



Computação Gráfica

UFRR – Departamento de Ciência da Computação
Computação Gráfica – Prof. Dr. Luciano F. Silva

Preenchimento

Professor: Luciano Ferreira Silva, Dr.



Preenchimento de áreas

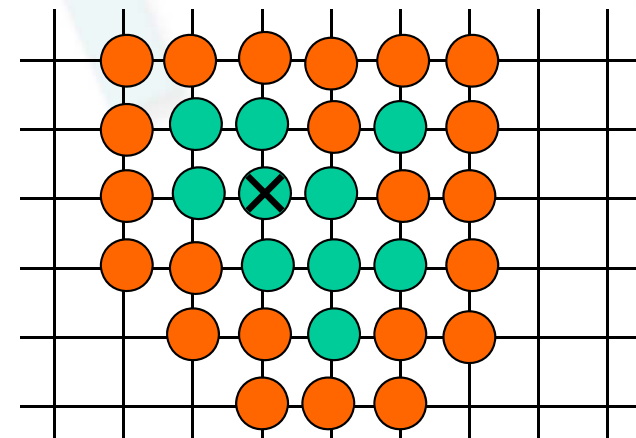
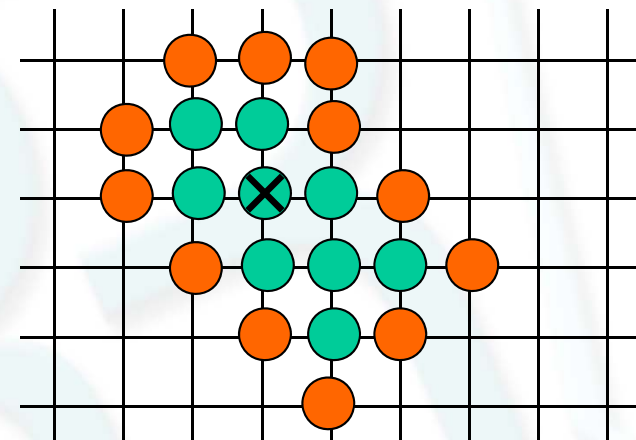
- Não é propriamente um processo de rasterização: operações no domínio da imagem;
- Procura-se as bordas de um dado elemento de forma a definir quando o pixel deve mudar de cor;
- Algoritmos:
 - ✓ *Flood Fill*;
 - ✓ de varredura com análise geométrica;



Algoritmo de preenchimento *Flood Fill*

■ Princípios:

- ✓ O polígono (área) já está desenhado na tela com uma dada cor;
- ✓ É necessário um pixel interno do corpo (semente) para dar início ao processo;
- ✓ Regiões são definidas por critérios de vizinhança ao pixel semente;
- ✓ Exemplo:
 - Pixels com cor semelhante à semente
 - Borda tem cor diferente
 - Pixels com cor diferente de uma cor dada
 - Borda tem cor igual à cor dada





Algoritmo de preenchimento *Flood Fill*

- **Algoritmo recursivo**

- ✓ Preenche vizinhos da semente que atendem ao critério
- ✓ Aplica o algoritmo recursivamente tomando esses vizinhos como sementes
- ✓ Termina quando nenhum vizinho atende o critério

- **Uso abusivo de recursão pode ser contornado preenchendo iterativamente intervalos horizontais**

- **Pode ser necessário usar um bitmap auxiliar para marcar os pixels visitados.**



Algoritmo de preenchimento *Flood Fill*

■ Pseudo-código:

Procedure *FloodFill* ($x, y, cor, novaCor$)

Se $pixel(x, y) = cor$ então

$pixel(x, y) \leftarrow novaCor$

FloodFill ($x + 1, y, cor, novaCor$)

FloodFill ($x, y + 1, cor, novaCor$)

FloodFill ($x - 1, y, cor, novaCor$)

FloodFill ($x, y - 1, cor, novaCor$)



Algoritmo de varredura com análise geométrica

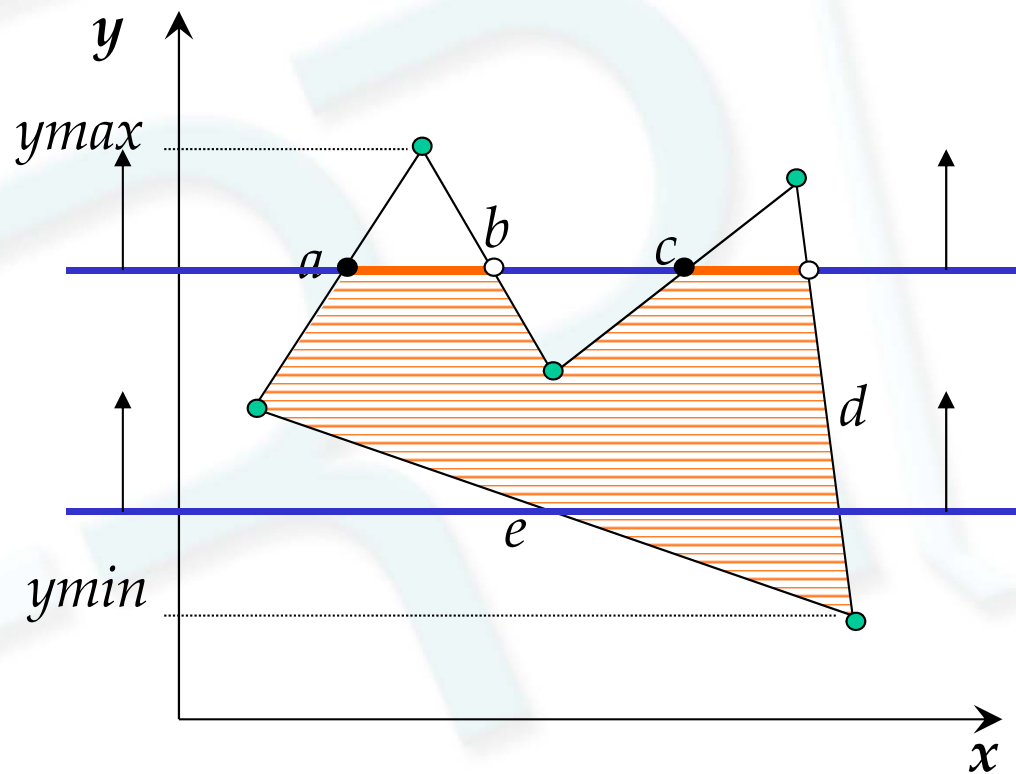
- **Baseia-se na descrição geométrica;**
 - ✓ como por exemplo, uma lista de vértices que formam o polígono;
- **Utiliza-se linhas de varredura ($y = \text{constante}$);**
- **Identifica-se pontos internos do polígono e as interseções das arestas com as linhas de varredura;**
- **Há uma tabela de lados para descrição do polígono em questão;**



Algoritmo de varredura com análise geométrica

■ Algoritmo clássico usa técnica de varredura

- ✓ Arestas são ordenadas
 - Chave primária: y mínimo
 - Chave secundária: x mín.
 - Exemplo: (e, d, a, b, c)
- ✓ Linha de varredura perpendicular ao eixo y percorre o polígono (desde y_{min} até y_{max})
- ✓ Intervalos horizontais entre pares de arestas são preenchidos

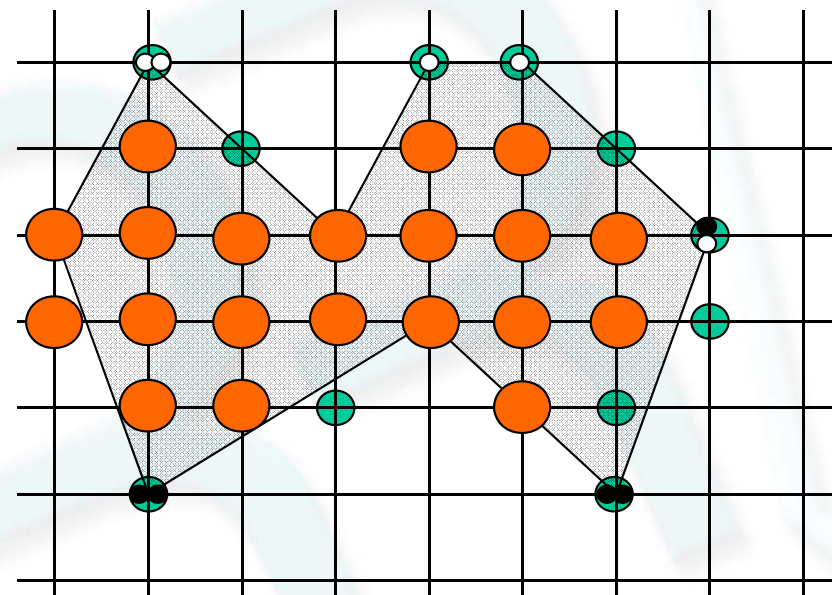




Algoritmo de varredura com análise geométrica

■ Intervalos de preenchimento

- ✓ Definidos sobre a linha de varredura
- ✓ Cada intervalo começa e termina sobre um pixel interceptado por uma aresta
 - Primeiro pixel é pintado, último não
- ✓ Arestas horizontais não são consideradas
- ✓ Um vértice de uma aresta horizontal é considerado apenas se for o vértice com *menor y*

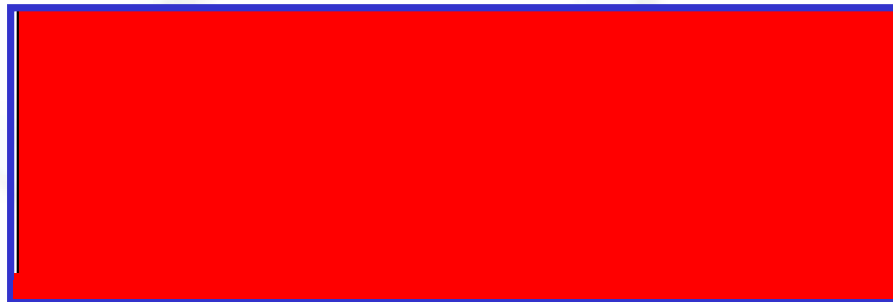


- Pixel resultante da interseção entre arestas e linhas de varredura
- Pixel pintado



Algoritmo de varredura com análise geométrica

- **Preenchimento de Retângulos: um dos mais simples;**

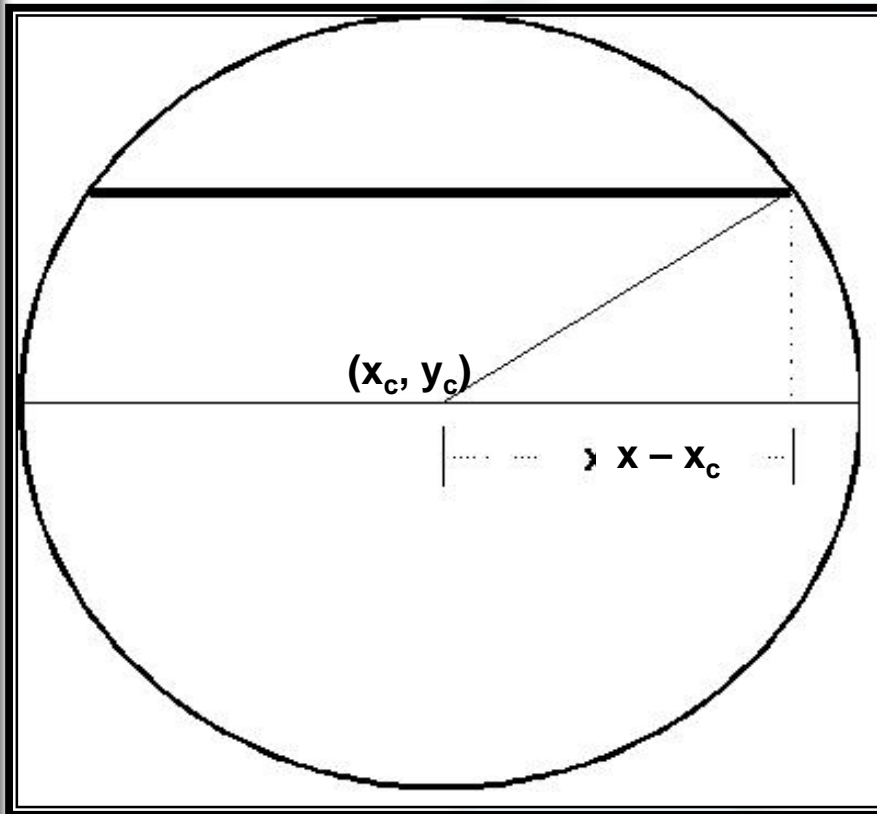


```
For (y=ymin; y<=ymax; y++)  
    for (x=xmin, x<=xmax;  
        x++)  
        writepixel(x,y,value);
```



Algoritmo de varredura com análise geométrica

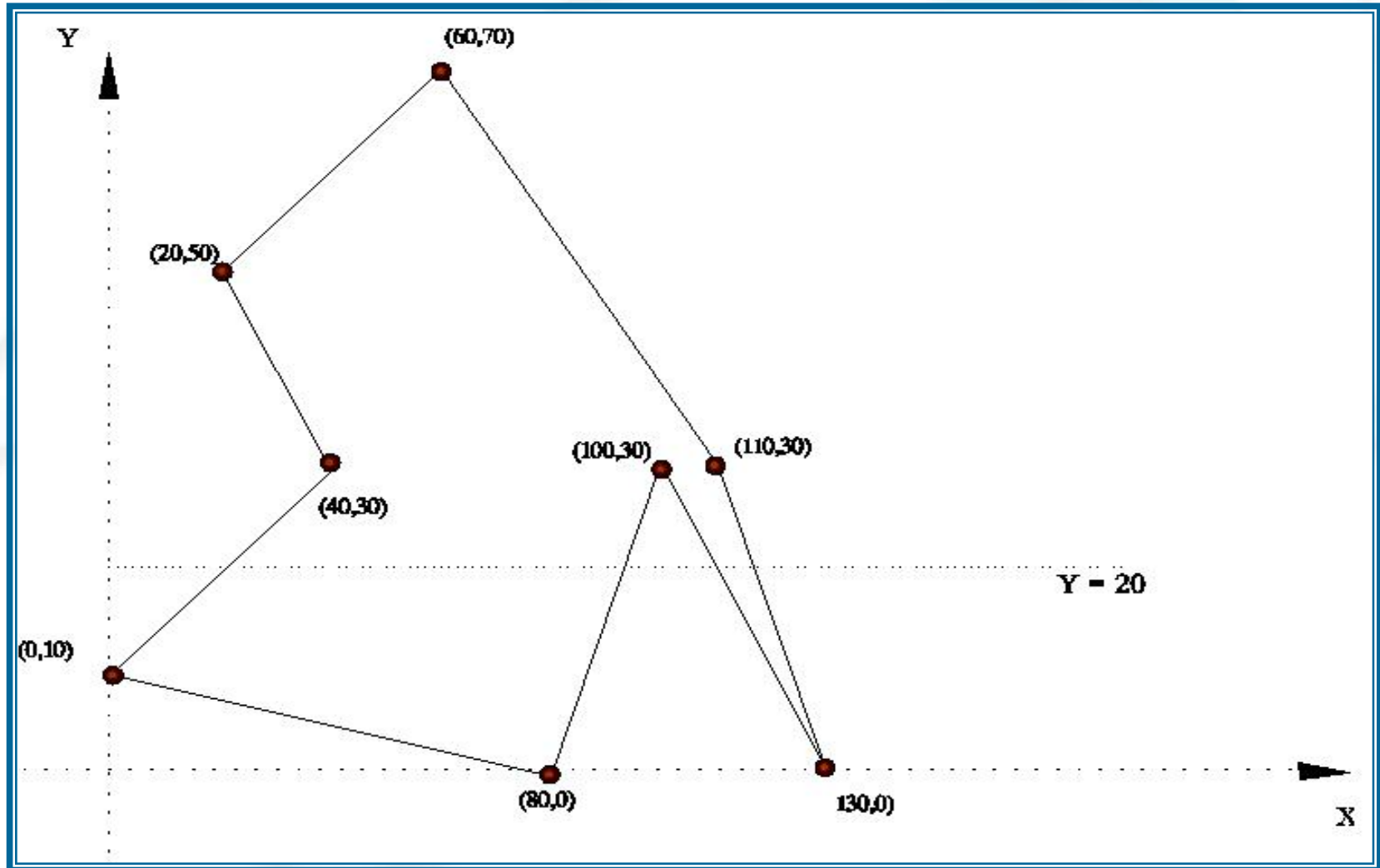
- **Preenchimento da Circunferência:** calcula-se facilmente a interseção da mesma com a linha de varredura



- $x_1 = x_c - \sqrt{R^2 - (y - y_c)^2}$
- $x_2 = x_c + \sqrt{R^2 - (y - y_c)^2}$
- Preenche-se de (x_1, y) até (x_2, y)
- Com $y_c - R < y < y_c + R$



Algoritmo de varredura com análise geométrica





Algoritmo de varredura com análise geométrica

1º passo: montar a Tabela de lados - descrição do polígono:

LADO	Y_{\min}	Y_{\max}	X para Y_{\min}	$1/m$
1	0	10	80	-8,0
2	10	30	0	+2,0
3	30	50	40	-1,0
4	50	70	20	+2,0
5	30	70	110	-1,25
6	0	30	130	-0,67
7	0	30	130	-1,0
8	10	30	80	+0,67

- É importante lembrar que: $\frac{1}{m} = \frac{\Delta x}{\Delta y}$



Algoritmo de varredura com análise geométrica

2º passo: identificar as diversas interseções com a linha de varredura: usamos a fórmula de cálculo - eq. da reta:

$$Y - Y_0 = m (X - X_0)$$

$$Y_{\text{varredura}} - Y_{\min} = m (X - X_{\min})$$

$$X - X_{\min} = \frac{1}{m} (Y_{\text{varredura}} - Y_{\min})$$

$$X = \frac{1}{m} \cdot (Y_{\text{varredura}} - Y_{\min}) + X_{\min}$$

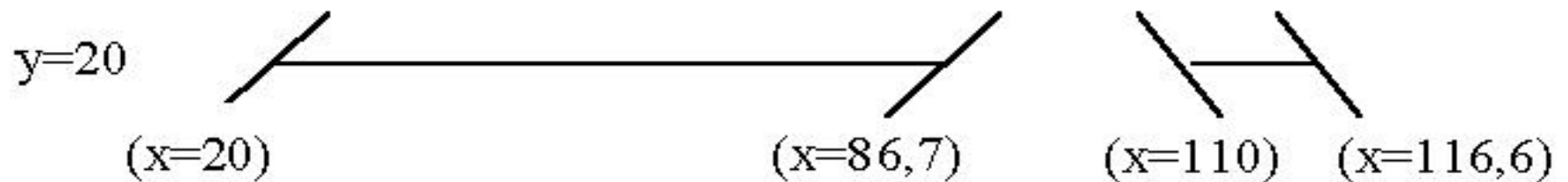
Com:

- $Y_{\text{varredura}} < Y_{\max}$
- $Y_{\text{varredura}} > Y_{\min}$



Algoritmo de varredura com análise geométrica

3º passo: ordenam-se os pontos e traçam-se linhas, tomando-as de duas em duas, a partir de x de valores crescentes:



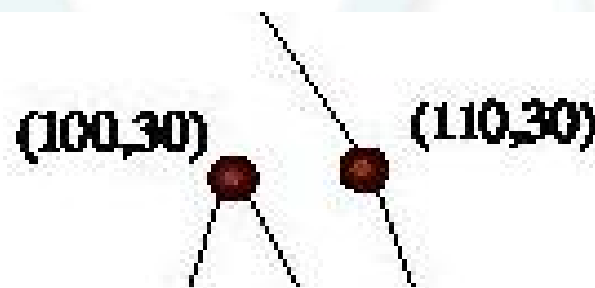
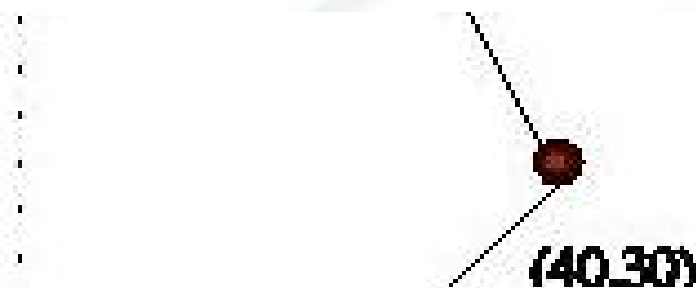
- Regra de paridade: iniciar contador com um número par, acrescentá-lo quando encontra uma intersecção e pintar quando ele for impar;



Algoritmo de varredura com análise geométrica

■ Problemas:

- ✓ Linha de varredura $y=30$ encontra um vértice



- ✓ Primeira interseção $(40,30) = 1$ único ponto para mudar o estado de preenchimento
- ✓ Interseção $(100,30) =$ não pode mudar o estado de preenchimento

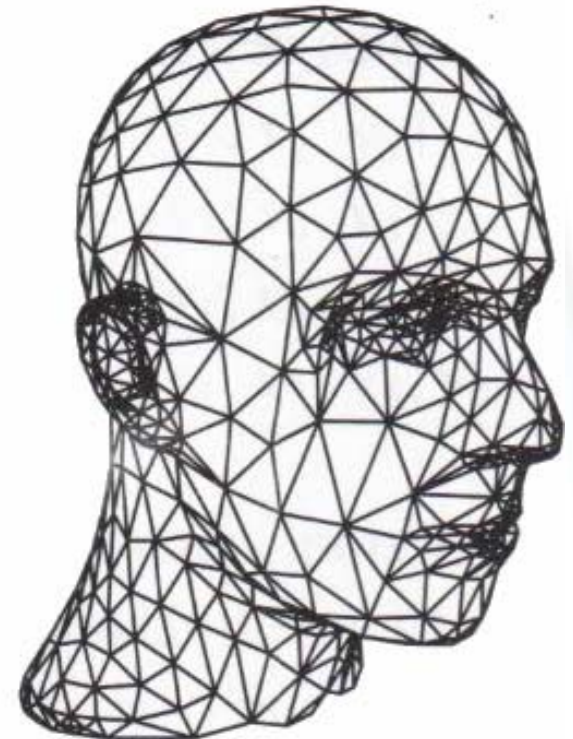
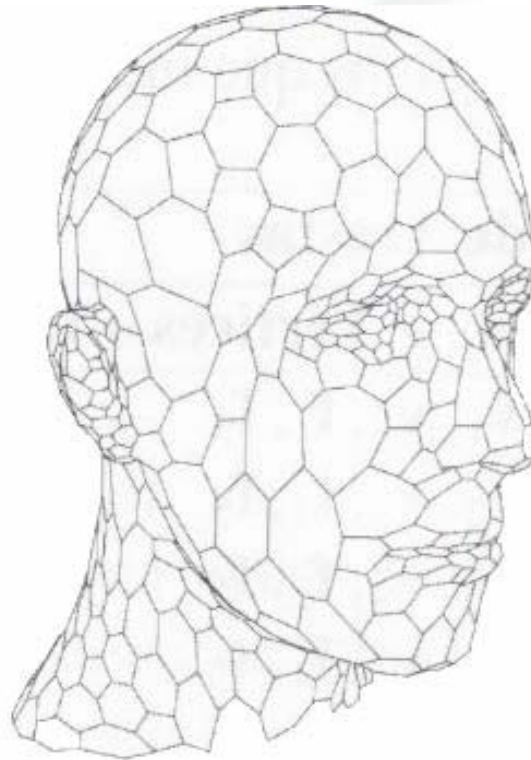
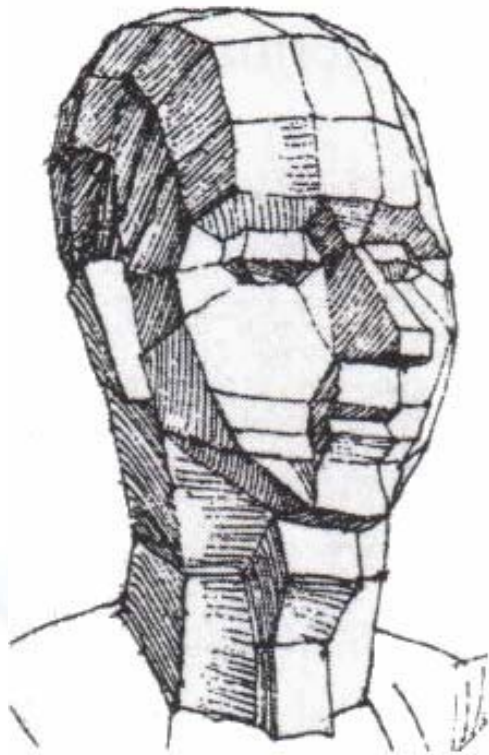


Rasterização de Polígonos

- **Operação fundamental em computação gráfica;**
 - ✓ Rasterização de linhas + preenchimento;
- **Polígono é dado por uma lista de vértices;**
 - ✓ Último vértice = primeiro vértice;
- **Obs.: OpenGL rasteriza apenas triângulos;**
 - ✓ Triângulos são casos especiais de polígonos;
 - ✓ Polígonos genéricos precisam ser triangulados;
 - Triangulação faz parte da biblioteca de utilitários (*gluTessellate*);



Triangulação



- **Principais vantagens:**

- ✓ **Planaridade** (triângulos são sempre curvas poligonais planas):
facilita enormemente os cálculos com as superfícies poliédricas.



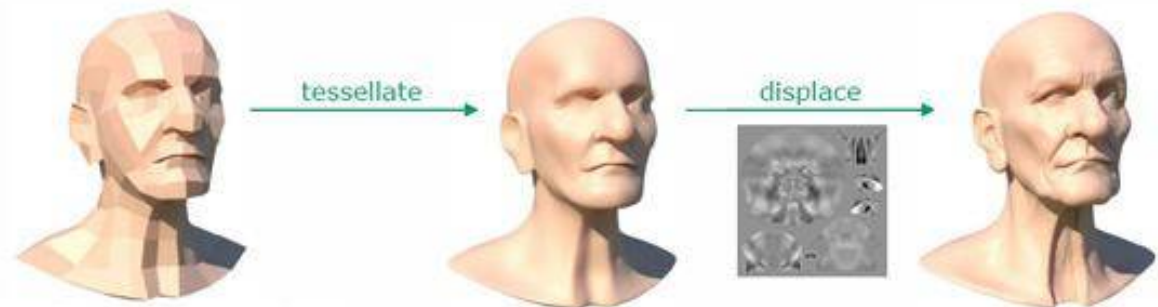
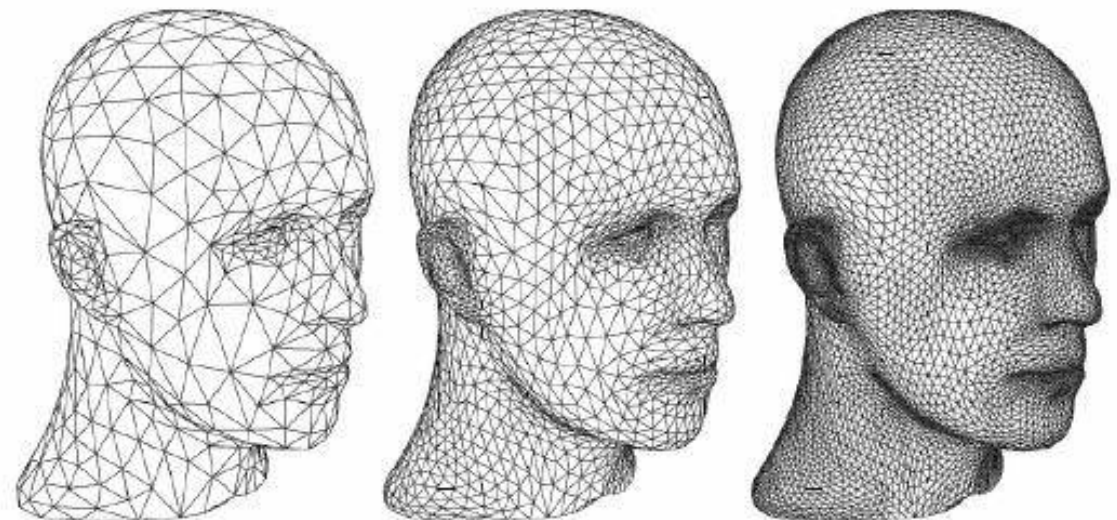
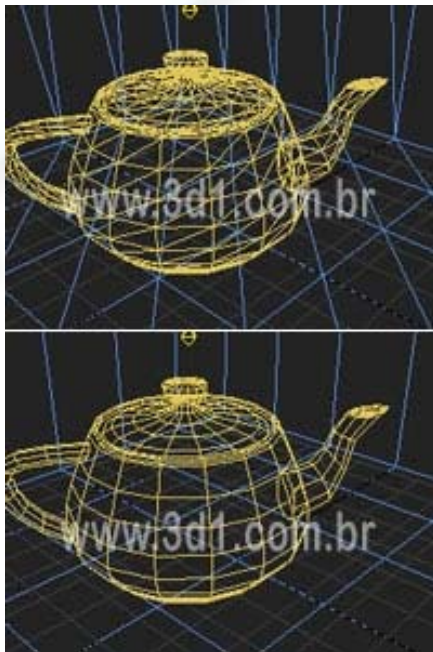
Triangulação

- ✓ **Sistema de Coordenadas** (um triângulo tem naturalmente associado um sistema de coordenadas lineares, coordenadas baricêntricas): o que pode ser usado para definir atributos da superfície definindo-os nos vértices de cada triângulo e fazendo interpolações nessas coordenadas.
- ✓ **Extensibilidade:** as triangulações se estendem naturalmente para \mathbb{R}^n com o conceito de *complexo simpliciais*.
- ✓ Qualquer polígono pode ser subdividido em triângulos.



Triangulação

- ✓ Maior controle da malha.
- ✓ Melhores resultados.

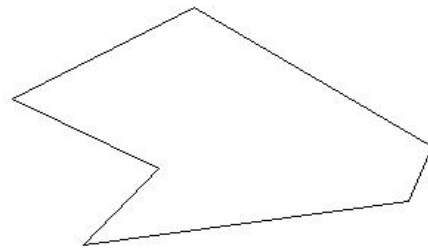




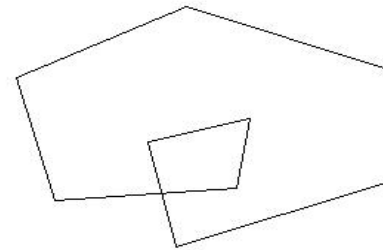
Quarto trabalho

- Construir um programa que permita preencher polígonos com os algoritmos de *Flood Fill* e de varredura com análise geométrica. Ele pintar uma circunferência, um retângulo e formas a, b, c e d. Descreva os resultados de maneira comparativa.

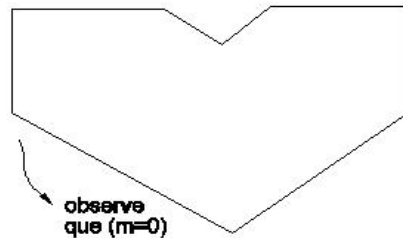
a)



b)



c)



d)

