

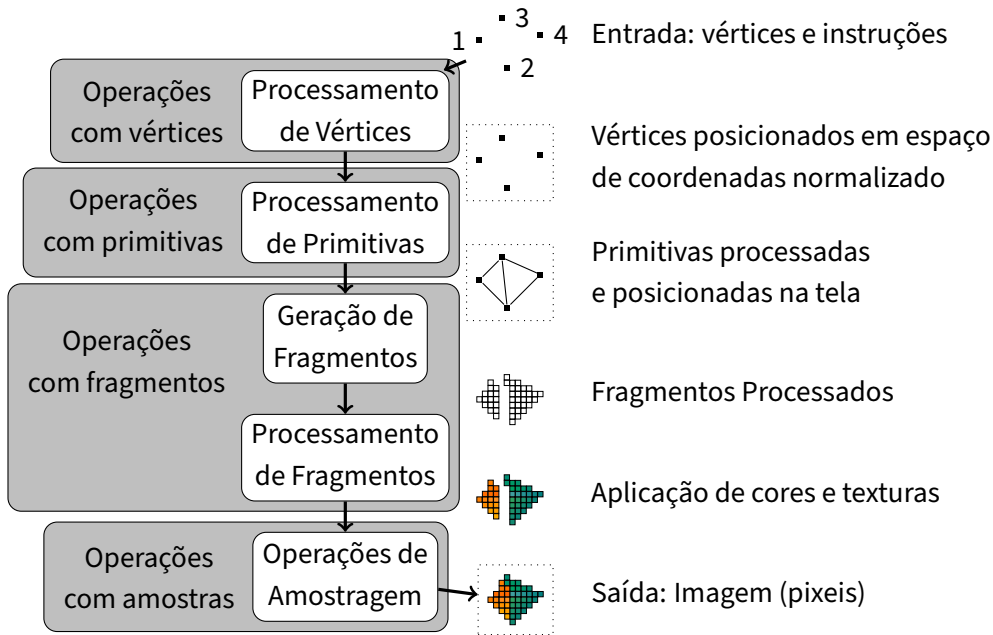
# Computação Gráfica

## Antialiasing

Prof. Alaor Cervati Neto



2022/1



# Sistema Gráfico

The background of the slide features a minimalist design with teal-colored geometric shapes. Two large triangles, one on the left and one on the right, point towards the center. These triangles overlap at the bottom, creating a darker teal triangular shape. The top half of the slide is a plain white background.

# Sistema Gráfico

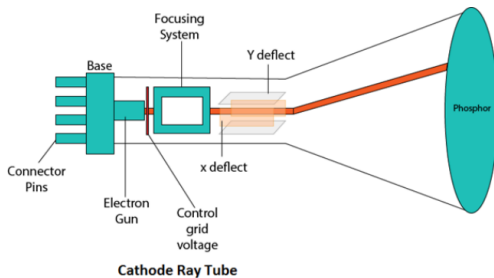
Partes do Sistema Gráfico:

- ▶ Dispositivos de Entrada
- ▶ Processador
- ▶ Memória
- ▶ *Frame Buffer*
- ▶ Dispositivos de Saída

# Sistema Gráfico

Monitor de Vídeo (dispositivo de saída):

- ▶ CRT (*Cathode Ray Tube*) foi usado por muito tempo.
- ▶ Feixe de elétrons em material fosforescente.

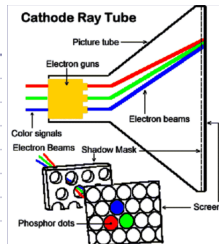


# Sistema Gráfico

Monitor de Vídeo (dispositivo de saída):

- ▶ CRT coloridos.
- ▶ Intensidade dos feixes corresponde à cor dos pixels.
- ▶ Primeiros tinham 8 cores.

Valores			Valor Binário	COR
R	G	B		
0	0	0	0	BLACK
0	0	1	1	BLUE
0	1	0	2	GREEN
0	1	1	3	CYAN
1	0	0	4	RED
1	0	1	5	MAGENTA
1	1	0	6	YELLOW
1	1	1	7	WHITE



# Sistema Gráfico

Monitor de Vídeo (dispositivo de saída):

- ▶ LCD (*Liquid Cristal Display*): tela é composta por cristais que são polarizados (via corrente elétrica) para gerar as cores.
- ▶ Plasma: tela composta por minúsculas cápsulas de vidro o gás (plasma) e uma capa de fósforo.
  - ▶ As cápsulas são os pixels e cada uma é composta de 3 subpixels que correspondem às cores RGB.
  - ▶ Corrente elétrica faz com que o plasma emita raios ultravioleta.
  - ▶ Essa luz faz o fósforo brilhar na cor apropriada.

# Sistema Gráfico

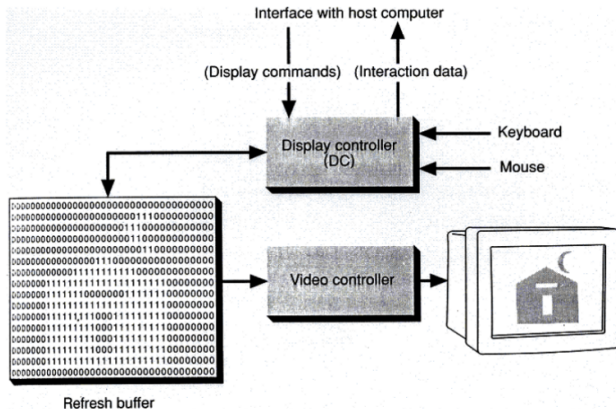
Monitor de Vídeo (dispositivo de saída):

- ▶ LED (*Light-Emitting Diode*): Uso de diodos (*Red, Green, Blue*) para emissão da luz. Pontos são iluminados de forma separada, melhorando a definição, cores e contraste.
- ▶ Outros dispositivos: projetores, hologramas, etc.



# Sistema Gráfico

Arquitetura simplificada de um dispositivo de exibição (matricial):



# Sistema Gráfico

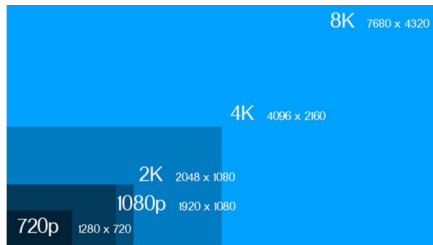
## Placas Gráficas:

- ▶ Hardware responsável por receber os comandos do processador e controlar o monitor de vídeo.
- ▶ *Drawing front end (drawing engine)*: recebe os comandos do processador com quais pixels a serem traçados e o sua intensidade (cores) e atualiza o “*bitmap*” do *frame-buffer*.
- ▶ *Video back-end*: interpreta o *bitmap* do *frame-buffer* e traduz para o monitor de vídeo.


# Sistema Gráfico

Imagem no dispositivo matricial:

- ▶ Cada imagem é mantida no *frame-buffer (fb)*, que contém uma posição associada a cada pixel da tela.
- ▶ Cada pixel tem um valor de intensidade (ou cor).
- ▶ Resolução é o número de pixels, que é igual à memória do *fb*.

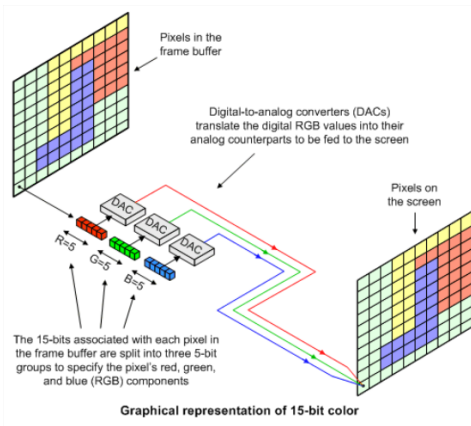


# Conversão Matricial

The bottom of the slide features a decorative graphic consisting of two large teal triangles pointing towards each other, meeting at a central point. Below this meeting point, there is a smaller, darker teal triangle pointing downwards.

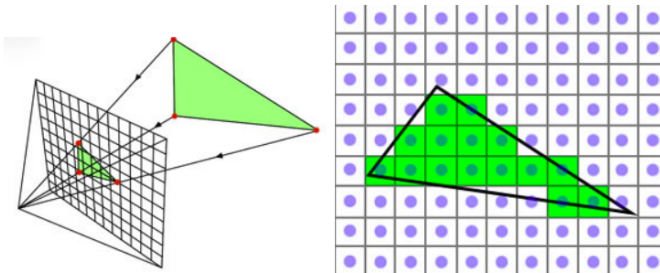
# Conversão Matricial

*Frame-buffer* contém representação matricial discreta da imagem (cena).



# Conversão Matricial

Objetos geométricos (vetoriais) precisam ser convertidos para a representação matricial (pixels).



Resulta em serrilhamento (*aliasing*)

# Conversão Matricial

Problema:

- ▶ Traçar primitivas geométricas (segmentos de reta, polígonos, circunferências, elipses, curvas, etc.) no dispositivo matricial.
- ▶ Rasterização: vetorial para matricial.
- ▶ Na prática, deseja-se converter do sistema de coordenadas reais (pontos flutuantes) para o sistema de coordenadas inteiras (discreto).

# Conversão Matricial

Características Desejáveis:

- ▶ Linearidade
- ▶ Precisão
- ▶ Espessura (Densidade Uniforme)
- ▶ Intensidade independente de inclinação
- ▶ Continuidade
- ▶ Rapidez no traçado



# Conversão Matricial

Considerando o caso mais simples, segmentos de reta:

- ▶ Equação explícita da reta:  $y = m \cdot x + b$ .
- ▶ Inclinação da reta:  $m = \frac{y_f - y_i}{x_f - x_i}$ .
- ▶ Interseção do eixo  $y$ :  $b = y_i - m \cdot x_i$ .

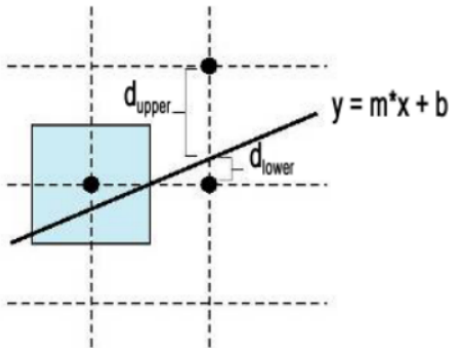
# Conversão Matricial

O algoritmo de Bresenham:

- ▶ Trabalha somente com inteiros.
- ▶ Incrementa  $x$  em intervalos unitários.
- ▶ Calcula o  $y$  correspondente.
- ▶ Partindo do ponto  $(x_k, y_k)$ :
  - ▶ Considere o caminho da esquerda para direita.
  - ▶ Assuma  $0 < |m| < 1$ .
  - ▶ Próximo ponto pode ser  $(x_{k+1}, y_k)$  ou  $(x_{k+1}, y_{k+1})$ .

# Conversão Matricial

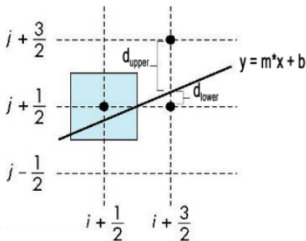
Regra de decisão:  $\begin{cases} (d_{\text{lower}} - d_{\text{upper}}) \geq 0 \implies \text{usar o pixel superior} \\ (d_{\text{lower}} - d_{\text{upper}}) < 0 \implies \text{usar o pixel inferior} \end{cases}$



# Conversão Matricial

Calculando  $d_{\text{lower}}$  e  $d_{\text{upper}}$ :

- ▶ Com base na equação da reta ( $y = m \cdot x + b$ ), na posição  $x_k + 1$ , a coordenada  $y$  é calculada como  $y = m \cdot (x_k + 1) + b$ .
- ▶ Então,  $d_{\text{lower}} = y - y_k = m \cdot (x_k + 1) + b - y_k$  e  $d_{\text{upper}} = (y_k + 1) - y = y_k + 1 - m \cdot (x_k + 1) + b$ .



# Conversão Matricial

Regra de decisão:

- Um teste rápido para saber a proximidade:

$$p_k = d_{\text{lower}} - d_{\text{upper}} = 2m(x_k + 1) - 2y_k + 2b - 1.$$

- Assim: 
$$\begin{cases} p_k > 0 \implies \text{pixel superior} \\ p_k < 0 \implies \text{pixel inferior} \end{cases}$$

- Porém, calcular  $m$  envolve operações de ponto flutuante:  $m = \frac{y_{\text{end}} - y_0}{x_{\text{end}} - x_0} = \frac{\Delta y}{\Delta x}.$

- Então, substituindo  $m$  por  $\frac{\Delta y}{\Delta x}$  e multiplicando tudo por  $\Delta x$ , tem-se:

$$p_k = \Delta x (d_{\text{lower}} - d_{\text{upper}}) = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c.$$

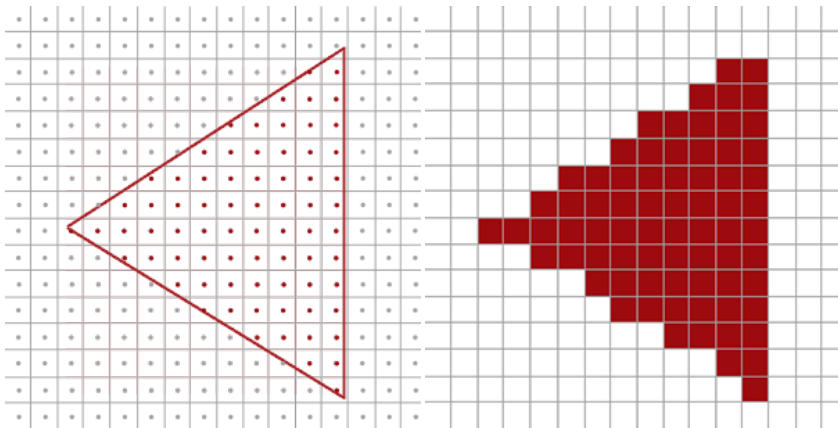
- Onde  $c = 2\Delta y + \Delta x (2b - 1)$  é um parâmetro constante independente da posição do pixel.

*Antialiasing*

The background of the slide features a white central area where the text is located. This central area is framed by two large teal-colored triangles that point towards each other, meeting at a point at the bottom center. The overall composition is minimalist and modern.

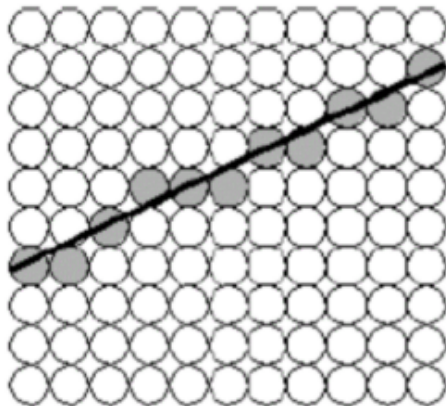
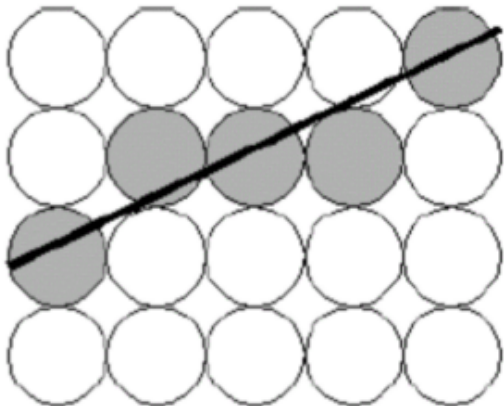
# Correção de Traçado (*Antialiasing*)

Voltando ao problema de serrilhado na conversão matricial.



## Correção de Traçado (*Antialiasing*)

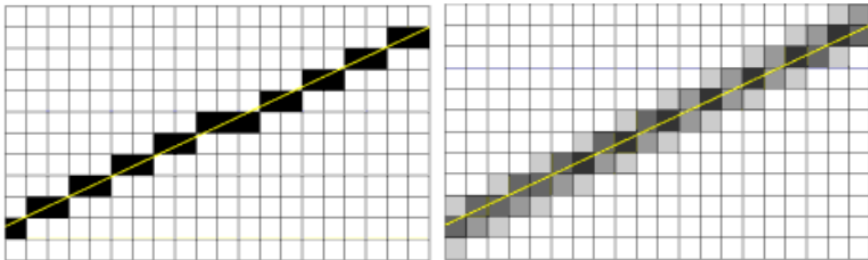
A forma mais simples de resolver esse problema é aumentando a resolução.





# Correção de Traçado (*Antialiasing*)

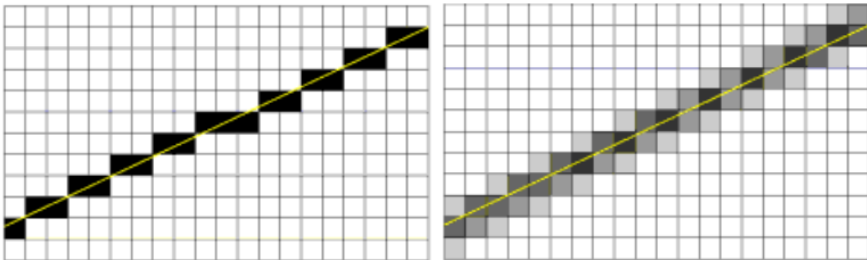
A estratégia mais simples é modificar a intensidade do pixel proporcional à cobertura do traçado.



Baseado em área: modifica a intensidade do pixel proporcional à cobertura do traçado.

# Correção de Traçado (*Antialiasing*)

A estratégia mais simples é modificar a intensidade do pixel proporcional à cobertura do traçado.



Funciona bem para gráficos simples (linhas e curvas), mas é ineficiente para cenários 3D.

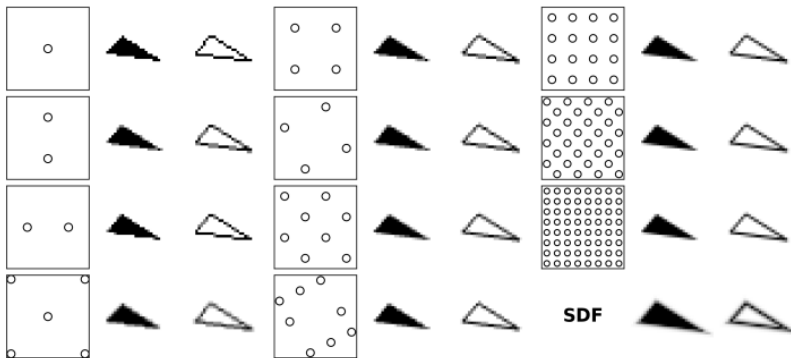
# Correção de Traçado (*Antialiasing*)

Técnica *Supersampling*:

- ▶ Simula um monitor com maior resolução (mais pixels).
- ▶ Na prática, cria (sub) pixels de tamanhos menores.
- ▶ Converte a imagem para esse monitor simulado.
- ▶ A intensidade do pixel real (monitor real) é definida com base na quantidade de subpixels cobertos.

# Correção de Traçado (*Antialiasing*)

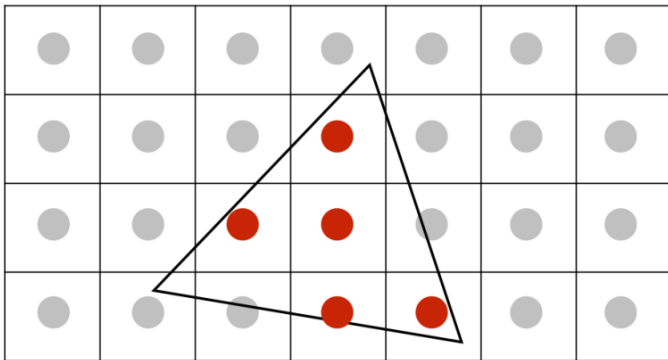
Técnica *Supersampling*:



*Antialiasing* com diferentes quantidades de subpixels.

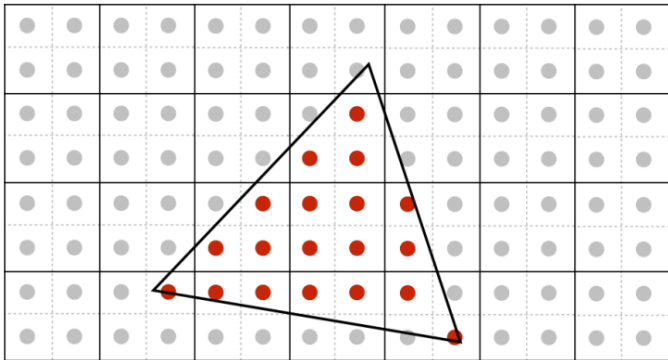
# Correção de Traçado (*Antialiasing*)

Técnica *Supersampling*:



# Correção de Traçado (*Antialiasing*)

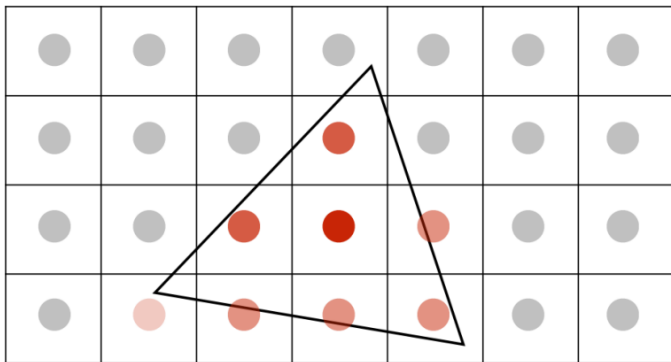
Técnica *Supersampling*:



Supersampling  $2 \times 2$

# Correção de Traçado (*Antialiasing*)

Técnica *Supersampling*:



# Correção de Traçado (*Antialiasing*)

Técnica *Supersampling*:

			75%			
		100%	100%	50%		
	25%	50%	50%	50%		



# Correção de Traçado (*Antialiasing*)

Técnica *Supersampling*:

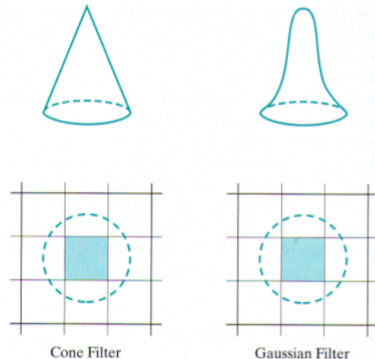
- ▶ Alguns autores defendem que o *supersampling* atinge qualidades próximas do ideal.
- ▶ No entanto, custos computacional e de memória inviabilizam seu uso para algumas aplicações (em tempo real).

Uma estratégia alternativa é *Multi-Sampling*:

- ▶ Técnicas para selecionar regiões de pixels que precisam de correção (por exemplo, regiões de fronteira dos objetos).
- ▶ Implementada em muitas GPUs.

# Correção de Traçado (*Antialiasing*)

Técnica de Filtragem: uma superfície contínua de ponderação é usada para determinar a cobertura do pixel.



# Correção de Traçado (*Antialiasing*)

DLSS (*Deep Learning Super Sampling*):



# Material de base para a aula

- ▶ Introduction to Computer Graphics. Version 1.2, January 2018, David J. Eck. Source: <http://math.hws.edu/graphicsbook/>.
- ▶ Imagem Rasterização: <https://www.techpowerup.com/review/nvidia-geforce-turing-geforce-rtx-architecture/5.html>.
- ▶ Imagem Antialiasing: <https://learnopengl.com/Advanced-OpenGL/Anti-Aliasing>.
- ▶ Computação Gráfica: Aula 11. Slides de Ricardo M. Marcacini. Disciplina SCC0250/0650, ICMC/USP, 2021.