


# Introdução a OpenGL



Renato Rodrigues Oliveira da Silva

Março 2009

Adaptado do material de

Marcela X. Ribeiro

Maria Cristina F. de Oliveira

Rosane Minghim



## Sumário

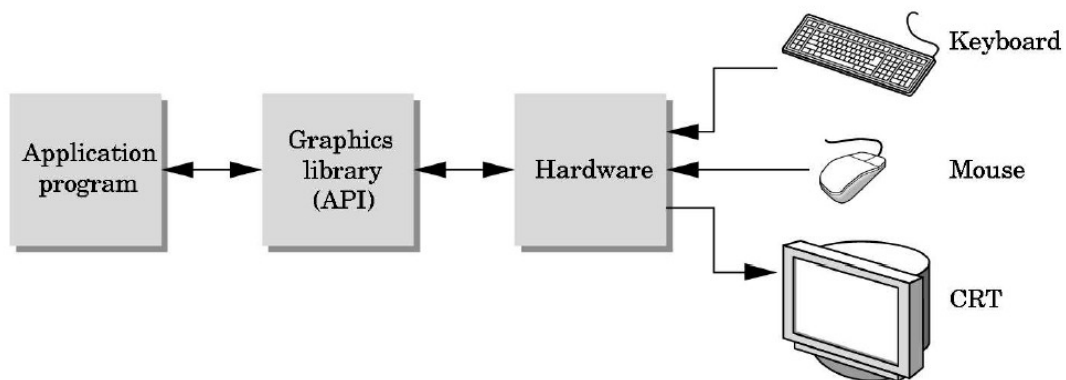


- # Introdução
- # Núcleo OpenGL, GLU, GLUT
- # Ambiente de trabalho
- # Instalação
- # Exemplo

# Introdução – O que é OpenGL?

## # OpenGL – *Open Graphics Library*

- É uma API (*Application Program Interface*) para o aplicações gráficas
- Abstrai a complexidade do hardware



3

# Introdução – O que é OpenGL?

- # Implementa rotinas gráficas e de modelagem bidimensional e tridimensional
- # Portável
- # Rápida

4

# Introdução - Histórico

- # Especificação gerenciada por um consórcio independente formado em 1992
  - Constituído por empresas líderes na área: NVIDIA, 3Dlabs, Apple Computer...
  - Responsável pela aprovação de novas funcionalidades, versões e extensões da OpenGL
  - <http://www.opengl.org>

5

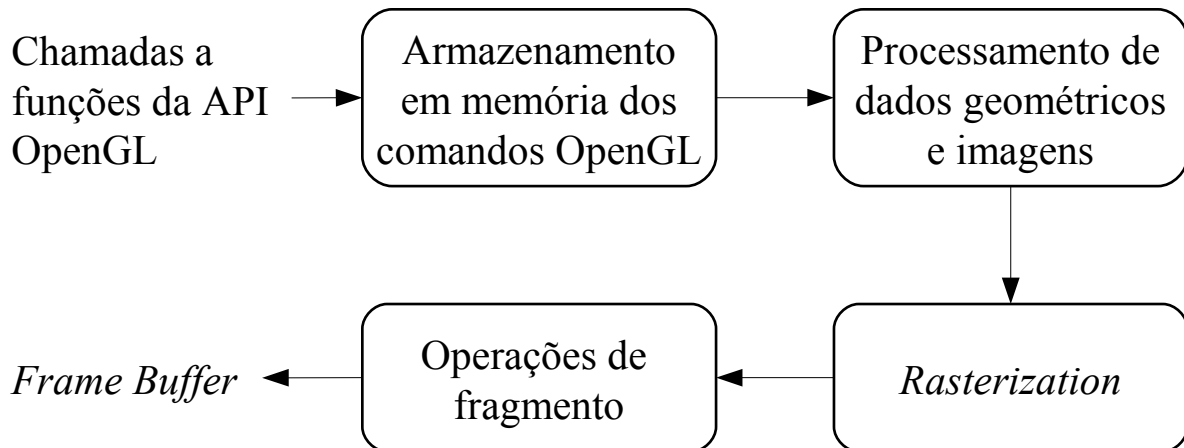
## Núcleo OpenGL

- # Possui aproximadamente 200 funções gráficas
- # Primitivas de baixo nível
- # Portável
  - Mesmo programa compilado em diferentes ambientes operacionais e sistemas gráficos
  - Aplicação não precisa ser alterada

6

# Núcleo OpenGL

## # Pipeline



7

# Núcleo OpenGL

- # Por ser portátil, a OpenGL não possui funções para gerenciamento de janelas, tratamento de eventos ou manipulação de arquivos
  - Podem ser utilizadas as funções específicas de cada plataforma
  - Ou usar uma biblioteca independente (GLUT, FLTK, wxWidgets, por exemplo)

8

# Exemplo – Primitivas Geométricas

## # Triângulos

```
glBegin(GL_TRIANGLES) ;  
    glVertex2f(x1, y1, z1) ;  
    glVertex3f(x2, y2, z2) ;  
    glVertex3f(x3, y3, z3) ;  
glEnd() ;
```

9

# Exemplo – Primitivas Geométricas

## # Quadriláteros

```
glBegin(GL_QUADS) ;  
    glVertex3f(x1, y1, z1) ;  
    glVertex3f(x2, y2, z2) ;  
    glVertex3f(x3, y3, z3) ;  
    glVertex3f(x4, y4, z4) ;  
glEnd() ;
```

10

# Exemplo

## Transformações Geométricas

### # Translação

■ `glTranslate(GLfloat tx, ty, tz)`

- Parâmetros: valores de translação aplicados aos eixos x,y,z

### # Escala

■ `glScaled(GLdouble tx, ty, tz)`

- Parâmetros: valores de escala aplicados aos eixos x,y,z

### # Rotação

■ `glRotatef(GLfloat ângulo, x, y, z)`

- Ângulo e eixo ao redor do qual será aplicada a rotação.

11

## Estados da OpenGL

### # OpenGL rastreia diversas variáveis de estado

■ Tamanho atual de um ponto, cor de fundo da janela, cor do desenho

■ O valor corrente permanece ativo até que seja alterado

- Tamanho de ponto: `glPointSize(3.0)`
- Cor de traçado: `glColor3f(red, green, blue)`
- Cor de fundo: `glClearColor(r, g, b, alpha)`
- ...

12

# Exemplo OpenGL

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

void Desenha(void) {

    //Limpa a janela de visualização
    glClear(GL_COLOR_BUFFER_BIT);
    //Altera a cor do traçado para preto
    glColor3f(0.0f, 0.0f, 0.0f);
```

13

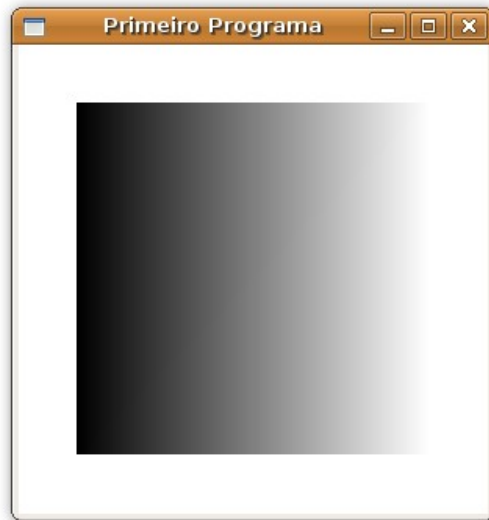
# Exemplo OpenGL

```
//Desenha um quadrado
glBegin(GL_QUADS)
    glVertex2f(-45.0f, -15.0f);
    glVertex2f(-45.0f, 15.0f);
    //Altera a cor do traçado para branco
    glColor3f(1.0f, 1.0f, 1.0f);
    glVertex2f(-15.0f, 15.0f);
    glVertex2f(-15.0f, -15.0f);
glEnd();

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

14

# Exemplo OpenGL



15

## GLU – OpenGL Utility Library

- # Instalada junto com a OpenGL
- # Contém uma série de funções que encapsulam comandos OpenGL de mais baixo nível
  - Definição de matrizes para projeção
  - Desenho de superfícies quádricas
  - Curvas e superfícies NURBS

16



# GLUT – OpenGL Utility Toolkit

- # Biblioteca que inclui alguns elementos de interface gráfica com o usuário
  - Criação de janelas e menus pop-up
  - Gerenciamento de eventos de mouse e teclado
- # Independente de plataforma

17

## GLUT - Inicialização

- # void glutInitDisplayMode(parâmetros)
  - GLUT\_DOUBLE: Define que a GLUT usará dois buffers de cor
  - GLUT\_SINGLE: Define apenas um buffer de cor
  - GLUT\_DEPTH: Define o uso de um buffer de profundidade (z-buffer), para remoção de superfícies escondidas
  - GLUT\_RGB: Define que as cores são especificadas por componentes RGB

18

# GLUT - Inicialização

- # **void glutInitWindowPosition(int x, int y)**
  - Define a posição inicial da janela. Os parâmetros representam o canto superior esquerdo
- # **void glutInitWindowSize(int width, int height)**
  - Define a largura e altura da janela
- # **int glutCreateWindow(char \*string)**
  - Cria a janela e define o seu título

19

# GLUT - Inicialização

## # Exemplo

```
int main(int argc, char *argv) {  
    glutInit(&argc, argv);  
    glutInitDisplayMode(GLUT_SINGLE |  
                        GLUT_RGB);  
    glutInitWindowSize(400, 400);  
    glutCreateWindow("Primeiro  
                    Programa");  
    glutMainLoop();  
}
```

20

# GLUT – Tratamento de eventos

- # Gerenciamento de eventos por meio de funções “callback”
- # Quem chama a função para o tratamento de um evento não é o programador e sim a GLUT
- # O programador define apenas a função a ser chamada, respeitando os parâmetros definidos pela GLUT

21

# GLUT – Tratamento de eventos

- # void glutDisplayFunc( Desenha)
  - Define que a função Desenha será a responsável por redesenhar a janela, sempre que necessário
  - A função deve ter o seguinte protótipo
    - void <nomeFunção> (void)

22

# GLUT – Tratamento de eventos

## # void glutReshapeFunc (AlteraTamanhoJanela)

- Define a função que será responsável por tratar o evento referente ao redimensionamento da janela
- Deve ter o protótipo
  - void <nomeFunção> (int largura, int altura)

23

# GLUT – Tratamento de eventos

```
int main(int argc, char *argv[]) {  
    glutInit(&argc, argv);  
    glutInitDisplayMode(GLUT_SINGLE |  
                        GLUT_RGB);  
    glutInitWindowSize(400, 400);  
    glutCreateWindow("Primeiro Programa");  
    glutDisplayFunc (Desenha) ;  
    glutReshapeFunc (AlteraTamanhoJanela) ;  
    ...  
}
```

24

# GLUT – Tratamento de eventos

```
void AlteraTamanhoJanela(GLsizei w,Glsizei h){
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    if (w <= h)
        gluOrtho2D(-40.0f, 40.0f, -40.0f*h/w,
                    40.0f*h/w);
    else
        gluOrtho2D(-40.0f*w/h, 40.0f*w/h,
                    -40.0f, 40.0f);
}
```

25

# GLUT – Interação com o teclado

# void glutKeyboardFunc(Teclado)

■ Trata o pressionamento de teclas comuns

■ void <nomeFunção> (unsigned char tecla, int x,  
int y)

■ Tecla representa o código ASCII da tecla pressionada

■ Os parâmetros x e y representam as coordenadas do mouse quando o evento ocorreu

26

# GLUT – Interação com o teclado

## # Exemplo

```
void Teclado(unsigned char tecla, int x,
              int y)
{
    if (tecla == 27) // tecla ESC
        exit(0);
    if (tecla == 'a') // muda para tela cheia
        glutFullScreen();
    if (tecla == 'A') // restaura tela
        glutReshapeWindow(400, 400);
}
```

27

# GLUT – Interação com o mouse

## # void glutMouseFunc(GerenciaMouse)

- Trata eventos de pressionamento e liberação de botões do mouse

- void <nomeFunção> (int botao, int estado, int x, int y)
- botão: GLUT\_LEFT\_BUTTON, GLUT\_MIDDLE\_BUTTON, GLUT\_RIGHT\_BUTTON
- estado: GLUT\_UP, GLUT\_DOWN

28

# GLUT – Interação com o mouse

## # Ex. posicionar pontos com o mouse

```
void myMouse(int button, int state, int x, int y){
    if (button == GLUT_LEFT_BUTTON && state ==
        GLUT_DOWN) {
        glColor3f(1.0f, 0.0f, 0.0f);
        glBegin(GL_POINTS)
            glVertex2f(x, y);
        glEnd();
        glFlush();
    }
}
```

29

# Padronização dos nomes das funções

## # Convenção:

<Prefixo Biblioteca> <Comando Raiz>  
<Contador de Argumentos> <Tipo  
Argumentos>

## # Visa padronizar e facilitar a utilização

## # Possibilita identificar a qual biblioteca a função pertence, quantos argumentos possui e quais são os tipos dos argumentos.

30

# Padronização dos nomes das funções

## # Exemplo

```
void glColor3f(GLfloat red, GLfloat green,  
               GLfloat blue)
```

//gl - É o prefixo da biblioteca GL

//Color - comando objetivo da função

//3 - contador para o número de argumentos

//f - indica o tipo dos argumentos

31

## Tipos de Dados

# Tipos de dados próprios para OpenGL

# Tornam o programa-fonte portátil

sufixo	Tipo	tipo C	nome
b	inteiro 8 bits	signed char	GLbyte
s	inteiro 16 bits	short	GLshort
i	inteiro 32 bits	int/long	GLint
f	float 32 bits	float	GLfloat
...	...	...	...

32



# Tipos de Dados

//Perigo!!: sistema passa int...

```
void drawDot(int x, int y) {  
    glBegin(GL_POINTS);  
    glVertex2i(x, y); // função “espera” inteiro 32 bits  
    glEnd();  
}
```

// código seguro.

```
void drawDot(GLint x, GLint y) {  
    glBegin(GL_POINTS);  
    glVertex2i(x, y);  
    glEnd();  
}
```

33

# Ambiente de Trabalho

- # Uma imagem consiste em uma matriz de pontos, já um modelo é uma representação computacional de um objeto
- # O modelo corresponde a uma estrutura de dados com a descrição geométrica da cena

34

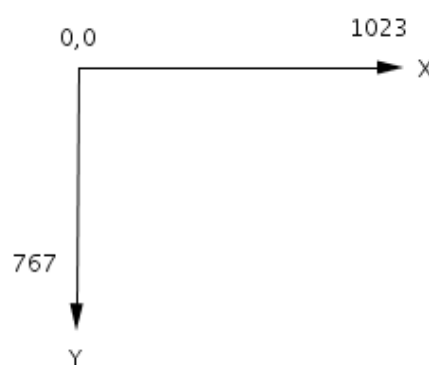
# Ambiente de Trabalho

- # Em OpenGL os objetos são representados no Sistema de Referência do Universo (SRU)
  - Plano cartesiano bi ou tridimensional, com eixos se interceptando na origem
- # Todos os comandos e modelos são definidos em relação a este sistema de referência

35

# Ambiente de Trabalho

- # No monitor do computador é adotado o SRT (Sistema de Referência da Tela)
  - No SRT a origem fica no canto superior esquerdo do monitor



36

# Ambiente de Trabalho - 2D

# No caso 2D, é necessário definir a porção o universo que desejamos mapear na tela.

- Essa área é chamada de janela de seleção, ou *window*

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

37

# Ambiente de Trabalho - 2D

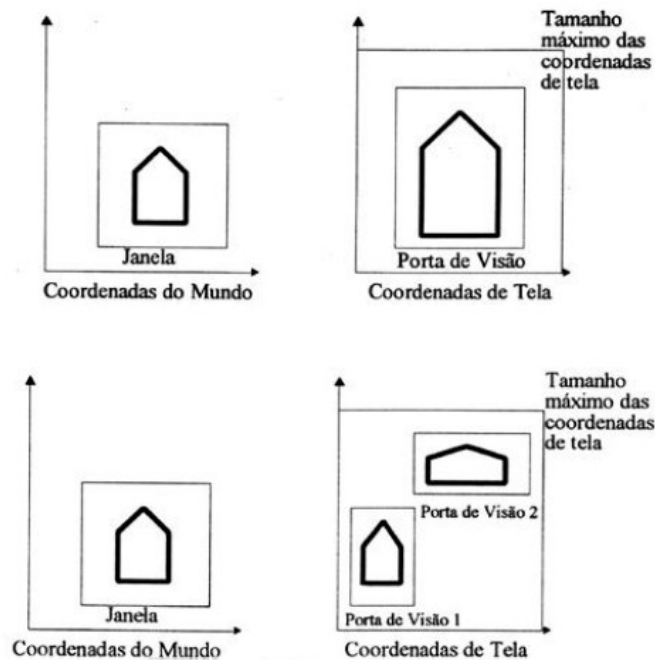
# É necessário definir também em que parte do monitor deseja-se exibir o conteúdo da *window*

- Chamamos essa região de *viewport* (ou janela de exibição)

```
void glViewport(GLint x, GLint y,  
               GLsizei width, GLsizei height)
```

38

# Ambiente de Trabalho - 2D



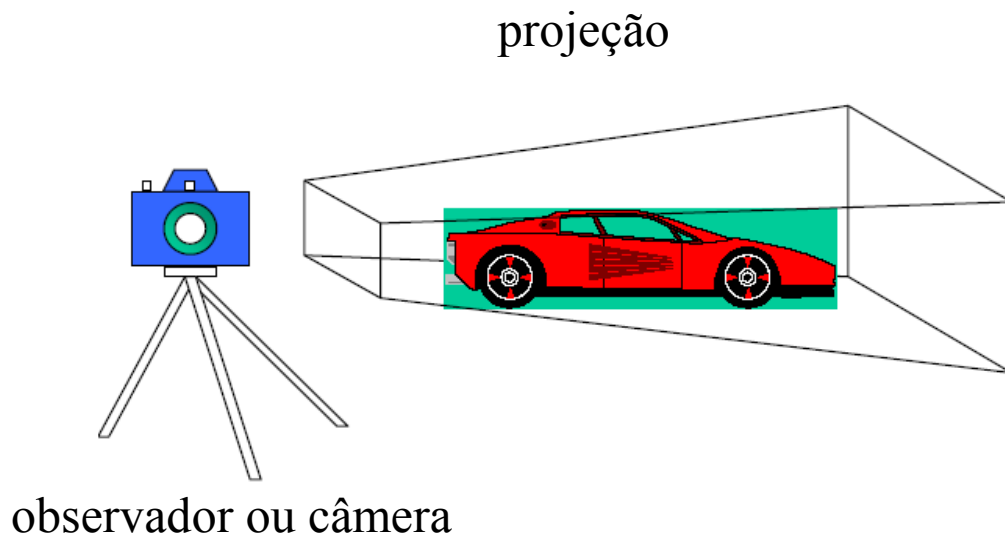
39

# Ambiente de Trabalho - 3D

- # No caso 3D, é preciso definir como a cena será visualizada e projetada em uma imagem 2D
- # Defini-se um observador virtual (ou câmera), que inclui a sua posição e orientação no universo
- # Como cada imagem gerada a partir da posição e orientação do observador é estática, faz-se analogia com uma foto

40

# Ambiente de Trabalho - 3D



41

# Ambiente de Trabalho - 3D

# A GLU oferece a seguinte função para posicionar e orientar a câmera

```
void gluLookAt(GLdouble obsx, obsy, obsz,  
               alvox, alvoy, alvoz,  
               upx, upy, upz)
```

//obs\*: define a posição da câmera

//alvo\*: ponto para onde o observador está olhando

//up\*: vetor que indica a 'vertical' da câmera

42

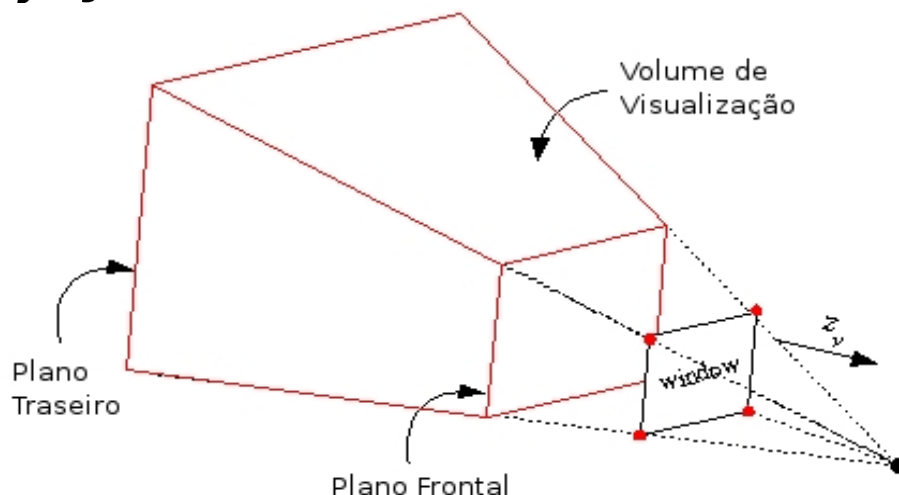
# Projeções

- # Obtém representações bidimensionais de objetos tridimensionais
- # A projeção é definida por raios de projeção (projetantes) que passam através de cada vértice dos objetos e interceptam o plano de projeção
- # Projeções divididas em 2 tipos:
  - Paralela Ortográfica
  - Perspectiva

43

## Projeção Perspectiva

- # As projetantes emanam de um único ponto, a uma distância finita do plano de projeção



44

# Projeção Perspectiva

```
void gluPerspective(GLdouble fovy, aspect,
                   zNear, zFar)

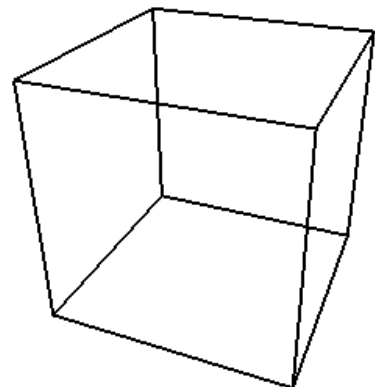
//fovy: ângulo de abertura da câmera em y
//aspect: área de visualização em x
//zNear: distância do observador ao plano
        de corte frontal
//zFar: distância do observador ao plano
        de corte traseiro
```

45

# Projeção Perspectiva

## # Exemplo

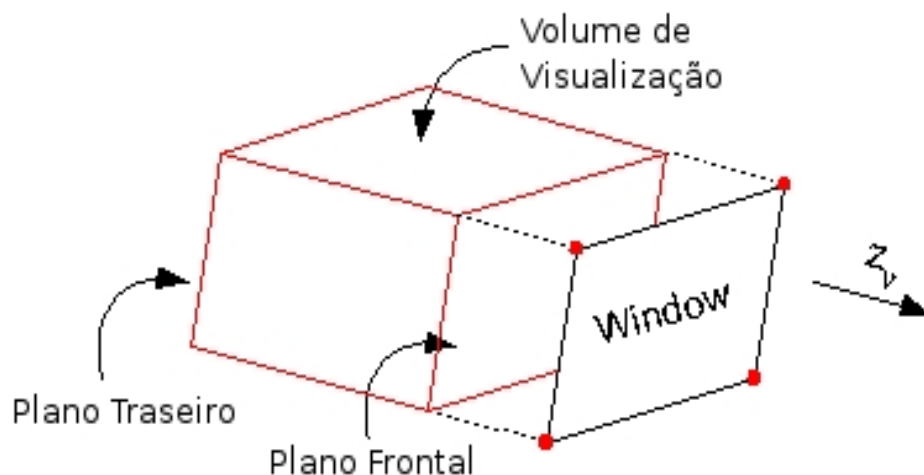
```
void Desenha(){
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(60, fAspect, 0.5, 500);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(40, 60, 100, 0, 0, 0, 0, 1, 0);
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0f, 0.0f, 1.0f);
    glutWireCube(50);
    glFlush();
}
```



46

# Projeção Paralela

- # As projetantes são paralelas entre si
- # Não há alteração nas medidas do objeto



47

# Projeção Paralela

```
void gluOrtho(GLdouble left, right,  
              bottom, top, near, far)
```

```
/*Os parâmetros definem os limites mínimo e  
máximo da janela de projeção em x, y e z.*/
```

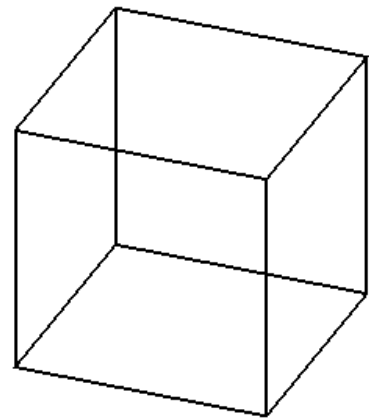
48



# Projeção Paralela

## # Exemplo

```
void Desenha(){
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(-65.0, 65.0, -65.0, 65.0, -400.0, 400.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    gluLookAt(40,60,100, 0,0,0, 0,1,0);
    glClearColor(1.0f, 1.0f, 1.0f, 1.0f);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(0.0f, 0.0f, 0.0f);
    glutWireCube(50);
    glFlush();
}
```



49

## Instalação - Windows

Para executar é necessário possuir as DLLs **Opengl32.dll**, **Glu32.dll** e **Glut32.dll** no diretório System32 do Windows

# Para compilar é necessário incluir as bibliotecas **Opengl32.lib**, **Glu32.lib** e **Glut32.lib** no projeto (Visual C ou DevC++), além de adicionar o *header* **<GL/glut.h>**

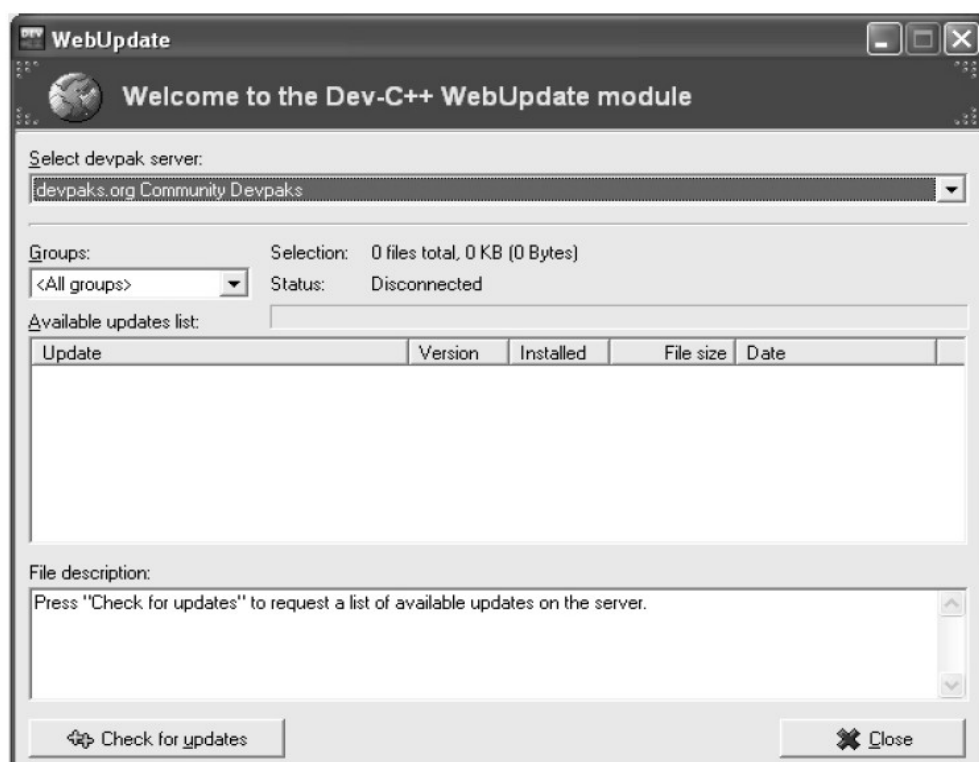
50

# Instalação – GLUT no Dev-C++

- # Selecione o menu Ferramentas → Atualizações
- # Em “select devpak server” escolha “devpaks.org Community Devpaks”
- # Clique no botão “Check for updates”

51

# Instalação – GLUT no Dev-C++



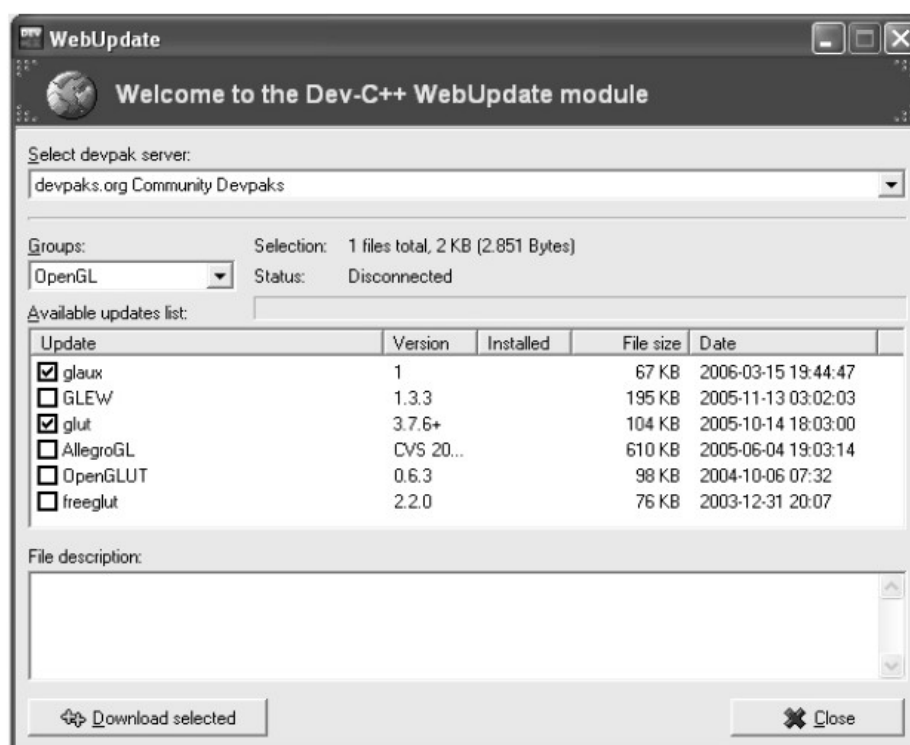
52

# Instalação – GLUT no Dev-C++

- # Na ComboBox “Groups”, selecione “OpenGL”
- # Marque a opção “glut”, “glaux”
- # Clique no botão “Download selected”
- # Aguarde e efetue a instalação

53

# Instalação – GLUT no Dev-C++



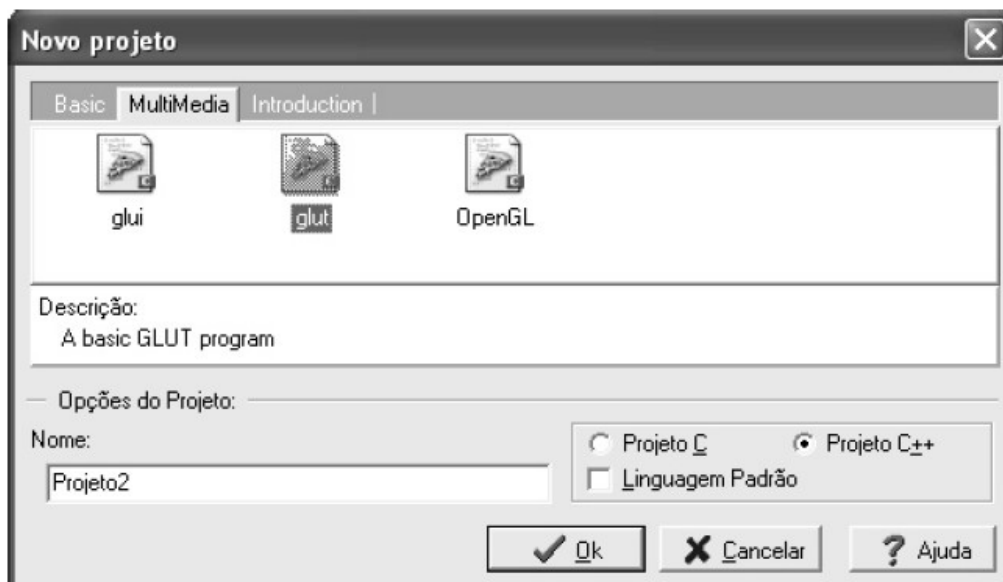
54

# Instalação – GLUT no Dev-C++

- # Para testar, selecione o menu: Arquivo → Novo → Projeto
- # Clique na aba multimídia
- # Selecione “GLUT” e clique em “Ok”

55

# Instalação – GLUT no Dev-C++



56

# Instalação – GLUT no Dev-C++

- # Salve o projeto e o arquivo com o código
- # Compile e execute
- # Se a cena gráfica aparecer, significa que tudo está instalado corretamente

57

# Instalação - Linux

- # Para execução deve possuir as bibliotecas **libGL.so**, **libGLU.so** e **libglut.so** no diretório `/usr/lib`
- # Para compilar verificar a existência dos *headers* **gl.h**, **glu.h** e **glut.h** em `/usr/include/GL`
- # Em geral as bibliotecas são fornecidas por um pacote denominado MesaGL

58

# Instalação - Linux

- # Ao compilar, importar as bibliotecas OpenGL
- # `gcc <nomeArquivo.c> -lglut -lGL -lGLU -lm -pthread -o <nomeExecutavel>`

59

# Exemplo – Sistema Solar

- # Código-fonte na CoTeia

60

# Bibliografia



---

- # Cohen, M. , Manssour, I. H, OpenGL – Uma Abordagem Prática e Objetiva