Módulo 3

Sistemas Gráficos e Interação
Instituto Superior de Engenharia do Porto

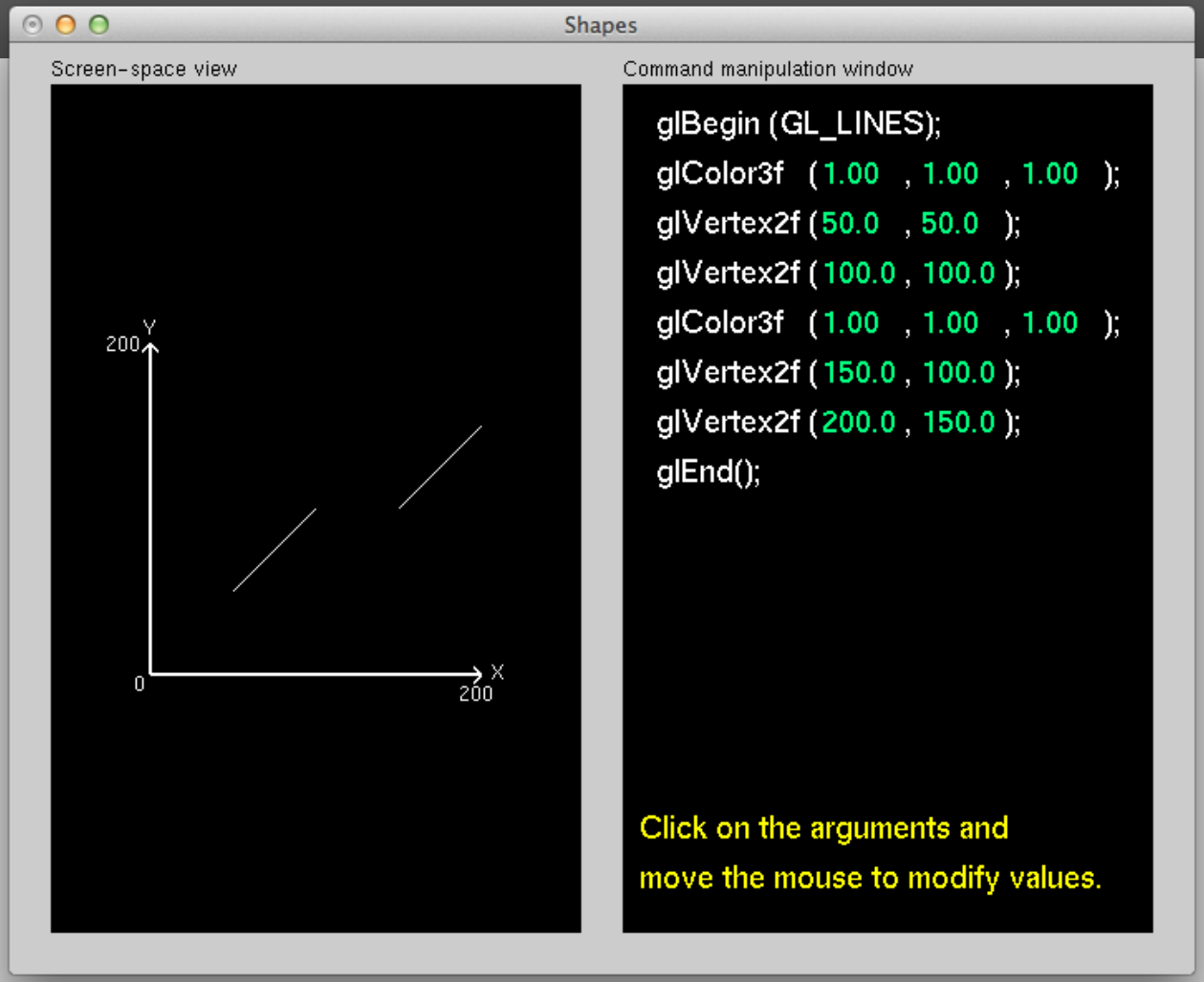
Filipe Pacheco
ffp@isep.ipp.pt

Pontos, Linhas e Polígonos

Conteúdo

- Instruções de desenho
- Especificidade de linhas, pontos e polígonos

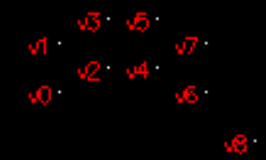
Demo


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.

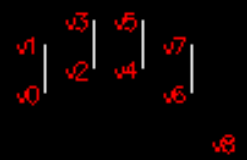
Desenho de objectos simples

- **`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
- **`glRectf(xi,yi,xf,yf)`**

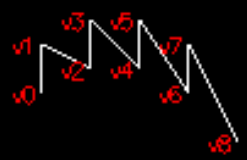
Demo



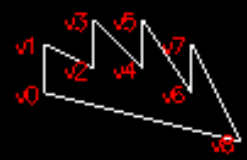
GL_POINTS



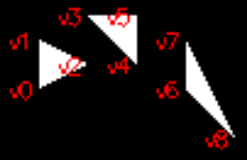
GL_LINES



GL_LINE_STRIP

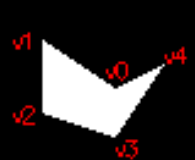


GL_LINE_LOOP

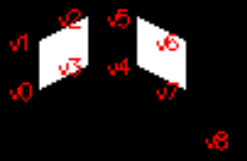


GL_TRIANGLES

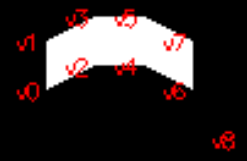
GL_TRIANGLE_STRIP



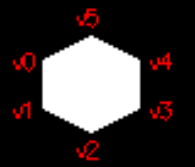
GL_TRIANGLE_FAN



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
```

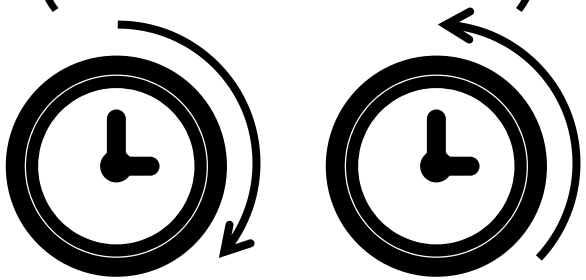
Instruções possíveis em glBegin()/glEnd()

- **`glColor`**
  - Cor em modo RGB
- **`glIndex`**
  - Cor em modo indexado
- **`glVertex`**
  - Coordenadas em 2D ou 3D
- **`glNormal`**
  - Perpendicular à superfície (utilizado para iluminação)
- **`glMaterial`**
  - Definições de material do objecto (iluminação)
- **`glCallList, glCallLists`**
  - Listas de vértices pre-compiladas

Vértices

- **`glVertex{2|3|4}`**
  - 2D (xy)
  - 3D (xyz)
  - Coordenadas homogéneas (xyzw)

- Indicar os vértices de cada face no mesmo sentido (CW ou CCW)

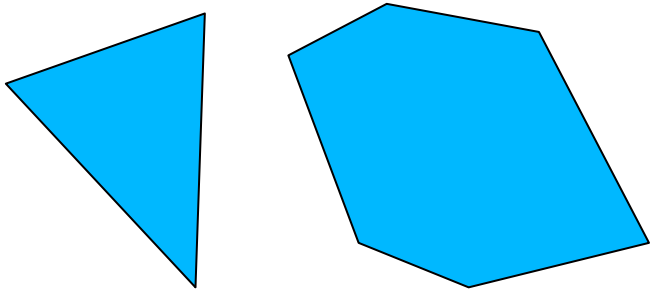


Modo de polígonos

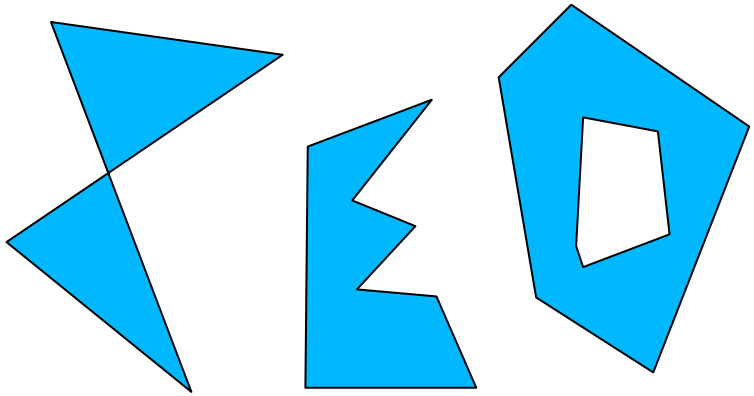
- **`glPolygonMode(`*`face, mode`*`)`**
  - *face* – faces para as quais queremos alterar modo
    - GL_FRONT
    - GL_BACK
    - GL_FRONT_AND_BACK
  - *mode* – modo de apresentação dos polígonos
    - GL_POINT
    - GL_LINE
    - GL_FILL
- Por omissão as faces cujos vértices são indicados CCW são faces “viradas para a frente”
- Por omissão as faces são preenchidas (GL_FILL)

Polígonos convexos

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



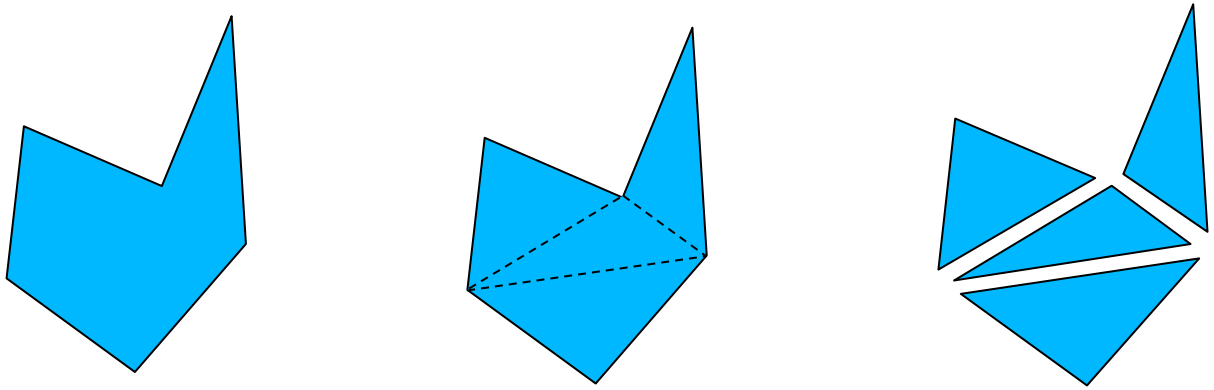
Convexo



Não Convexo

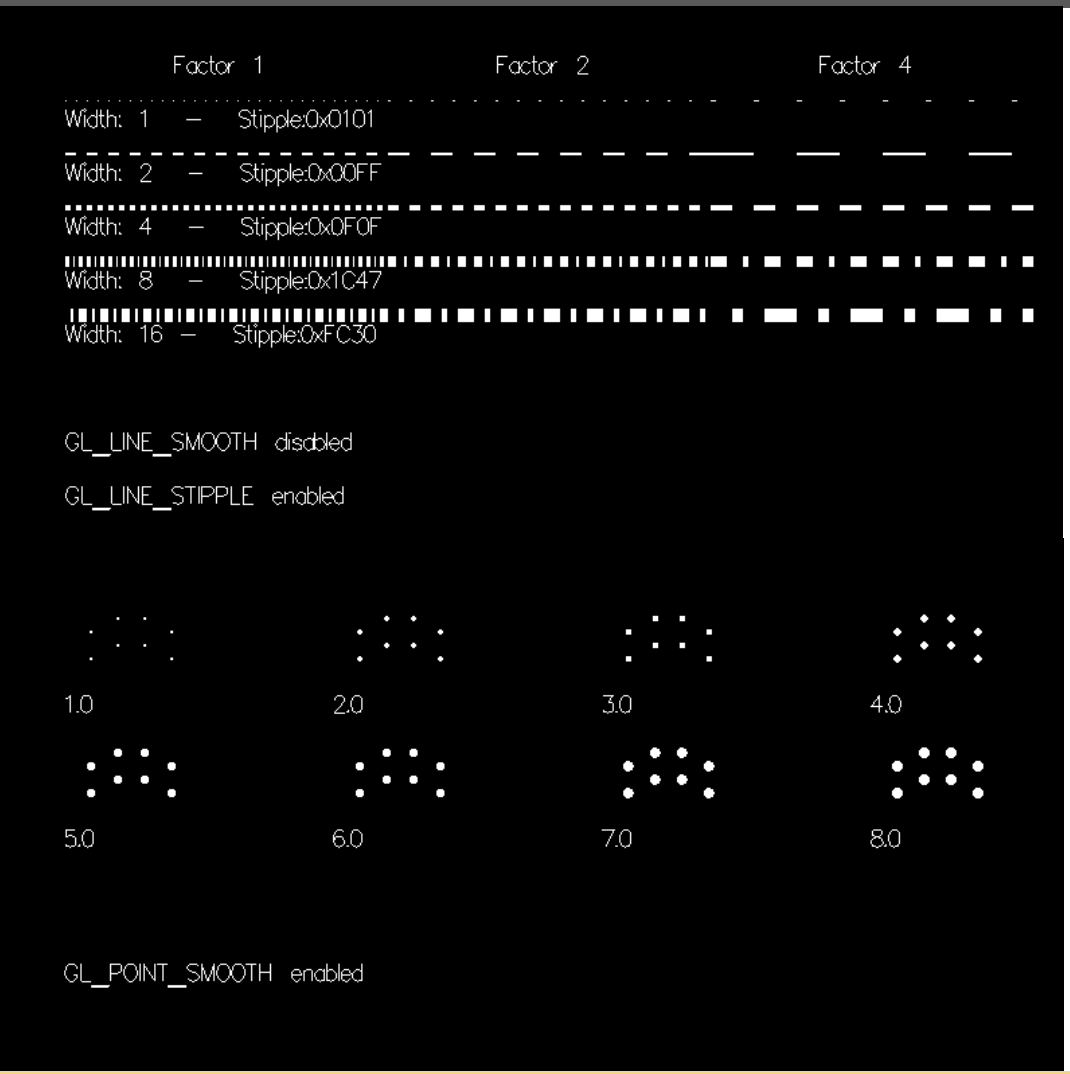
Polígonos não convexos

- 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

Linhas e pontos

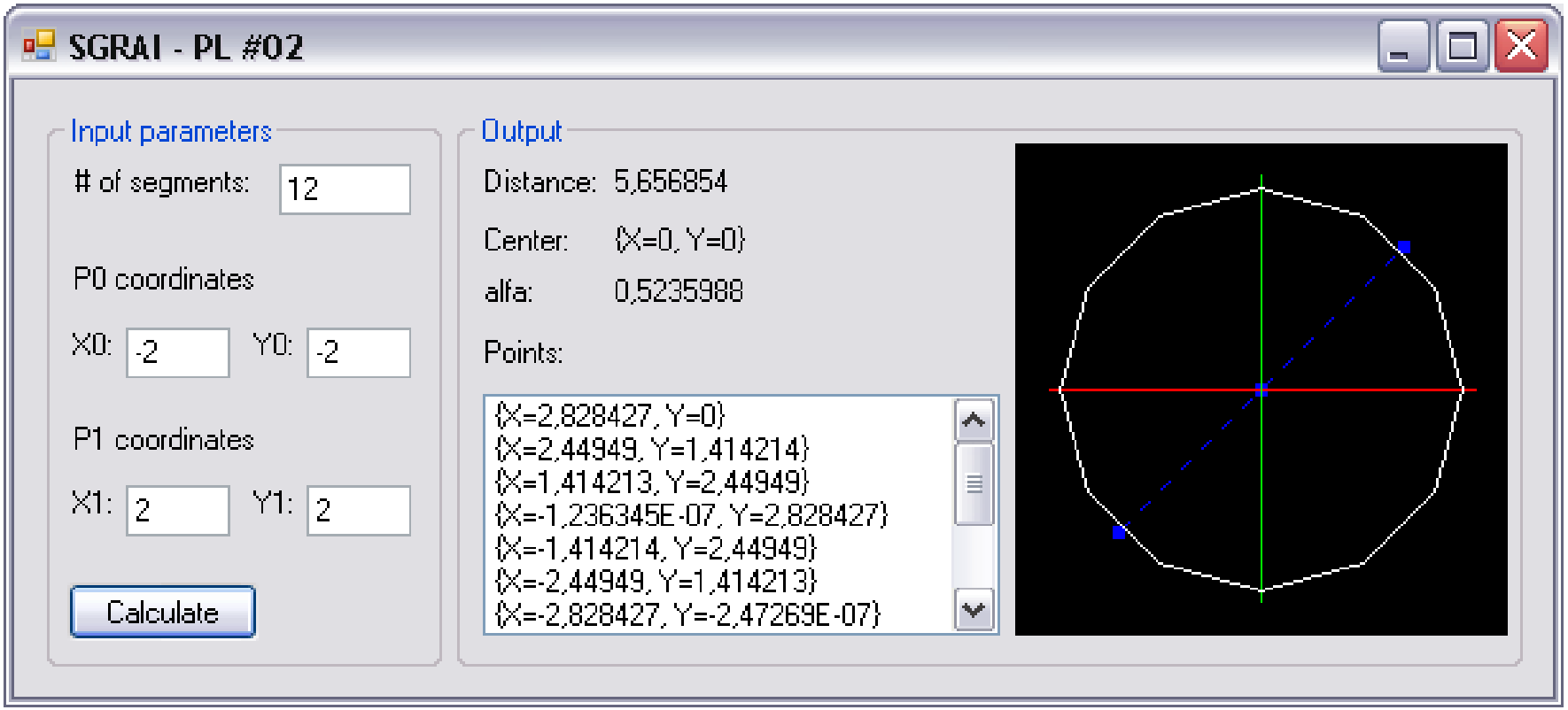


- **glLineWidth**
- **glLineStipple**
  - **0x0101 // dotted**
  - **0x00FF // dashed**
  - **0x1C47 // dash-dot-dash**
- **glEnable(GL_LINE_STIPPLE)**

- **glPointSize**

- Fora de glBegin/glEnd

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}

Módulo 6

Sistemas Gráficos e Interação

Instituto Superior de Engenharia do Porto

Filipe Pacheco

ffp@isep.ipp.pt

Animações

Contéudo

- Double buffer
- Animações em GLUT

Animações

- *Double buffer*
  - Desenhar próximo *frame* num *buffer* escondido e não no *buffer* de ecrã
  - Quando a cena estiver completa, trocar o *buffer* de ecrã pelo *buffer* escondido
- **`glutInit(GLUT_DOUBLE|...)`**
- **`glutSwapBuffers()`** em vez de **`glFlush()`** na *callback* de *display*

Animação usando GLUT

```
#include <GL/glut.h>

static GLfloat spin = 0.0;
int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
   glutInitWindowSize (250, 250);
   glutInitWindowPosition (100, 100);
   glutCreateWindow (argv[0]);
   glClearColor (0.0, 0.0, 0.0, 0.0);
   glShadeModel (GL_FLAT);
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutMouseFunc(mouse);
   glutMainLoop();
   return 0;
}
```

Animação usando GLUT

```
void display(void)
{
  glClear(GL_COLOR_BUFFER_BIT);
  glColor3f(1.0, 1.0, 1.0);
  glBegin(GL_QUADS);
    glVertex2f(25*cos(RAD(spin)), 25*sin(RAD(spin)));
    glVertex2f(25*cos(RAD(spin+90)), 25*sin(RAD(spin+90)));
    glVertex2f(25*cos(RAD(spin+180)), 25*sin(RAD(spin+180)));
    glVertex2f(25*cos(RAD(spin+270)), 25*sin(RAD(spin+270)));
  glEnd();
  glutSwapBuffers();
}
```

O efeito de animação é conseguido alterando a variável **spin**

Animação usando GLUT

```
void mouse(int button, int state, int x, int y)  {
   switch (button) {
      case GLUT_LEFT_BUTTON:
         if (state == GLUT_DOWN) glutIdleFunc(spinDisplay);
         break;
      case GLUT_MIDDLE_BUTTON:
         if (state == GLUT_DOWN) glutIdleFunc(NULL);
         break;
      default:
         break;
   }
}
void spinDisplay(void) {
   spin = spin + 2.0;
   if (spin > 360.0)
       spin = spin - 360.0;
   glutPostRedisplay();
}
```

Usando temporizadores

```
...

void main(int argc, char** argv)
{
    ...
    glutTimerFunc(10, anima, 1);
    glutMainLoop();
}

void anima(int v)
{
    glutTimerFunc(10, anima, 1);
    spin = spin + 2.0;
    if (spin > 360.0)
    spin = spin - 360.0;
    glutPostRedisplay();
}
```

NUNCA usamos instruções de desenho num timer…