Fundação CECIERJ - Vice Presidência de Educação Superior a Distância

Curso de Tecnologia em Sistemas de Computação
Disciplina: Computação Gráfica
AD1 - 2° semestre de 2012.

1) Descreva uma aplicação da área de visão computacional em problemas encontrados na sociedade (1.0 ponto).

Uma aplicação de visão computacional é na construção de veículos autônomos. A visão computacional pode, neste caso, auxiliar um carro a se manter na pista, através da detecção das faixas e outros carros, ou então evitar obstáculos, analisando imagens obtidas por meio de uma câmera.
(ver http://cvrr.ucsd.edu/publications/2010/IV10_SSivaraman.pdf).

A maioria dos veículos autônomos, entretanto, utiliza técnicas de visão computacional em conjunto com GPS, radares e LIDAR, uma tecnologia que mede distâncias a um alvo iluminando-o com pulsos de laser.

2) Descreva como calcular a interseção entre 2 segmentos de reta usando a representação paramétrica da reta (1.0 ponto)

Sejam dois segmentos l_1 e l_2. O segmento l_1 conecta p_0 e p_1 e o segmento l_2 conecta p_3 e p_2. Pode-se então escrever as equações paramétricas das retas que passam por l_1 e l_2 como:

$$p_a(t_a) = p_0 + t_a(p_1 - p_0)$$
$$p_b(t_b) = p_2 + t_b(p_3 - p_2)$$

Fazendo $p_a(t_a) = p_b(t_b)$, tem-se duas equações em duas incógnitas t_a e t_b.

$$x_1 + t_a(x_1 - x_0) = x_2 + t_b(x_3 - x_2)$$
$$y_1 + t_a(y_1 - y_0) = y_2 + t_b(y_3 - y_2)$$

Resolvendo para t_a e t_b obtem-se:

Se os denominadores de t_a e t_b forem zero as retas suporte são paralelas, e se tanto os denominadores quanto os numeradores forem zero, as retas suporte são coincidentes.

Para checar se os segmentos se intersectam, basta verificar se t_a ou t_b estão no intervalo [0,1] e aplicar os t_a e t_b nas respectivas equações paramétricas para obter as coordenadas da interseção.

3) Explique as vantagens em se utilizar o algoritmo de ponto médio para o problema de rasterização de segmentos (1.0 ponto).

O algoritmo do ponto médio permite evitar cálculos com números em ponto flutuante, além de facilitar o uso de cálculos incrementais para determinação dos pixels a serem acesos.

4) Faça uma pesquisa sobre o algoritmo de recorte de *Weiler-Atherton* (2.0 pontos).

O algoritmo *Weiler-Atherton* é um algoritmo de recorte capaz de recortar um polígono côncavo, com buracos, em relação a borda de um outro polígono, com as mesmas características. O polígono a ser recortado é denominado *polígono alvo* (PA) enquanto a região de recorte é dada pelo *polígono de recorte* (PR).

Ambos os polígonos são descritos por uma lista circular de vértices. A borda exterior dos polígonos é especificada no sentido horário e as bordas interiores, isto é, os buracos, são especificadas no sentido anti-horário, fazendo com que o lado interno dos polígonos, esteja sempre à direita das bordas.

As interseções, quando existem, ocorrem aos pares: uma delas, quando uma aresta do polígono alvo penetra o interior do polígono de recorte, e a outra, quando uma aresta do polígono alvo sai do polígono de recorte.

Ideia geral do algoritmo:

1. Começamos com uma interseção de entrada.
2. Percorremos a borda externa do polígono alvo no sentido horário até encontrar uma nova interseção.
3. Neste momento, viramos para à direita e percorremos a borda externa do polígono de recorte no sentido horário, até encontrar uma nova interseção.
4. Neste ponto, retornamos ao polígono alvo e continuamos o processo até que o ponto de início seja alcançado.

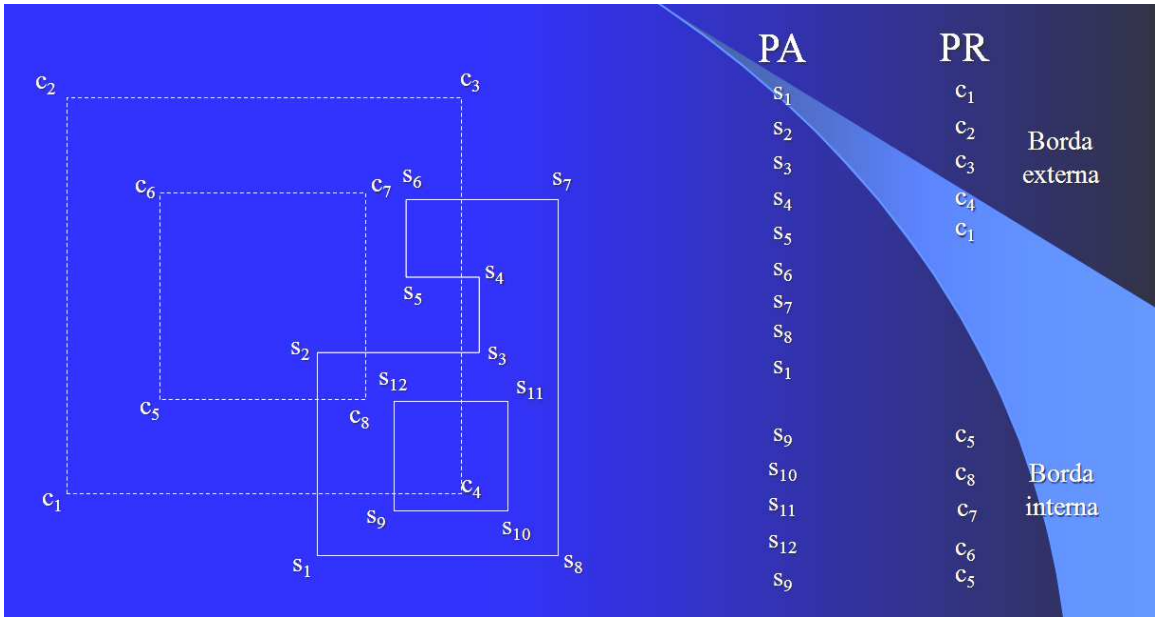
Obs: as bordas internas do polígono alvo são percorridas no sentido anti-horário.

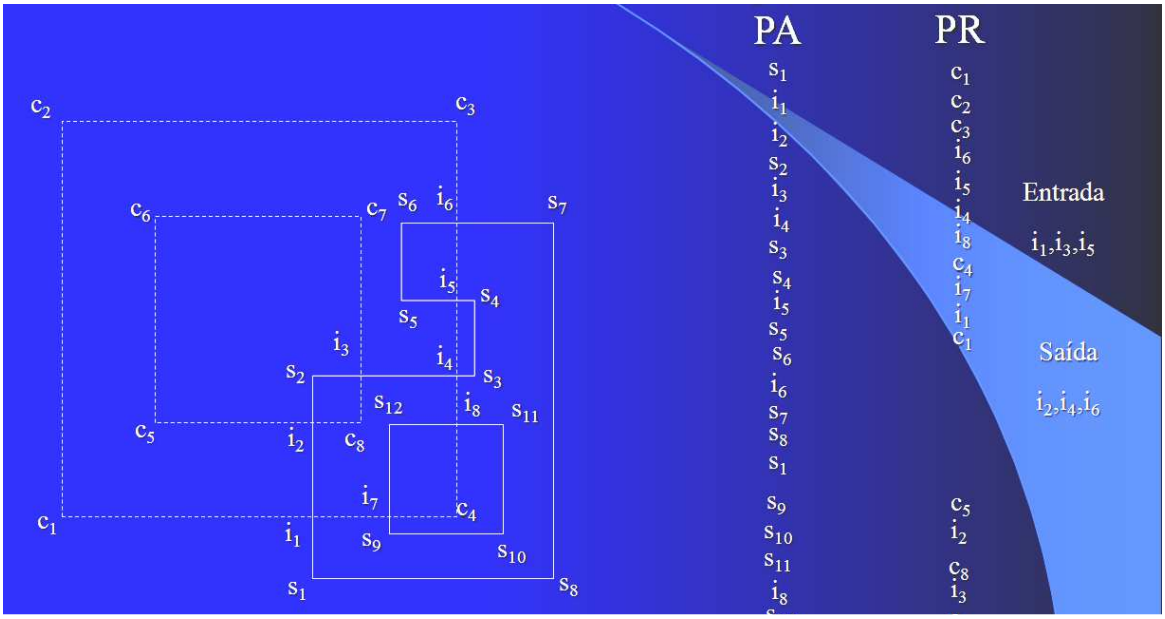
Inicialização do Algoritmo

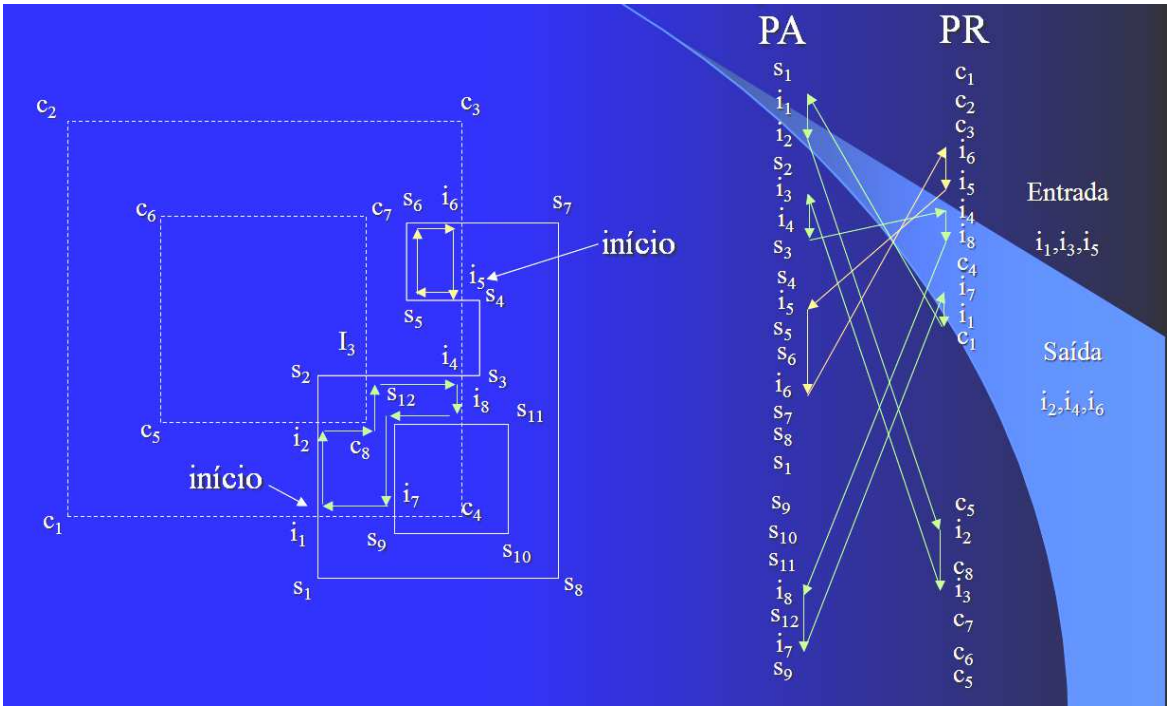
1. Determine as interseções entre os polígonos alvo e de recorte.
    a. Adicione cada interseção às listas de vértices de ambos os polígonos.
    b. Crie uma ligação bidirecional entre as duas listas de vértices para cada interseção.
2. Crie duas listas de armazenamento.
    a. Uma para as bordas que pertençam ao interior do polígono de recorte e a outra para as bordas que pertençam ao seu exterior.
    b. Bordas do polígono de recorte no interior do polígono alvo forma buracos. Neste caso, tais bordas pertenceram às duas listas.
3. Crie duas listas de interseções.
    a. Crie uma lista de interseções de entrada e uma outra de interseções de saída.
    b. Os tipos de interseção se alternam ao longo da borda dos polígonos.

Algoritmo de recorte

1. Remova um vértice da lista de interseções de entrada. Se não houver nenhum o processo terminou.
2. Siga a lista de vértices do polígono alvo até que uma interseção seja encontrada.
3. Neste momento, copie os vértices percorridos até o momento para a lista de bordas internas.
4. Salte para a posição correspondente na lista de vértices do polígono de recorte .
5. Siga a lista de vértices do polígono de recorte até que uma interseção seja encontrada.
6. Neste momento, copie os vértices percorridos do polígono de recorte, até o momento, para a lista de bordas internas.
7. Salte para a posição correspondente na lista de vértices do polígono alvo.
8. Repita o processo até que o ponto inicial seja alcançado. Nesse momento, um polígono interno foi fechado.

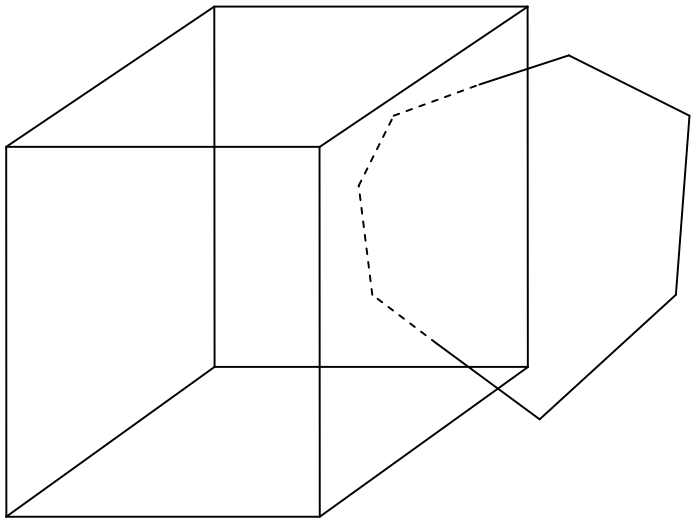

c2
c3
c6
c7
s6
s7
s4
s5
s2
s3
s12
s11
c5
c8
c4
c1
s9
s10
s1
s8
PA
PR
s1
s2
s3
s4
s5
s6
s7
s8
s1
s9
s10
s11
s12
s9
c1
c2
c3
c4
c1
c5
c8
c7
c6
c5
Borda externa
Borda interna


c2
c3
c6
c7
s6
i6
s7
i5
s4
s5
i3
i4
s2
s3
s12
i8
s11
c5
i2
c8
i7
c4
c1
i1
s9
s10
s1
s8
PA
PR
s1
i1
i2
s2
i3
i4
s3
s4
i5
s5
s6
i6
s7
s8
s1
s9
s10
s11
i8
c1
c2
c3
i6
i5
i4
i8
c4
i7
i1
c1
c5
i2
c8
i3
Entrada
i1,i3,i5
Saída
i2,i4,i6



5) Descreva uma versão tridimensional para o algoritmo de *Sutherland-Hodgman*, isto é, um algoritmo para recorte de polígonos em relação uma região do espaço 3d (1.0 ponto).

O algoritmo de Sutherland-Hodgman em 3D é idêntico ao caso bidimensional sendo a única diferença, o fato de que o recorte é feito entre um polígono e uma região do espaço. Para utilizar o algoritmo basta fazer o recorte das arestas do polígono em relação aos planos que definem as faces que delimitam a região, calculando, conforme necessário a interseção entre um segmento e um plano.



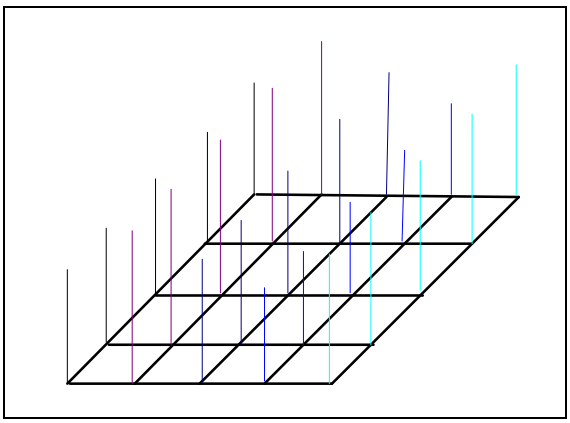
6) Explique o que é *Level-of-Detail* (LOD ou nível de detalhe) e cite uma aplicação em computação gráfica (1.0 ponto).

Segundo Luebke et al, *Level-of-Detail* é uma área da computação gráfica que lida com complexidade e desempenho através do controle da quantidade de detalhe usada para representar um mundo virtual. A ideia subjacente a LOD é muito simples: deve-se utilizar representações com menor nível de detalhe para objetos que estão distantes, são pequenos ou não são tão importantes na cena. (Luebke, D., Reddy, M., Cohen, J. D., Varshney, A., Watson, B., & Huebner, R. (2003). *Level of Detail for Computer Graphics.* (M. K. Publishers, Ed.)). Várias são as aplicações de LOD incluindo, jogos, animação, visualização de dados de terreno muito densos e outras.

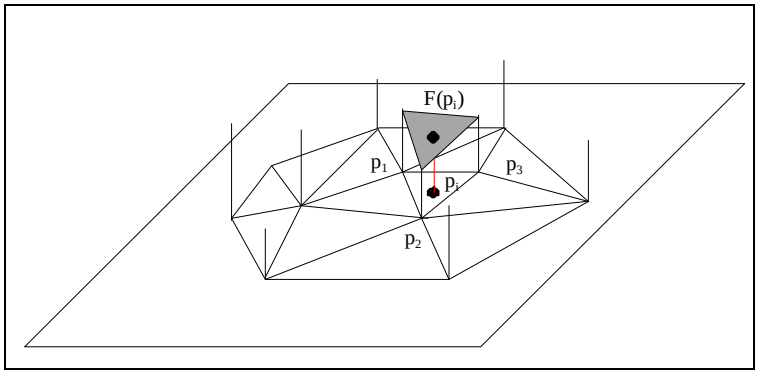
7) Como você modelaria um terreno em computação gráfica? (1.0 ponto).

Um terreno, na maioria dos casos, pode ser visto como o gráfico de uma função $h=f(x,y)$, isto é, um mapa de elevações.

Existem diferentes formas de se modelar um terreno, considerado como um mapa de elevações, e tais formas dependem de como a função é amostrada. Se o mapa de elevações for amostrado em um reticulado uniforme, então um terreno tem uma estrutura idêntica a uma imagem. Por outro lado se a amostragem for irregular, então é necessário algum mecanismo para criar uma estrutura (organização nos dados).

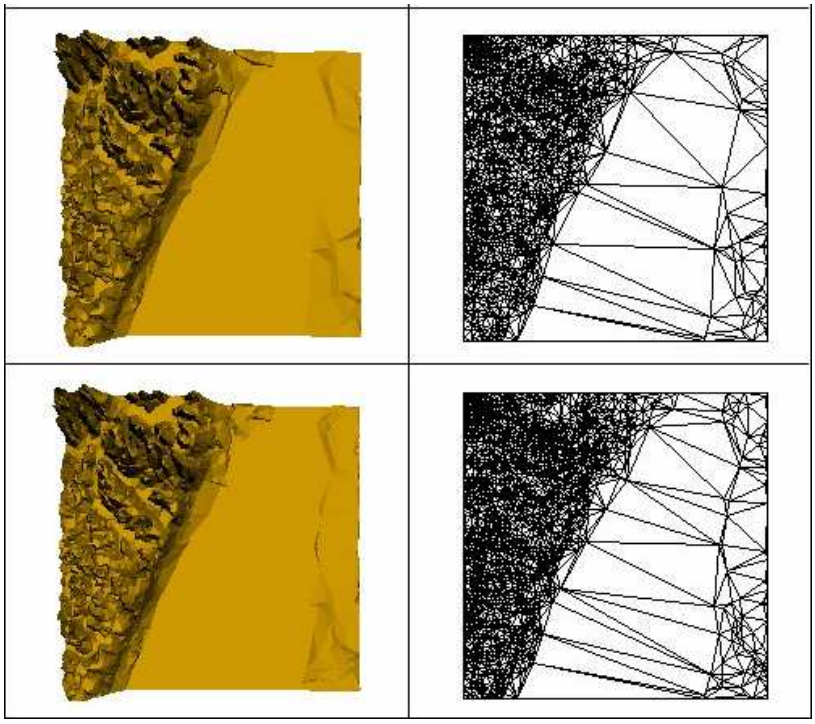


Amostragem uniforme



Amostragem não uniforme

Em ambas as situações, a superfície do terreno pode ser reconstruída através de uma malha de triângulos, via triangulação. A triangulação no caso da amostragem uniforme é direta, bastando triangular os quadrados formados por pontos vizinhos no reticulado, por exemplo, conectando pontos em uma diagonal. No caso não uniforme, pode-se utilizar uma triangulação de Delaunay, conforme a figura a seguir.



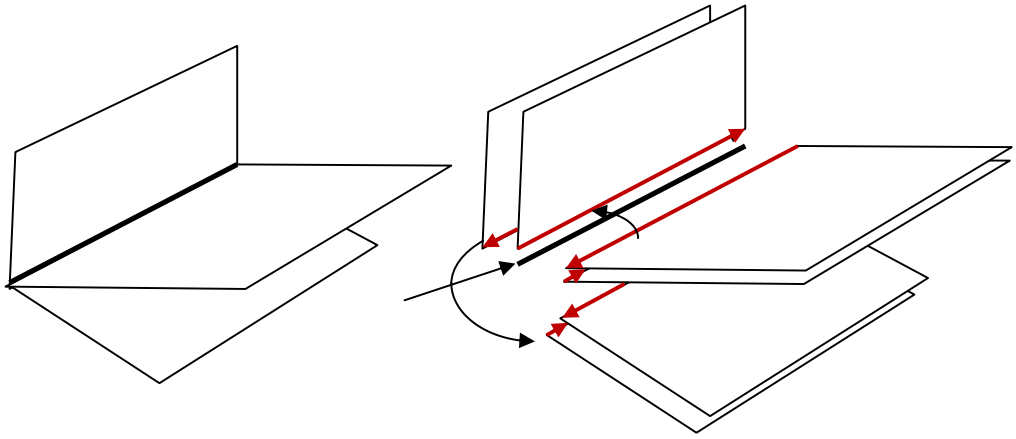
Obs.: Existem outras formas mais avançadas de se modelar terrenos que levam em consideração níveis de detalhe (ver questão 6).

8) Faça uma pesquisa sobre a estrutura de dados *Radial-Edge* e cite em que situações deve ser empregada (2.0 pontos).

A *Radial-Edge* é uma estrutura de dados topológica, proposta por Weiler (Weiler, K. (1988). The radial edge structure: a topological representation for non-manifold geometric boundary modeling. In Wozney M, McLaughlin H, E. J., editor, Geometric Modeling for CAD Applications, pages 3-36.), para representação de objetos gráficos que não são variedades (*non-manifolds*), isto é, objetos que localmente não se assemelham a um subconjunto do espaço euclidiano.

Um exemplo é o objeto abaixo, em que uma aresta tem várias faces adjacentes. Logo, sua vizinhança não se assemelha a um disco (subconjunto aberto do plano), como em uma superfície.

Para o exemplo abaixo, a *Radial-Edge* armazena uma lista de *edgeuses*, ordenada radialmente em torno da *edge*. Uma *edgeuse* nada mais é do uma aresta orientada, em uma face orientada do modelo (perceba que a aresta pode ser percorrida em sentidos diferentes, dependendo da face adjacente, e que neste modelo cada face tem dois lados, como uma folha). No caso, existe no exemplo abaixo 6 *edgeuses*, que são armazenadas em uma lista em ordem "radial" em torno da *edge*.



edge

edgeuse