Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AD1 - 2° semestre de 2010.

1) A qual categoria de objeto gráfico pertencem os terreno utilizados em jogos e simuladores de vôo? Explique sua afirmação. (1.0 ponto).

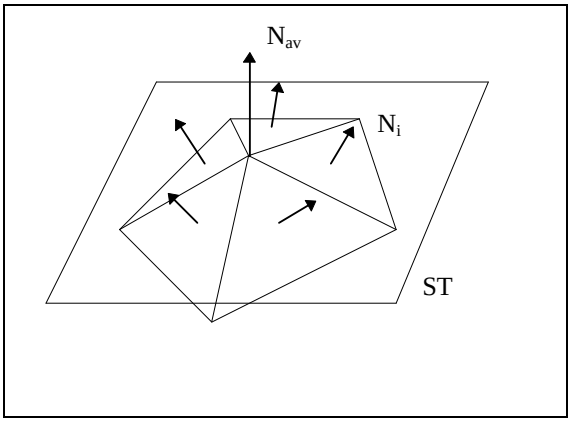
   Terrenos são tipicamente representados através da triangulação de um conjunto de vértices representando pontos sobre uma superfície que, por sua vez corresponde a um mapa de elevação. Logo, são objetos espaciais de dimensão dois (superfícies), já que não possuem área. Mais formalmente, a interseção de uma bola de raio $\varepsilon>0$ centrada em qualquer ponto que pertence a um terreno com o próprio terreno produz um disco cuja topologia é a de um pedaço do plano. Observe que tal afirmação só é válida para terrenos que podem ser expressos com uma função $f(x,y)$. Terrenos contendo cavernas e outras estruturas que não possuem a topologia do plano não podem ser descritas por uma superfície.

2) Descreva uma estrutura de dados capaz de representar dados de terreno (1.0 ponto).

   Terrenos são normalmente descritos por triangulações que, por sua vez, são expressas através de estruturas que representam o grafo induzido pelos vértices e arestas da triangulação, juntamente com o seu grafo dual. Uma estrutura de dados bem simples pode ser obtida armazenando uma lista de triângulos onde cada triângulo faz referência aos seus vértices. Podemos tornar tal estrutura mais eficiente criando uma lista de vértices, uma lista de arestas e uma lista de faces. Cada face (triângulo) da lista de faces referencia arestas na lista de arestas e cada aresta desta última, por sua vez, referencia um par de vértices na lista de vértices. Estruturas mais sofisticadas como a winged-edge e half-edge se baseiam em construções similares. Para uso de nível de detalhe (level of detail) em terrenos podem ser utilizadas estruturas mais sofisticadas para agrupar representações do terreno em diferentes níveis de detalhe como, por exemplo, as estruturas utilizadas nos algoritmos ROAM (real-time optimally adapting meshes).

3) Um terreno pode conter muitos triângulos. Descreva uma estratégia para reduzir o número de triângulos em um modelo sem aumentar consideravelmente o erro geométrico introduzido pela simplificação. Dica: pense em como remover vértices que não são importantes da triangulação. Considere uma função dos ângulos entre a normal de um vértice v e as normais dos vértices adjacentes a v como uma medida de sua importância q(v) (1.0 ponto).

Um método de dizimação(decimação) bastante elegante foi proposto no trabalho de Florian Shröder e Patrick Roβbach (Managing the complexity of digital terrain models, Comput. & Graphics,Vol. 18, No.6, pp. 775-783,1994) *para a simplificação de terrenos e sólidos geométricos em geral. O algoritmo descrito procura remover a cada passo o ponto que menos contribui para os detalhes de aproximação da superfície. O critério de seleção do ponto menos importante é baseado no grau de rugosidade em relação aos seus triângulos vizinhos. Uma superfície tangente ST é ajustada sobre um vértice* ***p*** *que é examinado conforme a figura abaixo:*



A orientação N_{av} de ST é dada pela média das normais n_i dos triângulos que envolvem p ponderados pela sua área A_i:

$$nav = \frac{\sum_{i=1}^{tno} \vec{n}_i \,.\, A_i}{\sum_{i=1}^{tno} A_i}$$

nav: normal média que determina a orientação da superfície tangente ST em p.

tno: número de triângulos que circundam o vértice examinado p.

$\vec{n}_i$ *: superfície normal do triângulo i.*

A_i*: área do triângulo i.*

Calcula-se então o ângulo máximo a_{max} entre a normal média nav e as normais das superfícies definidas pelos triângulos que circundam o ponto p.

$$a_{\max} = \max_{i=1}^{tno}\left(\arccos\frac{\vec{n}_{av}.\vec{n}_{i}}{|\vec{n}_{av}|.|\vec{n}_{i}|}\right)$$

O resultado em a_{max} representa o maior ângulo entre nav e n_i e assume valores entre 0 e π. Se n_{av} e todas as outras normais forem normalizadas, podemos então usar a seguinte expressão:

$$a_{\max} = \max_{i=1}^{tno}\left(\arccos\left(\vec{n}_{av}.\vec{n}_{i}\right)\right)$$

Pelo critério de qualidade, o algoritmo procurará remover todos os pontos cujo valor a_{max} é menor que o valor limite estabelecido AngMax. Se o objetivo é remover certa quantidade de pontos, então deve-se ordenar previamente todos os pontos por a_{max} em ordem crescente de forma a manter os mais significativos.

O autor deste trabalho propõe três técnicas para retriangular o conjunto de pontos resultante do processo de decimação:

***Triangulação de Delaunay**: retriangula-se segundo o critério de Delaunay todos os pontos que sobrevivem ao processo.*

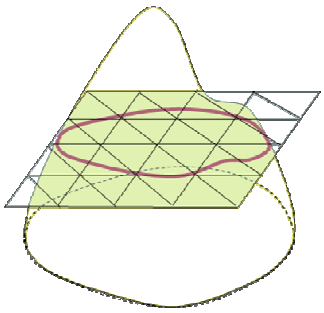
***Small Hole approach**: a cada ponto removido, utiliza-se um algoritmo de triangulação simples para retriangular os "buracos" gerados.*

***Big Hole approach**: permite-se que os pontos sejam removidos e ao final do processo, retriangula-se os buracos restantes.*

4) Descreva um método para converter a representação implícita de uma curva em uma curva poligonal (1.0 ponto).

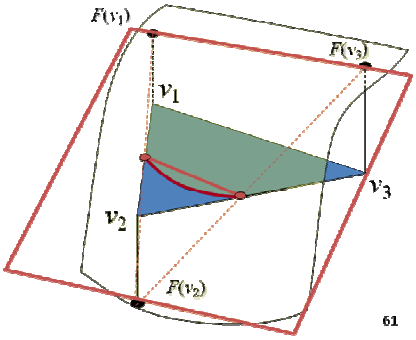
A dificuldade para poligonizar uma curva implícita se deve ao fato de que a solução de uma equação implícita determina um conjunto de pontos não-estruturados. Entretanto, é possível introduzir uma estruturação em tal conjunto através de uma triangulação do domínio da função. Com efeito, para gerar uma curva poligonal podemos aplicar o seguinte método:

1. Determinar uma triangulação do domínio de F.

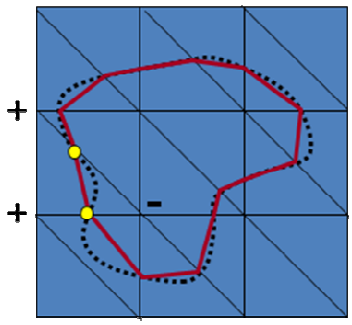


2. *Aproximar F em cada triângulo por uma função linear F'.*

3. Solucionar F'(x,y)=0 em cada triângulo onde houver diferença de sinais entre os valores da função nos vértices. A solução é, em geral, um segmento de reta.



4. Cada segmento gerado é conectado aos segmentos nas células adjacentes, de acordo com a triangulação definida sobre o reticulado.



5) Descreva o algoritmo de *De Casteljau* para construção de uma curva de *Bézier* cúbica (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 :

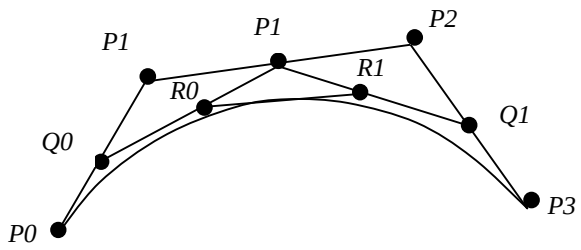
$Q1 = (1 - t0)P1 + t0P2$ *e* $Q2 = (1 - t0)P2 + t0P3.$

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

$R0 = (1 - t0)Q0 + t0Q1$ *e* $R1 = (1 - t0)Q1 + t0Q2.$

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

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



6) Descreva as principais diferenças entre uma curva de *Bézier* e uma curva *B-Spline* (1.0 ponto).

As B-Splines, diferentemente das curvas de Bézier, são formadas por segmentos de curva. As B-splines permitem criar curvas com muitos pontos de controle, sem a necessidade de se aumentar o grau do polinômio da base ou então colar diferentes curvas de menor grau utilizando, adicionalmente, um mecanismo que garanta a continuidade e suavidade entre os segmentos nos pontos de junção. Isso se deve ao fato de que, por definição, B-splines descrevem curvas suaves por partes, sendo que a suavidade é garantida automaticamente através do compartilhamento de pontos de controle entre segmentos de curva consecutivos que compõe a curva maior.

Nas B-Splines o controle local ocorre em um grau bem maior que nas Curvas de Bézier. Isto se deve ao fato de que, nas B-Splines, os pontos de controle influenciam apenas um subconjunto dos segmentos que compõem a curva total, algo que não é possível de se obter através de curvas de Bézier, nas quais a modificação de um ponto de controle causa uma modificação em toda a curva.

7) Plote os nós de uma *B-Spline* cúbica determinada pelos seis pontos de controle (0.0,0.0), (0.3,0.8),(0.7,0.8),(1.0,0.0), (1.0,1.0),(0.0,1.0) (1.0 ponto).

A B-Spline cúbica uniforme com m+1= 6 pontos de controle contém m-2=3 segmentos. Sejam Q_3, Q_4 e Q_5 os três segmentos que formam a curva. Além disso, uma curva com m+1 = 6 pontos de controle contém m+5 = 10 valores de nó, a saber, U=0,1,2,4,...,9.

Sabendo que a expressão que define um segmento de B-Spline é dada por $Q_i(u) = \sum_{k=0}^{3} B_{i-3+k}(u)\, p_{i-3+k}$ *,0≤u≤1 e que as bases de B-Spline cúbicas são dadas por:*

$$
\begin{aligned}
B_i(u) &= 1/6u^3 \\
B_{i-1}(u) &= 1/6(-3u^3 + 3u^2 + 3u + 1) \\
B_{i-2}(u) &= 1/6(3u^3 - 6u^2 + 4) \\
B_{i-3}(u) &= 1/6(1-u^3)
\end{aligned}
\quad ,
$$

podemos calcular os pontos correspondentes aos valores de nó no parâmetro global U iguais a 4 e 5. O valor de nó U=4 é o valor correspondente ao nó de junção entre os segmentos Q_3 e Q_4. Respectivamente, o valor U=5 corresponde ao nó de junção entre os segmentos Q_4 e Q_5. Obtendo o valor de nó no parâmetro local u = U-i, 0≤u≤1, temos:

$$Q_3(4-3) = \sum_{k=0}^{3} B_{i-3+k}(1)\, p_{i-3+k} = B_0(1)\, p_0 + B_1(1)\, p_1 + B_2(1)\, p_2 + B_3(1)\, p_3 =$$

$$0.0 * (0.0,0.0) + 1/6 * (0.3,0.8) + 4/6 * (0.7,0.8) + 1/6 * (1.0,0.0) =$$

$$(0.0,0.0) + (0.05,0.13) + (0.46,0.53) + (0.17,0.0) = (0.68,0.66)$$

$$Q_4(5-4) = \sum_{k=0}^{3} B_{i-3+k}(1)\, p_{i-3+k} = B_1(1)\, p_1 + B_2(1)\, p_2 + B_3(1)\, p_3 + B_4(1)\, p_4 =$$

$$0.0 * (0.3,0.8) + 1/6 * (0.7,0.8) + 4/6 * (1.0,0.0) + 1/6 * (1.0,1.0) =$$

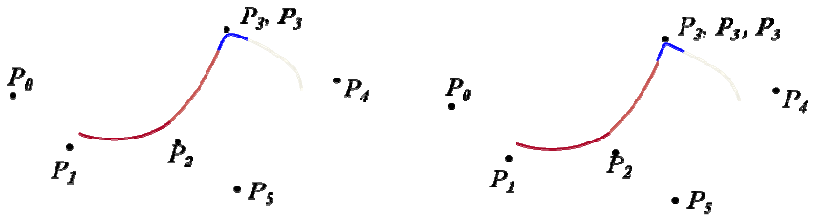
$$(0.0,0.0) + (0.12,0.13) + (0.67,0.0) + (0.17,0.17) = (0.96,0,3)$$

O exercício também pode ser resolvido utilizando a definição Cox-deBoor para obtenção das bases cúbicas da B-Spline.

Descreva duas estratégias para fazer com que uma *B-Spline* passe por um ponto de controle (1.0 ponto).

É possível fazer com que uma B-Spline interpole um ponto de controle através de dois esquemas:

- *Replicando pontos de controle, o que pode causar perdas de continuidade na curva*



- *Aumentando a multiplicidade dos nós, o que não causa perda de continuidade, mas que gera curvas B-Spline não uniformes, já que o intervalo no espaço de parâmetros entre os valores de nós não é mais o mesmo.*

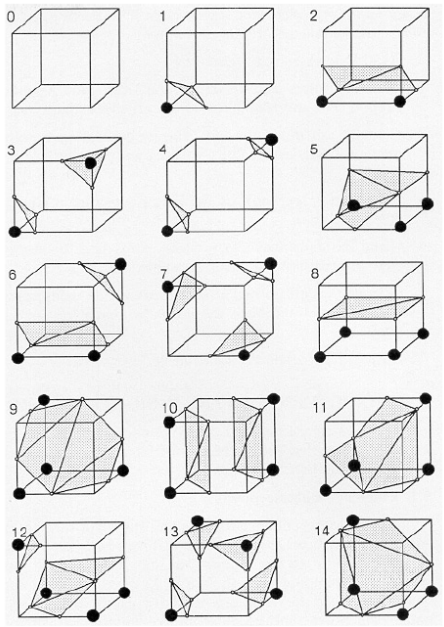
8) O que são *Quadtrees*? Descreva uma aplicação desta estrutura (1.0 ponto).

Quadtrees são estruturas de dados hierárquicas utilizadas para determinar a ocupação de uma região bi-dimensional. Uma Quadtree é construída subdividindo o plano sucessivamente em ambas as dimensões formando quatro quadrantes. Quando utilizada para representar uma área do plano, cada quadrante pode ser cheio, parcialmente cheio ou vazio, dependendo do quanto o quadrante intersecta a área. Um quadrante parcialmente cheio é subdividido em subquadrantes e o processo de subdivisão continua até que todos os quadrantes são homogêneos ou então uma resolução máxima foi alcançada. O processo de particionamento se reflete em uma estrutura de árvore quaternária.

9) Faça uma pesquisa sobre o algoritmo *Marching Cubes* (1.0 ponto).

*O **Marching Cubes** é um algoritmo porposto por Lorensen e Cline, que procura extrair uma superfície poligonal a partir modelos volumétricos. Uma superfície é construída ajustando-se um ou mais polígonos em cada voxel do modelo volumétrico que contenha parte da superfície. O algoritmo associa a cada voxel um número de 8 bits, onde cada bit associado a um único vértice, tem valor igual a 1, quando o vértice é interior a superfície, e zero quando ele é exterior. A configuração de polígonos no interior do voxel representando a superfície depende da configuração de valores de bits associada ao voxel. O algoritmo utiliza esse número para consultar uma tabela que mostrará a disposição dos vértices e faces geradas para cada configuração.*

Para saber se uma superfície passa por uma aresta, basta que exista um vértice da aresta que esteja dentro do volume e outro que esteja fora. As 256 possíveis combinações geradas pelos vértices geram uma tabela de configurações de faces. Estas se resumem a 15 variações mostradas na figura abaixo, porém com rotações diferentes:


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14