PPKE ITK

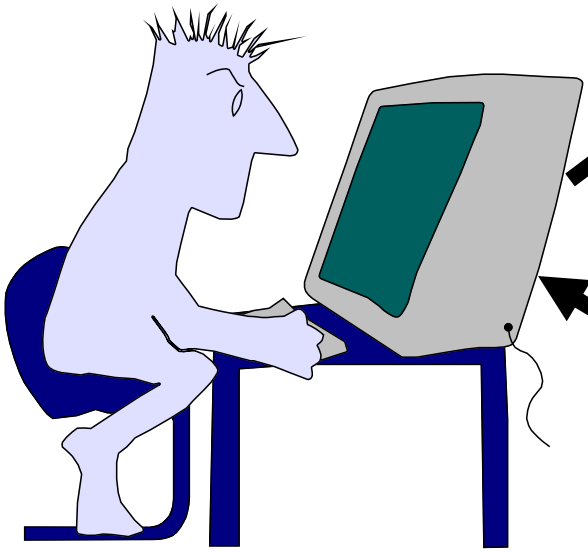
A számítógépes grafika alapjai

2D képszintézis

Előadó: Benedek Csaba

Tananyag : Szirmay-Kalos László, Benedek Csaba

Számítógépes grafika feladata



modellezés

képszintézis

Virtuális világ modell

világ = sík

Metafórák:

- 2D rajzolás

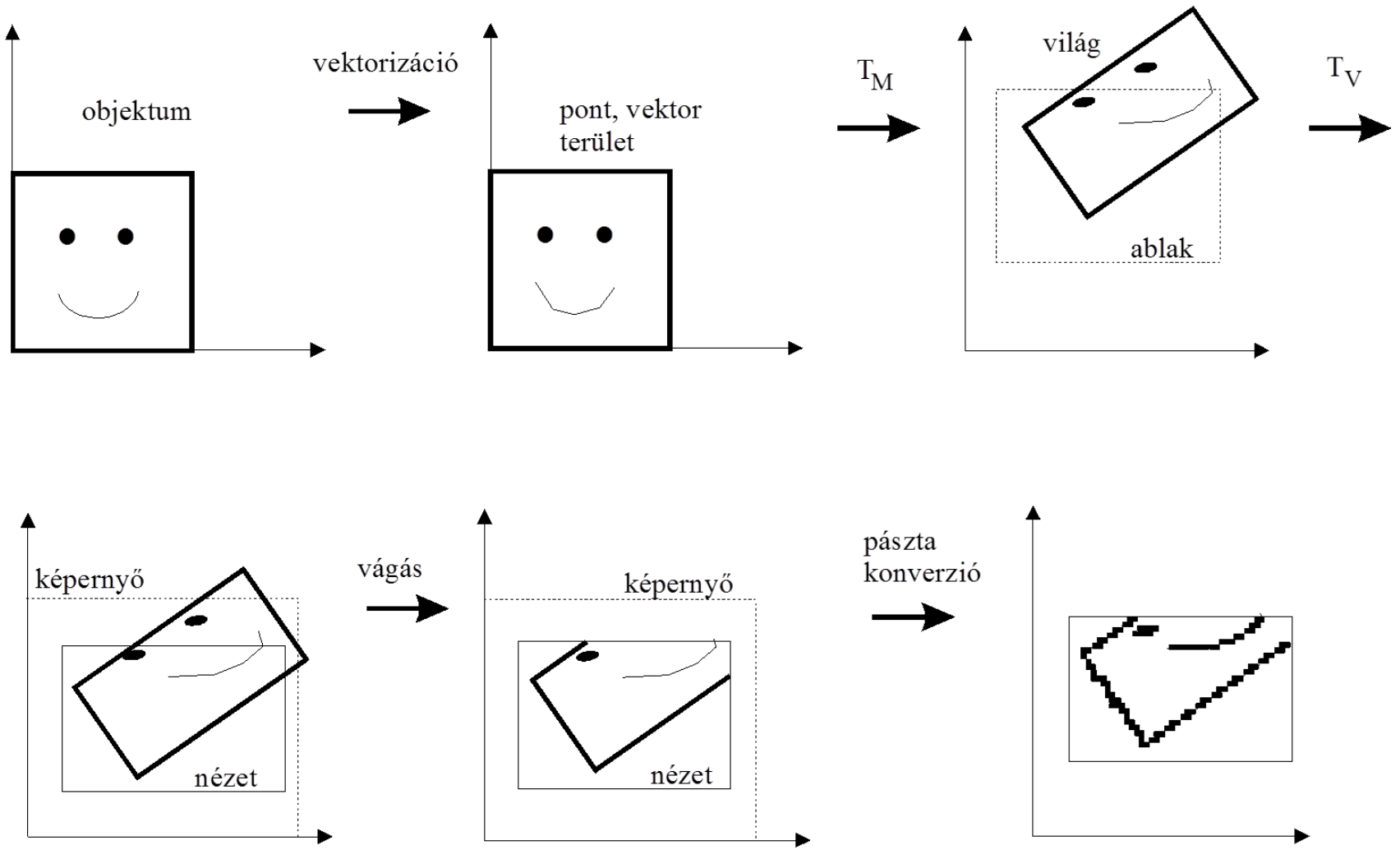
2D képszintézis

- **Cél:** objektumok képernyőre transzformálása és a nézetbe eső részek megjelenítése
  - Megjelenítés = a megfelelő pixelek kiszínezése
- Kiindulás:
  - lokális koordináta rendszerekben definiált objektumok
- Fő feladatok:
  - Objektumok transzformálása a világkoord.rsz-be
  - Ablak téglalap elhelyezése (2D kamera)
  - A képernyő koordinátarendszerébe való vetítés
  - A geometriai elemeket közelítő pixelek meghatározása

2D képszintézis lépései

- 1. *Vektorizáció:*
  - *A virtuális világban tárolt szabadformájú elemeket (például körívek,* interpolációs vagy spline görbék) pontok, szakaszok és poligonok halmazával közelítjük.
- 2. *Transzformáció:*
  - *a lokális koord.rsz →világ* koord.rsz → képernyő koord.rsz, homogén lineáris transzformáció
- 3. *Vágás:*
  - *csak az jeleníthető meg, ami a nézet téglalapján belül van,*
- 4. *Pásztakonverzió, avagy raszterizáció:*
  - *geometriai elemek közelítése pixelek átszínezésével*

2D képszintézis lépései

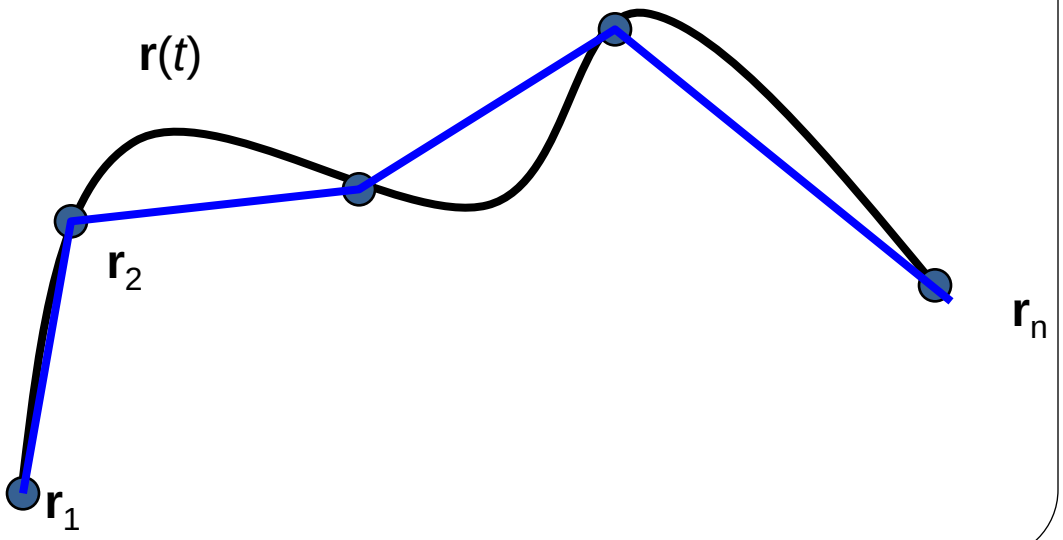


Vektorizáció

- A szabadformájú elemek közelítése pontokkal, szakaszokkal vagy poligonokkal
  - Homogén lineáris transzformáció és vágás nem változtatja meg az elemek típusát
  - Felosztás finomsága paraméter
    - Pl néhány pixel hosszú szakaszok nem növelik jelentősen a raszterizáció amúgy is jenlévő hibáját

$[0,1]$: $0, t_1, t_2, \ldots, t_n, 1$

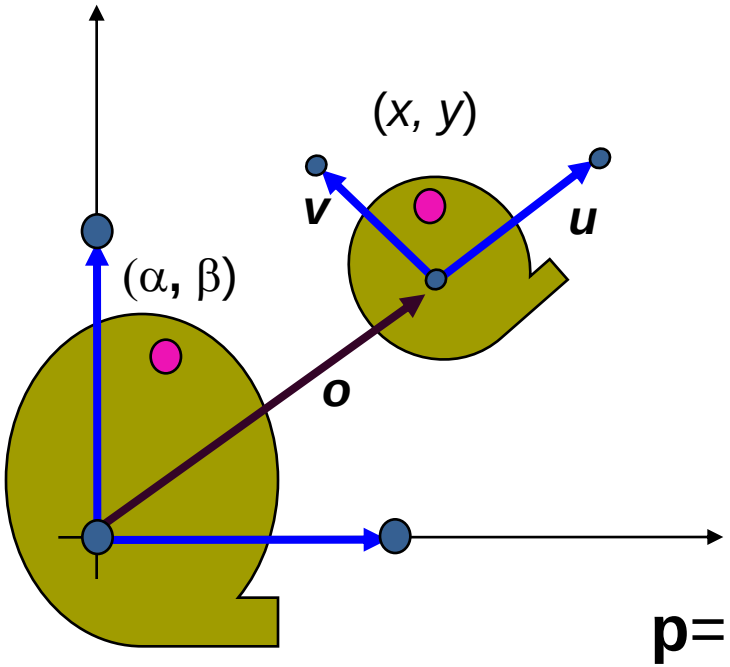
$\mathbf{r}_1 = \mathbf{r}(0), \mathbf{r}_2 = \mathbf{r}(t_2), \ldots, \mathbf{r}_n = \mathbf{r}(1)$



Modellezési transzformáció

$$\begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ p_x & p_y & 1 \end{bmatrix}$$

$\mathbf{T_M}$



$$[x, y, 1] = [\alpha, \beta, 1] \cdot \begin{bmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ o_x & o_y & 1 \end{bmatrix}$$

$$\mathbf{p}=(x, y) = \mathbf{o} + \alpha\boldsymbol{u} + \beta\boldsymbol{v}$$

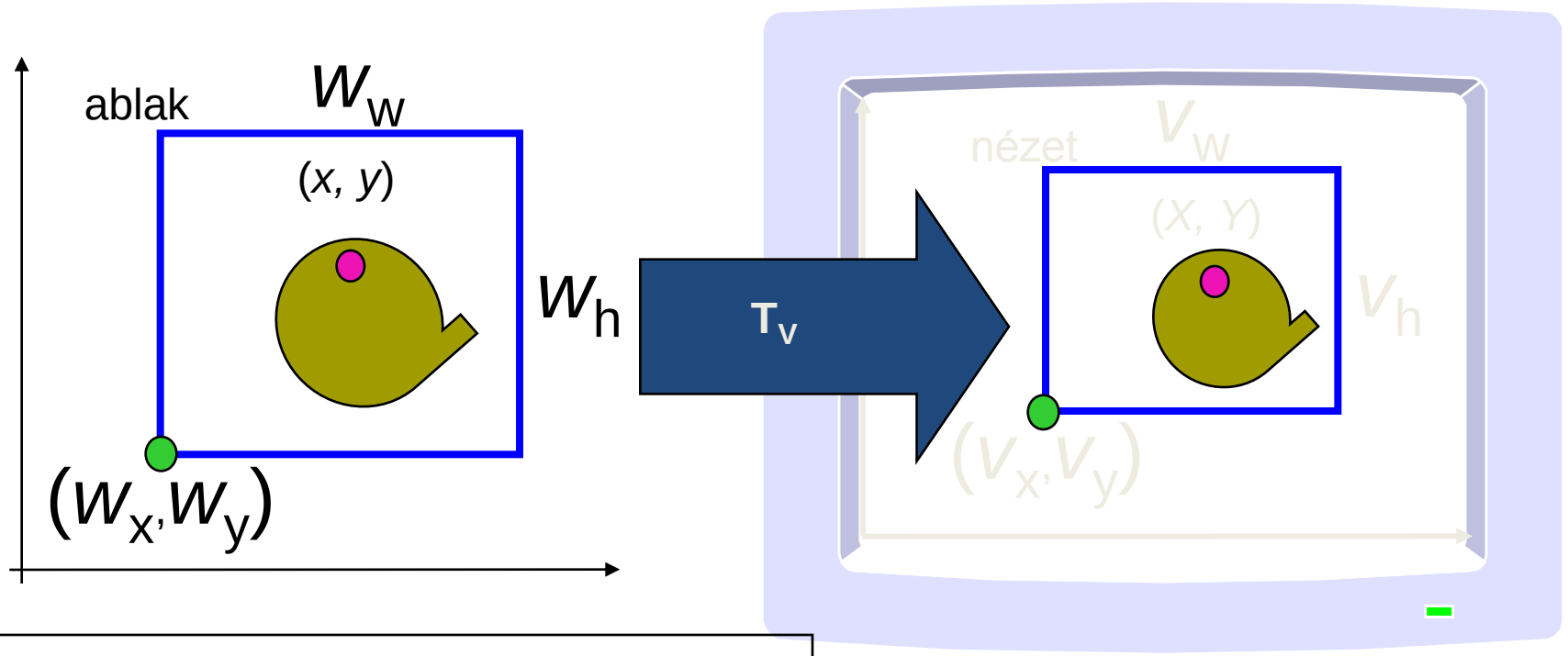
$\mathbf{p}$= [x,y]
$\mathbf{u}$=[u_x,u_y]
$\mathbf{v}$=[v_x,v_y]

[0, 0, 1] [o_x o_y 1]

[1, 0, 1] [o_x+u_x o_y+u_y 1]

[0, 1, 1] [o_x+v_x o_y+v_y 1]

Ablak-nézet transzformáció



$$X = (x\text{-}w_x)\ v_w/w_w + v_x$$
$$Y = (y\text{-}w_y)\ v_h/w_h + v_y$$

$$[X, Y, 1] = [x, y, 1]$$

$$\begin{bmatrix} v_w/w_w & 0 & 0 \\ 0 & v_h/w_h & 0 \\ v_x\text{-}w_x v_w/w_w & v_y\text{-}w_y v_h/w_h & 1 \end{bmatrix}$$

Összetett transzformáció

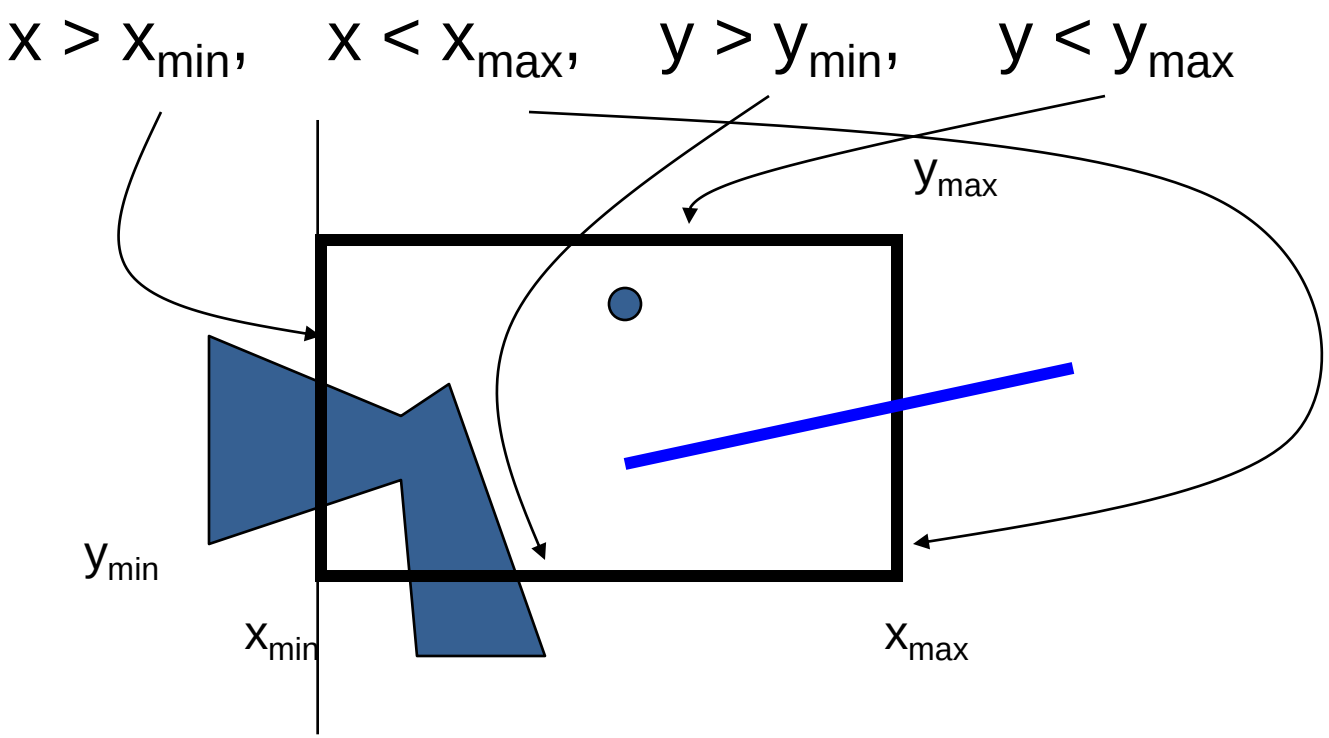
$$[X, Y, 1] = ([\alpha, \beta, 1]\ \mathbf{T}_M)\ \mathbf{T}_V$$

$$[X, Y, 1] = [\alpha, \beta, 1]\ (\mathbf{T}_M\ \mathbf{T}_V) = [\alpha, \beta, 1]\ \mathbf{T}_C$$

$\mathbf{T}_V$ nézeti transzformáció számítása
for each object o
  $\mathbf{T}_M$ előállítása
  $\mathbf{T}_C = \mathbf{T}_M\ \mathbf{T}_V$
  for each point of object o:
    transzformáció $\mathbf{T}_C$ -vel
  enfor
endfor

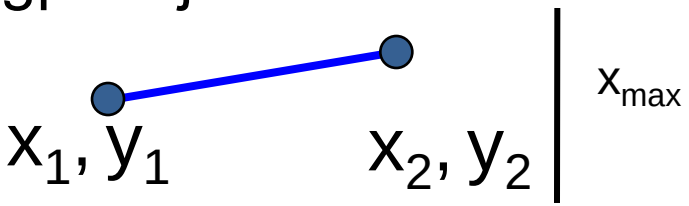
Vágás

- Ablakra vs. nézetre vágás
- Vektorizáció miatt: pont, szakasz, poligon
  - Pontok vágása

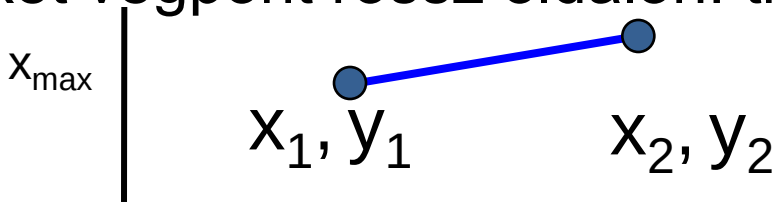


Szakasz vágás félsíkra

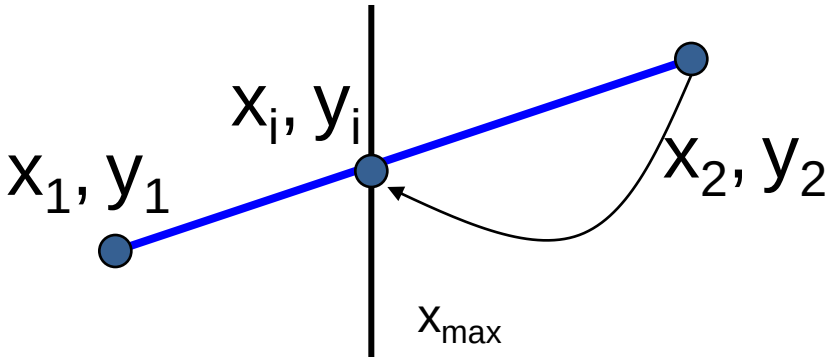
- Esetek (példa: x_{max} egyenes):
  - minkét végpont jó oldalon: triviális elfogadás



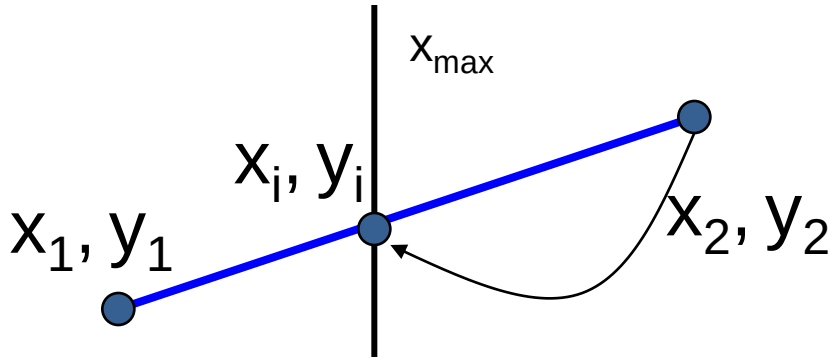
  - minkét végpont rossz oldalon: triviális eldobás



  - végpontok ellentétes oldalon: metszéspont számítás kell



Szakasz vágás félsíkra: metszéspont számítás

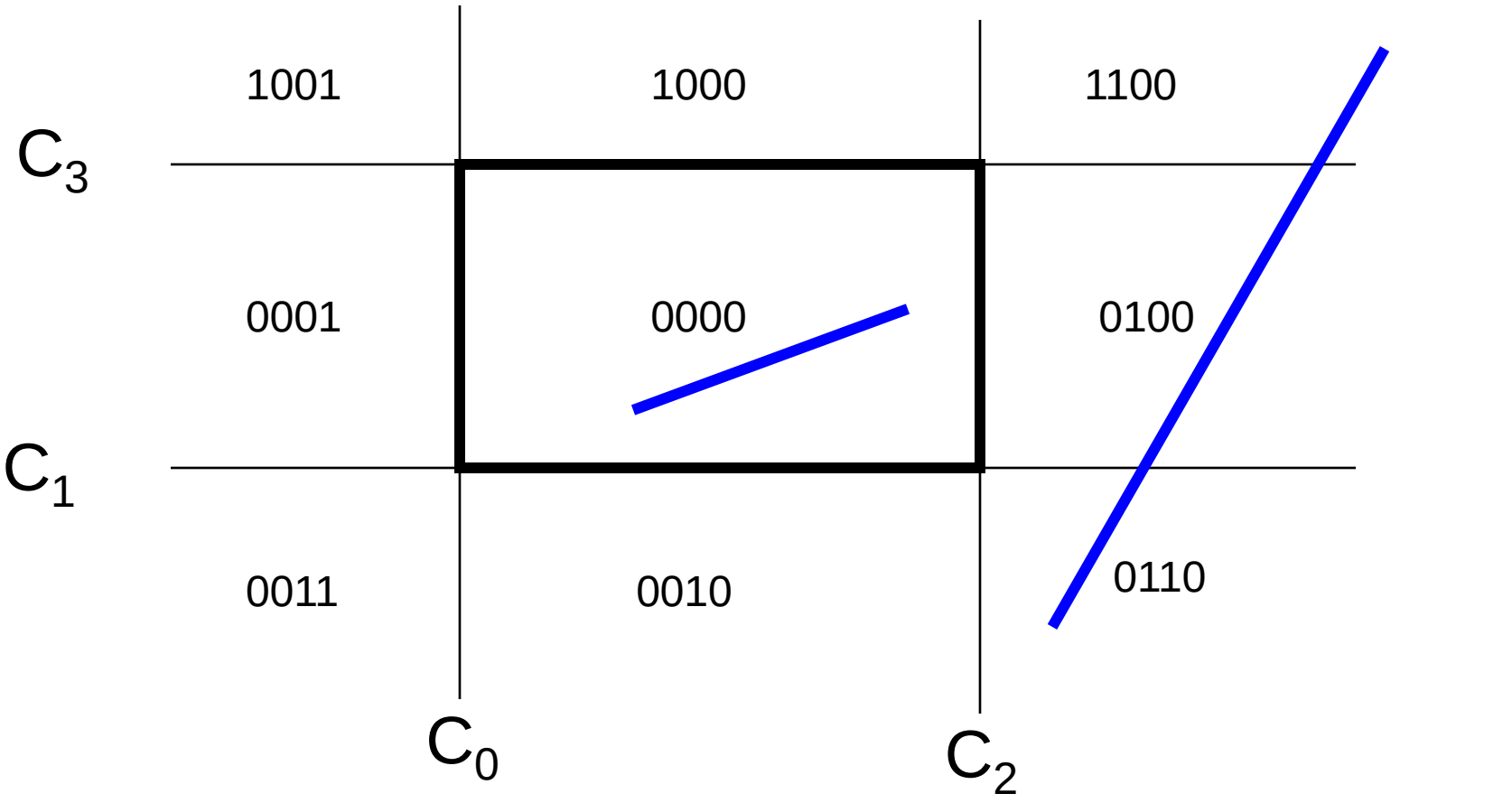


- $x(t) = x_1 + (x_2 - x_1)t, \quad y(t) = y_1 + (y_2 - y_1)t$
- $x = x_{max}$
  - Metszéspont:

    $x_{max} = x_1 + (x_2 - x_1)t \Rightarrow t = (x_{max}-x_1)/(x_2-x_1)$
- $x_i = x_{max} \qquad y_i = y_1 + (y_2 - y_1)\,(x_{max}-x_1)/(x_2-x_1)$

Cohen-Sutherland vágás

Kód=$C_3C_2C_1C_0$

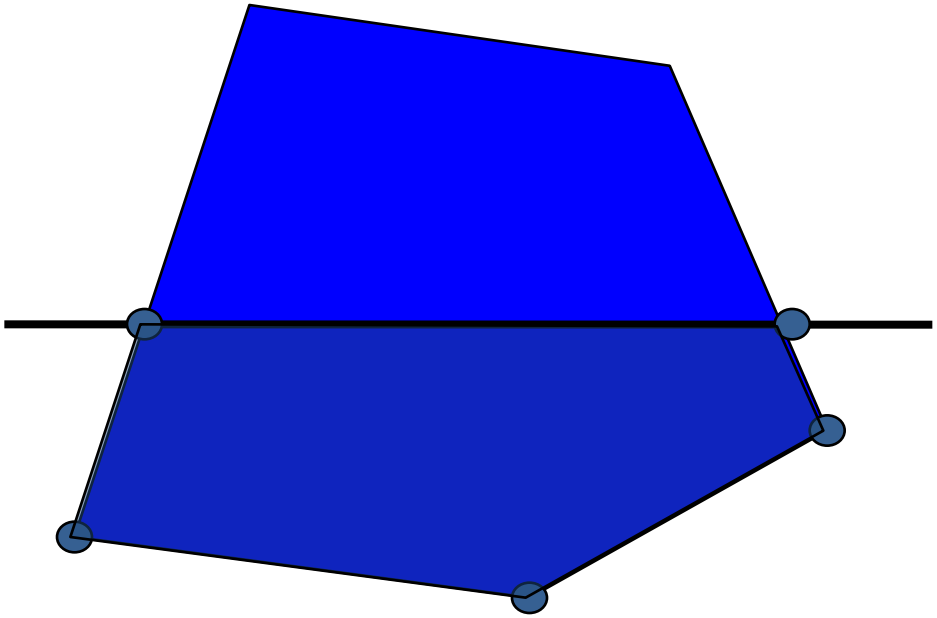


Cohen-Sutherland algoritmus

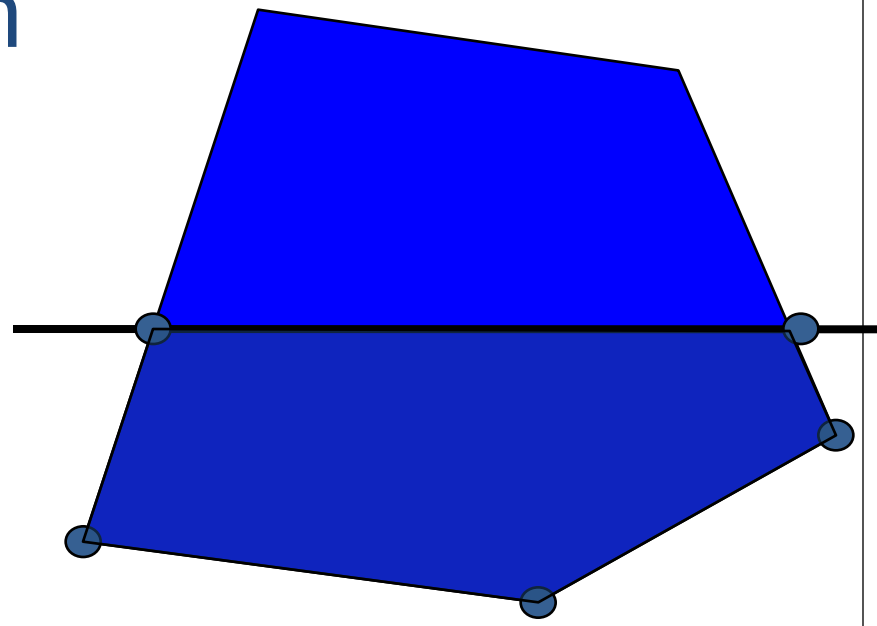
```
#define Code(p) ((p.x < xmin)      + ((p.y < ymin)<<1) + \\
                 ((p.x > xmax)<<2) + ((p.y > ymax)<<3)

bool LineClip(Vector& p1, Vector& p2) {
   for( ; ; ) {
      int c1 = Code(p1), c2 = Code(p2);
      if (c1 == 0 && c2 == 0) return true;
      if ((c1 & c2) != 0)     return false;

      if ((c1 & 1))       p1 = IntersectX(p1, p2, xmin);
      else if ((c1 & 2)) p1 = IntersectY(p1, p2, ymin);
      else if ((c1 & 4)) p1 = IntersectX(p1, p2, xmax);
      else if ((c1 & 8)) p1 = IntersectY(p1, p2, ymax);
      else if ((c2 & 1)) p2 = IntersectX(p1, p2, xmin);
      else if ((c2 & 2)) p2 = IntersectY(p1, p2, ymin);
      else if ((c2 & 4)) p2 = IntersectX(p1, p2, xmax);
      else if ((c2 & 8)) p2 = IntersectY(p1, p2, ymax);
   }
}
```

Poligonok vágása

- Továbbra is: négy félsíkra vágás...
  - Csúcspont belső pont: a vágott poligonnak is belső pontja
  - Csúcspont külső pont: eldobhatjuk, de egy új csúcspont is létrejöhet, ha az aktuális él metszi a vágás egyenesét!



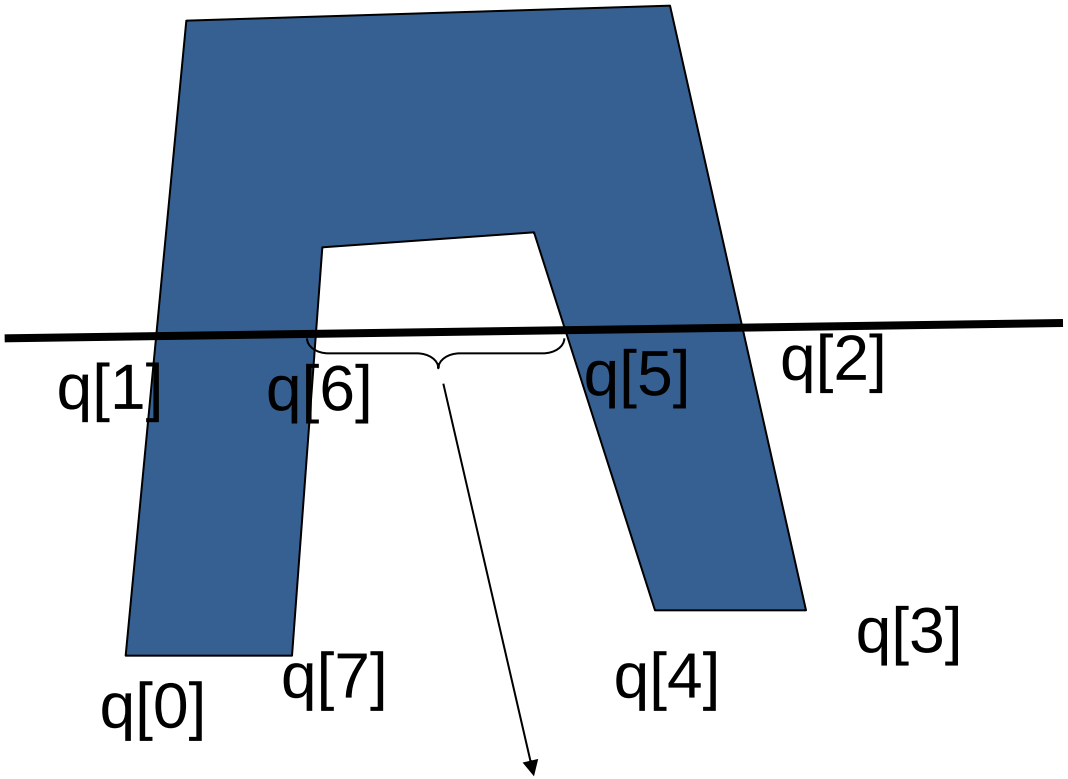
Sutherland-Hodgeman poligonvágás



```
PolygonClip(p[n] ⇨ q[m])
   m = 0;
   for( i=0; i < n; i++) {
      if (p[i] belső) {
         q[m++] = p[i];
         if (p[i+1] külső)
            q[m++] = Intersect(p[i], p[i+1], vágóegyenes);
      } else {
         if  (p[i+1] belső)
            q[m++] = Intersect(p[i], p[i+1], vágóegyenes);
      }
   }
}
```

Konkáv poligonok vágása

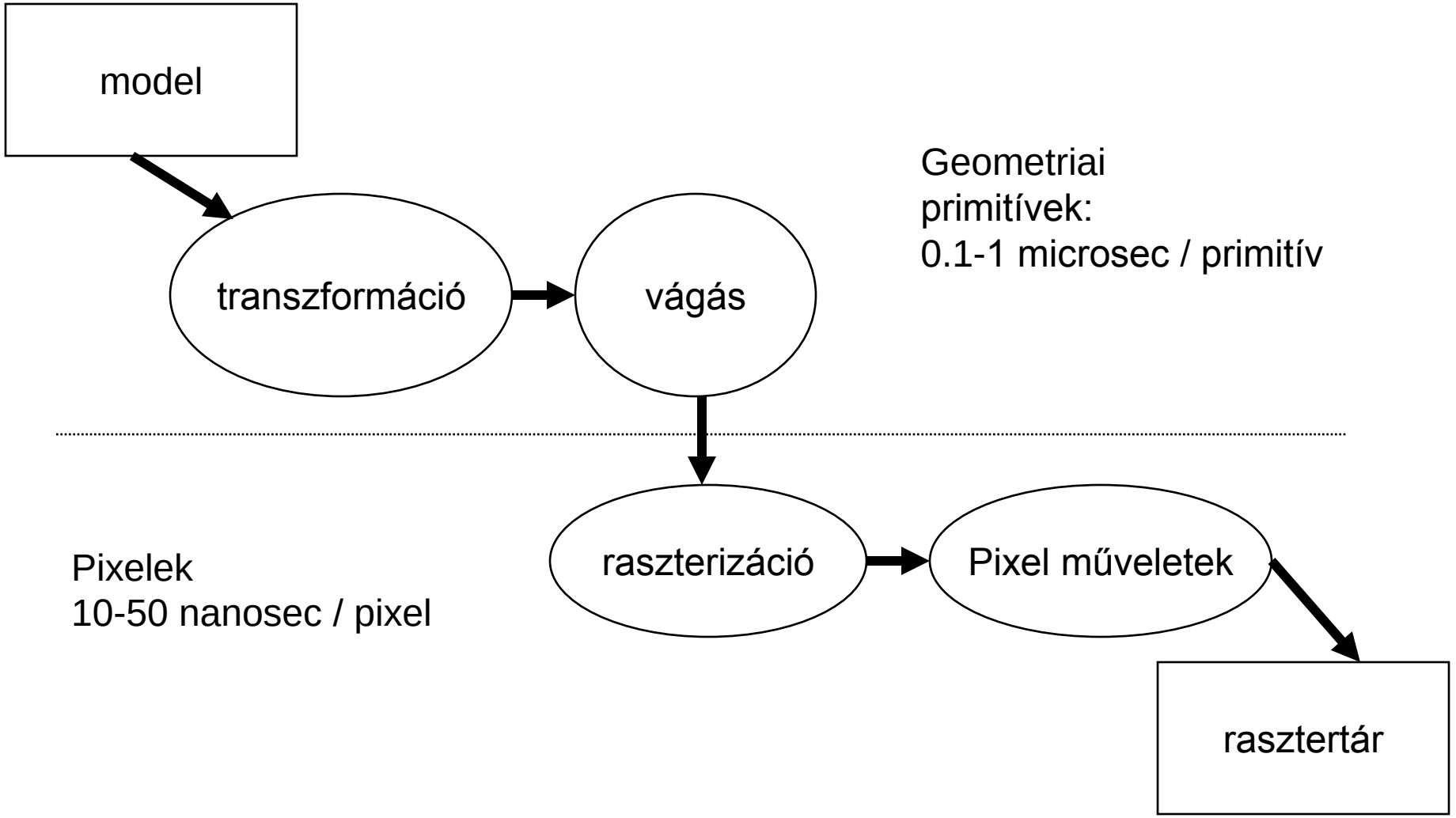
- Konkáv poligon több részre eshet szét



Raszterizáció

- Alakzatok formáinak közelítése: pixelek átszínezése útján
- Alapegység: pixel
  - egy geometria objektumhoz több ezer/millió pixel tartozhat
  - Következmény: az elvárt sebesség nagyságrendileg haladja meg az eddigi műveleteket!

Pászta konverzió (raszterizáció)

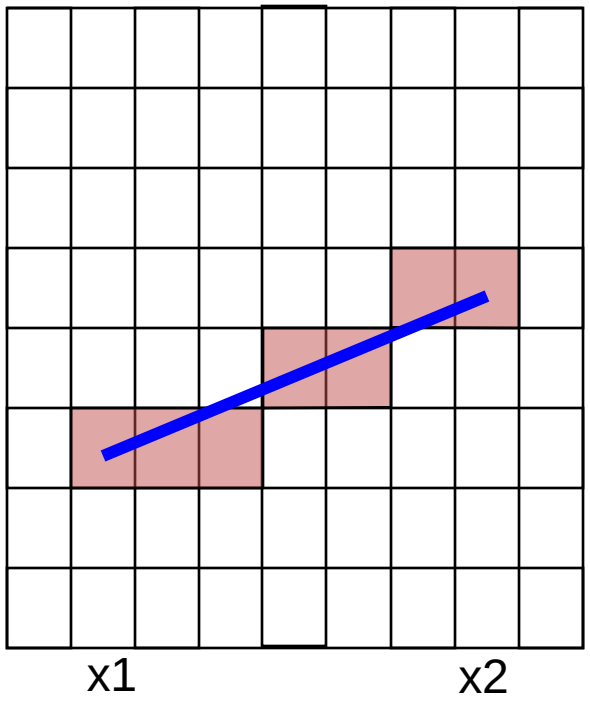


Szakasz rajzolás

Elvárások: a szakasz

(1) ne legyen lyukas (2) ne legyen túl vastag

Példa: enyhén emelkedő egyenes: 0 < m < 1:
minden oszlopban 1 pixel



Egyenes egyenlete:

$$y = mx + b$$

Egyeneshúzás (0 < m < 1):

```
m=(y2-y1)/(x2-x1)
b=y1-x1*(y2-y1)/(x2-x1)
for( x = x1; x <= x2; x++) {
        Y = m*x + b;
        y = Round( Y );
        write( x, y );
}
```

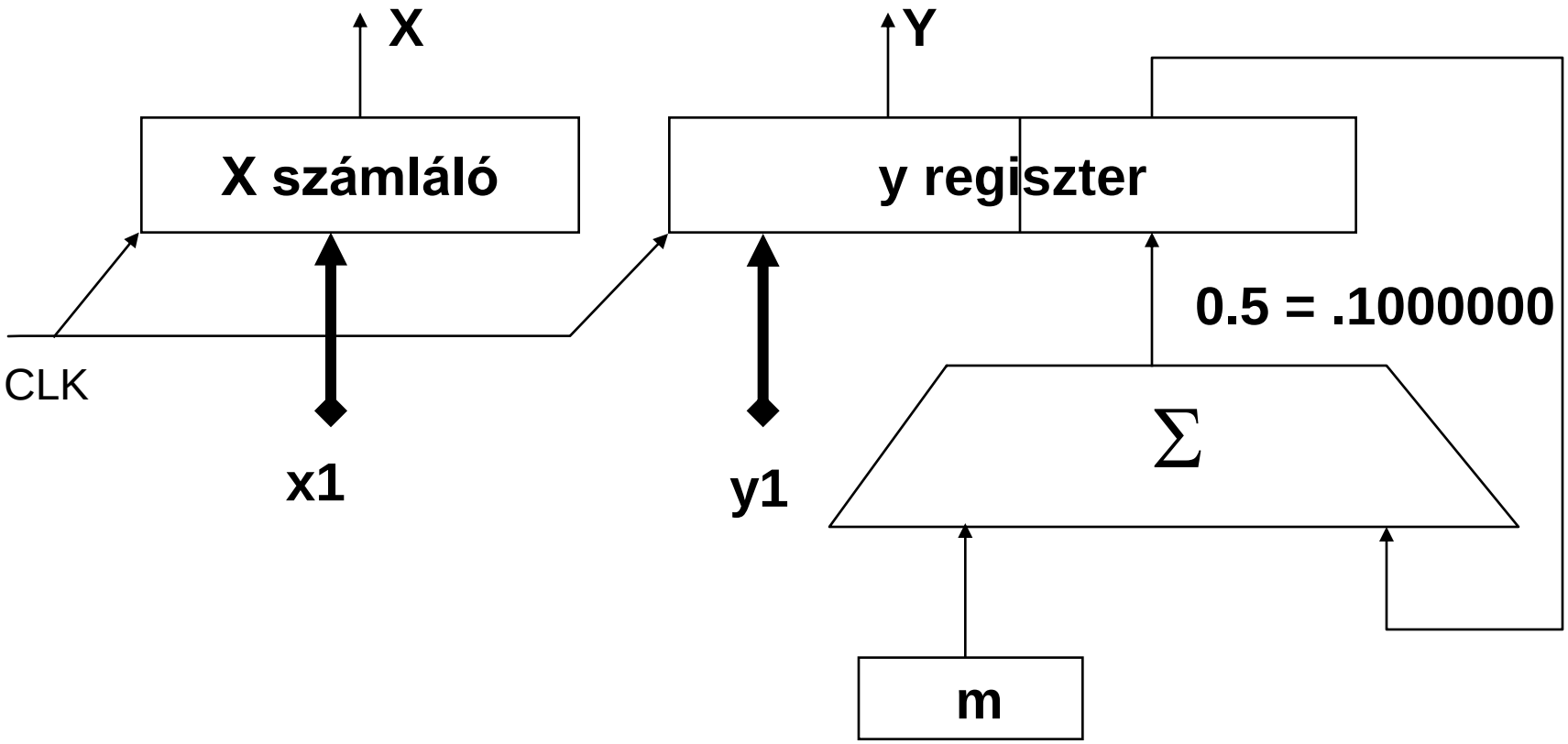
Inkrementális elv

- Y(X) kiszámítása
  - $Y(X) = F(Y(X-1)) = Y(X-1) + dY/dX$
- Szakasz rajzolás:
  - $Y(X) = mX + b = m(X-1)+b+m$
  - $Y(X) = F(Y(X-1)) = Y(X-1) + m$
- Összeadás: fixpontos ábrázolás: $y = Y\ 2^T$
  - T: hiba < 1 a leghosszabb műveletsornál
  - $N\ 2^{-T} < 1$: $T > \log_2 N$
    - N: leghosszabb szakasz hossza
  - round(y): y+0.5-t csonkítjuk (= „tört” bitek levágása)

DDA szakaszrajzolás

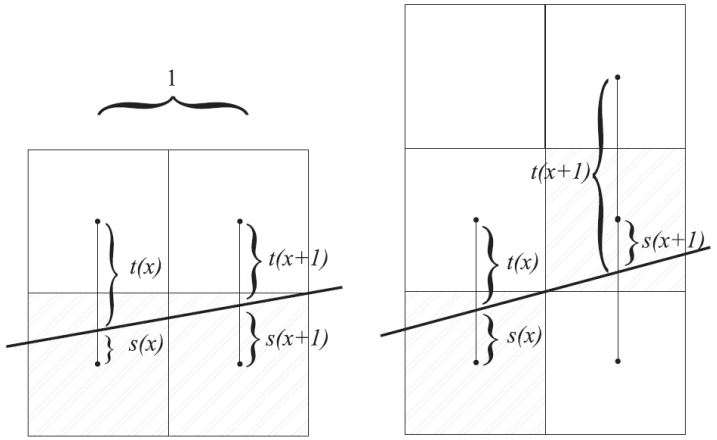
```
DDADrawLine(x1, y1, x2, y2) {
        m = (y2 - y1)/(x2 - x1)
        y = y1                                  + 0.5
        FOR X = x1 TO x2 {
                Y = round(y)                    trunc(y)
                WRITE(X, Y, color)
                y = y+m
        }
}
```

DDA szakaszrajzoló hardver



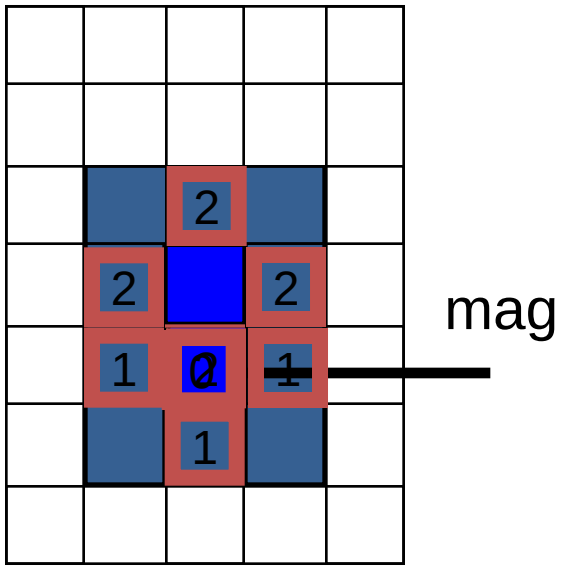
Bresenham szakaszrajzoló algoritmus

- DDA: gyors, de
  - a fixpontos ábrázolás és egészrész képzés eltolást (shift) igényel
  - szakaszonként egyszer lebegőpontos osztás (m)
- Megoldás: Bresenham algorithmusa (lásd tankönyv 103. oldal...



Terület elárasztás működése

Cél: adott kezdeti pontból egy adott határvonal átlépése nélkül elérhető pixelek átszínezése



Belső pont = sötétkék

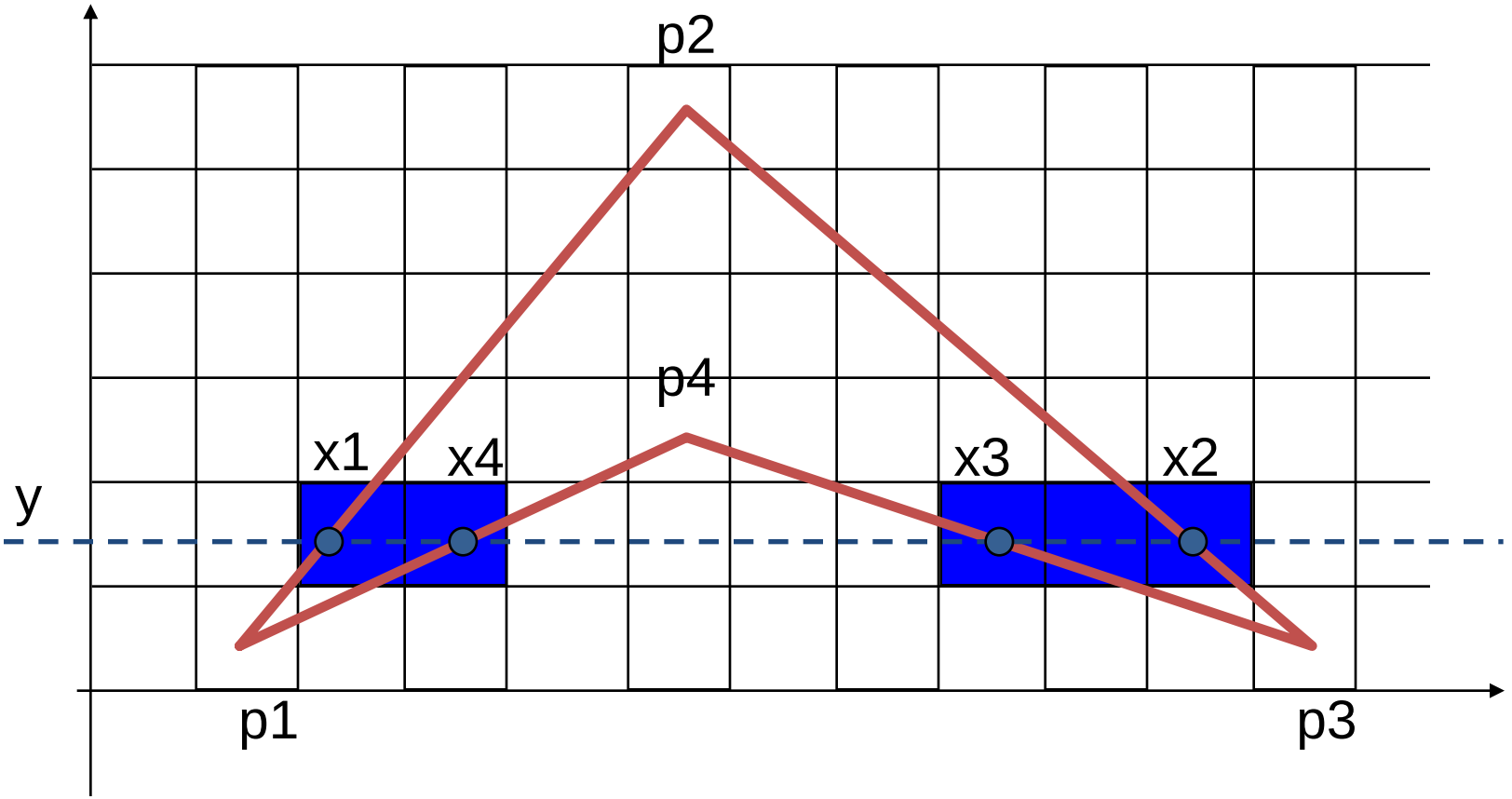
```
Flood(x, y) {
        if (pixel[x][y] belső pont) {
                Write(x, y, color);
                Flood(x, y-1);
                Flood(x, y+1);
                Flood(x-1, y);
                Flood(x+1, y);
        }
}
```

Területkitöltés

- Geometriai reprezentáció alapján
  - p1,p2,...,pn csúcsok
  - p1-p2, p2-p3,...,pn-p1 élek
- Belső pixel
  - páratlanszor metszünk élt, ha végtelenből jövünk
  - végtelen: vágás után a nézet széle

Területkitöltés

- Feldolgozás: vízszintes pásztánként:



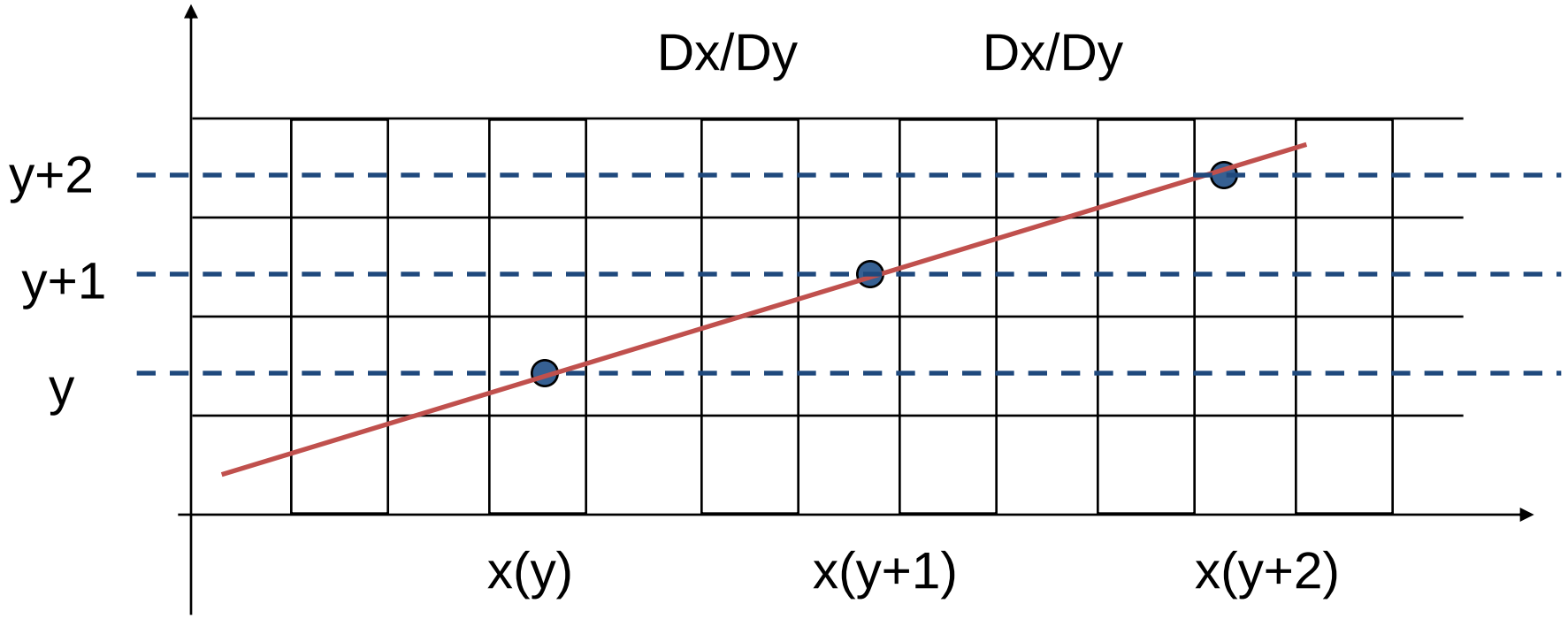
Naív módszer:

```
FillPolygonSlow(q, m)                % q = [q[0],…,q[m-1]]
    for Y = 0 to Ymax do
        scanline = Y
        k=0
        for e=0 to m-1 do
            if scanline a q[e] és q[e+1] között van then
              x[k++]=(q[e],q[e+1]) szakasz és a scanline
              metszéspontja
            endif
        endfor
        x[k] tömb rendezése
        for i=0 to k/2-1 do
              for X=x[2i] to x[2i+1] do
                 Pixel(X,Y,Color(X,Y))
              endfor
        endfor
end
```

Nem hatékony, mert minden pásztánál minden élt figyelünk ☹

Kitöltés gyorsítása

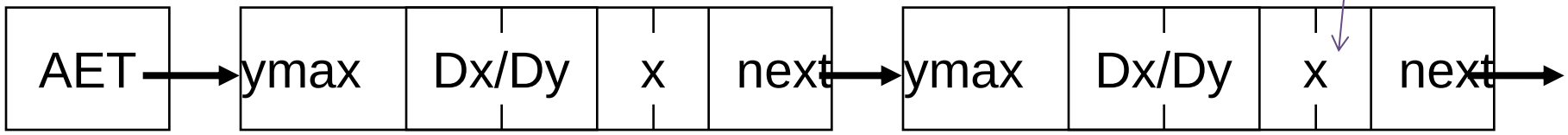
- Vizsgálatok csak az aktív élekre
  - aktív élek frissítése a megelőző pásztából
- Metszéspontszámítás inkrementális elv szerint a megelőző pászta metszéspontjából:



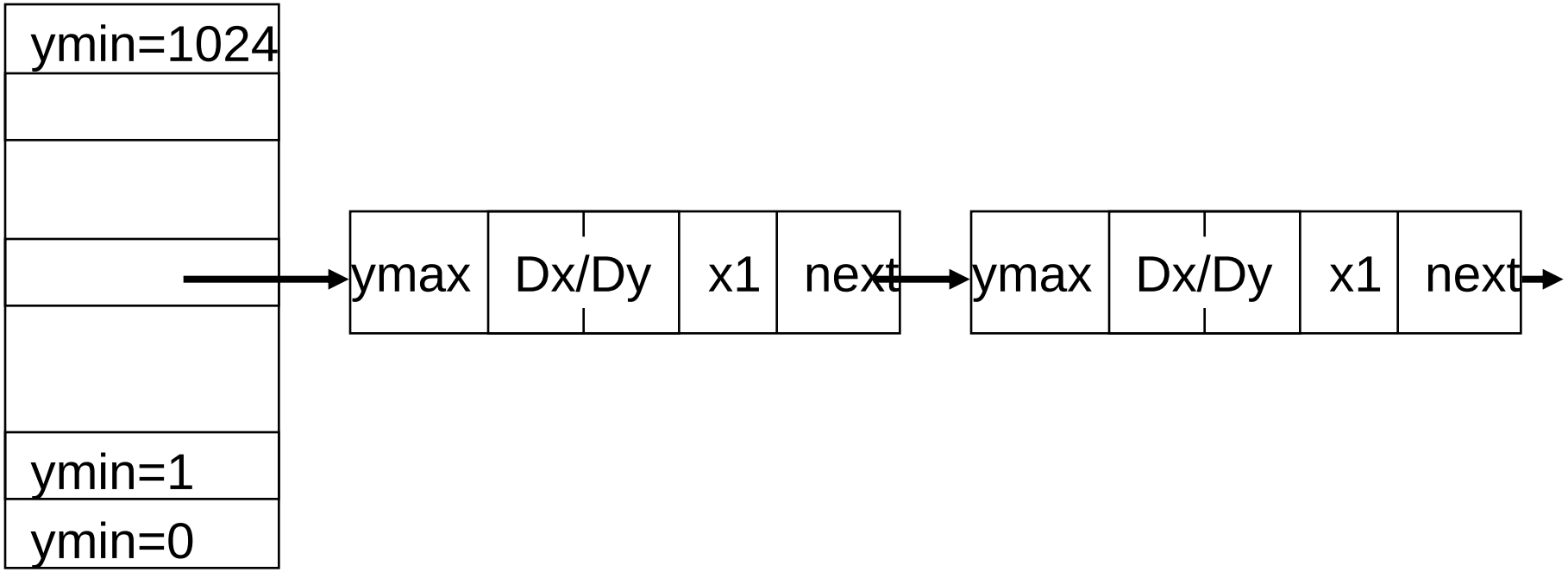
Aktív él lista

Metszéspont x koordinátája az aktuális pásztával

AET: aktív éltábla – az aktuális pásztánál releváns élek



ET éltábla: előzetesen rendezzük valamennyi élet ymin érték szerint



Poligonkitöltés

```
FillPolygonFast(q, m) % q = [q[0],…,q[m-1]]
    for Y = 0 to Ymax do
        for e=0 to m-1 do
        edge = (q[e],q[e+1])
              if ymin(edge) == Y then Put_AET(edge)
        endfor
        for minden edge élre az AET-ben do
               if ymax(edge) >= Y then Delete_AET(edge)
        endfor
        Resort_AET
        for minden második l élre az AET-ben do
              for X=round(x[l]) to round(x[l+1]) to
                    SetPixelColor(X,Y,Color(X,Y))
              endfor
        endfor
        for minden l élre az AET-ben do x[l]+=dx/dy
    endfor
end
```