OPENGL :

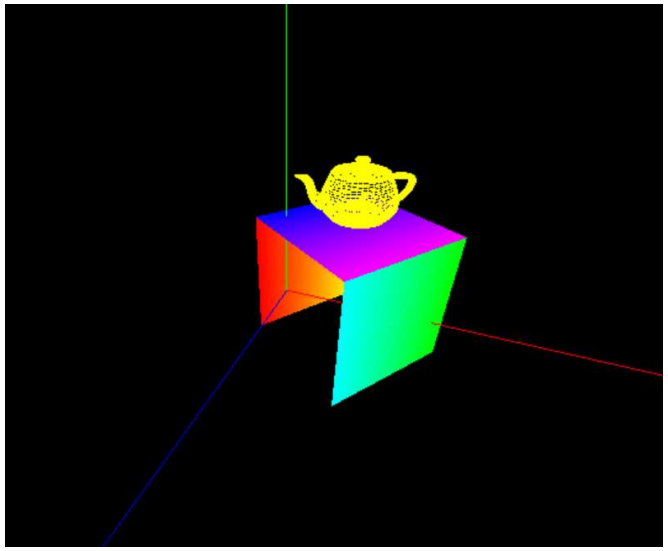
vertex_array,
normais,
faces visíveis, …
MESA

Visualização – comando lookAt

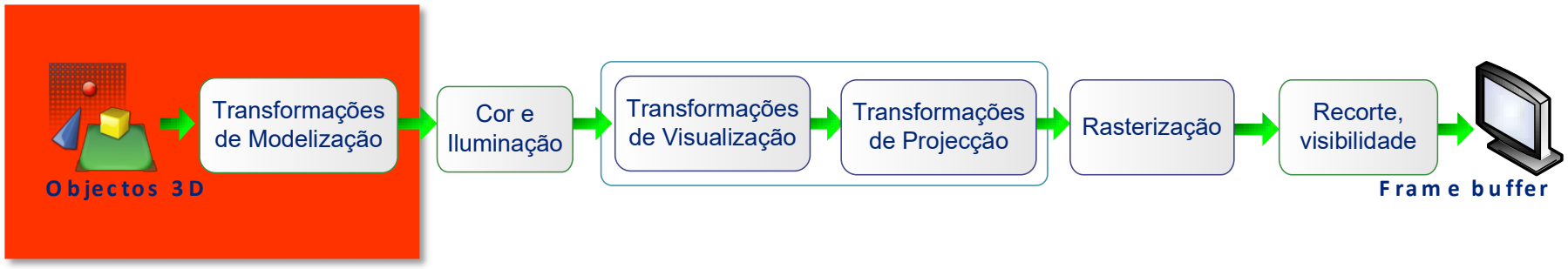
TP2 - Computação Gráfica

Sumário

- **Objectos: MESA**



  - **1. Geometria: Objectos**
    - Mesa
    - Bule
  - **2. Transformações geométricas**
    - Escalas, rotações, translações
    - (?)
  - **3. Visibilidade**
    - Lados de um polígonos
    - Normais



Objectivos

1. Objectos: MESA

- **Construção da mesa**
  - Uso da técnica ***VERTEX_ARRAY***, facilitar a construção de objectos
  - Três polígonos: esquerda, direita, cima
  - Básico para a escada …..

- **Teapot**
  - glutSolideTeapot (tamanho);
  - glutWireTeapot (tamanho);

  - Desenhado centrado na origem !!

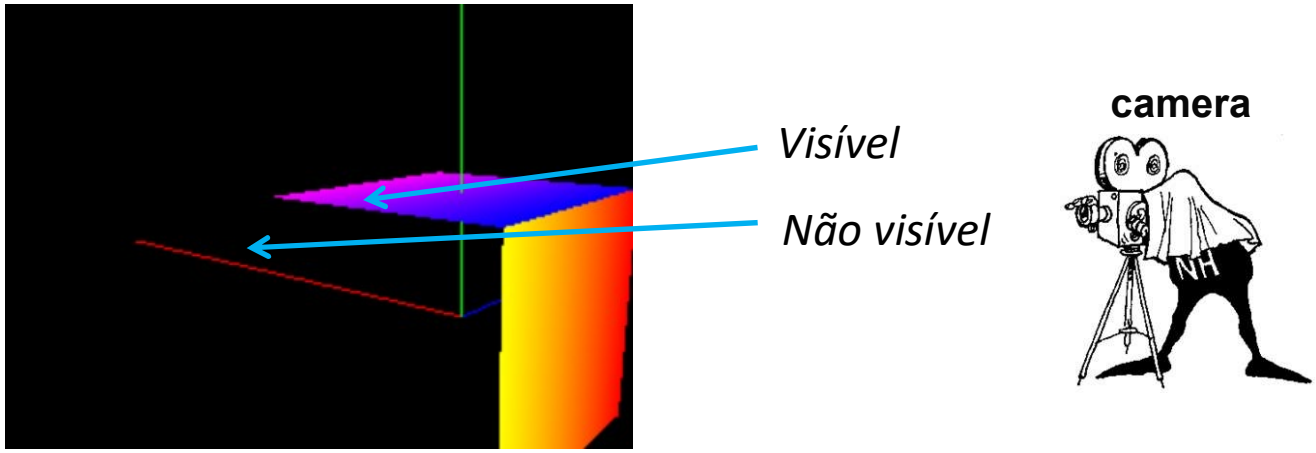
Objectivos

- **3. Visibilidade**

Apenas as faces voltadas para o observador devem ser visíveis.

Deve ser possível definir a face visível - exterior/interior.

- Regra da mão direita
- Normais

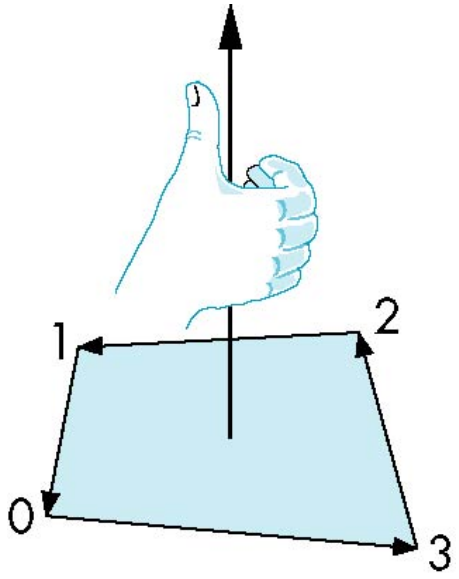


Objectos

- Faces Visiveis
  - ***Definição***
- Normais
- Vertex Array
- Faces Visiveis
  - ***Visualização***
- ***LookAt***

Faces Visíveis

- **Lado da frente e de trás de um polígono:**
- **Regra mão direita** (sentido anti-horário)
  - Frente ou fora (ordem 0,3,2,1)
  - Trás ou dentro (ordem 0,1,2,3)



TP2 – Visibilidade

- Faces Visíveis
- Normais
- Vertex Array
- Faces Visiveis
  - ***Visualização***
- ***LookAt***

Normais: definição

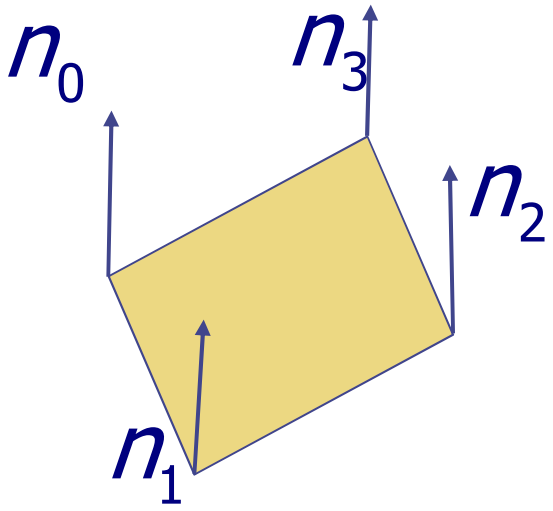
- Além das propriedades da luz e materiais (a ver em capitulos futuros), a geometria do objecto é também importante:

- **Vértice (objecto) definido por !!**
  - **Coordenadas**
  - **Cor**
  - **Normal**

- O vector ***normal*** é fundamental
  - Necessita de ser especificada com **glNormal ()**
  - Por omissão “aponta” para fora

Normais

- Definição vértice a vértice:




```
glBegin (GL_POLYGON);
      glNormal3fv(n0);
      glVertex3fv(v0);
      glNormal3fv(n1);
      glVertex3fv(v1);
      glNormal3fv(n2);
      glVertex3fv(v2);
      glNormal3fv(n3);
      glVertex3fv(v3);
glEnd();
```

```
glBegin (GL_POLYGON);
        glNormal3fv(n0);
        glVertex3fv(v0);
        glVertex3fv(v1);
        glVertex3fv(v2);
        glVertex3fv(v3);
glEnd();
```

Normais

- Normalização:
  - Por omissão normais não são normalizadas
  - Para o fazer
    - glEnable(GL_NORMALIZE)
  - Implica cálculos adicionais, …

TP2 – Visibilidade

- Faces Visíveis
- Normais

- Vertex Array

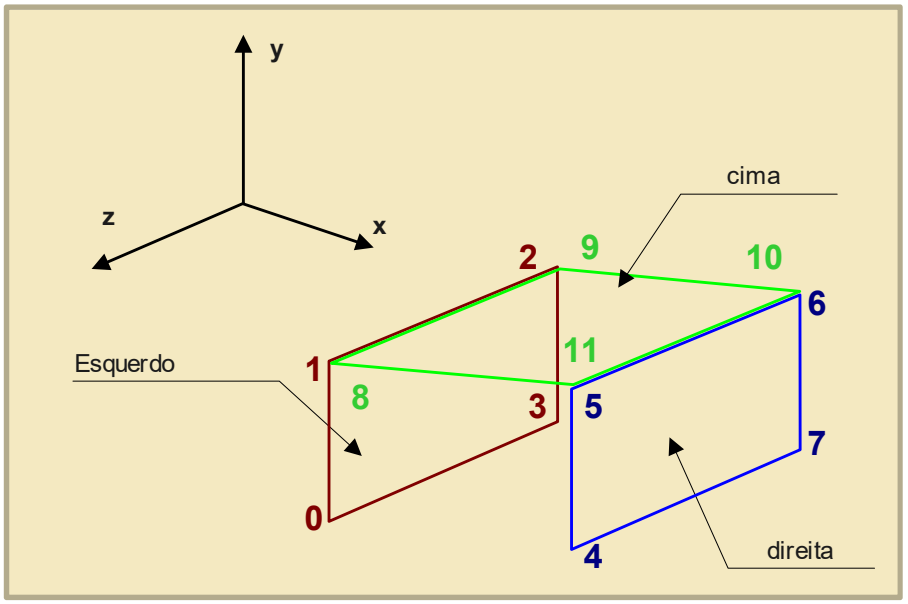
- Faces Visíveis
  - ***Visualização***

- ***LookAt***

Vertex Arrays

- Forma de facilitar a definição de um objecto

```
static GLfloat vertices[]={
//.....Esquerda
      -tam, -tam,  tam,  // 0
      -tam,  tam,  tam,  // 1
      -tam,  tam, -tam,  // 2
      -tam, -tam, -tam,  // 3
//..... Direita
      tam, -tam,  tam,   // 4
       tam,  tam,  tam,  // 5
       tam,  tam, -tam,  // 6
       tam, -tam, -tam,  // 7
//..... (Cima
      -tam,  tam,  tam,  // 8
      -tam,  tam, -tam,  // 9
       tam,  tam, -tam,  // 10
       tam,  tam,  tam,  // 11
};
```



- O OpenGL disponibiliza fundamentalmente três funções para o efeito:
  - glEnableClientState( )
  - glVertexPointer( )
  - glDrawElements()

Vertex Arrays

- 1. Activar o modo em causa
  - **glEnableClientState(GL_VERTEX_ARRAY);**

- 2. Atribuir vértices
  - **void glVertexPointer(GLint *size*, GLenum *type*, GLsizei *stride*, const GLvoid **pointer*);**

glVertexPointer(3, GL_FLOAT, 0, vertices);

  - *Indica que as coordenadas do objecto*
    - *são constituídas por pontos de dimensão 3,*
    - *do tipo GL_FLOAT e*
    - *previamente definidas no array vertices.*
    - *O parâmetro (O-zero) especifica o offset entre vértices.*

Vertex Arrays

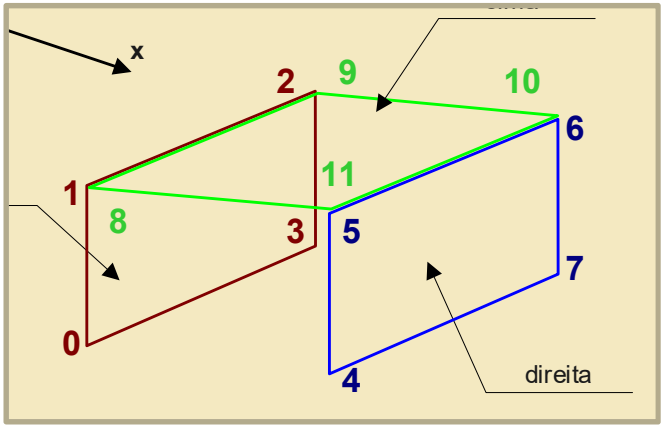
Simplificação !!!

- 3. Desenhar
  - **void glDrawElements(GLenum *mode*, GLsizei *count*, GLenum *type*, void **indices*);**

glDrawElements(GL_POLYGON, 4, GL_UNSIGNED_BYTE, esquerda);

- *Desenha uma primitiva do tipo polígono (GL_POLYGON), definida por 4 vértices, especificados no vector de nome esquerda e do tipo GL_UNSIGNED_BYTE.*
- ***Objecto é apenas definido por uma lista de vértices***

- ***Esquerda definida como***
  - static GLuint esquerda[ ]= {0, 1, 2, 3};
  - static GLuint direita[ ]= {4, 7, 6, 5};



Normais

- ***4. Definição das normais e das cores é equivalente***
  - *usando apenas um vetor*

- **glNormalPointer(GL_FLOAT, 0, normais);**

```
static GLfloat normais[] = {
//…… x=tam (Esquerda)
    -1.0, 0.0, 0.0,
    -1.0, 0.0, 0.0,
    -1.0, 0.0, 0.0,
    -1.0, 0.0, 0.0,
//…… x=2*tam (Direita)
     1.0, 0.0, 0.0,
     1.0, 0.0, 0.0,
     1.0, 0.0, 0.0,
     1.0, 0.0, 0.0,
//…… y=tam (Cima)
     0.0, 1.0, 0.0,
     0.0, 1.0, 0.0,
     0.0, 1.0, 0.0,
     0.0, 1.0, 0.0,
};
```

TP2 – Visibilidade

- Faces Visiveis
- Normais

- Vertex Array

- Faces Visiveis
  - ***Visualização***

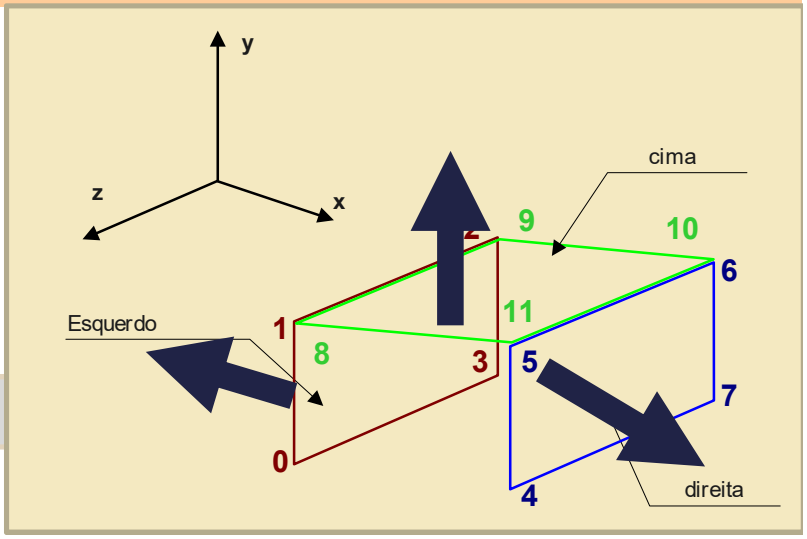
- ***LookAt***

Faces visiveis: desenho

- **Eliminar uma das faces**

  - **1. Activar** Eliminar faces

```
glEnable(GL_CULL_FACE)
```



  - **2. Seleccionar** a face a eliminar

```
void glCullFace (face);
```

    - Qual a *face* eliminar
      - GL_FRONT
      - GL_BACK
      - GL_FRONT_AND_BACK

```
void glCullFace (GL_BACK);
```

2. Visibilidade faces interior/exterior

- É possível modificar a regra da mão direita
  - CW-clockwise
  - CCWcounterclockwise

- glFrontFace(GL_CW) *// ao contrário !!!*
- glFrontFace(GL_CCW) *// a normal*

TP2 – Visibilidade

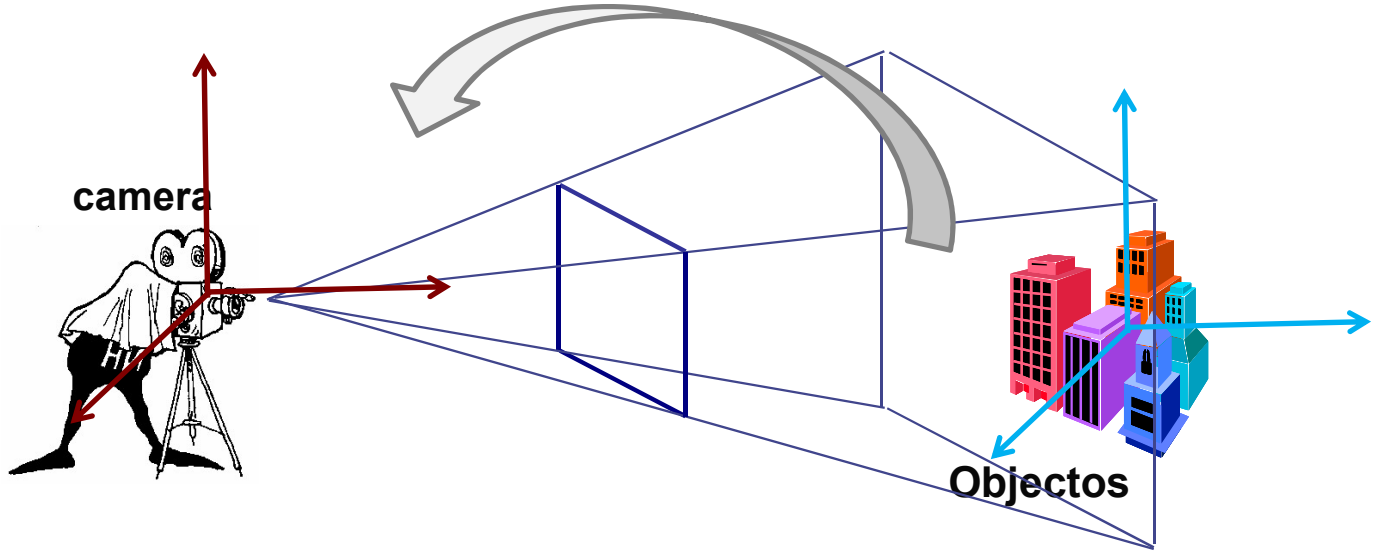
- Faces Visiveis
- Normais

- Vertex Array

- Faces Visiveis

- ***LookAt***

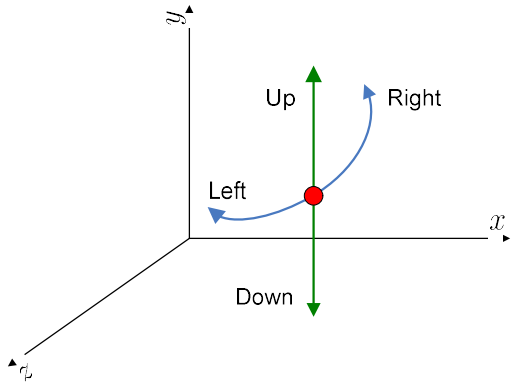

Transformações de Modelização
Cor e Iluminação
Transformações de Visualização
Transformações de Projecção
Rasterização
Operações, visibilidade

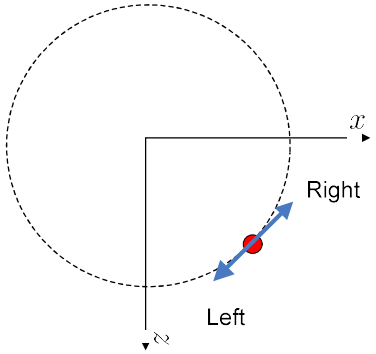

camera
Objectos

3D → 3D

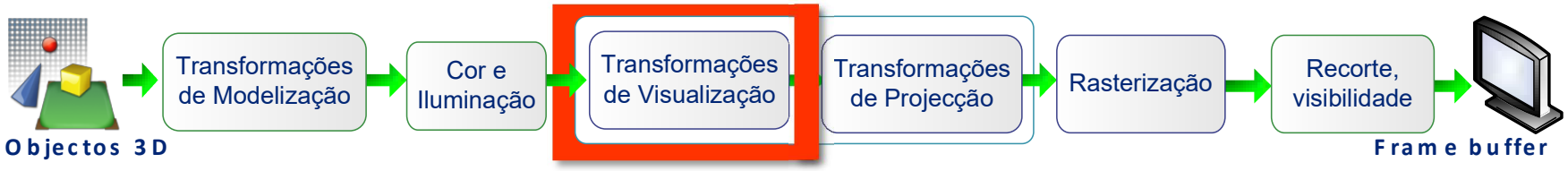
Visualização 3D

- **2. Observador**
  - Subir/descer
  - Girar
  - *Uso das SETAS*



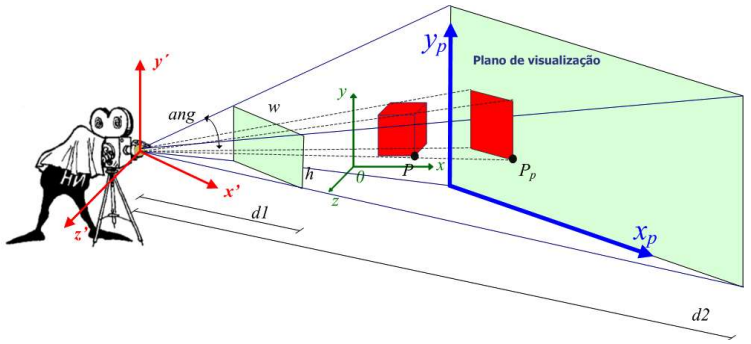


gluLookAt(Ox, Oy, Oz, Dx, Dy, dz, UPx, UPy, UPz);



Projecção

3. Projecção perspectiva

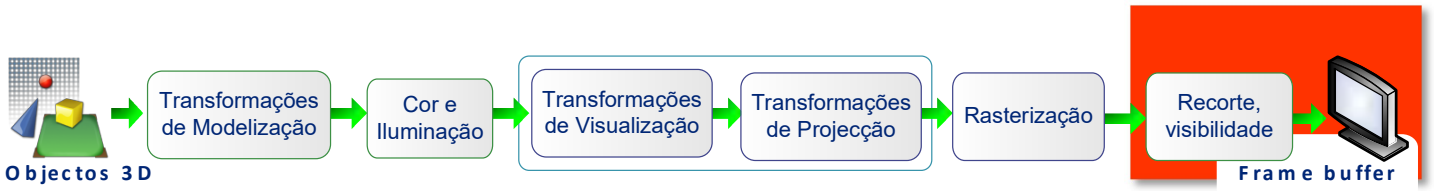


gluPerspective (***angulo***, wScreen/hScreen, d1, d2);

Projecção

- 4. Janela

```
glViewport (0, 0, 3.0*wScreen/4.0, hScreen);
```



Projecção

- **4. Codigo openGL**

```
glViewport (0, 0, wScreen, hScreen);

glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluPerspective(angZoom, (float)wScreen/hScreen, 0.1, 3*zC);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

Visualização

```
> gluLookAt(obX, obY, obZ 0,0,0, 0, 1, 0);
```

Transformações geométricas

```
> glTranslate(2,3,4);
> desenhaObjecto();
```

- Comandos importantes

```
glEnable(GL_DEPTH_TEST);        //……Profundidade / 3D

glNormal3fv(n);                 //…….. Normal

glEnable(GL_CULL_FACE);         //……Permitir eliminar faces
glCullFace(GL_BACK);            //……Qual delas eliminar
```