Lecture 15:
Ray Polygon Intersection

October 18, 2016
Outline

• Universal Basics
  – Polygon edges and tangents
    • Lines, rays, segments, tangents and normals
    • Generating the equation of the plane (infinite)
  – Ray intersection with a plane (infinite)
• Intersection with a convex Polygon
  – Inside-outside tests in 2D
  – Inside-outside tests in 3D
• Odd-even test for a 2D non-convex polygon
Review: A Line

• One dimensional object in n-dimensions
• Ambiguous terminology
  – May be bounded or infinite
  – May be directed or undirected
• Algebraic expression / representation
  – Implicit function
    • Distance from origin and a direction
  – Parametric
    • Two points define a line
Review: A Plane

• Two dimensional object in n-dimensions
• Less ambiguous than Line
  – Generally assumed to be unbounded
• Algebraic expression / representation
  – Implicit function
    • Distance from origin and a direction
  – Parametric
    • Three points define a plane
Review: A Segment

• Like a line, but!
• Term is short for Line Segment.
• Clearly indicates a start and end point.
• Algebraic expression / representation
  – Parametric
  – Generally implicit form is NOT used.
• Consider this question:
  – Do segments $S_1$ and $S_2$ intersect?
Review: A Vector

- Direction and magnitude in n-dimensions
- Does NOT have a location in space!
- Where can we draw vectors in illustrations?
  - Anywhere we want, provided it serves our purpose.
- Algebraic expression / representation
  - First choice is an n-tuple of scalars
  - Second choice is a unit length vector and a separate magnitude
Review: A Ray

• One dimensional object in n-dimensions
• Between and infinite line and a segment
  – Bounded at one end only
• Algebraic expression / representation
  – Parameterized line bounded at one end
  – In code, often a tuple (Point, Unit length vector)
  – Is the following a Line, Segment or Ray?

\[ P(t) = t(P_2 - P_1) + P_1 \]
Review: A Polyline

- One dimensional object in n-dimensions
- A sequence of one or more Segments
- Algebraic expression / representation
  - Top-level is a tuple: S1, S2, S3
  - Down one level
    - Segment parameterized between start and end points
  - Shorthand
    - A k Segment Polyline expressed as k+1 Points
- Question
  - What is the area of a Polyline?
Review: A Polygon

• Two dimensional object in n-dimensions
• Algebraic expression / representation
  – Bounded by k Segments
  – Defined by k+1 Points (vertices)
• But testing for inclusion is more involved
  – We cross over into the realm where short algorithms are more helpful than any single algebraic expression.
Convexity

• To start, you know it when you see it.

How would you formally confirm (or refute) convexity?
Edge Normal & Edge Tangent

Is this vertex inside/outside the highlighted edge Segment?
Ray/Polygon Intersection

• All polygons must be planar!
  – How would you test for this condition?

• Intersect rays with…
  – General Polygons
    • Slow, complex
  – Convex Polygons
    • Fast, pretty simple
  – Triangles
    • Faster
    • We’ve covered this now.
General Ray/Surface Intersection

• Implicit surfaces are defined by

\[ f(p) = 0 \]

• Given a ray

\[ L + tU \]

• The intersection is solved by

\[ f(L + tU) = 0 \]
Ray/Polygon Intersection

• For convex polygons there are two parts.
  – Find point of intersection P on infinite plane.
  – Test if point P inside polygon.
• General equation for a plane in 3D:
  \[ ax + by + cz + d = 0 \]
• Therefore an (x,y,z) point P is on a plane iff:
  \[ N \cdot P = -d \]
• Where
  \[ N = [a, b, c] \]
Ray/Polygon Intersection (II)

• What is N?
  – It’s the surface normal of the plane!
  – All points on a plane project to the same point on the normal vector (so $N \cdot P = -d$)

• How do we find N?
  – Given vertices \{A, B, C…\}
  – $N = (B-A) \times (C-B)$, then normalize
  – Unless A,B & C are collinear
    • Then $N = (C-B) \times (D-C)$
    • Unless A, B, C and D are collinear…
Ray/Polygon Intersection (III)

• The surface normal \( \mathbf{N} \) is \((a,b,c)\)
  in \( ax + by + cz + d = 0 \)

• How do you compute \( d \)?
  – Plug in \((x,y,z)\) coordinates of any vertex

• Notes
  – You will need the unit surface normal \( \mathbf{N} \) of every polygon for ray tracing
  – Therefore, compute \( \mathbf{N} \) for every polygon when you read it in
  – You only need the \( d \) term if you intersect convex polygons (not just triangles)
Planar Intersection

- Given the equation of the plane, calculate where the ray intersects the plane

\[ N \cdot (L + tU) = -d \]

\[ tN \cdot U = -(N \cdot L + d) \]

\[ t = \frac{-(N \cdot L + d)}{N \cdot U} \]

*Be careful. What if this is zero?*
Ray / Polygon Intersection

• Given $t$, we know the point where the ray intersects the plane of the polygon
  \[ P = L + tU \]

• But is it inside the polygon?
• For the convex case see the Sage Notebook.
  – Mostly omitted from this slide deck.
• For non-convex learn the odd-even rule
  – Covered in the remainder of this slide deck.
You WILL want to play with this Sage Notebook. There is a great deal going on and you should understand the steps.
Making the 3D problem 2D

• Once a 3D polygon and 3D point on the plane are found, throw away one dimension.
• Note that inside/outside is the same in both cases, provided we don’t collapse to a line.
• In these examples, which dimension should be dropped?

\[
\begin{bmatrix}
\left(\frac{1}{14} \sqrt{14}\right), & \left(\frac{4}{29} \sqrt{29}\right), & \left(\frac{6}{41} \sqrt{41}\right), & \left(\frac{3}{22} \sqrt{22}\right)
\end{bmatrix}
\]
Odd/Even Parity Rule

• Tests whether a 2D point P is inside or outside of a polygon

• Step 1: draw a ray from P in any direction in the plane.

• Step 2: count how many boundaries are crossed
  – Odd # of intersections ⇒ inside
  – Even # of intersections ⇒ outside
Odd/Even Illustrated

Direction doesn’t matter!
P1

P2

Q

D

Full Math – Horizontal Ray

```python
var('t','s')
P1 = matrix(SR, 2,1, var('px1','py1'))
P2 = matrix(SR, 2,1, var('px2','py2'))
Q = matrix(SR, 2,1, var('qx','qy'))
D = matrix(SR, 2,1, (1,0))
P = (P2 - P1) * t + P1
R = Q + s * D
eq1 = P[0,0] == R[0,0]
eq2 = P[1,0] == R[1,0]
res = solve([eq1,eq2],s,t)
pretty_print(eq1)
pretty_print(eq2)
pretty_print(res[0][1], res[0][0])
```

\[-(px_1 - px_2)t + px_1 = qx + s\]

\[-(py_1 - py_2)t + py_1 = qy\]

\[t = \frac{py_1 - qy}{py_1 - py_2}\]

\[s = \frac{px_2(py_1 - qy) - px_1(py_2 - qy) - (py_1 - py_2)qx}{py_1 - py_2}\]
Interesting Consequence

• Consider the following Polygon filled by PowerPoint: