# OpenGL: Introdução


**Professor:** Odilon Corrêa da Silva

**Curso: Engenharia de Computação**

**Disciplina: Computação Gráfica**

# OpenGL: Introdução

- Introdução
  - OpenGL é uma biblioteca com rotinas gráficas de modelagem bi (2D) e tridimensional (3D).
  - Os nomes das funções da biblioteca OpenGL seguem uma convenção e são divididos em quatro partes:

    *<PrefixoBiblioteca> <ComandoRaiz> <ContadorArgumentosOpcional> <TipoArgumentosOpcional>*

    - Por exemplo:

      *void glColor3f(GLfloat red, GLfloat greem, GLfloat blue);*

| Argumento | Descrição |
|---|---|
| gl | é o prefixo que representa a biblioteca gl |
| Color | é o comando raiz que indica o objetivo da função |
| 3 | é o contador para o número de argumentos que a função possui |
| f | indica que os argumentos são valores de ponto flutuante |

Fonte: https://www.inf.pucrs.br/~manssour/OpenGL/NomesFuncoes.html

# OpenGL: Introdução

- Introdução
  - Cada tipo de dado OpenGL está associado a um tipo de dado na linguagem de programação C/C++:

| Sufixo | Tipo de dado C/C++ | Tipo de dado OpenGL |
|---|---|---|
| b | signed char | GLbyte |
| s | short | GLshort |
| i | int ou long | GLint, GLsizei |
| f | float | GLfloat, GLclampf |
| d | double | GLdouble, GLclampd |
| ub | unsigned char | GLubyte, GLboolean |
| us | unsigned short | GLushort |
| ui | unsigned long ou unsigned int | GLuint, GLenum, GLbitfield |

Fonte: https://www.inf.pucrs.br/~manssour/OpenGL/TiposDados.html

# OpenGL: Introdução

- Definição do espaço de trabalho
  - Para entender como funciona o processo de visualização bidimensional em OpenGL é importante conhecer o conceito de “universo”, que pode ser definido como a região do plano (ou espaço) utilizado em uma aplicação. Como a descrição geométrica de um modelo normalmente envolve coordenadas geométricas, é preciso adotar um sistemas de referência que irá definir uma origem em relação à qual todos os posicionamentos do universo são descritos.

# OpenGL: Introdução

- Definição do espaço de trabalho
  - Sistema de Referência do Universo (SRU)
    - Consiste em um plano cartesiano com dois eixos (*x* e *y*) conforme a Figura 1. Todos os modelos e comandos do OpenGL são definidos em relação a este sistema de referência.
  - Sistema de Referência da Tela ou Dispositivo (SRT ou SRD)
    - No monitor do computador é adotado o SRT que possui algumas diferenças em relação ao SRU (Figura 2).

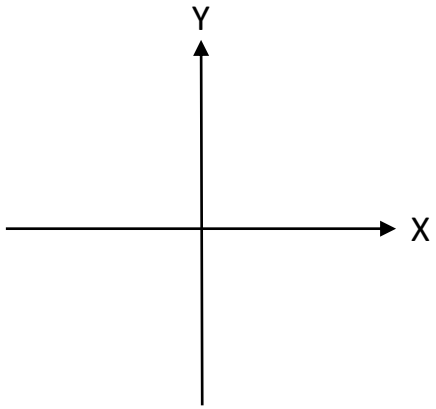



*Figura 01 – Plano cartesiano do SRU*

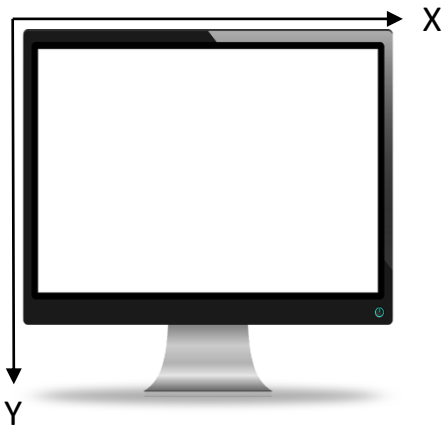



*Figura 02 – SRT*

# OpenGL: Introdução

- Definição do espaço de trabalho
  - Conversão ou Mapeamento
    - Para possibilitar a correta visualização na tela dos modelos definidos no SRU, é necessário fazer uma conversão ou mapeamento (Figura 3).

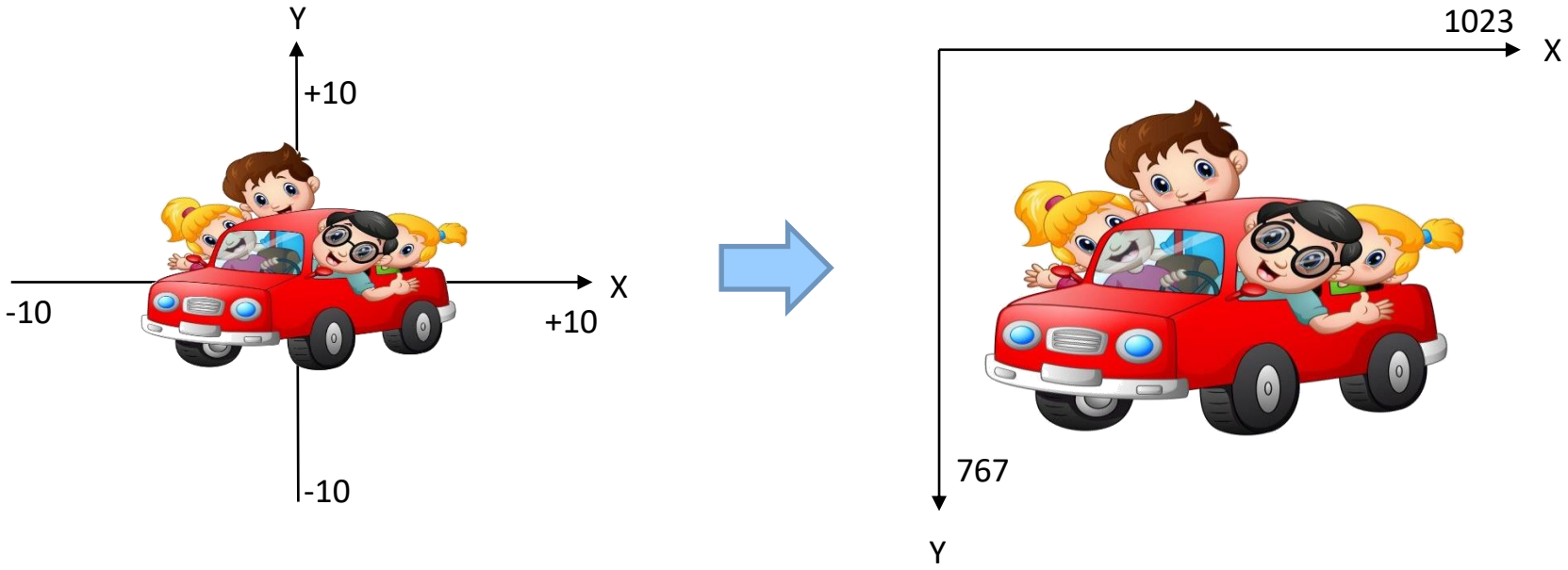



*Figura 03 – Mapeamento do SRU para o SRT*

# OpenGL: Introdução

- Definição do espaço de trabalho
  - Janela de seleção e exibição
    - A área do universo que delimita a região de interesse do usuário é chamada de *window* ou janela de seleção. Uma *window* é delimitada pelas coordenadas de seus cantos (esquerda, direita, superior e inferior), as quais sempre são dadas em valores que dizem respeito ao SRU.
    - É necessário definir em que parte do monitor deseja-se exibir o conteúdo da *window*. Chamamos essa região de *viewport* ou *janela de exibição*.
    - Em OpenGL uma *viewport* é normalmente delimitada pelo tamanho da janela (GLUT), que é definido sempre em valores que dizem respeito ao SRT.

# OpenGL: Introdução

- Definição do espaço de trabalho
  - Janela de seleção e exibição
    - A Figura 4 ilustra os conceitos de *window* e *viewport*.

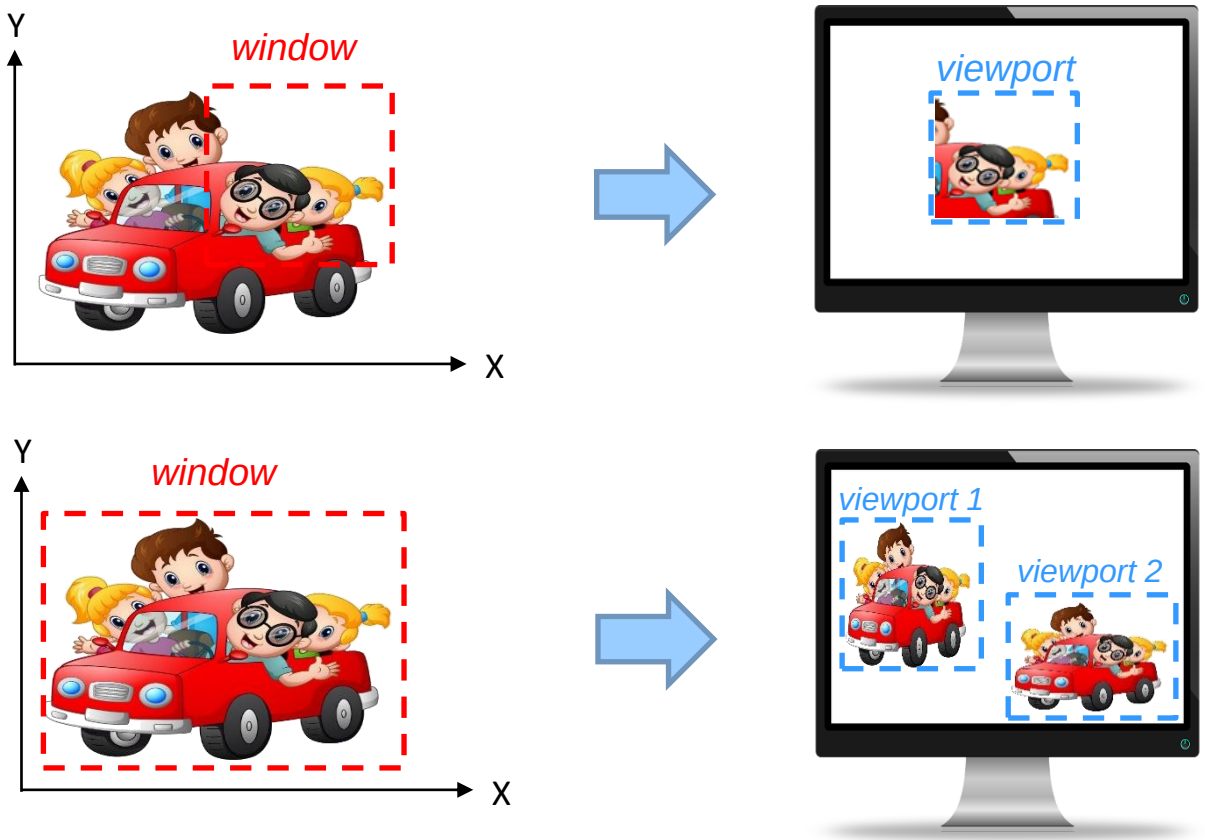


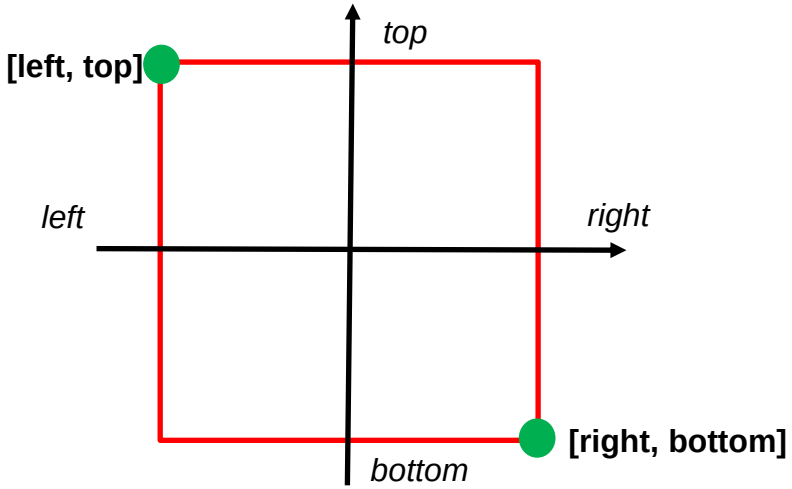

*Figura 04 – Window e ViewPort*

# OpenGL: Introdução

- Definição do espaço de trabalho
  - Janela de seleção e exibição
    - É possível realizar o mapeamento do SRU para o SRT ao especificar a *viewport* e *window* em OpenGL utilizando as funções:
      - A função gluOrtho2D determina que a projeção ortográfica será utilizada para exibir na tela a imagem 2D que está na janela de seleção

        *void gluOrtho2D(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top);*

        - (*left*, *right*) especificam os limites mínimo e máximo no eixo X.
        - (*bottom*, *top*) especificam os limites mínimo e máximo no eixo Y

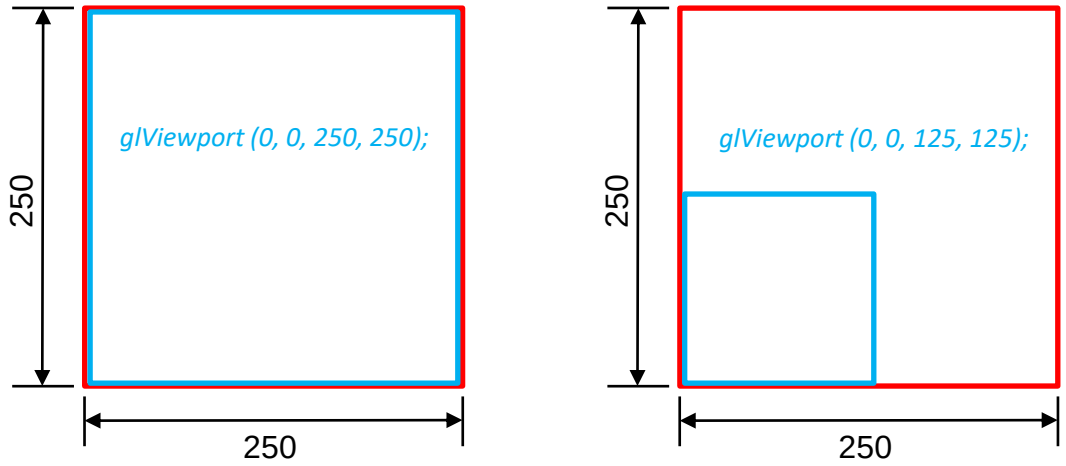



Fonte: https://www.inf.pucrs.br/~manssour/OpenGL/Desenhando.html

# OpenGL: Introdução

- Definição do espaço de trabalho
  - Janela de seleção e exibição
    - É possível realizar o mapeamento do SRU para o SRT ao especificar a *viewport* e *window* em OpenGL utilizando as funções:
      - A função *glViewport* é usada para definir a *viewport*

        *void glViewport (GLint x, GLint y, GLsizei width, GLsizei height);*
        - (*x*, *y*) especificam o canto inferior esquerdo em relação à janela na tela
        - (*width*, *height*) especificam a largura e altura em pixels

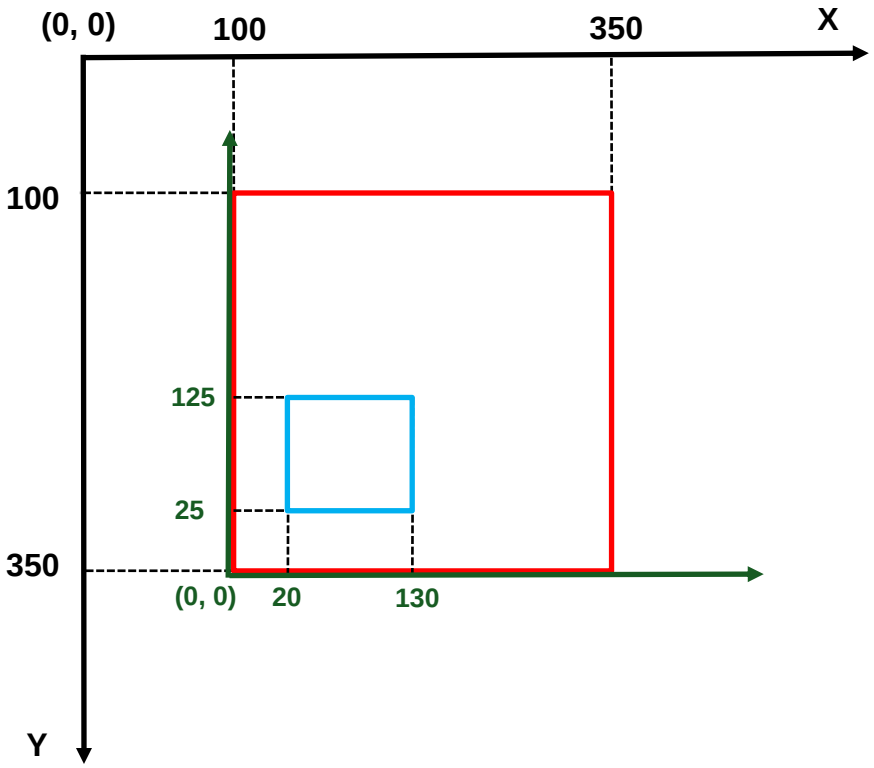



Fonte: http://edeleastar.github.io/opengl-programming/topic01/pdf/3.OpenGL_Fundamentals.pdf

# OpenGL: Introdução

- Definição do espaço de trabalho
  - Janela de seleção e exibição
    - Exemplo

```
glutInitWindowSize(250, 250);

glutInitWindowPosition(100, 100);

gluOrtho2D (0, 250, 0, 250);

glViewport(20, 25, 110, 100);
```




Fonte: https://shms-prod.s3.amazonaws.com/media/editor/142610/Computer_Graphics_courses_Part_II-net.pdf

# OpenGL: Introdução

- Primitivas gráficas
  - Com apenas algumas primitivas simples, tais como pontos, linhas e polígonos, é possível criar estruturas complexas. Em outras palavras, objetos e cenas criadas com OpenGL consistem em simples primitivas gráficas que podem ser combinadas de várias maneiras.
  - OpenGL fornece ferramentas para desenhar pontos, linhas e polígonos, que são formados por um ou mais vértices. Neste caso, é necessário passar uma lista de vértices entre duas chamadas de funções OpenGL.
  - Por exemplo:

```
glBegin(GL_POINTS);
    glColor3f(0.0f, 0.0f, 0.0f);
    glVertex2i(100, 50);
    glVertex2i(100, 130);
    glVertex2i(150, 130);
glEnd();
```

Fonte: https://www.inf.pucrs.br/~manssour/OpenGL/LinhaPontoPolig.html

# OpenGL: Introdução

- Primitivas gráficas
  - Podem ser passadas como argumento para a função *glBegin* uma das constantes abaixo:

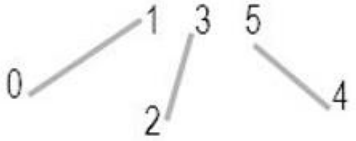
| Valor | Descrição |
|---|---|
| GL_POINTS | exibe um ponto para cada chamada ao comando glVertex |
| GL_LINES | exibe uma linha a cada dois comandos glVertex |
| GL_LINE_STRIP | exibe uma sequência de linhas conectando os pontos definidos por glVertex |
| GL_LINE_LOOP | exibe uma sequência de linhas conectando os pontos definidos por glVertex e ao final liga o primeiro como último ponto |
| GL_POLYGON | exibe um polígono convexo preenchido, definido por uma seqüência de chamadas a glVertex |
| GL_TRIANGLES | exibe um triângulo preenchido a cada três pontos definidos por glVertex |
| GL_TRIANGLE_STRIP | exibe uma sequência de triângulos baseados no trio de vértices v0, v1, v2, depois, v2, v1, v3, depois, v2, v3, v4 e assim por diante |
| GL_TRIANGLE_FAN | exibe uma sequência de triângulos conectados baseados no trio de vértices v0, v1, v2, depois, v0, v2, v3, depois, v0, v3, v4 e assim por diante |
| GL_QUADS | exibe um quadrado preenchido conectando cada quatro pontos definidos por glVertex; |
| GL_QUAD_STRIP | exibe uma sequência de quadriláteros conectados a cada quatro vértices; primeiro v0, v1, v3, v2, depois, v2, v3, v5, v4, depois, v4, v5, v7, v6, e assim por diante |

Fonte: https://www.inf.pucrs.br/~manssour/OpenGL/LinhaPontoPolig.html

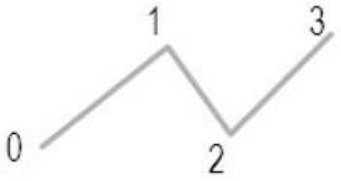
# OpenGL: Introdução

- Primitivas gráficas
  - Tipos de primitivas em OpenGL:

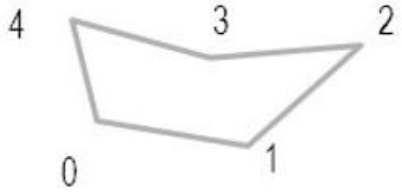
GL_POINTS



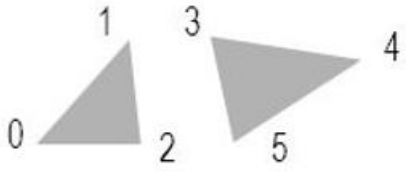
GL_LINES



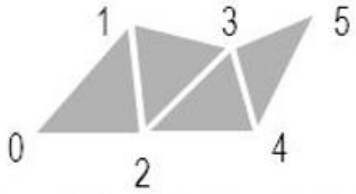
GL_LINE_STRIP



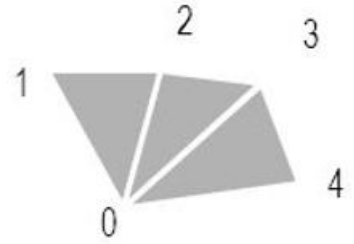
GL_LINE_LOOP



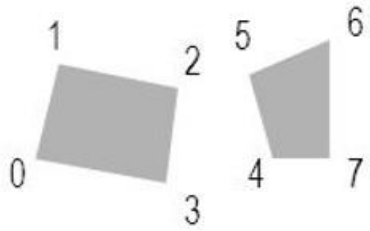
GL_TRIANGLES



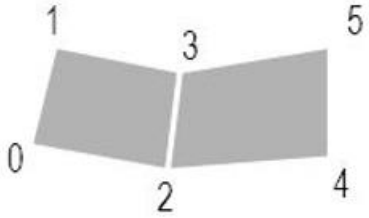
GL_TRIANGLE_STRIP



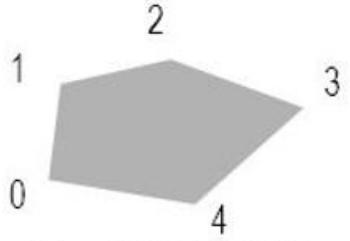
GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP

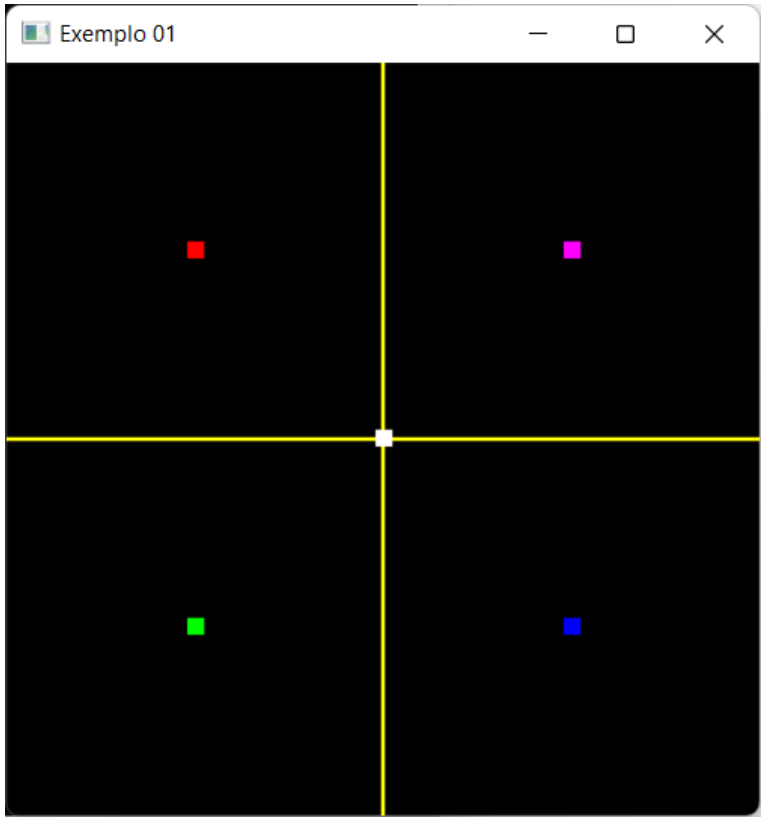


GL_POLYGON
*(convexo)*

# OpenGL: Introdução

- Primitivas gráficas
  - Exemplo 01

```
glutInitWindowPosition(100, 100);
glutInitWindowSize(400, 400);

glViewport(0, 0, 400, 400);
gluOrtho2D (-40.0f, 40.0f, -40.0f, 40.0f);

glLineWidth(2.0f);

glBegin(GL_LINES);
   glColor3f(1.0f, 1.0f, 0.0f); //cor amarela
   glVertex2i(0, 40); // linha do eixo Y
   glVertex2i(0, -40);
   glVertex2i(-40, 0);// linha do eixo X
   glVertex2i(40, 0);
glEnd();

glPointSize(9.0f);
glBegin(GL_POINTS);
   glColor3f(1.0f, 0.0f, 0.0f); // ponto vermelho
   glVertex2i(-20, 20);
   glColor3f(0.0f, 1.0f, 0.0f); // ponto verde
   glVertex2i(-20, -20);
   glColor3f(0.0f, 0.0f, 1.0f); // ponto azul
   glVertex2i(20, -20);
   glColor3f(1.0f, 0.0f, 1.0f); // ponto rosa
   glVertex2i(20, 20);
   glColor3f(1.0f, 1.0f, 1.0f); // ponto branco
   glVertex2i(0, 0);
 glEnd();
```

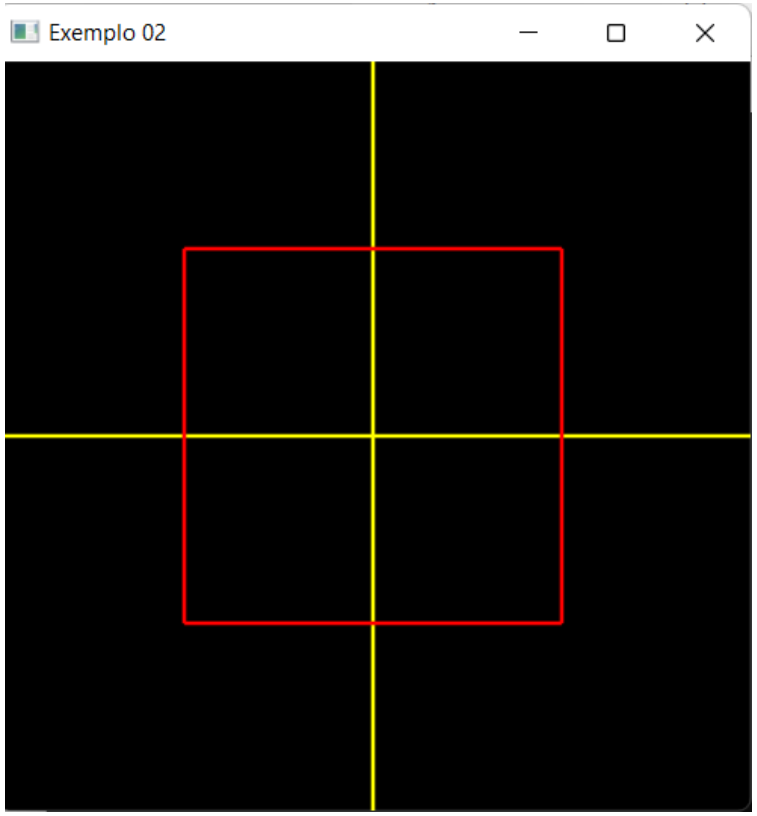



Projeto: Exemplo01

# OpenGL: Introdução

- Primitivas gráficas
  - Exemplo 02

```
glutInitWindowPosition(100, 100);
glutInitWindowSize(400, 400);

glViewport(0, 0, 400, 400);
gluOrtho2D (-40.0f, 40.0f, -40.0f, 40.0f);

glLineWidth(2.0f);

glBegin(GL_LINES);
   glColor3f(1.0f, 1.0f, 0.0f); //cor amarela
   glVertex2i(0, 40); // linha do eixo Y
   glVertex2i(0, -40);
   glVertex2i(-40, 0);// linha do eixo X
   glVertex2i(40, 0);
glEnd();

glBegin(GL_LINE_LOOP);
   glColor3f(1.0f, 0.0f, 0.0f); // vermelho em RGB
   glVertex2i(-20, 20);
   glVertex2i(-20, -20);
   glVertex2i(20, -20);
   glVertex2i(20, 20);
glEnd();
```

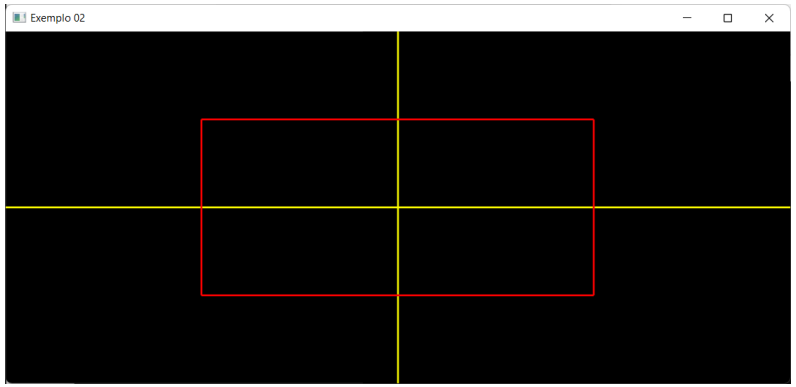


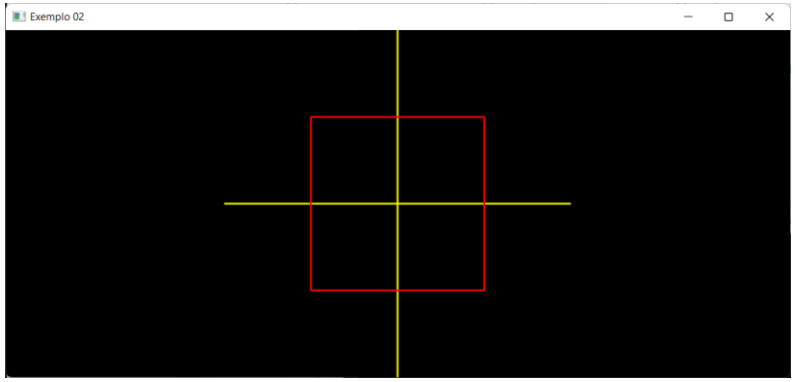

Projeto: Exemplo02

# OpenGL: Introdução

- Primitivas gráficas
  - Exemplo 02

```
glViewport(0, 0, 400, 400);
gluOrtho2D (-40.0f, 40.0f, -40.0f, 40.0f);
```

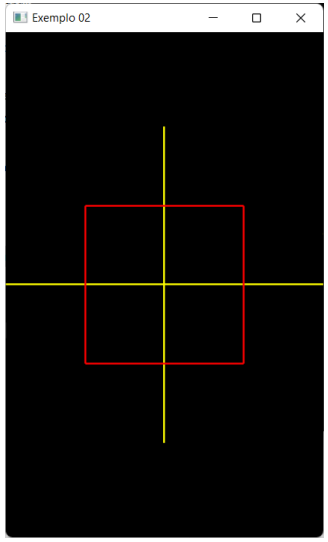


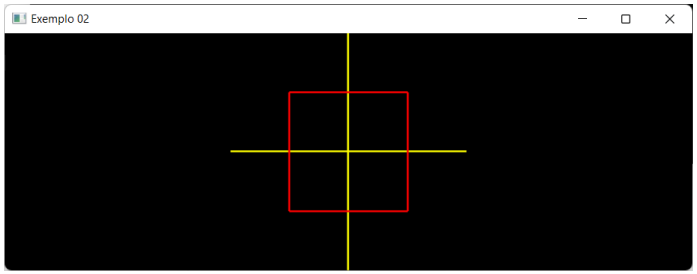

```
glViewport(0, 0, largura, altura);

if (largura <= altura)
    gluOrtho2D (-40.0f, 40.0f, -40.0f*altura/largura, 40.0f*altura/largura);
else
    gluOrtho2D (-40.0f*largura/altura, 40.0f*largura/altura, -40.0f, 40.0f);
```

# OpenGL: Introdução

- Primitivas gráficas
  - Exemplo 02
    - É possível manter a correção de aspecto caso a janela tenha seu tamanho alterado. A multiplicação pela razão entre largura e altura (relação de aspecto) garante que a *window* será definida de maneira que a imagem final não seja deformada.




Largura 400
Altura 1000
gluOrtho2D (-40.0, 40.0, -80, 80);

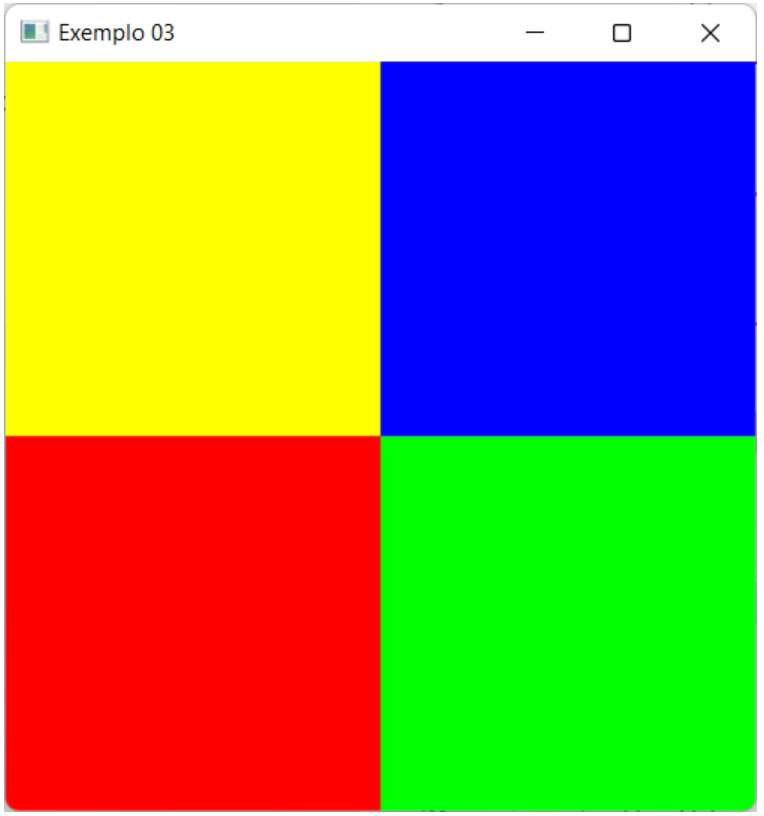



Largura: 1000
Altura: 400
gluOrtho2D(-80, 80, -40.0, 40.0);

# OpenGL: Introdução

- Primitivas gráficas
  - Exemplo 03

```
void desenhaQuadrado(void){
    glBegin(GL_QUADS);
      glVertex2i(-40, 40);
      glVertex2i(-40, -40);
      glVertex2i(40, -40);
      glVertex2i(40, 40);
    glEnd();
}

//Viewport 1
glViewport(0, 0, auxLargura/2, auxAltura/2);
glColor3f(1.0f, 0.0f, 0.0f); // vermelho em RGB
desenhaQuadrado();

//Viewport 2
glViewport(auxLargura/2, 0, auxLargura/2, auxAltura/2);
glColor3f(0.0f, 1.0f, 0.0f); // verde em RGB
desenhaQuadrado();

//Viewport 3
glViewport(auxLargura/2, auxAltura/2, auxLargura/2, auxAltura/2);
glColor3f(0.0f, 0.0f, 1.0f); // azul em RGB
desenhaQuadrado();

//Viewport 4
glViewport(0, auxAltura/2, auxLargura/2, auxAltura/2);
glColor3f(1.0f, 1.0f, 0.0f); // amarelo em RGB
desenhaQuadrado();
```




Projeto: Exemplo03

Viewport 1: (0, 0, 200, 200);
Viewport 2: (200, 0, 200, 200);
Viewport 3: (200, 200, 200, 200);
Viewport 4: (0, 200, 200, 200);

# OpenGL: Introdução

- Sugestão de videoaula
  - Projeção - glOrtho ( Desenhando um Quadrado)
    - https://www.youtube.com/watch?v=FwOw70_2UAc
  - Redimensionamento de Janela - glViewport
    - https://www.youtube.com/watch?v=VXtROZCAOXU

- Sugestão de leitura
  - Capítulo 2 do livro “Computação gráfica: geração de imagens” do livro do Azevedo e Conci
  - Capítulos 3, 4, 7 e 8 do livro “OpenGL - Uma Abordagem Prática e Objetiva” do Cohen e Manssour

# OpenGL: Introdução

- Referência Bibliográfica
  - AZEVEDO, Eduardo; CONCI, Aura. Computação gráfica: geração de imagens. Rio de Janeiro: Campus, Elsevier, 2003
  - COHEN, Marcelo; MANSSOUR, Isabel. OpenGL - Uma Abordagem Prática e Objetiva. São Paulo: Novatec, 2006. 486 p.