Algoritmusok raszteres grafikához

Egyenes rajzolása

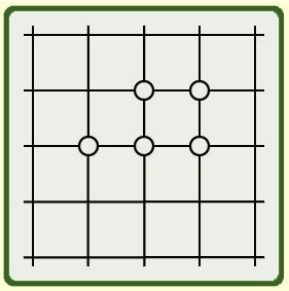
Kör rajzolása

Ellipszis rajzolása

Algoritmusok raszteres grafikához

Feladat:
Grafikai primitíveket (pl. vonalat, síkidomot) ábrázolni kép-mátrixszal, meghatározni azokat a képpontokat, amelyek a primitív pontjai, vagy közel vannak a primitívhez.

Modell:



Képpont (= körlap), amely a négyzetháló csúcspontjaiban helyezhető el.
A koordináták: egész számok

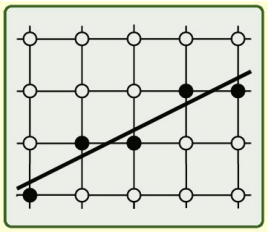
A raszteres grafika egyik kulcsproblémája, hogy a folytonos geometriai primitíveket egy képmátrixon kell megjeleníteni, vagyis csak bizonyos pontokat tudunk kirajzolni.
Az ábrán a képpontok a rács metszéspontjaira esnek, ezeket kör jelöli.
A rácspontok egész koordinátájú pontok.

Egyenes rajzolása

Tegyük fel, hogy egy "vékony" egyenes leírható az alábbi képlettel:

$$y = m \cdot x + b$$

A meredeksége: $0 < m < 1$
($m = 0, 1$ triviális, más esetekben pedig visszavezetjük
$0 < m < 1$-re)



Legyen a vonal két végpontja (x_0, y_0) és (x_1, y_1) úgy hogy $x_0 < x_1$ és $y_0 < y_1$

Tegyük fel, hogy a kirajzolandó egyenes leírható a $y = m \cdot x + b$ képlettel, ahol a $0 < m < 1$. Ez nem túl nagy megkötés viszont hasznos lesz a továbbiakban.
Vegyük észre, hogy ha

- $m = 0$, akkor vízszintes vonalat rajzolunk, vagyis csak x változik x_0-tól x_1-ig
- $m = 1$, akkor egy 45°-os egyenest rajzolunk.

Minden más esetet visszavezethetünk erre például a koordináták felcserélésével.
Az (x_0, y_0) és (x_1, y_1) közötti pontokat kerekítéssel határozzuk meg a képlet alapján.

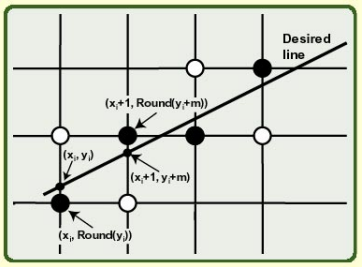
Egyenes rajzolása: Alap inkrementális algoritmus

(x_0, y_0)-t kirajzoljuk, haladjunk Δx növekménnyel balról jobbra és válasszuk a legközelebbi pontot:

$$(x_{i+1}, [y_{i+1} + 0,5]) = (x_{i+1}, [m \cdot x_{i+1} + b + 0,5])$$

A szorzás kiküszöbölhető inkrementálással:

$$y_{i+1} = m \cdot x_{i+1} + b = m \cdot (x_i + \Delta x) + b = \underbrace{y_i}_{m \cdot x_i + b} + m \cdot \Delta x = y_i + m$$



Az alap inkrementális algoritmus kihasználja azt a feltételt, hogy $x_0 < x_1$, valamint, hogy balról jobbra haladva x-et mindig csak 1-gyel kell növelni. Az egyenlet alapján minden x értékhez meg tudjuk mondani a hozzájuk tartozó y értéket, viszont ezt még kerekíteni kell.
Az egyenes egyenletében az $m \cdot x$ szorzást helyettesíthetjük inkrementálással. Nézzük meg, hogy a dia második egyenletében milyen összefüggés van y_{i+1} és y_i között! Azt vesszük észre, hogy y_{i+1} kiszámolható az y_i és m ismeretében, méghozzá elegendő egy összeadást elvégezni. Vagyis a következő y koordinátát úgy kapjuk, hogy az előzőhöz hozzáadunk m-et és ezt kerekítjük. (Itt már csak y-t kell kerekíteni, x mindig egész szám lesz.)
Jelentős számításigényt spórulunk azzal, hogy a szorzás helyett inkrementálást hajtunk végre.

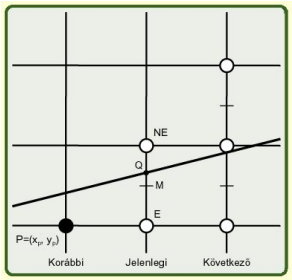
Egyenes rajzolása: Alap inkrementális algoritmus

Algoritmus: (ha $|m| > 1$, akkor x-et cseréljük ki y-nal)

```
void Line(int x0, int y0, int x1, int y1) {
   int x;
   double dx, dy, m, y;
   dy = y1-y0;
   dx = x1-x0;
   m = dy / dx;
   y = y0;
   for (x=x0; x < x1; x++) {
     WritePixel(x, Round(y), value); // pontot rajzol
     y += m;
   }
} // END Line
```

Egyenes rajzolása: Felezőpont algoritmus

Egész aritmetika elegendő (Bresenham)



Elv: Azt a képpontot válasszuk NE és E közül, amelyik a Q metszésponthoz közelebb van.

Másképpen: A választásban az döntsön, hogy Q az M **felezőpont** melyik oldalán van.

Tegyük fel, hogy $x_0 < x_1$ és $y_0 < y_1$

A felezőpont algoritmus az kerekítést fogja kiküszöbölni azzal, hogy megvizsgálja, hogy az aktuális x-kooridinátához tartozó függőleges rácsegyenest hol metszi a kirajzolandó egyenes. Ennek a két egyenesnek a metszéspontját a Q pont jelöli. A lehetséges y-kooridináták csak a $P(x_p, y_p)$ E és NE szomszédjainak y-koordinátái lehetnek. Hogy ezek közül melyiket válasszuk döntse el az, hogy a Q metszéspont melyikhez van közelebb.

Egyenes rajzolása: Felezőpont algoritmus

Az (x_0, y_0) és (x_1, y_1) pontokon áthaladó egyenes egyenlete:

$$\frac{(x - x_0)}{(x_1 - x_0)} = \frac{(y - y_0)}{(y_1 - y_0)}$$

innen átrendezve: $(x - x_0)(y_1 - y_0) = (y - y_0)(x_1 - x_0)$

Legyen $dx = (x_1 - x_0)$ és $dy = (y_1 - y_0)$, ekkor

$$(x - x_0) \cdot dy = (y - y_0) \cdot dx,$$

amelyből: $x \cdot dy - x_0 \cdot dy - y \cdot dx + y_0 \cdot dx = 0$

Legyen

$$F(x, y) = x \cdot dy - x_0 \cdot dy - y \cdot dx + y_0 \cdot dx = 0$$

Világos, hogy

$$F(x, y) \begin{cases} > 0, & \text{az egyenes az } (x, y) \text{ felett fut} \\ = 0, & \text{az egyenes áthalad az } (x, y) \text{ ponton} \\ < 0, & \text{az egyenes az } (x, y) \text{ alatt fut} \end{cases}$$

Egyenes rajzolása: Felezőpont algoritmus

$$\boxed{F(x,y) = x \cdot dy - x_0 \cdot dy - y \cdot dx + y_0 \cdot dx = 0}$$

(x_p, y_p) rajzolása után **felezőpont kritérium**:

$$d = F(M) = F(x_p{+}1, y_p{+}\tfrac{1}{2}) \begin{cases} > 0, & M \text{ felett fut} & \rightarrow \textbf{NE} \\ = 0, & \text{áthalad } M\text{-en} & \rightarrow \textbf{NE} \text{ vagy } \underline{\textbf{E}} \\ < 0, & M \text{ alatt fut} & \rightarrow \textbf{E} \end{cases}$$

d: a döntési változó
d változása a következő pontnál:

- ha előzőleg E-t választottuk, akkor
  $\Delta_E = d_{\text{új}} - d_{\text{régi}} = F(x_p + 2, y_p + \frac{1}{2}) – F(x_p + 1, y_p + \frac{1}{2}) = dy,$
- ha előzőleg NE-t választottuk, akkor
  $\Delta_{NE} = d_{\text{új}} - d_{\text{régi}} = F(x_p + 2, y_p + \frac{3}{2}) – F(x_p + 1, y_p + \frac{1}{2}) = dy - dx,$

A bekeretezett formula emlékeztetőként szolgál az előző diáról.
A d döntési változót úgy kapjuk, hogy az M felezőpont koordinátáit behelyettesítjük az egyenes egyenletébe.

- Ha a $d > 0$, akkor az M metszéspont felett halad az egyenes, és az NE szomszédot kell választani.
- Ha $d = 0$, akkor pontosan az M metszésponton halad keresztül az egyenes, és ekkor akár az NE, akár az E szomszédot választhatjuk, megállapodás szerint azonban E-t választjuk.
- Ha $d < 0$, akkor az egyenes az M metszéspont alatt halad el, így az E szomszédot választjuk.

Van két további segédváltozó, amelyet figyelembe fogunk venni az algoritmusnál. Attól függően, hogy az előző lépésben az NE vagy az E szomszédot választottuk, a következő lépésben a Δ_{NE} és a Δ_E változó értékével fogjuk növelni a d döntési változó értékét. Tehát a d döntési változó értékét elég egyetlen egyszer kiszámolni, továbbá ki kell számolni a Δ_{NE} és Δ_E változó értékét is, és ezekkel az értékekkel fogjuk növelni a d értékét minden egyes pont meghatározása során.

Egyenes rajzolása: Felezőpont algoritmus

$$\boxed{F(x,y) = x \cdot dy - x_0 \cdot dy - y \cdot dx + y_0 \cdot dx = 0}$$

Kezdés:

$$d_{start} = F(x_0 + 1, y_0 + \tfrac{1}{2}) = F(x_0, y_0) + dy - \tfrac{d_x}{2} = dy - \tfrac{dx}{2}$$

Azért, hogy egész aritmetikával számolhassunk, használjuk inkább a

$$2 \cdot F(x,y) = 2 \cdot (x \cdot dy – y \cdot dx + y_0 \cdot dx – x_0 \cdot dy)$$

függvényt, ennek előjele megegyezik F előjelével, és ekkor

$$d_{start} = 2 \cdot dy – dx$$

már egész szám.

Már csak azt kell kiszámolnunk, hogy az első lépéskor az NE vagy az E szomszédját válasszuk az (x_0, y_0) pontnak. Itt is a felezőpont koordinátáit helyettesítjük be az egyenes egyenletébe, és ez alapján döntünk.
Mivel a döntési változó nem egész szám (és a célunk az, hogy egész aritmetikával számoljunk), ezért lehet venni a döntési változó pozitív egész számú többszörösét ez ugyanis nem befolyásolja az előjelet.

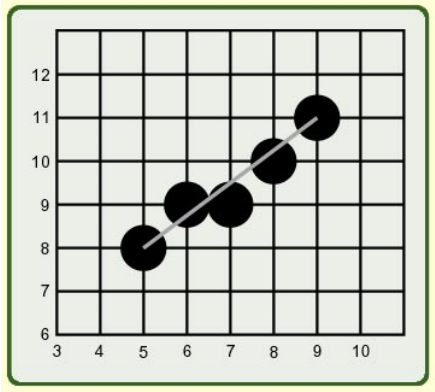
Egyenes rajzolása: Felezőpont algoritmus

```
void MidpointLine(int x0, int y0,
                  int x1, int y1, int value){
  int dx, dy, incrE, incrNE, d, y, x;
  dx = x1-x0;   dy = y1-y0;
  d = 2*dy-dx;   incrE = 2*dy;   incrNE = 2*(dy - dx);
  x = x0;   y = y0;
  WritePixel(x, y, value);
  while(x < x1) {
    if ( d <= 0 ) { // E-t valasztjuk
      x++;   d += incrE;
    } else { // NE-t valasztjuk
      x++; y++;   d += incrNE;
    }
    WritePixel(x, y, value);
  } // END while
} // END Midpoint
```


A felezőpont algoritmus egyenesre. Néhány utasítást egy sorba írtunk, hogy kiférjen egy diára.

Egyenes rajzolása: Felezőpont algoritmus

Eredmény: például



Tulajdonságok:

- csak összeadás és kivonás
- általánosítható ellipszisre és körre

Egyenes rajzolása

Megjegyzés:
Nem mindig lehet csak balról jobbra haladva rajzolni az egyeneseket.
Például szaggatott vonallal zárt poligon

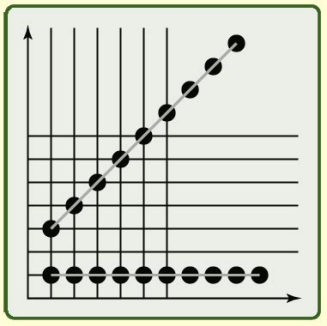
Problémák:
Különböző pontsorozat lesz az eredmény, ha balról jobbra vagy jobbról balra haladva rajzolunk.

Legyen a választás:

- balról jobbra: $d = 0 \rightarrow$ E-t választani
- jobbról balra: $d = 0 \rightarrow$ SW-t választani

Egyenes rajzolása

Problémák:
A vonal pontjainak sűrűsége függ a meredekségtől.



Megoldás:

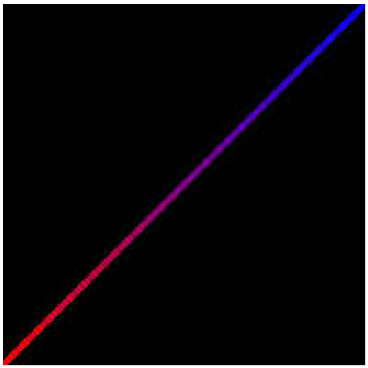
- intenzitás változtatása
- kitöltött téglalapnak kell tekinteni az egyenes pontjait

OpenGL

Egyenes szakasz rajzolása

Program: Szakaszrajzoló I.

Rajzoljunk egy 5 pixel vastagságú egyenest, melynek egyik végpontja piros, a másik végpontja kék!



Program: Szakaszrajzoló II.

```
void display() {
  glClear(GL_COLOR_BUFFER_BIT);
  glLineWidth(5.0); // 5 pixel vastag vonal
  glShadeModel(GL_SMOOTH);
  glBegin(GL_LINES);
    glColor3d(1.0, 0.0, 0.0); //A piros vegpont
    glVertex2d(0.0,0.0);
    glColor3d(0.0, 0.0, 1.0); // A kek végpont
    glVertex2d(200.0, 200.0);
  glEnd();
  glFlush();
}
```

Program: Szakaszrajzoló III.

```
int main(int argc, char* argv[]) {
  glutInit(&argc, argv);
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(200,200);
  glutInitWindowPosition(100,100);
  glutCreateWindow(''szakasz'');
  init();
  glutDisplayFunc(display);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```


Megjegyezzük, hogy csak a `display()` függvény változott a korábbi pontrajzolóhoz képest.

Program

Megjegyzés:
`glShadeModel(GL_SMOOTH)`

- `GL_SMOOTH`: a két végpont között a hozzájuk megadott színekkel interpolál
- `GL_FLAT`: az utolsó végpont színével rajzol
  (`GL_POLYGON` esetén az elsőével)

Algoritmusok raszteres grafikához

Kör rajzolása

Kör rajzolása

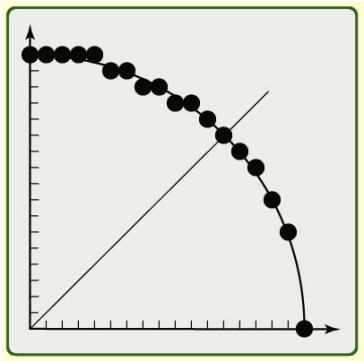
A kör egyenlete: $x^2 + y^2 - R^2 = 0$, R egész szám

Elég egy kör negyedet/nyolcadot megrajzolni, a többi rész a szimmetria alapján transzformációkkal (pl. tükrözés) előáll.

x 0-tól R-ig növekszik
$y = \sqrt{R^2 - x^2}$

Drága eljárás
(szorzás, gyökvonás)

Nem egyenletes



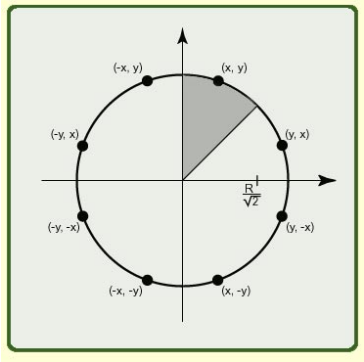
Kör rajzolása

Polárkoordinátás alak:
Most is elég egy nyolcadot kiszámítani.

$x = R \cdot \cos\theta$
$y = R \cdot \sin\theta$

θ 0°-tól 90°-ig növekszik

Drága eljárás
(sin, cos)



Program: nyolcadkör

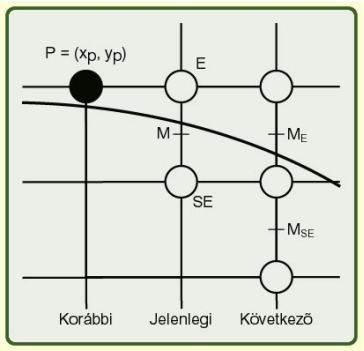
Egyszerre 8 pontot helyezünk el.

```
void Circlepoints(int x, int y, int value) {
   WritePixel( x, y, value)
   WritePixel( x, -y, value)
   WritePixel(-x, y, value)
   WritePixel(-x, -y, value)
   WritePixel( y, x, value)
   WritePixel(-y, x, value)
   WritePixel( y, -x, value)
   WritePixel(-y, -x, value)
} // END Circlepoints
```

Kör rajzolása

Felezőpont algoritmus körre:
x 0-tól $\frac{R}{\sqrt{2}}$-ig, (amíg $x \leq y$)

Elv:
E és SE közül azt a pontot választjuk, amelyikhez a körív metszéspontja közelebb van.



A felezőpont algoritmus általánosítható körre is. Itt elegendő csupán egy nyolcadot meghatározni, a többi pont a szimmetriából adódik. A körív felső nyolcadrésze pont $\frac{R}{\sqrt{2}}$-nál ér véget.
A kör rajzolását mindig az $(0, R)$ pontban kezdjük.

Felezőpont algoritmus körre

$$F(x,y) = x^2 + y^2 - R^2 \begin{cases} > 0, & \text{az } (x,y) \text{ a körön kívül van} \\ = 0, & \text{az } (x,y) \text{ a köríven van} \\ < 0, & \text{az } (x,y) \text{ a köríven belül van.} \end{cases}$$

$$d = F(M) = F(x_p{+}1, y_p{-}\tfrac{1}{2}) \begin{cases} > 0, & \rightarrow \text{ SE-t kell választani} \\ = 0, & \rightarrow \text{ SE-t vagy E-t választjuk} \\ < 0, & \rightarrow \text{ E-t kell választani.} \end{cases}$$

A felezőpont algoritmus kör esetében hasonlóan működik, mint egyenes esetében. Az aktuális $P = (p_x, p_y)$ pont után megnézzük, hogy a következő M felezőpontot behelyettesítve a kör egyenletébe pozítív, 0 vagy negatív értéket kapunk. Ennek megfelelően rendre SE-t, SE-t vagy E-t, illetve E-t kell választani.

Kör rajzolása: Felezőpont algoritmus

$$\boxed{F(x,y) = x^2 + y^2 - R^2 = 0}$$

d változása a következő pontnál:

- ha előzőleg E-t választottuk:

$$\begin{aligned}\Delta_E = d_{\text{új}} - d_{\text{régi}} &= F(x_p + 2, y_p - \tfrac{1}{2}) - F(x_p + 1, y_p - \tfrac{1}{2}) \\ &= 2 \cdot x_p + 3\end{aligned}$$

- ha előzőleg SE-t választottuk:

$$\begin{aligned}\Delta_{SE} = d_{\text{új}} - d_{\text{régi}} &= F(x_p + 2, y_p - \tfrac{3}{2}) - F(x_p + 1, y_p - \tfrac{1}{2}) \\ &= 2 \cdot x_p - 2 \cdot y_p + 5\end{aligned}$$

Kör rajzolása: Felezőpont algoritmus körre

Az iterációs lépések:

1. a döntési változó előjele alapján választjuk a következő képpontot
2. $d = d + \Delta_{SE}$ vagy $d + \Delta_E$ (a választástól függően).

Figyeljük meg, hogy d egész számmal változik.

Kezdés:

- kezdőpont: $(0, R)$
- felezőpont: $(1, R - \frac{1}{2})$
- $d = F(1, R - \frac{1}{2}) = \frac{5}{4} - R$ **nem egész szám!**

Kör rajzolása: Felezőpont algoritmus körre

Nem tudunk egész aritmetikát használni, ezért legyen h új döntési változó:

$$h = d - \frac{1}{4}$$

Ekkor kezdéskor

$$h = \frac{5}{4} - R - \frac{1}{4} = 1 - R$$

Kezdetben és a későbbiek során h egész szám

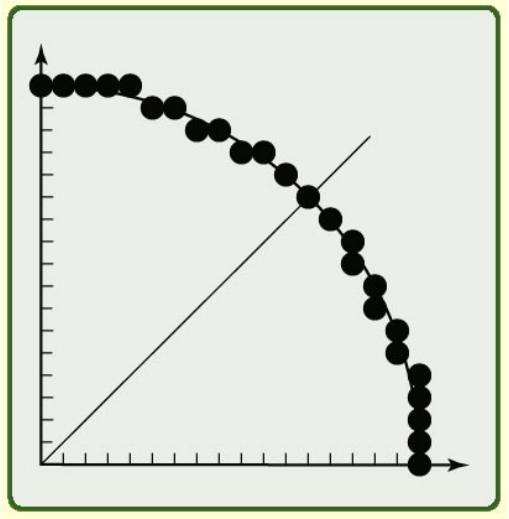
Igaz, hogy $d < 0$ helyett $h < -\frac{1}{4}$-et kellene vizsgálni, de h egész volta miatt ez ekvivalens $h < 0$-val, tehát egész aritmetika használható

Megjegyzés: F helyett $4F$-fel is dolgozhatnánk.

Kör rajzolása: Felezőpont algoritmus körre

```
void MidpointCircle(int R, int value) {
   int h;
   x=0; y = R; h = 1-R
   CirlcePoints(x,y,value);
   while(y >= x) {
     if (h < 0) {
       x++; h+=2*x+3;
     } else {
       x++; y--;
       h += 2*(x-y)+5
     } // END if-else
     CirclePoints(x,y,value);
   } // END while
}
```

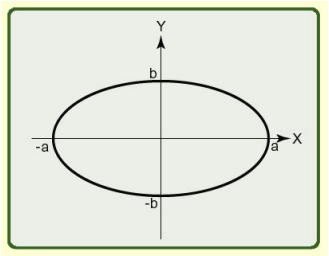
Kör rajzolása: Felezőpont algoritmus körre



Algoritmusok raszteres grafikához

Ellipszis rajzolása

Ellipszis rajzolása



$$a^2 + b^2 = 1$$

$$b^2x^2 + a^2y^2 - a^2b^2 = 0$$

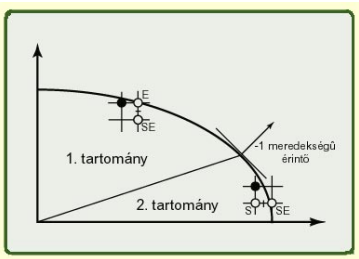
a és b egész számok

$$F(x, y) = b^2x^2 + a^2y^2 - a^2b^2$$

A szimmetria miatt elég az első síknegyeben megrajzolni.

Ellipszis rajzolása

Da Silva algoritmusa: (felezőpont algoritmus)
Bontsuk a negyedeket két tartományra:



Az 1. tartományban $a^2(y_p - \frac{1}{2}) > b^2(x_p + 1)$

Ellipszis rajzolása: Da Silva algoritmusa

Az 1. tartományban:

$$d_1 = F(x_p + 1, y_p - \tfrac{1}{2}) \begin{cases} >= 0, & E\text{-t választjuk} \\ < 0, & SE\text{-t választjuk} \end{cases}$$

Az 1. tartományban d változása:

$$\begin{aligned} \Delta_E = & \quad d_{\text{új}} - d_{\text{régi}} = F(x_p + 2, y_p - \tfrac{1}{2}) - F(x_p + 1, y_p - \tfrac{1}{2}) \\ = & \quad b^2(2 \cdot x_p + 3) \end{aligned}$$

$$\begin{aligned} \Delta_{SE} = & \quad d_{\text{új}} - d_{\text{régi}} = F(x_p + 2, y_p - \tfrac{3}{2}) - F(x_p + 1, y_p - \tfrac{1}{2}) \\ = & \quad b^2(2 \cdot x_p + 3) + a^2(-2 \cdot y_p + 2) \end{aligned}$$

Δ_E és Δ_{SE} egész szám.

Ellipszis rajzolása: Da Silva algoritmusa

$$\boxed{F(x,y) = b^2x^2 + a^2y^2 - a^2b^2}$$

Kezdés:

- kezdőpont: $(0, b)$
- felezőpont: $(1, b - \frac{1}{2})$
- $d = F(1, b - \frac{1}{2}) = b^2 + a^2(-b^2 + \frac{1}{4})$

Ha F helyett $4F$-el dolgozunk, akkor egész aritmetikát használhatunk.

Házi feladat
Az algoritmus a 2. tartományban.

Kör razolása: Da Silva algoritmusa

```
void MidpointEllipse(int a, int b, int value)
  int x, y, a2, b2;
  double d1, d2;
  x=0; y=b; a2=a*a; b2=b*b;
  d1 = b2-a2*b+a2/4;
  EllipsePoints(x,y,value);
  ////// Region 1 //////
  while (a2*(y-1/2) > b2*(x+1)) {
    if (d1 < 0) { // E-t valasztottuk
      d1+=b2*(2*x+3); x++;
    } else { // SE-t valasztottuk
      d1+=b2*(2*x+3)+a2*(-2*y+2); x++; y--;
    }
    EllipsePoints(x,y,value);
  } // END Region1
```

folytatódik a következő dián ...

Kör rajzolása: Da Silva algoritmusa

... folytatás

```
   ////// Region2 ///////
   while (y > 0) {
     if (d2 < 0) { // SE-t valasztjuk
       d2+=b2*(2*x+2) +a2*(-2*y+3); x++; y--;
     } else { // S-et valasztjuk
       d2+= a2*(-2*y+3); y--;
     }
     EllipsePoints(x,y,value);
   } // END Region2
} // MidpointEllipse
```

OpenGL

Feladat:
Kör rajzolása felezőpont algoritmussal