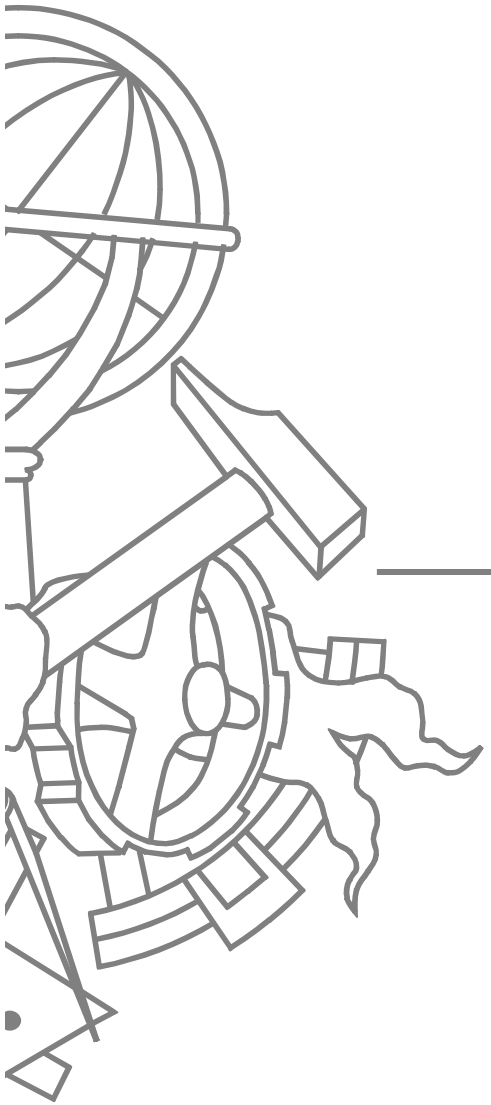


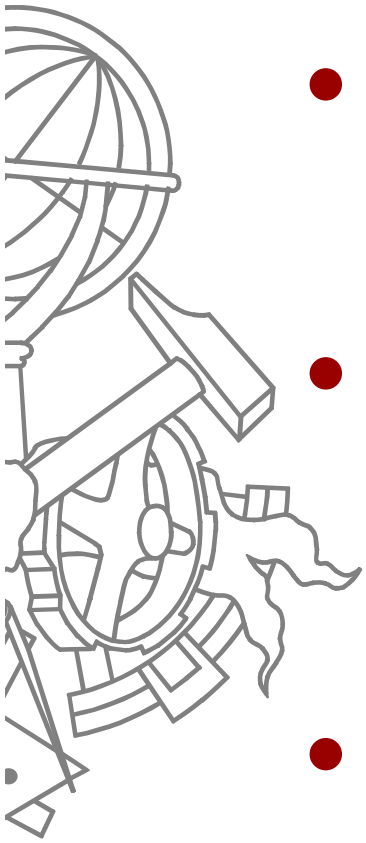
Transformações geométricas

Aula 4

Sistemas Gráficos e Interactivos
Instituto Superior de Engenharia do Porto

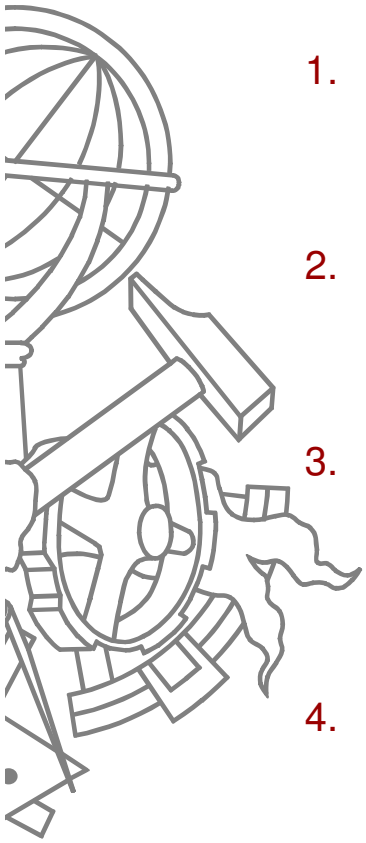
Paulo Gandra de Sousa
psousa@dei.isep.ipp.pt

Conteúdo

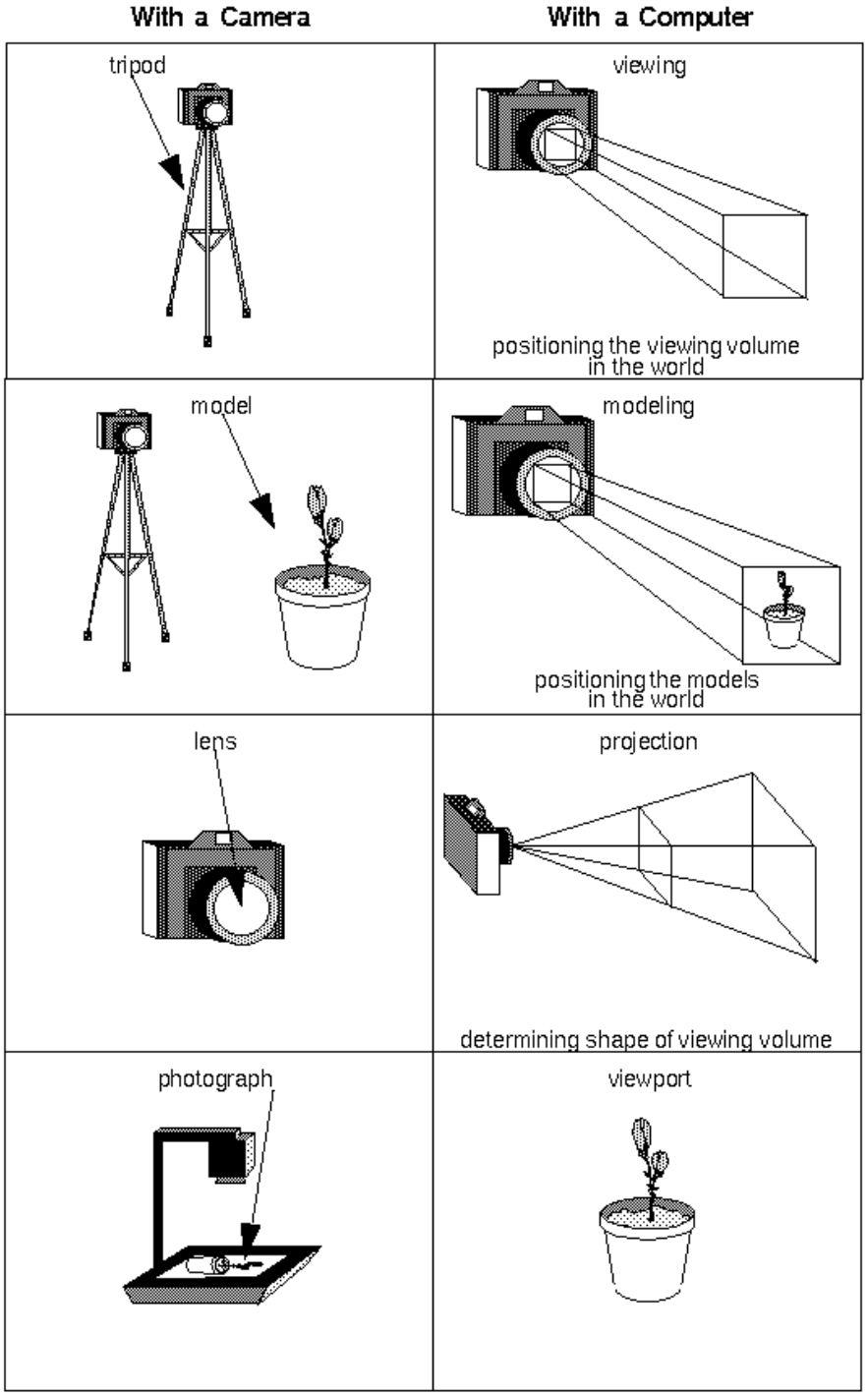


- Tipos de transformações
  - Model/view
  - Viewport
- Operações de transformação
  - Rotação
  - Translação
  - Escalamento
- Posicionamento da câmara
- Composição de transformações

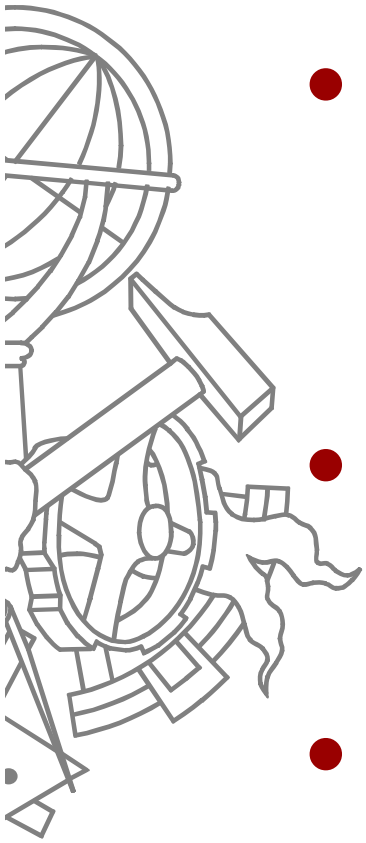
Analogia da câmara fotográfica



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).



Tipos de transformações

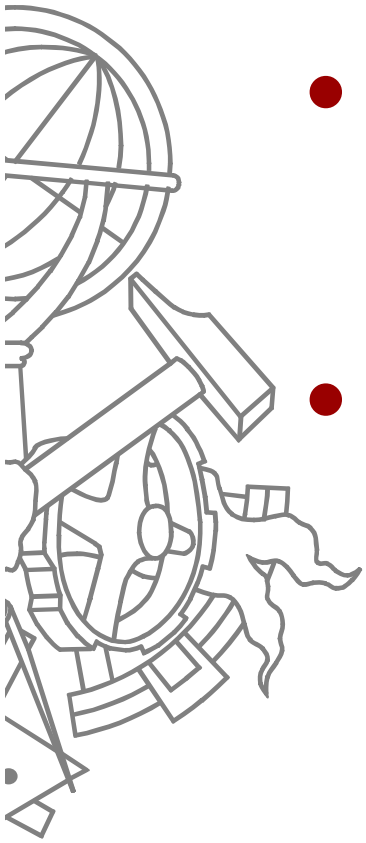


- View e Model
  - glMatrixMode(GL_MODELVIEW)
  - Posicionamento da câmara
  - Rotações, translações, escalamentos
- Projection
  - glMatrixMode(GL_PROJECTION)
  - Definição do volume de projecção
- Viewport
  - glViewport (x, y, width, height)

Esqueleto de código

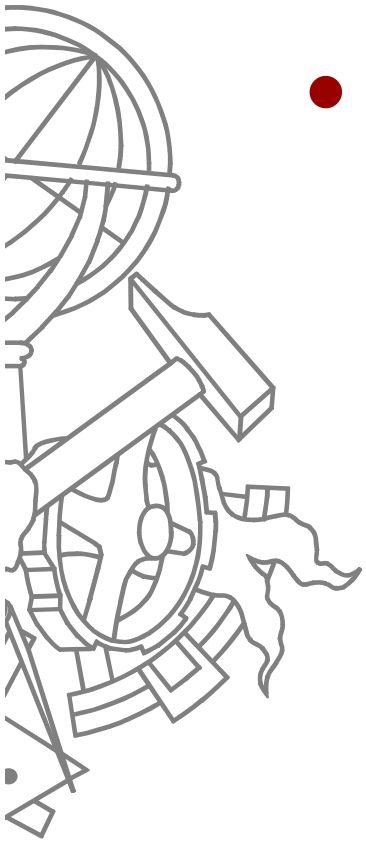
```
void reshape(int w, int h) {
   // viewport transformation
   glViewport(0, 0, w, h);
   // projection transformation
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   projeccao();
   ...
}

void display() {
   // modelview transformation
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   // posicionamento da câmara
   camara();
   // transformações do modelo
   ...
}
```

Inicializar as matrizes

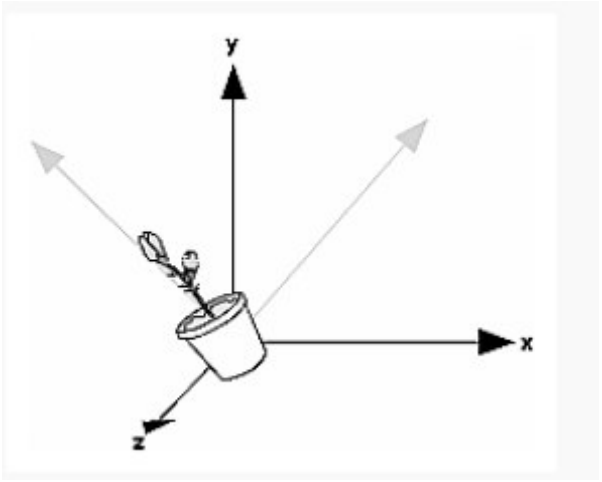


- Antes de definir transformações deve-se inicializar a matriz de transformação usando **`glLoadIdentity`**
- Deve-se evitar cálculos acumulados nas matrizes
  - Exemplo: não ir aplicando rotações na mesma matriz, mas sim ter uma variável com o angulo de rotação

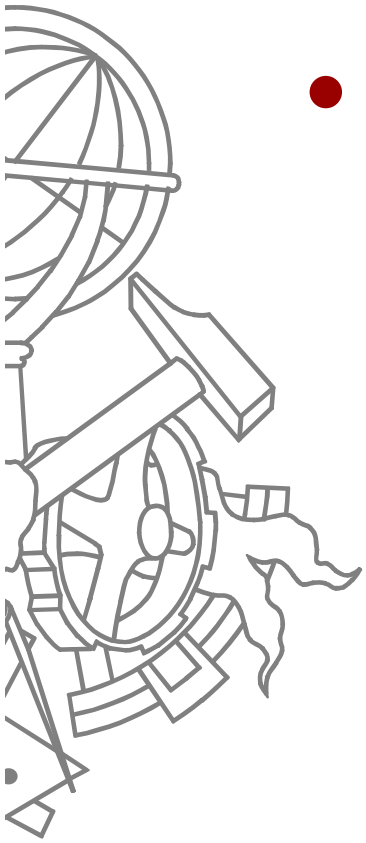
Rotações



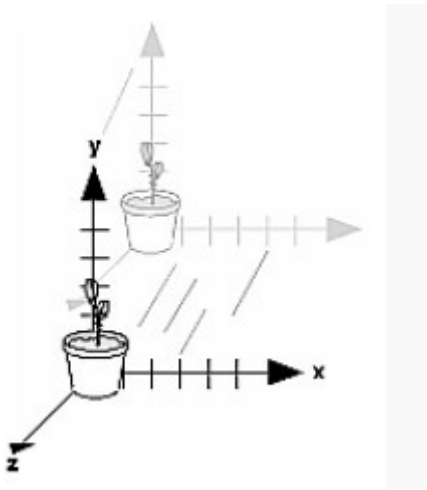
- void glRotate{ f | d }(angle, x, y, z)



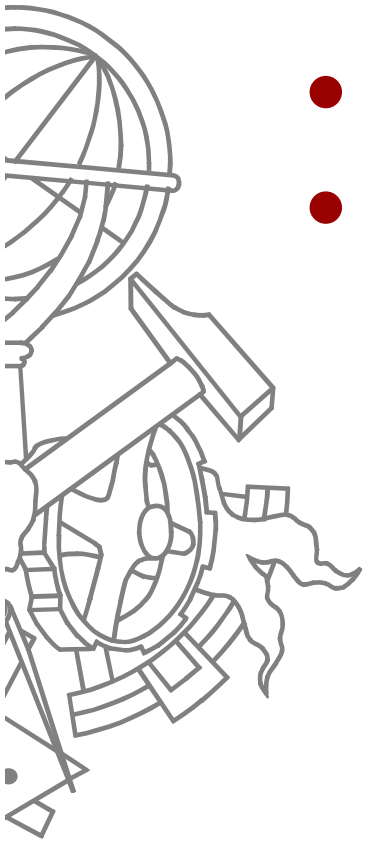
Translações



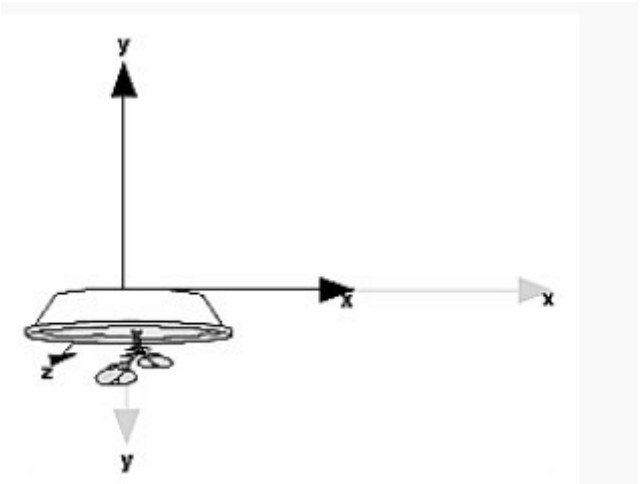
- void glTranslate{ f | d}(x, y, z);



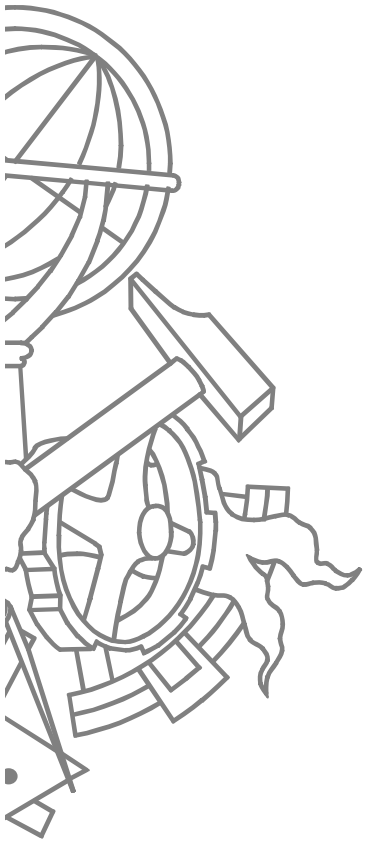
Escalamento

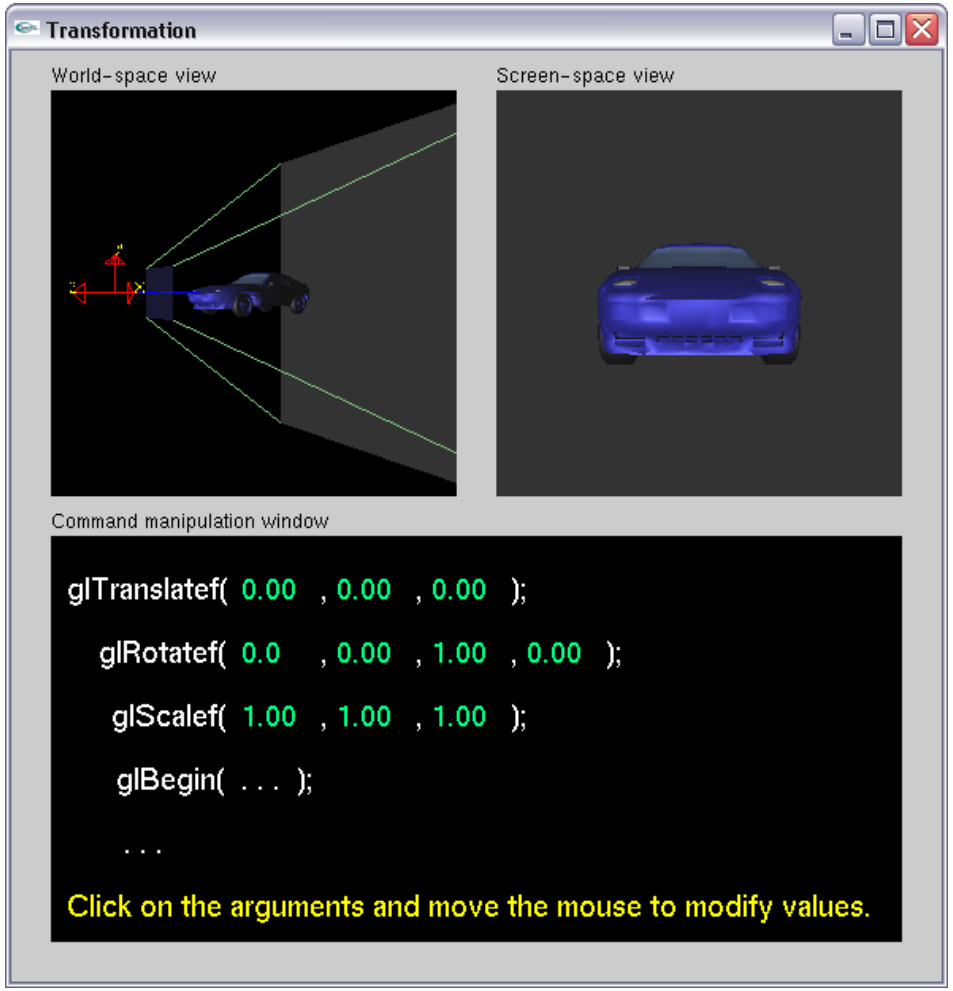


- void glScale{ f | d }(x, y, z)
- a evitar pois altera vectores normal

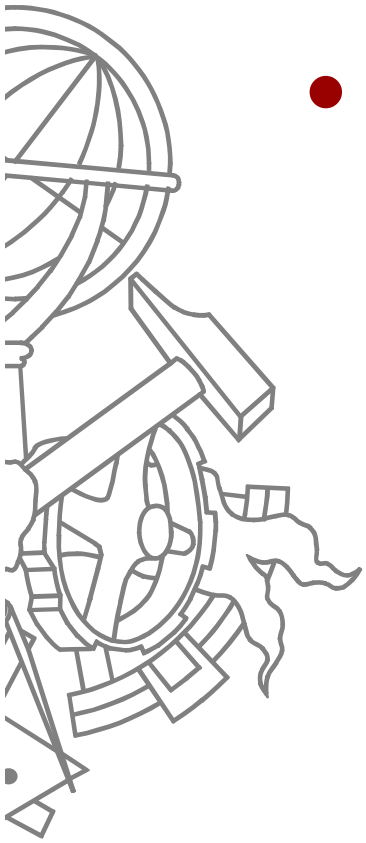


Demo

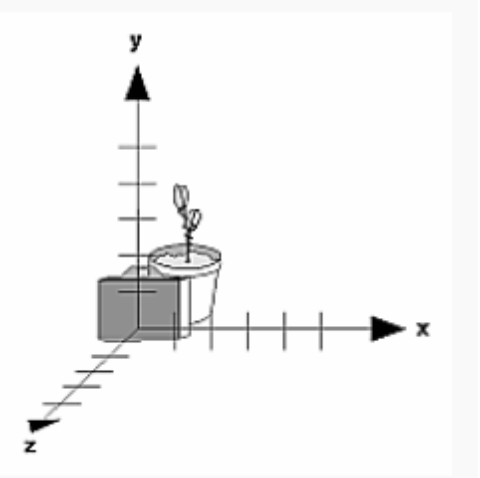



Transformation
World-space view
Screen-space view
Command manipulation window
glTranslatef(0.00 , 0.00 , 0.00);
glRotatef(0.0 , 0.00 , 1.00 , 0.00);
glScalef(1.00 , 1.00 , 1.00);
glBegin(. . .);
. . .
Click on the arguments and move the mouse to modify values.

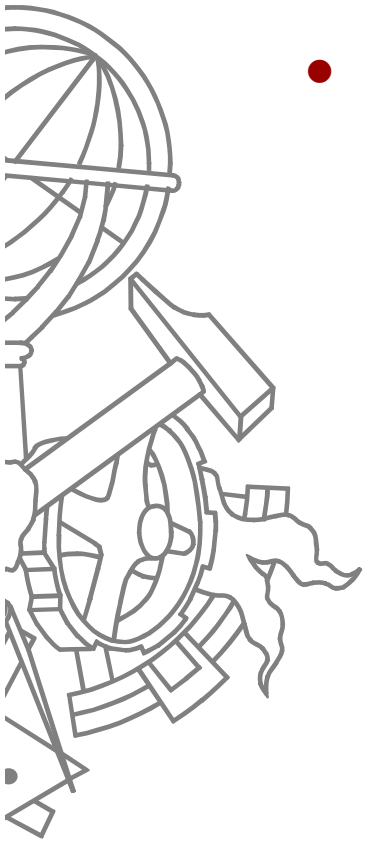
Posição da câmara



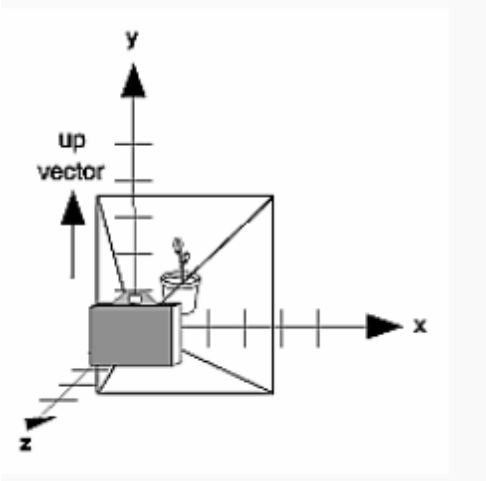
- Por omissão a câmara está colocada na origem (0, 0, 0) dirigida para o eixo negativo dos zz

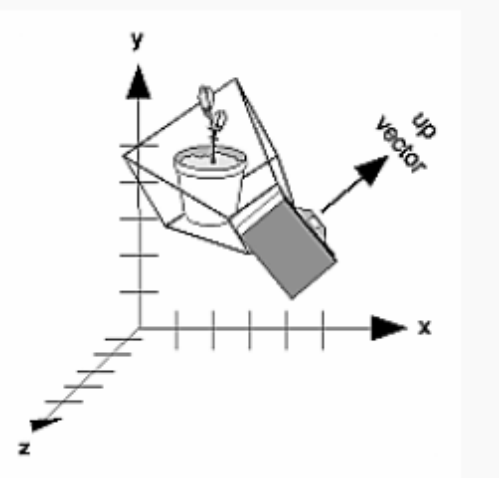


Posicionamento da câmara

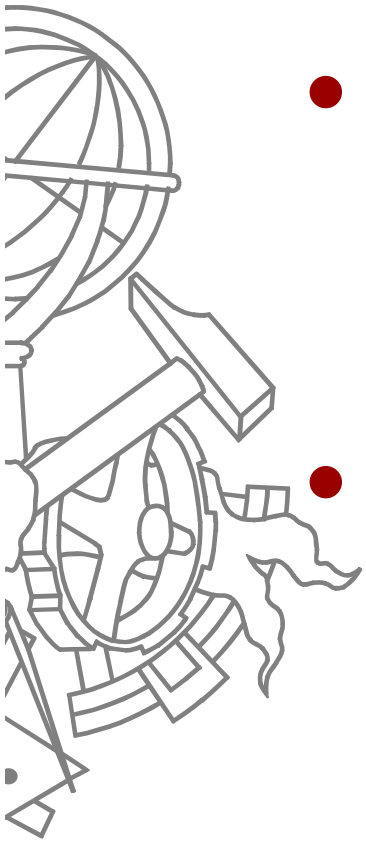


- `void gluLookAt(eyex, eyey, eyez, centerx, centery, centerz, upx, upy, upz)`



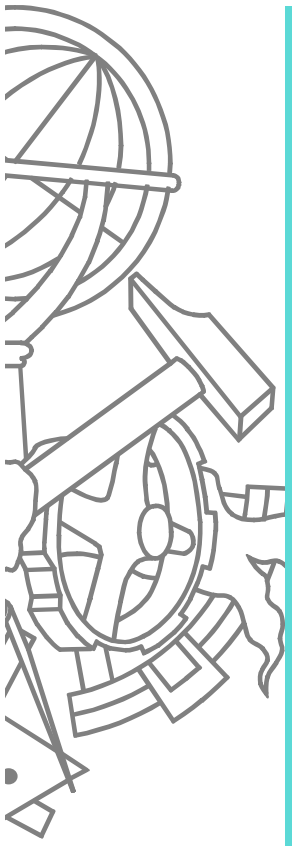


Posição da câmara

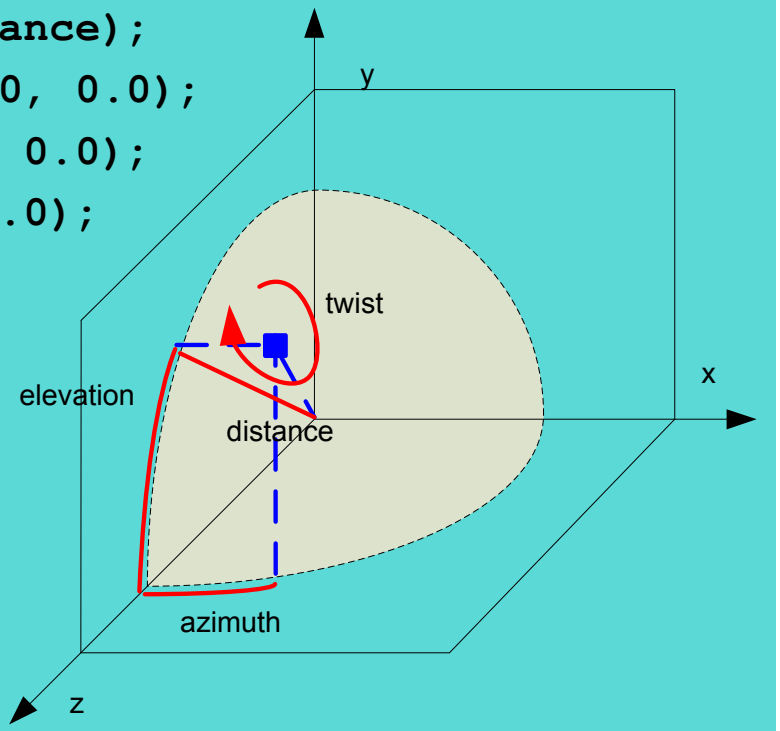


- Mover a câmara ou mover a cena obtém o mesmo resultado
  - `gluLookAt(0, 0, +5, 0, 0, 0, 0, 1, 0)`
  - `glTranslatef(0, 0, -5)`
- Podem usar **`gluLookAt`** ou construir a vossa própria rotina de visualização, por exemplo, câmara com movimento polar usando as operações básicas de transformação

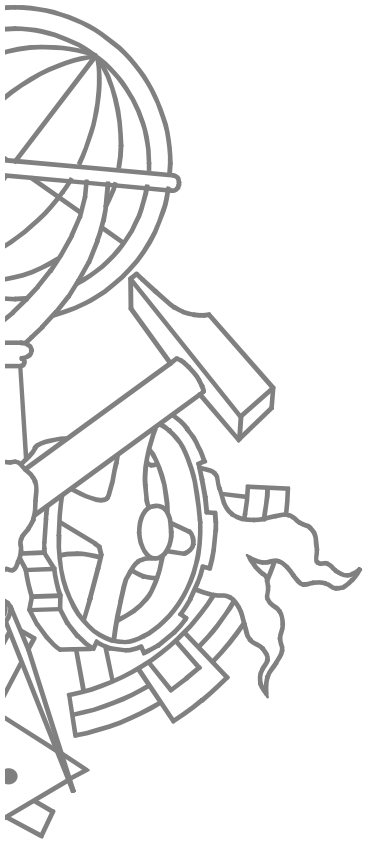
Câmara "polar"

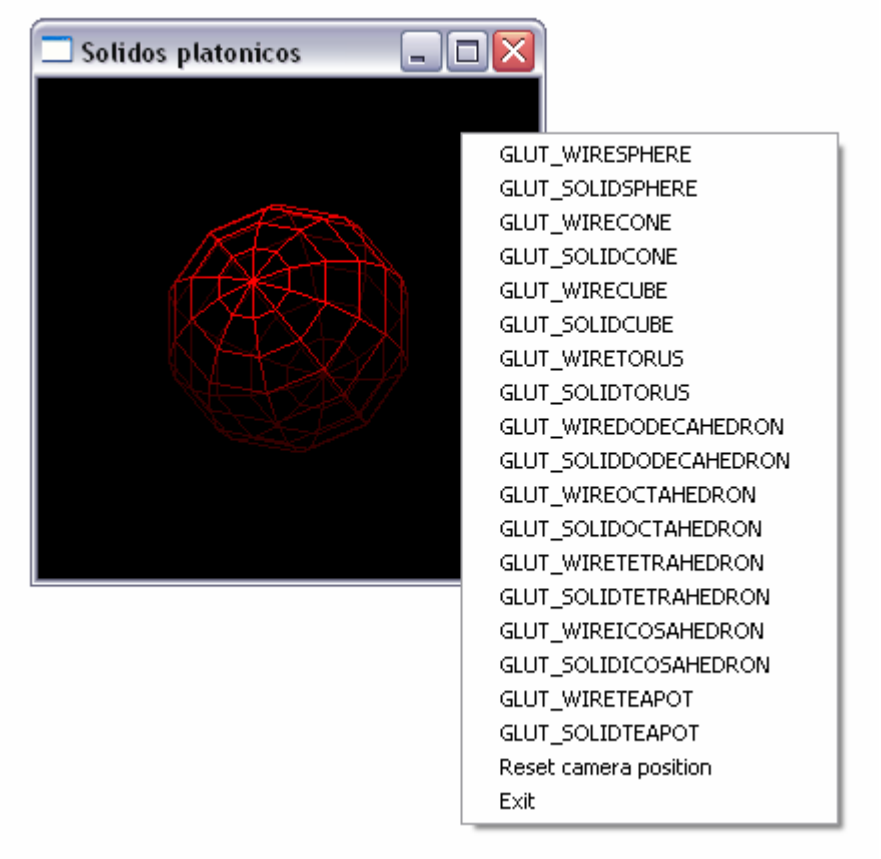


```
void polarView(GLdouble distance,
               GLdouble twist,
               GLdouble elevation,
               GLdouble azimuth)
{
   glTranslated(0.0, 0.0, -distance);
   glRotated(elevation, 1.0, 0.0, 0.0);
   glRotated(azimuth, 0.0, 1.0, 0.0);
   glRotated(twist, 0.0, 0.0, 1.0);
}
```

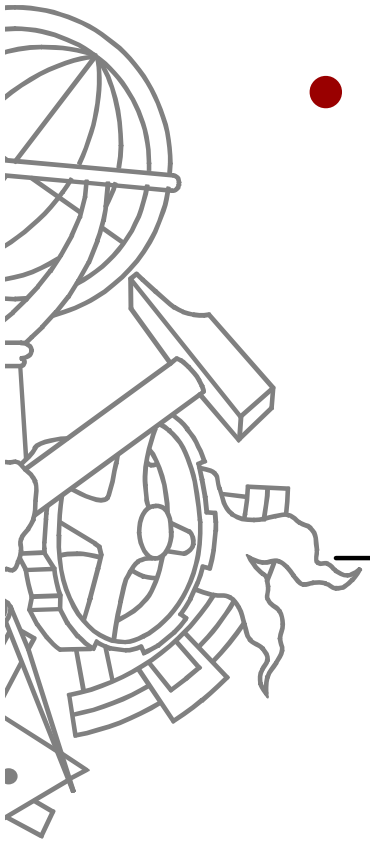


Demo – câmara polar

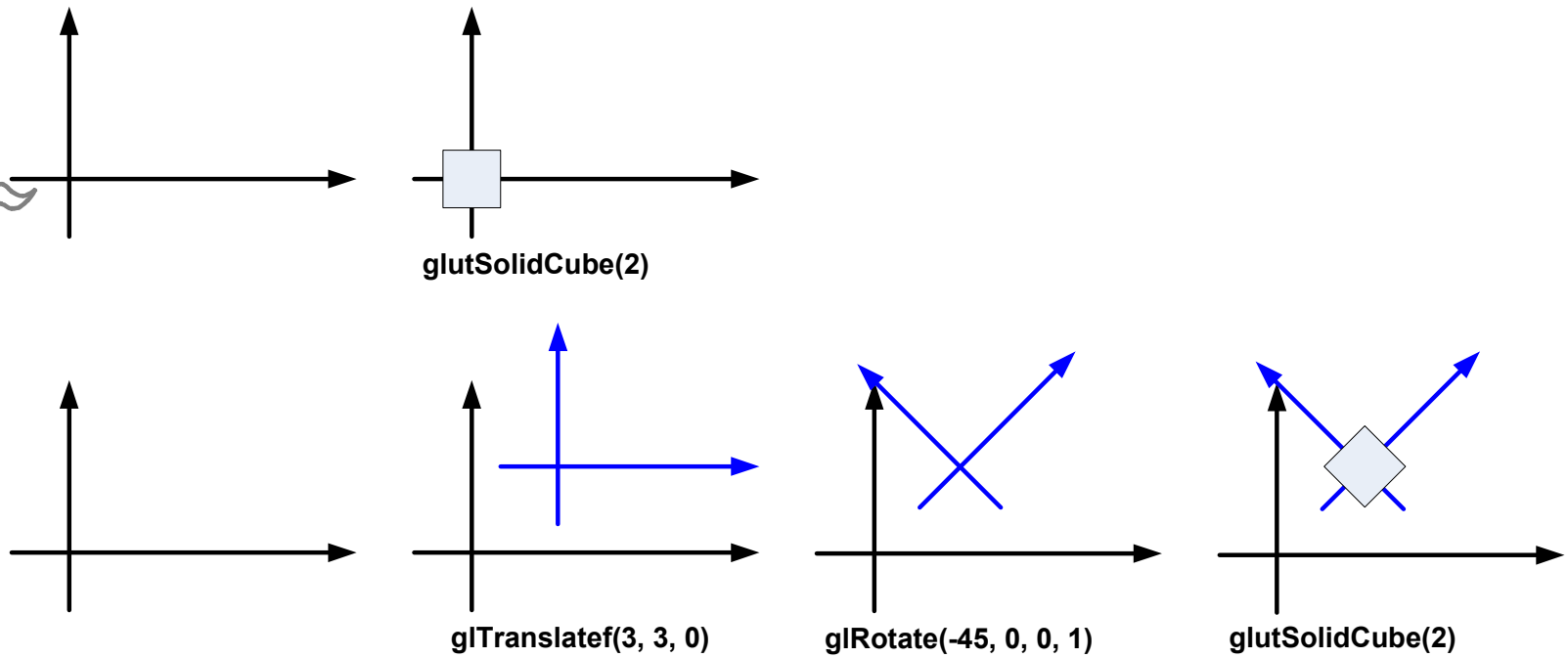




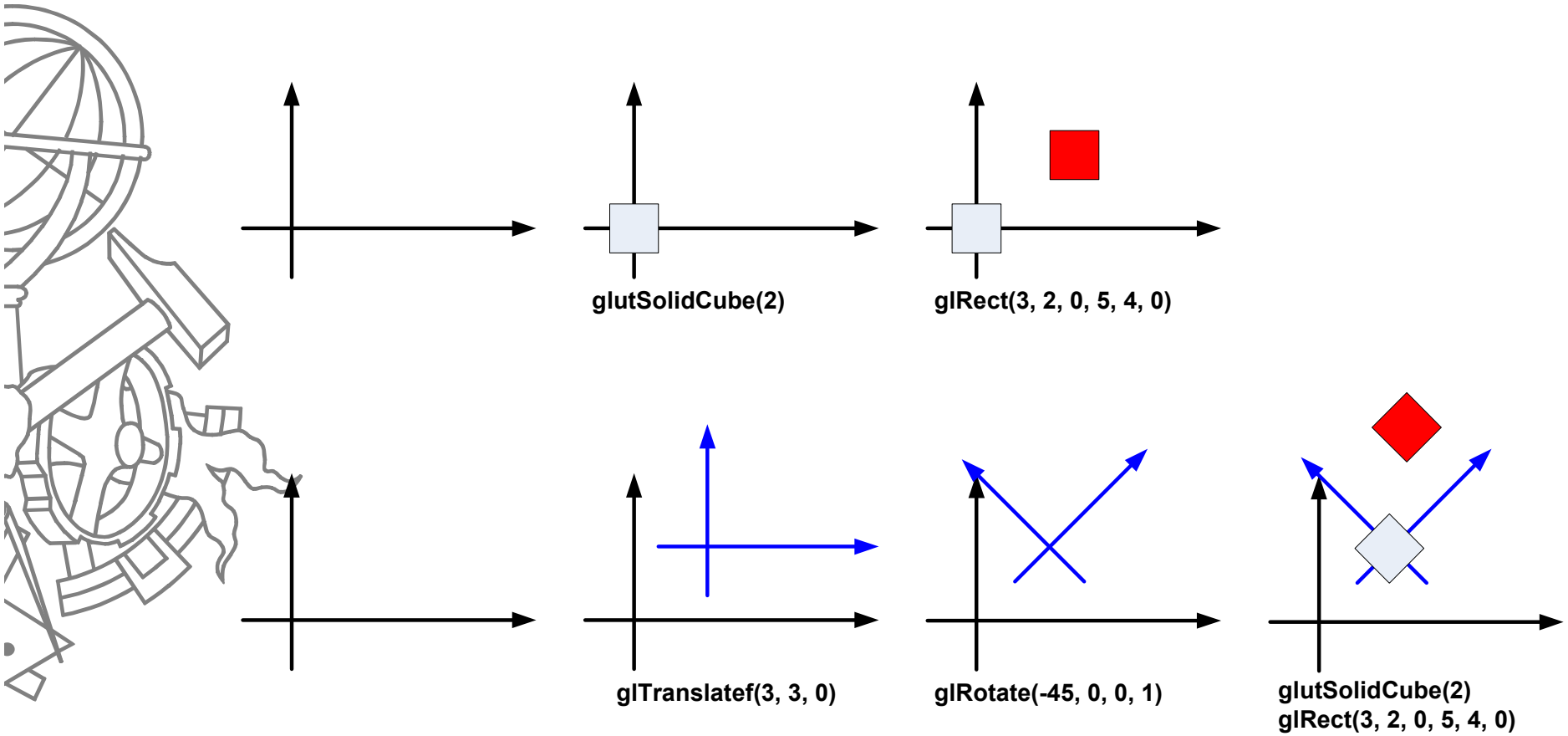
O sistema de coordenadas local



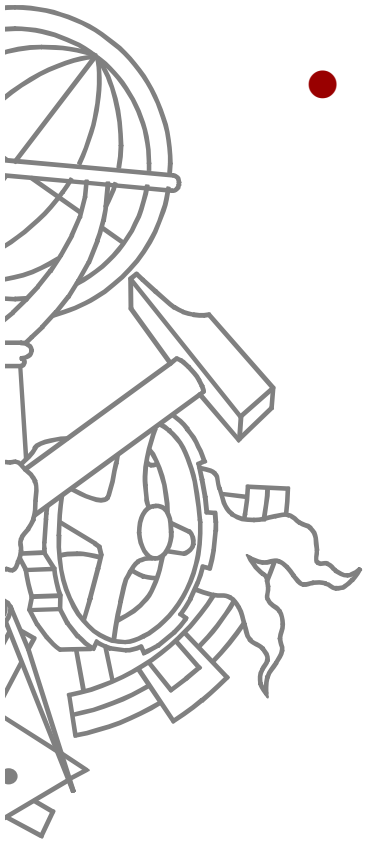
- Ao aplicar uma transformação estão na realidade a mover um sistema de coordenadas “agarrado” ao modelo



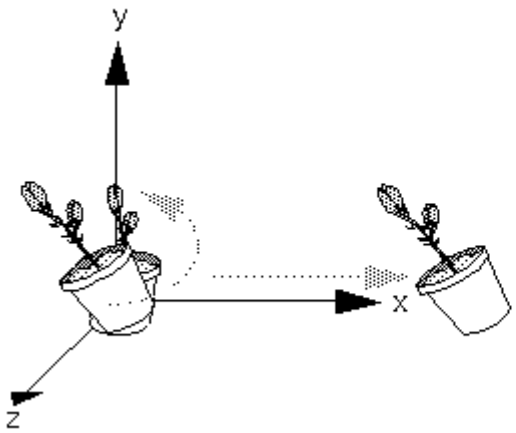
O sistema de coordenadas local



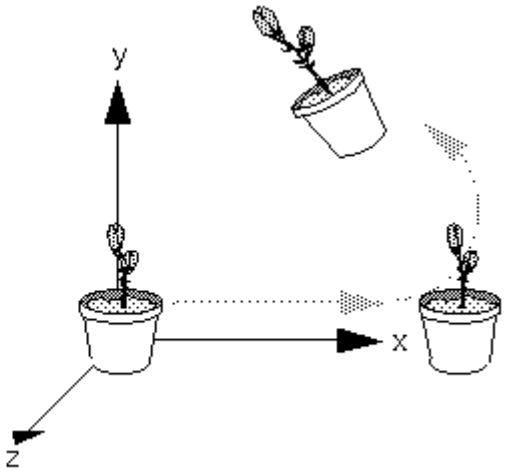
Efeito cumulativo de transformações



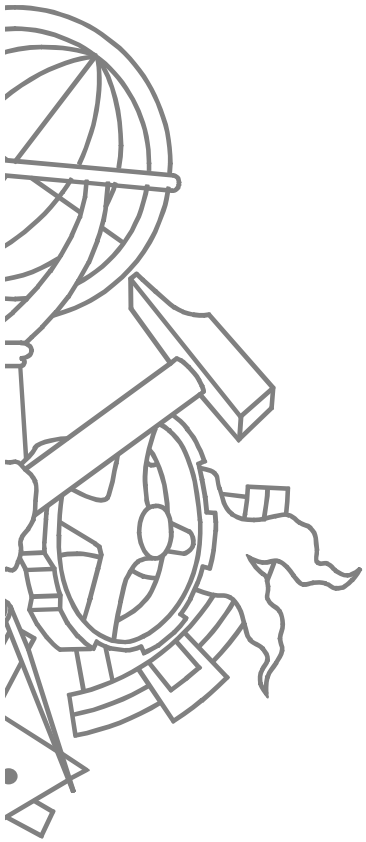
- Rotação + translação

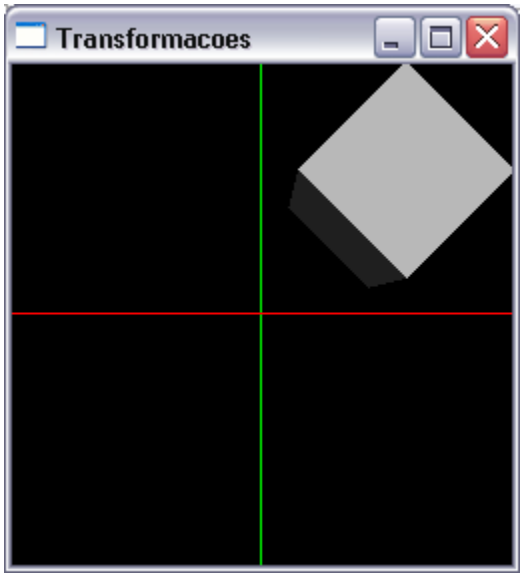


- Translação + rotação

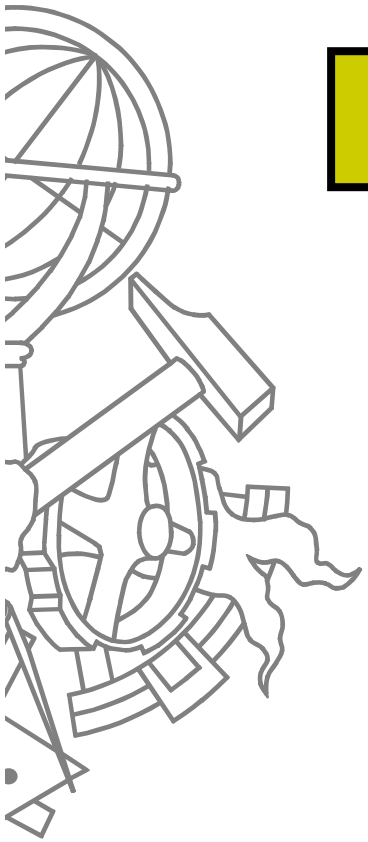


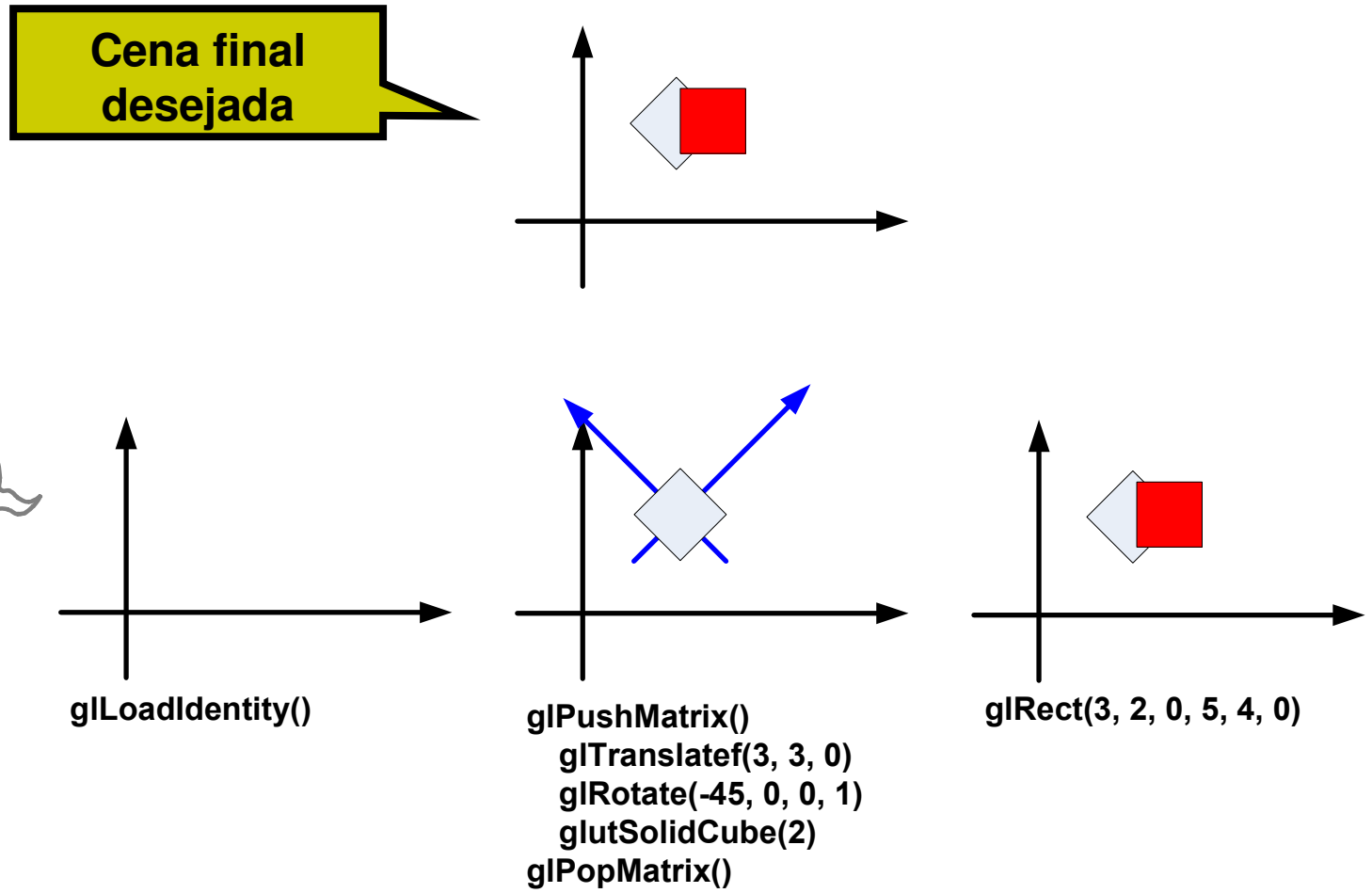
Demo



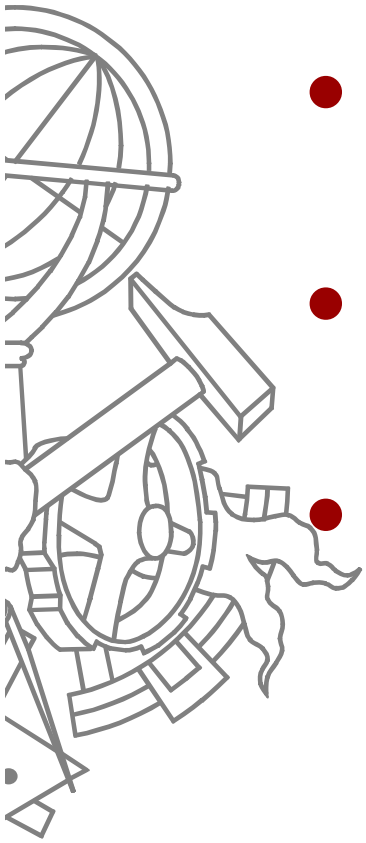


Efeito cumulativo de transformações



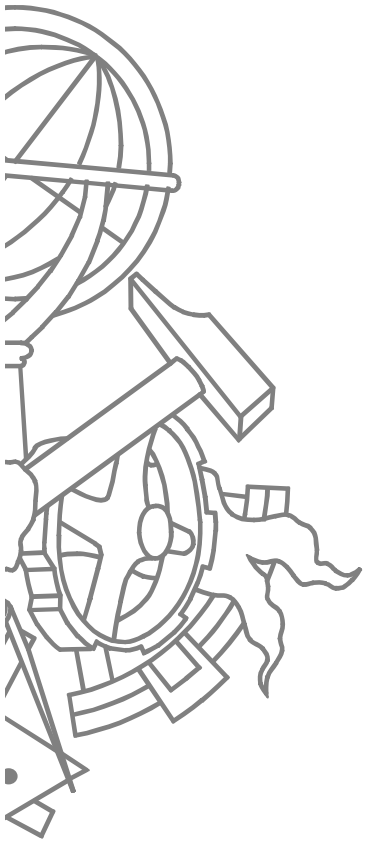


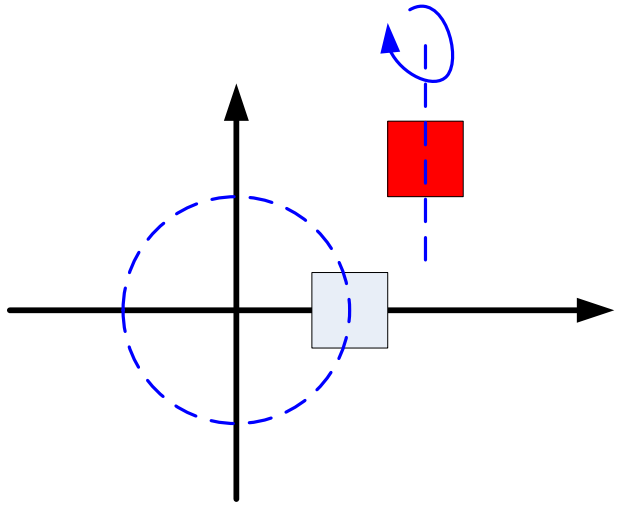
Transformações “locais”



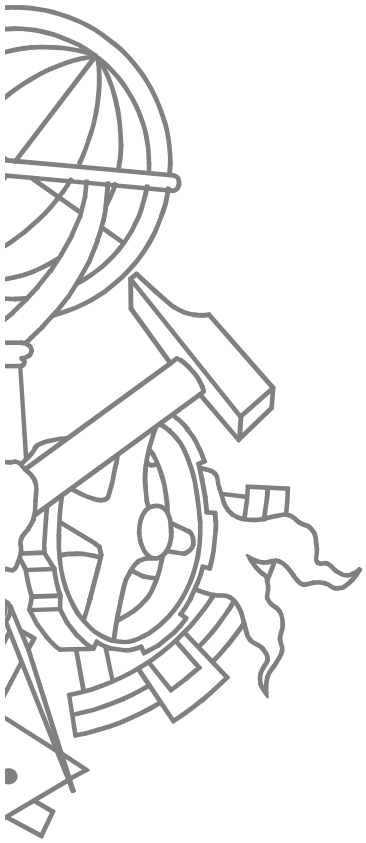
- Guardar a matriz de transformação usando **`glPushMatrix`**
- Recuperar a matriz anterior usando **`glPopMatrix`**
- Push e pop matrix podem ser usados para a matriz de projecção ou para a matriz de modelo/vista
  - Devem ter atenção a qual a matriz activa no momento

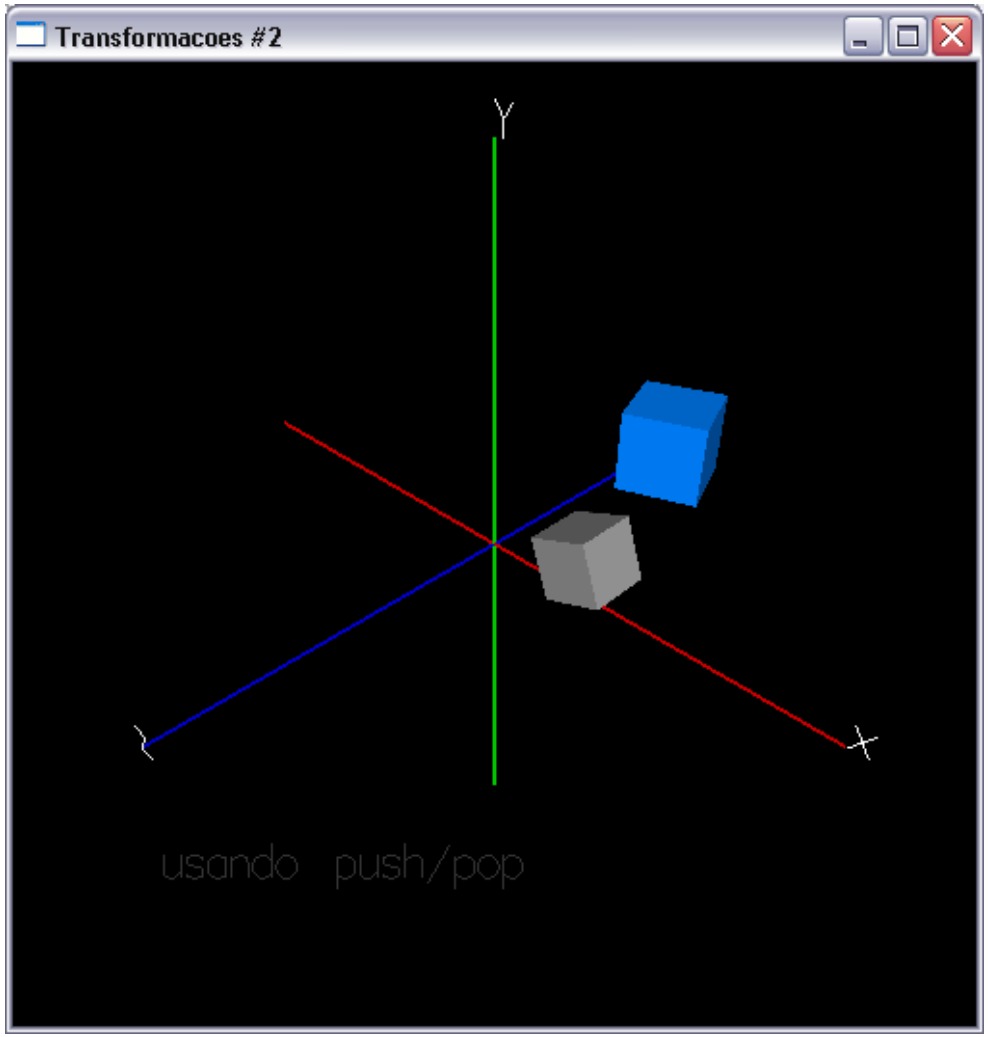
Transformações (in)dependentes



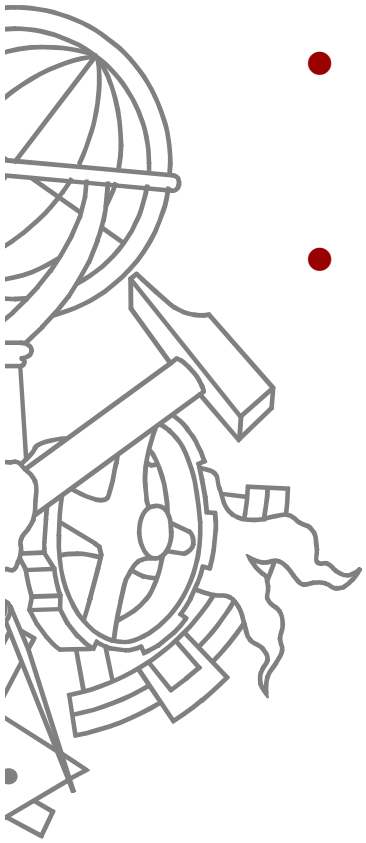


Demo



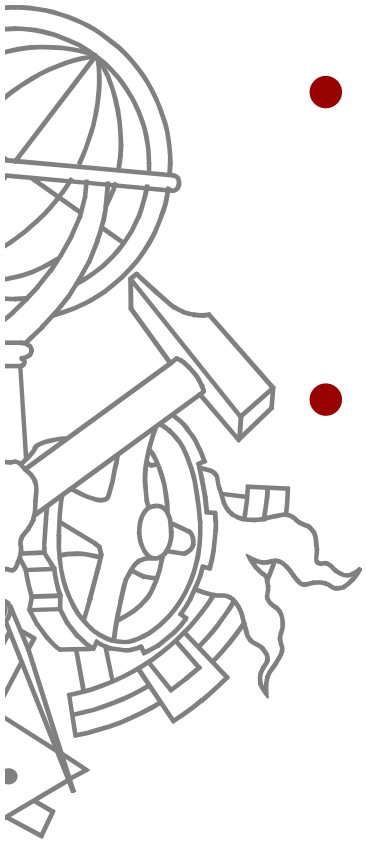


Texto em GLUT



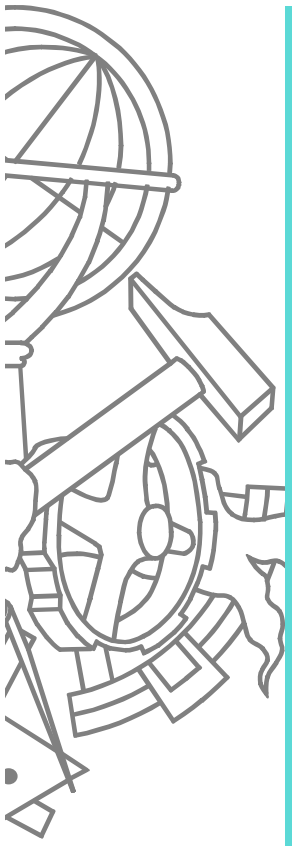
- Desenhar um caracter
  - glutStrokeCharacter(*fonte*, *caracter*)
  - glutBitmapCharacter(*fonte*, *caracter*)
- Fontes
  - GLUT_STROKE_ROMAN
  - GLUT_STROKE_MONO_ROMAN
  - GLUT_BITMAP_9_BY_15
  - GLUT_BITMAP_8_BY_13
  - GLUT_BITMAP_TIMES_ROMAN_10
  - GLUT_BITMAP_TIMES_ROMAN_24
  - GLUT_BITMAP_HELVETICA_10
  - GLUT_BITMAP_HELVETICA_12
  - GLUT_BITMAP_HELVETICA_18

Texto em GLUT



- Desenhado usando o sistema de coordenadas locais e todas as transformações em vigor
- O tamanho de cada caracter é tipicamente 120 unidades
  - Necessário escalar
  - Proteger a matriz de transformação usando push e pop matrix
- Ciclo para imprimir uma string completa

Função escreve()



```
void drawString(GLfloat x, GLfloat y, GLfloat z,
        GLfloat scale,
        char* msg)
{
   glPushMatrix();
     glTranslatef(x, y, z);
     glScalef(scale, scale, scale);
     for (int i = 0; i < strlen(msg); i++)
        glutStrokeCharacter(GLUT_STROKE_ROMAN, msg[i]);
   glPopMatrix();
}
```

Texto sem usar transformações do modelo

```
// exemplo
glPushMatrix();
   glLoadIdentity();
   gluLookAt(0, 0, 5, 0, 0, 0, 0, 1, 0);

   glColor4fv(mat_w);
   glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_w);
   drawString(-2, -2, 0, 0.002, msg);
glPopMatrix();
```

Demo