Geometria Computacional

Cristina G. Fernandes

Departamento de Ciência da Computação do IME-USP

http://www.ime.usp.br/~cris/

segundo semestre de 2009

Representação de ponto

Ponto: vetor de dimensão apropriada

Representação de ponto

Ponto: vetor de dimensão apropriada

Ficar nos inteiros enquanto for possível.

Representação de ponto

Ponto: vetor de dimensão apropriada

Ficar nos inteiros enquanto for possível.

```
#define X 0
#define Y 1
#define DIM 2 /* dimensão do espaço */

/* tipo ponto inteiro */
typedef int tPointi[DIM];

/* tipo ponto real */
typedef double tPointd[DIM];
```

Representação de polígono

Polígono: vetor ou lista ligada de pontos

Representação de polígono

Polígono: vetor ou lista ligada de pontos

Qual das duas opções escolher? Depende...

Representação de polígono

Polígono: vetor ou lista ligada de pontos

Qual das duas opções escolher? Depende...

Com vetor...

```
/* número máximo de pontos em um polígono */
#define PMAX 1000

/* tipo polígono de pontos inteiros */
typedef tPointi tPolygoni[PMAX];

/* tipo polígono de pontos reais */
typedef tPointd tPolygond[PMAX];
```

Cálculos de área

Triângulo

```
int Area2 (tPointi a, b, c) {
   return a[X]*b[Y]-a[Y]*b[X]+a[Y]*c[X]
          -a[X]*c[Y]+b[X]*c[Y]-c[X]*b[Y];
}
```

Cálculos de área

Triângulo

```
int Area2 (tPointi a, b, c) {
   return a[X]*b[Y]-a[Y]*b[X]+a[Y]*c[X]
          -a[X]*c[Y]+b[X]*c[Y]-c[X]*b[Y];
}
```

Com menos multiplicações e em pseudocódigo:

Área2(a, b, c)
1 **devolva** $(a[X] - c[X]) * (b[Y] - c[Y]) -$
$(a[Y] - c[Y]) * (b[X] - c[X])$

Cálculos de área

Triângulo

Área2(a, b, c)
1 **devolva** $(a[X] - c[X]) * (b[Y] - c[Y]) -$
$(a[Y] - c[Y]) * (b[X] - c[X])$

Polígono

Cálculos de área

Triângulo

Área2(a, b, c)
1 **devolva** $(a[X] - c[X]) * (b[Y] - c[Y]) -$
$(a[Y] - c[Y]) * (b[X] - c[X])$

Polígono

ÁreaPolígono2(n, P)
1 $s \leftarrow 0$
2 **para** $i \leftarrow 1$ **até** $n - 2$ **faça**
3 $s \leftarrow s +$Área2$(P[0], P[i], P[i + 1])$
4 **devolva** s

Segmentos e pontos

Ponto c à esquerda da reta dada por $\vec{ab}$

Esquerda$^+(a, b, c)$

1 **devolva** Área2$(a, b, c) > 0$

Segmentos e pontos

Ponto c à esquerda da reta dada por $\vec{ab}$

Esquerda$^+(a, b, c)$

1 **devolva** Área2$(a, b, c) > 0$

Ponto c à esquerda ou sobre a reta dada por $\vec{ab}$

Esquerda(a, b, c)

1 **devolva** Área2$(a, b, c) \geq 0$

Segmentos e pontos

Ponto c à esquerda da reta dada por $\vec{ab}$

Esquerda$^+(a, b, c)$

1 **devolva** Área2$(a, b, c) > 0$

Ponto c à esquerda ou sobre a reta dada por $\vec{ab}$

Esquerda(a, b, c)

1 **devolva** Área2$(a, b, c) \geq 0$

Pontos a, b e c são colineares

Colinear(a, b, c)

1 **devolva** Área2$(a, b, c) = 0$

Interseção de segmentos

Interseção própria entre ab e cd:
a interseção é um único ponto no interior dos segmentos.

IntersectaProp(a, b, c, d)
1 **se** Colinear(a, b, c) **ou** Colinear(a, b, d)
ou Colinear(c, d, a) **ou** Colinear(c, d, b)
2 **então devolva** FALSO
3 **devolva** Xor(Esquerda$^+(a, b, c)$,Esquerda$^+(a, b, d)$)
e Xor(Esquerda$^+(c, d, a)$,Esquerda$^+(c, d, b)$)

A rotina Xor devolve o
ou exclusivo entre duas expressões booleanas.

Interseção de segmentos

Ponto c está no segmento ab

Entre(a, b, c)

1 **se não** colinear(a, b, c)
2 **então devolva** FALSO
3 **se** $a[X] \neq b[X]$ ▷ ab não é vertical
4 **então devolva** $a[X] \leq c[X] \leq b[X]$ **ou** $b[X] \leq c[X] \leq a[X]$
5 **senão devolva** $a[Y] \leq c[Y] \leq b[Y]$ **ou** $b[Y] \leq c[Y] \leq a[Y]$

Interseção de segmentos

Ponto c está no segmento ab

Entre(a, b, c)
1 **se não** Colinear(a, b, c)
2 **então devolva** FALSO
3 **se** $a[X] \neq b[X]$ ▷ ab não é vertical
4 **então devolva** $a[X] \leq c[X] \leq b[X]$ **ou** $b[X] \leq c[X] \leq a[X]$
5 **senão devolva** $a[Y] \leq c[Y] \leq b[Y]$ **ou** $b[Y] \leq c[Y] \leq a[Y]$

Interseção entre ab e cd

Intersecta(a, b, c, d)
1 **se** IntersectaProp(a, b, c, d)
2 **então devolva** VERDADE
3 **devolva** Entre(a, b, c) **ou** Entre(a, b, d)
ou Entre(c, d, a) **ou** Entre(c, d, b)

Dentro ou fora?

Candidata a diagonal $P[i]P[j]$ está no interior do polígono?

Está no cone das arestas vizinhas do polígono?

```
NoCone(n, P, i, j)
1  u ← i − 1 (mod n)
2  w ← i + 1 (mod n)
3  se Esquerda(P[u], P[i], P[w])          ▷ P[i] é convexo
4     então devolva Esquerda⁺(P[i], P[j], P[u]) e
                    Esquerda⁺(P[j], P[i], P[w])
5     senão devolva não (Esquerda(P[i], P[j], P[w]) e
                         Esquerda(P[j], P[i], P[u]))
```

Teste de diagonal

Quase uma diagonal...

QuaseDiagonal(n, P, i, j)

```
1  para k ← 0 até n − 1
2     kp ← k + 1 ( mod n)
3     se k ≠ i e k ≠ j e kp ≠ i e kp ≠ j
4        então se Intersecta(P[i], P[j], P[k], P[kp])
5                 então devolva FALSO
6  devolva VERDADE
```

Teste de diagonal

Quase uma diagonal...

QuaseDiagonal(n, P, i, j)

```
1  para k ← 0 até n − 1
2     kp ← k + 1 ( mod n)
3     se k ≠ i e k ≠ j e kp ≠ i e kp ≠ j
4        então se Intersecta(P[i], P[j], P[k], P[kp])
5                 então devolva FALSO
6  devolva VERDADE
```

Diagonal de fato...

Diagonal(n, P, i, j)

```
1  devolva NoCone(n, P, i, j) e QuaseDiagonal(n, P, i, j)
```

Teste de diagonal

Quase uma diagonal...

QuaseDiagonal(n, P, i, j)

```
1  para k ← 0 até n − 1
2     kp ← k + 1 ( mod n)
3     se k ≠ i e k ≠ j e kp ≠ i e kp ≠ j
4        então se Intersecta(P[i], P[j], P[k], P[kp])
5                 então devolva FALSO
6  devolva VERDADE
```

Diagonal de fato...

Diagonal(n, P, i, j)

```
1  devolva NoCone(n, P, i, j) e QuaseDiagonal(n, P, i, j)
```

Tempo de execução: $\Theta(n)$