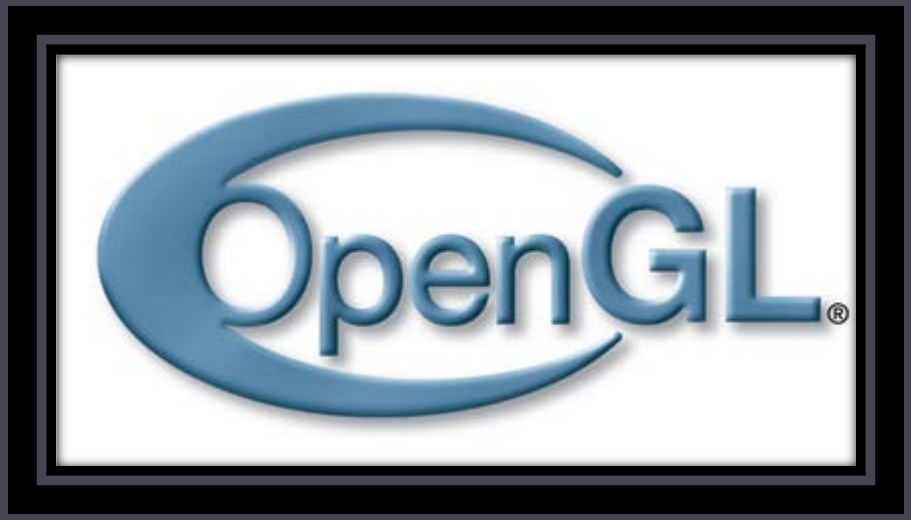


Iniciando com OpenGL

Rossana Baptista Queiroz

UNISINOS

6 de março de 2014

O que é OpenGL?

- OpenGL (Open Graphics Library)
  - “interface para hardware gráfico”
    - Biblioteca de rotinas gráficas e de modelagem
    - 2D e 3D
    - Portável e rápida
  - Especificação aberta
    - Implementação distribuída em SDKs e *drivers* de placas gráficas
    - Licenças variadas
  - Versão atual: 4.4

OpenGL

- Início:
  - Biblioteca gráfica desenvolvida para rodar aplicações gráficas sobre o hardware proprietário da Silicon Graphics Inc. (SGI).
  - Primeira versão foi o IRIS GL, biblioteca gráfica das estações IRIS Graphics da SGI.
    - Hardware que provia transformações de matrizes com altíssima performance.
  - OpenGL surge da decisão de portar a IRIS GL para um padrão aberto, para suportar outros fabricantes de dispositivos gráficos, outras plataformas de hardware e outros sistemas operacionais.

OpenGL

- OpenGL ARB:
  - API aberta não pode ser de propriedade de uma empresa apenas.
  - Surge então o *OpenGL Architecture Review Board (ARB)*
  - Consórcio, originalmente, formado por SGI, Digital Equipament Corporation, IBM, Intel e Microsoft. Outras empresas fazem parte do consórcio, como: SUN, nVIDIA, ATI, ...
  - 01/07/1992 sai a primeira versão da especificação da OpenGL.
  - A partir de 2006 SGI transfere o controle sobre o padrão OpenGL para o grupo de trabalho *The Khronos Group* (www.khronos.org).
  - Atualmente, este grupo promove o OpenGL e outros padrões como o OpenGL ES (para dispositivos móveis) e WebGL (para web).

OpenGL

- Software x hardware:
  - SW tem muito menos performance que HW e alguns efeitos especiais podem ser proibitivos nas aplicações gráficas.
  - SW pode executar em qualquer plataforma sem que o usuário necessite de hardware especial, no caso hardware gráfico.
  - HW tem performance, porém maior custo.
  - OpenGL interage com o hardware. Uma aplicação gráfica que use a OpenGL necessitará que o dispositivo gráfico (placa-de-vídeo) e o driver gráfico implementem a especificação da OpenGL.

OpenGL

- Quando usamos a OpenGL, em vez de descrever detalhadamente uma cena 2D ou 3D, basta especificar o conjunto de passos que devem ser seguidos para se obter o aspecto ou efeito desejado.
- Funções com prefixo `gl`

- Não possui funções para o gerenciamento de janelas e eventos
  - GLU – OpenGL Utility Library
  - GLUT – OpenGL Utility Toolkit
  - FLTK, wxWidgets, SDL, Qt...

OpenGL

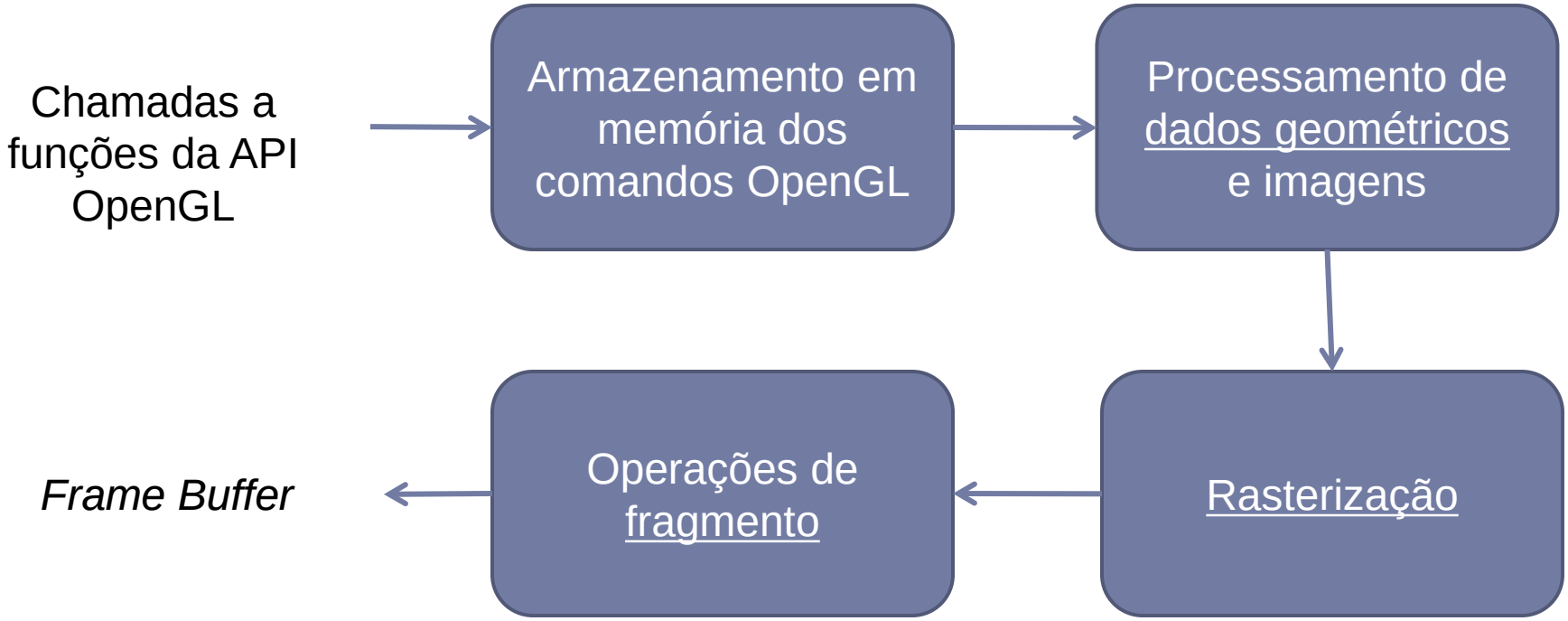
- GLU
  - Consiste de funções que utilizam os recurso de baixo nível da biblioteca OpenGL para prover rotinas de desenho de alto nível.
  - Normalmente é distribuída junto com o pacote básico do OpenGL
  - Mapeamento de coordenadas entre o espaço de tela e do objeto
  - Desenho de superfícies quádricas e NURBS
  - Geração de *mipmaps* de textura
  - *Tesselation* de primitivas poligonais
  - Interpretação dos códigos de erros do OpenGL
  - Funções com prefixo `glu`

OpenGL

- GLUT
  - criação e controle de janelas
  - tratamento de eventos de dispositivos de entrada (mouse e teclado)
  - desenho de formas tridimensionais pré-definidas (como cubo, esfera, bule, etc)
  - Funções com prefixo `glut`

  - **OpenGL + GLUT**: a combinação mais adequada para aprender a programar em OpenGL e para construir aplicações pequenas

Pipeline do OpenGL



O que é *pipeline*? É uma palavra usada para descrever um processo composto de duas ou mais etapas para a geração de uma imagem.

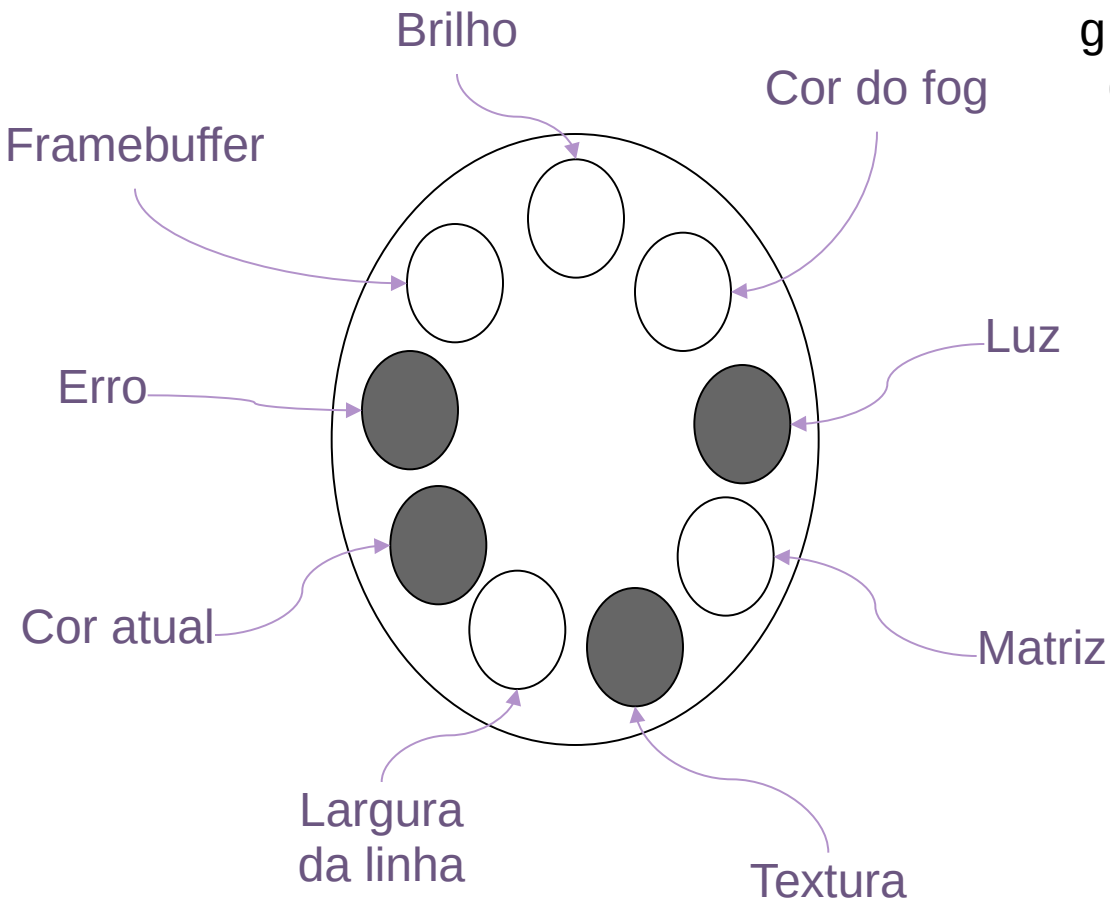
Como funciona a OpenGL

- API (Application Prigramming Interface) procedural. Programador invoca comandos OpenGL para compor uma cena.
- De forma mais básica, os objetos são desenhados com primitivas gráficas: pontos, linhas e polígonos em 3D.
- É uma **máquina de estados**. É necessário alterar/habilitar os estados de acordo com a descrição da cena que se deseja gerar, através da chamada de funções
  - Ex.: a cor corrente
- Mantém uma série de **variáveis de estado**, tais como estilo (ou padrão) de uma linha, posições e características das luzes, e propriedades do material dos objetos que estão sendo desenhados.

Como funciona a OpenGL

- Permite visualização de objetos em qualquer ponto de um espaço 3D.
- Suporta também iluminação e sombreamento, mapeamento de textura, *blending*, transparência, animação e diversos outros efeitos especiais.
- Faz a conversão das primitivas para imagem – rasterização.
- Também não existe nenhum formato de arquivo associado ao OpenGL para modelos ou ambientes virtuais. O programador
deve fazer a carga e interpretação dos formatos de arquivos comuns e converter para primitivas gráficas.

Como funciona a OpenGL




```
glEnable() e
 glDisable()
```

Iniciando com o VS2010

- Windows fornece os arquivos *opengl32.dll* e *glu32.dll*, necessários para execução de programas OpenGL
- Para desenvolver aplicações, é necessário as bibliotecas estáticas (.lib) e os arquivos de cabeçalhos (.h). A SDK do Visual Studio já fornece:
  - *opengl32.lib* (OpenGL) e *glu32.lib* (GLU)
  - *gl.h* e *glu.h*.
  - Estes arquivos normalmente estão localizados em uma pasta especial no *path* do *include*.

Iniciando com o VS2010

- GLUT
  - É necessário baixar a GLUT (pode ser a versão pré-compilada)
  - Colocar o arquivo de cabeçalho (glut.h) no diretório **include**/GL do VS 2010
    - Ex.: C:\Program Files\Microsoft Visual Studio 10.0\VC\include\GL
  - Colocar a biblioteca estática (glut32.lib) no diretório **lib**
    - C:\Program Files\Microsoft Visual Studio 10.0\VC\lib
  - Colocar a biblioteca dinâmica (glut32.dll) no diretório System32 do Windows
    - C:\Windows\System32

Iniciando um programa

- Inserir os arquivos de cabeçalho no seu programa:

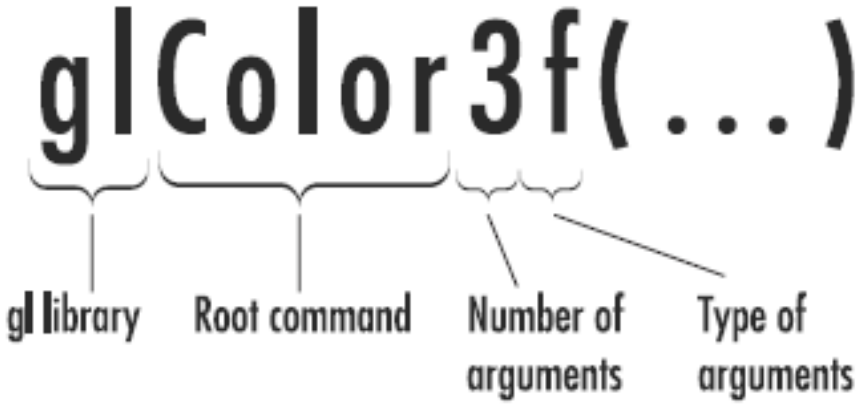
```
#include <GL/gl.h>
#include <GL/glu.h>
```

- Utilizando a GLUT, só precisa incluir o cabeçalho dela, pois as outras estão incluídas já no arquivo

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

Programando com OpenGL

- Convenção de nomes




```
glVertex3f(0.0f, 0.0f, 0.0f);
glVertex3i(0, 0, 0);
glVertex3v(vertex[0]);
```

Programando com OpenGL

- Tipos de dados para se usar com a API OpenGL

OpenGL Data Type	Internal Representation	Defined as C Type	C Literal Suffix
GLbyte	8-bit integer	signed char	b
GLshort	16-bit integer	short	s
GLint, GLsizei	32-bit integer	long	l
GLfloat, GLclampf	32-bit floating point	float	f
GLdouble, GLclampd	64-bit floating point	double	d
GLubyte, GLboolean	8-bit unsigned integer	unsigned char	ub
GLushort	16-bit unsigned integer	unsigned short	us
GLuint, GLenum, GLbitfield	32-bit unsigned integer	unsigned long	ui
GLchar	8-bit character	char	None
GLsizeiptr, GLintptr	native pointer	ptrdiff_t	None

Programando com OpenGL

- Convenção de nomes
  - Começam com `gl...`
  - Próximas palavras começa com a primeira letra maiúscula.
    - `glBegin()`
  - Constantes começam com GL_ e são sempre em maiúsculas.
    - `GL_COLOR_BUFFER_BIT`

Programando com OpenGL

- Lembre-se:
  - OpenGL é **máquina de estados**, desenha primitivas e o estado das variáveis altera o resultado da síntese da imagem.
    - Um objeto é desenhado com a cor que está definida, com iluminação que está definida, com as transformações previamente definidas e etc.
    - Portanto, um vez definida um propriedade, ela ficará “residente” até tenha o seu estado limpo ou alterado com outro valor.
    - Por exemplo, quando uma cor é definida ela será usada em qualquer processo de colorização até que ocorra um novo comando de cor.
  - OpenGL não faz interação com usuário, portanto, não gerencia entrada e saída de dados. É necessário o uso de alguma biblioteca para gerenciamento de GUI. Padrão GLUT.

Programando com OpenGL

- Como uma máquina de estados, os comandos também podem ter seu estado definido/alterado apenas como ligado ou desligado.
  - `glEnable()` e `glDisable()`
- É o caso da iluminação:
  - `glEnable(GL_LIGHTING)`
- **CUIDADO**: a troca frequente de estados afeta a performance. O melhor é “setar” os estados uma vez só, quando possível.

Funções de *callback*

- A GLUT é baseada em eventos
- Funções de *callback*
  - Funções que não são chamadas pelo programador, e sim pela GLUT
  - Eventos de mouse, teclado, redesenho,
  - Precisam ser registradas no início do programa
- Uma das funções de *callback* que todo programa com OpenGL+GLUT deve ter é uma função que desenha na tela

Inicializando a GLUT

- Na própria função *main* ou em uma função criada por você, chamada no *main*

```
// 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");

                                                    continua...
```

Inicializando a GLUT


```
// 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;
}
```

Registro de Funções de *Callback*

- Exemplo:

```
glutDisplayFunc(Desenha);
glutReshapeFunc(AlteraTamanhoJanela);
glutKeyboardFunc(Teclado);
glutSpecialFunc(TeclasEspeciais);
glutMouseFunc(GerenciaMouse);
glutMotionFunc(MoveMouseBotaoPressionado);
glutPassiveMotionFunc(MoveMouse);
glutIdleFunc(Idle);
```

Inicializando a OpenGL


```
// 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);
}
```

- `glMatrixMode(Glenum mode)`
  - Permite identificar com qual tipo de matriz se vai trabalhar (de projeção e de modelo)
- `gluOrtho2D(GLDouble left, GLDouble right, GLDouble bottom, GLDouble top)`
  - Define que está trabalhando com cenas 3D

Callback de Desenho

```
// 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();
}
```

Callback de Teclado


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

Programando com OpenGL

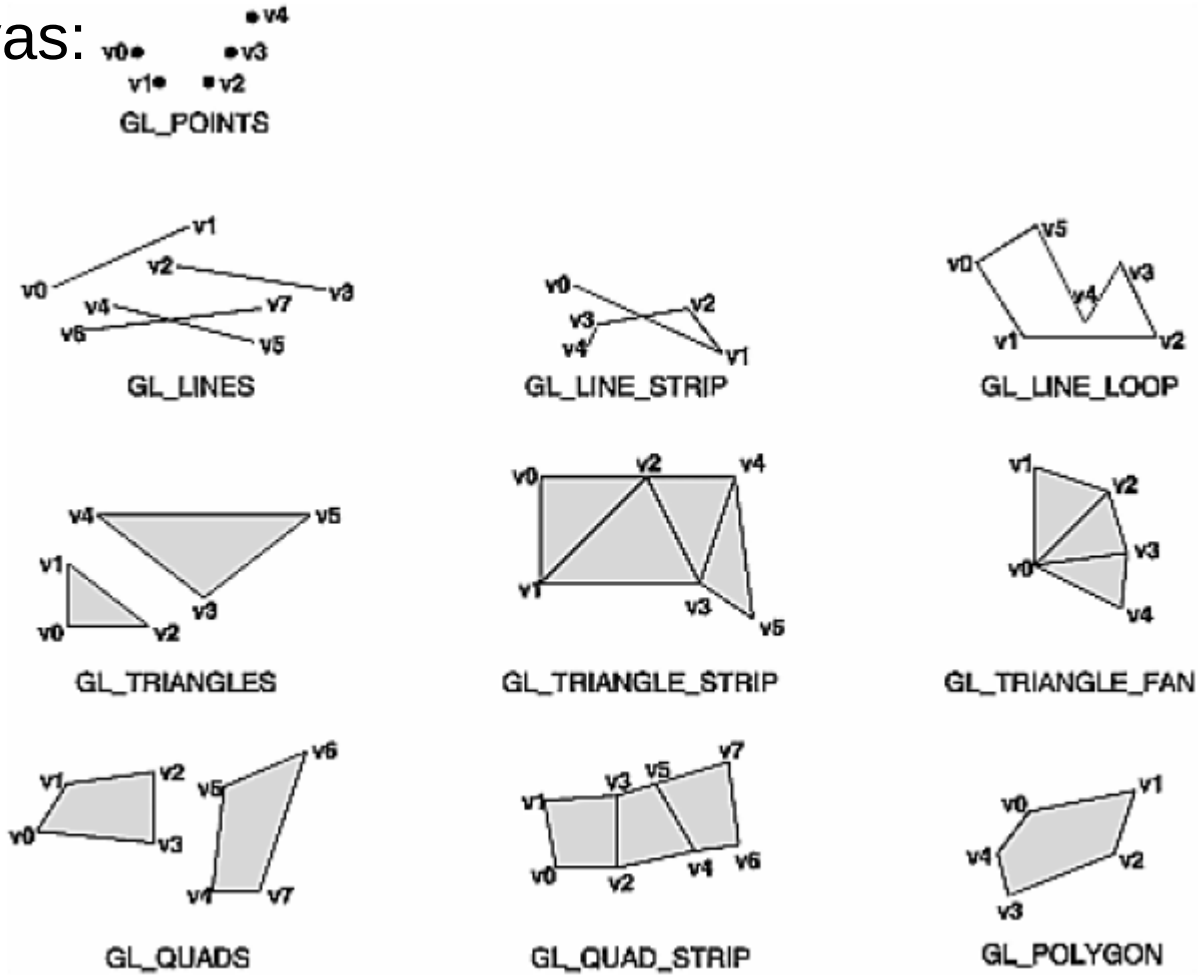
- Desenho de primitivas:
  - Basicamente, é a definição dos vértices das primitivas e das propriedades de cor, normais, texturas e etc.
  - Tudo definido entre `glBegin(<`*`primitiva`*`>)` e `glEnd()`
  - Exemplo:

```
glBegin(GL_TRIANGLES);
   glColor3f(0.0, 0.0, 1.0);
   glVertex3f(0.0, 0.0, 0.0);
   glVertex3f(1.0, 0.0, 0.0);
   glVertex3f(0.5, 0.5, 0.0);
glEnd();
```

  - Outros comandos utilizados dentro do bloco:
    - `glColor*(), glNormal*(), glMaterial*()`

Programando com OpenGL

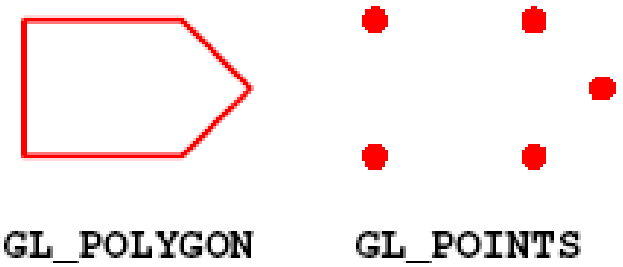
- Primitivas:



Programando com OpenGL

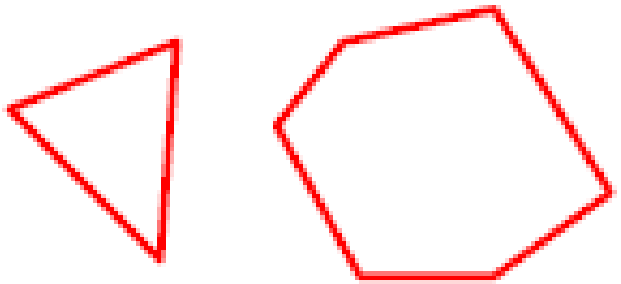
- Primitivas. Exemplos:

```
glBegin(GL_POLYGON); // ou glBegin(GL_POINTS);
  glVertex2f(0.0, 0.0);
  glVertex2f(0.0, 3.0);
  glVertex2f(4.0, 3.0);
  glVertex2f(6.0, 1.5);
  glVertex2f(4.0, 0.0);
glEnd();
```

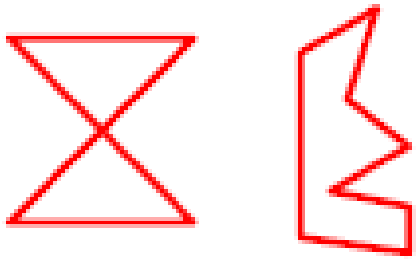


Programando com OpenGL

- Tipos de polígonos:



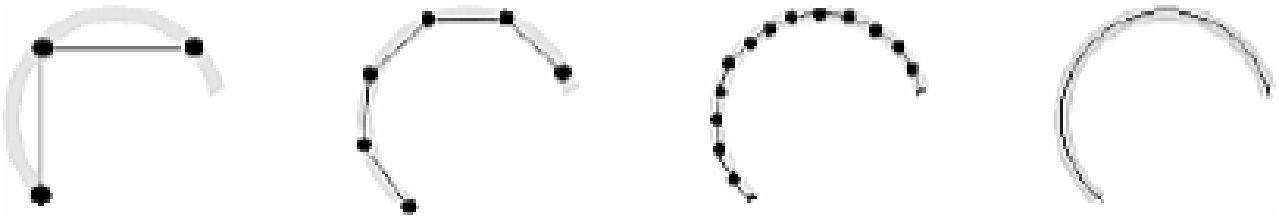
Válidos



Inválidos

Programando com OpenGL

- Objetos curvos são aproximados por retas:



Programando com OpenGL

- Exemplo. Desenhando um círculo:

```
#define PI 3.1415926535898
GLfloat circle_points = 100.0f;
GLfloat angle, raioX=1.0f, raioY=1.0f;
glBegin(GL_LINE_LOOP);
  for (int i = 0; i < circle_points; i++) {
        angle = 2*PI*i/circle_points;
        glVertex2f(cos(angle)*raioX,
  sin(angle)*raioY);
  }
glEnd();
```

Note que:

- O tamanho do círculo é dado pelo raio, no exemplo é desenhado um círculo com raio 1
- Definindo o valor dos raios X e Y com valores diferentes, pode-se desenhar uma elipse.

Programando com OpenGL

- Modo de polígono:
  - `glPolygonMode(<lado>,<modo>)`
  - **`<lado>`**
    - `GL_FRONT_AND_BACK`
    - `GL_FRONT`
    - `GL_BACK`
  - **`<modo>`**
    - `GL_POINT`
    - `GL_LINE`
    - `GL_FILL`

Programando com OpenGL

- Matrizes:
  - Três tipos de matrizes:
  - GL_MODELVIEW:
    - A matriz *modelview* controla as transformações dos vértices dos objetos da cena
  - GL_PROJECTION
    - A matriz de projeção controla como a cena 3-D é projetada em 2-D
  - GL_TEXTURE
    - A matriz de texturas (geralmente pouco conhecida e utilizada) transforma as coordenadas das textura para obter efeitos como projetar e deslocar texturas

Programando com OpenGL

- Matrizes de projeção:
  - `glFrustum(left,right,bottom,top,near,far)`
  - `glOrtho(left,right,bottom,top,near,far)`
  - `gluPerspective(fovy,aspect,zNear,zFar)`
  - `gluOrtho2D(left,right,bottom,top)`
  - `gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)`
- Coordenadas de Tela
  - `glViewport(x, y, width, height)`

Referências bibliográficas

1. http://www.inf.pucrs.br/~manssour/OpenGL/Tutorial.html
2. COHEN, Marcelo, MANSSOUR, Isabel. OpenGL: uma abordagem prática e objetiva. São Paulo : Novatec, 2006. 478 p.
3. Notas de Aula do professor Leandro Tonietto (http://professor.unisinos.br/ltonietto/jed/pgr/pgr2011_02.html)

Referências bibliográficas

- www.opengl.org
- www.khronos.org
- OpenGL Programming Guide (Woo, Neider, Davis - Addison-Wesley)
- OpenGL Programming for the X Window System
- OpenGL Game Programming (Astle, Hawkins, LaMothe)

Referências bibliográficas

- Red Book
  - http://fly.cc.fer.hr/~unreal/theredbook/
- Tutorial de OpenGL (Português)
  - http://www.ingleza.com.br/opengl/index.html
- Referência (API) para PyOpenGL
  - http://pyopengl.sourceforge.net/documentation/manual/
- Documentação para PyOpenGL
  - http://pyopengl.sourceforge.net/documentation/index.html
- Referência (API) para wxPython
  - http://www.wxpython.org/onlinedocs.php
- Tutorial básico de wxPython
  - http://www.wxpython.org/tutorial.php