# Introdução à Computação Gráfica
# OpenGL Básico

Claudio Esperança
Paulo Roma Cavalcanti

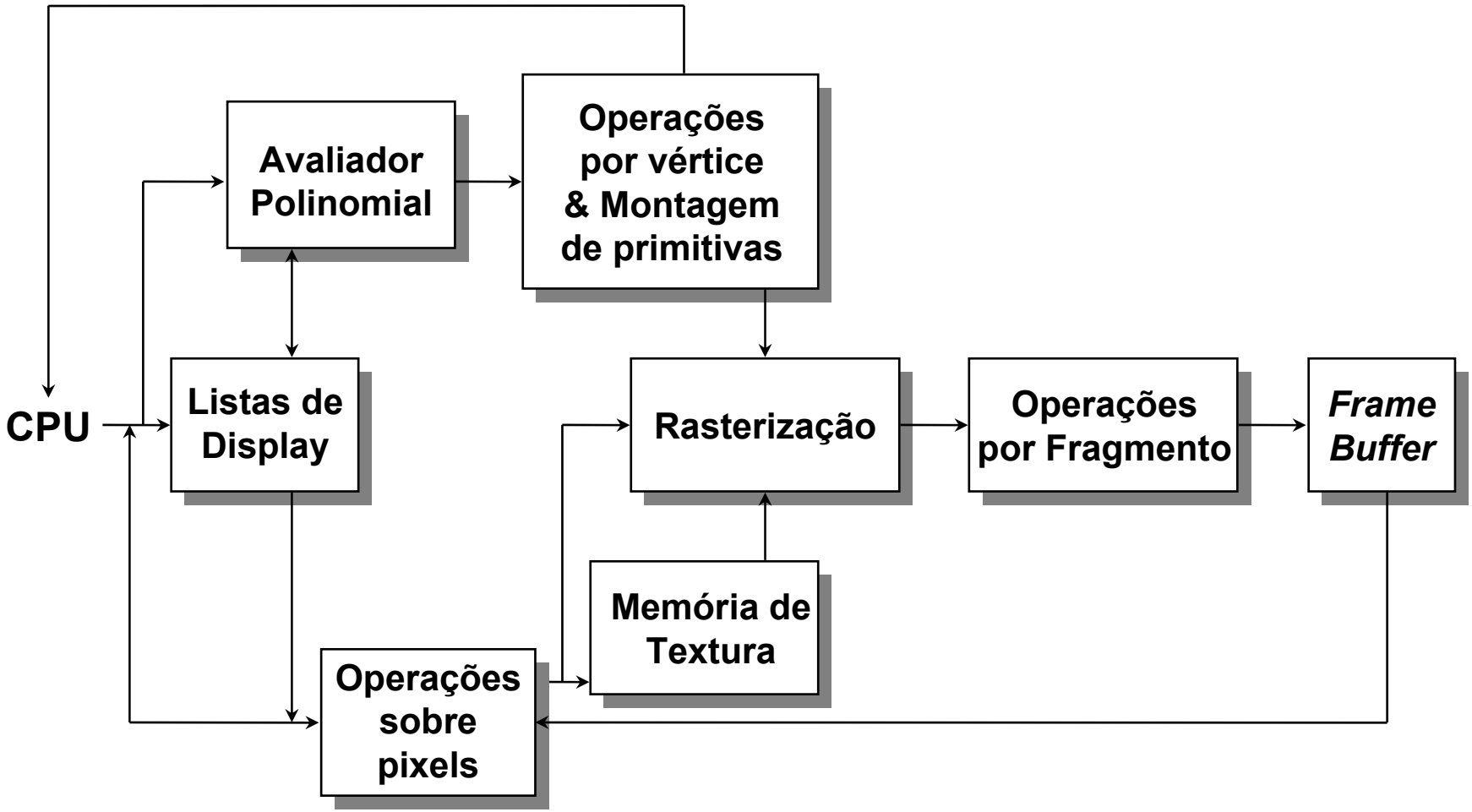
# OpenGL – O que é?

- Uma API para geração de gráficos
  - 3D e 2D
  - Primitivas vetoriais e matriciais (imagens)
  - Capaz de gerar imagens de alta qualidade
  - Comumente implementado de forma a tirar partido da aceleração gráfica (se disponível)
  - Independente de plataforma
  - Independente de sistema de janelas

# Sistemas de Janela

- Principal meio de interação homem/máquina em ambientes de computação modernos
- Tela é dividida em janelas (eventualmente superpostas)
- Janelas são controladas por aplicações que têm a incumbência de mantê-las sempre atualizadas
- Interação do usuário e do próprio sistema de janelas são comunicados à aplicação através de *eventos*, ex.:
  - Mouse foi apertado
  - Janela foi redimensionada
- Eventos são tratados por rotinas *callback* da aplicação. Ex.:
  - Redesenhar o conteúdo da janela
  - Mover um objeto de um lado para outro da janela
- Cada Sistema de Janelas possui uma API distinta
  - MS Windows, X, Apple
  - Portabilidade: Camada de interface com diversos SJ mas com API única (ex.: GLUT)

# Arquitetura do OpenGL



# Desenhando com OpenGL

- OpenGL funciona como uma máquina de estados
- API tem rotinas para
  - Desenhar primitivas geométricas e imagens
  - Alterar variáveis de estado (ex.: cor, material, fontes de iluminação, etc)
  - Consultar variáveis de estado
- OpenGL é um padrão em evolução
  - Mecanismo padronizado de extensões
  - Novas versões são estabelecidas por um comitê (ARB) de usuários e fabricantes

# APIs Relacionadas

- GLU (OpenGL Utility Library)
  - Parte do padrão OpenGL
  - NURBS, trianguladores, quádricas, etc.
- 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 official do OpenGL

# Anatomia de um programa OpenGL/GLUT

```
#include <GL/glut.h>
/* Outros headers */

void display (void) {
   …
}
/* Outras rotinas callback */

int main (int argc, char *argv[]) {
   glutInit (argc, argv);
   glutInitDisplayMode( modo );
   glutCreateWindow( nome_da_janela );
   glutDisplayFunc( displayCallback );
   glutReshapeFunc( reshapeCallback );
   /* Registro de outras rotinas callback */
   glutMainLoop();
   return 0;
}
```

Headers

Rotinas Callback

Inicialização do GLUT

Inicialização da janela

Registro de callbacks

Laço principal

# Headers OpenGL/GLUT

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

- Já inclui automaticamente os headers do OpenGL:

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

- Se GLUT não for usado, os headers OpenGL têm que ser incluídos explicitamente, junto com os de outra camada de interface
- Há APIs para construção de interfaces gráficas (GUI) construídas sobre o GLUT cujos headers incluem os do GLUT
  - Por exemplo, o pacote GLUI requer:

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

    - (Já inclui `glut.h`)

# GLUT – Registrando Callbacks

- Callbacks são rotinas que serão chamadas para tratar eventos.
- Para uma rotina callback ser efetivamente chamada ela precisa ser registrada através da função

  `glutXxxFunc (`*`callback`*`)`

  - Onde `Xxx` designa uma classe de eventos e *`callback`* é o nome da rotina
- Por exemplo, para registrar uma callback de desenho chamada `Desenho`, usa-se

```
glutDisplayFunc (Desenho);
```

# GLUT – Callback de desenho

- É a rotina chamada automaticamente sempre que a janela ou parte dela precisa ser redesenhada (ex.: janela estava obscurecida por outra que foi fechada)
- Todo programa 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

- Chamada sempre que a janela é redimensionada, isto é, teve seu tamanho alterado
- Tem a forma

```
void reshape (int width, int height){...}
```

  - width/height são a nova largura/altura da janela (em pixels)
- Se uma rotina de redimensionamento não for especificada, o GLUT usa uma rotina de redimensionamento "default" que simplesmente ajusta o *viewport* para usar toda a área da janela

# GLUT - Callbacks

- Outras callbacks comumente usadas

```
void keyboard (unsigned char key, int x, int
  y)
```

  - Eventos de teclado

```
void mouse(int button, int state, int x, int y)
void motion(int x, int y)
void passiveMotion(int x, int y)
```

  - Eventos de mouse

```
void idle (void)
```

  - Chamada continuamente quando nenhum outro evento ocorre

- Várias outras

# 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

  `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
- 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 (do livro vermelho)

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

void display(void)
{
  /* Limpar todos os pixels  */
  glClear (GL_COLOR_BUFFER_BIT);

  /* Desenhar um polígono branco (retângulo) */
  glColor3f (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();

  /* Não esperar! */
  glFlush ();
}
```

# Exemplo (do livro vermelho)

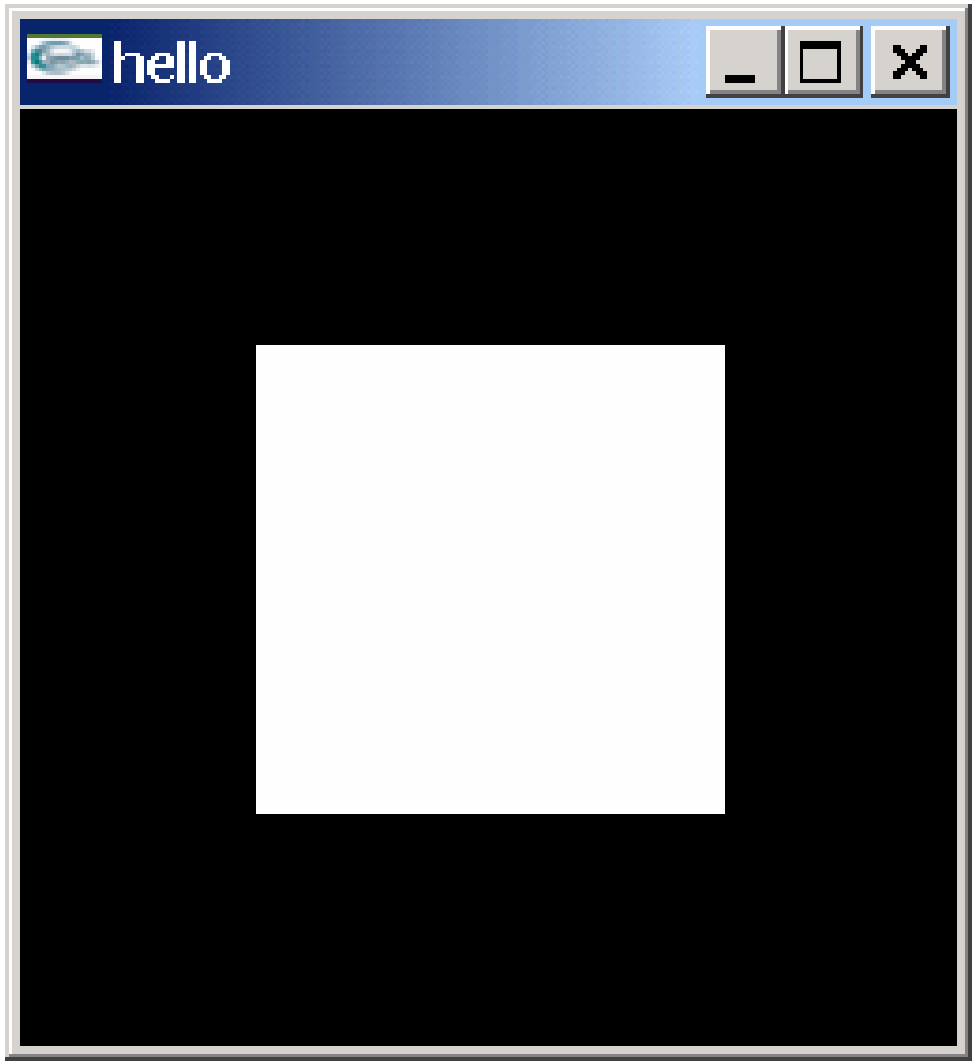
```
void init (void)
{
  /* selecionar cor de fundo (preto) */
  glClearColor (0.0, 0.0, 0.0, 0.0);

  /* inicializar sistema de viz. */
  glMatrixMode(GL_PROJECTION);
  glLoadIdentity();
  glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

```
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode
      (GLUT_SINGLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
   glutCreateWindow ("hello");
   init ();
   glutDisplayFunc(display);
   glutMainLoop();

  /* C ANSI requer que main retorne um
     inteiro */
   return 0;
}
```

# Resultado do Exemplo



# OpenGL – Primitivas de desenho

```
glBegin ( PRIMITIVA );
```

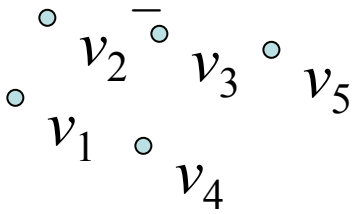
*especificação de vértices, cores, coordenadas de textura, propriedades de material*

```
glEnd ();
```

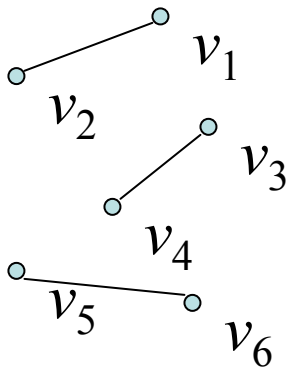
- Entre `glBegin()` e `glEnd()` apenas alguns comandos podem ser usados. Ex.:
  - `glMaterial`
  - `glNormal`
  - `glTexCoord`
- Uma vez emitido um vértice (`glVertex`), este é desenhado com as propriedades (cor, material, normal, coordenadas de textura etc) registradas nas variáveis de estado correspondentes
- Conclusão: Antes de emitir um vértice, assegurar-se que cor, material, normal, etc têm o valor certo

# OpenGL – Primitivas de desenho

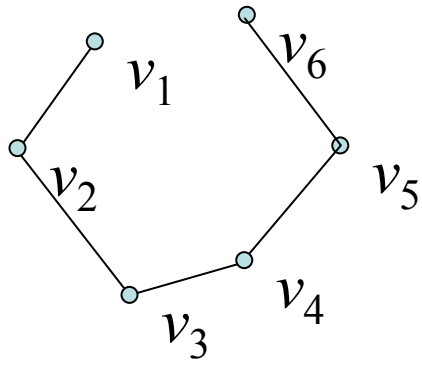
GL_POINTS



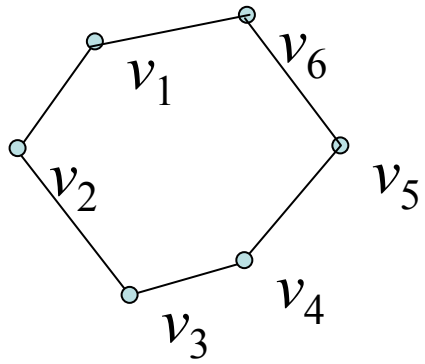

GL_LINES



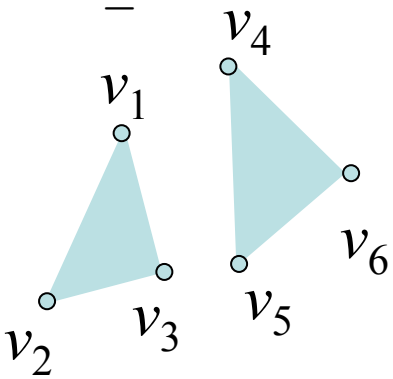

GL_LINE_STRIP



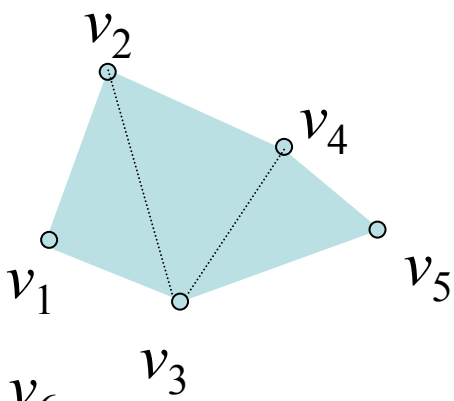

GL_LINE_LOOP



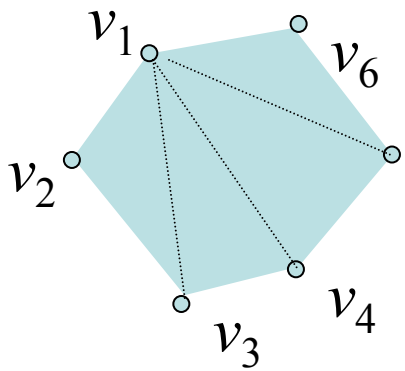

GL_TRIANGLES



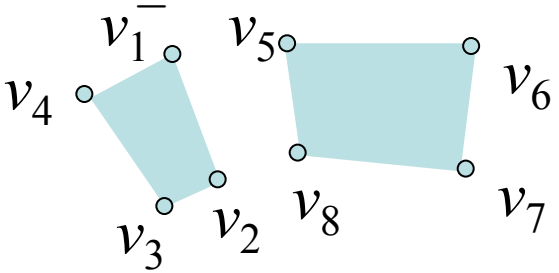

GL_TRIANGLE_STRIP



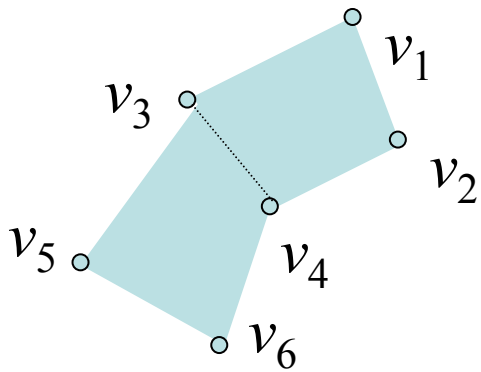

GL_POLYGON
GL_TRIANGLE_FAN




GL_QUADS



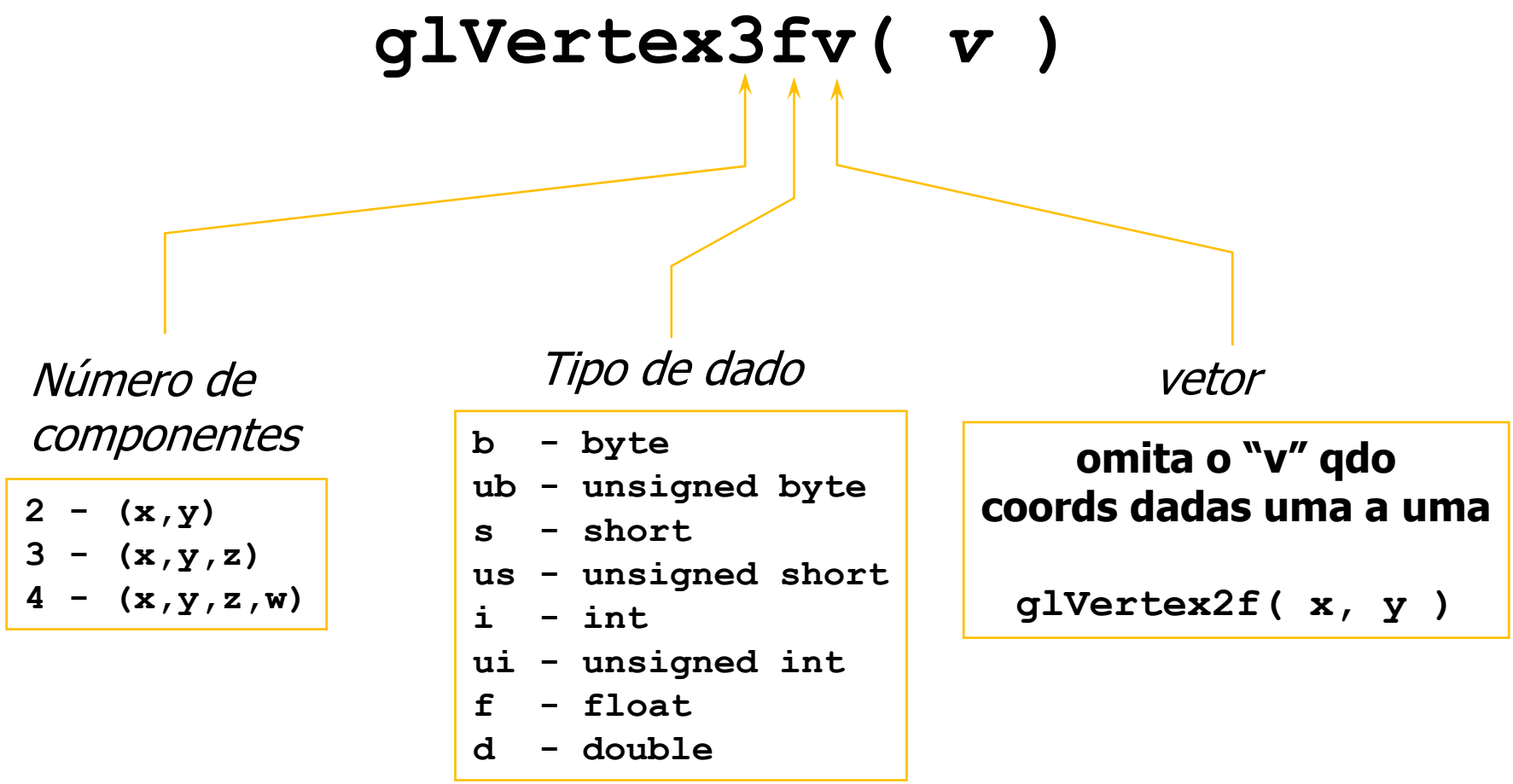

GL_QUAD_STRIP



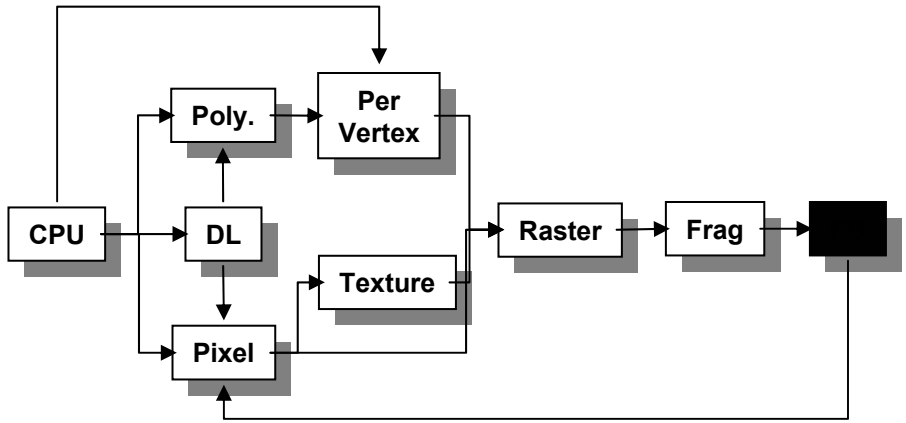
# OpenGL – Exemplo de desenho simples

```
void drawRhombus( GLfloat color[] )
{
 glBegin( GL_QUADS );
 glColor3fv( color );
 glVertex2f( 0.0, 0.0 );
 glVertex2f( 1.0, 0.0 );
 glVertex2f( 1.5, 1.118 );
 glVertex2f( 0.5, 1.118 );
 glEnd();
}
```

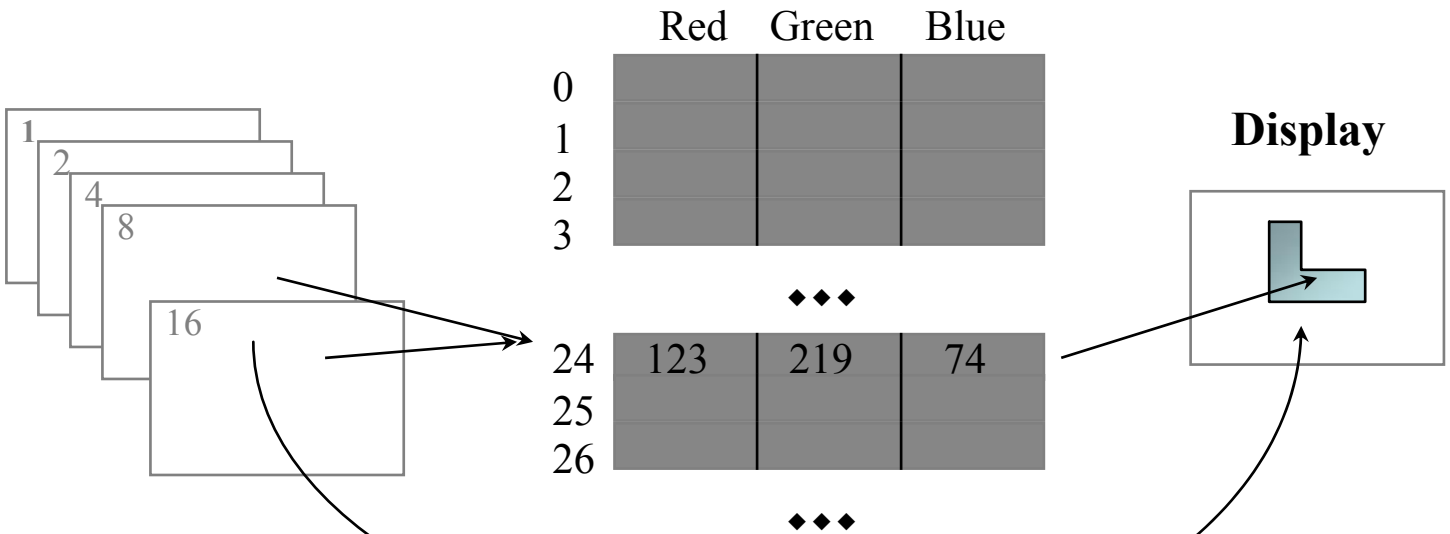
# OpenGL – Convenções de Nome



# OpenGL - Especificando Cores




*color index mode*

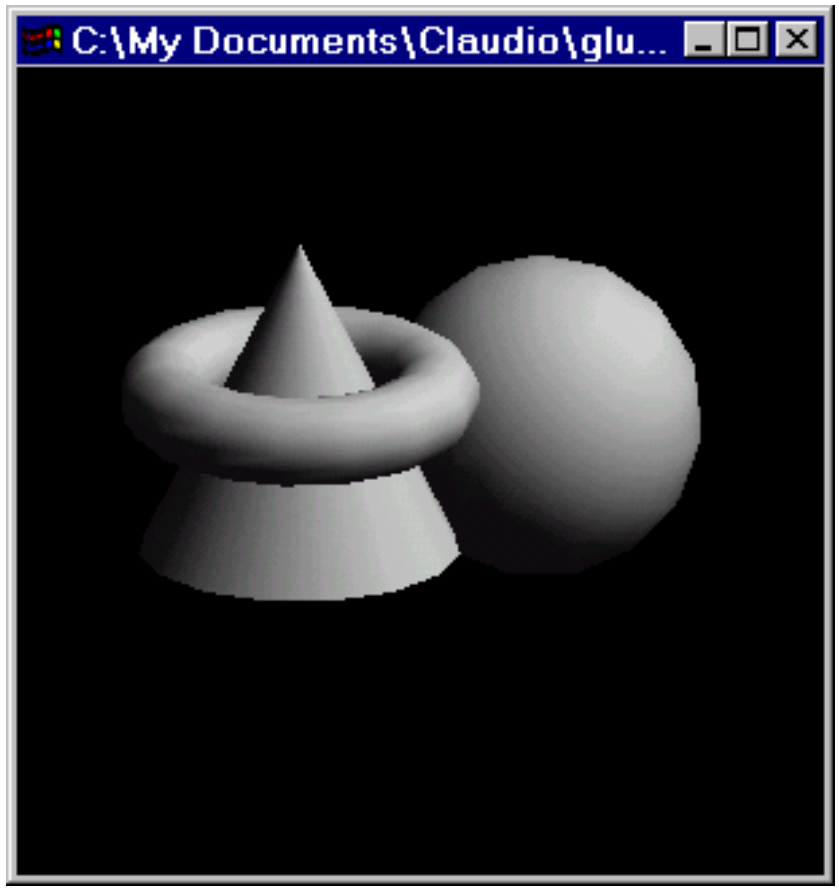



*RGBA mode*

# OpenGL – Controlando as cores

- Cores especificadas diretamente (default)
  - Usar `glColorIndex()` ou `glColor()`
- Computadas a partir de um modelo de iluminação
  - Ligar a iluminação: `glEnable (GL_LIGHTING);`
  - Escolher modelo de sombreamento:
    - Constante por face: `glShadeModel (GL_FLAT);`
    - Gouraud (default): `glShadeModel (GL_SMOOTH);`
  - Ligar ao menos uma fonte de luz. Ex: `glEnable(GL_LIGHT0);`
  - Especificar propriedades da(s) fonte(s) de luz: `glLight()`
  - Especificar propriedades de material de cada objeto: `glMaterial()`
  - Especificar normais de cada face ou de cada vértice: `glNormal()`

# OpenGL - Sombreamento constante por face e Gouraud

# OpenGL – Exemplo de Inicialização

```
void myinit(void)
{
    GLfloat light_ambient[] = { 0.0, 0.0, 0.0, 1.0 };
    GLfloat light_diffuse[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };

    glLightfv(GL_LIGHT0, GL_AMBIENT, light_ambient);
    glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
    glLightfv(GL_LIGHT0, GL_SPECULAR, light_specular);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);

    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
    glShadeModel (GL_SMOOTH);
}
```