Project #2: Classes for shapes (spheres and triangles)

Assign: 8/22

Due: 11:59pm 8/29

Submission: send all your java sources in a zip file and send it to me. Note that the project is accumulated. You may update files in previous projects. Thus, for each submission, please include all files including ones from previous projects.

goals

- In this project, you are asked to implement Java classes, Shape, Sphere, Triangle and Intersection.
- Your classes should support the functions listed in the following slides. You are free to design the interface as long as you support the operations.

Shapes

- One advantages of ray tracing is it can support various kinds of shapes as long as we can find ray-shape intersection.
- Careful abstraction of geometric shapes is a key component for a ray tracer. Ideal candidate for object-oriented design. Scan conversion may not have such a neat interface.
- All shape classes implement the same interface and the other parts of the ray tracer just use this interface without knowing the details about this shape.

Shapes

```
public abstract class Shape
{
public abstract Intersection Intersect(Ray ray);
public abstract bool IntersectP(Ray ray);
}

public class Intersection
{
private Point p;
private Vector n;
Private Shape s;
}
```

Sphere

```
public class Sphere extends Shape {
  private Point origin;
  private float radius;
}

 Sphere(Point p, float r);
```

Intersection

$$(x - O_x)^2 + (y - O_y)^2 + (z - O_z)^2 = r^2$$

$$\left(o_x + td_x - O_x\right)^2 + \left(o_y + td_y - O_y\right)^2 + \left(o_z + td_z - O_z\right)^2 = r^2$$

$$At^2 + Bt + C = 0$$

Step 1

$$A = d_x^2 + d_y^2 + d_z^2$$

$$B = 2[d_x(o_x - O_x) + d_y(o_y - O_y) + d_z(o_z - O_z)]$$

$$C = (o_x - O_x)^2 + (o_y - O_y)^2 + (o_z - O_z)^2 - r^2$$

Intersection

$$t_0 = \frac{-B - \sqrt{B^2 - 4AC}}{2A} \qquad t_1 = \frac{-B + \sqrt{B^2 - 4AC}}{2A}$$

Step 2

If (B^2-4AC<0) then the ray misses the sphere

Step 3

Calculate t_0 and test if $t_0 < 0$ (actually mint, maxt)

Step 4

Calculate t_1 and test if $t_1 < 0$

Normal is P-O where P is the intersection and O is the origin of the sphere.

check the real source code in sphere.cpp, but mind the differences

Triangles

The most commonly used shape.

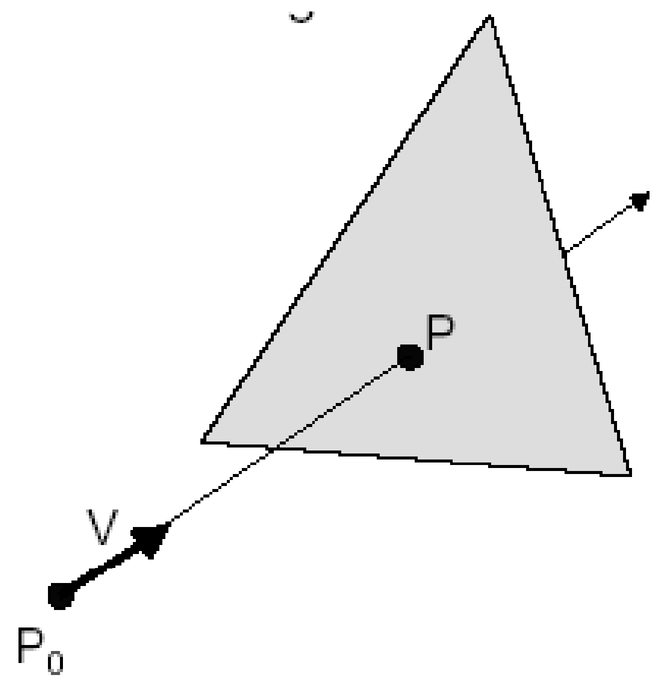
Some ray tracers only support triangle meshes.

Point p1, p2, p3;

Vector n1, n2, n3;

Ray triangle intersection

1. Intersect ray with plane
2. Check if point is inside triangle



Ray plane intersection

$$Ray : P = P_0 + tV$$

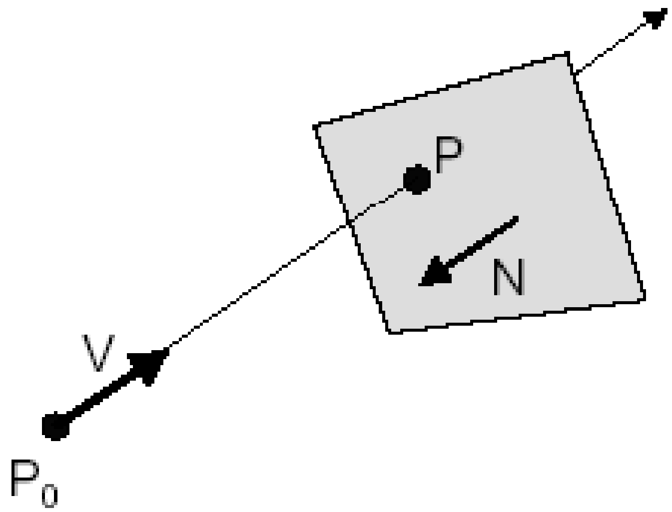
$$Plane : P \cdot N + d = 0$$

Substituting for P, we get:

$$\left(P_0 + tV\right) \cdot N + d = 0$$

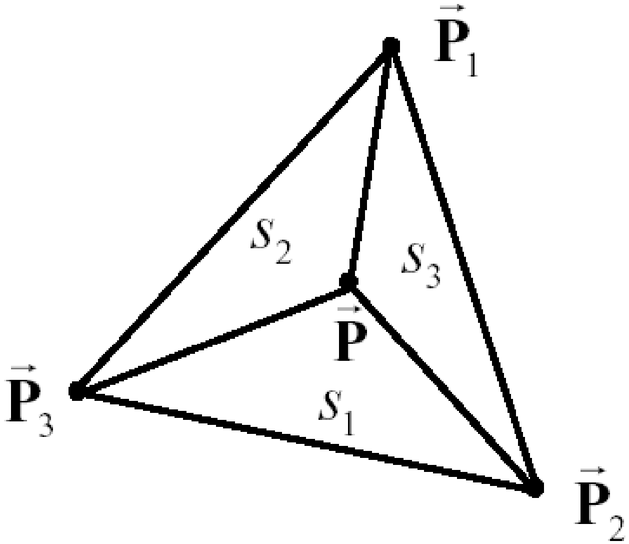
Solution:

$$t = \frac{-\left(P_0 \cdot N + d\right)}{\left(V \cdot N\right)}$$



$$P = P_0 + tV$$

Ray triangle intersection I (recommended)



$$s_1 = \text{area}(\Delta \mathbf{P}\mathbf{P}_2\mathbf{P}_3)$$

$$s_2 = \text{area}(\Delta \mathbf{P}\mathbf{P}_3\mathbf{P}_1)$$

$$s_3 = \text{area}(\Delta \mathbf{P}\mathbf{P}_1\mathbf{P}_2)$$

Barycentric coordinates

$$\vec{\mathbf{P}} = s_1\vec{\mathbf{P}}_1 + s_2\vec{\mathbf{P}}_2 + s_3\vec{\mathbf{P}}_3$$

Inside criteria

$$0 \le s_1 \le 1$$

$$0 \le s_2 \le 1$$

$$0 \le s_3 \le 1$$

$$s_1 + s_2 + s_3 = 1$$

Ray triangle intersection I

- Normal is the linear combination of normals:

$$N = s_1 N_1 + s_2 N_2 + s_3 N_3$$

Ray triangle intersection II

Algebraic Method

For each side of triangle:

$$V_1 = T_1 - P_0$$

$$V_2 = T_2 - P_0$$

$$N_1 = V_1 \times V_2$$

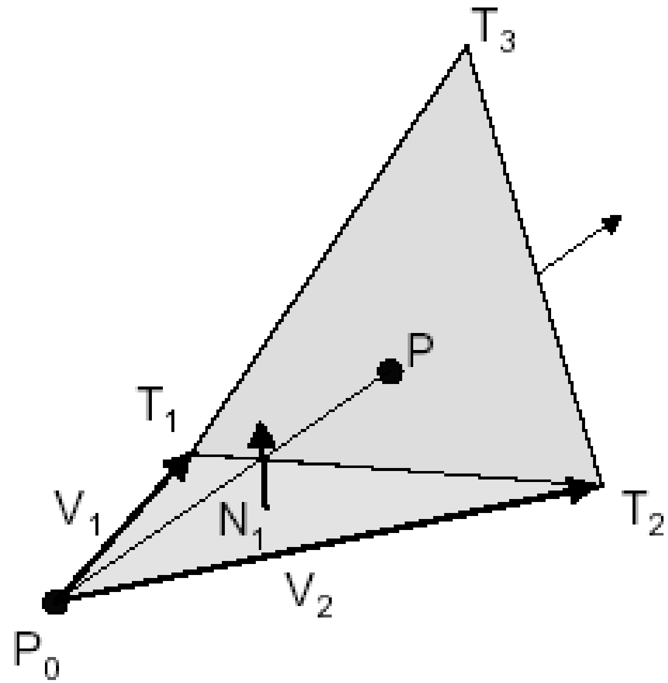
$$Normalize\ \ N_1$$

$$d_1 = -P_0 \cdot N_1$$

$$if\ \left(\left(P \cdot N_1 + d_1\right) < 0\right)$$

$$return\ \ FALSE$$

end



Ray triangle intersection III

Parametric Method

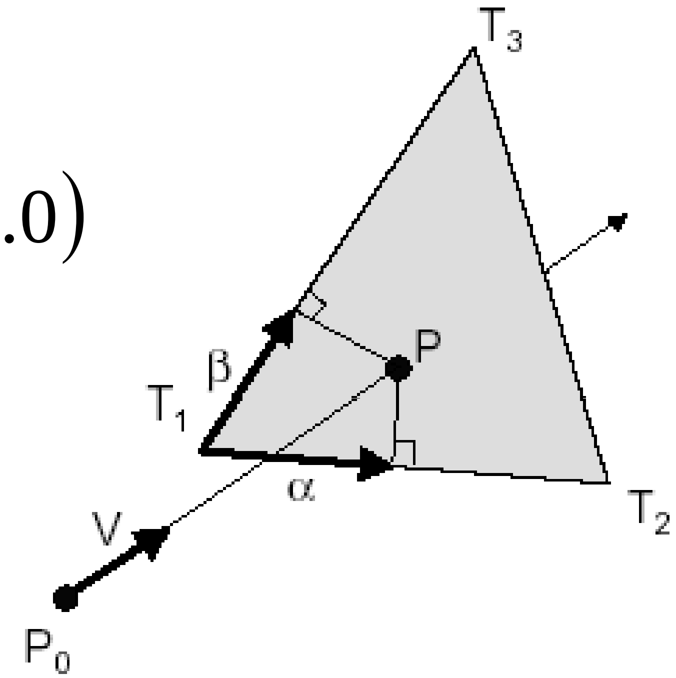
$$Compute\ \alpha, \beta:$$

$$P = \alpha(T_2 - T_1) + \beta(T_3 - T_1)$$

$$if\ (0.0 \le \alpha \le 1.0)\ and\ (0.0 \le \beta \le 1.0)$$

$$and\ (\alpha + \beta \le 1.0)$$

$$then\ P\ is\ inside\ triangle$$



Appendix

Quadric (in pbrt.h) (You can ignore this)

```
inline bool Quadratic(float A, float B, float C,
                      float *t0, float *t1) {
  // Find quadratic discriminant
  float discrim = B * B - 4.f * A * C;
  if (discrim < 0.) return false;
  float rootDiscrim = sqrtf(discrim);
  // Compute quadratic _t_ values
  float q;
  if (B < 0) q = -.5f * (B - rootDiscrim);
  else       q = -.5f * (B + rootDiscrim);
  *t0 = q / A;
  *t1 = C / q;
  if (*t0 > *t1) swap(*t0, *t1);
  return true;
}
```

Why?

- Cancellation error: devastating loss of precision when small numbers are computed from large numbers by addition or subtraction.

```
double x1 = 10.000000000000004;
double x2 = 10.000000000000000;
double y1 = 10.00000000000004;
double y2 = 10.00000000000000;
double z = (y1 - y2) / (x1 - x2); // 11.5
```

$$t_0 = \frac{q}{A} \qquad t_1 = \frac{C}{q}$$

$$q = \begin{cases} -\frac{1}{2}\left(B - \sqrt{B^2 - 4AC}\right) & \text{if } B < 0 \\ -\frac{1}{2}\left(B + \sqrt{B^2 - 4AC}\right) & \text{otherwise} \end{cases}$$

Fast minimum storage intersection

a point on the ray

a point inside the triangle

$$O + tD = (1 - u - v)V_0 + uV_1 + vV_2$$

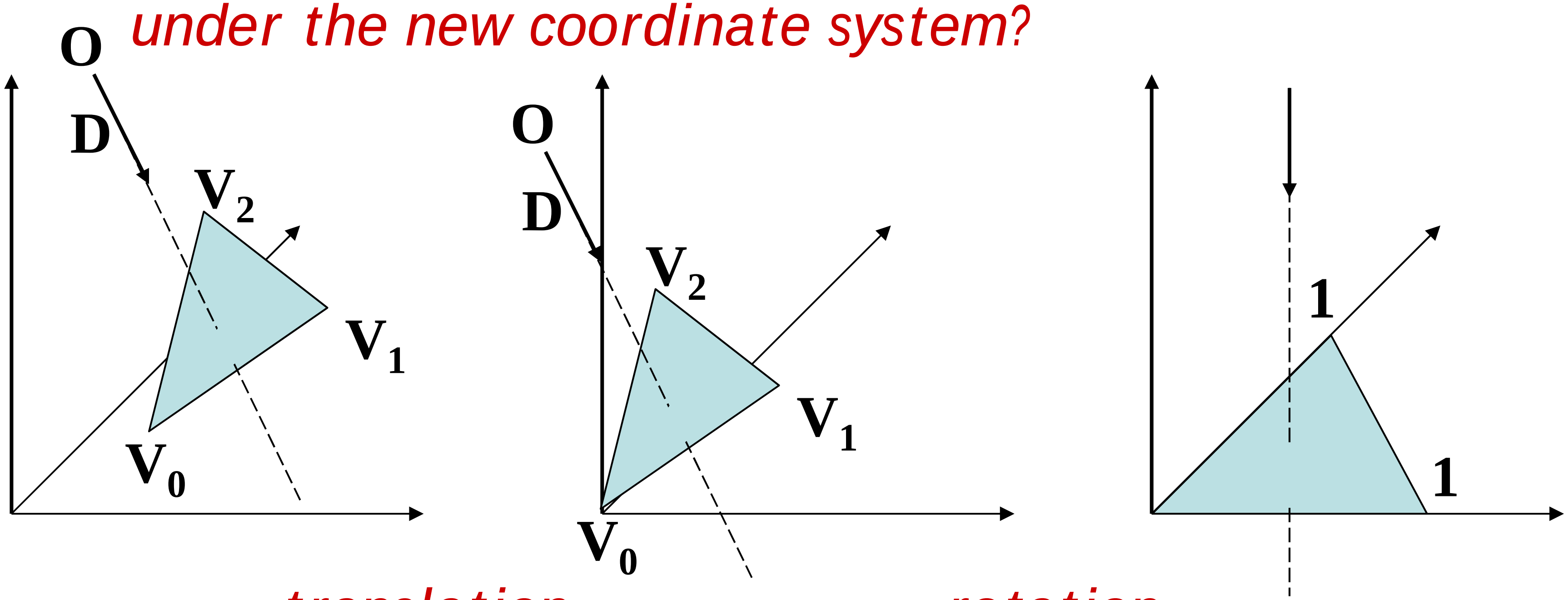
$$u, v \geq 0 \text{ and } u + v \leq 1$$

$$\begin{bmatrix} -D & V_1 - V_0 & V_2 - V_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0$$

Fast minimum storage intersection

$$\begin{bmatrix} -D & V_1 - V_0 & V_2 - V_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0$$

Geometric interpretation: what is O's coordinate under the new coordinate system?



translation *rotation*

Fast minimum storage intersection

$$\begin{bmatrix} -D & V_1 - V_0 & V_2 - V_0 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = O - V_0$$

$$E_1 = V_1 - V_0 \qquad E_2 = V_2 - V_0 \qquad T = O - V_0$$

$$\begin{bmatrix} -D & E_1 & E_2 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = T$$

Fast minimum storage intersection

- Cramer's rule

$$\begin{bmatrix} -D & E_1 & E_2 \end{bmatrix} \begin{bmatrix} t \\ u \\ v \end{bmatrix} = T$$

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\left| -D, E_1, E_2 \right|} \begin{bmatrix} \left| T, E_1, E_2 \right| \\ \left| -D, T, E_2 \right| \\ \left| -D, E_1, T \right| \end{bmatrix}$$

$$\left| A, B, C \right| = -(A \times C) \cdot B = -(C \times B) \cdot A$$

Fast minimum storage intersection

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{\left| -D, E_1, E_2 \right|} \begin{bmatrix} \left| T, E_1, E_2 \right| \\ \left| -D, T, E_2 \right| \\ \left| -D, E_1, T \right| \end{bmatrix}$$

$$Q = T \times E_1 \qquad P = D \times E_2$$

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{P \cdot E_1} \begin{bmatrix} Q \cdot E_2 \\ P \cdot T \\ Q \cdot D \end{bmatrix}$$

1 division
27 multiplies
17 adds