Prof. Erickson R. Nascimento
erickson@dcc.ufmg.br

INTRODUÇÃO À OPENGL

VERLab Computer Vision and Robotics
DCC DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
UFMG

1

O que é OpenGL

VERLab Computer Vision and Robotics

- É uma especificação de uma interface de programação (API) para renderização de gráficos 2D e 3D.
  - É independente de plataforma
  - É independente do sistema de janelas

UFMG

2

O que é OpenGL

VERLab Computer Vision and Robotics

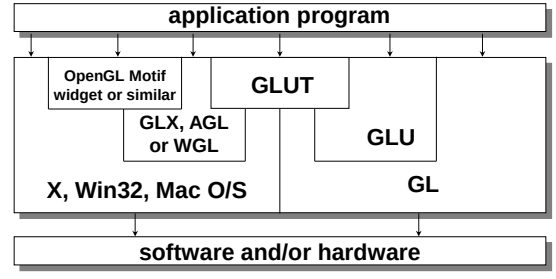
- Bibliotecas auxiliares
  - GLU (OpenGL Utility Library)
    - É parte da OpenGL
    - NURBS, tessellators, quadric shapes, etc.
  - GLUT (OpenGL Utility Toolkit)
    - Sistema de gerenciamento de janelas e I/O
    - Não é parte da OpenGL
  - AGL, GLX, WGL
    - Liga OpenGL com sistemas de janela

UFMG

3

OpenGL e sua relação com outras APIs

VERLab Computer Vision and Robotics



UFMG

4

Instalação OpenGL em Python

- Baixe os arquivos em http://www.lfd.uci.edu/~gohlke/pythonlibs/:
  - PyOpenGL-3.1.1-cp27-cp27m-win_amd64.whl
  - PyOpenGL_accelerate-3.1.1-cp27-cp27m-win_amd64.whl
- Use o pip para instalar as libs:
  - pip install PyOpenGL-3.1.1-cp27-cp27m-win_amd64.whl
  - pip install PyOpenGL_accelerate-3.1.1-cp27-cp27m-win_amd64.whl

5

5

Estrutura de uma aplicação OpenGL

- Configurar e abrir uma janela para renderização
- Inicialização dos estados da OpenGL
  - `glutInitDisplayMode, glClearColor, glClearDepth, glEnable`
- Registro das funções de callback
  - Render, resize, input: keyboard, mouse, etc.
- Loop de processamento

6

Hello OpenGL!

```
glutInit()
glutInitWindowSize( 640, 480 )
glutCreateWindow("2D")

glutInitDisplayMode( GLUT_DOUBLE | GLUT_RGB)
glClearColor(0.0, 0.0, 0.0,0.0)

glutDisplayFunc(displayFun)
glutReshapeFunc(resize)
glutKeyboardFunc(keyboard)

glutMainLoop()
```

7

7

Hello OpenGL!

- Quais funções serão responsáveis por:
  - Ajuste das dimensões da janela
  - Input
  - Animação
- Registro das funções callbacks com GLUT

```
glutDisplayFunc( display );
glutIdleFunc( idle );
glutKeyboardFunc( keyboard );
```

8

8

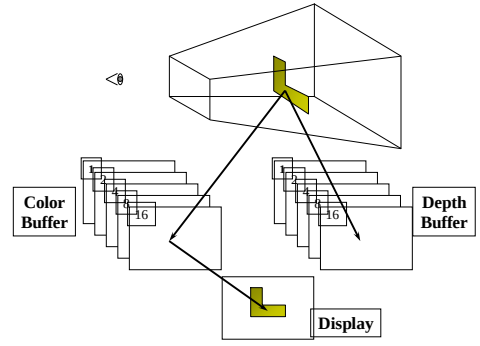
Animação utilizando Double Buffering

- Ajustando um double
  - `glutInitDisplayMode( GLUT_RGB | GLUT_DOUBLE );`
- Limpando o buffer de cores
  - `glClear( GL_COLOR_BUFFER_BIT );`
- Renderização da cena no buffer
- Mostra o buffer na tela
  - `glutSwapBuffers();`

UFMG

9

Double Buffer



UFMG

10

Hello OpenGL!

```
def draw_rect(x, y, width, height):
    glBegin(GL_QUADS)
    glVertex2f(x, y)
    glVertex2f(x + width, y)
    glVertex2f(x + width, y + height)
    glVertex2f(x, y + height)
    glEnd()

def display():
    glClear(GL_COLOR_BUFFER_BIT|GL_DEPTH_BUFFER_BIT)

    glColor3f(0.0, 1.0, 1.0)
    draw_rect(10, 10, 200, 100)

    glutSwapBuffers()
```

11

UFMG

11

Primitivas Geométricas

double v[3];

glVertex3fv(v)

glVertex3f(x, y, z)

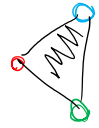
Number of components	*Data Type*	*Vector*
2 - (x,y) 3 - (x,y,z) 4 - (x,y,z,w)	b - byte ub - unsigned byte s - short us - unsigned short i - int ui - unsigned int f - float d - double	omit "v" for scalar form glVertex2f(x, y)

UFMG

12

Primitivas Geométricas

- As primitivas são especificadas usando as funções:
  - `glBegin( tipo )` e `glEnd();`
  - `tipo`: determina como os vértices serão conectados

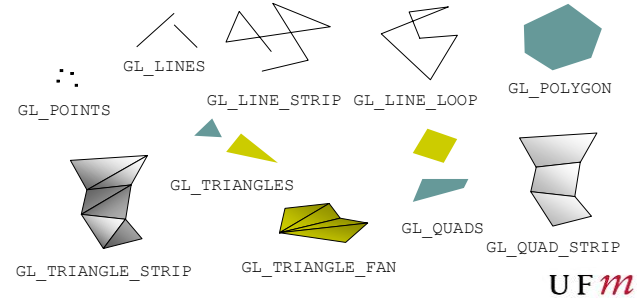


```
GLfloat red, greed, blue;
Glfloat coords[3];
glBegin( primType );
for ( i = 0; i < nVerts; ++i ) {
  glColor3f( red, green, blue );
  glVertex3fv( coords );
}
glEnd();
```

13

Primitivas Geométricas

- Todas as primitivas geométricas em OpenGL são definidas por vértices



14

Primitivas Geométricas

```
void drawQuad ( GLfloat color[] )
{
   glBegin( GL_QUADS );
   glColor3fv( color );
   glVertex2f( 0.0, 0.0 );
   glVertex2f( 1.0, 0.0 );
   glVertex2f( 1.0, 1.0 );
   glVertex2f( 0.0, 1.0 );
   glEnd();
}
```

15

Transformações em OpenGL

- Transformações de projeção
  - Ajustes dos parâmetros intrínsecos da câmera
- Transformações de visualização
  - Define a posição e orientação do volume de visualização no mundo

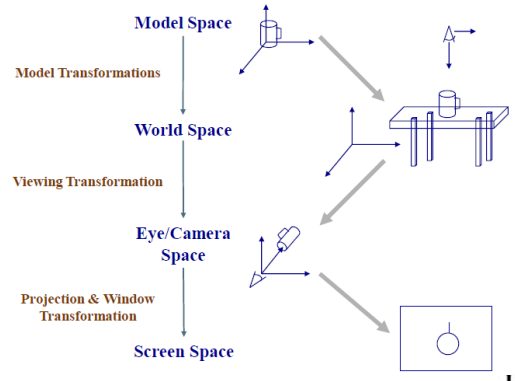
16

Transformações em OpenGL

- Transformações de modelagem
  - Move o modelo
- Transformações de Viewport
  - Aumenta ou reduz o tamanho do sensor

17

Transformações em OpenGL



18

Transformações em OpenGL

- Vértices são representados por um vetor coluna

$$\vec{v} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix}$$

  - w = 1.0 (geralmente)
  - Operações são realizados por multiplicação de matrizes
  - Direções (segmentos de reta direcionado) podem ser representadas usando w = 0.0

19

Transformações em OpenGL

- Um vértice é transformado por matrizes 4×4
  - Todas as transformações afins são multiplicações de matrizes.
  - As matrizes são armazenas concatenando as colunas (column-major).
  - As matriz são sempre pos-multiplicadas.
  - O produto de uma matriz e um vetor é dado por:

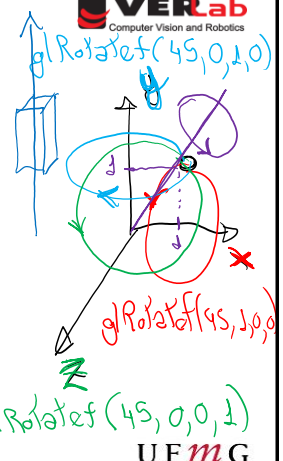
$$\mathbf{M}\vec{v}$$

$$\mathbf{M} = \begin{bmatrix} m_0 & m_4 & m_8 & m_{12} \\ m_1 & m_5 & m_9 & m_{13} \\ m_2 & m_6 & m_{10} & m_{14} \\ m_3 & m_7 & m_{11} & m_{15} \end{bmatrix}$$

20

Transformações rígidas no Modelo

- Mover
  - `glTranslate{fd}( x, y, z )`
- Rotacionar em volta de um eixo (x, y, z)
  - `glRotate{fd}( angle, x, y, z )`
  - angle is in degrees
- Aumentar, diminuir ou espelhar
  - `glScale{fd}( x, y, z )`



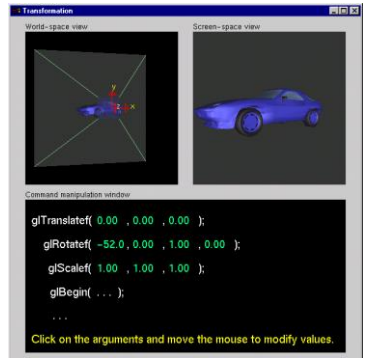
glRotatef(45, 1, 1, 0)

UFMG

21

Transformações rígidas no Modelo

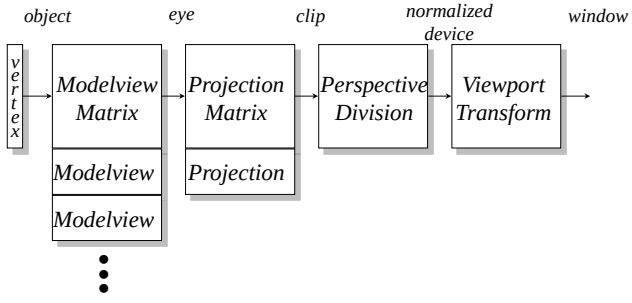
- Exemplo



UFMG

22

Pipeline de transformações geométricas



UFMG

23

Operações com matrizes

- Escolha da matriz na pilha interna
  - `glMatrixMode(GL_MODELVIEW or GL_PROJECTION)`
- Outras operações:
  - `glLoadIdentity()`: Inicializa a matriz com a matriz identidade.
  - `glPushMatrix()`: Salva a matriz atual em uma pilha.
  - `glPopMatrix()`: Restaura a última matriz sala na pilha.

UFMG

24

Ajustes para visualização da cena

- Viewport: janela onde será feito o desenho
  - Em geral tem as mesmas dimensões da janela
  - `glViewport( x, y, width, height )`

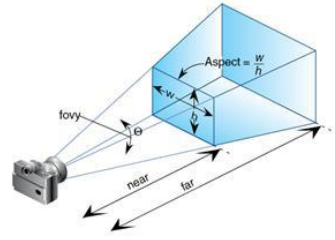
- Projeção Perspectiva
  - `gluPerspective( fovy, aspect, zNear, zFar )`
  - `glFrustum( left, right, bottom, top, zNear, zFar )`

UFMG

25

Ajustes para visualização da cena

- Projeção Perspectiva
  - `gluPerspective( fovy, aspect, zNear, zFar )`
  - `glFrustum( left, right, bottom, top, zNear, zFar )`



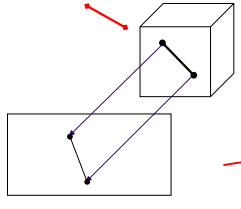
http://www.codersource.net

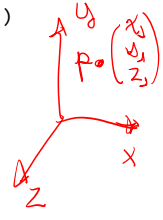
UFMG

26

Ajustes para visualização da cena

- Projeção Ortográfica Paralela
  - `glOrtho( left, right, bottom, top, zNear, zFar )`
  - `gluOrtho2D( left, right, bottom, top )`





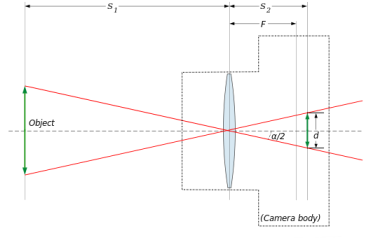
UFMG

27

Angle-of-view

- Precisamos fornecer
  - `gluPerspective(fovy,aspect,near,far)`

- $\tan\frac{\alpha}{2} = \frac{\frac{d}{2}}{F}$
- $\tan\frac{\alpha}{2} = \frac{d}{2F}$
- $\alpha = 2\arctan\frac{d}{2f}$



UFMG

28

Matriz de projeção e os parâmetros intrínsecos

```
def ajuste_intrinsecos_camera(width, height, K)
    fx = K[0,0];
    fy = K[1,1];
    fovy = 2*arctan(0.5*height/fy)*180/pi;
    aspect = (width*fy)/(height*fx);

    near = 0.1;
    far = 100.0;

    gluPerspective(fovy,aspect,near,far);
```

29

29

Ajustes para visualização da cena

- Mudança na pose da câmera
  - gluLookAt(eyex, eyey, eyez, aimx, aimy, aimz, upx, upy, upz)

30