max}}$ (negative is fine)

- For the 3D box, $t_{\text{enter}} = \max\{t_{\text{min}}\}$, $t_{\text{exit}} = \min\{t_{\text{max}}\}$

- If $t_{\text{enter}} < t_{\text{exit}}$, we know the ray **stays a while** in the box (so they must intersect!) (not done yet, see the next slide)
Ray Intersection with Axis-Aligned Box

• However, ray is not a line
  - Time $t$ should always be positive for physical correctness!

• What if $t_{\text{exit}} < 0$?
  - The box is “behind” the ray — no intersection!

• What if $t_{\text{enter}} < 0$?
  - The ray’s origin is inside the box — still intersection!
Optimize Ray-Plane Intersection For Axis-Aligned Planes?

General

Perpendicular to x-axis

$t = \frac{(p' - o) \cdot N}{d \cdot N}$

3 subtractions, 6 multiplies, 1 division

$t = \frac{p'_{x} - o_{x}}{d_{x}}$

1 subtraction, 1 division
Uniform Spatial Partitions (Grids)
Preprocess – Build Acceleration Grid

1. Find bounding box
Preprocess – Build Acceleration Grid

1. Find bounding box
2. Create grid
Preprocess – Build Acceleration Grid

1. Find bounding box
2. Create grid
3. Store each object in overlapping cells
Ray-Scene Intersection

- Step through grid in ray traversal order
- For each grid cell, test intersection with all objects stored at that cell
Grid Resolution?

One cell

- No speedup
Grid Resolution?

- Too many cells
  - Inefficiency due to extraneous grid traversal
Grid Resolution?

Heuristic:

- \#cells = C \times \#objs
- C \approx 27 \text{ in 3D}
Uniform Grids – When They Work Well

Grids work well on large collections of objects that are distributed evenly in size and space
“Teapot in a stadium” problem
Spatial Partitions
Spatial Hierarchies
Spatial Hierarchies

Slide courtesy of Prof. Ren Ng, UC Berkeley
Spatial Hierarchies

Slide courtesy of Prof. Ren Ng, UC Berkeley
Spatial Partitioning Examples

Oct-Tree  KD-Tree  BSP-Tree

Note: you could have these in both 2D and 3D. In lecture we will illustrate principles in 2D.
Object Partitions & Bounding Volume Hierarchy (BVH)
Spatial vs Object Partitions

Spatial partition (e.g. KD-tree)
- Partition space into non-overlapping regions
- Objects can be contained in multiple regions

Object partition (e.g. BVH)
- Partition set of objects into disjoint subsets
- Bounding boxes for each set may overlap in space
Bounding Volume Hierarchy (BVH)
Bounding Volume Hierarchy (BVH)
Bounding Volume Hierarchy (BVH)
Bounding Volume Hierarchy (BVH)
Bounding Volume Hierarchy (BVH)

Internal nodes store
- Bounding box
- Children: reference to child nodes

Leaf nodes store
- Bounding box
- List of objects

Nodes represent subset of primitives in scene
- All objects in subtree
BVH Pre-Processing

- Find bounding box
- Recursively split set of objects in two subsets
- Recompute the bounding box of the subsets
- Stop when there are just a few objects in each set
- Store obj reference(s) in each leaf node
BVH Pre-Processing

Choosing the set partition

• Choose a dimension to split
• Heuristic #1: Always choose the longest axis in node
• Heuristic #2: Split node at location of median object

Termination criteria?

• Heuristic: stop when node contains few elements (e.g. 5)
BVH Recursive Traversal

```java
Intersect(Ray ray, BVH node) {
    if (ray misses node.bbox) return;

    if (node is a leaf node)
        test intersection with all objs;
    return closest intersection;

    hit1 = Intersect(ray, node.child1);
    hit2 = Intersect(ray, node.child2);

    return the closer of hit1, hit2;
}
```
Thank you!