Computer Graphics

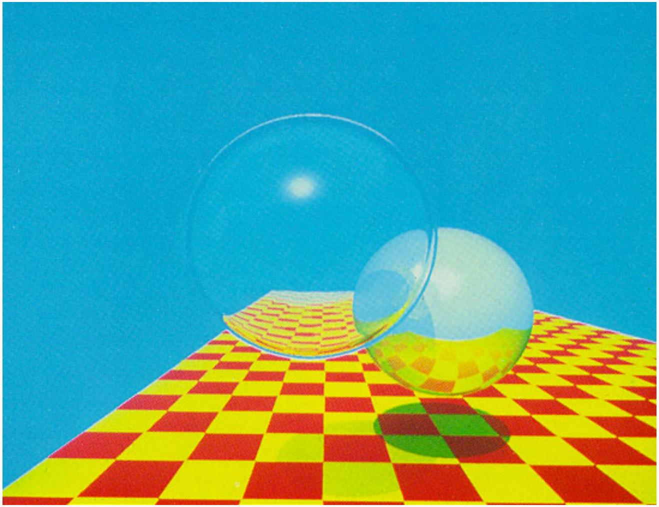
Dr. rer. nat. Martin Möddel

April 6, 2021

Institut für Biomedizinische Bildgebung

Ray Tracing

Ray Tracing



Ray Tracing

Definition

A ray tracer is an IOR algorithm, where a ray is traced into a scene until it hits a scene object. Information on hit point and object are used to color image pixels. A basic ray tracer has three parts

1. Ray generation: compute origin and direction of pixels viewing ray based on camera geometry and orientation
2. Ray intersection: Find closest object intersecting the viewing ray
3. Shading: Compute pixel color based on the result of the ray intersection

Rays

Definition
A ray is a 3D parametric line

$$\mathrm{p}(t) = \mathrm{o} + t(\mathrm{s} - \mathrm{o}). \tag{1}$$

Here o is the origin of the ray, $\mathrm{d} = \mathrm{s} - \mathrm{o}$ is the direction and $t \in \mathbb{R}$ is the fractional distance of $\mathrm{p}(t)$ to o.

- $p(0) = \mathrm{o}$ and $\mathrm{p}(1) = \mathrm{s}$
- If $0 < t_1 < t_2$ then $p(t_1)$ is closer to the origin o than $\mathrm{p}(t_2)$
- If $t < 0$ then $\mathrm{p}(t)$ is behind o

Remark
Ray generation is tied to the camera and is usually performed in the camera coordinate system.

Ray Intersection

Apart from ray generation, intersection methods are an integral part of ray tracing, though they find their use in many other applications in computer graphics. Intersection methods should answer the following questions:

- Is my ray hitting any object?
- Are the objects in front of my camera, on its back or surrounding the camera?
- Which object is closest in front of the camera?
- Where does the object get hit?
- Where does the normal of the surface points towards?

Ray Sphere Intersection

Using the implicit formula we can describe a sphere centered at $\mathrm{c} \in \mathbb{R}^3$ with radius r by $f(\mathrm{x}) = \|\mathrm{x} - \mathrm{c}\|_2 - r = 0$. Let $\mathrm{p}(t) = \mathrm{o} + t\mathrm{d}$ be a ray then

$$\begin{aligned} & 0 = \|\mathrm{o} + t\mathrm{d} - \mathrm{c}\| - r \\ \Leftrightarrow\ & r = \|\mathrm{o} + t\mathrm{d} - \mathrm{c}\| \\ \Leftrightarrow\ & r^2 = \|\mathrm{o} + t\mathrm{d} - \mathrm{c}\|^2 \\ \Leftrightarrow\ & 0 = t^2 + t\frac{2\mathrm{d} \cdot (\mathrm{o} - \mathrm{c})}{\mathrm{d} \cdot \mathrm{d}} + \frac{(\mathrm{o} - \mathrm{c}) \cdot (\mathrm{o} - \mathrm{c}) - r^2}{\mathrm{d} \cdot \mathrm{d}} \\ \Leftrightarrow\ & 0 = t^2 + 2tb + c, \end{aligned}$$

where $b = \frac{\mathrm{d} \cdot (\mathrm{o} - \mathrm{c})}{\mathrm{d} \cdot \mathrm{d}}$ and $c = \frac{(\mathrm{o} - \mathrm{c}) \cdot (\mathrm{o} - \mathrm{c}) - r^2}{\mathrm{d} \cdot \mathrm{d}}$.

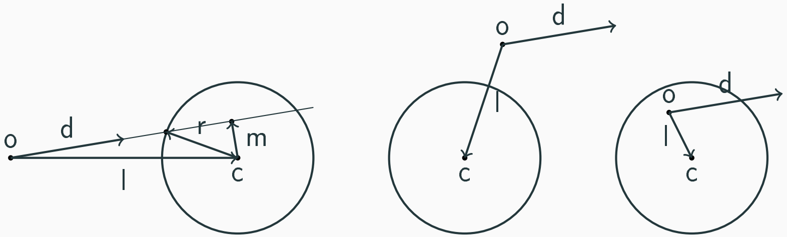
Ray Sphere Intersection

The possibly complex roots are then given by

$$t_{\pm} = -b \pm \sqrt{b^2 - c}. \tag{2}$$

- If $b^2 - c < 0$ then the roots are complex. Meaning the ray does not intersect with the sphere.
- If $b^2 = c$ then the two roots coincide and the ray intersects with the sphere at one point.
- If $b^2 - c > 0$ then there are two intersection points.
    - If $b > \sqrt{b^2 - c}$ then the sphere is behind the camera
    - If $b < -\sqrt{b^2 - c}$ then the sphere is in front of the camera
    - If $\sqrt{b^2 - c} > b > -\sqrt{b^2 - c}$ then camera is inside the sphere

Ray Sphere Intersection - Optimizations



- Compute the vector $\mathrm{l} = \mathrm{c} - \mathrm{o}$ and its length $l^2 = \mathrm{l} \cdot \mathrm{l}$
- If $l^2 < r^2$ the ray origin is inside the sphere (right)
- Compute $s = \mathrm{l} \cdot \mathrm{d}$
- If $s < 0$ and the origin is outside the sphere the sphere is behind the camera
- Compute distance between ray and sphere center $m^2 = l^2 - s^2$
- If $m^2 > r^2$ the sphere is going to be missed
- If not intersections occur at $t_\pm = s \pm q$, where $q^2 = r^2 - m^2$

Ray OBB Intersection

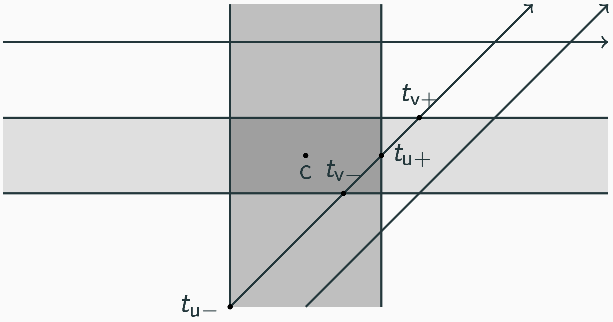
Definition
An oriented bounding box is a box whose faces have normals u, v, and w which make up an orthonormal basis. It is described by its center c and its positive half-lengths h_{u}, h_{v}, and h_{w}, i.e. the shortest distance from center to face along the respective normal directions.

Alternative Description
Consider a slab which is the infinitely large volume in between two parallel planes. For each direction consider the slab defined by the opposing faces of an OBB then the intersection of all three slabs equals the oriented bounding box.

Ray OBB Intersection

Slab Ray intersection
Each ray intersects the plane twice at $(t_{\mathsf{u}-}, t_{\mathsf{v}-}, t_{\mathsf{w}-})$ and $(t_{\mathsf{u}+}, t_{\mathsf{v}+}, t_{\mathsf{w}+})$ respectively. A ray hits the OBB if it enters all slabs before it leaves any

$$\max(t_{\mathsf{u}-}, t_{\mathsf{v}-}, t_{\mathsf{w}-}) \leq \min(t_{\mathsf{u}+}, t_{\mathsf{v}+}, t_{\mathsf{w}+}). \tag{3}$$



Ray OBB Intersection

```
procedure RaysOBBIntersect(Ray, OBB)
    $t_{\text{min}} = -\infty$
    $t_{\text{max}} = \infty$
    for direction $\text{a} \in \{\text{u}, \text{v}, \text{w}\}$ do
        $e = \text{a} \cdot (\text{c} - \text{o})$
        $f = \text{a} \cdot \text{d}$
        if $|f| >$ floating point precision then
            $t_1 = (e + h_{\text{a}})/f$
            $t_2 = (e - h_{\text{a}})/f$
            if $t_1 > t_2$ then swap$(t_1, t_2)$
            end if
            if $t_1 > t_{\text{min}}$ then $t_{\text{min}} = t_1$
            end if
            if $t_2 < t_{\text{max}}$ then $t_{\text{max}} = t_2$
            end if
```

Ray OBB Intersection

```
16:               if $t_{\text{min}} > t_{\text{max}}$ then return OBB not hit
17:               end if
18:               if $t_{\text{max}} < 0$ then return OBB behind camera
19:               end if
20:           else if $-e - h_a > 0$ or $-e + h_a < 0$ then return Ray parallel to OBB but not hitting
21:           end if
22:       end for
23:       if $t_{\text{min}} > 0$ then return intersect at $t_{\text{min}}$
24:       else return intersect at $t_{\text{max}}$
25:       end if
26:   end procedure
```

Ray Polygon Intersection

Definition Plane
A plane can be defined by a normal vector n and a point on the plane p by $\{\mathrm{x}|\mathrm{n} \cdot (\mathrm{x} - \mathrm{p}) = 0\}$. Alternatively, we can use the distance of the plane to the origin $\delta = \mathrm{n} \cdot \mathrm{p}$ to define the plane

$$\{\mathrm{x}|\mathrm{n} \cdot \mathrm{x} - \delta = 0\}\,. \tag{4}$$

Definition Polygon
A simple planar polygon of n vertices (n-gon) is defined by an ordered vertex list $\{\mathrm{p}_i\}_{i=1}^{n}$ and a plane with normal vector n and its distance to the origin δ. The vertices p_i and p_{i+1} form an edge for all $i = 1, \ldots, n-1$ and the polygon is closed by the edge from p_n to p_1, where non of the edges intersect each other.

Ray Polygon Intersection

- To intersect a ray $\mathrm{p}(t) = \mathrm{o} + t\mathrm{d}$ and a polygon first calculate the intersection between the polygon plane and the ray

$$\mathrm{n} \cdot (\mathrm{o} + t\mathrm{d}) + \delta = 0. \tag{5}$$

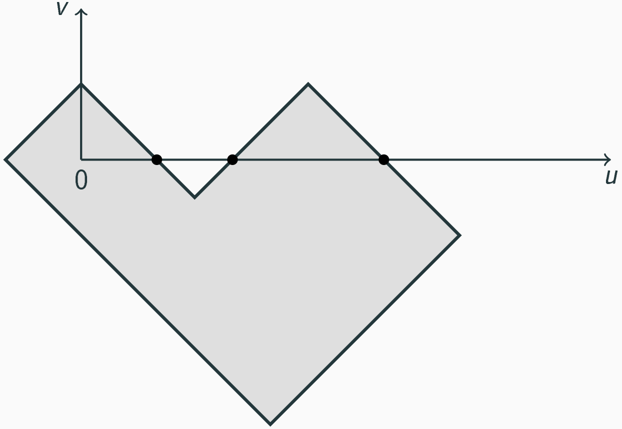
- If $|\mathrm{n} \cdot \mathrm{d}| \ll 1$ the ray is considered parallel to the plane (avoid overflow when dividing), otherwise the intersection occurs for

$$t_{\mathrm{intersect}} = \frac{-\delta - \mathrm{n} \cdot \mathrm{o}}{\mathrm{n} \cdot \mathrm{d}}. \tag{6}$$

- Project all vertices and the intersection point $\mathrm{p}(t_{\mathrm{intersect}})$ to one of the xy-, yz-, or xz-plane, where the area of the projected polygon is maximized and relabel the new coordinates by u and v. I.e. skip the coordinate component, where n is largest (skip y if $\mathrm{n} = (1, 2, 0.5)$).
- Shift the projected intersection point into the origin.

Ray Polygon Intersection

- Count crossings with positive u-axis.
  - odd number: Origin inside polygon
  - even number: Origin outside polygon
  - infinite crossings: Rotate vertices slightly and repeat test



Ray Polygon Intersection

```
procedure CountCrossings
    $n_{\text{crossings}} = 0$
    for edge vertex pairs $(\mathrm{a}, \mathrm{b})$ do
        if $(a_v > 0$ and $b_v < 0)$ or $(a_v < 0$ and $b_v > 0)$ then
            if $(a_u > 0$ and $b_u > 0)$ then
                $n_{\text{crossings}} = n_{\text{crossings}} + 1$
            else if intersection edge $u$-axis positive then
                $n_{\text{crossings}} = n_{\text{crossings}} + 1$
            else no crossing possible
            end if
        else no crossing possible
        end if
    end for return $n_{\text{crossings}}$
end procedure
```

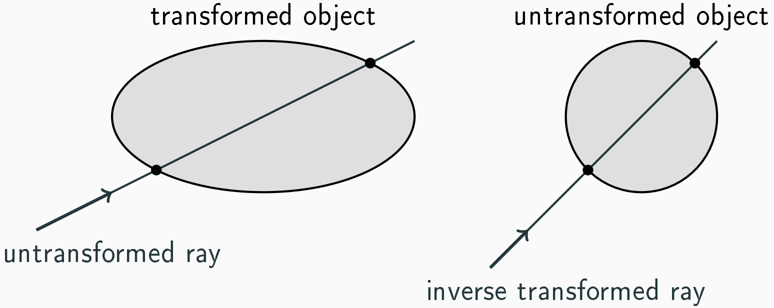
Instancing

Geometry instancing is the practice of rendering multiple copies of the same possibly distorted object in a scene. This method can also be applied to ray tracing where one can chose to

1. Transform the object and intersect with the untransformed ray
2. Apply the inverse transformation to the ray and intersect with the untransformed object

- Homogeneous coordinates can be used to transform the ray origin (point) and ray direction.
- Untransformed objects may have simpler intersection routines (sphere vs. ellipsoid)
- Many transformed objects might share the same untransformed object reducing storage

Instancing


transformed object
untransformed object
untransformed ray
inverse transformed ray

Optimization Techniques in Intersection

- Order comparisons such that easy to compute reject or accept intersections come first to possibly escape expensive computations
- Exploit the results from previous tests whenever possible
- Try out different orders if you have multiple tests. Minor changes might result in a significant performance boost
- Postpone expensive calculations such as trigonometric functions and square roots until they are required
- Dimensionality reduction simplify an intersection problem quite significantly
- In some cases pre-calculations can be worth the effort, though they require memory
- Take all special cases into account even if they are unlikely to occur
- Intersect with bounding boxes first, for complicated calculation

Bounding Volumes

- A Bounding Volume is a volume enclosing an object which is used for performant intersection rejection
- Common bounding volumes are spheres, axis aligned bounding boxes and oriented bounding boxes
- Finding optimal boxes is usually a non-trivial problem
- Often suboptimal boxes are chosen trading speed vs. efficiency
- In order to minimize intersection costs it is important to find tight fittings for an object.

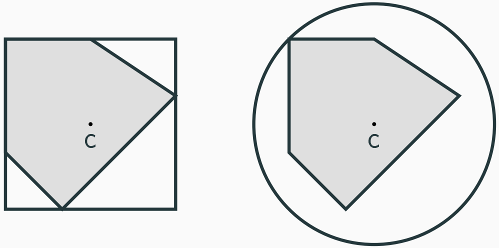
Bounding Volumes

Axis Aligned Bounding Box (AABB)

- Find extreme coordinate values for each direction
- create tight axis aligned box from these values

Sphere

- Sphere center set to coincide with center from AABB
- radius given by largest distance to center position



Spatial Data Structures

- In case not only a few but a large number n of objects is considered the intersection costs increase by $\mathcal{O}(n)$
- Spatial data structures organizes objects in some space to accelerate intersection queries typically to $\mathcal{O}(\log n)$
- Most spatial data structures organize objects hierarchically
- Construction of these structures is expensive and usually done in a preprocessing step
- Common types are bounding volume hierarchies (BVHs) and binary space partitioning (BSP) trees
    - In BHVs each object belongs to exactly one of a number of nodes within each hierarchy level, whereas spatial points may belong to a number of nodes
    - In BSPs each spatial point belongs to exactly one of a number of nodes within each hierarchy level, whereas an object may belong to a number of nodes

Tree Structures in Computer Graphics

The scene is organized in a hierarchical tree structure consisting of a root, internal nodes, and leaves.

- The topmost node is the root containing all scene objects
- A leaf node holds a single scene object
- An internal node holds a number of children (nodes and/or leafs)
- Each node/leaf has a bounding volume enclosing all objects in its entire sub-tree

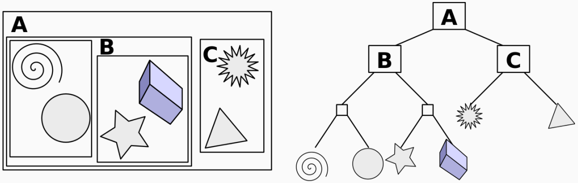


Figure 1: Example of bounding volume hierarchy (BVH) in two dimensions, where bounding volumes are AABB by Schreiberx licencend under CC BY-SA 3.0.

Spatial Data Structures - Ray Scene Intersection

Find first object in hierarchy

- Start intersection with the root bounding volume
- If the ray misses it misses all objects
- otherwise continue recursively by testing the bounding volumes of the children
- Whenever a bounding volume is missed the entire sub-tree can be dismissed
- If the bounding volume of a leaf node is hit intersection with the object contained within is tested

Spatial Data Structures - Ray Scene Intersection

Find closest object in hierarchy

- Store the identity and distance of the first object hit
- When intersecting bounding volumes discard sub-trees if the distance to the volume is larger than the stored distance
- Update identity and distance if a closer object is hit
- Continue until the whole tree is traversed

Bounding Volume Hierarchy

Example Hierarchy Creation

```
 1: Chose an integer constant $c$
 2: Decide on method to chose direction e, e.g. cycle through axis-
    directions as we go deeper into the hierarchy
 3: Create AABB for all objects of scene
 4: procedure bvh(objects)
 5:     if Number of objects $\leq c$ then
 6:         Create Bounding box enclosing all objects
 7:         Create a leaf for each object
 8:     else
 9:         Order objects along e (AABB center)
10:         Partition objects into $c$ sets
11:         Call bhv for each set
12:     end if
13: end procedure
```

Binary Space partitioning trees

Example Space Partitioning

```
 1: Create AABB for all objects of scene
 2: procedure bps(AABB, objects)
 3:     if Some continuation criterion is met then
 4:         Divide the AABB in two along an axis, e.g. the axis where
    the AABB is largest
 5:         for each sub-box AABB$_i$ do
 6:             $O_i$ = set of objects (partially) inside AABB$_i$
 7:             bps(AABB$_i$, $O_i$)
 8:         end for
 9:     else
10:         create one leaf containing the objects
11:     end if
12: end procedure
```

Spatial Data Structures

Remarks

- There is no recipe on which data structure to use
  - Depends on task to be accelerated
  - Depends on scene to be rendered
- There a more spatial data structures availible, which might be usefull depending on your application
  - Regular grids
  - Octrees
  - Constructive Solid Geometry trees