Bresenham-Gerade: für den ersten Oktanten

Gerade durch A(x1,y1),B(x2,y2): y= kx+d

T oberes Pixel, S unteres Pixel =>
Abstand von S zur wahren Linie: s = y – y_
Abstand von T zur wahren Linie: t = (y_+1)-y
falls s<t, dann S: x_=x_+1, y_ bleibt
falls s>=t, dann T: x_=x_+1, y_=y_+1,
Dabei gilt: s-t = (y-y_)-(y_+1)+y = 2y-2y_-1 = 2(kx_+d)-2y_-1
Sei k= dy/dx mit dx>0 und setzen wir d'= dx (s-t) =>
Entscheidungsvariable sei d'= 2dy x_ - 2dx y_ + dx(2d-1) =>
=> d''= 2 dy (x_+1) – 2dx (y__) +dx(2d-1) =>
d''-d' = 2dy -2dx (y__-y_)
falls S: y__= y_ => d'' = d'+ 2dy
falls T: y__= y_+1 => d'' = d'+ 2dy-2dx
Berechnung für den 1.Punkt: d1 = 2dy-dx

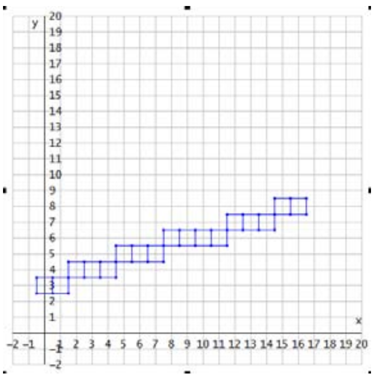
Algorithmus:
x_ = x1, y_ = y1
dx = x2 - x1, dy = y2 – y1
dT = 2 (dy – dx), dS = 2 dy
d_ = 2 dy –dx
zeichne (x_,y_)
solange (x_< x2) {
    x_++
    wenn (d_<0) { d_ = d_+dS }
    sonst { y_++, d_ = d_+ dT }
    zeichne (x_y_)
}

DERIVE:

```
kaestchen(P) := [P + [0.5, 0.5], P + [-0.5, 0.5], P + [-0.5, -0.5], P + [0.5, -0.5], P + [0.5, 0.5]]

bresenham_gerade(A, B, x, y, dx, dy, dT, dS, d, e) :=
  Prog
    x := A↓1
    y := A↓2
    dx := B↓1 - A↓1
    dy := B↓2 - A↓2
    dT := 2·(dy - dx)
    dS := 2·dy
    d := 2·dy - dx
    e := [kaestchen([x, y])]
    Loop
      If x < B↓1
         [x := x + 1, IF(d < 0, d := d + dS, [y := y + 1, d := d + dT]), e := APPEND(e, [kaestchen([x, y])])]
         RETURN e
```

```
A1 := [0, 3]
B1 := [16, 8]
bresenham_gerade(A1, B1)
```



Bresenham- Kreis: (für den zweiten Oktanten)

wir beginnen „oben“ (0,r) und rechnen nach „rechts“.
S Pixel innerhalb des Kreises: x_=x_+1, y_=y_-1 und $B(S) = (x\_+1)^2+(y\_-1)^2-r^2 <0$
T Pixel außerhalb des Kreises: x_=x_+1, y_ bleibt und $B(T) = (x\_+1)^2+y\_^2-r^2 >0$
Entscheidungsvariable sei $d' = B(T)+B(S) = 2(x\_+1)^2+y\_^2+(y\_-1)^2-2r^2$
=> $d'' = 2(x\_+2)^2+y\_\_^2+(y\_\_-1)^2-2r^2$
falls S: $d''-d' = 4(x\_-y\_)+10$
falls T: $d''-d' = 4x\_+6$
Wir berechnen noch den ersten Punkt (0,r):
$d1 = 2(0+1)^2+r^2+(r-1)^2-2r^2 = 3-2r$

Algorithmus:

x_=0, y_= r,
d_ = 3-2r
solange (x_<=y_) {
    zeichne (x_y_)
    wenn (d_<0) { d_= d_+4x_+6 }
    sonst { d_=d_+4(x_-y_)+10, y_-- }
    x_++
}

DERIVE:

```
kaestchen(P) := [P + [0.5, 0.5], P + [-0.5, 0.5], P + [-0.5, -0.5], P + [0.5, -0.5], P + [0.5, 0.5]]

bresenham_kreis(r, x_, y_, d_, e_) :=
  Prog
    x_ := 0
    y_ := r
    d_ := 3 - 2·r
    e_ := []
    Loop
      If x_ ≤ y_
         [e_ := APPEND(e_, [kaestchen([x_, y_])]), IF(d_ < 0, d_ := d_ + 4·x_ + 6, [d_ := d_ + 4·(x_ - y_) + 10, y_ := y_ - 1]), x_ := x_ + 1]
         RETURN e_
```

bresenham_kreis(15)