DIEGO DANTAS ALMEIDA - LISTA 5

1- O shading consiste em calcular a iluminação apenas de alguns pixels da imagem (ex: vértices de pequenos triângulos) e fazer uso de algoritmos de shading para determinar a iluminação dos outros pixels por aproximações (ex.: interpolação).

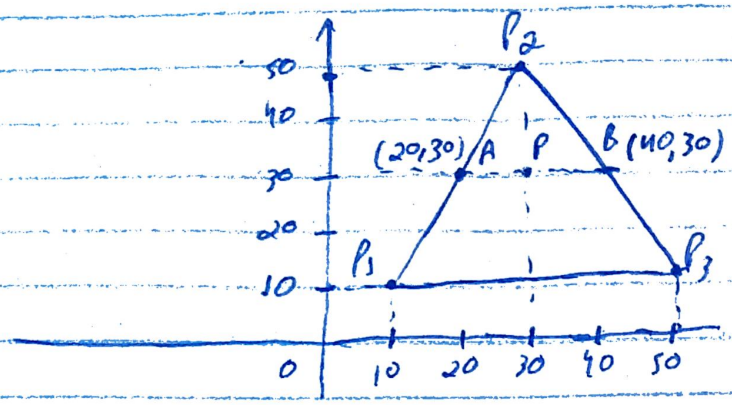
2- Flat - Determina as coordenadas do ponto central de cada polígono (baricentro), calcula a normal e outros raios e o valor da iluminação para esse ponto, que será o mesmo para todos os pixels dentro do polígono.

Gouraud - Especifica um valor de intensidade (cor) diferente para cada vértice do polígono (polígonos adjacentes tem o valor da borda em comum), interpola na hora de fazer a renderização.

Phong - Calcula a normal em cada vértice, posteriormente, para determinar a cor de cada pixel, é feita uma interpolação das normais linearmente, usando a mesma ideia do algoritmo de Gouraud. Finalmente, calcula o valor de shading usando a normal interpolada (com a equação de iluminação).

3- O algoritmo de shading default do OpenGL é o de Gouraud.

4-



$\mu_1 = d(P_2,A)/d(P_1,P_2)$ $\quad P = (30,30)$

$\mu_1 = \frac{22,36}{44,72} \cong 0,5$

$A_3 = \mu_1 \cdot S_1 + (1-\mu_1)\cdot S_2$

$A_3 = 0,5 \cdot 200 + 0,5 \cdot 180 = 190$

$\mu_2 = \frac{d(P_2,B)}{d(P_1,P_3)} = \frac{22,36}{44,72} \cong 0,5$

$B_3 = 0,5 \cdot 230 + 0,5 \cdot 180 = 205$

$\sigma_1 = \frac{205 - 190}{20} = 0,75$ $\quad P_x - A_x = 10$

$P_3 = A_3 + 10 \cdot 0,75 \cong 197,5 = 198$

1=RETA

$y = 2x + b$

$10 = 2 \cdot 10 + b$

$10 = 20 + b$

$b = -10$

$y = 2x - 10$

$30 = 2x - 10$

$40 = 2x$

$x = 20$

2=RETA

$y = -2x + b$

$10 = -2 \cdot 50 + b$

$10 = -100 + b$

$b = 110$

$y = -2x + 110$

$30 = -2x + 110$

$2x = 80$

$x = 40$

5- $P = \alpha P_1 + \beta P_2 + \gamma P_3$

$(30,30) = \alpha(10,20) + \beta(30,50) + \gamma(50,10)$

$30 = 10\alpha + 30\beta + 50\gamma$

$30 = 10\alpha + 50\beta + 10\gamma$

$1 = \alpha + \beta + \alpha$

$$\begin{bmatrix} 10 & 30 & 50 \\ 10 & 50 & 10 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 30 \\ 30 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} 10 & 30 & 50 \\ 10 & 50 & 10 \\ 1 & 1 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 30 \\ 30 \\ 1 \end{bmatrix} = \begin{bmatrix} 0,25 \\ 0,5 \\ 0,25 \end{bmatrix}$$

$\alpha = 0,25 \; ; \; \beta = 0,5 \; ; \; \gamma = 0,25$

6 - $P_5 = 0{,}25 \cdot 200 + 0{,}5 \cdot 180 + 0{,}25 \cdot 230 = 197{,}5 \leq 198$

7 - Consiste em mapear uma imagem na superfície de um objeto usando uma função paramétrica que mapeia pontos (u, v) em coordenadas de imagem (x, y). É utilizado para determinar a cor (textura) de cada pixel no objeto.

8 - Utilizando funções paramétricas. Dependendo do formato do objeto existem funções paramétricas mais adequadas.
Ex: - Esferas: coordenadas esféricas $(\phi, \theta) \Rightarrow (\alpha\pi u, \pi v)$
- Cilindro: coordenadas cilíndricas $(u, \theta) \Rightarrow (u, \alpha\pi v)$
- Superfícies paramétricas (B-splines, Bézier): (u, v)

9 - Cor da superfície, refletância, vetor normal, geometria, transparência, radiância.

10 - No mapa de deslocamentos há realmente a alteração da superfície do objeto, com "deslocamento" da normal em determinados pontos, posteriormente é feita a rasterização. Já o "bump map" pode ser feito em tempo de rasterização somente modificando a cor do objeto para simular as alterações da normal.

11 - A ideia é gerar um array 3D de valores de textura (como se fosse um bloco de mármore) e usar uma função $(x, y, z) \rightarrow (R, G, B)$ para mapear cores em pontos do espaço de acordo com o formato do objeto.