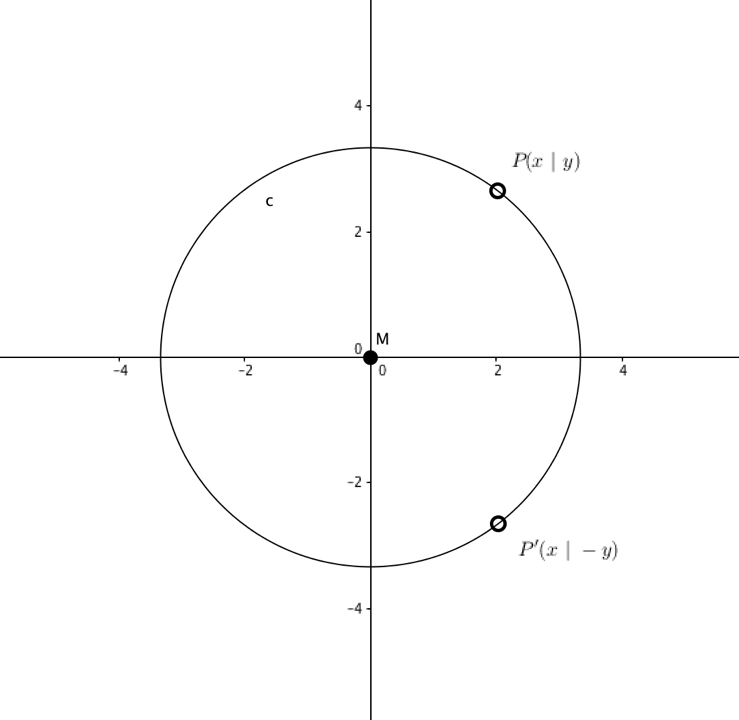
# Midpoint circle Algorithmus

Das Ziel des Midpoint circle Algorithmus ist es, die Pixelkoordinaten eines Kreises möglichst schnell und effizient zu berechnen. Wie er dies anstellt, werden wir Ihnen nun erklären.

Wen wir einen Kreis im Koordinatensystem betrachten erkennen wir schnell, dass er achsensymetrisch ist. Und zwar an der Y- sowie der X-Achse.

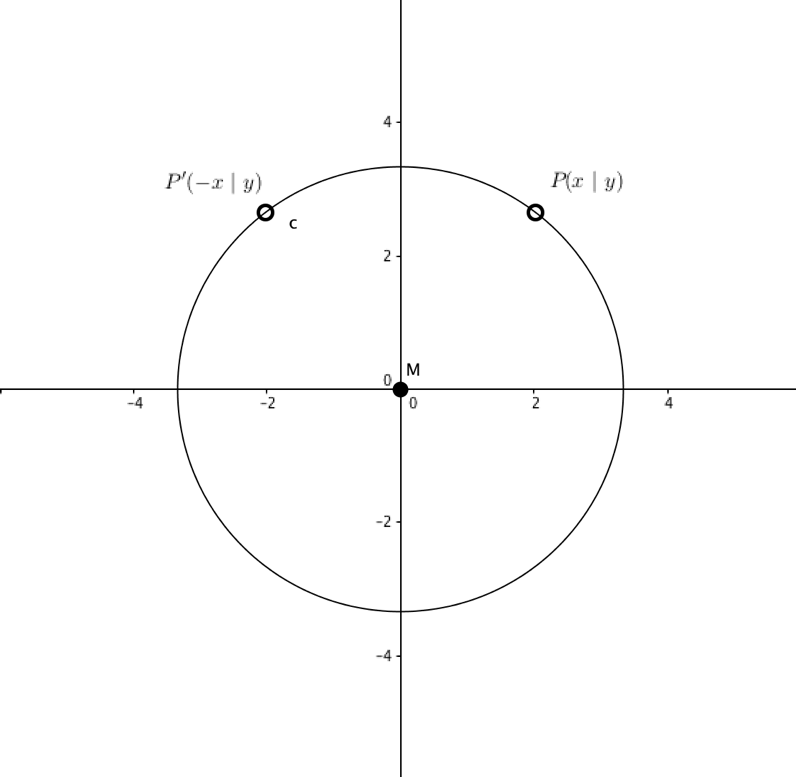
**Spiegelung an der Y-Achse:**



Wie wir in dieser Abbildung sehen, können wir jeden zu jedem berechneten Kreispunkt einen anderen spiegeln. Dies halbiert die Anzahl Kreispunkte, die wir berechnen müssen, um unseren Kreis zu zeichnen.

Um die Koordinaten des gegenüberliegenden Kreispunktes zu erhalte, multiplizieren wir die X-Koordinate unseres Originalpunktes mit -1.

**Spiegelung an der X-Achse:**

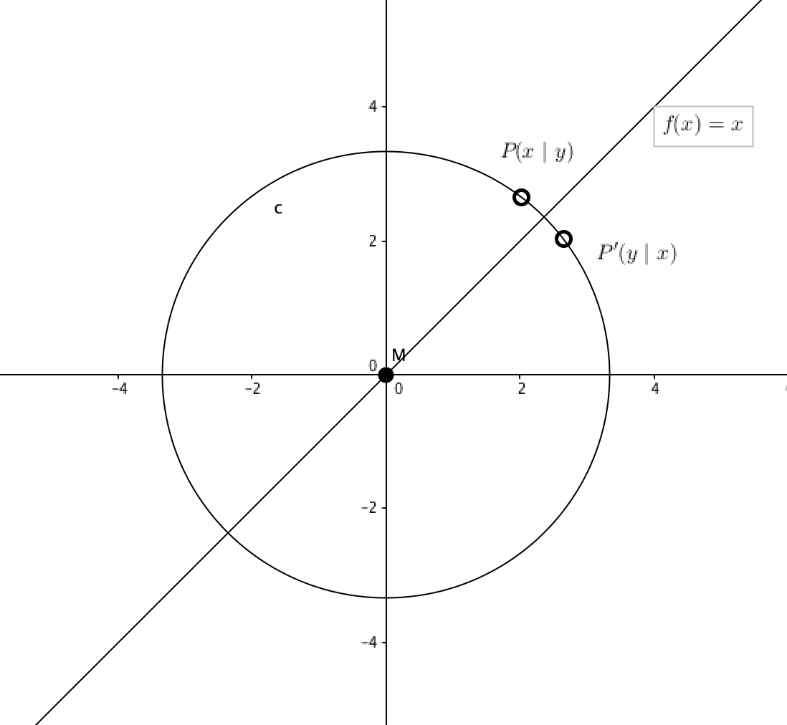


Dieselbe Spiegelung wie an der Y-Achse können wir auch an der   
X-Achse vornehmen. Wir haben nun die Menge an Kreispunkten, die zu berechnen sind ein weiteres Mal halbiert.

Um die Koordinaten des neuen Kreispunktes zu erhalte, multiplizieren wir die Y-Koordinate unseres Originalpunktes mit -1.

Da ein Kreis an allen Geraden, welche durch den Mittelpunkt verlaufen, gespiegelt werden kann, können wir die Mengen der zu berechnenden Kreispunkte weiter senken.

**Spiegelung an der Y=X Achse:**



Auch können wir unsere Bildpunkte an der Gerade Y = X spiegeln. Somit müssen wir nur noch die ein Sechstel des Kreises berechnen. Wir konnten den Rechenaufwand nochmals senken.

Um die Koordinaten des neuen Kreispunktes zu erhalte, tauschen wir die X und Y Koordinaten unseres berechneten Punktes.

**Spiegelung an der Y=-X Achse:**

**Ein Bild, das Text, Karte enthält.

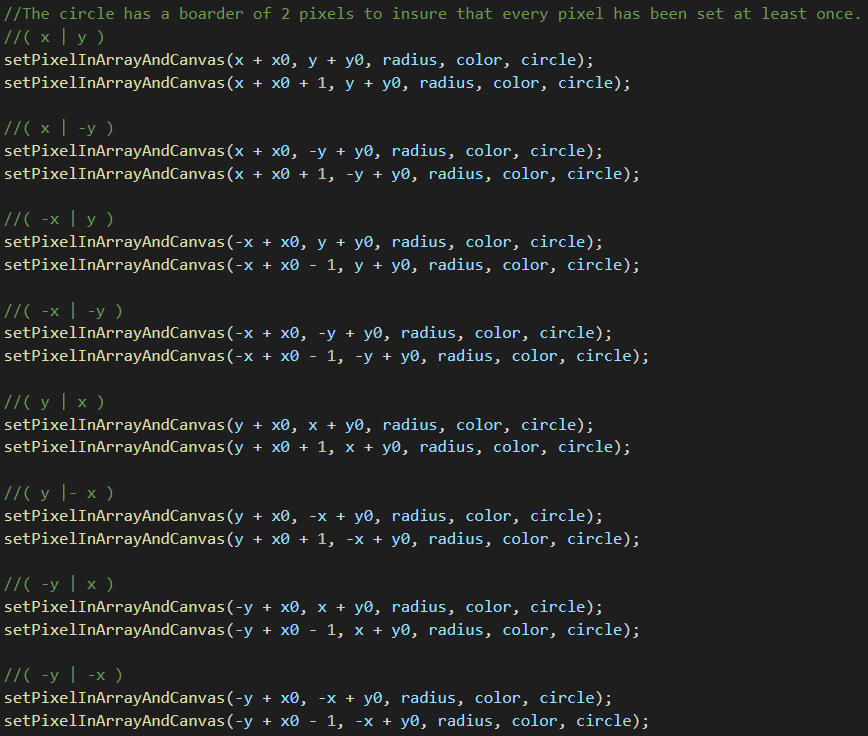
Automatisch generierte Beschreibung**

Dieselbe Spiegelung wie an der Geraden können wir auch an der Geraden vornehmen. Wir haben nun die originale Menge an Punkten, die wir berechnen müssen, um ein achtfaches verkleinert.

**Spiegelung an allen Achsen:**

Wir haben nun unseren Kreis in acht Sektoren geteilt und demonstriert, dass wir mit den Kreispunkten einem dieser Sektoren unseren Kreis um jeden anderen Sektor erweitern können.

Nun können wir die jedem Sektor eine Formel zuweisen. Wir müssen nur noch den Sektor berechnen und können die restlichen Kreispunkte mit den entsprechenden Formeln ableiten.

Wir sehen diese Ableitung gut in diesem Ausschnitt aus unserem Code. Um unseren Kreis dicker erscheinen zu lassen, haben wir an der Aussenseite des Kreises immer einen weiteren Pixel eingefärbt.

**Berechnung der Kreispixel:**

Nun müssen wir nur noch die Punkte dieses einen Sektors berechnen. Wie wir deren Koordinaten erhalten wird im nächsten Abschnitt erklärt.

Was uns interessiert ist, welche Pixel wir einfärben sollen, um unseren Kreis zu zeichnen.

Pixel im Array

Wir können die theoretische Kreislinie mathematisch berechnen. Jedoch müssen wir eine Formel entwickeln, mit der wir bestimmen, ob wir einen Pixel einfärben oder nicht.

Theoretischer Kreis

**Auswahl der Kreispixel:**

Wir berechnen den Theoretischen Mittelpunkt jedes Pixels. Mit diesen bilden wir ein Zweierpaar von Pixeln, die sich berühren.

Was uns nun interessiert ist, ob die theoretische Kreislinie zwischen den Mittelpunkten eines Pares verläuft. Um dies zu erfahren berechnen wir die X-Koordinate des Kreises beim Y-Wert der beiden Pixelmittelpunkten berechnen und diese mit den X-Koordinaten der beiden Punkte vergleichen.

Verläuft die theoretische Kreislinie zwischen einem Mittelpunktpaar, so werden wir diese beiden Pixel einfärben

Nun bilden wir nochmals Pixelpaare. Diesmal mit allen Pixel die übereinander sind. Auch bei ihnen führen wir denselben Prozess durch

**Das Resultat:**

Nachdem wir dies für einen Sektor des Kreises gemacht haben, können wir die Restlichen Kreispixel mit der schon beschriebenen Formel berechnen.

Auf diese Art können wir mit dem Midpoint Circle Algorithmus schnell und effizient einen Kreis aus Pixeln berechnen. Perfekt für unser Projekt.