Módulo 4

Sistemas Gráficos e Interação

Instituto Superior de Engenharia do Porto

Filipe Pacheco

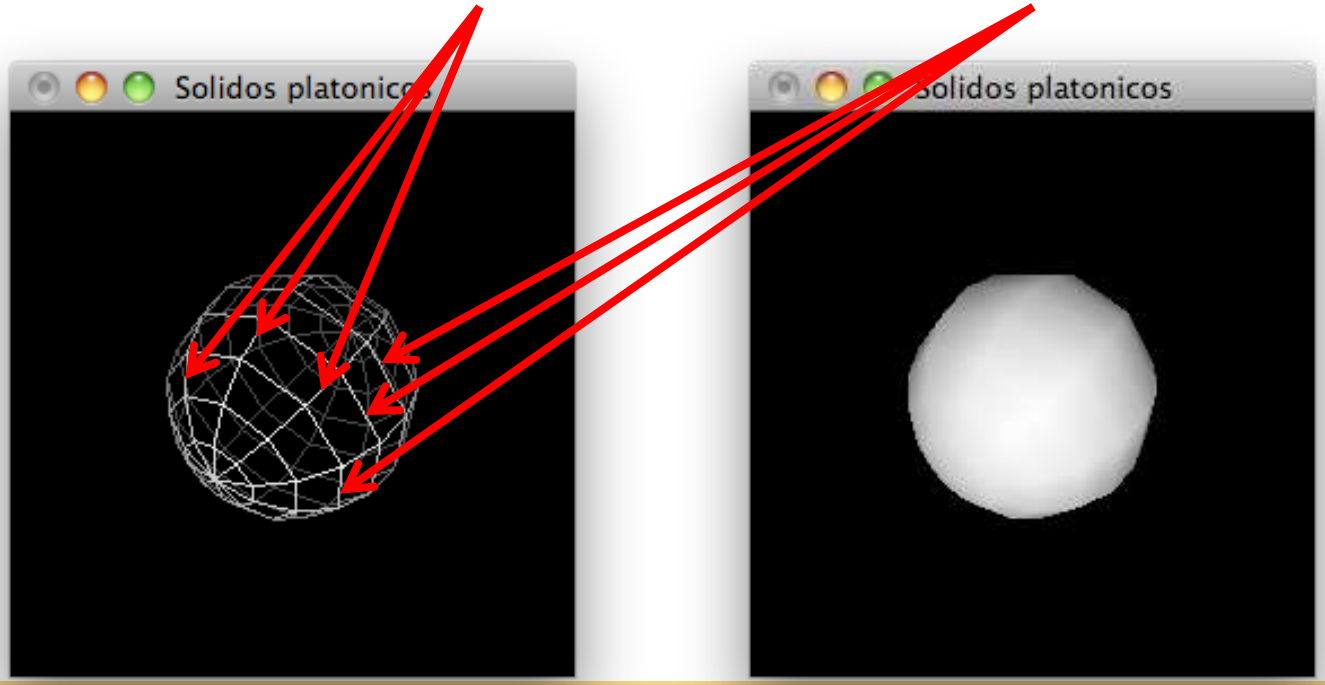
ffp@isep.ipp.pt

Sólidos "Primitivos"

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

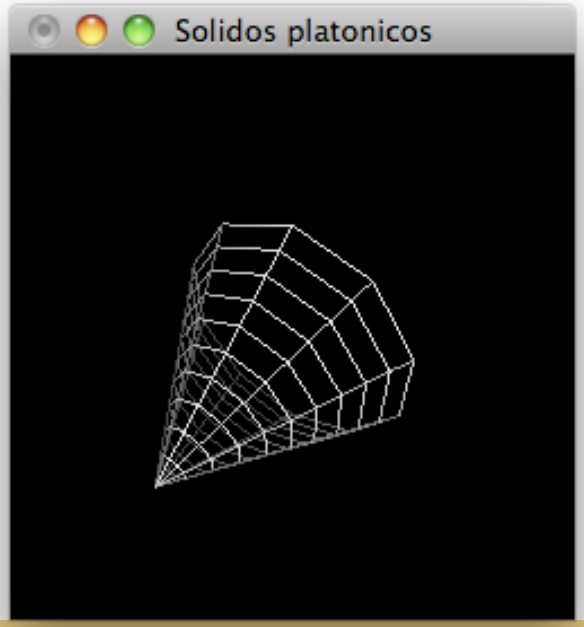
Esferas

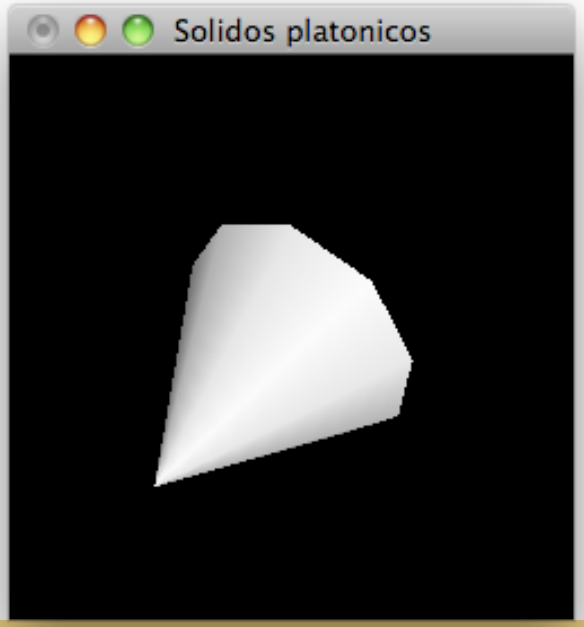
- `glutWireSphere(GLdouble radius, GLint slices, GLint stacks);`
- `glutSolidSphere(GLdouble radius, GLint slices, GLint stacks);`



Cones

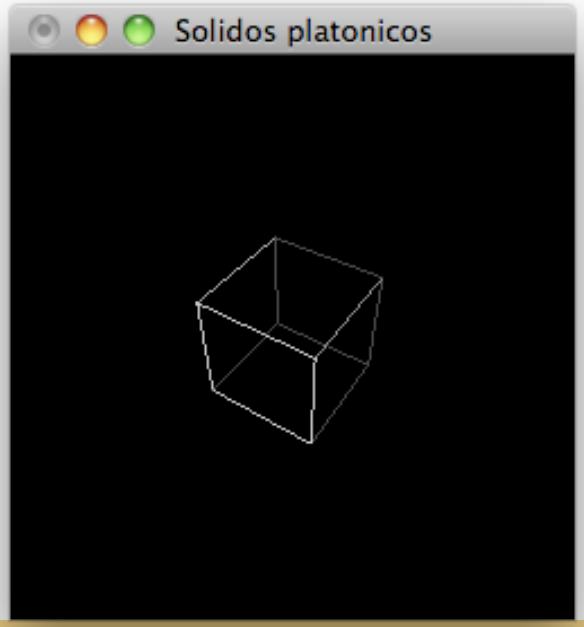
- `glutWireCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);`
- `glutSolidCone(GLdouble base, GLdouble height, GLint slices, GLint stacks);`

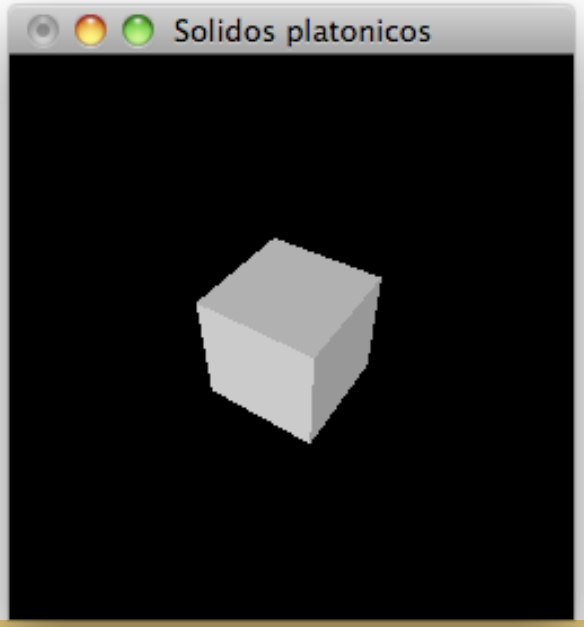




Cubos

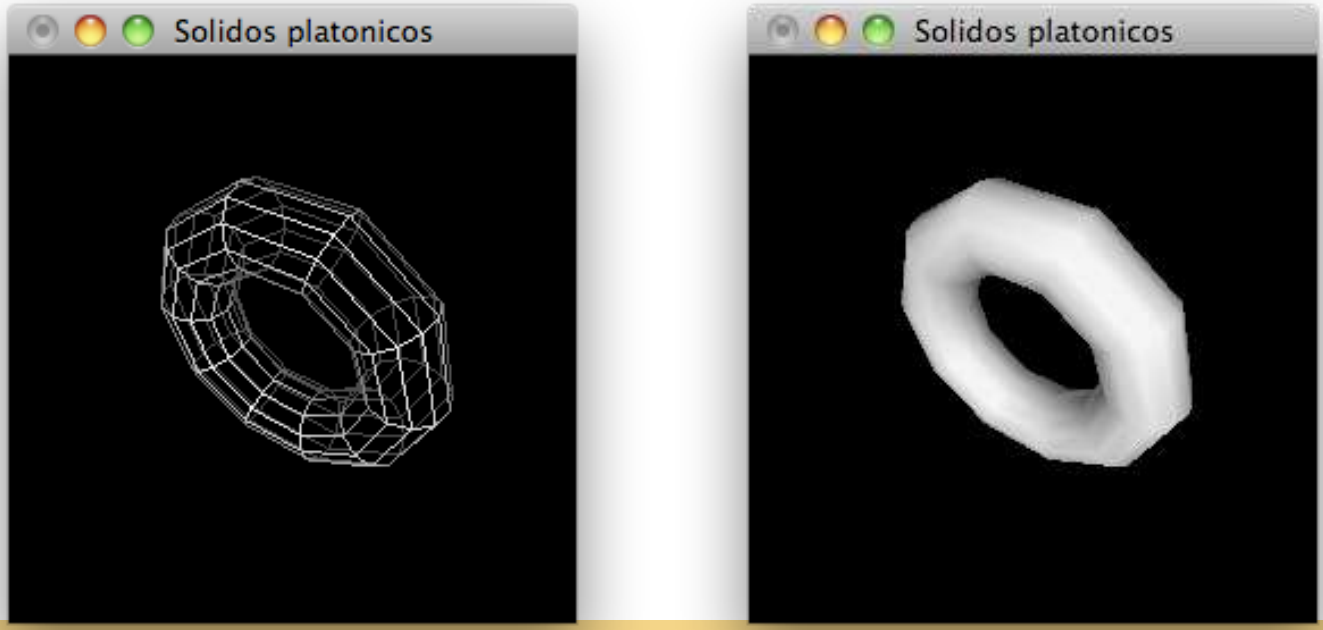
- `glutWireCube(GLdouble size);`
- `glutSolidCube(GLdouble size);`





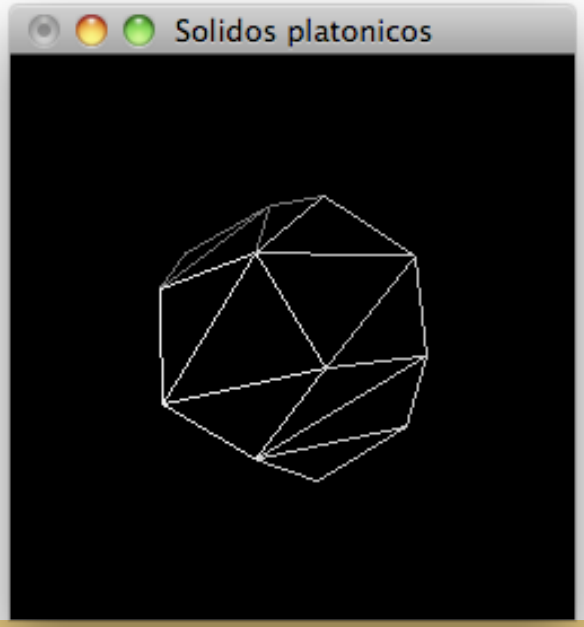
Torus

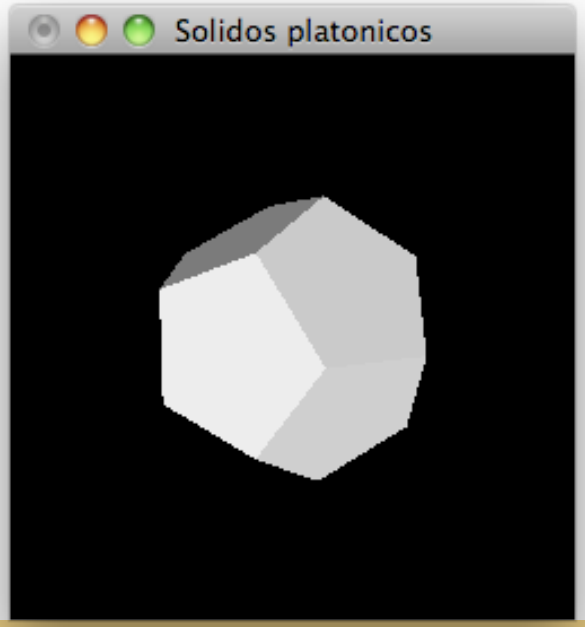
- `glutWireTorus(GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings);`
- `glutSolidTorus(GLdouble innerRadius, GLdouble outerRadius, GLint sides, GLint rings);`



Dodecaedro

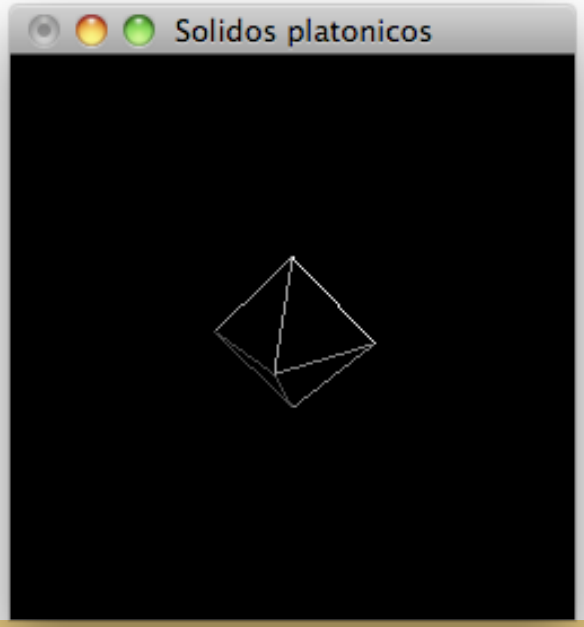
- `glutWireDodecahedron()`
- `glutSolidDodecahedron()`

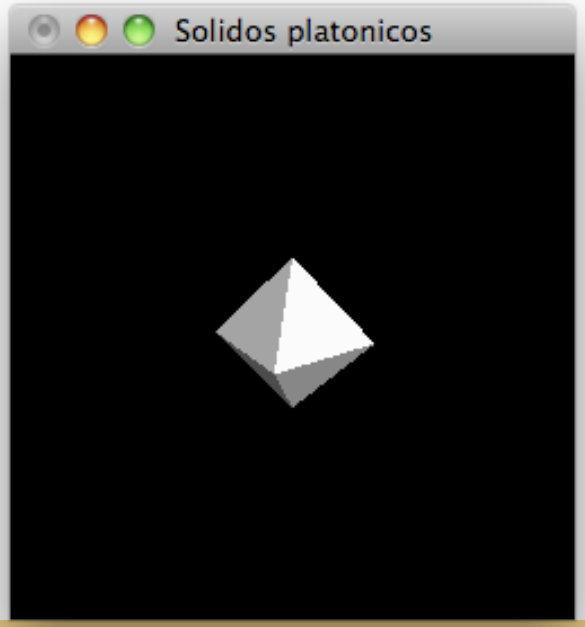




Octaedro

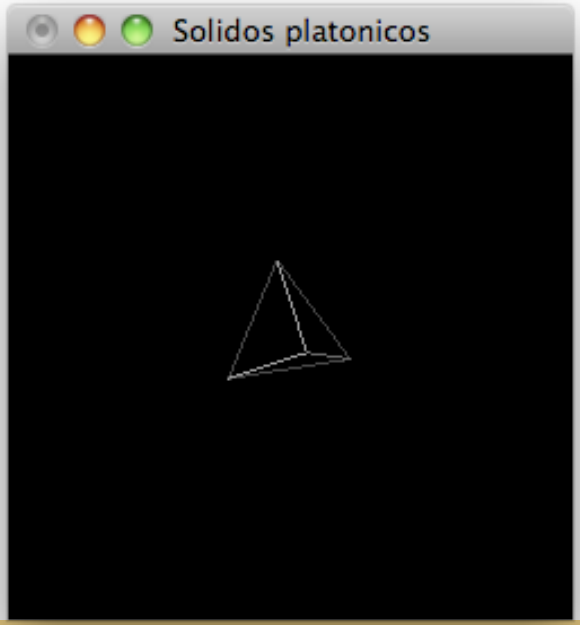
- `glutWireOctahedron()`
- `glutSolidOctahedron()`

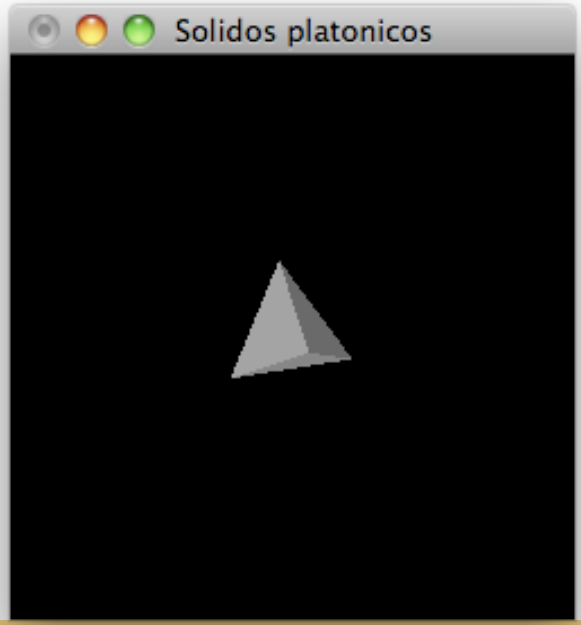




Tetraedro

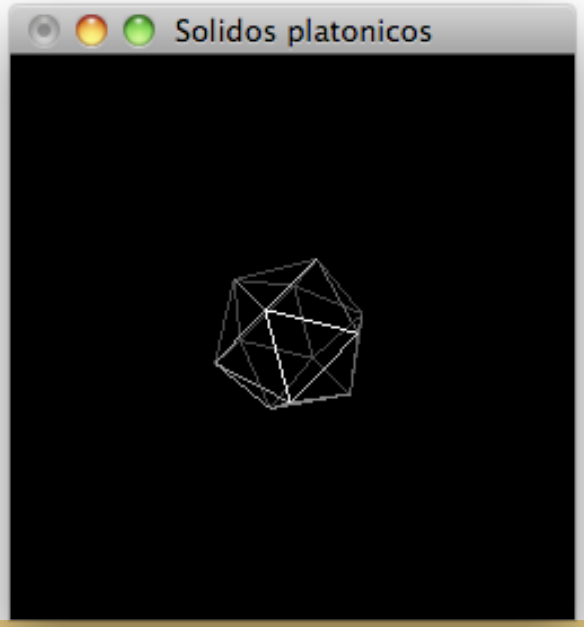
- `glutWireTetrahedron()`
- `glutSolidTetrahedron()`

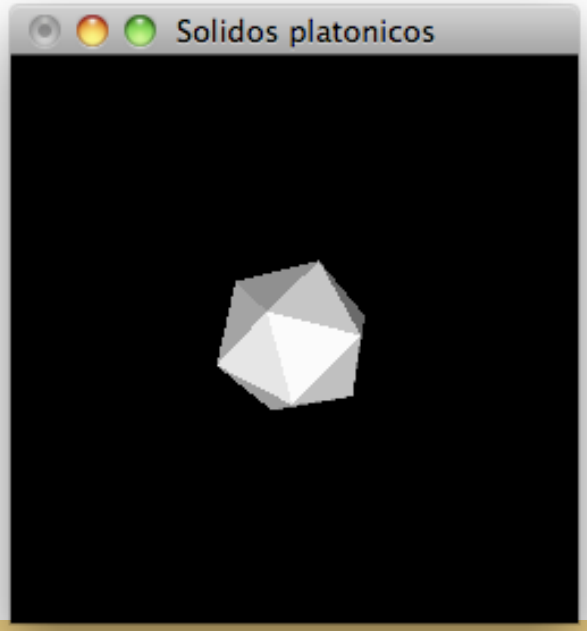




Icosaedro

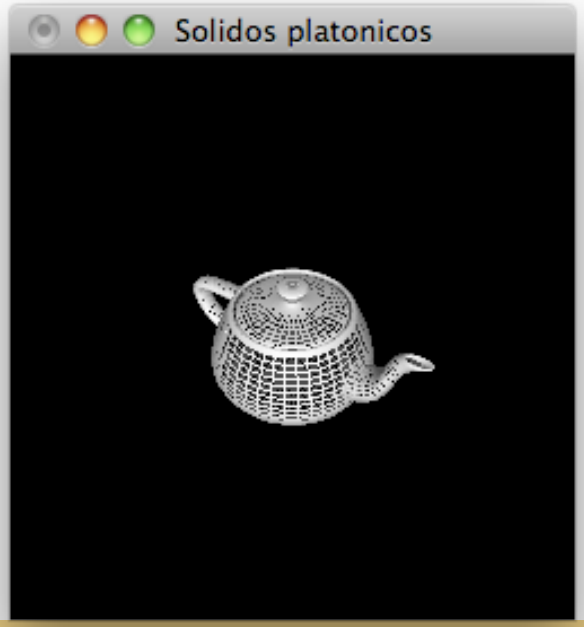
- `glutWireIcosahedron()`
- `glutSolidIcosahedron()`

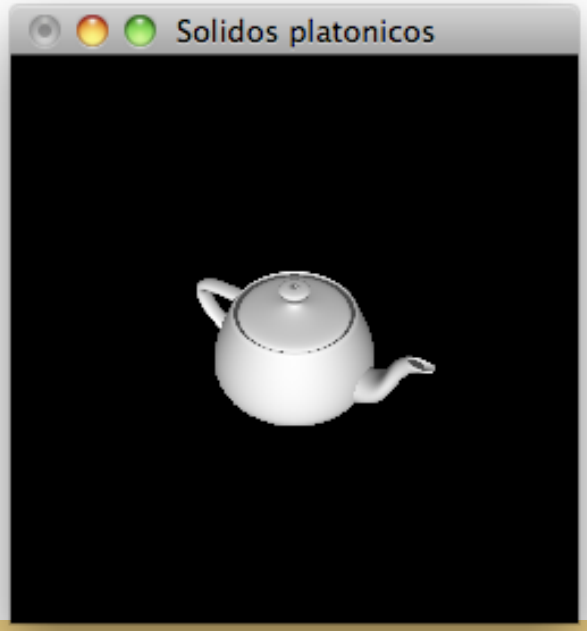




Bule de chá

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





GLU quadrics

- Objects described by a quadratic equation
- Using GLU API:
  1. Create a GLUQuadric objecto (can be global in init)
     ```
     GLUQuadric *quad=gluNewQuadric()
     ```
  2. Specify attributes:
     1. gluQuadricOrientation () controls whether the front of the solid is inward (GLU_OUTSIDE) or outward (GLU_INSIDE)
     2. gluQuadricDrawStyle () controls the representation of the solid (GLU_FILL - solid, GLU_LINE - only edges, GLU_POINT - only vertices or GLU_SILHOUETTE identical to GLU_LINE but omits edges of coplanar faces)
     3. gluQuadricNormals () specifies the type of normals to be generated (GLU_NONE - none, GLU_FLAT - by face, GLU_SMOOTH - by vertex)

GLU quadrics

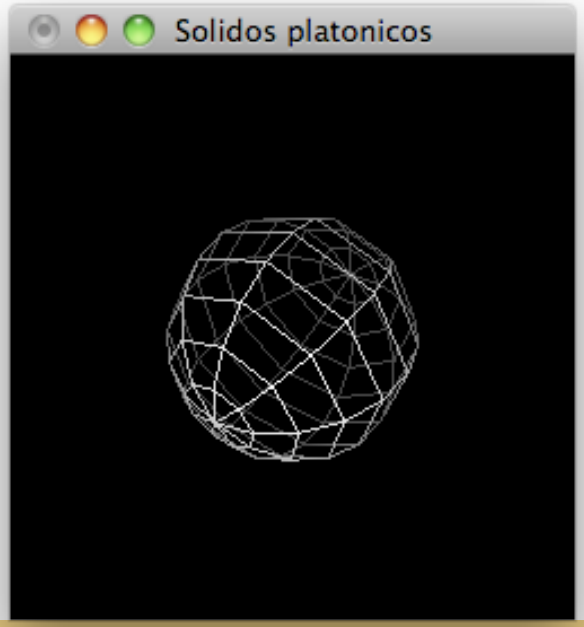
- Utilização da interface GLU quadrics:
  1. Criar um objecto (pode ser global na função init)
     `GLUQuadric *quad=gluNewQuadric()`
  2. Especificar atributos de desenho (opcional):
     1. `...`
     2. `...`
     3. `...`
     4. **`gluQuadricTexture()`** – controls the generation of coordinates for texture mapping (GLU_TRUE or GLU_FALSE)
  3. Register error *callback* (optional)
     `gluQuadricCallback()`
  4. Build the object **`gluSphere()`**, **`gluCylinder()`**, **`gluDisk()`**, or **`gluPartialDisk()`**

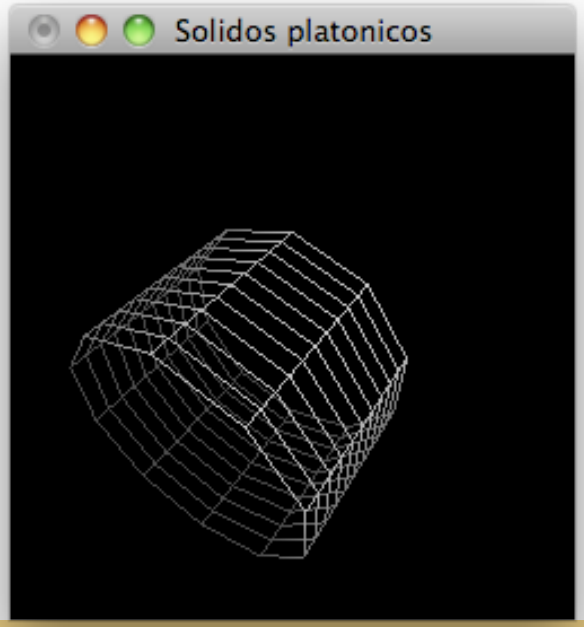
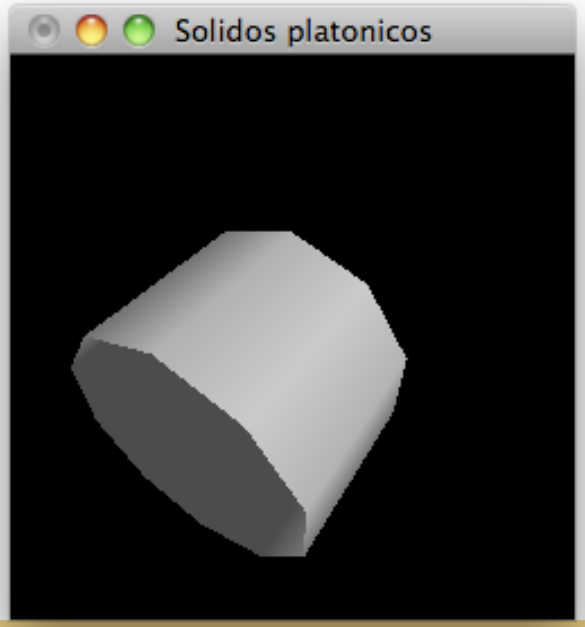
GLU quadrics

```
// 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

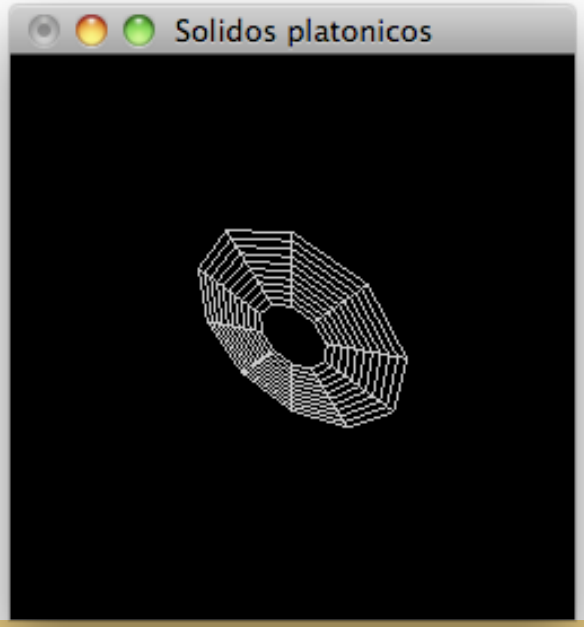
- ```
  void gluSphere (GLUquadric *qobj,
                  GLdouble radius,
                  GLint slices,
                  GLint stacks);
  ```

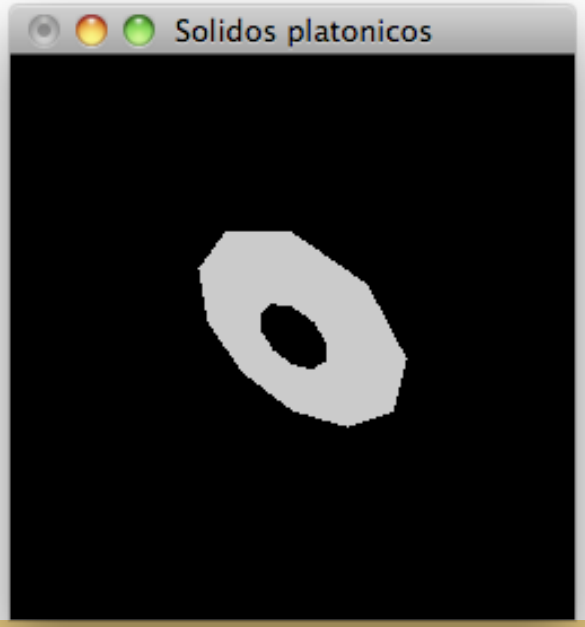



  ```

GLU Cylinder

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




  ```

GLU Disk

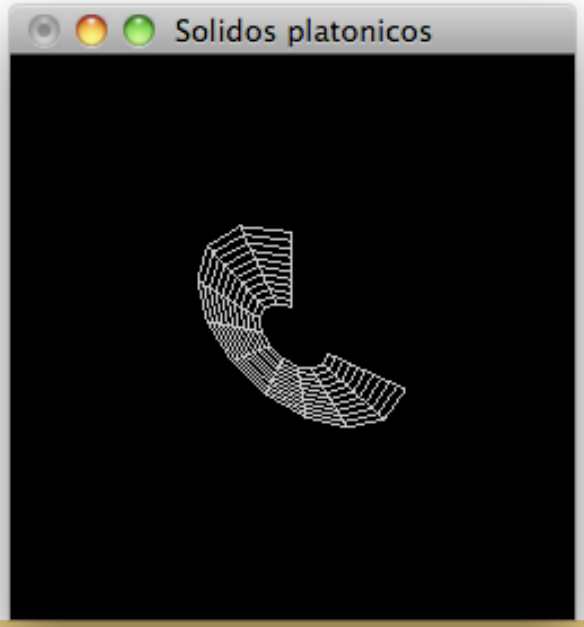
- `void gluDisk(GLUquadric *qobj,`
  `GLdouble innerRadius,`
  `GLdouble outerRadius,`
  `GLint slices, GLint loops);`



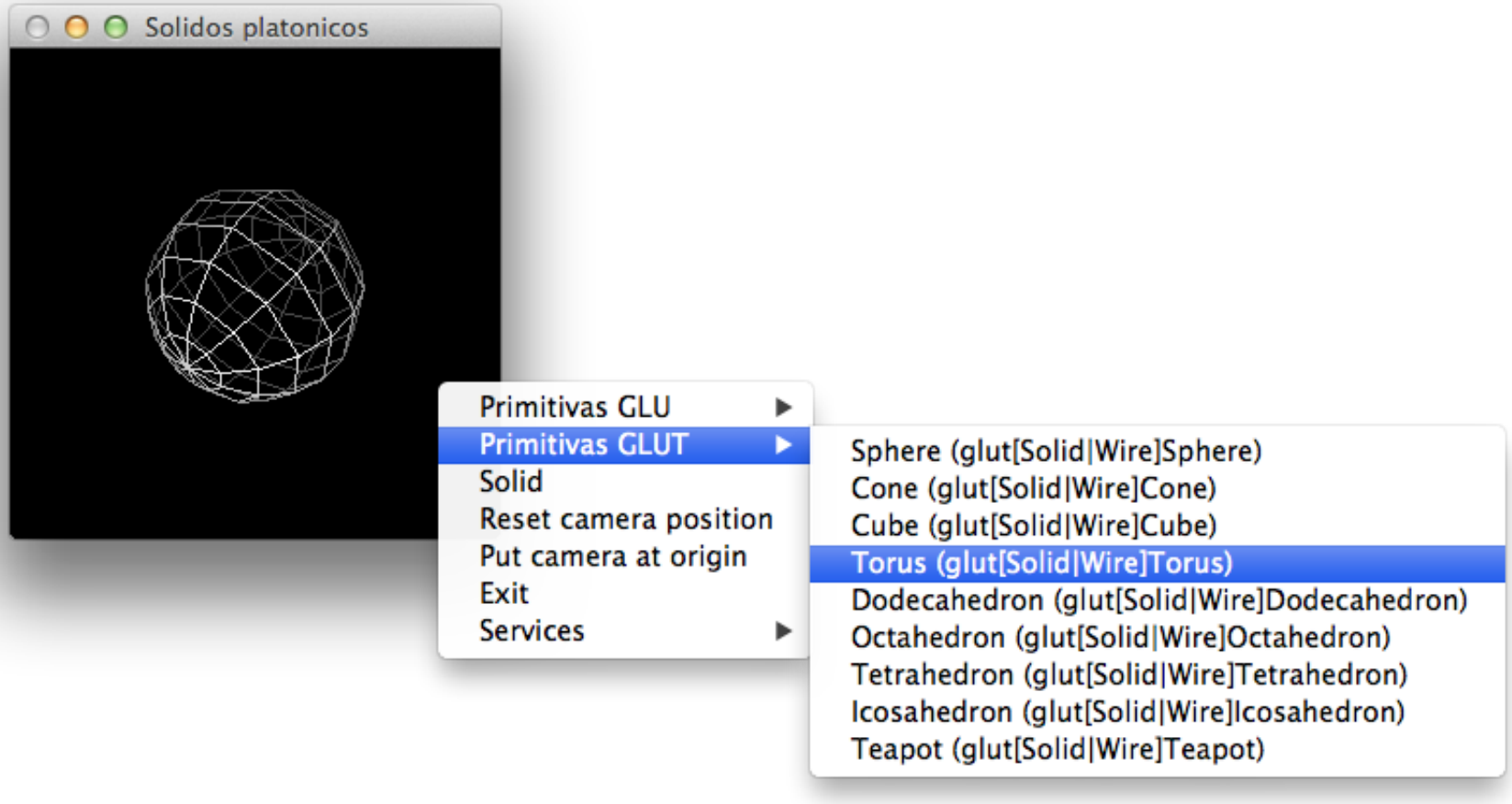


GLU Partial disk

- `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); // solve issue in teapot object
    glEnable(GL_COLOR_MATERIAL); // Use color as material
}

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

"Turn on" light 0 (white by default) and depth test

Módulo 7

Sistemas Gráficos e Interação
Instituto Superior de Engenharia do Porto

Filipe Pacheco
ffp@isep.ipp.pt

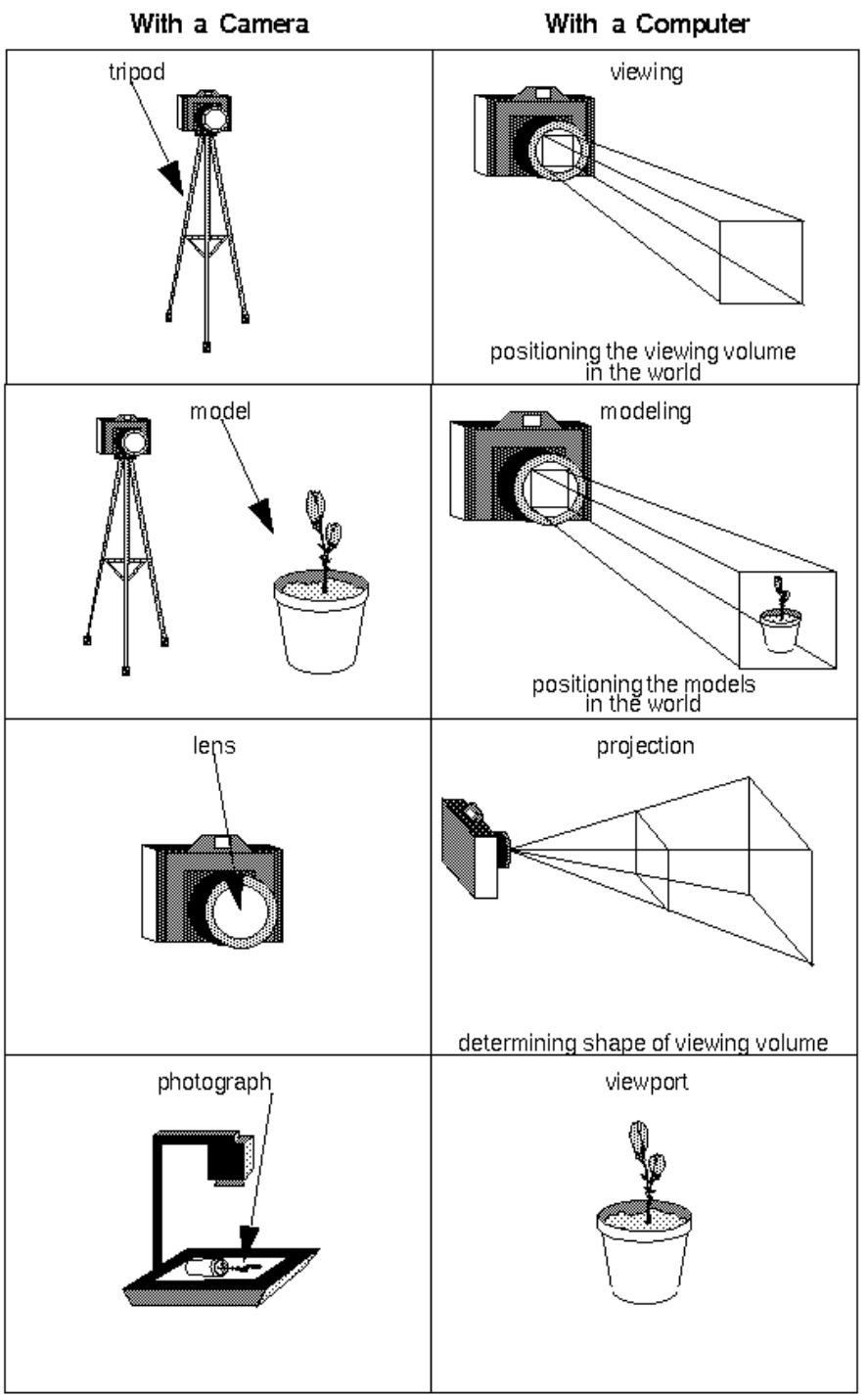
Transformações geométricas

Conteúdo

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

Analogia da câmara fotográfica

1. Point the camera at the scene (viewing transformation).
2. Compose the scene (modeling transformation).
3. Choose the type of lens and adjust the zoom (projection transformation).
4. Determine the physical size of the scene (viewport transformation).



Tipos de transformações

- View e Model
  - **`glMatrixMode(GL_MODELVIEW)`**
  - Position the camara
  - Rotations, translations, sclae
- Projection
  - **`glMatrixMode(GL_PROJECTION)`**
  - Projection volume definition
- 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); // usually in reshape func
    glLoadIdentity();
    // camera position
    camara();
    // model transformations
    ...
}
```

Evitar transformações desnecessárias

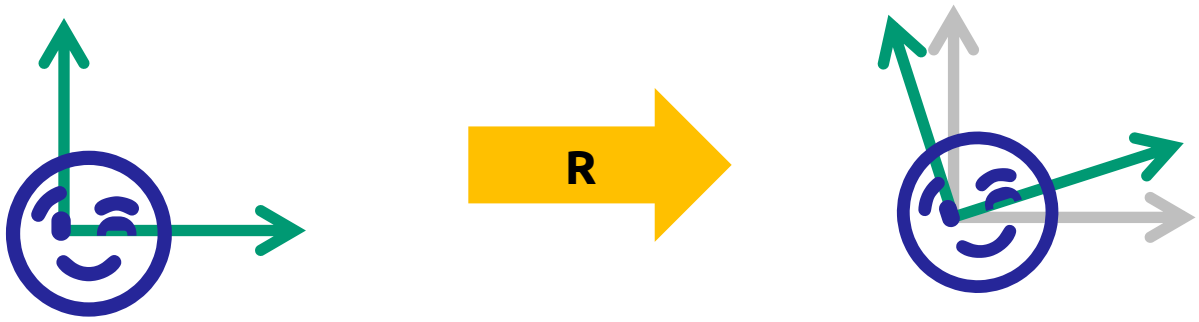
- Accumulated calculations in matrices should be avoided
- **Example:** do not apply consecutive rotations in the same matrix, but have a variable with the rotation angle

Inicializar as matrizes

- Before defining transformations, the transformation matrix must be initialized using **`glLoadIdentity()`**

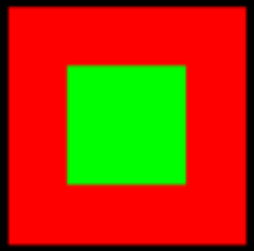
Rotações

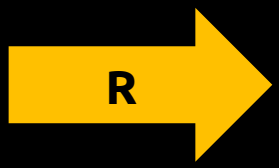
- `void glRotate{f|d}(angle, ex, ey, ez)`

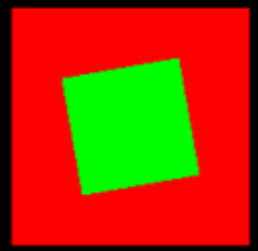


Rotações

```
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);
```

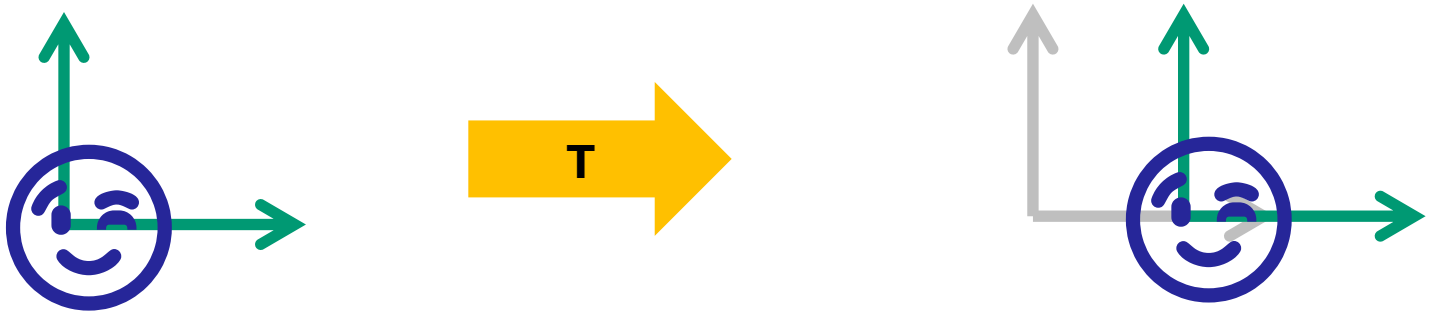






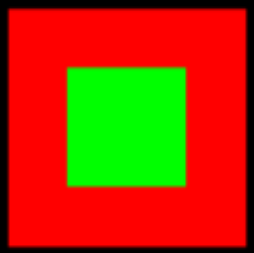
Translações

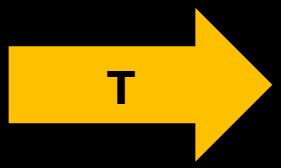
- `void glTranslate{f|d}(dx, dy, dz);`

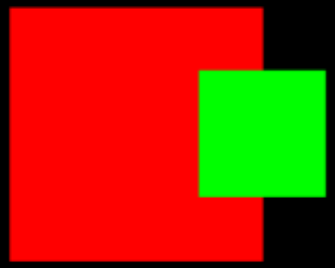


Translações

```
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);
```

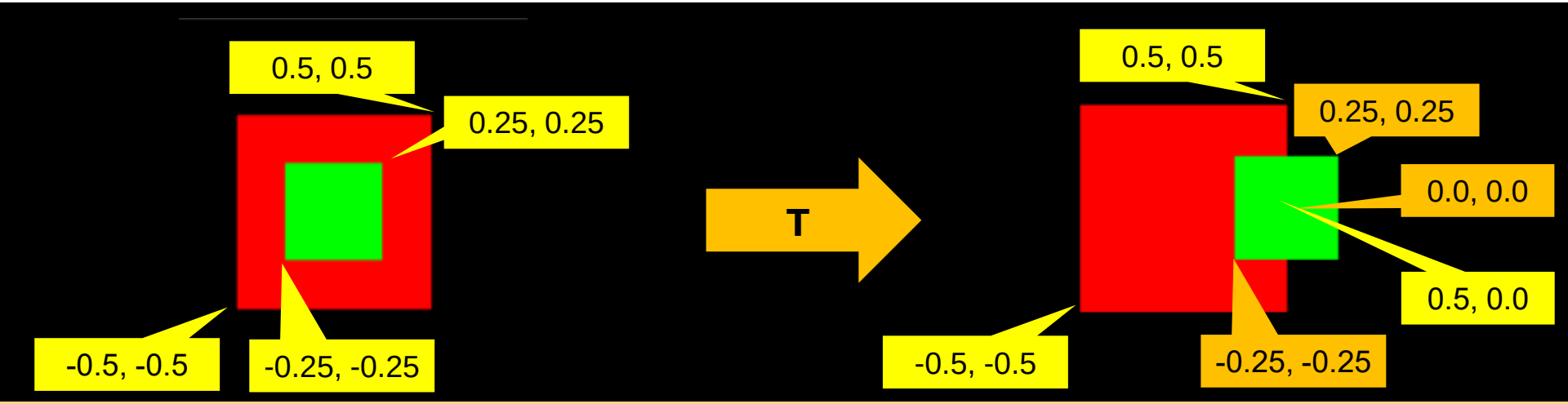






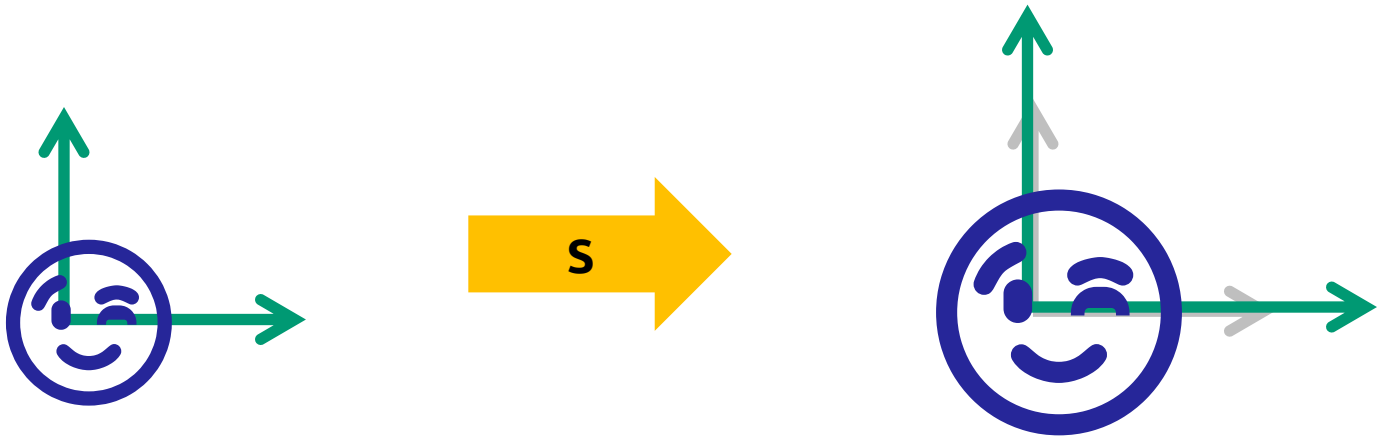
Translações

```
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);
```



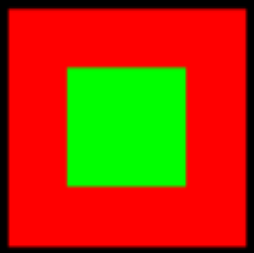
Escalamento

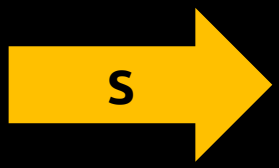
- `void glScale{f|d} sx, sy, sz)`
- avoid non-proportional scales (sx! = sy! = sz) as they alter normal vectors, rotations, etc ...

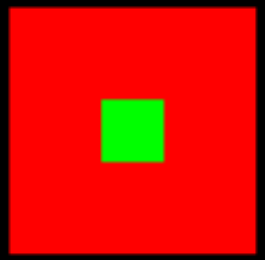


Escalamento

```
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);
```

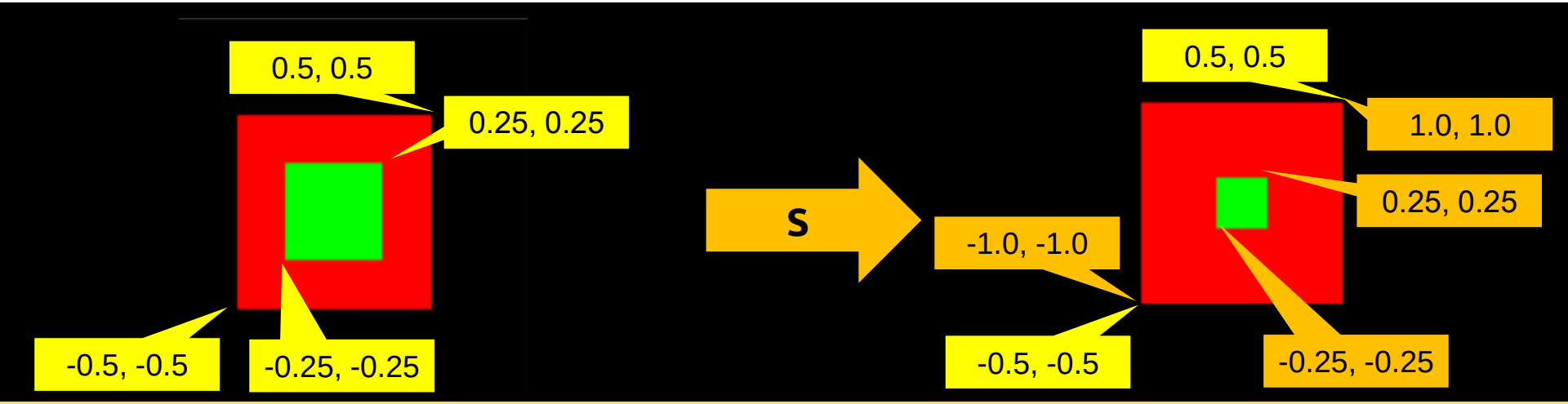




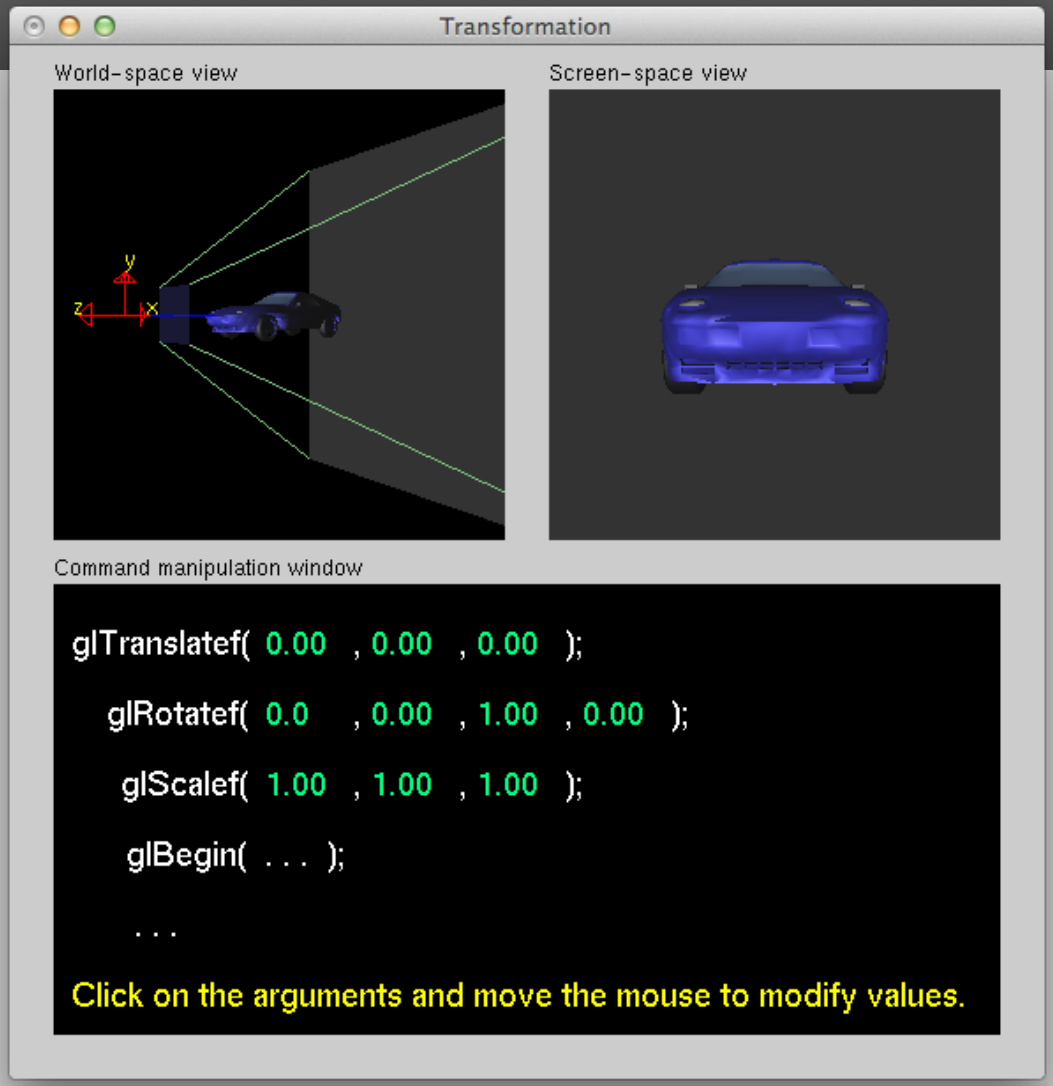


Escalamento

```
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.