UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Introdução ao OpenGL em C

Disciplina: Computação Gráfica (BCC35F)
Curso: Ciência da Computação

Prof. Walter T. Nakamura
waltertakashi@utfpr.edu.br

Campo Mourão - PR

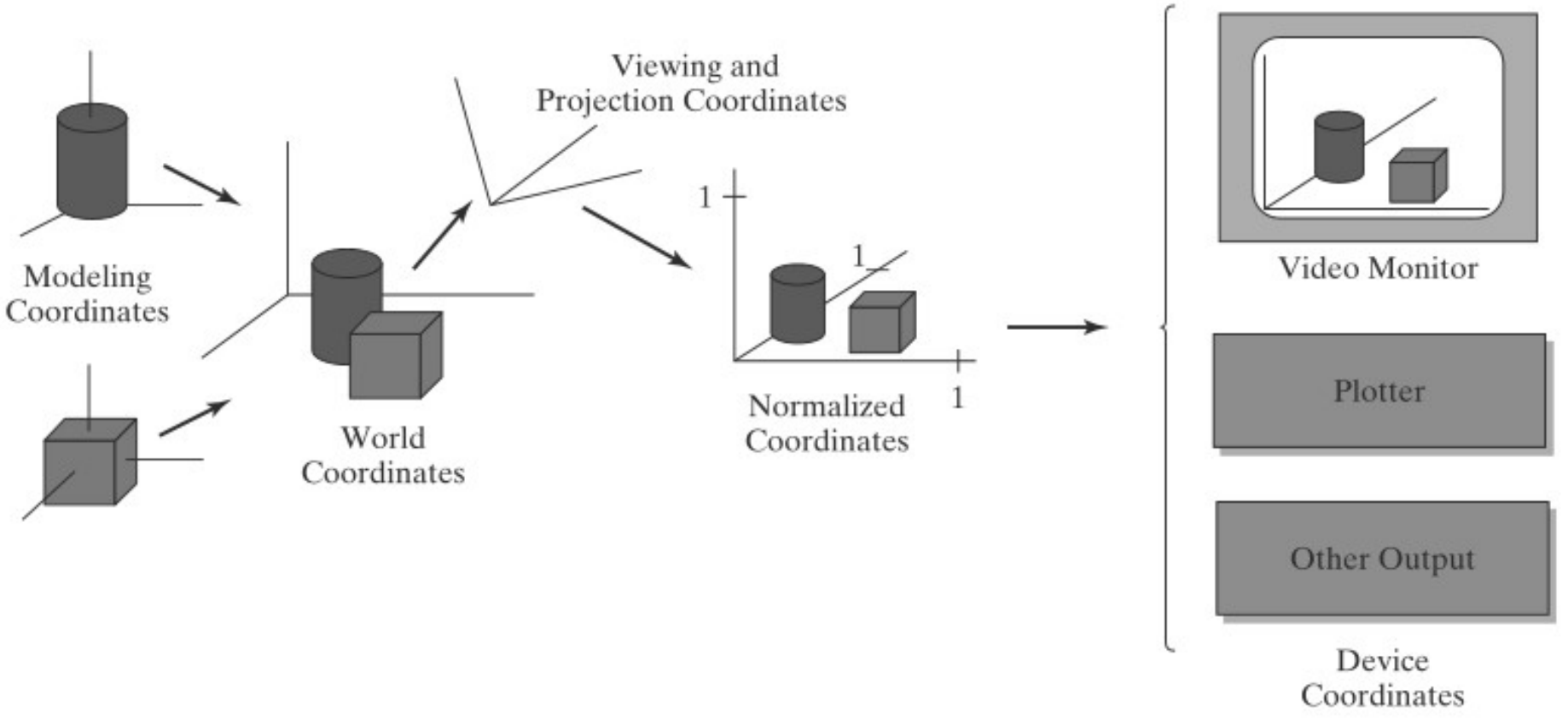
Baseados nos materiais elaborados pelas professoras Aretha Alencar (UTFPR) e Rosane Minghim (USP)

Representação de coordenadas

- Para gerar uma imagem é preciso fornecer a **descrição** dos objetos geométricos
  - Local e formato dos objetos
- Diversos **planos cartesianos** são utilizados para exibir uma cena
  - Coordenadas de modelagem ou coordenadas locais
  - Coordenadas do mundo
  - **Exemplo:** construção de uma bicicleta
- Após todas as partes da cena serem especificadas, as coordenadas do mundo são processadas e exibidas em um dispositivo de saída (**viewing pipeline**)

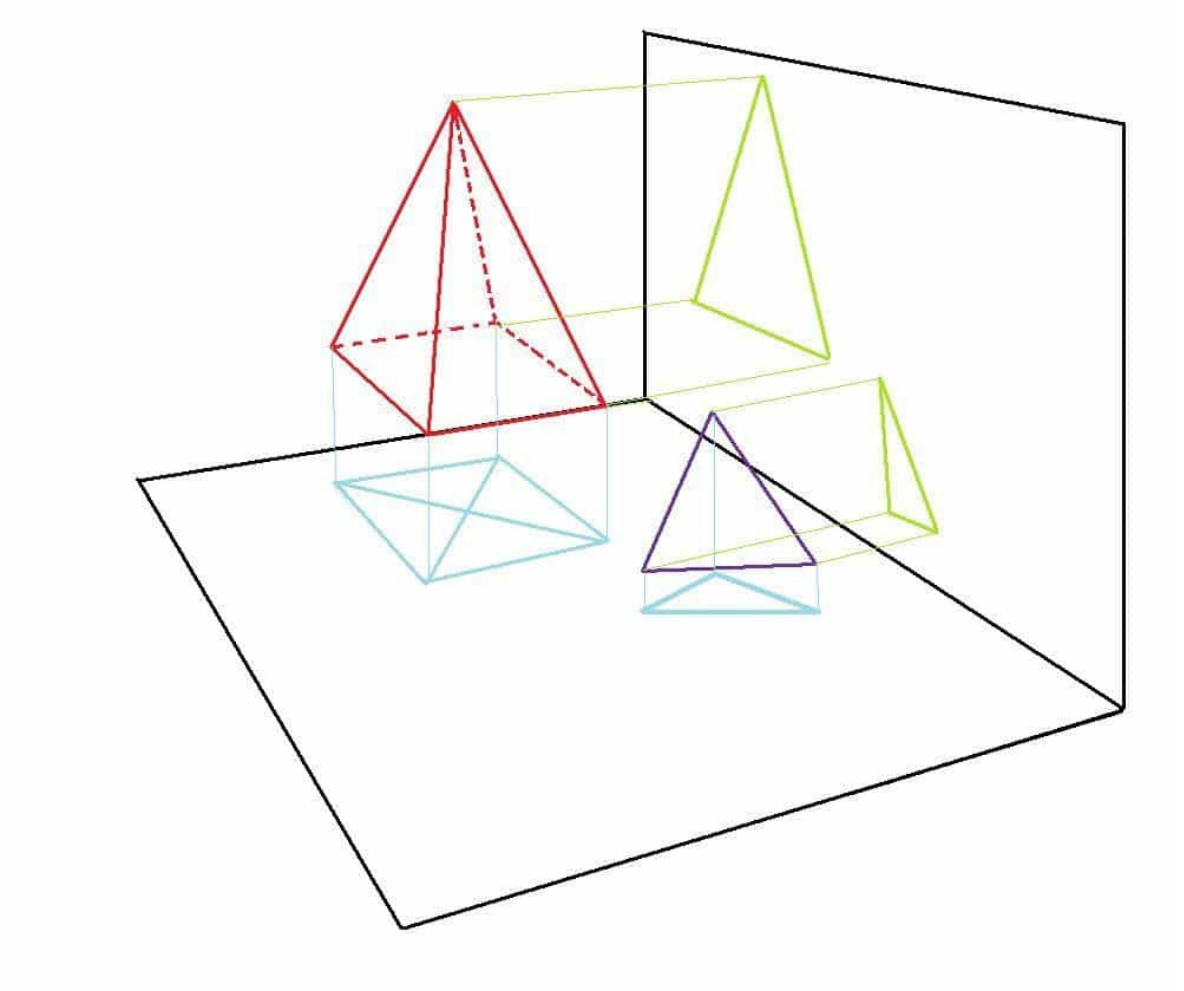
Representação de coordenadas

- O local dos objetos é transformado em uma **projeção 2D** da cena correspondente ao que veremos no dispositivo de saída



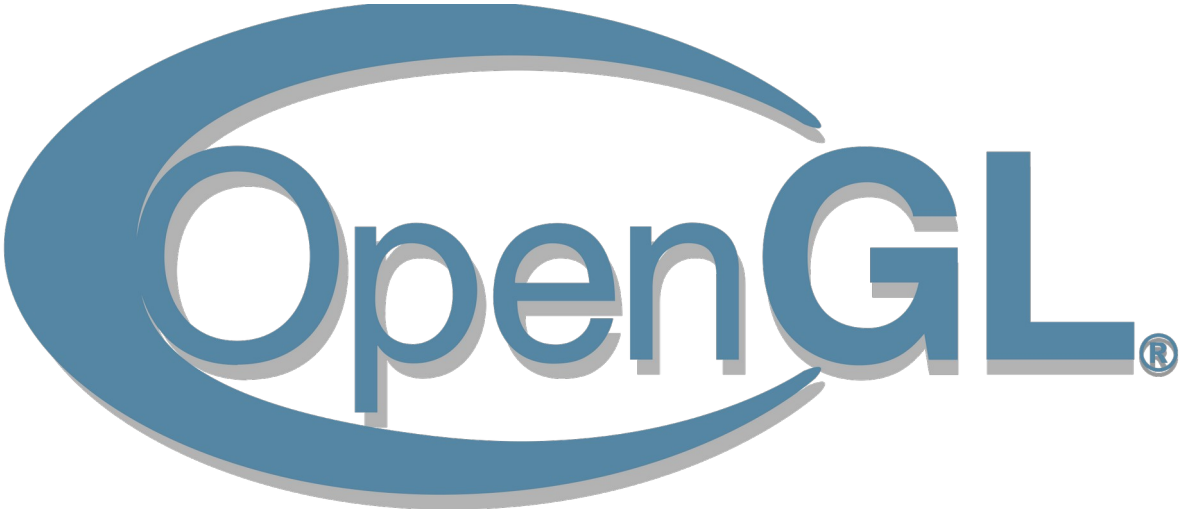
Representação de coordenadas

- O local dos objetos é transformado em uma **projeção 2D** da cena correspondente ao que veremos no dispositivo de saída



Funções gráficas

- Um pacote gráfico fornece diversas funções para criar e manipular imagens
  - **Primitivas gráficas:** blocos de construção básicas
    - Curvas, linhas, áreas coloridas, formas
  - **Atributos:** descreve como uma primitiva será exibida
    - Cores, estilo de linha e de texto, padrões de preenchimento
  - **Transformações geométricas:** altera o tamanho, posição e orientação em uma cena
  - **Transformações de visualização:** seleciona uma visão da cena, tipo de projeção e local da visão no monitor
  - **Funções de entrada:** controla e processa os dados de dispositivos interativos
  - **Operações de controle:** realiza tarefas “domésticas”


OpenGL®

Instalação OpenGL no Windows

- Baixar o instalador do **MinGW** (clique aqui)
  - (a) Instalar em `C:\MinGW` (`<MINGW_HOME>`)
  - (b) Na instalação do MinGW, instalar os pacotes básicos "`mingw32-base`", "`mingw32-gcc-g++`" e "`msys-base`"
    - Clicar em cada pacote e escolher "Mark for installation"
    - Ir em "Installation" no menu superior → Apply changes

Instalação OpenGL no Windows

- Para instalar a biblioteca **GLUT** no MinGW:
  - Faça o download da biblioteca GLUT (clique aqui)
  - Descompacte o arquivo baixado no passo anterior. Copie o arquivos que estão em "`include\GL`" para "`<MINGW_HOME>\include\GL`"
  - Os arquivos da pasta "`lib`" para "`<MINGW_HOME>\lib`"
  - Os arquivos da pasta "`bin`" para "`<MINGW_HOME>\bin`"

Instalação OpenGL no Linux

- Instalar os seguintes pacotes na sua distribuição Linux:
  - `freeglut3`
  - `freeglut3-dev`
  - `make`
  - `gcc`
- Instalar o CodeBlocks ou Visual Studio Code, caso goste de usar uma IDE para compilar seus programas em C

Diretivas de compilação

- As seguintes diretivas de compilação devem ser usadas no seu projeto OpenGL, dependendo do sistema operacional sendo usado:
  - **Windows:** "`-lopengl32 -lglu32 -lfreeglut`"
  - **Linux:** "`-lGL -lglut -lGLU -lm`"
  - **Exemplo:**
    - `gcc -o exemplo exemplo.c -lGL -lglut -lGLU -lm`

Configuração de IDEs: CodeBlocks

- Baixar e Instalar o CodeBlocks (versão sem MinGW) (http://www.codeblocks.org)
- Configurar as diretivas de compilação no CodeBlocks:
  - Settings → Compiler → Aba “Linker Settings”
  - Colocar em “Other linker options”:
    - Se for Windows:
      - `"-lopengl32 -lglu32 -lfreeglut -lm"`
    - Se for Linux:
      - `"-lGL -lglut -lGLU -lm"`

Configuração de IDEs: Visual Studio Code

- No Visual Studio Code:
  - Criar uma pasta para o projeto e abrir
  - Ir em "Terminal" → "Configure default build task"
  - Escolher o compilador "`C/C++:g++.exe`"
  - Na janela que abrir (`tasks.json`), incluir no parâmetro "`args`":
    - Se for Windows:
      - `"-lopengl32", "-lglu32", "-lfreeglut", "-lm"`
    - Se for Linux:
      - `"-lGL", "-lglut", "-lGLU", "lm"`
  - Salvar o arquivo de configuração (Ctrl + S)

Introdução

- **Application Programming Interface (API)**
  - Coleção de rotinas que o programador pode chamar
  - Modelo de como estas rotinas **operam** em conjunto para gerar gráficos
  - Programador “enxerga” apenas a interface
  - Não precisa lidar com aspectos específicos do **hardware** ou peculiaridades de **software** no sistema gráfico residente (independente do dispositivo)
  - Oferece suporte para gerar e exibir **cenas 3D** complexas, e também para **gráficos 2D** simples

OpenGL

- Ambiente p/ escrever e executar programas gráficos
  - Monitor (tela) + biblioteca de software para desenhar primitivas gráficas na tela
- API pode ser vista como uma **“caixa preta”**:
  - **Entradas:** Chamadas a funções da biblioteca feitas pelo programa do usuário; Medidas fornecidas por dispositivos de entrada; etc.
  - **Saídas:** Os gráficos exibidos no monitor
  - Descrita em termos das funções que disponibiliza

API OpenGL

- **Programa**
  - Em geral, trabalha com um sistema de janelas (window system)
  - **Inicializações:** modo de exibição (display mode), janela de desenho e sistema de coordenadas de referência (associado à janela)
- API oferece centenas de **funções**
  - Diferentes funcionalidades:
    - **Funções primitivas:** o que?
    - **Funções de atributos:** como?
    - ...

API OpenGL

- Direcionada a **eventos** (event-driven)
  - **Programa responde a eventos:** clique do mouse, tecla pressionada, redimensionamento da janela, ...

Programação em OpenGL

- OpenGL rastreia diversas **variáveis de estado**:
  - Tamanho atual de um ponto, cor de fundo da janela, cor do desenho, etc.
- O **valor corrente** permanece ativo até que seja alterado:
  - **Tamanho de ponto:** `glPointSize(3.0)`
  - **Cor de desenho:** `glColor3f(red, green, blue)`
  - **Cor de fundo:** `glClearColor(red, green, blue, alpha)`
  - **Limpar janela:** `glClear(GL_COLOR_BUFFER_BIT)`
    - Os bits do color buffer serão modicados para a cor de fundo

Estados do OpenGL

- OpenGL é utilizada junto com outras bibliotecas auxiliares:
  - **OpenGL Utility (GLU):** define a visão, matrizes de projeção, aproximação poligonal, desenho de superfícies, etc.
  - **OpenGL Utility Toolkit (GLUT):** define o sistema de janelas, e outras funções de desenho de superfície

Estrutura básica de um programa em OpenGL

- O arquivo `glut.h` contém os protótipos das funções utilizadas pelo programa, e também inclui os headers `gl.h` e `glu.h` (OpenGL e GLU)
- Somente inclua o arquivo de cabeçalho `windows.h`, se estiver usando o Windows

```
#include <windows.h>
#include <GL/glut.h>

int init(void){
/* inserir codigo */
}

void display(void){
/* inserir codigo */
}

int main(int argc, char** argv){
/* inserir codigo */
}
```

Estrutura básica de um programa em OpenGL

- Precisamos de três funções principais:
  - **init()** – inicializa os parâmetros do rendering
  - **display()** – função callback chamada para fazer o desenho
  - **main()** – função principal

```
#include <windows.h>
#include <GL/glut.h>

int init(void){
/* inserir codigo */
}

void display(void){
/* inserir codigo */
}

int main(int argc, char** argv){
/* inserir codigo */
}
```

Programando em OpenGL: “Hello World”

- Gerenciamento da janela de exibição usando o GLUT

```
int main(int argc, char** argv){
    glutInit(&argc, argv);  // inicializa o GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB); // define o modo de display
    glutInitWindowPosition(200,0); // posição inicial da janela
    glutInitWindowSize(400,300); // largura e altura da janela
    // cria a janela de exibição
    glutCreateWindow("Exemplo de programa OpenGL");

    init(); // chamam a função de inicialização
    // define a função display() como a função callback de exibição
    glutDisplayFunc(display);
    glutMainLoop();// mostre tudo e espere
}
```

Programando em OpenGL: “Hello World”

- Define o método de inicialização:

```
int init(void){
    // define a cor de fundo
    glClearColor(1.0, 1.0, 1.0, 0.0);
    // carrega a matriz de projeção
    glMatrixMode(GL_PROJECTION);
    // define projeção ortogonal 2D
    gluOrtho2D(0.0, 200.0, 0.0, 150.0);
}
```

Programando em OpenGL: “Hello World”

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

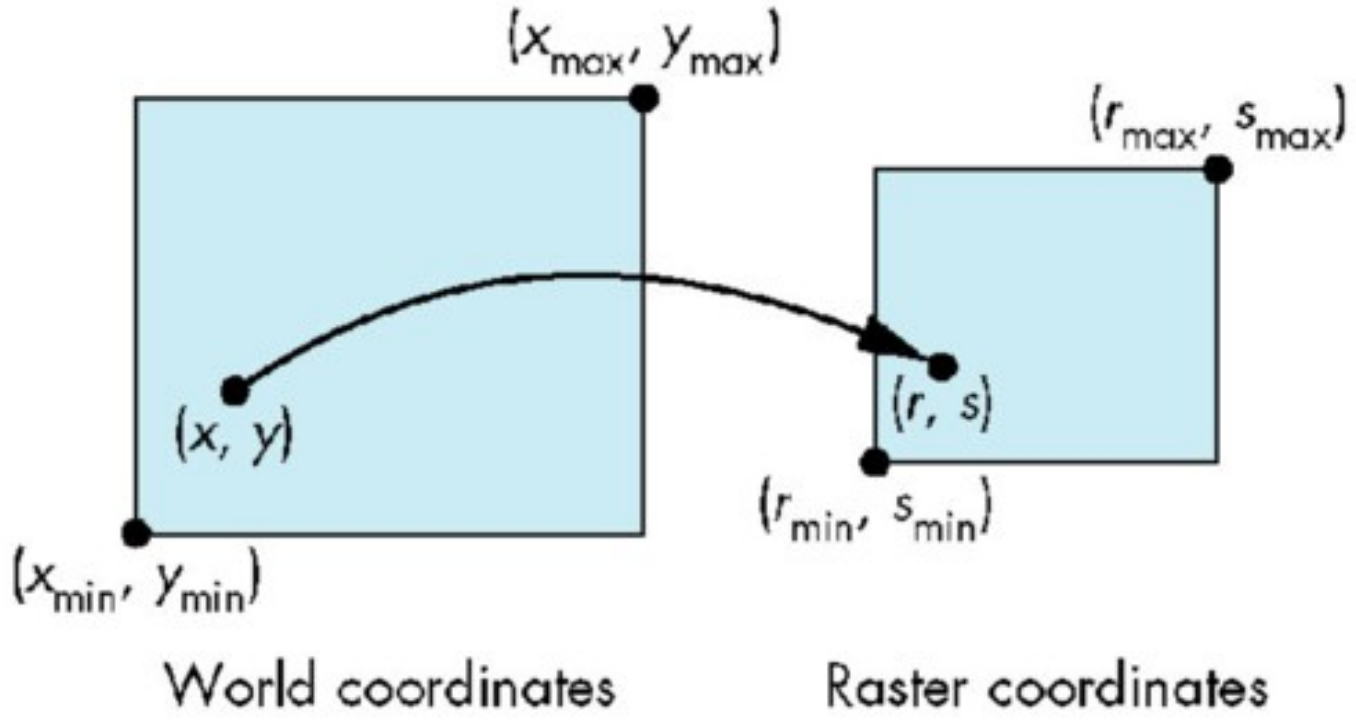
- Define o método de desenho:

```
void display(void){
    // desenha o fundo (limpa a janela)
    glClear(GL_COLOR_BUFFER_BIT);

    glColor3f(1.0, 0.0, 0.0); // altera o atributo de cor
    glBegin(GL_LINES); // desenha uma linha
        glVertex2i(180, 15);
        glVertex2i(20, 15);
    glEnd();
    // esvazia o buffer, enviando todos os comandos emitidos
    // até o momento para a GPU, sem esperar pelo render
    glFlush();
}
```

Programando em OpenGL

- O pipeline da OpenGL é sempre 3D, mas é possível criar desenhos 2D, definindo:
  - glMatrixMode(GL_PROJECTION)
  - gluOrtho2D(xw_min, xw_max, yw_min, yw_max)



Programamando em OpenGL

- OpenGL permite desenhar gráficos de modo independente do dispositivo
- É possível especificar o Sistema de Coordenadas do Mundo (ponto flutuante) onde os objetos são definidos
- Os elementos são traçados no Sistema de Coordenadas do Dispositivo, ou Sistema de Coordenadas da Tela (inteiro)
  - O mapeamento entre esses sistemas de coordenadas é feito de forma transparente pelo OpenGL.

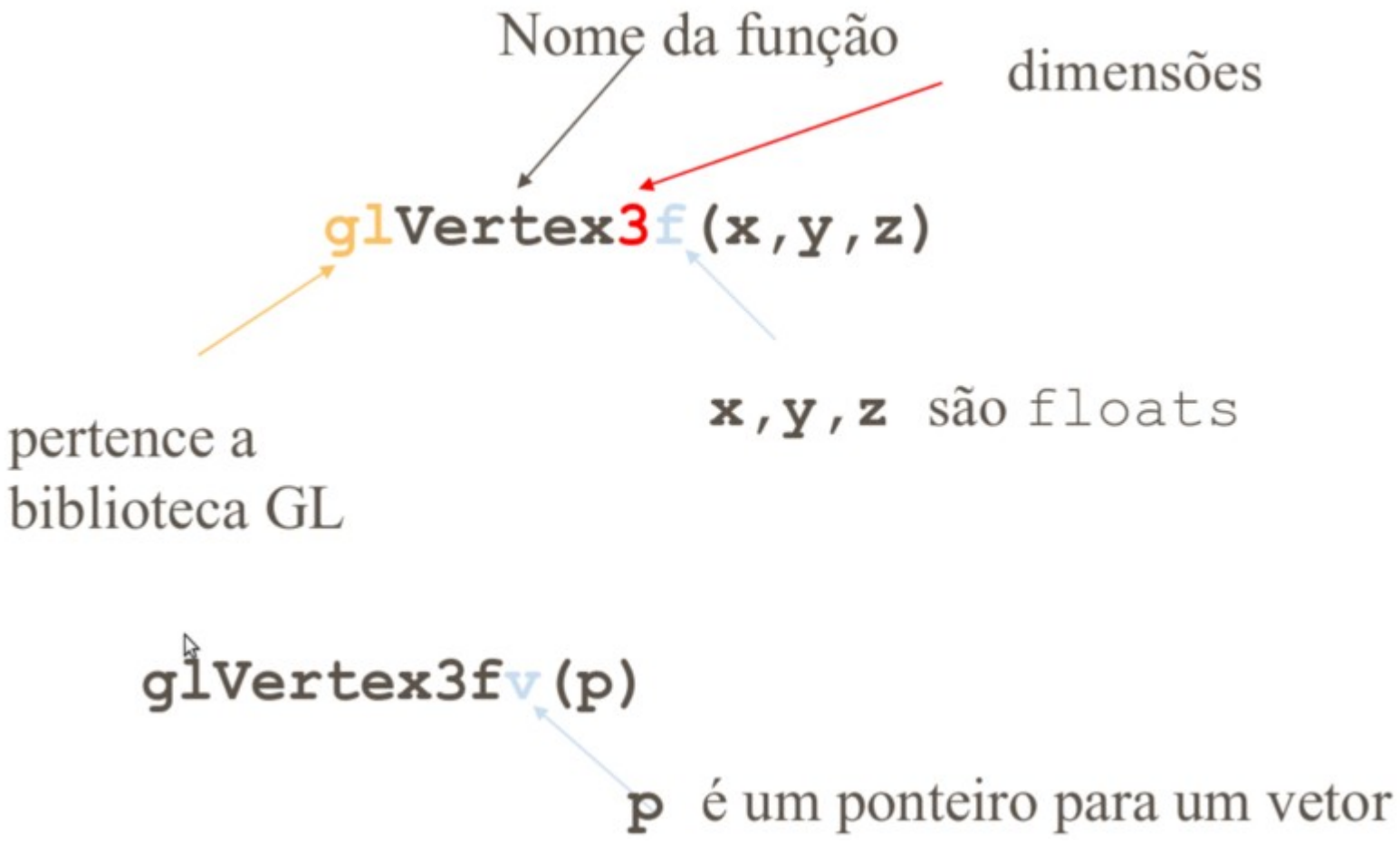
Primitivas gráficas

- Primitivas básicas
  - Pontos, linhas, polilinhas, polígonos
    - `GL_POINTS`, `GL_LINES`, `GL_POLYGON`, etc.
  - Para descrever um objeto, uma lista de vértices é informada:

```
glBegin(GL_POINTS); // desenha 3 pontos
    glVertex2i(100, 50);
    glVertex2i(100, 130);
    glVertex2i(150, 130);
glEnd();
```

Primitivas gráficas

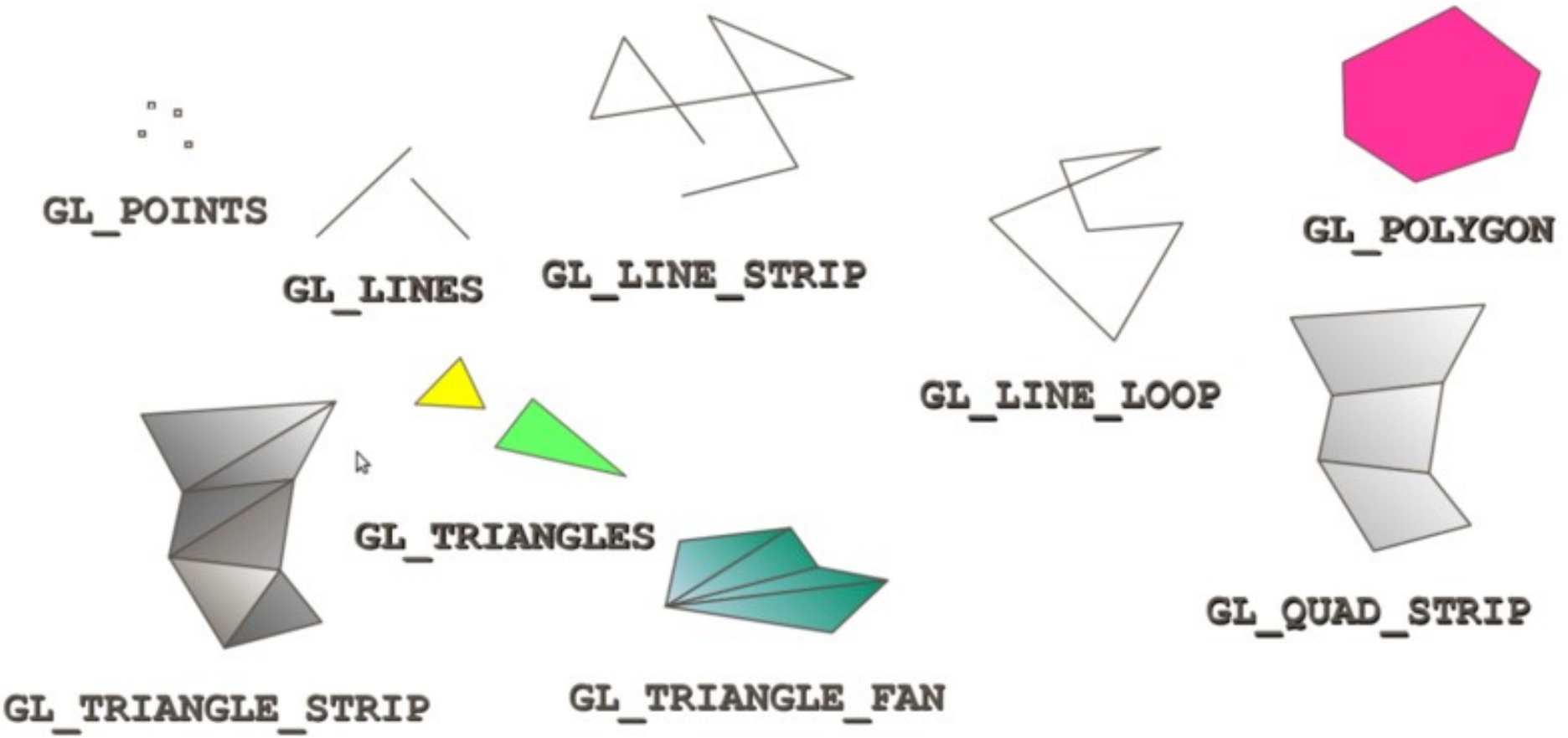
- Convenções na nomenclatura das funções



Outras primitivas gráficas

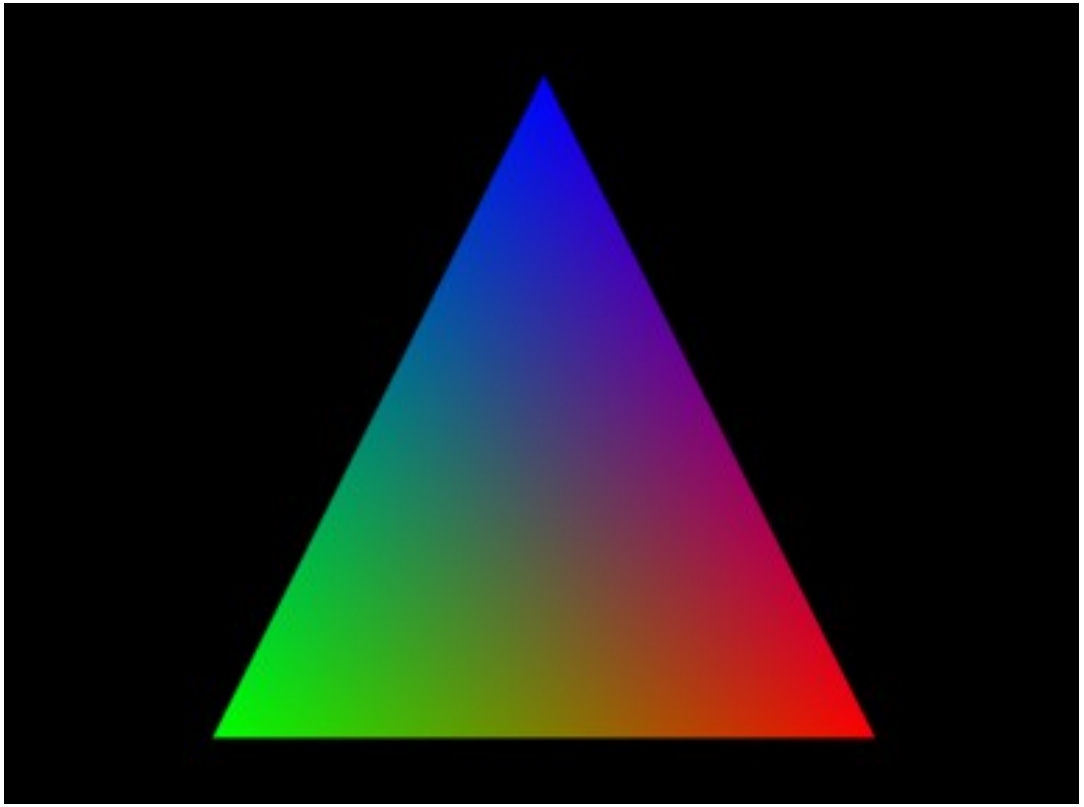
- **Polilinha:** sequência de linhas conectadas, fechada ou não
  - `GL_LINE_STRIP, GL_LINE_LOOP`
- **Outras primitivas:**
  - `GL_TRIANGLES`
  - `GL_QUADS`
  - `GL_TRIANGLE_STRIP`
  - `GL_TRIANGLE_FAN`
  - `GL_QUAD_STRIP`

Outras primitivas gráficas



Exercício 1

- Fazer um programa que desenhe o triangulo 2D abaixo.
- Documentação do OpenGL:
  - https://www.opengl.org/sdk/docs/man



Exercício 2

- Crie um programa que exiba uma imagem similar ao exemplo abaixo:

Exercícios

- 3) Crie um programa que exiba uma estrela de 5 pontas.
- 4) Identifique uma forma de inicializar a janela da aplicação OpenGL do exercício 3 em tela cheia.
- 5) Crie um programa que desenhe um círculo na matriz de projeção.