Gabarito da P3 CG: 2021.1 - Exame

**Questao 1**

Tem-se uma engrenagem 3D definida por faces triangulares. A engrenagem tem raio 300. O

desenvolvedor observou que a iluminação não está correta e desconfia que as normais estejam

erradas. Elabore um algoritmo para calcular e exibir o vetor normal para cada face desta

engrenagem para auxiliar o programador na verificação. Utilize vetores, operações e figuras

ilustrativas.

R:

funcao calculaNormal(p1, p2, p3)

v1 = p2 - p1

v2 = p2 - p3

n = v1 cross v2 //cria vetor perpendicular ao triangulo

n = n.normalize

n \*= 100 //vetor normal com 100 unidades de comprimento para poder visualizar na tela

retorna n

fim

para cada triangulo

normal = calculaNormal(p1, p2, p3) //passa os 3 vertices de cada triangulo

centro = (p1 + p2 + p3 ) / 3 //centro do triangulo

normal = normal + centro; //move a normal na posicao do triangulo

line(centro, normal) //desenha o vetor normal

fim

**Questao 2**

O que é a iluminação difusa? Mostre a diferença entre a iluminação difusa por vértice e por pixel.

Utilize figuras ilustrativas e comente.

R:

Iluminação difusa ocorre em superfícies rugosas e retorna a cor do objeto, em função da normal da face, coeficiente de reflexão difusa e posição da fonte luminosa (ângulo entre normal e fonte), conforme equação.

Id = Kd \* Ifonte \* N . L

Na iluminação por vértice (como o opengl trabalha), calcula-se a cor de cada vértice com a equação acima. Após, faz-se interpolação dessas cores para calcular a cor de cada ponto da face

Não iluminação por pixel, calcula-se uma normal para cada pixel pela interpolação das 3 normais do triangulo. Após, aplica-se para cada pixel, com sua normal, a equação acima.

**Questao 3**

Dado uma sequência de N pontos de controle (N ≥ 4), elabore um algoritmo detalhado para desenhar uma curva fechada que liga todos os pontos de controle. A curva deve ter continuidade C2. A curva deve ser desenhada como uma sequência de linhas. Coloque comentários sobre a teoria, e defina as funções e parâmetros.

R:

Para se ter continuidade C2, deve-se usar curvas B-spline. Cada curva cobre 4 pontos de controle. Como a curva deve ser fechada, para, por exemplo, 4 pontos de controle, deve-se ter 4 patches.

array[N] //sao os N pontos de controle

//faz incremento circular dos índices para ao final da curva pegar novamente os pontos de controle iniciais. Para o primeiro patch: 0,1,2,3 para o segundo: 1,2,3,0, e assim por diante.

função inc(int i)

return i%N

fim

funcao avaliaBspline(i, s) //indice do vetor e variavel paramétrica s = [0,1]

retorna computaBsline(array[inc(i)], array[inc(i+1)], array[inc(i+2)], array[inc(i+3)], s )

fim

funcao desenhaCurva()

p1 = avaliaBspline(0, 0) //primeiro ponto da curva

for( i = 0, i < N; i++) //percorre cada patch ateh fechar toda curva

{

for(s = 0; s<=1; s+=0.01) //avalia cada patch

{

p2 = avaliaBspline(i, s)

line(p1, p2) //liga ponto antigo com o ponto novo

p1 = p2

}

}

fim