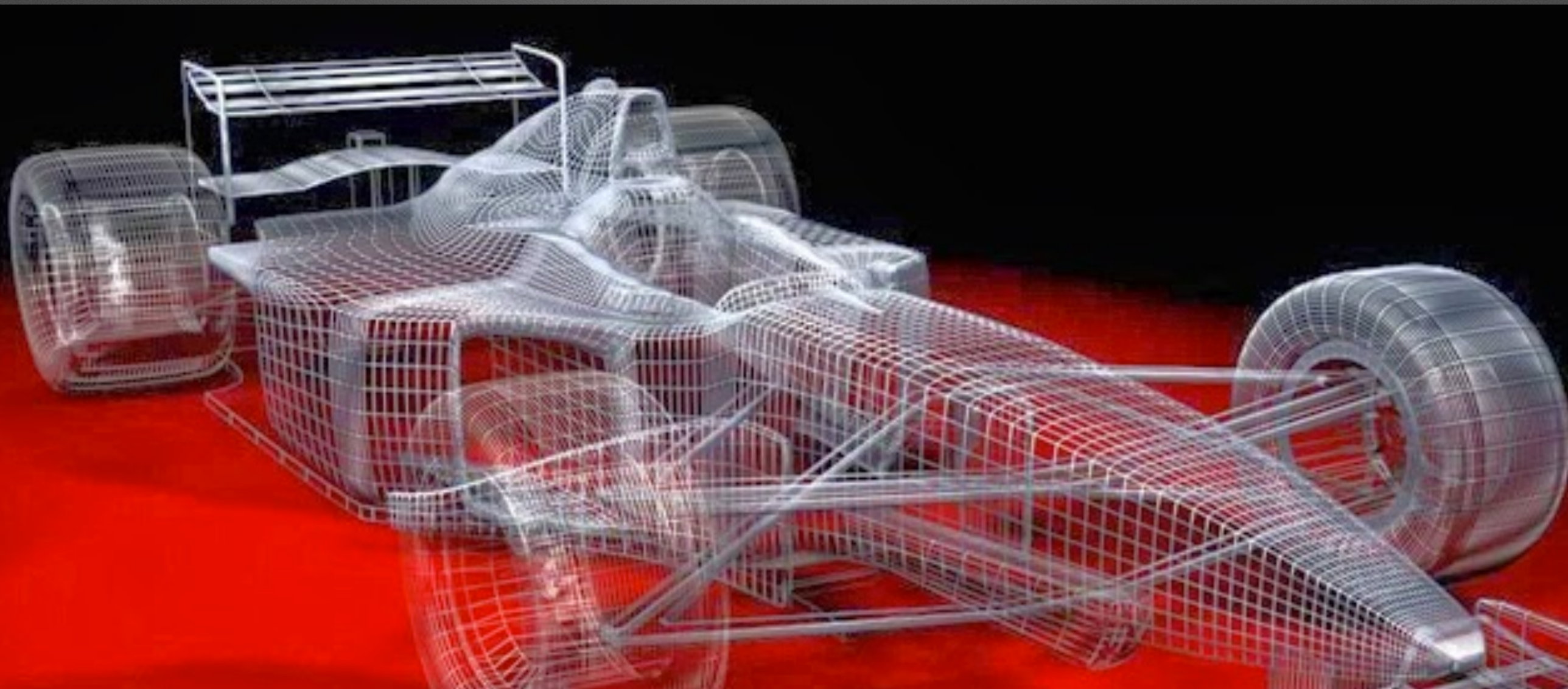
Computação Gráfica

Nécio de Lima Veras
necio.veras@ifce.edu.br

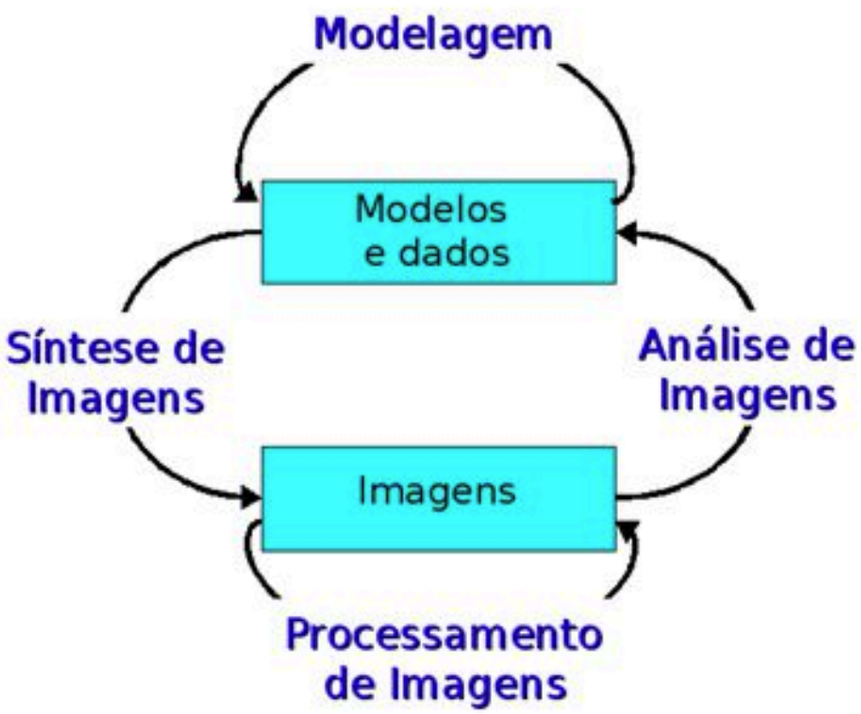
Introdução ao OpenGL

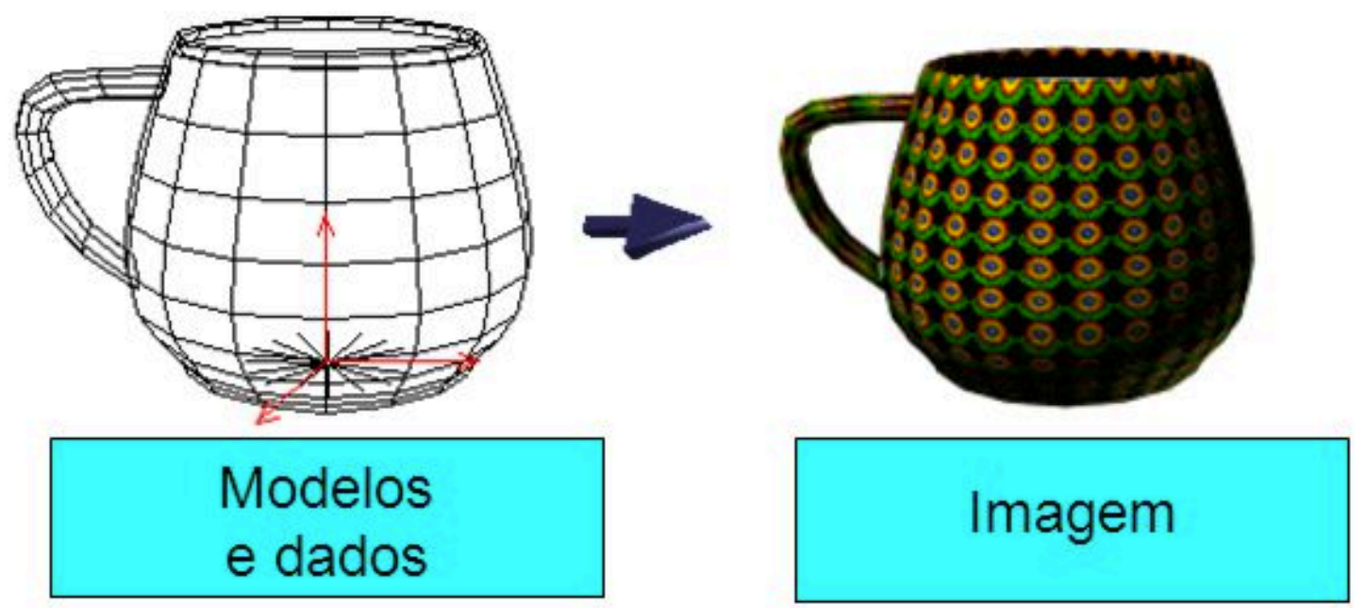
Créditos:
Josué Machado
Profa. Isabel Harb
Adaptações:
Nécio de Lima Veras

Síntese de Imagens



Síntese de Imagens





OpenGL

- Surge o OpenGL:
  - Sucessora da biblioteca gráfica IRIS GL, ambas desenvolvidas pela Silicon Graphics, Inc. (SGI) – 1992;
- É mantida por:
  - Um consórcio chamado ARB (*OpenGL Architecture Review Board*);

Mas, afinal o que é OpenGL?

- “GL” significa *Graphics Library*
- Interface de programação para aplicações gráficas 2D ou 3D
- Independente do sistema de janelas:
  - GLX (*OpenGL Extension for X*);
  - GLUT (*OpenGL Utility Toolkit);*
- Palavras reservadas começam com:
  - gl ou GL_

OpenGL

- Não é uma linguagem de programação, mas, uma API gráfica com cerca de 150 funções para manipulação de imagem;
- Interface independente de hardware e plataforma;
- Possui implementações para: Windows, Linux e MacOS;
- Pode ser utilizada por diversas linguagens de programação: C/C++, Java, Delphi, etc;

Principais pontos...

- Máquina de estados:
  - Vértices são afetados pelo estado atual;
  - Matrizes de transformação, cor, iluminação, textura
- Controle encapsulado:
  - Iluminação;
  - Textura;
- Interface procedural:
  - Programador especifica exatamente como a cena será renderizada;

Onde é utilizado?

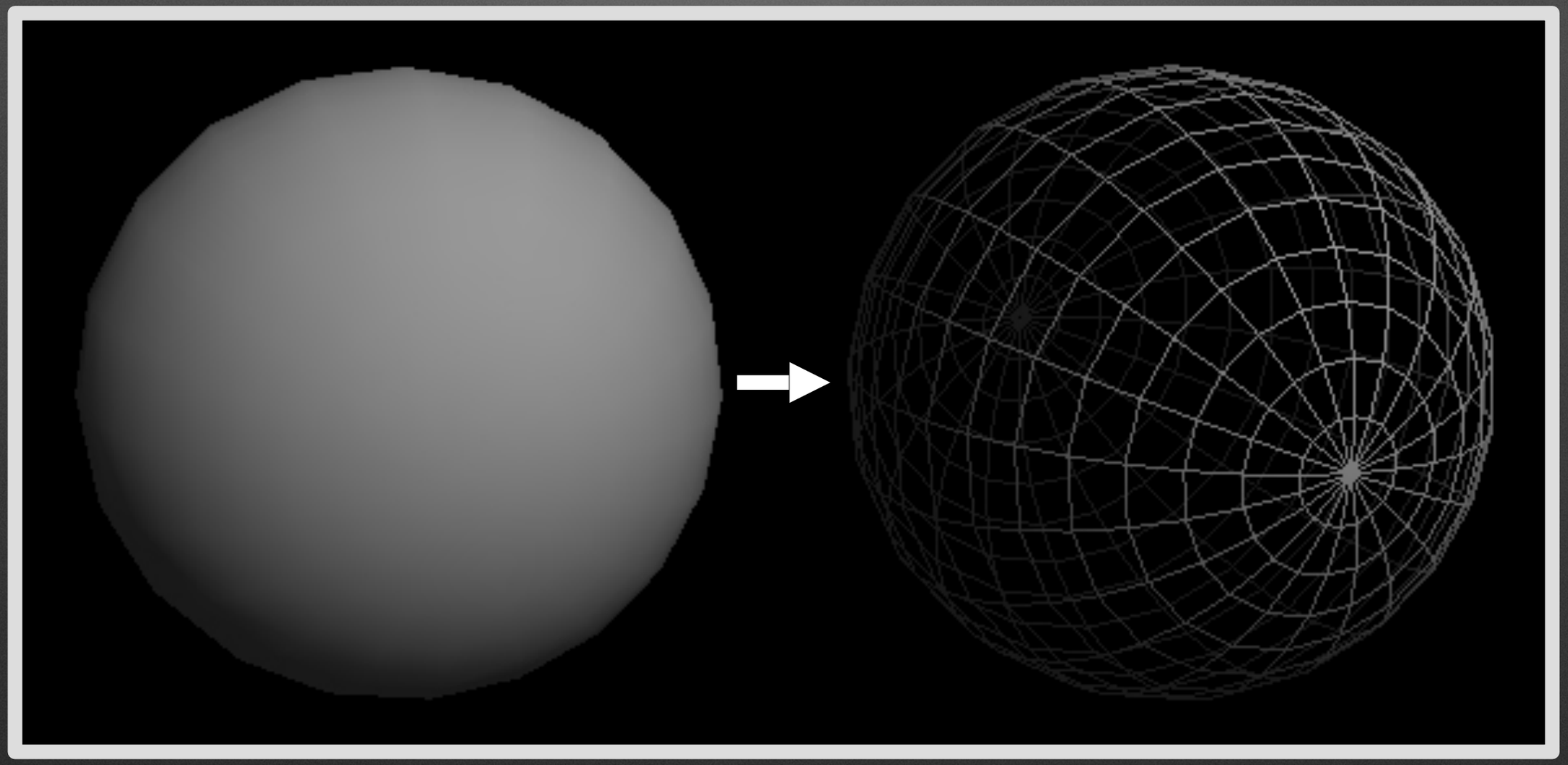
- Programas iterativos que produzem imagens coloridas de objetos móveis tridimensionais;
- Jogos 2D/3D;
- Filmes;
- Realidade Virtual;
- Arquitetura;
- etc...

Que linguagem devo usar?

- C++;
- Principais razões:
  - Fortemente orientada a objetos;
  - Independente de plataforma;
  - Possuir alto desempenho;
  - Permitir otimizações de baixo nível;
  - Atualmente, a mais utilizada em desenvolvimento de jogos;
  - Nativamente, ela é C++;

Como os objetos são feitos?

- Por meio de Vértices no espaço;



Bibliotecas do padrão OpenGL:

- GLU (OpenGL Utility Library);
- Possui funções para manipulação de todo o posicionamento da cena, alguns objetos complexos e criação de superfícies curvas;

Bibliotecas do padrão OpenGL:

- GLUT (OpenGL Utility Toolkit);
- Possui funções para manipulação de janela e tratamento de eventos, tais como: teclado, mouse, timers, etc;
- Muito útil em aplicações de pequeno ou médio porte;

Máquina de estados

- Internamente OpenGL trabalha com matrizes, estas são guardadas em modo de pilhas;
- Alterando-se algum valor interno, todos os comandos ligados a este que foi alterado, também sofrerá a alteração;

Parte II

Fundamentos

Tipos de dados

- OpenGL tem declarado internamente seus próprios tipos de dados:

Linguagem C++	API OpenGL
int	GLint
float	GLfloat
char	GLbyte
unsigned int	GLuint

Sintaxe das funções

- Todas suas funções seguem os seguintes critérios de nomenclatura:

`glVertex3fv` ← **v** se estiver presente, indica formato vetorial

tipo de dado: **f** float
d double float
s signed short integer
i signed integer

número de componentes: **2**, **3** ou **4**

Sintaxe das funções

- Todas suas funções seguem os seguintes critérios de nomenclatura:

`glVertex3fv` ← **v** se estiver presente, indica formato vetorial

tipo de dado: **f** float
d double float
s signed short integer
i signed integer

número de componentes: **2**, **3** ou **4**

- Outros tipos de dados:

b byte
ub unsigned byte
us unsigned short integer
u unsigned integer

Sintaxe das constantes

- Iniciam pelo nome da biblioteca;
- Todas escritas em maiúsculas;
- Separam palavras pelo caractere “_”;
- Exemplos:
  - GL_LIGHTING;
  - GLUT_RGB;
- As funções da GLUT iniciam com a palavra glut;

Tratamento de eventos

- GLUT utiliza funções de CallBack para tratar eventos, alguns possíveis são:
  - Tecla pressionada;
  - Clique ou movimento do mouse;
  - Redimensionamento da janela, ...;

- Para utilizar essas funções é necessário registrar uma função qualquer para ser chamada sempre que acontecer o evento;

Registrando funções CallBack

- glutDisplayFunc (x)
  - registra a função x para ser chamada sempre que um pixel da tela necessitar ser redesenhado;
  - **Protótipo para a função x: void x (void);**

- glutKeyboardFunc (x)
  - registra a função x para ser chamada sempre que uma tecla ASCII for pressionada;
  - **Protótipo para a função x: void xxx (unsigned char key, int posX, int posY);**

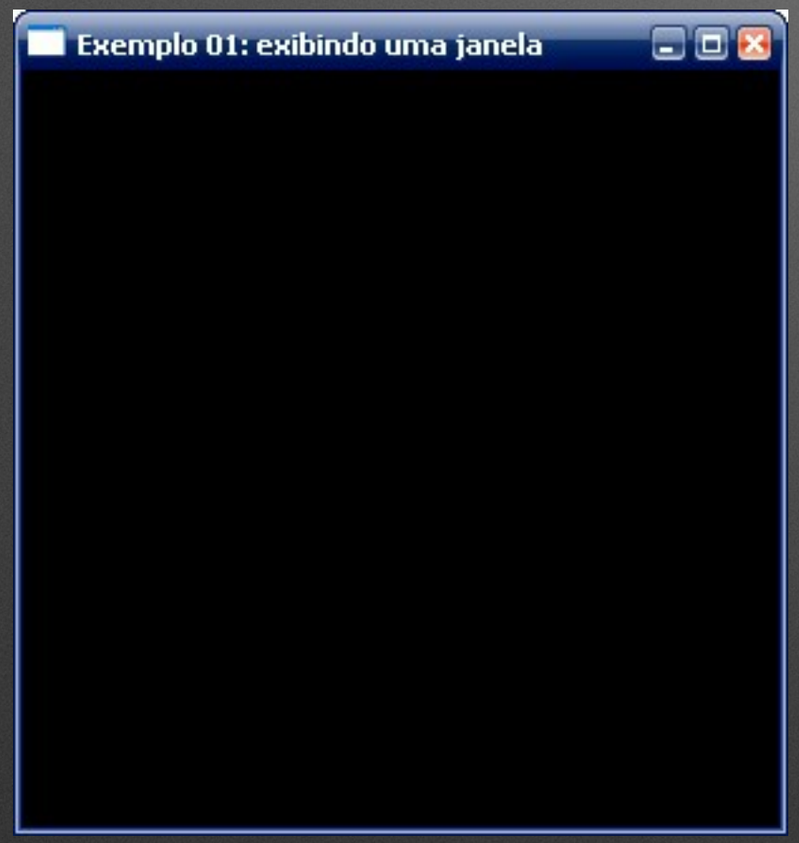
- glutReshapeFunc (x)
  - registra a função x para ser chamada sempre que a janela for redimensionada;
  - **Protótipo para a função x: void xxx (int height, int wirth);**

Registrando funções CallBack

- void glutSpecialFunc (xxx);
  - chamada sempre que uma tecla especial for pressionada (Home, F1...);
  - **Protótipo para a função xxx:**
    - **void xxx (int specialKey, int posX, int posY);**

- void glutIdleFunc (xxx);
- chamada sempre que nada estiver acontecendo;
- **Protótipo para a função xxx:**
    - **void xxx (void);**

Estrutura básica



main.cpp

```
1  // arquivos de cabeçalho
2  #include <stdlib.h>
3  #include <GL\gl.h>
4  #include <GL\glu.h>
5  #include <GL\glut.h>
6
7  // função de desenho
8  void draw ()
9  {
10     glClear (GL_COLOR_BUFFER_BIT);
11
12     glFlush ();
13 }
14
15 // função para inicializar as variaveis de estado
16 void init ()
17 {
18     glClearColor (0.0, 0.0, 0.0, 0.0);
19 }
20
21 // função principal
22 int main (int argc, const char * argv[])
23 {
24     glutInit(&argc, (char **)argv);
25     // configurando e exibindo uma janela
26     glutInitDisplayMode (GLUT_SINGLE | GLUT_RGBA);
27     glutCreateWindow ("Exemplo 01: exibindo uma janela");
28
29     // registrando função de callback
30     glutDisplayFunc (draw );
31
32     // inicialização das variáveis de estado
33     init ();
34
35     // loop de tratamento de eventos
36     glutMainLoop ();
37
38     return 0;
39 }
40
```

Line 33, Column 12

Parte III

Primitivas e Objetos Geométricos

Primitivas OpenGL

- Todas as primitivas são formadas por vértices e todos os objetos são formados por um conjunto finito de primitivas;
- OpenGL em si, trabalha apenas com primitivas simples:
  - Pontos;
  - Segmentos de reta (linhas);
  - Triângulos;
  - Quadrados;
  - Polígonos;

Desenhando primitivas

- Toda primitiva é desenhada pelo seguinte critério:

```
glBegin ( <tipo de primitiva> );
    < vertices da primitiva >
glEnd ( );
```

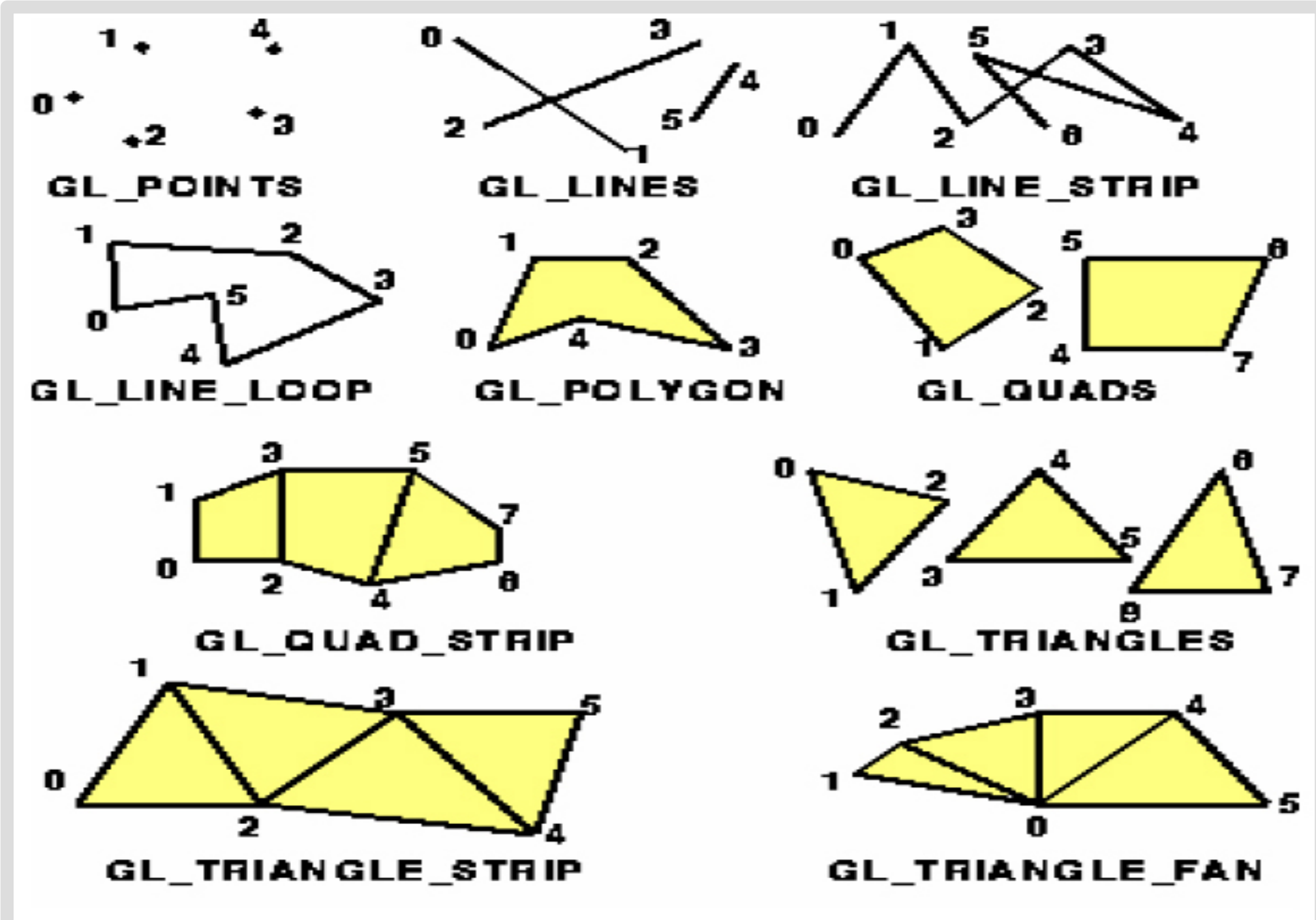
Desenhando primitivas

- Toda primitiva é desenhada pelo seguinte critério:

```
glBegin ( <tipo de primitiva> );
    < vertices da primitiva >
glEnd ( );
```

- Para definirmos um vértice no espaço utilizamos uma das seguintes funções:
  - glVertex2f (float x, float y); (2D)
  - glVertex3f (float x, float y, float z); (3D)

Tipos de primitivas

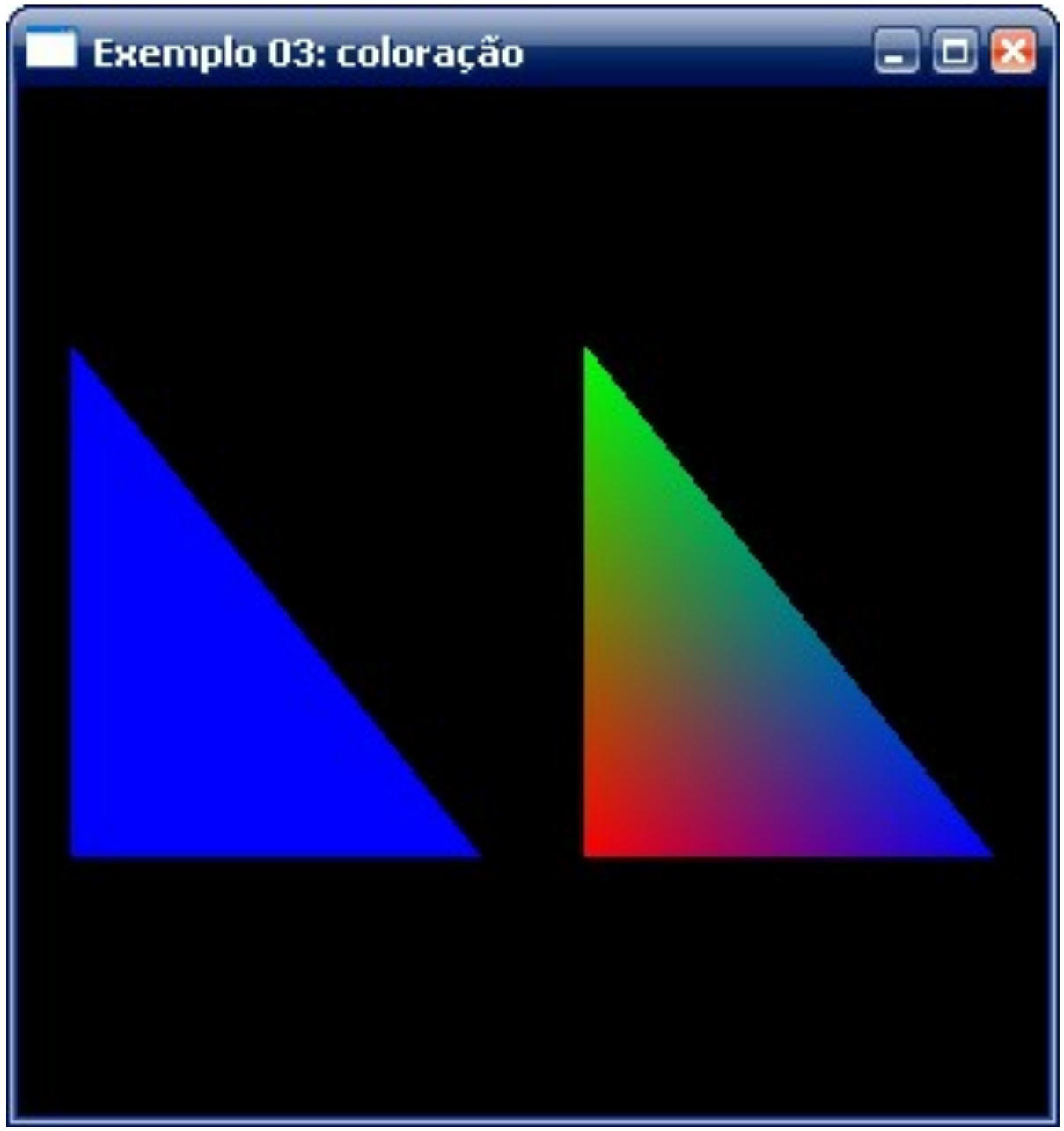


Coloração

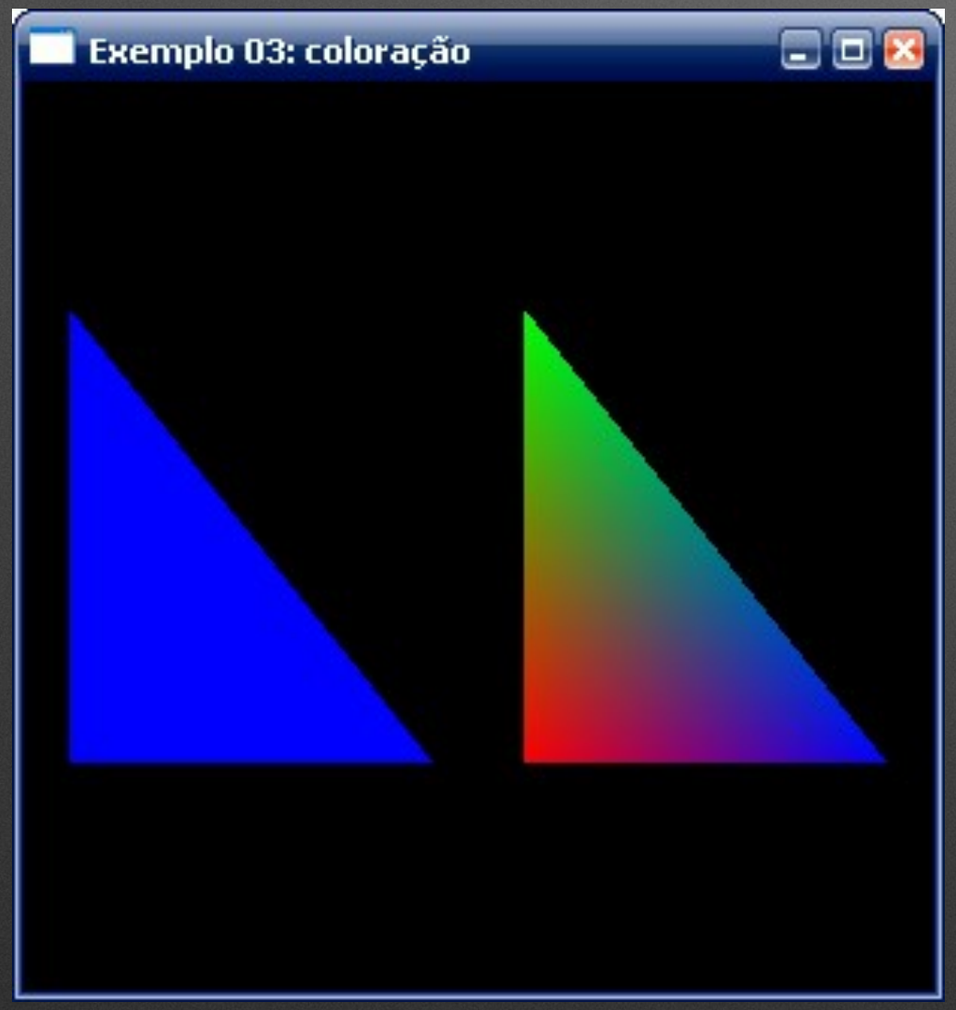
- OpenGL suporta dois tipos de coloração (preenchimento):
  - GL_SMOOTH – a cor final é resultado da interpolação entre todos os vértices da primitiva;
  - GL_FLAT – a cor final é a mesma do ultimo vértice desenhado;
- A escolha do modo de coloração a ser trabalho é definido pela função:

```
glShadeModel ( <modo> );
```

EXEMPLO



EXEMPLO (função draw)



main.cpp

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

// função de desenho
void draw ()
{
    glClear (GL_COLOR_BUFFER_BIT);

    // desenhando o triangulo da esquerda
    glShadeModel (GL_FLAT);
    glBegin (GL_TRIANGLES);
        glColor3f  ( 1.0,  0.0,  0.0);
        glVertex3f (-0.9, -0.5,  0.0);
        glColor3f  (-0.1,  1.0,  0.0);
        glVertex3f (-0.9,  0.5,  0.0);
        glColor3f  ( 0.0,  0.0,  2.0);
        glVertex3f (-0.1, -0.5,  0.0);
    glEnd ();


    // desenhando o triangulo da direita
    glShadeModel (GL_SMOOTH);
    glBegin (GL_TRIANGLES);
        glColor3f  ( 1.0,  0.0,  0.0);
        glVertex3f ( 0.1, -0.5,  0.0);
        glColor3f  ( 0.0,  1.0,  0.0);
        glVertex3f ( 0.1,  0.5,  0.0);
        glColor3f  ( 0.0,  0.0,  1.0);
        glVertex3f ( 0.9, -0.5,  0.0);
    glEnd ();

    glFlush ();
}
```

Parte IV

Posicionamentos

Campo de visualização

- OpenGL manipula toda a cena por meio de duas pilhas (matrizes):
  - GL_MODELVIEW – responsável pela modelagem e visualização da cena;
  - GL_PROJECTION – responsável pela projeção utilizada na cena;

Campo de visualização

- OpenGL manipula toda a cena por meio de duas pilhas (matrizes):
  - GL_MODELVIEW – responsável pela modelagem e visualização da cena;
  - GL_PROJECTION – responsável pela projeção utilizada na cena;
- A escolha da pilha a ser utilizada é feita pela função:

  glMatrixMode (<pilha>);
- Câmera e objetos são originalmente posicionados na origem;

Porque utilizar pilhas?

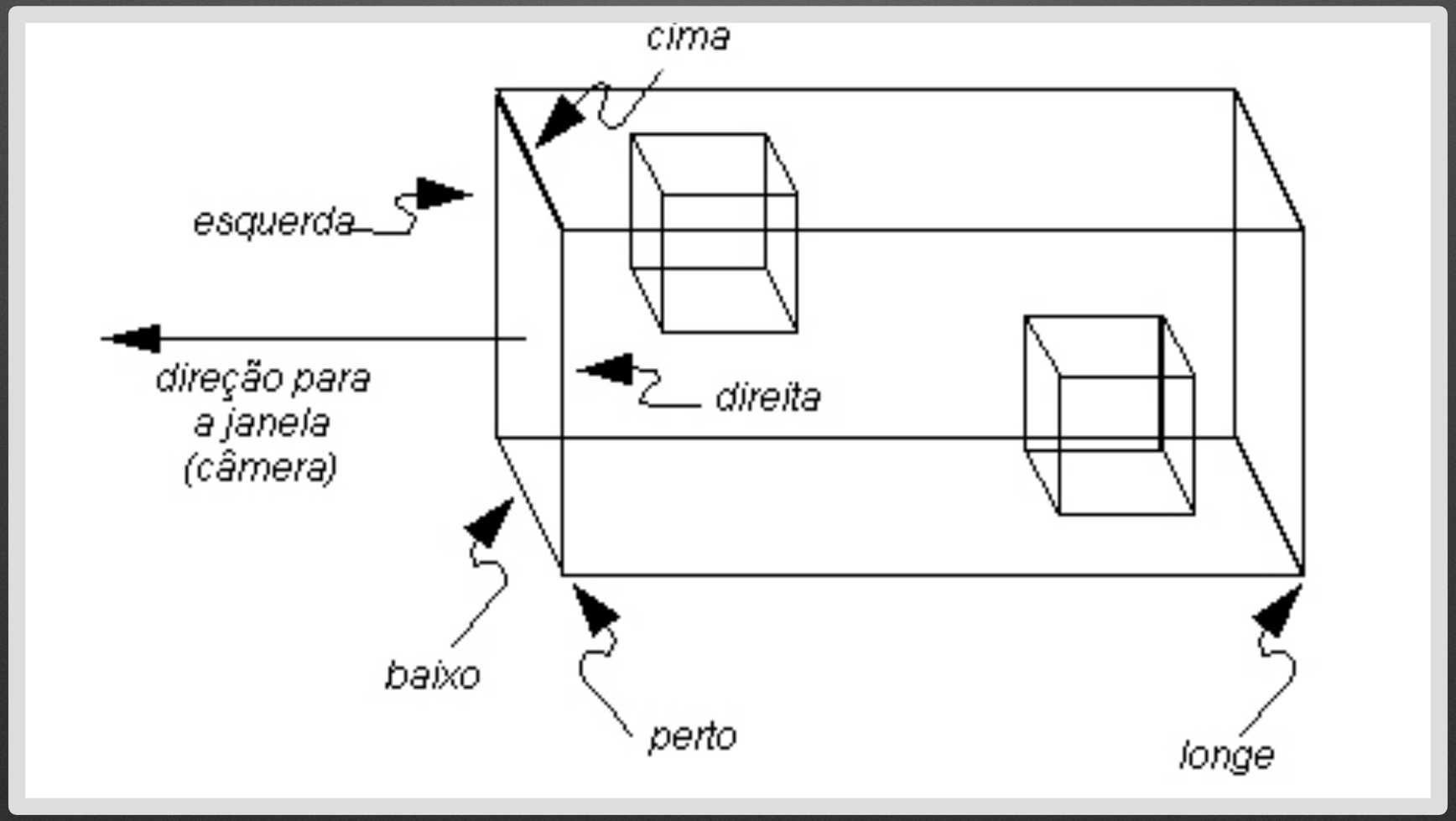
- Organizar a seqüência de operações sobre as matrizes de transformação;
- Facilitar a construção de modelos hierárquicos;
- Através da rotina glViewport(), define uma correspondência entre os vértices transformados e os pixels da tela;
- O controle sobre a pilha é realizado pelas rotinas: glPushMatrix() e glPopMatrix()

Projeções

- As projeções dividem-se em duas:
  - Ortogonal – a distância dos objetos em relação ao observador não afeta seu tamanho;
  - Perspectiva – a distância dos objetos em relação ao observador afeta seu tamanho;

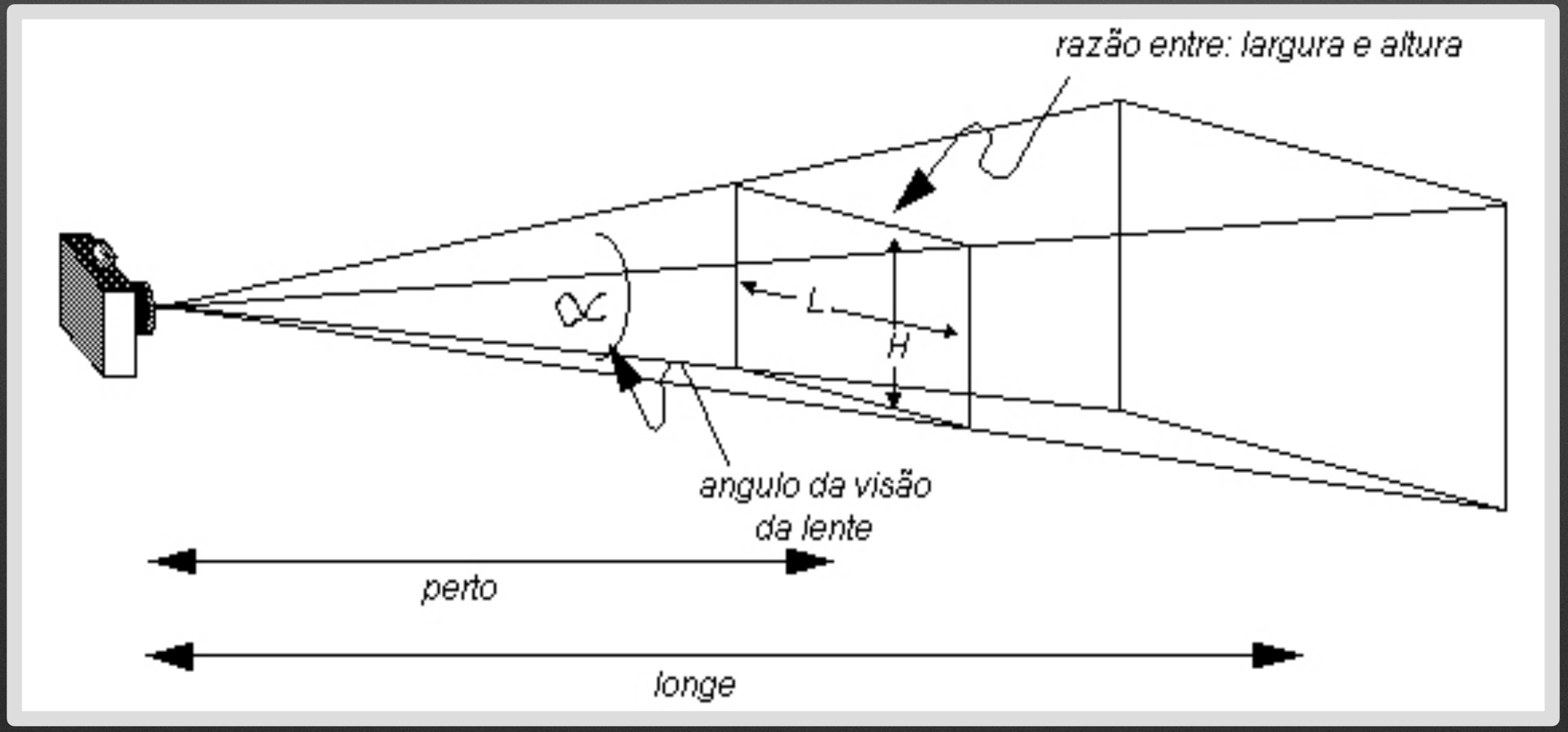
Projeção ortogonal

- glOrtho (left, right, botton, up, near, far);



Projeção perspectiva

- glFrustum(...) ou
- gluPerspective (angle, raz , near, far);



Posição do observador

- Após termos a projeção definida, temos que informar qual a posição do observador (câmera), alvo e sentido da câmera:

- gluLookAt (cx, cy, cz, ax, ay, az, vx, vy, vz);

Criando o campo de visualização

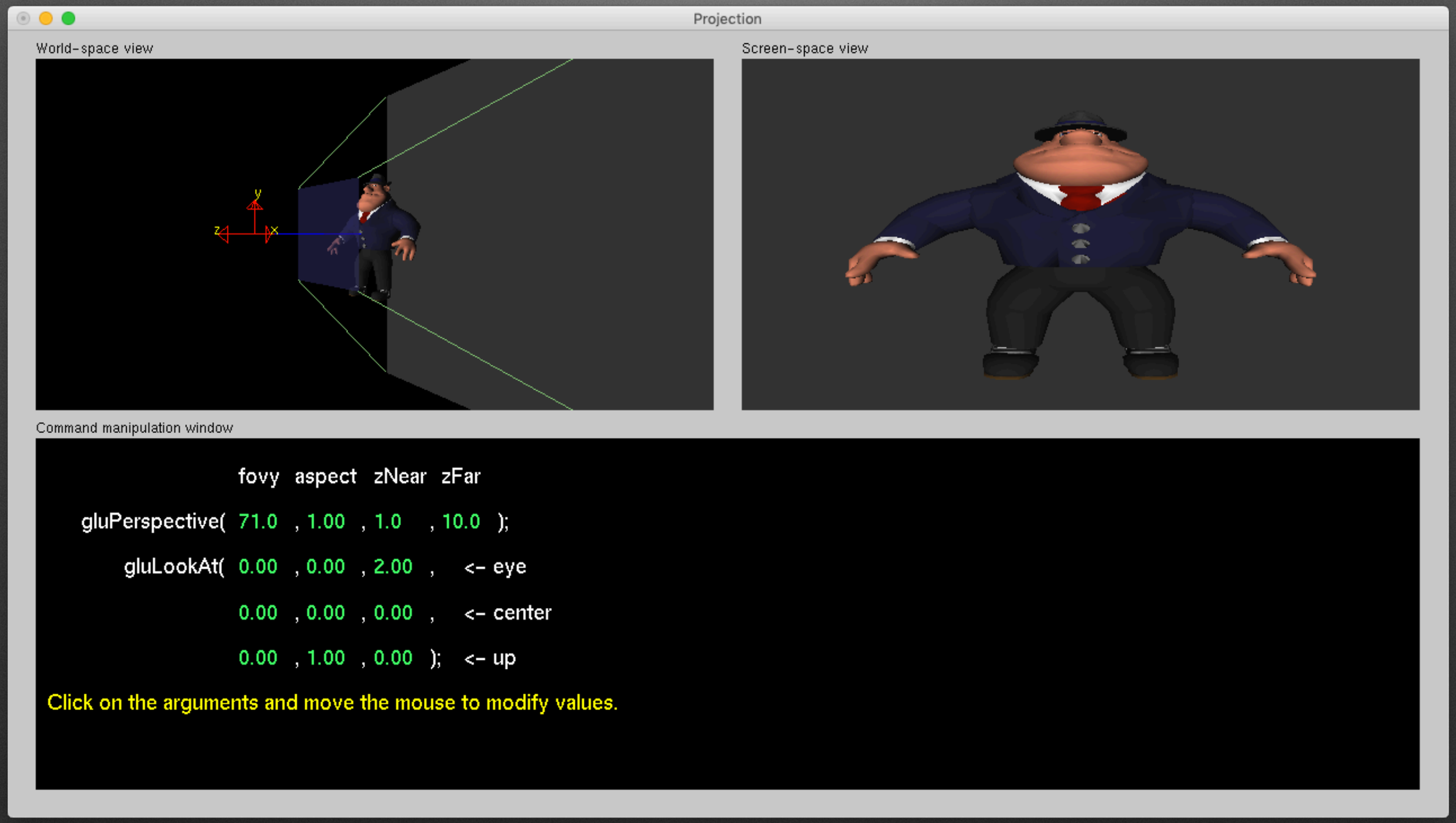
- void confCamera ( )

```
{
    glMatrixMode (GL_PROJECTION);
    gluPerspective ( ... );   {ou}  glOrtho ( ... );


    glMatrixMode (GL_MODELVIEW);
    gluLookAt ( ... );

}
```

Para um melhor entendimento



https://www.dropbox.com/s/xfgbdhquotz1o35/Exemplos%20de%20proje%C3%A7%C3%A3o%20e%20outros%20-%20openGL%20-%20Win32.zip?dl=0

glLoadIdentity ()

- Função responsável por multiplicar a matriz (pilha) atual por uma matriz identidade, resultando em uma anulação de todos os seus dados;
- Normalmente é utilizada logo após definirmos qual matriz iremos usar em glMatrixMode (...);

Parte V

GLUT

Objetos predefinidos da GLUT

- Além de gerenciar janelas e eventos, a GLUT possui funções para a criação de alguns objetos mais complexos;

Objetos predefinidos da GLUT

- Além de gerenciar janelas e eventos, a GLUT possui funções para a criação de alguns objetos mais complexos;

- Todos os seus objetos possui um critério:

  glut + <modo> + <objeto> (...);

- modo – varia entre Wire (linhas) e Solid (polígonos);
- objeto – o nome do objeto em si: cubo, esfera, etc;

Principais objetos

- Cubo:
  - glutSolidCube (raio);
- Cone:
  - glutSolidCone (raio, altura, planos, fatias);
- Esfera:
  - glutSolidSphere (raio, planos, fatias);
- Toróide (rosquinha):
  - glutSolidTorus (raio int, raio ext, planos, fatias);
- Chaleira:
  - glutSolidTeapot (raio);

glutKeyboardFunc (X);

- Função da GLUT que registra a função X para ser chamada sempre que uma tecla ASCII for pressionada;

glutKeyboardFunc (X);

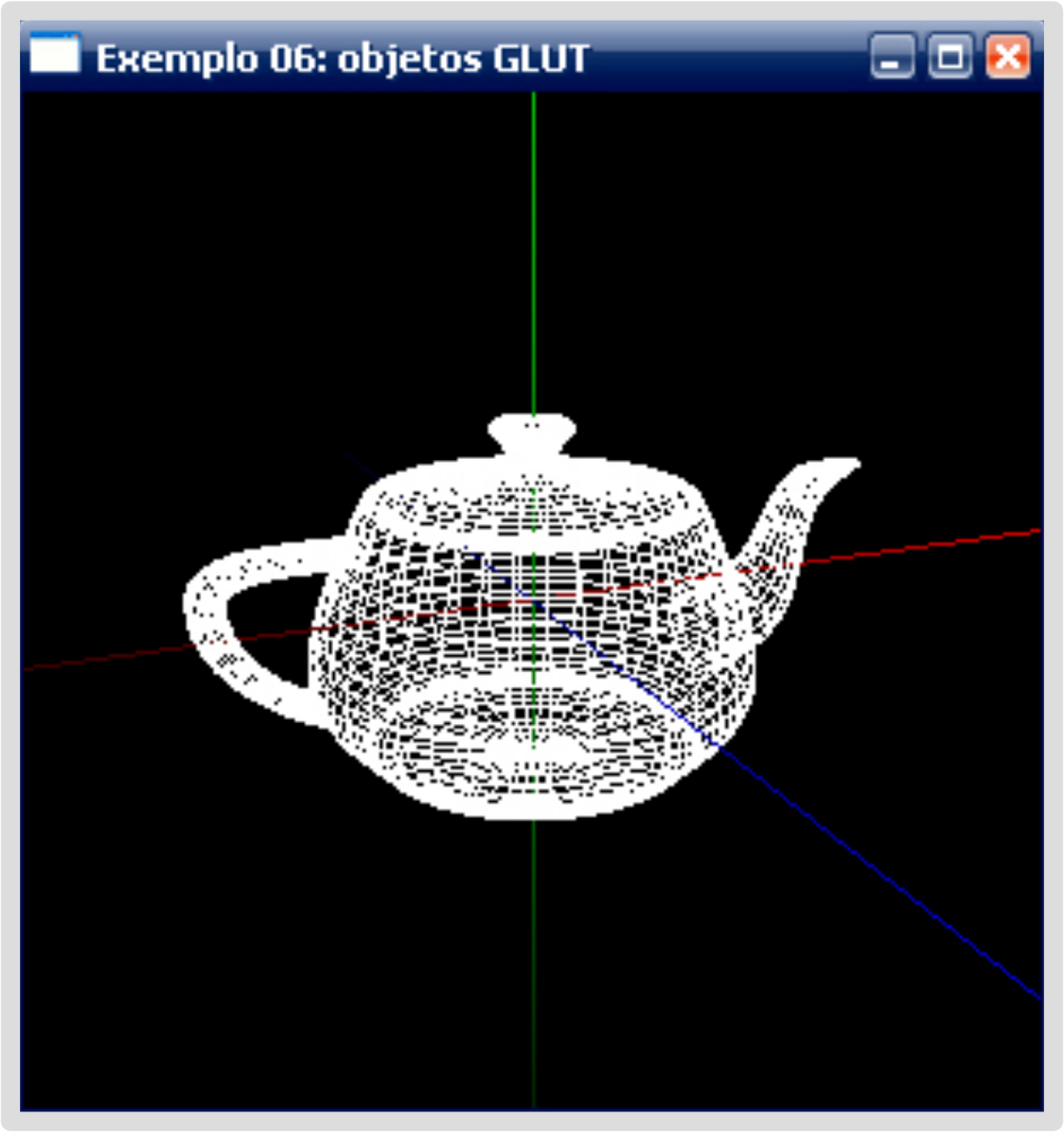
- Função da GLUT que registra a função X para ser chamada sempre que uma tecla ASCII for pressionada;

- A função X deve ter o seguinte protótipo:

```
void X (unsigned char key, int x, int y);
```

- key – guarda o código da tecla pressionada;
- x e y – possuem a coordenada do mouse no momento em que a tecla foi pressionada;

EXEMPLO



Parte VI

Transformações Geométricas

Transformações geométricas

- Por padrão OpenGL possui três transformações já definidas:
  - Rotação;
  - Translação;
  - Escalonamento;
- Todas as transformação são acumulativas!

Rotação

- glRotatef (angle, x, y, z);
- angle – representa um valor em graus que corresponderá ao giro;
- x, y e z – representam os eixos que ocorrerá o giro; cada variável assume normalmente valor zero (0) ou um (1);

Rotação

- glRotatef (angle, x, y, z);
- angle – representa um valor em graus que corresponderá ao giro;
- x, y e z – representam os eixos que ocorrerá o giro; cada variável assume normalmente valor zero (0) ou um (1);

- Ex.:
  - giro de 90° no eixo y ?

```
glRotatef (90,  0, 1, 0);
```

Translação

- glTranslatef (x, y, z);
- x, y e z – representam as novas coordenadas dentro do espaço atual, estas serão o centro do sistema de coordenadas (0.0, 0.0, 0.0);

Translação

- glTranslatef (x, y, z);
- x, y e z – representam as novas coordenadas dentro do espaço atual, estas serão o centro do sistema de coordenadas (0.0, 0.0, 0.0);

- Ex.:
  - Mover o centro do sistema de coordenadas para o ponto (0, 5, 3) ?

```
glTranslatef ( 0, 5, 3 );
```

Escalonamento

- glScalef (x, y, z);
- x, y e z – valores que serão multiplicados pelo eixo correspondente do sistema atual;

Escalonamento

- glScalef (x, y, z);
- x, y e z – valores que serão multiplicados pelo eixo correspondente do sistema atual;

- Ex.:
  - desenho duas vezes maior ?

glScalef (2, 2, 2);

Manipulando matrizes de transformação

- Analisando o seguinte código:

```
...
glRotatef (90, 0, 1, 0);
...
glRotatef (90, 0, 1, 0);
glutSolidTeapot ( 0.5 );
...
```

A chaleira terá um giro de quantos graus ?

Manipulando matrizes de transformação

- Analisando o seguinte código:

```
...
glRotatef (90, 0, 1, 0);
...
glRotatef (90, 0, 1, 0);
glutSolidTeapot ( 0.5 );
...
```

A chaleira terá um giro de quantos graus ?

180° no eixo y

Manipulando matrizes de transformação

- Problema:
  - Nem sempre queremos que as transformações sejam acumulas;
- Solução encontrada:
  - Girar o sistema de coordenadas na mesma quantidade de graus só que com sinal contrário;
    - glRotatef (90, 0, 1, 0); *// girando 90° em y*
    - glRotatef (-90, 0, 1, 0); *// desfazendo o giro*

Salvando e recuperando matrizes

- A forma mais elegante de resolver o problema seria:
  1. Salvar a matriz atual;
  2. Fazer as transformações necessárias;
  3. Restaurar a matriz já salva;

- glPushMatrix ( ) – salva;

- glPopMatrix ( ) – recupera;

Parte VII

Prática