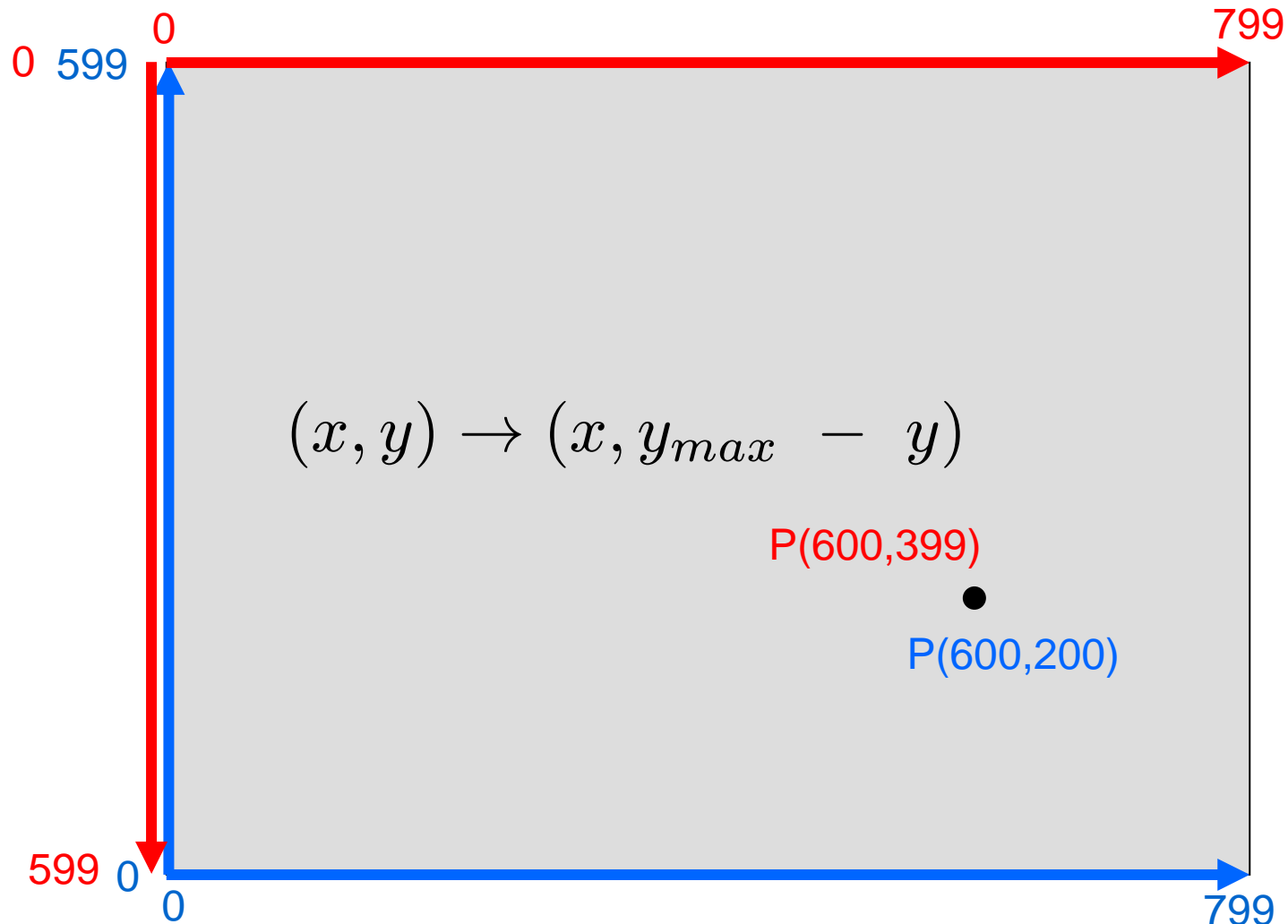


Computergrafik SS 2008

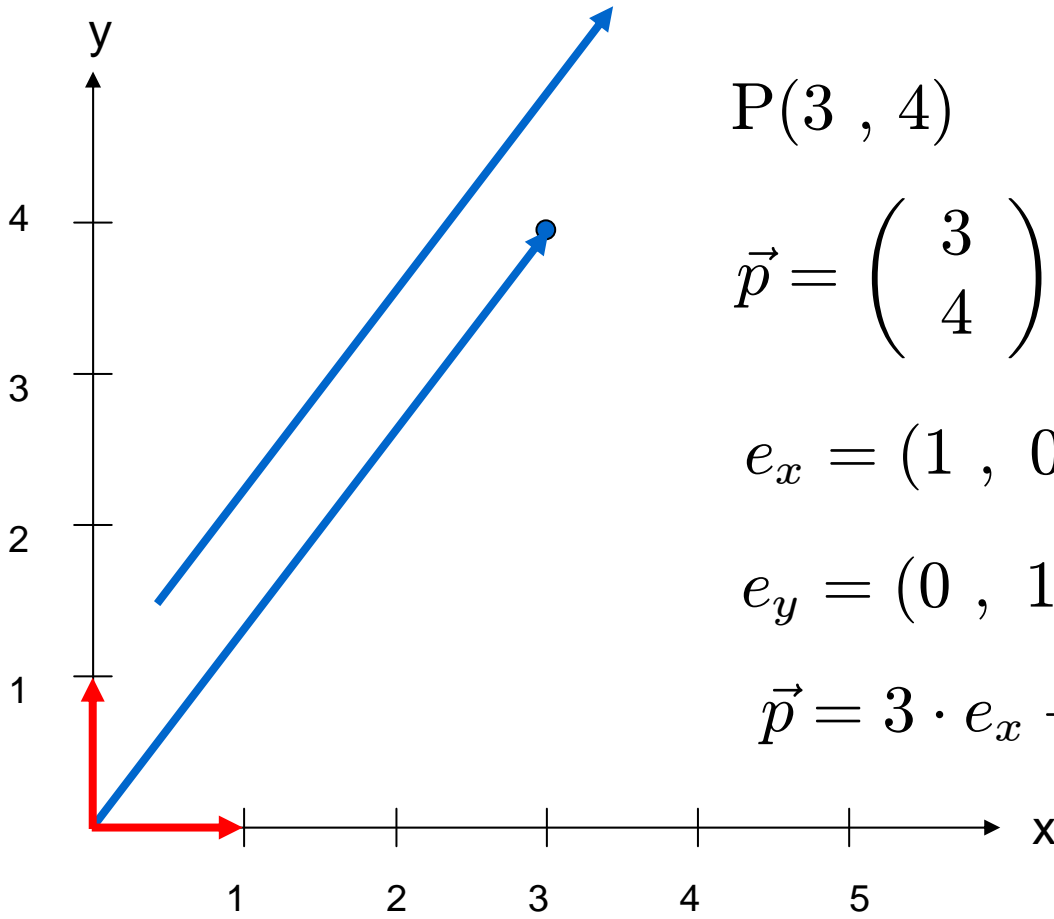
Oliver Vornberger

Kapitel 3: 2D-Grundlagen

Koordinatensysteme



Punkt + Vektor



$$P(3, 4)$$

$$\vec{p} = \begin{pmatrix} 3 \\ 4 \end{pmatrix} = (3, 4)^T$$

$$e_x = (1, 0)^T$$

$$e_y = (0, 1)^T$$

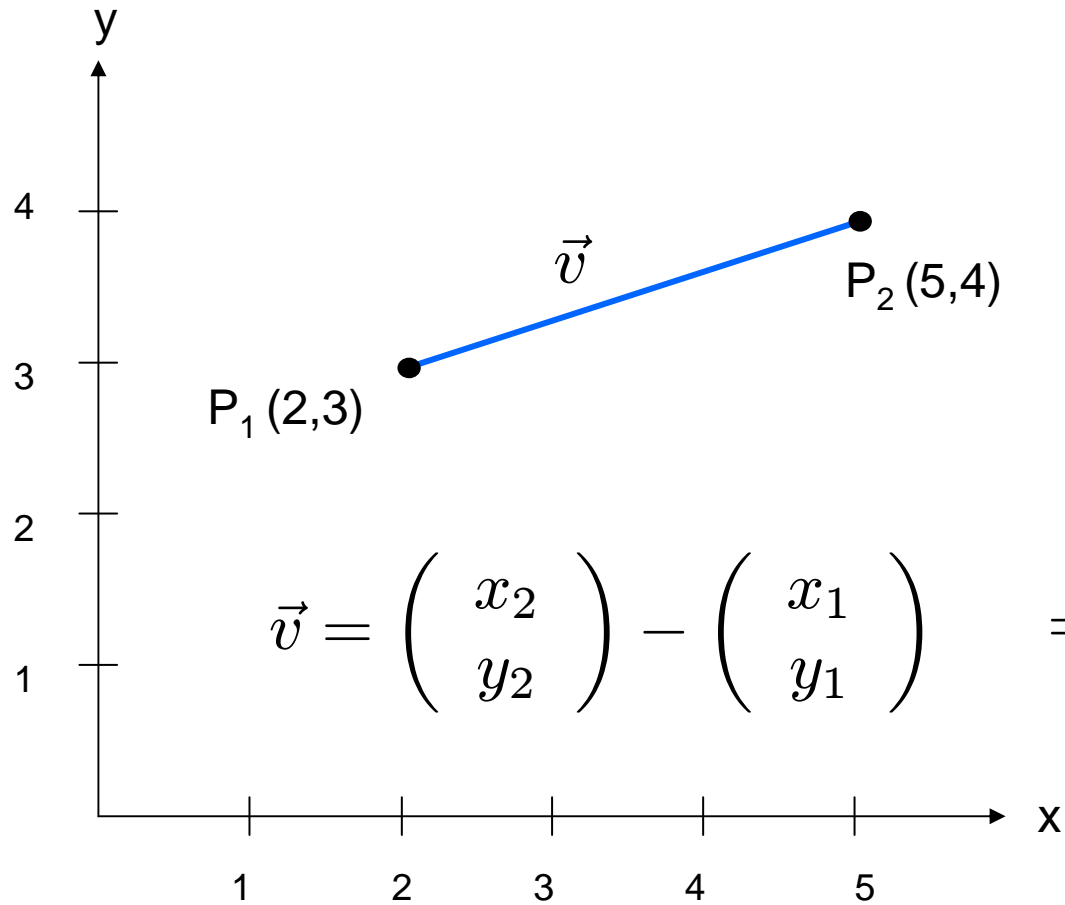
$$\vec{p} = 3 \cdot e_x + 4 \cdot e_y$$

setPixel(int x, int y)

```
setPixel(3,4);
```

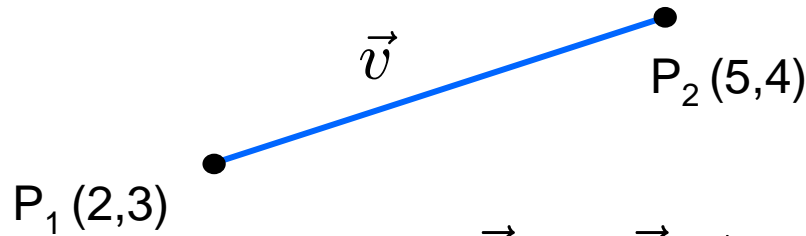
```
setPixel((int)(x+0.5),(int)(y+0.5));
```

Linie



$$\vec{v} = \begin{pmatrix} x_2 \\ y_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} = \begin{pmatrix} x_2 - x_1 \\ y_2 - y_1 \end{pmatrix}$$

Parametrisierte Gradengleichung



$$g : \vec{u} = \vec{p}_1 + r \cdot \vec{v}; \quad r \in \mathbb{R}$$

$$l : \vec{u} = \vec{p}_1 + r \cdot \vec{v}; \quad r \in [0; 1]$$

$$d = \|\overline{P_1P_2}\| = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

VectorLine

```
int x1,y1,x2,y2,x,y,dx,dy;
double r, step;

dy = y2-y1;
dx = x2-x1;

step = 1.0/Math.sqrt(dx*dx+dy*dy);
for (r=0.0; r <= 1; r=r+step) {
    x = (int)(x1+r*dx+0.5);
    y = (int)(y1+r*dy+0.5);
    setPixel(x,y);
}
```

Gradengleichung als Funktion

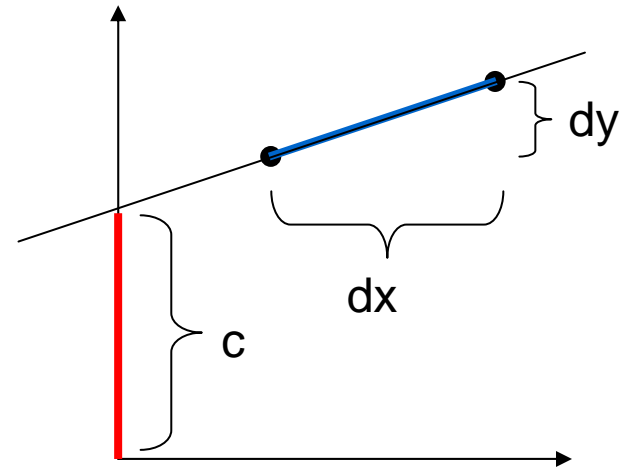
$$y = f(x) = s \cdot x + c$$

$$s = \frac{y_2 - y_1}{x_2 - x_1}$$

$$\frac{y_1 - c}{x_1 - 0} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$c = \frac{y_1 \cdot x_2 - y_2 \cdot x_1}{x_2 - x_1}$$

$$y = \frac{y_2 - y_1}{x_2 - x_1} \cdot x + \frac{x_2 \cdot y_1 - x_1 \cdot y_2}{x_2 - x_1}$$



StraightLine

von links nach rechts

```
s = (double)(y2-y1)/(double)(x2-x1);  
c = (double)(x2*y1-x1*y2)/(double)(x2-x1);  
  
for (x=x1; x <= x2; x++) {  
    y = (int)(s*x+c+0.5);  
    setPixel(x,y);  
}
```

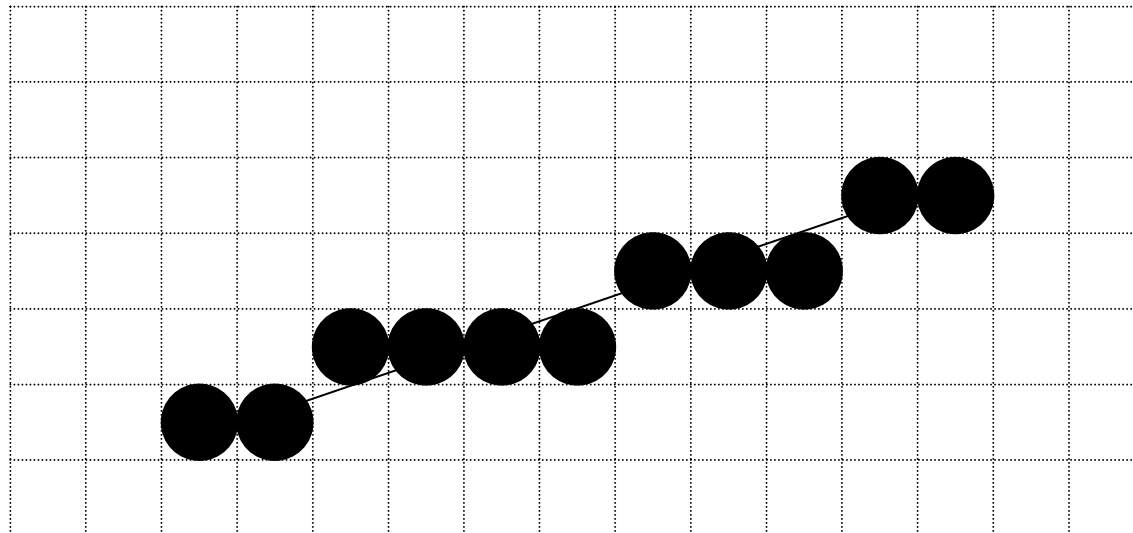
Oktanden

1.

Bresenham

Steigung $s = \Delta y / \Delta x = 3/10 = 0.3$

Fehler $error = y_{ideal} - y_{real}$



0.0 0.3 0.6 -0.1 0.2 0.5 0.8 0.1 0.4 0.7 0.0
-0.4 -0.2 -0.3

BresenhamLine, die 1.

```
dy = y2-y1; dx = x2-x1;
s = (double)dy/(double)dx;
error = 0.0;
x = x1;
y = y1;
while (x <= x2){
    setPixel(x,y);
    x++;
    error = error + s;
    if (error > 0.5) {
        y++;
        error = error - 1.0;
    }
}
```

Integer-Arithmetik

Mache Steigung + Fehler ganzzahlig:

$$dx := x_2 - x_1$$

$$dy := y_2 - y_1$$

$$s_{neu} = s_{alt} \cdot 2dx = \frac{dy}{dx} \cdot 2dx = 2dy$$

BresenhamLine, die 2.

```
dy = y2-y1; dx = x2-x1;  
s = (double)dy/(double)dx; delta = 2*dy  
error = 0.0;  
x = x1;  
y = y1;  
while (x <= x2){  
    setPixel(x,y);  
    x++;  
    error = error + s; delta  
    if (error > 0.5) { dx  
        y++;  
        error = error - 1.0; 2*dx  
    }  
}
```

multipliziere Steigung mit 2dx

Vergleich mit 0

- vergleiche **error** mit 0,
d.h. verschiebe **error** um $(x_2 - x_1)$ nach unten
- verwende **schrift** für $-2 * dx$

BresenhamLine, die 3.

```
dy = y2-y1; dx = x2-x1;
delta = 2*dy;
error = 0.0; -dx
x = x1; schritt= -2*dx
y = y1;
while (x <= x2){
    setPixel(x,y);
    x++;
    error = error + delta;
    if (error > dx) { 0
        y++;
        error = error -2*dx; + schritt
    }
}
```

Verschiebe error um 2dx nach unten

BresenhamLine

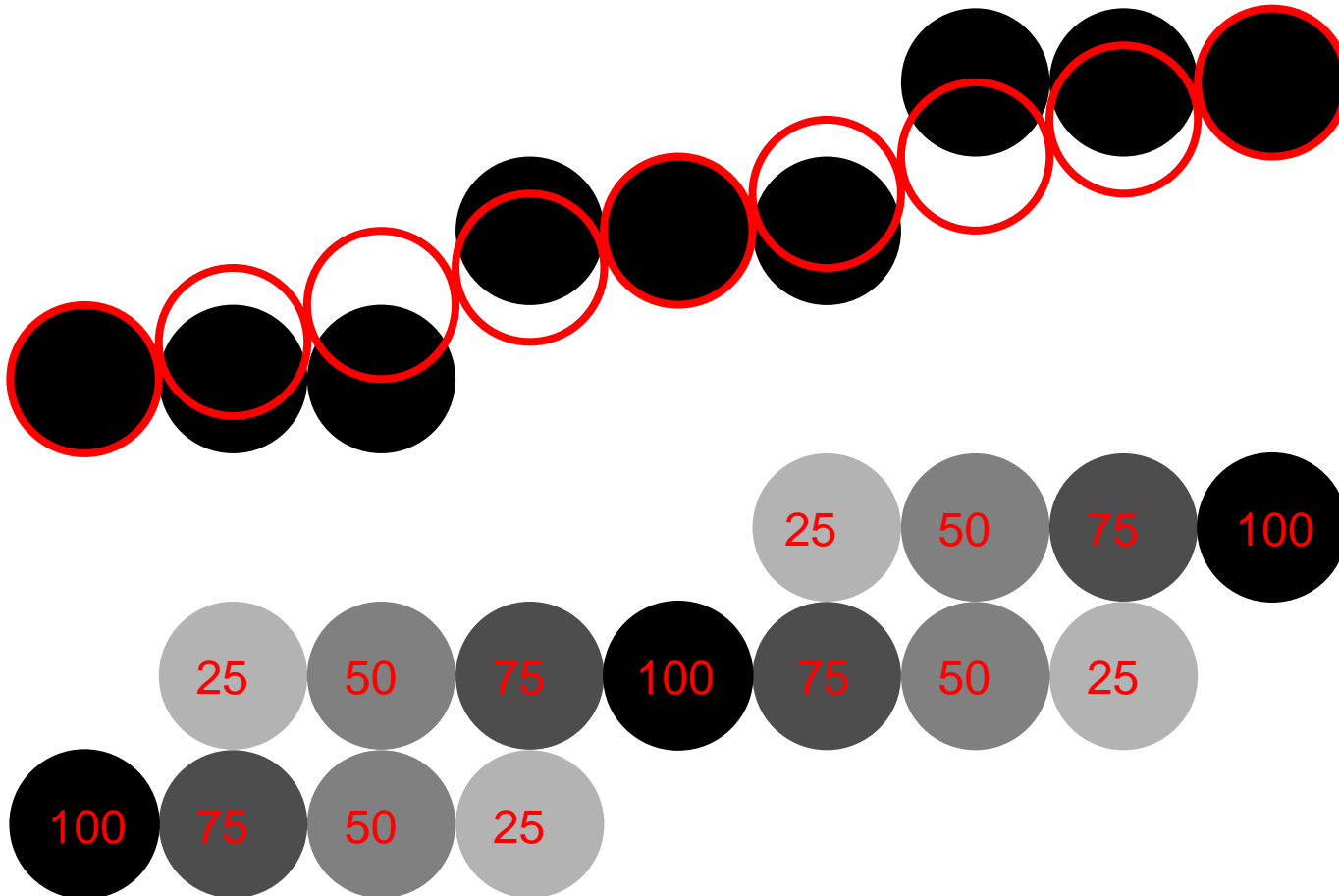
alle 8 Oktanten durch Fallunterscheidung abhandeln:

~cg/2008/skript/Sources/drawBresenhamLine.jav.html

Java-Applet:

~cg/2008/skript/Applets/2D-basic/App.html

Antialiasing

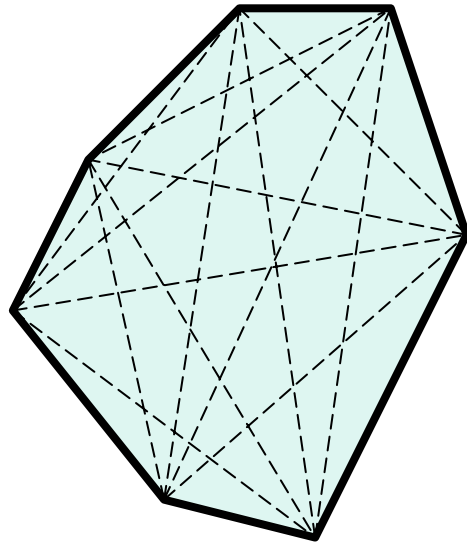


Antialiasing in Adobe Photoshop

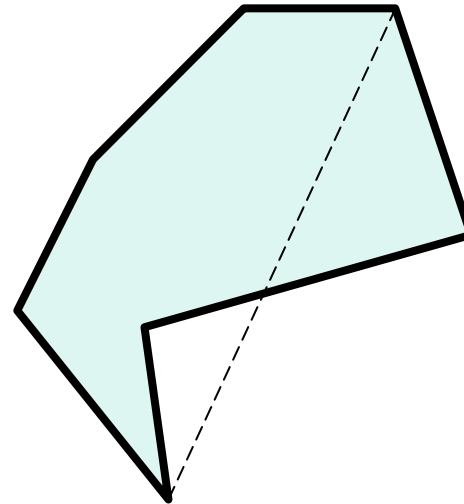


The image displays two rows of the lowercase letters 'a', 'b', and 'c'. The top row shows the letters with a pixelated, jagged appearance, representing text without antialiasing. The bottom row shows the same letters with smooth, blurred edges, representing text with antialiasing. The antialiased version uses a grayscale gradient to blend the black text with the white background, reducing the visible pixelation.

Polygon



konvex



konkav

Punkt versus Gerade

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} + r \cdot \begin{pmatrix} 7-2 \\ 5-3 \end{pmatrix}$$

$$x = 2 + 5r$$

$$y = 3 + 2r$$

$$2x = 4 + 10r$$

$$-5y = -15 - 10r$$

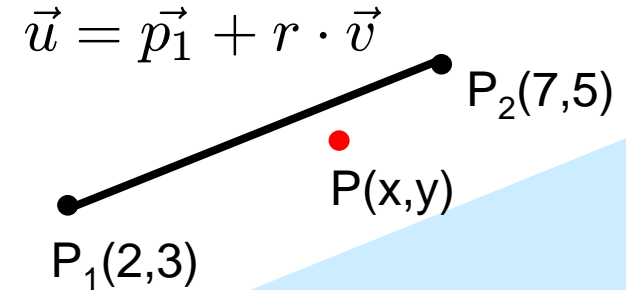
$$2x - 5y = -11$$

$$2x - 5y + 11 = 0$$

$F(x,y) = 0$ falls P auf der Geraden

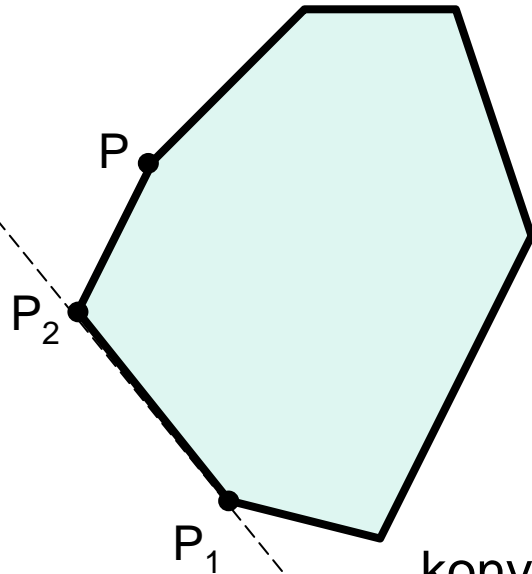
> 0 falls P rechts von der Geraden

< 0 falls P links von der Geraden



$$2x > 5y$$

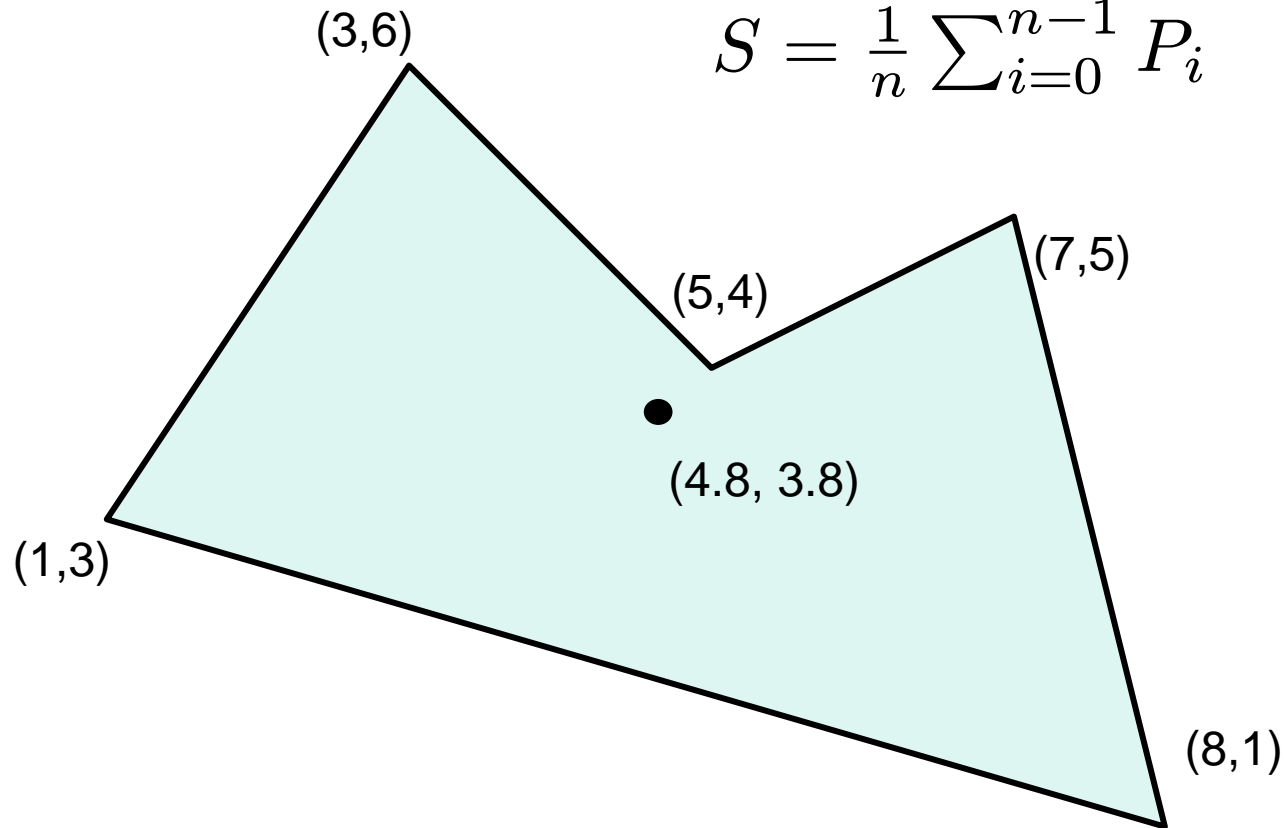
Konvexitätstest nach Paul Bourke



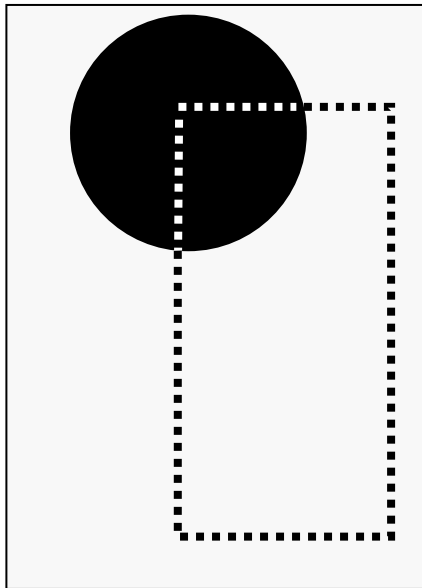
konvex, falls für alle $P(x,y)$
kein Vorzeichenwechsel bei $F(x,y)$

Schwerpunkt

$$S = \frac{1}{n} \sum_{i=0}^{n-1} P_i$$



Zeichnen und Löschen mit XOR

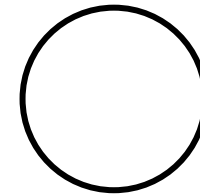
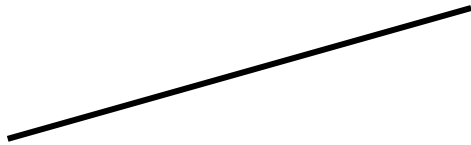


Pixel:	01101011
Gummiband:	11111111
XOR ergibt:	10010100
Gummiband:	11111111
XOR ergibt:	01101011

Beispiel für Gummiband:

~cg/2008/skript/Applets/2D-basic/App.html

Algorithmen zum Zeichnen



Parametrisiert:

$$x := f_1(t); y := f_2(t)$$

Gradengleichung:

$$y := f(x)$$

Bresenham:

```
x++; if (...) {y++; ... }
```

Parametrisiert:

$$x = f_1(t); y := f_2(t)$$

Kreisgleichung:

$$y := f(x)$$

Bresenham:

```
x++; if (...) {y--; ... }
```

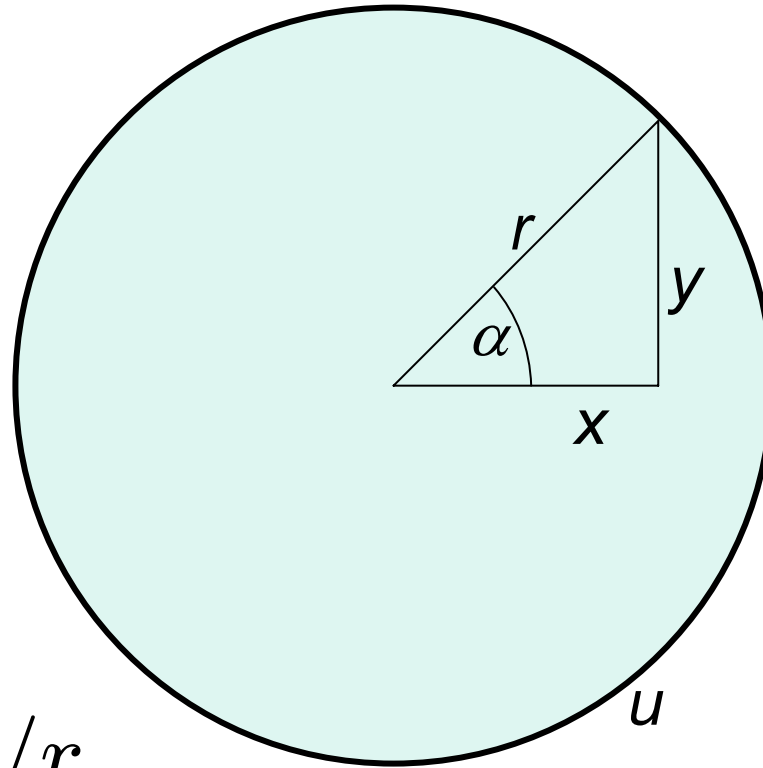
Kreis um (0,0), parametrisiert

$$x = r \cdot \cos(\alpha)$$

$$y = r \cdot \sin(\alpha)$$

$$u = 2 \cdot \pi \cdot r$$

$$step = \frac{2 \cdot \pi}{2 \cdot \pi \cdot r} = 1/r$$



TriCalcCircle

```
double step = 1.0/(double r);  
double winkel;  
  
for (winkel = 0.0;  
     winkel < 2*Math.PI;  
     winkel = winkel+step){  
  
    setPixel((int) r*Math.sin(winkel)+0.5,  
            (int) r*Math.cos(winkel)+0.5);  
}
```

TriTableCircle

```
// Tabellen sin + cos seien berechnet
// für ganzzahlige Winkel von 0..360

int winkel;

for (winkel = 0;
     winkel < 360;
     winkel++){

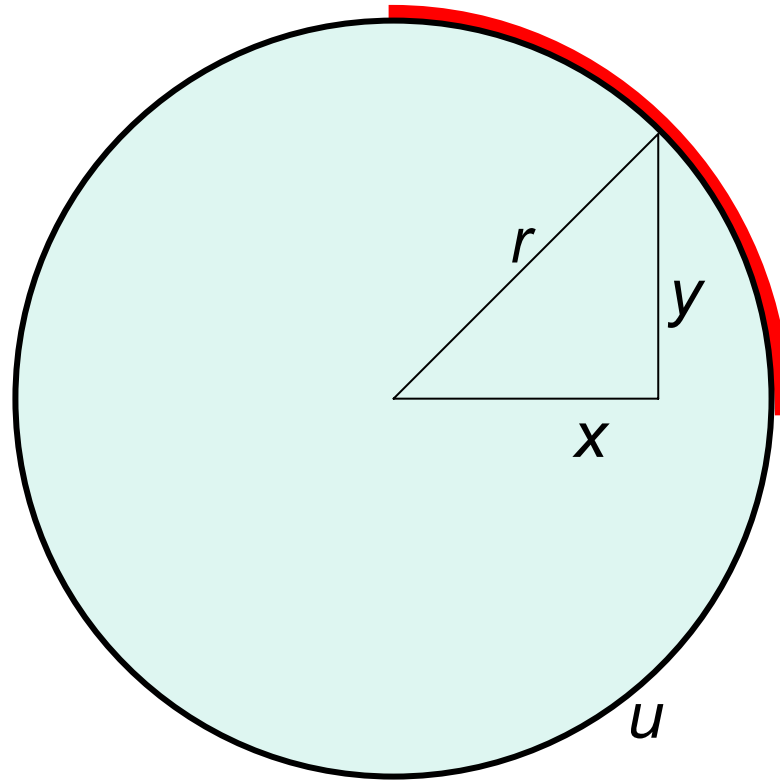
    setPixel((int) r*sin[winkel] + 0.5,
             (int) r*cos[winkel] + 0.5);
}
```

Problem: konstante Zahl von Kreispunkten !

Kreis als Funktion im 1. Quadranten

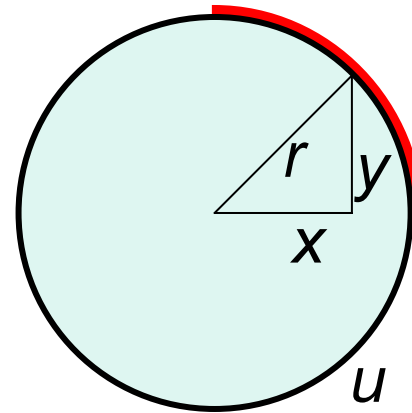
$$x^2 + y^2 = r^2$$

$$y = \sqrt{r^2 - x^2}$$



PythagorasCircle, die 1.

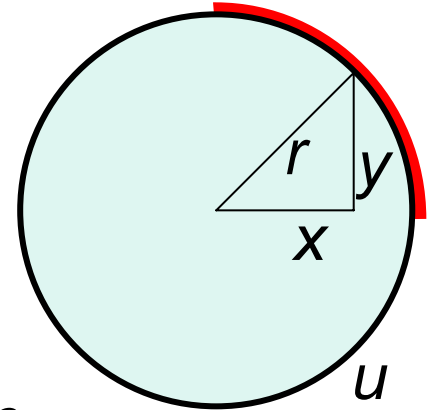
$$y = \sqrt{r^2 - x^2}$$



```
for (x=0; x <=r; x++){  
    y = (int) Math.sqrt(r*r-x*x);  
    setPixel(x,y);  
}
```

PythagorasCircle, die 2.

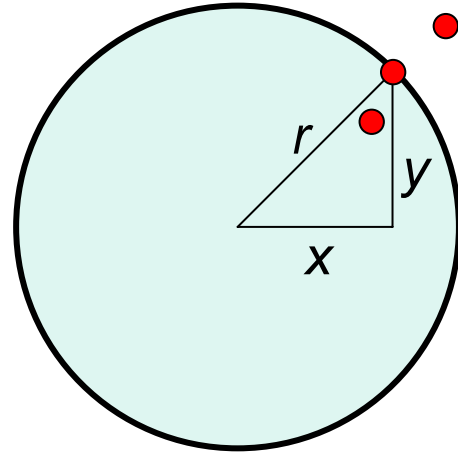
$$y = \sqrt{r^2 - x^2}$$



$$w = r^2 - 1, r^2 - 4, r^2 - 9, r^2 - 16, \dots$$

```
d = 1;
w = r*r;
for (x=0; x <= r; x++) ){
    y = (int) Math.sqrt(w);
    setPixel(x,y);
    w = w-d
    d = d+2;
}
```

Punkt versus Kreis



$$x^2 + y^2 = r^2$$

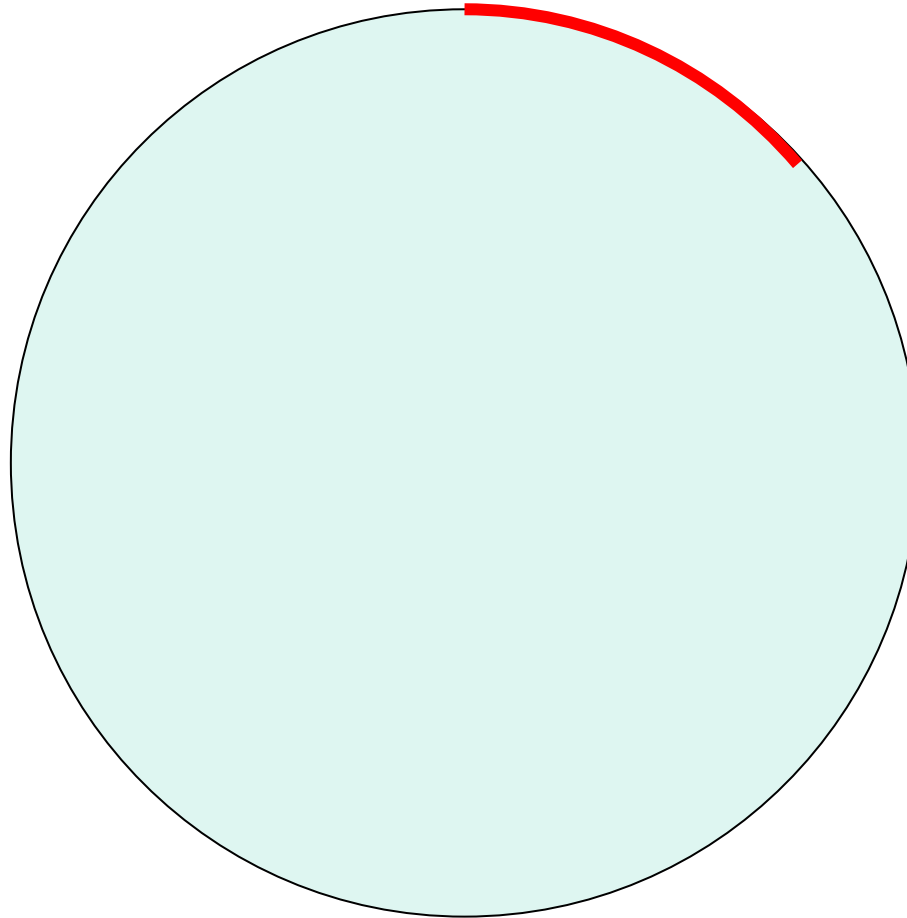
$$F(x, y) = x^2 + y^2 - r^2$$

$F(x, y) = 0$ für (x, y) auf dem Kreis

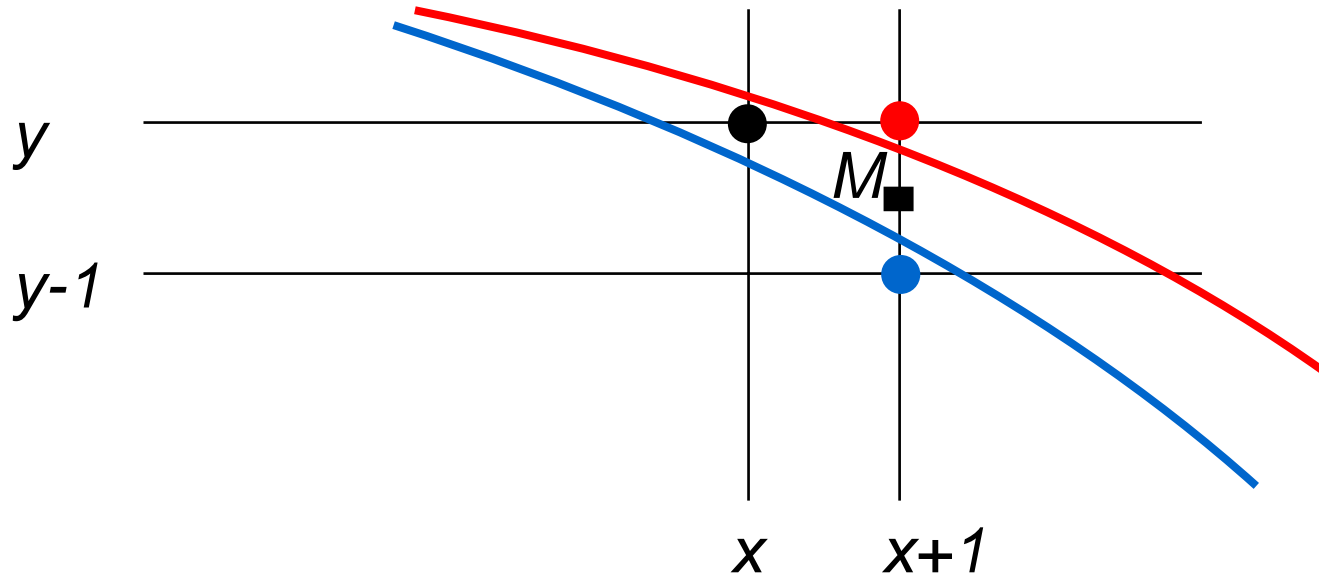
< 0 für (x, y) innerhalb des Kreises

> 0 für (x, y) außerhalb des Kreises

Kreis im 2. Oktanten



Entscheidungsvariable Δ



$$\Delta = F(x+1, y-1/2)$$

$\Delta < 0 \Rightarrow M$ liegt innerhalb \Rightarrow wähle $(x+1, y)$

$\Delta \geq 0 \Rightarrow M$ liegt außerhalb \Rightarrow wähle $(x+1, y-1)$

Berechnung von Δ

$$\Delta = F(x+1, y-\frac{1}{2}) = (x+1)^2 + (y-\frac{1}{2})^2 - r^2$$

$$\Delta < 0 \Rightarrow$$

$$\Delta' = F(x+2, y-\frac{1}{2}) = (x+2)^2 + (y-\frac{1}{2})^2 - r^2 =$$

$$\Delta + 2x + 3$$

$$\Delta \geq 0 \Rightarrow$$

$$\Delta' = F(x+2, y-\frac{3}{2}) = (x+2)^2 + (y-\frac{3}{2})^2 - r^2 =$$

$$\Delta + 2x - 2y + 5$$

$$\text{Startwert } \Delta = F(1, r-\frac{1}{2}) = 1^2 + (r-\frac{1}{2})^2 - r^2 =$$

$$5/4 - r$$

BresenhamCircle, die 1.

```
x = 0;
y = r;
delta = 5.0/4.0 - r;
while (y >= x) {
    setPixel(x,y);
    if (delta < 0.0) {
        delta = delta + 2*x + 3.0;
        x++;
    }
    else {
        delta = delta + 2*x - 2*y + 5.0;
        x++;
        y--;
    }
}
```

Substitutionen

$$d := \text{delta} - \frac{1}{4}$$

$$dx := 2x + 3$$

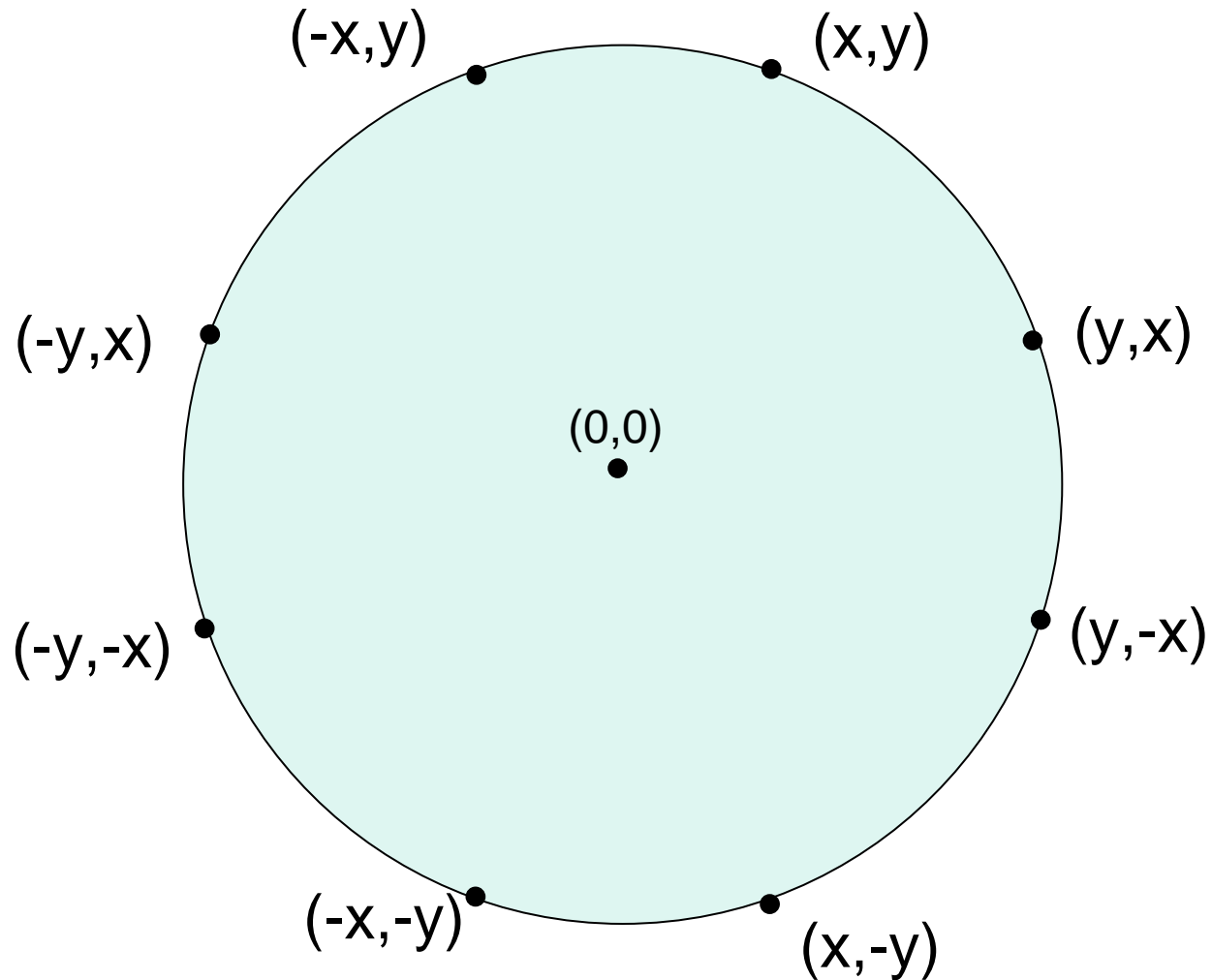
$$dxy := 2x - 2y + 5$$

BresenhamCircle, die 2.

```
x = 0;
y = r;
delta = 5.0/4.0 - r;
while (y >= x) {
    setPixel(x,y);
    if (delta < 0.0) {
        delta = delta + 2*x + 3.0;
        x++;
    } else {
        delta = delta + 2*x - 2*y + 5.0;
        x++;
        y--;
    }
}
d:=delta-1/4    dx:=2x+3    dxy:= 2x-2y+5
```

~~$d = 1 - r;$~~
 ~~$dx = 3;$~~
 ~~$dxy = -2*r + 5;$~~
 ~~$(d < 0.0)$~~
 ~~$d = d + dx;$~~
 ~~$dx = dx + 2;$~~
 ~~$dxy = dxy + 2;$~~
 ~~$d = d + dxy;$~~
 ~~$dx = dx + 2;$~~
 ~~$dxy = dxy + 4;$~~

Oktanden-Symmetrie



BresenhamCircle, die 3.

```
x=0; y=r; d=1-r; x=3; dx=3; dxy=-2*r+5;

while (y>=x){

    setPixel(+x,+y);
    setPixel(+y,+x);
    setPixel(+y,-x);
    setPixel(+x,-y);
    setPixel(-x,-y);
    setPixel(-y,-x);
    setPixel(-y,+x);
    setPixel(-x,+y);

    if (d<0) {d=d+dx; dx=dx+2; dxy=dxy+2; x++;}
    else     {d=d+dxy; dx=dx+2; dxy=dxy+4; x++; y--;}
}
```


BresenhamCircle

$$\begin{aligned} \text{Zahl der erzeugten Punkte} &= 4 \cdot \sqrt{2} \cdot r \\ &= 10\% \text{ unterhalb von } 2 \cdot \pi \cdot r \end{aligned}$$

Kreis mit Radius r um Mittelpunkt (x,y) :

~cg/2008/skript/Sources/drawBresenhamCircle.jav

Java-Applet:

~cg/2008/skript/Applets/2D-basic/App.html

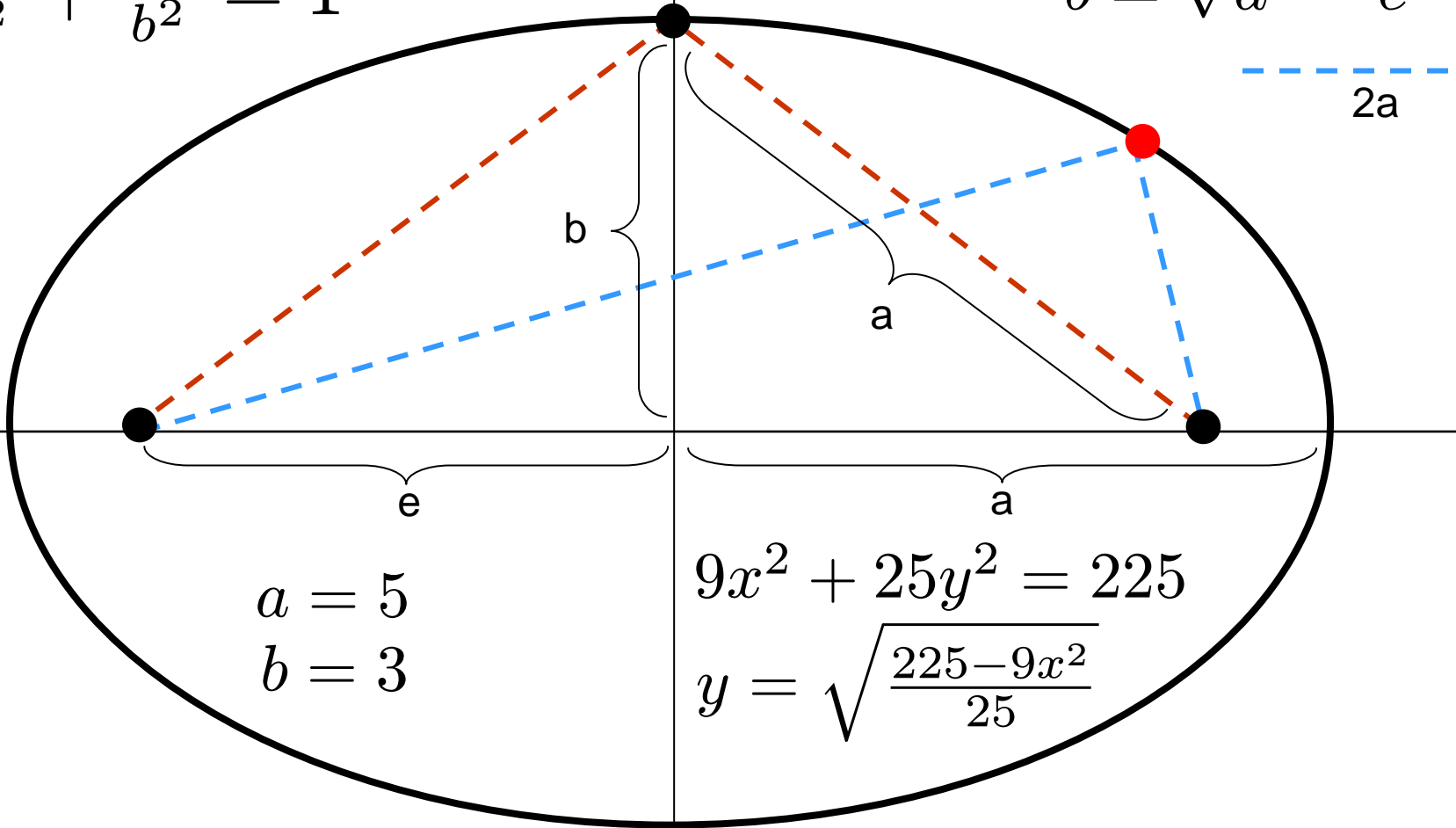
Java-Applet für Performance-Messung

~cg/2008/skript/Applets/circle/App.html

Ellipse um (0,0)

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

$$b = \sqrt{a^2 - e^2}$$

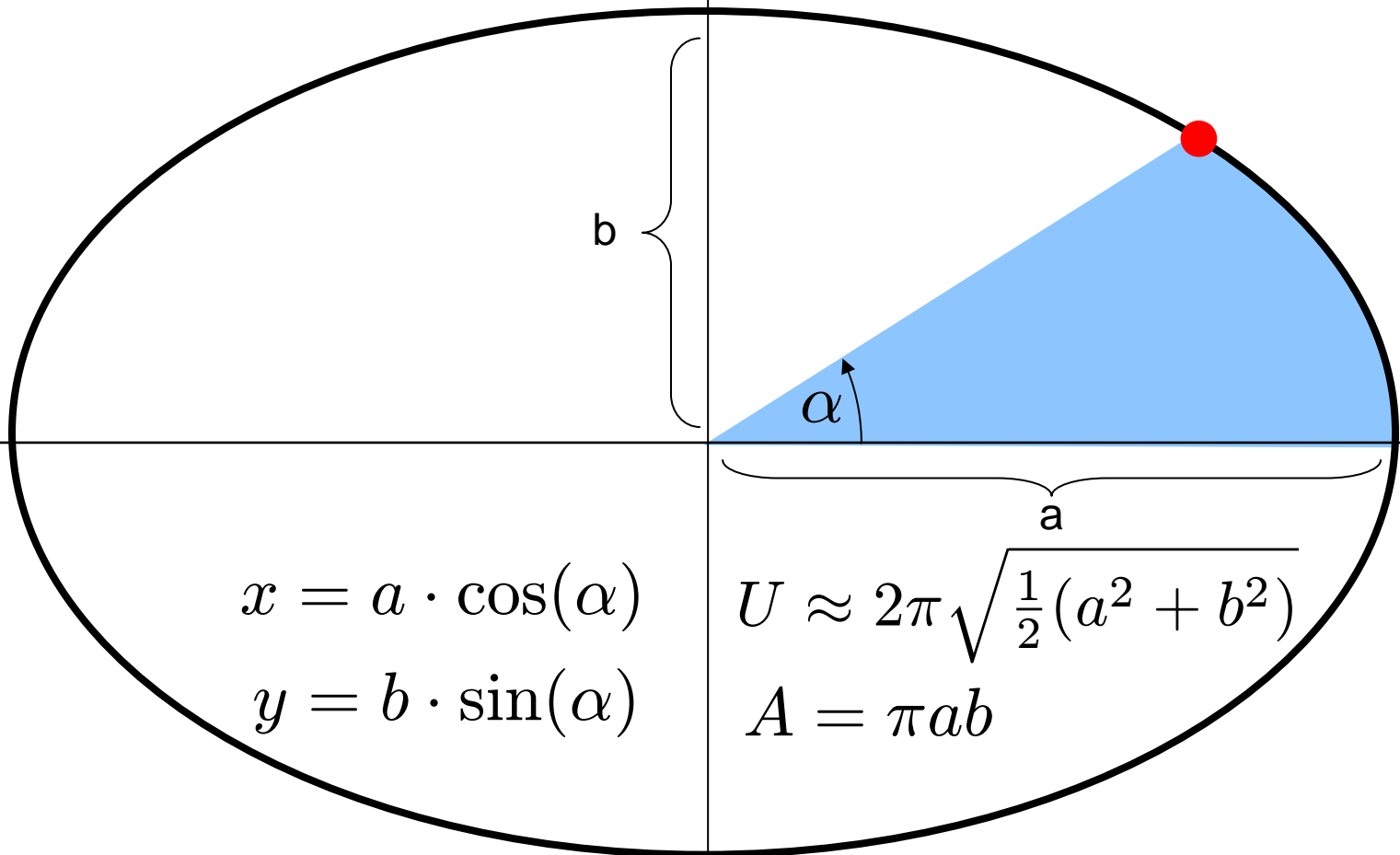


$$a = 5$$
$$b = 3$$

$$9x^2 + 25y^2 = 225$$

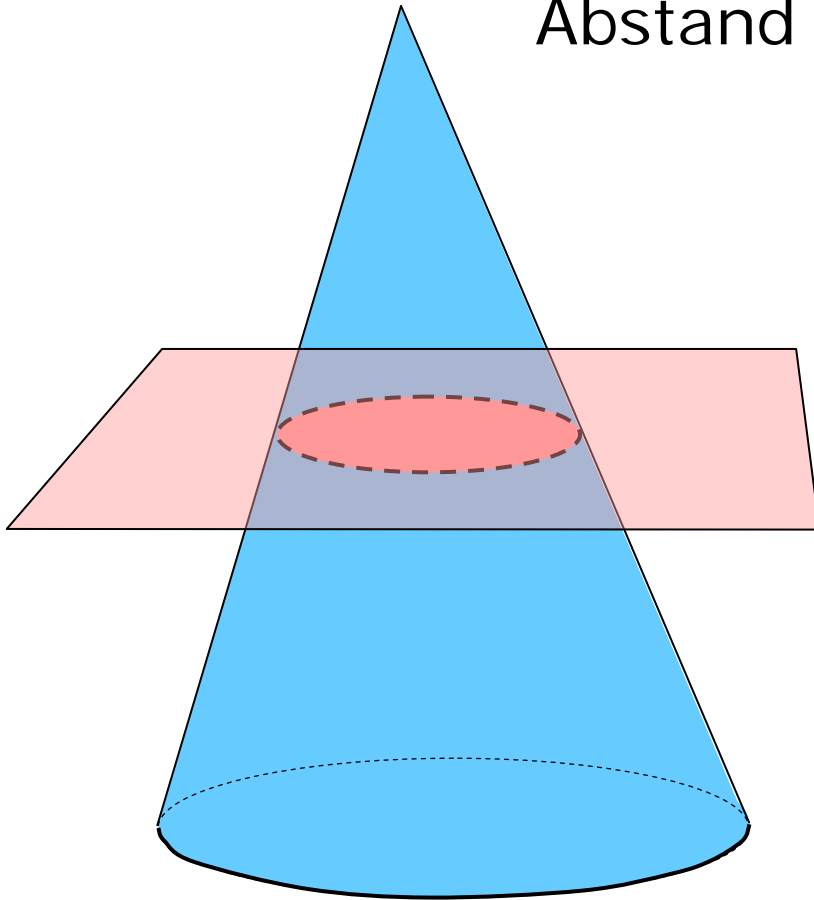
$$y = \sqrt{\frac{225-9x^2}{25}}$$

Ellipse um (0,0) parametrisiert

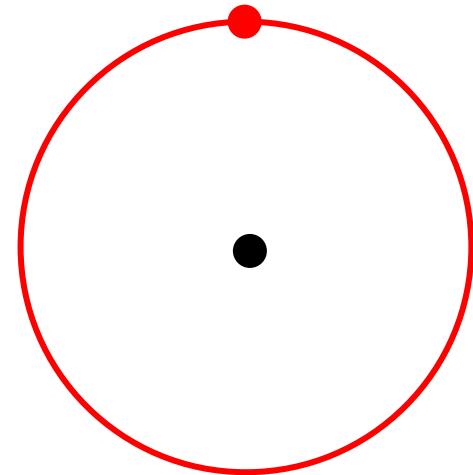


Kegelschnitt: Kreis

Abstand zu einem Punkt ist konstant



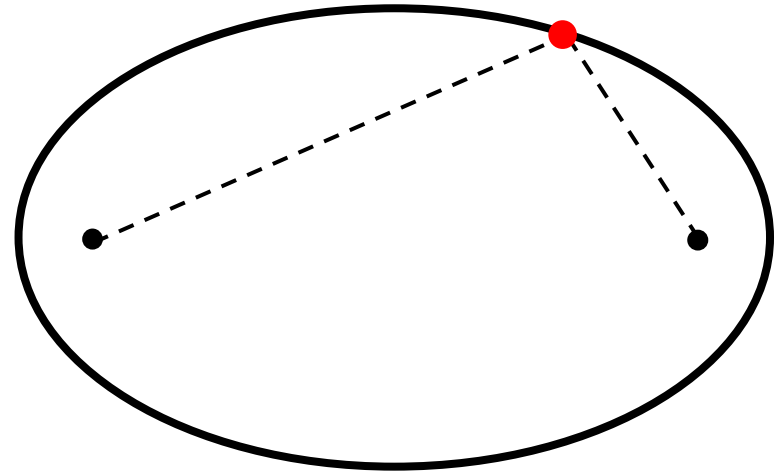
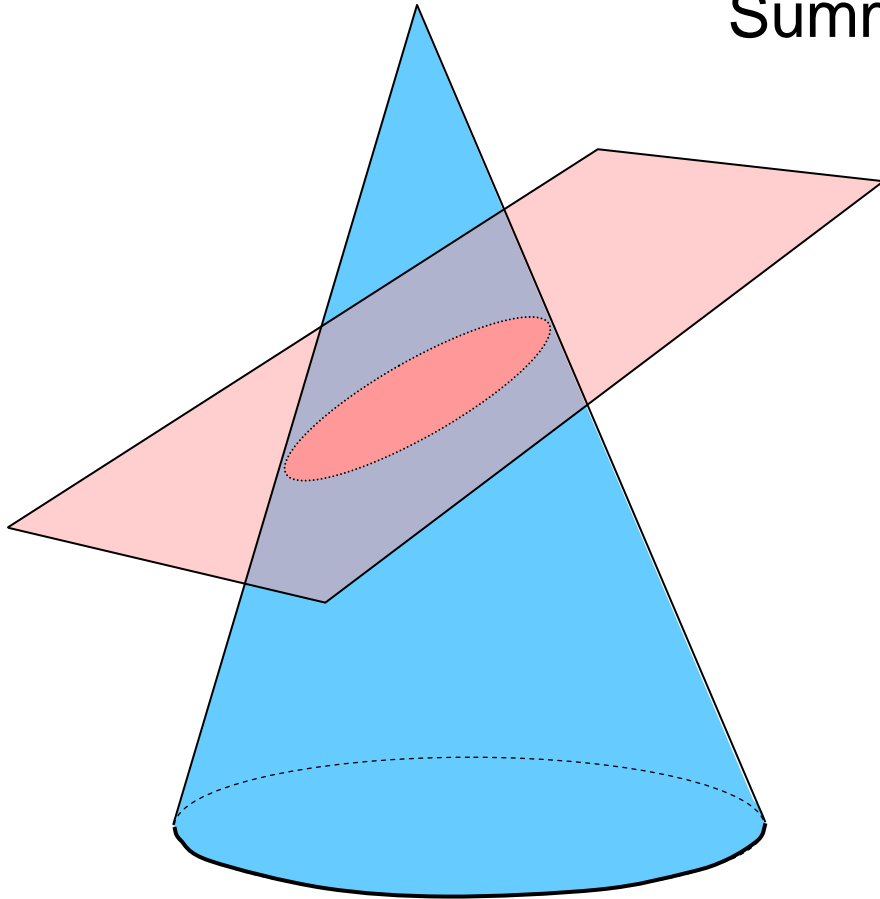
$$x^2 + y^2 = 1$$



Kegelschnitt: Ellipse

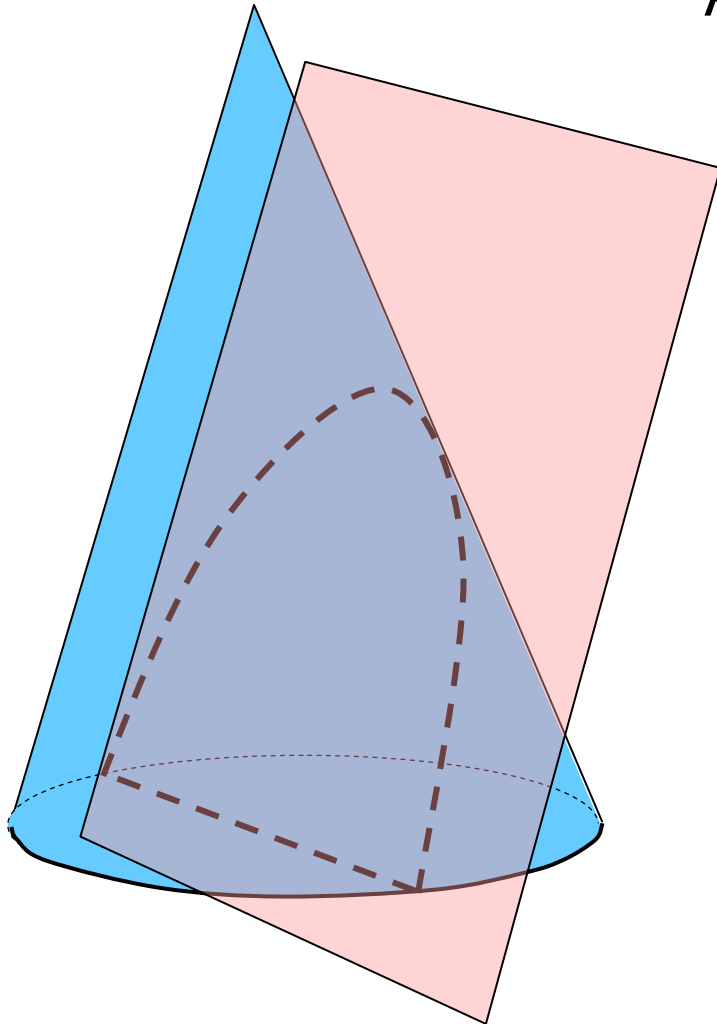
Summe der Abstände zu 2 Punkten
ist konstant

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

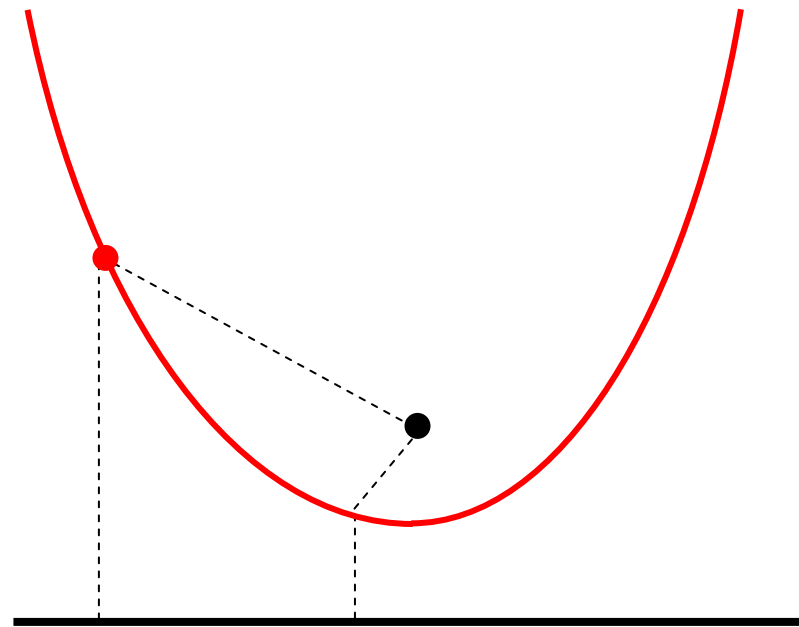


Kegelschnitt: Parabel

Abstand zu Punkt und Gerade
ist gleich

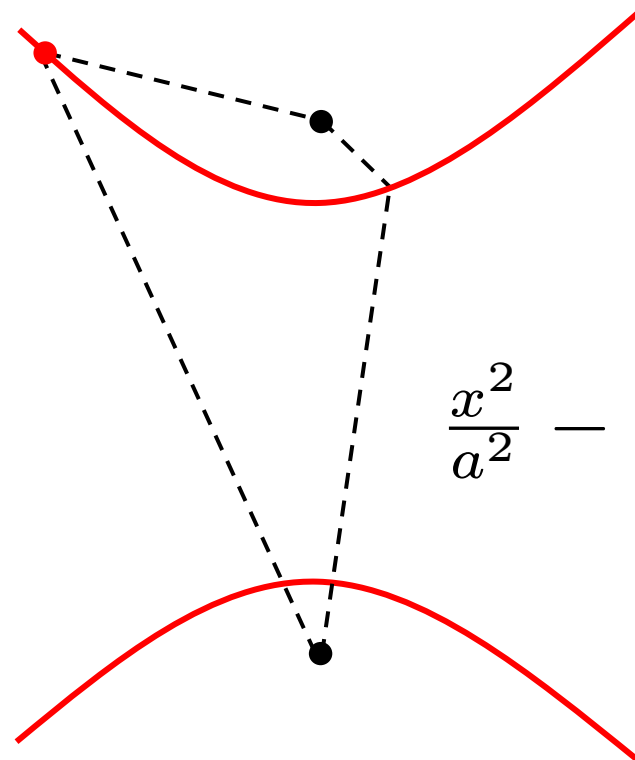
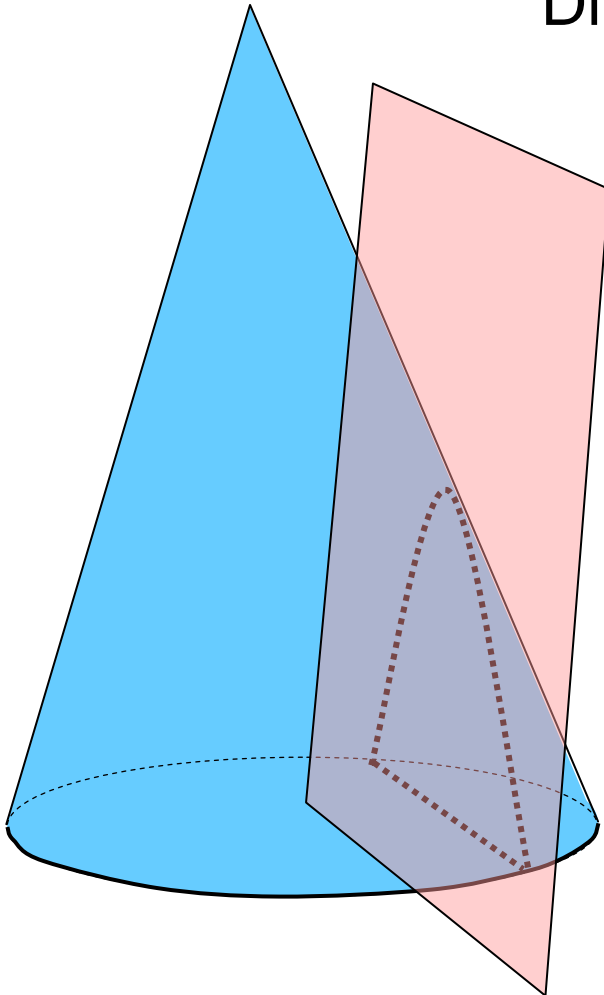


$$y = ax^2 + bx + c$$



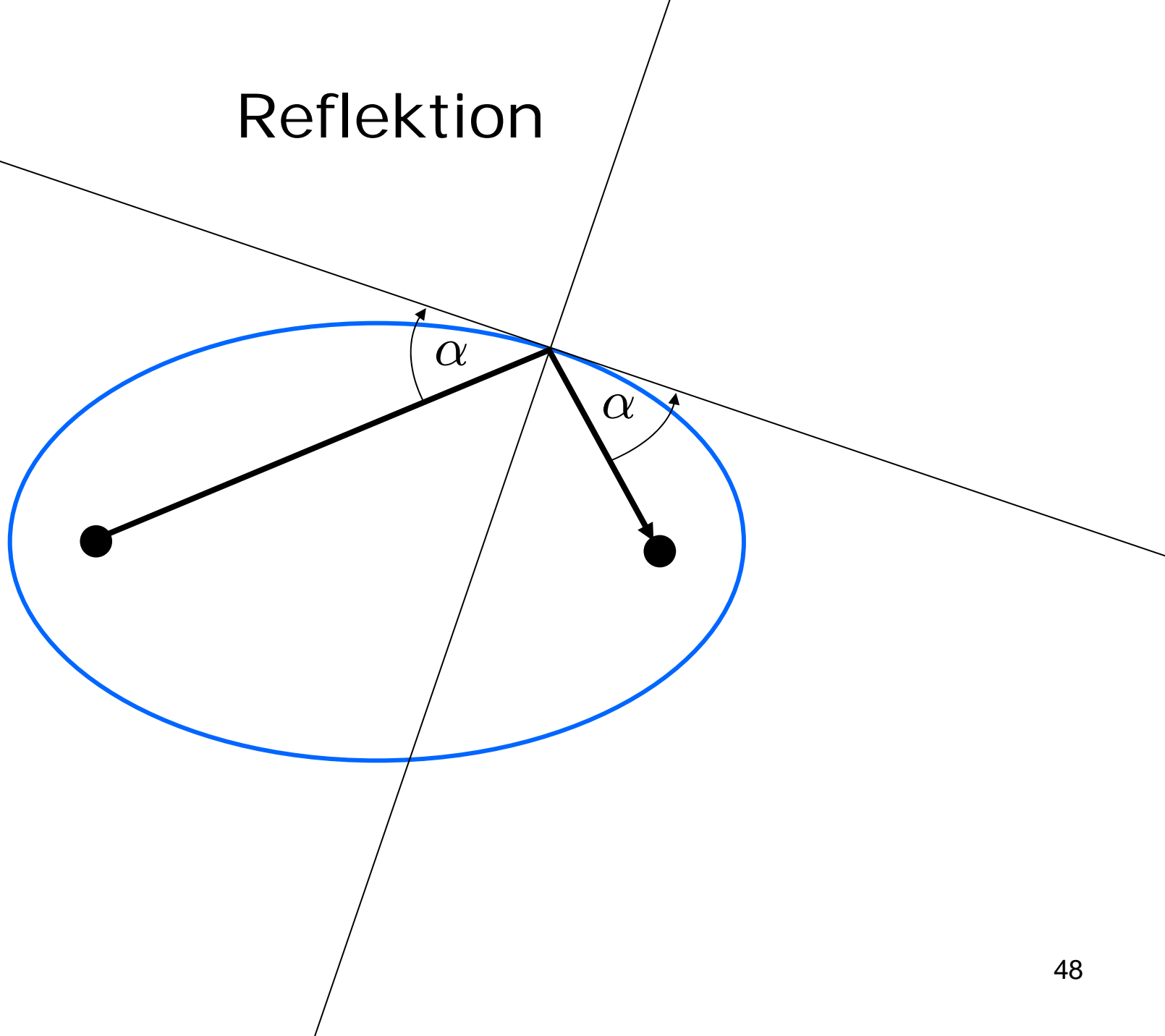
Kegelschnitt: Hyperbel

Differenz der Abstände zu 2 Punkten
ist konstant

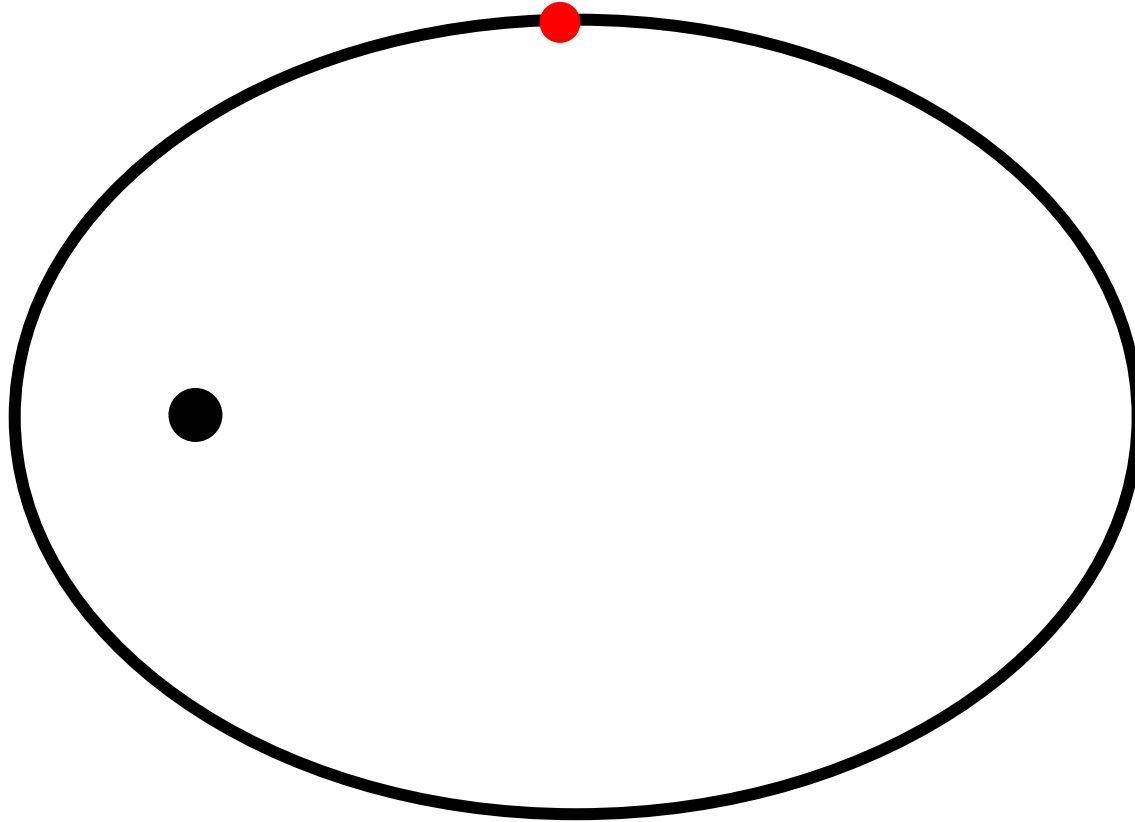


$$\frac{x^2}{a^2} - \frac{y^2}{b^2} = 1$$

Reflektion

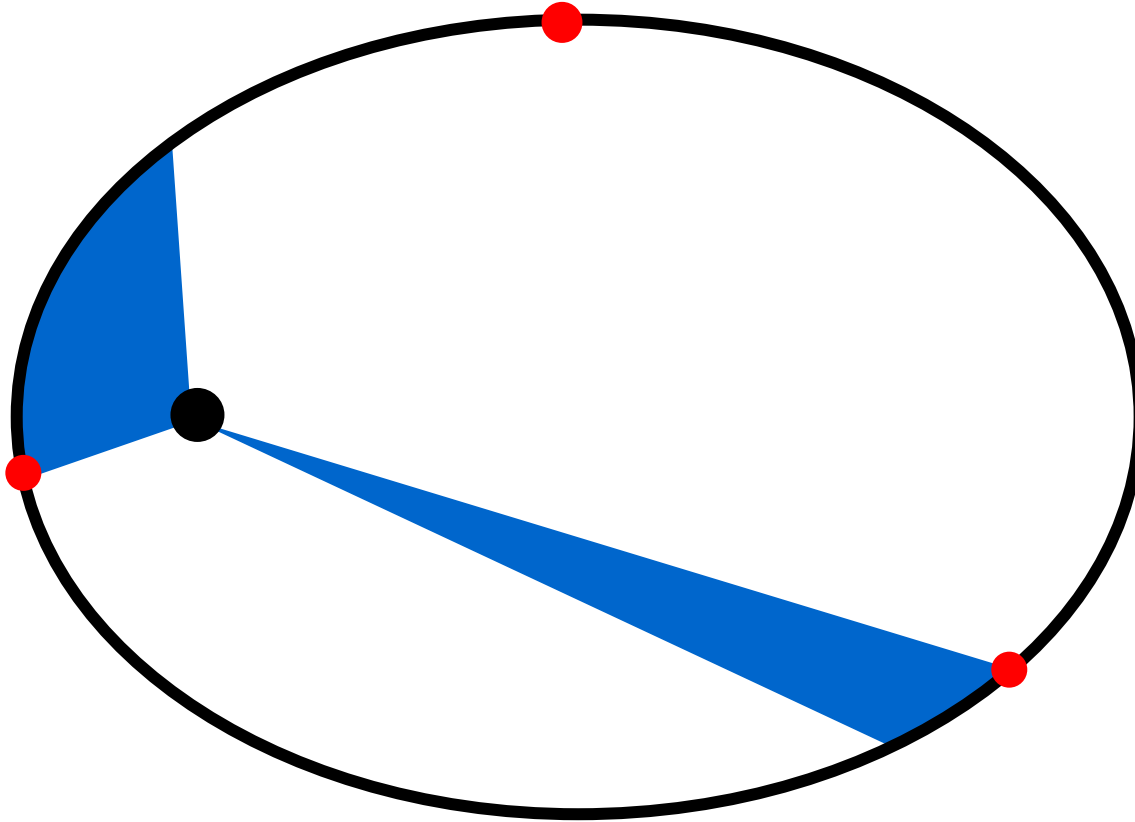


1. Keplersches Gesetz



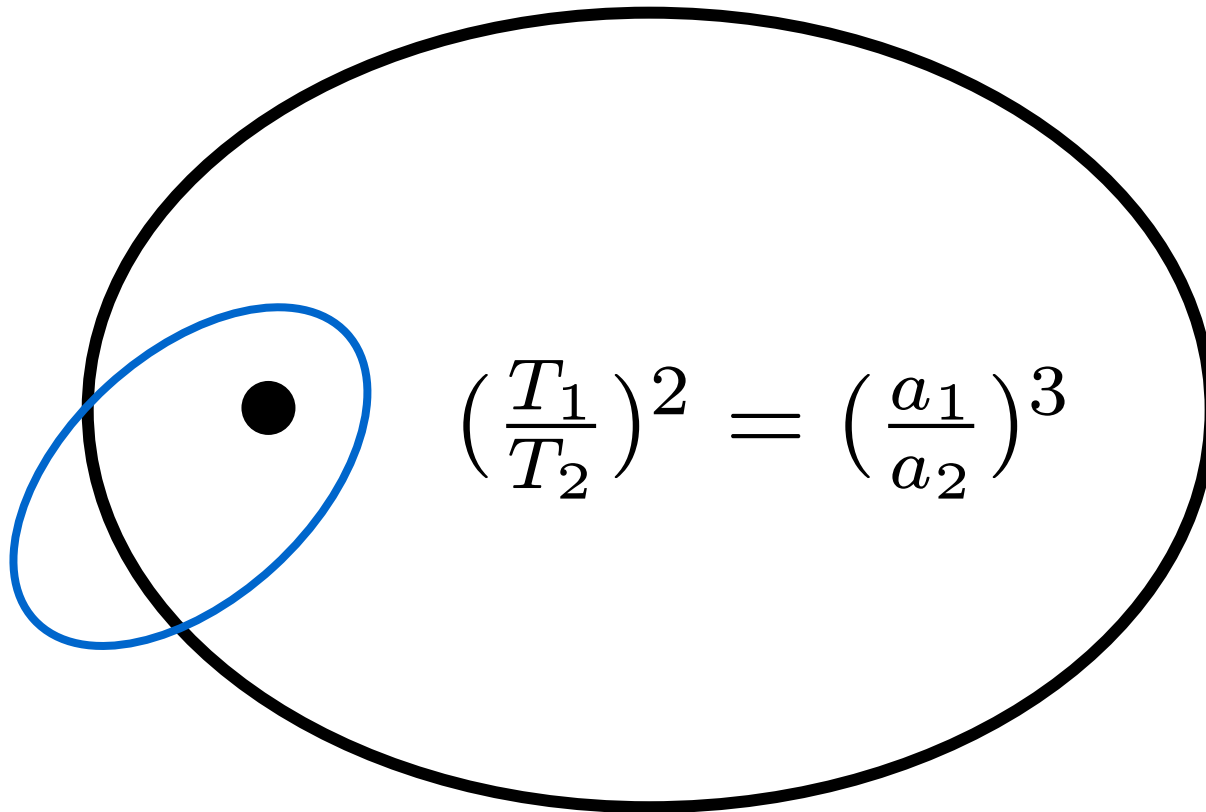
Die Planeten umkreisen die Sonne auf einer Ellipse

2. Keplersches Gesetz



In gleichen Zeiten überstreicht der Fahrstrahl gleiche Flächen

3. Keplersches Gesetz



Die Quadrate der Umlaufzeiten verhalten sich wie die Kuben der großen Halbachsen