Departamento de Engenharia Informática

TP3 - Visualização 3D

DEI – CG

2019/20

Transformações Geométricas (Pipeline)

- 1. Definição ***objectos*** e ***transformações*** (rotação, translação, escala, …)
- ~~2. Definição de modelos de luz e **cor**~~

Esquecer a cor

- 3. Visualização / Projecção
  - 3.1 Localização e orientação do ***observador***
    - Coordenad1as dos objectos em função das coordenadas do observador
  - **3.2 Volume de visualização**
    - Tipo de ***projecção*** (perspectiva/paralela)

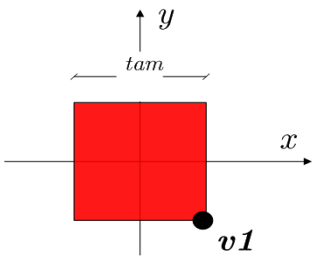
Coordenadas !

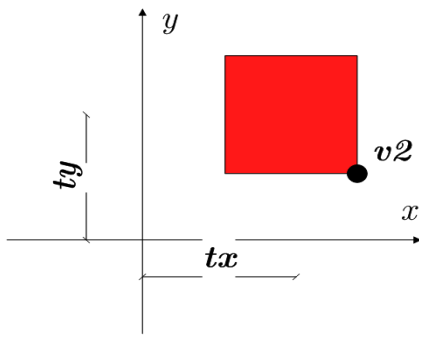
- 4. ***ViewPort***

Transformações Geométricas

1. Definição **Objectos** e transformações (rotação, translação, escala, …)

```
glTranslatef(tx, ty, 0);                    // transformações
glColor4f(VERMELHO);                        // Cores
glBegin(GL_POLYGON);                        // primitivas (objectos)
        glVertex3i(  0,   0, 0);
        glVertex3i(tam,   0, 0);
        glVertex3i(tam, tam, 0);
        glVertex3i(0, tam, 0);
glEnd();
```



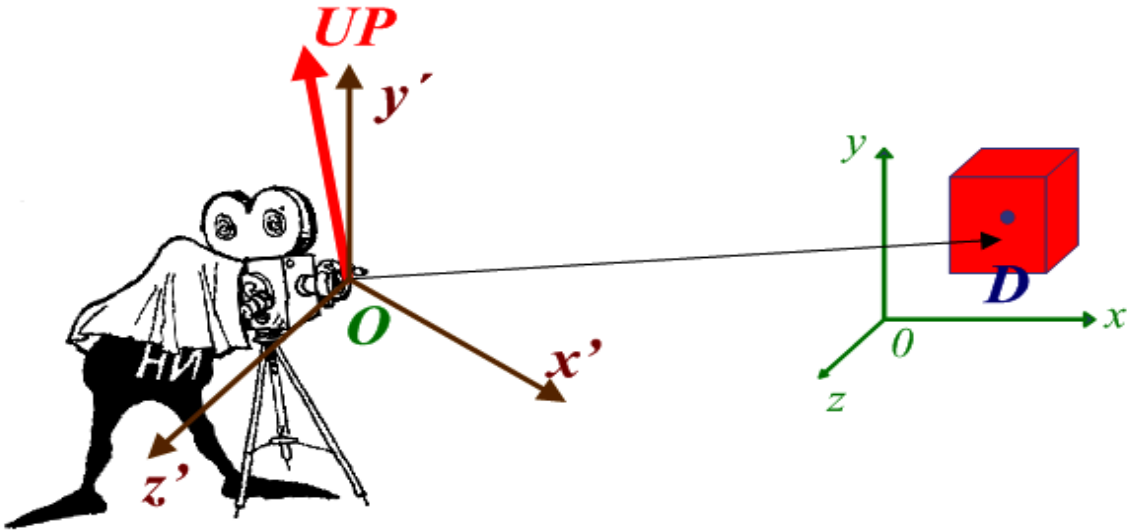


$$\begin{bmatrix} x2 \\ y2 \\ z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & tx \\ 0 & 1 & 0 & ty \\ 0 & 0 & 1 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1 \\ y1 \\ z1 \\ 1 \end{bmatrix}$$

Transformações Geométricas

- 2. Transformações de Visualização (***Definir a localização do observador e sua orientação***).

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

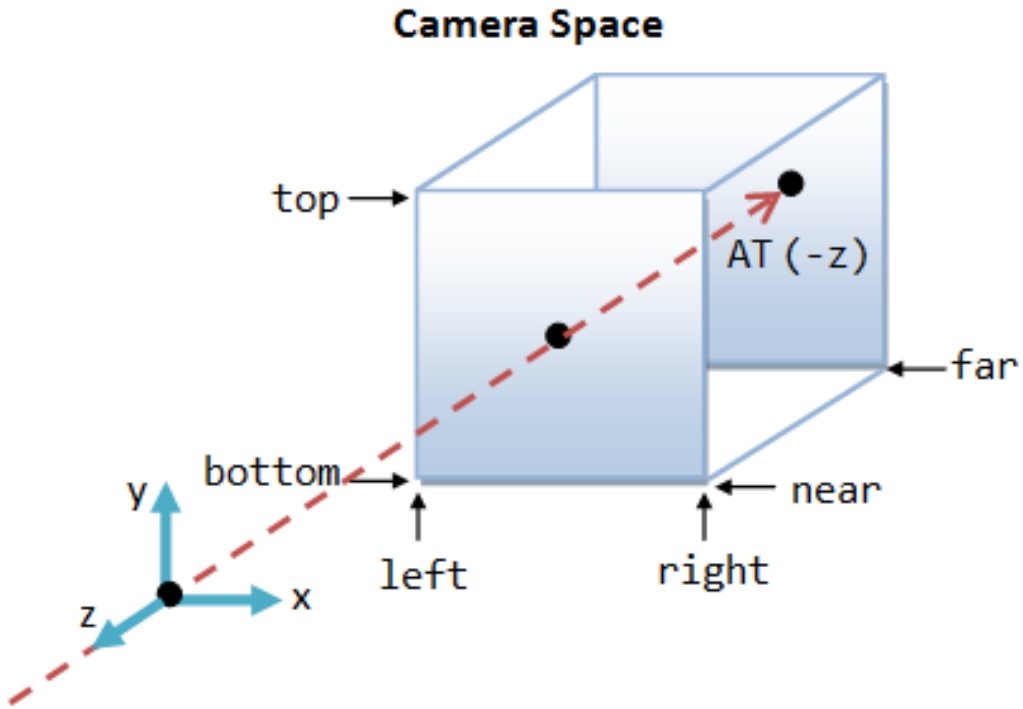


$$\begin{bmatrix} Px' \\ Py' \\ Pz' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{1x} & R_{2x} & R_{3x} & Rx \square Q \\ R_{1y} & R_{2y} & R_{3y} & Ry \square Q \\ R_{1z} & R_{2z} & R_{3z} & Rz \square Q \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Px \\ Py \\ Pz \\ 1 \end{bmatrix}$$

Transformações Geométricas

- 3. Projecção e Volume de Visualização

glOrtho (left, right, bottom, top, near, far);



Transformações Geométricas

- 3. Projecção e Volume de Visualização

Nota: fez-se, nos trabalhos anteriores

- glOrtho2D(-5, 5, -5, 5)
- [xmin, xmax, ymin, ymax]= (-5, 5, -5, 5)

- Observador origem olhar –z
- Volume de visualização (-5,5,-5,5)

Eixos XY (2D)

Transformações Geométricas

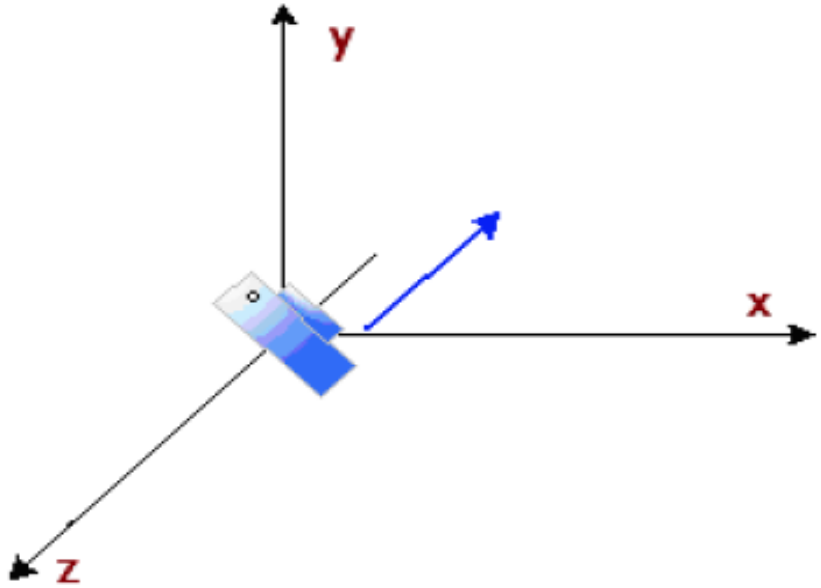
3. Projecção e Volume de Visualização

- Por Omissão

  glOrtho (-1, 1, -1, 1, -1, 1);

Observador:

- **na origem (0,0,0)**
- **A olhar no sentido contrário ao eixo Z**

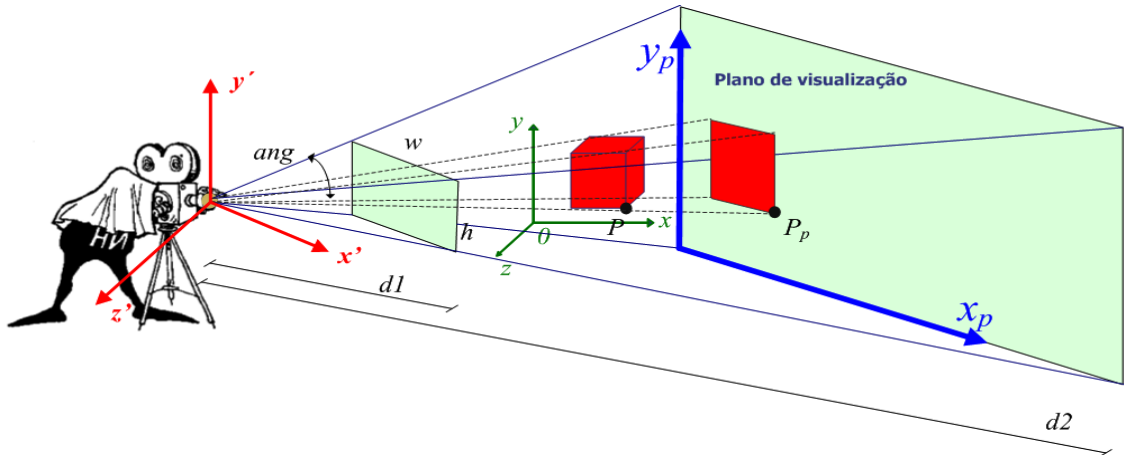


Transformações Geométricas

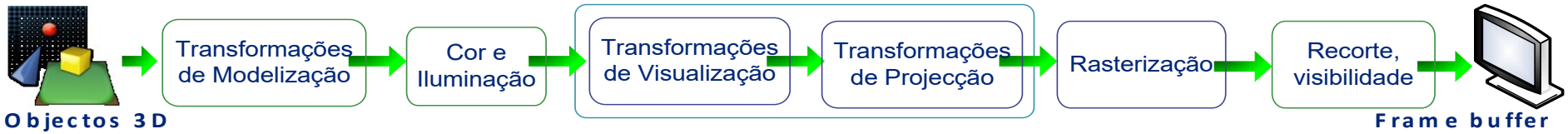
- 3. Projecção e Volume de Visualização

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

*d1,d2 - distâncias (***sempre positivos***)*

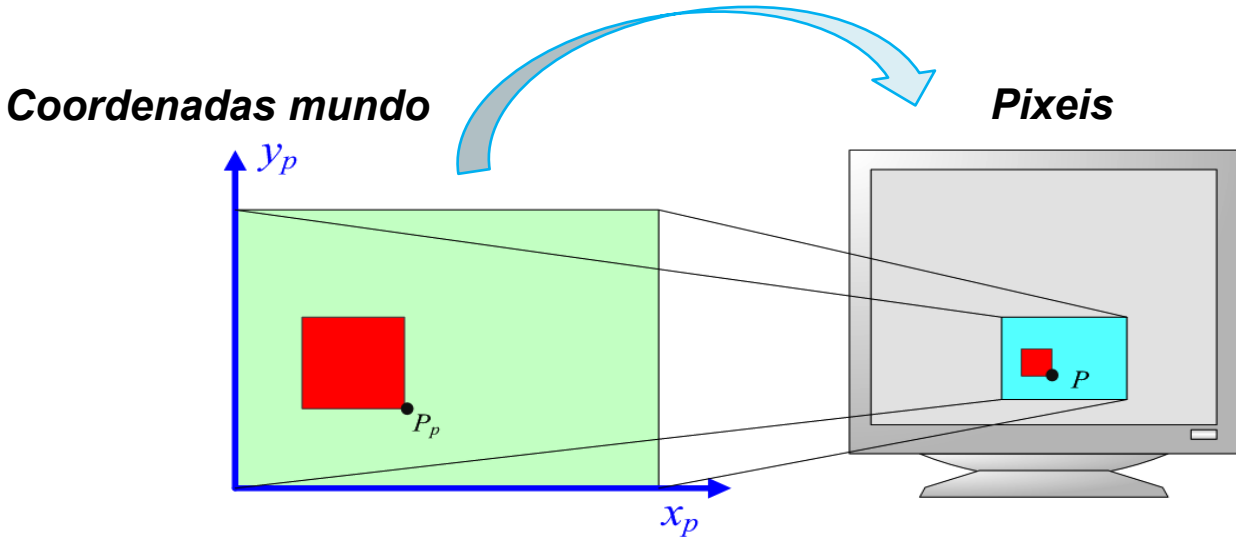


Transformações Geométricas

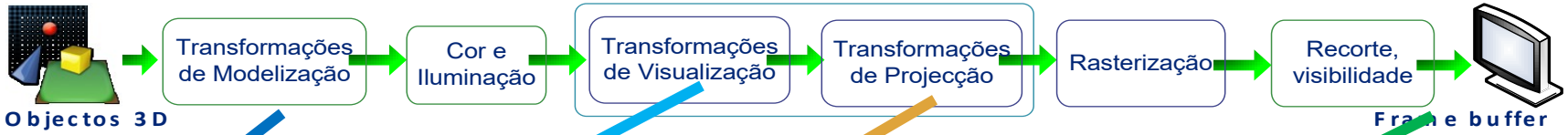


- *4. ViewPort*

glViewport (Xo, Yo, wScreen, hScreen); **//pixeis !!**



Transformações Geométricas



Ordem Conceptual:

1. **Definir a cena** para ser fotografada, incluindo a especificação dos objectos
   Equivalente à etapa de modelização

2. **Posicionar a câmara** para fotografar a cena
   Equivalente a especificar as transformações de visualização

3. **Seleccionar a lente da câmara** ou ajustar o zoom
   Equivalente a especificar as transformações de projecção geométricas

4. **Determinar o tamanho final da foto** (maior ou menor)
   Equivalente a especificar a janela de visualização (viewport)

Transformações Geométricas

	glViewport (0,0,wScreen, hScreen);	Janela Visualização	M1

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-dX, dX, -dY, d\Y, -dZ, dZ);
		Projecção	M2=**Projection**

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(Ox,Oy,Oz, Dx,Dy,Dz, UPx,UPy,UPz);
		View (observador)	M3a

	glTranslatef(tx, ty,0);
	glColor4f(VERMELHO);
	glBegin(GL_POLYGON);
	glVertex3i(0, 0, 0);
	glVertex3i(tam, 0, 0);
	glVertex3i(tam, tam, 0);
	glVertex3i(0, tam, 0);
	glEnd();
		Model (objectos)	M3b
			
			M3=**ModelView**

Matrizes ModelView e Projection

- **PROJECTION**
  - Definição do modelo de perspectiva / volume de visualização.

- **MODELVIEW**
  - Posição e orientação os ***objectos*** na cena de visualização (**MOD**elos)
  - Localização e orientação do ***observador*** **/*camera*** **(VIEW)**

- Para especificar qual a matriz em operação no momento o OpenGL disponibiliza os comandos:

  glMatrixMode (GL_PROJECTION)

  glMatrixMode (GL_MODELVIEW)

- Para permitir que a transformação actual seja re-inicializada existe o comando

  glLoadIdentity().

openGL

Atenção: *outros comandos*

1. Activar buffer de profundidade

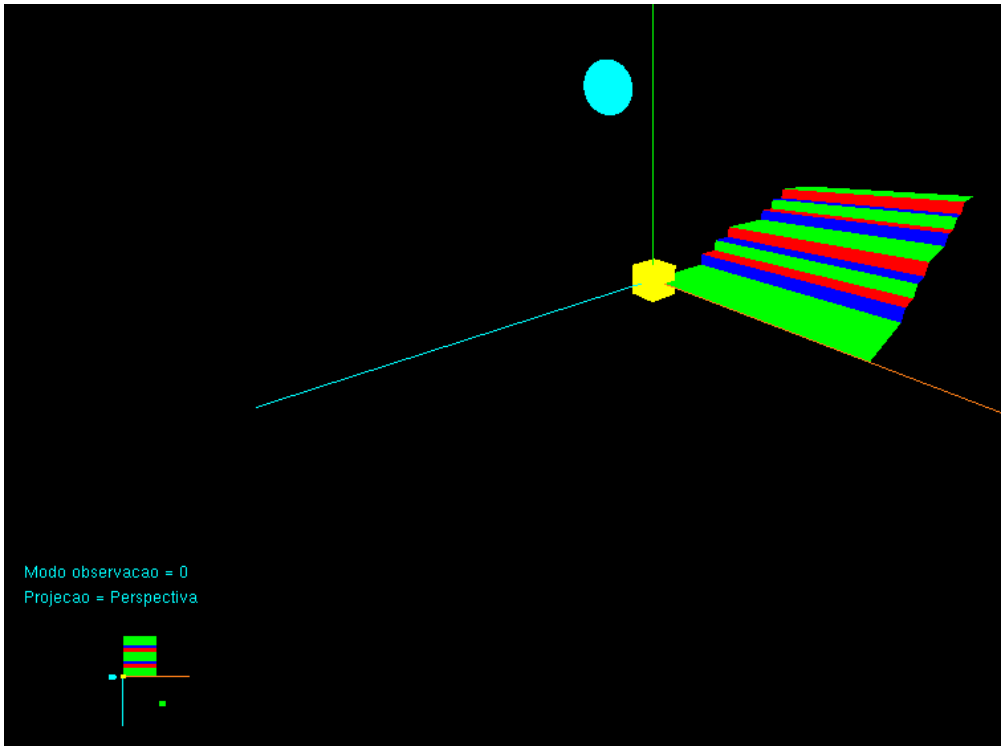
```
glClear ( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
```

2. Cada vez que desenha "limpa" buffer de profundidade

```
glutInitDisplayMode (GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH );
```

Visualização 3D

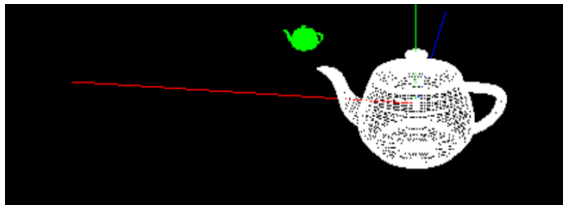
- Trabalho
  - **1. Objectos**
  - 2. Observador
  - **3**. Projecções
  - **4.** Janela visualização



Visualização 3D

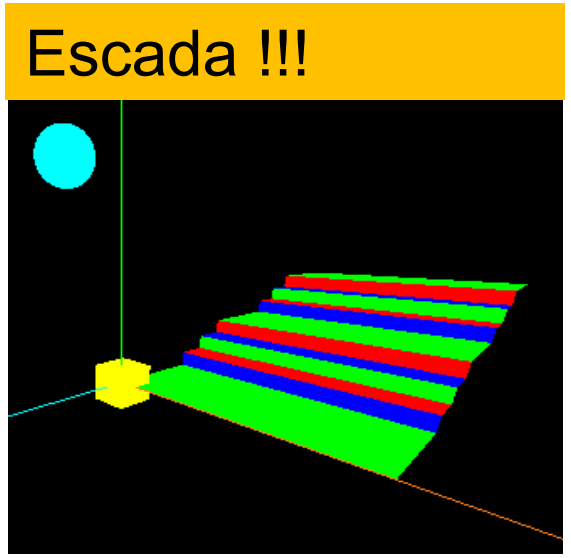
1. Objectos

- **GLUT**



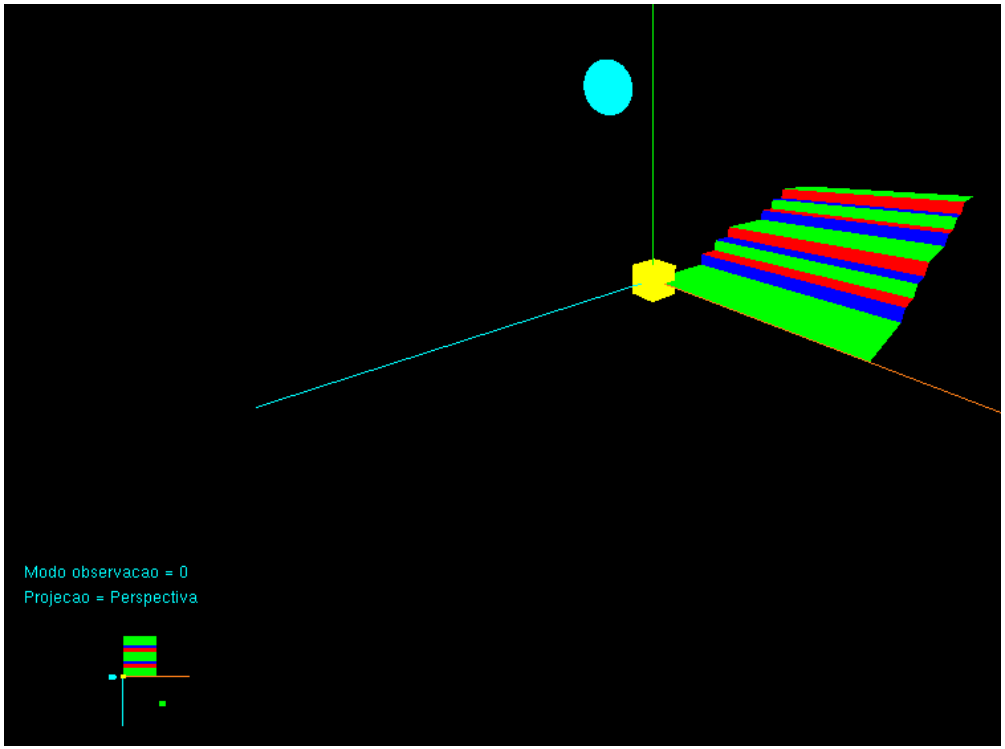
- **glutSolidCube, glutWireCube**
- **glutSolidTeapot, glutWireTeapot**

- glutSolidSphere, glutWireSphere
- glutSolidCone, glutWireCone
- glutSolidTorus, glutWireTorus
- glutSolidDodecahedron, glutWireDodecahedron
- glutSolidOctahedron, glutWireOctahedron
- glutSolidTetrahedron, glutWireTetrahedron
- glutSolidIcosahedron, glutWireIcosahedron

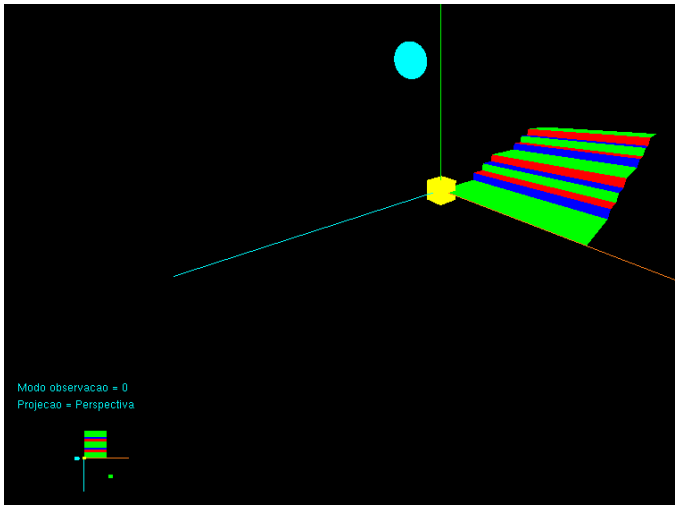


Visualização 3D

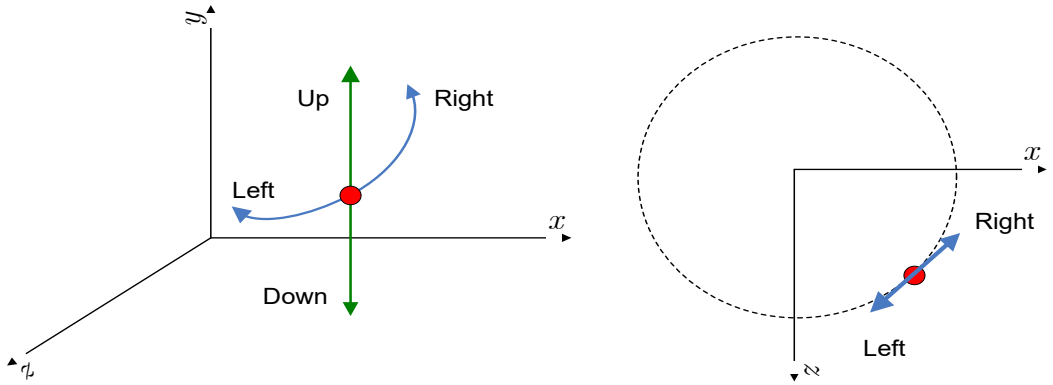
- Trabalho
  - 1. Objectos
  - **2. Observador**
  - **3**. Projecções
  - **4.** Janela visualização



Visualização 3D



- **Aula anterior – olha para a origem (cubo amarelo)**
  - Subir/descer
  - Girar



Observador/camera

- **Para a aula de hoje (visualização)**

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

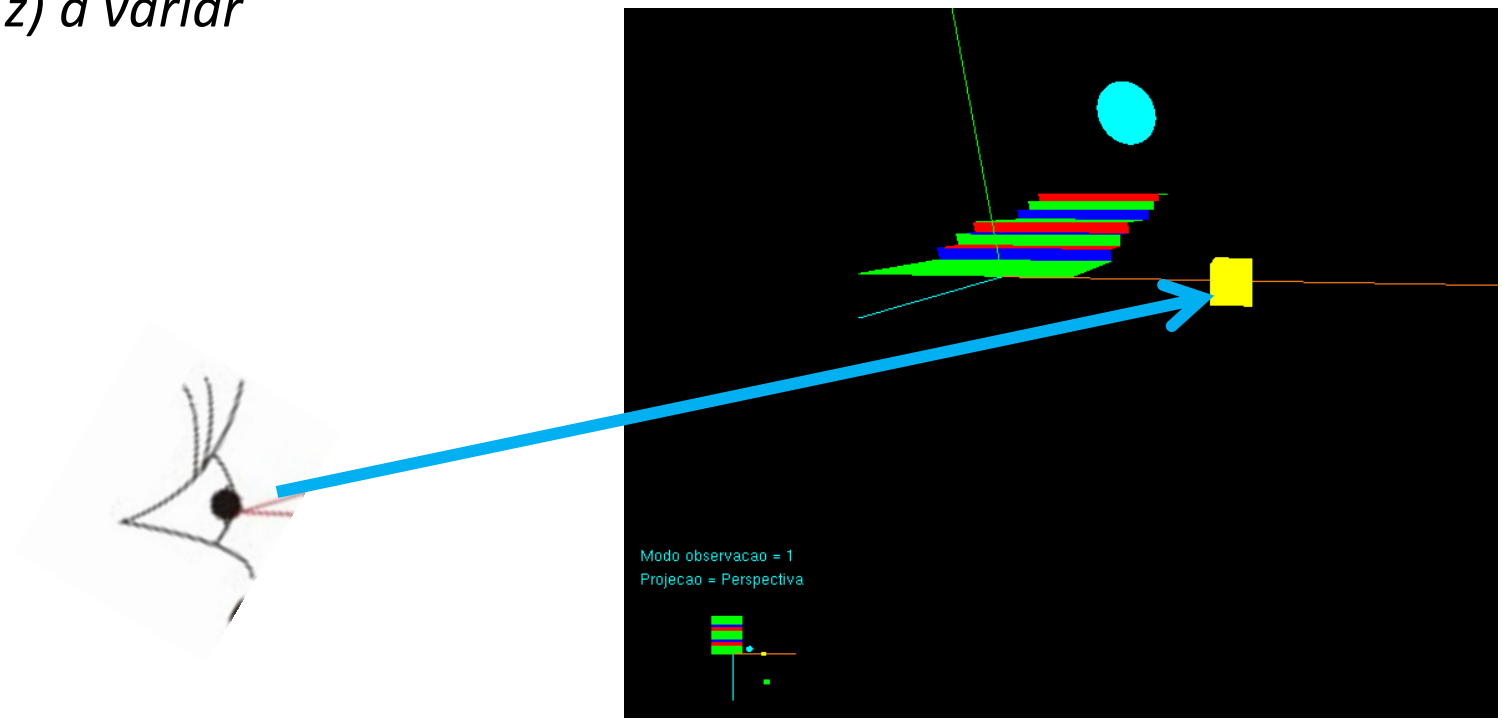
  - **Depende do objectivo !!!**
  - **Pode/há mais que uma alternativa!**
    - Na aula anterior move-se, mas olha sempre para o mesmo local (origem)
    - Por exemplo, pode estar numa posição fixa sempre a olhar para um objecto.
    - Pode andar no “meio dos objectos”

Modo 1

- **Observador fixo – foco móvel**
- Cubo amarelo desloca-se no plano horizontal

gluLookAt(Ox, Oy, Oz, Tx, 0, Tz, UPx, UPy, UPz)

(Tx, 0, Tz) a variar

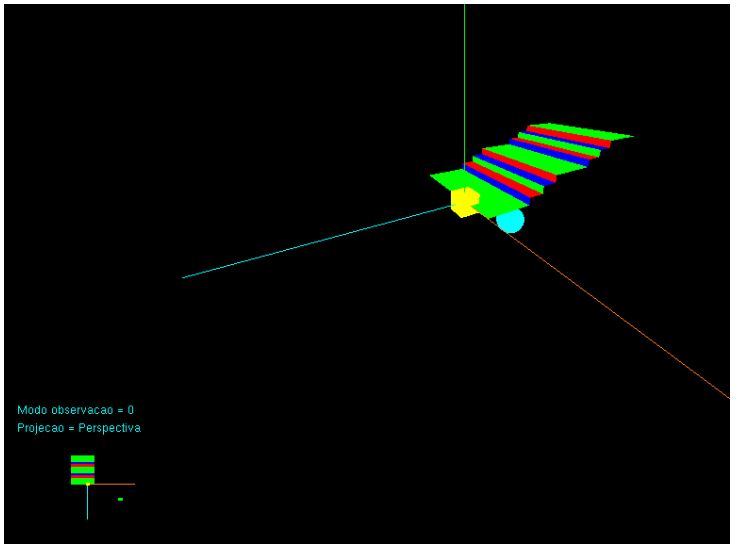


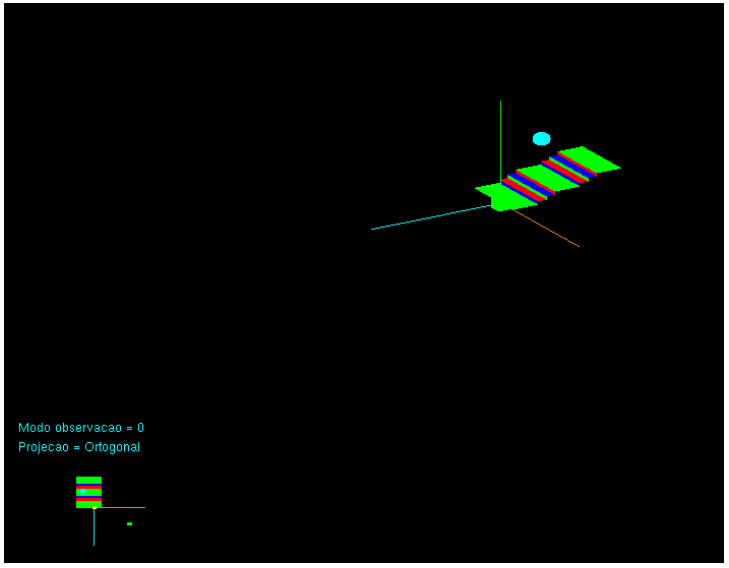
Visualização 3D

- Trabalho
  - 1. Objectos
  - 2. Observador
  - **3. Projecções**
  - 4. Janela visualização

Visualização 3D

- **3. Projecções**
  - Ortogonal
  - Perspectiva



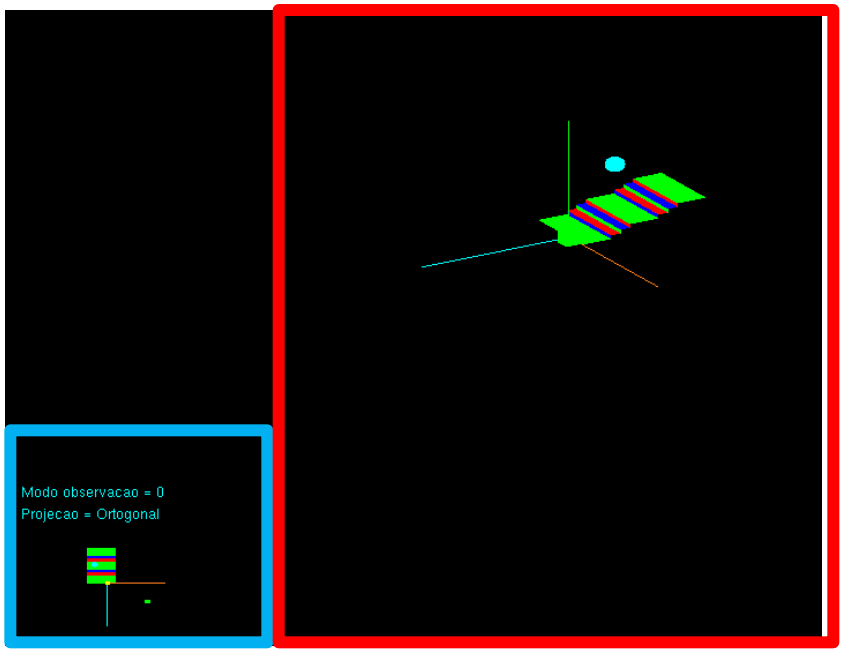


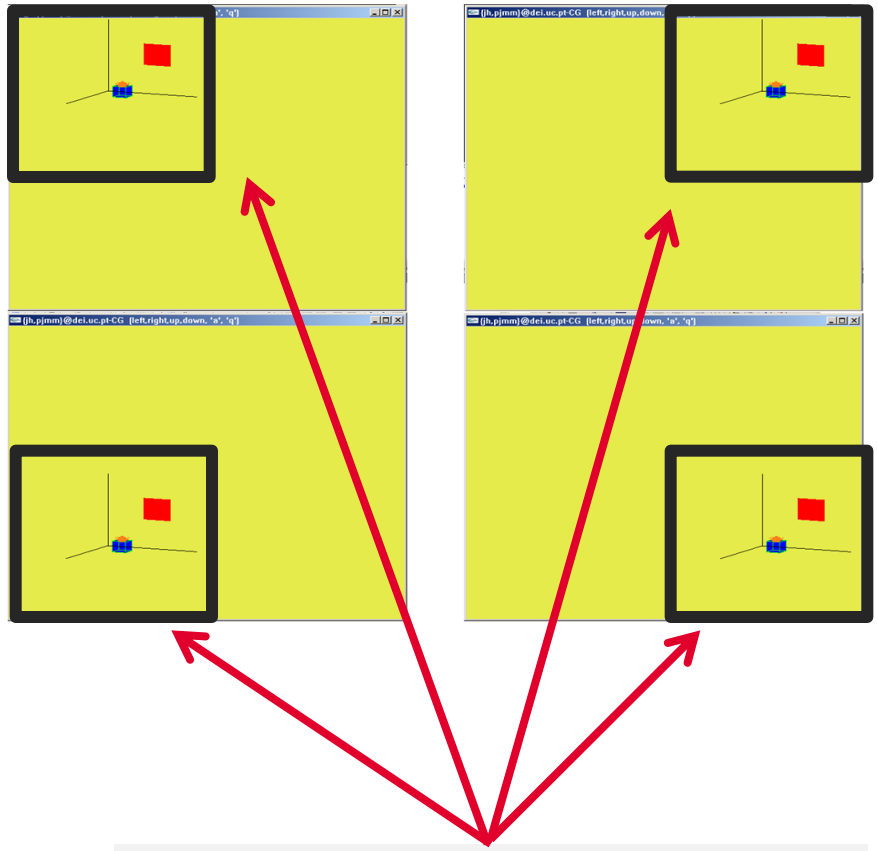
Visualização 3D

- Trabalho
  - 1. Objectos
  - 2. Observador
  - 3. Projecções
  - **4. Janela visualização**

Visualização 3D

- **4. Janela visualização**
  - ***glViewport (Xo, Yo, wScreen, hScreen);***





4 possíveis localizações da cena na janela de visualização

4 possíveis viewports

Trabalho

1- Dois viewports

glViewport (Xo, Yo, wScreen, hScreen);
Admitindo janela de dimensão 800,600

1.1- O "mapa"

- Projecao ortogonal

```
…
glViewport(0,0,100,100)
glOrtho(…)
Lookat( "vista de cima" )
desenhar cena
```

1.2- A cena

- Comutar projecção ortogonal e perspectiva
- Poder fazer zoom do local onde está a olhar

```
glViewport(100,100, 700, 500)
glPerspective(…)
Lookat(olhar naturalmente para os objectos)
desenhar cena
…
```

Transformações Geométricas

	glViewport (0,0,wScreen, hScreen);	Janela Visualização	M1

	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(-dX, dX, -dY, d\Y, -dZ, dZ);
		Projecção	M2=**Projection**

	glMatrixMode(GL_MODELVIEW);
	glLoadIdentity();
	gluLookAt(Ox,Oy,Oz, Dx,Dy,Dz, UPx,UPy,UPz);
		View (observador)	M3a

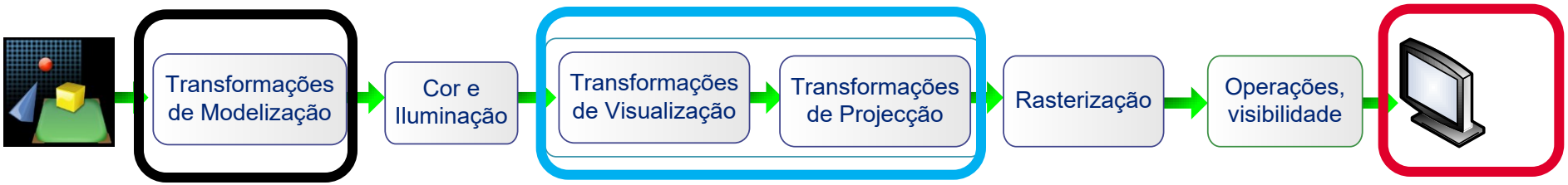
	glTranslatef(tx, ty,0);
	glColor4f(VERMELHO);
	glBegin(GL_POLYGON);
	glVertex3i(0, 0, 0);
	glVertex3i(tam, 0, 0);
	glVertex3i(tam, tam, 0);
	glVertex3i(0, tam, 0);
	glEnd();
		Model (objectos)	M3b
			
			M3=**ModelView**

2. Zoom

perspective(angulo, razao, d1, d2);

Admitindo o observador fixo que acontece se variar o anglo ?

Concluindo: até agora **COORDENADAS**



- Transformadas modelização/geométricas –

- **MESA a deslocar-se (T + R + S) ???**

Observador a movimentar-se (?)
Uso de projeccao ortogonal e perspectiva (?)
Uso de viewports (?)

)