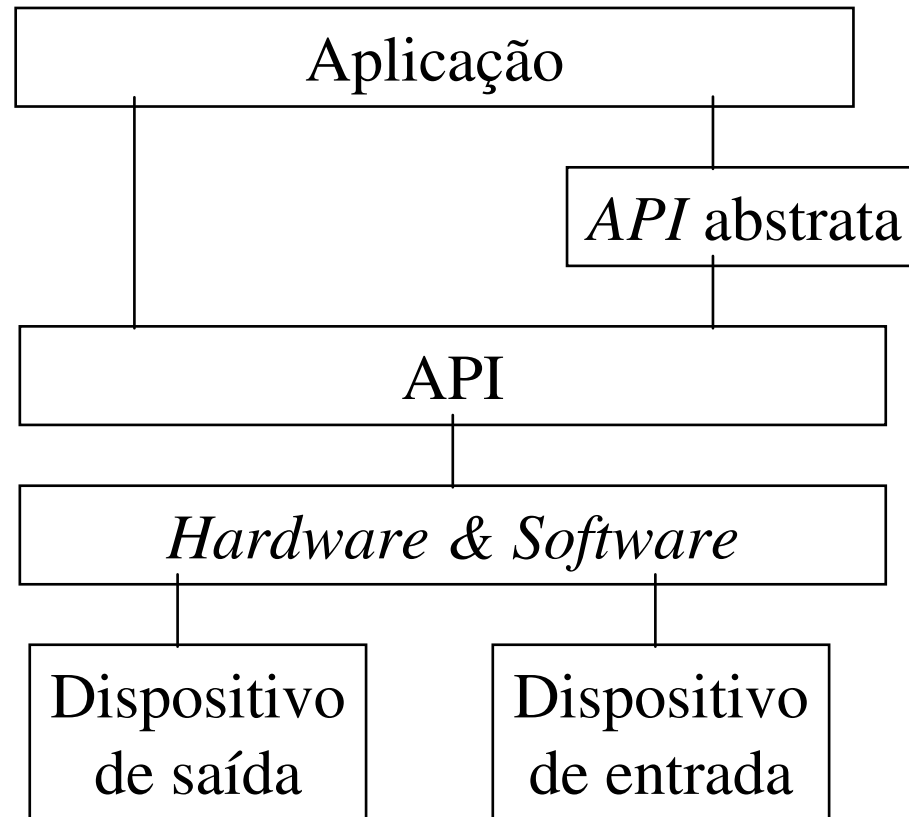


<http://www.opengl.org>

Alberto B. Raposo

OpenGL: o que é?

- API
 - Interface para programador de aplicação



Por que OpenGL?

- rápido
- relativamente simples
- arquitetura bem definida
- bem documentado
- independente de sistemas de janelas
- primitivas geométricas e imagens
- padrão
 - disponível em diversas plataformas

Bibliotecas

- O OpenGL propriamente dito:
 - **libGL.so** ou **opengl32.dll** e **opengl32.lib**
 - **GL.h** ou **gl.h**
- OpenGL Utility library (GLU): já vem com o OpenGL
 - **libGLU.so** ou **glu32.dll** e **glu32.lib**
 - **GLU.h** ou **glu.h**
- O **glut toolkit**:
 - **libglut.a** ou (**glut32.dll** e **glut32.lib**)
 - **glut.h**

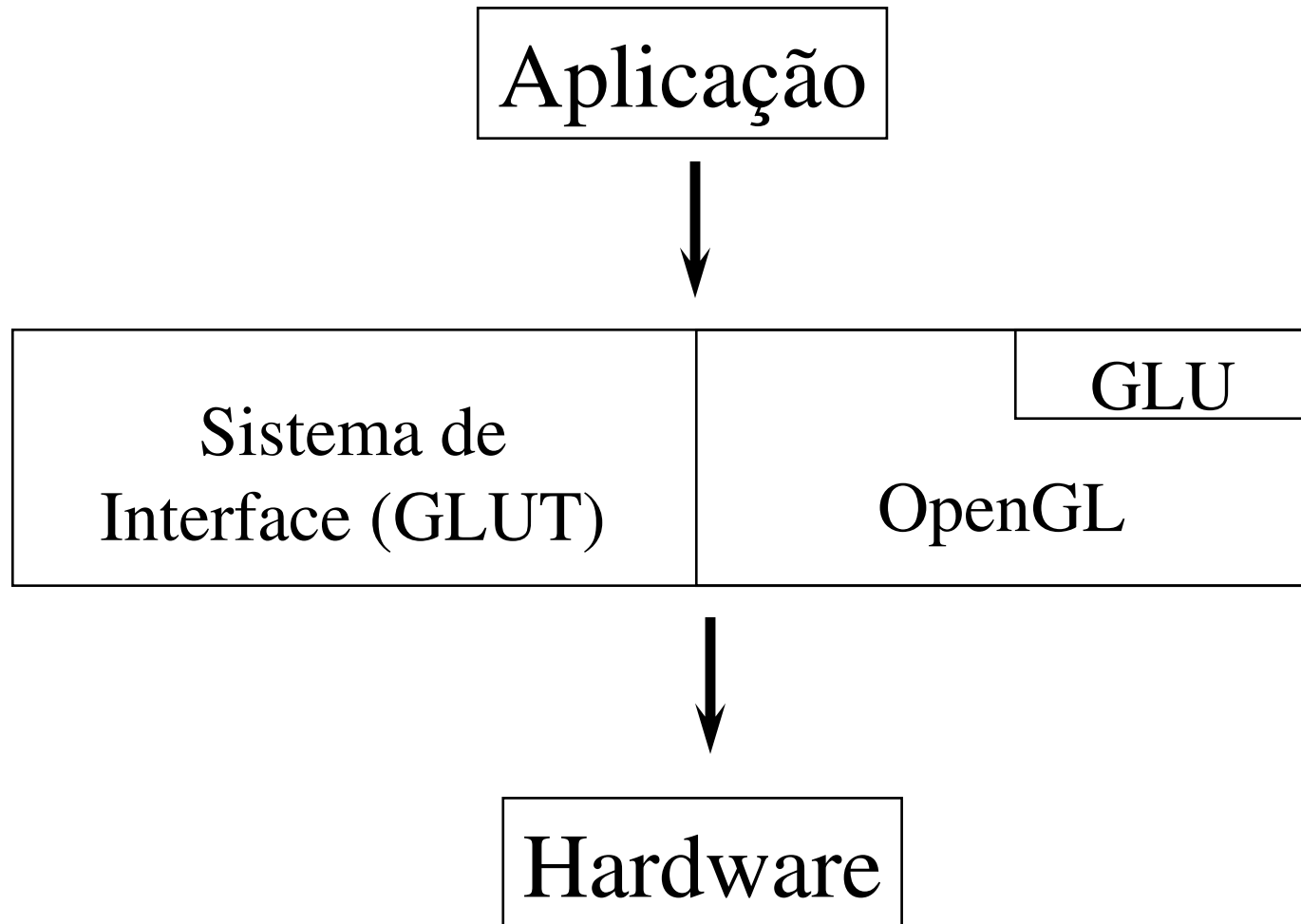
OpenGL / GLU

- Aproximadamente 250 comandos e funções
 - 200 do *core* OpenGL
 - 50 da GLU – OpenGL Utility Library
- GLU
 - contém várias rotinas que utilizam os comandos OpenGL de baixo nível para executar tarefas como *setar* as matrizes para projeção e orientação da visualização, e fazer o *rendering* de uma superfície
- No Microsoft Visual C++
 - `#include <windows.h>`
 - `#include <gl/gl.h>`
 - `#include <gl/glu.h>`

GLUT

- **GLUT - *OpenGL Utility Toolkit***
 - *toolkit* independente de plataforma
 - Por ser portátil, OpenGL não possui funções para gerenciamento de janelas, tratamento de eventos e manipulação de arquivos
 - GLUT faz isso (criação de janelas e menus popup, gerenciamento de eventos de mouse e teclado, etc)
 - <http://www.opengl.org/resources/libraries/glut/>
 - <http://www.xmission.com/~nate/glut.html>
 - criada para facilitar o aprendizado e a elaboração de programas OpenGL
 - independente do ambiente de programação
 - pegar *glut.dll* e *glut32.dll*

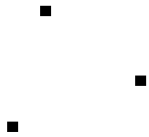
Aplicação típica



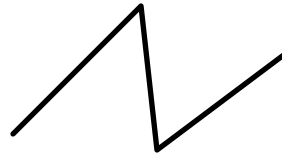
Nomes de funções

- Todos os nomes das funções OpenGL seguem uma convenção que indica de qual biblioteca a função faz parte e, freqüentemente, quantos e que tipos de argumentos a função tem.
 - **<PrefixoBiblioteca> <ComandoRaiz>
<ContadorArgumentosOpcional>
<TipoArgumentosOpcional>**
- Exemplos:
 - *glColor3f* possui *Color* como raiz. O prefixo *gl* representa a biblioteca *gl*, e o sufixo *3f* significa que a função possui três valores de ponto flutuante como parâmetro.
 - Variações da função: recebe três valores inteiros como parâmetro (*glColor3i*), três *doubles* (*glColor3d*) e assim por diante.
 - *glVertex2f...*

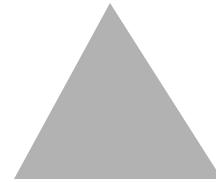
Primitivas geométricas básicas



Ponto



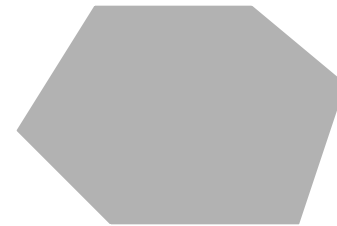
Linha



Triângulo

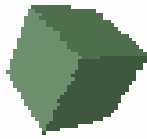


Quadrado



Polígono

Objetos 3D



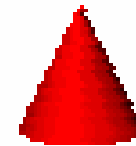
Polyhedra



Sphere



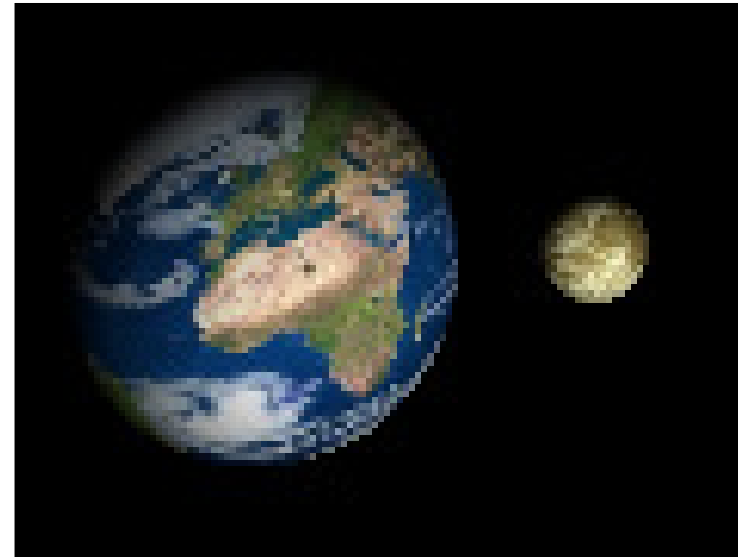
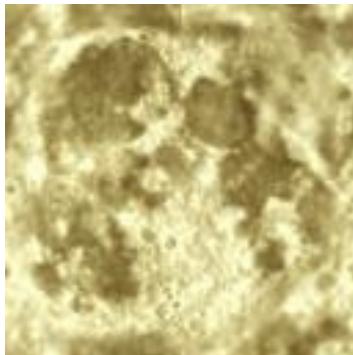
Bezier Surfaces



Quadric

From SIGGRAPH'97 course

Imagem e Textura

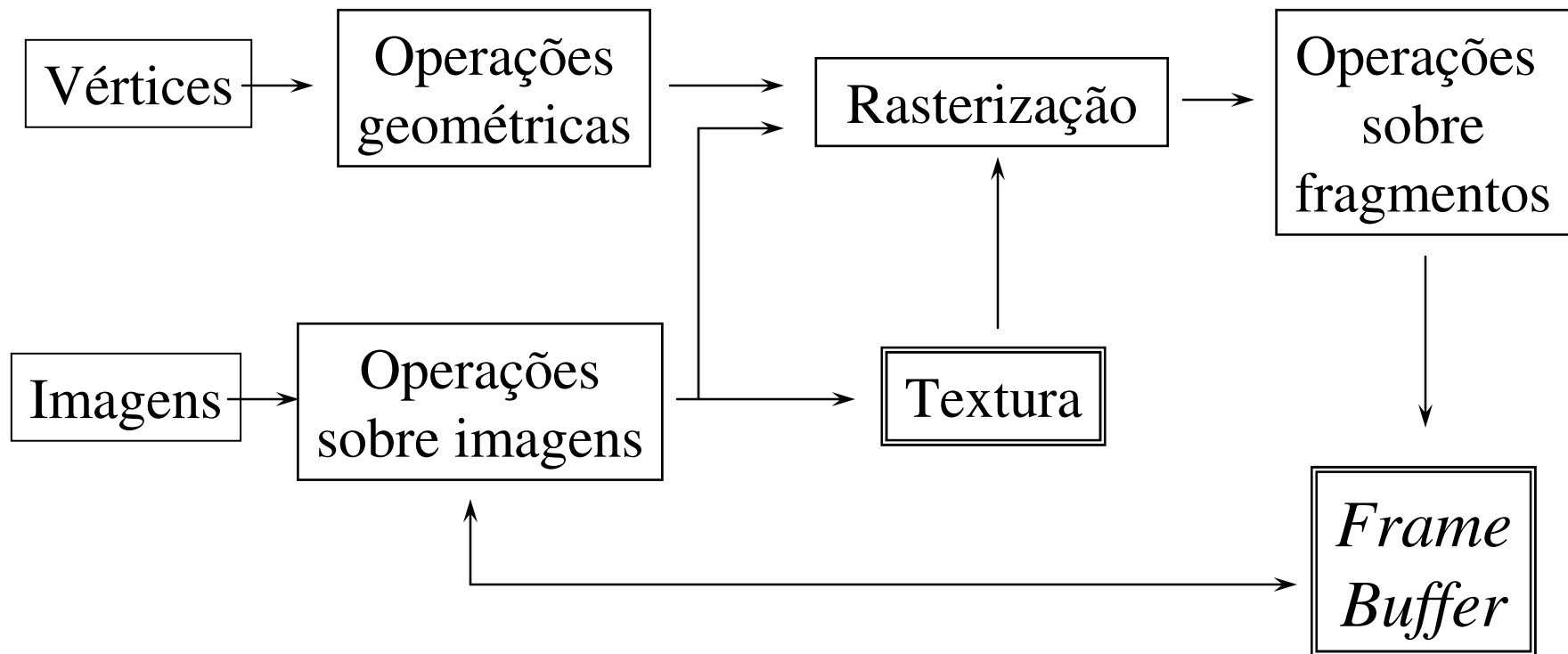


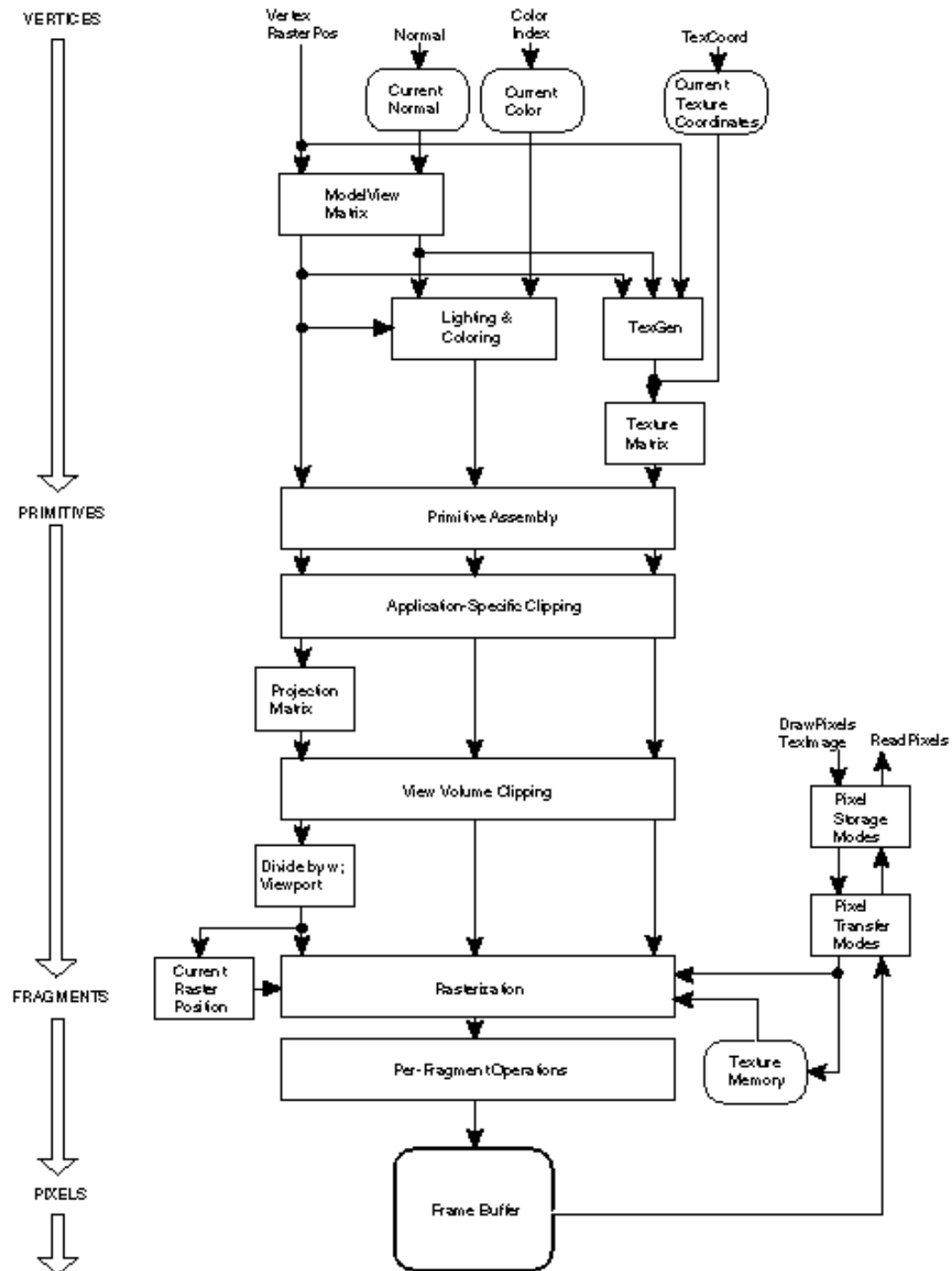
Pipeline OpenGL



- Como uma aplicação faz chamadas às funções API OpenGL, os comandos são colocados em um *buffer* de comandos. Este *buffer* é preenchido com comandos, vértices, dados de textura, etc. Quando este *buffer* é "esvaziado", os comandos e dados são passados para o próximo estágio
- Após a etapa de aplicação das transformações geométricas e da iluminação, é feita a rasterização, isto é, é gerada a imagem a partir dos dados geométricos, de cor e textura. A imagem final, então, é colocada no *frame buffer*, que é a memória do dispositivo gráfico. Isto significa que a imagem é exibida no monitor .

Pipeline de renderização





Máquina de Estados

- OpenGL é uma máquina de estados. É possível colocá-la em vários estados (ou modos) que não são alterados, a menos que uma função seja chamada para isto.
 - Ex.: a cor corrente é uma variável de estado que pode ser definida como branco. Todos os objetos, então, são desenhados com a cor branca, até o momento em que outra cor corrente é especificada.
 - Outros exemplos de variáveis de estado: estilo (padrão) de linha, posições e características de luzes, propriedades de materiais de objetos que estão sendo desenhados, etc.
 - Agrupar objetos por estados pode ser interessante para ganhar performance na etapa de renderização (i.e., envia-se objetos com mesmos estados em sequência para a GPU, diminuindo os comandos de troca de estados).

Maquina de Estados

- Trecho de programa a seguir mostra um exemplo da utilização dos estados.

```
int luz;  
:  
glEnable(GL_LIGHTING); //Habilita luz - GL_LIGHTING é a variável de estado  
:  
luz = glIsEnabled(GL_LIGHTING); // retorna 1 (verdadeiro)  
:  
glDisable(GL_LIGHTING); //Desabilita luz  
:  
luz = glIsEnabled(GL_LIGHTING); // retorna 0 (falso)  
:
```


Primeiro Exemplo

```
// 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();
}

// Inicializa parâmetros de rendering
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(void)
{
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("Primeiro Programa");
    glutDisplayFunc(Desenha);
    Inicializa();
    glutMainLoop();
}
```

Define modo de operação da GLUT

“Título” da janela criada

Define a função de desenho

glutInitDisplayMode

- Parâmetros
 - GLUT_SINGLE: usa apenas um buffer de cor. A função que efetua exibição da imagem é *glFlush*
 - GLUT_DOUBLE: usa 2 buffers de cor. Um (visível) armazena a imagem mostrada e outro (invisível) armazena a imagem que está sendo construída. Ao finalizar a construção da imagem, força-se a troca dos buffers com a função *glutSwapBuffers*
 - Double buffering é bom para animações
 - GLUT_DEPTH: define que se necessita do Z-buffer
 - GLUT_RGB/GLUT_RGBA: define que cores são especificadas por componentes RGB e A (alfa – transparência)
 - GLUT_INDEX: cores do programa são especificadas por tabela de cores

Definindo primitivas

- Sempre entre *glBegin* e *glEnd*

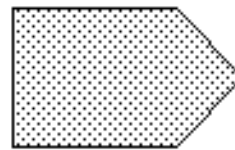
```
glBegin(tipo_de_prim);
```

...define atributo de vértice

...define vértice

```
glEnd();
```

```
glBegin(GL_POLYGON);  
    glVertex2f(0.0, 0.0);  
    glVertex2f(0.0, 3.0);  
    glVertex2f(3.0, 3.0);  
    glVertex2f(4.0, 1.5);  
    glVertex2f(3.0, 0.0);  
glEnd();
```

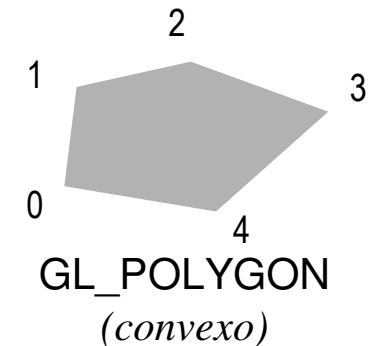
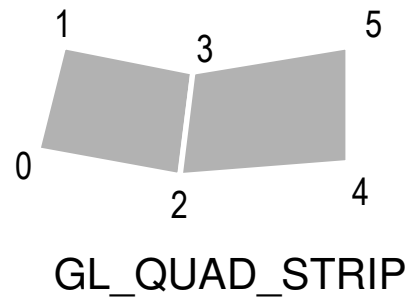
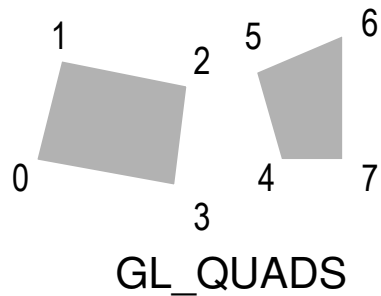
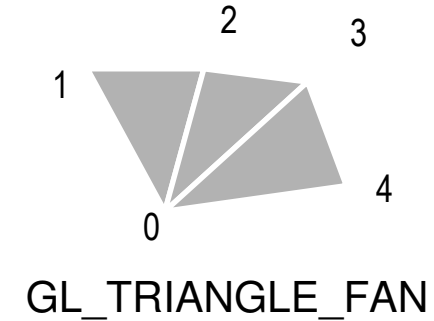
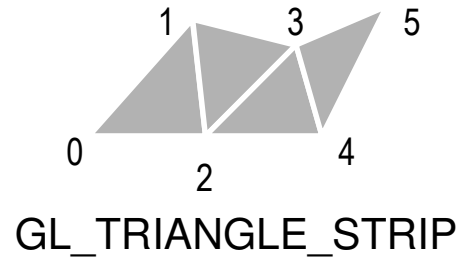
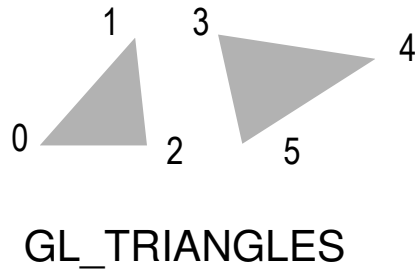
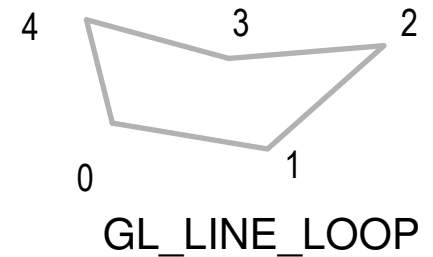
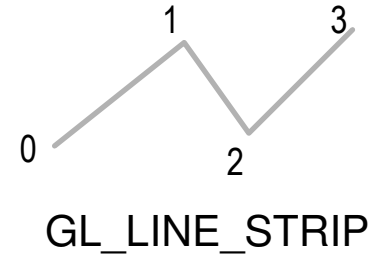
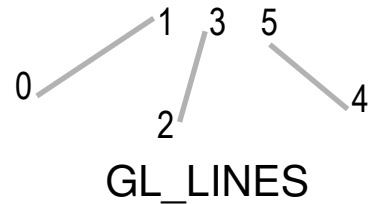


GL_POLYGON



GL_POINTS

Tipos de primitivas em OpenGL



Segundo Exemplo

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

// Função callback de redesenho da janela de visualização
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);
        glVertex3f(-0.5,-0.5,0);
        glVertex3f( 0.0, 0.5,0);
        glVertex3f( 0.5,-0.5,0);
    glEnd();

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

OpenGL: Uma Abordagem Prática e Objetiva
M. Cohen e I. H. Manssour

Segundo Exemplo (cont.)

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

// Função responsável por inicializar 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);
}
```

Define a janela quando se trabalha com projeção 2D:
Parâmetros: x mínimo, x máximo, y mínimo, y máximo

Indica qual matriz estará no topo da pilha para operações subsequentes:

GL_MODELVIEW: matriz do modelo

GL_PROJECTION: matriz de projeção

GL_TEXTURE: matriz de textura

Segundo Exemplo (cont.)

```
// Programa Principal
int main(void)
{
    // Define do modo de operação da GLUT
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);

    // Especifica o tamanho inicial em pixels da janela GLUT
    glutInitWindowSize(400,400);

    // Cria a janela passando como argumento o título da mesma
    glutCreateWindow("Primeiro Programa");

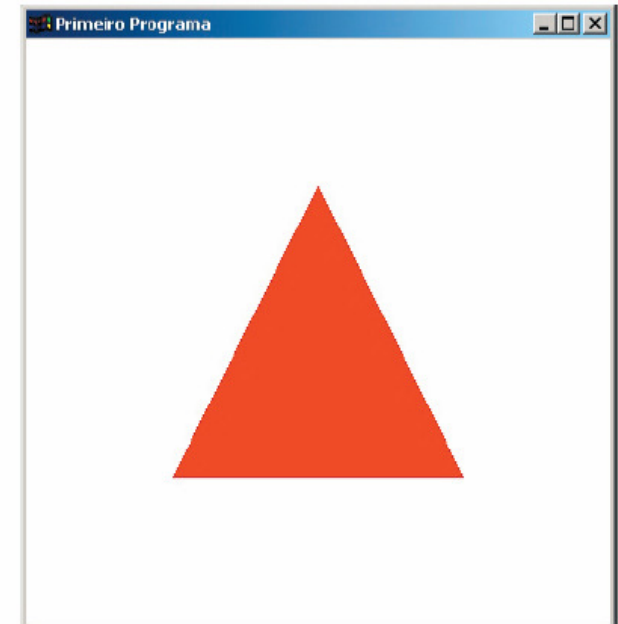
    // Registra a função callback de redesenho da janela de visualização
    glutDisplayFunc(Desenha);

    // Registra a função callback para tratamento das teclas ASCII
    glutKeyboardFunc (Teclado);

    // Chama a função responsável por fazer as inicializações
    Inicializa();

    // Inicia o processamento e aguarda interações do usuário
    glutMainLoop();

    return 0;
}
```



Terceiro Exemplo

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

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

```
// Função callback chamada para fazer o desenho
```

```
void Desenha(void)
```

```
{
```

```
    glMatrixMode(GL_MODELVIEW);
```

```
    glLoadIdentity();
```

Inicializa sistema de coordenadas: sem esse comando, poderia dar erro



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

```
    // Especifica que a cor corrente é vermelha
```

```
    //      R      G      B
    glColor3f(1.0f, 0.0f, 0.0f);
```

```
    // Desenha um quadrado preenchido com a cor corrente
```

```
    glBegin(GL_QUADS);
```

```
        glVertex2i(100,150);
```

```
        glVertex2i(100,100);
```

```
        // Especifica que a cor corrente é azul
```

```
        glColor3f(0.0f, 0.0f, 1.0f);
```

```
        glVertex2i(150,100);
```

```
        glVertex2i(150,150);
```

```
    glEnd();
```

```
    // Executa os comandos OpenGL
```

```
    glFlush();
```

```
}
```


Terceiro Exemplo (cont.)

```
// Inicializa parâmetros de rendering
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)
{
    // Evita a divisao por zero
    if(h == 0) h = 1;

    // Especifica as dimensões da Viewport
    glViewport(0, 0, w, h);

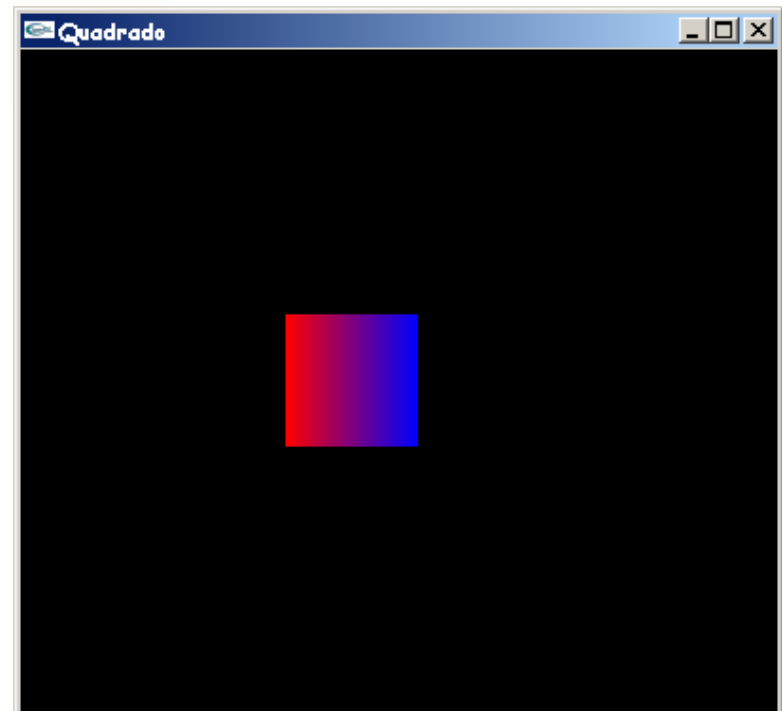
    // Inicializa o sistema de coordenadas
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Estabelece a janela de seleção (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);
}
```

A *viewport* define a área dentro janela, em coordenadas de tela, que OpenGL pode usar para fazer o desenho. O volume de visualização é, então, mapeado para a nova *viewport*. Os 2 primeiros parâmetros são as coordenadas do canto inferior esquerdo na tela, e as 2 últimas são largura e altura.

Terceiro Exemplo (cont.)

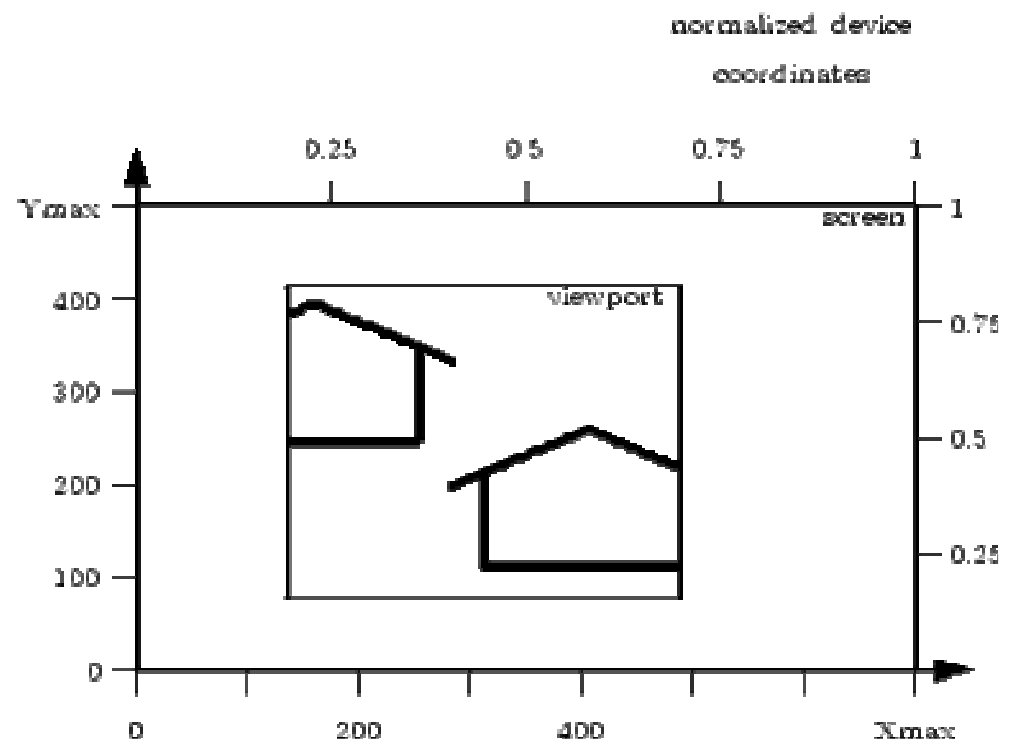
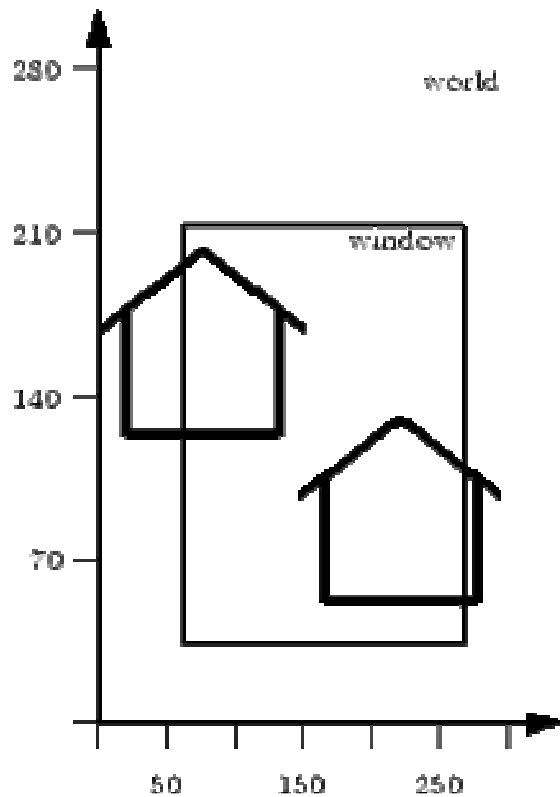
```
// Programa Principal
int main(void)
{
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(400,350);
    glutInitWindowPosition(10,10);
    glutCreateWindow("Quadrado");
    glutDisplayFunc(Desenha);
    glutReshapeFunc(AlteraTamanhoJanela);
    Inicializa();
    glutMainLoop();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



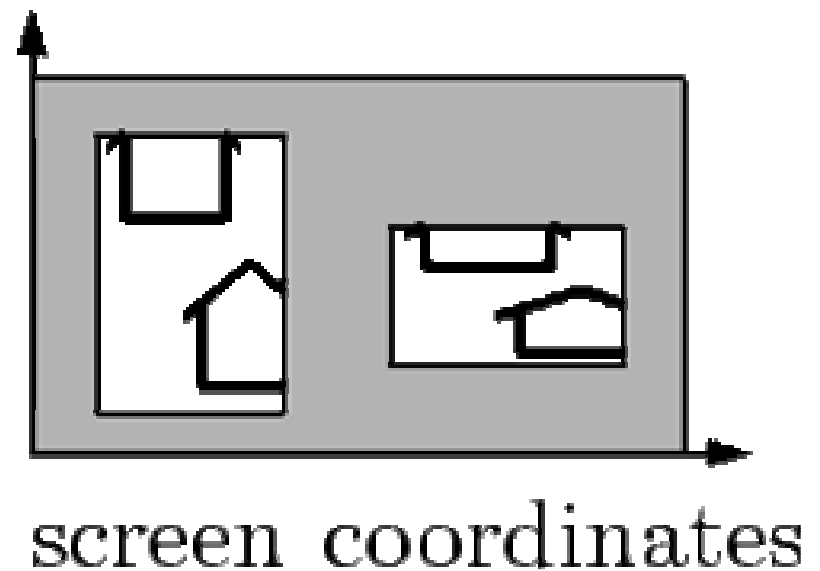
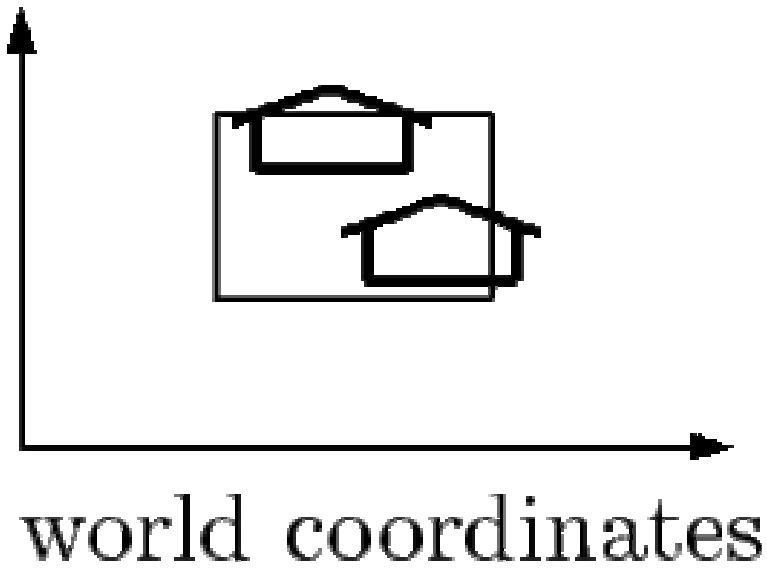
Window vs Viewport

- Window
 - Parte do universo que é do interesse (parte do mundo que está sendo visualizada).
 - SRU: Sistema de Referência do Universo
- Viewport
 - Parte do monitor onde será visualizado o conteúdo

Window vs Viewport



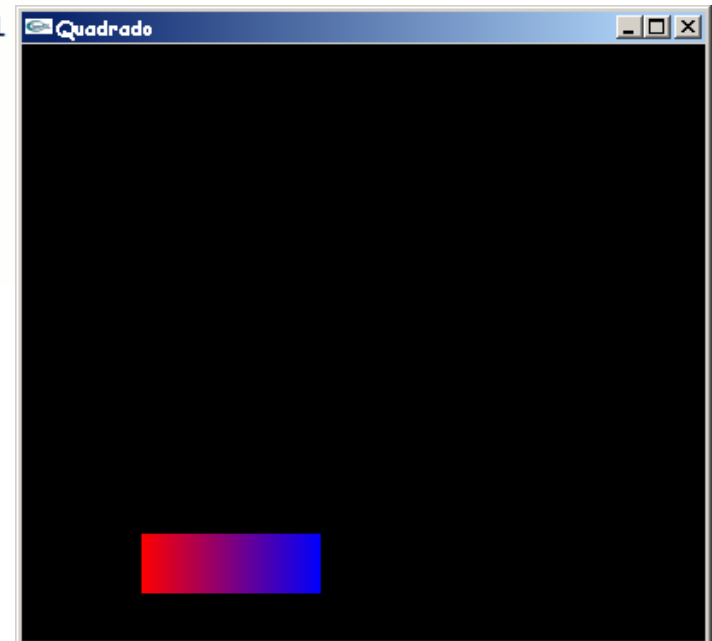
Window vs Viewport



Transformações geométricas

```
void Desenha(void)
{
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f, 0.0f, 0.0f);
    glTranslatef(-100.0f, -30.0f, 0.0f);
    glScalef(1.5f, 0.5f, 1.0f);

    glBegin(GL_QUADS);
        glVertex2i(100,150);
        glVertex2i(100,100);
        // Especifica que a cor corrente é azul
        glColor3f(0.0f, 0.0f, 1.0f);
        glVertex2i(150,100);
        glVertex2i(150,150);
    glEnd();
    glFlush();
}
```



Quarto Exemplo

```
// TeaPot3D.c - Isabel H. Manssour
// Um programa OpenGL que exemplifica a visualização
// de objetos 3D.
// Este código está baseado nos exemplos disponíveis no livro
// "OpenGL SuperBible", 2nd Edition, de Richard S. e Wright Jr.

#include <gl/glut.h>

GLfloat angle, fAspect;

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

    glColor3f(0.0f, 0.0f, 1.0f);

    // Desenha o teapot com a cor corrente (wire-frame)
    glutWireTeapot(50.0f);

    // Executa os comandos OpenGL
    glutSwapBuffers();
}

// Inicializa parâmetros de rendering
void Inicializa (void)
{
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);
    angle=45;
}
```

Primitiva “clássica” em CG: bule

Quarto Exemplo (cont.)

```
// Função usada para especificar o volume de visualização
void EspecificaParametrosVisualizacao(void)
{
    // Especifica sistema de coordenadas de projeção
    glMatrixMode(GL_PROJECTION);
    // Inicializa sistema de coordenadas de projeção
    glLoadIdentity();

    // Especifica a projeção perspectiva
    gluPerspective(angle, fAspect, 0.1, 500);

    // Especifica sistema de coordenadas do modelo
    glMatrixMode(GL_MODELVIEW);
    // Inicializa sistema de coordenadas do modelo
    glLoadIdentity();

    // Especifica posição do observador e do alvo
    gluLookAt(0, 80, 200, 0, 0, 0, 0, 1, 0);
}

// Função callback chamada quando o tamanho da janela é alterado
void AlteraTamanhoJanela(GLsizei w, GLsizei h)
{
    // Para prevenir uma divisão por zero
    if ( h == 0 ) h = 1;

    // Especifica o tamanho da viewport
    glViewport(0, 0, w, h);

    // Calcula a correção de aspecto
    fAspect = (GLfloat)w/(GLfloat)h;

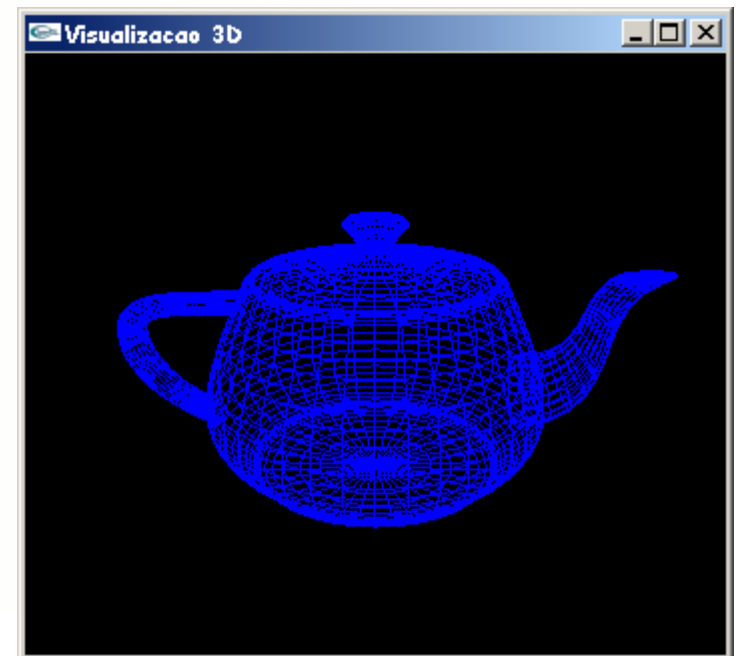
    EspecificaParametrosVisualizacao();
}
```


Quarto Exemplo (cont.)

```
// Função callback chamada para gerenciar eventos do mouse
void GerenciaMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
        if (state == GLUT_DOWN) { // Zoom-in
            if (angle >= 10) angle -= 5;
        }
    if (button == GLUT_RIGHT_BUTTON)
        if (state == GLUT_DOWN) { // Zoom-out
            if (angle <= 130) angle += 5;
        }
    EspecificaParametrosVisualizacao();
    glutPostRedisplay();
}

// Programa Principal
int main(void)
{
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(350,300);
    glutCreateWindow("Visualizacao 3D");
    glutDisplayFunc(Desenha);
    glutReshapeFunc(AlteraTamanhoJanela);
    glutMouseFunc(GerenciaMouse);
    Inicializa();
    glutMainLoop();
}
```

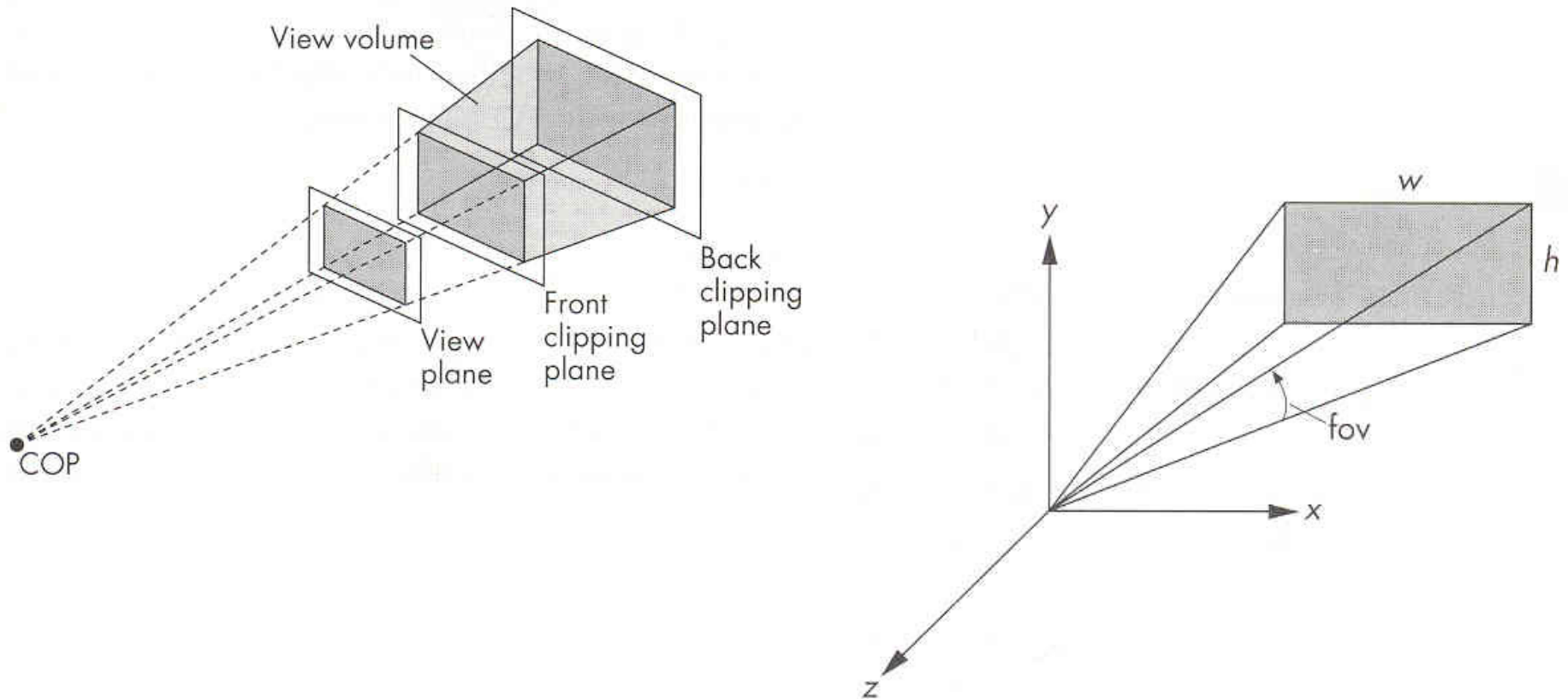
marks the normal plane of current window as needing to be redisplayed.



gluPerspective(angle,fAspect,0.1,500); segundo [Wright 2000], esta função estabelece os parâmetros da Projeção Perspectiva, atualizando a matriz de projeção perspectiva. Seu protótipo é: *void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);*. Descrição dos parâmetros: *fovy* é o ângulo, em graus, na direção y (usada para determinar a "altura" do volume de visualização); *aspect* é a razão de aspecto que determina a área de visualização na direção x, e seu valor é a razão em x (largura) e y (altura); *zNear*, que sempre deve ter um valor positivo maior do que zero, é a distância do observador até o plano de corte mais próximo (em z); *zFar*, que também sempre tem um valor positivo maior do que zero, é a distância do observador até o plano de corte mais afastado (em z). Esta função sempre deve ser definida ANTES da função *gluLookAt*, e no modo GL_PROJECTION.

gluLookAt(0,80,200, 0,0,0, 0,1,0); define a transformação de visualização. Através dos seus argumentos é possível indicar a posição da câmera e para onde ela está direcionada. Seu protótipo é: *void gluLookAt(GLdouble eyex, GLdouble eyey, GLdouble eyez, GLdouble centerx, GLdouble centery, GLdouble centerz, GLdouble upx, GLdouble upy, GLdouble upz);*. Descrição dos parâmetros: *eyex*, *eyey* e *eyez* são usados para definir as coordenadas x, y e z, respectivamente, da posição da câmera (ou observador); *centerx*, *centery* e *centerz* são usados para definir as coordenadas x, y e z, respectivamente, da posição do alvo, isto é, para onde o observador está olhando (normalmente, o centro da cena); *upx*, *upy* e *upz* são as coordenadas x, y e z, que estabelecem o vetor *up* (indica o "lado de cima" de uma cena 3D) [Wright 2000].

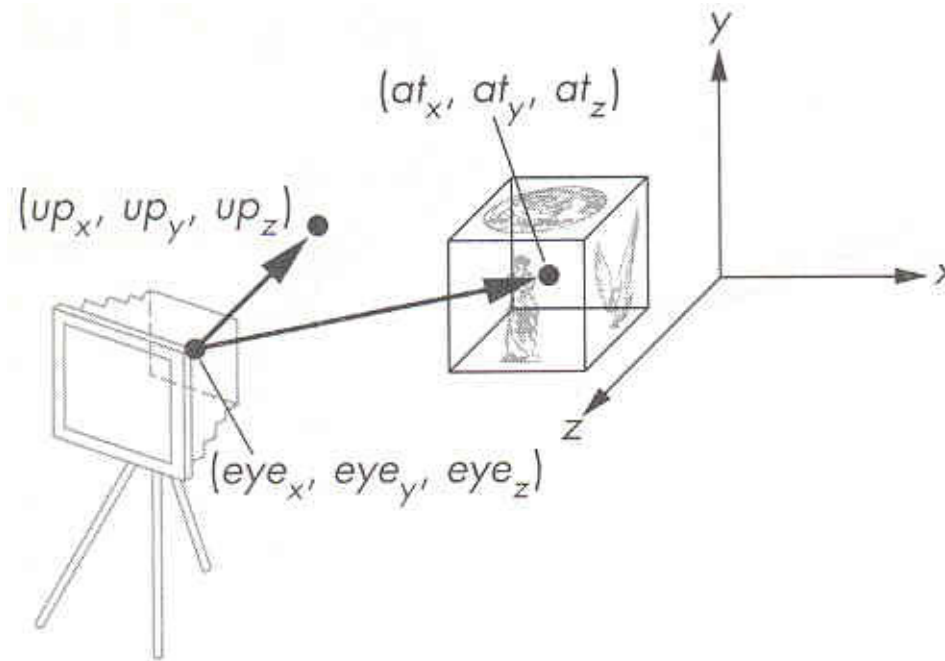
gluPerspective



```
glMatrixMode(GL_PROJECTION);  
glLoadIdentity();  
gluPerspective(fovy, aspect, near, far);
```

w / h

gluLookAt



```
gluLookAt(eyex, eyey, eyez, atx, aty, atz, upx, upy, upz);
```

Outras Primitivas 3D

- `void glutWireCube(GLdouble size);`
- `void glutWireSphere(GLdouble radius, GLint slices, GLint stacks);`
- `void glutWireCone(GLdouble radius, GLdouble height, GLint slices, GLint stacks);`
- `void glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint nsides, GLint rings);`
- `void glutWireIcosahedron(void);`
- `void glutWireOctahedron(void);`
- `void glutWireTetrahedron(void);`
- `void glutWireDodecahedron(GLdouble radius);`

Os parâmetros *slices* e *stacks* que aparecem no protótipo de algumas funções, significam, respectivamente, o número de subdivisões **em torno do eixo z** (como se fossem linhas longitudinais) e o número de subdivisões **ao longo do eixo z** (como se fossem linhas latitudinais). Já *rings* e *nsides* correspondem, respectivamente, ao número de seções que serão usadas para formar o *torus*, e ao número de subdivisões para cada seção. A figura 14.2 exibe um exemplo de um *torus* com *rings*=6 e *nsides*=20, e a figura 14.3 exibe um exemplo com *rings*=20 e *nsides*=20.

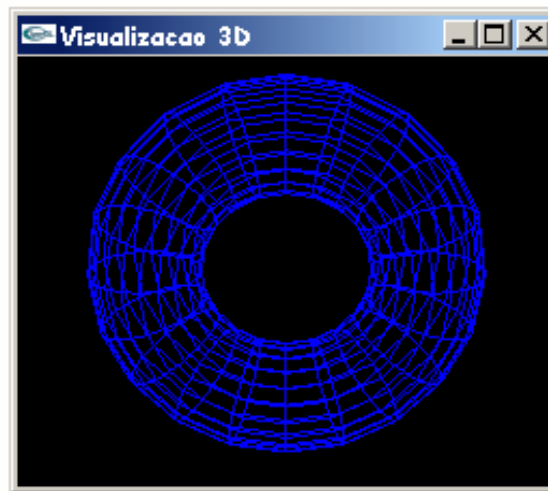
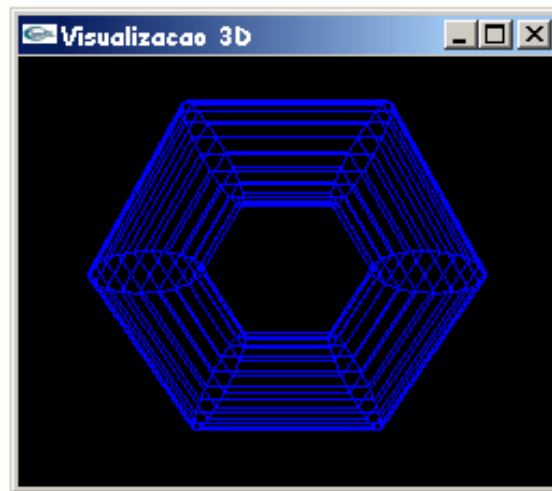


Figura 14.2 - *Torus* (*rings*=6, *nsides*=20) Figura 14.3 - *Torus* (*rings*=20, *nsides*=20)

<http://www.inf.pucrs.br/~manssour/OpenGL/Tutorial.html>

Solid

- Todas as primitivas anteriores podem ser também sólidas, ao invés de Wireframe:
 - Ex.: *glutSolidTeapot(50.0f)*



Exemplo de Animação

```
// Anima.c - Isabel H. Manssour
// Um programa OpenGL simples que mostra a animação
// de quadrado em uma janela GLUT.
// Este código está baseado no Bounce.c, exemplo
// disponível no livro "OpenGL SuperBible",
// 2nd Edition, de Richard S. e Wright Jr.

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

// Tamanho e posição inicial do quadrado
GLfloat x1 = 100.0f;
GLfloat y1 = 150.0f;
GLsizei rsize = 50;

// Tamanho do incremento nas direções x e y
// (número de pixels para se mover a cada
// intervalo de tempo)
GLfloat xstep = 1.0f;
GLfloat ystep = 1.0f;

// Largura e altura da janela
GLfloat windowHeight;
```

Exemplo de Animação (cont.)

```
// 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 vermelha
    //           R      G      B
    glColor3f(1.0f, 0.0f, 0.0f);

    // Desenha um quadrado preenchido com a cor corrente
    glBegin(GL_QUADS);
        glVertex2i(x1,y1+rsize);
        glVertex2i(x1,y1);
        // Especifica que a cor corrente é azul
        glColor3f(0.0f, 0.0f, 1.0f);
        glVertex2i(x1+rsize,y1);
        glVertex2i(x1+rsize,y1+rsize);
    glEnd();

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


C

```
// Função callback chamada pela GLUT a cada intervalo de tempo
// (a window não está sendo redimensionada ou movida)
void Timer(int value)
{
    // Muda a direção quando chega na borda esquerda ou direita
    if(x1 > windowWidth-rsize || x1 < 0)
        xstep = -xstep;

    // Muda a direção quando chega na borda superior ou inferior
    if(y1 > windowHeight-rsize || y1 < 0)
        ystep = -ystep;

    // Verifica as bordas. Se a window for menor e o
    // quadrado sair do volume de visualização
    if(x1 > windowWidth-rsize)
        x1 = windowWidth-rsize-1;

    if(y1 > windowHeight-rsize)
        y1 = windowHeight-rsize-1;

    // Move o quadrado
    x1 += xstep;
    y1 += ystep;

    // Redesenha o quadrado com as novas coordenadas
    glutPostRedisplay();
    glutTimerFunc(33,Timer, 1);
}
```

Tem que chamar a
função timer de novo



Exemplo de Animação (cont.)

```
// Inicializa parâmetros de rendering
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)
{
    // Evita a divisao por zero
    if(h == 0) h = 1;

    // Especifica as dimensões da Viewport
    glViewport(0, 0, w, h);


    // Inicializa o sistema de coordenadas
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();

    // Estabelece a janela de seleção (left, right, bottom, top)
    if (w <= h) {
        windowHeight = 250.0f*h/w;
        windowWidth = 250.0f;
    }
    else {
        windowWidth = 250.0f*w/h;
        windowHeight = 250.0f;
    }

    gluOrtho2D(0.0f, windowWidth, 0.0f, windowHeight);
}
```

Exemplo de Animação (cont.)

```
// Programa Principal
int main(void)
{
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
    glutInitWindowSize(400,350);
    glutInitWindowPosition(10,10);
    glutCreateWindow("Anima");
    glutDisplayFunc(Desenha);
    glutReshapeFunc(AlterarTamanhoJanela);
    glutTimerFunc(33, Timer, 1);
    Inicializa();
    glutMainLoop();
}
```



glutTimerFunc(33, Timer, 1); estabelece a função *Timer* previamente definida como a função *callback* de animação. Seu protótipo é: *void glutTimerFunc(unsigned int msec, void (*func)(int value), int value);*. Esta função faz a GLUT esperar *msecs* milissegundos antes de chamar a função *func*. É possível passar um valor definido pelo usuário no parâmetro *value*. Como esta função é "disparada" apenas uma vez, para se ter uma animação contínua é necessário reinicializar o *timer* novamente na função *Timer*.

<http://www.inf.pucrs.br/~manssour/OpenGL/Tutorial.html>

Iluminação em OpenGL

- Shading
 - Função *glShadeModel*.
 - Parâmetros: GL_FLAT ou GL_SMOOTH (Gouraud shading, default)

- Material

*glMaterialfv(GLenum face, GLenum pname, const GLfloat *params);*

face determina se as propriedades do material dos polígonos que estão sendo especificadas são *front* (GL_FRONT), *back* (GL_BACK) ou ambas (GL_FRONT_AND_BACK);

pname: pode determinar as seguintes propriedades do material: GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_EMISSION, GL_SHININESS, GL_AMBIENT_AND_DIFFUSE ou GL_COLOR_INDEXES

params: vetor que contém as componentes da propriedade que está sendo especificada

Iluminação em OpenGL

```
// Capacidade de brilho do material
GLfloat especularidade[4]={1.0,1.0,1.0,1.0};
GLint especMaterial = 60;

// Especifica que a cor de fundo da janela será preta
glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

// Habilita o modelo de colorização de Gouraud
glShadeModel(GL_SMOOTH);

// Define a refletância do material
glMaterialfv(GL_FRONT, GL_SPECULAR, especularidade);
// Define a concentração do brilho
glMateriali(GL_FRONT, GL_SHININESS, especMaterial);
```

Variação da função



Iluminação em OpenGL

- Modelo de iluminação

*glLightModelfv(GLenum pname, const GLfloat *params);*

*glLightModeliv(GLenum pname, const GLint *params);*

pname especifica um parâmetro do modelo de iluminação:
GL_LIGHT_MODEL_AMBIENT, GL_LIGHT_MODEL_LOCAL_VIEWER ou
GL_LIGHT_MODEL_TWO_SIDE;

- GL_LIGHT_MODEL_AMBIENT é usado para especificar a luz ambiente *default* para uma cena, que tem um valor RGBA *default* de (0.2, 0.2, 0.2, 1.0);
- GL_LIGHT_MODEL_TWO_SIDE é usado para indicar se ambos os lados de um polígono são iluminados (por *default* apenas o lado frontal é iluminado);
- GL_LIGHT_MODEL_LOCAL_VIEWER modifica o cálculo dos ângulos de reflexão especular;

params (GLfloat* ou GLint*) para GL_LIGHT_MODEL_AMBIENT ou GL_LIGHT_MODEL_LOCAL_VIEWER, aponta para um vetor de números inteiros ou reais; para GL_LIGHT_MODEL_AMBIENT o conteúdo do vetor indica os valores das componentes RGBA da luz ambiente [Wright 2000].

Iluminação em OpenGL

- Fontes de luz: Ambiente, Difusa, Especular, Emissiva

```
glLightf(GLenum light, GLenum pname, GLfloat param);  
glLighti(GLenum light, GLenum pname, GLint param);  
glLightfv(GLenum light, GLenum pname, const GLfloat *params);  
glLightiv(GLenum light, GLenum pname, const GLint *params);
```

As duas primeiras variações requerem apenas um único valor para determinar uma das seguintes propriedades: GL_SPOT_EXPONENT, GL_SPOT_CUTOFF, GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION e GL_QUADRATIC_ATTENUATION. As duas últimas variações são usadas para parâmetros de luz que requerem um vetor com múltiplos valores (GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION e GL_SPOT_DIRECTION).

Descrição dos parâmetros:

light especifica qual fonte de luz está sendo alterada (varia de 0 a GL_MAX_LIGHTS); valores constantes de luz são enumerados de GL_LIGHT0 a GL_LIGHT7

pname especifica qual parâmetro de luz está sendo determinado pela chamada desta função (GL_AMBIENT, GL_DIFFUSE, GL_SPECULAR, GL_POSITION, GL_SPOT_DIRECTION, GL_SPOT_EXPONENT, GL_SPOT_CUTOFF, GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION, GL_QUADRATIC_ATTENUATION);

param (GLfloat ou GLint) para parâmetros que são especificados por um único valor (*param*); estes parâmetros, válidos somente para *spotlights*, são GL_SPOT_EXPONENT, GL_SPOT_CUTOFF, GL_CONSTANT_ATTENUATION, GL_LINEAR_ATTENUATION e GL_QUADRATIC_ATTENUATION.

params (GLfloat* ou GLint*) um vetor de valores que descrevem os parâmetros que estão sendo especificados [Wright 2000].

Iluminação em OpenGL

```
// Define a refletância do material
glMaterialfv(GL_FRONT, GL_SPECULAR, especularidade);
// Define a concentração do brilho
glMateriali(GL_FRONT, GL_SHININESS, especMaterial);

// Ativa o uso da luz ambiente
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, luzAmbiente);

// Define os parâmetros da luz de número 0
glLightfv(GL_LIGHT0, GL_AMBIENT, luzAmbiente);
glLightfv(GL_LIGHT0, GL_DIFFUSE, luzDifusa );
glLightfv(GL_LIGHT0, GL_SPECULAR, luzEspecular );
glLightfv(GL_LIGHT0, GL_POSITION, posicaoLuz );

// Habilita a definição da cor do material a partir da cor corrente
glEnable(GL_COLOR_MATERIAL);
//Habilita o uso de iluminação
glEnable(GL_LIGHTING);
// Habilita a luz de número 0
glEnable(GL_LIGHT0);
// Habilita o depth-buffering
glEnable(GL_DEPTH_TEST);
```


Exemplo de Programa com Iluminação

```
// Iluminacao.c - Isabel H. Manssour
// Um programa OpenGL que exemplifica a visualização
// de objetos 3D com a inserção de uma fonte de luz.
// Este código está baseado nos exemplos disponíveis no livro
// "OpenGL SuperBible", 2nd Edition, de Richard S. e Wright Jr.

#include <gl/glut.h>

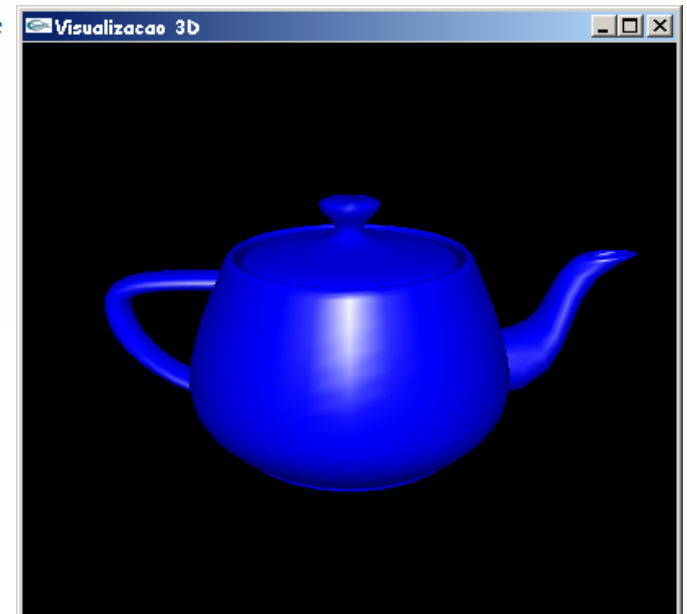
GLfloat angle, fAspect;

// Função callback chamada para fazer o desenho
void Desenha(void)
{
    // Limpa a janela e o depth buffer
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    glColor3f(0.0f, 0.0f, 1.0f);

    // Desenha o teapot com a cor corrente (solid)
    glutSolidTeapot(50.0f);

    glutSwapBuffers();
}
```



Exemplo de Programa com Iluminação

```
// Inicializa parâmetros de rendering
void Inicializa (void)
{
    GLfloat luzAmbiente[4]={0.2,0.2,0.2,1.0};
    GLfloat luzDifusa[4]={0.7,0.7,0.7,1.0};    // "cor"
    GLfloat luzEspecular[4]={1.0, 1.0, 1.0, 1.0}; // "brilho"
    GLfloat posicaoLuz[4]={0.0, 50.0, 50.0, 1.0};

    // Capacidade de brilho do material
    GLfloat especularidade[4]={1.0,1.0,1.0,1.0};
    GLint especMaterial = 60;

    // Especifica que a cor de fundo da janela será preta
    glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

    // Habilita o modelo de colorização de Gouraud
    glShadeModel(GL_SMOOTH);

    // Define a refletância do material
    glMaterialfv(GL_FRONT, GL_SPECULAR, especularidade);
    // Define a concentração do brilho
    glMateriali(GL_FRONT, GL_SHININESS, especMaterial);

    // Ativa o uso da luz ambiente
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT, luzAmbiente);
```

Exemplo de Programa com Iluminação

```
// Define os parâmetros da luz de número 0
glLightfv(GL_LIGHT0, GL_AMBIENT, luzAmbiente);
glLightfv(GL_LIGHT0, GL_DIFFUSE, luzDifusa );
glLightfv(GL_LIGHT0, GL_SPECULAR, luzEspecular );
glLightfv(GL_LIGHT0, GL_POSITION, posicaoLuz );

// Habilita a definição da cor do material a partir da cor corrente
glEnable(GL_COLOR_MATERIAL);
//Habilita o uso de iluminação
glEnable(GL_LIGHTING);
// Habilita a luz de número 0
glEnable(GL_LIGHT0);
// Habilita o depth-buffering
glEnable(GL_DEPTH_TEST);

angle=45;
}

// Função usada para especificar o volume de visualização
void EspecificaParametrosVisualizacao(void)
{
    // Especifica sistema de coordenadas de projeção
    glMatrixMode(GL_PROJECTION);
    // Inicializa sistema de coordenadas de projeção
    glLoadIdentity();

    // Especifica a projeção perspectiva
    gluPerspective(angle,fAspect,0.4,500);

    // Especifica sistema de coordenadas do modelo
    glMatrixMode(GL_MODELVIEW);
    // Inicializa sistema de coordenadas do modelo
    glLoadIdentity();

    // Especifica posição do observador e do alvo
    gluLookAt(0,80,200, 0,0,0, 0,1,0);
}
```

Exemplo de Programa com Iluminação

```
// Função callback chamada quando o tamanho da janela é alterado
void AlteraTamanhoJanela(GLsizei w, GLsizei h)
{
    // Para prevenir uma divisão por zero
    if ( h == 0 ) h = 1;

    // Especifica o tamanho da viewport
    glViewport(0, 0, w, h);

    // Calcula a correção de aspecto
    fAspect = (GLfloat)w/(GLfloat)h;

    EspecificaParametrosVisualizacao();
}

// Função callback chamada para gerenciar eventos do mouse
void GerenciaMouse(int button, int state, int x, int y)
{
    if (button == GLUT_LEFT_BUTTON)
        if (state == GLUT_DOWN) { // Zoom-in
            if (angle >= 10) angle -= 5;
        }
    if (button == GLUT_RIGHT_BUTTON)
        if (state == GLUT_DOWN) { // Zoom-out
            if (angle <= 130) angle += 5;
        }
    EspecificaParametrosVisualizacao();
    glutPostRedisplay();
}
```

Exemplo de Programa com Iluminação

```
// Programa Principal
int main(void)
{
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB | GLUT_DEPTH);
    glutInitWindowSize(400,350);
    glutCreateWindow("Visualizacao 3D");
    glutDisplayFunc(Desenha);
    glutReshapeFunc(AlterarTamanhoJanela);
    glutMouseFunc(GerenciaMouse);
    Inicializa();
    glutMainLoop();
}
```

Material Adicional

- <http://www.novateceditora.com.br/livros/opengl> →
- <http://www.inf.pucrs.br/~manssour/OpenGL/Tutorial.html>
- http://www.cs.trinity.edu/About/The_Courses/cs357/gl.html
- <http://www.opengl.org/resources/code/samples/s2001/>
- <http://fly.cc.fer.hr/~unreal/theredbook/> →

