2.1 Introduction to computer algorithm

Before getting started in computer graphics algorithms we need to know what the term *'computer algorithm*' means.

An algorithm is any well-defined computational procedure that takes some values as input and produces some values as output. Like a cooking recipe, an algorithm provides a step-by-step method for solving a computational problem. Unlike programs, algorithms are not dependent on a particular programming language, machine, system, or compiler (usually described using natural language and pseudo code). They are mathematical entities, which can be thought of as running on some sort of idealized computer with an infinite random access memory and an unlimited word size. Algorithm design is all about the mathematical theory behind the design of good programs. The process of writing an algorithm is just taking your problem, and breaking it down, abstracting it to be able to be solved without a computer. Then you can translate those steps into code.

2.1 مقدمة في خوارزمية الحاسوب

قبل البدء في خوارزميات رسومات الكمبيوتر ، نحتاج إلى معرفة معنى مصطلح "خوارزمية الكمبيوتر".

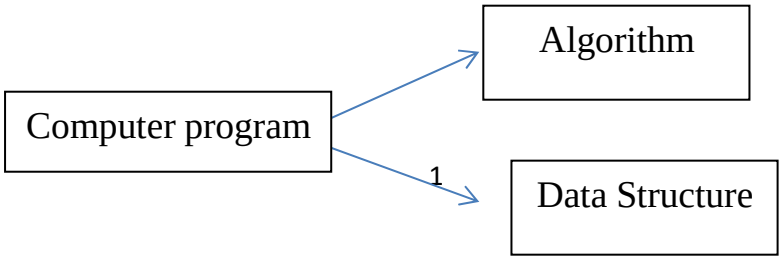
الخوارزمية هي أي إجراء حسابي محدد جيدًا يأخذ بعض القيم كمدخلات وينتج بعض القيم كإخراج. مثل وصفة الطهي ، توفر الخوارزمية طريقة خطوة بخطوة لحل مشكلة حسابية. على عكس البرامج ، لا تعتمد الخوارزميات على لغة برمجة معينة أو آلة أو نظام أو مترجم (يوصف عادةً باستخدام لغة طبيعية ورمز زائف). إنها كيانات رياضية ، يمكن اعتبارها تعمل على نوع من الكمبيوتر المثالي مع ذاكرة وصول عشوائية غير محدودة وحجم كلمة غير محدود. يدور تصميم الخوارزمية حول النظرية الرياضية وراء تصميم البرامج الجيدة. إن عملية كتابة الخوارزمية هي مجرد حل لمشكلتك ، وتقسيمها ، وتجريدها لتكون قادرة على حلها بدون جهاز كمبيوتر. ثم يمكنك ترجمة هذه الخطوات إلى رمز.

Computer algorithm has the following features:
– A detailed step-by-step instruction.
– ”No ambiguity” - no fuzz!
– Careful specification of the input and the output.

تحتوي خوارزمية الكمبيوتر على الميزات التالية:
- تعليمات مفصلة خطوة بخطوة.
- "لا لبس" - لا ضباب!
- تحديد دقيق للمدخلات والمخرجات.

Note that:لاحظ أن:
• The same algorithm may be represented in many different ways
• Several [different] algorithms may exist for solving a particularproblem
• Algorithms for the same problem can be based on very differentideas and have very different properties.

• قد يتم تمثيل نفس الخوارزمية بعدة طرق مختلفة
• قد توجد عدة خوارزميات [مختلفة] لحل مشكلة معينة
• يمكن أن تستند الخوارزميات الخاصة بنفس المشكلة إلى أفكار مختلفة جدًا ولها خصائص مختلفة جدًا.



2.2 Line Drawing Algorithm

The Cartesian slope-intercept equation for a straight line is

$$y = m \cdot x + b \qquad (2\text{-}1)$$

with m representing the slope of the line and b as they intercept. Given that the two endpoints of a line segment are specified at positions (x1, y1) and (x2, y2), as shown in Fig. 2-1, we can determine values for the slope m and y intercept b with the following calculations:

$$m = \frac{y2-y1}{x2-x1} \qquad (2\text{-}2)$$

$$b = y1 - m.x1 \qquad (2\text{-}3)$$

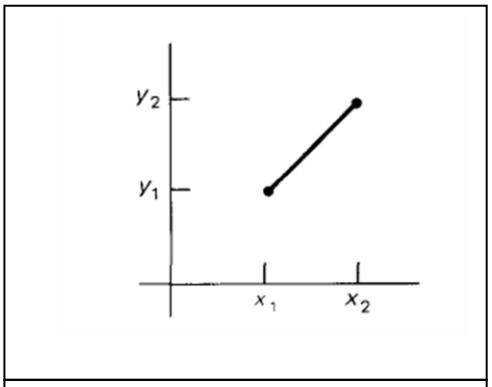


Figure (2.1) Line path between endpoint positions (x1,y1) and (x2,y2).

معادلة ميل وتقاطع الديكارتي للخط المستقيم هي

ص = م • س + ب (2-1)

حيث تمثل m ميل الخط بينما تمثل b عند التقاطع. بالنظر إلى أن نقطتي النهاية لقطعة مستقيمة محددة في المواضع (x1، y1) و (x2، y2) ، كما هو موضح في الشكل 2-1 ، يمكننا تحديد قيم المنحدر m وتقاطع y b بالحسابات التالية :

Algorithms for displaying straight lines are based on the line equation (2-1) and the calculations given in Eqs. (2-2) and (2-3).

For any given x interval Δx along a line, we can compute the corresponding y interval Δy from Eq.(2-2) as:

تعتمد خوارزميات عرض الخطوط المستقيمة على معادلة الخط (2-1) والحسابات الواردة في المعادلات. (2-2) و (2-3).

لأي فاصل زمني x معين Δx على طول خط ، يمكننا حساب الفاصل y المقابل Δy من المعادلة (2-2) على النحو التالي:

$$\Delta y = m.\Delta x \qquad (2\text{-}4)$$

Similarly, we can obtain the x interval Δx corresponding to a specified Δy

وبالمثل ، يمكننا الحصول على الفاصل الزمني x المقابل لـ yΔ المحدد

as:

$$\Delta x = \frac{\Delta y}{m} \qquad — \qquad (2\text{-}5)$$

These equations form the basis for determining deflection voltages in analog devices. For lines with slope magnitudes |m| < 1, Δx can be set proportional to a small horizontal deflection voltage and the corresponding vertical deflection is then set proportional to Δy as calculated from Eq. (2-4). For lines whose slopes have magnitudes |m| > 1, Δy can be set proportional to a small vertical deflection voltage with the corresponding horizontal deflection voltage set proportional to Δx, calculated from Eq. (2-5). For lines with m = 1, Δx = Δy and the horizontal and vertical deflections voltages are equal. In each case, a smooth line with slope m is generated between the specified endpoints.

تشكل هذه المعادلات الأساس لتحديد جهد الانحراف في الأجهزة التناظرية. للخطوط ذات الميل بمقادير | م | <1 ، يمكن ضبط xΔ بشكل متناسب مع جهد انحراف أفقي صغير ثم يتم تعيين الانحراف الرأسي المقابل متناسبًا مع yΔ كما هو محسوب من Eq. (2-4). للخطوط التي لها منحدرات مقادير | م | > 1 ، يمكن ضبط yΔ بشكل متناسب مع جهد انحراف رأسي صغير مع ضبط جهد الانحراف الأفقي المقابل المتناسب مع xΔ ، محسوب من Eq. (2-5). بالنسبة للخطوط التي تحتوي على م = 1 ، yΔ = xΔ والجهدان الأفقي والرأسي للانحراف متساويان. في كل حالة ، يتم إنشاء خط أملس بميل م بين نقاط النهاية المحددة.

On raster systems, lines are plotted with pixels, and step sizes in the horizontal and vertical directions are constrained by pixel separations. That is, we must "sample" a line at discrete positions and determine the nearest pixel to the line at each sampled position. The scan conversion process for straight lines is illustrated in Fig.(2-2), for a near horizontal line with discrete sample positions along the x axis.

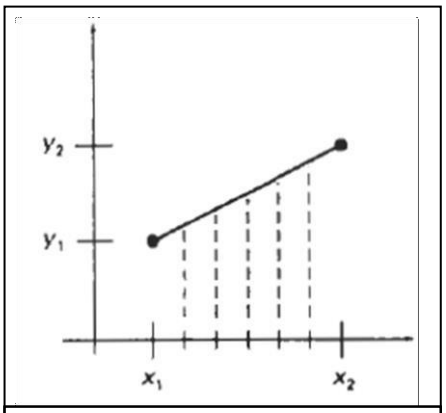


Figure (2-2) Straight line segment with five sampling positions along the x axis between x1 and x2.

في الأنظمة النقطية ، يتم رسم الخطوط بالبكسل ، ويتم تقييد أحجام الخطوات في الاتجاهين الأفقي والرأسي بفواصل البكسل. وهذا يعني أنه يجب علينا "أخذ عينات" لخط ما في مواضع منفصلة وتحديد أقرب بكسل للخط في كل موضع تم أخذ عينات منه. يوضح الشكل (2-2) عملية تحويل المسح للخطوط المستقيمة لخط أفقي قريب مع مواضع عينة منفصلة على طول المحور س.

DDA Algorithm:

The digital differential analyzer (DDA) is a scan-conversion line algorithm based on calculating either Δy or Δx, using Eq.(2-4) or Eq.(2- 5). We sample the line at unit intervals in one coordinate and determine corresponding integer values nearest the line path for the othercoordinate.

Consider first a line with positive slope, as shown in Fig.(2-1). If the slope is less than or equal to 1, we sample at unit x intervals ($\Delta x = 1$) and compute each successive y value as:

خوارزمية DDA:

محلل التفاضل الرقمي (DDA) عبارة عن خوارزمية لخط تحويل المسح تعتمد على حساب Δy أو Δx ، باستخدام المعادلة (2-4) أو المعادلة (2-5). نقوم بتجربة الخط على فترات وحدة في إحداثي واحد ونحدد قيم الأعداد الصحيحة المقابلة الأقرب لمسار الخط للإحداثي الآخر.

ضع في اعتبارك أولاً خطًا ذا ميل موجب ، كما هو موضح في الشكل (2-1). إذا كان الميل أقل من أو يساوي 1 ، فنحن نختبر في فواصل زمنية للوحدة س (x = 1Δ) ونحسب كل قيمة y متتالية على النحو التالي:

$$yk + 1 = yk + m \qquad (2\text{-}6)$$

Subscript k takes integer values starting from 1, for the first point, and increases by 1 until the final endpoint is reached. Since m can be any real number between 0 and 1, the calculated y values must be rounded to the nearest integer.

For lines with a positive slope greater than 1, we reverse the roles of x and y. That is, we sample at unit y intervals (Δy = 1) and calculate each succeeding x value as:

يأخذ Subscript k قيمًا صحيحة تبدأ من 1 ، للنقطة الأولى ، ويزيد بمقدار 1 حتى يتم الوصول إلى نقطة النهاية النهائية. نظرًا لأن m يمكن أن يكون أي عدد حقيقي بين 0 و 1 ، يجب تقريب قيم y المحسوبة إلى أقرب عدد صحيح.

بالنسبة للخطوط التي يكون ميلها موجبًا أكبر من 1 ، نعكس دور كل من x و y. أي أننا نأخذ عينات على فترات زمنية للوحدة (Δy = 1) y ونحسب كل قيمة x تالية على النحو التالي:

$$xk + 1 = xk + \frac{1}{m} \qquad (2\text{-}7)$$

Equations (2-6) and (2-7) are based on the assumption that lines are to be processed from the left endpoint to the right endpoint (Fig. 2-1). If this processing is reversed, so that the starting endpoint is at the right, then either we have Δx = -1 and

تعتمد المعادلتان (2-6) و (2-7) على افتراض معالجة الخطوط من نقطة النهاية اليسرى إلى نقطة النهاية اليمنى (الشكل 2-1). إذا تم عكس هذه المعالجة ، بحيث تكون نقطة نهاية البداية على اليمين ، فإما أن يكون لدينا Δ x = -1 و

$$yk + 1 = yk - m \qquad (2\text{-}8)$$

or (when the slope is greater than 1) we have Δy = -1 with

$$xk + 1 = xk - \frac{1}{m} \qquad (2\text{-}9)$$

Equations (2-6) through (2-9) can also be used to calculate pixel positions along a line with negative slope. If the absolute value of the slope is less than 1 and the start endpoint is at the left, we set Δx = 1 and calculate y values with Eq.(2-6).

When the start endpoint is at the right (for the same slope), we set Δx = -1 and obtain y positions from Eq. (2-8). Similarly, when the absolute value of a negative slope is greater than 1, we use Δy = -1 and Eq.(2-9) or we use Δy = 1 and Eq.(2-7).

This algorithm is summarized in the following procedure, which accepts as input the two end point pixel positions. Horizontal and vertical differences between the endpoint positions are assigned to parameters dx and dy. The difference with the greater magnitude determines the value of parameter steps. Starting with pixel position (x1, y1), we determine the offset needed at each step to generate the next pixel position along the line path. We loop through this process steps times. If the magnitude of dx is greater than the magnitude of dy and x1 is less than x2, the values of the increments in the x and y directions are 1 and m, respectively. If the greater change is in the x direction, but x1 is greater than x2, then the decrements - 1 and -m are used to generate each new point on the line.

Otherwise, we use a unit increment (or decrement) in the y direction and an x increment (or decrement) of l / m.

يمكن أيضًا استخدام المعادلات (2-6) إلى (2-9) لحساب مواضع البكسل على طول خط ذي ميل سلبي. إذا كانت القيمة المطلقة للميل أقل من 1 وكانت نقطة نهاية البداية على اليسار ، فإننا نضع x = 1Δ ونحسب قيم y بالمكافئ (2-6).
عندما تكون نقطة نهاية البداية على اليمين (لنفس المنحدر) ، نقوم بتعيين x = -1Δ ونحصل على مواضع y من Eq. (2-8). وبالمثل ، عندما تكون القيمة المطلقة لمنحدر سالب أكبر من 1 ، فإننا نستخدم y = -1Δ و Eq. (2-9) أو نستخدم y = 1Δ و Eq. (2-7).
يتم تلخيص هذه الخوارزمية في الإجراء التالي ، والذي يقبل كإدخال مواضع نقطة نهاية البكسل. يتم تعيين الفروق الأفقية والعمودية بين مواضع نقطة النهاية للمعلمات dx و dy. يحدد الاختلاف مع الحجم الأكبر قيمة خطوات المعلمة. بدءًا من موضع البكسل (x1، y1) ، نحدد الإزاحة المطلوبة في كل خطوة لإنشاء موضع البكسل التالي على طول مسار الخط. نحن نمر عبر هذه العملية خطوات الخطوات. إذا كان حجم dx أكبر من حجم dy و x1 أقل من x2 ، فإن قيم الزيادات في الاتجاهين x و y هي 1 و m على التوالي. إذا كان التغيير الأكبر في اتجاه x ، ولكن x1 أكبر من x2 ، فسيتم استخدام التناقصات - 1 و -m لإنشاء كل نقطة جديدة على الخط.
بخلاف ذلك ، نستخدم زيادة (أو إنقاص) وحدة في الاتجاه y و x زيادة (أو إنقاص) من m / l.

The steps of LineDDA Algorithm can be described as follows:

1- Get the two points: start point(x1,y1) and end point(x2,y2).2- Calculate dx & dy:
dx=x2-x1dy=y2-y1
3- Calculate length of the line (L):
L=max(|dx|,|dy|)
4- Calculate INCx & INCy:INCx=dx/L
INCy=dy/L
5- Set x=x1,y=y1 6- Repeat L times:
Pset(x,y) x=x+INCx
y=y+INCy
7-end