Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AP1 - 1° semestre de 2018.

Nome –

Assinatura –

Observações:

i) Prova sem consulta e sem uso de máquina de calcular.
ii) Use caneta para preencher o seu nome e assinar nas folhas de questões e nas folhas de respostas.
iii) Você pode usar lápis para responder as questões.
iv) Ao final da prova devolva as folhas de questões e as de respostas.
v) Todas as respostas devem ser transcritas nas folhas de respostas. As respostas nas folhas de questões não serão corrigidas.

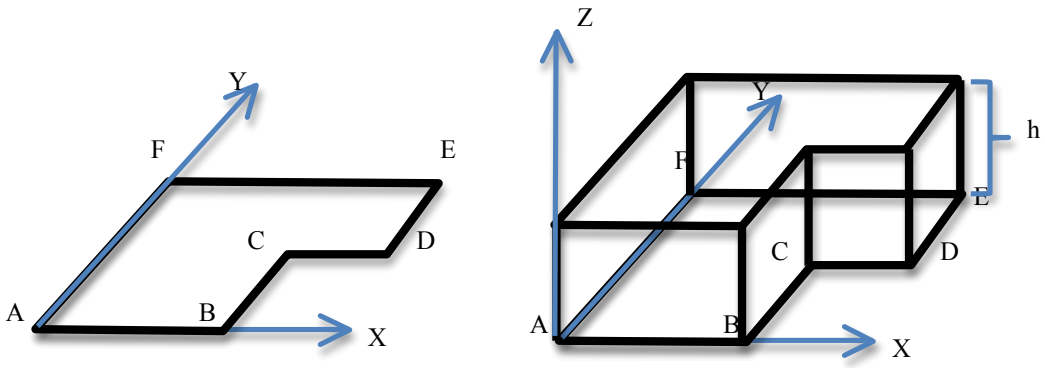
Na última página encontra-se a folha de respostas. Preencha corretamente e sem rasuras. Todas as questões tem o mesmo peso.

1) Descreva as principais diferenças e os principais pontos em comum entre as subáreas de Processamento de Imagens e Visão Computacional (2.5 pontos)

A área de Processamento de Imagens tem como intuito investigar e resolver problemas que recebem uma imagem digital como parte principal de sua entrada de dados e geram como saída uma outra imagem, em geral com diferentes características que podem auxiliar na solução de algum problema. Exemplos de problemas tratados em processamento de imagens são: realce de imagens, segmentação, detecção de arestas e quinas (*corners*), suavização, compressão de dados, dentre outros. Já a Visão Computacional lida com problemas que envolvem simular, através de sistemas computacionais, processos que ocorrem em sistemas visuais, como o sistema visual humano, a partir de uma imagem. Os métodos propostos em Visão Computacional podem ou não ser inspirados por sistemas visuais biológicos. Exemplos de problemas tratados em Visão Computacional são: reconhecer formas, interpretar uma forma reconhecida, extraindo uma informação não visual (por exemplo, reconhecer as letras e dígitos de uma placa de automóvel), calcular a disparidade entre pares de imagem estereográficas, reconstruir objetos tridimensionais a partir de imagens e detectar formas simples como retas, círculos e elipses em imagens.

Ambas as áreas tem como objeto de entrada uma imagem e usam muitas ferramentas de processamento de sinais para resolver os problemas de interesse.

2) Considere uma planta baixa simplificada correspondendo a um único cômodo, delimitado por uma curva poligonal fechada *p* com vértices A, B, C, D, E e F, conforme a figura à esquerda abaixo. Escreva uma algoritmo que tome como entrada a curva poligonal *p* e gere um sólido delimitado pelas 8 faces poligonais na figura à direita. O algoritmo deve gerar uma lista de faces, onde cada face é representada pelas coordenadas 3D dos seus vértices. Considere que as coordenadas 2D de um vértice, por exemplo o vértice A, sejam denotadas por (A.x, A.y) e suas coordenadas 3D por (A.x, A.y, A.z) (2.5 pontos).



Algoritmo. Constrói modelo 3d a partir de planta baixa

Entrada: *p* – curva poligonal fechada na forma de um array de vértices
n – número de vértices de *p*

Saida: l - lista de faces do modelo 3d

```
l ← {}

teto ←{} // cria a face superior (uma lista de vértices)
chao ←{} // cria a face inferior (uma lista de vértices)

Para i=0 até n-1 faça
   inserir_vertice(chao, p[i].x, p[i].y, 0)
   inserir_vertice(teto, p[i].x, p[i].y, h)
Fim_para

inserir_face(l, chao)
inserir_face(l, teto)

Para i = 0 até n-1 faça
        parede ← {} // cria uma nova parede (uma lista de vértices)
        inserir_vertice(parede, p[i].x, p[i].y, 0)
        inserir_vertice(parede, p[(i+1) % n].x, p[(i+1) % n ].y, 0)
        inserir_vertice(parede, p[(i+1) % n].x, p[(i+1) % n].y, h)
        inserir_vertice(parede, p[i].x, p[i].y, h)
        inserir_face( l, parede)
Fim_para

Retorne total

Fim_Algoritmo
```

3) Considere um pedaço de superfície definida por um paraboloide elíptico dado pela função $z = x^2 + y^2$, tomando valores na região $U = \{(x,y) \mid -1 \leq x \leq 1, -1 \leq y \leq 1\}$ do plano. Descreva um método que gere uma triangulação da superfície usando um reticulado regular definido em U com 10x10 pontos (2.5 pontos).

Um método para gerar uma triangulação da superfície consiste dos seguintes passos:

a) Gerar um reticulado no domínio $U = \{(x,y) \mid -1 \leq x \leq 1, -1 \leq y \leq 1\}$, dado pelo produto cartesiano de uma partição dos intervalos $-1 \leq x \leq 1$, $-1 \leq y \leq 1$ considerando valores Δx e Δy para definição do intervalo de amostragem.

b) Definir uma triangulação em U, adicionando um aresta diagonal a cada uma das células quadriláteras do reticulado, gerando assim 2 triângulos por célula.

c) Aplicar a função para obter as coordenadas z de cada ponto do reticulado transportando a triangulação no domínio U para o R^3, deste modo definindo a triangulação da superfície.

4) Considere uma região do espaço $U = \{(x,y,z) \mid -1 \leq x \leq 1, -1 \leq y \leq 1, -1 \leq z \leq 1\}$. Considere ainda um reticulado formado por 10x10x10 voxels em U. Escreva um algoritmo que determine o total de células do reticulado cujos centroides estejam dentro de uma esfera de raio unitário centrada na origem O = (0,0,0) (2.5 pontos).

Algoritmo. Conta células interiores a bola unitária

Entrada: *dx* = 10, *dy* = 10, *dz* =10, dimensões do reticulado
a = -1, *b* =1 limites do intervalo em *x*
c = -1, *d* =1 limites do intervalo em *y*
e = -1, *f* =1 limites do intervalo em *z*

Saída: *total* – total de células interiores

```
  total ← 0
  delta_x ← (b-a)/dx
  delta_y ← (d-c)/dy
  delta_z ← (f-e)/dz

  Para i=0 até dx-1 faça
    Para j=0 até dy-1 faça
      Para k=0 até dz-1 faça

        cx ←  a + i*delta_x + delta_x/2
        cy ←  c + j*delta_y + delta_y/2
        cz ←  e + k*delta_z + delta_z/2

        Se (cx*cx+cy*cy+cz*cz<1.0) então
          total ← total +1
        Fim_se

      Fim_para
    Fim_para
  Fim_para
Retorne total
Fim_Algoritmo
```