Computação Gráfica (IME 04-10842)
2022.2

Introdução ao OpenGL

Gilson. A. O. P. Costa (IME/UERJ)

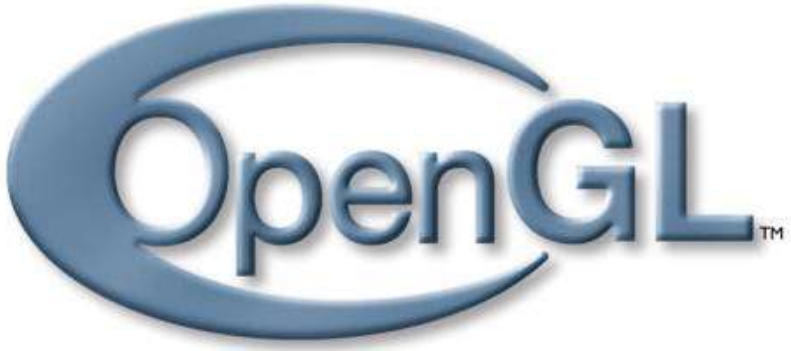
gilson.costa@ime.uerj.br

Conteúdo

- Aplicações
- Instalação
- Arquitetura
- Sintaxe
- Primitivas de Desenho
- APIs Complementares
- Callbacks
- Window × Viewport
- Exemplos

OpenGL

- OpenGL (Open Graphics Library) é uma API livre utilizada na computação gráfica, para desenvolvimento de aplicativos gráficos, ambientes 3D, jogos, entre outros.



- Desenvolvimento iniciado na década de 90 pela Silicon Graphics, Inc.
- Desde 2006 a biblioteca é mantida pelo consórcio chamado Khronos Group.

OpenGL

Promoter Members

OpenGL

Aplicações:

http://www.rage.com/

OpenGL

Aplicações:

https://quake.bethesda.net/

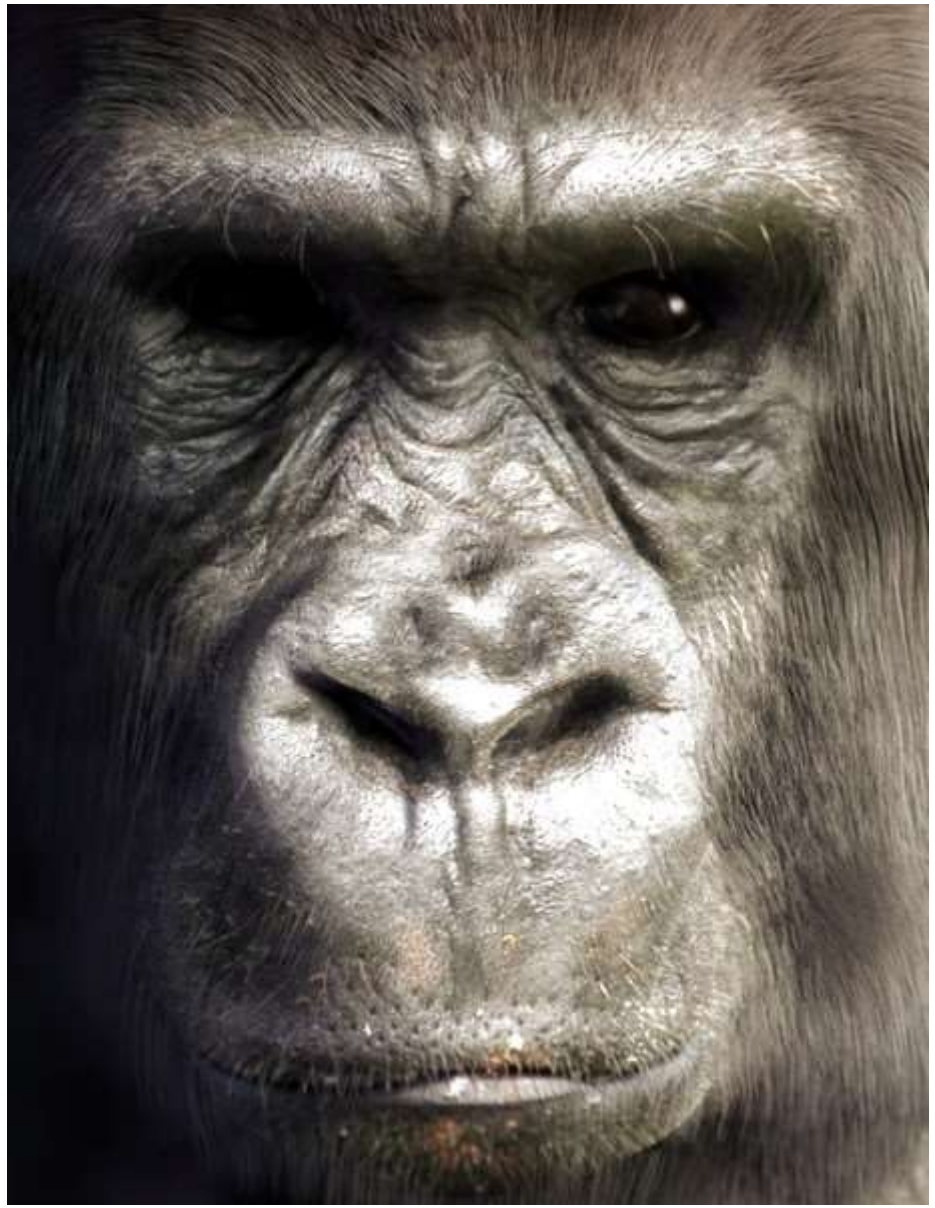
OpenGL

Aplicações:

http://www.aerofly.de/

OpenGL

Aplicações:



http://www.blender.org/

Instalação

Ubuntu (Debian):

- http://www.codebind.com/linux-tutorials/install-opengl-ubuntu-linux/
- https://medium.com/geekculture/a-beginners-guide-to-setup-opengl-in-linux-debian-2bfe02ccd1e

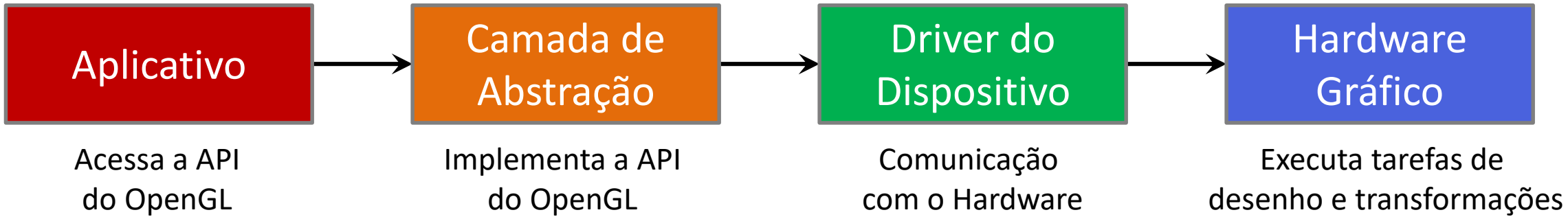
Windows:

- https://medium.com/swlh/setting-opengl-for-windows-d0b45062caf

Visual Studio:

- https://www.geeksforgeeks.org/how-to-setup-opengl-with-visual-studio-2019-on-windows-10/
- https://www.youtube.com/watch?v=XOtY4yzitdk&ab_channel=HamzahAsyrani

Arquitetura



- Independente de plataforma.
- Comumente implementado de forma a tirar partido da aceleração gráfica (se disponível).
- Não gerencia janelas nem trata eventos produzidos por dispositivos de interação.

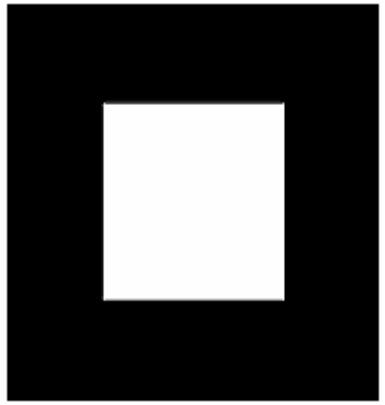
Arquitetura

- Permite a criação de descrições matemáticas de objetos a partir de primitivas geométricas (pontos, linhas e polígonos) e imagens/bitmaps.
- Permite a organização dos objetos no espaço e a seleção de um ponto de vista adequado para a cena.
- Converte as descrições matemáticas + cores em pixels (rasterização).
- Calcula as cores dos objetos por:
  - Atribuição direta.
  - Modelos de iluminação.
  - Mapeamentos de texturas.
  - Combinações.

Sintaxe OpenGL

Exemplo:

```
#include <whateverYouNeed.h>
main() {
   InitializeAWindowPlease();

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();
   glFlush();

   UpdateTheWindowAndCheckForEvents();
}
```



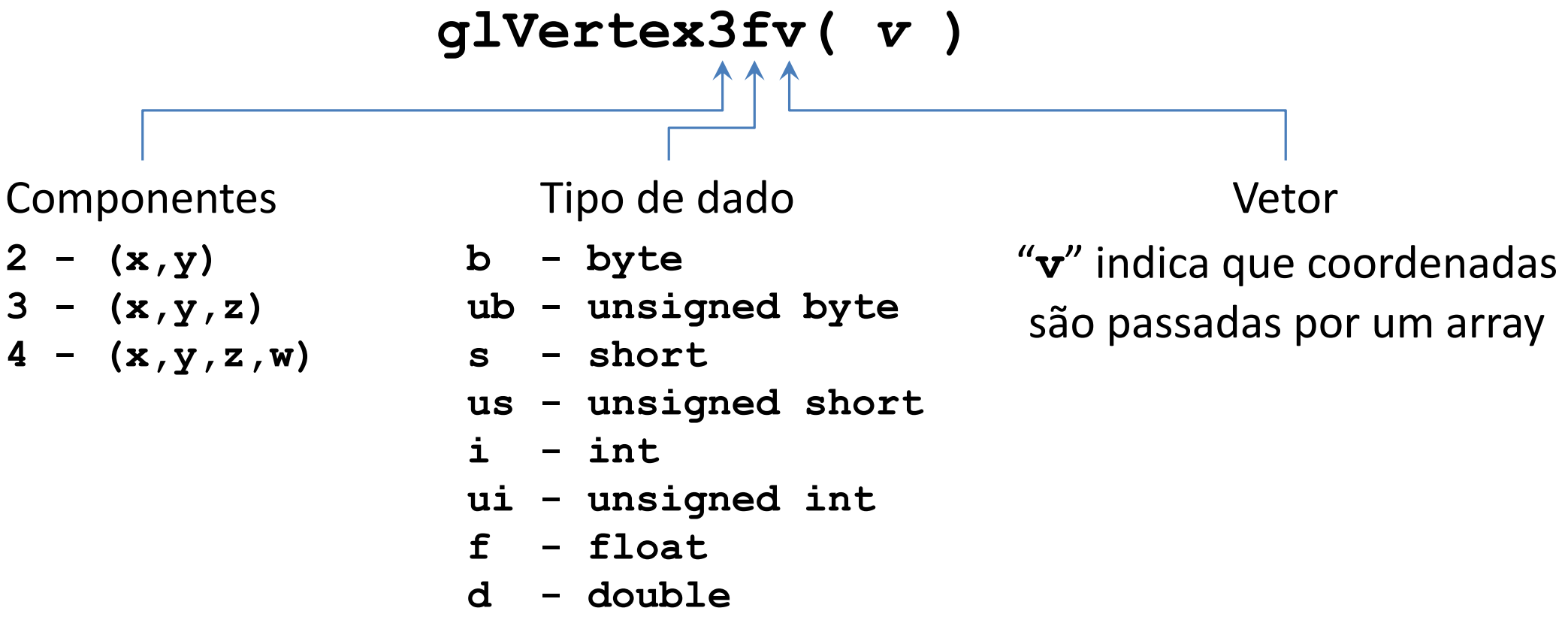
Sintaxe OpenGL

Convenções de nomes:

- Todas as funções começam com o prefixo **`gl`**, e.g., **`glClearColor()`**
- Padrão *lower camel case*, e.g., **`glColor()`**
- O sufixo indica a quantidade e o tipo dos argumentos, e.g., **`glVertex2i(1,3)`**
- Constantes: **`GL_COLOR_BUFFER_BIT`**

Sintaxe OpenGL

Convenções de nomes:

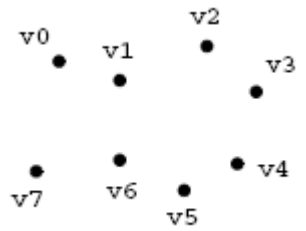


Sintaxe OpenGL

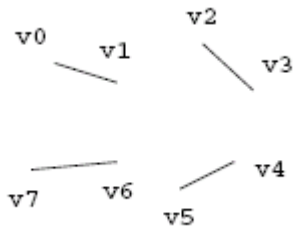
Convenções de nomes (tipos):

Nome	Tipo	C	OpenGL
b	inteiro 8-bits	signed char	GLbyte
s	inteiro 16-bits	short	GLshort
i	inteiro 32-bits	long	GLint, GLsizei
f	ponto flutuante 32-bits	float	GLfloat, GLclampf
d	ponto flutuante 64-bits	double	GLdouble, GLclampd
ub	caractere sem sinal 8-bits	unsigned char	GLubyte, GLboolean
us	caractere sem sinal 16-bits	unsigned short	GLushort
ui	caractere sem sinal 32-bits	unsigned long	GLuint, GLenum, GLbitfield

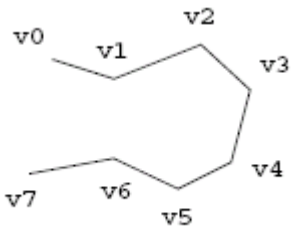
Primitivas de Desenho



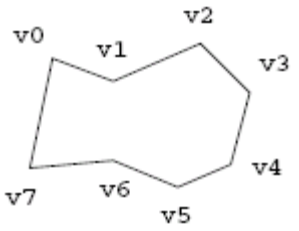
GL_POINTS



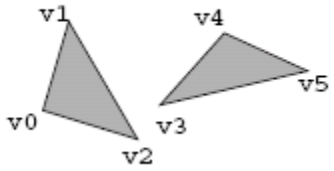
GL_LINES



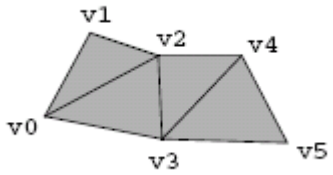
GL_LINE_STRIP



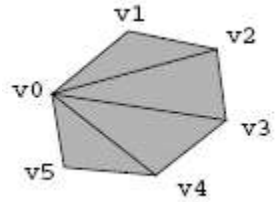
GL_LINE_LOOP



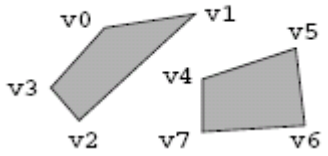
GL_TRIANGLES



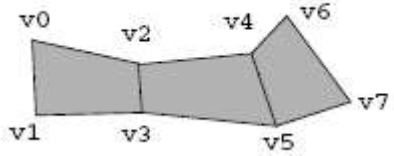
GL_TRIANGLE_STRIP



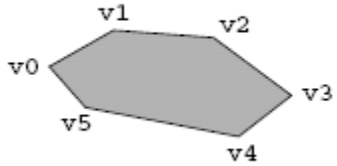
GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP



GL_POLYGON

Primitivas de Desenho

Primitiva	Significado
`GL_POINTS`	Pontos individuais
`GL_LINES`	Pares de vértices interpretados como segmentos de reta individuais
`GL_LINE_STRIP`	Série de segmentos de reta conectados
`GL_LINE_LOOP`	Igual ao anterior. Ultimo vértice conectado a primeiro
`GL_TRIANGLES`	Triplas de vértices interpretados como triângulos
`GL_TRIANGLE_STRIP`	Cadeia triângulos conectados
`GL_TRIANGLE_FAN`	Leque de triângulos conectados
`GL_QUADS`	Quadrupla de vértices interpretados como quadriláteros
`GL_QUAD_STRIP`	Cadeia de quadriláteros conectados
`GL_POLYGON`	Borda de um polígono simples

Sintaxe OpenGL

- Utilizando as primitivas:

```
glBegin(PRIMITIVA);
// comandos para a especificação de vértices, cores,
// coordenadas de textura, propriedades de material
glEnd();
```

- Entre **`glBegin()`** e **`glEnd()`** apenas alguns comandos podem ser usados, e.g., **`glVertex()`**, **`glMaterial()`**, **`glNormal()`**, **`glTexCoord()`**

Máquina de Estados

- Uma aplicação OpenGL funciona como uma **máquina de estados**.
- Os estados correntes permanecem ativos até que sejam modificados.
- Exemplo: a cor de desenho é aplicada a qualquer primitiva geométrica até que a cor corrente seja modificada.
- Existem várias variáveis de estados, por exemplo:
  - cor de desenho corrente
  - transformações de visualização e projeção
  - padrões de linhas e polígonos
  - modo de desenho dos polígonos
  - atributos das fontes de luz
  - propriedades de reflexão e textura dos materiais associados aos objetos

Máquina de Estados

- Alguns comandos para ler um estado:

  `glGetBooleanv()`, `glGetDoublev()`, `glGetFloatv()`,
  `glGetIntegerv()`, `glPointerv()`, `glIsEnabled()`

- Comandos para salvar um estado:

  `glPushAttrib()` e **`glPushClientAttrib()`**

- Comandos para restaurar um estado:

  `glPopAttrib()` e **`glPopClientAttrib()`**

Máquina de Estados

Exemplo **GL_LIGHTING**:

```
int luz;

glEnable(GL_LIGHTING);   //Habilita iluminação
luz = glIsEnabled(GL_LIGHTING); // retorna 1 (verdadeiro)

glDisable(GL_LIGHTING); //Desabilita luz
luz = glIsEnabled(GL_LIGHTING); // retorna 0 (falso)
```

APIs Complementares

GLU (OpenGL Utility Library)

- Parte do padrão OpenGL.
- NURBS, trianguladores, quádricas, mapeamento, mipmaps, superfícies quadráticas, transformação e posicionamento da câmera e primitivas de desenho adicionais.

AGL, GLX, WGL

- Camadas entre o OpenGL os diversos sistemas de janelas.

GLUT (OpenGL Utility Toolkit)

- API portátil de acesso aos sistemas de janelas.
- Encapsula e esconde as camadas proprietárias.
- Não é parte oficial do OpenGL.
- http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

GLUT (OpenGL Utility Toolkit)

- Software gratuito, não open-source (Licença MIT): **freeglut**.
- API que permite a implementação de aplicativos interativos simples com uso do OpenGL.
- Independente do sistema de janelas do sistema operacional.
- Permite a implementação de aplicativos multiplataforma.
- Bindings para diversas linguagens de programação.
- Fornece um conjunto de primitivas para desenho de objetos mais complexos como quádricas e etc.

Callbacks do GLUT

Callbacks são trechos de código (rotinas/funções) chamadas para tratar **eventos** síncronos ou assíncronos:

```
glutDisplayFunc
glutOverlayDisplayFunc
glutReshapeFunc
glutKeyboardFunc
glutMouseFunc
glutMotionFunc
glutPassiveMotionFunc
glutVisibilityFunc
glutEntryFunc
glutSpecialFunc
glutSpaceballMotionFunc
glutSpaceballRotateFunc
glutSpaceballButtonFunc
glutButtonBoxFunc
glutDialsFunc
glutTabletMotionFunc
glutTabletButtonFunc
glutMenuStatusFunc
glutIdleFunc
glutTimerFunc
```

Callbacks do GLUT

- Para uma rotina/função *callback* ser ativada ela precisa ser **registrada**.
- Os eventos são monitorados pelo GLUT e as respectivas funções *callback* são chamadas a partir do **`GlutMainLoop()`**.

```
void Teclado(unsigned char key, int x, int y){
     // comandos...
}
int main(int argc, char ** argv){
     // comandos...
     glutKeyboardFunc(Teclado);
     glutMainLoop();
}
```

Callbacks do GLUT

- Para uma rotina/função *callback* ser ativada ela precisa ser **registrada**.
- Percebam que a função definida para tratar um evento deve ter a mesma assinatura (parâmetros) da função de registro:

```
void Teclado(unsigned char key, int x, int y)

void glutKeyboardFunc(void (*func)(unsigned char key,
int x, int y))
```

Callback de Desenho

`glutDisplayFunc(void(*func)(void))`

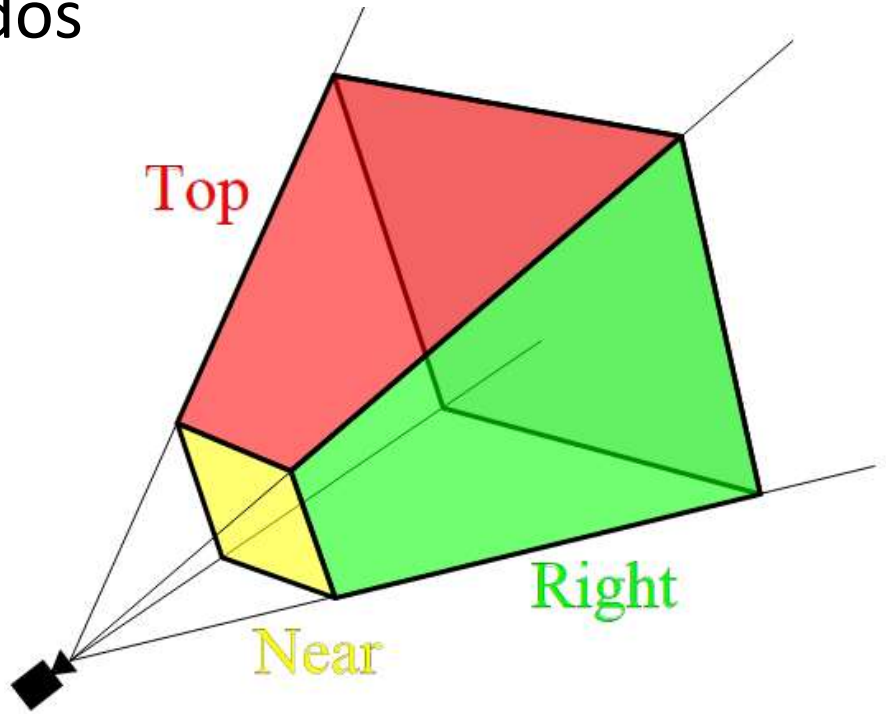
- Chamada automaticamente sempre que a janela ou parte dela precisa ser redesenhada.
- Todo programa OpenGL/GLUT precisa ter uma!
- Exemplo:

```
void Desenha (void){
  glBegin(GL_TRIANGLE_STRIP);
    glVertex3fv( v[0] );
    glVertex3fv( v[1] );
    glVertex3fv( v[2] );
    glVertex3fv( v[3] );
  glEnd();
  glFlush(); // força a execução de funções pendentes
}
```

Callback de Redimensionamento

```
glutReshapeFunc((void(*func)(int width, int height))
```

- Chamada sempre que a janela é redimensionada: **`width`** e **`height`** são a nova largura/altura da janela (em pixels).
- Os valores fornecidos podem ser usados para recalcular o ***frustum***:



Callback de Teclado

```
void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y))
```

- Chamada sempre que um caractere é pressionado: **`key`** indica o caractere; **`x`** e **`y`** indicam a posição relativa do mouse no momento.
- **`glutGetModifiers`** pode ser chamado para determinar outras teclas pressionadas simultaneamente (Shift, Ctrl, Alt).
- Para desabilitar esta *callback* é preciso passar **`NULL`** para **`glutKeyboardFunc`**.

Callback de Temporização

```
void glutTimerFunc(unsigned int msecs, void (*func)(int value), value)
```

- Após **`msecs`** milissegundos, a chamada para a *callback* se dará assim que for possível .
- O parâmetro **`value`** pode ser usado dentro do código, para, por exemplo, ignorar a *callback* de temporização.
- Podem ser registradas múltiplas *callbacks* de temporização.

Outros Callbacks

- Eventos de mouse:

```
void glutMouseFunc(int button, int state, int x, int y)
void glutMotionFunc(int x, int y)
void glutPassiveMotionFunc(int x, int y)
```

- Chamada continuamente quando nenhum outro evento ocorre:

```
void glutIdleFunc(void)
```

Criando janelas do GLUT

`int glutCreateWindow (char* name)`

- Cria uma nova janela primária (*top-level*).
- O parâmetro **`name`** é usado para rotular a janela.
- O número inteiro retornado é usado pelo GLUT para identificar a janela.

`glutInitWindowPosition (int x, int y)`

- Estabelece a posição inicial do canto superior esquerdo da janela.

`glutInitWindowSize (int width, int height)`

- Estabelece o tamanho (em pixels) da janela.

Laço principal do GLUT

`glutMainLoop (void)`

- Entrega o controle ao sistema de janelas.
- Esta rotina na verdade é o “despachante” de eventos.
- *Loop* infinito: função nunca retorna.

Exemplo #1: Janela GLUT

```
// PrimeiroPrograma.c - Isabel H. Manssour
// Um programa OpenGL simples que abre uma janela GLUT
// Este código está baseado no Simple.c, exemplo
// disponível no livro "OpenGL SuperBible",
// 2nd Edition, de Richard S. e Wright Jr.

#include <gl/glut.h>

// Função callback chamada para fazer o desenho
void Desenha(void) {

      //Limpa a janela de visualização com a cor de fundo especificada
      glClear(GL_COLOR_BUFFER_BIT);

      //Executa os comandos OpenGL
      glFlush();
}
```

Exemplo #1: Janela GLUT

```
// Inicializa parâmetros de desenho
void Inicializa(void) {
      // Define a cor de fundo da janela de visualização como preta
      glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
}

// Programa Principal
int main(int argc, char ** argv) {
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutCreateWindow("Primeiro Programa");
      glutDisplayFunc(Desenha);
      Inicializa();
      glutMainLoop();
}
```

Exemplo #2: Desenho 2D

```
#include <gl/glut.h>

// Função callback chamada para fazer o desenho
void Desenha(void) {
	//Limpa a janela de visualização com a cor branca
	glClearColor(1, 1, 1, 0);
	glClear(GL_COLOR_BUFFER_BIT);
	//Define a cor de desenho: vermelho
	glColor3f(1, 0, 0);
	//Desenha um triângulo no centro da janela
	glBegin(GL_TRIANGLES);
		glVertex2f(-0.5, -0.5);
		glVertex2f(0.0, 0.5);
		glVertex2f(0.5, -0.5);
	glEnd();
	//Executa os comandos OpenGL
	glFlush();
}
```

Exemplo #2: Desenho 2D

```
//Função callback chamada para gerenciar eventos de teclas
void Teclado(unsigned char key, int x, int y) {
      if (key == 27)
            exit(0);
}

// Inicializa parâmetros e variáveis
void Inicializa(void) {
      // Define a janela de visualização 2D
      glMatrixMode(GL_PROJECTION);
      gluOrtho2D(-1.0, 1.0, -1.0, 1.0);
      glMatrixMode(GL_MODELVIEW);
}
```

Exemplo #2: Desenho 2D

```
// Programa Principal
int main(int argc, char ** argv) {
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutInitWindowSize(400, 400);
      glutCreateWindow("Segundo Programa");
      glutDisplayFunc(Desenha);
      glutKeyboardFunc(Teclado);
      Inicializa();
      glutMainLoop();
      return 0;
}
```

Exemplo #2: Desenho 2D

Experimente:

- Altere no código a cor de cada um dos vértices do triângulo: coloque cores diferente para cada um.
- Em tempo de execução, altere o tamanho da janela criada pelo programa.

Exemplo #3: Window × Viewport

```
#include <windows.h>
#include <gl/glut.h>
// Função callback chamada para fazer o desenho
void Desenha(void) {
      glMatrixMode(GL_MODELVIEW);
      glLoadIdentity();
      // Limpa a janela de visualização com a cor de fundo especificada
      glClear(GL_COLOR_BUFFER_BIT);
      // Especifica que a cor corrente é verde
      glColor3f(0.0f, 1.0f, 0.0f);
      glBegin(GL_QUADS); // Desenha quadrado com a cor corrente
            glVertex2i(100,150);
            glVertex2i(100,100);
            glVertex2i(150,100);
            glVertex2i(150,150);
      glEnd();
      glFlush(); // Executa os comandos OpenGL
}
```

Exemplo #3: Window × Viewport

```
// Inicializa parâmetros de desenho
void Inicializa (void) {
    // Define a cor de fundo da janela de visualização como preta
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
}
// Função callback chamada quando o tamanho da janela é alterado
void AlteraTamanhoJanela(GLsizei w, GLsizei h) {
      if(h == 0) h = 1; // Evita a divisao por zero
      glViewport(0, 0, w, h); // Especifica as dimensões da Viewport
      glMatrixMode(GL_PROJECTION); // Inicializa a matriz de projeção
      glLoadIdentity();
      // Estabelece World Window (left, right, bottom, top)
      if (w <= h)
            gluOrtho2D (0.0f, 250.0f, 0.0f, 250.0f*h/w);
      else
            gluOrtho2D (0.0f, 250.0f*w/h, 0.0f, 250.0f);
}
```

Exemplo #3: Window × Viewport

```
// Programa Principal
int main(int argc, char ** argv) {
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
      glutInitWindowSize(400,350);
      glutInitWindowPosition(10,10);
      glutCreateWindow("Quadrado");
      glutDisplayFunc(Desenha);
      glutReshapeFunc(AlteraTamanhoJanela);
      Inicializa();
      glutMainLoop();
}
```

Exemplo #3: Window × Viewport

Experimente:

- Altere no código a cor de cada um dos vértices do quadrado.
- Em tempo de execução, altere o tamanho da janela criada pelo programa.
- Por que o redimensionamento da janela provoca um efeito diferente em relação ao Exemplo #2?

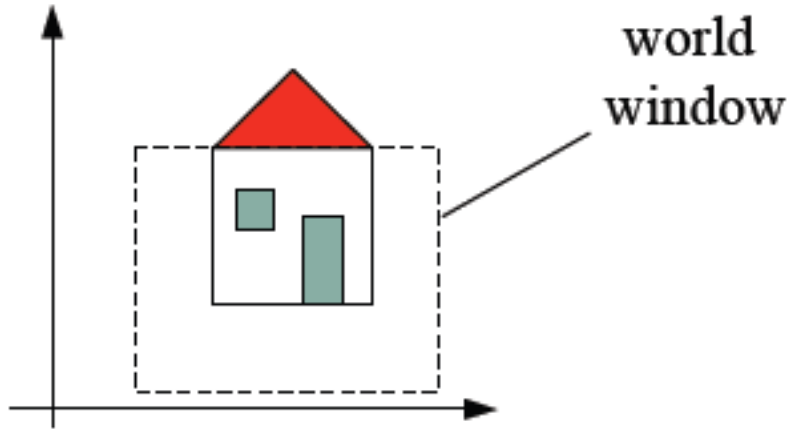
Window × Viewport

Sistema de Coordenadas Globais (**World Coordinate System**)

- Domínio da Cena: espaço dos objetos.
- Espaço em que se encontram os objetos geométricos.
- Espaço onde a geometria do objeto e o modelo de aplicação é definido.

Subdomínio da Cena (**World Window**)

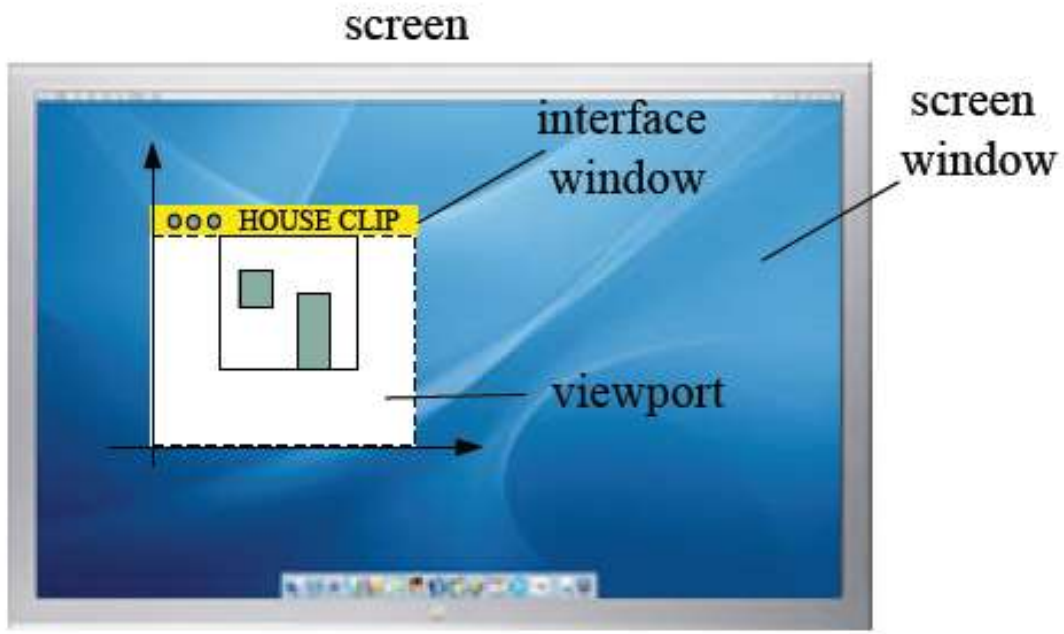
- Retângulo que define a parte da cena (*world*) que se pretende visualizar.



Window × Viewport

Sistema de Coordenadas da Imagem (**Screen Coordinate System**)

- Espaço no qual a imagem é mostrada; e.g., 800x600 pixels.
- Espaço no qual a imagem rasterizada do objeto é definida.

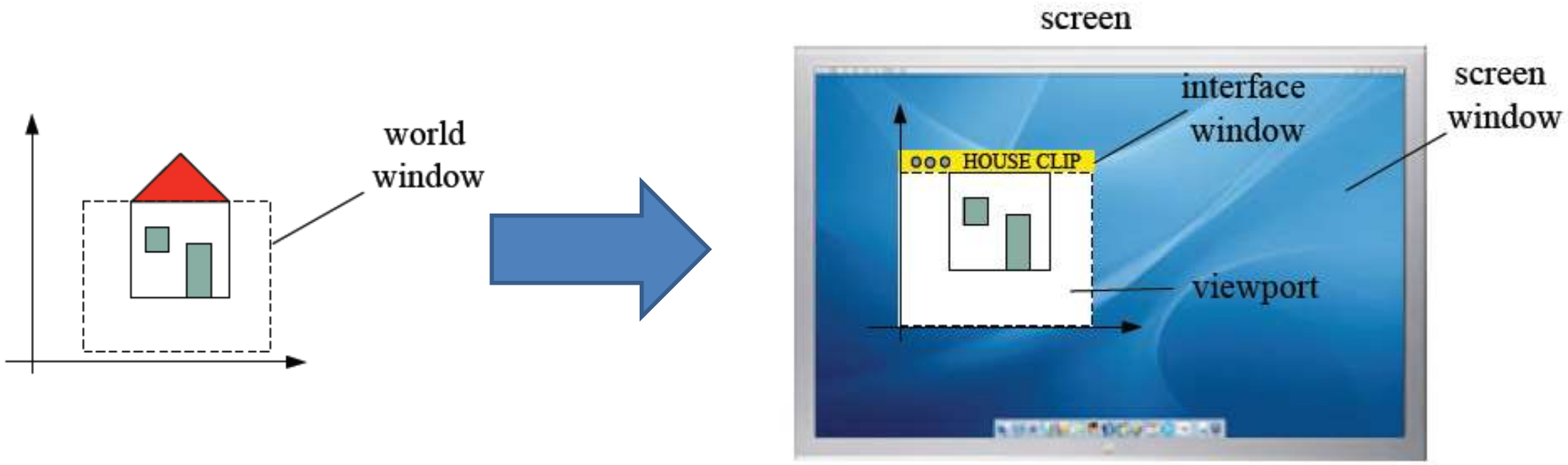


Subdomínio de Imagem (**Viewport**)

- Sistema de coordenadas da tela (janela do sistema de janelas).
- Sistema de coordenadas se move com a janela.

Mapeamento Window × Viewport

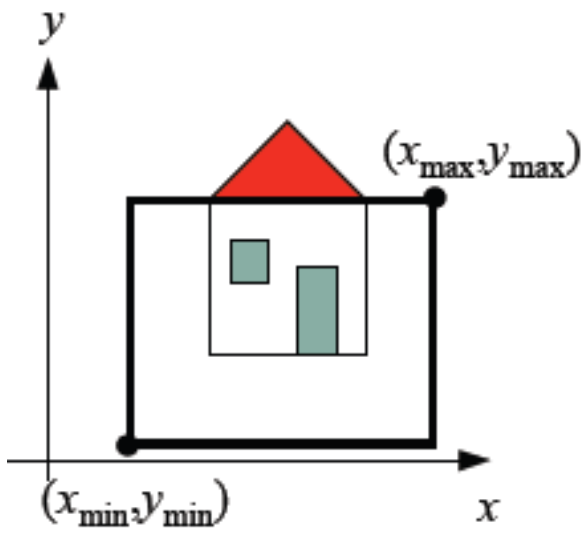
- Transformação de visualização: *Window* × *Viewport*.
- Mapeamento de uma janela no domínio da cena (**World Coordinates**) para o *viewport* (**Screen Coordinates**).



Mapeamento Window × Viewport

`gluOrtho2D(left, right, bottom, top)`

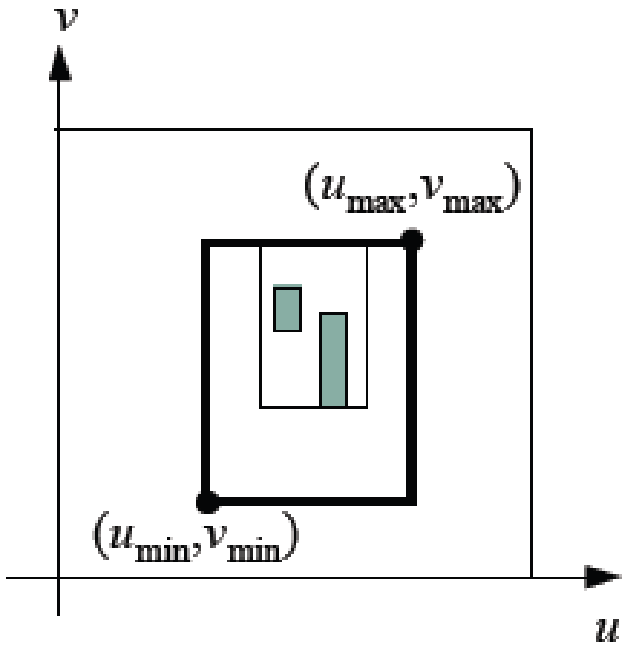
- Define um retângulo de visualização ortogonal 2D (*Window*): parte visível da cena.
- Define dois planos verticais (**`left`**, **`right`**) e dois planos horizontais (**`bottom`**,**`top`**).
- Os valores (**`left`**,**`right`**,**`bottom`**,**`top`**) são (`-1`,`1`,`-1`,`1`) por default.
- Define uma matriz de projeção ortogonal 2D.
- Define ainda a transformação *Window* × *Viewport*, em combinação com a função **`glViewport`**.



Mapeamento Window × Viewport

`glViewport (x, y, width, height)`

- Define as dimensões do *Viewport* numa janela do sistema de janelas.
- Onde **`x`** e **`y`** indicam o canto inferior esquerdo; e **`widht`** e **`height`** a extensão do *Viewport*.
- Por default o *Viewport* ocupa toda a janela.
- Podem existir vários *Viewports* dentro de uma janela.

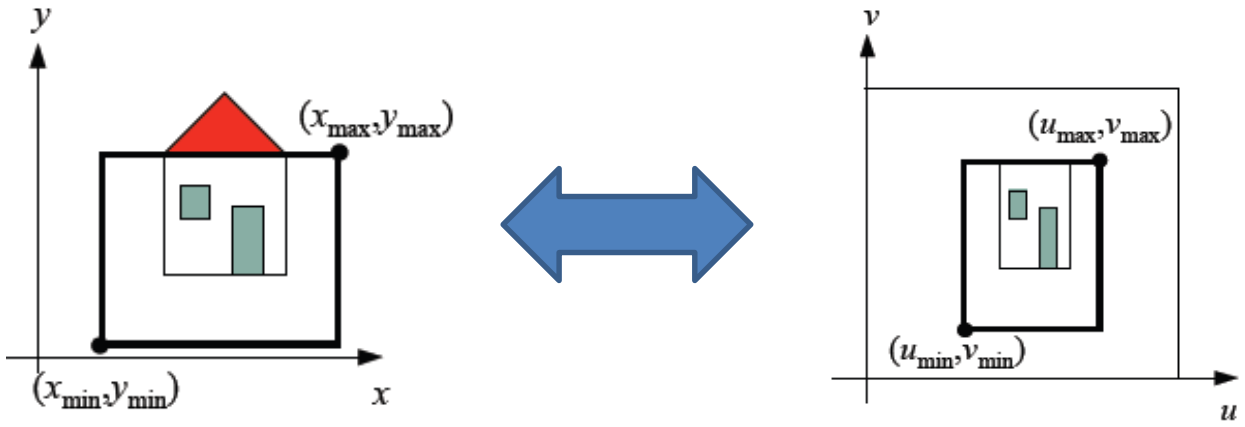


Mapeamento Window × Viewport

- Manutenção automática das proporções na transformação de visualização.
- Para evitar distorção quando a janela na tela é redimensionada.
- Inserir na função *callback* de redimensionamento de janela, e.g., **`AlteraTamanho`** (registrada através de **`glutReshapeFunc`**):

```
void AlteraTamanho(GLsizei w, GLsizei h){
      glViewport(0, 0, w, h);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity(); // redefine a matriz de projeção
      if (w <= h)
            gluOrtho2D (xmin, xmax, ymin, ymax*h/w);
      else
            gluOrtho2D (xmin, xmax*w/h, ymin, ymax);
}
```

Mapeamento Window × Viewport




```
void AlteraTamanho(GLsizei w, GLsizei h){
      glViewport(0, 0, w, h);
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity(); // redefine a matriz de projeção
      if (w <= h)
            gluOrtho2D (xmin, xmax, ymin, ymax*h/w);
      else
            gluOrtho2D (xmin, xmax*w/h, ymin, ymax);
}
```

Exemplo #4: Interface-Movimento

```
#include <GL/glut.h>
// Funções callback
void desenha(void);
void movimento(int x, int y);
void botao(int button, int state, int x, int y);
void alteraTamanhoJanela(GLsizei w, GLsizei h);
// Variáveis globais
float largura, altura; // tamanho da world window e do viewport
float rx, ry; // posição inicial do retângulo vermelho

int main(int argc, char *argv[]) {
      largura = 600;
      altura = 600;
      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_RGBA | GLUT_DOUBLE);
      glClearColor(0, 0, 0, 1);
```

Exemplo #4: Interface-Movimento

```
        glutPositionWindow(200, 100);
        glutReshapeWindow(largura, altura);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluOrtho2D(0, largura, 0, altura);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glutDisplayFunc(desenha);
        glutIdleFunc(desenha);
        glutMotionFunc(movimento);
        glutMouseFunc(botao);
        glutReshapeFunc(alteraTamanhoJanela);
        glutMainLoop();
        return 0;
}
```

Exemplo #4: Interface-Movimento

```
// Função callback chamada para fazer o desenho
void desenha(void){
      glClear(GL_COLOR_BUFFER_BIT);
      glColor3f(1, 1, 0); // desenha em amarelo
      glRectf(100, 100, 300, 200); // desenha retângulo amarelo
      glColor3f(1, 0, 0); // desenha em vermelho
      glRectf(rx, ry, 100 + rx, 20 + ry); // desenha retângulo vermelho
      glutSwapBuffers(); // buffer de cor duplo
}
// Função callback para o mouse se movimentando com botão pressionado
void movimento(int x, int y){
      rx = x; ry = altura - y;
}
// Função callback chamada quando o botão do mouse é pressionado
void botao(int button, int state, int x, int y){
      if (button == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
            rx = x; ry = altura - y;
}
```

Exemplo #4: Interface-Movimento

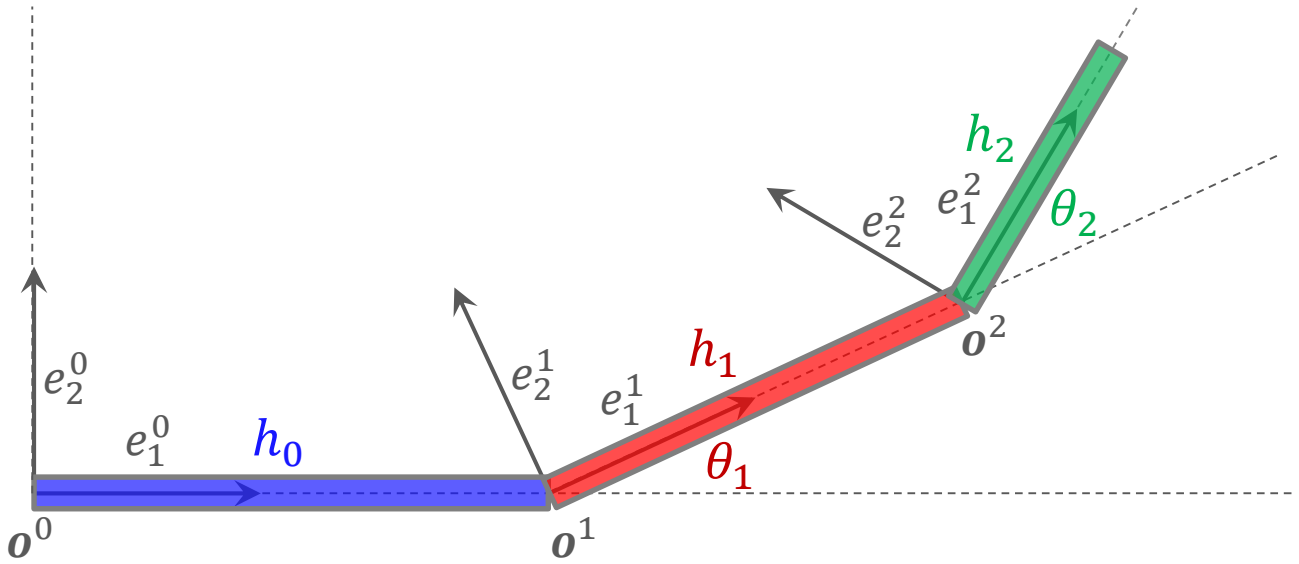
```
// Função callback chamada quando o tamanho da janela é alterado
void alteraTamanhoJanela(GLsizei w, GLsizei h){
      largura = w;
      altura = h;

      // Especifica as dimensões da Viewport
      glViewport(0, 0, largura, altura);  // comando não necessário:
                                          // comportamento default
      // Inicializa o sistema de coordenadas do mundo
      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();

      // Dimensões da World Window (left, right, bottom, top)
      gluOrtho2D(0, largura, 0, altura);
}
```

Exemplo #4: Interface-Movimento

Experimente:

- Dê um clique do mouse para movimentar o retângulo vermelho.
- Clique e arraste (com o botão esquerdo pressionado).
- Em tempo de execução, altere o tamanho da janela criada pelo programa.
- Por que o redimensionamento da janela provoca um efeito diferente em relação aos exemplos 2 e 3?

Exemplo #5: Animação-Timer



Exemplo #5: Animação-Timer

```
#include <GL/freeglut.h>

#define PI 3.141592654
#define tAnima 5.0
#define deltaT 33

float theta1, theta2, thetaMax, thetaMin, hMax, h0, h1, h2,incremento;

void draw(void);
void keyboard(unsigned char key, int x, int y);
void setTimer(int value);
void reshapeWindow(GLsizei w, GLsizei h);
```

Exemplo #5: Animação-Timer

```
int main(int argc, char ** argv){
      h0 = 6; h1 = 4; h2 = 2;
      hMax = h0 + h1 + h2;
      theta1 = theta2 = 0;
      thetaMax = PI/2;
      thetaMin = -PI/2;
      incremento = 0.1;

      glutInit(&argc, argv);
      glutInitDisplayMode(GLUT_RGB | GLUT_DOUBLE);
      glutCreateWindow("Braço Mecânico 2D");
      glutPositionWindow(200, 100);
      glutReshapeWindow(200, 400);

      glMatrixMode(GL_PROJECTION);
      glLoadIdentity();
      gluOrtho2D(0, hMax, -hMax, hMax);
```

Exemplo #5: Animação-Timer

```
        glutDisplayFunc(draw);
        glutKeyboardFunc(keyboard);
        glutTimerFunc(deltaT, setTimer, 1);
        glutReshapeFunc(reshapeWindow);
        glutMainLoop();
        return 0;
}
void draw(void){
        glClearColor(1, 1, 1, 1);
        glClear(GL_COLOR_BUFFER_BIT);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glLineWidth(20);
        glBegin(GL_LINES); /* haste 0 */
                glColor3f(0, 0, 1);
                glVertex2f(0.0, 0.0);
                glVertex2f(h0, 0.0);
        glEnd();
```

Exemplo #5: Animação-Timer

```
        /* haste 1 */
        glTranslatef(h0, 0, 0);
        glRotatef(180.0*theta1 / PI, 0.0, 0.0, 1.0);
        glBegin(GL_LINES);
              glColor3f(1, 0, 0);
              glVertex2f(0.0, 0.0);
              glVertex2f(h1, 0.0);
        glEnd();
        /* haste 2 */
        glTranslatef(h1, 0, 0);
        glRotatef(180.0*theta2 / PI, 0.0, 0.0, 1.0);
        glBegin(GL_LINES);
              glColor3f(0, 1, 0);
              glVertex2f(0.0, 0.0);
              glVertex2f(h2, 0.0);
        glEnd();
        glutSwapBuffers();
}
```

Exemplo #5: Animação-Timer

```
void keyboard(unsigned char key, int x, int y){
      if (key == 27) exit(0);
}
void setTimer(int value){
      theta1 = theta1 + incremento;
      if ((theta1>=thetaMax)||(theta1<=thetaMin)) incremento *= -1;
      theta2 = theta1 * 2;
      glutPostRedisplay();
      glutTimerFunc(deltaT, setTimer, 1);
}
void reshapeWindow(GLsizei w, GLsizei h){
      if (h == 0) h = 1;
      glViewport(0, 0, h/2, h);
}
```

Exemplo #5: Animação-Timer

Experimente:

- Em tempo de execução, altere o tamanho da janela criada pelo programa.
- Modifique o código para fazer com que os ângulos θ_1 e θ_2 sejam alterados pelo teclado.

Computação Gráfica (IME 04-10842)
2022.2

Introdução ao OpenGL

Gilson. A. O. P. Costa (IME/UERJ)

gilson.costa@ime.uerj.br