

## 2D grafikai algoritmusok

[A quadtree/octtree algoritmus](#)

[A floodfill algoritmus](#)

[Belső vagy külső pont?](#)

[Baricentrikus koordináták](#)

[Körüljárási irány eldöntése](#)

[Animáció](#)

## A quadtree/octtree algoritmus

Legyen  $\Omega_0 \subset \mathbf{R}^2$  egy négyzet,  $S_0 := \{x_1, x_2, \dots, x_N\} \subset \Omega_0$  egy véges ponthalmaz (vezérlő pontok),  $N_{\min}, L_{\max} \in \mathbf{N}$  adottak.

```
procedure build( $\Omega, S, L$ );  
begin  
  if ( $L < L_{\max}$ ) and ( $|S| \geq N_{\min}$ ) then begin  
    ... (bontsuk fel  $\Omega$ -t az  $\Omega_1, \Omega_2, \Omega_3, \Omega_4$   
      egybevágó rész-négyzetekre) ...  
    build( $\Omega_1, S \cap \Omega_1, L+1$ );  
    build( $\Omega_2, S \cap \Omega_2, L+1$ );  
    build( $\Omega_3, S \cap \Omega_3, L+1$ );  
    build( $\Omega_4, S \cap \Omega_4, L+1$ );  
  end;  
end;
```

Ezekután a QT-háló *generálása*:

```
build(  $\Omega_0, s_0, 0$  );
```

*Műveletigény*:  $O(N \cdot L_{\max})$ .

*Reprezentálása*: fa-gráffal (elemek: cellák, élek: felbontások).

*3D algoritmus*: octtree (kocka 8-felé darabolása).

*Reguláris QT-háló*: bármely két szomszédos cella méretének aránya legfeljebb 2.

Minden QT-háló regularizálható pótlólagos cellafelbontásokkal.

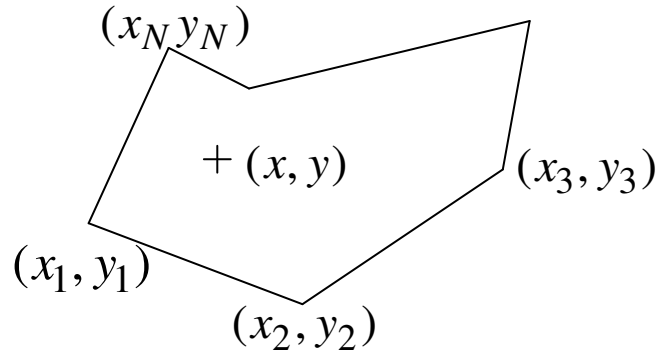
## A floodfill algoritmus

*Belső* pontok (pixelek) automatikus megkeresése

```
procedure fill(x, y: integer);  
begin  
    if (Pixels[x, y] <> col_border) and  
        (Pixels[x, y] <> col_paint) then begin  
        Pixels[x, y] := col_paint;  
        fill(x+1, y);  
        fill(x, y+1);  
        fill(x-1, y);  
        fill(x, y-1);  
    end;  
end;
```

Más adatstruktúrákban (pl. QT-hálón) is működik.

## Belső vagy külső pont?

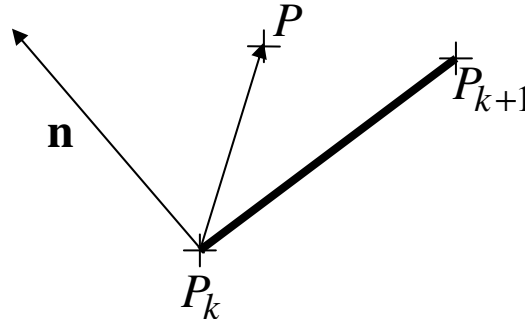


*Probléma:* Adott egy  $N$ -szög a csúcspontjai sorozatával, és egy  $(x, y)$  pont. Döntsük el, hogy  $(x, y)$  az  $N$ -szög belsejében vagy külsejében (esetleg a határán) van!

*1. megoldás:* Jelölje  $P_k := (x_k, y_k)$  ( $k = 1, 2, \dots, N$ ), és  $P := (x, y)$ .  
( $P_{N+1} := (x_1, y_1)$ .) Számítsuk ki az  $\alpha_k := \angle P_k P P_{k+1}$  előjeles szögeket  
( $k = 1, 2, \dots, N$ ). Ha a  $\sum_{k=1}^N \alpha_k$  szögösszeg  $2\pi$ , akkor  $P$  belső pont, ha 0,  
akkor külső pont.

*2. megoldás:* Húzzunk  $P$ -n keresztül egy tetszőleges félegyenest. Ha ez páratlan sokszor metszi a peremet, akkor  $P$  belső pont, ha páros sokszor, akkor külső pont.

**3. megoldás (csak konvex sokszögekre működik):** Ha a  $P$  pont mindegyik  $P_k P_{k+1}$  egyenesnek ugyanazon oldalán fekszik, akkor  $P$  belső pont, egyébként külső pont.

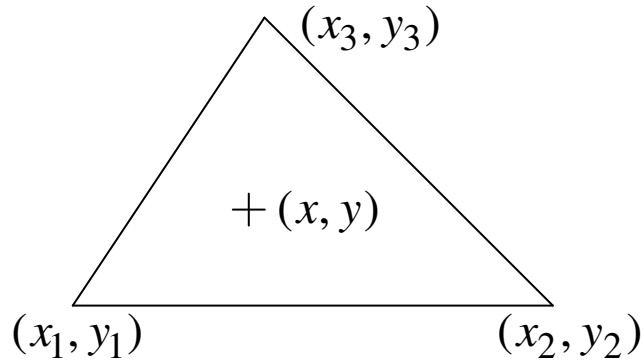


$$\mathbf{n} := (P_k P_{k+1})^\perp = (-(y_{k+1} - y_k), (x_{k+1} - x_k))$$

$$\mathbf{e} := P_k P = (x - x_k, y - y_k)$$

Ha  $\langle \mathbf{n}, \mathbf{e} \rangle > 0$ , akkor  $P$  a  $P_k P_{k+1}$  egyenes bal oldalán, ha  $\langle \mathbf{n}, \mathbf{e} \rangle < 0$ , akkor a jobb oldalán fekszik.

## Baricentrikus koordináták



A háromszög belső  $(x, y)$  pontjai (és csak ezek) előállnak

$$(x, y) = \sum_{j=1}^3 \lambda_j \cdot (x_j, y_j)$$

konvex kombinációként:  $\lambda_1, \lambda_2, \lambda_3 > 0$ ,  $\lambda_1 + \lambda_2 + \lambda_3 = 1$ .

$\lambda_1, \lambda_2, \lambda_3$ : az  $(x, y)$  pontnak a  $(x_1, y_1)$ ,  $(x_2, y_2)$ ,  $(x_3, y_3)$  pontokra vonatkoztatott *baricentrikus koordinátái*.

Megoldva tehát a

$$\lambda_1 x_1 + \lambda_2 x_2 + \lambda_3 x_3 = x$$

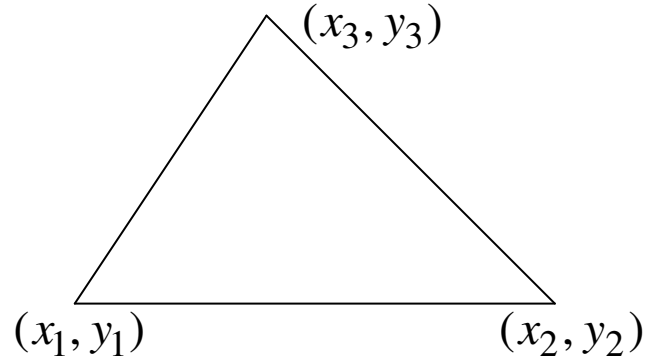
$$\lambda_1 y_1 + \lambda_2 y_2 + \lambda_3 y_3 = y$$

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

egyenletrendszer, ha  $\lambda_1, \lambda_2, \lambda_3 > 0$ , akkor  $(x, y)$  belső pont: ha valamelyik  $\lambda_j$  negatív, akkor külső pont.

Az eljárás konvex sokszögekre általánosítható.

## Körüljárási irány eldöntése



A  $P_1P_2P_3$  körüljárási irány pozitív, ha  $P_3$  a  $P_1P_2$  egyenes bal oldalán fekszik (ill. negatív, ha a jobb oldalán).

$$\mathbf{n} := (P_1P_2)^\perp = (-(y_2 - y_1), (x_2 - x_1))$$
$$\mathbf{e} := P_1P_3 = (x_3 - x_1, y_3 - y_1)$$

Ha  $\langle \mathbf{n}, \mathbf{e} \rangle > 0$ , akkor a  $P_1P_2P_3$  körüljárási irány pozitív, ha  $\langle \mathbf{n}, \mathbf{e} \rangle < 0$ , akkor negatív.

## Animáció

A mozgás újabb és újabb fázisait nem célszerű közvetlenül a grafikus memóriába írni (nem folyamatos mozgás illúzióját adja), hanem először egy háttérben alakítjuk ki a képet, amit utána közvetlenül (és gyorsan) megjelenítünk.

```
var Bitmap: TBitmap;
```

Inicializálás:

```
Bitmap := TBitmap.Create;  
Bitmap.Width := Form1.ClientWidth;  
Bitmap.Height := Form1.ClientHeight;
```

## A háttér puffer törlése:

```
Brush.Color:=clWhite;  
Rectangle(0,0,Bitmap.Width,Bitmap.Height);
```

## Rajzolás a háttér pufferbe: a Bitmap Canvas objektumán keresztül, pl.

```
Bitmap.Canvas.Pen.Color := clBlue;  
Bitmap.Canvas.MoveTo(x1,y1);  
Bitmap.Canvas.LineTo(x2,y2);
```

## A háttér puffer megjelenítése:

```
Form1.Canvas.Draw(0,0,Bitmap);
```