CS 636
Advanced Rendering Techniques

Dr. David Breen
Online
Wednesday 6PM → 8:50PM
Presentation 4
4/20/22

Questions from Last Week?
- Color models
- Light models
- Phong shading model
- Assignment 2

Slide Credits
- Leonard McMillan, Seth Teller, Fredo Durand, Barb Cutler - MIT
- David Luebke - University of Virginia
- Matt Pharr - Stanford University
- Jonathan Cohen - Johns Hopkins U.
- Kevin Suffern - University of Technology, Sydney, Australia

Motivation

Extra rays needed for these effects
- Distributed Ray Tracing
  - Soft shadows
  - Anti-aliasing (getting rid of jaggies)
  - Glossy reflection
  - Motion blur
  - Depth of field (focus)

Shadows
- one shadow ray per intersection per point light source
20

Soft Shadows
- Multiple shadow rays to sample area light source

21

Antialiasing – Supersampling
- Multiple rays per pixel

22

Reflection
- One reflection ray per intersection

23

Glossy Reflection
- Multiple reflection rays

24

Motion Blur
- Sample objects temporally

25

Depth of Field
- Multiple rays per pixel
Algorithm Analysis

- Ray casting
- Lots of primitives
- Recursive
- Distributed Ray Tracing Effects
- Soft shadows
- Anti-aliasing
- Glossy reflection
- Motion blur
- Depth of field

\[ \text{cost} \leq \text{height} \times \text{width} \times \text{num primitives} \times \text{intersection cost} \times \text{num shadow rays} \times \text{supersampling} \times \text{num glossy rays} \times \text{num temporal samples} \times \text{max recursion depth} \times \ldots \]

can we reduce this?

26

Bounding Regions

Acceleration of Ray Casting

- Goal: Reduce the number of ray/primitive intersection tests

Conservative Bounding Region

- First check for an intersection with a conservative bounding region
- Early reject

29

27

28

Conservative Bounding Regions

- tight → avoid false positives
- fast to compute
- fast to intersect

Bounding Volumes

What makes a "good" bounding volume?
- Tightness of fit (expressed how?)
- Easy to compute
- Simplicity of intersection
- Total cost = \( b\cdot B + i\cdot I + S \)
  - \( b \): # times volume tested for intersection
  - \( B \): cost of ray-volume intersection test
  - \( i \): # times item is tested for intersection
  - \( I \): cost of ray-item intersection test
  - \( S \): cost to compute BV parameters

Spheres
- Cheap intersection test
- Poor fit
- Somewhat costly to fit to data

Axis-aligned bounding boxes (AABBs)
- Relatively cheap intersection test
- Usually better fit
- Trivial to fit to data

Oriented bounding boxes (OBBs)
- Medium-expensive intersection test
- Very good fit (asymptotically better)
- Medium-difficult to fit to data
Bounding Volumes

- Slabs (parallel planes)
  - Comparatively expensive
  - Very good fit
  - Very difficult to fit to data

Intersection with Axis-Aligned Box

- For all 3 axes, calculate the intersection distances $t_1$ and $t_2$
- $t_{near} = \max (t_{x_1}, t_{y_1}, t_{z_1})$
- $t_{far} = \min (t_{x_2}, t_{y_2}, t_{z_2})$
- If $t_{near} > t_{far}$, box is missed
- If $t_{far} < 0$, box is behind
- If box survived tests, report intersection at $t_{near}$

Bounding Box of a Triangle

- $(x_{min}, y_{min}, z_{min})$
- $(x_{max}, y_{max}, z_{max})$

Bounding Box of a Sphere

- $(x, y, z) = (x + r, y + r, z + r)$
- $(x_{min}, y_{min}, z_{min})$
- $(x_{max}, y_{max}, z_{max})$

Bounding Box of a Group

- $(x_{min_a}, y_{min_a}, z_{min_a})$
- $(x_{max_a}, y_{max_a}, z_{max_a})$
- $(x_{min_b}, y_{min_b}, z_{min_b})$
- $(x_{max_b}, y_{max_b}, z_{max_b})$

Acceleration Spatial Data Structures
Spatial Data Structures

- Spatial partitioning techniques classify all space into non-overlapping portions
- Easier to generate automatically
- Can "walk" ray from partition to partition
- Hierarchical bounding volumes surround objects in the scene with (possibly overlapping) volumes
- Often tightest fit

Spatial Partitioning

- Some spatial partitioning schemes:
  - Regular grid (2-D or 3-D)
  - Octree
  - k-D tree
  - BSP-tree

Acceleration Spatial Data Structures

Regular Grid

Create grid

- Find bounding box of scene
- Choose grid spacing
- grid, need not = grid,

Insert primitives into grid

- Primitives that overlap multiple cells?
- Insert into multiple cells (use pointers)
For each cell along a ray
- Does the cell contain an intersection?
  - Yes: return closest intersection
  - No: continue
- Use algorithm to step through cells

Preventing repeated computation
- Perform the computation once, "mark" the object
- Don't re-intersect marked objects

Don't return distant intersections
- If intersection t is not within the cell range, continue (there may be something closer)

Where do we start?
- Intersect ray with scene bounding box
- Ray origin may be inside the scene bounding box

Is there a pattern to cell crossings?
- Yes, the horizontal and vertical crossings have regular spacing

What's the next cell?
- \( t_{\text{next}_x} < t_{\text{next}_h} \)
  - \( i \leftarrow \text{sign} \)
  - \( t_{\text{min}} = t_{\text{next}_x} \)
  - \( t_{\text{next}_x} += dt_x \)
- \( j \leftarrow \text{sign} \)
  - \( t_{\text{min}} = t_{\text{next}_h} \)
  - \( t_{\text{next}_h} += dt_h \)

if \( (\text{dir}_x > 0) \text{ sign} = 1 \) else \( \text{sign} = -1 \)
if \( (\text{dir}_y > 0) \text{ sign} = 1 \) else \( \text{sign} = -1 \)
What's the next cell?

- 3DDDA – Three Dimensional Digital Difference Analyzer
- 3D Bresenham Algorithm

Pseudo-code

create grid
insert primitives into grid
for each ray $r$
find initial cell $c(i,j)$, $t_{min}$, $t_{next_v}$ & $t_{next_h}$
compute $dt_v$, $dt_h$, sign_v and sign_h
while $c$ is NOT NULL
for each primitive $p$ in $c$
intersect $r$ with $p$
if intersection in range found
return
$c = $ find next cell

Regular Grid Discussion

- Advantages?
  - easy to construct
  - easy to traverse

- Disadvantages?
  - may be only sparsely filled
  - geometry may still be clumped in a small number of cells

Acceleration Spatial Data Structures

Adaptive Grids

- Subdivide until each cell contains no more than $n$ elements, or maximum depth $d$ is reached

Adaptive Grids

- Can live at intermediate levels, or be pushed to lowest level of grid

Primitives in an Adaptive Grid

- Nested Grids
- Octree(Quadtree)
Adaptive Grid Discussion
- Advantages?
  - Grid complexity matches geometric density
- Disadvantages?
  - More expensive to traverse (especially octree)

k-D Trees
- k-D tree pros:
  - Moderately simple to generate
  - More adaptive than octrees
- k-D tree cons:
  - Less efficient to trace rays across
  - Moderately complex data structure

BSP Trees
- BSP tree pros:
  - Extremely adaptive
  - Simple & elegant data structure
- BSP tree cons:
  - Very hard to create optimal BSP
  - Splitting planes can explode storage
  - Simple but slow to trace rays across

Acceleration Spatial Data Structures
Bounding Volume Hierarchy
- What makes a “good” bounding volume hierarchy?
  - Grouped objects (or volumes) should be near each other
  - Volume should be minimal
  - Sum of all volumes should be minimal
  - Top of the tree is most critical
  - Constructing the hierarchy should pay for itself!
Bounding Volume Hierarchy

- Find bounding box of objects
- Split objects into two groups
- Recurse

Where to split objects?

- At midpoint OR
- Sort, and put half of the objects on each side OR
- Use modeling hierarchy

Data Structure Pseudo-code

```plaintext
sort_axis = 0;
Make_BVH(object_list, sort_axis, ptr);
struct.bbox = BoundingBox(object_list);
if # of objects < Threshold
    struct.obj_list = object_list
else
    if ((sort_axis % 3)== 0) Sort object centroids in X
    elseif ((sort_axis % 3)== 1)
        Sort object centroids in Y
    else
        Sort object centroids in Z
    end
    Split sorted list into two halves
    Make_BVH(left_obj_list, sort_axis++, lptr)
    Make_BVH(right_obj_list, sort_axis++, rptr)
    struct.lptr = lptr; struct.rptr = rptr;
    ptr = &struct;
    return
```

Intersection with BVH

- Check subvolume with closer intersection first

Intersection with BVH

- Don’t return intersection immediately if the other subvolume may have a closer intersection

Intersection Pseudo-code

```plaintext
Does ray intersect box?
intersect_BVH(box, ray, xsect_pt, t)
  If no more subboxes
    Intersect geometry and return nearest xsect_pt & t
  Intersect ray with both subboxes
    No hits: return xsect_pt = Null;
    Sort t’s
    Call subbox of nearest t subbox1
    intersect_BVH(subbox1, ray, xsect_pt, t)
    If hit_subbox2?
      If xsect_pt == Null || t2near <= t1far
        intersect_BVH(subbox2, ray, xsect_pt, t)
      Set nearest xsect_pt and t
    Return
```

Bounding Volume Hierarchy Discussion

- Advantages
  - easy to construct
  - easy to traverse
  - binary
- Disadvantages
  - may be difficult to choose a good split for a node
  - poor split may result in minimal spatial pruning
- Hint
  - Alternate sorting in X, Y & Z

Transformation Hierarchy

- Group & Transformation hierarchy may not be a good spatial hierarchy

What’s the best method?

- What kind of scene are you rendering?
  - Teapot in a stadium vs. uniform distribution
  - Impact on surface tessellation on distribution
- Parameter values are critical
Shoot Fewer Rays

- Adaptive depth control
  - Naïve ray tracer: spawn 2 rays per intersection until max recursion limit
  - In practice, few surfaces are transparent or reflective
  - Stop shadow ray at first intersection between start and light source
  - Just shoot the rays you need
  - Determine contribution of ray
    - Don’t shoot rays w/ contribution near 0%

Shoot Fewer Rays

- Adaptive sampling
  - Shoot rays coarsely, interpolating their values across pixels
  - Where adjacent rays differ greatly in value, sample more finely
  - Stop when some maximum resolution is reached

Generalized Rays

- Beams, cones, pencils
- Area sampling, rather than point sampling
- Geometric computations are tricky (expensive?)
- Problems with reflection/refractions

Wrap Up

- Discuss next programming assignment
  - Add an acceleration technique
    - Adaptive grid
    - Bounding volume hierarchy
  - Supersample image
- Discuss status/problems/issues with this week’s programming assignment