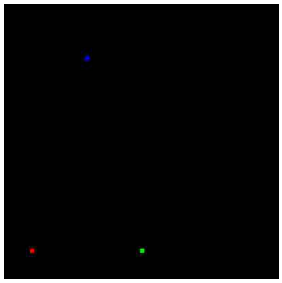
OpenGL

Pontok rajzolása

Pontok rajzolása

Rajzoljunk egy piros pontot a (10, 10), egy zöld pontot az (50, 10) és egy kék pontot a (30, 80) koordinátákba (az ablak 100*100-as méretű)



Színek és színmódok

RGBA színmód:
Minden színt négy komponens definiál: (**R**, **G**, **B**, **A**)
vörös (Red), zöld (Green), kék (Blue),
alfa (Alpha)

Minél nagyobb az **RGB** komponens értéke, annál intenzívebb a színkomponens

A (átlátszóság): 1.0 - nem átlátszó,
0.0 - teljesen átlátszó

Pl.: (0.0, 0.0, 0.0, 0.0) – átlátszó fekete

Törlő szín

```
void glClearColor(
     GLclampf red,
     GLclampf green,
     GLclampf blue,
     GLclampf alpha);
```

Aktuális törlő szín beállítása
Alapértelmezés: `(0, 0, 0, 0)`

`GLclampf - float`

Mátrix mód beállítása

Transzformációk: mátrixokkal definiálva
nézeti (viewing),
modellezési (modelling),
vetítési (projection)

```
void glMatrixMode(enum mode);
```

Ha *mode* == `GL_PROJECTION`, akkor vetítési mátrix
pl.:

```
glMatrixMode(GL_PROJECTION);
```

```
void glLoadIdentity(void);
```

az érvényes mátrix az egységmátrix lesz

Vetítési mátrix megadása (2D)

```
void gluOrtho2D(double left, double
     right, double bottom, double top);
```

a (*left, right, bottom, top*) téglalapban levő objektumok 2D párhuzamos vetítése
pl.:

```
gluOrtho2D(0, 100, 0, 100);
```

Program (pontrajzoló) I

```
#include "stdafx.h" // Visual C++ esetén
      // más fordítónál kicsit másképp
void init(void) {
     glClearColor(0.0,0.0,0.0,0.0);
      // fekete a törlőszín
     glMatrixMode(GL_PROJECTION);
       // vetítés az aktuális mátrix mód
     glLoadIdentity();
       // legyen az egységmátrix
     gluOrtho2D(0,100,0,100);
       // párhuzamos vetítés specifikálása
}
```

Pufferek törlése

```
void glClear(GLbitfield mask);
```

Pufferek tartalmának a törlése

A pufferek:

`GL_COLOR_BUFFER_BIT`,
`GL_DEPTH_BUFFER_BIT`,
`GL_STENCIL_BUFFER_BIT` vagy
`GL_ACCUM_BUFFER_BIT`

Pl. a szín puffer törlése az aktuális törlőszínnel:

```
glClear(GL_COLOR_BUFFER_BIT);
```

Objektumok megadása

```
void glBegin(enum mode);
…
void glEnd(void);
```

geometriai objektumok specifikációja

mode értéke lehet pl.
`POINTS, LINES, POLYGON`

Színbeállítás

```
void glColor{34}{bsifd ubusui}
     (T components);
```

Színbeállítás csúcspontokhoz van hozzárendelve

Pl.

```
glColor3f(1.0,0.0,0.0);    // piros
glColor3f(0.0,1.0,0.0);    // zöld
glColor3f(0.0,0.0,1.0);    // kék
```

Csúcspontok megadása

```
void glVertex{234}{sifd}( T coords );
```

Csúcspont(ok) (vertex) megadása

Pl.:

```
glVertex2i(10,10);
     // a pont koordinátája (10, 10)
```

Program (pontrajzoló) II

```
void display(void) {
     glClear(GL_COLOR_BUFFER_BIT);
               // képernyő tötlés
     glBegin(GL_POINTS);
               // pontokat specifikálunk
          glColor3f(1.0,0.0,0.0);  // piros
          glVertex2i(10,10); // piros pont
          glColor3f(0.0,1.0,0.0);  // zöld
          glVertex2i(50,10);  // zöld pont
          glColor3f(0.0,0.0,1.0);  // kék
          glVertex2i(30,80);  // kék pont
     glEnd();       // több pont nem lesz
     glFlush();     // rajzolj!
}
```

Program (pontrajzoló) III

```
void keyboard
  (unsigned char key, int x, int y){
    switch(key) {  // billentyű kezelés
      case 27:     // ha escape
      exit(0);     // kilép a programból
      break;
    }
}
```

Képernyő mód

```
void glutInitDisplayMode
     (unsigned int mode);
```

A képernyő módot definiálja

Pl. ha ***mode***

```
GLUT_SINGLE | GLUT_RGB
```

akkor az ún. *egyszeresen pufferelt*, ***RGB*** *módban* specifikál ablakot

Ablak

```
void glutInitWindowSize
     (int width, int height);
```

Az ablak méretét definiálja pixelekben

```
void glutInitWindowPosition(int x, int y);
```

Az ablak bal felső sarkának pozíciója

```
int glutCreateWindow(char *name);
```

Megnyit egy ablakot az előző rutinokban specifikált jellemzőkkel. Ha az ablakozó rendszer lehetővé teszi, akkor ***name*** megjelenik az ablak fejlécén. A visszatérési érték egy egész, amely az ablak azonosítója.

Callback függvények

```
void glutDisplayFunc(void(*func)(void));
```

Azt a ***callback*** függvényt specifikálja, amelyet akkor kell meghívni, ha az ablak tartalmát újra akarjuk rajzoltatni. Pl.:

```
glutDisplayFunc(display);
```

```
void glutKeyboardFunc(void(*func)
     (unsigned char key, int x, int y);
```

Azt a ***callback*** függvényt specifikálja, melyet egy billentyű lenyomásakor kell meghívni. ***key*** egy **ASCII** karakter. Az ***x*** és ***y*** paraméterek az egér pozícióját jelzik a billentyű lenyomásakor (ablak relatív koordinátákban). Pl.:

```
glutKeyboardFunc(keyboard);
```

Program (pontrajzoló) IV

```
int APIENTRY WinMain(HINSTANCE hInstance,
                     HINSTANCE hPrevInstance,
                     LPSTR     lpCmdLine,
                     int       nCmdShow) {
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
        //az ablak egyszeresen pufferelt,és RGB módú
    glutInitWindowSize(100, 100); // 100x100-as
    glutInitWindowPosition(100, 100);
        // az ablak bal felső sarkának koordinátája
    glutCreateWindow("3point");  // neve 3point
    init();                      // inicializálás
    glutDisplayFunc(display);
        // a képernyő események kezelése
    glutKeyboardFunc(keyboard);
        // billentyűzet események kezelése
    glutMainLoop()   // belépés az esemény hurokba...
    return 0;
}
```

ALGORITMUSOK RASZTERES GRAFIKÁHOZ

Egyenes rajzolása

Kör rajzolása

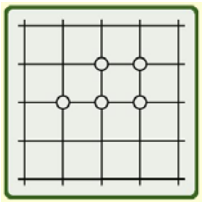
Ellipszis rajzolása

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

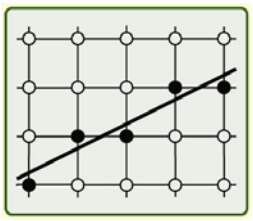
Modell:



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

Egyenes rajzolása

Tegyük fel, hogy "vékony" egyenes: $y = mx + b$
meredeksége: $0 < m < 1$
($m = 0,1,...$ triviális speciális esetek)
más esetekben visszavezetjük $0 < m < 1$ -re



Legyen: $x_0 < x_1$, $y_0 < y_1$

Egyenes rajzolása

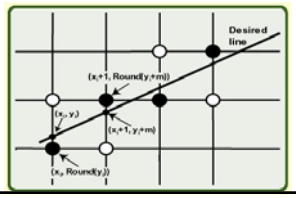
1. Alap inkrementális algoritmus

Haladjunk $\Delta x = 1$ növekménnyel balról jobbra, válasszuk a legközelebbi képpontot:

$(x_i, [y_i+0.5]) = (x_i, [mx_i+b+0.5])$

A szorzás kiküszöbölhető inkrementálással:

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



Alap inkrementális algoritmus

Algoritmus:

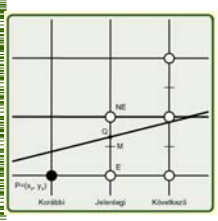
(ha $|m|>1$, akkor x-et cseréljük y-nal)

```
procedure Line(x0,y0, x1,y1, value: integer);
var x : integer;
  dy,dx,y,m : real;
begin
  dy := y1-y0; dx := x1-x0;
  m := dy/dx; y := y0;
  for x := x0 to x1 do
     begin
       WritePixel(x, Round(y), value);
       y := y+m
     end
end; {Line}
```

Egyenes rajzolása

2. Felezőpont algoritmus egyenesre

egész aritmetika elegendő (Bresenham)



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

Másképp: az döntsön a választásban, hogy *Q* az *M* felezőpont melyik oldalán van.

Tegyük fel, hogy:

$x_0 < x_1$, $y_0 < y_1$

Felezőpont algoritmus egyenesre

Az ($x0, y0$) és ($x1, y1$) ponton átmenő egyenes egyenlete: $(x - x0) / (y - y0) = (x1 - x0) / (y1 - y0)$

Legyen $dx = x1 - x0 \; (> 0)$,

$dy = y1 - y0 \; (> 0)$,

akkor: $x\,dy - y\,dx + y0\,dx - x0\,dy = 0$

Legyen: $F(x,y) = x\,dy - y\,dx + y0\,dx - x0\,dy$

$F(x,y)$ { > 0, ha az egyenes (x, y) fölött fut,
$= 0$, ha (x, y) az egyenesen van,
< 0, ha az egyenes (x, y) alatt fut.

Felezőpont algoritmus egyenesre

$F(x,y) = x\,dy - y\,dx + y0\,dx - x0\,dy$

Felezőpont kritérium:

$d = F(M) = F(x_p+1, y_p+½)$ (*d* : döntési változó)

az egyenes		választás:
> 0,	*M* fölött,	***NE***
= 0,	*M*-en át,	***NE*** vagy ***E***
< 0,	*M* alatt fut	***E***

A következő pontnál:

ha ***E***-t választottuk, akkor

$\Delta_E = d_{új} - d_{régi} = F(x_p+2, y_p+½) - F(x_p+1, y_p+½) = dy,$

ha ***NE***-t választottuk, akkor

$\Delta_{NE} = F(x_p+2, y_p+3/2) - F(x_p+1, y_p+½) = dy - dx$

Felezőpont algoritmus egyenesre

$F(x,y) = x\,dy - y\,dx + y0\,dx - x0\,dy$

Kezdés:

$d_{start} = F(x_0+1, y_0+½) = F(x_0, y_0)+dy - dx/2 = dy - dx/2$

Azért, hogy egész aritmetikával számolhassunk, használjuk inkább az

$F(x,y) = 2\cdot(x\cdot dy - y\cdot dx + y0\cdot dx - x0\cdot dy)$

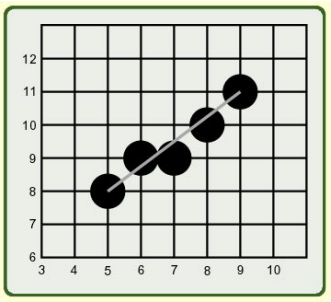
függvényt.

```
procedure MidpointLine
     (x0,y0,x1,y1,value: integer);
var dx,dy,incrE,incrNE,d,x,y : integer;
begin
   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 do  begin
     if d <= 0 then begin
        x := x + 1; d := d + incrE;
     end
     else begin
        x := x + 1; y := y + 1; d := d + incrNE;
     end;
     WritePixel(x,y,value)
   end {while}
end; {MidpointLine}
```

Felezőpont algoritmus egyenesre

Felezőpont algoritmus egyenesre

Eredmény: pl.



Tulajdonságok:

- csak összeadás és kivonás
- általánosítható körre, ellipszisre

Egyenes rajzolása

Megjegyzés:

Nem mindig lehet csak balról jobbra haladva rajzolni az egyeneseket.
Pl. szaggatott vonallal rajzolt zárt poligon

Problémák:

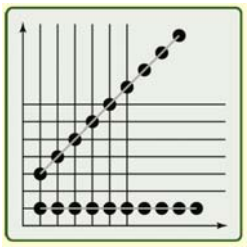
1. Különböző pontsorozat lesz az eredmény, ha balról jobbra, vagy ha jobbról balra haladunk.

Legyen a választás:

balról jobbra: *d = 0* *E*-t választani
jobbról balra: *d = 0* *SW*-t választani

Egyenes rajzolása

2. A vonal pontjainak a sűrűsége függ a meredekségétől



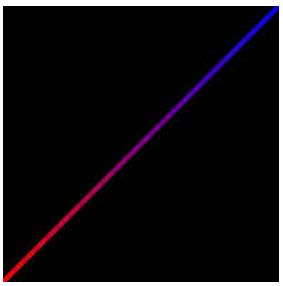
Megoldás: - intenzitás változtatása,
- kitöltött téglalapnak tekinteni az egyenes pontjait

OpenGL

Egyenes szakasz rajzolása

Program (szakaszrajzoló) I

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



Program (szakaszrajzoló) II

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

Program (szakaszrajzoló) III

Megjegyzés:
`glShadeModel (GL_SMOOTH)`

`GL_SMOOTH`: ekkor a két végpont között a hozzájuk megadott színekkel interpolál

`GL_FLAT`: utolsó végpont színével rajzol (`GL_POLYGON` esetében az elsőével)

Program (szakaszrajzoló) IV

```
int APIENTRY WinMain
     (HINSTANCE hInstance,
      HINSTANCE hPrevInstance,
      LPSTR     lpCmdLine,
      int       nCmdShow) {
  glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
  glutInitWindowSize(200,200);
  glutInitWindowPosition(100,100);
  glutCreateWindow("colors");
  init();
  glutDisplayFunc(display);
  glutKeyboardFunc(keyboard);
  glutMainLoop();
  return 0;
}
```

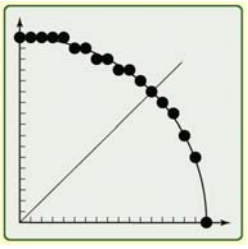
Kör rajzolása

$x^2+y^2 = R^2$ R : egész

1. Elég egy kör-negyedet 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

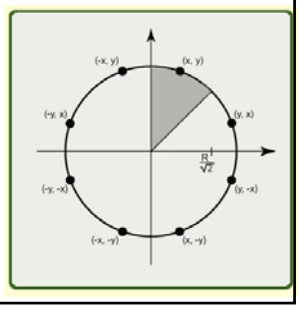
2. Polárkoordinátás alak

Elég egy nyolcad kört kiszámítani:

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

Θ *0°*-tól *90°*-ig növekszik

Drága eljárás (*sin, cos*)



Program (polárkoordinátás alak)

```
procedure Circlepoints(x,y,value: integer);
begin
   WritePixel (x, y, value);
   WritePixel (y, x, value);
   WritePixel (y, -x, value);
   WritePixel (x, -y, value);
   WritePixel (-x, -y, value);
   WritePixel (-y, -x, value);
   WritePixel (-y, x, value);
   WritePixel (-x, y, value);
end; {CirclePoints}
```

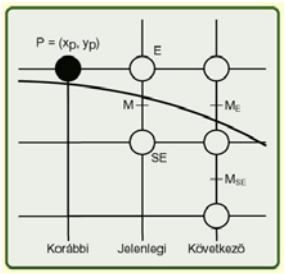
Kör rajzolása

3. Felezőpont algoritmus körre

x *0*-tól $R/\sqrt{2}$-ig (amíg $x \le y$)

Elv:

E és ***SE*** közül azt a pontot választjuk, amelyikhez a körív metszéspontja közelebb van



Felezőpont algoritmus körre

$$F(x,y) = x^2+y^2 - R^2 \begin{cases} > 0, \text{ ha } (x,y) \text{ kívül van,} \\ = 0, \text{ ha } (x,y) \text{ rajta van,} \\ < 0, \text{ ha } (x,y) \text{ belül van.} \end{cases}$$

$d = F(M) = F(x_p+1, y_p - ½)$

- >0 ▶▶▶ ***SE***-t választani
- $=0$ ▶▶▶ ***SE*** vagy ***E***
- <0 ▶▶▶ ***E***-t választani

Felezőpont algoritmus körre

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

A következő pontnál:

ha ***E***-t választottuk, akkor

$$\Delta_E = d_{új} - d_{régi} = F(x_p+2, y_p - ½) - F(x_p+1, y_p - ½) =$$
$$= 2x_p+3$$

ha ***SE***-t választottuk, akkor

$$\Delta_{SE} = F(x_p+2, y_p - 3/2) - F(x_p+1, y_p - ½) =$$
$$= 2x_p - 2y_p+5$$

Felezőpont algoritmus körre

Az iterációs lépések:

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

Figyeljük meg: *d* értéke egész számmal változik!

Kezdés:

kezdőpont: (*0, R*)
felezőpont: (*1, R – 1/2*)
$d = F(1, R - 1/2) = 5/4 - R$

Felezőpont algoritmus körre

```
procedure MidpointCircle (radius, value : integer);
var x, y : integer; d: real;
begin
   x := 0; y := radius; d := 5/4-radius;
   CirclePoints (x,y,value);
   while y>x do
       begin
          if d<0 then
             begin
                 x := x+1; d := d+2*x+3;
             end
          else
             begin
                x := x+1; y := y-1
                d := d+2*(x-y)+5;
             end;
          CirclePoints (x,y,value)
       end {while}
end; {MidpointCircle}
```

Felezőpont algoritmus körre

Nem egész aritmetika, ezért legyen h új döntési változó:

$h = d - ¼$ $\qquad$ $h+¼ = d < 0$

Ekkor kezdéskor

$h = 1 - R$

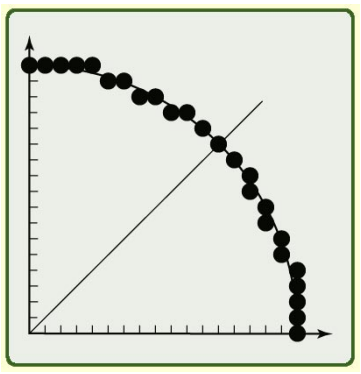
Kezdetben, és a későbbiek során is h egész szám!

Igaz, hogy $d < 0$ helyett $h < -¼$ -et kellene vizsgálni, de ez h egész volta miatt ekvivalens $h < 0$ -val, tehát egész aritmetika használható.

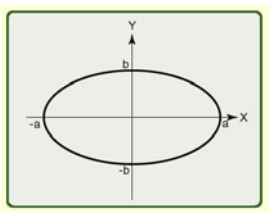
Felezőpont algoritmus körre

```
procedure MidpointCircle (radius, value : integer);
var x,y,d : integer;
begin
   x := 0; y := radius; h := 1-radius;
   CirclePoints (x,y,value);
   while y>x do
       begin
          if h<0 then
            begin
               x := x+1; h := h+2*x+3;
            end
          else
            begin
               x := x+1; y := y-1
               h := h+2*(x-y)+5;
            end;
          CirclePoints (x,y,value)
       end {while}
end; {MidpointCircle}
```

Felezőpont algoritmus körre



Ellipszis rajzolása



$x^2/a^2 + y^2/b^2 = 1$

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

a, b egész

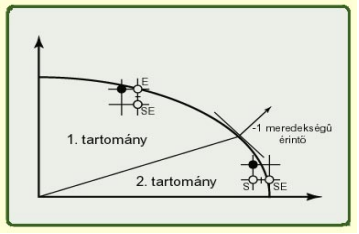
$F(x,y) = b^2x^2 + a^2y^2 - a^2b^2$

Szimmetria miatt:
elég az első síknegyedben megrajzolni

Ellipszis rajzolása

Da Silva algoritmusa (felezőpont módszer)

Bontsuk a negyedet két tartományra:



Az 1. tartományban $\quad a^2 (y_p - ½) > b^2 (x_p + 1)$

Da Silva algoritmusa

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

Az 1. tartományban:

$d_1 = F(x_p+1, y_p - ½)$ { ≥ 0 ***E***-t választjuk ; < 0 ***SE***-t választjuk }

$$\begin{cases} d_{új} - d_{régi} = F(x_p+2, y_p - ½) - F(x_p+1, y_p - ½) \\ d_{új} - d_{régi} = F(x_p+2, y_p - 3/2) - F(x_p+1, y_p - ½) \end{cases}$$

$$\begin{cases} \Delta_E = b^2(2x_p+3) \\ \Delta_{SE} = b^2(2x_p+3) + a^2(-2y_p+2) \end{cases}$$

Da Silva algoritmusa

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

Kezdés:

kezdőpont: $(0, b)$

felezőpont: $(1, b - ½)$

$d = F(1, b - ½) = b^2 + a^2(-b + ¼)$

Házi feladat

Az algoritmus a 2. tartományban

Da Silva algoritmusa

```
procedure MidpointEllipse (a,b,value : integer);
var x,y: integer; d1,d2: real;
begin
     x := 0;  y := b;  a2:=a*a; b2:= b*b;
     d1 := b2-a2b+a2/4;
     EllipsePoints (x,y,value);
     while (a2(y-1/2) > b2(x+1)) do
          begin
               if d1<0 then
                    begin
                         d1 := d1+b2(2x+3); x := x+1
                    end
               else
                    begin
                         d1 := d1+b2(2x+3)+a2(-2y+2);
                         x := x+1; y := y-1
                    end
               EllipsePoints (x,y,value)
          end; {Region1}
     d2 := b2(x+1/2)2+a2(y-1)2-a2b2;
     while (y>0) do
          begin
               if d2<0 then
                    begin
                         d2 := d2+b2(2x+2)+a2(-2y+3);
                         x := x+1; y := y-1
                    end
               else
                    begin
                         d2 := d2+a2(-2y+3); y := y-1
                    end;
               EllipsePoints (x,y,value)
          end (Region2)
end; {MidpointEllipse}
```

OpenGL

Feladat:

Kör rajzolása felezőpont algoritmussal