

# Módulo 12

## Sistemas Gráficos e Interação

Instituto Superior de Engenharia do Porto

Filipe Pacheco

ffp@isep.ipp.pt

# Seleção & feedback

# Conteúdo

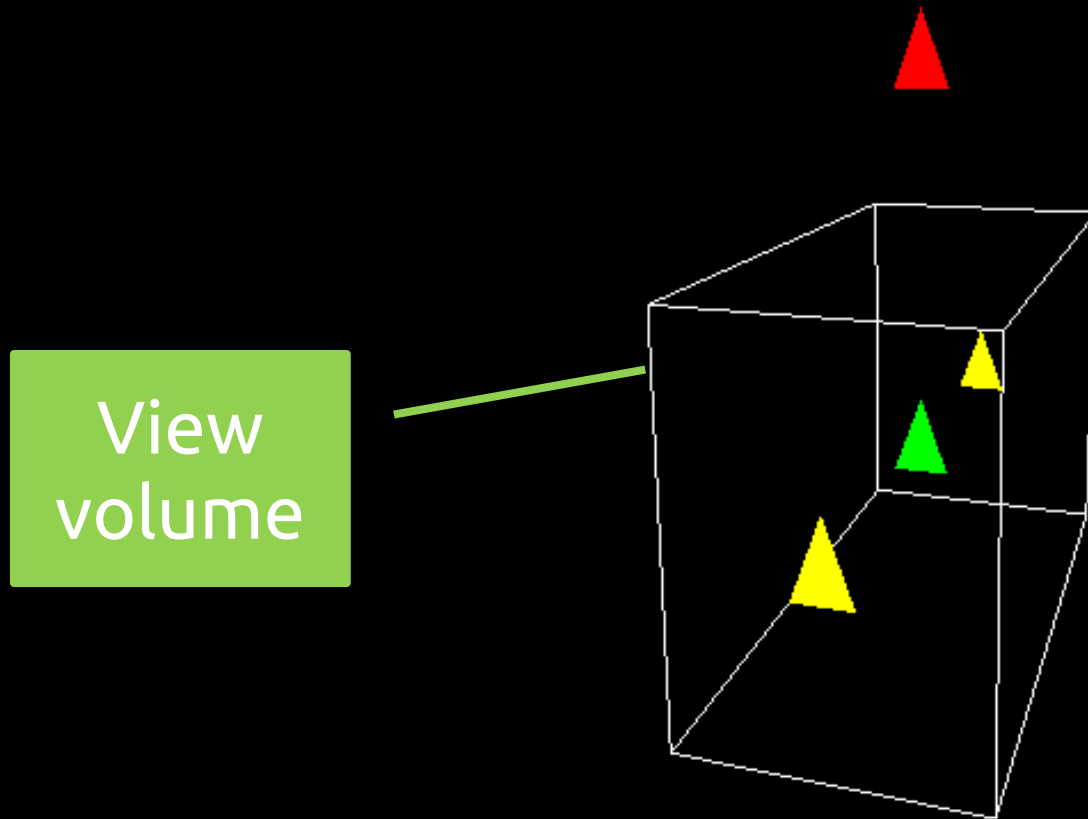


- ⦿ Modos do OpenGL
- ⦿ Seleção
- ⦿ Picking
- ⦿ Feedback

# Modos do OpenGL

- ⊙ `glRenderMode(mode)`
  - ⊙ `GL_RENDER`
    - ⊙ Normal operating mode: drawing primitives on the screen
  - ⊙ `GL_SELECTION`
    - ⊙ Selection mode: does not draw on the screen but returns information (symbolic name) about the objects that would be drawn
    - ⊙ Picking mode: identical but restricting information area, using the position of an input device (usually the mouse)
  - ⊙ `GL_FEEDBACK`
    - ⊙ Feedback mode: does not draw on the screen but returns information about the graphic elements that would be drawn on the screen (vertices, colors, ...)

# Exemplo seleção



# Exemplo seleção

```
void display(void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // draw normal scene
    drawScene();

    // "draw" in selection mode
    selectObjects();

    glFlush();
}
```

# Exemplo seleção

```
void selectObjects(void) {
    GLuint selectBuf[BUFSIZE];
    glSelectBuffer(BUFSIZE, selectBuf);
    glRenderMode(GL_SELECT);
    glInitNames();
    glPushName(0); //place initial "name" in stack - 0
    // projection, view, draw
    glPushMatrix();
    glMatrixMode (GL_PROJECTION);
    glLoadIdentity ();
    glOrtho(0.0, 5.0, 0.0, 5.0, 0.0, 10.0);
    glMatrixMode (GL_MODELVIEW);
    glLoadIdentity ();
    drawTriangles();
    glPopMatrix();
    glFlush();
    GLint hits = glRenderMode(GL_RENDER);
    processHits (hits, selectBuf);
}
```

# Exemplo seleção

```
void drawTriangles(void)
{
    /* green triangle */
    glColor3f (0.0, 1.0, 0.0);
    glLoadName(1);
    drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -5.0);

    /* red triangle */
    glColor3f (1.0, 0.0, 0.0);
    glLoadName(2);
    drawTriangle (2.0, 7.0, 3.0, 7.0, 2.5, 8.0, -5.0);

    /* yellow triangles */
    glColor3f (1.0, 1.0, 0.0);
    glLoadName(3);
    drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, 0.0);
    drawTriangle (2.0, 2.0, 3.0, 2.0, 2.5, 3.0, -10.0);
}
```

# Instruções

- ⊙ `glSelectBuffer (size, buffer)`
  - ⊙ set **before** entering selection mode
- ⊙ `glInitNames ()`
  - ⊙ call before drawing the scene
- ⊙ `glPushName (name) / glPopName ()`
  - ⊙ Insert / remove a symbolic name from the stack
- ⊙ `glLoadName (name)`
  - ⊙ set the symbolic name at the top of the stack



# Seleção

## ◎ Steps:

1. Initialize return buffer
2. Enter selection mode
3. Initialize symbolic name stack
4. Set viewing volume
5. “Draw” the scene and define the symbolic name of the objects
6. Exit selection mode and process the return buffer record

# Processor resultados

- ⊙ When exiting the selection mode (invoking `glRenderMode`), information about selection is received
  - ⊙ Variable length array with variable length records
  - ⊙ *Hit record = #names, z1, z2, (names)\**

#names	z1	z2	names
1	0.3	0.43	27
2	0.55	0.62	27 35
1	0.23	0.23	29

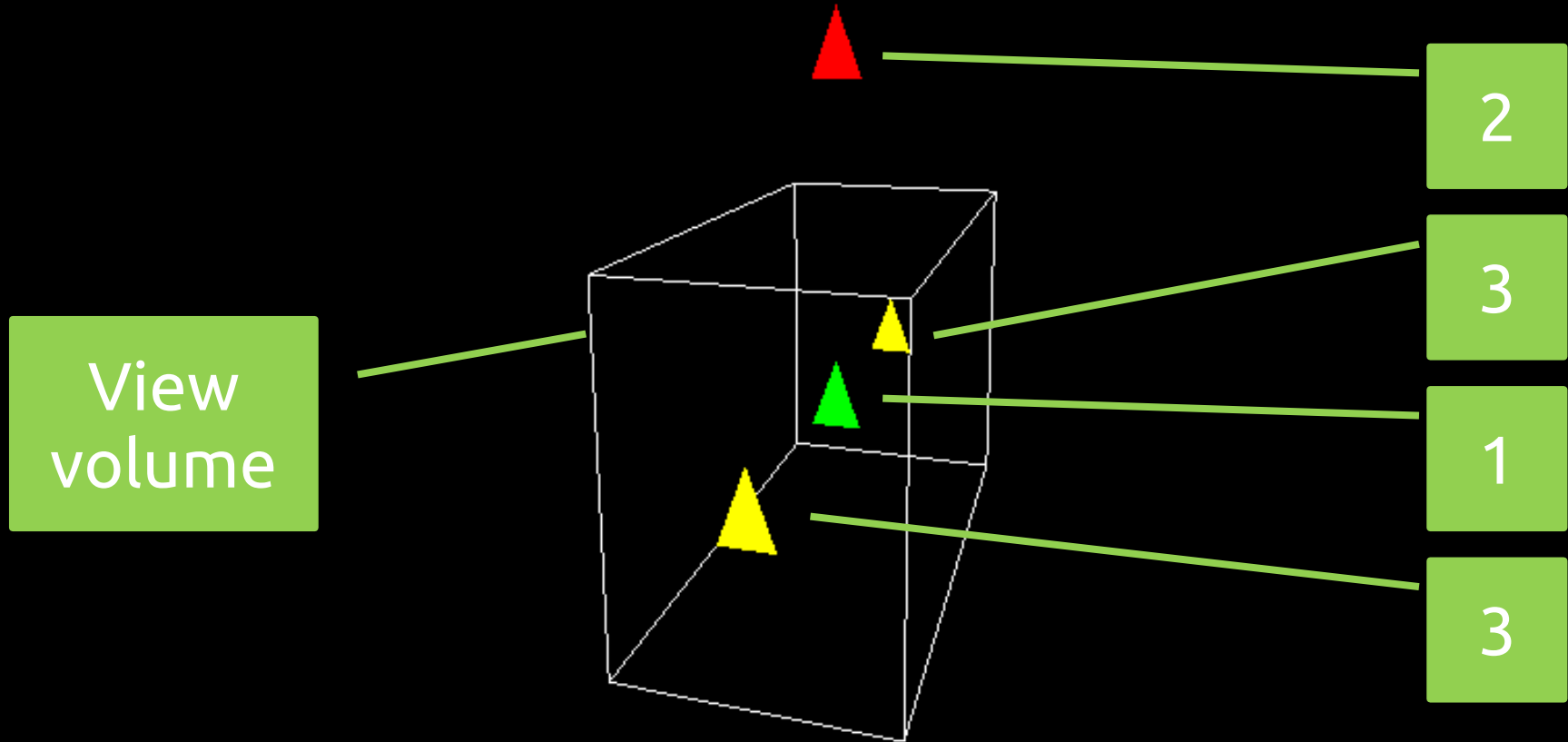
#n	z1	z2	name	#n	z1	z2	name1	name2	#n	z1	z2	name
1	3	43	27	2	55	62	27	35	1	23	23	29

# Exemplo seleção

```
void processHits(GLint hits, GLuint buffer[])
{
    int i, j, names;
    GLuint *ptr;

    printf ("hits = %d\n", hits);
    ptr = (GLuint *) buffer;
    for (i = 0; i < hits; i++) { /* for each hit */
        names = (int) *ptr;
        printf (" number of names for hit = %d\n", names); ptr++;
        printf(" z1 is %g;", (float) *ptr/UINT_MAX); ptr++;
        printf(" z2 is %g\n", (float) *ptr/UINT_MAX); ptr++;
        printf (" the name is ");
        for (j = 0; j < names; j++) { /* for each name */
            printf ("%d ", *ptr); ptr++;
        }
        printf ("\n");
    }
}
```

# Demo - select



# Picking

## ◎ Passos a seguir:

1. Initialize return buffer
2. Enter selection mode
3. Initialize symbolic name stack
4. **Define sensitivity matrix based on the position of the input device**
5. Set viewing volume
6. “Draw” the scene and define the symbolic name of the objects
7. Exit selection mode and process the return buffer records

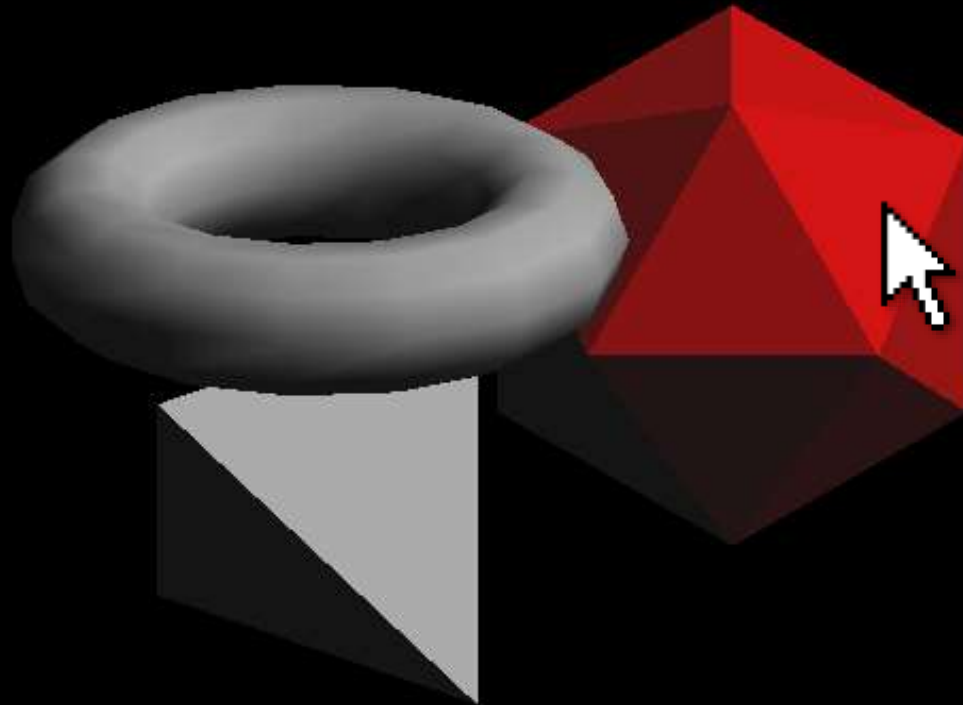
# Exemplo picking

- ⊙ Enter selection mode via mouse event
  - ⊙ `glutPassiveMotionFunc`
  - ⊙ `glutMouseFunc`
- ⊙ Define the sensitivity matrix - `GL_PROJECTION`

```
glMatrixMode(GL_PROJECTION);  
glPushMatrix();  
glLoadIdentity();  
GLint viewport[4];  
glGetIntegerv(GL_VIEWPORT, viewport);  
// 5x5 pixel picking region around the mouse pointer  
gluPickMatrix((GLdouble)x, (GLdouble)(viewport[3]-y),  
             5.0, 5.0, viewport);  
// define projection  
...
```

# Demo - highlight

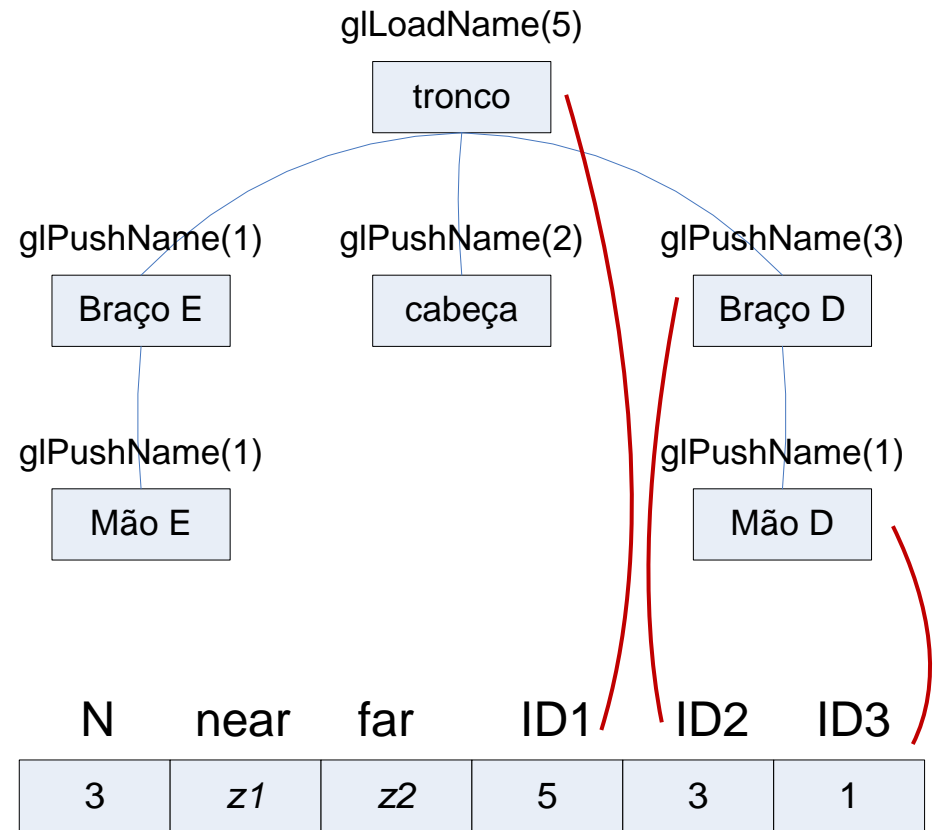
Automatically names object under mouse.



Located: icosahedron.

# Múltiplos níveis de nomes

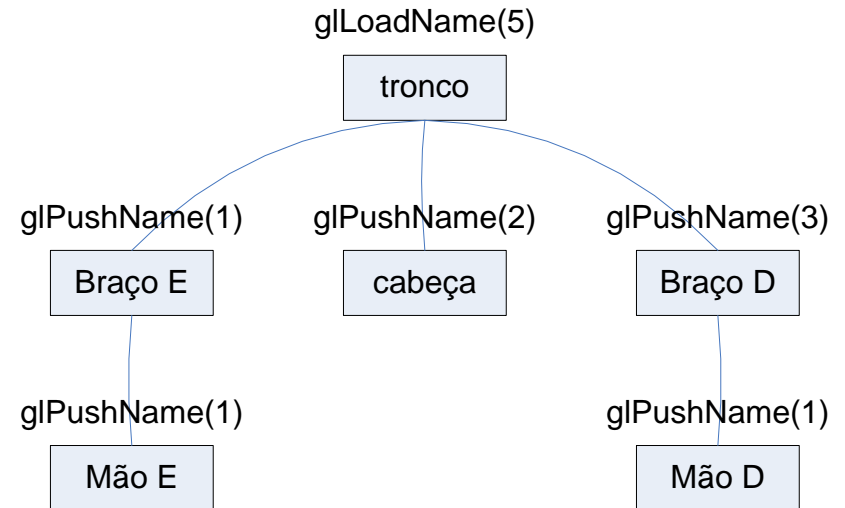
- To allow working with hierarchical objects it is possible to define a stack of names and not just a name for the object
- If the user selected the object “hand D”, the hit record would be:





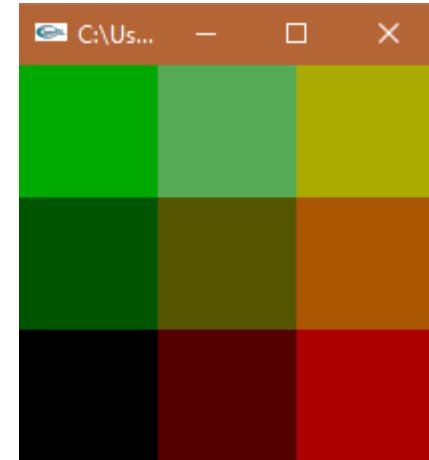
# Exemplo

```
void drawRobot()  
{  
    glLoadName(5);  
    drawTronco();  
    // braço D  
    glPushMatrix();  
    glPushName(3);  
    drawBraco();  
    glPushName(1);  
    drawMao();  
    glPopName();  
    glPopMatrix();  
    // cabeça  
    glPushMatrix();  
    glPushName(2);  
    drawCabeca();  
    glPopName();  
    glPopMatrix();  
    // braço E  
    ...  
}
```



# Exemplo

```
void drawSquares(GLenum mode)
{
    GLuint i, j;
    for (i = 0; i < 3; i++)
    {
        if (mode == GL_SELECT)
            glLoadName(i);
        for(j = 0; j < 3; j ++ )
        {
            if (mode == GL_SELECT)
                glPushName(j);
            glColor3f(i/3.0, j/3.0, board[i][j]/3.0);
            glRecti(i, j, i+1, j+1);
            if (mode == GL_SELECT)
                glPopName();
        }
    }
}
```



# Demo - picksquare

```
hits = 1  
  number of names for this hit = 2  
    z1 is 0.9999999; z2 is 0.9999999  
      names are 1 2
```



```
hits = 1  
  number of names for this hit = 2  
    z1 is 0.9999999; z2 is 0.9999999  
      names are 0 1
```



# feedback

- ◎ Why?
  - ◎ Know what would be drawn
  - ◎ Vector information
- ◎ What is it for?
  - ◎ Generate printer commands
  - ◎ Save to file in vector format, eg DXF, windows metafile

# feedback

## ◎ Steps:

1. Set return buffer
2. Enter feedback mode
3. “Draw” objects
4. Exit feedback mode
5. Process return information

# Exemplo

```
void display(void)
{
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 100.0, 0.0, 100.0, 0.0, 1.0);

    glClearColor(0.0, 0.0, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    drawGeometry(GL_RENDER);

    GLfloat feedBuffer[1024];
    glFeedbackBuffer (1024, GL_3D_COLOR, feedBuffer);
    glRenderMode (GL_FEEDBACK);
    drawGeometry (GL_FEEDBACK);

    GLint size = glRenderMode (GL_RENDER);
    printBuffer (size, feedBuffer);
}
```

# Exemplo

```
void drawGeometry(GLenum mode)
{
    glBegin(GL_LINE_STRIP);
        glNormal3f(0.0, 0.0, 1.0);
        glVertex3f(30.0, 30.0, 0.0);
        glVertex3f(50.0, 60.0, 0.0);
        glVertex3f(70.0, 40.0, 0.0);
    glEnd();
    if (mode == GL_FEEDBACK)
        glPassThrough(1.0);
    glBegin(GL_POINTS);
        /* will be clipped */
        glVertex3f(-100.0, -100.0, -100.0);
    glEnd();
    if (mode == GL_FEEDBACK)
        glPassThrough(2.0);
    glBegin(GL_POINTS);
        glNormal3f(0.0, 0.0, 1.0);
        glVertex3f(50.0, 50.0, 0.0);
    glEnd();
}
```

# Informação de retorno

## ◎ glFeedbackBuffer(*size, type, buffer*)

Type	Coord.	Cor	Texturas	Nr. Valores
GL_2D	x, y			2
GL_3D	x, y, z			3
GL_3D_COLOR	x, y, z	k		3 + k
GL_3D_COLOR_TEXTURE	x, y, z	k	4	7 + k
GL_4D_COLOR_TEXTURE	x, y, z, w	k	4	8 + k

k=1 (Color Index)

k=4 (RGBA)



# Informação de retorno

The return buffer is filled with variable size information

Primitive code, data

Vertices are in window coordinates

Primitive	Type Code	Associated Data
Point	GL_POINT_TOKEN	vertex
Line	GL_LINE_TOKEN or GL_LINE_RESET_TOKEN	vertex vertex
Polygon	GL_POLYGON_TOKEN	n vertex vertex ... vertex
Bitmap	GL_BITMAP_TOKEN	vertex
Pixel Rectangle	GL_DRAW_PIXEL_TOKEN or GL_COPY_PIXEL_TOKEN	vertex
Pass-through	GL_PASS_THROUGH_TOKEN	a floating-point number

# Demo

GL\_LINE\_RESET\_TOKEN

36.00 30.00 0.00 0.84 0.84 0.84 1.00

60.00 60.00 0.00 0.84 0.84 0.84 1.00

GL\_LINE\_TOKEN

60.00 60.00 0.00 0.84 0.84 0.84 1.00

84.00 40.00 0.00 0.84 0.84 0.84 1.00

GL\_PASS\_THROUGH\_TOKEN

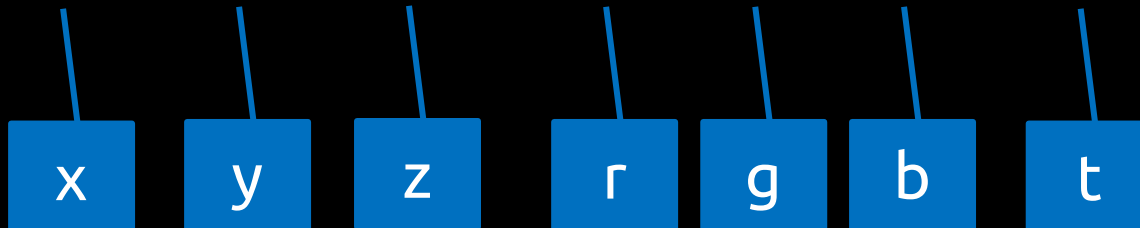
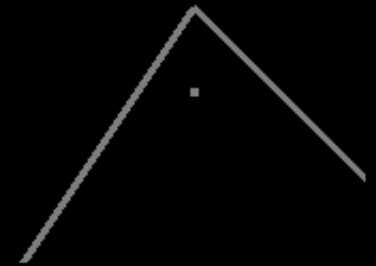
1.00

GL\_PASS\_THROUGH\_TOKEN

2.00

GL\_POINT\_TOKEN

60.00 50.00 0.00 0.84 0.84 0.84 1.00



# Exportar para PS e WMF

- ⦿ **Examples at:**

- ⦿ <http://www.codeproject.com/opengl/glexport.asp>

- ⦿ <http://www.geuz.org/gl2ps/>