Módulo 4

Sistemas Gráficos e Interação

Instituto Superior de Engenharia do Porto

Filipe Pacheco

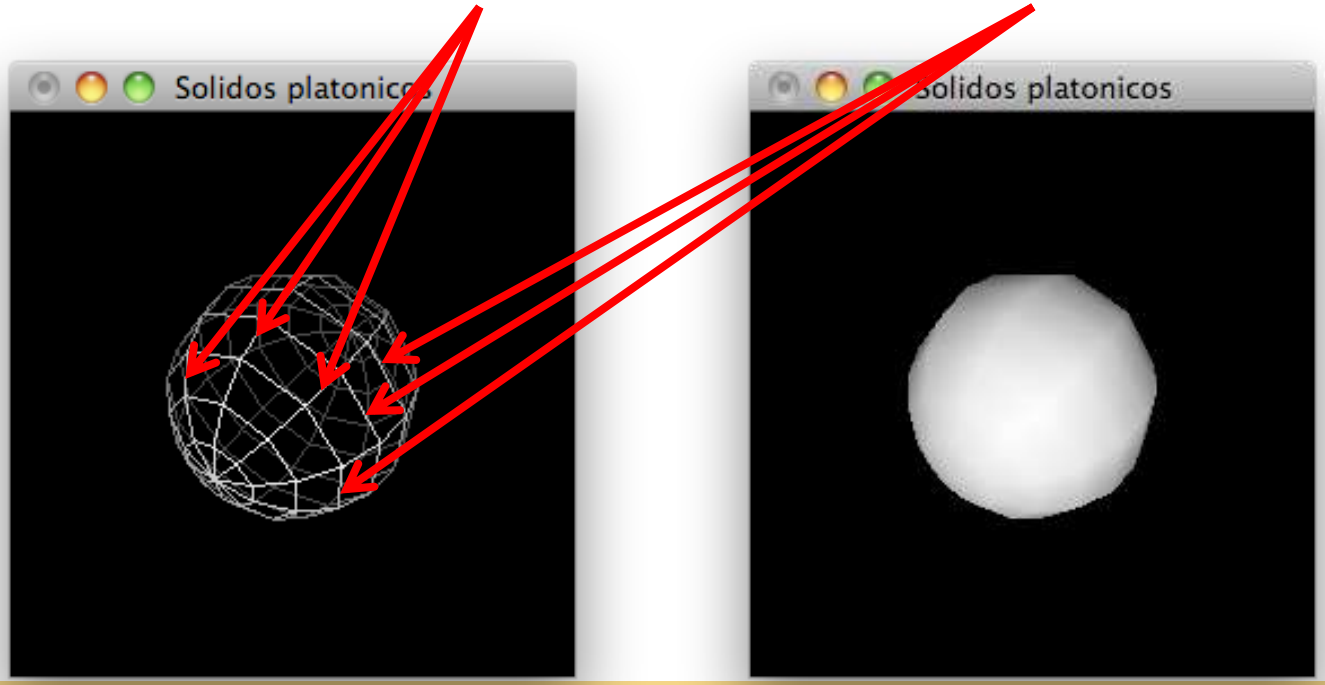
ffp@isep.ipp.pt

Sólidos "Primitivos"

- Sólidos primitivos da GLUT
- Sólidos primitivos da GLU
- Iluminação básica

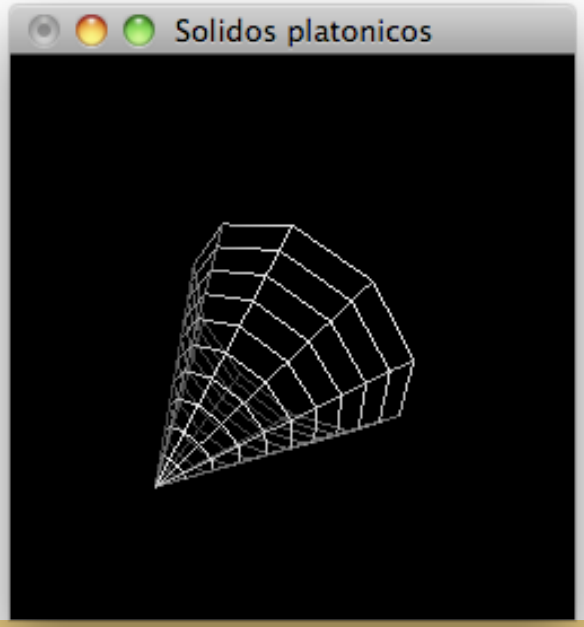
Esferas

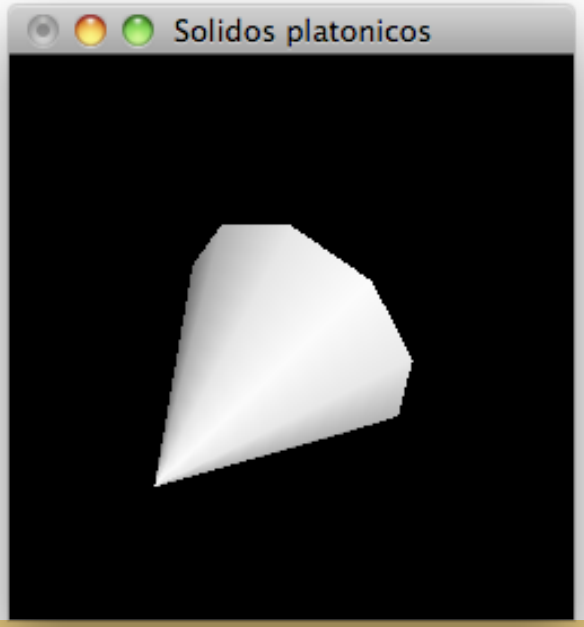
- `glutWireSphere(GLdouble radius, GLint slices, GLint stacks);`
- `glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);`



Cones

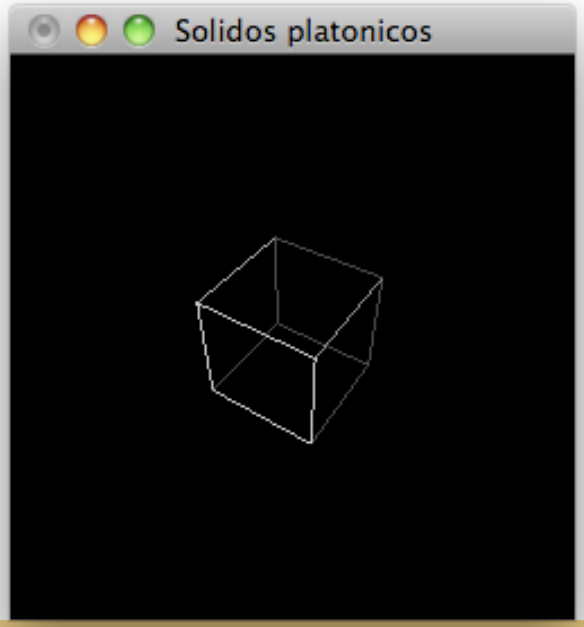
- `glutWireCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);`
- `glutSolidCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);`

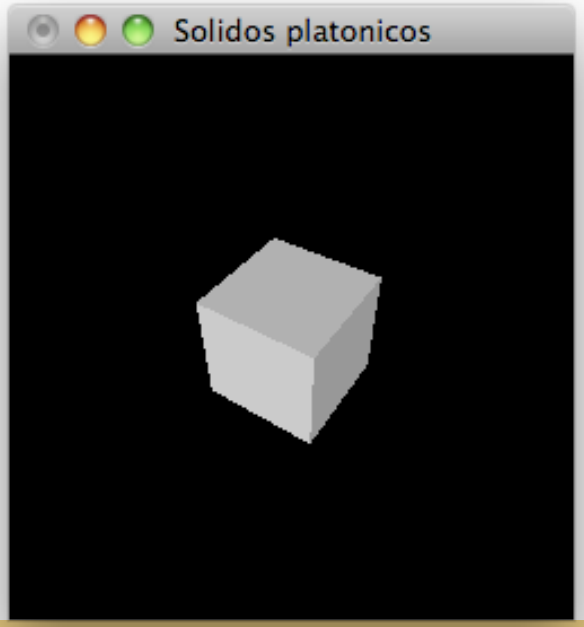




Cubos

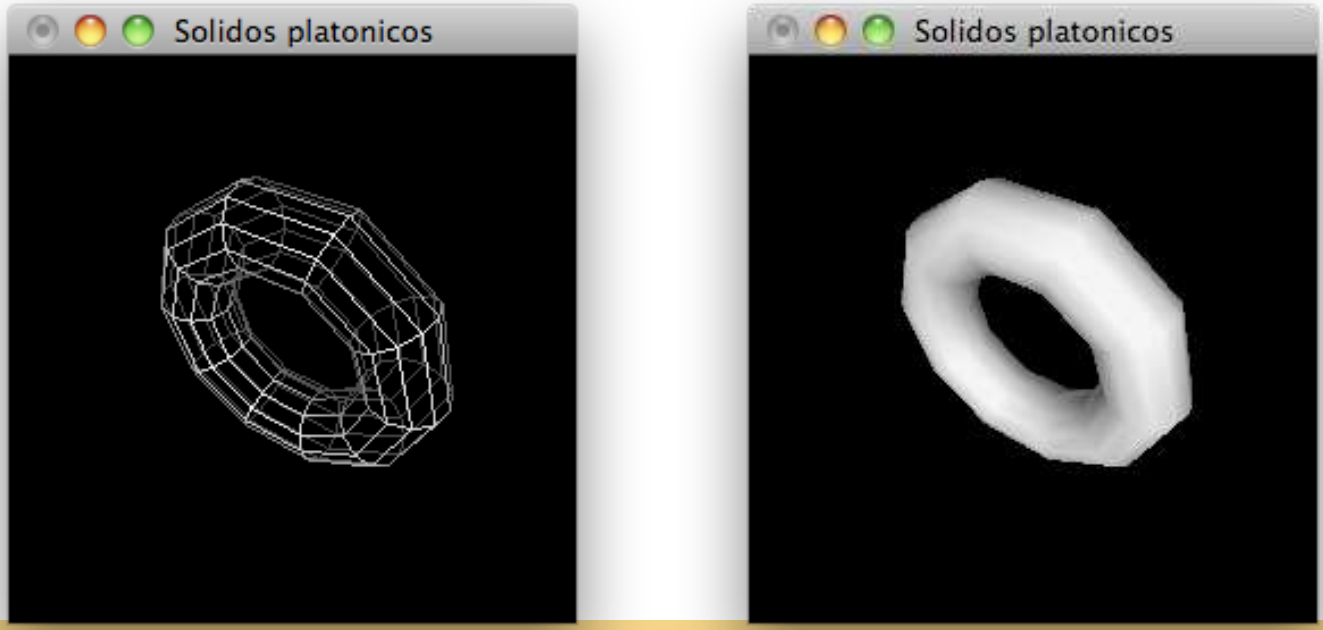
- `glutWireCube(GLdouble size);`
- `glutSolidCube(GLdouble size);`





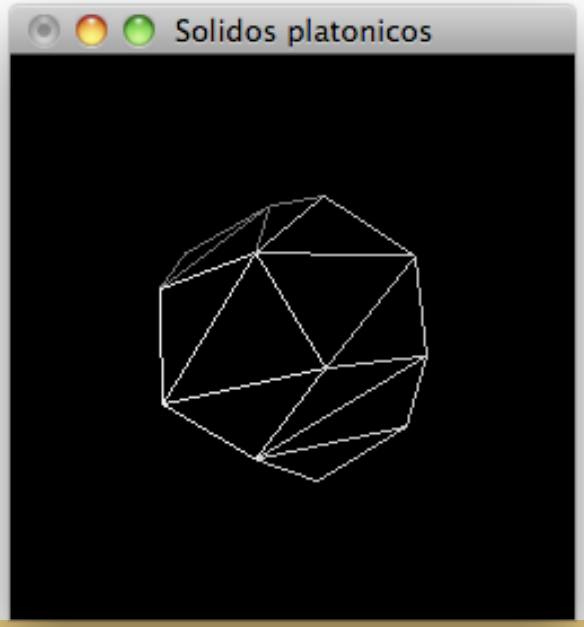
Torus

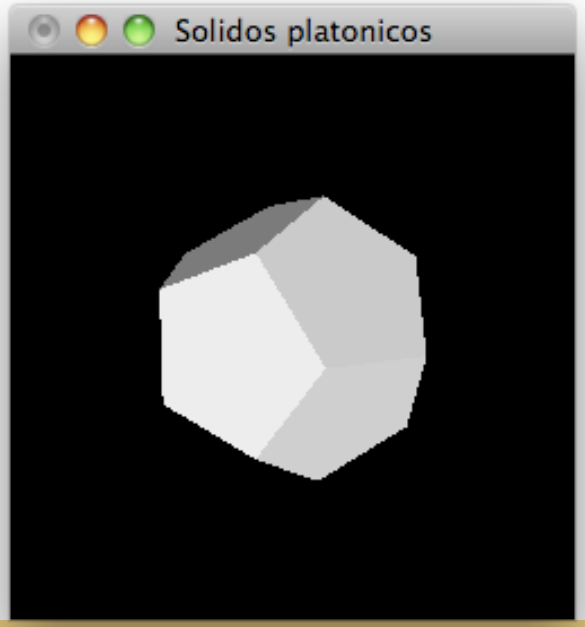
- `glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings);`
- `glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings);`



Dodecaedro

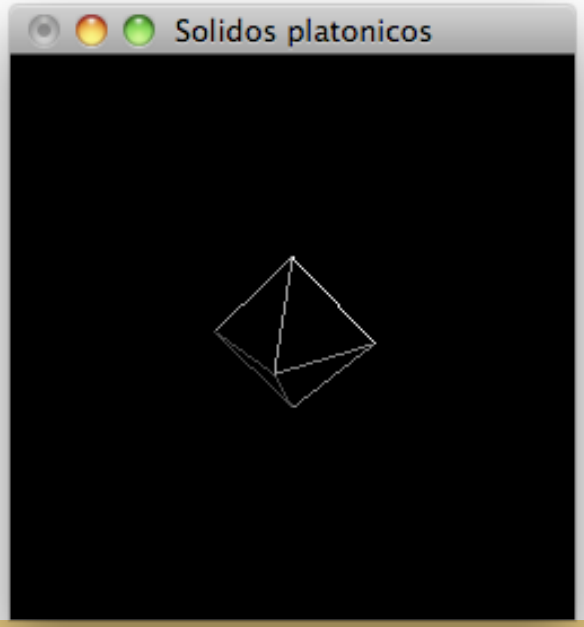
- `glutWireDodecahedron()`
- `glutSolidDodecahedron()`

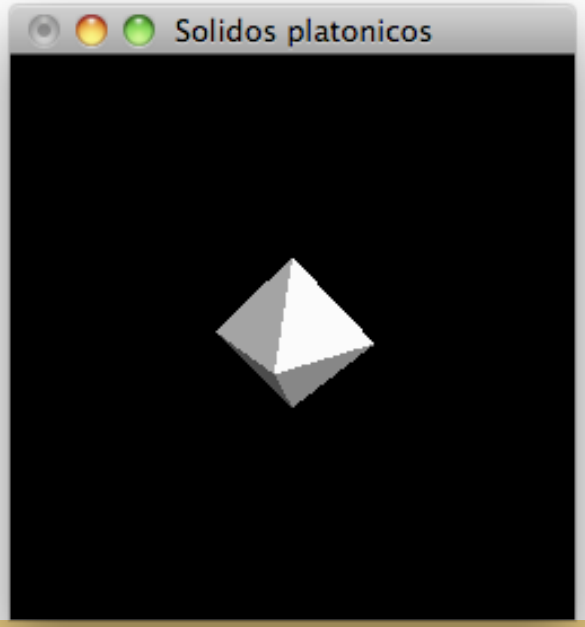




Octaedro

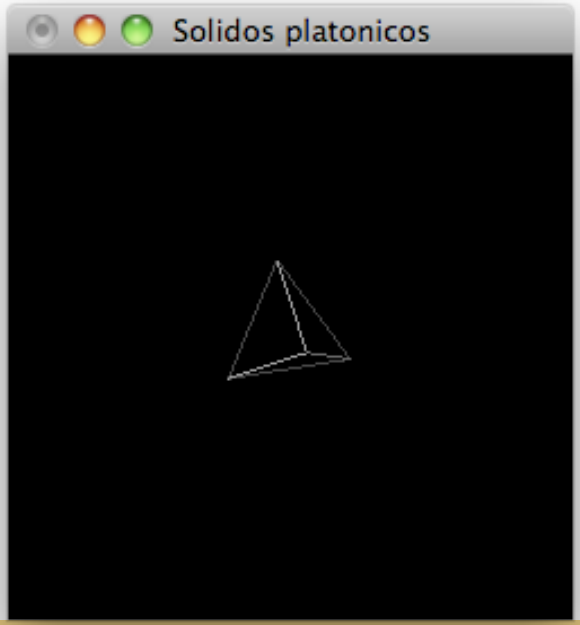
- `glutWireOctahedron()`
- `glutSolidOctahedron()`

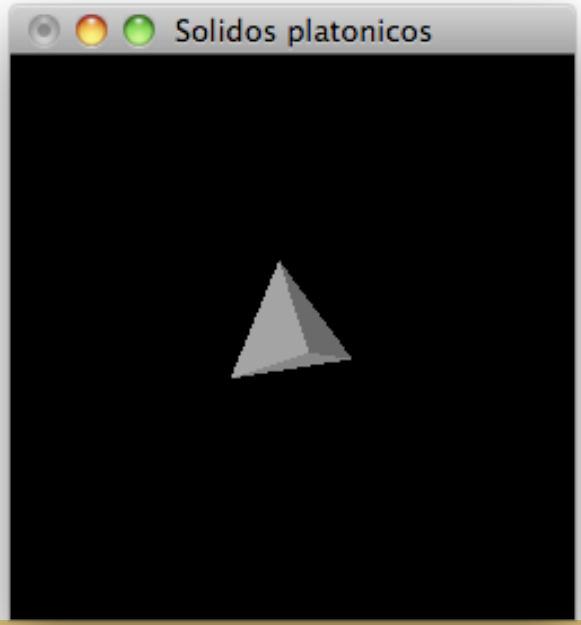




Tetraedro

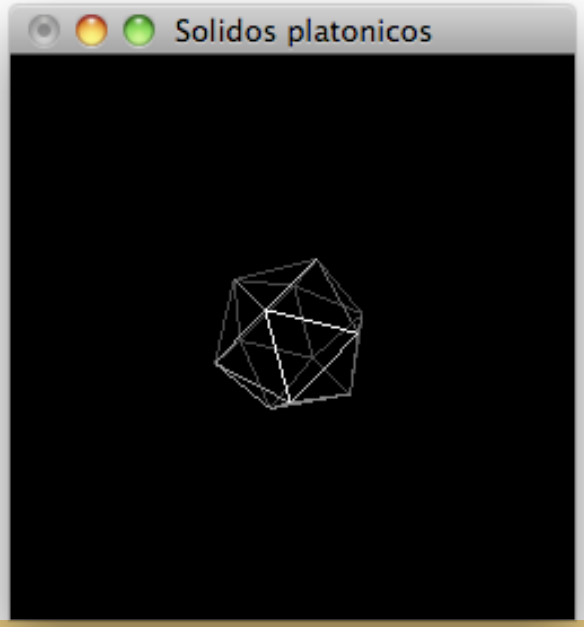
- `glutWireTetrahedron()`
- `glutSolidTetrahedron()`

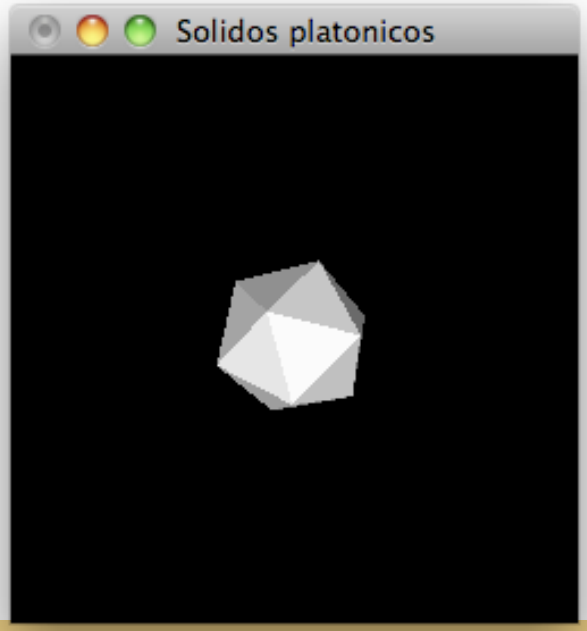




Icosaedro

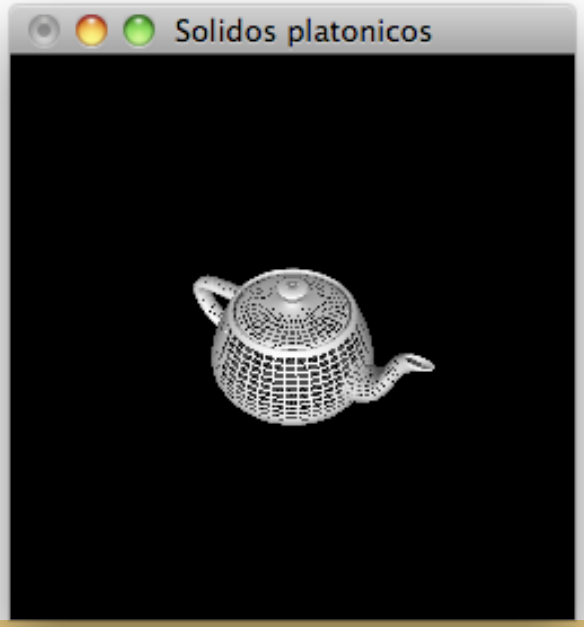
- `glutWireIcosahedron()`
- `glutSolidIcosahedron()`

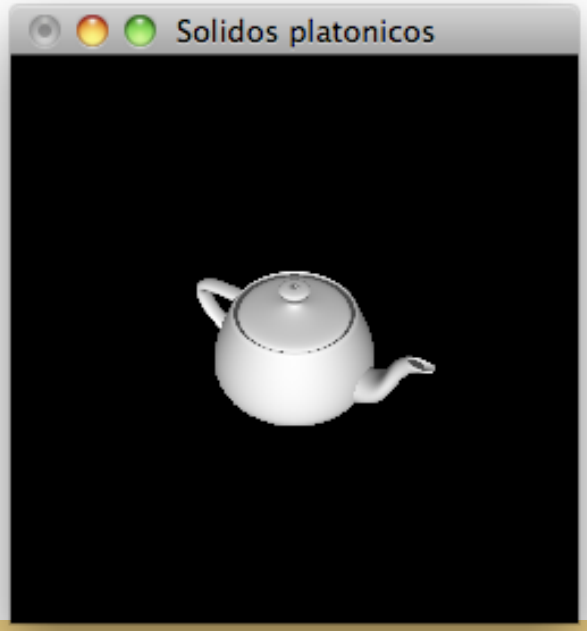




Bule de chá

- `glutWireTeapot(GLdouble size)`
- `glutSolidTeapot(GLdouble size)`





GLU quadrics

- Objectos descritos por uma equação quadrática
- Utilização da interface GLU quadrics:
  1. Criar um objecto (pode ser global na função init)
     `GLUQuadric *quad=gluNewQuadric()`
  2. Especificar atributos de desenho:
     1. **`gluQuadricOrientation()`** controla se a frente do sólido é para o interior (GLU_OUTSIDE) ou exterior (GLU_INSIDE)
     2. **`gluQuadricDrawStyle()`** controla a representação do sólido (GLU_FILL - a cheio, GLU_LINE - só arestas, GLU_POINT – só os vértices ou GLU_SILHOUETTE identico a GLU_LINE mas omite arestas de faces co-planares)
     3. **`gluQuadricNormals()`** específica o tipo de normais a ser geradas (GLU_NONE - nenhuma , GLU_FLAT - por face, GLU_SMOOTH - por vértice)

GLU quadrics

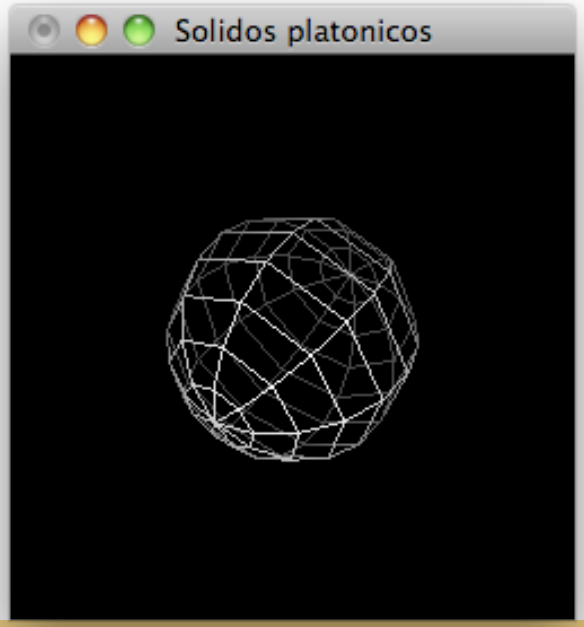
- Utilização da interface GLU quadrics:
  1. Criar um objecto (pode ser global na função init)
     `GLUQuadric *quad=gluNewQuadric()`
  2. Especificar atributos de desenho (opcional):
     1. `...`
     2. `...`
     3. `...`
     4. **`gluQuadricTexture()`** – controla a geração de coordenadas para o mapeamento de texturas (GLU_TRUE ou GLU_FALSE)
  3. Registar *callback* de erros com (opcional)
     `gluQuadricCallback()`
  4. Construir o objecto **`gluSphere()`**, **`gluCylinder()`**, **`gluDisk()`**, ou **`gluPartialDisk()`**

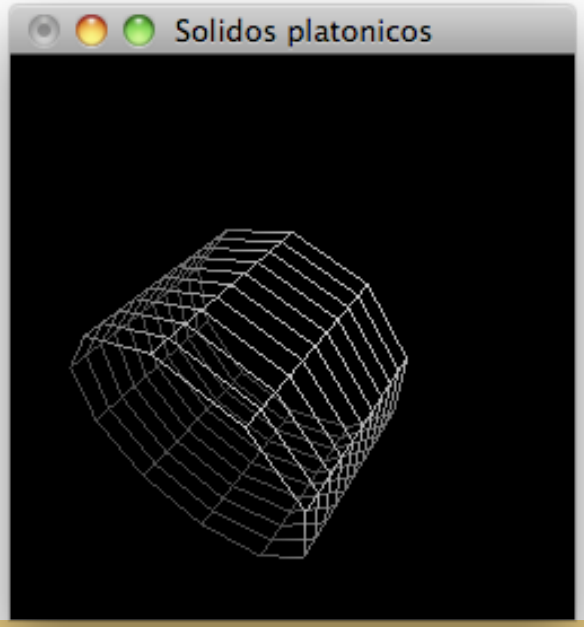
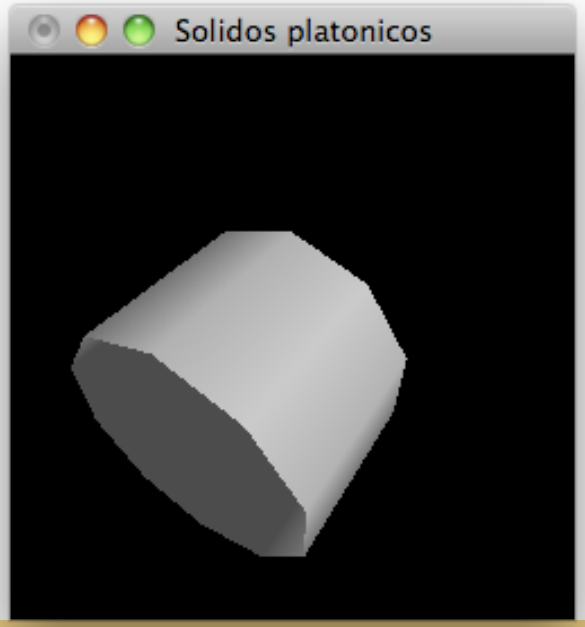
GLU quadrics

```
// criar e destruir objectos quadric não é o mais eficiente
// o objecto deveria ser colocado numa Display List e
// reutilizado quando necessário

void cylinder(GLenum mode)
{
    GLUquadricObj* qobj = gluNewQuadric();
    gluQuadricDrawStyle(qobj, mode); //GLU_LINE ou GLU_FILL
    gluQuadricNormals(qobj, GLU_SMOOTH);
    gluCylinder(qobj, 1.5, 1.5, 2, slices, stacks);
    gluDeleteQuadric(qobj);
}
```

GLU Sphere

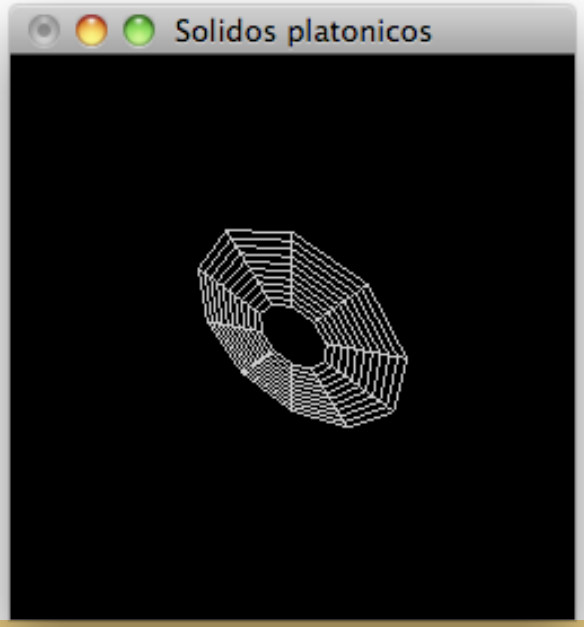
- ```
  void gluSphere (GLUquadric *qobj,
                  GLdouble radius,
                  GLint slices,
                  GLint stacks);
  ```

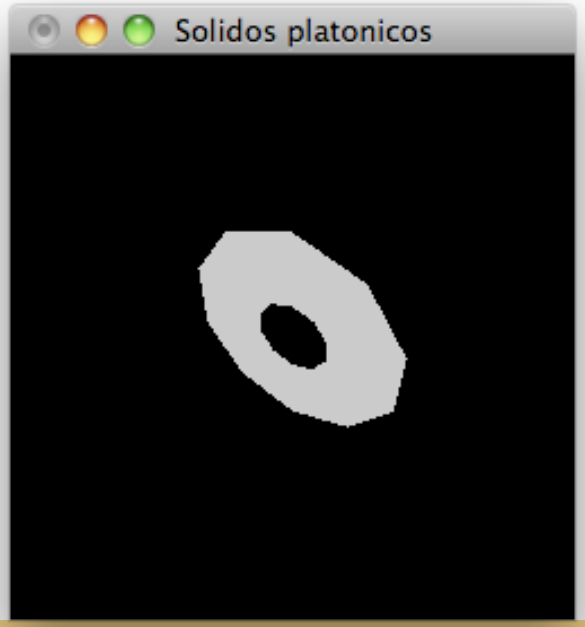



  ```

GLU Cylinder

- ```
  void gluCylinder(GLUquadric *qobj,
       GLdouble baseRadius,
       GLdouble topRadius,
       GLdouble height,
       GLint slices, GLint stacks)
  ```




  ```

GLU Disk

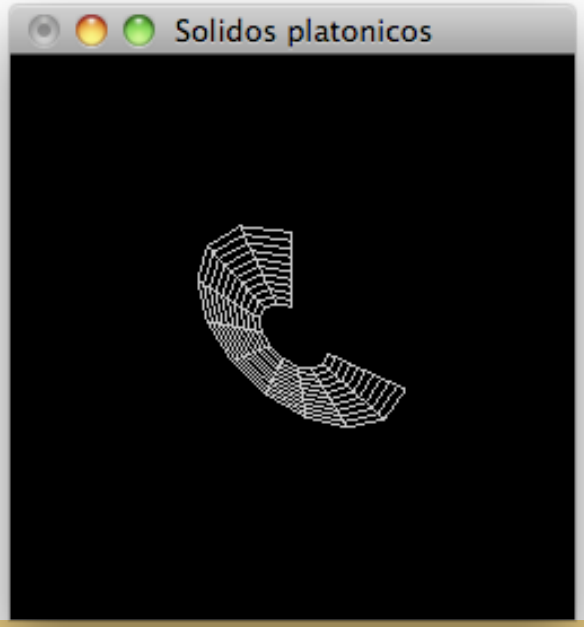
- `void gluDisk(GLUquadric *qobj,`
  `GLdouble innerRadius,`
  `GLdouble outerRadius,`
  `GLint slices, GLint loops);`



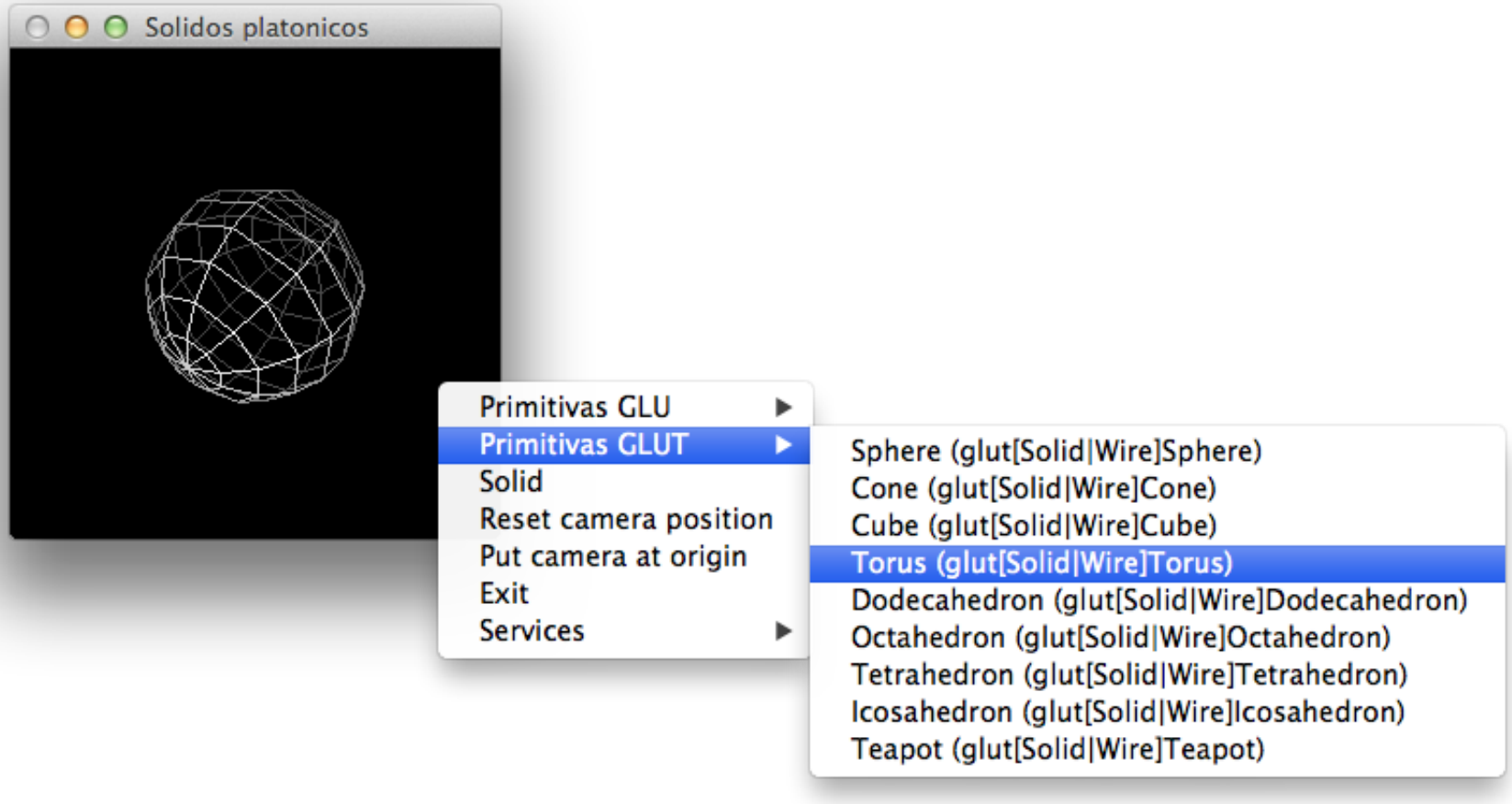


GLU Partial disk

- `void gluPartialDisk(GLUquadric *qobj, GLdouble innerRadius, GLdouble outerRadius, GLint slices, GLint loops, GLdouble startAngle, GLdouble sweepAngle);`



Demo


Solidos platonicos
Primitivas GLU
Primitivas GLUT
Solid
Reset camera position
Put camera at origin
Exit
Services
Sphere (glut[Solid|Wire]Sphere)
Cone (glut[Solid|Wire]Cone)
Cube (glut[Solid|Wire]Cube)
Torus (glut[Solid|Wire]Torus)
Dodecahedron (glut[Solid|Wire]Dodecahedron)
Octahedron (glut[Solid|Wire]Octahedron)
Tetrahedron (glut[Solid|Wire]Tetrahedron)
Icosahedron (glut[Solid|Wire]Icosahedron)
Teapot (glut[Solid|Wire]Teapot)

Iluminação básica

```
void init()
{
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);

    glFrontFace(GL_CW); // Problema nas normais da teapot
    glEnable(GL_COLOR_MATERIAL); // Usar cor como material
}

void display()
{
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    ...
    glColor3f(1.0f,1.0f,1.0f);
    glutSolidTeapot(1);
    ...
}
```

"ligar" luz 0
(branca por omissão)
e teste de profundidade

Módulo 7

Sistemas Gráficos e Interação
Instituto Superior de Engenharia do Porto

Filipe Pacheco
ffp@isep.ipp.pt

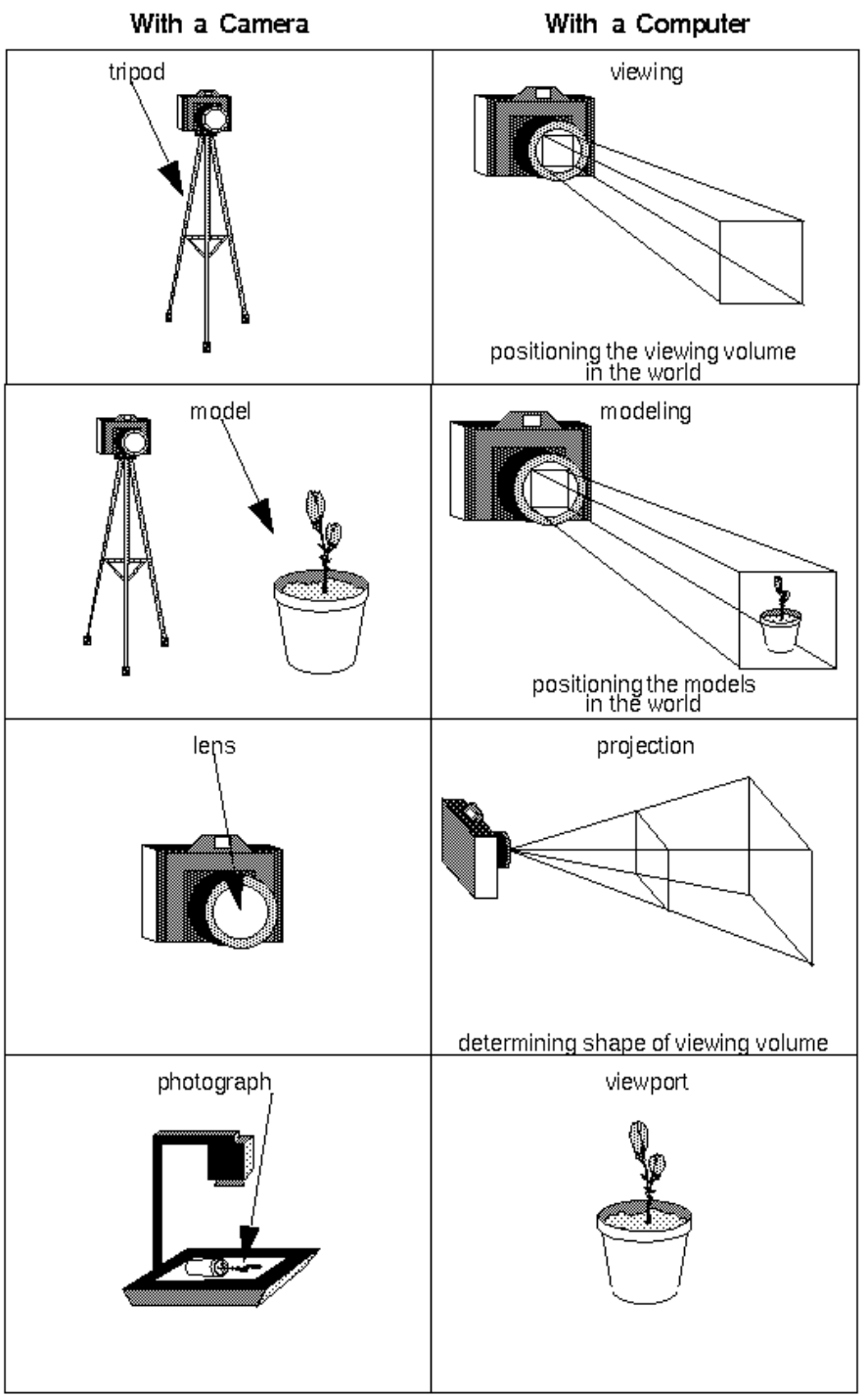
Transformações geométricas

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 projeção
- 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();
    projecao();
    ...
}
void display() {
    // modelview transformation
    glMatrixMode(GL_MODELVIEW); // costuma estar na reshape
    glLoadIdentity();
    // posicionamento da câmara
    camara();
    // transformações do modelo
    ...
}
```

Evitar transformações desnecessárias

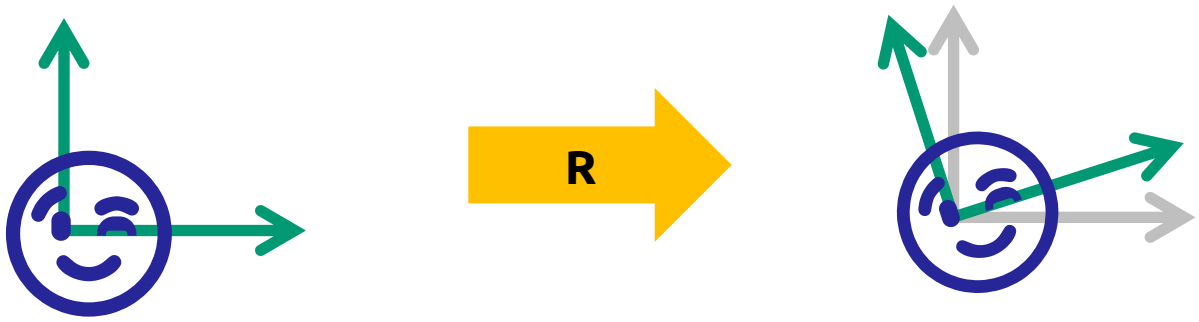
- Deve-se evitar cálculos acumulados nas matrizes
  - **Exemplo:** não ir aplicando rotações na mesma matriz, mas sim ter uma variável com o ângulo de rotação

Inicializar as matrizes

- Antes de definir transformações deve-se inicializar a matriz de transformação usando **`glLoadIdentity()`**

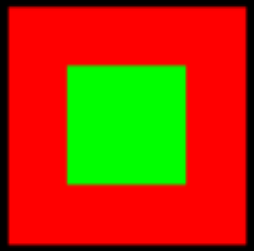
Rotações

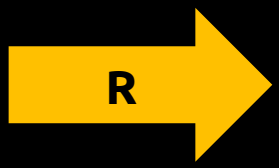
- `void glRotate{f|d}(angle, ex, ey, ez)`

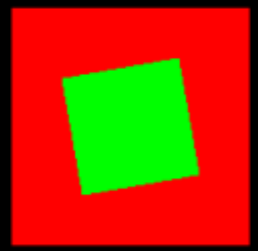


Rotações

```
glLoadIdentity();
glColor3f(1, 0, 0);
glRectf(-0.5,-0.5,0.5,0.5);
glRotatef(10, 0, 0, 1);
glColor3ub(0, 255, 0);
glRectf(-0.25,-0.25,0.25,0.25);
```

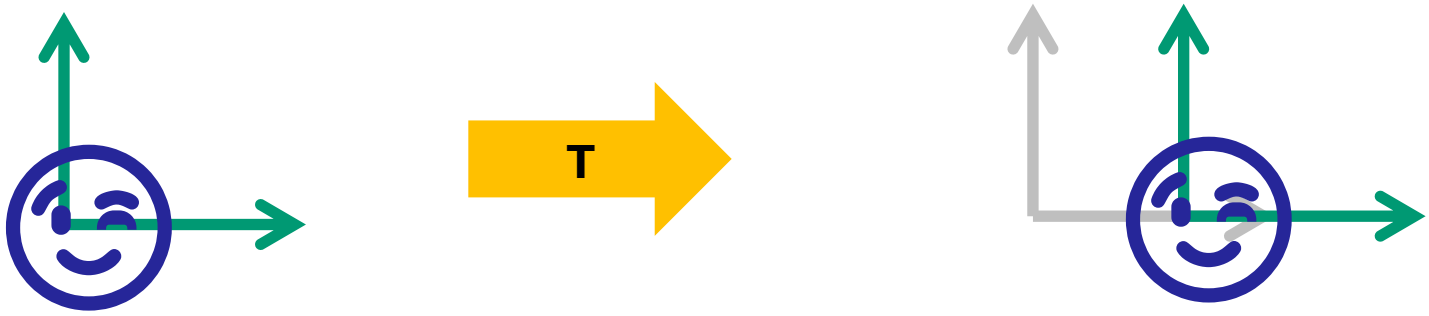






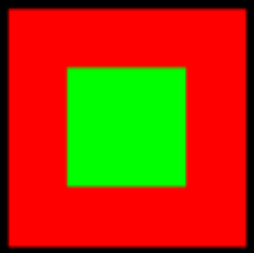
Translações

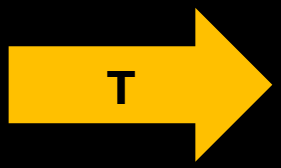
- `void glTranslate{f|d}(dx, dy, dz);`

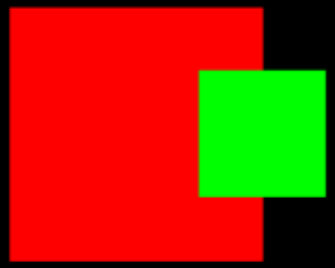


Translações

```
glLoadIdentity();
glColor3f(1, 0, 0);
glRectf(-0.5,-0.5,0.5,0.5);
glTranslatef(0.5, 0.0, 0.0);
glColor3ub(0, 255, 0);
glRectf(-0.25,-0.25,0.25,0.25);
```

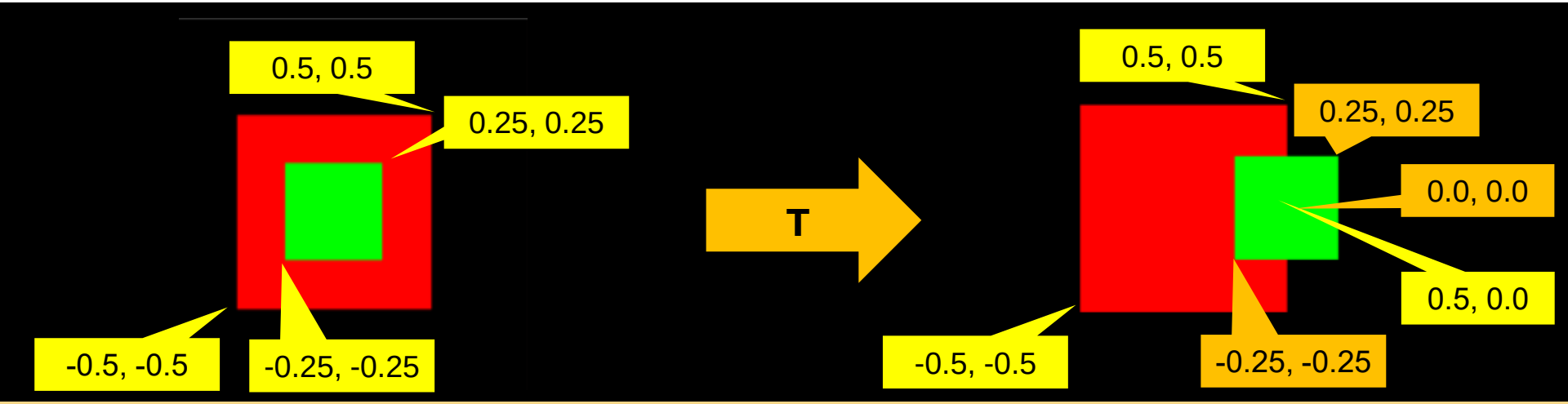






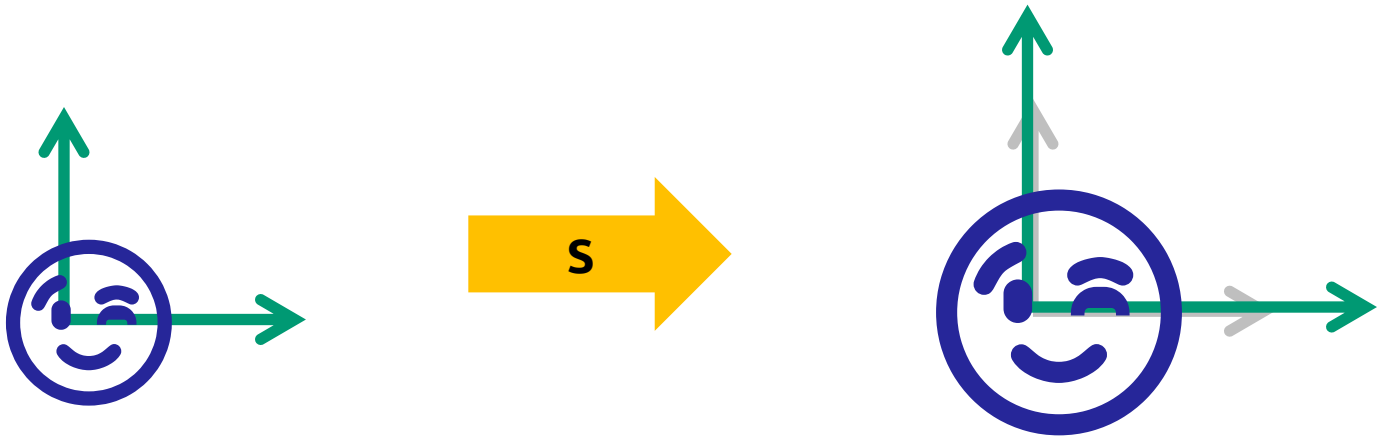
Translações

```
glLoadIdentity();
glColor3f(1, 0, 0);
glRectf(-0.5,-0.5,0.5,0.5);
glTranslatef(0.5, 0.0, 0.0);
glColor3ub(0, 255, 0);
glRectf(-0.25,-0.25,0.25,0.25);
```



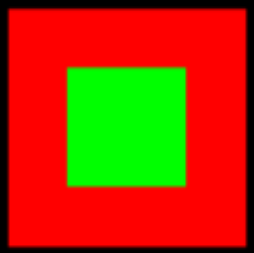
Escalamento

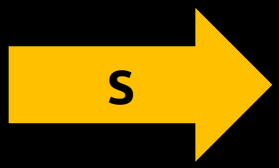
- **`void glScale{f|d}`**
  `sx, sy, sz)`
- evitar escalas não proporcionais (sx != sy != sz) pois alteram vectores normal, rotações, etc…

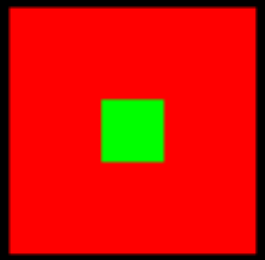


Escalamento

```
glLoadIdentity();
glColor3f(1, 0, 0);
glRectf(-0.5,-0.5,0.5,0.5);
glScalef(0.5, 0.5, 0.5);
glColor3ub(0, 255, 0);
glRectf(-0.25,-0.25,0.25,0.25);
```

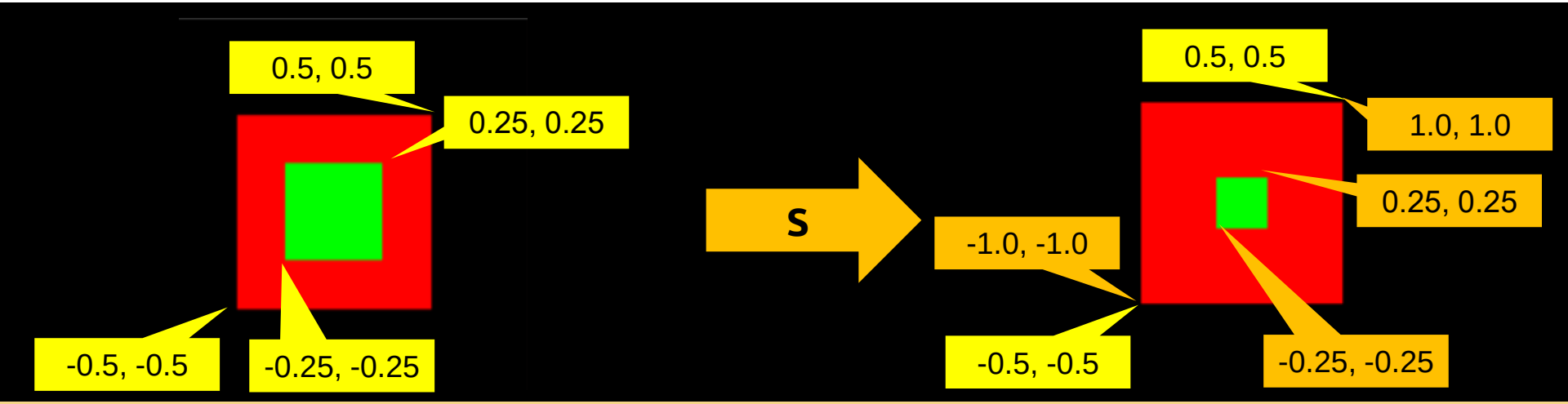




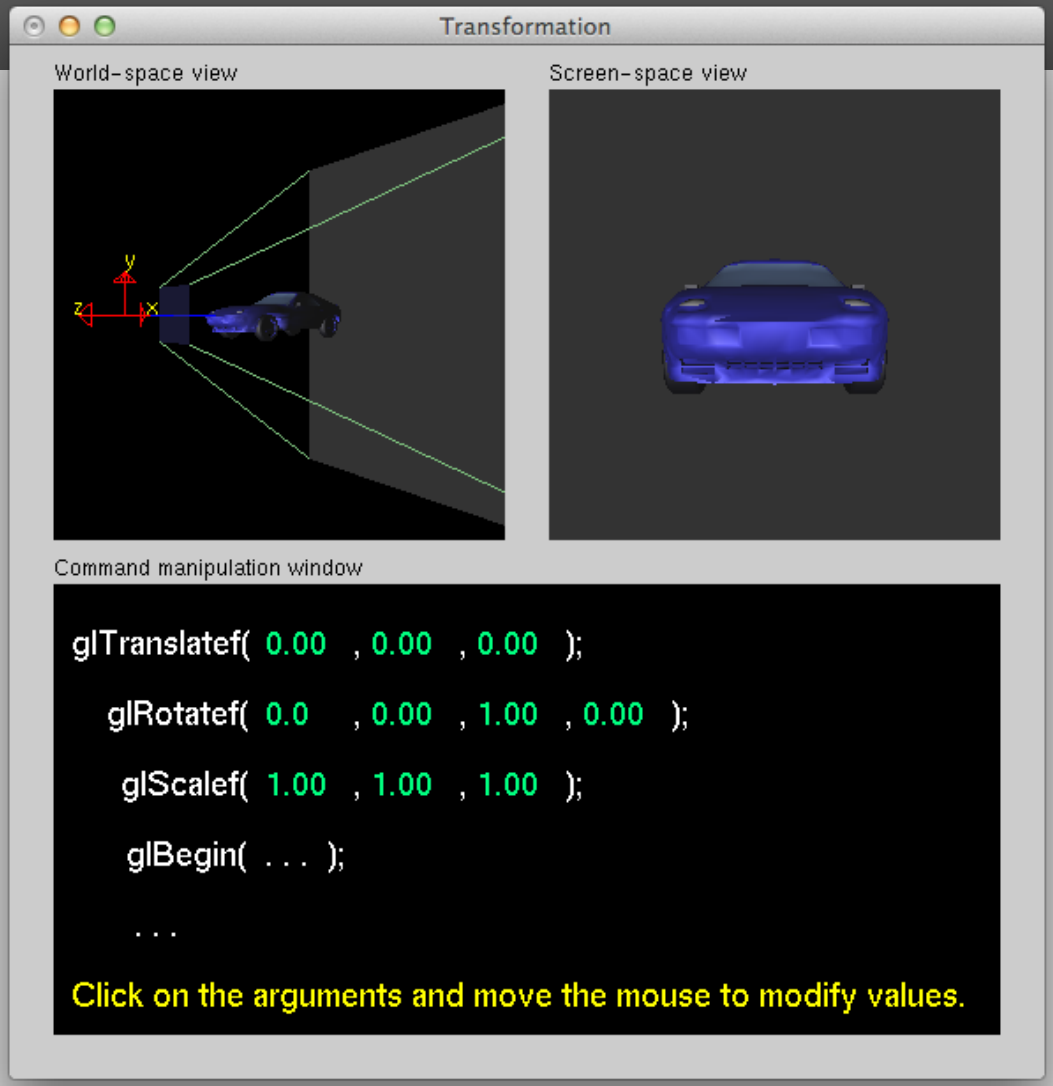


Escalamento

```
glLoadIdentity();
glColor3f(1, 0, 0);
glRectf(-0.5,-0.5,0.5,0.5);
glScalef(0.5, 0.5, 0.5);
glColor3ub(0, 255, 0);
glRectf(-0.25,-0.25,0.25,0.25);
```



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.