Unidade 6 - Introdução ao OpenGL

IME 04-10842
Computação Gráfica
Professor Guilherme Mota

Definição

Padrão aberto de arquitetura para computação gráfica que especifica hardware e uma API de software .

Exemplos de OpenGL

http://www.rage.com/

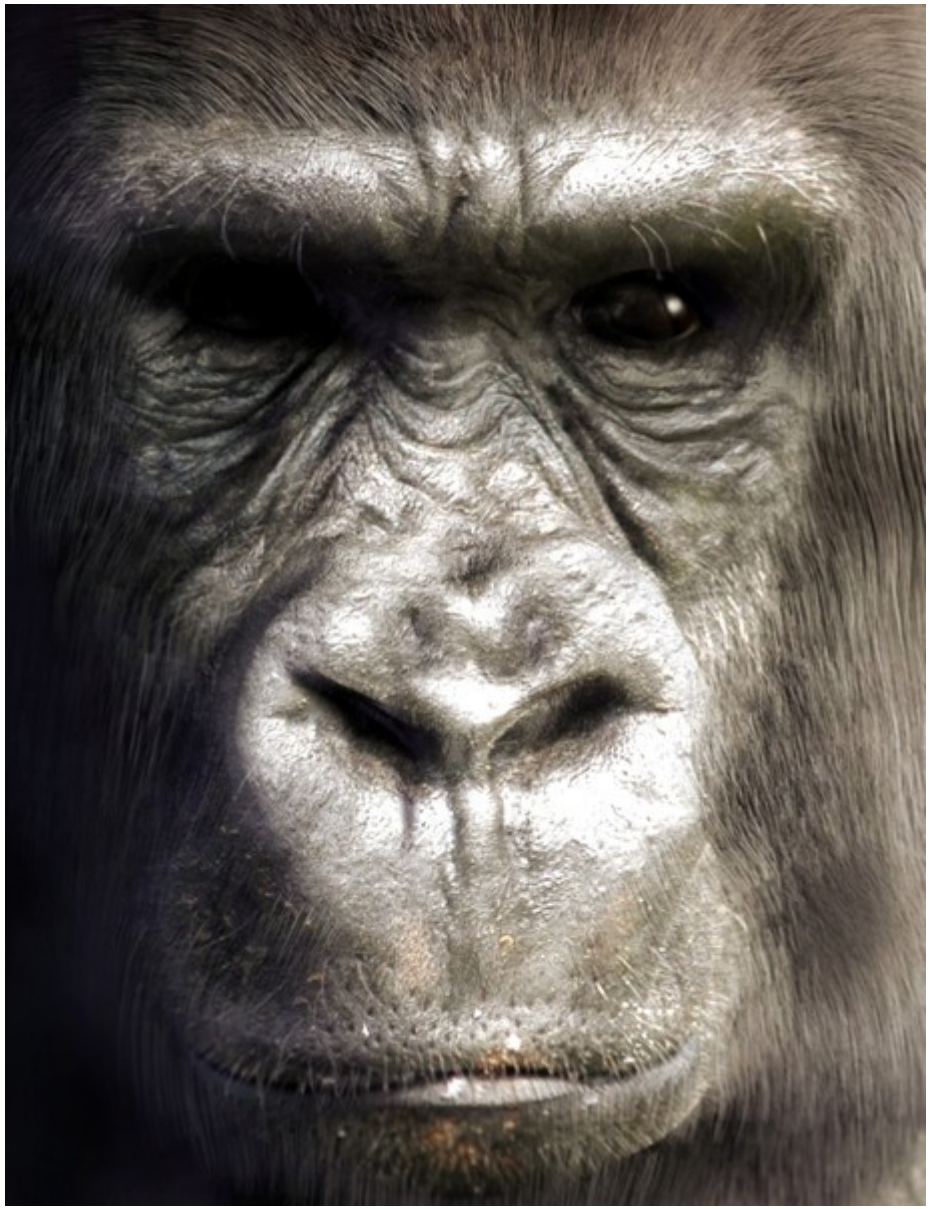
Exemplos de OpenGL

http://en.wikipedia.org/wiki/Quake_4

Exemplos de OpenGL

http://www.aerofly.de/

Exemplos de OpenGL



http://www.blender.org/

Pipeline de Software

Uso da API do OpenGL

Implementação da API do OpenGL

Comunicação com o hardware OpenGL

Executa as tarefas de desenho e transformações geométricas

Arquitetura

Arquitetura

- Possui primitivas e operações para a geração e manipulação de dados vetoriais e matriciais.
- Capaz de gerar imagens de alta qualidade.
- Comumente implementado de forma a tirar partido da aceleração gráfica (se disponível).
- Independente de plataforma.

Arquitetura

- Não gerencia janelas nem trata eventos produzidos por dispositivos de interação.
- Não possui comandos de alto nível para especificação de objetos 3D complexos.
- Objetos complexos devem ser construídos a partir de primitivas geométricas simples.

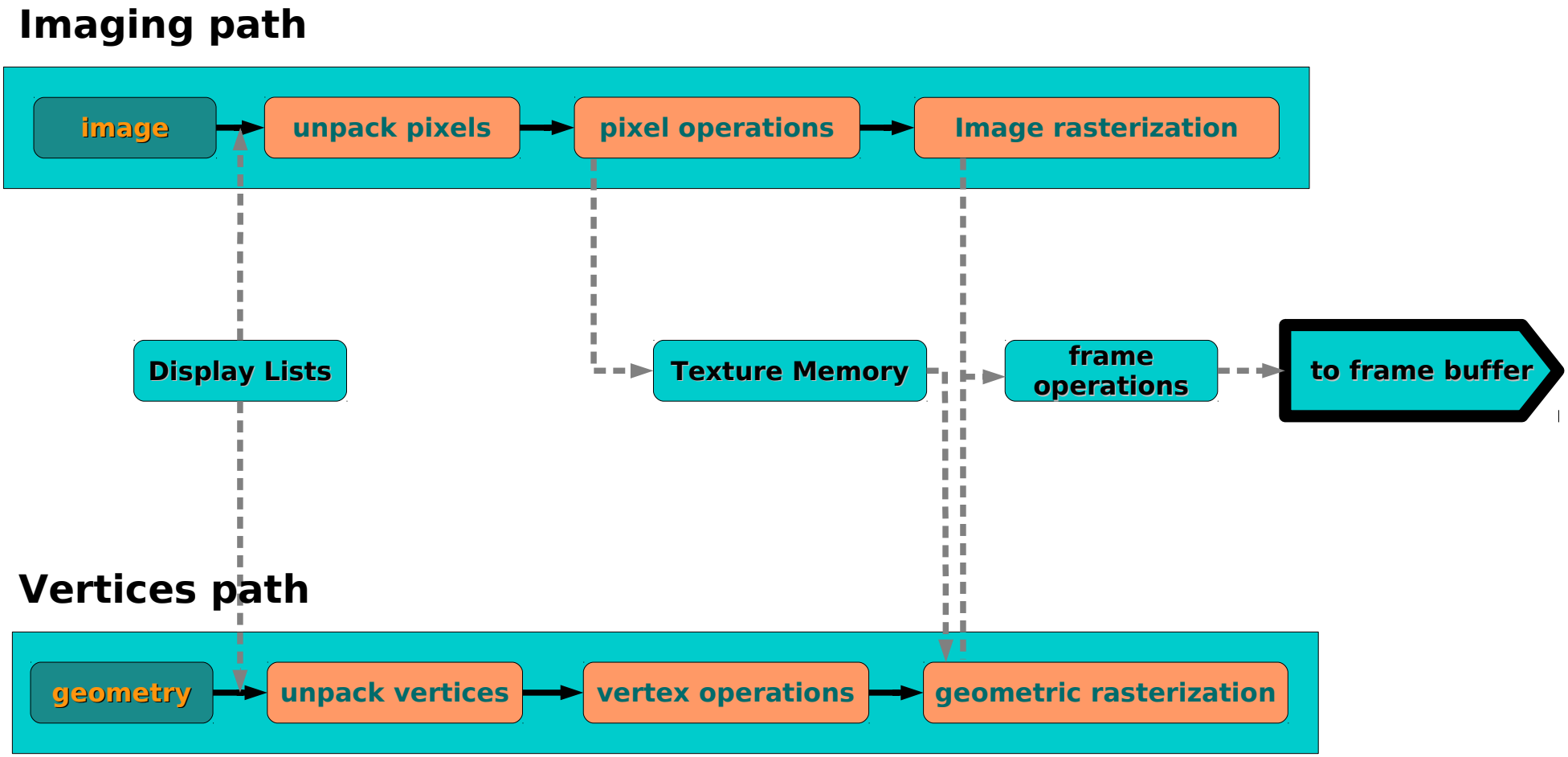
Arquitetura

- Cria descrições matemáticas de objetos a partir de primitivas geométricas (pontos, linhas e polígonos) e imagens/bitmaps.
- Organiza os objetos no espaço 3D e seleciona o ponto de vista adequado para a cena

Arquitetura

- Calcula as cores dos objetos por:
  - Atribuição direta.
  - Modelos de iluminação.
  - Mapeamentos de texturas.
  - Combinações.
- Converte as descrições matemáticas + cores em pixels (rasterização).

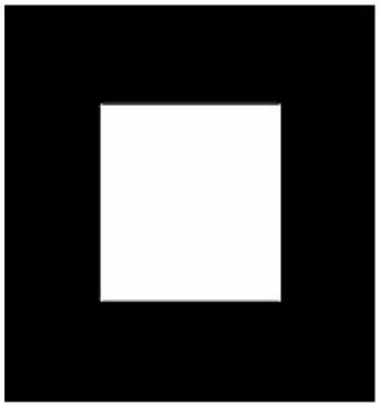
Arquitetura - Pipeline Programável de Visualização



Sintaxe OpenGL

Código OpenGL

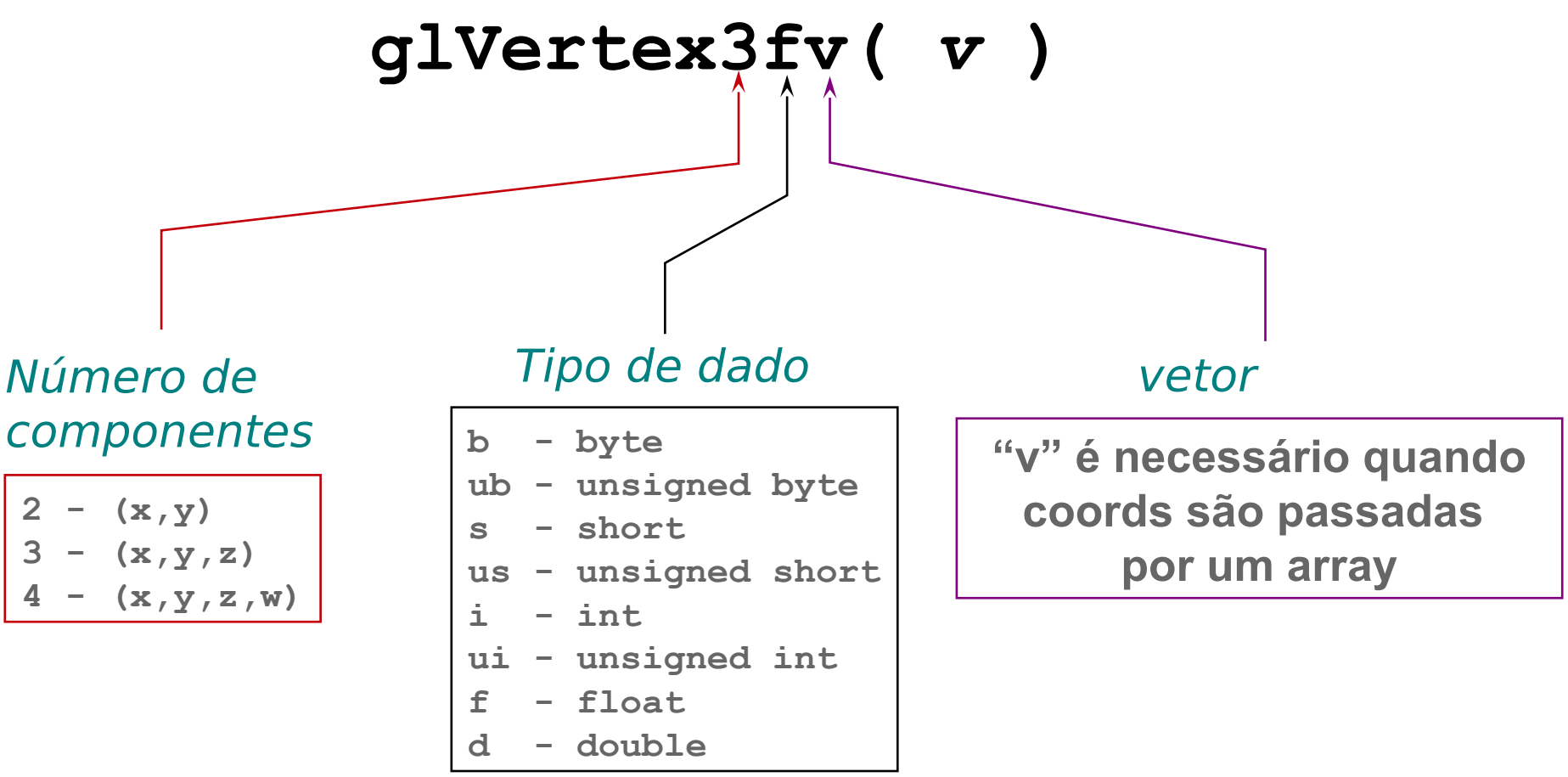
```
#include <whateverYouNeed.h>

main() {

   InitializeAWindowPlease();

   glClearColor (0.0, 0.0, 0.0, 0.0);
   glClear (GL_COLOR_BUFFER_BIT);
   glColor3f (1.0, 1.0, 1.0);
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
   glEnd();
   glFlush();

   UpdateTheWindowAndCheckForEvents();
}
```



Sintaxe das funções da API

- Todos as funções começam com o prefixo `gl` (Ex.: `glClearColor()`).
- Padrão lower camel case (Ex.: `glColor()`).
- O sufixo indica a quantidade e o tipo dos argumentos (Ex.: `glVertex2i(1,3)`).
- Constantes: `GL_COLOR_BUFFER_BIT`.

Sintaxe das funções da API



Sufixos e tipos dos argumentos

Tipo	Tipo	C	OpenGL
`b`	Inteiro 8-bits	signed char	GLbyte
`s`	Inteiro 16-bits	short	GLshort
`i`	Inteiro 32-bits	long	GLint, GLsizei
`f`	Ponto-flutuante 32-bit	float	GLfloat, GLclampf
`d`	Ponto-flutuante 64-bit	double	GLdouble, GLclampd
`ub`	Caractere s/ sinal 8-bit	unsigned char	GLubyte, GLboolean
`us`	Caractere s/ sinal 16-bit	unsigned short	GLushort
`ui`	Caractere s /sinal 32-bit	unsigned long	GLuint, GLenum, GLbitfield

Primitivas de Desenho OpenGL

OpenGL- Primitivas de desenho

- glBegin (PRIMITIVA);
  - especificação de vértices, cores, coordenadas de textura, propriedades de material
- glEnd ();
- Entre glBegin() e glEnd() apenas alguns comandos podem ser usados. Ex.:
  - glMaterial
  - glNormal
  - glTexCoord

OpenGL - Primitivas de desenho

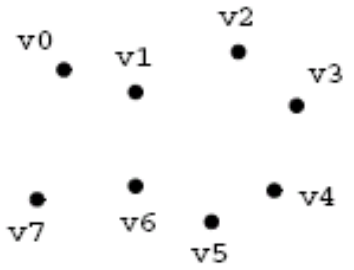
- Uma vez emitido um vértice (glVertex), este é desenhado com as propriedades (cor, material, normal, coordenadas de textura etc) registradas nas variáveis de estado correspondentes.

  - Atenção: Antes de emitir um vértice, assegurar-se que cor, material, normal, etc têm o valor certo.

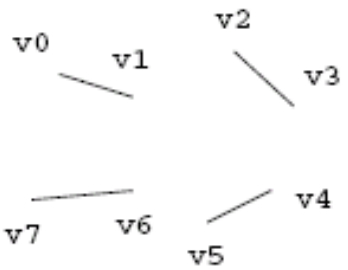
Primitivas de Desenho

Valor	Significado
`GL_POINTS`	Pontos individuais
`GL_LINES`	Pares de vértices interpretados como segmentos de reta individuais.
`GL_LINE_STRIP`	Serie de segmentos de reta conectados.
`GL_LINE_LOOP`	Igual ao anterior. Ultimo vertice conectado a primeiro
`GL_TRIANGLES`	Triplas de vértices interpretados como triângulos.
`GL_TRIANGLE_STRIP`	Cadeia triângulos conectados.
`GL_TRIANGLE_FAN`	Leque de triângulos conectados.
`GL_QUADS`	Quadrupla de vértices interpretados como quadriláteros.
`GL_QUAD_STRIP`	Cadeia de quadriláteros conectados.
`GL_POLYGON`	Borda de um polígono convexo simples.

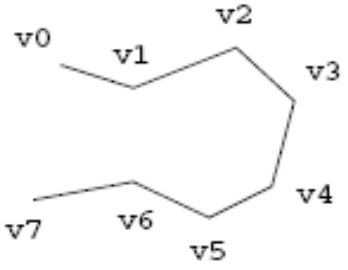
Primitivas de Desenho



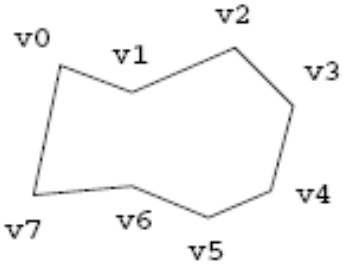
GL_POINTS



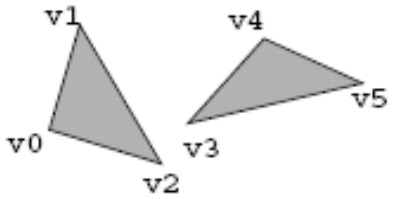
GL_LINES



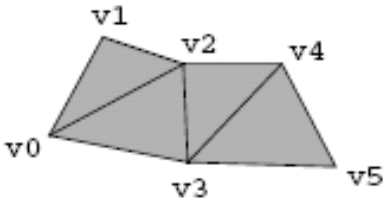
GL_LINE_STRIP



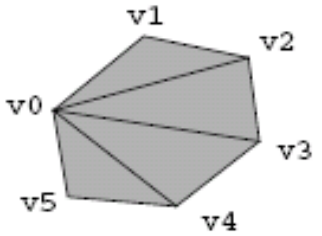
GL_LINE_LOOP



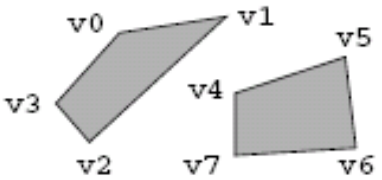
GL_TRIANGLES



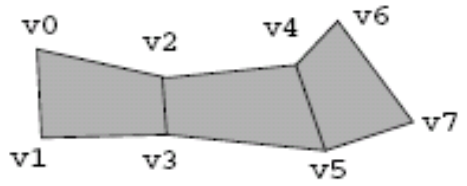
GL_TRIANGLE_STRIP



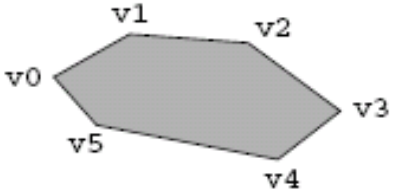
GL_TRIANGLE_FAN



GL_QUADS



GL_QUAD_STRIP



GL_POLYGON

OpenGL como máquina de estados

OpenGL como máquina de estados

- Uma aplicação OpenGL funciona como uma máquina de estados.
- Os estados correntes permanecem ativos.
- Exemplo:
  - A cor de desenho é aplicada a qualquer primitiva geométrica até que a cor corrente seja modificada.

OpenGL como maquina de estados

- Existem várias variáveis de estados, por exemplo:
  - cor de desenho corrente
  - transformações de visualização e projeção
  - padrões de linhas e polígonos
  - modo de desenho dos polígonos
  - atributos das fontes de luz
  - propriedades de reflexão e textura dos materiais associados aos objetos

OpenGL como maquina de estados

- Vários estados se referem a modos que estão habilitados ou desabilitados.
- Estes estados são modificados através dos comandos glEnable() e glDisable().
- Exemplo: glEnable(GL_LIGHTINING).

OpenGL como maquina de estados

- Alguns comandos para ler um estado:
  - glGetBooleanv(), glGetDoublev(), glGetFloatv(), glGetIntegerv(), glPointerv() ou glIsEnabled().
- Comandos para salvar um estado:
  - glPushAttrib() e glPushClientAttrib().
- Comandos para restaurar um estado:
  - glPopAttrib() e glPopClientAttrib().

APIs complementares

API's relacionadas

- GLU (OpenGL Utility Library)
  - Parte do padrão OpenGL
  - NURBS, trianguladores, quádricas, mapeamento, mipmaps, superfícies quadráticas, transformação e posicionamento da câmera e primitivas de desenho adicionais
- AGL, GLX, WGL
  - Camadas entre o OpenGL os diversos sistemas de janelas
- GLUT (OpenGL Utility Toolkit)
  - API portátil de acesso aos sistemas de janelas
  - Encapsula e esconde as camadas proprietárias
  - Não é parte oficial do OpenGL
  - http://www.opengl.org/resources/libraries/glut/spec3/spec3.html

GLUT - *OpenGL Utility Toolkit*

- API permite a implementação de aplicativos simples com uso do OpenGL
- Independente do sistema de janelas
- Útil para fins didáticos
- Permite a implementação de aplicativos multiplataforma
- Projetada para o desenvolvimento de programas de porte pequeno e médio
- *Bindings* para diversas linguagens de programação
- É um software gratuito e não *open source*
  - *freeglut* (Licença MIT)
- Fornece um conjunto de primitivas para desenho de objetos mais complexos como quádricas e etc

GLUT Callbacks

GLUT - *Callbacks*

- *Callbacks* são trechos de código (rotinas) chamadas para tratar eventos síncronos ou assíncronos
- Na linguagem C são implementadas a partir de ponteiros para funções, instrumento que permite que funções sirvam de argumento para outras. Exemplo:

```
<type> callerFunc(<type> (*argFunc)(<argList>))
{
    /* Código de callerFunc */
}
```

- Nas chamadas à `callerFunc` precisa ser fornecida a função `argFunc`
- A assinatura da função passada como parâmetro precisa ser idêntica à da função `argFunc`

GLUT - *Callbacks*

- As *callbacks* do GLUT são monitoradas e chamadas a partir do `GlutMainLoop`
- A API do GLUT possui diversas *callbacks* predefinidas
- Para uma rotina *callback* ser ativada ela precisa ser registrada através da função de registro
  - `glut<Callbackname>Func ((<type> (*callBackFunc)(<argList>)))`
    - `<Callbackname>` revela a classe do evento
    - `callBackFunc` é o nome da rotina de Callback
- Exemplo, função de registro da *callback* de desenho:

```
void glutDisplayFunc(void (*func)(void));
```

GLUT - Callback de desenho

```
glutDisplayFunc (void(*func)(void));
```

- Chamada automaticamente sempre que a janela ou parte dela precisa ser redesenhada
- Todo programa OpenGL/GLUT precisa ter uma
- Exemplo:

```
void display ( void )
{
  glClear( GL_COLOR_BUFFER_BIT );
  glBegin( GL_TRIANGLE_STRIP );
    glVertex3fv( v[0] );
    glVertex3fv( v[1] );
    glVertex3fv( v[2] );
    glVertex3fv( v[3] );
  glEnd();
  glutSwapBuffers(); /* Usamos double-buffering! */
}
```

GLUT - Callback de redimensionamento

```
glutReshapeFunc ((void (*func)(int width,
int height));
```

- Chamada sempre que a janela é redimensionada
- `width` e `height` são a nova largura/altura da janela (em pixels)
- Os valores fornecidos servem para o recálculo do *frustrum*

- Há uma rotina *default* que simplesmente ajusta o *viewport* para usar toda a área da janela gráfica

GLUT - *Callback* de Teclado

```
void glutKeyboardFunc(void (*func)
(unsigned char key, int x, int y));
```

  - Chamada sempre que um caracter é emitido
  - `key` indica o caracter e `x` e `y` a posição relativa do mouse no momento do envio
  - glutGetModifiers pode ser chamado para determinar as teclas pressionadas

- Para desabilitar esta *callback* é preciso passar `NULL` para `glutKeyboardFunc`
- Não há *callback* de teclado *default*

GLUT - *Callback* de Temporização

```
void glutTimerFunc(unsigned int msecs, void
(*func)(int value), value);
```

- Registra uma *callback* de temporização
- Após `msecs` milissegundos, a chamada para a *callback* se dará assim que for possível
- `value` tem o mesmo valor em (*func) e `glutTimerFunc`
- Podem ser registradas múltiplas *callbacks* de temporização.
- Uma vez registrada, não é possível cancelar uma *callback* de temporização. Contudo, pode-se ignorá-la em função de `value`

GLUT - Outras Callbacks

- Eventos de mouse
  - void mouse(int button, int state,int x,int y)
  - void motion(int x, int y)
  - void passiveMotion(int x, int y)
- Chamada continuamente quando nenhum outro evento ocorre
  - void idle(void)

Inicialização GLUT

Programa OpenGL/GLUT - Inicialização

- Inicialização do GLUT
  - glutInit (int* argc, char** argv)
- Estabelece contato com sistema de janelas.
  - Em X, opções de linha de comando são processadas e removidas.

Programa OpenGL/GLUT - Inicialização

- Inicialização da(s) janela(s) - glutInitDisplayMode (int modo)
- Estabelece o tipo de recursos necessários para as janelas que serão criadas.
  - Modo é um "ou" bit-a-bit de constantes:
    - GLUT_RGB cores dos pixels serão expressos em RGB.
    - GLUT_DOUBLE bufferização dupla (ao invés de simples).
    - GLUT_DEPTH buffer de profundidade (z-buffer).
    - GLUT_ACCUM buffer de acumulação.
    - GLUT_ALPHA buffer de cores terá componente alfa.

Programa OpenGL/GLUT - Inicialização

- glutInitWindowPosition (int x, int y)
  - Estabelece a posição inicial do canto superior esquerdo da janela a ser criada.
- glutInitWindowSize (int width, height)
  - Estabelece o tamanho (em pixels) da janela a ser criada.

Programa OpenGL/GLUT - Inicialização

- Criação da(s) janela(s)
  - `int glutCreateWindow (char* nome)`
    - Cria uma nova janela primária (top-level)
    - Nome é tipicamente usado para rotular a janela
    - O número inteiro retornado é usado pelo GLUT para identificar a janela

Programa OpenGL/GLUT - Inicialização

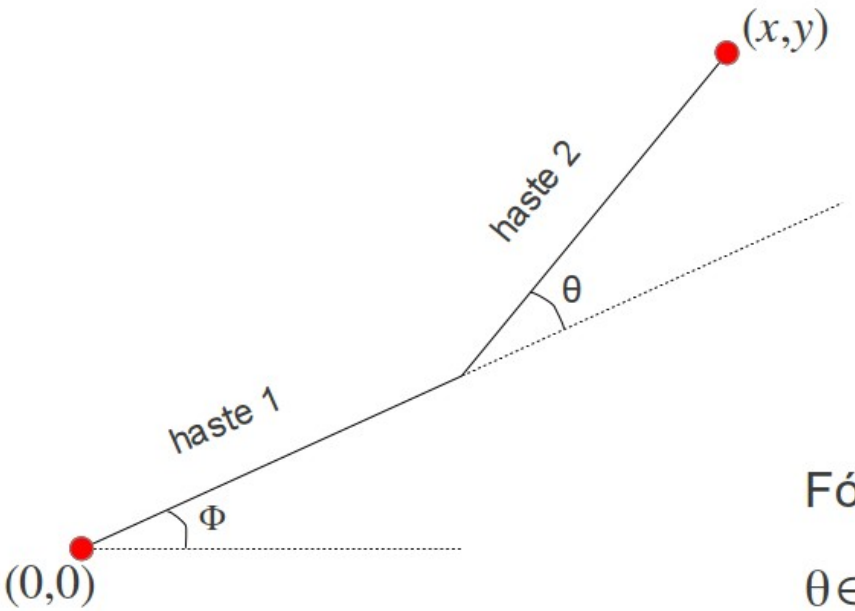
- Outras inicializações
- Após a criação da janela é costumeiro configurar variáveis de estado do OpenGL que não mudarão no decorrer do programa. Por exemplo:
  - Cor do fundo
  - Tipo de sombreamento de desejado

Programa OpenGL/GLUT - Laço principal

- Depois de registradas as callbacks, o controle é entregue ao sistema de janelas:
  - `glutMainDisplayLoop (void)`
- Esta rotina na verdade é o “despachante” de eventos.
- Ela nunca retorna.

Exemplo
GLUT + OpenGL

Exemplo GLUT + OpenGL



Fórmulas:

$$\theta \in [0°, 180°]$$

$$\Phi \in [0°, 360°]$$

$$\theta = 2 \cdot \arccos\left(\frac{\sqrt{(x^2+y^2)}}{2 \cdot l}\right)$$

$$\Phi = \arccos\left(\frac{x}{\sqrt{(x^2+y^2)}}\right) - \frac{\theta}{2}$$

Exemplo GLUT + OpenGL

```
// This program is free software; you can redistribute it
// and/or modif it under the terms of the GNU General Public
// License as published by the Free Software Foundation;
// either version 2 of the License, or (at your option) any
// later version.
//
/* Exemplo Braco Mecanico */
#include <GL/gl.h>
#include <GL/glu.h>
#include <GL/freeglut.h>
#include <stdlib.h>
#include <math.h>
#include <stdio.h>
#include <time.h>

#define PI 3.141592654
#define tAnima 5.0
#define deltaT 33

float theta, phi, theta0, phi0, theta1, phi1, l, TempoDecorrido;
```

Exemplo GLUT + OpenGL

```
int main(int argc, char ** argv)
{
   float x0, y0, x1, y1;
   /* A inicialização foi omitida */
   theta = CalcTheta( x0, y0, l);
   phi = CalcPhi(x0,y0,theta);
   theta0 = CalcTheta( x0, y0, l);
   phi0 = CalcPhi(x0,y0,theta0);
   theta1 = CalcTheta( x1, y1, l);
   phi1 = CalcPhi(x1,y1,theta1);
   TempoDecorrido=0;

   glutInit(&argc, argv);
   glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE);
   glutInitWindowSize( 500,500 );
   glutCreateWindow("Default viewport");

   glutDisplayFunc(draw);
   glutKeyboardFunc(keyboard);
   glutTimerFunc(deltaT,Timer, 1);
   glutMainLoop();

   return 0;
}
```

Exemplo GLUT + OpenGL

```
void draw()
{
   glClearColor( 100, 100, 0, 0 );
   glClear ( GL_COLOR_BUFFER_BIT );
   glViewport(0,0,500,500);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluOrtho2D(-2.1*l,2.1*l,-2.1*l,2.1*l);
   glRotatef(180.0*phi/PI, 0.0, 0.0,1.0);
   /* haste 1 */
   glColor3f( 0, 0, 0 );
      glBegin(GL_LINES);
         glVertex2f(0.0,0.0);
         glVertex2f(l,0.0);
   glEnd();
   glTranslatef(l,0,0);
   glRotatef(180.0*theta/PI, 0.0, 0.0,1.0);
   /* haste 2 */
   glColor3f( 0, 0, 1 );
   glBegin(GL_LINES);
         glVertex2f(0.0,0.0);
         glVertex2f(l,0.0);
   glEnd();
   glutSwapBuffers();
}
```

Exemplo GLUT + OpenGL

```
void keyboard(unsigned char key,int x,int y)
{
   if(key==27) exit(0);
}
```

Exemplo GLUT + OpenGL

```
void Timer(int value)
{

   TempoDecorrido += deltaT/1000.0;
   if (TempoDecorrido > tAnima)
      TempoDecorrido =tAnima;

   phi=(phi0*(tAnima-TempoDecorrido)
  +phi1*TempoDecorrido)/tAnima;
   theta = (theta0*(tAnima-
  TempoDecorrido)
  +theta1*TempoDecorrido)/tAnima;
   glutPostRedisplay();
   if (TempoDecorrido<tAnima)
      glutTimerFunc(deltaT,Timer, 1);
}
```

Fim