CS 543: Computer Graphics
Lecture 13: Raytracing (Part 5)

Emmanuel Agu

Today..

- Cube and mesh hit( ) functions
- Antialiasing

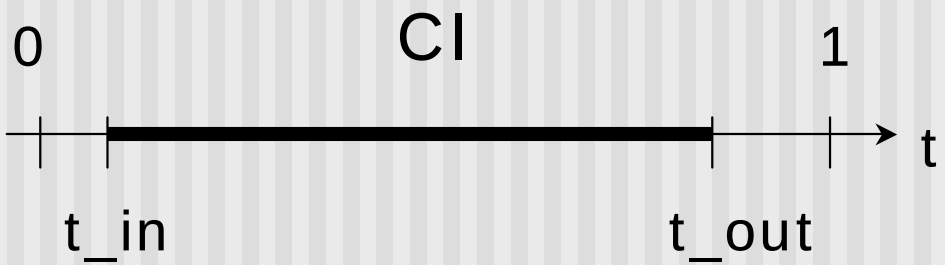
Intersection with Cube (or convex Polyhedra)

- Cubes, Meshes and convex polyhedra used a lot in graphics
- Start with generic cube (center at origin, corners at ±1, ±1, ±1)
- All 8 combinations of +1 and –1 are used (i.e. 2^3)
- Intersection algorithm is essentially Cyrus-Beck clipping
- Six faces lie on planes shown below

Plane	Name	Equation	Outward Normal	Spot
0	top	y=1	(0,1,0)	(0,1,0)
1	bottom	y=-1	(0,-1,0)	(0,-1,0)
2	right	x=1	(1,0,0)	(1,0,0)
3	left	x=-1	(-1,0,0)	(-1,0,0)
4	front	z=1	(0,0,1)	(0,0,1)
5	back	z=-1	(0,0,-1)	(0,0,-1)

Recall: Candidate Interval (CI)

- Define Candidate Interval (CI) as time interval during which edge might still be inside CVV. i.e. CI = t_in to t_out

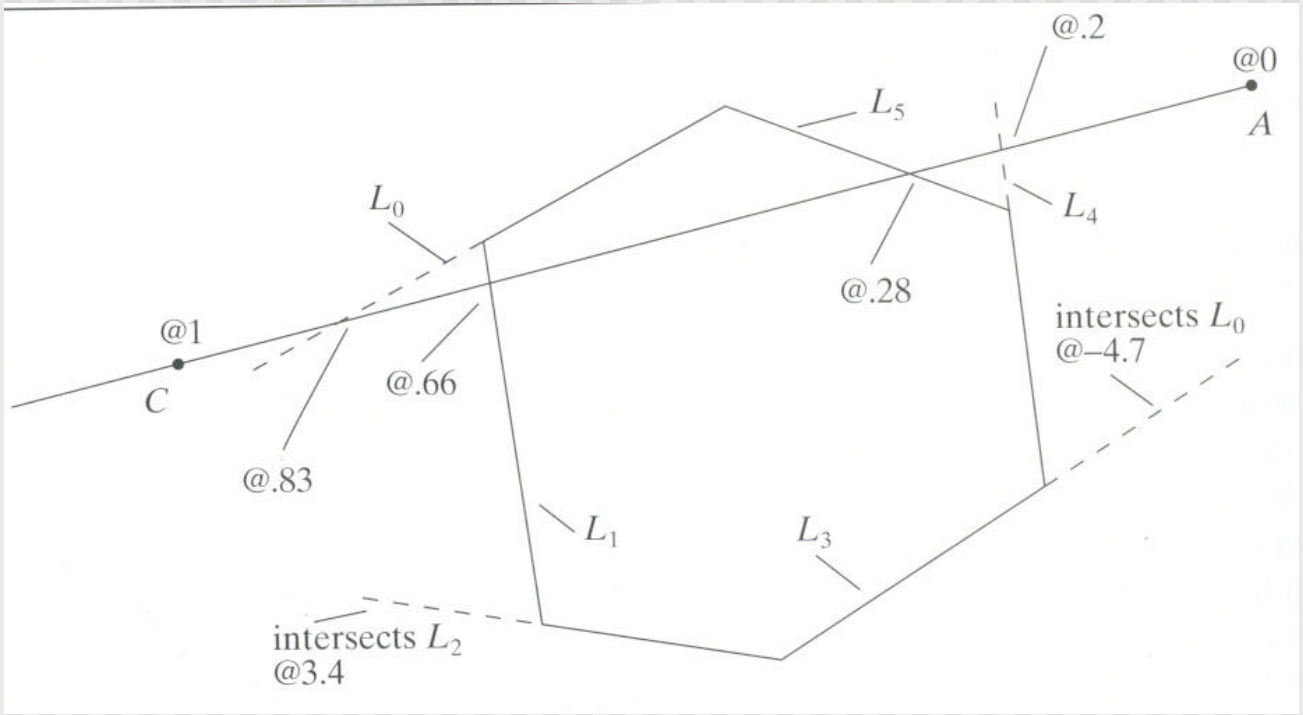


- Conversely: values of t outside CI = edge is outside CVV
- Previously used CI initialized to [0,1], now can exceed this range
- Initialize CI to (-infinity, infinity)

Candidate Interval (CI)

Example to illustrate search for t_in, t_out

Note: CVV is different shape. This is just example



Line test	t_{in}	t_{out}
0	0	0.83
1	0	0.66
2	0	0.66
3	0	0.66
4	0.2	0.66
5	0.28	0.66

Summary of CI

- Track CI
- As we test each plane, chop away at interval
  - Try to reduce t_out
  - Try to increase t_in
- If ever t_out < t_in, STOP!!
- Actual testing is done by

```
if(the ray is entering at t_hit)
    t_in = max(t_in, t_hit)
else if(the ray is exiting at t_hit)
    t_out = min(t_out, t_hit)
```

Testing against Planes

- To solve for intersection times with each plane, put ray equation into implicit equation for plane

$$F(P) = \mathbf{m} \bullet (P - B)$$

$$\mathbf{m} \bullet (S + \mathbf{c}t - B) = 0$$

- And t_hit is given as

$$t = \frac{numer}{denom}$$

- Where numer = $\mathbf{m}.(B - S)$
- And denom = $\mathbf{m}.\mathbf{c}$

Testing against Planes

- Where numer = **m.***(B - S)*, denom = **m.c**
- If denom = 0, ray is parallel to plane, numer determines if it is wholly inside or outside
  - numer > 0, wholly inside
  - numer < 0, wholly outside
- If denom > 0, means ray is passing into **outside** half of plane, since **m.c** are less than 90 degrees apart
- If denom < 0, means ray is passing into **inside** half of plane, since **m.c** are less than 90 degrees apart

Testing against Planes

- Can easily show that numer and denom can be found using the following short forms

Plane	numer	denom
0	1 - Sy	cy
1	1 + Sy	-cy
2	1 - Sx	cx
3	1 - Sx	-cx
4	1 - Sz	cz
5	1 + Sz	-cz

Intersection with Convex Polyhedra

- Implementation for cube function is nicely laid out in figure 14.23 of text. Please read it..
- We've seen enough hit functions to last you a life time
- Read on your own..
- For convex polyhedra, instead of 6 faces for cube, store i faces
- Find normal to face i, m_i and hit point B_i. Use loop then as

```
for(int i=0;i < N;i++)
{
    numer = dot3D(m_i, B_i - S)
    denom = dot3D(m_i, c)
    ... continue same as cube
}
```

Intersection with Mesh

- We can then extend method to develop hit function for a mesh
- Retrieve normal of mesh and vertex 0
- Read mesh intersection part from book (1/2 page)

```
for(int f=0;f < numFaces;f++)
{
    Vector3 diff;
    Vector3 normal(norm[face[f].vert[0].normIndex);
    Point3 point(pt[face[f].vert.vertIndex);
    form diff = point – genRay.start
    numer = dot3D(normal, diff)
    denom = dot3D(normal, genRay.dir)
    … continue same as cube
}
```

References

- Hill, chapter 14