

Introdução ao OpenGL



Maria Cristina F. de Oliveira

Rosane Minghim

Fernando V. Paulovich

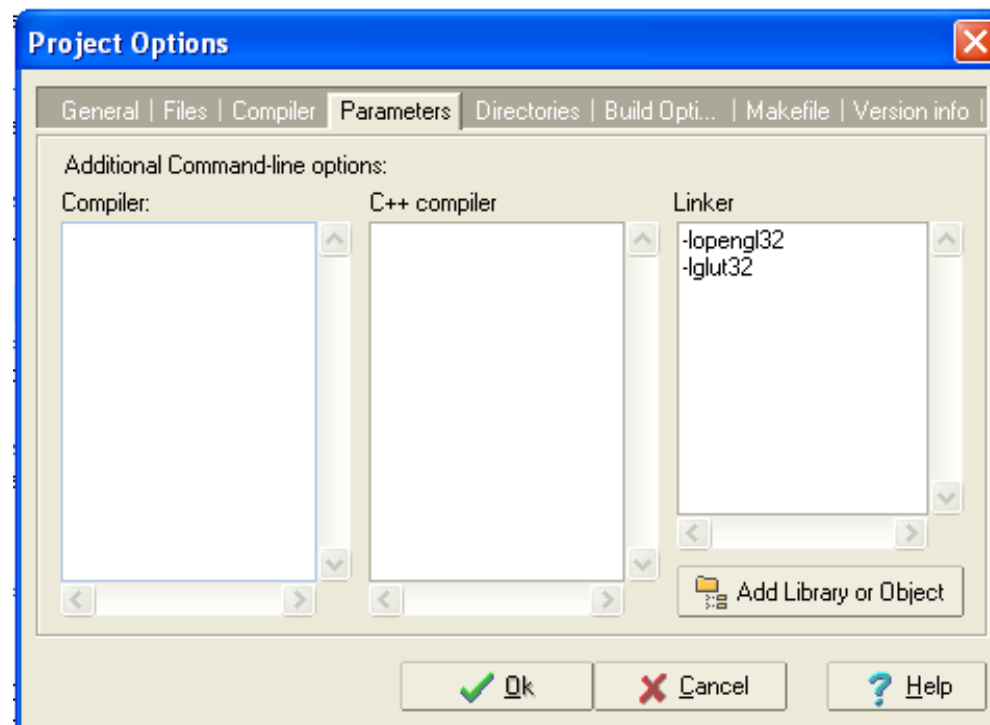


Instalando (DevC++)

- Faça o *download* (<http://www.inf.pucrs.br/~manssour/OpenGL/glut-devc.zip>) e descompacte o mesmo
- Mova o arquivo *glut.h* para a pasta GL do DevC++ (C:\Dev-C++\Include\GL)
- Mova os arquivos *glut32.def* e *libglut.a* para a pasta Lib do DevC++ (C:\Dev-C++\Lib)
- Mova o arquivo *glut32.dll* para a pasta onde se encontram os arquivos *opengl32.dll* e *glu32.dll* (c:/windows/system32)
- <http://www.inf.pucrs.br/~manssour/OpenGL/Devc++.html>

Instalando (DevC++)

- Para cada projeto gerado deve ser configurado



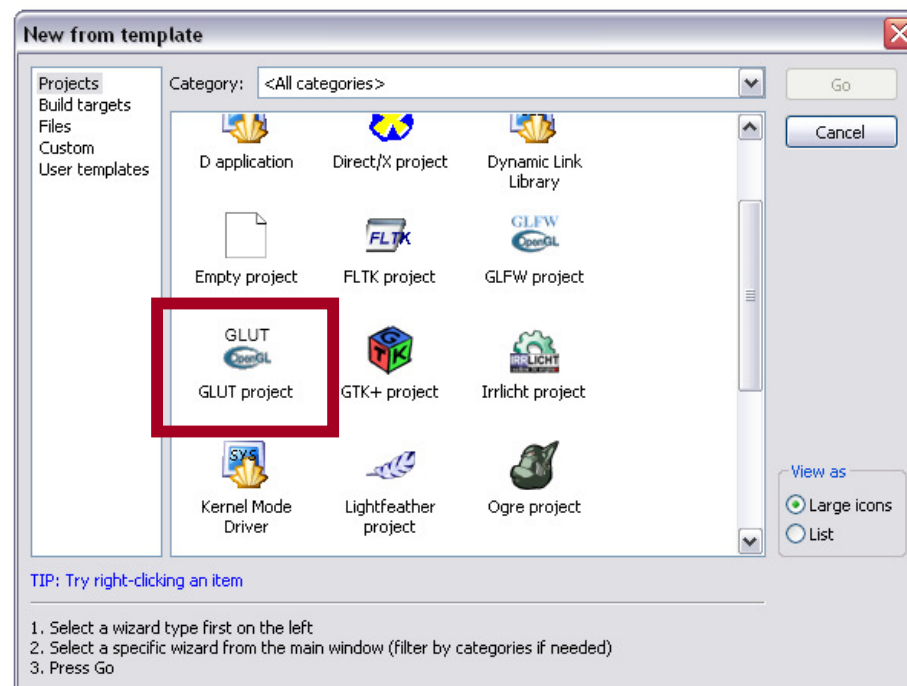
Instalando (DevC++)

- A biblioteca do GLUT deve estar na pasta do programa executável gerado. Isso permite que ele execute em ambientes onde o GLUT está instalado.



Instalando (CodeBlocks)

- Semelhante, mas criar um projeto GLUT



<http://www.codeblocks.org/>



OpenGL

- *Application Programming Interface (API)*
 - Coleção de rotinas que o programador pode chamar
 - Modelo de como estas rotinas operam em conjunto para gerar gráficos
 - Programador 'enxerga' apenas a interface
 - Não precisa lidar com aspectos específicos do hardware ou idiossincracias de software no sistema gráfico residente (independente do dispositivo)
 - Oferece suporte para gerar e exibir cenas 3D complexas, e também para gráficos 2D simples



OpenGL

- Ambiente p/ escrever e executar programas gráficos
 - Monitor (‘tela’) + biblioteca de software
 - para desenhar primitivas gráficas na tela
- API pode ser vista como uma ‘caixa preta’
 - Entradas:
 - Chamadas a funções da biblioteca feitas pelo programa do usuário
 - Medidas fornecidas por dispositivos de entrada
 - ...
 - Saídas:
 - Os gráficos exibidos no monitor
 - Descrita em termos das funções que disponibiliza



API Open GL

■ Programa

- Em geral, trabalha com um sistema de janelas (‘window system’)
- Inicializações: modo de exibição (‘display mode’), janela de desenho e sistema de coordenadas de referência (associado à janela)

■ API oferece centenas de funções...

- diferentes funcionalidades
 1. Funções primitivas: o que
 2. Funções de atributos: como
 3. ...



Programação Dirigida a Eventos

- Direcionada a eventos (*event-driven*)
 - programa responde a eventos: clique do mouse, tecla pressionada, redimensionamento da janela
- Fila de eventos
 - Política FIFO de tratamento dos eventos
 - Programa organizado como coleção de *callback functions*
 - Cada tipo de evento associado a uma *callback* que é ativada quando ele ocorre
 - Modelo de programação diferente do 'procedimental seqüencial'...
 - *Forever*: 'não faça nada até que um evento ocorra, quando isso acontece, trate o evento (ative sua *callback*)...'



Estados do OpenGL

- OpenGL rastreia diversas variáveis de estado
 - Tamanho atual de um ponto, cor de fundo da janela, cor do desenho, etc.
 - O valor corrente permanece ativo até que seja alterado
 - Tamanho de ponto: `glPointSize(3.0)`
 - Cor de desenho: `glColor3f(red, green, blue)`
 - Cor de fundo: `glClearColor(red, green, blue, alpha)`
 - Limpar janela: `glClear(GL_COLOR_BUFFER_BIT)`



Programando

- OpenGL é utilizada junto com outras bibliotecas auxiliares
 - **OpenGL Utility (GLU):** definir a visão, matrizes de projeção, aproximação poligonal, desenho de superfícies, etc
 - **OpenGL Utility Toolkit (GLUT):** define o sistema de janelas, e outras funções de desenho de superfície
- Importar **#include <GL/glut.h>** que todas as bibliotecas serão corretamente importadas



Programando

- Inicializar o sistema
 - `glutInit(&argc, argv)`
- Criar a janela de exibição
 - `glutCreateWindow(“Título”)`
- Registrar qual será a função de desenho
 - `glutDisplayFunc(...)`
- Ativar as janelas e o sistema de desenho
 - `glutMainLoop()` – último a ser chamado



Programando

- Definir a posição da janela
 - `glutInitWindowPosition(50, 100)`
- Definir o tamanho da janela
 - `glutInitWindowSize(400, 300)`
- Definir o modo de desenho
 - `glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB)`
 - Podem ser usadas várias constantes



Programando

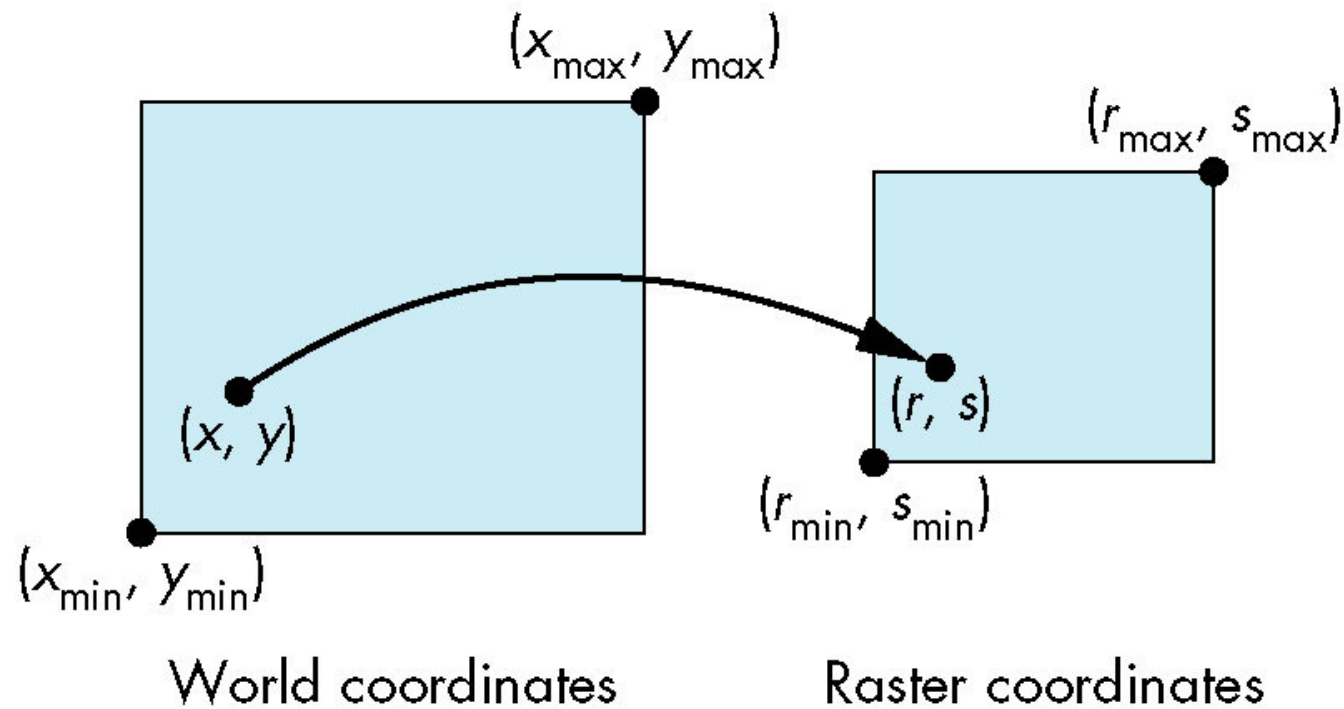
- Definir a cor de fundo da janela
 - `glClearColor(1.0, 1.0, 1.0, 0.0)`
- Invocar a função para o desenho do fundo
 - `glClear(GL_COLOR_BUFFER_BIT)`
 - `GL_COLOR_BUFFER_BIT`: os bits do “color buffer” serão modificados para a cor de fundo



Programando

- O pipeline da OpenGL é sempre 3D, mas é possível criar desenhos 2D, definindo
 - `glMatrixMode(GL_PROJECTION)`
 - `gluOrtho2D(0.0, 200.0, 0.0, 150.0)`
- Projeção ortogonal, com x variando de 0 a 200 e y de 0 a 150

Sistemas de Coordenadas





Sistemas de Coordenadas

- OpenGL permite desenhar gráficos de modo independente do dispositivo
- Usuário especifica elementos de interesse no Sistema de Coordenadas do Usuário, ou Sistema de Coordenadas do Mundo (ponto flutuante)
- Os elementos são traçados no sistema de coordenadas do dispositivo, ou sistema de coordenadas da tela (inteiro)
- OpenGL faz o mapeamento de forma transparente para o usuário



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

```
#include <stdlib.h>
```

```
void init(void) {
```

```
    glClearColor(1.0, 1.0, 1.0, 0.0);
```

```
    glMatrixMode(GL_PROJECTION);
```

```
    gluOrtho2D(0.0, 200.0, 0.0, 150.0);
```

```
}
```

```
void desenha(void) {
```

```
    glClear(GL_COLOR_BUFFER_BIT); //desenha o fundo (limpa a janela)
```

```
    glColor3f(1.0, 0.0, 0.0); //altera o atributo de cor
```

```
    glBegin(GL_LINES); //desenha uma linha
```

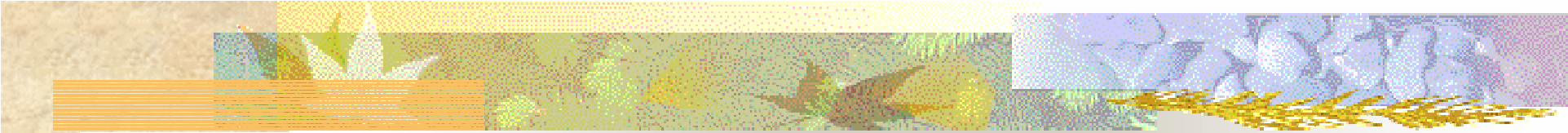
```
        glVertex2i(180, 15);
```

```
        glVertex2i(10, 145);
```

```
    glEnd();
```

```
    glFlush(); //processa as rotinas OpenGL o mais rápido possível
```

```
}
```



```
int main(int argc, char**argv) {  
    glutInit(&argc, argv);  
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);  
    glutInitWindowPosition(50, 100);  
    glutInitWindowSize(400, 300);  
    glutCreateWindow("Titulo");  
  
    init();                // inicialização (após a criação da janela)  
    glutDisplayFunc(desenha); // registra a função de desenho  
    glutMainLoop();         // desenha tudo e espera por eventos  
  
    return EXIT_SUCCESS;  
}
```



Primitivas

- Traçado requer um sistema de referência para posicionamento espacial
 - Em CG trabalha-se com diversos sistemas de coordenadas... Inicialmente, adotamos um muito simples
 - Associado ao sistema de coordenadas da janela
 - Distâncias medidas em pixels
 - Zero no canto inferior esquerdo da janela
- Primitivas básicas
 - Pontos, linhas, poli-linhas, polígonos
 - Definidos em termos de **vértices**



Primitivas

■ Desenho de primitivas

- Diversos objetos: GL_POINTS, GL_LINES, GL_POLYGON, etc.
- Para descrever objeto, usuário informa a lista de vértices

```
glBegin(GL_POINTS);  
    glVertex2i(100, 50); // desenha 3 pontos  
    glVertex2i(100, 130);  
    glVertex2i(150, 130);  
glEnd();
```



Tipos de Dados

- OpenGL suporta um conjunto fixo de tipos de dados.

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
...



Tipos de Dados

```
void drawDot(int x, int y)           //Perigo!!: sistema passa int...
{
    glBegin(GL_POINTS);
        glVertex2i(x, y);           // função c/ sufixo i 'espera' inteiro 32 bits
    glEnd();
}
```

```
void drawDot(GLint x, GLint y)      // código seguro... compilação associa os
{                                   // tipos adequadamente (GL.h)
    glBegin(GL_POINTS);
        glVertex2i(x, y);
    glEnd();
}
```

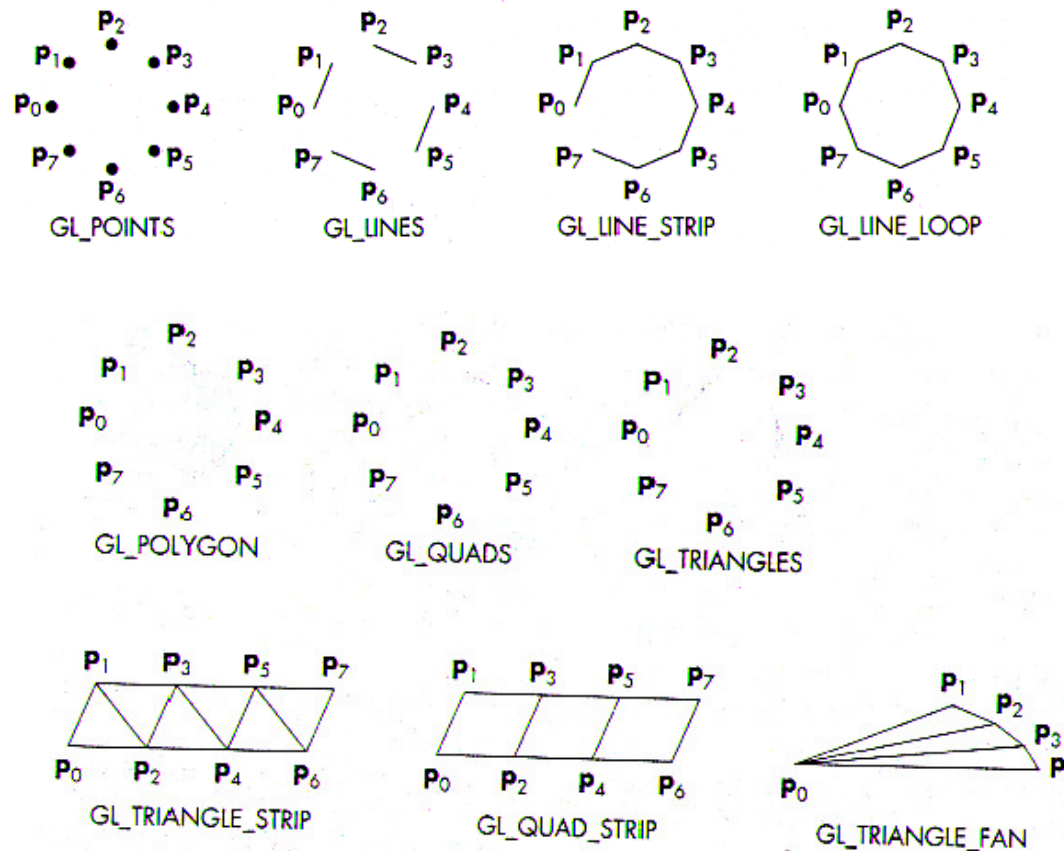


Outras Primitivas: Poli-linhas e Polígonos

- Poli-linha: GL_LINE_STRIP, GL_LINE_LOOP
 - sequência de linhas conectadas... fechada ou não

- Outras primitivas
 - GL_TRIANGLES
 - GL_QUADS
 - GL_TRIANGLE_STRIP
 - GL_TRIANGLE_FAN
 - GL_QUAD_STRIP

Outras Primitivas: Poli-linhas e Polígonos



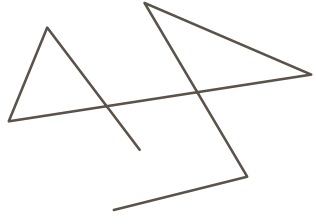
Outras Primitivas: Poli-linhas e Polígonos



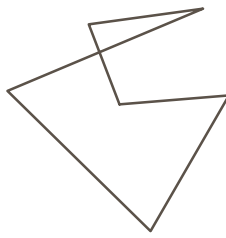
GL_POINTS



GL_LINES



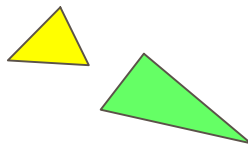
GL_LINE_STRIP



GL_LINE_LOOP



GL_POLYGON



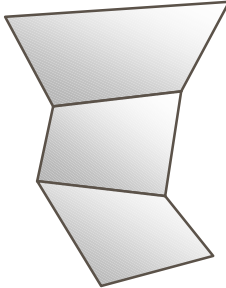
GL_TRIANGLES



GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN



GL_QUAD_STRIP



Interação com Mouse e Teclado

- `glutMouseFunc(myMouse)`, registra a fç de tratamento do evento de pressionar/soltar botão do mouse
- `glutMotionFunc(myMovedMouse)`, registra a fç de tratamento do evento de mover o mouse com um botão pressionado
- `glutKeyboardFunc(myKeyboard)`, registra a fç de tratamento do evento tecla pressionada



Interação com Mouse e Teclado

- `void myMouse(int button, int state, int x, int y)`
 - Button: `GLUT_LEFT_BUTTON`,
`GLUT_MIDDLE_BUTTON`, `GLUT_RIGHT_BUTTON`
 - State: `GLUT_UP`, `GLUT_DOWN`
 - `x, y`: posição do mouse no momento da ocorrência do evento
 - Posição do pixel em relação ao sistema de coordenadas com origem no canto superior esquerdo da janela



Interação com Mouse

```
void myMouse(int button, int state, int x, int y)
{
    if(button == GLUT_LEFT_BUTTON && state ==
        GLUT_DOWN)
        ...
    else if(button == GLUT_RIGHT_BUTTON && state ==
        GLUT_DOWN)
        ...
}
```



Movimento do Mouse

■ void myMovedMouse(int x, int y)

```
void myMovedMouse(int x, int y)
{
    ...
}
```



Interação com Teclado

- `void myKeyboard(unsigned int key, int x, int y)`
 - Key: valor ASCII da tecla pressionada
 - x e y: localização do mouse

```
void myKeyboard(unsigned int key, int x, int y) {  
    if(key == 'q') exit(1);  
}
```



Outras funções de *Callback*

- Para se registrar outros eventos, diferentes funções de *callback* devem ser registradas
 - `glutReshapeFunc(...)` : chamada quando é alterado o tamanho da janela – importante quando a razão de aspecto entre o tamanho da janela e a projeção ortogonal são diferentes



Exercício

- Fazer um programa que desenhe alguma imagem 2D
 - Esse programa deve possibilitar fazer zoom e pan



Bibliografia

- Computer Graphics Using OPEN GL, F.S. Hill, Prentice-Hall 2001
- E. Angel, Interactive Computer Graphics, 3a. Edição, Adison Wesley, 2003