Bilgisayar Grafiği
HAFTA 3
Doğru Çizme Algoritmaları

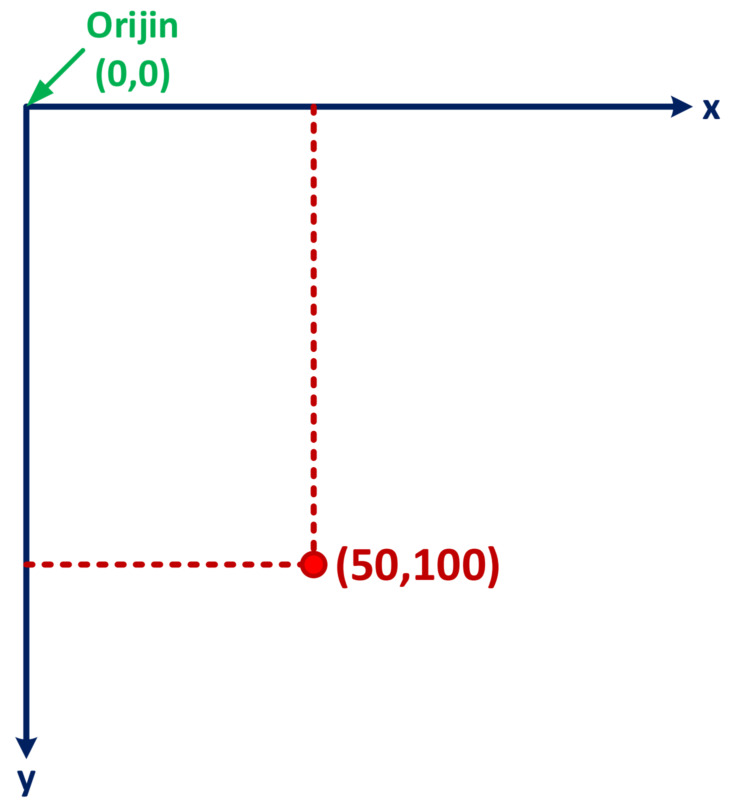
Arş. Gör. Dr. Gülüzar ÇİT
Bilgisayar ve Bilişim Bilimleri Fakültesi
Bilgisayar Mühendisliği Bölümü
gulizar@sakarya.edu.tr

Konu & İçerik

- 2B Koordinat Sistemi
- Doğru Çizme Algoritmaları
  - DDA Doğru Çizme Algoritması
  - Bresenham Doğru Çizme Algoritması
- Çember Çizme Algoritması
- Elips Çizme Algoritması
- Poligon Doldurma

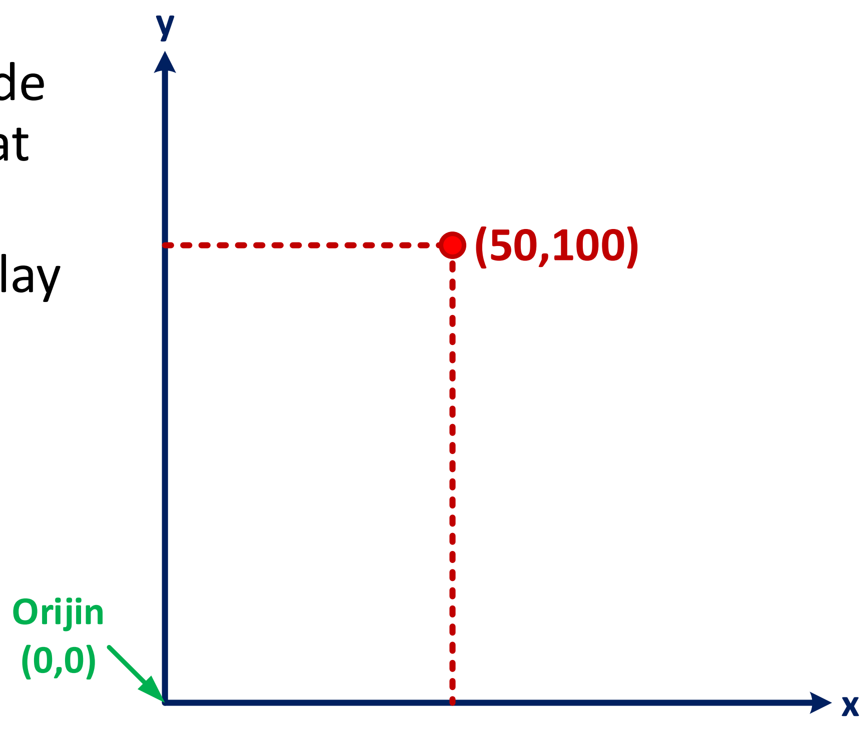
2B Koordinat Sistemleri

- **Sol El Koordinat Sistemi**
  - Genellikle görüntülerin kullandığı sistem
  - x ekseni -> sağa
  - y ekseni -> aşağıya
  - Raster ekranların donanımından dolayı
    - Görüntü tazelenirken ışın demeti, yukarıdan aşağıya doğru, her satırı soldan sağa tarayarak hareket eder.



2B Koordinat Sistemleri…

- **Sağ El Koordinat Sistemi**
  - X ekseninin sağa, y ekseninin de yukarıya doğru arttığı koordinat sistemi
  - Matematiksek ifadesi daha kolay



Doğru Çizme Algoritmaları

- İki boyutlu doğrunun kapalı gösterimi
  - $ax + by + c = 0$
- (x_1, y_1) ve (x_2, y_2) koordinatlarıyla belirtilen iki noktadan geçen doğru denkleminin katsayıları
  - $a = y_2 - y_1$
    $b = -(x_2 - x_1)$
    $\mathrm{c} = -(ax_1 + by_1)$

Doğru Çizme Algoritmaları…

- **ÖRNEK:** (4,-1) ve (8,5) noktalarından geçen doğrunun kapalı denklemini bulunuz.
  - 1. YOL:
    - $a = y_2 - y_1$
      $b = -(x_2 - x_1)$
      $\mathrm{c} = -(ax_1 + by_1)$
  - 2. YOL:
    - $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5-(-1)}{8-4} = {}^3/_2$

      (8,5) noktasını kullanırsak;

      $m = \frac{y-5}{x-8} = {}^3/_2$

      $3x - 24 = 2y - 10 \Rightarrow 3x - 2y - 14 = 0$

Doğru Çizme Algoritmaları…

- Doğruları göstermek için *y* koordinatının *x* cinsinden gösterildiği denklem
  - $y = mx + n$
  - *m*: doğrunun değimi
  - *n*: doğrunun y eksenini kestiği nokta
- (x_1, y_1) ve (x_2, y_2) koordinatlarıyla belirtilen iki noktadan geçen doğru denkleminin katsayıları
  - $m = \frac{y_2 - y_1}{x_2 - x_1}$

    $n = y_1 - mx_1$

Doğru Çizme Algoritmaları…

- **ÖRNEK:** (4,-1) ve (8,5) noktalarından geçen doğru denklemini $y = mx + n$ şeklinde hesaplayın.
  - $y = mx + n$
  - $m = \frac{y_2 - y_1}{x_2 - x_1} = \frac{5-(-1)}{8-4} = {}^3/_2$
  - (8,5) noktasını kullanırsak;

    $5 = {}^3/_2 . 8 + n \Rightarrow n = -7$

    $y = {}^3/_2 x - 7$

Doğru Çizme Algoritmaları…

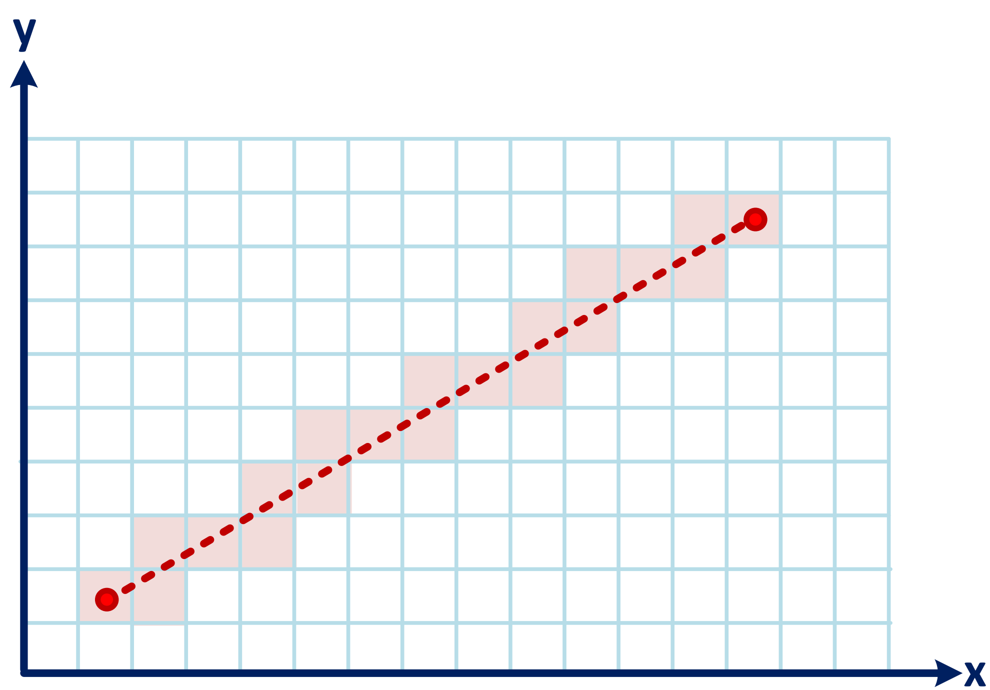
- **Çizim**
  - Ekranda çizimi yapılacak olan şekle bağlı olarak ekran noktalarının (piksellerin) önceden belirlenen renkte aydınlatılması, diğerlerinin aydınlatılmaması
- **Doğru Çizme**
  - Algoritmada verilen başlangıç ve bitiş noktalarına bağlı olarak doğrunun izlediği yol boyunca kesiştiği ardışık piksellerin (hangilerinin) aydınlatılması?

Doğru Çizme Algoritmaları…

- Doğru çizme algoritmalarında yaygın yaklaşım, doğruyu oluşturan pikseller için her bir sütuna (ya da satıra) tek bir pilsel olacak şekilde çizim yapılmasıdır. Fakat, tüm piksellerin aydınlatılması gereken durumlar da vardır.
- Aydınlatılacak olan piksellere karar verilirken dikkat edilecek en önemli nokta karar verme hızıdır.
  - Maksimum doğruluğu yakalayacak şekilde karar vermek ve bunu minimum sürede yapmak
- Karar verme sürecini etkileyen faktörler
  - Doğru, düzgün bir yol boyunca ilerlemeli
  - Doğru yerde başlayıp bitmeli
  - Doğrunun kalınlığı sabit olmalı
    - Kimi yerde kalınlaşıp kimi yerde incelmemeli
  - Hızlı çizilmeli

Doğru Çizme Algoritmaları…

- Doğrunun çakıştığı piksellerden hangisi/hangileri aydınlatılacak?
  - Her satır veya sütun boyunca bir piksel
  - Ekranda görüntülenecek olan piksel sayısı belirlenir ve hesaplanarak önceden belirlenen renk ile aydınlatılır.



Doğru Çizme Algoritmaları…

- **DDA(Digital Differential Analyzer) Algoritması**
  - En kolay doğru çizme algoritmalarından birisi
  - Doğruyu oluşturan ardışıl noktalar arasındaki x ve y koordinat farklarının hesaplanması prensibine dayanır.

$$\left.\begin{matrix}\Delta x = x_2 - x_1 \\ \Delta y = x_2 - x_1\end{matrix}\right\} piksel\_sayısı = \Delta x > \Delta y \ ? \ \Delta x : \Delta y$$

$$x_{fark} = {}^{\Delta x}/_{piksel\_sayısı}$$

$$y_{fark} = {}^{\Delta y}/_{piksel\_sayısı}$$

$$x_{n+1} = x_n + x_{fark}$$

$$y_{n+1} = y_n + y_{fark}$$

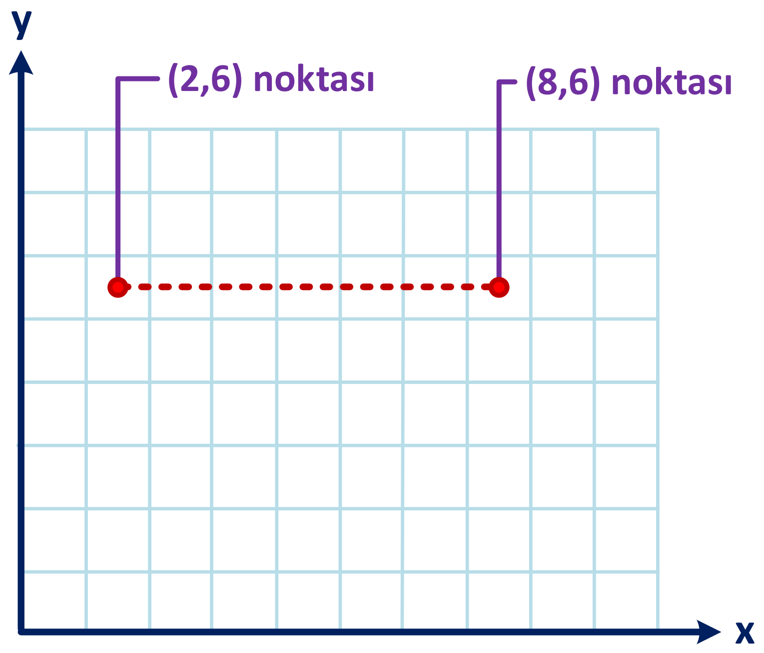
Doğru Çizme Algoritmaları…

- **DDA(Digital Differential Analyzer) Algoritması…**
  - ÖRNEK: (2,6) ve (8,6) noktalarını DDA algoritmasına göre çizersek aydınlatılacak olan noktalar?

$$\left.\begin{array}{l}\Delta x = 8 - 2 = 6 \\ \Delta y = 6 - 6 = 0\end{array}\right\} piksel\_sayısı = 6$$

$$x_{fark} = {}^{\Delta x}/_{piksel\_sayısı} = {}^{6}/_{6} = 1$$

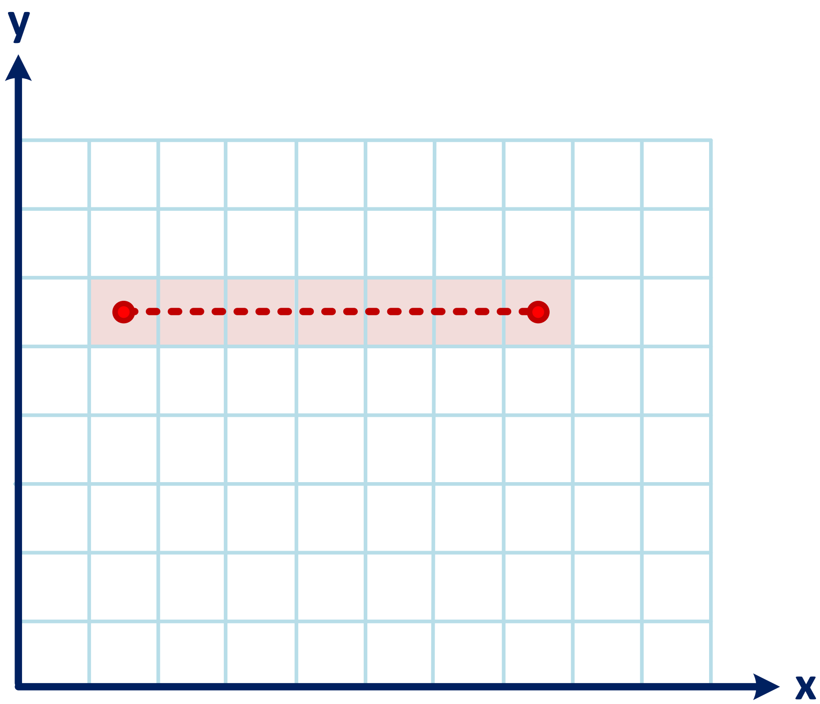
$$x_{fark} = {}^{\Delta y}/_{piksel\_sayısı} = {}^{0}/_{6} = 0$$



Doğru Çizme Algoritmaları…

DDA(Digital Differential Analyzer) Algoritması…

ÖRNEK:…



$x_0 = 2, y_0 = 6$

n	$x_{n+1} = x_n + x_{fark}$	$y_{n+1} = y_n + y_{fark}$
0	3	6
1	4	6
2	5	6
3	6	6
4	7	6
5	8	6

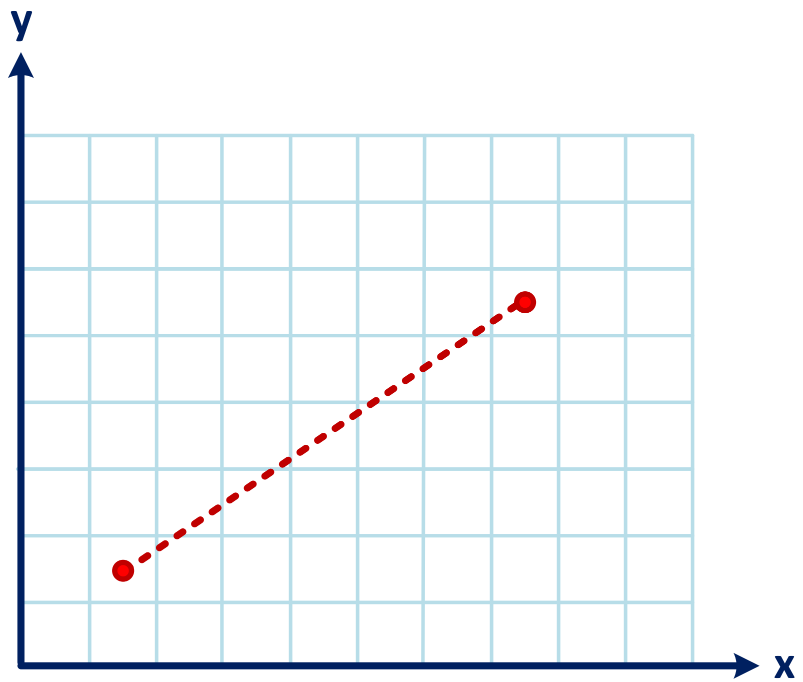
Doğru Çizme Algoritmaları…

- **DDA(Digital Differential Analyzer) Algoritması…**
  - <u>ÖRNEK:</u> (2,2) ve (8,6) noktalarını DDA algoritmasına göre çizersek aydınlatılacak olan noktalar?

$$\left.\begin{matrix}\Delta x = 8 - 2 = 6 \\ \Delta y = 6 - 2 = 4\end{matrix}\right\} piksel\_sayısı = 6$$

$$x_{fark} = \Delta x / _{piksel\_sayısı} = 6/_{6} = 1$$

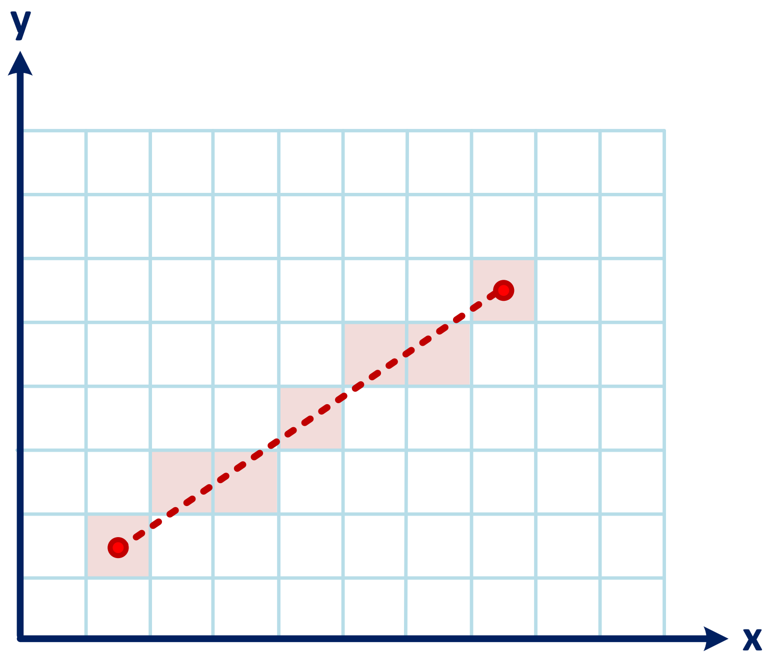
$$x_{fark} = \Delta y / _{piksel\_sayısı} = 4/_{6} \cong 0{,}66$$



Doğru Çizme Algoritmaları…

DDA(Digital Differential Analyzer) Algoritması…

ÖRNEK:…



$$x_0 = 2, y_0 = 2$$

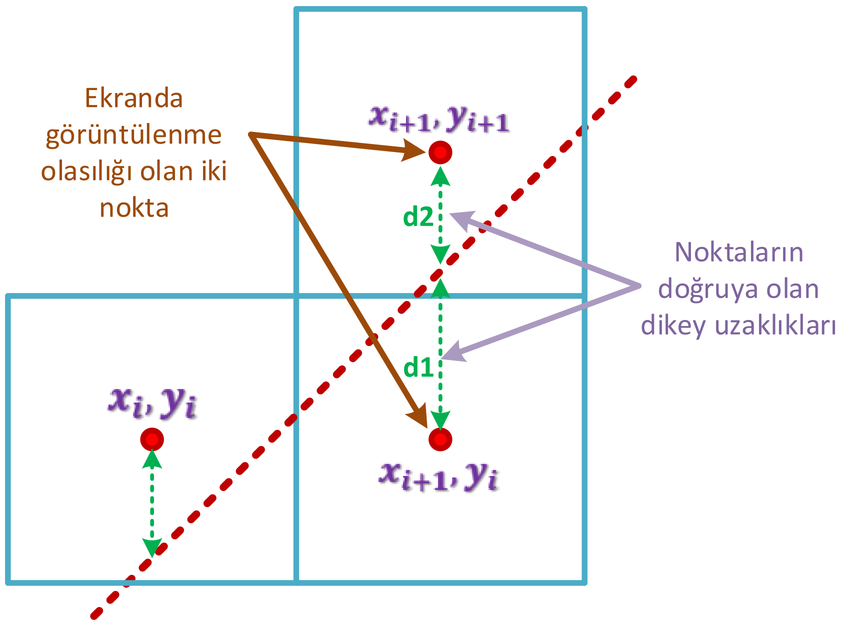
n	$x_{n+1} = x_n + x_{fark}$	$y_{n+1} = y_n + y_{fark}$
0	3	6
1	4	6
2	5	6
3	6	6
4	7	6
5	8	6

Doğru Çizme Algoritmaları…

- **DDA(Digital Differential Analyzer) Algoritması…**
  - Dezavantajları
    - Kayan noktalı sayılar ile işlem yapar
    - Girilen her doğru için bölme işlemi yapılır
  - Tamsayı işlemleri, kayan noktalı sayı işlemlerinden daha hızlı
  - Toplama ve çıkartma, çarpmadan hızlı
  - Çarpma, bölmeden hızlı
  - Bir sayıyı sıfır ile karşılaştırma, iki sayının karşılaştırılmasından daha hızlı

Doğru Çizme Algoritmaları…

- **Bresenham Doğru Çizme Algoritması**
  - 1965 yılında Jack Bresenham tarafından geliştirilmiştir
  - Tamsayılarla işlem yapar
  - Doğrunun bitim noktalarının birbirlerine göre olan konumlarına ve doğrunun eğimine bağlı olarak algoritmanın yapısı değişir.



Doğru Çizme Algoritmaları…

- **Bresenham Doğru Çizme Algoritması…**
  - Algoritma, x'in yatay ve y'nin dikey yönde arttığı koşul için yazılmıştır. Yani eğim 0 ile 1 arasındadır.

$x_1 < x_2$ ve $y_1 < y_2$

$$\left.\begin{array}{l}\Delta x = x_2 - x_1 \\ \Delta y = y_2 - y_1\end{array}\right\} \Delta x > \Delta y \Rightarrow m \in [0,1]$$

---

x, birer birer artar

y, bazen artar, bazen sabit

$$(x, y) \Rightarrow \left.\begin{array}{c} x+1, y \\ x+1, y+1 \end{array}\right\} doğruya\ olan\ dikey\ uzaklığı\ fazla\ olan\ seçilir.$$

Doğru Çizme Algoritmaları…

➢ Bresenham Doğru Çizme Algoritması…

$$y = mx + n \Longrightarrow y = m.(x_{i+1}) + n$$

$$d1 = y - y_i = m.(x_{i+1}) + n - y_i$$

$$d2 = y_{i+1} - y = y_{i+1} - m.(x_{i+1}) + n$$

$$d1 - d2 = m.(x_{i+1}) + n - y_i - (y_{i+1} - m.(x_{i+1}) + n\,)$$

$$\Rightarrow 2m.(x_{i+1}) - 2y_i + 2n - 1$$

$$\Rightarrow 2.{}^{\Delta y}/_{\Delta x}\,(x_{i+1}) - 2y_i + 2n - 1$$

$$\Delta x.(d1 - d2) = 2.\Delta y.(x_{i+1}) - 2y_i.\Delta x + \Delta x(2n - 1)$$

$$P_i = \Delta x.(d1 - d2) = 2.\Delta y.x_i - 2y_i.\Delta x + 2.\Delta y + \Delta x(2n - 1)$$

Doğru Çizme Algoritmaları…

Bresenham Doğru Çizme Algoritması…

$P_i = 2.\Delta y.x_i - 2.\Delta x.y_i + c$

$P_{i+1} = 2.\Delta y.x_{i+1} - 2.\Delta x.y_{i+1} + c$

$P_{i+1} - P_i = 2.\Delta y.(x_{i+1} - x_i) - 2.\Delta x.(y_{i+1} - y_i)$

$\Rightarrow (x_{i+1} - x_i) = 1$ (x birer birer artar)

$\Rightarrow (y_{i+1} - y_i) = 0 \mid 1$ (y artar, ya da sabit kalır)

$P_i < 0 \Rightarrow P_{i+1} = P_i + 2.\Delta y$ (d1 $< d2$, yani y seçilirse)

$P_i > 0 \Rightarrow P_{i+1} = P_i + 2.\Delta y + 2.\Delta x$ ($d2 < d1$, yani y+1 seçilirse)

Doğru Çizme Algoritmaları…

Bresenham Doğru Çizme Algoritması…

$$P_0 = 2.\Delta y.x_0 - 2.\Delta x.y_0 + 2.\Delta y + \Delta x(2n - 1)$$

$$y = mx + n \Rightarrow n = y_0 - {}^{\Delta y}/_{\Delta x}.x_0$$

$$P_0 = 2.\Delta y.x_0 - 2.\Delta x.y_0 + 2.\Delta y + \Delta x\left(2.y_0 - 2.{}^{\Delta y}/_{\Delta x}.x_0 - 1\right)$$

$$\Rightarrow \cancel{2.\Delta y.x_0} - \cancel{2.\Delta x.y_0} + 2.\Delta y + 2.\Delta \cancel{x.y_0} - \cancel{2.\Delta y.x_0} - \Delta x$$

$$P_0 = 2.\Delta y - \Delta x$$

Doğru Çizme Algoritmaları…

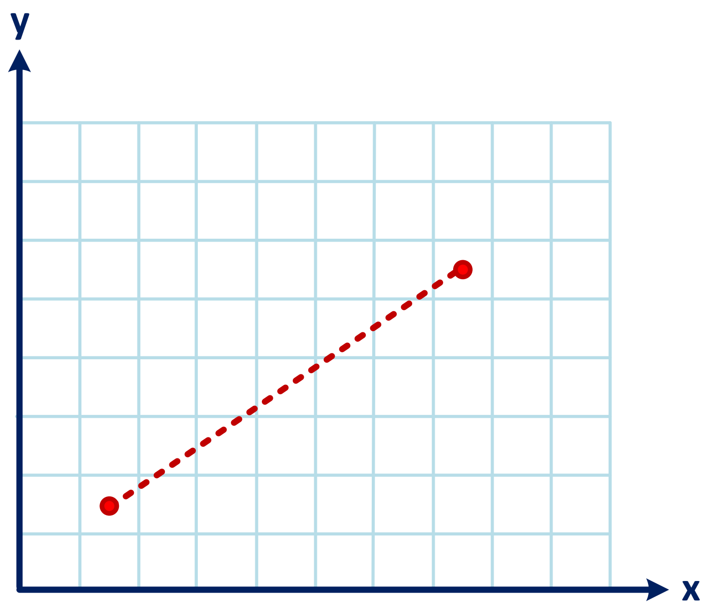
➢ DDA(Digital Differential Analyzer) Algoritması…

➢ <u>ÖRNEK:</u> (2,2) ve (8,6) noktalarını Bresenham algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?

$\Delta x = 8 - 2 = 6 \Rightarrow 2\,\Delta x = 12$

$\Delta y = 6 - 2 = 4 \Rightarrow 2\,\Delta y = 8$

$x_0 = 2, y_0 = 2$



Doğru Çizme Algoritmaları…

➢DDA(Digital Differential Analyzer) Algoritması…

➢ÖRNEK: (2,2) ve (8,6) noktalarını Bresenham algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?

$P_0 = 2.\Delta y - \Delta x = 4 - 6 = 2 > 0 \Rightarrow x_1 = 3, y_1 = 3$

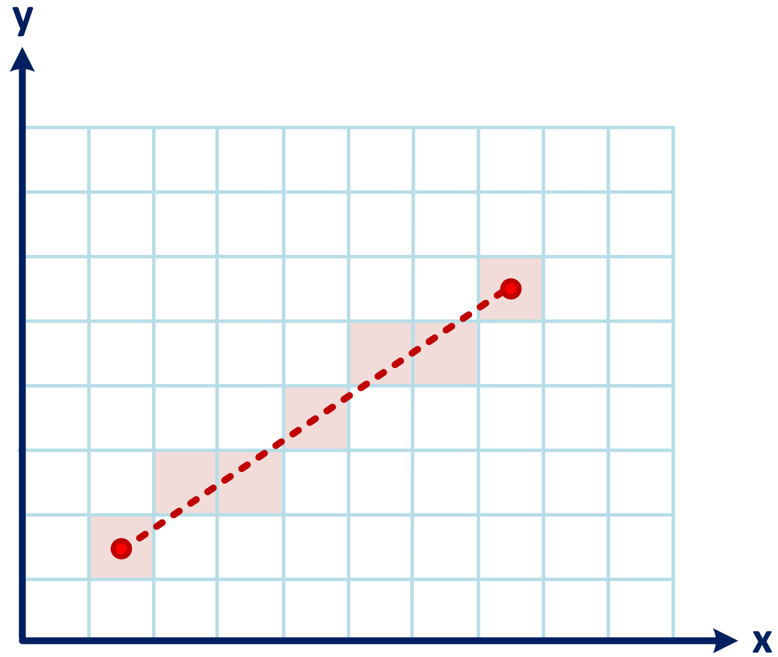
$P_1 = P_0 + 2.\Delta y - 2.\Delta x = -2 < 0 \Rightarrow x_2 = 4, y_2 = 3$

$P_2 = P_1 + 2.\Delta y = 6 > 0 \Rightarrow x_3 = 5, y_3 = 4$

$P_3 = P_2 + 2.\Delta y - 2.\Delta x = 2 > 0 \Rightarrow x_4 = 6, y_4 = 5$

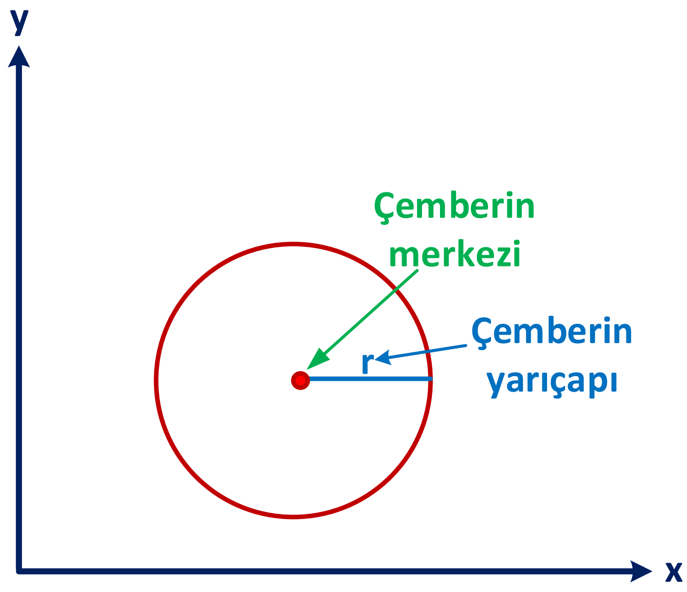
$P_4 = P_3 + 2.\Delta y - 2.\Delta x = -2 < 0 \Rightarrow x_5 = 7, y_5 = 5$

$P_5 = P_4 + 2.\Delta y = 6 > 0 \Rightarrow x_6 = 8, y_6 = 6$



Çember Çizme Algoritmaları…

- **Çember**
  - Sabit bir noktadan aynı uzaklıkta ve aynı düzlemdeki noktalar kümesinin oluşturduğu kapalı eğri



Çember Çizme Algoritmaları…

- **Bresenham Çember Çizme Algoritması**
  - Piksel-tabanlı çember çizme algoritması
  - Pisagor teoremi – Çemberin kapalı gösterimi
    - $f(x,y) = (x - {x_0}^2) + (y - {y_0}^2) - r^2$=0
  - Çemberin üzerinde bulunan tüm noktalar bu fonksiyonu sağlar
  - Çemberin içinde bulunan noktalar için çemberin kapalı fonksiyonu negatif, dışında bulunan noktalar için pozitif
  - Çemberin merkezi orjin(0,0) olduğu varsayılırsa, çemberin üzerindeki bir noktada olma(ma)sı
    - $hata(x,y) = x^2 + y^2 + r^2$

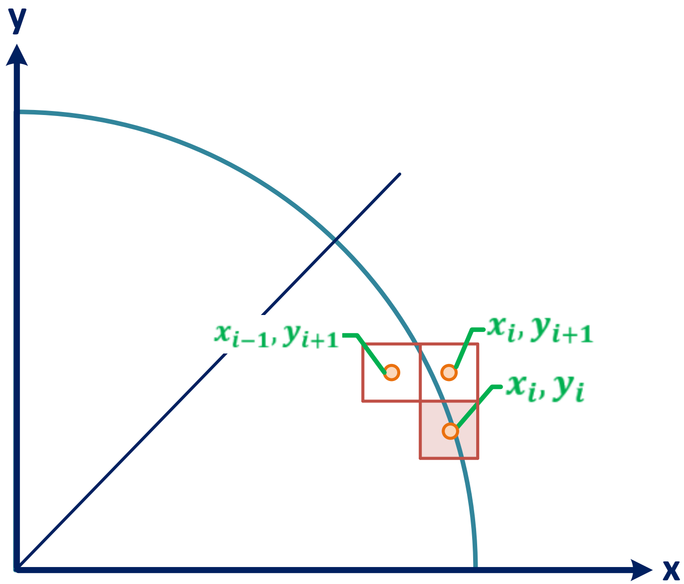
Çember Çizme Algoritmaları…

- **Bresenham Çember Çizme Algoritması…**
  - Bresenham doğru çizme algoritmasında ekranda görüntülenecek olan piksellerin değeri bir hata değerine bağlı olarak yapılan hesaplamadakine benzer mantık Bresenham çember algoritmasında da kullanılır.
  - (x_0, y_0) merkezli, *r* yarıçaplı çemberi Bresenham çember algoritması ile çizmek için;
    1. Çemberin merkezi orjine taşınır.
    2. Orjinde(0,0) r yarıçaplı çemberin 0-45° arasında kalan dilimi için piksel koordinatları hesaplanır
    3. Çemberin simetri özelliğinden dolayı diğer 7 dilim, 0-45° arasında kalan dilimin piksel koordinatları kullanılarak hesaplanır.
    4. Orjindeki çember, (x_0, y_0) noktasında geri ötelenir (Hesaplanan her piksel değerine (x_0, y_0) eklenir.)

Çember Çizme Algoritmaları…

Bresenham Çember Çizme Algoritması…

- Hesaplanan yeni hata değerinden mutlak değeri küçük olan piksel seçilir, ekranda görüntülenir ve hata değeri güncellenir.



$$hata(x, y) = x^2 + y^2 + r^2$$

$$hata(x, y+1) = x^2 + (y+1)^2 - r^2$$
$$\Rightarrow hata(x, y) + 2y + 1$$

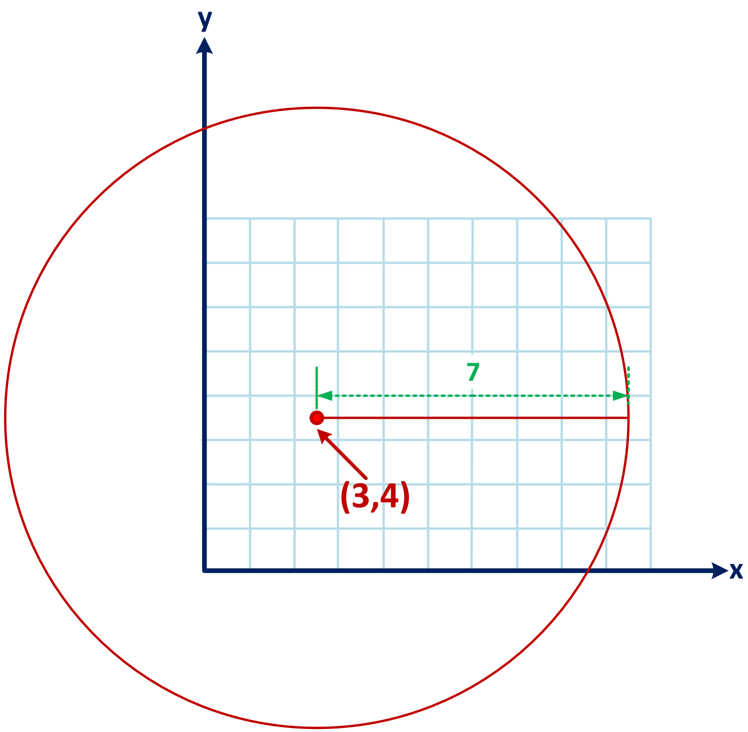
$$hata(x-1, y+1) = (x-1)^2 + (y+1)^2 - r^2$$
$$\Rightarrow hata(x, y) + 2y + 1 - (2x - 1)$$

$$hata(r, 0) = 0$$

Çember Çizme Algoritmaları…

➢ Bresenham Çember Çizme Algoritması…

- ➢ ÖRNEK: Merkezi (3,4) noktasında ve yarıçapı 7 birim olan çemberi Bresenham çember çizme algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?



Çember Çizme Algoritmaları…

➢ Bresenham Çember Çizme Algoritması…

➢ ÖRNEK: Merkezi (3,4) noktasında ve yarıçapı 7 birim olan çemberi Bresenham çember çizme algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?

1. Çemberin merkezini orjine taşı
2. Orjinde(0,0) ve7 birim yarıçaplı çemberin 0-45° arasında kalan dilimi için piksel koordinatları hesapla
3. Çemberin simetri özelliğinden dolayı diğer 7 dilimi, 0-45° arasında kalan dilimin piksel koordinatları kullanılarak hesapla
4. Orjindeki çemberi, $(3,4)$ noktasında geri ötele

Çember Çizme Algoritmaları…

Bresenham Çember Çizme Algoritması…

- <u>ÖRNEK:</u> Merkezi (3,4) noktasında ve yarıçapı 7 birim olan çemberi Bresenham çember çizme algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?

$$(7,0) \Rightarrow hata = x^2 + y^2 - r^2 = 0$$

$$\left.\begin{array}{l} hata1 = hata + 2y + 1 = 1 \\ hata2 = hata + 2y + 1 - (2x - 1) = -12 \end{array}\right\} |1| < |-12| \Rightarrow (7,1)$$

$$hata = 1$$

$$\left.\begin{array}{l} hata1 = hata + 2y + 1 = 4 \\ hata2 = hata + 2y + 1 - (2x - 1) = -9 \end{array}\right\} |4| < |-9| \Rightarrow (7,2)$$

$$hata = 4$$

Çember Çizme Algoritmaları…

➢ Bresenham Çember Çizme Algoritması…

- ➢ ÖRNEK: Merkezi (3,4) noktasında ve yarıçapı 7 birim olan çemberi Bresenham çember çizme algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?

$$\left.\begin{aligned} hata1 &= hata + 2y + 1 = 9 \\ hata2 &= hata + 2y + 1 - (2x - 1) = -4 \end{aligned}\right\} |-4| < |9| \Rightarrow (6,3)$$

$$hata = -4$$

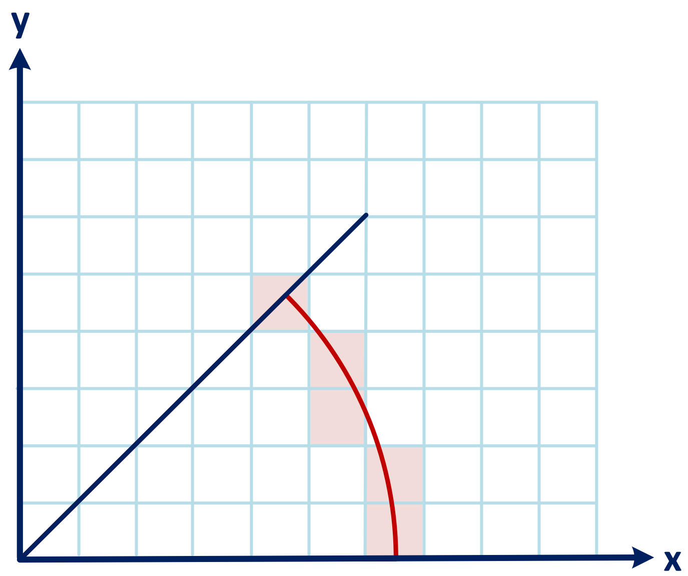
$$\left.\begin{aligned} hata1 &= hata + 2y + 1 = 3 \\ hata2 &= hata + 2y + 1 - (2x - 1) = -8 \end{aligned}\right\} |3| < |-8| \Rightarrow (6,4)$$

$$hata = 3$$

$$\left.\begin{aligned} hata1 &= hata + 2y + 1 = 1 \\ hata2 &= hata + 2y + 1 - (2x - 1) = 1 \end{aligned}\right\} |1| < |12| \Rightarrow (5,5)$$

Çember Çizme Algoritmaları…

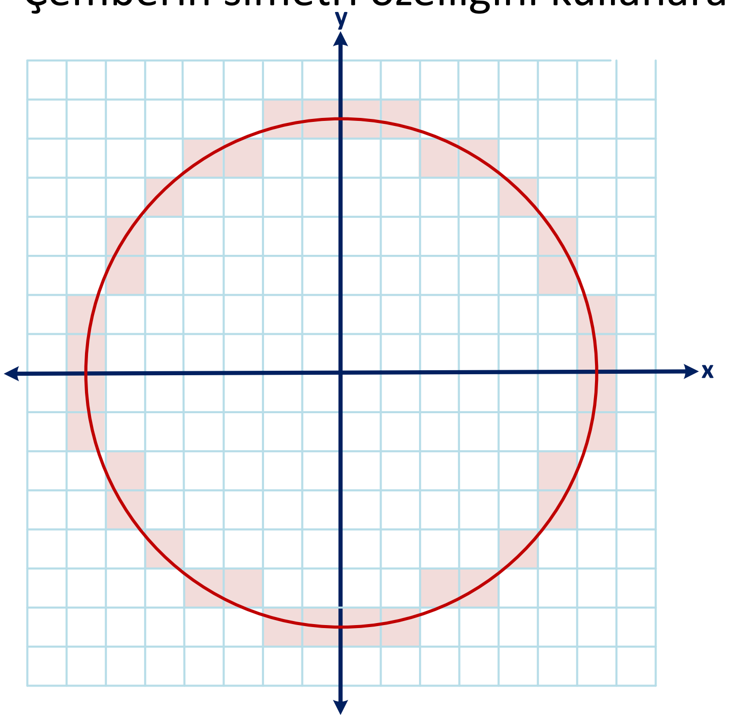
- **Bresenham Çember Çizme Algoritması…**
  - ÖRNEK: Merkezi (3,4) noktasında ve yarıçapı 7 birim olan çemberi Bresenham çember çizme algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?



Çember Çizme Algoritmaları…

Bresenham Çember Çizme Algoritması…

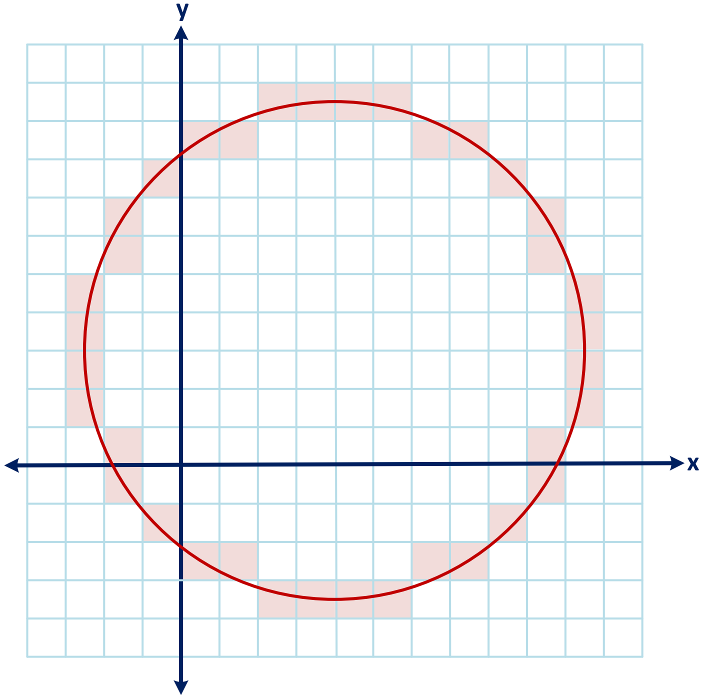
- ÖRNEK: Merkezi (3,4) noktasında ve yarıçapı 7 birim olan çemberi Bresenham çember çizme algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?
  - Çemberin simetri özelliğini kullanarak diğer 7 dilimin koordinatları hesaplanır.



Çember Çizme Algoritmaları…

Bresenham Çember Çizme Algoritması…

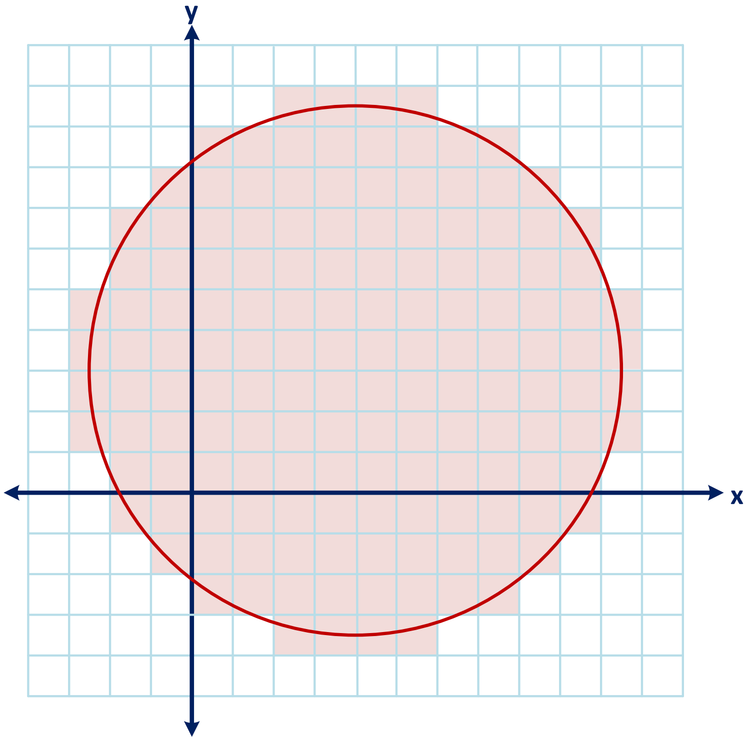
- **ÖRNEK:** Merkezi (3,4) noktasında ve yarıçapı 7 birim olan çemberi Bresenham çember çizme algoritmasına göre çizersek aydınlatılacak olan noktalar hangileridir?
  - Orjindeki çember $(3,4)$ noktasına geri ötelenir.



Daire Çizme Algoritması

- **Daire**
  - Çember tarafından sınırlandırılan düzlemsel alan
  - (x_0, y_0) merkezli, *r* yarıçaplı daireyi çizmek için;
    - Çember çizme algoritması kullanılarak çembere ait pikseller belirlenir
    - Çemberin içinde kalan pikseller belirlenir

$$
\begin{aligned}
&for\ (y = 1; y \leq r; y++)\\
&\quad for\ (y = 1; y \leq r; y++)\\
&\qquad if\ (x^2 + y^2 < r^2)\\
&\qquad\quad piksel\ (x_0 + x, y_0 + y);
\end{aligned}
$$



Elips Çizme Algoritmaları

- **Elips**
  - Çemberin x veya y eksenlerinde ölçeklendirilmiş, uzatılmış, daraltılmış, esnetilmiş hali
  - Bir merkez ve iki odak noktasından oluşur.
  - Odak noktalarından elips üzerinde çizilecek herhangi bir noktaya çizilecek olan iki doğrunun uzunlukları toplamı sabit
  - Merkezi (x_0, y_0), asal eksen yarıçapı r_x ve yedek eksen yarıçapı r_y olan elipsin kapalı denklemi;
    - $\left(\frac{x-x_0}{r_x}\right)^2 + \left(\frac{y-y_0}{r_y}\right)^2 = 1$

Elips Çizme Algoritmaları…

- **Bresenham Elips Çizme Algoritması**
  - Piksel-tabanlı elips çizme algoritması
  - Elipsin merkezi orjin(0,0) olduğu varsayılırsa, elipsin üzerindeki bir noktada olma(ma)sı
    - $hata(x,y) = {r_x}^2 {r_y}^2 - {r_x}^2 y^2 - {r_y}^2 x^2$

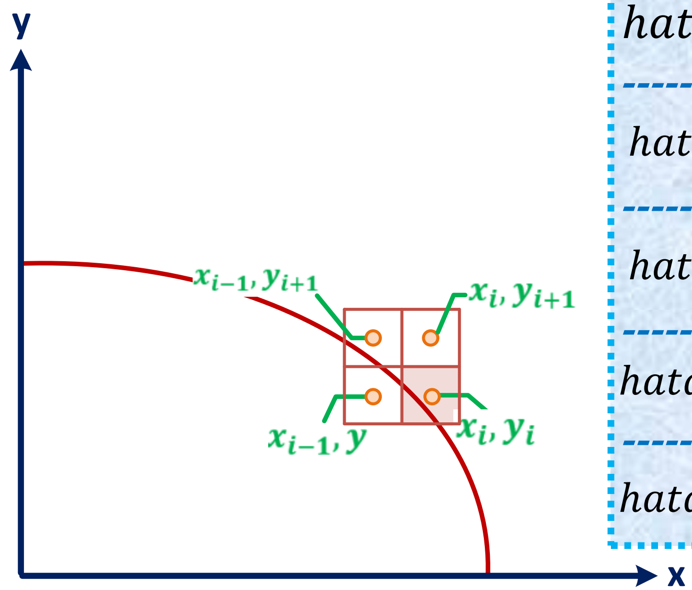
Elips Çizme Algoritmaları…

- **Bresenham Elips Çizme Algoritması…**
  - Bresenham çember çizme algoritmasında ekranda görüntülenecek olan piksellerin değeri bir hata değerine bağlı olarak yapılan hesaplamadakine benzer mantık Bresenham elips algoritmasında da kullanılır.
  - (x_0, y_0) merkezli, *r* yarıçaplı elipsi Bresenham çember algoritması ile çizmek için;
    1. Elipsin merkezi orjine taşınır.
    2. Orjinde(0,0), asal eksen yarıçapı r_x ve asal eksen yarıçapı r_y olan elipsin 0-90° arasında kalan dilimi için piksel koordinatları hesaplanır
    3. Elipsin simetri özelliğinden dolayı diğer 3 dilim, 0-90° arasında kalan dilimin piksel koordinatları kullanılarak hesaplanır.
    4. Orjindeki elips, (x_0, y_0) noktasında geri ötelenir (Hesaplanan her piksel değerine (x_0, y_0) eklenir.)

Elips Çizme Algoritmaları…

➢ Bresenham Elips Çizme Algoritması…

- ➢ Hesaplanan yeni hata değerinden mutlak değeri küçük olan piksel seçilir, ekranda görüntülenir ve hata değeri güncellenir.



$$hata(x,y) = {r_x}^2{r_y}^2 - {r_x}^2y^2 - {r_y}^2x^2$$

$$hata(x-1,y) = hata(x,y) + 2{r_y}^2x - {r_y}^2$$

$$hata(x-1,y+1) = hata(x,y) + 2{r_y}^2x - 2{r_x}^2y - {r_x}^2 - {r_y}^2$$

$$hata(x,y+1) = hata(x,y) - 2{r_x}^2y - {r_x}^2$$

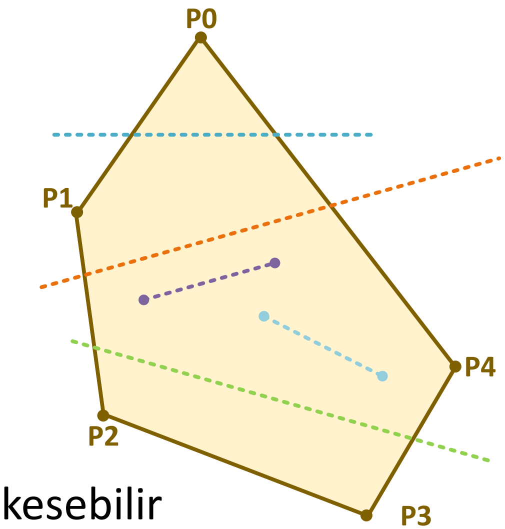
$$hata(r_x,0) = 0$$

Poligon Doldurma Algoritmaları

- Poligonsal nesneleri görüntülemek için kullanılırlar
- Görüntüleme işlemi, hızlı ve doğru olmalı
  - Yanlış olursa;
    - Pikseller yanlış yere basılır
    - Pikseller arası boşluk oluşur
    - Pikseller üst üste basılı, vs.
  - Hızlı değilse;
    - Grafik uygulaması yavaşlar
- İki temel poligon sınıfı var
  - Basit poligonlar
  - Genel poligonlar

Poligon Doldurma Algoritmaları…

- **Basit Poligonlar**
  - Kenarları birbirleriyle kesişmez
  - İçerisinde boşluk bulundurmaz
  - Her köşe iki kenar tarafından paylaşılır
  - İçbukey veya dışbükey



  - **Dışbukey Poligonlar**
    - Herhangi bir doğru dışbükey poligonu en fazla iki yerden kesebilir
    - İçerisinde bulunan herhangi iki noktayı birleştiren doğru poligonun dışına çıkamaz
    - Tüm iç açıları 180° den küçüktür.
  - İçbukey poligonlar bu özelliklerden hiç birisini sağlamaz.

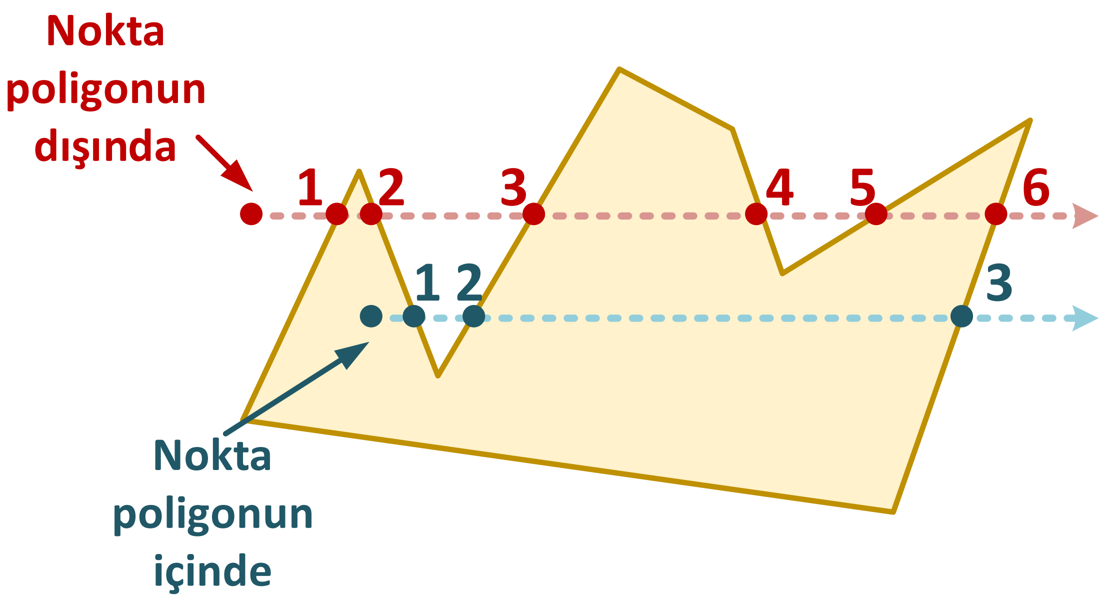
  - NOT: Bir çok donanım, temel grafik öğesi poligonu olarak üçgenleri kullanır. Bu durumda, köşe sayısı üçten büyük olan poligonlar ilk önce üçgenlerle belirlenir.

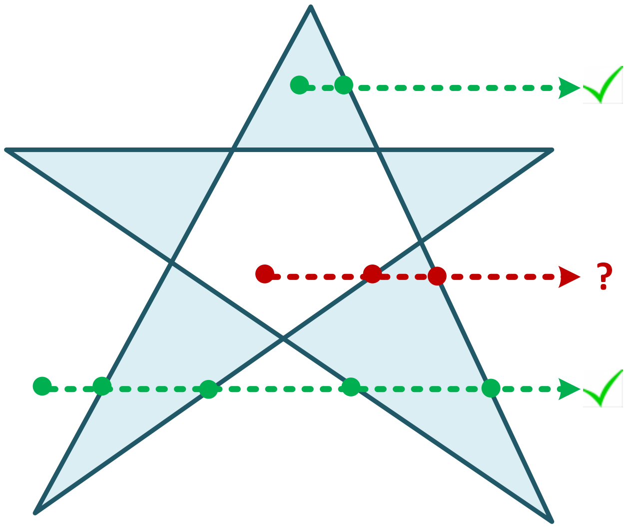
Poligon Doldurma Algoritmaları…

- **İç-Dış Testleri**
  - Verilen bir noktanın bir poligonun içerisinde olup olmadığını belirlemek için kullanılır.
  - Jordan Eğri Teoremi
    - En yaygın kullanılan iç-dış testi algoritmasıdır.
    - Tek-çift kuralı olarak da adlandırılır.
    - Poligonun içinde bulunan bir noktadan herhangi bir yöne doğru gönderilen ışın, her zaman tek sayıda kenarı keser.
    - Eğer, ışın çift sayıda kenarı kesiyorsa nokta poligonun dışındadır.
    - İçbukey, dışbükey ya da içerisinde boşluklar barındıran poligonlar için düzgün sonuçlar verir.
    - Kenarları birbirleriyle kesişen poligonlar için her zaman istenilen sonucu vermeyebilir.

Poligon Doldurma Algoritmaları…

İç-Dış Testleri…





KAYNAKLAR

- Atılım Çetin, Bilgisayar Grafikleri, Seçkin Yayıncılık, 2003.