Q1 $\cancel{(dy, dx)}$ $\cancel{(-dy, dx)}$ 1155127464

(i) $\vec{P_2} - \vec{P_1} = (10, 10)$ any point P in $\vec{P_1P_2}$ is $(3,5)+t(10,10)$

We locate P such that $\vec{P_1P} \perp \vec{XP}$

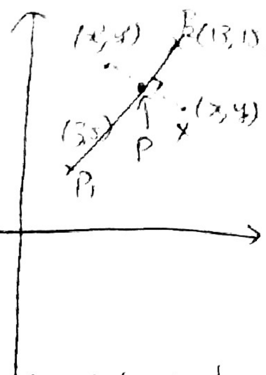
$[(3,5)+t(10,10)] \cdot (x, y) = 0$

$(3+10t)x + (5+10t)y = 0$

$3x+5y + (10x+10y)t = 0$

$t = \frac{3x+5y}{10x+10y}$, $P = (3,5) + \left(\frac{3x+5y}{x+y}\right)(1,1)$

$= \left(\frac{6x+8y}{x+y}, \frac{8x+10y}{x+y}\right)$



We translate the point P to origin, then perform rotation by 180°, then translate back

So the required matrix:

$$\begin{pmatrix} 1 & 0 & \frac{6x+8y}{x+y} \\ 0 & 1 & \frac{8x+10y}{x+y} \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -\frac{6x+8y}{x+y} \\ 0 & 1 & -\frac{8x+10y}{x+y} \\ 0 & 0 & 1 \end{pmatrix}$$

(ii) $$\begin{pmatrix} 1 & 0 & 7 \\ 0 & 1 & 9 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -7 \\ 0 & 1 & -9 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 10 \\ 10 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 7 \\ 0 & 1 & 9 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 1 \\ 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 7 \\ 0 & 1 & 9 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -3 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 8 \\ 1 \end{pmatrix}$$, the result is $(4,8)$ //

(iii) $S = \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $T = \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$, $R = \begin{pmatrix} \cos 90^\circ & -\sin 90^\circ & 0 & 0 \\ \sin 90^\circ & \cos 90^\circ & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

So the product matrix is $R \cdot T \cdot S$

$$= \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(iv) $(RTS)^{-1} = S^{-1} T^{-1} R^{-1}$

$$= \begin{pmatrix} \frac{1}{4} & 0 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & 0 \\ 0 & 0 & \frac{1}{6} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & -3 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

1155127464

Q2 a) $P = V1 - (V1-V3)\frac{y_1 - y_s}{y_1 - y_3}$

$= (50,32,78) - (-20,-56,36)\frac{-4}{-8}$

$= (60,60,60)$ //

$R = V1 - (V1-V2)\frac{y_1 - y_s}{y_1 - y_2}$

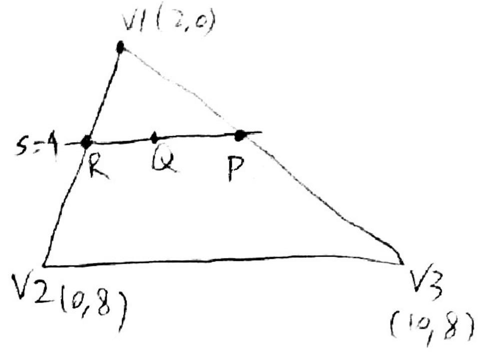
$= (50,32,78) - (50,32,78)\frac{-4}{-8})$

$= (25,16,39)$

$Q = P - (P-R)\frac{x_P - x_Q}{x_P - x_R}$

$= (60,60,60) - (35,44,21)\frac{3}{4}$

$= (33.75, 27, 44.25)$ //



b) $P = V1 - (V1-V3)\frac{x_1 - x_a}{x_1 - x_3}$

$= (50,32,78) - (-20,-56,36)\frac{-4}{-8}$

$= (60,60,60)$

$Q_1 = V1 - (V1-V3)\frac{x_1 - x_b}{x_1 - x_3}$

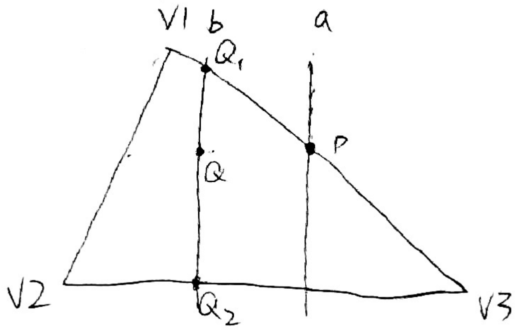
$= (50,32,78) - (-20,-56,36)\frac{-1}{-8} = (52.5, 39, 82.5)$

$Q_2 = V2 - (V2-V3)\frac{x_2 - x_b}{x_2 - x_3}$

$= (0,0,0) - (-70,-88,-42)\frac{-3}{-10} = (21, 26.4, 12.6)$

$Q = Q_1 - (Q_1 - Q_2)\frac{y_{Q1} - y_Q}{y_{Q1} - y_{Q2}}$

$= (52.5, 39, 82.5) - (31.5, 12.6, 69.9)\frac{-3}{-8} = (40.6875, 34.275, 56.2875)$



Which is completely different with the value obtained on (a). It is because the x-component distance may to be the same with the y-component distant. The RGB will only be the same when the pixel is on the edge.

c) Barycentric coordinates: $P = \lambda_1 * V_1 + \lambda_2 * V_2 + \lambda_3 * V_3$, Area (V1, V2, V3) $= \frac{10\times 8}{2} = 40$

For point P: note that Area(V1, V3, P) = 0, $\lambda_2 = 0$

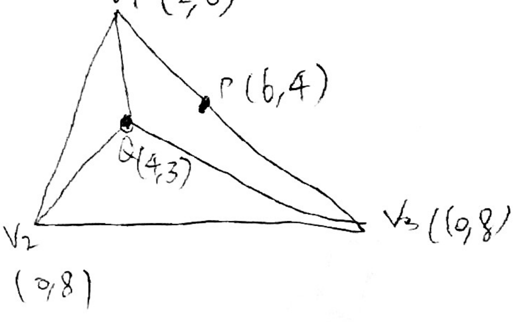
Area (V2, V3, P) $= \frac{10\times 4}{2} = 20$

Area (V1, V2, P) $= 40 - 20 = 20$

So $P = \frac{1}{2} V1 + 0 + \frac{1}{2} V3$

$= \frac{1}{2}(50,32,78) + \frac{1}{2}(70,88,42)$

$= (60,60,60)$

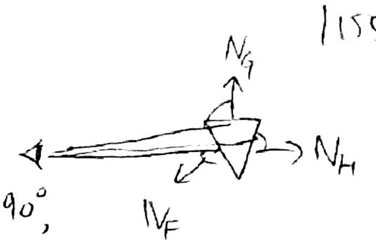


For point Q: Area (V2, V3, Q) $= \frac{10\times 4}{2} = 20$

The RGB color of P will always receive the same colour, because it lies on the edge.
So Area (V1, V3, P) = 0, the Barycentric coordinates become bilinear interpolation

Q3 1155127464

a) Perform dot product



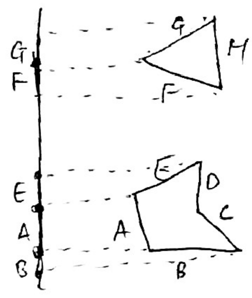
Note that if the angle between V_x and $N_x > 90^\circ$,
we cannot see the face, i.e. $N_x \cdot V_x < 0$

So if $N_x \cdot V_x < 0$, the line segment x should be removed

b) B, C, D, G, H will be removed

c) Yes. We should use orthogonal projection instead of perspective projection.

C, D, H will be removed



Q4 1155127464

(a) $\vec{P_2}-\vec{P_1}=(-1,2,0)$

$\vec{P_3}-\vec{P_1}=(-1,0,3)$

$$(\vec{P_2}-\vec{P_1})\times(\vec{P_3}-\vec{P_1})=\begin{vmatrix}\hat{i} & \hat{j} & \hat{k}\\ -1 & 2 & 0\\ -1 & 0 & 3\end{vmatrix}=6\hat{i}+3\hat{j}+2\hat{k}=(6,3,2)$$

The normal vector is $(6,3,2)$ //

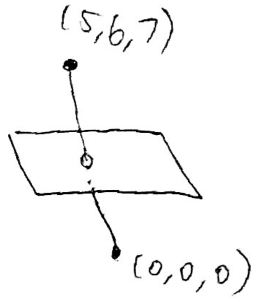
Put $(1,0,0)$ into $6x+3y+2z+D=0$

$D=-6$

So the implicit equation of the plane S is $6x+3y+2z-6=0$ //

(b)

$$d=\frac{(6,3,2,-6)\cdot(5,6,7,1)}{\sqrt{6^2+3^2+2^2}}=\frac{56}{7}=8 \quad //$$



(c) We perform Line clipping

$(6,3,2)\cdot(5,6,7)=62>0$

$(6,3,2)\cdot(0,0,0)=0$

So the line pass through the plane S //

line equation: $(0,0,0)+t[(5,6,7)-(0,0,0)]$

$(5t,6t,7t)$

Substitute $(5t,6t,7t)$ into $S: 6x+3y+2z-6=0$

$30t+18t+14t-6=0$

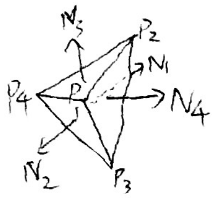
$62t=6,\ t=\frac{3}{31}$

So the intersection point should be $\left(\frac{15}{31},\frac{18}{31},\frac{21}{31}\right)$ //

(d) We can perform dot product. We first obtain four normal vector from four faces

If $P_5\cdot N_i<0$ for all $i=1,2,3,4$, it means all faces do not face toward P_5, P_5 is within the tetrahedron

If $P_5\cdot N_i>0$ for some $i=1,2,3,4$, it means some (at least one) faces do face toward P_5, P_5 is outside the tetrahedron



Q5 1155127464

ai) To perform Flat shading, we just colored all pixels.
Since there are 70 pixels,
the total operation cost is 70C

aii) Gouraud shading: Calculate the intensity at each vertex
For each pixel, we need to apply linear scalar interpolation to obtain its RGB color,
We need to perform 66 times linear scalar interpolation (all pixel unless vertex)
The total operation cost is 66L + 70C

aiii) Phong shading: Interpolates the normal at each vertex
There are four vertex, four averaging normal can be obtained
The total operation cost is 4S + 66L + 70C

bi) K is the diffuse reflection coefficient,
I is the intensity of the light source,
$\hat{N}, \hat{L}, \hat{R}, \hat{V}$ are vectors related with the direction of the light source
K should be the parameters we need.
We should set $\hat{n}$ to (0, 1, 0)

bii) K should be the parameters we need.
We should set $\hat{n}$ to (0, 0, 1)

Alpha Channel | $R = R1 * (1-\alpha * A2) + R2 * (\alpha * A2)$ for each pixel

Barycentric Coordinate

$P = \alpha P_0 + \beta P_1 + \gamma P_2$ where $\alpha + \beta + \gamma = 1$

$A_0 = \alpha A$

$A_1 = \beta A$ i.e. $\alpha = \frac{A_0}{A}$, $\beta = \frac{A_1}{A}$, $\gamma = \frac{A_2}{A}$

$A_2 = \gamma A$

Spherical Linear Interpolation (slerp)

$\vec{v}(t) = \vec{V_0}\,\frac{\sin((1-t)\theta)}{\sin\theta} + \vec{V_1}\,\frac{\sin(t\theta)}{\sin\theta}$

Rotation about X-axis

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation about Y-axis

$$\begin{pmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation about Z-axis

$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Aliasing causes 1. Jagged profile
2. Loss of Details
3. Disintegrating Textures

Anti-aliasing concepts: 1. pre-filtering: Treat each pixel as an area
Compute relative area ratio and mix the colors
2: post-filtering: Combine multiple samples per pixel

Anti-aliasing in Graphics Hardware

1. No AA
2: SSAA (render higher resolution then downsample)
3: MSAA — render multiple samples per pixel
4. FXAA: Fast Approximate — Find all edges in image and smooth the edges
5: TXAA: Specifically to reduce Temporal aliasing

Object space $\xrightarrow{\text{Linear Transformation}}$ World space $\xrightarrow{\text{Linear Transformation}}$ Eye/Camera space $\xrightarrow{\text{Non-linear Transformation}}$ Screen space $\xrightarrow{\text{Rasterization}}$ Raster space

Viewing Coordinate System

Parallel — Orthographic, Oblique
Perspective
Orthographic — Orthogonal, Axonometric

$Z_v = \frac{P-L}{|P-L|}$; $X_v = \frac{V \times Z}{|V \times Z|}$; $Y_v = Z_v \times X_v$

$$M = \begin{pmatrix} X_v & 0 \\ Y_v & 0 \\ Z_v & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 & -P_x \\ 0 & 1 & 0 & -P_y \\ 0 & 0 & 1 & -P_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = RT$$

l, r, b, t, n, f

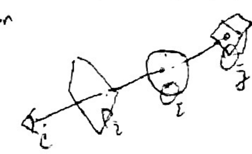
$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & -\frac{2}{f-n} & -\frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

(1) Ray Casting

- For each pixel, construct ray from COP through PP at that pixel and into scene
- Intersect the ray with every object in the scene, color the pixel according to the object with the closest intersection
(image order: pixel only once)

implementation

- parameterize: $R(t) = (1-t)\,c + t p_i$



(2) Z-buffer

- An additional channel in memory for the pixel's depth (Z-buffer)
- Consider object first (once)
- Compare its depth with the depth in Z buffer. Replace if it's closer.

Z Value Interpretation

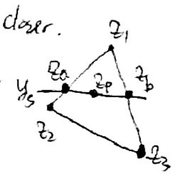
① : bilinear interpolation

② : barycentric coordinates

$z_a = z_1 - (z_1 - z_2)\frac{y_1 - y_s}{y_1 - y_2}$

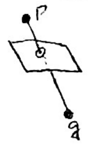
$z_b = z_1 - (z_1 - z_3)\frac{y_1 - y_s}{y_1 - y_3}$

$z_p = z_b - (z_b - z_a)\frac{x_b - x_p}{x_b - x_a}$



$$\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Line clipping

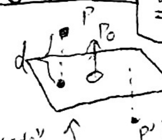


- If $H \cdot p > 0$ and $H \cdot q < 0$
⇒ clip q to plane
- If $H \cdot p < 0$ and $H \cdot q > 0$
⇒ clip p to plane
- If $H \cdot p > 0$ and $H \cdot q > 0$
⇒ pass through
- If $H \cdot p < 0$ and $H \cdot q < 0$
⇒ clipped out

Point Clipping

- $P = (a, b, c)$
- $d = H \cdot P = \frac{(A\,B\,C\,D)(a\ b\ c\ 1)^T}{\sqrt{A^2+B^2+C^2}}$

positive = "inside" negative = "outside"



If $d = H \cdot p \geq 0$, inside

If $d = H \cdot p < 0$, outside

$H_{near} = (0\ 0\ {-1}\ {-near})$ $H_{top} = (0\ d\ {-h}\ 0)$

$H_{far} = (0\ 0\ 1\ far)$ $H_{left} = (d\ 0\ h\ 0)$

$H_{bottom} = (0\ d\ h\ 0)$ $H_{right} = (d\ 0\ {-h}\ 0)$

Outcodes

1010	0010	0110
1000	0000	0100
1001	0001	0101

(3) Back Face Culling

- Used in conjunction with polygon-based algorithms
- No need to draw polygons that face away from the viewer, so we can eliminate (cull) back-facing polygons in the drawing process
- Can be used with ray casting and Z-buffer algorithm
- Use normal vector to obtain $\hat{n}$
- Use dot product to test (visibility test: $N_p \cdot V > 0$)

(4) Potentially Visible Set (PVS)

- Divide the scene into cells (regions)
- Precompute the objects that may be visible in each cell
- acceleration method

Light source | Type of Light source:

- Directional: source infinity away, $(x, y, z, 0)$, Incoming light rays are parallel and of same intensity
- point: finite distance, $(x, y, z, 1)$, goes out from the source in all direction / Attenuation
- spot light source: special point light

Rasterization Algorithms: Line

$y = mx + b$

$m = \frac{y_2 - y_1}{x_2 - x_1}$

$b = \frac{y_1 x_2 - y_2 x_1}{x_2 - x_1}$

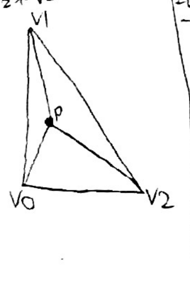
Barycentric coordinates

$P = \lambda_0 * V0 + \lambda_1 * V_1 + \lambda_2 * V2$

$\lambda_0 = \frac{Area(V1, V2, P)}{Area(V0, V1, V2)}$

$\lambda_1 = \frac{Area(V2, V0, P)}{Area(V0, V1, V2)}$

$\lambda_2 = \frac{Area(V0, V1, P)}{Area(V0, V1, V2)}$



Phong Illumination Model

Iteration Zero (Emission)

Given a point P on a surface

- $I = k_e$
- I is the resulting intensity
- k_e is the intrinsic shade

Iteration One (Ambient)

- $I = k_e + k_a I_a$
- k_a is the ambient reflection coefficient of the object
- I_a is the ambient intensity

Interpolative shading

Assumption:
1) Averaging normal
2) shading of a pixel can be obtained by a bilinear interpolation or barycentric-coordinate based interpolation

- Extended Light Source
  - Line or Area
  - more realistic, but computation is very hard
- Environment as a Light source

Iteration Two (Diffuse)

- $I = k_e + k_a I_a + k_d I_1 \cos\theta$ $(\cos\theta = \hat{N} \cdot \hat{L})_+$
- k_d is the diffuse reflection coefficient
- I_1 is the intensity of the light source
- N is the normal at the surface point
- L is the direction to the light source
- $(x)_+$ means $\max\{0, x\}$

Iteration Three (Specular)

- $I = k_e + k_a I_a + \sum_i f(d_i) I_i \left[k_d (\hat{N} \cdot \hat{L}_i)_+ + k_s (\hat{V} \cdot \hat{R})_+^{n_s}\right]$

Phong Illumination model

Depends on surface property
Depends on light intensity
Depends on light direction
Depends on viewing direction

(1) Gouraud shading
Calculate the intensity at each vertex

(2) Phong shading
Interpolates the normal at each vertex