جامعة حماة
الكلية التطبيقية
قسم تقنيات الحاسوب

- المقرر: بيانيات الحاسب Computer Graphics

Chapter 3

مقدمة في الأنظمة البيانية المادية والبرمجية

Overview of Graphics System

3. البرمجيات البيانية Graphics Software:

نستطيع أن نميز عائلتين مستقلتين من الحزم البرمجية:

- الحزم ذات الاستخدامات الخاصة special purpose packages: صممت لتستخدم من قبل المستثمر العادي الراغب بتوليد صورة معينة، دون الاهتمام بالإجرائيات والمعالجات اللازمة للإظهار. يتفاعل المستثمر مع هذه البرمجيات بواسطة واجهات بيانية تتألف من قوائم **menus**.
- الحزم ذات الاستخدامات الخاصة العامة general programming packages: تؤمن مكتبات بيانية graphics library يتم فيها استخدام التوابع الموجودة من قبل مبرمجين بواسطة لغات برمجية مثل **C++, Java**. تتضمن هذه المكتبات توابع أساسية مثل إظهار الأوليات: مستقيم، مضلع، كرة، مخروط...، تحديد ألوان الأغراض، والتحويلات الهندسية الثنائية والثلاثية البعد. من هذه المكتبات على سبيل المثال GL (Graphics Library)، OpenGL، VRML (Virtual-Reality Modeling Language)، Java 2D، Java 3D

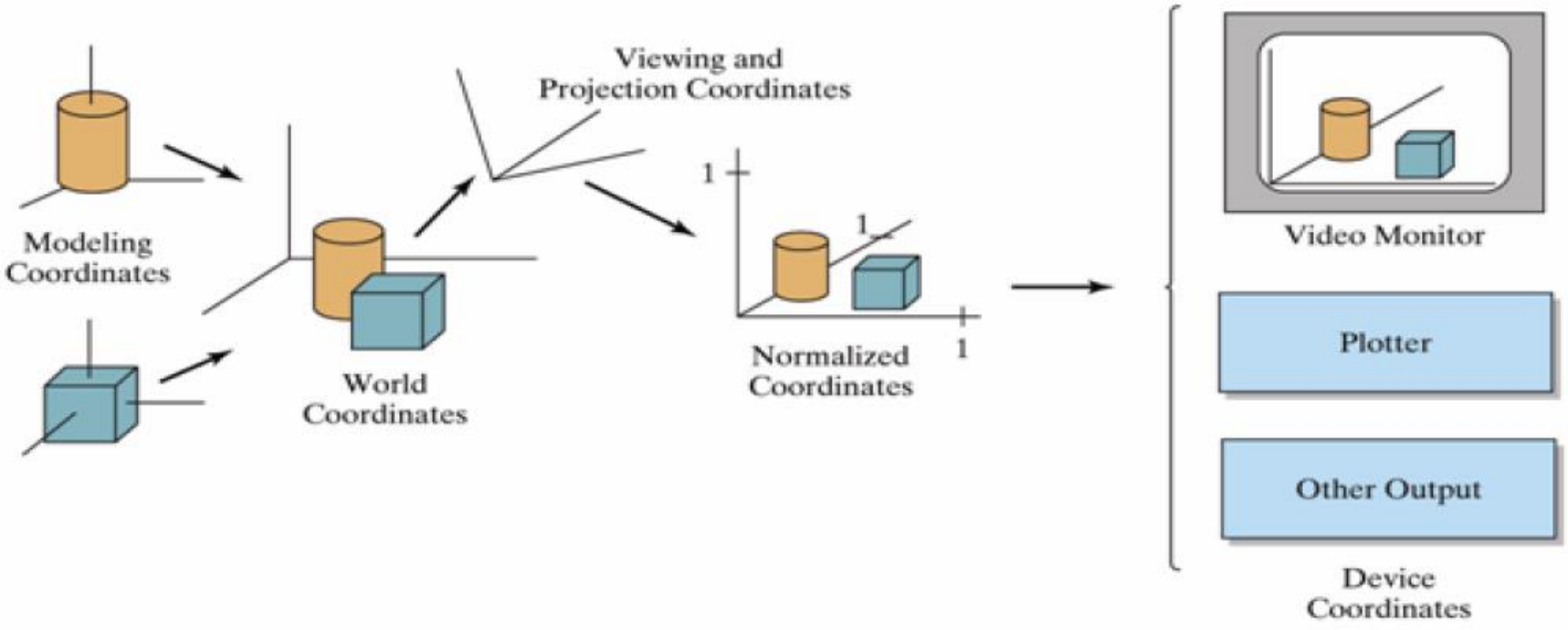
غالبًا ما تسمى هذه التوابع بـ .

Computer-graphics application programming interface **(CG API)**

لأنها تؤمن الواجهات الضرورية اللازمة للربط بين لغة البرمجة المستخدمة والبنى المادية.

1. التوابع البيانية

نحتاج إليها دومًا لوصف أغراض المشهد وإجراء التحويلات الهندسية عليها، تحريكها، إضاءتها، أو تشويهها. فمثً لا فلوصف البنية **structure** لغرض أولي (مكعب مثً لا) يتوجب تحديد الإحداثيات الديكارتية، الصحيحة **integer** أو الحقيقية **floating point** لنقاطه **vertices** وتمريرها إلى الحزم البرمجية **graphics packages** تقوم الحزمة البرمجية بتطبيق جملة من التحويلات الهندسية **viewing pipeline** تنقل عناصر الغرض من فضاء إحداثياته المحلية إلى فضاء إحداثيات وحدة الإخراج **output devices** الشاشة مثلاً الشكل (**11 -2**)



الشكل (2-11)

وبالتالي تؤمن الحزم البرمجية مكتبة من الإجرائيات (التوابع) مقسمة إلى مجموعات بعضها يتعلق بالتحويلات الهندسية **transformation** ومنها بالإظهار **viewing**، وأخرى بالإسقاط **projection** أو الإضاءة والتلوين. كما تؤمن بعض الحزم البرمجية بعض الإجرائيات الخاصة لإظهار بعض الأشكال المعقدة..

2. الأنظمة البيانية القياسية

تم تطوير عدة مكتبات برمجية بعضها يؤمن إجرائيات بيانية عامة، بينما بعضها الآخر موجه نحو تطبيقات خاصة مثل تطبيقات الإحياء **animation** وتطبيقات الحقيقة الافتراضية أو البيانيات عبر الانترنت. إلا أنه يبقى الهدف الأساسي من تقييس **standardization** الحزم البرمجية البيانية هو تأمين المحمولية **portability** والتي تتيح للمستثمر تنفيذ التطبيقات البرمجية على بنى مادية مختلفة. يمكن أن نميز عدد من المكتبات البيانية القياسية:

- المكتبة البيانية Graphical Kernel System تم تطويرها في عام 1984 وتعرف اختصارًا بـ **GKS** تم استخدامها في التطبيقات الثنائية والثلاثية البعد.

- المكتبة البيانية Programmer’s Hierarchical Interactive Graphics Standard وتعرف اختصاراً بـ**PHIGS** تم بواسطتها تحسين نمذجة الأغراض الهيكلية Hierarchical Object Modelingوالخصائص اللونية، وتقانات إظهار السطوح ومعالجة الصور.

- المكتبة البيانية **Graphics Library** وتعرف اختصارًا بـ **GL** قامت بتطويرها شركة **Silicon Graphics** تتميز هذا المكتبة بقدرتها على الإظهار في الزمن الحقيقي.
- المكتبة البيانية **Open Graphic Library** وتعرف اختصارًا بـ **OpenGL** وتعني مكتبة الرسوميات المفتوحة. تم تطويرها انطلاقًا من النظام البياني GL تعمل الهيئة **OpenGL Architecture Review Board** والممثلة لمجموعة من المنظمات والشركات على تطوير وتحديث المكتبة بشكل منتظم.

من أهم ميزاتها قدرتها الكبيرة على معالجة الأغراض الثلاثية البعد إضافةً إلى الأغراض الثنائية البعد. يمكن للمبرمج الاستفادة من التوابع البرمجية باستخدام إحدى اللغات البرمجية ADA، C، C++, Fortran

- المكتبة البيانية Open Inventor وهي مزودة بمجموعة من الصفوف والإجرائيات غرضية التوجه تساعد في وصف المشهد الذي يمكن إظهاره بواسطة المكتبية OpenGL
- المكتبة البيانية Virtual-Reality Modeling Language وهو ما يعرف إختصاراً بـ VRML والتي ظهرت في البداية كمكتبة جزئية ضمن Open Inventor تسمح هذه المكتبة بنمذجة الأغراض ثلاثية البعد في الواقع الإفتراضي على الانترنت.

كما يمكن أيضًا بناء المشاهد ثنائية وثلاثية البعد عبر الوب باستخدام المكتبة **Java D2** , **Java 3D** على التتالي.

خوارزميات تحويل المسح البيانية
Basic Raster Graphics Algorithms for Drawing 2D Primitives

تؤمن كافة الأنظمة البيانية مجموعة من الإجرائيات والخوارزميات تسمى خوارزميات تحويل المسح scan converting algorithm أو الـ sterization التي تقوم بتقريب approximate الرياضي المثاليideal للأوليات والموصَّفة بإحداثيات نقاطها vertices الديكارتية ، إلى مجموعة من البكسلات ذات شدة ضوئية intensity (قيم لونية) تحفظ في الـ frame buffer .

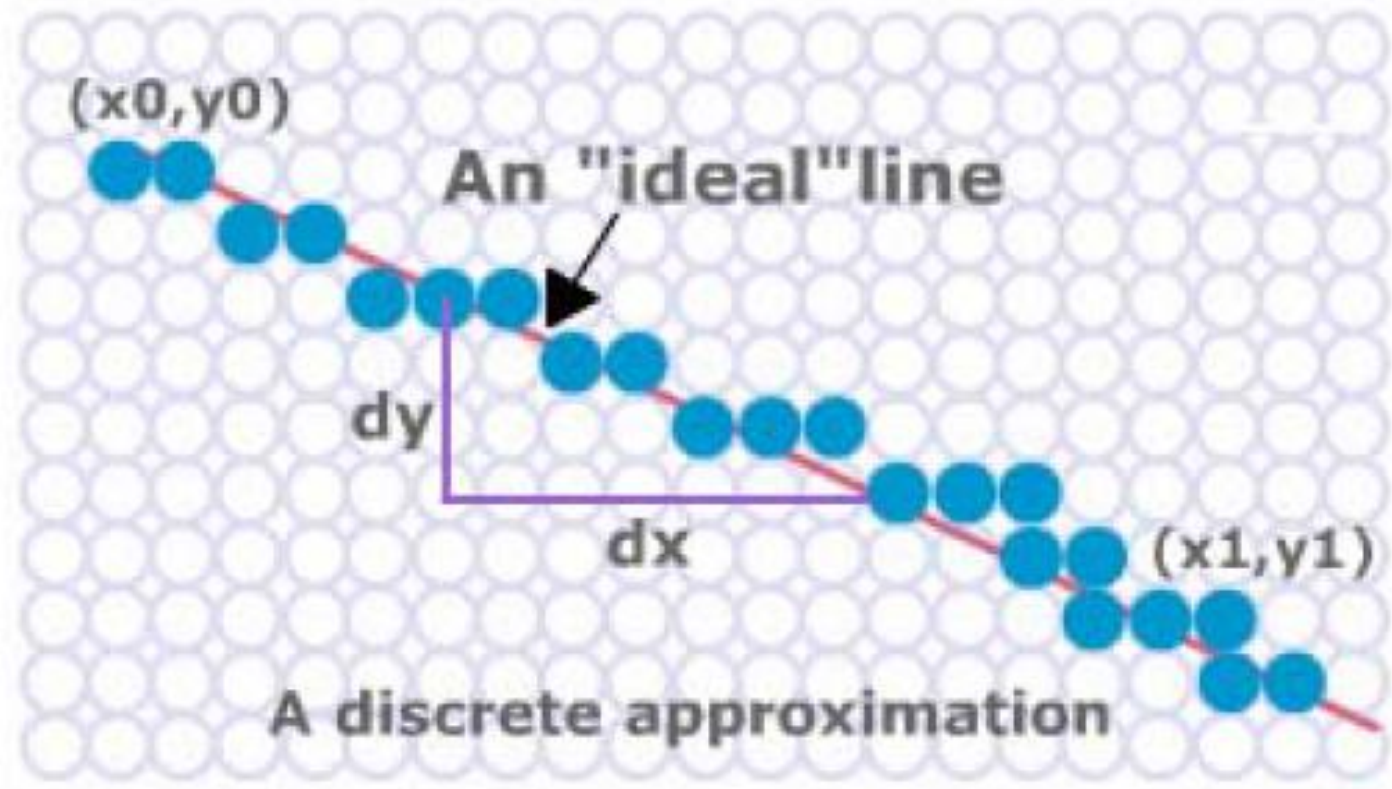
يهدف هذا الفصل إلى دراسة خوارزميات المسح، كما سنعرف القص **Clipping**وندرس أهم خوارزمياته.

1. الإحداثيات المرجعية للإطار

تتطابق مواضع النقاط على الشاشة **screen coordinates** مع مواقع البكسلات في الـ **frame Buffer** والتي تعرّف برقم سطر المسح **scan line number (y)** ورقم العمود **column number (x)** المنسوب إلى إحداثيات مرجعية عادةً ما تكون الزاوية العلوية اليسرى من الشاشة.

يتحدد موضع البكسل **pixel position** ، عند تقاطع خطوط الشبكة، على المحور **y** بقيمة صحيحة تقع ضمن المجال **[0–y_{max}]** (يعبر الرقم **0** عن رقم المسح الأول في أعلى الشاشة والرقم y_{max} عن سطر المسح الأخير في أسفل الشاشة)، وعلى المحور **x** بقيمة صحيحة تقع ضمن المجال **[0 – x_{max}]** . وبالتالي فموقع البكسل والذي يعبر عنه بقيم صحيحة هو تقريب للقيمة الحقيقية لهذه النقطة والواقعة على الشكل الرياضي المثالي (مستقيم، مضلع، منحن...).

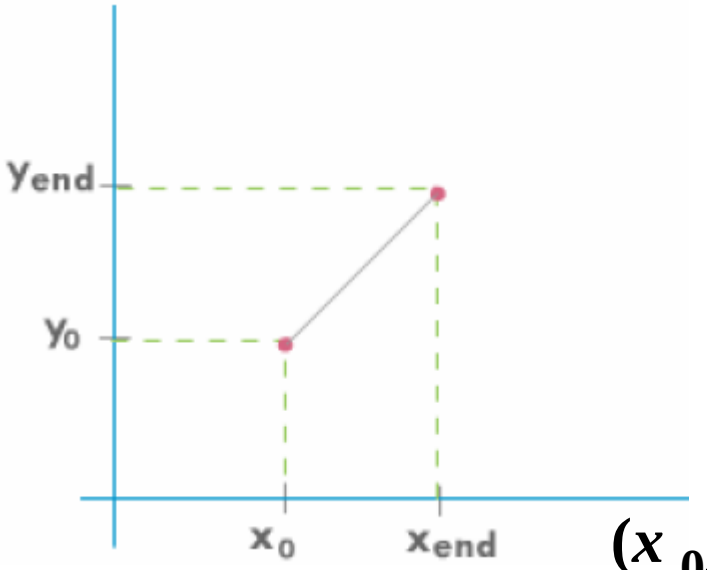
ينتج عن هذا التقريب تشويه في الشكل الظاهر على الشاشة وهو ما يعرف بظاهرة الدرج المتكسر كما في الشكل **(3-1)** التالي:



الشكل(3-1)

2. خوارزمية رسم مستقيم LINE-DRAWING ALGORITHMS

تعرّف قطعة مستقيمة بتحديد إحداثيات طرفيها (x_0, y_0) , (x_{end}, y_{end}) كما في الشكل(2-3)



الشكل(2-3)

1. خوارزمية algorithm slope-intercept:

تعرّف معادلة المستقيم **y** حيث **m** هو الميل و **b** هو محصور **y** للمستقيم $y = m \cdot x + b$

يتم حساب **b** و **m** بدلالة (x_0, y_0) , (x_{end}, y_{end})

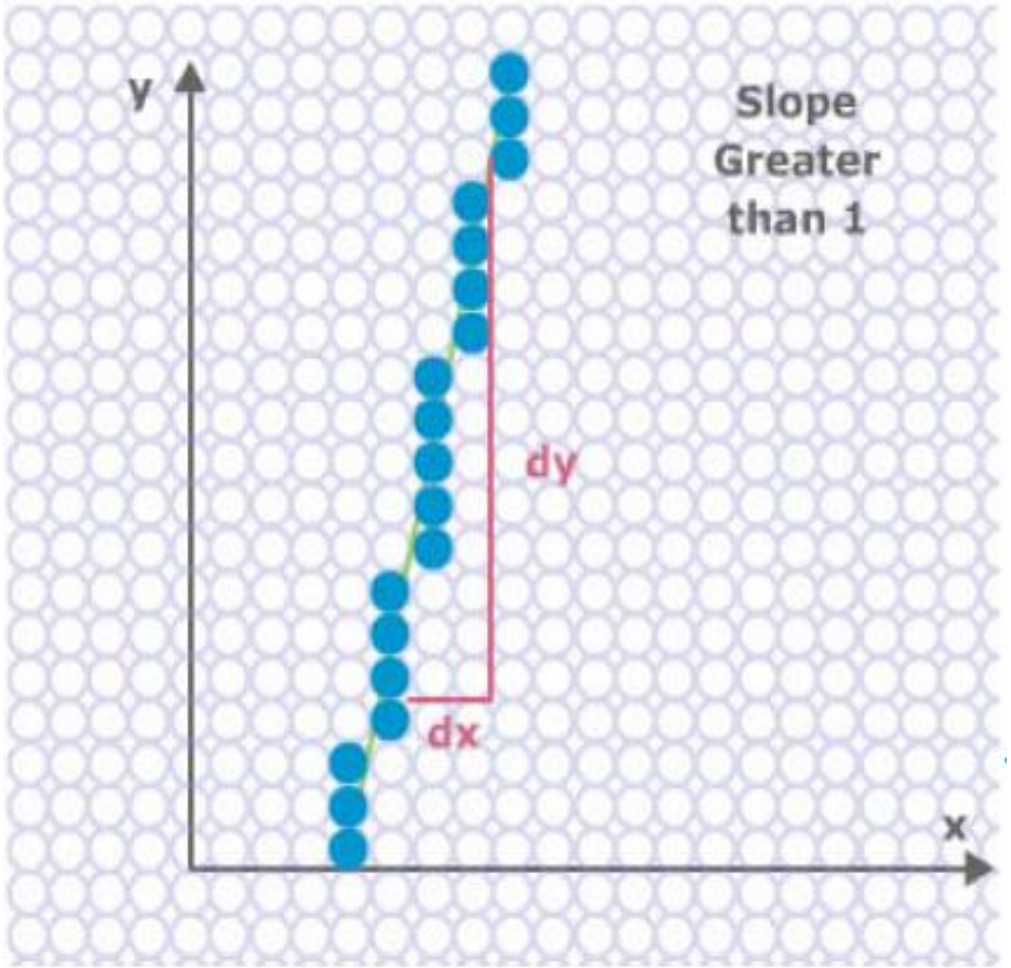
$$m = \frac{y_{end} - y_0}{y_{end} - x_0} \quad , \quad b = y_0 - m \cdot x_0$$

تتطلب الخوارزمية السابقة إجراء عمليتي الضرب والجمع على الأعداد الحقيقية عند كل خطوة إضافة إلى عملية التدوير **round**

لا تعطي الخوارزمية نتائج مرضية عندما يكون الميل $m > 1$ كما في الشكل (3-3) التالي:

2. خوارزمية المسح التفاضلي Digital Differential Analyzer

تعتبر خوارزمية المسح التفاضلي **Digital Differential Analyzer** والتي تعرف اختصارًا بـ**DDA** خوارزمية مسح متزايدة **incremental algorithm** بمعنى أنها تهدف إلى إلغاء عملية الضرب وتحويلها إلى عملية جمع.



الشكل (3-3)

سندرس الخوارزمية من أجل المستقيمات ذات الميول الموجبة.

في حال $m > 1$ تتم الزيادة على المحور y بمقدار 1 وتحسب قيم x_{i+1} استناداً لـ y

في حال $m <= 1$ تتم الزيادة على المحور x بمقدار 1 وتحسب قيم y_{i+1} المتتالية

$$\Delta x = x_{i+1} - x_i$$

$$y_{i+1} = m * x_{i+1} + b$$

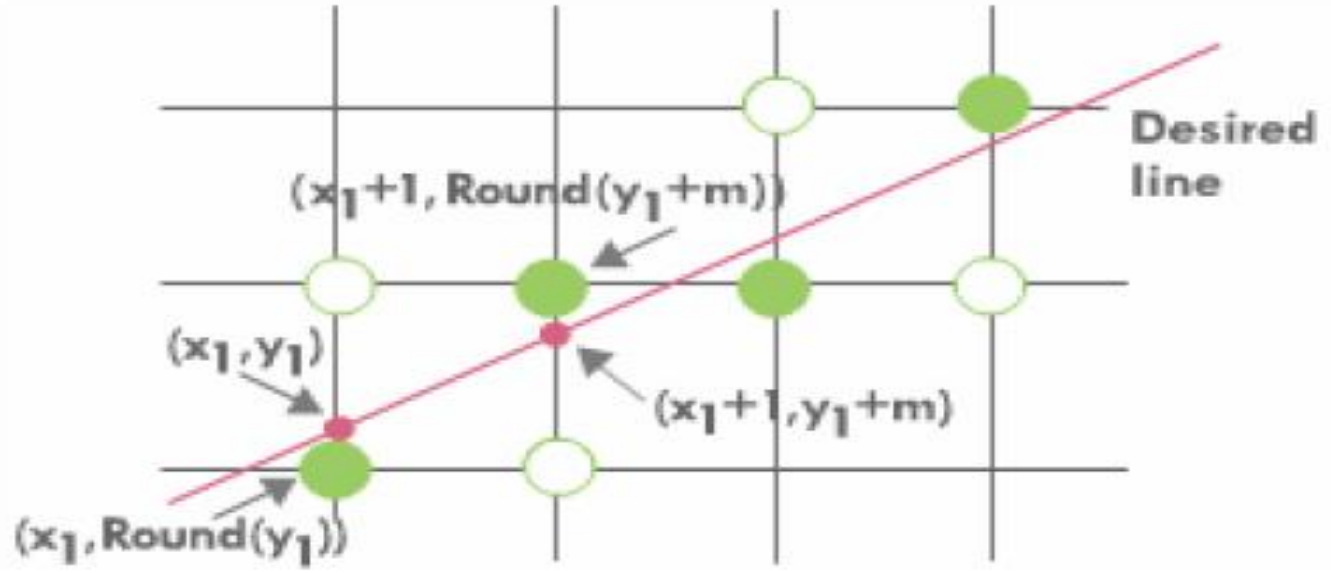
$$= m*(x_i + \Delta x) + b$$

$$= (m\, x_i + b) + m*\Delta x$$

$$= y_i + m*\Delta x$$

$$= y_i + m, \text{ if } \Delta x = 1$$

تفترض المعادلات السابقة أن رسم المستقيم يتم من اليسار إلى اليمين. كما يمكن تعميم الخوارزمية من أجل قيم سالبة للميول. من أجل إيجاد موضع النقطة الصحيح يتم تدوير القيمة الناتجة لأقرب موضع للبكسل round الشكل (3–5)



الشكل (3-4)

تعتبر خوارزمية **DDA** أسرع من خوارزمية **slope-intercept** نتيجة إلغاء عملية الضرب. لكنها ما زالت تعاني من تزايد الأخطاء التراكمية نتيجة عمليات التدوير، أضف إلى أن العمليات الحساب تتم بالفاصلة العائمة.

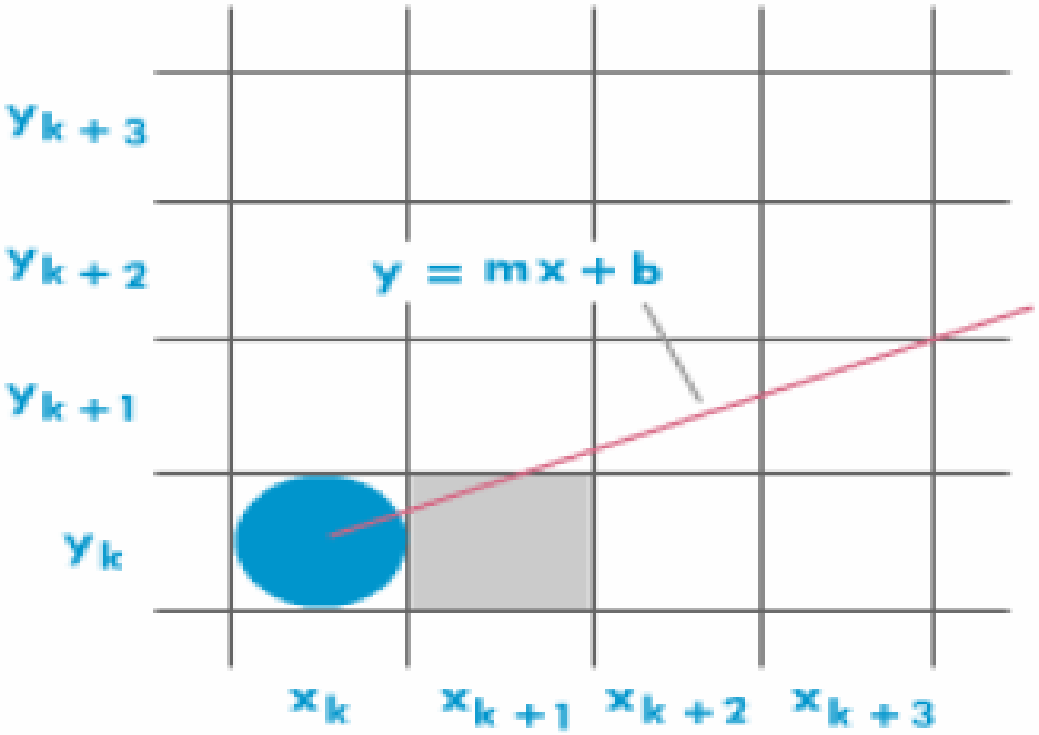
يمكن كتابة خوارزمية المسح التفاضلي العامة:

```
inline int round (const float a)
{ return int (a + 0.5); }
void lineDDA (int x0, int y0, int xend, int yend) {
int dx = xend - x0, dy = yend - y0, steps, k;
float xIncrement, yIncrement, x = x0, y = y0;
if (fabs (dx) > fabs (dy))
steps = fabs (dx);
else
steps = fabs (dy);
xIncrement = float (dx) / float (steps);
yIncrement = float (dy) / float (steps);
setPixel (round (x), round (y));
for (k = 0; k < steps; k++) {
x += xIncrement;
y += yIncrement;
setPixel (round (x), round (y));
}
}
```

3. خوارزمية Bresnham's:

سنشرح خوارزمية **Bresnham** في تحويل مسح المستقيم المبين في الشكل (3-5) من أجل $0 < m < 1$. تتقاطع المستقيمات العمودية مع المستقيمات الأفقية لتحدد موضع النقاط التي سترسم المستقيم. عند إختيار قيمة على المحور المستقيم. عند إختيار قيمة على المحور **x**، نحتاج إلى تحديد قيمة **y** الأقرب الى المستقيم. تتجاوز هذه الخوارزمية سيئات الخوارزمية السابقة حيث يستخدم **Bresnham** عمليات حسابية صحيحة متزايدة تعتمد على عمليات الجمع والطرح.

فمثلا عند إظهار النقطة (x_k , y_k) ، نريد تحديد البكسل الجديد المراد إظهاره من إجل x_{k+1} حيث $x_{k+1} = x_k + 1$ من الواضح إننا أمام خيارين وحيدين يتوجب إختيار إحداهما فقط و هما البكسل (x_{k+1} , y_k) أو البكسل (x_{k+1} , y_{k+1})



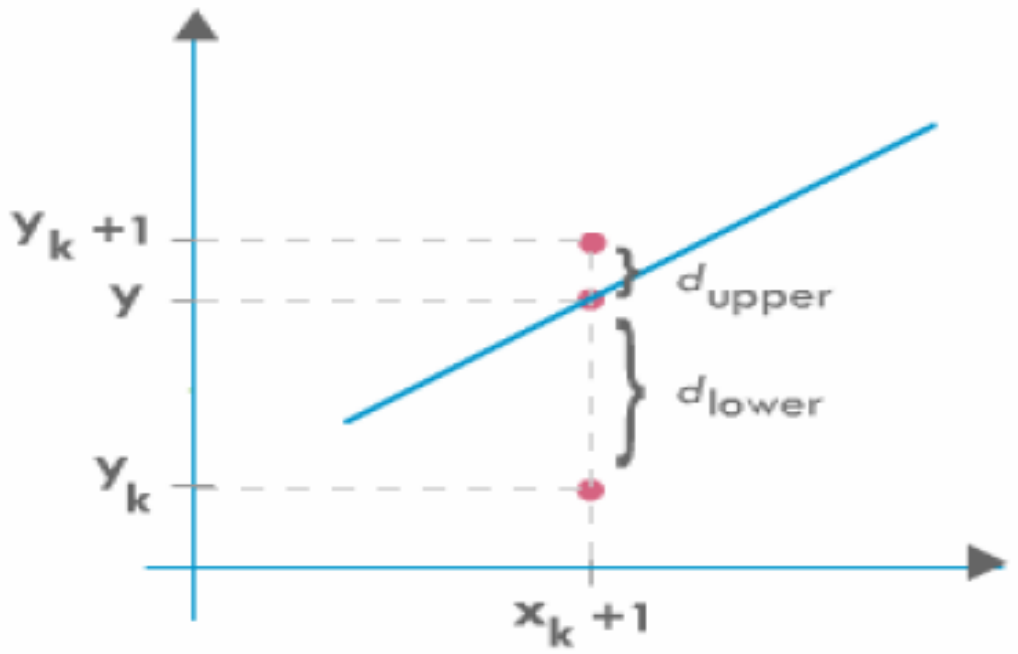
الشكل (3-5)

من أجل الإحداثي x_{k+1} سنسمي المسافة العمودية بين y_k والمستقيم بـ d_{lower}

والمسافة العمودية بين y_{k+1} والمستقيم d_{upper}

تعطى الإحداثية y من أجل $x+1$ وفقاً لمعادلة المستقيم $y = m(x_k + 1) + b$ وبالتالي

تحسب d_{lower} و d_{upper} كما في الشكل (3-6)



الشكل (3-6)

$$d_{lower} = y - y_k = m(x_k + 1) + b - y_k$$
$$d_{upper} = (y_k + 1) - y = y_k + 1 - m(x_k + 1) - b$$

لتحديد أي من البكسلين أقرب إلى المستقين نحسب الفرق بين العلاقتين الأخيرتين:

$$d_{lower} - d_{upper} = 2m(x_k + 1) - 2yx + 2b - 1$$

بترتيب العلاقة الأخيرة يمكن الحصول على $\mathbf{p_k}$ وتسمى **Decision parameter**:

$$p_k = \Delta x(d_{lower} - d_{upper}) = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + c$$

حيث c ثابت ويساوي $\boldsymbol{c} = 2\Delta \boldsymbol{y} + \Delta \boldsymbol{x}(2b - 1)$ ولا يتعلق بموضع البكسل.

- في حال $d_{lower} < d_{upper}$ $(p_k < 0)$ يتم إختيار البكسل y_k كونه أقرب إلى المستقيم من البكسل $y_k + 1$.
- في حال $d_{lower} > d_{upper}$ $(p_k > 0)$ يتم إختيار البكسل $y_k + 1$ كونه أقرب إلى المستقيم من البكسل y_k.

يمكن الحصول على القيم المتعاقبة لـ p_k بإستخدام متكرر متزايد بالقيم الصحيحة. عند الخطوة $k+1$ يتم حساب $p_k + 1$

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

بطرح $p_k + 1$ من p_k ينتج:

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

وبما أن $x_{k+1} = x_k + 1$ يمكن حساب $p_k + 1$:

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

حيث دوماً $y_{k+1} - y_k$ تساوي الصفر أو الواحد وفقاً لإشارة p_k.

بذلك يتم حساب p_k بشكل متكرر متزايد من أجل قيم x من اليسار إلى اليمين وحتى نهاية المستقيم. يتم حساب p_0 من أجل النقطة (x_0 , y_0) :

$$p_0 = 2\Delta y - \Delta x$$

وبالتالي يمكن كتابة خوارزمية **Bresnham** من أجل المستقيم حيث $0 < m < 1$ حسب ما هو مبين أدناه حيث يتم حساب الثوابت $2\Delta x , 2\Delta y - 2\Delta x$ مرة واحدة.

يمكن تعميم الخوارزمية من إجل مستقيمات ذات قيم ميول مختلفة بالإخذ بعين الإعتبار التناظر **symmetry** ضمن الأثمان **octants** والأرباع **quadrtants** في المستو **xy** .

يتم رسم المضلع **polyline** بتطبيق خوارزمية **Bresnham** أيضًا من أجل **(n-1)** مستقيم يصل بين n نقطة. كما تطبق الخوارزمية نفسها من أجل رسم المنحنيات.

```
/* Bresenham line-drawing procedure for |m| < 1.0. */
void lineBres (int x0, int y0, int xEnd, int yEnd){
int dx = fabs (xEnd - x0), dy = fabs(yEnd - y0);
int p = 2 * dy - dx;
int twoDy = 2 * dy, twoDyMinusDx = 2 * (dy - dx);
int x, y;
/* Determine which endpoint to use as start position. */
if (x0 > xEnd) {
x = xEnd;
y = yEnd;
xEnd = x0;
}
else {
```

```
x = x0;
y = y0;
}
setPixel (x, y);
while (x < xEnd ) {
x++;
if (p < 0)
p += twoDy;
else {
y++;
p += twoDyMinusDx;
}
setPixel (x, y);
}
}
```