OpenGL

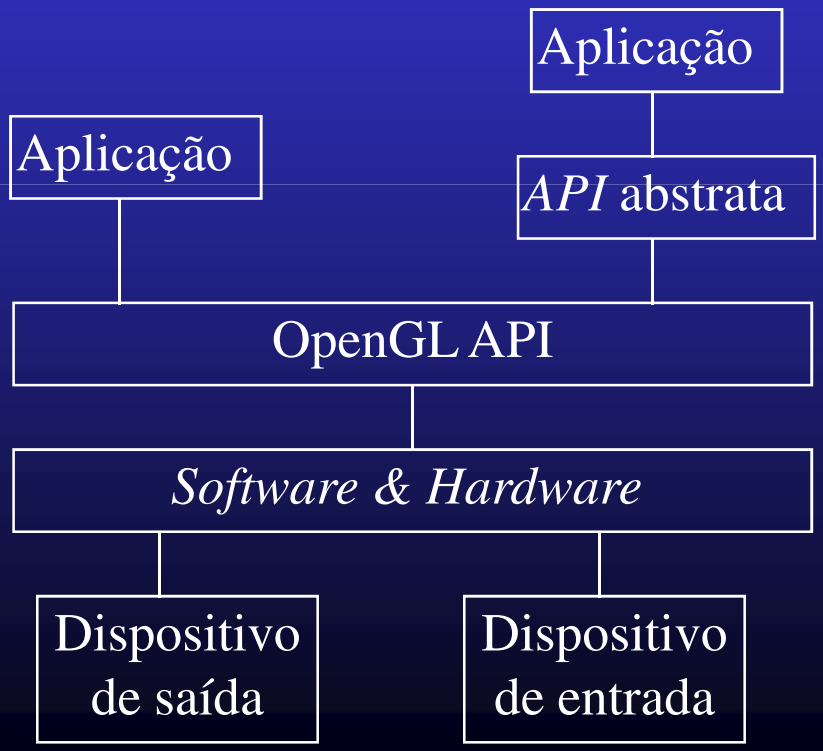
Conceitos Básicos

Waldemar Celes

Tecgraf/PUC-Rio

OpenGL: o que é?

- API
  - Interface para programador de aplicação



Por que OpenGL?

- primitivas geométricas e imagens
- arquitetura bem definida
- relativamente simples
- boa performance (sw & hw)
- bem documentado
- independente de sistemas de janelas
- padrão
  - disponível em diversas plataformas

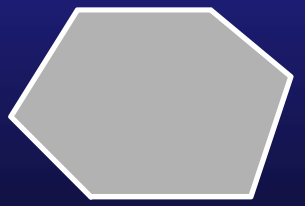
Primitivas geométricas básicas

Ponto

Linha

Triângulo

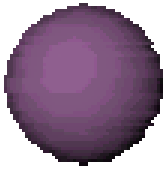
Quadrado



Polígono (convexo)

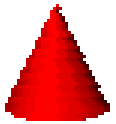
Objetos 3D

Polyhedra



Sphere

Bezier Surfaces

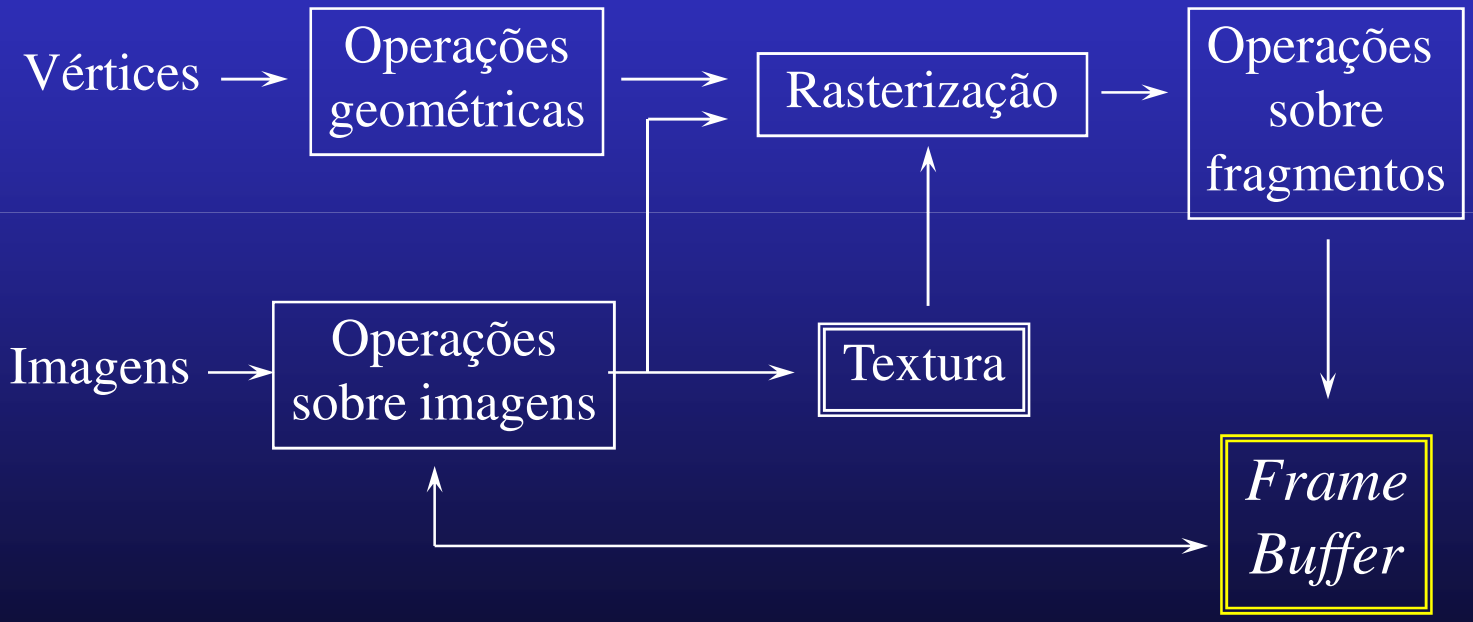


Quadric

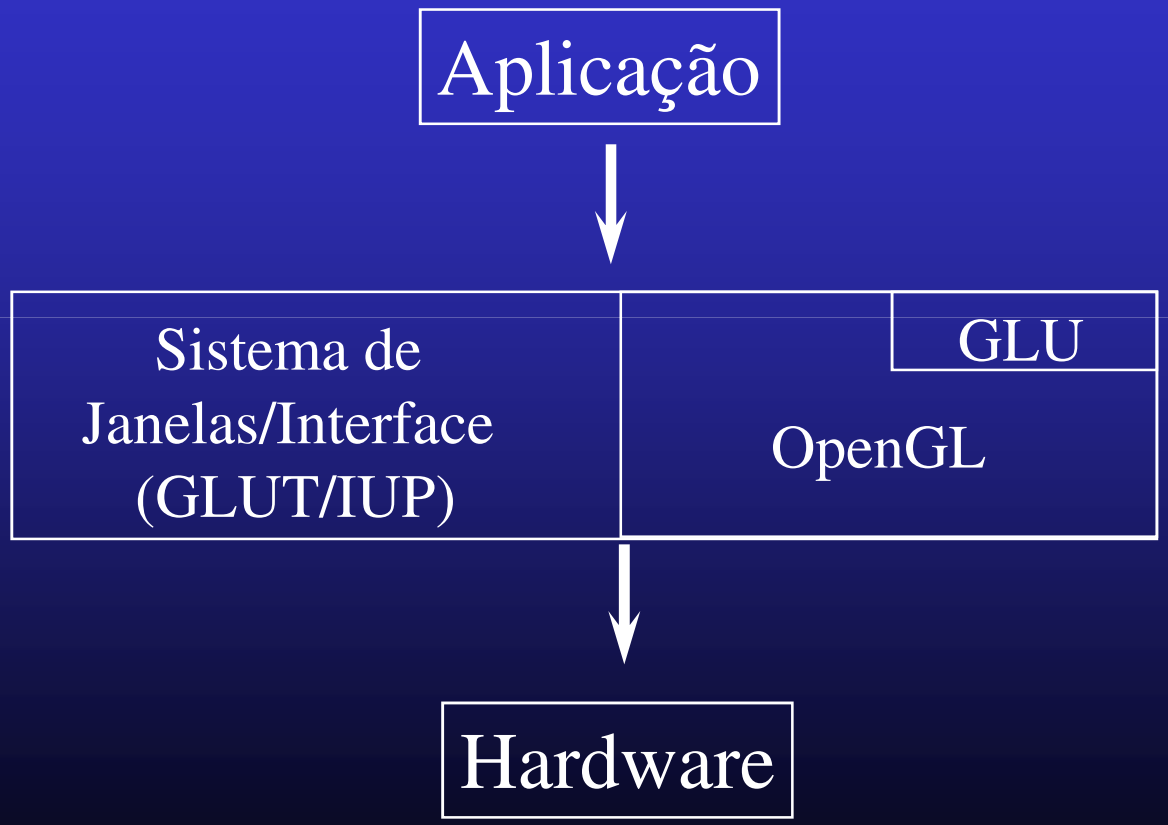
From SIGGRAPH'97 course

Imagem e Textura

OpenGL rendering pipeline



Aplicação típica



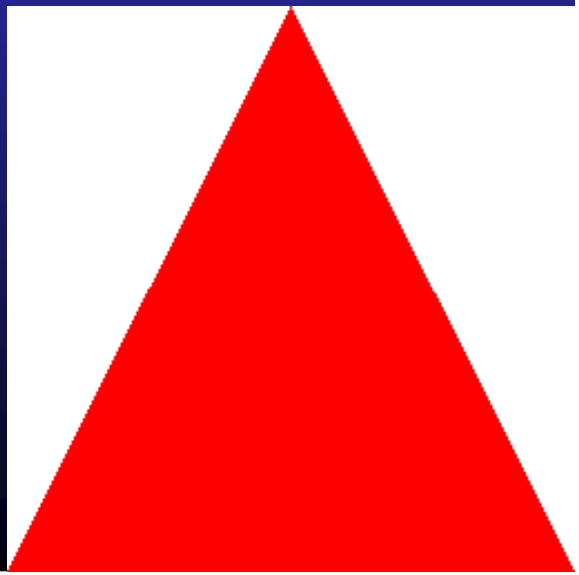
Programa simples (usando GLUT)

```
#ifdef _WIN32
#include <windows.h>
#endif
#include "GL/gl.h"
#include "GL/glu.h”
#include “GL/glut.h”
```

```
int main (int argc, char* argv[])
{
 /* openg GLUT */
 glutInit(&argc, argv);
 glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
 glutInitWindowSize (250, 250);

 /* create window */
 glutCreateWindow (”simple");
 glutDisplayFunc(display);

 /* interact … */
 glutMainLoop();
 return 0;
}
```

Programa simples (usando GLUT) - cont.

```
void display (void)
{
 /* clear window */
 glClearColor(1,1,1,1);
 glClear(GL_COLOR_BUFFER_BIT);

 /* draw red triangle */
 glColor3d(1,0,0);
 glBegin(GL_TRIANGLES);
  glVertex2d(-1,-1);
  glVertex2d(1,-1);
  glVertex2d(0,1);
 glEnd();

 /* update screen */
 glFlush();
}
```



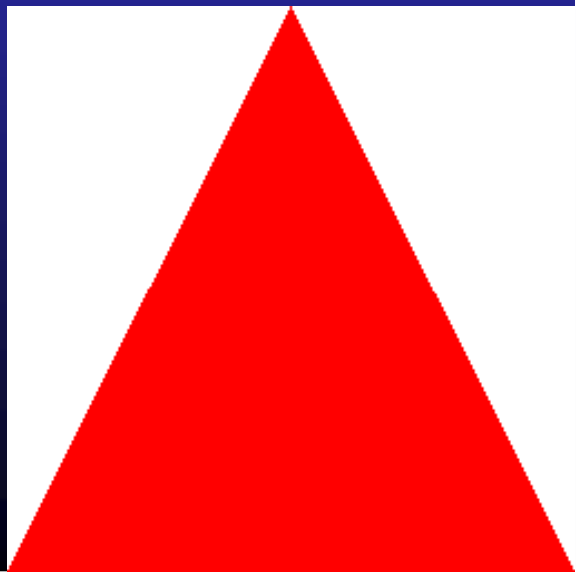
Programa simples (usando IUP)

```
#include "iup.h"
#include "iupgl.h"

#ifdef _WIN32
#include
<windows.h>
#endif
#include
"GL/gl.h"
#include "GL/glu.h"
```

```
int main (void)
{
 Ihandle  *dg, *cv;

 /* open GUI */
 IupOpen();
 IupGLCanvasOpen();

 /* create canvas and dialog */
 cv = IupGLCanvas( "redraw" );
 dg = IupDialog(cv);
 IupSetAttribute(dg,"SIZE","200x200");
 IupSetFunction("redraw",(Icallback)redraw);
 IupShow(dg);

 /* interact... */
 IupMainLoop();
 IupClose();
 return 0;
}
```

Programa simples (cont.)

```
int redraw (Ihandle *cv, double x, double y)
{
 IupGLMakeCurrent(cv);

 /* clear window */
 glClearColor(1,1,1,1);
 glClear(GL_COLOR_BUFFER_BIT);

 /* draw red triangle */
 glColor3d(1,0,0);
 glBegin(GL_TRIANGLES);
  glVertex2d(-1,-1);
  glVertex2d(1,-1);
  glVertex2d(0,1);
 glEnd();

 /* update screen */
 glFlush();
 return IUP_DEFAULT;
}
```



OpenGL & IUP

```
#include “iup.h”
#include “iupgl.h”
```

Inicialização

```
IupOpen();
IupGLCanvasOpen();
```

Bibliotecas:
- IUP
  - iup.lib & iupgl.lib
- OpenGL
  - opengl.lib & glu.lib (SGI)
  - opengl32.lib & glu32.lib (MS)

Criação em C

```
cv = IupGLCanvas( "redraw" );
```

Criação em LED

```
cv = GLCanvas(redraw)
```

Atributos

BUFFER = SINGLE ou DOUBLE
COLOR = RGBA ou INDEX
...

OpenGL: máquina de estado

- Trabalha com o conceito de valor corrente
  - Iluminação
  - Shading
  - Textura
  - etc.

glEnable/glDisable

Inicialização da área de desenho

```
glClearColor(red,green,blue,alpha);
glClear(GL_COLOR_BUFFER_BIT);
```

Atualização da área de desenho

```
glFlush( );
glFinish( );     // modal
```

Primitivas geométricas

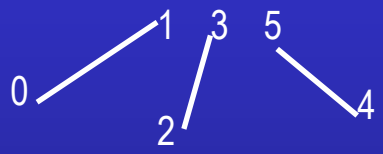
glBegin(*tipo_de_prim);*

…define attributo de vértice
…define vértice

glEnd();

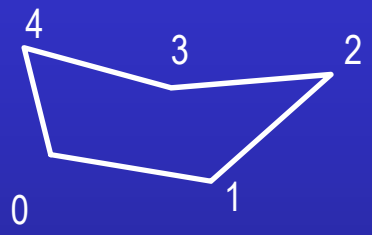
Tipos de primitivas

GL_POINTS

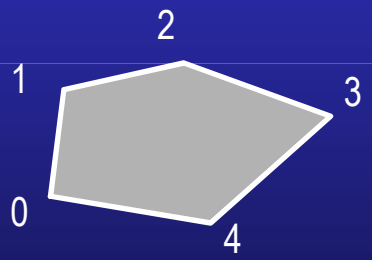


GL_LINES

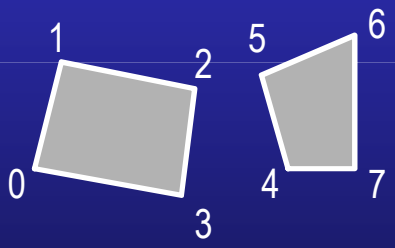
GL_LINE_STRIP



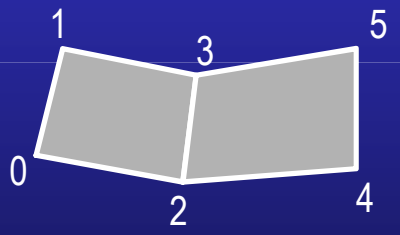
GL_LINE_LOOP



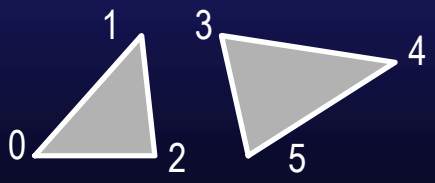
GL_POLYGON



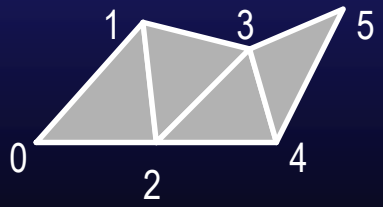
GL_QUADS



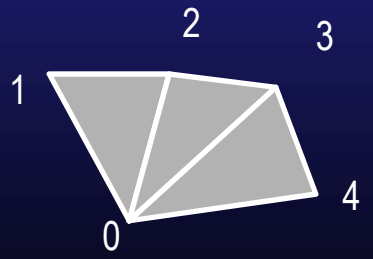
GL_QUAD_STRIP



GL_TRIANGLES



GL_TRIANGLE_STRIP



GL_TRIANGLE_FAN

Especificação de vértice

glVertex{tam}{tipo}{vetor} (…);

exemplo:

```
GLdouble pos[ ] = {0.4,9.0,2.0};
glVertex3dv(pos);
```

ou

```
glVertex3d(0.4,9.0,2.0);
```

- *OpenGL trabalha com coordenadas homegêneas*

Especificação de atributos: Cor

- Modelo de cor
  - RGB

```
glColor3d(red,green,blue);
```

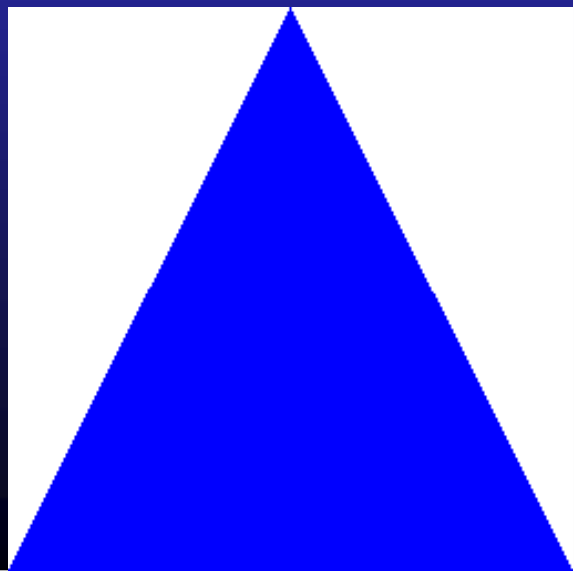
  - *Color index*
    - Paleta previamente definida

```
lupGLPalette (handle, index, red, green, blue);
...
glIndexi(index);
```

Modelo de Shading

- *Flat*

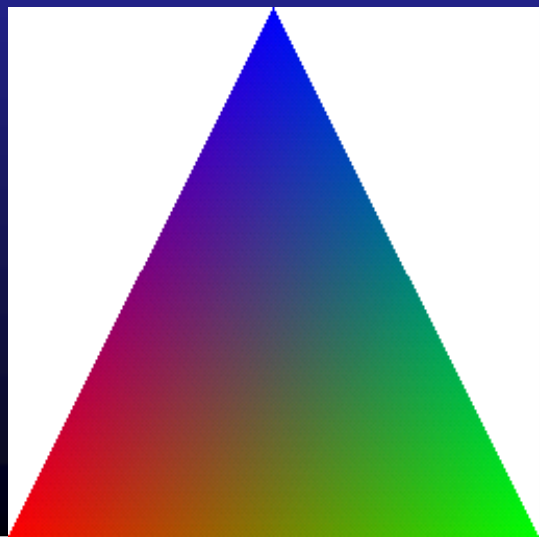
```
glShadeModel(GL_FLAT);
glBegin(GL_TRIANGLES);
   glColor3f(1.0,0.0,0.0);  // red
   glVertex2f(-1.0,-1.0);
   glColor3f(0.0,1.0,0.0); // green
   glVertex2f(1.0,-1.0);
   glColor3f(0.0,0.0,1.0); // blue
   glVertex2f(0.0,1.0);
glEnd( );
```



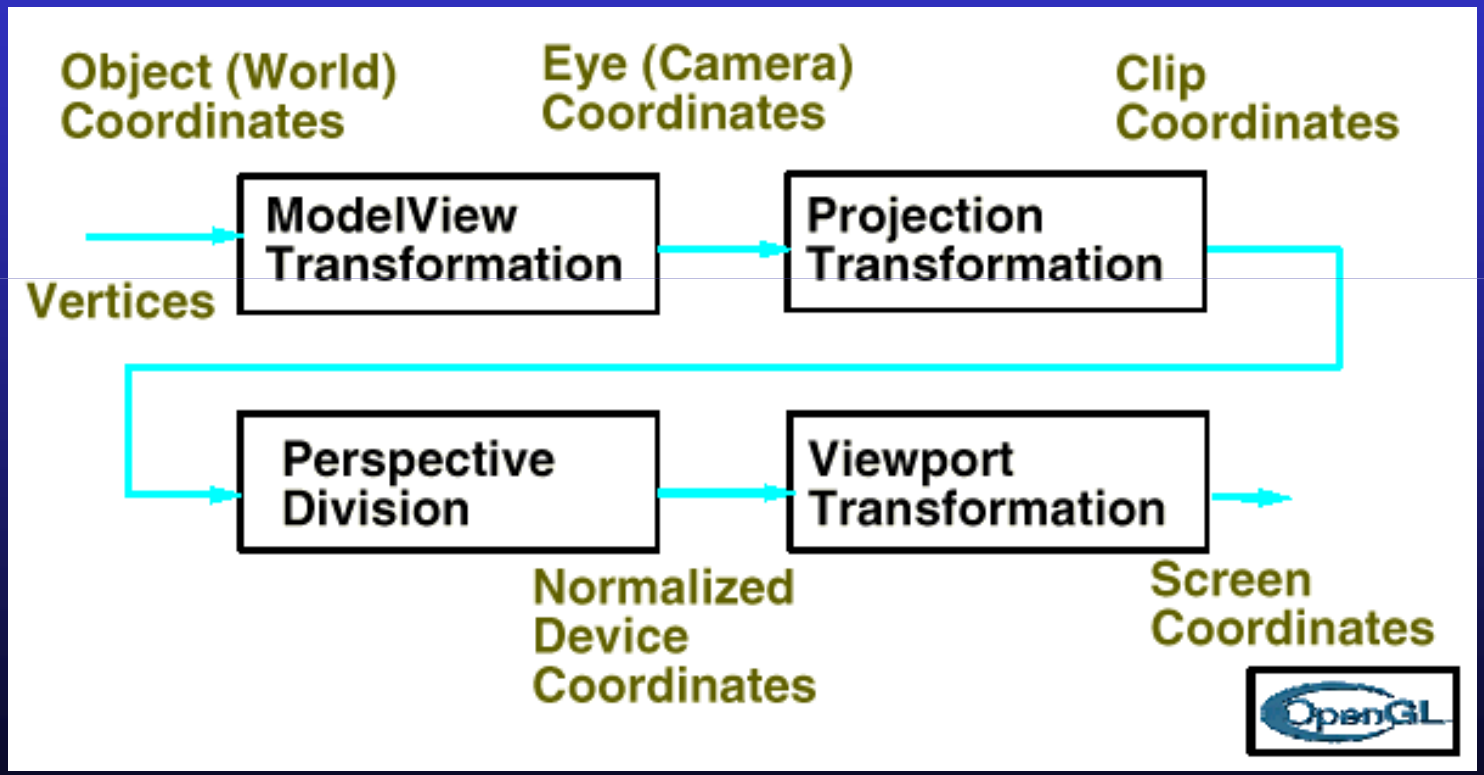
Modelo de *Shading*

- *Smooth (Gouraud)*

```
glShadeModel(GL_SMOOTH);   // default
glBegin(GL_TRIANGLES);
  glColor3f(1.0,0.0,0.0);  // red
  glVertex2f(-1.0,-1.0);
  glColor3f(0.0,1.0,0.0); // green
  glVertex2f(1.0,-1.0);
  glColor3f(0.0,0.0,1.0); // blue
  glVertex2f(0.0,1.0);
glEnd( );
```



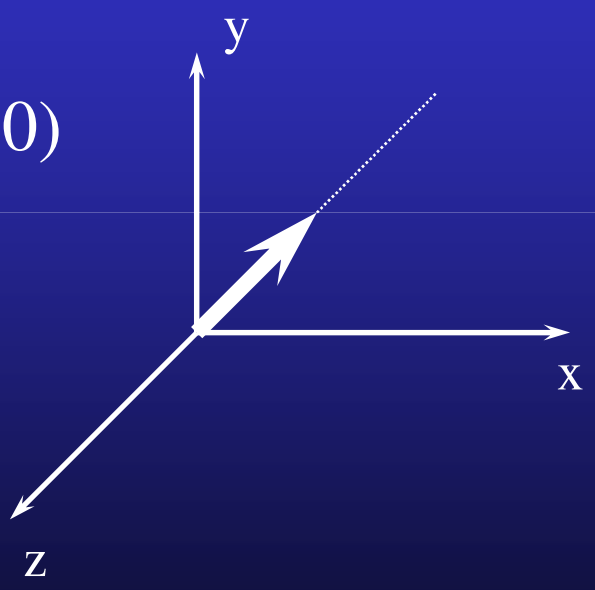
Transformações 3D e Sistemas de Coordenadas



From SIGGRAPH'97 course

Visualização 3D

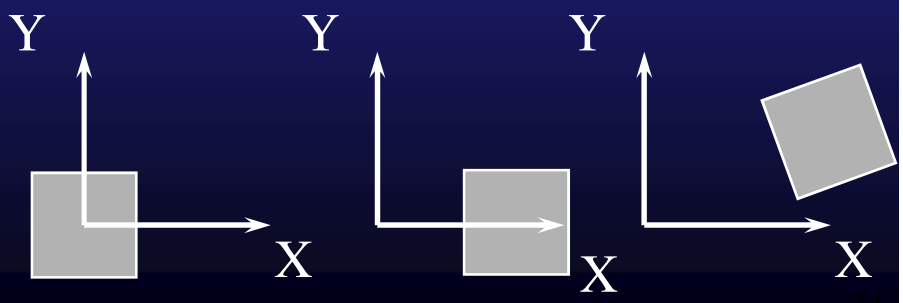
- Camera
  - Posição fixa: (0.0,0.0,0.0)
  - Direção: -z
- Composição da cena
  - move camera *ou*
  - move objetos



Transformação de Modelagem x Visualização

- Transformação de modelagem
  - Sistema global fixo
  - Ordem inversa para especificação

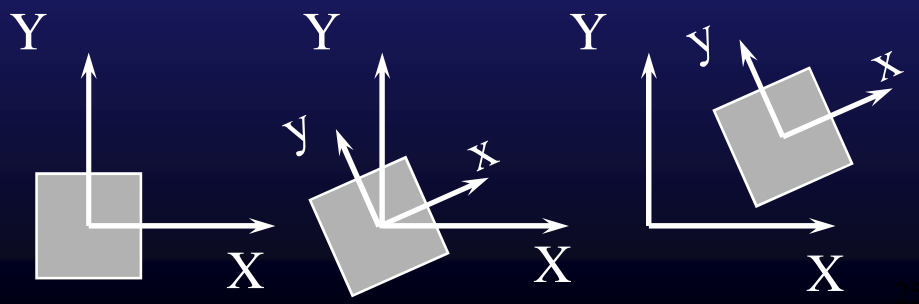
```
...
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
glRotatef(30,0,0,1);
glTranslatef(10,0,0);
...
```



Transformação de Modelagem x Visualização (cont.)

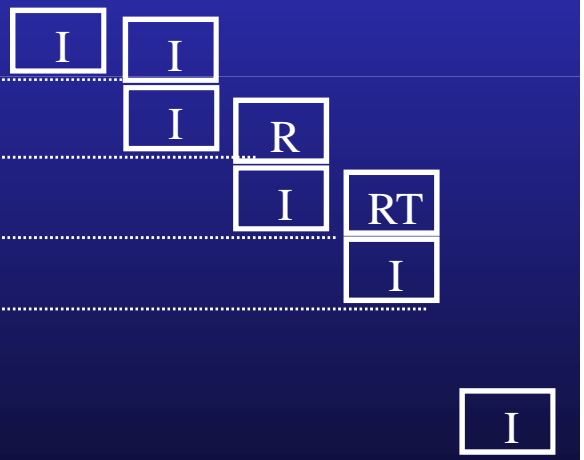
- Transformação de visualização
  - Sistema local móvel
  - Ordem natural para especificação

```
...
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glRotatef(30,0,0,1);
glTranslatef(10,0,0);
...
```



Manipulação da pilha de matrizes

```
…
glMatrixMode(GL_MODELVIEW);
glLoadIdentity( );
glPushMatrix( );
glRotate(30,0,0,1);
glTranslate(10,0,0);
draw_object_1( );
glPopMatrix( );
...
```



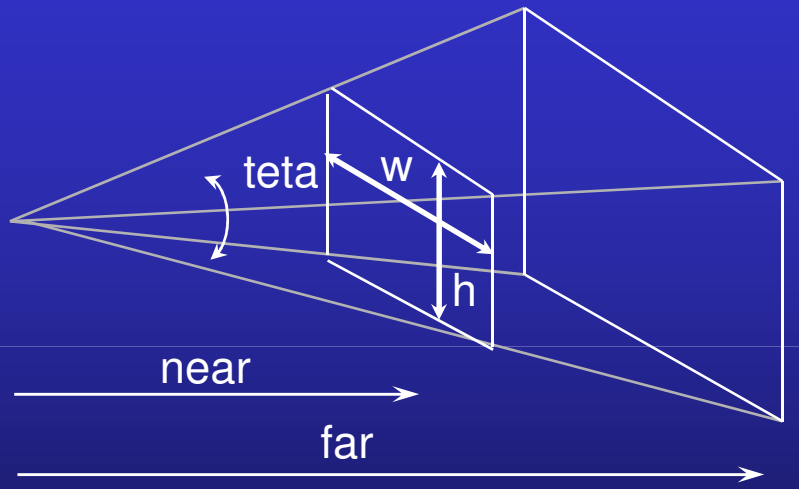
Posicionamento da camera

- Função auxiliar

```
...
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt(eye_x, eye_y, eye_z,
          center_x, center_y, center_z,
          up_x, up_y, up_z
         );
...
```

Projeção: tipo de camera

- Perspectiva

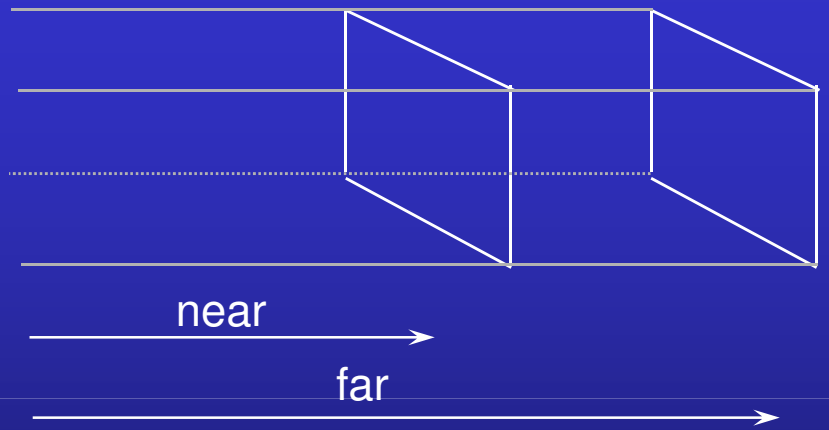


aspect = w / h

```
...
glMatrixMode(GL_PROJECTION);
glLoadIdentity( );
gluPerspective (teta_y,aspect,znear,zfar);
...
```

Projeção: tipo de camera (cont.)

- Ortográfica




```
…
glMatrixMode(GL_PROJECTION);
glLoadIdentity( );
glOrtho (xleft,xright,ybottom,ytop,znear,zfar);
...

2D:
gluOrtho2D (xleft,xright,ybottom,ytop);
```

Transformação de *viewport*

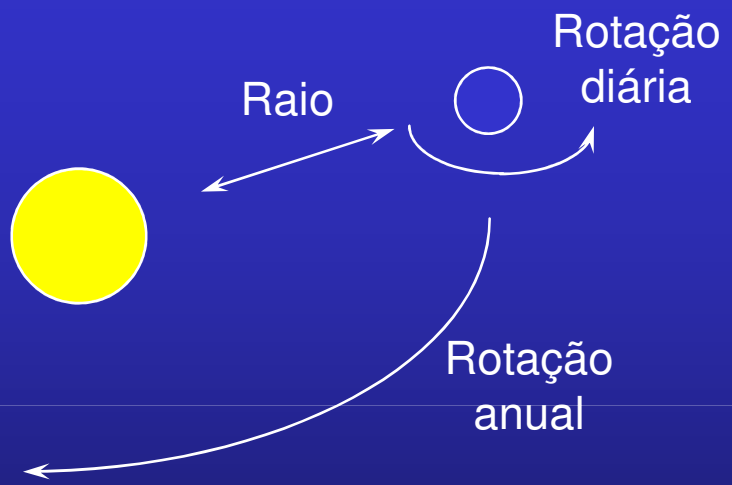
```
...
glViewport (x, y, width, height);
...
```

IUP & GLUT:

• A função default de “resize” define a *viewport* como sendo a área total do canvas.

Exemplo: sistema solar

- Sol e um planeta



Sol: desenhado na origem

Planeta:

Pensando em sistema local

- Rotação anual
- Translação em x
- Rotação diária

Remoção de superfícies ocultas

- Z-BUFFER
  - Inicializa window (default)
  - Habilita teste em Z

    glEnable (GL_DEPTH_TEST);
  - Define teste

    glDepthFunc (GL_LESS);
  - Limpa buffer

    glClear (GL_DEPTH_BUFFER_BIT);

Animação

- *Double color buffer: BACK & FRONT*
  - Inicialização
    - GLUT

      glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGB);
    - IUP/C

      cv = IupGLCanvas (“redraw”);

      IupSetAttribute (cv, IUP_BUFFER, IUP_DOUBLE);
    - IUP/LED

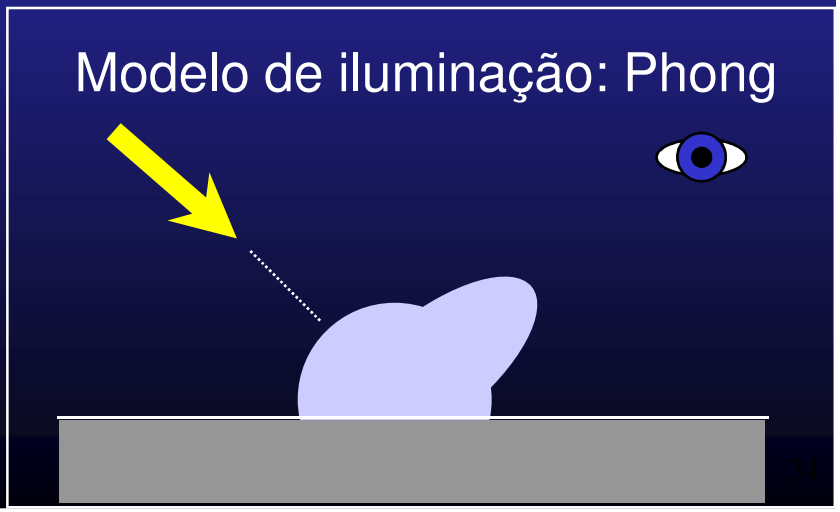
      cv = GLCanvas [BUFFER=DOUBLE] (redraw)
  - Atualização da tela

    IupGLSwapBuffers (cv);

    glutSwapBuffers( );

Rendering

- Cor do objeto depende de:
  - fonte de luz
  - orientação da superfície
  - posição do observador
  - reflexividade do material
    - ambiente
    - difusa
    - especular



Especificação da orientação

- Vetor normal em cada vértice

  glNormal3d(nx,ny,nz);

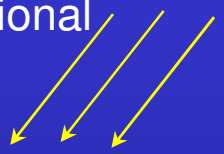
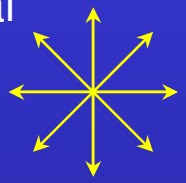
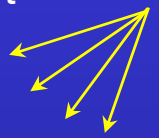
- Se não for normalizado

  glEnable (GL_NORMALIZE);

Obs: cálculo de normal é caro!

Fontes de luz

- Tipos

Direcional  Pontual  Spot 

```
GlfloatGlfloat pos[ ] = {x,y,z,w};
glLightf (GL_LIGHT0, GL_POSITION, pos);
```

- Cor e intensidade: ambiente, difusa, especular

```
Glfloat dif[ ] = {red,green,blue,alpha};
glLightf (GL_LIGHT0, GL_DIFFUSE, dif);
```

- Habilitação

```
glEnable (GL_LIGHTING);
glEnable (GL_LIGHT0);
```

Parâmetros adicionais de iluminação

- Luz ambiente global

```
GLfloat amb[ ] = {0.2,0.2,0.2,1.0};
glLightModelfv(GL_LIGHT_MODEL_AMBIENT, amb);
```

- Posição do observador: local ou infinito

```
glLightModeli (GL_LIGHT_MODEL_VIEWER, GL_TRUE);
```

- Iluminação de faces: *back e front*

```
glLightModeli(GL_LIGHT_MODEL_TWO_SIDE, GL_TRUE);
```

Material

- Cor (reflexividade)
  - Ambiente
    - não depende de orientação
  - Difusa
    - depende da orientação da superfície e da posição da fonte de luz
  - Especular
    - depende da orientação da superfície, da posição da fonte de luz e posição do observador
  - Brilho (*shininess*)
    - fator de brilho da reflexão especular
  - Emissão
    - para representação de fontes de luz na cena

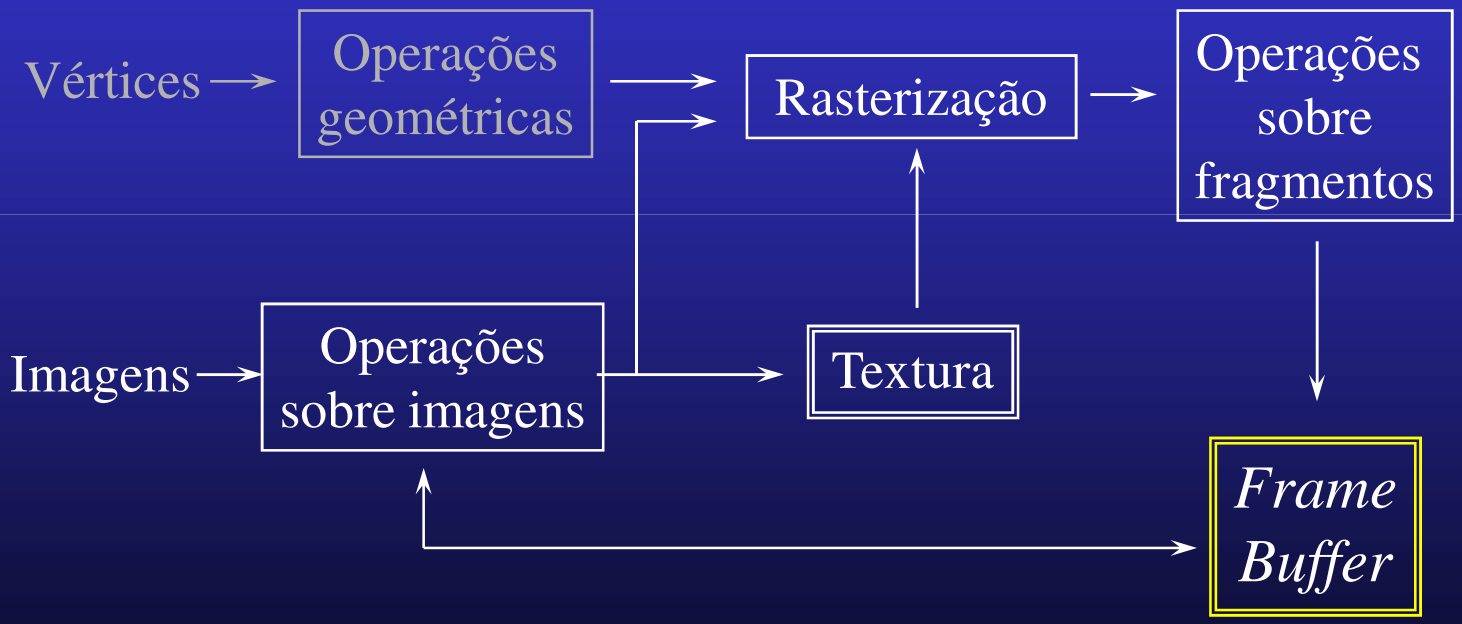
```
Glfloatcolorcolorcolorcolorcolor [ ] = { red, green, blue, alpha };
glMaterialf (GL_BACK_AND_FRONT,
                    GL_AMBIENT_AND_DIFFUSE,
                     color);
```

Cor como material

- Usando cor para definição de material

```
glColorMaterial (GL_BACK_AND_FRONT,
                 GL_AMBIENT_AND_DIFFUSE);

glEnable (GL_COLOR_MATERIAL);

...
glColor3f (red, green, blue);
...
```

OpenGL rendering pipeline



Blending

- Combinação da cor do fragmento sendo processado com a cor do pixel
  - depois da rasterização e antes do fragmento ser desenhado no *framebuffer*.

- Aplicações
  - transparência
  - composição digital
  - pintura

obs: *blending* não funciona com *color index*.

Blending: exemplos de uso

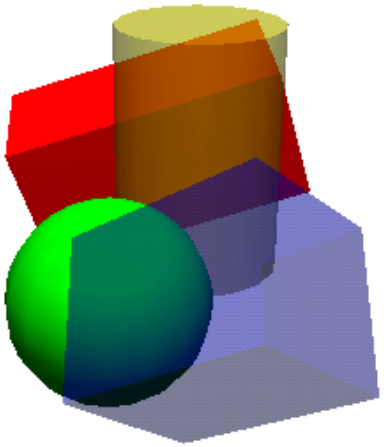
- Desenho temporário sobre imagem

```
glEnable (GL_BLEND);
glBlendFunc (GL_ONE_MINUS_DST_COLOR,
                    GL_ZERO);
glColor3d (1.0, 1.0, 1.0);
```

- Objetos transparentes
  - Cor dada por: (*red, green, blue, opacity*)

```
glEnable (GL_BLEND);
glBlendFunc (GL_SRC_ALPHA,
                        GL_ONE_MINUS_SRC_APHA );
```

Transparência em cena 3D



- habilita-se z-buffer
- desenha-se objetos opacos
- define-se z-buffer como *read-only*

```
glDepthMask (GL_FALSE);
```

- desenha-se objetos com transparência em ordem

Imagens

- Representa uma área retangular de valores associados aos pixels
- Fatores complicantes
  - existem diferentes dados associados aos pixels
  - existem diferentes formas de armazenar uma imagem
  - existem diferentes conversões de dados quando operamos sobre pixels

Operações sobre imagens

- Read

  Frame buffer  Memória

- Draw

  Memória  Frame buffer

- Copy

  Frame buffer  Frame buffer

Formato de cada pixel

- De 1 a 4 elementos representam um pixel

GL_RGB
GL_RGBA
GL_RED
GL_GREEN
GL_BLUE
GL_ALPHA
GL_LUMINANCE
GL_LUMINANCE_ALPHA
GL_DEPTH_COMPONENT
GL_STENCIL_INDEX
GL_COLOR_INDEX

Tipo de cada elemento

UNSIGNED_BYTE ..... 8 bits
BYTE ..... 7 bits
UNSIGNED_SHORT ..... 16 bits
SHORT ..... 15 bits
UNSIGNED_INT ..... 32 bits
INT ..... 31 bits
FLOAT ..... [0.0,1.0]

BITMAP..... 1 bit

Desenhando imagens

- Posição da imagem

  glRasterPos* (x, y, z, w);

- Especificação da imagem

  glDrawPixels (width, height, format, type, pixels)

- Especificação de bitmap
  - Projetado para suportar desenho de fontes raster

  glBitmap (width, height, x_0, y_0, x_{inc}, y_{inc}, bitmap);

Textura

- Mapeamento de imagens sobre primitivas
- Imagem composta por *texels*
- Largura e altura: 2^n

```
gluScaleImage (format,
                width_in, height_in, type_in, data_in,
                width_out, height_out, type_out, data_out);
```

- Especificação: 1D e 2D

```
glEnable (GL_TEXTURE_2D or GL_TEXTURE_1D);
glTexImage2D (GL_TEXTURE_2D, level, components,
                width, height, border, format, type, pixels);
glTexImage1D (GL_TEXTURE_1D, level, components,
                width, border, format, type, pixels);
```

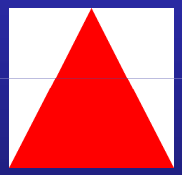
Coordenada de textura

- Para cada vértice
  - coordenada: identifica qual o pixel no buffer
  - coordenada de textura: identifica qual o texel
- Coordenadas de textura: s, t, r, q
- Coordenadas de textura são linearmente interpoladas entre vértices

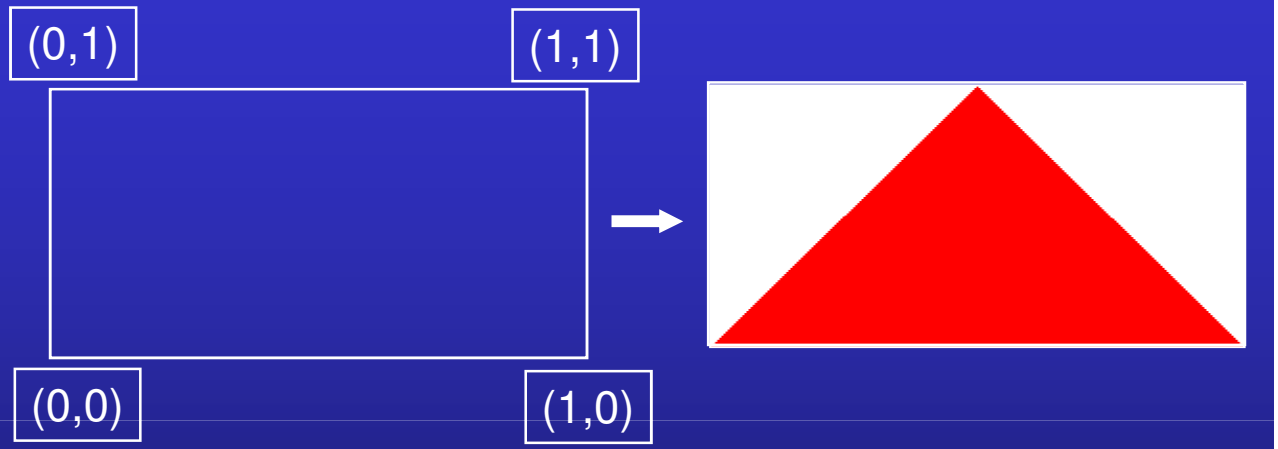
```
glTexCoord* (s, t, r, q);
```

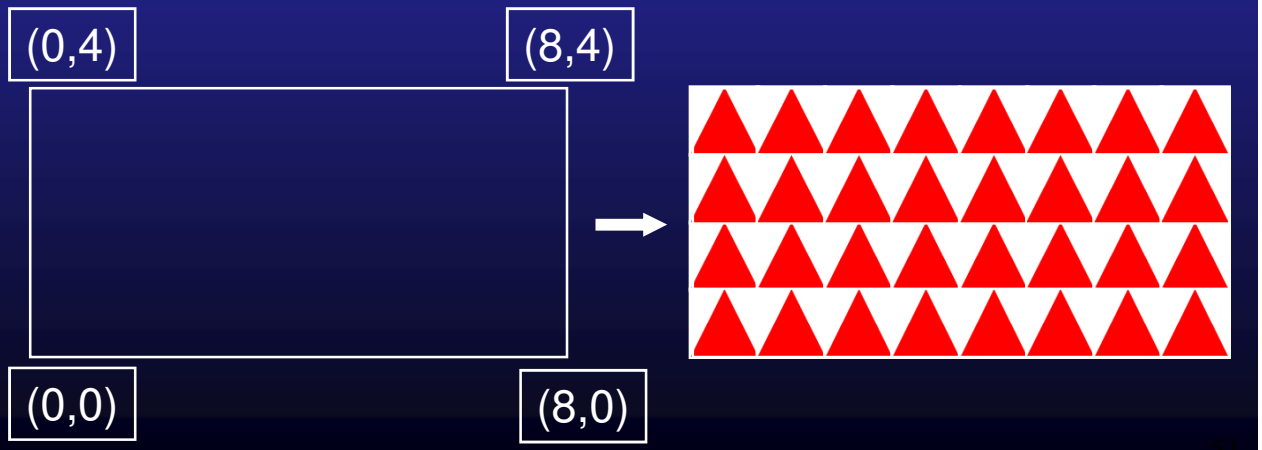
- Intervalo [0,1], senão *wrap*?
- Transformados pela matriz de textura

Mapeamento de textura



Textura





Combinação de pixel com texel

- *Decal*
  - Cor definida pelo texel
- *Modulate*
  - Cor do pixel é modulada pela cor do texel
- *Blend*
  - Cor combinada com uma cor adicional de ambiente
- Exemplo
  - Modular com a cor branca para dar iluminação

```
glTexEnvi (GL_TEXTURE_ENV,
           GL_TEXTURE_ENV_MODE,
           GL_MODULATE);
```

Exemplo de modulação

DECAL

MODULATE

Referências

- "The Red Book"

  OpenGL: Programming Guide

  Release 1.2

  M. Woo, J. Neider, T. Davis

- Web sites

  The official OpenGL web page

  http://www.opengl.org

  SGI's OpenGL WWW Center

  http://www.sgi.com/Technology/openGL

  Gateway to OpenGL

  http://reality.sgi.com/mjk_asd/opengl-links.html