Shading Algorithms

Ron Goldman
Department of Computer Science
Rice University

Illumination and Shading

Standard Assumptions

- Curved surfaces are approximated by planar polygons.
- All light sources are point light sources (at infinity).

Illumination Models

- Ambient Light
- Diffuse Reflection (Scattering)
- Specular Reflection

Shading Models

- Uniform (Flat)
- Gouraud
- Phong

<u>Shading</u>

Algorithms

- Uniform -- constant intensity for each polygon (Flat)
- Gouraud -- linear interpolation of intensity for each polygon along scan lines
- Phong -- linear interpolation of normal vectors (non-linear interpolation of intensity) for each polygon along scan lines

Uniform Shading

Assumptions

1. The surfaces really are polygonal.
   - N = surface normal = constant

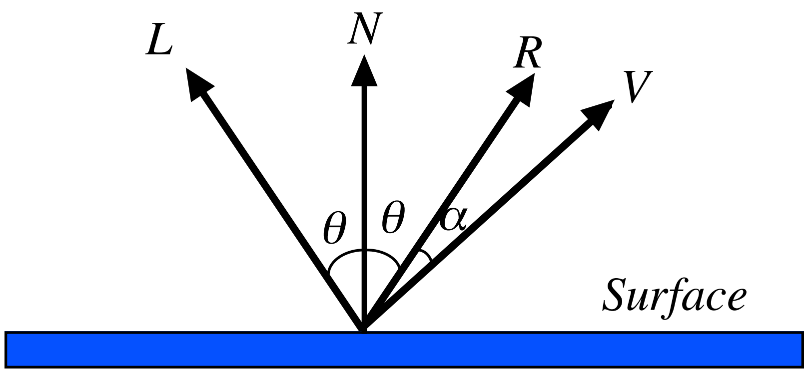
2. Light source at infinity
   - L = vector to light source = constant
   - $L \bullet N =$ constant
   - $R = 2(L \bullet N)N - L =$ constant

3. Eye at infinity
   - V = vector to eye = constant
   - $N \bullet V =$ constant
   - $R \bullet V =$ constant

Uniform Shading (continued)

Intensity

- $$I_{uniform} = \underbrace{I_a k_a}_{ambient} + \underbrace{I_p k_d (L \bullet N)}_{diffuse} + \underbrace{I_p k_s (R \bullet V)^n}_{specular}$$
- N, L, R, V constant $\Rightarrow I_{uniform}$ constant along each polygon



Mach Bands

- Discontinuities in intensity along polygon edges.
- Individual polygons highly visible.
- Heightened by physiological effects of the eye.

<u>Gouraud Shading</u>

Purpose

- To Reduce Mach Bands

Method

- Linear Interpolation of Intensities

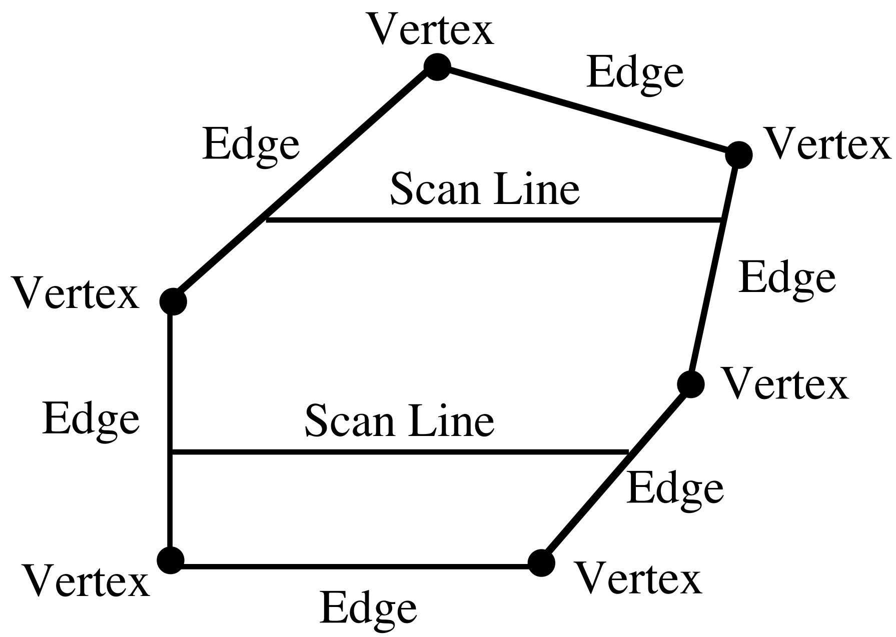
Strategy

1. Compute intensity at vertices, using an average unit normal vector.
   - $N_{vertex} = \dfrac{\sum_{vertex \in Polygon} N_{polygon}}{\left|\sum_{vertex \in Polygon} N_{polygon}\right|}$ (Normalized Average)
2. Interpolate intensity along edges.
3. Interpolate intensity along scan lines.

Observation

- Polygon intensities agree along common edges.
- Integrates well with hidden surface scan line algorithm.

Polygon


Vertex
Edge
Vertex
Edge
Scan Line
Vertex
Edge
Vertex
Edge
Scan Line
Edge
Vertex
Vertex
Edge

Linear Interpolation

Points

$$L(t) = (1 - t)P_1 + tP_2$$

$$L(t) = P_1 + t(P_2 - P_1)$$

$$L(t + \Delta t) = P_1 + (t + \Delta t)(P_2 - P_1)$$

Intensities

$$I(t) = (1 - t)I_1 + tI_2$$

$$I(t) = I_1 + t(I_2 - I_1)$$

$$I(t + \Delta t) = I_1 + (t + \Delta t)(I_2 - I_1)$$

$$\Delta L = \Delta t(P_2 - P_1)$$

- $\Delta x = \Delta t(x_2 - x_1)$
- $\Delta y = \Delta t(y_2 - y_1)$
- $\Delta z = \Delta t(z_2 - z_1)$

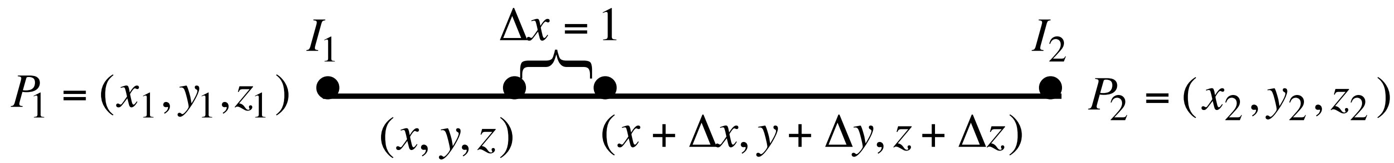
$$\Delta I = \Delta t(I_2 - I_1)$$

$$I_{new} = I_{old} + \Delta I$$

Observation: If we know Δt, then we can compute ΔI.

Incremental Intensity Computation

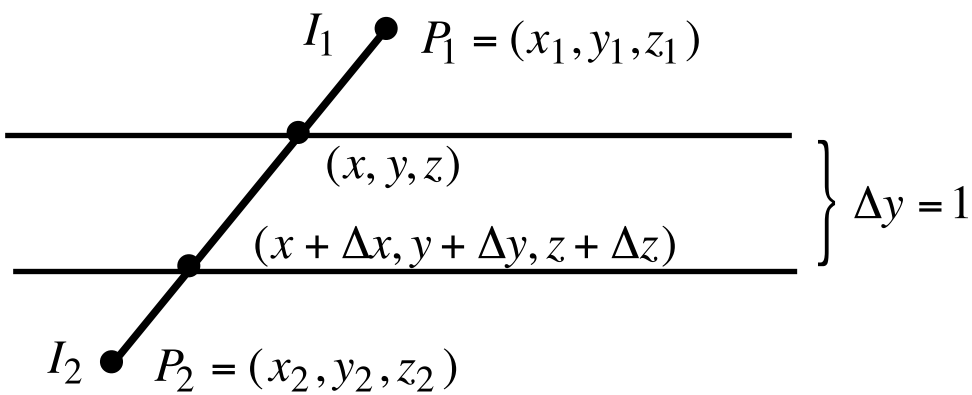
Along a Scan Line



$$\Delta x = 1 \Rightarrow \Delta t = 1/(x_2 - x_1)$$

$$\Delta I = (I_2 - I_1)\Delta t = (I_2 - I_1)/(x_2 - x_1)$$

Next Scan Line



$$\Delta y = 1 \Rightarrow \Delta t = 1/(y_2 - y_1)$$

$$\Delta I = (I_2 - I_1)\Delta t = (I_2 - I_1)/(y_2 - y_1)$$

Orientation Dependence and Independence

Problem

- Gouraud (Phong) Shading is Orientation Dependent

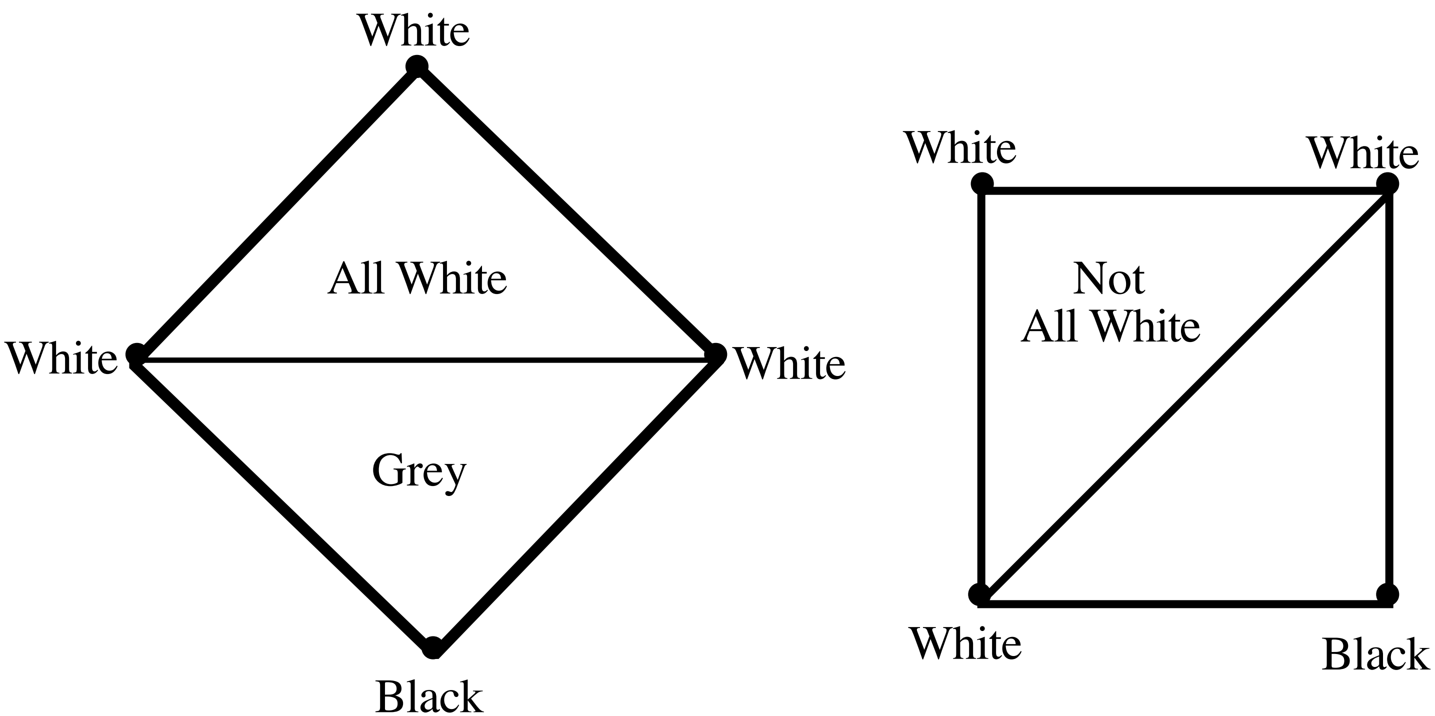
Solution

- Subdivide the Polygons into Triangles
- Gouraud (Phong) Shading for Triangles is Orientation Independent

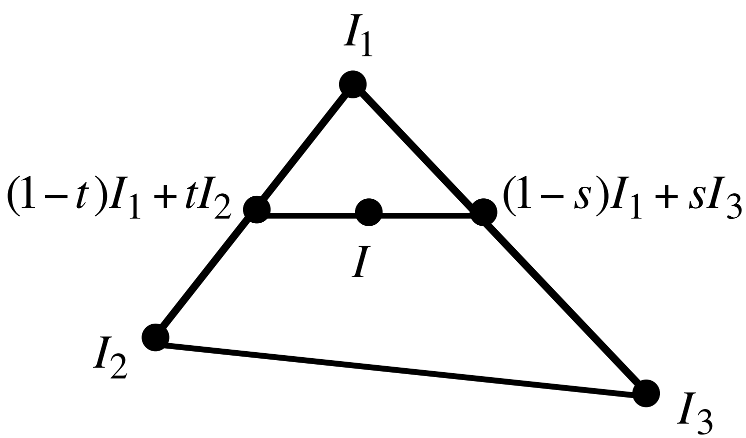
Note

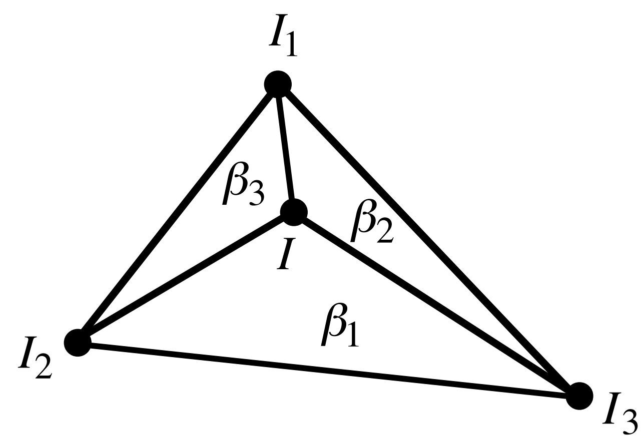
- Shading Algorithm Depends on the Particular Triangular Subdivision
- Shading Algorithm Gives Different Results for Different Subdivisions

Coordinate Dependence



Barycentric Coordinates and Coordinate Independence





$$I = (1-u)\big((1-t)I_1 + tI_2\big) + u\big((1-s)I_1 + sI_3\big)$$

$$I = \beta_1 I_1 + \beta_2 I_2 + \beta_3 I_3$$

$$I = \underbrace{\big((1-u)(1-t) + u(1-s)\big)}_{\beta_1} I_1 + \underbrace{\big((1-u)t\big)}_{\beta_2} I_2 + \underbrace{(us)}_{\beta_3} I_3$$

Barycentric Coordinates

- $\beta_1 + \beta_2 + \beta_3 = 1$
- $P = \beta_1 P_1 + \beta_2 P_2 + \beta_3 P_3 = P_1 + \beta_2 (P_2 - P_1) + \beta_3 (P_3 - P_1)$
- $\beta_1, \beta_2, \beta_3$ unique

Phong Shading

Purpose

- To Reduce Mach Bands
- To Mimic Curved Surfaces

Method

- Linear Interpolation of Normal Vectors

Strategy

1. Compute unit normals at vertices by averaging the unit normals of the polygons to which the vertex belongs.
   - $N_{vertex} = \dfrac{\sum_{vertex \in Polygon} N_{polygon}}{\left|\sum_{vertex \in Polygon} N_{polygon}\right|}$ (Normalized Average)
2. Interpolate normals along edges. (Renormalize)
3. Interpolate normals along scan lines. (Renormalize)
4. Use normals to calculate intensities.

Linear Interpolation

Points

$$L(t) = (1-t)P_1 + tP_2$$

$$L(t) = P_1 + t(P_2 - P_1)$$

$$L(t+\Delta t) = P_1 + (t+\Delta t)(P_2 - P_1)$$

Normals

$$N(t) = (1-t)N_1 + tN_2$$

$$N(t) = N_1 + t(N_2 - N_1)$$

$$N(t+\Delta t) = N_1 + (t+\Delta t)(N_2 - N_1)$$

$$\Delta L = \Delta t(P_2 - P_1)$$

- $\Delta x = \Delta t(x_2 - x_1)$
- $\Delta y = \Delta t(y_2 - y_1)$
- $\Delta z = \Delta t(z_2 - z_1)$

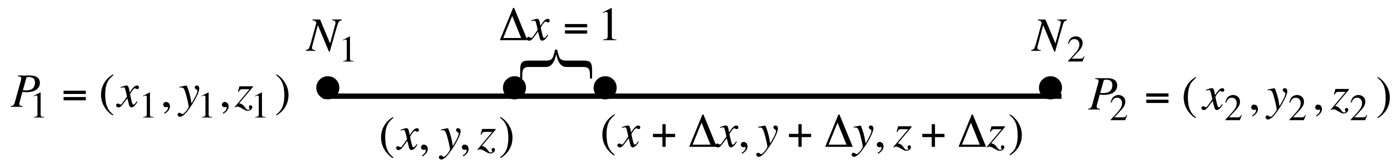
$$\Delta N = \Delta t(N_2 - N_1)$$

$$N_{new} = \frac{N_{old} + \Delta N}{\left|N_{old} + \Delta N\right|}$$

Observation: If we know Δt, then we can compute ΔN.

Incremental Normal Computation

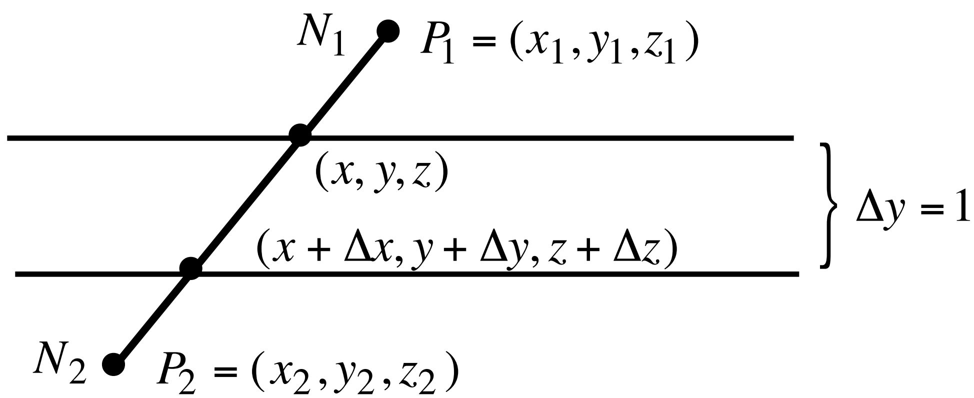
Along a Scan Line



$$\Delta x = 1 \Rightarrow \Delta t = 1/(x_2 - x_1)$$

$$\Delta N = (N_2 - N_1)\Delta t = (N_2 - N_1)/(x_2 - x_1)$$

Next Scan Line



$$\Delta y = 1 \Rightarrow \Delta t = 1/(y_2 - y_1)$$

$$\Delta N = (N_2 - N_1)\Delta t = (N_2 - N_1)/(y_2 - y_1)$$

Clever Implementation -- Diffuse Reflection

Diffuse Reflection

$$I_{diffuse} = I_p k_d (L \bullet N) = I_p k_d \frac{(1-t)(L \bullet N_1) + t(L \bullet N_2)}{\|(1-t)N_1 + t\,N_2\|}$$

- $(1-t)\underbrace{(L \bullet N_1)}_{I_1} + t\underbrace{(L \bullet N_2)}_{I_2}$ -- Similar to Gouraud (Scalars)
- $d^2 = \|(1-t)N_1 + t\,N_2)\|^2 = \|N + \Delta N\|^2 = (N + \Delta N) \bullet (N + \Delta N)$
- $d^2 = N \bullet N + 2\,N \bullet \Delta N + \Delta N \bullet \Delta N = 1 + 2\,N \bullet \Delta N + \Delta N \bullet \Delta N$

Square Root by Newton's Method

- $x^2 - d^2 = 0$
- $x_{n+1} = x_n - \dfrac{x_n^2 - d^2}{2x_n}$

Clever Implementation -- Diffuse Reflection (continued)

Length of Normal Vectors

- $$D(t) = \|(1-t)N_1 + t\,N_2)\|^2 = \big((1-t)N_1 + t\,N_2)\big) \bullet \big((1-t)N_1 + t\,N_2)\big)$$

- $$D(t) = \left((1-t)^2 (\underbrace{N_1 \bullet N_1}_{1}) + t^2 (\underbrace{N_2 \bullet N_2}_{1}) + 2t\,(1-t)(N_1 \bullet N_2) \right)$$

- $$D(t) = \big((1-t)^2 + t^2 + 2t(1-t)(N_1 \bullet N_2)\big)$$

- $$D(t+\Delta t) = \big((1-t-\Delta t)^2 + (t+\Delta t)^2 + 2(t+\Delta t)(1-t-\Delta t)(N_1 \bullet N_2)\big)$$

Change in Length

- $$\Delta D = \big(2(1-t)\Delta t + \Delta t^2\big) + (2t\,\Delta t + \Delta t^2) + 2(\Delta t\,(1-t) - t\,\Delta t - \Delta t^2)(N_1 \bullet N_2)\big)$$

-- $$\Delta D = 2\Delta t\big(1 + \Delta t + (1 - 2\,t + \Delta t)(N_1 \bullet N_2)\big)$$

-- $$\Delta D = \underbrace{2\Delta t\big(1 + \Delta t + (1 + \Delta t)(N_1 \bullet N_2)\big)}_{constant} - \underbrace{2t(N_1 \bullet N_2)}_{linear\ in\ t}$$

Clever Implementation -- Diffuse Reflection (continued)

Change in D

- $\Delta D = \underbrace{2\Delta t\left(1 + \Delta t + (1 + \Delta t)(N_1 \bullet N_2)\right)}_{constant} - \underbrace{2t(N_1 \bullet N_2)}_{linear\ in\ t}$
  - -- $E = 2\Delta t\left(1 + \Delta t + (1 + \Delta t)(N_1 \bullet N_2)\right)$
  - -- $F(t) = 2t(N_1 \bullet N_2)$
- $\Delta D = E - F(t)$

Change in F

- $F(t) = 2t(N_1 \bullet N_2)$
- $F(t + \Delta t) = 2(t + \Delta t)(N_1 \bullet N_2)$
  - -- $\Delta F = \underbrace{2\,\Delta t(N_1 \bullet N_2)}_{constant}$

Clever Implementation -- Specular Reflection

Specular Reflection

$$I_{specular} = I_p k_s (R \bullet V)^n$$

- $R = 2(L \bullet N)N - L$

$$I_{specular} = I_p k_s \left(2(L \bullet N)(N \bullet V) - (L \bullet V)\right)^n$$

$$I_{specular} = I_p k_s \left(\frac{2\left\{\underbrace{(1-t)(L \bullet N_1) + t(L \bullet N_2)}_{Same\ \ as\ Gouraud}\right\}\left\{\underbrace{(1-t)(V \bullet N_1) + t(V \bullet N_2)}_{Same\ \ as\ Gouraud}\right\}}{\left\|(1-t)N_1 + t\ N_2\right\|^2} - \underbrace{(L \bullet V)}_{Constant} \right)^n$$

- $\left\|(1-t)N_1 + t\ N_2\right\|^2 = \left\|N + \Delta N\right\|^2$ -- No Square Root Required

Spherical Linear Interpolation

Slerp

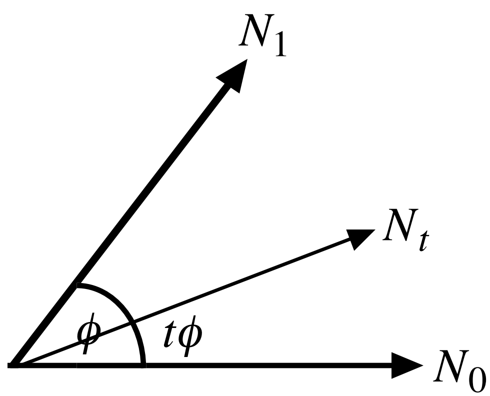
- $slerp(N_1, N_2, t) = \dfrac{\sin\left((1-t)\phi\right)}{\sin(\phi)} N_1 + \dfrac{\sin\left(t\phi\right)}{\sin(\phi)} N_2$
  - $\cos(\phi) = N_1 \bullet N_2$ (dot product)

- *slerp* maps unit vectors to unit vectors (along geodesics)
  - more natural than linear interpolation

- Expand incrementally
  - Expand Δsine and Δcosine
  - Precompute $\sin(\Delta t\, \phi)$ and $\cos(\Delta t\, \phi)$

Interpolating Unit Vectors

Vector Interpolation Problem



$N_0, N_1 =$ unit vectors

Vector Interpolation Formula

- $$N_t = \frac{\sin\big((1-t)\phi\big)}{\sin(\phi)} N_0 + \frac{\sin\big(t\phi\big)}{\sin(\phi)} N_1$$

Derivation

Theorem: $N_t = \dfrac{\sin\big((1-t)\phi\big)}{\sin(\phi)} N_0 + \dfrac{\sin\big(t\phi\big)}{\sin(\phi)} N_1$

Proof: Rotate N_0 by $t\phi$ around $w = \dfrac{N_0 \times N_1}{\left|N_0 \times N_1\right|} = \dfrac{N_0 \times N_1}{\sin(\phi)}$.

- $N_t = \cos(t\phi)N_0 + \big(1-\cos(t\phi)\big)\underbrace{(N_0 \bullet w)}_{0} w + \sin(t\phi)(w \times N_0)$
  - $(N_0 \times N_1) \times N_0 = (N_0 \bullet N_0)N_1 - (N_0 \bullet N_1)N_0 = N_1 - \cos(\phi)N_0$
- $N_t = \cos(t\phi)N_0 + \sin(t\phi)\left(\dfrac{N_1 - \cos(\phi)N_0}{\sin(\phi)}\right)$

$$= \left(\frac{\sin(\phi)\cos(t\phi) - \sin(t\phi)\cos(\phi)}{\sin(\phi)}\right) N_0 + \left(\frac{\sin(t\phi)}{\sin(\phi)}\right) N_1$$

$$= \left(\frac{\sin(\phi - t\phi)}{\sin(\phi)}\right) N_0 + \left(\frac{\sin(t\phi)}{\sin(\phi)}\right) N_1$$

Alternative Derivation

Theorem: $N_t = \dfrac{\sin\big((1-t)\phi\big)}{\sin(\phi)} N_0 + \dfrac{\sin\big(t\phi\big)}{\sin(\phi)} N_1$

Proof: The vectors N_0, N_1, N_t are unit vectors, and

$$N_t = \alpha N_0 + \beta N_1.$$

Dotting both sides with N_0 and N_1 yields:

$$N_0 \bullet N_t = \alpha + \beta\, N_0 \bullet N_1 \Rightarrow \cos(t\phi) = \alpha + \cos(\phi)\, \beta$$

$$N_1 \bullet N_t = \alpha\, N_0 \bullet N_1 + \beta \Rightarrow \cos\big((1-t)\phi\big) = \cos(\phi)\, \alpha + \beta$$

Solving for α, β by Cramer's rule:

$$\alpha = \frac{\det\begin{pmatrix} \cos(t\phi) & \cos(\phi) \\ \cos\big((1-t)\phi\big) & 1 \end{pmatrix}}{\det\begin{pmatrix} 1 & \cos(\phi) \\ \cos(\phi) & 1 \end{pmatrix}} = \frac{\cos(t\phi) - \cos(\phi)\cos\big((1-t)\phi\big)}{1-\cos^2(\phi)}$$

Alternative Derivation (continued)

Therefore

$$\alpha = \frac{\cos(t\phi) - \cos(\phi)\big(\cos(\phi)\cos(t\phi) + \sin(\phi)\sin(t\phi)\big)}{\sin^2(\phi)}$$

$$= \frac{\cos(t\phi)\left(1 - \cos^2(\phi)\right) + \cos(\phi)\sin(\phi)\sin(t\phi)}{\sin^2(\phi)}$$

$$= \frac{\cos(t\phi)\sin(\phi) - \cos(\phi)\sin(t\phi)}{\sin(\phi)}$$

$$= \frac{\sin(\phi - t\phi)}{\sin(\phi)}$$

Similarly

$$\beta = \frac{\sin(t\phi)}{\sin(\phi)} \, .$$

Incremental Spherical Linear Interpolation

Slerp

- $slerp(N_1, N_2, t) = \dfrac{\sin\big((1-t)\phi\big)}{\sin(\phi)} N_1 + \dfrac{\sin\big(t\phi\big)}{\sin(\phi)} N_2$
  - $\cos(\phi) = N_1 \bullet N_2$

Incremental Computation

- $slerp(N_1, N_2, t + \Delta t) = \dfrac{\sin\big((1-t-\Delta t)\phi\big)}{\sin(\phi)} N_1 + \dfrac{\sin\big((t+\Delta t)\phi\big)}{\sin(\phi)} N_2$

-- $\sin\big((t+\Delta t)\phi\big) = \sin(t\phi)\cos(\Delta t\,\phi) + \cos(t\phi)\sin(\Delta t\,\phi)$

-- $\sin\big((1-t-\Delta t)\phi\big) = \sin((1-t)\phi)\cos(\Delta t\,\phi) - \cos((1-t)\phi)\sin(\Delta t\,\phi)$

-- $\cos\big((t+\Delta t)\phi\big) = \cos(t\phi)\cos(\Delta t\,\phi) - \sin(t\phi)\sin(\Delta t\,\phi)$

-- $\cos\big((1-t-\Delta t)\phi\big) = \cos((1-t)\phi)\cos(\Delta t\,\phi) + \sin((1-t)\phi)\sin(\Delta t\,\phi)$

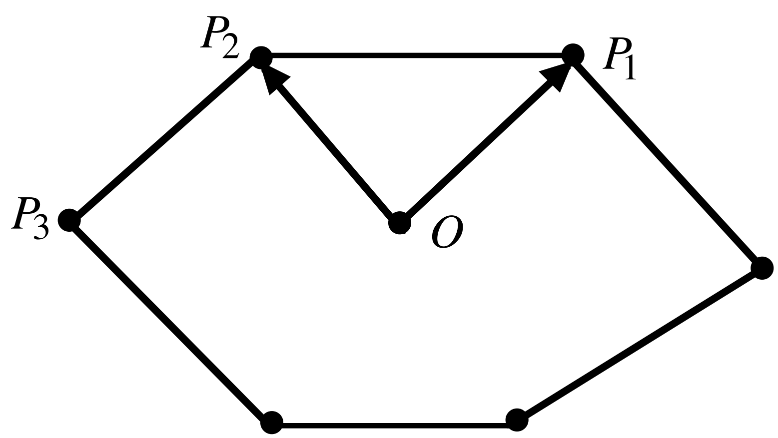
Polygon Normals

Normal to a Polygon

- $N = (P_k - O) \times (P_{k+1} - O) = P_k \times P_{k+1} - P_k \times O + P_{k+1} \times O$

Newell's Formula

- $N = \sum_{k=0}^{m} P_k \times P_{k+1}$
- $|N| = 2 \times Area(Polygon)$ (Green's Theorem)
- Avoids problem of collinear vertices



<u>Comparison to Gouraud Shading</u>

Similar to Gouraud Shading

- Intensity calculations replaced by normal calculations
- Integrates well with hidden surface scan line algorithm
- Coordinate dependent method

Slower than Gouraud Shading

- Intensities must be recalculated at each point from the normal vector
- $$I = \underbrace{I_a k_a}_{ambient} + \underbrace{I_p k_d (L \bullet N)}_{diffuse} + \underbrace{I_p k_s (R \bullet V)^n}_{specular}$$
  - N is different for each point
  - R is different for each point

More Accurate than Gouraud Shading

- Simulates curved surfaces (varying normal vectors)
- Further reduces Mach Bands
- Tighter highlights (and spotlights) than Gouraud shading
- Gouraud shading interpolates intensities -- smooths out highlights

Problems with Gouraud and Phong Shading

Orientation Dependence

- Problem: Linear Interpolation for Polygons is Orientation Dependent
- Visual Effect: Shading Changes Abruptly During Animation
- Solution: Subdivide into Triangles

Distortion

- Problem: Intensities are Interpolated Along Projected Edges,
  Not Along True Edges
- Visual Effect: Introduces Distortions
- Solution: Introduce Mass into Intensity Calculations

Distortion

Linear Interpolation (Before Pseudoperspective)

$$P = (1-t)P_1 + tP_2$$

$$I = (1-t)I_1 + tI_2$$

$$t = \frac{|P - P_1|}{|P_2 - P_1|}$$

Linear Interpolation (After Pseudoperspective)

$$P^{**} = (1-t^*)P_1^{**} + t^* P_2^{**}$$

$$t^* = \frac{|P^{**} - P_1^{**}|}{|P_2^{**} - P_1^{**}|} \neq \frac{|P - P_1|}{|P_2 - P_1|} = t$$

$$I^{**} = (1-t^*)I_1 + t^* I_2 \neq I$$

<u>**Fixing Distortion**</u>

Pseudoperspective

$$Pseudo(P) = (1-t)Pseudo(P_1) + tPseudo(P_2)\ .$$

$$(mP^{**}, m) = (1-t)(m_1 P_1^{**}, m_1) + t(m_2 P_2^{**}, m_2)$$

$$mP^{**} = (1-t)m_1 P_1^{**} + tm_2 P_2^{**}$$

$$m = (1-t)m_1 + tm_2$$

Linear Interpolation (Right Way)

$$P^{**} = \frac{mP^{**}}{m} = \frac{(1-t)m_1 P_1^{**} + tm_2 P_2^{**}}{(1-t)m_1 + tm_2}\ .$$

$$P^{**} = (1-t^*)P_1^{**} + t^* P_2^{**}$$

$$t^* = \frac{tm_2}{(1-t)m_1 + tm_2}$$

$$t = \frac{t^* m_1}{(1-t^*)m_2 + t^* m_1}$$

Fixing Distortion Incrementally

Blinn's Way

$$P_1 \to (m_1 P_1^{**}, m_1) \qquad P_2 \to (m_2 P_2^{**}, m_2)$$

$$I_1 \to \left(\frac{I_1}{m_1}, \frac{1}{m_1}\right) \qquad I_2 \to \left(\frac{I_2}{m_2}, \frac{1}{m_2}\right)$$

$$P^{**} = (1-t^*)P_1^{**} + t^* P_2^{**} \qquad t^* = \frac{tm_2}{(1-t)m_1 + tm_2}$$

$$I = \frac{(1-t^*)\dfrac{I_1}{m_1} + t^* \dfrac{I_2}{m_2}}{(1-t^*)\dfrac{1}{m_1} + t^* \dfrac{1}{m_2}} = (1-t)I_1 + tI_2$$

$$\Delta(I/m) = \Delta t^* \left(I_2/m_2 - I_1/m_1\right)$$

$$\Delta(1/m) = \Delta t^* \left(1/m_2 - 1/m_1\right)$$