Johns Hopkins
Engineering for Professionals

605.767 Applied Computer Graphics

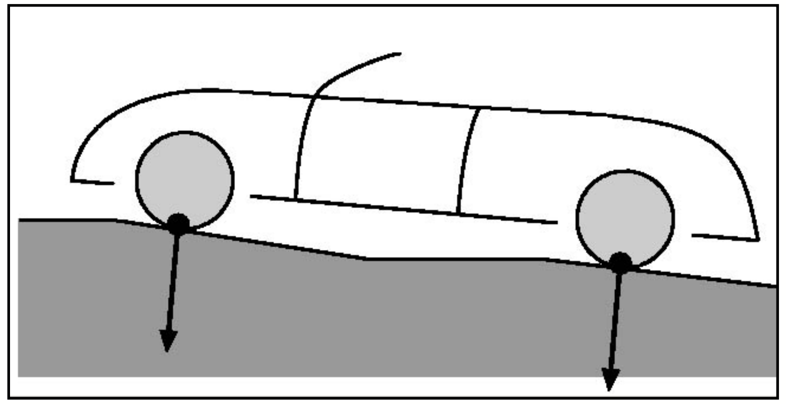
Brian Russin

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

Module 10B
Ray Intersections
Planar Intersections

Collision Detection with Rays

- Can approximate collision detection with rays in certain conditions
- Example: car driving on a road
  - Put a ray at each wheel and intersect it with road geometry
  - 4 wheels are the only points that the car touches the ground
  - Set origin of each ray at the point where wheel touches ground
  - Test intersection of rays with ground:
    - d = 0, wheel is touching the ground
    - d > 0, wheel is above the ground
    - d < 0, wheel penetrates the environment
    - Use these distances for collision response
      - Move wheel to stay in touch with ground
- Requires distance further back along the ray
  - Can have a negative distance
  - Usually move the ray origin in a negative direction rather than testing for intersections in both directions



Triangle/Triangle Intersection

- Triangles are the predominant drawing primitive
  - Collision detection at lowest level requires an efficient method to determine if 2 triangles intersect
- Various Methods
  - Brute force
  - Interval Overlap
    - Devillers-Guigue Method
      - Best performance, but complicated
  - ERIT’s method
    - Simpler than Interval Overlap with near-equal performance

Triangle/Triangle Intersection (cont.)

- Problem description
  - 2 triangles defined by their vertices and plane equations
  - $T_1 = \Delta p_0 p_1 p_2$, In plane: π_1
  - $T_2 = \Delta q_0 q_1 q_2$, In plane: π_2
- Common trivial rejection case using a separating plane
  - Plane of one of the triangles
    - Triangle T_1 lies on plane π_1
      - No intersection if T_2 is entirely above/below π_1
    - Triangle T_2 lies on plane π2
      - No intersection if T_1 is entirely above/below π_2

Co-planar Triangle Intersection

- If all signed distances = 0 then the triangles are coplanar
  - Project onto 2D plane – maximizing area of triangle
    - e.g. if(abs(normal.x) >= abs(normal.y) && abs(normal.x) >= abs(normal.z)) // use the y-z plane
  - 2D triangle-triangle overlap test
    - Test all edges of T_1 for intersection with T_2
      - If any edges intersect then the triangles intersect
    - Test if T_1 is totally enclosed in T_2 and vice versa
      - Point in triangle tests
  - Point-in-triangle test
    - Compute barycentric coordinate of q for Triangle (p_0, p_1, p_2)
    - $u*p_0 + v*p_1 + (1-u-v)*p_2 = q \iff u*(p_2-p_0) + v*(p_2-p_1) = q$
    - Point inside if u >= 0 and v >= 0 and (u+v) <= 1

$$\begin{bmatrix} (p_0 - p_2)_x & (p_1 - p_2)_x \\ (p_0 - p_2)_y & (p_1 - p_2)_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} q_x \\ q_y \end{bmatrix}$$

Line-Segment/Plane Intersection

- Solve plane equation for each end of the line segment
  - $d_1 = \mathrm{n} \cdot p_1 - \mathrm{d} \qquad d_2 = \mathrm{n} \cdot p_2 - \mathrm{d}$
  - No intersection if signs are equal
- Intersection Point
  - $p_i = p_0 + \dfrac{d_1}{d_1 - d_2}\left(p_1 - p_0\right)$

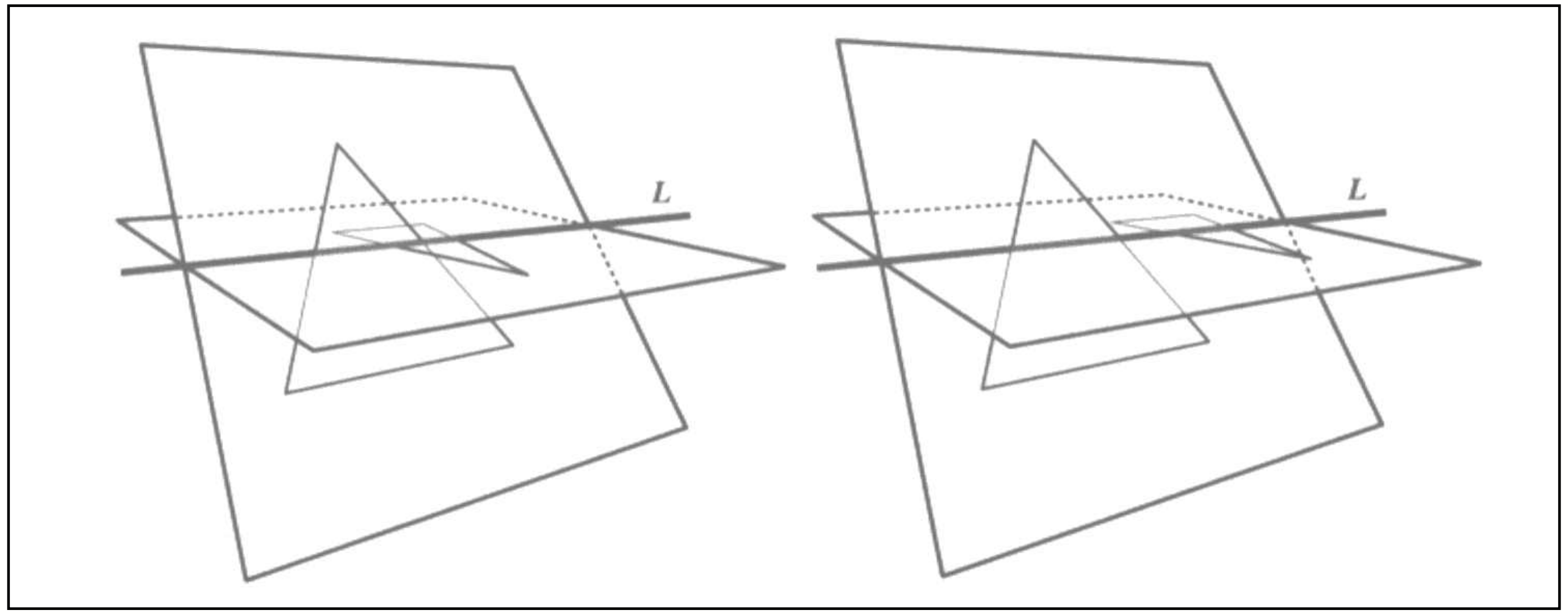
Brute Force

- Method
  - Given two triangles T_1 (p_0, p_1, p_2) and T_2 (q_0, q_1, q_2)
  - Test if p_i-p_{i+1} intersects T_2
  - Test if q_i-q_{i+1} intersects T_1
    - 6 tests
    - Uses the line-segment/plane intersection
      - Test if intersection point is inside the triangle
- Optimization
  - Trivially reject if T_2 above/below π_1
    - or if T_1 above/below π_2
  - Performs relatively well

Interval Overlap Method

- Method
  - Compute the plane equation of T_2
    - Trivial rejection: all points of T_1 on same side of T_2
  - Compute plane equation of T_1
    - Trivial rejection: all points of T_2 on same side of T_1
  - Compute intersection line and project onto largest axis
    - Compute intervals for each triangle
    - Intersect the intervals

Computing Intersect Intervals



- Intersection of Plane1 and Plane2 is a line, l = o + td
- Intersections form intervals on l
  - If these intervals overlap, the triangles overlap as well
  - Figure 22.19 (16.15 in 3rd Edition) - illustration of intersection intervals
  - Due to prior tests, both triangles are guaranteed to intersect l

Computing Intersect Intervals

- Assume p_0 lies on one side of π_2, and p_1 and p_2 lie on the other side
  - Compute intersection of p_0-p_1 and p_0-p_2 with π_2
  - Use line-segment/plane intersection from previous slide
  - Intersection points u_1, u_2
  - Repeat for T_2 against π_1
    - Results in intersection points v_1, v_2
- Treat line as a vector and project the intersection points onto it

$$d_\mathrm{i} = \mathrm{n} \cdot u_\mathrm{i}$$

  - Distance of each point along the line from the origin
    - Triangles intersect if distances for u_i overlap distances for v_i
- See Figure 22.20 (16.16 in 3rd Edition) for geometric representation

Devillers-Guigue Method

- https://inria.hal.science/inria-00072100/document
- Simplified Interval Overlap
  - Compute π_1 for T_1, π_2 for T_2
  - Rearrange T_1 (p_0 above π_2, p_1 and p_2 below π_2)
  - Rearrange T_2 (q_0 above π_1, q_1 and q_2 below π_1)
  - Compute overlap using the “screw” test
    - determinant of 4x4 matrix where columns (or rows) define two line segments
    - uses homogeneous coordinates
  - The entire test becomes
    - T1 intersects T2 if $\text{determinant}_1 >= 0$ and $\text{determinant}_2 >= 0$

$$\det_1 = \begin{vmatrix} p_0 & p_1 & q_0 & q_1 \\ 1 & 1 & 1 & 1 \end{vmatrix} \qquad \det_2 = \begin{vmatrix} p_0 & p_2 & q_2 & q_0 \\ 1 & 1 & 1 & 1 \end{vmatrix}$$

ERIT's Method

- Method from ERIT by Martin Held
  - A collection of Efficient and Reliable Intersection Tests
- Method
  1. Compute plane for T_1 (π_1)
     - Trivial reject: all points of T_2 are on same side of plane
       - Store the signed distances
  2. If triangles are coplanar: use coplanar intersect test
  3. Compute intersection between π_1 and T_2
     - Results in a line segment that is coplanar with π_1
     - Test if this line segment intersects or is totally contained in T_1
       - If so then the triangles intersect, otherwise they do not