Doğru denklemi

Düz bir çizgi için eğim kesişim denklemi aşağıdaki gibidir.

$$y = mx + b \tag{1}$$

m doğrunun eğimini b ve y ise kesişimi temsil eder. Bir doğrunun iki bitiş noktası (x_1, y_1) ve (x_2, y_2) ile gösterilir. m ve y değerlerini aşağıdaki gibi hesaplayabiliriz.

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{2}$$

$$b = y_1 - mx_1 \tag{3}$$

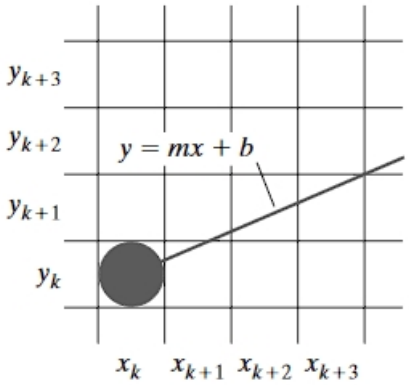
Çizgi çizme algoritmaları yukarıdaki denklemler üzerine kuruludur.
x ve y farkını $\Delta x = x_2 - x_1$, $\Delta y = y_2 - y_1$ ile göstererek denklemleri tekrar düzenlersek eğer.

$$\Delta y = m\Delta x \tag{4}$$

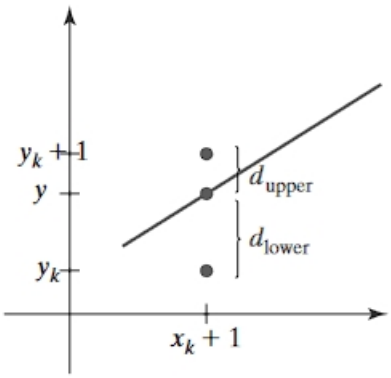
$$\Delta x = \frac{\Delta y}{m} \tag{5}$$

Bresenham Çizgi Algoritmasını Türetmek

Kayan noktalı aritmetik kullanmadığı ve yalnızca tamsayılarla işlem yaptığı için oldukça hızlı bir algoritmadır. Algoritmanın ilk halinde eğimi birden küçük olan doğrularla çalışacağız. Birim x aralıklarında ilerleyerek doğrunun üzerinde bulunduğu noktaları belirleyeceğiz. Verilen bir doğru için sol tarafta kalan (x_0, y_0) noktasından başlayacağız, ardışık gelen her bir sütunda (x konumunda) ilerleyeceğiz ve gerçek doğruya en yakın olan y değerine sahip pikseli yerleştireceğiz.
Amacımız başlangıçta (x_k, y_k) konumunda bulunuyorsak eğer, x ekseninde bir adım ilerlerken y ekseninde bir adım yukarıya mı pikseli yerleştirmeliyim $(x_k + 1, y_k + 1)$ yoksa aynı yüksekliğe mi $(x_k + 1, y_k)$ sorusuna cevap bulmaya çalışmak. Neden böyle bir şey yapmaya çalıştığımızı daha iyi anlayabilmeniz için resimi inceleyelim.



Yuvarlak ile gösterilen kısmı başlangıç pikseli olarak düşünecek olursanız, istediğimiz şey bu yuvarlak pikselin devam eden çizgiye olabildiğince yakın olması -çizgi aslında orada yok ve o çizginin adı esas çizgi- fakat x ekseninde birer adım ilerlediğinizde gördüğünüz üzere çizgi aynı sütunda hem y_k hem y_{k+1} satırlarından -yüksekliklerinden- geçiyor. Bu nedenle pikselin bir adım yukarıya mı yoksa aynı yüksekliğe mi koyulması gerektiği konusu önemli bir konu, eğer doğrumuz gerçeğe en yakın doğru olsun istiyorsak.



Bu kararı verebilmek için, gerçek doğrunun bir alttaki ve bir üstteki koordinatlara olan uzaklığını öncelikle d_{alt} ve d_{ust} olarak resimdeki gibi temsil etmemiz gerekiyor. Tahmin edebileceğiniz üzere alttaki y koordinatına daha yakınsa $d_{alt} > d_{ust}$, üsttekine daha yakınsa $d_{alt} < d_{ust}$.

Yatay eksende bir birim ilerledikten sonra koyulacak piksel koordinatının denklemi:

$$y = m(x_k + 1) + b \tag{6}$$

O halde,

$$\begin{aligned} d_{alt} &= y - y_k && (7)\\ &= m(x_k + 1) + b - y_k && (8)\\ d_{ust} &= (y_k + 1) - y && (9)\\ &= y_k + 1 - m(x_k + 1) + b && (10) \end{aligned}$$

Esas çizgiye hangi pikselin daha yakın olduğunu belirlemek için $d_{alt} - d_{ust}$ ifadesine bakarız, eğer alt uzunluk üst uzunluktan daha büyükse bu ifade sıfırdan büyük -pozitif- olacaktır, değilse sıfırdan küçük -negatif- olacaktır.

$$d_{alt} - d_{ust} = 2m(x_k + 1) - 2y_k + 2b - 1 \tag{11}$$

Fakat şu anda yukarıdaki (11) denklemde bulunan eğim $m = \frac{\Delta y}{\Delta x}$ değeri nedeniyle halen kayan noktalı aritmetik yapmaktayız, bu kesirli sayıdan kurtulmak için m değerini yerine koyup denklemin her iki tarafını da Δx ile çarpacağız.

$$\Delta x(d_{alt} - d_{ust}) = \Delta x(2\frac{\Delta y}{\Delta x}(x_k + 1) - 2y_k + 2b - 1) \tag{12}$$

Denklemin sol tarafına p_k -karak parametresi- dersek ve sağ tarafındaki çarpımı dağıtırsak aşağıdaki sonuca ulaşırız.

$$\begin{aligned} p_k &= \Delta x(d_{alt} - d_{ust}) \\ &= 2\Delta y x_k - 2\Delta x y_k + c && (13) \end{aligned}$$

Δx ifadesi bu örnekte daima sıfırdan büyük olduğu için, p_k işareti $d_{alt} - d_{ust}$ ifadesine bağlıdır. c ise denklemde bir değişken bulunmayan bölümü tarif eder ve değeri $2\Delta y + \Delta x(2b - 1)$ gibidir.

Eğer ki y_k konumunda bulunan piksel esas doğruya $y_k + 1$ konumundan daha yakınsa, - yani $d_{alt} < d_{ust}$ karar parametresi p_k negatiftir. Bu durumda aşağıdaki piksele, tersi durumunda yukarıdaki piksele nokta koyacağız.

Çizgiyi oluşturmak için ilerledikçe koordinatlar değiştiği için, ardışık karar parametrelerine yani rekürsif bir tanıma ihtiyacımız var. Burada yapacağımız şey bir sonraki karar parametresini bir önceki karar parametresinden hesaplamak. k+1 adımındaki karar parametresini 13. denklemden elde edelim.

$$p_{k+1} = 2\Delta y x_{k+1} - 2\Delta x y_{k+1} + c \tag{14}$$

Şimdi bu denklemden 13. denklemi çıkartalım.

$$p_{k+1} - p_k = 2\Delta y(x_{k+1} - x_k) - 2\Delta x(y_{k+1} - y_k) \tag{15}$$

15. denklemde denklemin sağ tarafında bulunan $(x_{k+1} - x_k)$ ifadesi bire eşittir, çünkü x ekseninde birer birim ilerliyoruz o nedenle gelecek koordinat ile şimdiki koordinatın farkı bir olacaktır, buna göre denklemi tekrar düzenlersek eğer.

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x(y_{k+1} - y_k) \tag{16}$$

16. denklemde denklemin sağ tarafında bulunan $(y_{k+1} - y_k)$ ifadesi ise, y ekseninde ya bir adım ilerleyeceğimiz ya da hiç gitmeyeceğimiz için 1 ve 0 değerlerinden birisini alabilir. Bu da aslında karar parametresinin işaretine bağlıdır. Şimdi başlangıç karar parametresi olan p_0 hesaplamalıyız. x_0, y_0 başlangıç noktasını, c sabitini ve en başta belirttiğim c sabitinde bulunan b değerini 13. denklemde yerine koymalıyız. Önce 13. denklemin açık halini yazalım ve ilk noktayı yerine koyalım.

$$p_0 = 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \Delta x(2b - 1) \tag{17}$$

Şimdi 3. denklemde belirttiğimiz b değerini ve eğim değerini $m = \frac{\Delta y}{\Delta x}$ yerine koyalım ve sonrasında çarpım dağıtma işlemlerini gerçekleştirelim.

$$\begin{aligned} p_0 &= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + \Delta x[2(y_0 - \frac{\Delta y}{\Delta x} x_0) - 1] \\ &= 2\Delta y x_0 - 2\Delta x y_0 + 2\Delta y + [2\Delta x y_0 - 2\Delta y x_0 - \Delta x] \\ &= 2\Delta y - \Delta x \end{aligned} \tag{18}$$

Her şey hazır, şimdi algoritmayı görebiliriz.

Bresenham Çizgi Algoritması

1. Başlangıç noktası (x_0, y_0) olan iki giriş al.
2. $\Delta x, \Delta y, 2\Delta y, 2\Delta y - 2\Delta x$ sabitlerini hesapla ve başlangıç karar parametresini $p_0 = 2\Delta y - \Delta x$ hesapla.
3. k=0'dan başlayarak, çizgi boyunca her bir x_k için karar parametresinin değerini test et, eğer negatifse yani $p_k < 0$ pikselin koyulacağı koordinat $(x_k + 1, y_k)$:

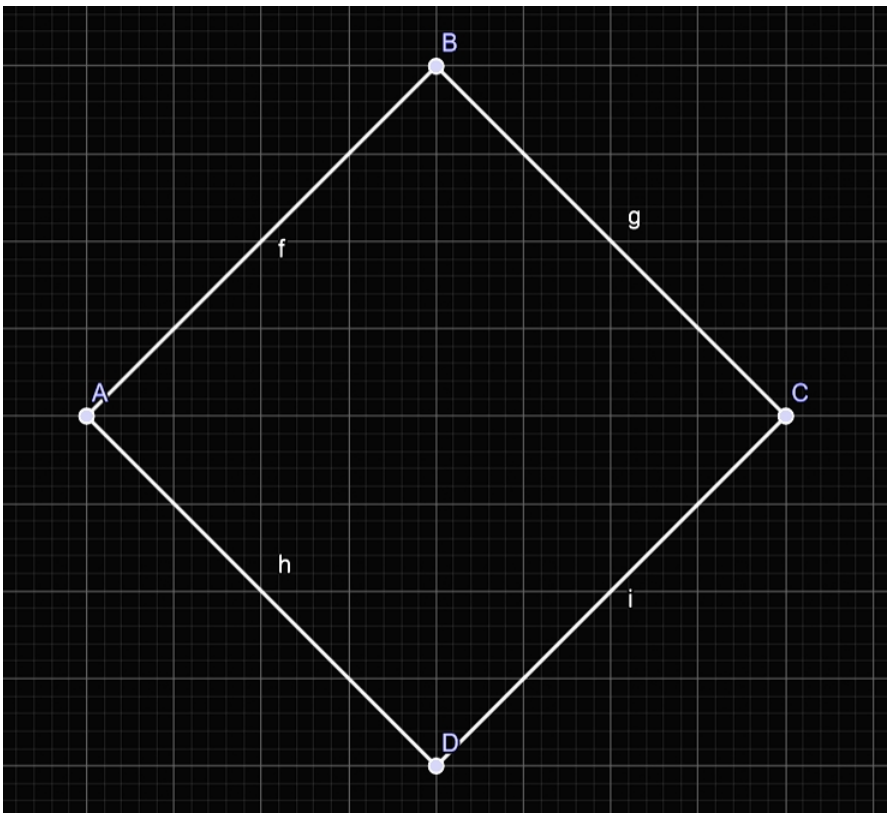
$$p_{k+1} = p_k + 2\Delta y$$

Eğer pozitifse pikselin koyulacağı koordinat $(x_k + 1, y_k + 1)$:

$$p_{k+1} = p_k + 2\Delta y - 2\Delta x$$

4. 3. adımı $\Delta x - 1$ kere daha tekrarla.

Burada dikkat edilmesi gereken nokta, bu algoritma yalnızca soldan sağa ve yukarıdan aşağı yönlü bir doğru çizmek için uygun, bu durumlar ufak modifikasyonlarla giderilebilir. Bir diğer husus ise, eğimi birden büyük olan doğruları çizebilmek için iki seçenek bulunuyor, ya x ve y koordinatlarını birbirleri ile değiştirmelisiniz ya da daha basit bir algoritma olan DDA algoritmasını eğim birden büyük olduğu zaman kullanmalısınız. Dikkat edilmesi gereken diğer bir nokta ise, bilgisayar ekranlarında x ve y koordinatları sol en üst köşeden sağ en üst köşeye doğru büyümekte, bunu da göz önünde bulundurmak zorundayız. Şimdi aşağıdaki figürle doğrunun başlangıç ve bitiş noktalarının bulunabileceği konumları ve bunu nasıl manipüle ederek algoritmayı tamamlayabileceğimizi düşünelim.



Resimde B noktasından A ve C noktalarına, aşağı yönlü uzanan doğru parçaları ve D noktasından A ve C noktalarına yukarı yönlü uzanan doğru parçaları görüyorsunuz. Bu durumları ele alacağız. Başlangıç noktası (x_0, y_0) ve bitiş noktası (x_1, y_1) ile temsil edilecek. Bunları ele alırken aklımızdan çıkartmamamız gerkeen iki şey var
1. Algoritma başlangıç noktasını (x_0, y_0) ile alıyor ve soldan sağa, aşağıdan yukarı doğru adım adım ilerliyor.
2. Bilgisayar ekranı sol en üst köşede (0,0) koordinatına sahip, aşağı ve sağa doğru ilerledikçe koordinat değerleri artıyor.

BA çizgisini ele aldığımızda, başlangıç noktasının x koordinatının, bitiş noktasının x koordinatına göre daha sağda kaldığını görüyoruz, bu durum istenmediği için noktalar arasında değişiklik yapacağız. Bu arada BA çizgisinde y ekseninin koordinatlarını aşağı yönlü yerleştirirken y ekseninde ilerlediğimizi unutmayalım, yani y değerlerini artırmalıyız.
BC çizgisini ele aldığımızda, başlangıç noktasının x koordinatı, bitiş noktasının x koordinatına göre daha geride olduğu için bir sorun yok fakat y ekseninin koordinat değerlerinin artacağını unutmamalıyız.

DA çizgisini ele aldığımızda, başlangıç noktasının x koordinatı, bitiş noktasının x koordinatına göre daha geride bu nedenle noktaları değiştirmeliyiz. Başlangıç noktasının y koordinatı ise bitiş noktasından daha fazla olduğu için, y ekseninde pikselleri yerleştirirken gerilememiz yani y değerlerini azaltmamız gerekiyor.

DC çizgisini ele aldığımızda, x ekseni koordinatları açısından herhangi bir sıkıntısı yok, y ekseninde değerleri azaltmalıyız.

Buradan çıkartacağımız iki sonuç var, başlangıç noktaları bitiş noktalarından geride ise noktaların yerini değiştir, yer değiştirdikten sonra çizgi y ekseninde aşağıya doğru ilerliyor ise y eksenindeki değerleri arttır, yukarıya doğru ilerliyorsa y değerlerini azalt.
Eğimin birden büyük olduğu durum içinse DDA algoritmasını kullanabilirsiniz.