DCC192

2025/2

UFMG

Desenvolvimento de Jogos Digitais

A3: Gráficos I — Fundamentos e Modelos

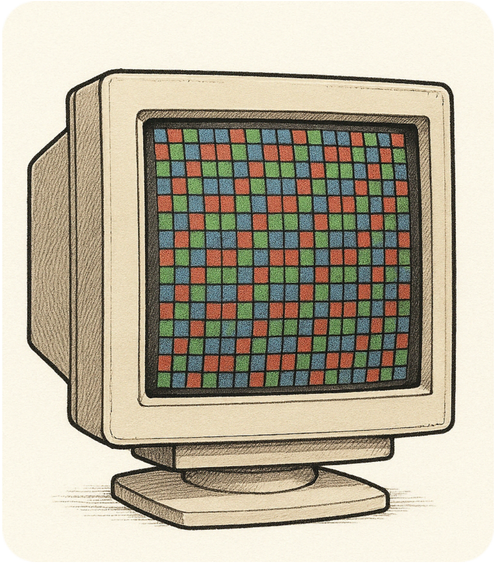
Prof. Lucas N. Ferreira

Plano de Aula

- Hardware Gráficos
  - Monitores, Imagens e Cores
- Computação Gráfica
- Modelos 3D
  - Vértices e Atributos
  - Formatos de especificação
- Pipeline Gráfico
-

Monitores

Monitores são dispositivos para mostrar imagens digitais. Para isso, eles possuem uma matriz de pixels atualizada a uma determinada frequência:



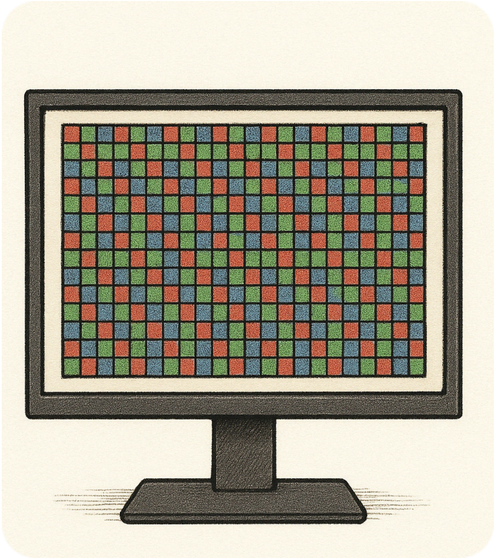
Resoluções típicas:

- 640x480
- 800x600
- 1024x768
- 1280 x 1024

Taxa de atualização:

- De 60Hz a 85Hz

Monitores CRT



Resoluções típicas:

- 1280x720
- 1920x1080
- 2560x1440
- 3840x2160

Taxa de atualização:

- 60Hz, 144Hz, 240Hz, ...

Monitores LCD/LED/OLED

A principal diferença entre as tecnologias de monitores é como elas representam e atualizam essa matriz de pixels.

Monitores CRT vs LCD/LED/OLED

Esse vídeo utiliza câmeras com taxas de quadros mais altas do que as de TVs para visualizar como cada tecnologia (CRT vs LCD/LED/OLED) atualiza as imagens na tela.

https://www.youtube.com/watch?v=3BJU2drrtCM&ab_channel=TheSlowMoGuys

Monitores CRT

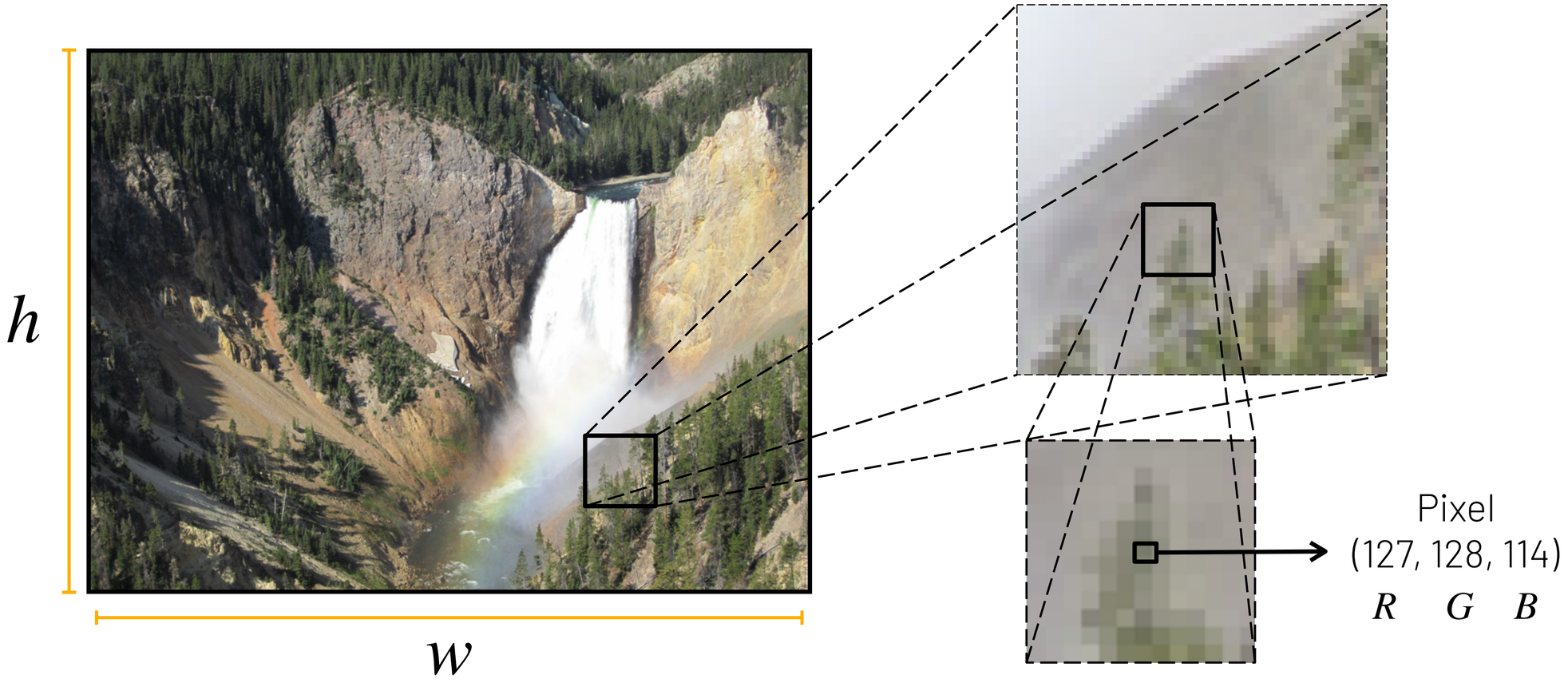
A imagem é formada por um canhão de elétrons que atira feixes de elétrons na parte de trás da tela de vidro de cima para baixo, em linhas, dezenas de vezes por segundo

Monitores LCD/LED/OLED

A tela é composta por uma matriz de lâmpadas (ex. LED), portante não há feixe percorrendo a tela, todos os pixels podem ser atualizados de forma independente.

Imagens

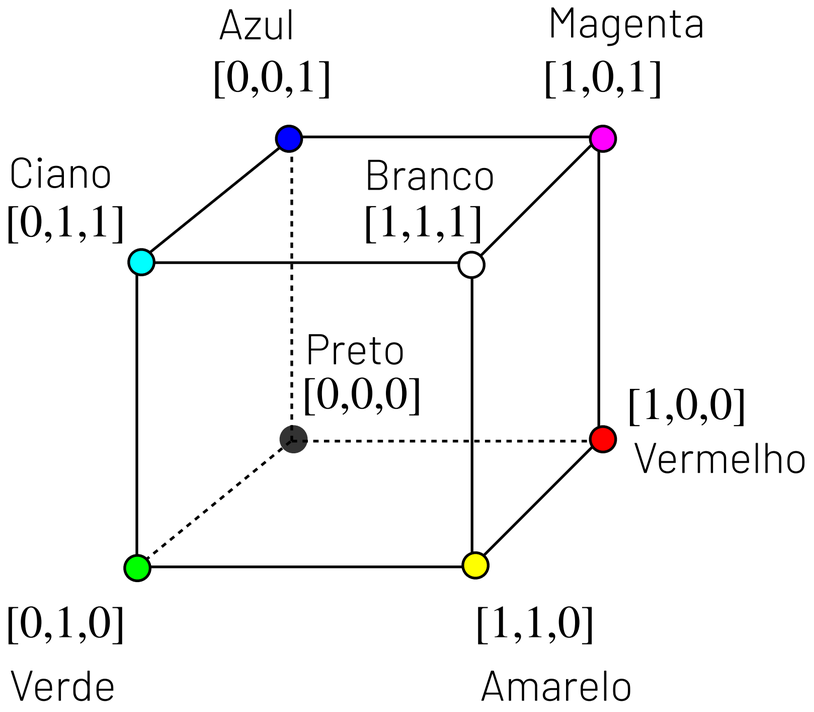
Imagens são **arranjos bidimensionais de pixels** ($w \times h$), onde cada pixel é representado por 3 valores (canais): vermelho (R), verde (G) e azul (B)



Cores

O sistema RGB é um dos principais padrões para definição de cores, onde uma cor é definida pela combinação de 4 canais: Vermelho (R), Verde (G), Azul (B) e Transparência (Alpha)

- Cores são representadas em 32 bits, 8 bits por canal
- Intensidade de cada canal varia entre 0 e 255
- 2^{32} or ~4 bilhões de cores diferentes
- O sistema RGB pode ser visualizado como um cubo, onde cada ponto interno representa uma cor



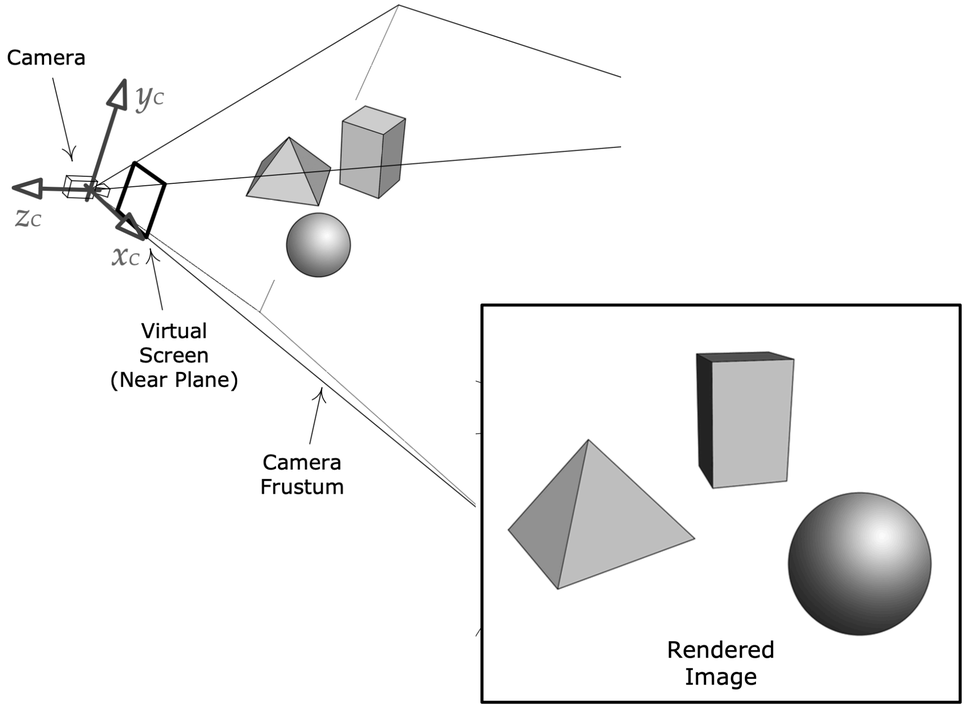
Transparência

O **Canal Alpha** é um quarto canal de cor usado para adicionar transparência às imagens. Esse canal é **uma adição de software,** monitores não desenham um canal de transparência!

Computação Gráfica

Um dos principais problemas da área de Computação Gráfica consiste em gerar uma *imagem* (i.e., arranjo bidimensional de pixels) a partir de uma cena composta por:

- **Objetos 3D**
  Geralmente representados por conjuntos de vértices.
- **Câmera**
  Geralmente representada por uma posição, orientação, distância focal e planos de recorte próximo e distante
- **Fonte de Luz**
  Vários tipos de fontes de luz podem ser especificadas: direcional, ambiente e spot.
- **Materiais**
  Propriedades visuais dos objetos, descrevendo como a luz deve interagir com os objetos.



Renderização em Tempo Real vs. Pré-Renderização

Há uma importante distinção dentro da Computação Gráfica quanto às restrições de tempo impostas no processo de geração de imagens:

Marvel's Spider-Man (2018) — Video Game

Renderização em Tempo Real

- As imagens são geradas instantâneamente, muitas vezes a 60 quadros por segundo;
- Algoritmos e técnicas priorizam a velocidade, mesmo que isso signifique sacrificar um pouco da qualidade visual;
- Possibilita interatividade no processo de geração de imagens.

Renderização em Tempo Real vs. Pré-Renderização

Há uma importante distinção dentro da Computação Gráfica quanto às restrições de tempo impostas no processo de geração de imagens:

Spider Man: No Way Home (2021) — Filme
https://www.youtube.com/watch?v=MH58W4oDig4&ab_channel=BBCClick

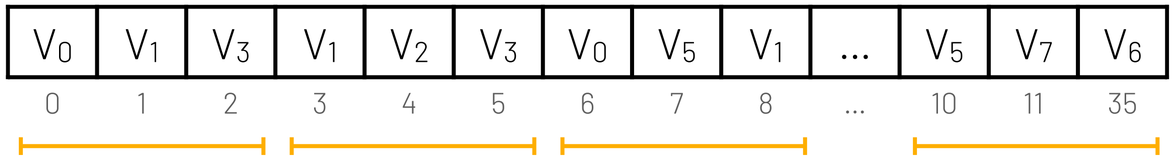
Pré-Renderização

- As imagens são geradas com antecedência, antes de serem mostradas ao usuário.
- Algoritmos e técnicas priorizam qualidade visual máxima, com efeitos realistas de luz, sombra, reflexo, refração, etc.
- É usada quando o conteúdo não precisa ser interativo, como em filmes.

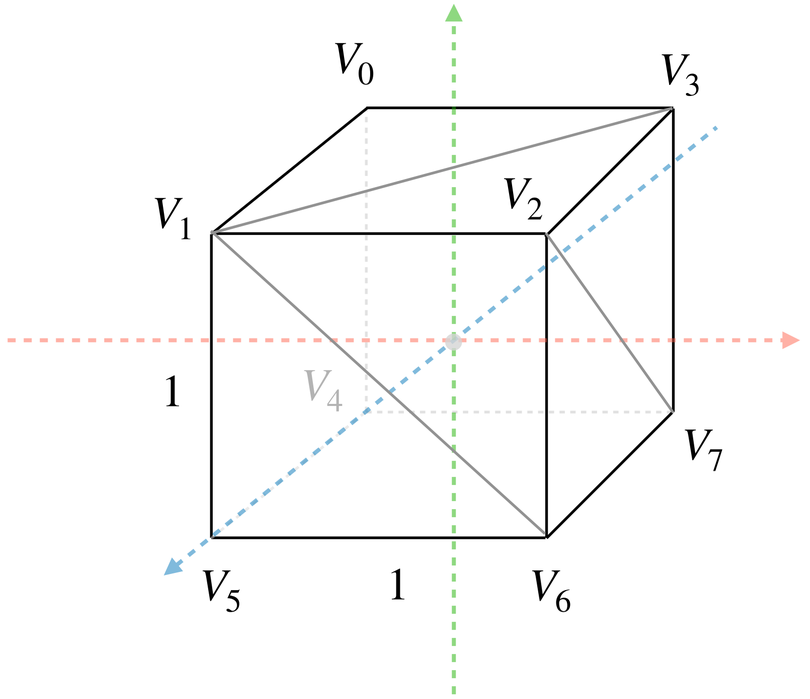
Objetos 3D — Lista de Triângulos

A maioria dos objetos (ou modelos) em jogos são representados por malhas de triângulos. A forma mais simples de definir uma malha é utilizar uma **lista de triângulos**:

- Cada triângulo é formado por um grupo de três vértices consecutivos $\{V_i, V_{i+1}, V_{i+2}\}, V_i \in \mathbb{R}^3$



- Os vértices são definidos no **espaço do objeto**, um sistema de coordenadas centrado no objeto. Por exemplo, no cubo ao lado:
  - $V_1 = [-0.5, 0.5, 0.5]$
  - $V_3 = [0.5, 0.5, -0.5]$
  - $V_5 = [-0.5, -0.5, 0.5]$

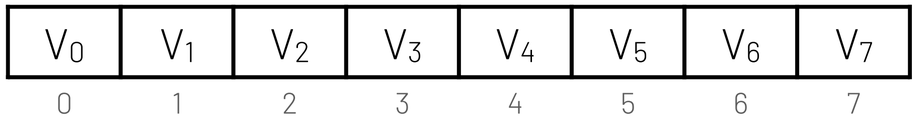


Por exemplo, um cubo pode ser representado por 12 triângulos, dois para cada face.

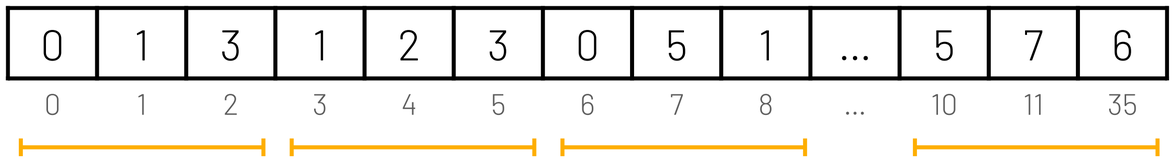
Objetos 3D — Lista de Triângulos Indexadas

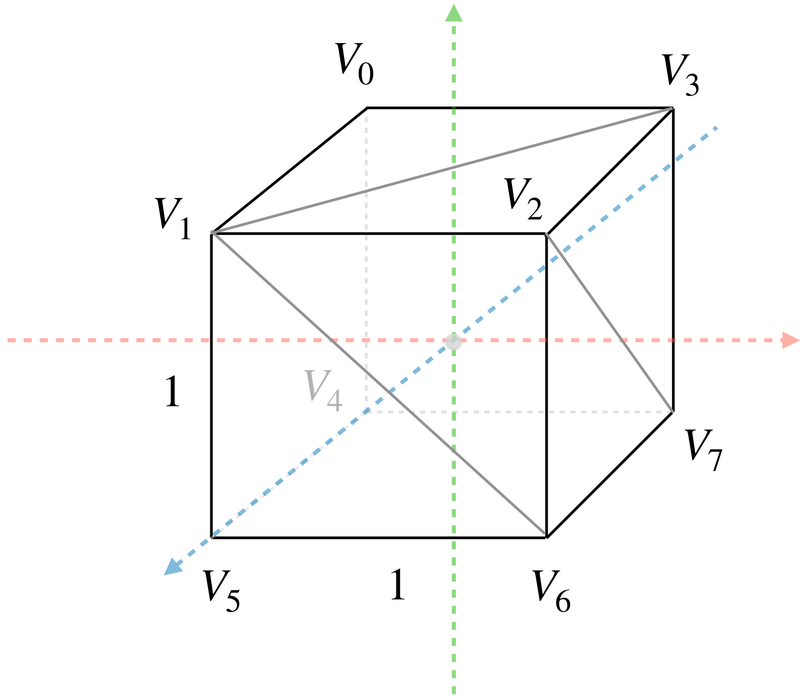
Na representação por lista de triâgulos, vários vértices são armazenados mais de uma vez, o que não é eficiente. A **lista indexada de triângulos** é uma estrutura mais eficiente:

- Criamos uma lista com os vértices do modelo aparecendo apenas uma vez:



- Criamos uma **lista de índices** (inteiros) adicional para definir as triplas de vértices que formam os triâgulos:



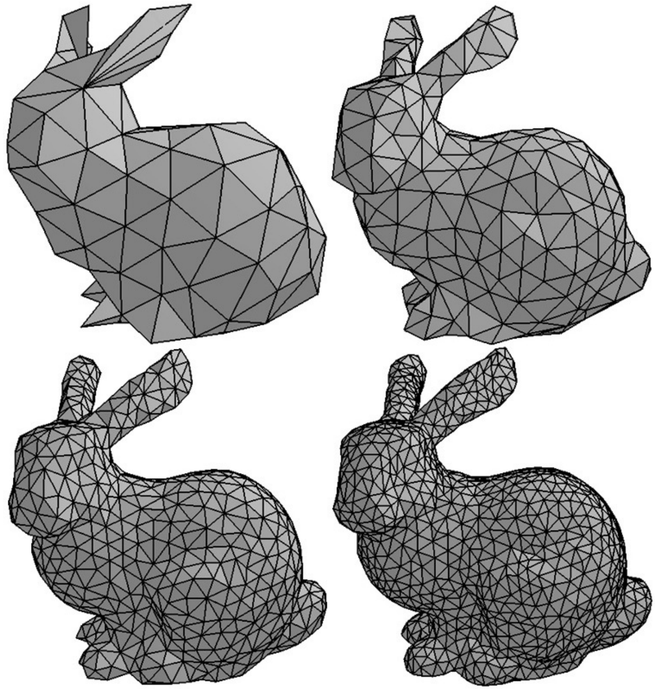


Geralmente armazenamos muitos metadados com cada vértice, então repetir esses dados em uma lista de triângulos desperdiça memória.

Por que triângulos?

Em jogos digitais, malhas triângulos são utilizados para representar os objetos 3D pois triângulos são:

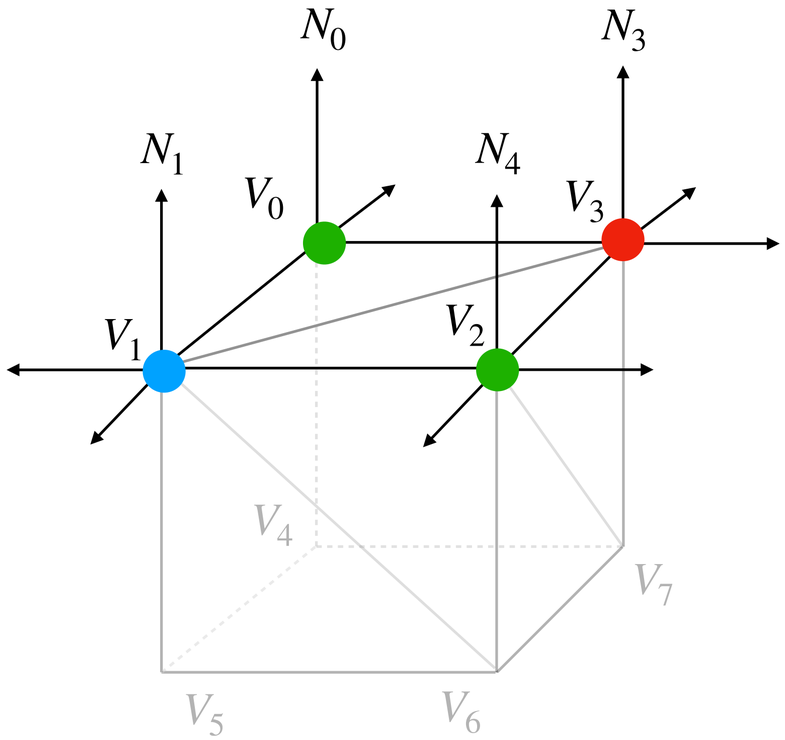
- O tipo de polígono mais simples;
- Sempre planares;
- Permanecem triângulos sob a maioria dos tipos de transformações;
- A maioria das GPUs para jogos são projetadas em torno da rasterização triangular.



Atributos dos Vértices

Além de suas posições $V_i = [v_x, v_y, v_z]$, cada vértice possue atributos associados, para definir propriedades visuais de cada região do objeto:

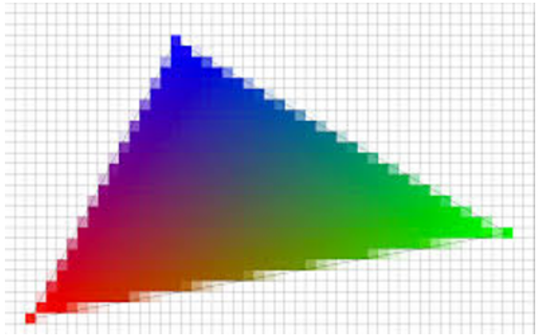
- Vetor normal: $N_i = [n_{ix}, n_{iy}, n_{iz}]$
- Cor: $D_i = [d_R, d_G, d_B, d_A]$
- Coordenadas de Texturas: $U_i = [u_{ij}, v_{ij}]$
- ...

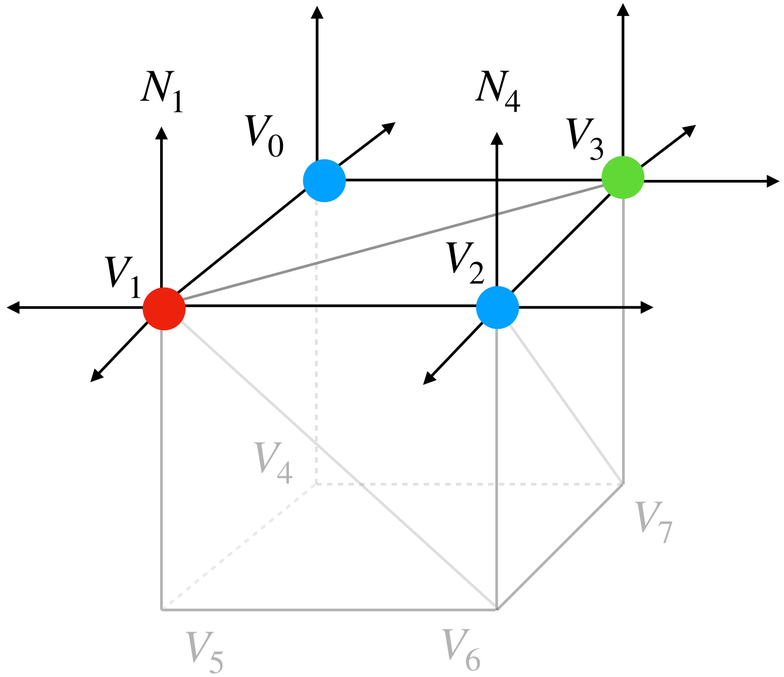


Atributos dos Vértices – Interpolação

Como especificamos apenas os atributos dos vértices, mas precisamos colorir as superfícies inteiras, a GPU interpola linearmente os valores dos atributos entre as superfícies:

- Todos os atributos são interpolados: posição, normal, cor, etc...
- Exemplo de como a interpolação de cor afeta a superfície do triângulo $\{V_0, V_1, V_3\}$:





Criando Modelos

Modelos 3D para jogos são tipicamente criados com editores gráficos especializados, por exemplo:

- Blender
  https://www.blender.org/
- Autodesk Maya
  https://www.autodesk.com/maya
- ZBrush
  https://www.maxon.net/zbrush
- BlockBench
  https://www.blockbench.net/

https://www.youtube.com/watch?v=Y05txBWzBqE&ab_channel=Lukky

Salvando/Carregando Modelos

Os editores de modelos 3D exportam objetos para arquivos to tipo FBX, OBJ, STL, etc. Esses formatos armazenam os vértices, as faces, as coordenadas de texturas, entre outros:

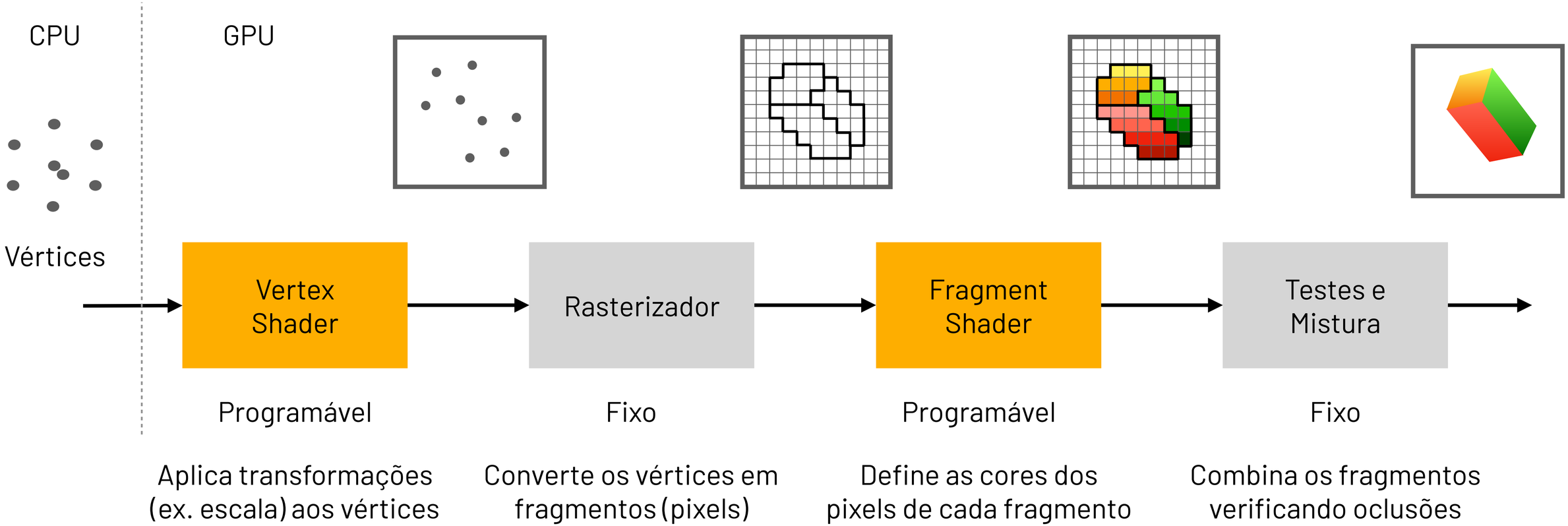
```
# OBJ file format with ext .obj
# vertex count = 2503
# face count = 4968
v -3.4101800e-003 1.3031957e-001 2.1754370e-002
v -8.1719160e-002 1.5250145e-001 2.9656090e-002
v -3.0543480e-002 1.2477885e-001 1.0983400e-003
v -2.4901590e-002 1.1211138e-001 3.7560240e-002
v -1.8405680e-002 1.7843055e-001 -2.4219580e-002
v 1.9067940e-002 1.2144925e-001 3.1968440e-002
…
f 1069 1647 1578
f 1058 909 939
f 421 1176 238
f 1055 1101 1042
f 238 1059 1126
f 1254 30 1261
```

https://graphics.stanford.edu/~mdfisher/Data/Meshes/bunny.obj

https://en.wikipedia.org/wiki/Stanford_bunny#/media/File:Stanford_Bunny.stl

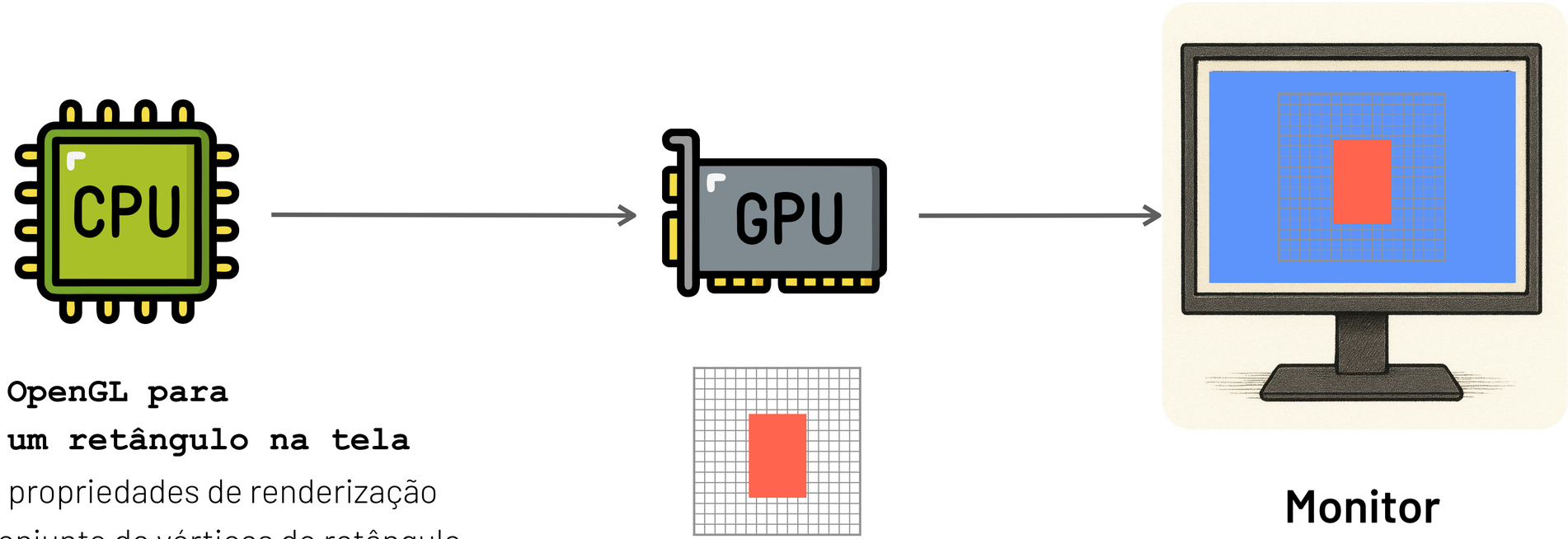
Pipeline Gráfico

Os frameworks de computação gráfica, como DirectX e OpenGL, geralmente produzem uma imagem a partir de uma cena em uma sequência de operações, chamado de **Pipeline Gráfico:**



Pipeline Gráfico

A imagem final profuzida pelo Pipeline Gráfico é escrita em em uma matriz de pixels da GPU chamada **Frame Buffer,** a qual é lida pelo monitor

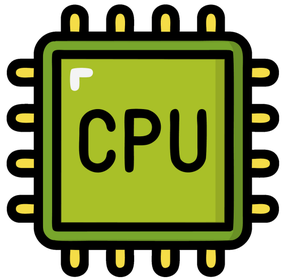


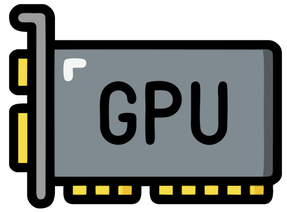
`Programa OpenGL para desenhar um retângulo na tela`

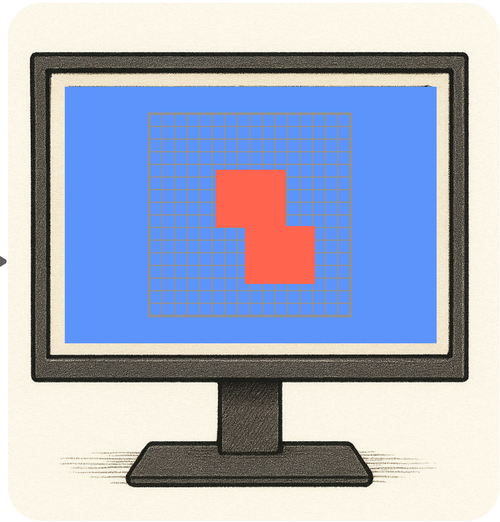
- Configura propriedades de renderização
- Carrega conjunto de vértices do retângulo
- Especifica shaders
- Compila e liga os shaders

Screen Tearing ("Rasgo na Tela")

Se um programa OpenGL (ex. um jogo) atualizar o frame buffer enquanto o monitor estiver atualizando a tela (ex. mover o retângulo para a direita), você pode ver um **rasgo na imagem:**





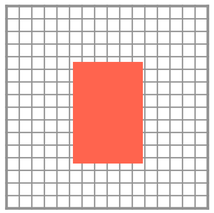


Programa OpenGL para desenhar um retângulo na tela

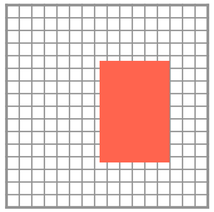
- Configura propriedades de renderização
- Carrega conjunto de vértices do retângulo
- Especifica shaders
- Compila e liga os shaders

Quadros

t_1



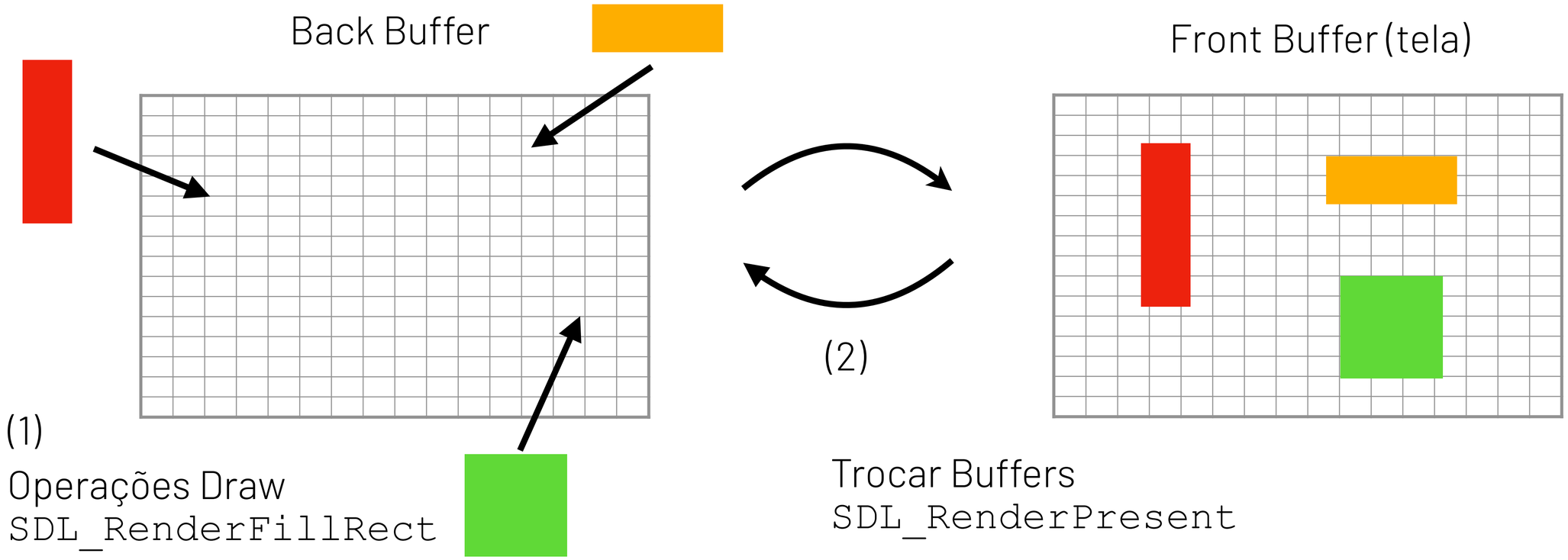
t_2



Monitor

Double Buffering

Jogos geralmente são renderizados usando **Double Buffering**, onde gráficos são (1) desenhados em um back buffer, que (2) é trocado com o front buffer quando o quadro inteiro foi desenhado



Próxima aula

A4: Gráficos II

- Visão geral de um programa OpenGL/GLSL
  - Compilando Shaders
  - Contexto OpenGL e Buffers
- Matemática para Jogos
  - Vetores
    - Soma, Multiplicação, Normalização, Produto Escalar, Produto Vetorial, ...
  - Transformações geométricas
    - Translação, Rotação e Escala
    - Coordenadas Homogêneas