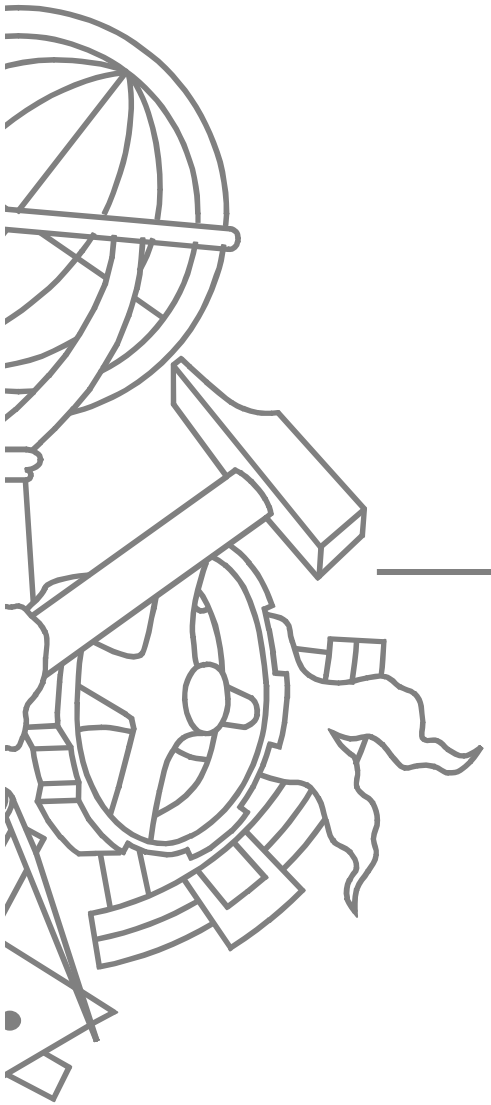


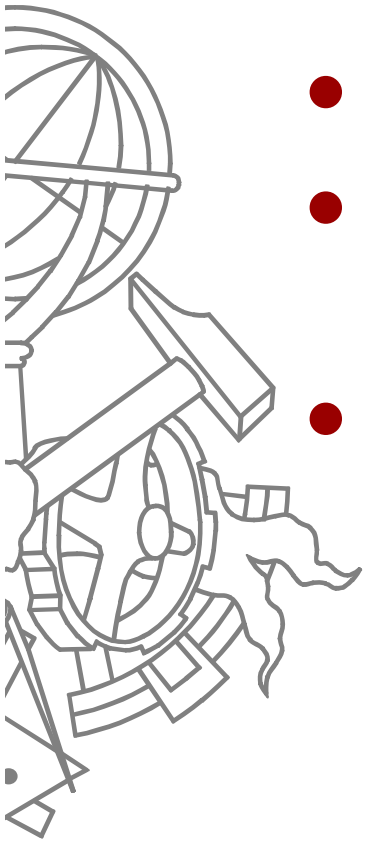
Pontos, Linhas e Polígonos

Aula 2

Sistemas Gráficos e Interactivos
Instituto Superior de Engenharia do Porto

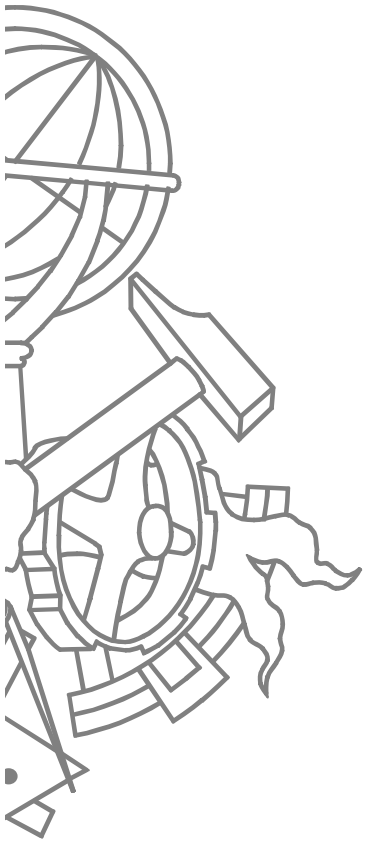
Paulo Gandra de Sousa
psousa@dei.isep.ipp.pt

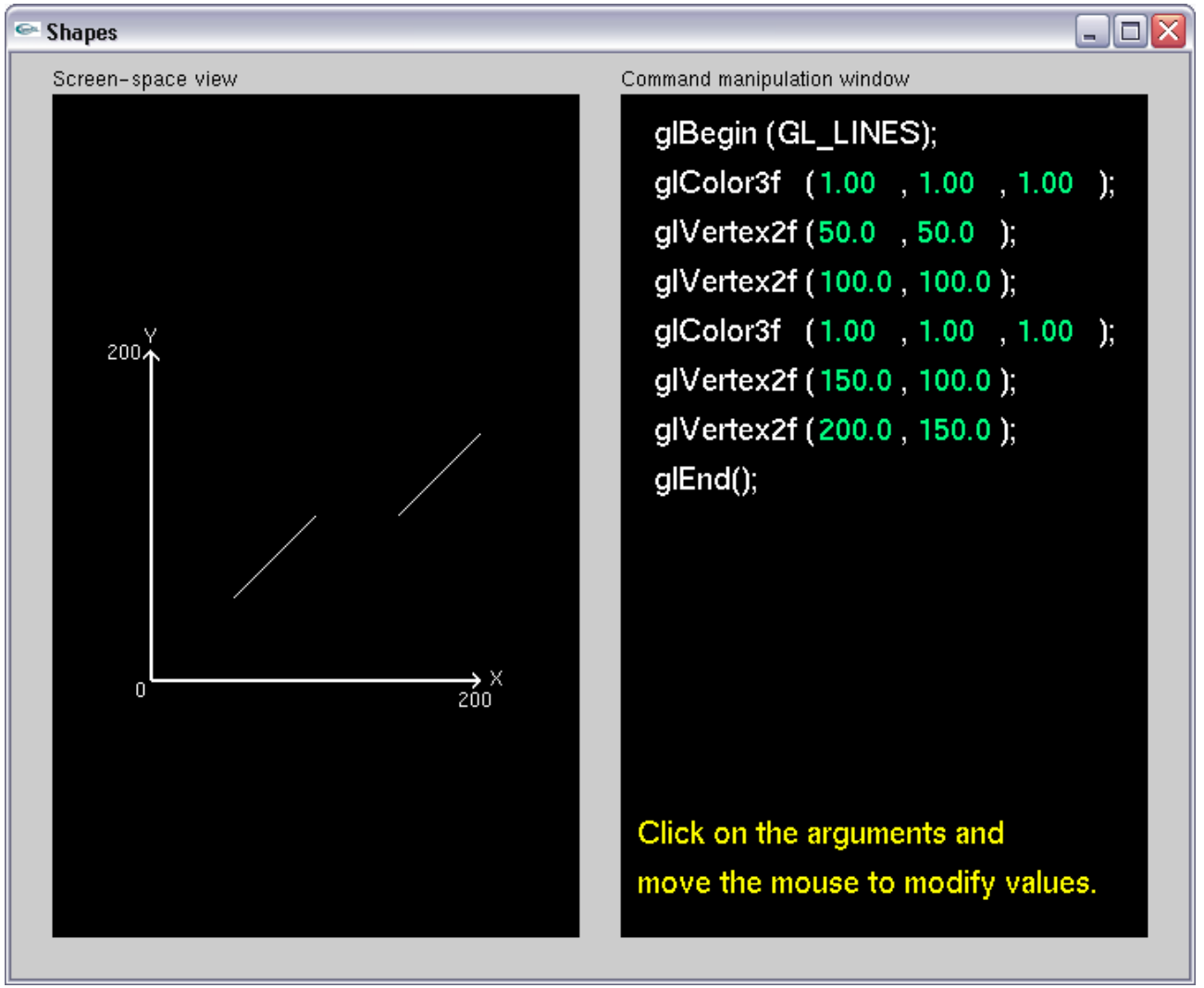
Conteúdo



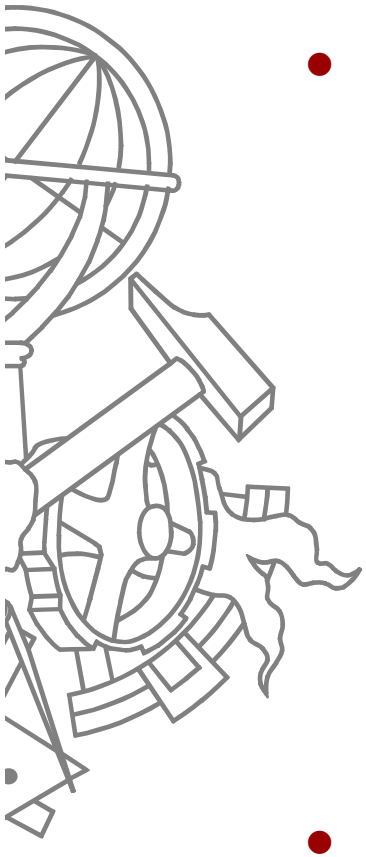
- Instruções de desenho
- Especificidade de linhas, pontos e polígonos
- GLUT
  - Menus
  - Tratamento de eventos

Demo



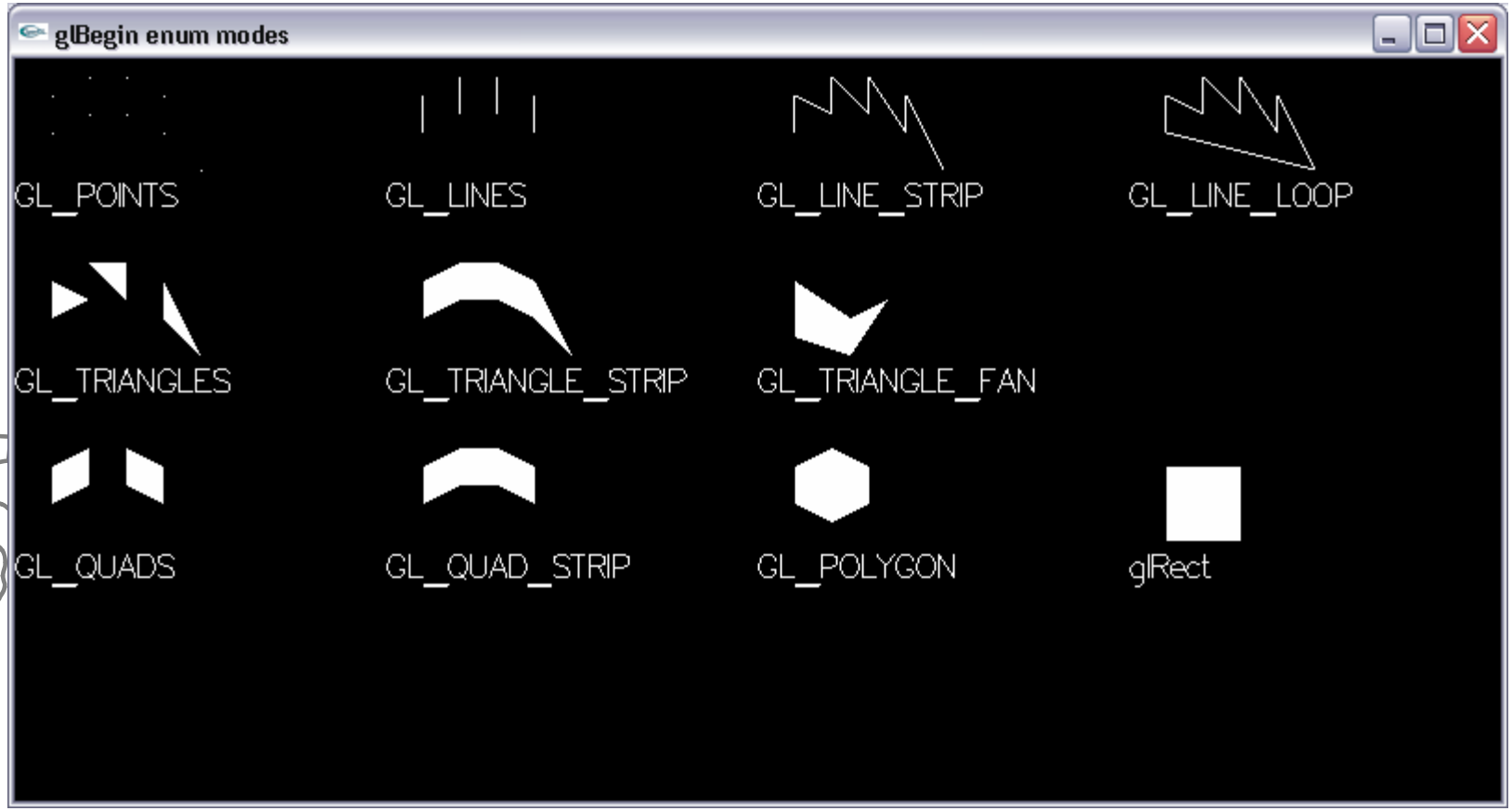

Shapes
Screen-space view
Command manipulation window
glBegin (GL_LINES);
glColor3f (1.00 , 1.00 , 1.00);
glVertex2f (50.0 , 50.0);
glVertex2f (100.0 , 100.0);
glColor3f (1.00 , 1.00 , 1.00);
glVertex2f (150.0 , 100.0);
glVertex2f (200.0 , 150.0);
glEnd();
Y
200
0
X
200
Click on the arguments and
move the mouse to modify values.

Desenho de objectos simples

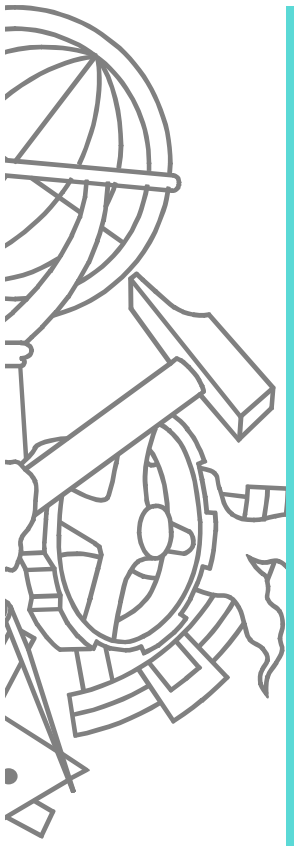


- **`glBegin(`*`mode`*`) / glEnd()`**
  - GL_POINTS
  - GL_LINES
  - GL_LINE_STRIP
  - GL_LINE_LOOP
  - GL_TRIANGLES
  - GL_TRIANGLE_STRIP
  - GL_TRIANGLE_FAN
  - GL_POLYGON
  - GL_QUADS
  - GL_QUAD_STRIP
- **`glRect`**

Demo


glBegin enum modes
GL_POINTS
GL_LINES
GL_LINE_STRIP
GL_LINE_LOOP
GL_TRIANGLES
GL_TRIANGLE_STRIP
GL_TRIANGLE_FAN
GL_QUADS
GL_QUAD_STRIP
GL_POLYGON
glRect

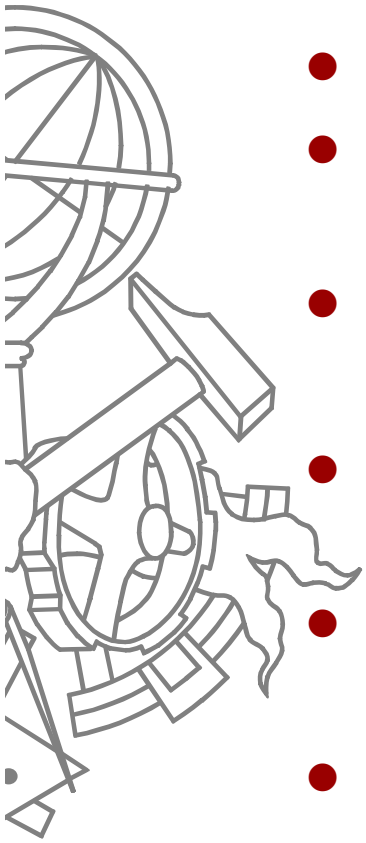
Vértices usados na demo



```
// Points, Lines

glBegin(mode);
   glVertex2i(10, 20);
   glVertex2i(10, 10);
   glVertex2i(20, 15);
   glVertex2i(20, 5);
   glVertex2i(30, 15);
   glVertex2i(30, 5);
   glVertex2i(40, 20);
   glVertex2i(40, 10);
   glVertex2i(50, 30);
glEnd();
```

```
// Quads

glBegin(GL_QUADS);
   glVertex2i(10, 20);
   glVertex2i(10, 10);
   glVertex2i(20, 5);
   glVertex2i(20, 15);
   glVertex2i(30, 5);
   glVertex2i(30, 15);
   glVertex2i(40, 20);
   glVertex2i(40, 10);
glEnd();

// vértices indicados em
// loop seguindo um sentido
// CW/CCW
```

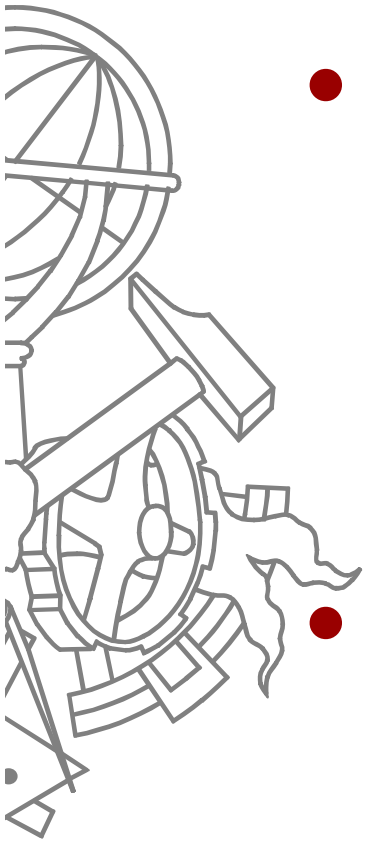
Instruções possíveis em glBegin/glEnd



- **`glColor`**
- **`glIndex`**
  - Cor em modo indexado
- **`glVertex`**

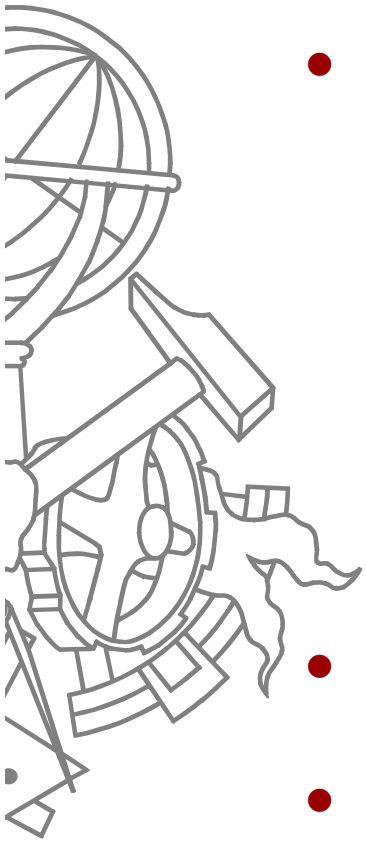
- **`glNormal`**
  - Perpendicular à superfície (utilizado para iluminação)
- **`glMaterial`**
  - Tipo de material do objecto
- **`glCallList, glCallLists`**
  - Objectos pré-construidos (aula 3)

Vértices



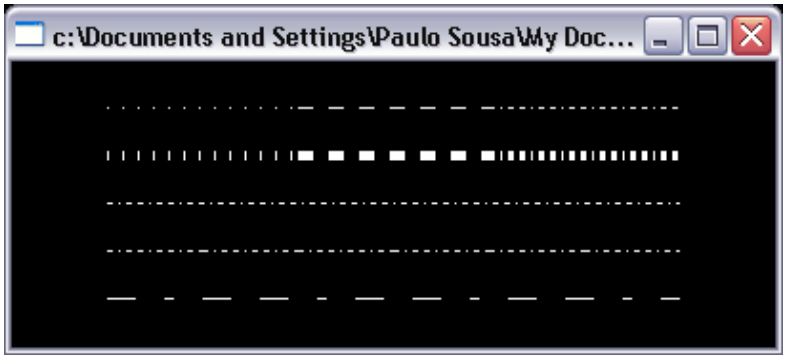
- `glVertex{2|3|4}`
  - 2D (xy)
  - 3D (xyz)
  - Coordenadas homogéneas (xyzw)
- Indicar os vértices de cada face no mesmo sentido (CW ou CCW)

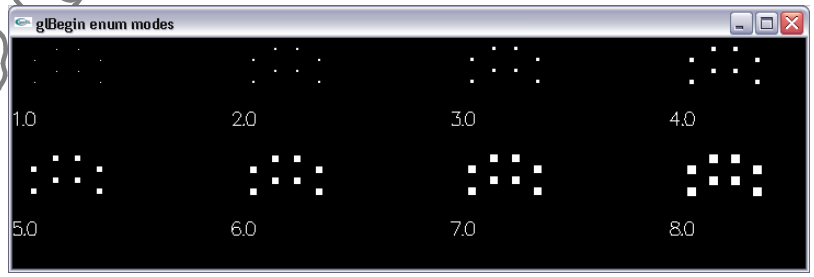
Modo de polígonos



- **`glPolygonMode(`*`face, mode`*`)`**
  - *Face*
    - GL_FRONT
    - GL_BACK
    - GL_FRONT_AND_BACK
  - *Mode*
    - GL_POINT
    - GL_LINE
    - GL_FILL
- Por omissão as faces cujos vértices são indicados CCW são faces “viradas para a frente”
- Por omissão as faces são preenchidas (GL_FILL)

Linhas e pontos



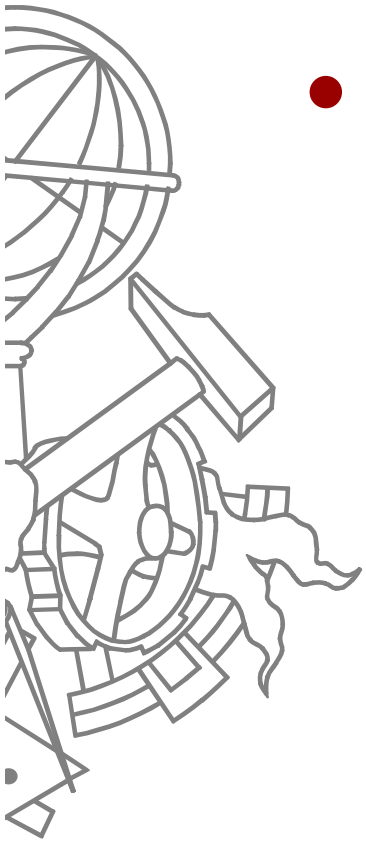


- `glLineWidth`
- `glLineStipple`
  - `0x0101 // dotted`
  - `0x00FF // dashed`
  - `0x1C47 // dash-dot-dash`
- `glEnable(GL_LINE_STIPPLE)`

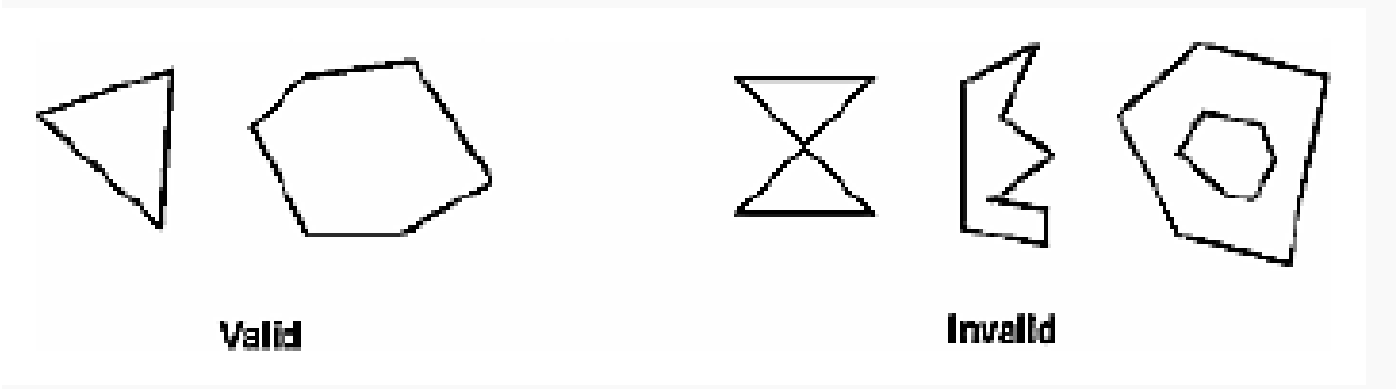
- `glPointSize`

- Fora de glBegin/glEnd

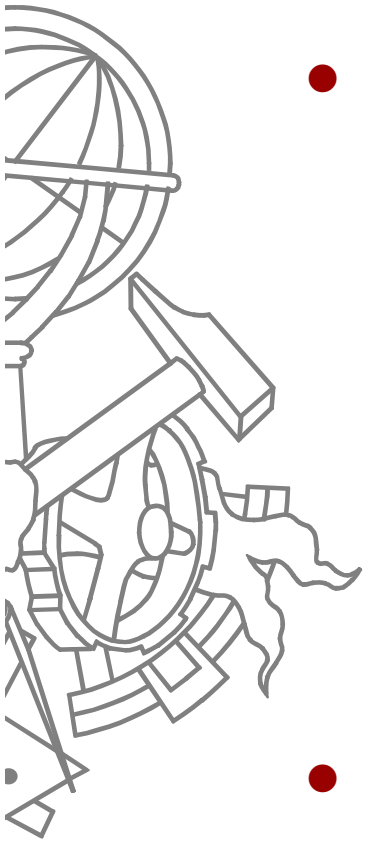
Polígonos convexos



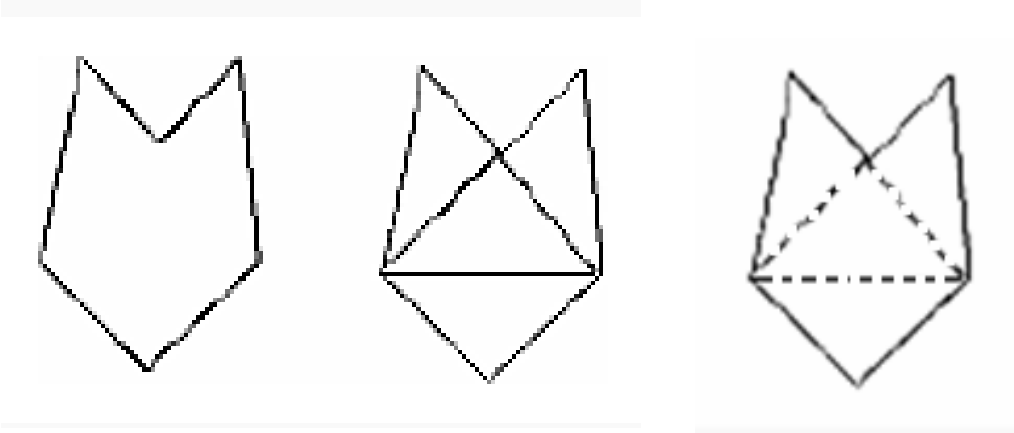
- Qualquer linha que “atravesse” um polígono só tem um segmento dentro do polígono.



Polígonos não convexos

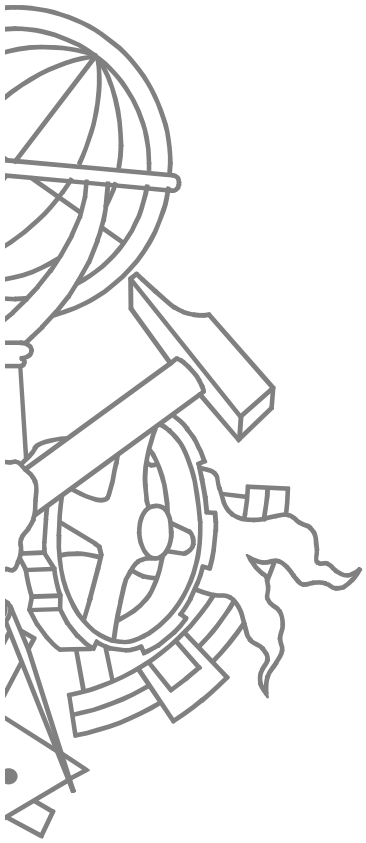


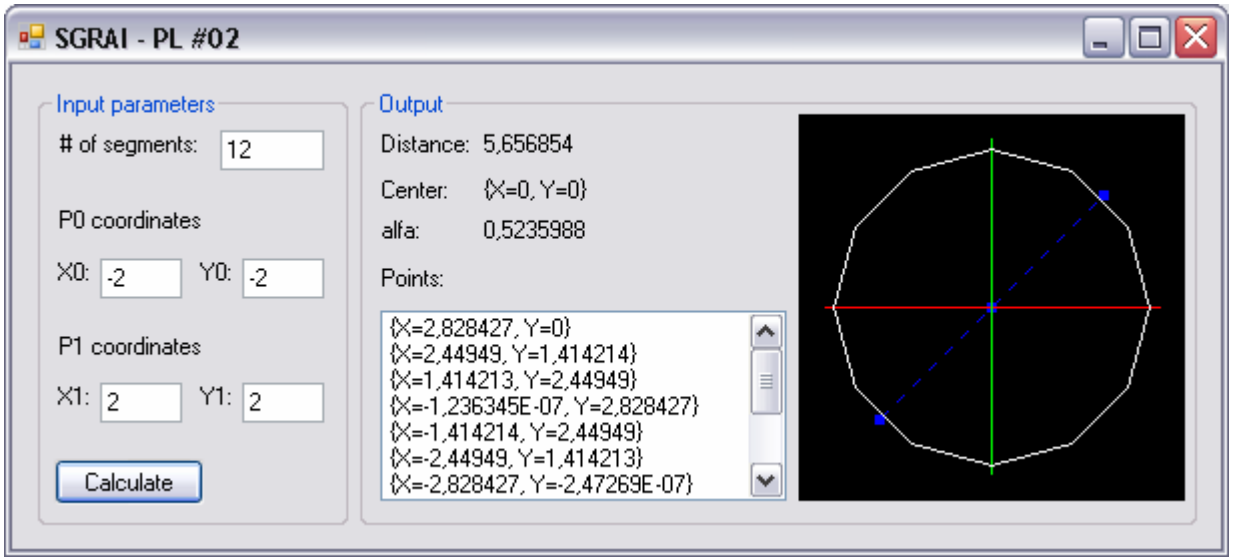
- Dividir em polígonos convexos (ex., triângulos) e usar **`glEdgeFlag`** para indicar os vértices que pertencem a arestas de bordo



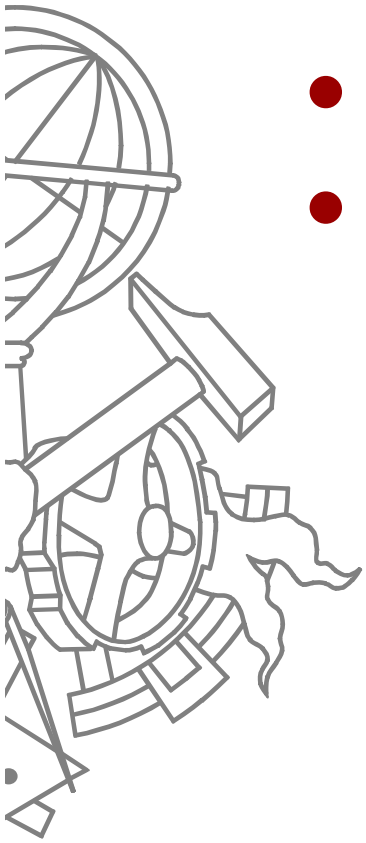
- Este processo tem o nome de “tesselation” e o GLU tem API própria para o fazer de forma eficiente

Círculos



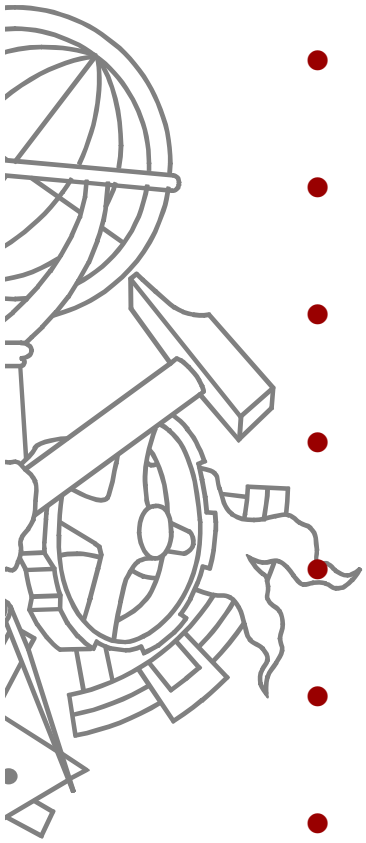

SGRAI - PL #02
Input parameters
# of segments: 12
P0 coordinates
X0: -2
Y0: -2
P1 coordinates
X1: 2
Y1: 2
Calculate
Output
Distance: 5,656854
Center: {X=0, Y=0}
alfa: 0,5235988
Points:
{X=2,828427, Y=0}
{X=2,44949, Y=1,414214}
{X=1,414213, Y=2,44949}
{X=-1,236345E-07, Y=2,828427}
{X=-1,414214, Y=2,44949}
{X=-2,44949, Y=1,414213}
{X=-2,828427, Y=-2,47269E-07}

Um pouco mais de GLUT



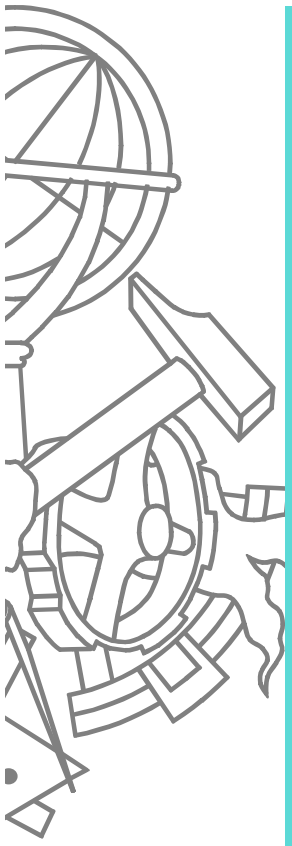
- *Callbacks*
- Menus

callbacks GLUT



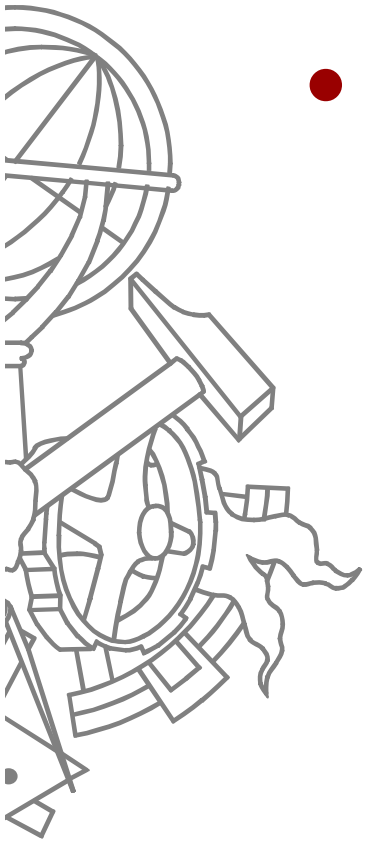
- glutDisplayFunc(*display*)
  - void *display*()
- glutReshapeFunc(*reshape*)
  - void *reshape*(int width, int height)
- glutKeyboardFunc(*keyboard*)
  - void *keyboard*(unsigned char key, int x, int y)
- glutMouseFunc(*mouse*)
  - void *mouse*(int button, int state, int x, int y)
- glutMotionFunc(*motion*)
  - void *motion*(int x, int y)
- glutIdleFunc(*idle*)
  - void *idle*(void)
- glutTimerFunc(unsigned int millis, *onTimer*, int value)
  - void *onTimer*(int value)
- glutSpecialFunc(*special*)
  - void *special*(int key, int x, int y)

Esqueleto *callbacks*



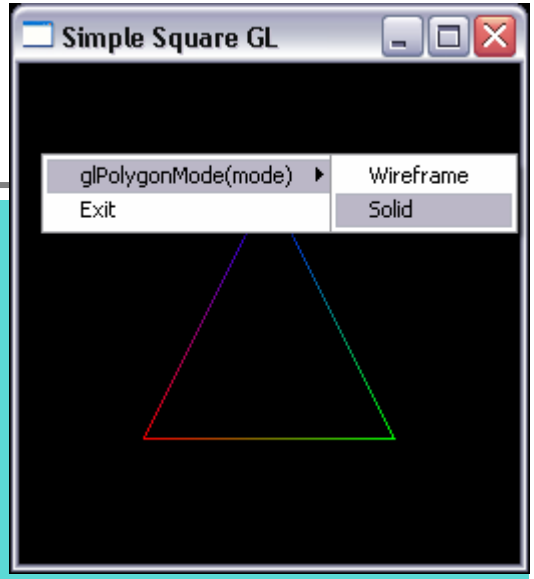
```
void reshape(int w, int h) {
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y) {
   switch (key) {
   case 27: // ESCAPE
        exit(0);
        break;
   case 'w':
   case 'W':
        g_conf.polygonWire = !g_conf.polygonWire;
        break;
   }
   glutPostRedisplay();
}
```

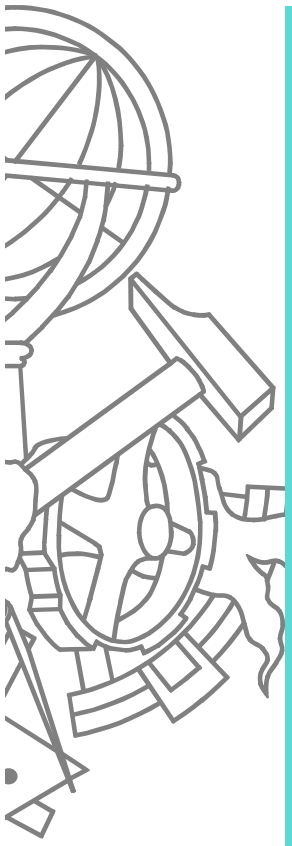
Forçar redesenho



- `glutPostRedisplay`

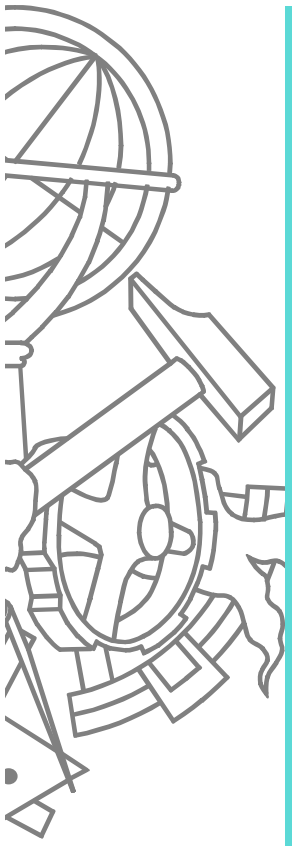
Menus em GLUT





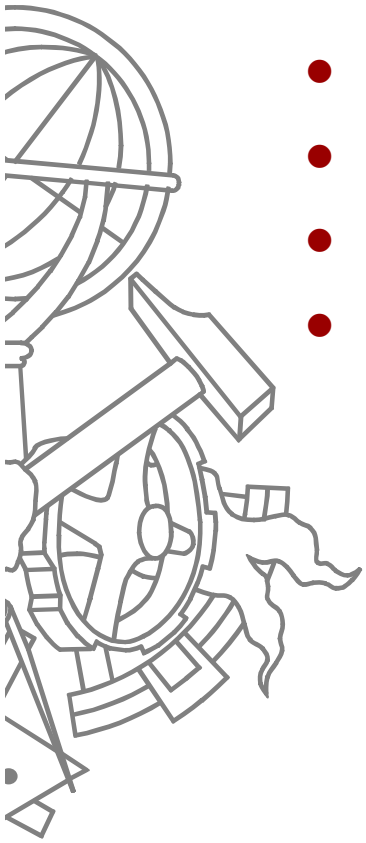
```
#define MENU_EXIT               1
#define MENU_POLYGON_WIRE       2
#define MENU_POLYGON_SOLID      3

void init_menus()
{
   int mnu, mnu1;

   mnu1 = glutCreateMenu(handle_menu);
   glutAddMenuEntry("Wireframe", MENU_POLYGON_WIRE);
   glutAddMenuEntry("Solid", MENU_POLYGON_SOLID);

   mnu = glutCreateMenu(handle_menu);
   glutAddSubMenu("glPolygonMode(mode)", mnu1);
   glutAddMenuEntry("Exit", MENU_EXIT);
   glutAttachMenu(GLUT_LEFT_BUTTON);
}
```

Menus em GLUT



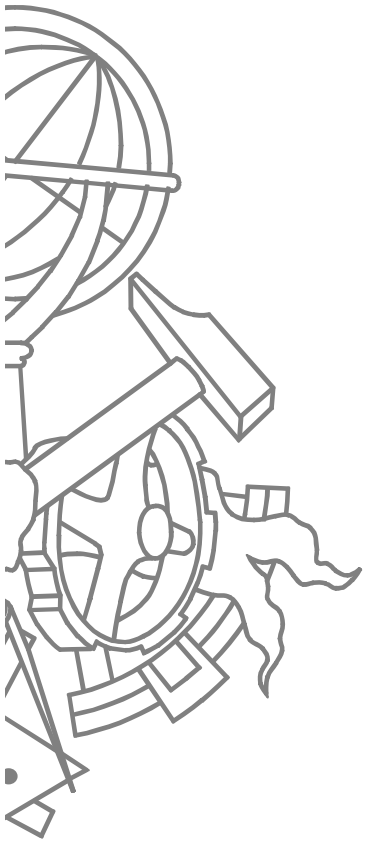
```
void handle_menu(int op)
{
   switch (op)
   {
   case MENU_EXIT:
        exit(0);
        break;
   case MENU_POLYGON_WIRE:
        g_conf.polygonWire = GL_TRUE;
        break;
   case MENU_POLYGON_SOLID:
        g_conf.polygonWire = GL_FALSE;
        break;
   }
   glutPostRedisplay();
}
```

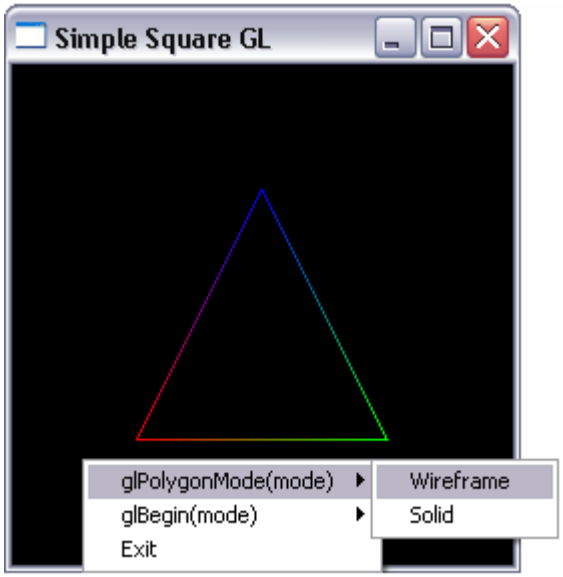
Menus em GLUT



- `int glutCreateMenu(void(*)(int op))`
- `glutAddMenuEntry(char* label, int value)`
- `glutAddSubMenu(int submenu)`
- `glutAttachMenu(int button)`

Demo




Simple Square GL
glPolygonMode(mode)
glBegin(mode)
Exit
Wireframe
Solid