TP3 – Visualização + vertex array

vertex_array, faces visíveis, normais …

lookAt

TP4 – Projecção - próxima semana

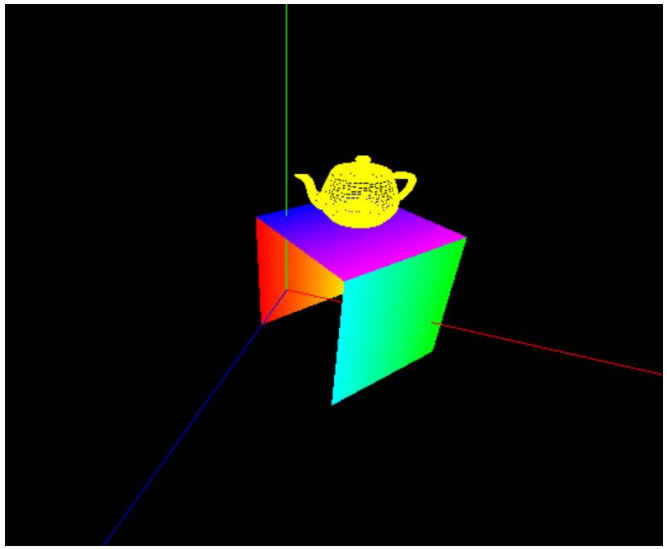
ortogonal

perspectiva

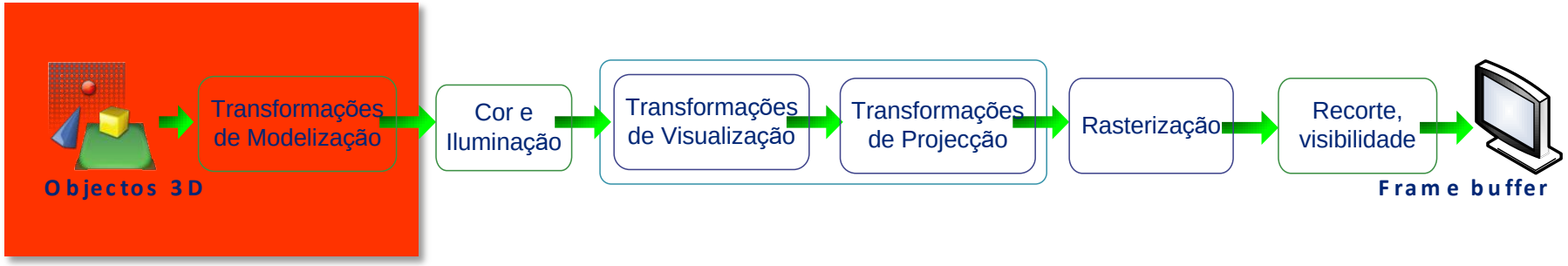
introdução Projecto

Sumário

- **Objectos: MESA**



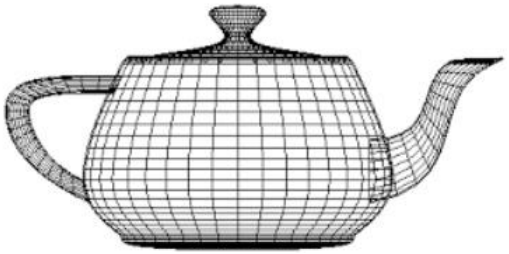
- **1. Geometria: Objectos**
  - Mesa (**vertex array**)
  - Bule (objectos GLUT)
- **2. Transformações geométricas**
  - Escalas, rotações, translações
- **2. Orientação / normais**
  - Lados da frente/trás de um polígono
  - Orientação – uso do vetor normal
- **4. Visualização**
  - Lookat(.)



Objectivos

1. Objectos – superfícies definidas por poligonos

- **Construção da mesa – *vamos nós fazer***
  - Uso da técnica ***VERTEX_ARRAY***, facilitar a construção de objectos
  - Três polígonos: esquerda, direita, cima



- **Objectos da GLUT – *alguém já fez !***
- **exemplo: teapot, toroid, cubo, cone, …**
  - glutSolideTeapot (tamanho);
  - glutWireTeapot (tamanho);
  - Desenhado centrado na origem !!

Pode haver problemas de compatibilidade com o freeGLUT !
Se sim, nada de usar objectos da glut (e não são necessários), ou então usar a GLUT

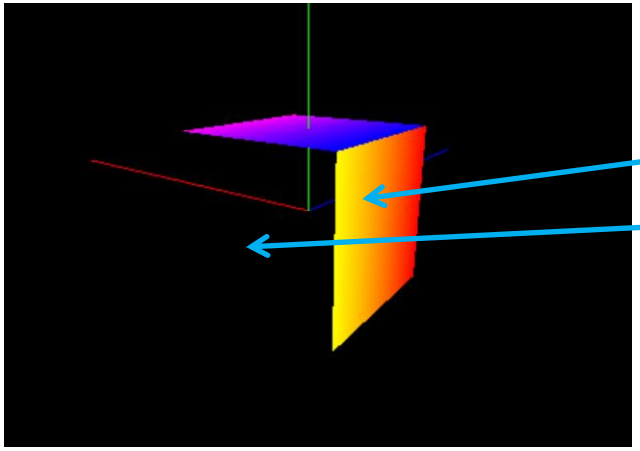
Objectivos

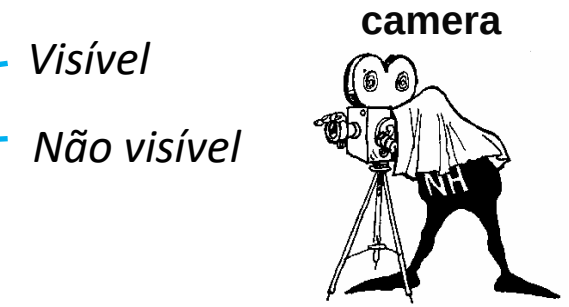
- **2 Mesa: visibilidade**

Apenas as faces voltadas para o observador devem ser visíveis.

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

- Regra da mão direita



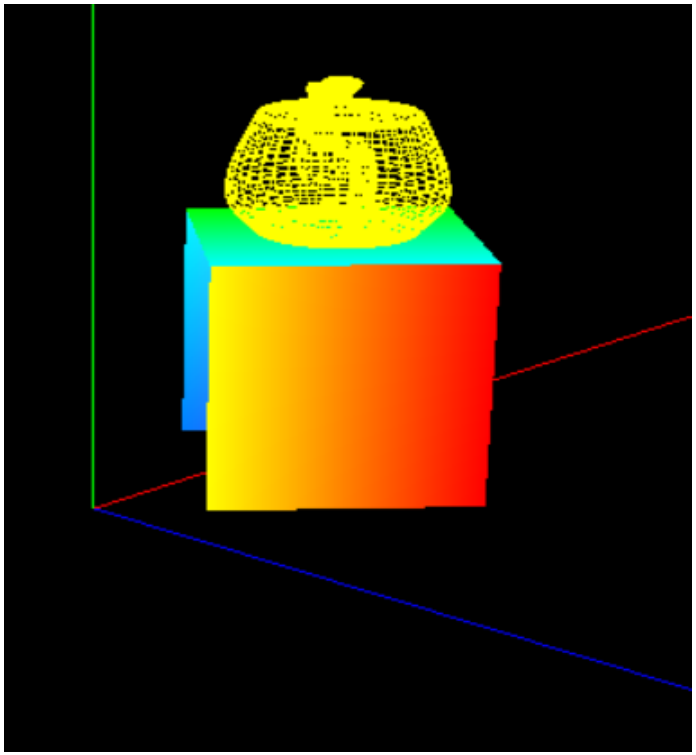


Objectivos

- **3 Mesa: alguma animação**

Com o que já sabemos de transformações geométricas

- Rotações, translações, escalas



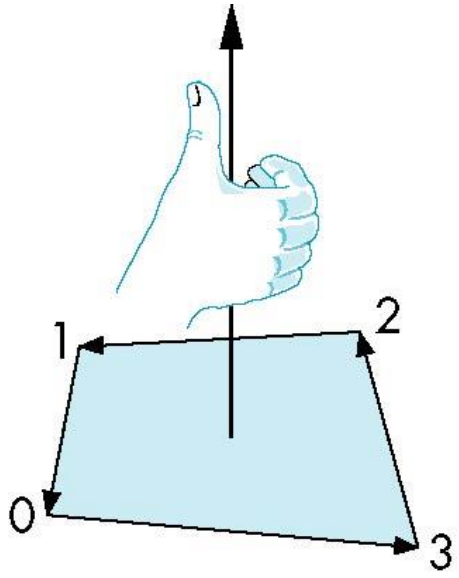
Sumário

- Faces Visiveis
  - ***Definição***

- Faces Visiveis
  - Visualização
- Normais
- Vertex Array
- 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)



Sumário

- Faces Visiveis

- Faces Visiveis
  - **Visualização**

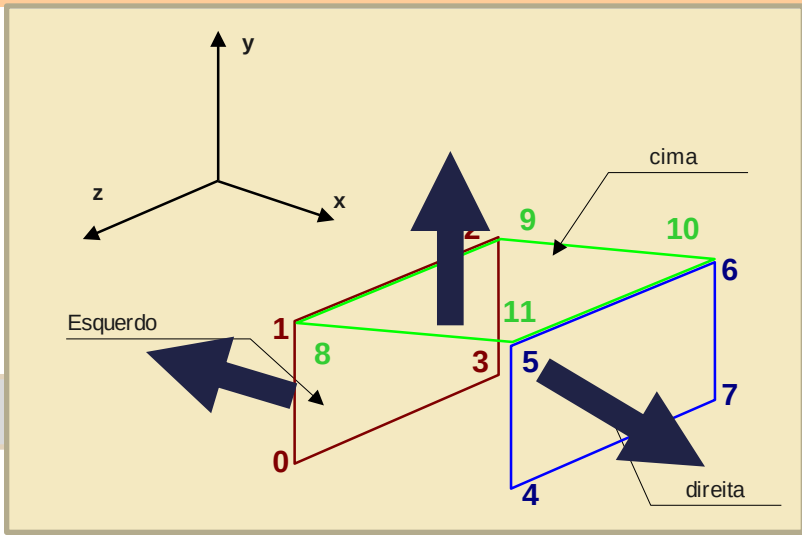
- Normais
- Vertex Array
- Visualização - LookAt

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

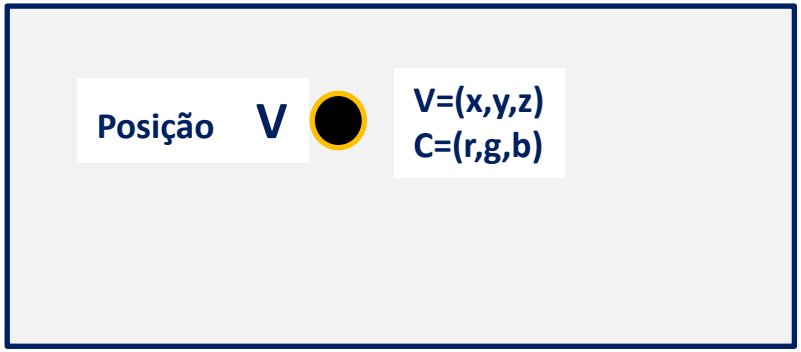
- É possível modificar a regra da mão direita
  - CW-clockwise
  - CCWcounterclockwise

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

Sumário

- Faces Visíveis
- Normais
- Vertex Array
- LookAt

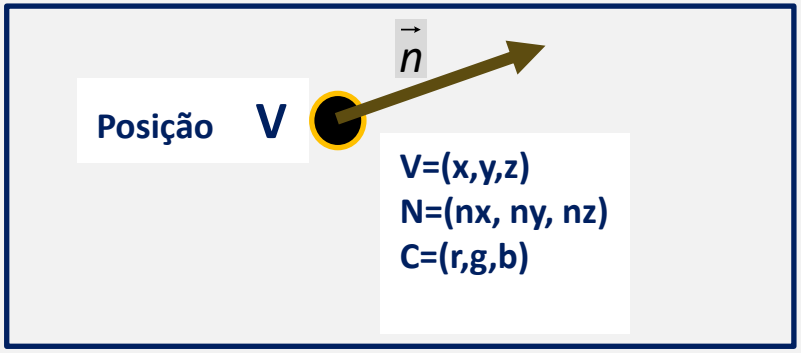
Já vimos
- posição (x,y,z)
- Cor (r,g,b)



Normais: definição

- **Iluminação:** Além das propriedades da luz e materiais (a ver em capítulos futuros), a **orientação** do objecto é fundamental:

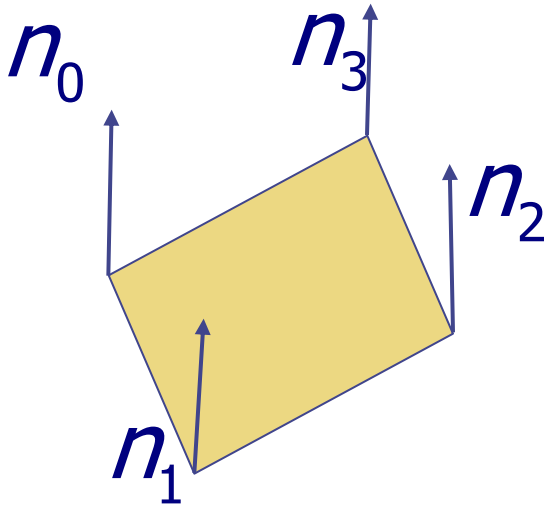
- **Vértice (objecto) definido por atributos**
  - **Coordenadas**
  - **Cor**
  - ***Normal***



- O vector ***normal*** é fundamental
  - Necessita de ser especificada com **glNormal ()**

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, …

Sumário

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

1. **Um objecto é definido por vértices**
2. **Os vértices são ligas por primitivas (polígonos)**
3. **Os polígonos definem a superfície do objecto**

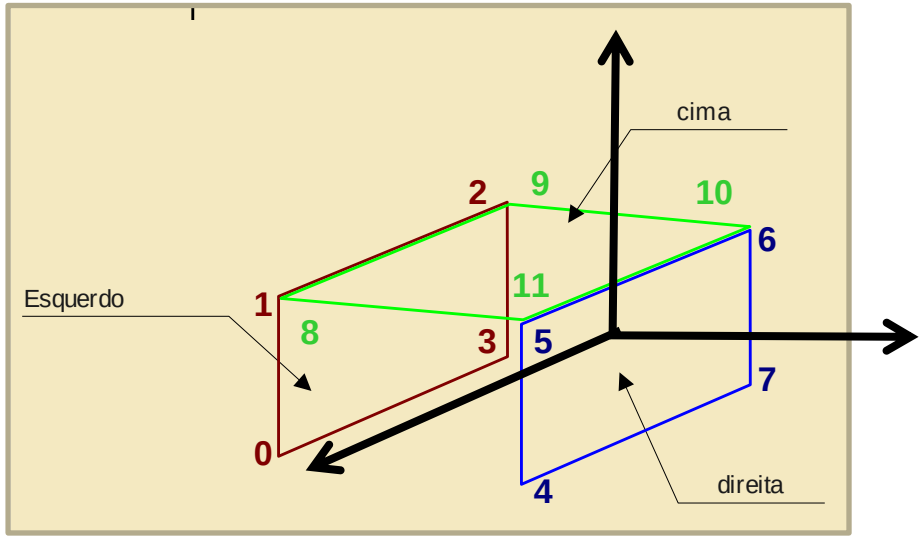
Temos de definir TODOS os vértices e ligá-los convenientemente !!

Moroso !!!

Vertex Arrays

- Forma de facilitar a definição de um objecto
  - Permite "gerir" a definição dos **atributos**:
    - **vértices + normais + cores + coordenadas texturas**

```
tam=0.5;
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
};
```



Vertex Arrays

- **VERTEX_ARRAY** O OpenGL disponibiliza fundamentalmente as funções para o efeito:

- ***1| Habilitar***
  - glEnableClientState( )
-
- ***2| Definir / atribuir***
  - glVertexPointer( )
  - glColorPointer( )
  - glNormalPointer( )

- ***2| Desenhar***
  - glDrawElements()

Vertex Arrays

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

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

glVertexPointer(3, GL_FLOAT, 0, vertices);

vertices

X0
Y0
Z0
X1
Y1
Z1
..

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

Normais

- 2b. Definição das normais e das cores é equivalente
  - *Definidas usando um vetor*
  - *Cada valor correspondente a um vértice*

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=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,
};
```

glColorPointer(3, GL_FLOAT, 0, cor);

Vertex Arrays

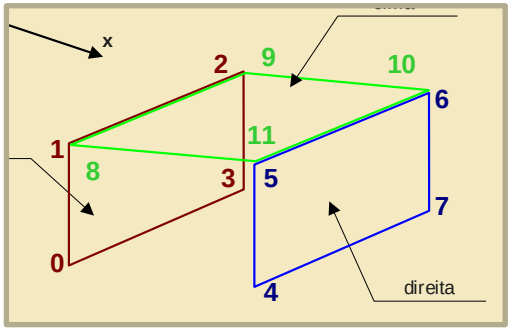
Simplificação !!!

- 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.*
- ***Usa as cores e as normais***
- ***Objecto é apenas definido por uma lista de vértices***

- ***Direita definida como***
  - static GLuint direita[ ]= {4, 7, 6, 5};

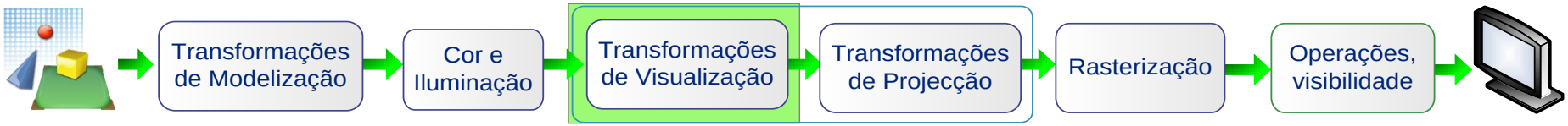


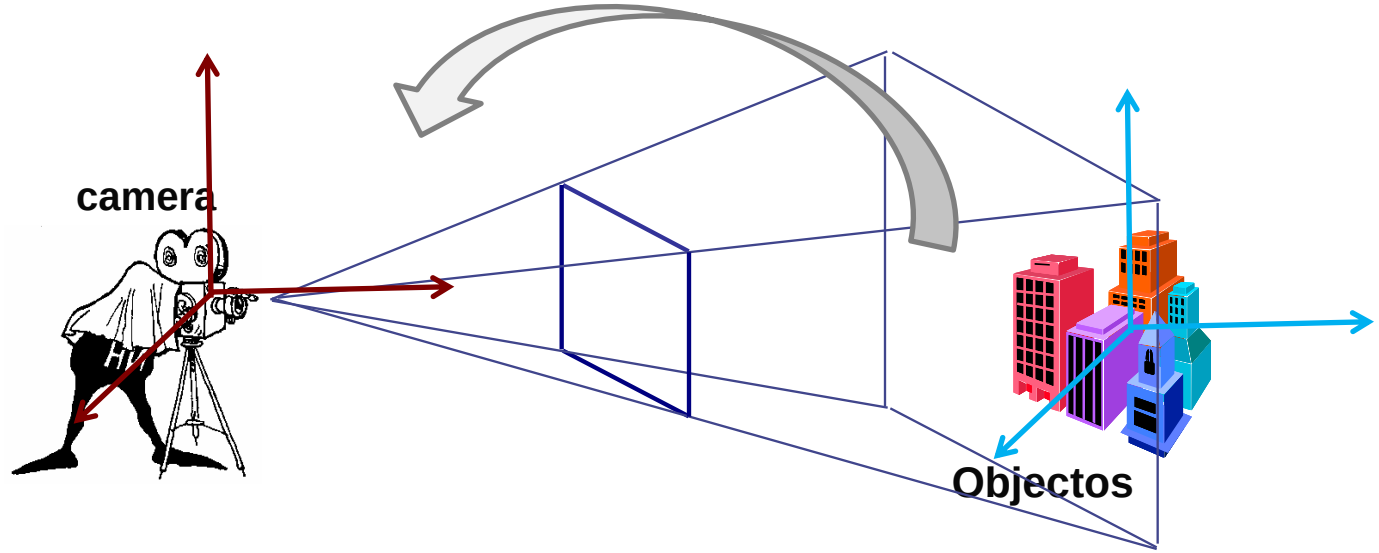
Sumário

- Faces Visiveis
- Normais

- Vertex Array

- ***LookAt**, perspective, viewport* ***(a rever na próxima aula)***

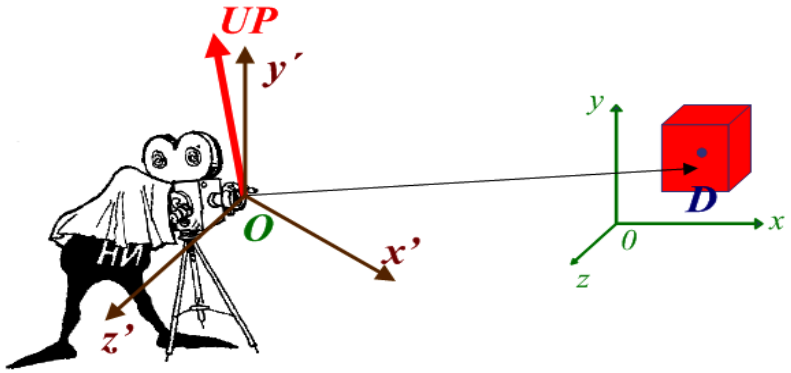

Transformações de Modelização
Cor e Iluminação
Transformações de Visualização
Transformações de Projecção
Rasterização
Operações, visibilidade


camera
Objectos

3D → 3D

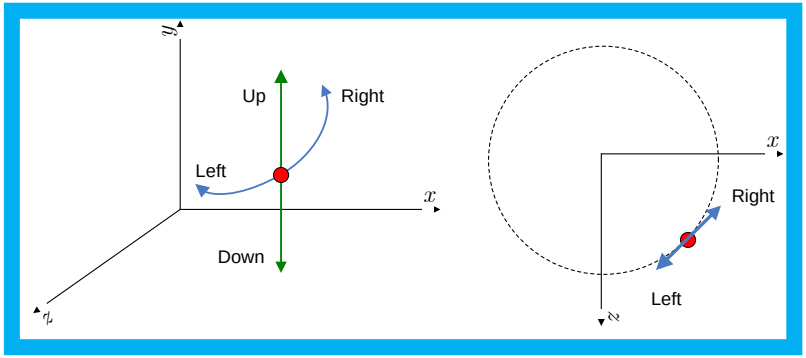
Visualização 3D

- **2. Observador**
  - Implementado !!
  - ***Voltamos aqui para a próxima aula***



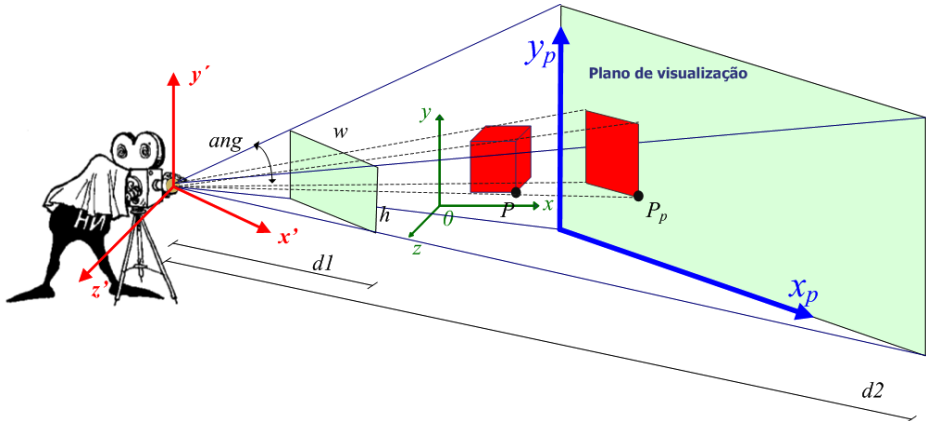
gluLookAt(Ox, Oy, Oz, Dx, Dy, Dz, UPx, UPy, UPz);

- Por agora
  - Subir/descer, girar
  - ***Uso das SETAS***



Projecção

3. Projecção perspectiva

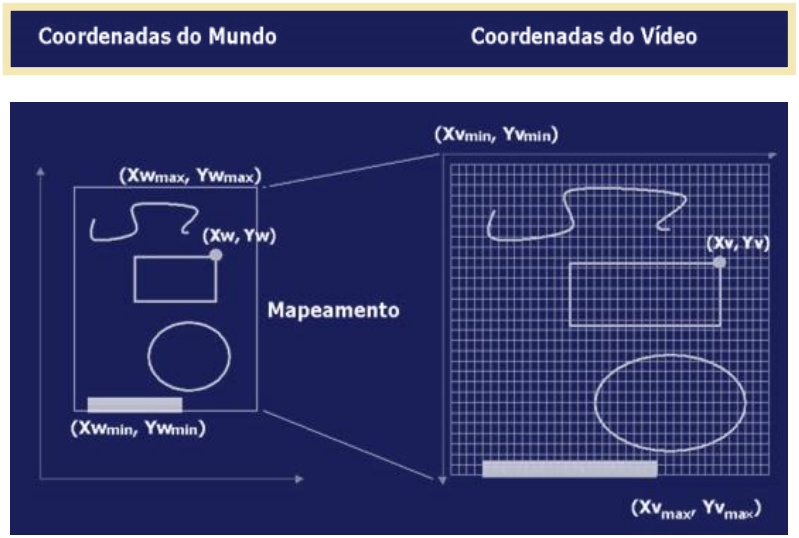


gluPerspective (***angulo***, wScreen/hScreen, d1, d2);

Projecção

- 4. Janela ecrã

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

Observador
Na posição (0bX, obY, obZ), a olhar para a origem, de pé

- **Concluindo: comandos importantes (além do vertex array)**

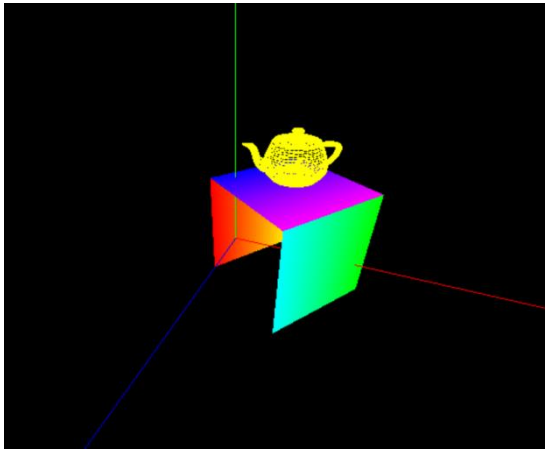
```
glEnable(GL_DEPTH_TEST);        //……Profundidade / 3D

glNormal3fv(n);                 //…… Normal

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

glutWireTeapot(tam);            //……chaleira tamanho=tam
```

CONCLUINDO: A IMPLEMENTAR



- 1. Objecto MESA

  VertexArray - atributos: posição, cor, normal

  Faces visíveis

  - Tecla 'F'

- 2. Transformações / **animação**
  - Teclas 'R' sempre a rodar
  - Teclas 'E/D' rodar esquerda / direita
  - Teclas 'A/S' andar esquerda/direita (eixo x)

- 3. Observador (**FEITO!**)
  - Teclas '←↑→↓' esquerda/cima/direita/baixo