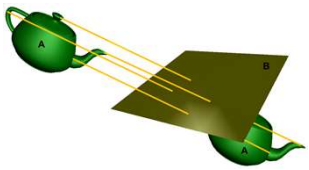
1

Collision Detection

Oregon State University
Mike Bailey
mjb@cs.oregonstate.edu



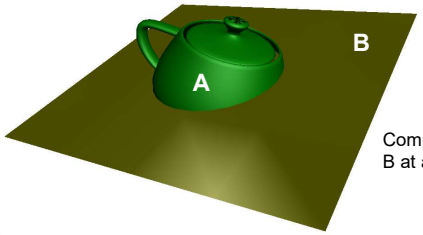
Oregon State University
Computer Graphics

1

2

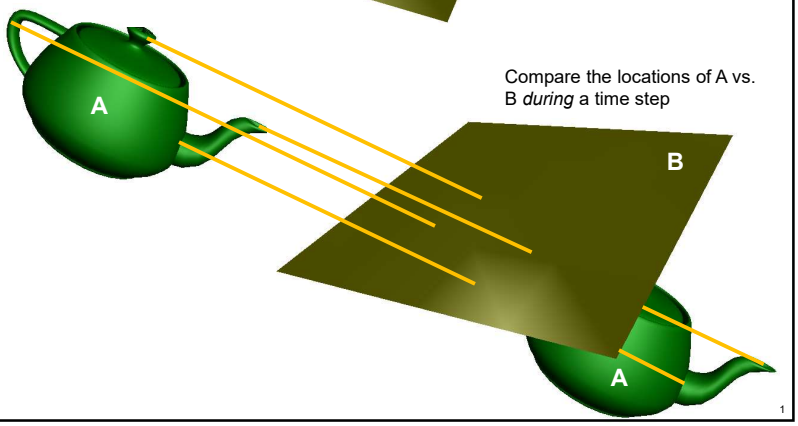
There are Two Types of Collision Detection

1. Discrete



Compare the locations of A vs. B at a *single snapshot* in time

2. Continuous



Compare the locations of A vs. B *during* a time step

Oregon State University
Computer Graphics

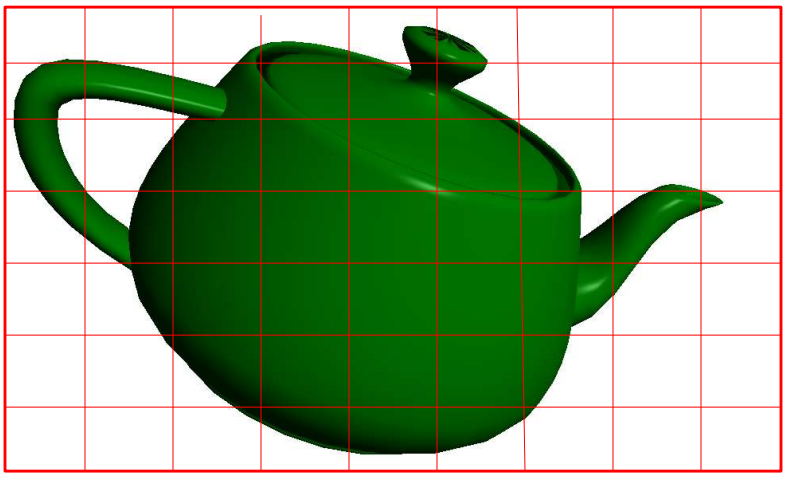
2

Detecting Collisions Between Two Objects

1. Do as many *fast rejections* as you can
2. Do *hierarchical* fast rejections
3. **Discrete:** compare all edges of Object A against all faces of Object B
3. **Continuous:** create "pseudo-edges" by connecting respective points in Object A across the time step, then compare all these pseudo-edges of Object A against all faces of Object B

Oregon State University
Computer Graphics

A Way to Simplify the Intersection Test – Break the Scene into a Grid



Oregon State University
Computer Graphics

Discrete: You only have to do intersection tests against objects that live in the same grid square.

A Way to Simplify the Intersection Test -- A Bounding Sphere

5

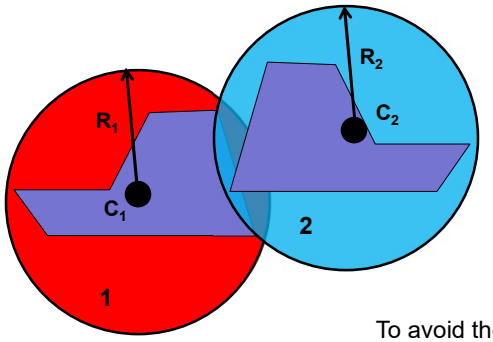
Oregon State University
Computer Graphics

mjb – August 6, 2021

5

A Way to Simplify the Intersection Test - Bounding Spheres

6



These spheres overlap if:

Distance(C_1,C_2) < R_1+R_2

To avoid the time-consuming square root:

Distance^2(C_1,C_2) < (R_1+R_2)^2

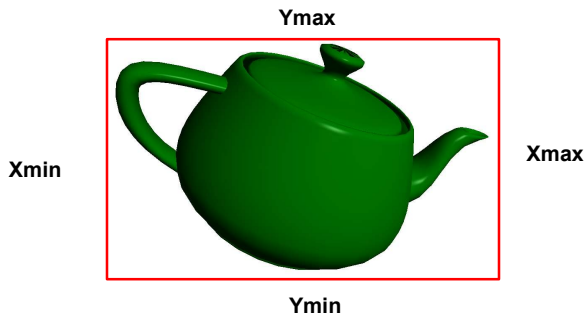
Oregon State University
Computer Graphics

mjb – August 6, 2021

6

A Way to Simplify the Intersection Test -- A Bounding Box

7



Oregon State University
Computer Graphics

mjb – August 6, 2021

7

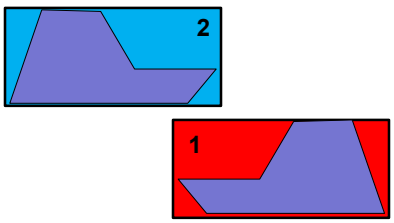
Try to Simplify the Intersection Test- Bounding Boxes

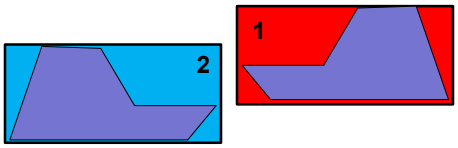
8

Quickly compare two objects by fitting each with a bounding box and then comparing the two bounding boxes.

We know that these bounding boxes ***do not*** overlap because:

$$\underbrace{Xmax_1 < Xmin_2 \;||\; Ymax_1 < Ymin_2}_{} \;||\; \underbrace{Xmax_2 < Xmin_1 \;||\; Ymax_2 < Ymin_1}_{}$$





Oregon State University
Computer Graphics

mjb – August 6, 2021

8

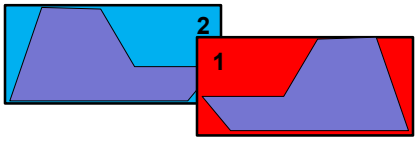
Try to Simplify the Intersection Test- Bounding Boxes

9

Quickly compare two objects by fitting each with a bounding box and then comparing the two bounding boxes.

We know that these bounding boxes ***do*** overlap because:

$Xmax_1 > Xmin_2$ && $Ymax_1 > Ymin_2$ && $Xmax_2 > Xmin_1$ && $Ymax_2 > Ymin_1$



Oregon State University
Computer Graphics

mjb – August 6, 2021

9

Try to Simplify the Intersection Test- -- Two Types of Bounding Boxes

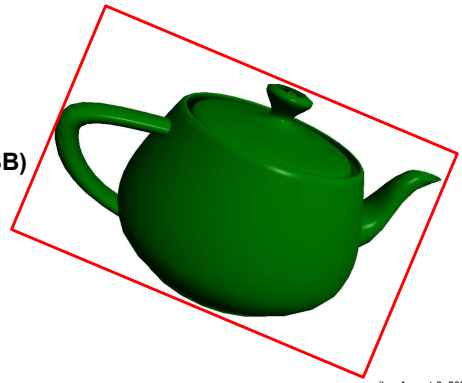
10

Axis-Aligned Bounding Box (AABB)

Check for overlap by looking for overlap in just X, then just Y, then just Z

Arbitrary-Oriented Bounding Box (AOBB)

This is a tighter fit around the object, but the overlap comparison is much more involved

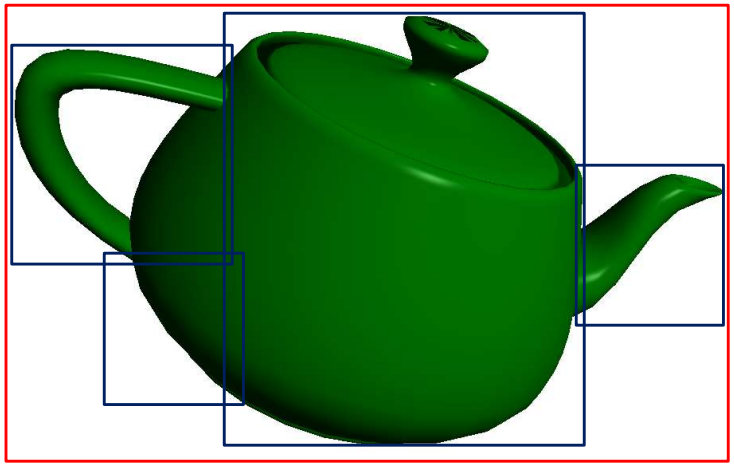


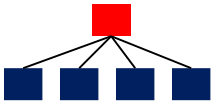
Oregon State University
Computer Graphics

mjb – August 6, 2021

10

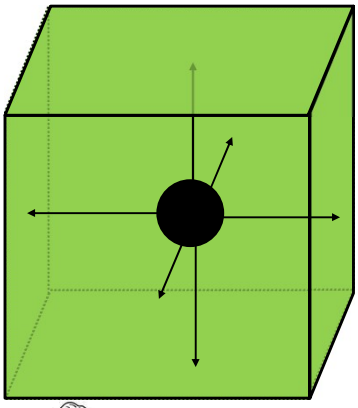
A Hierarchy of Bounding Boxes





Oregon State University
Computer Graphics

Discrete – It is Straightforward to Tell if a Point is Inside a *Convex* Polyhedron



Assume that all surface normals point **outwards** (usual convention).

Use the Distance-from-a-Point-to-a-Plane formula for each face of the polyhedron.

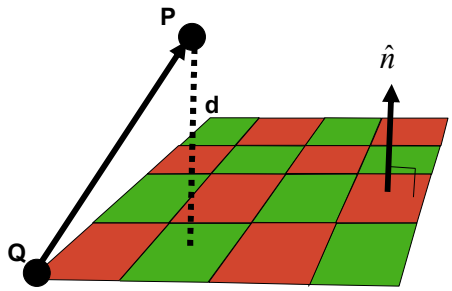
If all distances are ***negative***, the point is inside the convex polyhedron

What if the polyhedron is not convex?

Put a ***Convex Hull*** around the polyhedron and test against that. If the point is not inside the convex hull, then it is not inside the polyhedron either. If it is inside the convex hull, then a more detailed analysis is needed.

Oregon State University
Computer Graphics

Distance from a Point to a Plane



The equation of the plane is:

$$\left((x,y,z)-\left(Q_x,Q_y,Q_z\right)\right)\bullet(n_x,n_y,n_z)=0$$

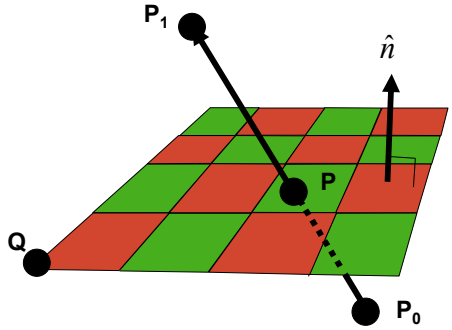
which expands out to become the more familiar Ax + By + Cz + D = 0

The distance from the point P to the plane is based on the plane equation:

$$d=\left(P-Q\right)\bullet\hat{n}$$

The dot product is answering the question "How much of (P-Q) is in the normal direction?". Note that this gives a *signed distance*. If d > 0., then P is on the same side of the plane as the normal.

Discrete and Continuous – Comparing an Edge on Object A against a Face on Object B



The equation of the line segment is:

$$P=(1-t)P_0+tP_1$$

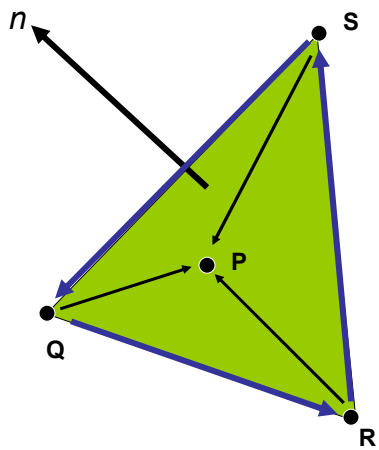
If point P wants to be a point in the plane, then:

$$\left((P_x,P_y,P_z)-\left(Q_x,Q_y,Q_z\right)\right)\bullet(n_x,n_y,n_z)=0$$

If we substitute the parametric expression for P into the plane equation, then the only thing we don't know in that equation is t. Knowing t* will let us compute the (x,y,z) of the actual intersection using the line equation. If t* has a zero in the denominator, then that tells us that t*=∞, and the line must be parallel to the plane.

This gives us the point of intersection with the infinite plane. We would now use the method covered a few slides ago to see if P lies inside the triangle in question.

Is a Point inside a Triangle?



Let:

$$n = (R - Q) \times (S - Q)$$

$$n_q = (R - Q) \times (P - Q)$$

$$n_r = (S - R) \times (P - R)$$

$$n_s = (Q - S) \times (P - S)$$

If $(n \bullet n_q), (n \bullet n_r), and (n \bullet n_s)$

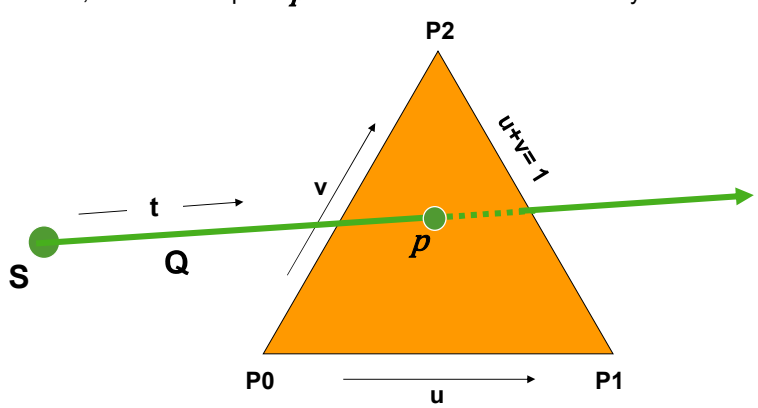
are all positive, then P is inside the triangle QRS

Oregon State University
Computer Graphics

From the Ray-Triangle Notes

We want to find out where the ray intersects the triangle.

That is, where is the point p that is common to both the ray and the triangle?



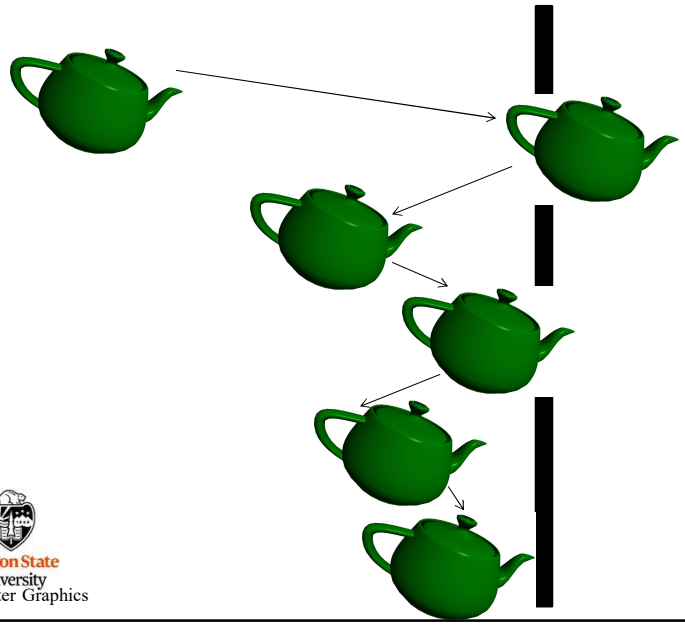
Such that:

$t \geq 0.$
$0. \leq u \leq 1.$
$0. \leq v \leq 1.$-u

Oregon State University
Computer Graphics

In the Discrete case, Find When the Exact Collision Occurs

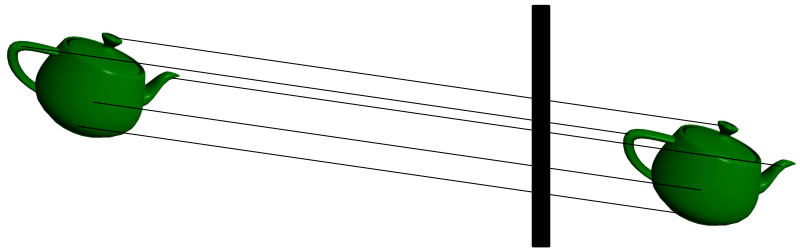
Do a binary search across the time step until you find the time of collision



Oregon State University
Computer Graphics

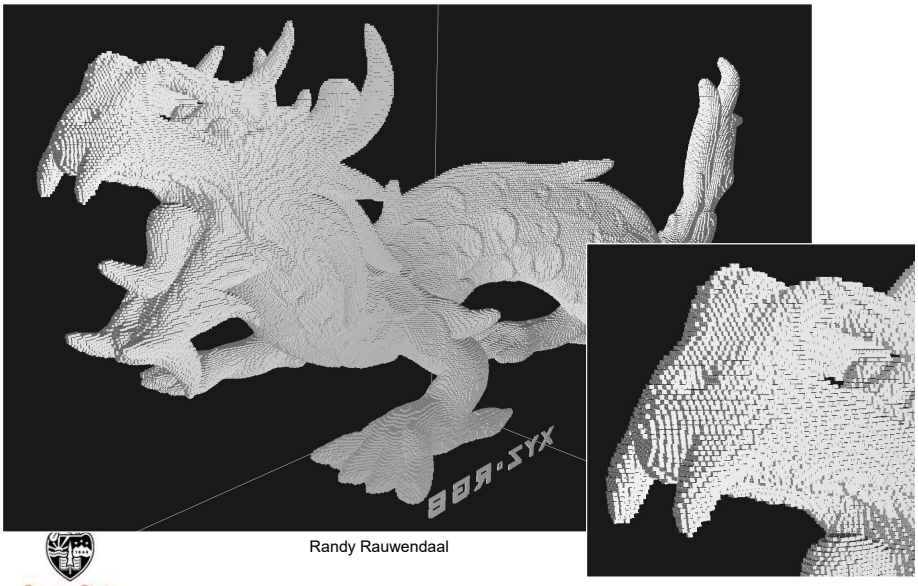
In the Continuous case, Find When the Exact Collision Occurs

Connect all points across time and look for the minimum t in an intersection with the boundary



Oregon State University
Computer Graphics

Voxelization – another way to do Collision Detection?



Randy Rauwendaal

Oregon State
University
Computer Graphics