INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SUL-RIO-GRANDENSE
Campus Pelotas

Computação Gráfica
Texto T02

(atualizada em: 12 mar. 2024)
glaucius@pelotas.ifsul.edu.br

OPENGL

O que é *OpenGL*?

OpenGL é uma biblioteca de rotinas gráficas de modelagem, manipulação de objetos e exibição tridimensional que permite a criação de aplicações que usam Computação Gráfica.

Seus recursos permitem ao usuário criar objetos gráficos com qualidade, de modo rápido, além de incluir recursos avançados de animação, tratamento de imagens e texturas é possível ter visualização em vários ângulos.

A biblioteca *OpenGL* foi introduzida em 1992 pela *Silicon Graphics*, no intuito de conceber uma *API* (Interface de Programação de Aplicação) gráfica independente de dispositivos de exibição.

Com isto, seria estabelecida uma ponte entre o processo de modelagem geométrica de objetos, situadas em um nível de abstração mais elevado, e as rotinas de exibição e de processamento de imagens implementadas em dispositivos (*hardware*) e sistemas operacionais específicos.

A função utilizada pelo *OpenGL* para desenhar um ponto na tela, por exemplo, possui os mesmos nomes e parâmetros em todos os sistemas operacionais nos quais *OpenGL* foi implementada, e produz o mesmo efeito de exibição em cada um destes sistemas.

Diante das funcionalidades providas pelo *OpenGL*, tal biblioteca tem se tornado um padrão amplamente utilizado na indústria de desenvolvimento de aplicações.

Este fato tem sido adotado também pela facilidade de aprendizado, pela estabilidade das rotinas e pelos resultados visuais consistentes para qualquer sistema de exibição concordante com este padrão.

Diversos jogos, aplicações científicas e comerciais tem utilizado *OpenGL* como ferramenta de apresentação de recursos visuais, principalmente com a adoção deste padrão por parte dos fabricantes de placas de vídeo destinadas aos consumidores domésticos.

Todas as rotinas do *OpenGL* são implementadas na linguagem C, tornando fácil sua utilização em qualquer programa escrito em C ou C++.

As implementações do *OpenGL* geralmente proveem bibliotecas auxiliares, tais como a *GLU* (*OpenGL Utility Librar*y), utilizada para realizar tarefas comuns, tais como manipulação de matrizes, geração de superfícies e construção de objetos por composição.

As especificações do OpenGL não descrevem as interações entre *OpenGL* e o sistema de janelas utilizado (*Windows*, *X Window etc.*).

Assim, tarefas comuns em uma aplicação, tais como criar janelas gráficas, gerenciar eventos provenientes de mouse e teclados, e apresentação de menus ficam a cargo de bibliotecas próprias de cada sistema operacional.

Recursos Gráficos da *OpenGL*

- Modos de desenho de pontos
- Ajuste de largura de linhas
- Aplicação de transparência
- Ativação/desativação de serrilhamento (*aliasing*)
- Mapeamento de superfícies com textura
- Seleção de janela de desenho
- Manipulação de fontes/tipos de iluminação e sombreamento
- Transformação de sistemas de coordenadas
- Transformações em perspectiva
- Combinação de imagens (*blending*)

Instalação/Compilação da *OpenGL* em *Linux*

Como instalar a *OpenGL* em *Linux*?

Instale a biblioteca:

- *freeglut3-dev*

Como administrador, em um terminal, tecle:

```
apt update
apt install freeglut3-dev
```

Como instalar a *OpenGL* no Windows, para utilização com o Dev-C++?

Fonte: CARVALHO, M. A. G. de. **Instalação da biblioteca OpenGL no Dev-C++**. Campinas: Unicamp, jul. 2006. Disponível em: https://wordpress.ft.unicamp.br/magic/instalacao-da-biblioteca-opengl-no-dev-c/. Acesso em: 12 mar. 2024.

Como instalar a *OpenGL* no Mac OS X?

Fonte: VAIDYA. Mithilesh. **OpenGL on MacOS**. Disponível em: https://methi1999.github.io/2020/08/19/opengl.html. Acesso em: 12 mar. 2024.

COMO COMPILAR ARQUIVOS COM INCLUSÃO DA OPENGL?

Inclua ao início do código fonte:

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

Compile assim:

```
g++ codigo_fonte.cpp -o arq_executavel -lglut -lGL -lGLU
```

ou, se estiver utilizando a biblioteca matemática (*cmath*):

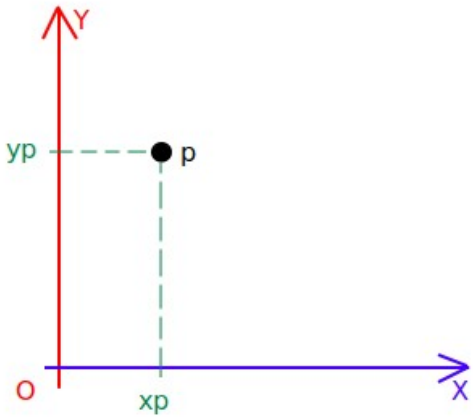
```
g++ codigo_fonte.cpp -o arq_executavel -lglut -lGL -lGL –lm
```

Como executar o programa compilado?

```
./arq_executavel
```

Sistema de Coordenadas 2D

Figura 1 – Sistema de coordenadas 2D.



Fonte: Autor.

Exemplo 1 - Desenho de pontos

Em um aplicativo para edição de textos gere o seguinte código fonte de programa, em Linguagem C++:

```
#include <GL/gl.h>      //biblioteca gl.h
#include <GL/glu.h>     //biblioteca glu.h
#include <GL/glut.h>    //biblioteca glut - ferramentas adicionais

void Inicializa(void) {
        //seleciona cor do fundo azul...
        glClearColor(0, 0, 1, 0);
}

void Desenha(void) {
        // Limpa a janela de visualização com a cor de fundo especificada
        glClear(GL_COLOR_BUFFER_BIT);
        // Especifica que a cor corrente é o branco
        glColor3f(1, 1, 1);
        // Desenha quatro pontos brancos
        glBegin(GL_POINTS);
                glVertex2i(10,10);
                glVertex2i(290,10);
                glVertex2i(290,290);
                glVertex2i(10,290);
        glEnd();
        // Executa os comandos OpenGL
        glFlush();
}

// 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, 300, 0, 300*h/w);
        else
                gluOrtho2D(0, 300*w/h, 0, 300);
}
```

```
//Declara o tamanho da janela, posicao e modo de
//visualizacao...(single buffer e RGBA).
//Abre uma janela com o título "4 Pontos" na barra de título.
//Chama as rotinas de inicializacao. Entra no loop principal e
//processa os eventos...
int main(int argc, char **argv) {
        //inicializa a lib glut com parametros default
        glutInit(&argc, argv);
        //define o modo de exibição, usando tipo simples e sistema de cor RGB
        glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        //define o tamanho da janela
        glutInitWindowSize(300, 300);
        //define a posição da janela
        glutInitWindowPosition(10,10);
        //define o título da janela
        glutCreateWindow("4 Pontos");
        //chama funcao callback de exibicao
        glutDisplayFunc(Desenha);
        //chama funcao para alterar o tamanho da janela corrente
        glutReshapeFunc(AlteraTamanhoJanela);
        //chama funcao de inicializacao
        Inicializa();
        //realiza um loopback na janela ativa
        glutMainLoop();
        //requer que funcao main retorne um inteiro (0)
        return 0;

}
```

Salve o arquivo como pontos.cpp.

Gere o código executável, teclando:

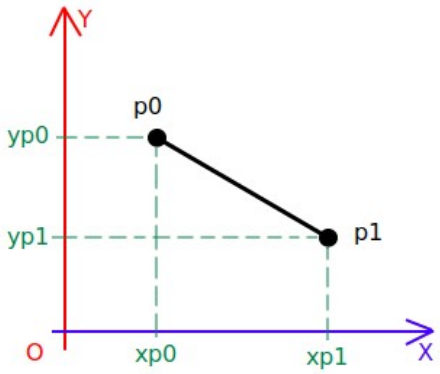
```
g++ pontos.cpp -o pontos -lglut -lGL -lGLU
```

Execute o programa, teclado:

```
./pontos
```

Desenho de linhas

Figura 2 – Modelagem de uma linha no espaço 2D.



Fonte: Autor.

Exemplo 2 - Desenho de linhas por coordenadas a cada dois pontos

```
...
        glBegin(GL_LINES);
                glVertex2i(10,10);
                glVertex2i(290,10);
                glVertex2i(290,290);
                glVertex2i(10,290);
        glEnd();
...
```

Estilo de linhas

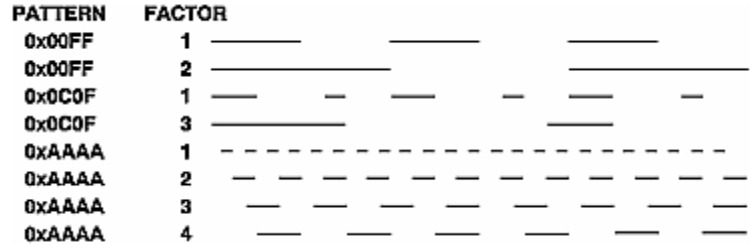
```
...
        //Estilo das linhas
        glEnable(GL_LINE_SMOOTH);
        // abilita o desenho de linha interrompida
        glEnable(GL_LINE_STIPPLE);
        // define máscara
        glLineStipple(2, 58360);
...


...
        //Desabilita estilo das linhas
        glDisable(GL_LINE_SMOOTH);
        glDisable(GL_LINE_STIPPLE);
...
```

A função glLineStipple(factor, pattern) é utilizada para definir qual padrão exatamente você quer aplicado à linha. O parâmetro do padrão representa 16 *bits* que definem onde a linha será pintada. Por exemplo, o padrão 0x0F0F representa em binário: 0000000011111111000000011111111. Agora, simplesmente imagine uma linha onde tem 1s e nenhuma onde tem zeros. Na verdade, o padrão é desenhado ao contrário, do último 1 até o primeiro.

Cada bit será considerado um número de vezes igual ao fator. Então um padrão que comece com: 01110111 será encarado como 000111111111000111111111 para um fator igual a 3.

Vejamos graficamente alguns exemplos, retirados do próprio manual da *OpenGL*:

Figura 3 – Padrões para definição de estilos de linhas.



Fonte: Stack Overflow (2021).
Disponível em: https://i.stack.imgur.com/pwGcB.gif . Acesso em: 22 mar. 2021.

Espessura das linhas

```
...
        glLineWidth(3);
...
```

Tamanho dos pontos

```
...
        glPointSize(5);
...
```