Computação Gráfica

Pipeline Gráfico, Primitivas Geométricas, e OpenGL

Prof. Alaor Cervati Neto

2023/1

Pipeline Gráfico

Pipeline Gráfico

Modelo Conceitual:

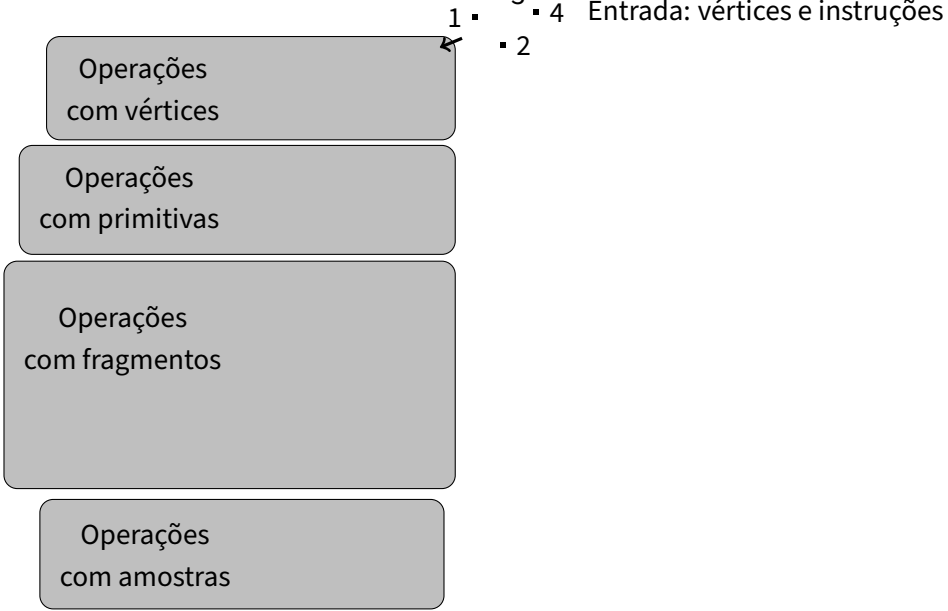
- Etapas envolvidas na renderização de um objeto.
- Objeto modelado em 2D ou 3D (via primitivas geométricas).

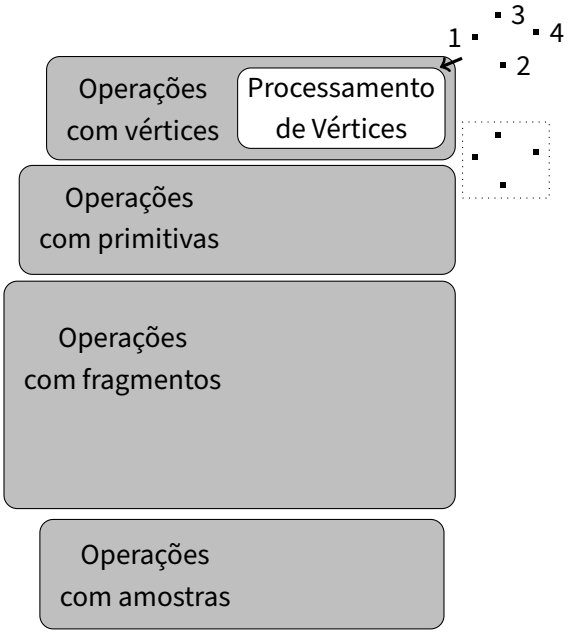
Operações
com vértices

Operações
com primitivas

Operações
com fragmentos

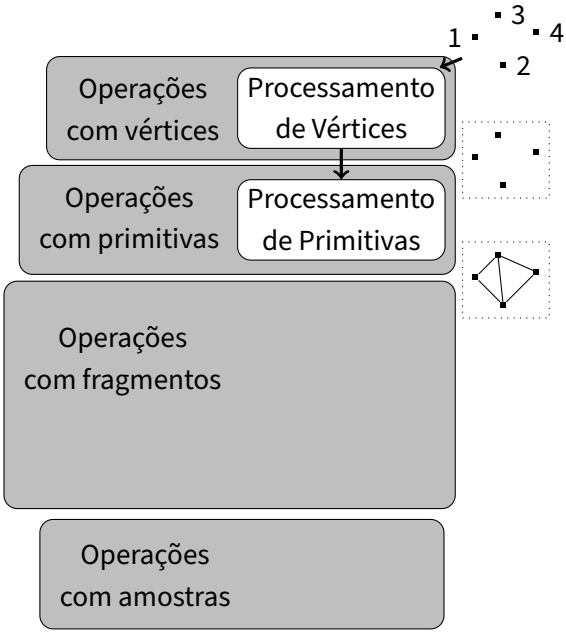
Operações
com amostras


1
2
3
4
Entrada: vértices e instruções
Operações
com vértices
Operações
com primitivas
Operações
com fragmentos
Operações
com amostras



Entrada: vértices e instruções

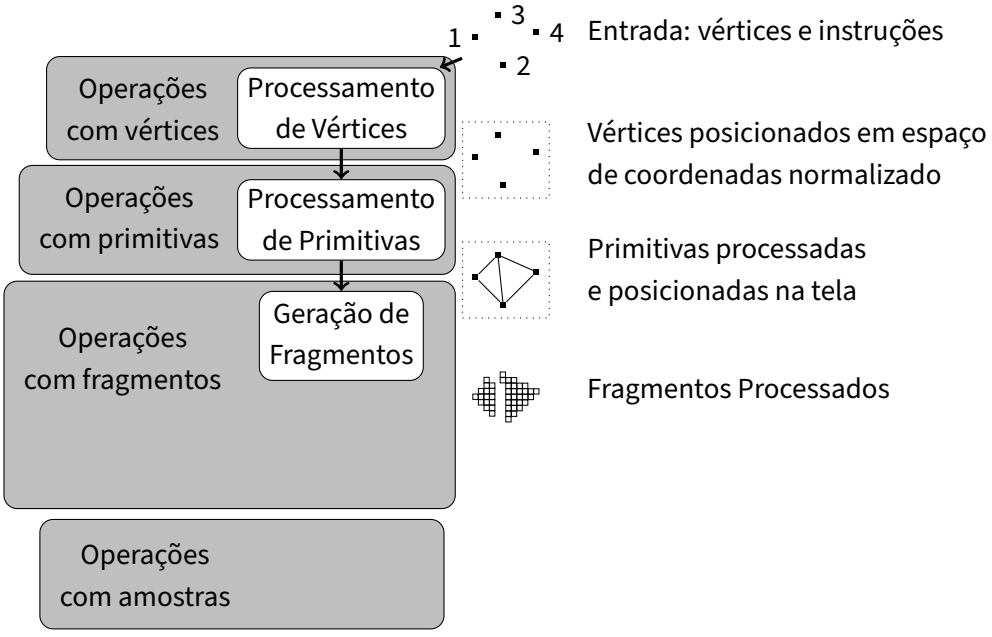
Vértices posicionados em espaço de coordenadas normalizado

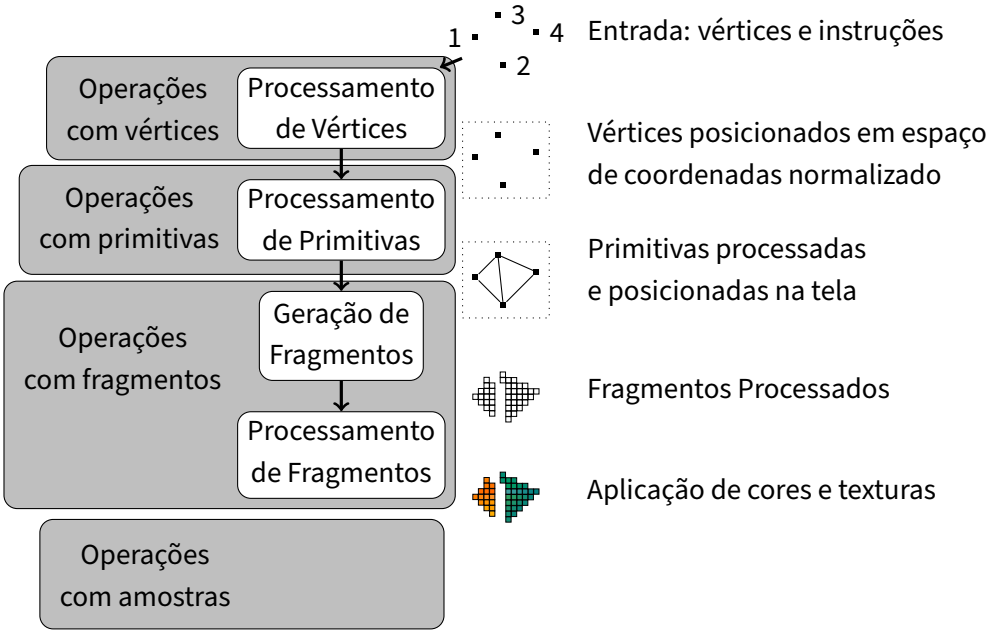


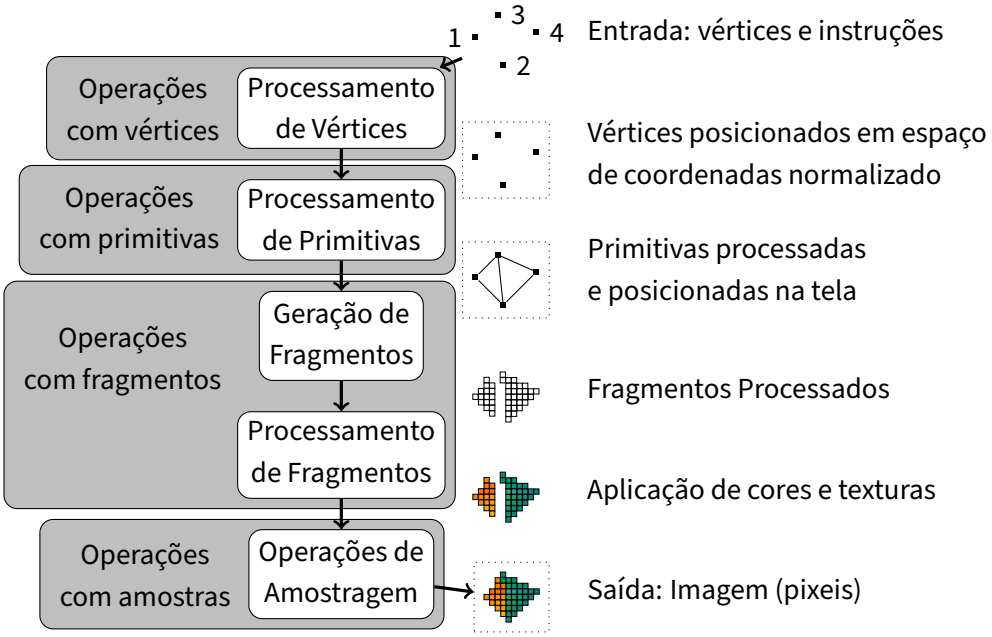
Entrada: vértices e instruções

Vértices posicionados em espaço de coordenadas normalizado

Primitivas processadas e posicionadas na tela


1
2
3
4
Entrada: vértices e instruções
Operações com vértices
Processamento de Vértices
Vértices posicionados em espaço de coordenadas normalizado
Operações com primitivas
Processamento de Primitivas
Primitivas processadas e posicionadas na tela
Operações com fragmentos
Geração de Fragmentos
Fragmentos Processados
Operações com amostras


1
2
3
4
Entrada: vértices e instruções
Operações com vértices
Processamento de Vértices
Vértices posicionados em espaço de coordenadas normalizado
Operações com primitivas
Processamento de Primitivas
Primitivas processadas e posicionadas na tela
Operações com fragmentos
Geração de Fragmentos
Fragmentos Processados
Processamento de Fragmentos
Aplicação de cores e texturas
Operações com amostras


1
2
3
4
Operações com vértices
Processamento de Vértices
Operações com primitivas
Processamento de Primitivas
Operações com fragmentos
Geração de Fragmentos
Processamento de Fragmentos
Operações com amostras
Operações de Amostragem
Entrada: vértices e instruções
Vértices posicionados em espaço de coordenadas normalizado
Primitivas processadas e posicionadas na tela
Fragmentos Processados
Aplicação de cores e texturas
Saída: Imagem (pixeis)

Pipeline Fixo × Pipeline Programável

Pipeline Fixo:

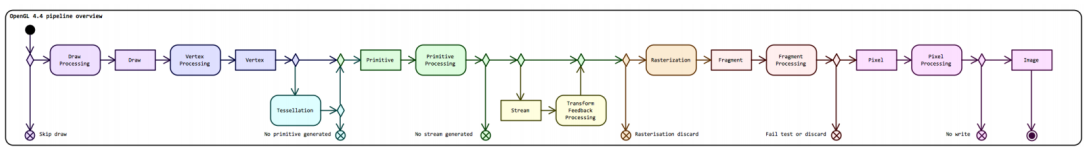
- Padrão até 2003, com renderização gráfica dependente de algoritmos internos de alguma GPU (tanto no OpenGL quanto no DirectX).
- Limitava um controle mais fino do processo. Atualmente obsoleto.

Pipeline Programável:

- A partir de 2004, inclusão de *Shaders* Programáveis.
- Permite programação com *Shader* de Vértices e *Shader* de Fragmentos.
- Linguagem de programação própria (exemplo: GLSL).

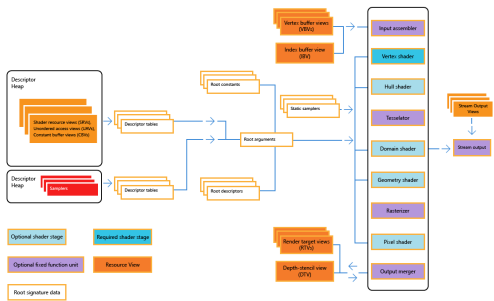
APIs para Computação Gráfica

- Exemplos: OpenGL, DirectX, WebGL, Vulkan.
- Cada uma implementa um Pipeline Gráfico.
- Exemplo: Pipeline Gráfico do OpenGL 4 (`https://www.seas.upenn.edu/~pcozzi/OpenGLInsights/OpenGL44PipelineMap.pdf`):



APIs para Computação Gráfica

- Exemplo: Pipeline Gráfico do Direct3D 12 (`https://docs.microsoft.com/en-us/windows/win32/direct3d12/pipelines-and-shaders-with-directx-12`):



APIs para Computação Gráfica

Neste curso: OpenGL.

- Modern Open GL (pipeline programável).
- API disponível para muitas linguagens de programação (`https://www.khronos.org/opengl/wiki/Language_bindings`).
- Documentação: `https://www.khronos.org/registry/OpenGL-Refpages/gl4/`.

Características do OpenGL:

- Multiplataforma (Linux, Windows, Mac, ...).
- Independente do Sistema de Janelas.

OpenGL

OpenGL

Um breve histórico

- OpenGL 1 lançada em 1994 (pipeline fixo).
- OpenGL 2 lançada em 2004 (*shaders* programáveis).
- OpenGL 3 adicionou mecanismo de remoção de funcionalidades obsoletas e compatibilidade.
  - OpenGL 3.2 incluiu *shaders* de geometria.
- OpenGL 4 adicionou novos *shaders* (tesselação).

Importante:

- OpenGL ES 3.2 (para dispositivos móveis).
- WebGL: implementação OpenGL ES em Javascript (`https://webglsamples.org/`).

Sistema de Janelas

Uma aplicação com OpenGL precisa de uma janela para renderizar a imagem. Alguns sistemas de janelas:

- GLUT (obsoleto): `https://www.opengl.org/resources/libraries/glut/`.
- FreeGLUT: `http://freeglut.sourceforge.net/`.
- GLFW: `https://www.glfw.org/`.
- QT: `https://www.qt.io/download-open-source`.
- SDL: `https://www.libsdl.org/`.

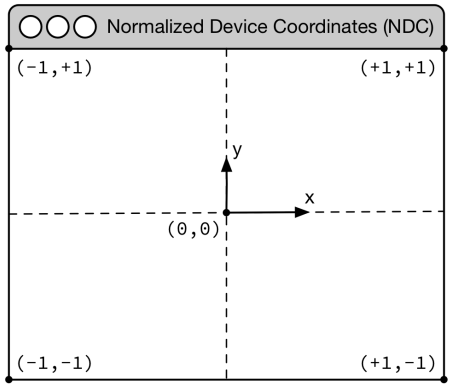
Janelas com OpenGL

Sistema normalizado de coordenadas

- Sistema cartesiano.
- A janela na qual o OpenGL é processado utiliza coordenadas no intervalo $-1 \leq x, y, z \leq 1$.
- O centro corresponde à coordenada $(0, 0, 0)$.

Janelas com OpenGL

Sistema normalizado de coordenadas



Primitivas Geométricas

Primitivas Geométricas

Para renderizar alguma imagem, precisamos utilizar Primitivas Geométricas.

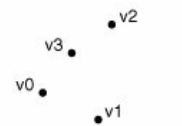
- Elementos gráficos simples que combinados permitem construir objetos complexos.
- 2D: ponto, linha, polilinha, polígono, elipse, arco.
- 3D: cubo, esfera, cilindro, cone, tubo.

Vértices são o ponto de partida.

- Representados por 4 coordenadas: (x, y, z, w).
- Motivo: coordenadas homogêneas.

Pontos e Linhas

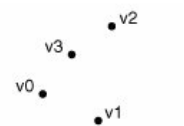
Há três formas de renderizar linhas. Considere o conjunto de vértices:

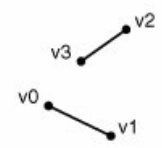


- GL_LINES: Renderiza a cada dois vértices:
  - $(\texttt{v0}, \texttt{v1})$, $(\texttt{v2}, \texttt{v3})$
- GL_LINE_STRIP: Renderiza de forma encadeada:
  - $(\texttt{v0}, \texttt{v1})$, $(\texttt{v1}, \texttt{v2})$, $(\texttt{v2}, \texttt{v3})$
- GL_LINE_LOOP: Renderiza de forma encadeada até atingir o primeiro vértice:
  - $(\texttt{v0}, \texttt{v1})$, $(\texttt{v1}, \texttt{v2})$, $(\texttt{v2}, \texttt{v3})$, $(\texttt{v3}, \texttt{v0})$

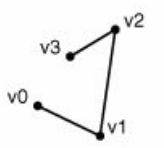
Pontos e Linhas

Há três formas de renderizar linhas. Considere o conjunto de vértices:

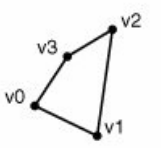




GL_LINES



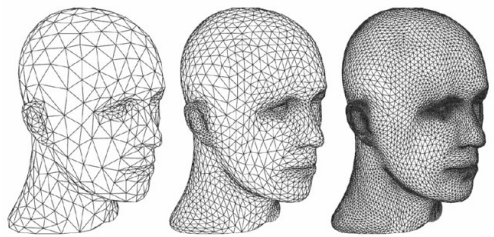
GL_LINE_STRIP



GL_LINE_LOOP

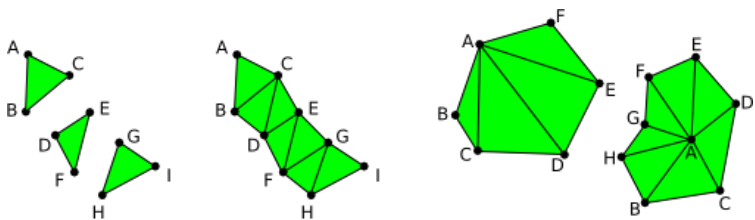
Triângulos

- Triângulos são primitivas importantes para a computação gráfica moderna.
- GPU divide malhas em pequenos triângulos antes de desenhá-las. Esse processo é chamado tesselação, tecelagem ou mosaico.



Triângulos

- Triângulos são primitivas importantes para a computação gráfica moderna.
- GPU divide malhas em pequenos triângulos antes de desenhá-las. Esse processo é chamado tesselação, tecelagem ou mosaico.
- Algumas primitivas para triângulos:

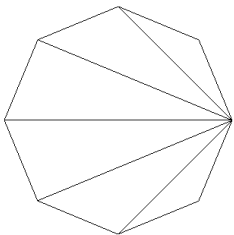


Quadrados

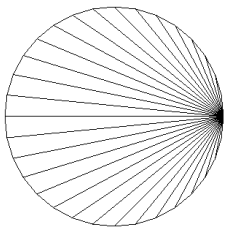
- Quadrados podem ser desenhados com dois triângulos.
- Modo mais prático usando GL_TRIANGLE_STRIP.
- Apenas quatro vértices são suficientes.

Círculos

- Círculos podem ser desenhados usando vários triângulos:



8 vértices

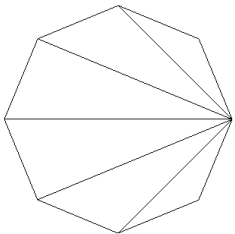


32 vértices

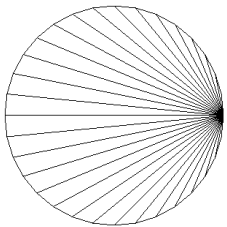
- Primitiva: GL_TRIANGLE_FAN.
- Problema: determinar as coordenadas dos vértices.

Círculos

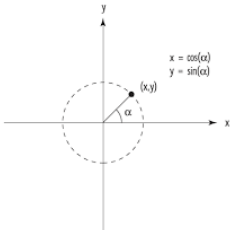
- Círculos podem ser desenhados usando vários triângulos:



8 vértices



32 vértices



- Primitiva: GL_TRIANGLE_FAN.
- Problema: determinar as coordenadas dos vértices.

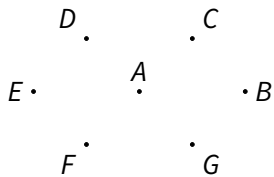
Material de base para a aula

- INTERACTIVE COMPUTER GRAPHICS. Stanford CS248 Course. Kayvon Fatahalian, 2018, `http://graphics.stanford.edu/courses/cs248-18-spring/courseinfo`.
- SHREINER, Dave et al. OpenGL programming guide: The Official guide to learning OpenGL, version 4.3. Addison-Wesley, 2013.
- Computação Gráfica: Aula 02. Slides de Ricardo M. Marcacini. Disciplina SCC0250/0650, ICMC/USP, 2021.
- Colaboração do monitor Matheus da Silva Araújo (Discord: `MatheusAraujo#6468`).

Exercícios

Exercícios

Considerando os vértices:



Utilize uma única primitiva para desenhar as seguintes figuras:

1. Um hexágono contendo todos os vértices.
2. Dois triângulos que não se tocam.
3. Um quadrado.