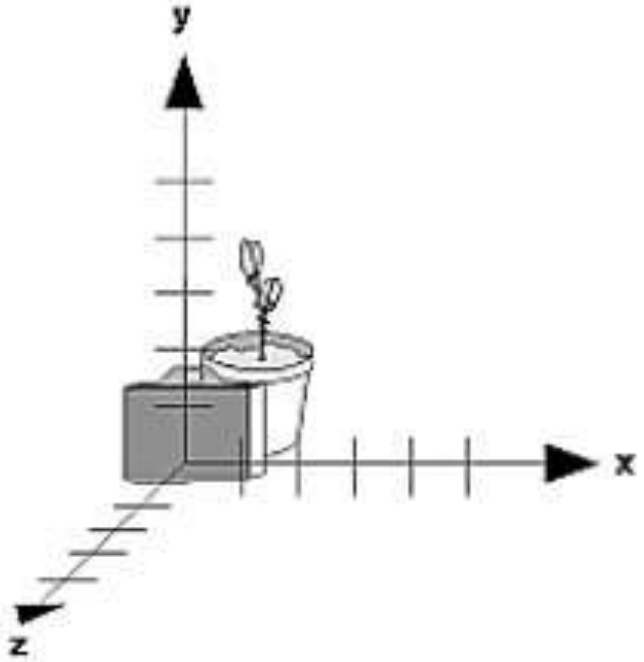
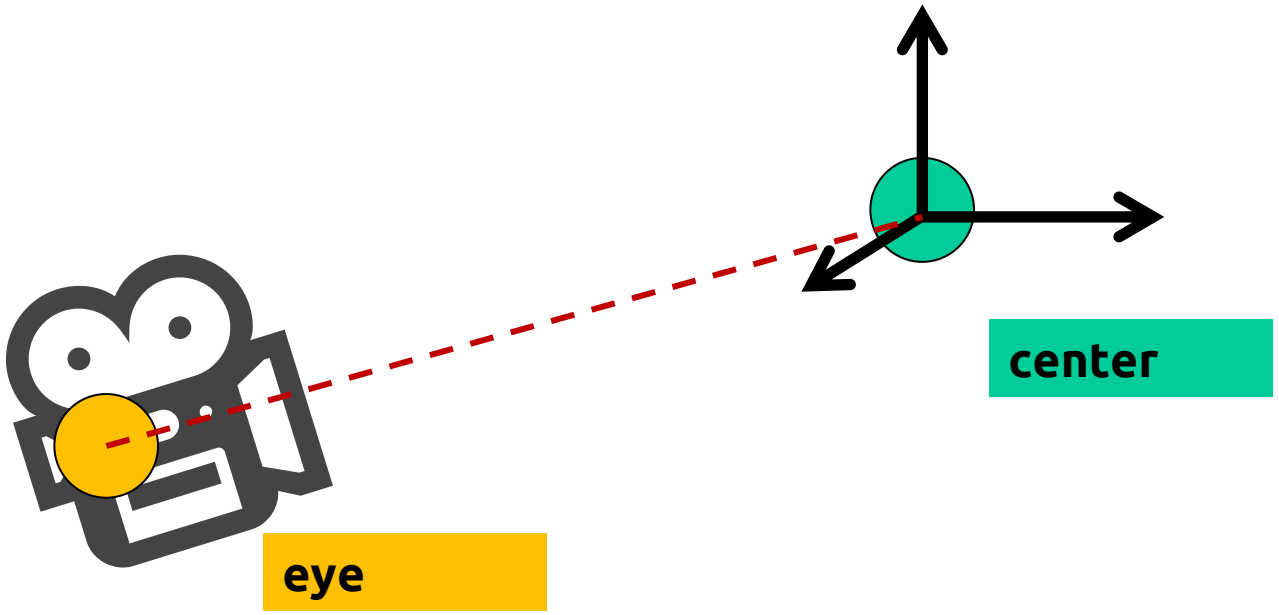
Posição da câmara

- By default, the camera is placed at the origin (0, 0, 0) directed to the negative zz axis



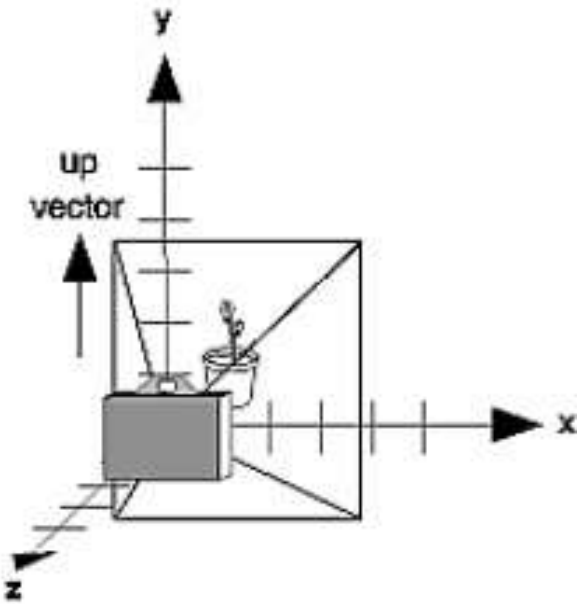
Posicionamento da câmara

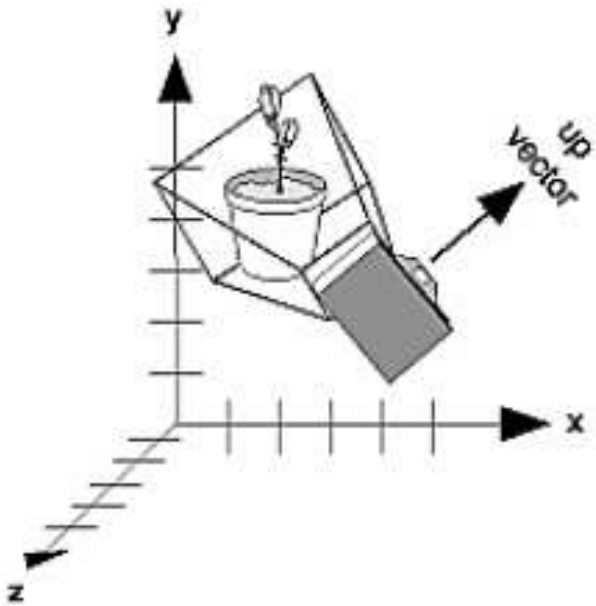
- `void gluLookAt(`
  `eyex, eyey, eyez,`
  `centerx, centery, centerz,`
  `upx, upy, upz)`



Posicionamento da câmara

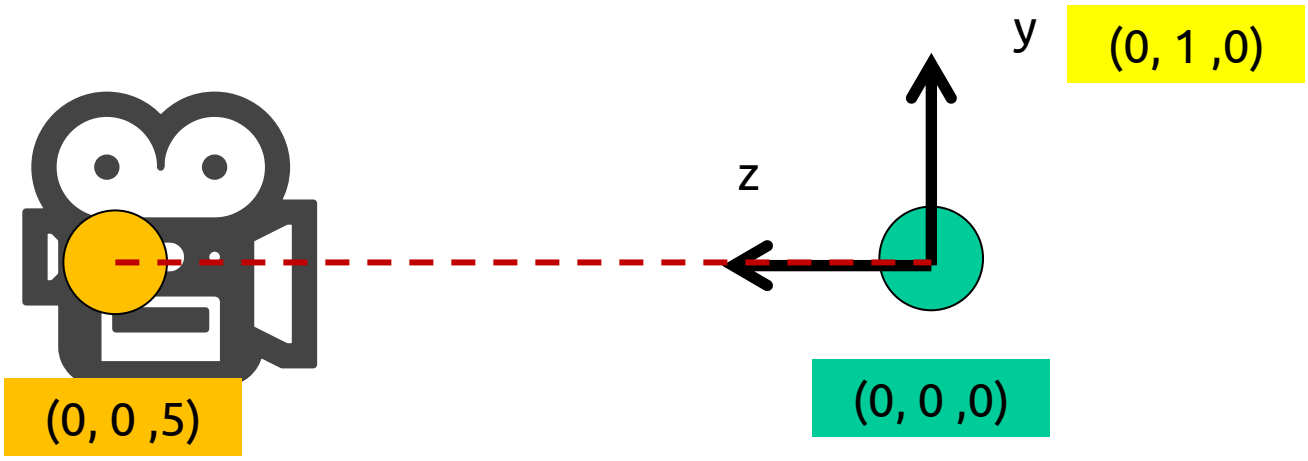
- `void gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)`





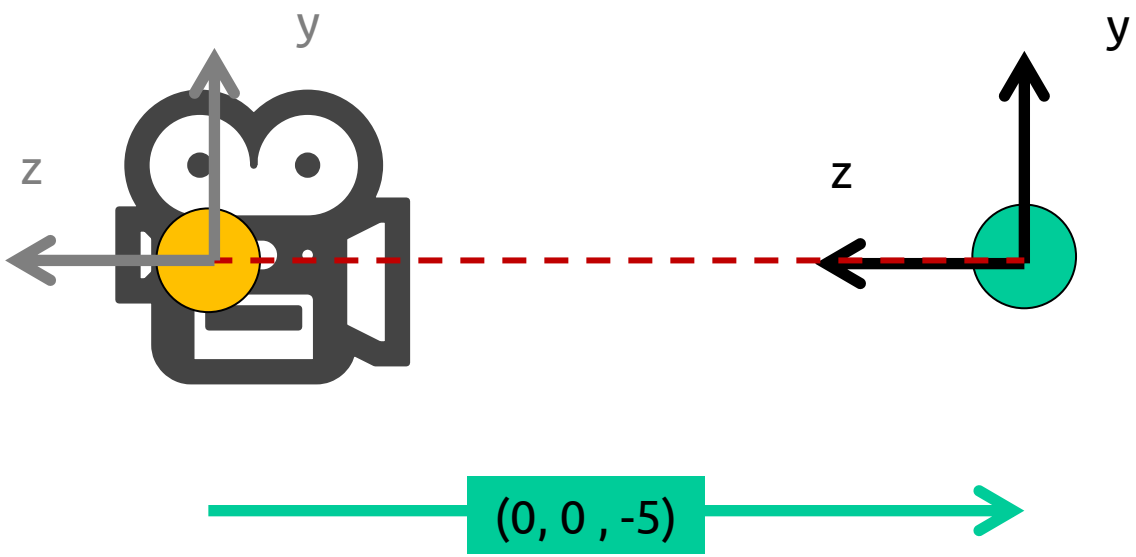
Posição da câmara

- Moving the camera or moving the scene has the same result
  - **`gluLookAt(0,0,+5, 0,0,0, 0,1,0)`**
  - `glTranslatef(0, 0, -5)`



Posição da câmara

- Moving the camera or moving the scene has the same result
  - `gluLookAt(0,0,+5, 0,0,0, 0,1,0)`
  - **`glTranslatef(0, 0, -5)`**

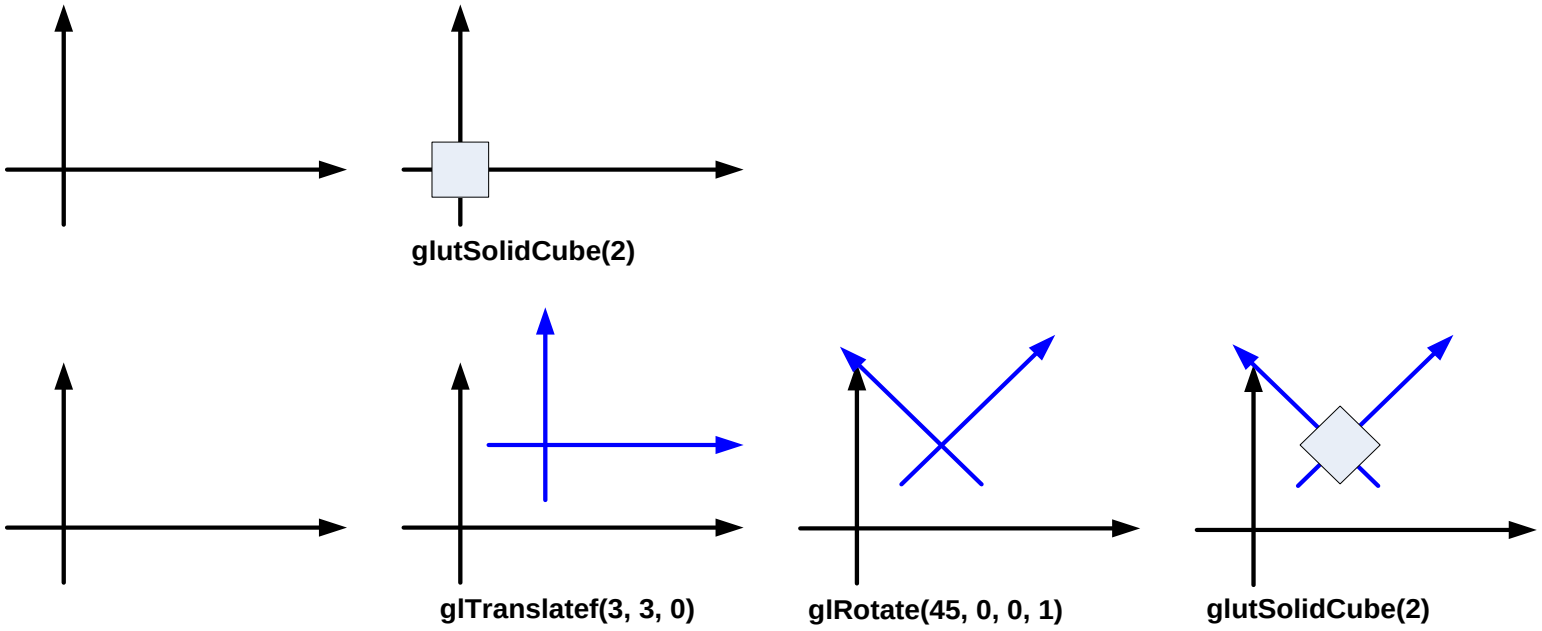


Posição da câmara

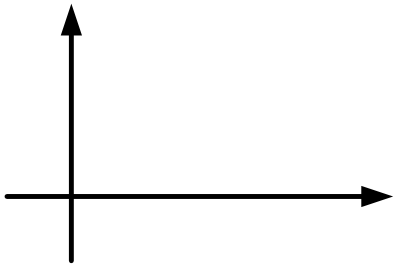
- You can use gluLookAt or build your own viewing routine, for example, polar motion camera using basic transformation operations

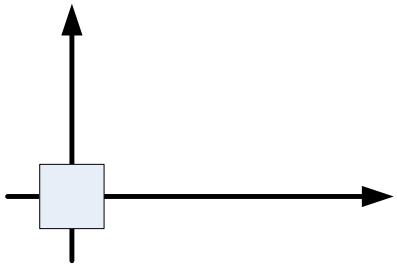
O sistema de coordenadas local

- When applying a transformation, we are actually moving the coordinate system “attached” to the model

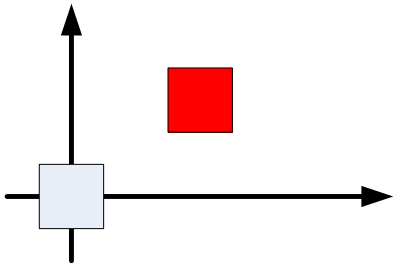


O sistema de coordenadas local

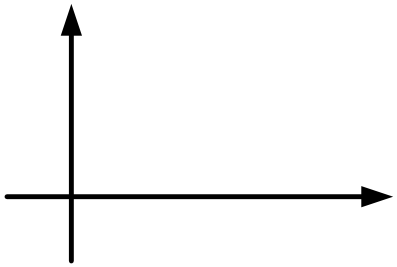


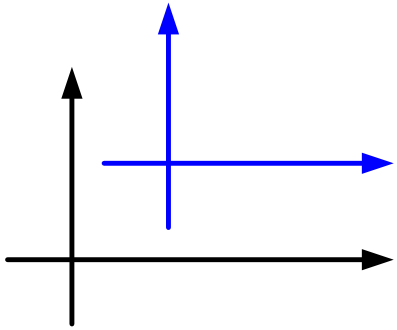


glutSolidCube(2)

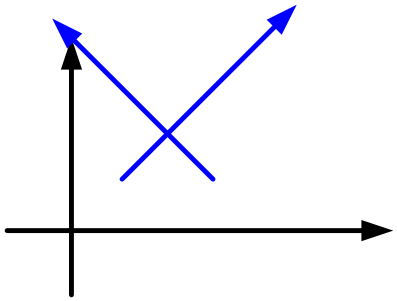


glRect(3, 2, 0, 5, 4, 0)

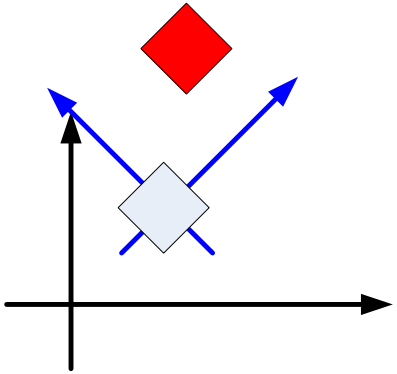




glTranslatef(3, 3, 0)



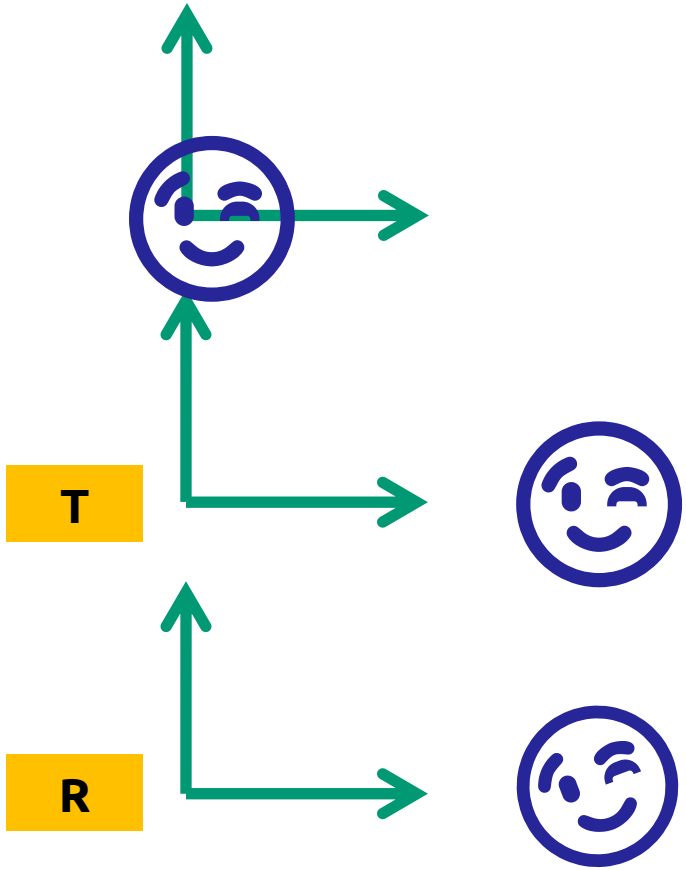
glRotate(45, 0, 0, 1)



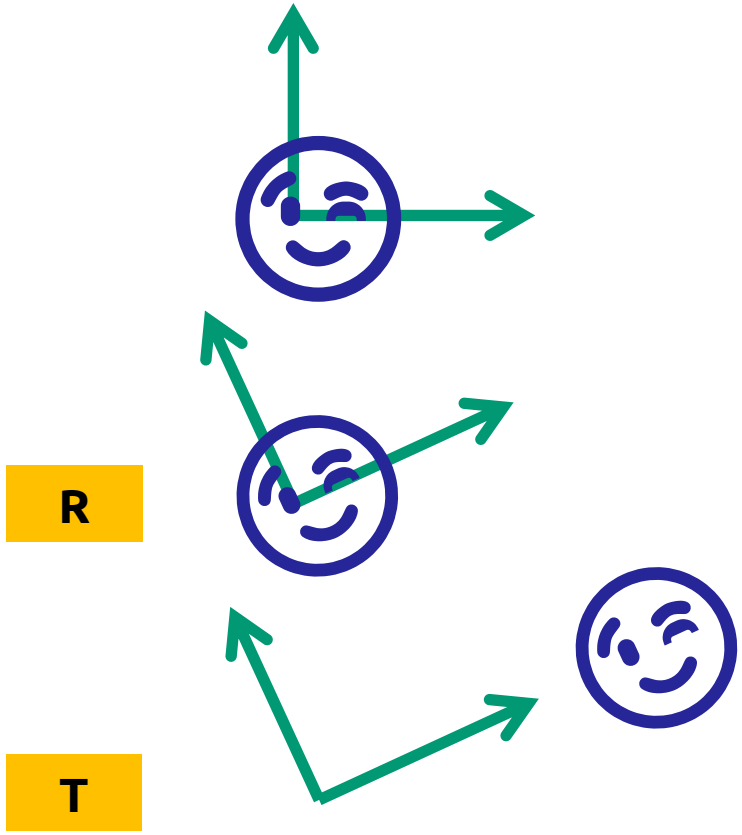
glutSolidCube(2)
glRect(3, 2, 0, 5, 4, 0)

Efeito cumulativo de transformações

- Translação + Rotação

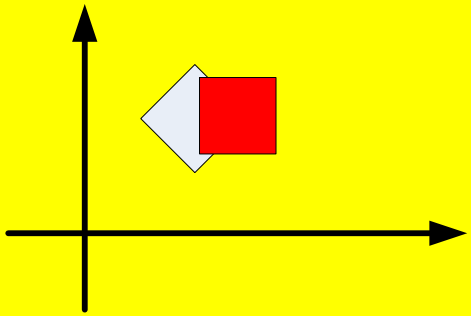


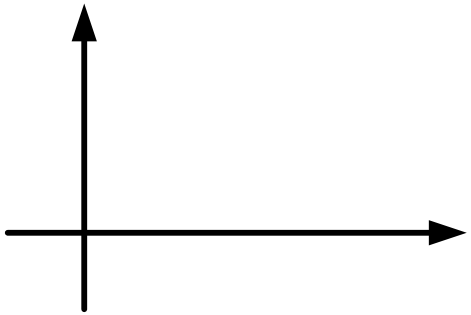
- Rotação + Translação



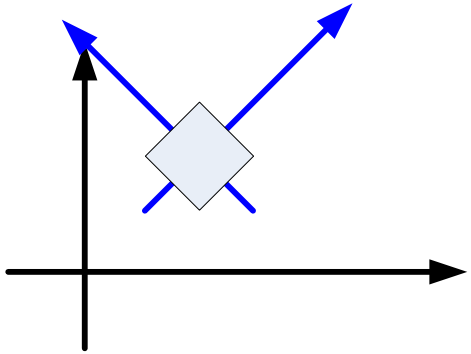
PushMatrix + PopMatrix

Final scene

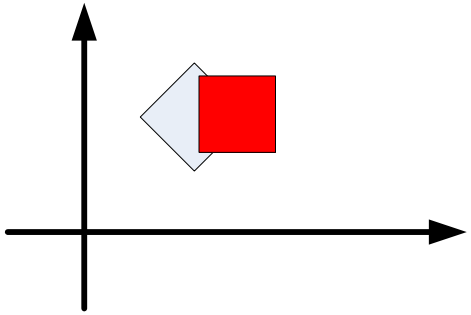




glLoadIdentity()



glPushMatrix()
  glTranslatef(3, 3, 0)
  glRotate(45, 0, 0, 1)
  glutSolidCube(2)
glPopMatrix()

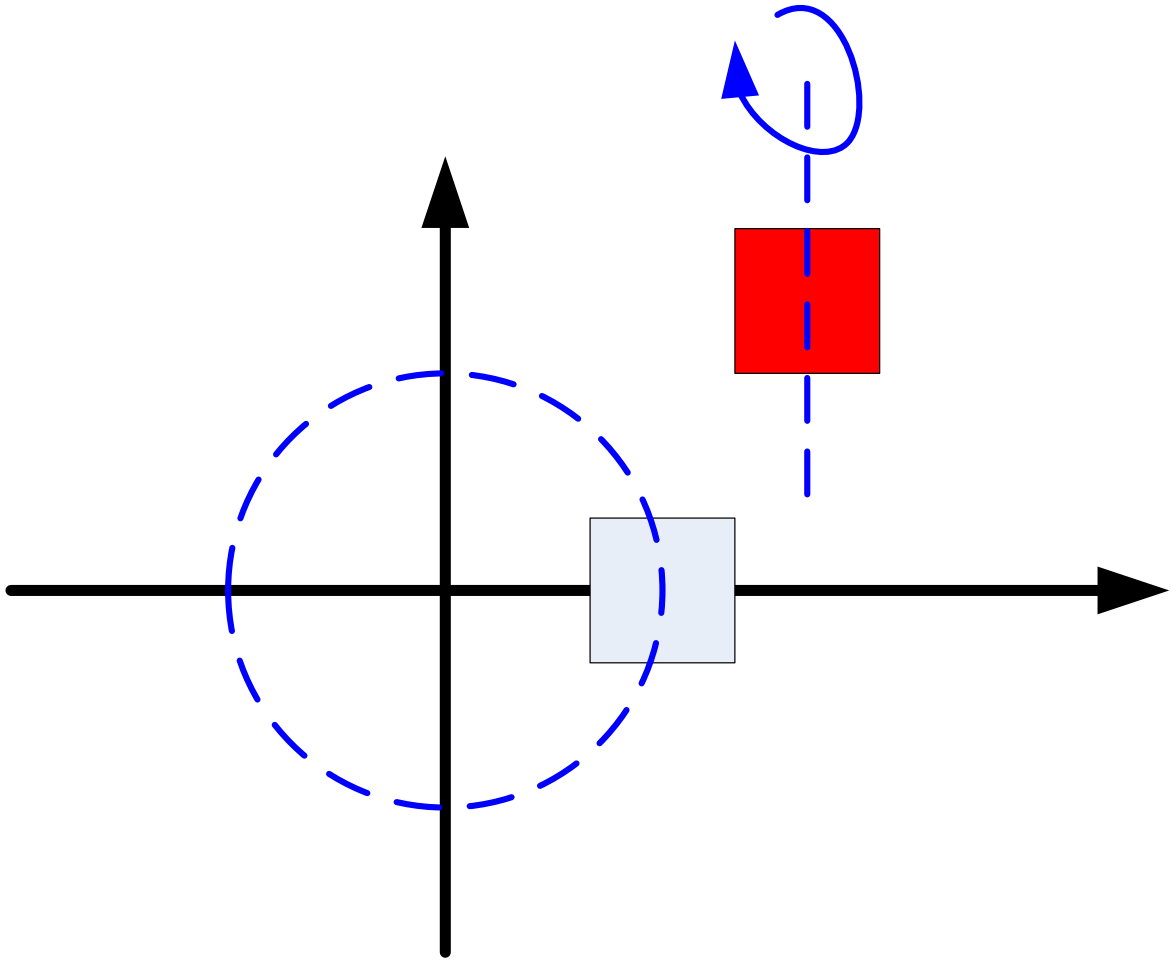


glRect(3, 2, 0, 5, 4, 0)

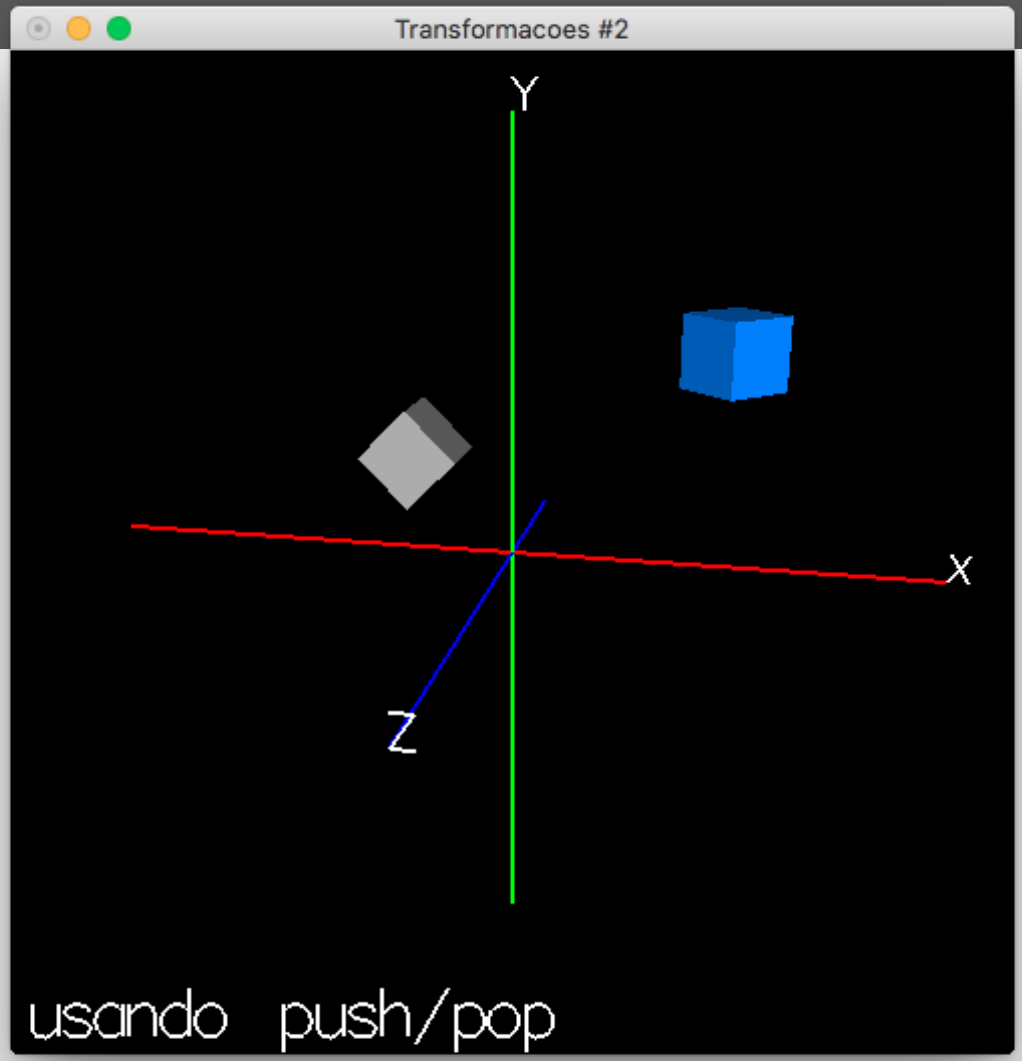
Transformações "locais"

- Save the transformation matrix using glPushMatrix ()
- Retrieve the previous matrix using glPopMatrix ()
- glPushMatrix and glPopMatrix can be used for the projection matrix or for the model/view matrix
  - You should be careful of which matrix is currently active

Transformações (in)dependentes



Demo



Texto em GLUT

- Draw a character
  - glutStrokeCharacter(*fonte*, *caracter*)
  - glutBitmapCharacter(*fonte*, *caracter*)
- Fonts
  - GLUT_STROKE_ROMAN
  - GLUT_STROKE_MONO_ROMAN
  - GLUT_BITMAP_9_BY_15
  - GLUT_BITMAP_8_BY_13
  - GLUT_BITMAP_TIMES_ROMAN_10
  - GLUT_BITMAP_TIMES_ROMAN_24
  - GLUT_BITMAP_HELVETICA_10
  - GLUT_BITMAP_HELVETICA_12
  - GLUT_BITMAP_HELVETICA_18

Módulo 8

Sistemas Gráficos e Interação

Instituto Superior de Engenharia do Porto

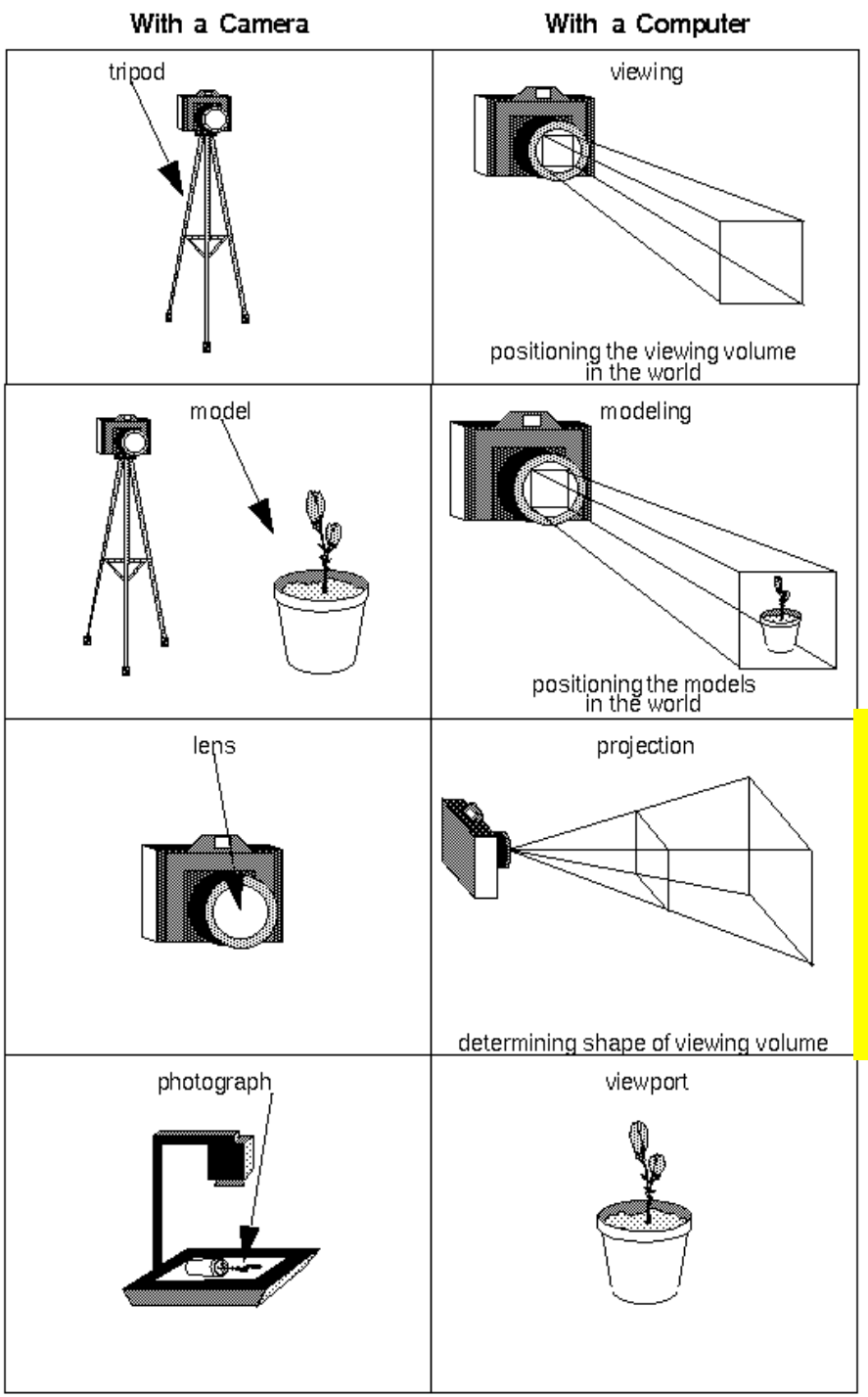
Filipe Pacheco

ffp@isep.ipp.pt

Projeções

Relembrando...

1. Apontar a câmara à cena (viewing transformation).
2. Compor a cena (modeling transformation).
3. **Escolher o tipo de lente e acertar o zoom (projection transformation).**
4. Determinar o tamanho físico da cena (viewport transformation).



Esqueleto de código

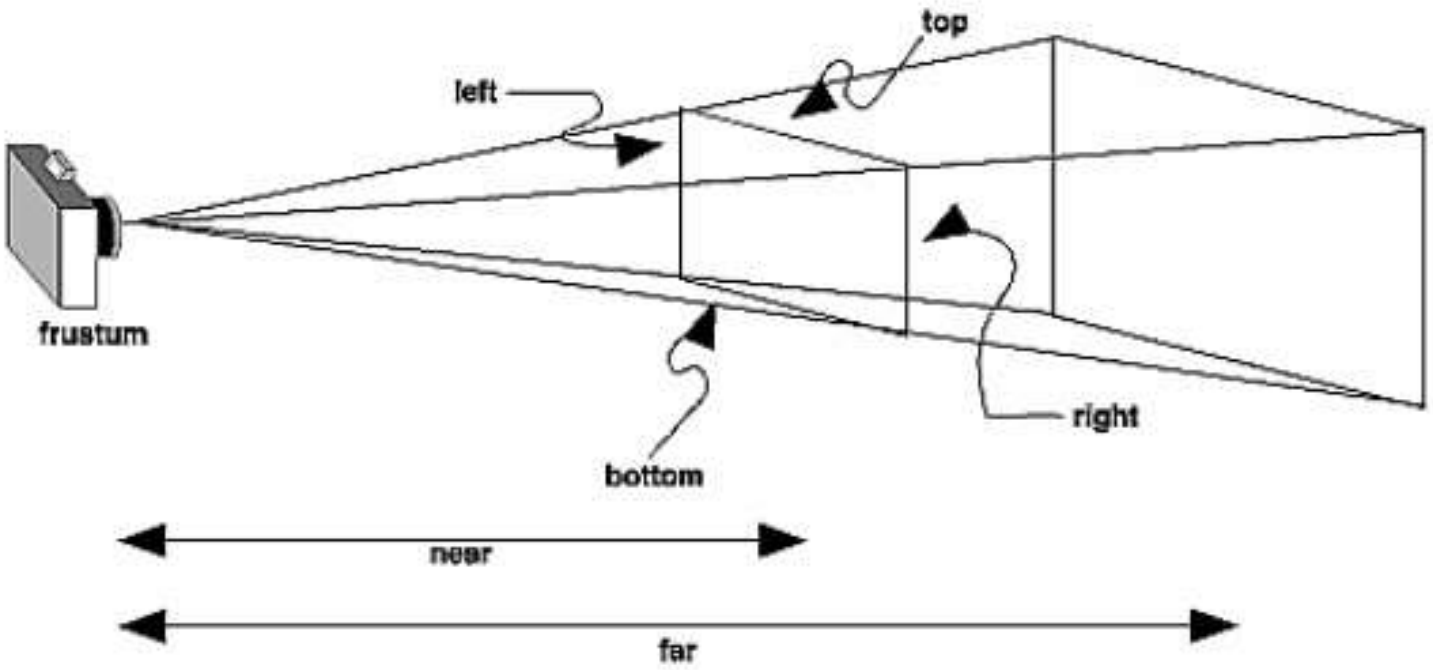
```
void reshape(int w, int h) {
    // viewport transformation
    glViewport(0, 0, w, h);
    // projection transformation
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    projeccao();
    ...
}
void display() {
    // modelview transformation
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
    // posicionamento da câmara
    camara();
    // transformações do modelo
    ...
}
```

What is a projection transformation?

- The purpose of the projection transformation is to define the viewing volume, which is used in two ways:
  - Determines how an object is projected on the screen (using a perspective or orthographic projection), and
  - Defines which objects or parts of these are eliminated from the final image.

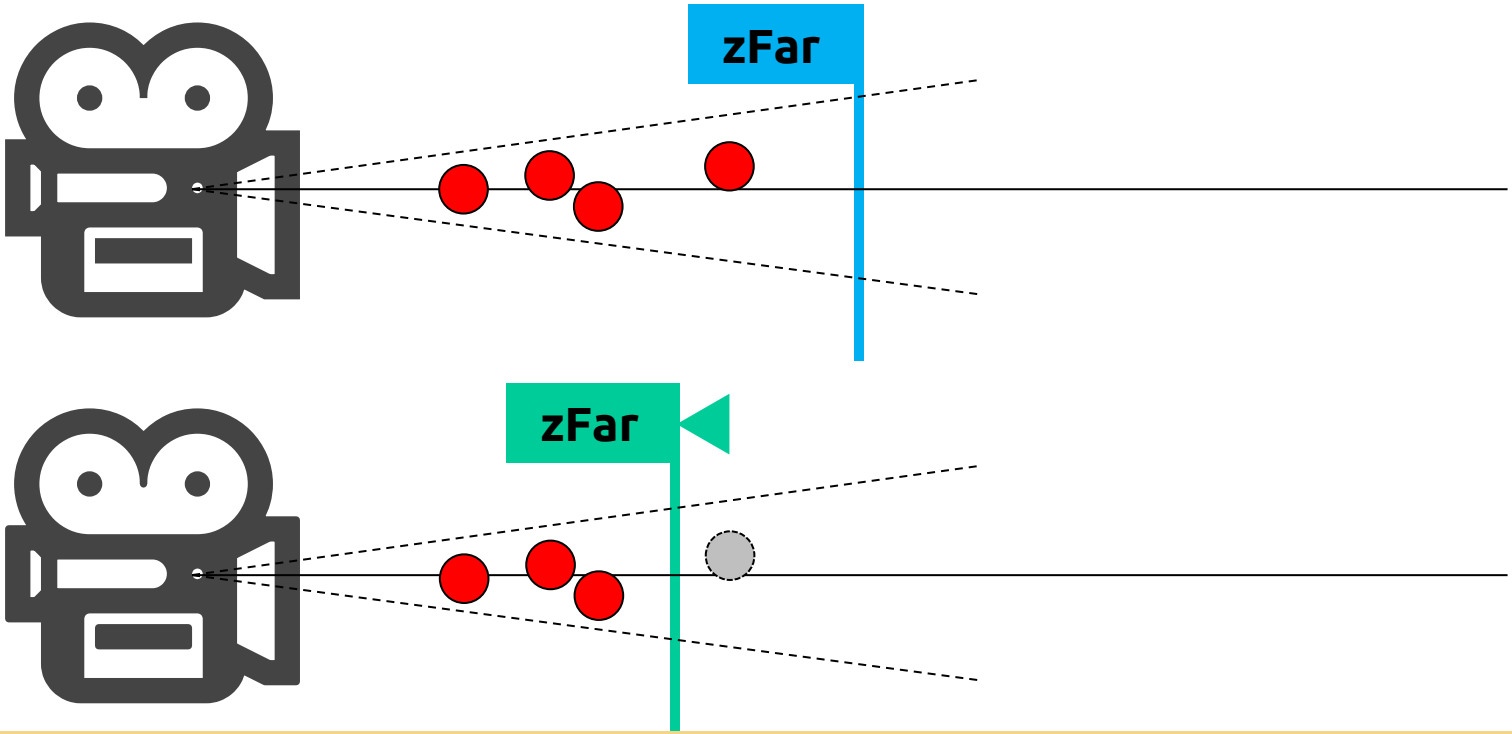
Perspetive

- `void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar)`



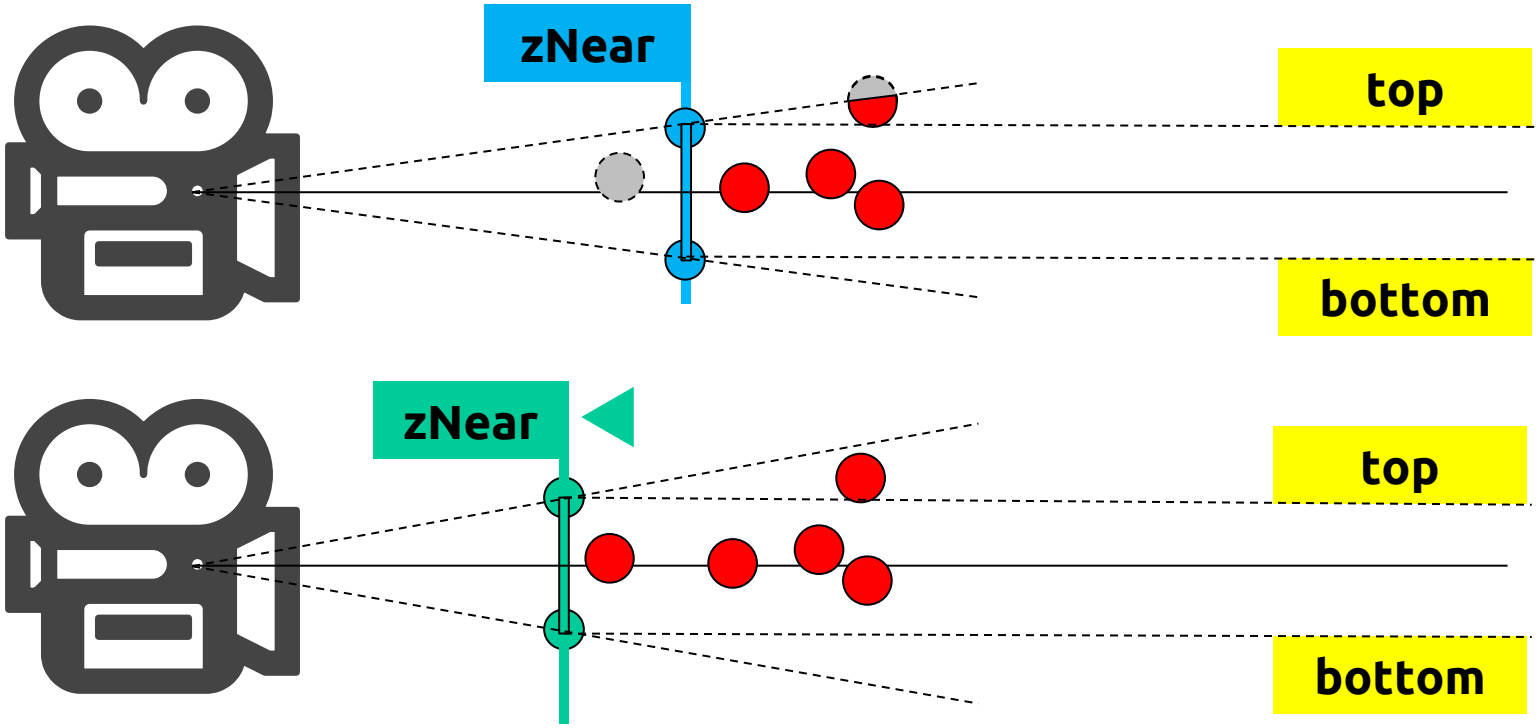
Perspetive

- `void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar)`



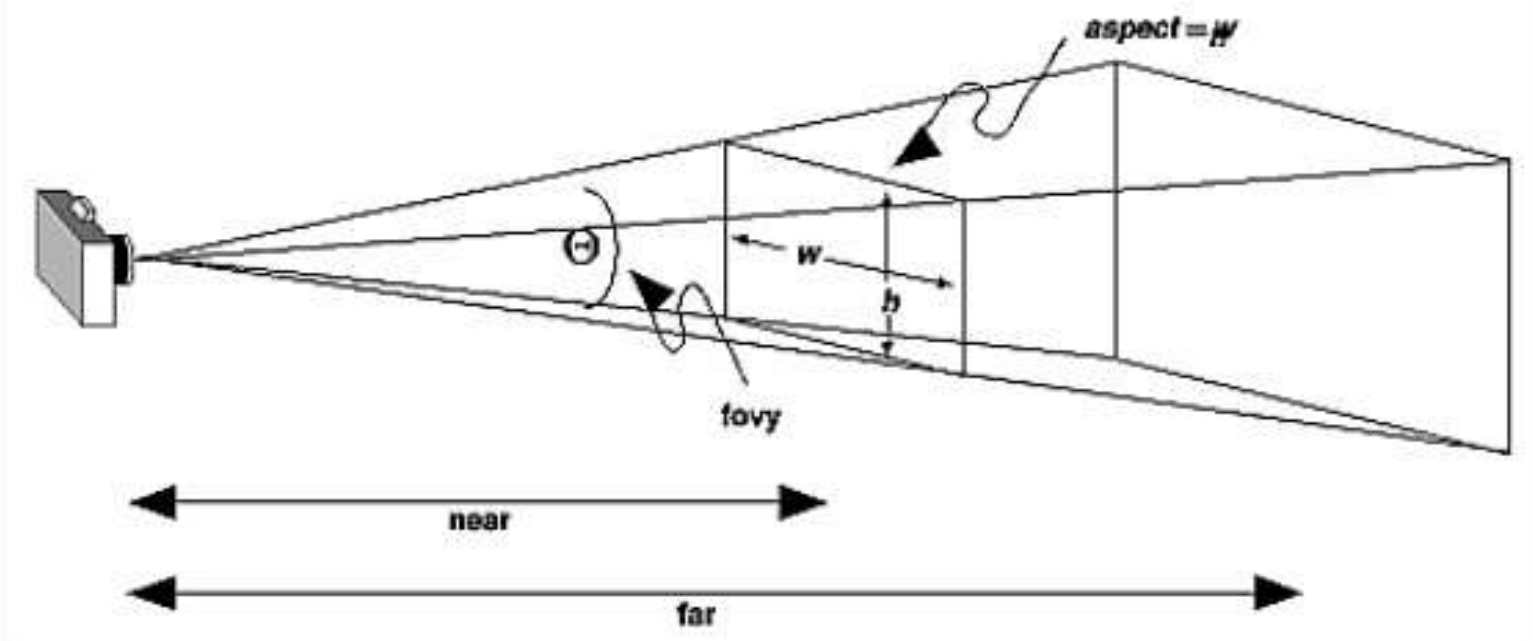
Perspetive

- `void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble zNear, GLdouble zFar)`



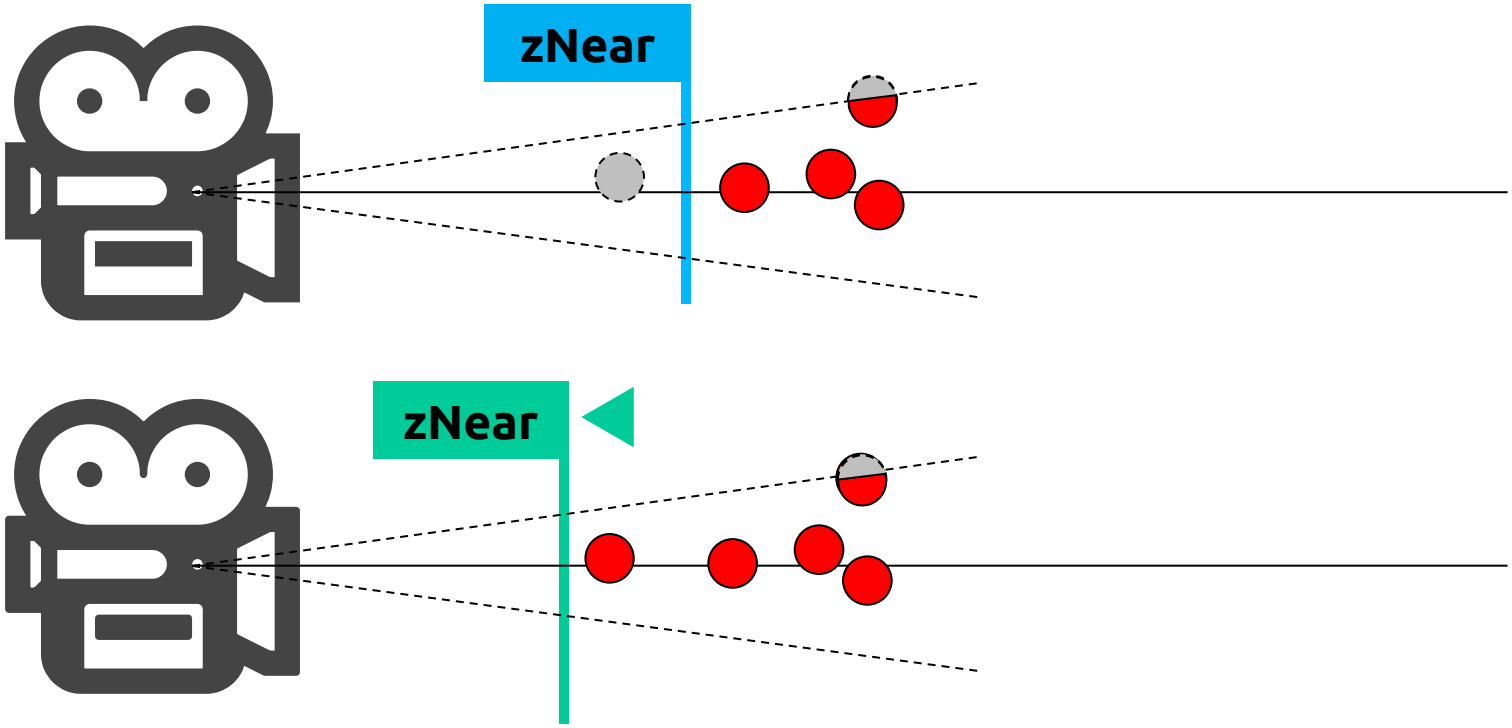
Perspetive

- `void gluPerspective( GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);`



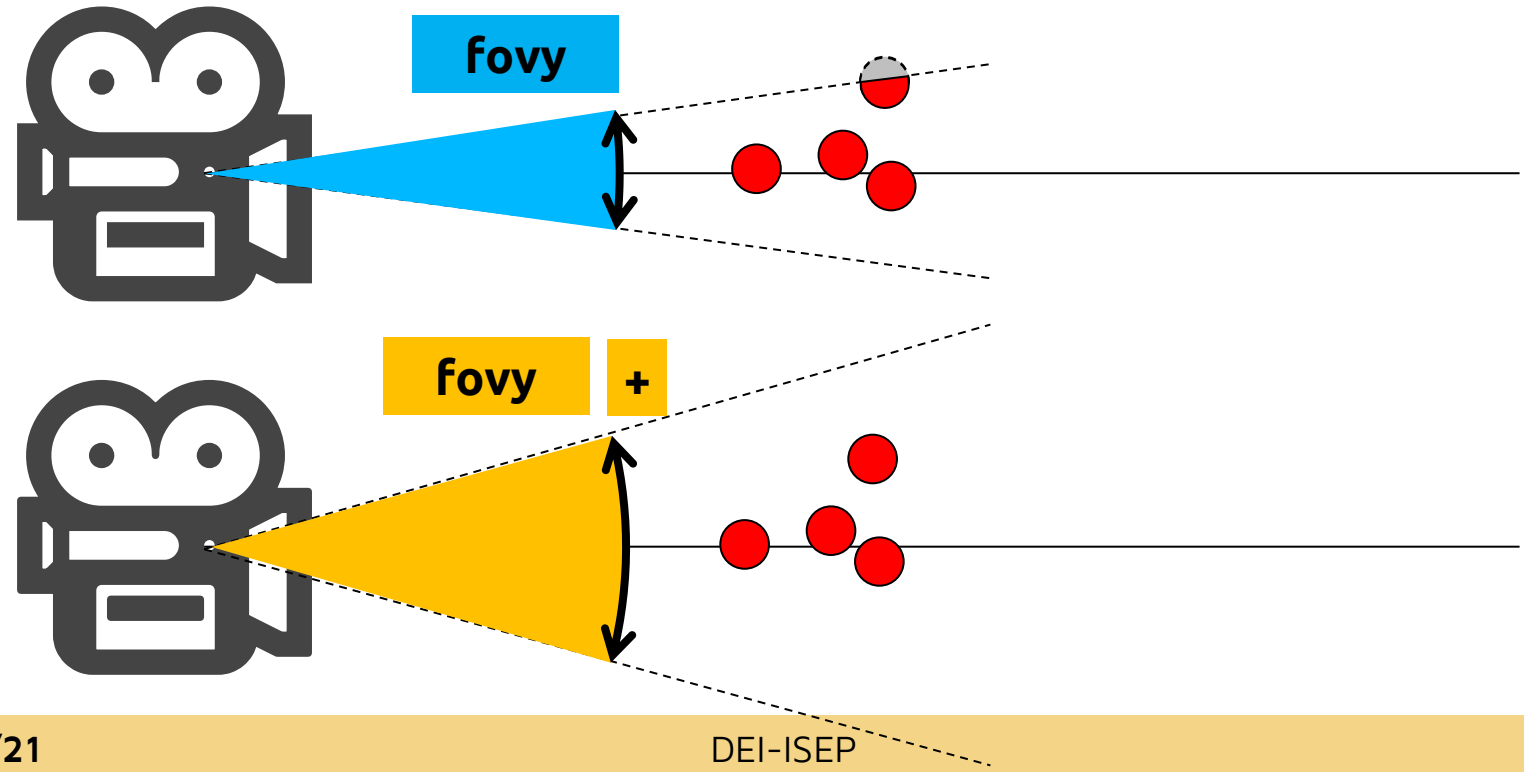
Perspetive

- `void gluPerspective( GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);`



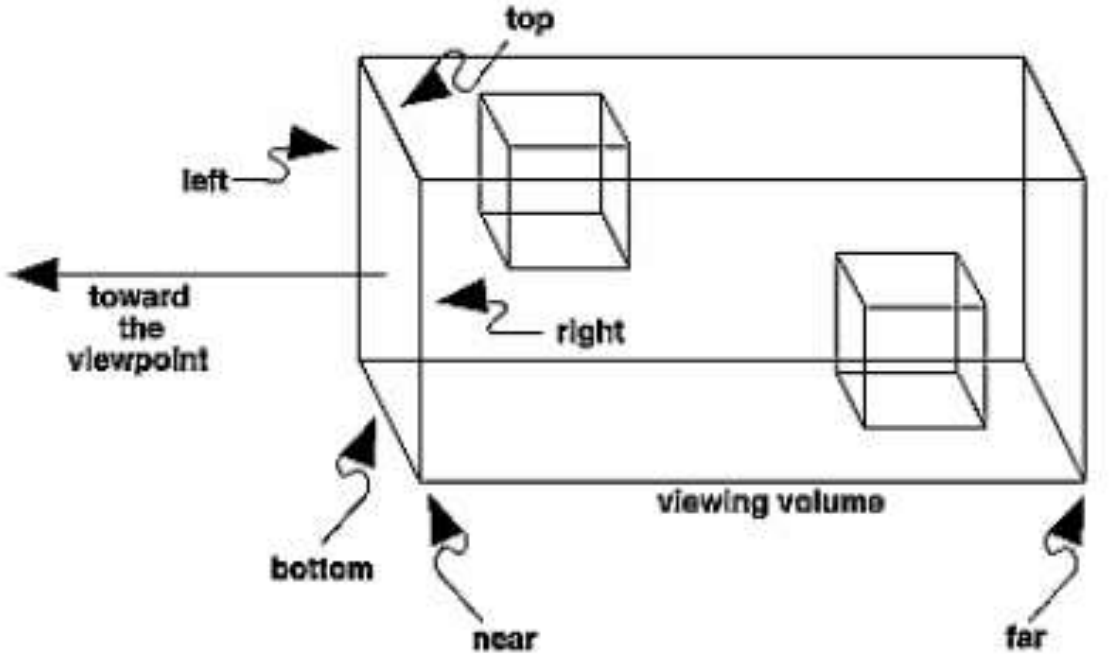
Perspetive

- `void gluPerspective( GLdouble fovy, GLdouble aspect, GLdouble zNear, GLdouble zFar);`



Ortográfica

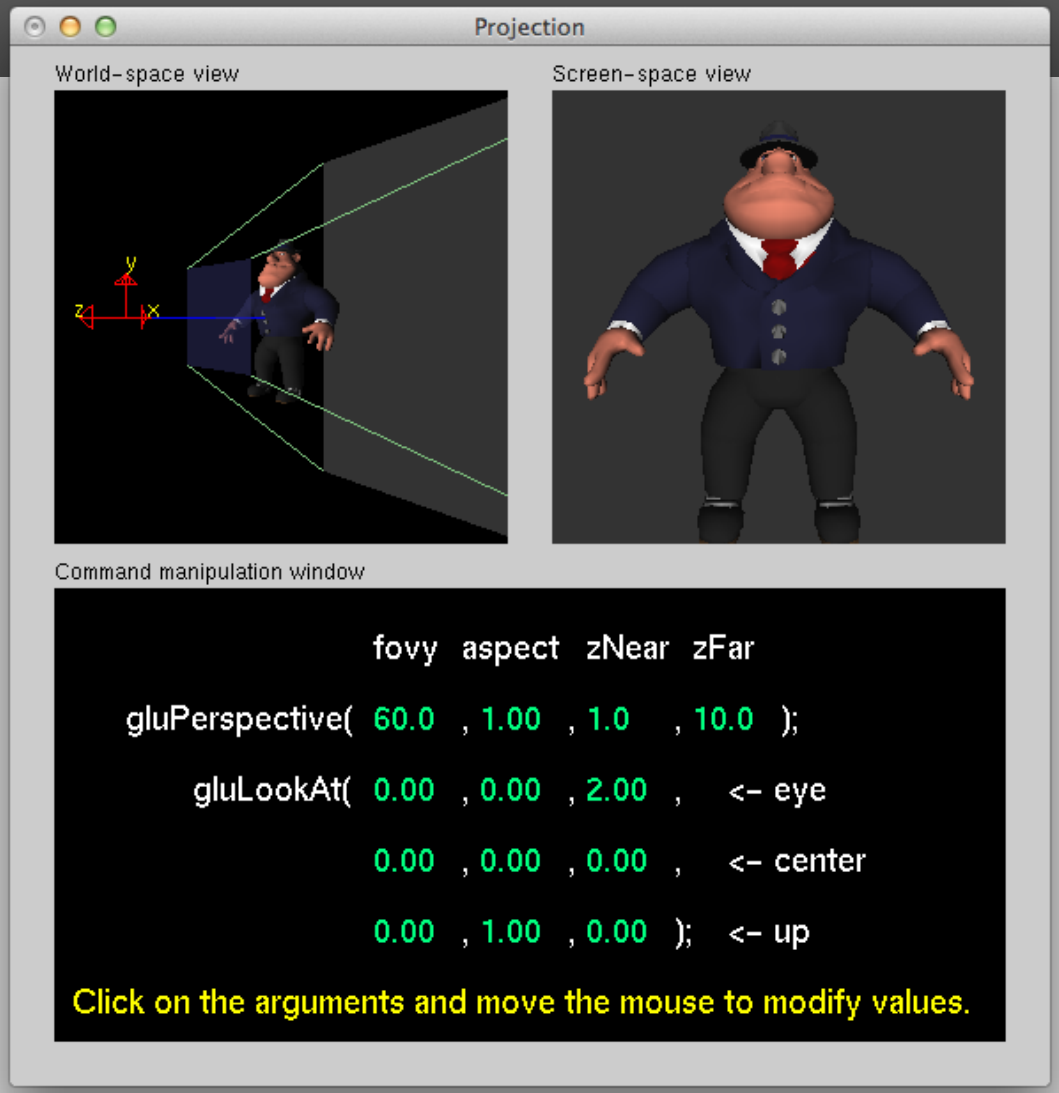
- `void glOrtho(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, Gldouble near, Gldouble far);`



zNear e zFar in Ortographic Projection

- zNear and zFar define *clipping planes* **relative to the camera position**
- Example
  - zNear = 1 & zFar = 3
  - Camera (0, 0, 0)
    - Only objects with z coordinate in the range [-1, -3] are visible
  - Move camera to (0, 0, 3)
    - Only objects with z coordinate in the range [2, 0] are visible

Demo


Projection
World-space view
Screen-space view
y
z
x
Command manipulation window
fovy aspect zNear zFar
gluPerspective(60.0 , 1.00 , 1.0 , 10.0);
gluLookAt(0.00 , 0.00 , 2.00 , <- eye
0.00 , 0.00 , 0.00 , <- center
0.00 , 1.00 , 0.00); <- up
Click on the arguments and move the mouse to modify values.