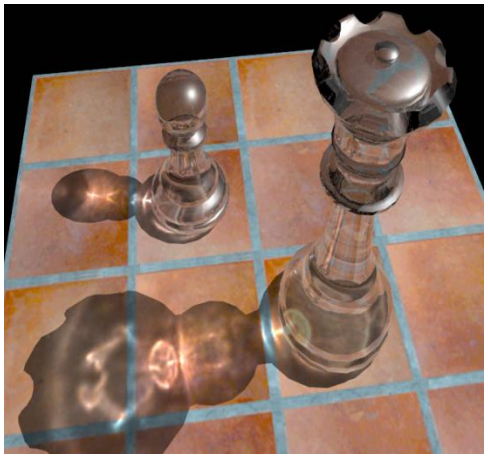
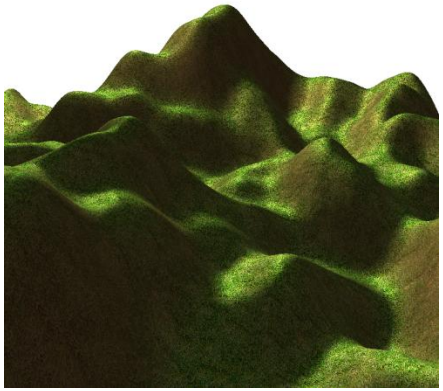
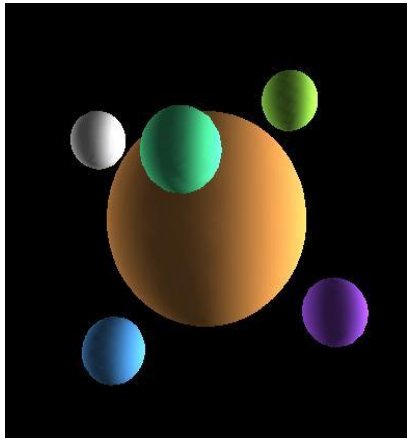
Introdução ao OpenGL



Professora: Maria Cristina Ferreira de Oliveira

PAE: Erick Gómez Nieto
egomezn@icmc.usp.br

Introdução as API's Gráficas

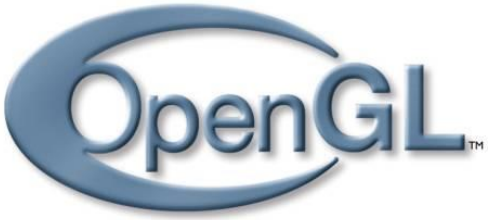
API (*Application programming interface)*

Interface implementada por um programa de software para permitir interação com outro software, similar a maneira que uma interface de usuário facilita a interação entre humanos e computador [1].

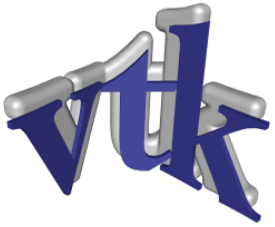
[1] Free On-line Dictionary of Computing. 1995-02-15. http://foldoc.org/Application+Program+Interface. Retrieved 2009-06-28.

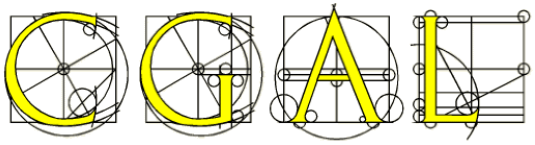
Graphical APIs

Best known



Also



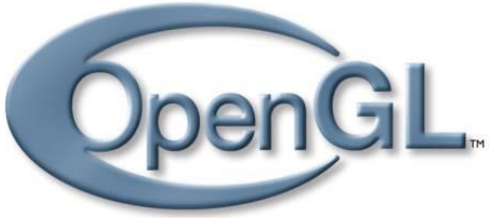


OpenGL

O OpenGL (Open Graphics Library) é uma API livre utilizada na computação gráfica, para desenvolvimento de aplicativos gráficos, ambientes 3D, jogos, entre outros.

OpenGL

- Independente de plataforma
- Fácil de usar
- Permite à aplicação extrair o melhor potencial do hardware



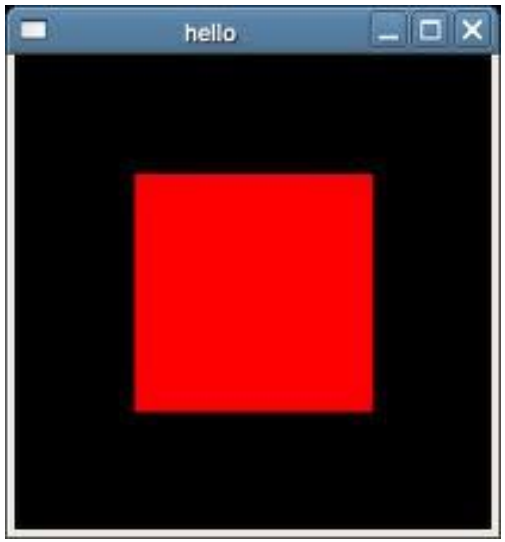
O que o OpenGL faz

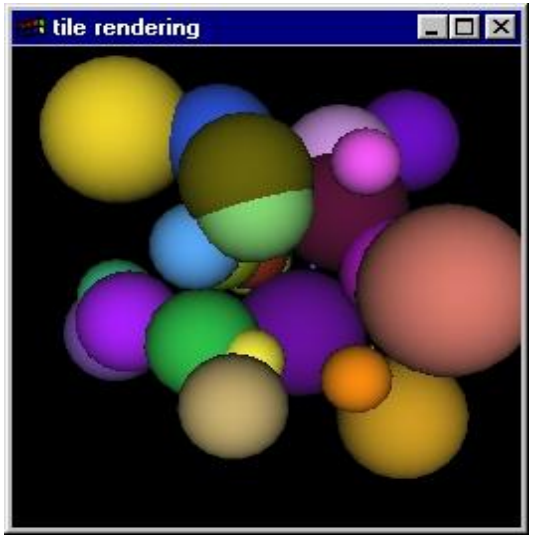
- Implementa *clipping* e algoritmos de rasterização
- Suporta linguagem para implementação de shaders: rotinas executadas no hardware para processamento de vértices e fragmentos
- Funções executadas no hardware: alto desempenho

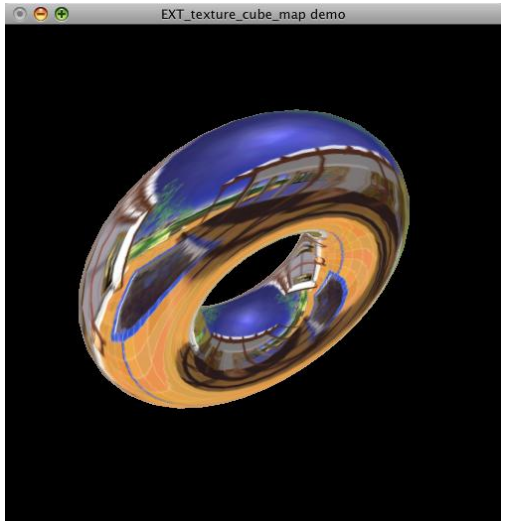
O que o OpenGL não faz

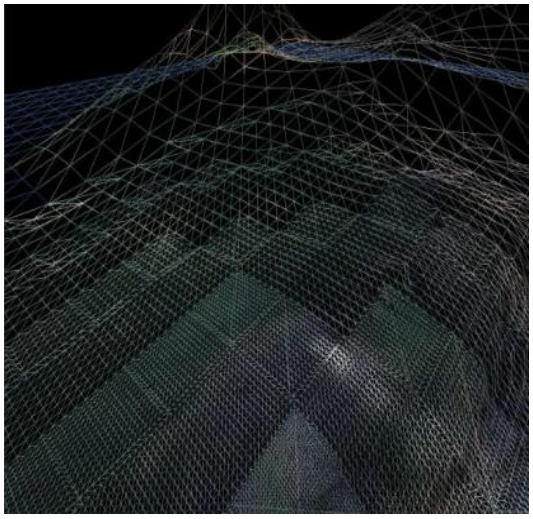
- Não inclui comandos para tratar janelas e entrada de dados
  - Feito pela GLUT
- Não inclui comandos de alto nível para descrever modelos geométricos complexos
  - Modelos de superfícies descritos por malhas de polígonos
- Não implementa algoritmos complexos de *rendering*, como *ray tracing* e radiosidade
  - Implementa o *scanline*

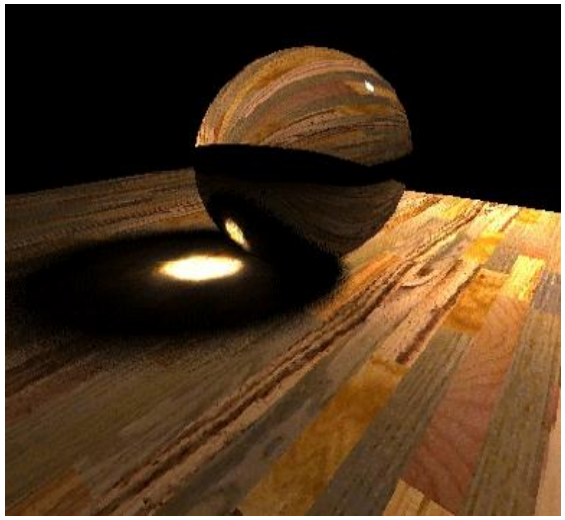
Exemplos:



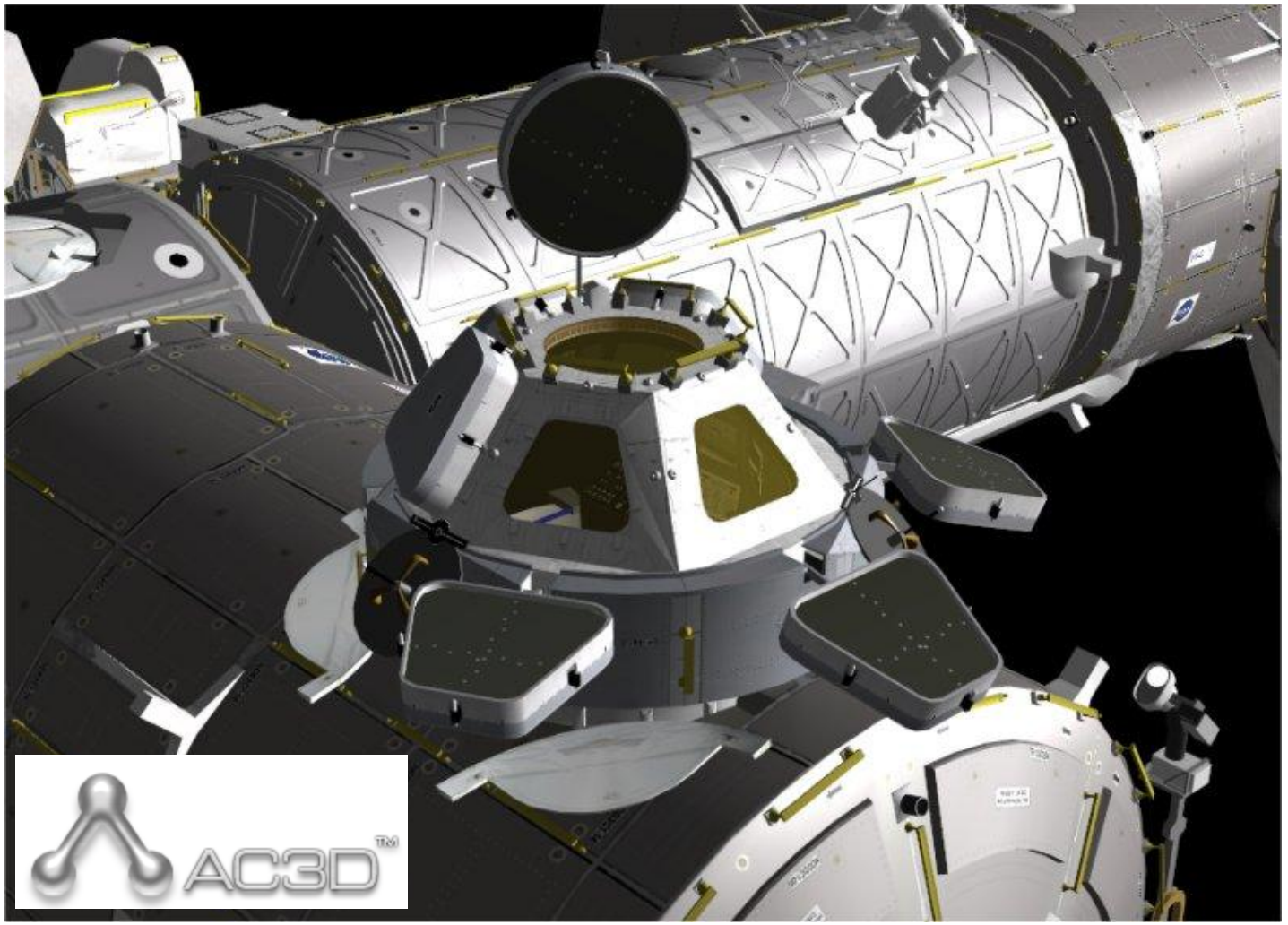








OpenGL Aplicações



AC3D http://www.inivis.com/index.html

OpenGL Aplicações

Alone in the Dark 4

Outros exemplos ...

Terrain

Collision

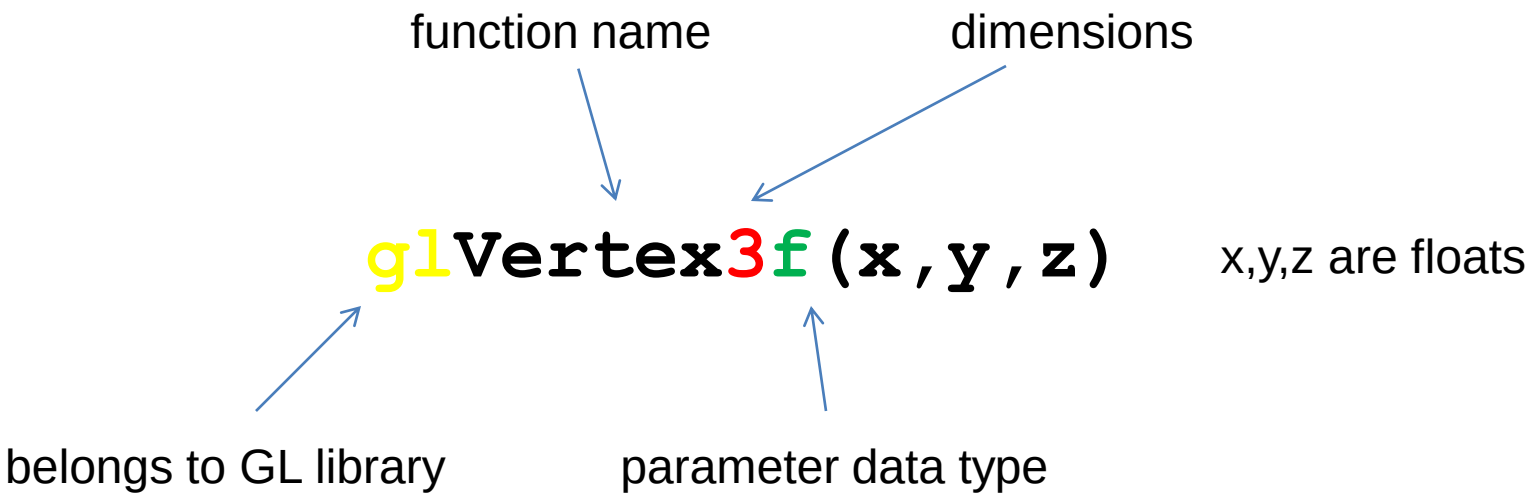
Environment

Xadrez

OpenGL – como funciona

- Fornece funções que permitem especificar os elementos necessários para gerar uma imagem:
  - Objetos, Posição da Câmera/Observador, Fonte de Luz, Materiais, Dispositivos de Interação, Capacidade do Sistema, etc.
- Programa OpenGl é uma máquina de estados, com 2 tipos de funções
  - Execução de primitivas: objetos são processados para atualizar a cena
  - Controle dos estados: mudança de estados e alteração dos valores dos atributos (ex., cor, espessura de primitiva...)

OpenGL Functions

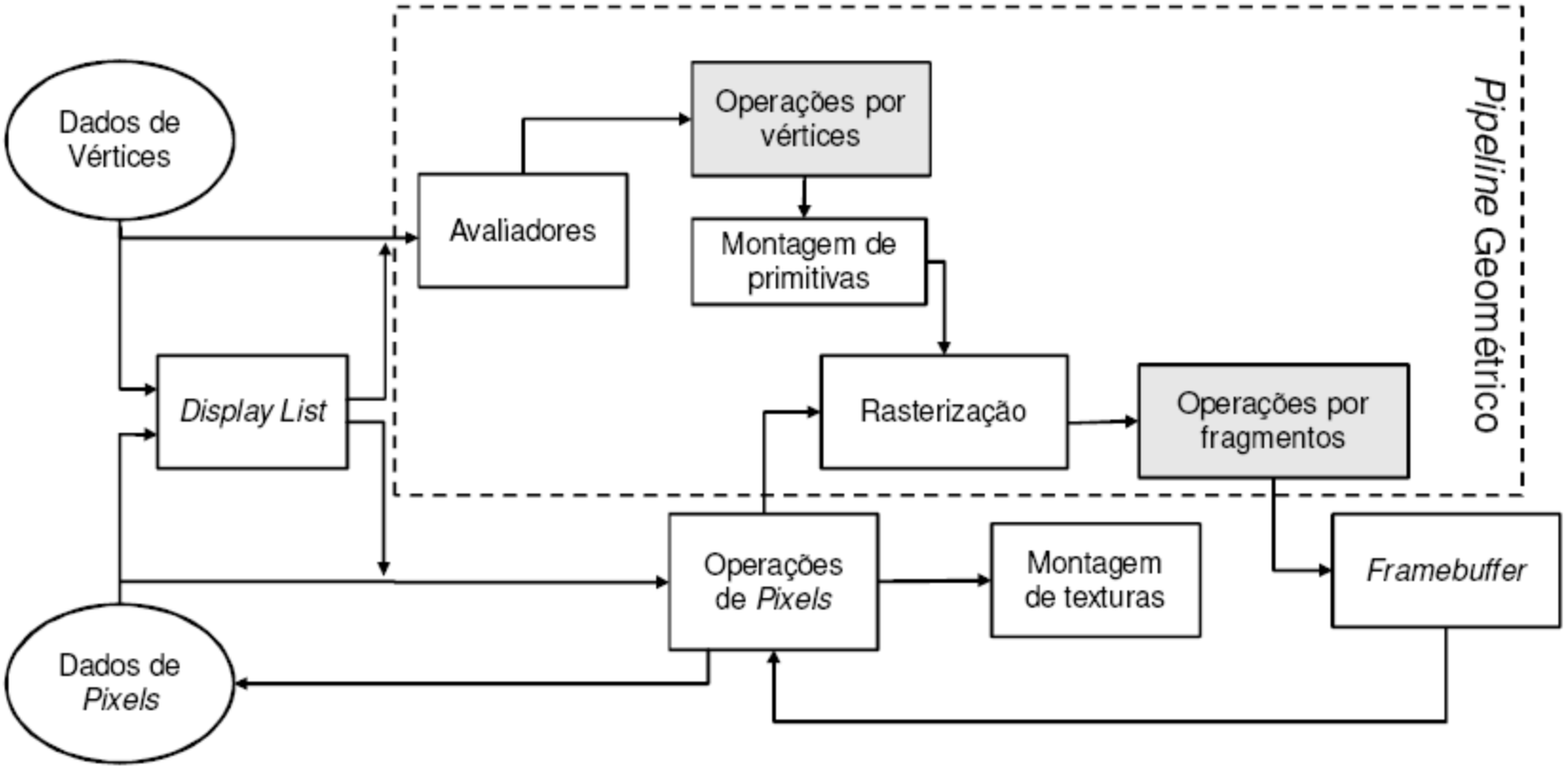



```
glVertex3f(p)
```

p is a pointer to an array

Primitives demo – nate robins

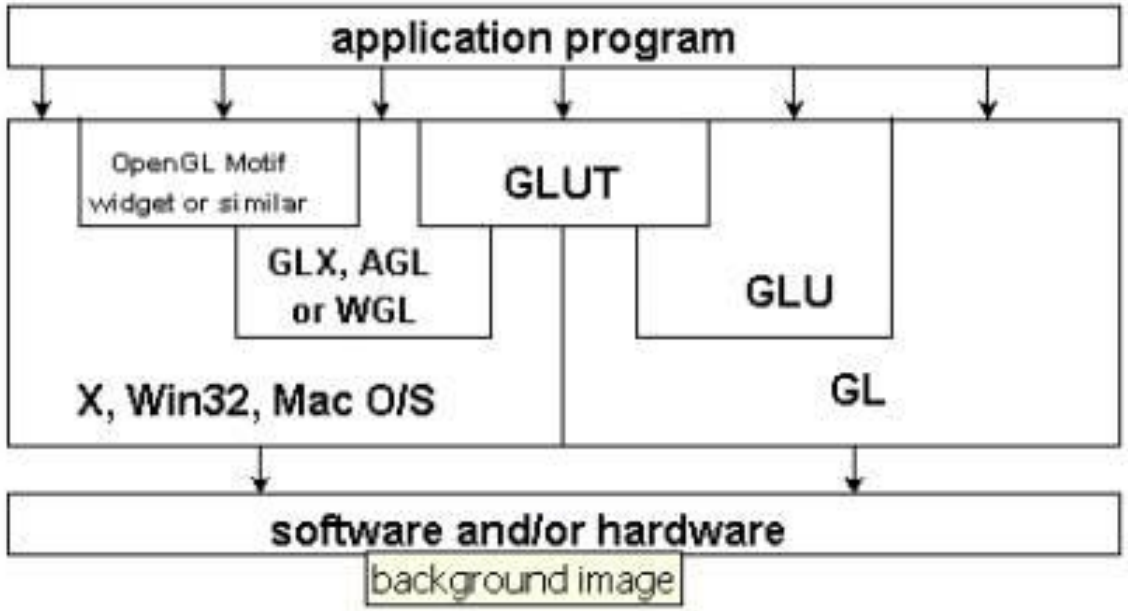
OpenGL - Pipeline de Rendering



Bibliotecas relacionadas

- **GLU:** definição de curvas e superfícies NURBs (*Non-Uniform Rational B-Spline*); Funções utilitárias para definir parâmetros de visualização
- **GLUT:** definição de janelas e tratamento de entrada de dados; criação de interfaces gráficas
- **GLUI:** criação de interfaces gráficas mais elaboradas

OpenGL and Related APIs



OpenGL Support Libraries

GLU Some additional functions for OpenGL programs.
GLUT The OpenGL utility toolkit.
freeglut Open source alternative to GLUT
GLUI a GUI toolkit made with GLUT
SDL The Simple DirectMedia Layer.
Glee The OpenGL Easy Extension library.
GLEW The OpenGL Extension Wrangler Library.
GLM C++ mathematics toolkit for OpenGL based on the GLSL specification.
SFML Simple and Fast Multimedia Library.
JOGL Java bindings for OpenGL API.

OpenGL Máquina de estados

GL é uma máquina de estados

- Têm muitas variáveis armazenadas como 'globais'
  - Que cor usar – glColor()
    - Todos os vértices a seguir (seguintes) usarão essa cor.
  - Qual modo de rendering (Aramado ou Sólido)?
    - Todos os poligonos seguintes serão *renderizados dessa forma.*
  - Que matriz está ativa (MODELVIEW or PROJECTION) – glMatrixMode()
    - Todos os comandos de matriz a seguir (subsequentes) afetarão essa matriz

OpenGL Tabela de estados

COLOR_CLEAR_VALUE **0,0,0,0**

Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute
DRAW_BUFFERi	$1 + \times Z_{10*}$	**GetIntegerv**	see 4.2.1	Draw buffer selected for output color i	4.2.1	color-buffer
INDEX_WRITEMASK	Z^+	**GetIntegerv**	1's	Color index writemask	4.2.2	color-buffer
COLOR_WRITEMASK	$4 \times B$	**GetBooleanv**	*True*	Color write enables; R, G, B, or A	4.2.2	color-buffer
DEPTH_WRITEMASK	B	**GetBooleanv**	*True*	Depth buffer enabled for writing	4.2.2	depth-buffer
STENCIL_WRITEMASK	Z^+	**GetIntegerv**	1's	Front stencil buffer writemask	4.2.2	stencil-buffer
STENCIL_BACK_WRITEMASK	Z^+	**GetIntegerv**	1's	Back stencil buffer writemask	4.2.2	stencil-buffer
COLOR_CLEAR_VALUE	C	**GetFloatv**	0,0,0,0	Color buffer clear value (RGBA mode)	4.2.3	color-buffer
INDEX_CLEAR_VALUE	CI	**GetFloatv**	0	Color buffer clear value (color index mode)	4.2.3	color-buffer
DEPTH_CLEAR_VALUE	R^+	**GetIntegerv**	1	Depth buffer clear value	4.2.3	depth-buffer
STENCIL_CLEAR_VALUE	Z^+	**GetIntegerv**	0	Stencil clear value	4.2.3	stencil-buffer
ACCUM_CLEAR_VALUE	$4 \times R^+$	**GetFloatv**	0	Accumulation buffer clear value	4.2.3	accum-buffer

OpenGL 2.0 Spec, Table 6.21. Framebuffer Control

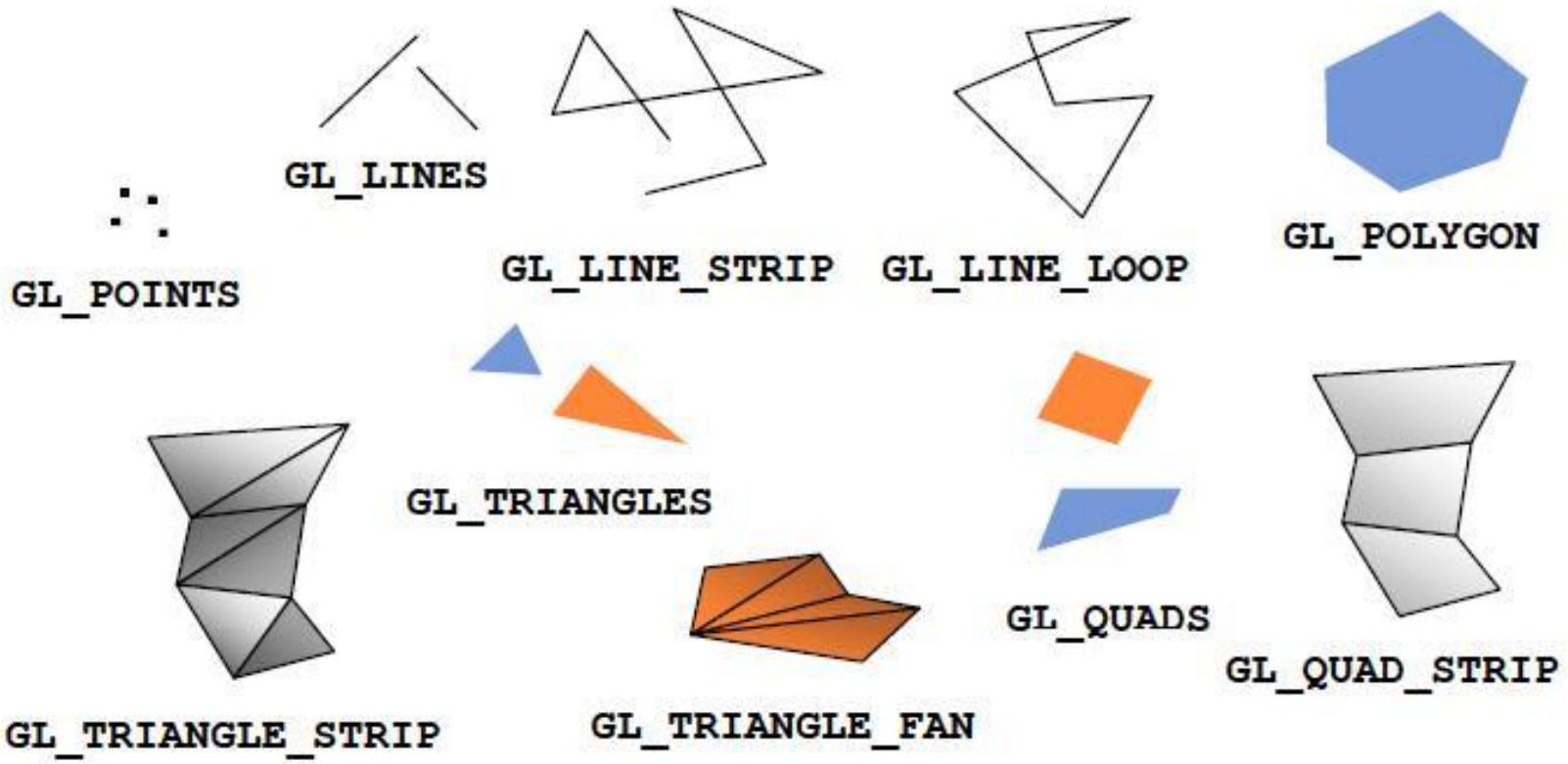
Get value	Type	Get Cmnd	Initial Value	Description	Sec.	Attribute
CURRENT_COLOR	C	**GetIntegerv, GetFloatv**	1,1,1,1	Current color	2.7	current
CURRENT_SECONDARY_COLOR	C	**GetIntegerv, GetFloatv**	0,0,0,1	Current secondary color	2.7	current
CURRENT_INDEX	CI	**GetIntegerv, GetFloatv**	1	Current color index	2.7	current
CURRENT_TEXTURE_COORDS	$2 * \times T$	**GetFloatv**	0,0,0,1	Current texture coordinates	2.7	current
CURRENT_NORMAL	N	**GetFloatv**	0,0,1	Current normal	2.7	current
CURRENT_FOG_COORD	R	**GetIntegerv, GetFloatv**	0	Current fog coordinate	2.7	current
–	C	–	-	Color associated with last vertex	2.6	–
–	CI	–	-	Color index associated with last vertex	2.6	–
–	T	–	-	Texture coordinates associated with last vertex	2.6	–
CURRENT_RASTER_POSITION	R^4	**GetFloatv**	0,0,0,1	Current raster position	2.13	current
CURRENT_RASTER_DISTANCE	R^+	**GetFloatv**	0	Current raster distance	2.13	current
CURRENT_RASTER_COLOR	C	**GetIntegerv, GetFloatv**	1,1,1,1	Color associated with raster position	2.13	current
CURRENT_RASTER_INDEX	CI	**GetIntegerv, GetFloatv**	1	Color index associated with raster position	2.13	current
CURRENT_RASTER_TEXTURE_COORDS	$2 * \times T$	**GetFloatv**	0,0,0,1	Texture coordinates associated with raster position	2.13	current
CURRENT_RASTER_POSITION_VALID	B	**GetBooleanv**	*True*	Raster position valid bit	2.13	current
EDGE_FLAG	B	**GetBooleanv**	*True*	Edge flag	2.6.2	current

OpenGL 2.0, Table 6.5. Current Values and Associated Data

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 de desenho
    - Distâncias medidas em pixels
    - Zero no canto superior esquerdo da janela
- Para desenho de primitivas 2D: GLOrtho2D define transformação necessária para a exibição
  - (janela 640 x 480)

Primitivas básicas: especificadas por seus vértices



Primitivas

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

```
glBegin(GL_POINTS);
  glVertex2i(100, 50);  // draw three points
  glVertex2i(100, 130);
  glVertex2i(150, 130);
glEnd();
```

  - Informação sobre cada vértice encaminhada para um ´*pipeline* gráfico´, no qual passa por várias etapas de processamento

Tipos de Dados

- OpenGL suporta um conjunto fixo de tipos de dados.

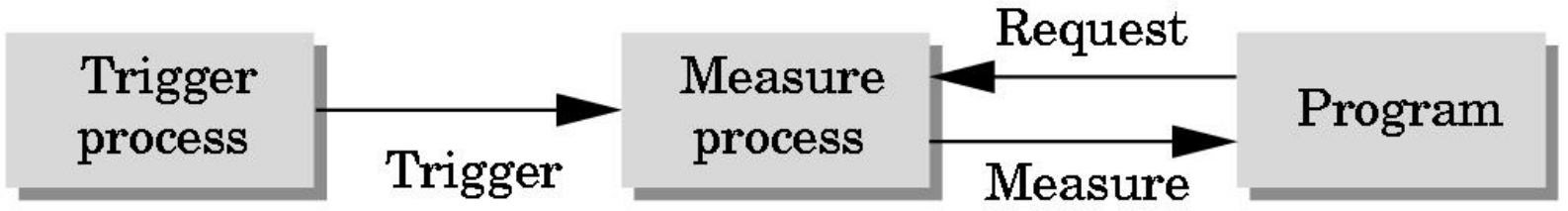
Suffix	Data Type	Typical Corresponding C-Language Type	OpenGL Type Definition
b	8-bit integer	signed char	GLbyte
s	16-bit integer	short	GLshort
i	32-bit integer	int or long	GLint, GLsizei
f	32-bit floating-point	float	GLfloat, GLclampf
d	64-bit floating-point	double	GLdouble, GLclampd
ub	8-bit unsigned integer	unsigned char	GLubyte, GLboolean
us	16-bit unsigned integer	unsigned short	GLushort
ui	32-bit unsigned integer	unsigned int or unsigned long	GLuint, GLenum, GLbitfield

Modos de Input

- Dispositivos de entrada incluem um *trigger* que sinaliza para o sistema operacional a ocorrência de um evento relevante
  - Botão no mouse pressionado/solto
  - Tecla pressionada/solta
  - …
- Quando ativados, dispositivos de entrada retornam informação (*measure*) para o sistema
  - Mouse retorna informação de posição
  - Teclado retorna código ASCII

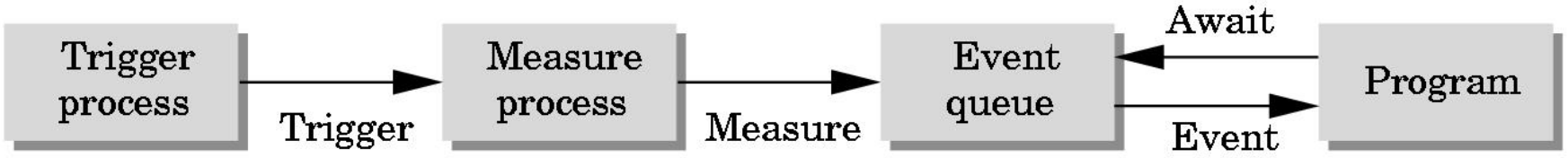
Modo Request

- Input fornecido ao programa somente quando o usuário ativa o dispositivo
- Típico para entrada via teclado
  - Pode apagar (backspace), editar, corrigir, etc. – até que a tecla enter (return) é pressionada (o *trigger*)



Modo Event

- Sistemas em geral tem mais de um dispositivo de entrada – cada qual pode ser ativado arbitrariamente pelo programador
- Cada trigger dispara um evento, cuja ´medida´ vai para uma fila de eventos, a qual é acessada pelo programa



Tipos de Eventos

- Window: resize, expose, iconify
- Mouse: click one or more buttons
- Motion: move mouse
- Keyboard: press/release a key
- Idle: ausência de eventos
  - Define o que deve ser feito se não há eventos na fila

Callbacks

- Interface de programação para entrada ´event-driven
- Define uma *callback function* para cada tipo de evento que o sistema gráfico reconhece
- Essa função, fornecida pelo programador, é executada quando o evento ocorre
- Exemplo GLUT: **`glutMouseFunc(mymouse)`**

mouse callback function

OpenGL Utility Toolkit (GLUT)

- Biblioteca para gerenciamento de eventos
  - Ex. glutMouseFunc(myMouse);
  - Registra função myMouse p/ tratar ocorrência de evento do mouse
  - Código da função definido pelo programador

GLUT callbacks

GLUT reconhece um subconjunto de eventos – os que são reconhecidos por sistemas de janelas (Windows, X, Macintosh)

- **`glutDisplayFunc`**
- **`glutMouseFunc`**
- **`glutReshapeFunc`**
- **`glutKeyboardFunc`**
- **`glutIdleFunc`**
- **`glutMotionFunc, glutPassiveMotionFunc`**

GLUT Event Loop

- A última linha em um programa **`main.c`** que usa a GLUT deve ser

```
glutMainLoop();
```

- Isso coloca o programa em um loop infinito de tratamento de eventos
- A cada passo desse loop de eventos, a GLUT
  - Verifica os eventos na fila
  - Para cada evento, executa a função callback apropriada (caso tenha sido definida)
  - Se nenhuma callback foi definida para o evento, ele é ignorado

Hello Opengl (*usando GLUT*)

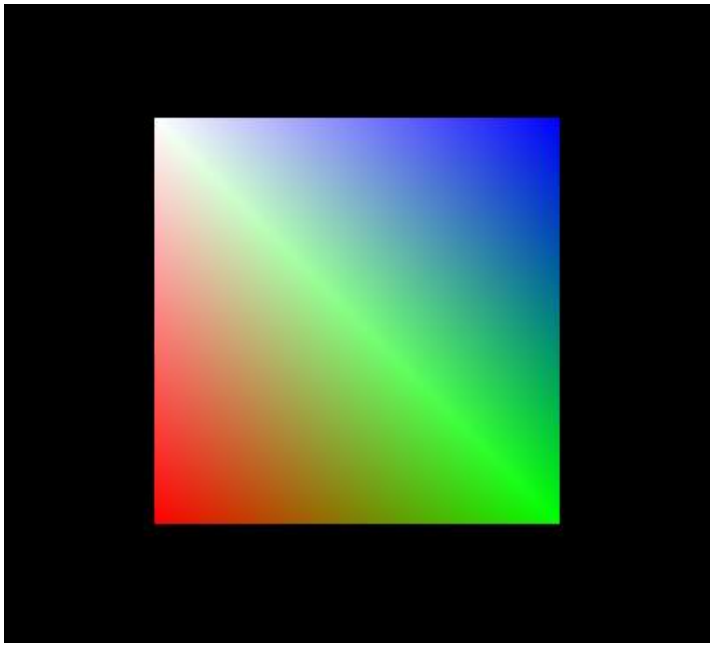
- Baixar *HelloGLUT.cpp* de
  http://www.lcad.icmc.usp.br/~egomez/CG/HelloGLUT.cpp
- Abrir o Dev–C++ e criar um projeto vazio.
- Adicionar ao projeto o *HelloGLUT.cpp*
- Compilar e rodar.
- Se tem problemas com o linker adicionar os parâmetros: *-lglut32 -lglu32 -lopengl32 -lwinmm -lgdi32*
  em Menú Projeto -> Opções de Projeto -> Parâmetros -> Linker

OpenGL Drawing a simple quad (*using GLUT*)

```
// Draw a colored square against a black background

#include <windows.h>
#include <stdio.h>
#define GLUT_DISABLE_ATEXIT_HACK
#include <GL/glut.h>

void draw() {
    glClear(GL_COLOR_BUFFER_BIT);
    glLoadIdentity();
    glOrtho(0, 4, 0, 4, -1, 1);
    glBegin(GL_POLYGON);
    glColor3f(1.0f,1.0f,1.0f);
    glVertex2i(1, 1);
    glColor3f(0.0f,0.0f,1.0f);
    glVertex2i(3, 1);
    glColor3f(0.0f,1.0f,0.0f);
    glVertex2i(3, 3);
    glColor3f(1.0f,0.0f,0.0f);
    glVertex2i(1, 3);
    glEnd();
    glFlush();
}

int main(int argc, char** argv) {
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    glutCreateWindow("whitesquare");
    glutDisplayFunc(draw);
    glutMainLoop();
}
```

OpenGL

GLUT



Run it !

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

- Ex. posicionar pontos com o mouse

```
void myMouse(int button, int state, int x, int y)
{
   if(button == GLUT_LEFT_BUTTON && state ==
   GLUT_DOWN)
     drawDot(x, screenHeight – y);
   else if(button == GLUT_RIGHT_BUTTON && state ==
   GLUT_DOWN)
     exit(-1)
}
```

Movimento do Mouse

- void myMovedMouse(int x, int y)

```
void myMovedMouse(int mouseX, int mouseY)
{
    GLint x = mouseX;
    GLint y = ScreenHeight - mouseY;
    GLint brushSize = 20;
    GLRecti(x,y, x + brushSize, y + brushSize);
    GLFlush();
}
```

Interação com Teclado

- void myKeyboard(unsigned int key, int x, int y)
- Key: valor ASCII da tecla pressionada

Exemplo de Interação com teclado e mouse

Terrain Rendering ....

Good Tutorials !!

Nate Robbins Tutorial	http://www.xmission.com/~nate/tutors.html
Neon Helion (NeHe)	http://nehe.gamedev.net/
Lighthouse 3D	http://www.lighthouse3d.com/opengl/
Introdução à OpenGL **Professora Isabel Harb Manssour**	http://www.inf.pucrs.br/~manssour/OpenGL/Tutorial.html
JPot Tutorial	http://www.cs.uwm.edu/~grafix2/

Bibliography

- HEARN, DONALD AND M. PAULINE BAKER . **Computer Graphics.**
- Martz, Paul. **OpenSceneGraph Quick Start Guide**
- **OpenGL Red Book.**
- **OpenGL Super Bible.**
- http://education.siggraph.org/resources/cgsource/instructional-materials
- http://graphics.stanford.edu/courses/cs248-07/