Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
Gabarito AD1 - 1° semestre de 2009.

1) Descreva as quatro principais sub-áreas da Computação Gráfica. Cite exemplos de problemas que são estudados por cada uma delas (1.0 ponto).

Síntese de imagens – Trata da geração de imagens a partir de um conjunto de dados e modelos. Os principais problemas estudados estão relacionados à produção de imagens realistas e visualização de dados, fenômenos e processos, em muitos casos de forma interativa e, até mesmo, em tempo real. A síntese de imagens se propõe a investigar diversos métodos, algoritmos e esquemas de representação e manipulação dos dados, de forma a solucionar tais problemas de modo eficiente e econômico. Nos estágios iniciais, a síntese de imagens introduziu os principais modelos e técnicas para geração de imagens 3D em dispositivos ***raster,*** *tais como, estruturas de dados para representação de objetos gráficos 2D e 3D, projeções e modelos de câmera, algoritmos para rasterização de polígonos e recorte, remoção de superfícies escondidas e iluminação direta, técnicas para interação e geração de curvas e superfícies. Em uma fase posterior, algoritmos de iluminação mais sofisticados como Raytracing e Radiosidade e técnicas de mapeamento de textura foram propostos. Atualmente são investigadas técnicas capazes de gerar imagens cada vez mais realistas, utilizando modelos sofisticados com o auxílio do avanço tecnológico das placas e processadores gráficos, assim como dos dispositivos de captura e visualização. A Síntese de Imagens tem grande aplicação na indústria, nas engenharias, nas diversas áreas da ciência, arquitetura, indústria do entretenimento, medicina e etc.*

Processamento de imagens – É a subárea responsável por estudar técnicas para representar, manipular e realizar operações sobre imagens digitais. Ao processar uma imagem digital as técnicas de processamento de imagens produzem outra imagem onde determinadas características são realçadas ou modificadas de forma a facilitar a realização de diferentes processos que utilizam as informações codificadas nas imagens. Cita-se como exemplo a aplicação de filtros para remover ou minimizar ruídos em uma imagem digitalizada por um scanner. A subárea de processamento de imagens considera imagens, em muitos casos, como um tipo particular de sinal, havendo desta forma uma considerável interseção com a área de processamento de sinais. Problemas comumente tratados pela área são os problemas de realce de características de imagens, segmentação, transformações geométricas aplicadas a imagens, composição, além de técnicas para armazenamento e transmissão. A área de processamento de imagens

possui inúmeras aplicações, como nos próprios processos de síntese de imagens, na ciência dos materiais, astronomia, geografia, microscopia, aerofotogrametria e etc.

Análise de imagens – Trata da aquisição de informação a partir de uma imagem digital, aquisição esta muita das vezes baseada em reconhecimento de padrões e nas características dos sistemas de formação de imagens. Temos como exemplos de aplicações a identificação de placas de automóveis, a identificação de áreas desmatadas, detecção de tumores em dados médicos, calibração automática de câmeras, determinação da estrutura tridimensional de objetos a partir de imagens e outras.

Modelagem – A modelagem, tipicamente denominada modelagem geométrica lida com problemas que envolvem a representação, geração e manipulação de formas como curvas, superfícies e sólidos em sistemas computacionais. Problemas normalmente estudados envolvem representação de formas por subdivisão, representações por multiresolução, simplificação de malhas, modelagem a partir de imagens e etc. As técnicas provenientes da área de modelagem têm inúmeras aplicações na indústria, em problemas de física e matemática, engenharias, projeto e manufatura auxiliados por computador (CAD e CAM, respectivamente) e etc.

2) Descreva alguns dos componentes de uma ferramenta de modelagem (1.0 ponto).

Dentre os diversos componentes em uma ferramenta de modelagem podemos citar como exemplos:

Componentes para modelagem baseada em:
- *Polígonos;*
- *Imagens.*

Editores de shaders.
Texturização.
Otimização de modelos baseada em:
- *Hierarquias de níveis de detalhe (LODs);*
- *Renderização em texturas;*
- *Redução de polígonos.*

Animação:
- *Cinemática inversa,*
- *Rigging,*
- *Skinning.*

Componentes para conversão entre diferentes formatos de modelos 3D.

3) Dê exemplos de funções de atributos de objetos gráficos bidimensionais (1.0 ponto).

Objetos gráficos bidimensionais são aqueles cuja dimensão do seu suporte geométrico é igual a dois. Exemplos de objetos gráficos de dimensão dois de interesse para a computação gráfica são:

- *Regiões do plano – subconjuntos do plano dados por regiões internas delimitadas por uma curva de Jordan, juntamente com um algoritmo de classificação ponto-conjunto;*

- *Superfícies topológicas – subconjunto S do espaço euclidiano R^3 que localmente é homeomorfo ao plano euclidiano R^2, o que intuitivamente significa que ela pode ser obtida colando-se pedaços deformados do plano.*

Como exemplo de função de atributos para uma região do plano temos uma função f que associa a cada um dos seus pontos uma cor: $f:U \subset R^2 \rightarrow R^3$.

Como exemplo de função de atributos para uma superfície paramétrica S destacamos uma função que associa a cada ponto de S o vetor normal: f: $U \subset R^2 \rightarrow R^3$, onde $f(u,v) = \vec{n} = \frac{\partial f}{\partial u} \times \frac{\partial f}{\partial v}$, onde $\frac{\partial f}{\partial u}$ e $\frac{\partial f}{\partial v}$ são as derivadas parciais de *f* nos parâmetros *u* e *v*.

4) Descreva um algoritmo capaz de verificar se uma curva poligonal possui auto-interseção e indique sua complexidade (1.0 ponto).

Um algoritmo força bruta com complexidade $O(n^2)$, onde n é o número de segmentos da curva poligonal, pode ser obtido através do seguinte conjunto de passos:

s – array com os segmentos da curva

```
Função DetectarSimplicidade(s):lógico
Para i:←1 até n faça
  Para j:←i+1 até n faça
        Se Intersectar(s[i],s[j]) então
           Retornar verdadeiro
        Fim_Se
  Fim_Para
Fim_para
Retornar falso;
```

*Um algoritmo mais eficiente, como complexidade O(n log n), onde n é o número de segmentos da curva poligonal, pode ser obtido através da ordenação lexicográfica (ordenação considerando as coordenadas x e y dos pontos) extremos do segmento, seguido da técnica de **varredura do plano**, a qual utiliza uma estrutura de dados L, do*

tipo dicionário, que armazena o status de linhas de varredura e uma estrutura que armazena eventos E, que no caso é apenas um array que armazena os pontos extremos dos segmentos.

A estrutura L armazena os segmentos conforme a relação $>_x$. Dados dois segmentos s_1 e s_2, comparáveis em uma abscissa x, isto é, se existe uma linha vertical passando pela abscissa x que intersecta tanto s_1 e s_2, dizemos que $s_1 >_x s_2$ se a interseção de s_1 com x for maior que a interseção de s_2 com x.

A relação $>_x$ define uma ordem total e somente muda nos seguintes casos:

- *O extremo esquerdo de um segmento s é encontrado e nesse caso s deve ser inserido em L.*
- *O extremo direito de um segmento s é encontrado e nesse caso s deve ser removido de L por não mais ser comparável com os demais segmentos.*
- *Um ponto de interseção entre dois segmentos s_1 e s_2 é alcançado e nesse caso s_1 e s_2 mudam de ordem na ordenação.*

O algoritmo apresentado abaixo, devido a Bentley e Ottmann, considera apenas o problema de determinar a existência de uma interseção ou não, isto é um problema de detecção. A estrutura de dados L suporta as operações:

- *Inserir(L,s) – insere em tempo logaritmo um segmento s.*
- *Deletar (L,s) – remove em tempo logaritmo um segmento s.*
- *Acima(s, L) – retorna em tempo constante o segmento acima de s em L.*
- *Abaixo(s,L) – retorna em tempo constante o segmento abaixo de s em L.*

```
s – array com os n segmentos da curva
pontos – array com os m pontos dos segmentos da linha poligonal

Função DetectarSimplicidadeVarreduraDoPlano(s):lógico
pontos – array com os m pontos dos segmentos da linha poligonal
OrdenarLexicograficamente(s,pontos) {ordenar os pontos dos segmentos em função de x e y e armazenar no array pontos}
Para i← 1 até m faça;
  p ←pontos[i];
  s ← semento em que p é um ponto extremo;
        Se (p é um ponto extremo esquerdo) então
                Inserir(L,s)
                s1←Acima(L,s)
                s2←Abaixo(L,s)
                Se Intersectar(s1,s) então
                        Retornar verdadeiro
                Fim_Se
                Se Intersectar(s2,s) então
                        Retornar verdadeiro
                Fim_Se
        Senão {p é um ponto extremo direito}
                s1←Acima(L,s)
                s2←Abaixo(L,s)
                Deletar(L,s)
                Se Intersectar(s1,s2) então
                        Retornar verdadeiro
                Fim_Se
        Fim_Se
  Fim_Para
Retornar falso;
```

O algoritmo somente retorna uma interseção se ela é encontrada e nunca retorna uma interseção inexistente. Também não deixa de apresentar pelo menos uma interseção quando interseções existem. Para isto basta mostrar que ele sempre retorna a interseção mais a esquerda qL (a primeira da esquerda para direita). Suponha que qL coincida com um ponto extremo esquerdo p de um segmento, então, neste caso, tal interseção é encontrada no processamento de p. Suponha agora o caso em que qL não seja um ponto extremo esquerdo p de um segmento. Neste caso, a estrutura L mantém a ordenação $>_x$, correta a esquerda de qL. Além disso, existe um pequeno intervalo de abscissas à esquerda de qL em que dois segmentos que se intersectam em qL encontram-se adjacente em L, o que faz com que qL acabe sendo descoberta em algum passo do algoritmo.

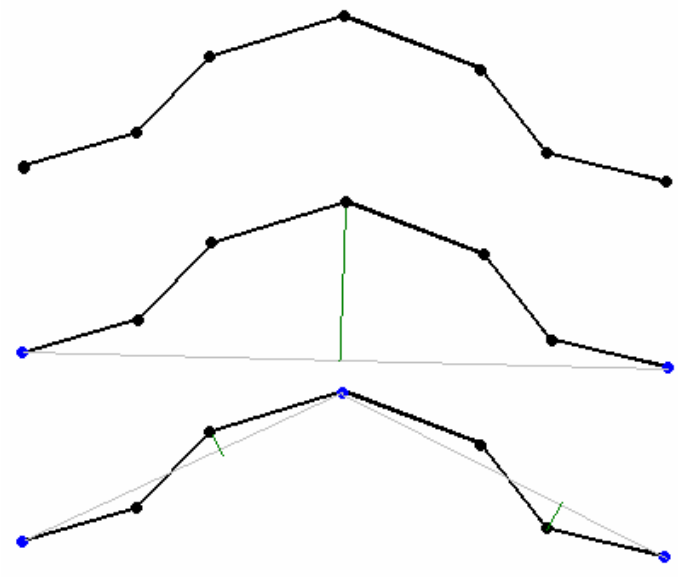
Como as seleções requerem tempo logarítmico O(log n) no pior caso, a complexidade da detecção das interseções é O(m log n) que é igual a O(n log n), já que o número de pontos m da curva poligonal é 2n-1, onde n é o número de segmentos. Tal custo coincide com o custo da ordenação o que indica que a complexidade total é O(n log n). Observe que somente O(n) testes de interseção são realizados o que pode fazer com que algumas interseções não sejam detectadas, entretanto isto não importa, já que o algoritmo sempre detecta a existência de pelo menos uma interseção, caso ela exista.

5) Faça uma pesquisa sobre o algoritmo Douglas-Peucker para simplificação de curvas poligonais. Descreva algumas de suas aplicações (1.0 ponto).

*O algoritmo de simplificação de curvas de **Douglas-Peucker** se baseia no processo de aproximação dos extremos da curva considerada, por um segmento de reta unindo seus primeiro e último pontos e um fator de tolerância que controla a validade da aproximação.*

O algoritmo, inicialmente, tem como entrada uma seqüência (P1, P2, .., Pn) de pontos da curva original, e cria um segmento ligando o primeiro P_1 ao último P_n ponto. Em seguida, é computado o ponto P_k da curva mais distante deste segmento. Se a distância de P_k for inferior ou igual à tolerância, o algoritmo retorna o par (P_1,P_n) de pontos que compõem a simplificação.

Se a distância de P_k for superior à tolerância, o algoritmo retorna a composição das simplificações obtidas pelas chamadas recursivas ao algoritmo, passando as entradas $(P_1,...,P_{k-1})$ e $(P_k,...,P_n)$. A ilustração abaixo demonstra o processo de simplificação, onde na primeira chamada é computado o ponto mais distante ao segmento. Como sua distância supera a tolerância, o algoritmo é re-executado para os segmentos restantes, que finalmente podem ser aproximados por segmentos, segundo a tolerância.



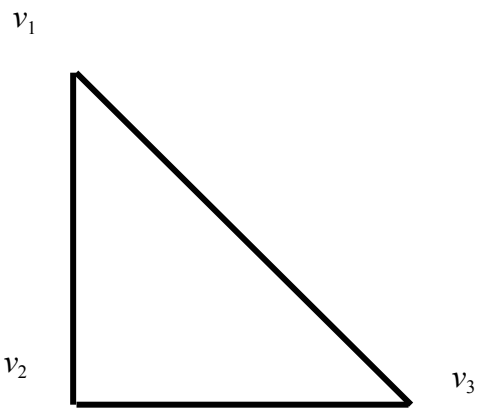
Simplificação de curva por Douglas-Peucker

6) Descreva um método para poligonização de curvas representadas de forma implícita (1.0 ponto).

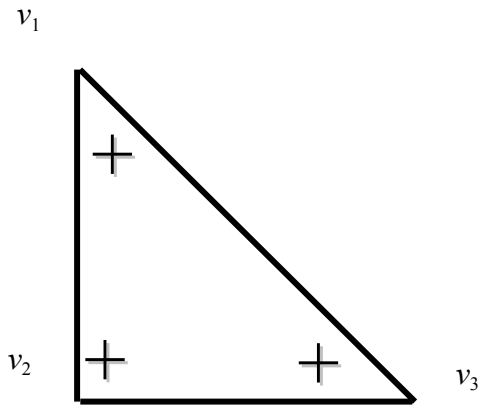
Solução:

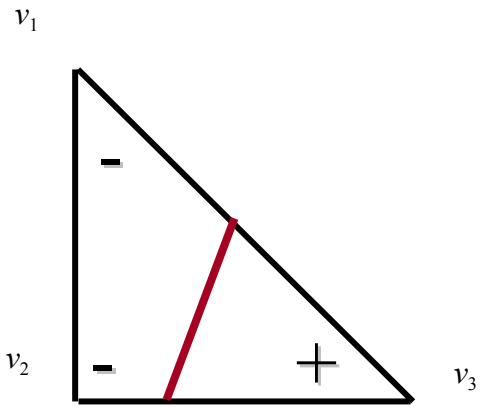
i. Determinar uma *triangulação do domínio* de F.

ii. *Aproximar F* em cada triângulo por uma *função linear F'*.

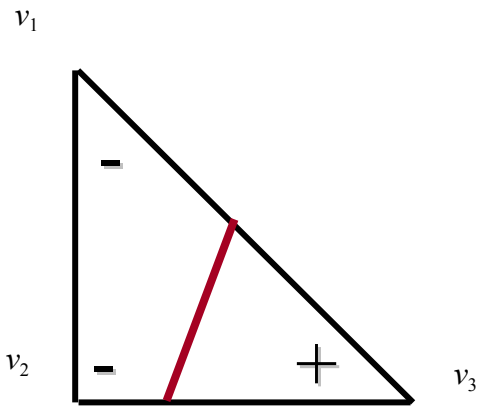


iii. Verificar se há mudança de sinais do valor da função dos pontos avaliados. Se não houver não fazer nada.

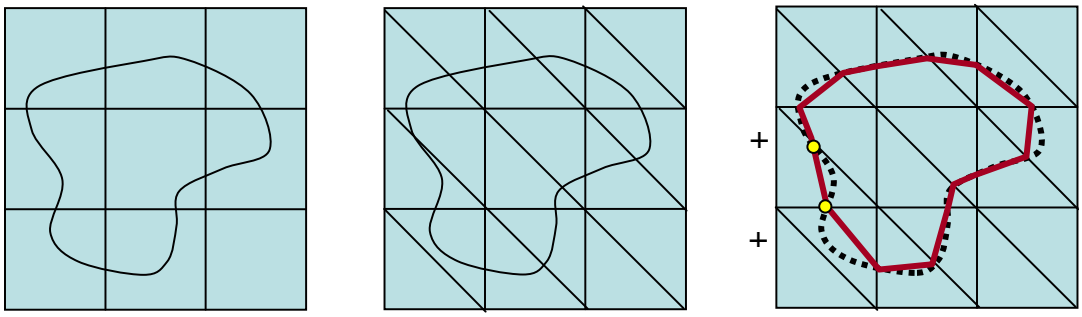




iv. Solucionar F’(x,y)=0 em cada triângulo. A solução é em geral um segmento de reta.



v. A estruturação das amostras é induzida pela estrutura da triangulação subjacente.



7) Faça uma pesquisa sobre um algoritmo capaz de triangularizar uma região do plano definida por uma curva simples fechada (1.0 ponto).

Um modo simples de triangularizar um polígono se baseia no fato de que todo polígono sem buracos possui duas "orelhas". Uma orelha é um triângulo cujos dois lados pertencem às arestas do polígono e cujo terceiro está completamente contido nele. O algoritmo consiste em encontrar uma orelha, removê-la do polígono, gerando um novo que satisfaz a mesma propriedade e repetir o processo até que reste somente um triângulo.

8) Como objetos volumétricos podem ser representados? Descreva as vantagens e desvantagens de cada tipo de representação (1.0 ponto).

Objetos volumétricos podem ser representados de dois modos:

Representação por bordo ou B-rep (boundary representation): baseada no teorema de Jordan, que afirma que um objeto volumétrico pode ser definido por uma superfície fechada e limitada (uma cápsula envolvente) mais um algoritmo para solução do problema de classificação ponto-conjunto.

Representação por decomposição: tipo de representação em que o espaço envolvente é subdivido em um conjunto de células e o objeto é descrito através da enumeração das células que o intersectam. Pode-se adotar uma decomposição uniforme, por exemplo, através de uma matriz ou então uma decomposição não uniforme, por exemplo, uma octree.

9) Faça uma pesquisa sobre o algoritmo de deCastlejau para geração de curvas de Bérzier (1.0 ponto).

O algoritmo de deCastlejau é um algoritmo utilizado para calcular um ponto sobre uma curva de Bérzier correspondente a um valor de parâmetro t = t0, 0≤t0≤1, baseado em aplicações repetidas de interpolações lineares. Considere, por exemplo, uma curva de Bézier de grau n=3 contendo os pontos P0,P1,P2 e P3. A parametrização do segmento P0P1 é dada por

(1 - t)P0 + tP1,

Para um dado valor de t=t0 podemos calcular um novo ponto de controle sobre P0P1 da seguinte forma:

Q0 = (1 - t0)P0 + t0P1.

Podemos aplicar o mesmo processo calculando para os segmentos P1P2 e P2P3 os pontos :

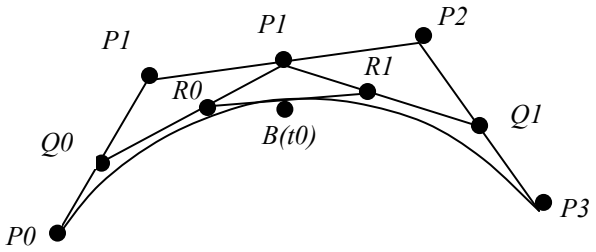
Q1 = (1 - t0)P1 + t0P2 e Q2 = (1 - t0)P2 + t0P3.

Em seguida, obtemos os pontos R0 e R1, interpolando para t=t0, respectivamente, os extremos dos segmentos Q0Q1 e Q1Q, conforme abaixo:

R0 = (1 - t0)Q0 + t0Q1 and R1 = (1 - t0)Q1 + t0Q2.

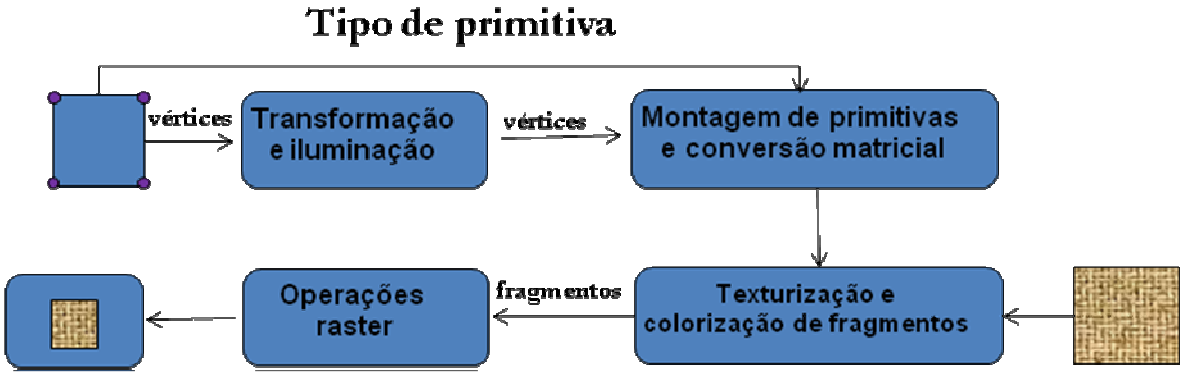
Finalmente, o ponto B(t0) na curva é obtido interpolando-se R0 e R1

B(t0) = (1 - t0)R0 + t0R1



10) Descreva os principais componentes do pipeline gráfico da OpenGL(1.0 ponto)?

Uma versão simplificada do pipeline gráfico do OpenGL, considerando a alimentação de primitivas vetoriais (formadas por vértices) pode ser delineada através do diagrama abaixo.



No primeiro estágio são aplicadas as transformações geométricas que convertem as coordenadas do objeto para coordenadas da câmera e aplica-se a projeção especificada. Isto corresponde à transformação do vértice por meio de sua multiplicação pelas matrizes modelview e projection. Neste mesmo estágio é feito o cálculo de iluminação por vértice.

No segundo estágio é montada a primitiva conforme especificada pelo usuário. O estágio anterior não dispõe de tal informação. Em seguida é feita a conversão matricial, onde são determinadas as coordenadas dos fragmentos que compõem a primitiva. Neste estágio são determinados os valores (cor, coordenada de textura, profundidade de cada fragmento através de interpolação).

No terceiro estágio as informações calculadas para cada fragmento podem ser modificadas em função do mapeamento de textura e aplicação de neblina (fog).

Finalmente as operações raster determinam se um fragmento efetivamente chegará à tela ou não, sendo aplicados testes de stencil, alpha e profundidade.