تفريغ مقرر بيانيات الحاسب| BCG601

Slash team

برنامج الهندسة المعلوماتية – الجـــــامعة الافتراضية السورية

تفريغ الطالب/ـــة:	ميادة ميرو
المدرّس المحاضر:	د. فاتن مشتا
عنوان المحاضرة	Scan conversion
رقم المحاضرة	3
تاريخ بث المحاضرة	2022/11/11

SCAN CONVERSION

مفهوم ال SCAN CONVERSION

مقدمة:

Scan conversion: أي تحويل الأغراض من التمثيل الهندسي الى صورة.

- لرسم مستقيم أنا بحاجة الى معادلة رسم مستقيم فقط

$$y = mx + b$$

عند رسمه أحدد نقطة البداية ونقطة النهاية وأرسم المستقيم الواصل بينهما

أو بحسب الميلان وتقاطع المستقيم مع محور ال y وأرسم المستقيم.

- كيف أحول هذا التمثيل الهندسي الى صورة؟

الصورة هي مصفوفة ثنائية البعد من البكسلات.

فرسم مستقيم على صورة يقابل تلوين البكسلات التي يمر بها المستقيم.

- ماذا سنتعلم ؟

سنتعلم كيف نختار البكسلات لرسم المستقيم بأفضل صورة ممكنة لتكون أقرب لشكل المستقيم المثالي.

وسنرى ماهي الخوارزميات الفعالة لرسم المستقيمات.

سنرى خوارزميات رسم الدوائر والخطوط وخوارزميات التلوين(flood fill , scan line)

وسنتعلم ال Anti-Aliasing لإزالة التكسرات من الخطوط لتبدو مثالية بأكبر شكل ممكن.

خطوات الرسم التفاعلي:

مرحلة الرسم التفاعلي على الحاسب تمر بعدة خطوات

1. Application model

- تمثل مجموعة المعلومات أو الأغراض الموجودة ضمن المشهد.
- تضم بيانات ثلاثية البعد بالاعتماد على تمثيلها الهندسي.
- تعرف كنموذج أو مشهد.
- يمكن رسم هذه الأغراض باليد أو بواسطة برامج على الحاسب.

2. Application program

- بناء النموذج الذي حصلنا عليه من Application model (جمع الأغراض بمشهد واحد).
- تنفيذ العمليات اللازمة للحصول على المشهد المطلوب والتفاعل مع الدخل.

3. Graphic system

- تتوسط ما بين Application program وشاشة العرض.
- مبرمج بلغة عالية المستوى لتنتج صور عالية المستوى ليتم عرضها على الشاشة.
- تمرر الدخل للمعالجة.

- Rendering

هي عملية تجهيز الصورة للعرض على الشاشة ولها الخطوات التالية:

1) أشكال أولوية بإحداثيات الأغراض object space
2) Modeling transformation لجمع الأغراض بمشهد واحد والإحداثيات هنا هي word space
3) الإضاءة والظلال والسطوح الخفية والإحداثيات هي camera space
4) الإسقاط
5) Scan conversion للحصول على الصورة بالبكسل.

RASTER_SCAN DISPLAY SYSTEM (SRGP)

هو نظام البيانات المستخدم مع شاشات المدفع الإلكتروني حيث ينقل المعلومات الهندسية ومواصفاتها ويحولها الى أوامر لتحويلها الى صور بواسطة ال SRGP

حيث كل بكسل يقابل صورة على الشاشة ونحن بحاجة الى التحكم بكل بكسل عن طريق Display Processer .

ثم يقوم ال video controller بتحويل هذه البكسلات الى قيم كهربائية.

المصفوفة الثنائية من البكسلات هنا حجمها هائل جداً ويتم مسحها ابتداءاً من الشاشة العليا اليسارية.

يتم هذا بواسطة الDouble buffering

Buffer A يقوم بعمل on/off للبكسلات

Buffer B يحول البكسلات لإشارات كهربائية

ملاحظة: سؤال امتحاني ماهي وظيفة الDouble Buffering

حتى يعرض الحركة ضمن الصور بشكل سلس.

بارامترات SCAN CONVERTING

التحويل من التمثيل الهندسي الى صورة.

الدخل: معادلة مستقيم لنقطتين

الخرج: قائمة بكسلات

نضيئ البكسلات حتى نحصل على أقرب ما يمكن لشكل المستقيم.

إحداثيات البكسلات هي أسطر وأعمدة نبدأها من الزاوية العلوية اليسارية.

- لرسم مستقيم نحتاج مراعاة هذه الأمور
  1. أن تكون النهايات أفضل ما يمكن (البداية واضحة والنهاية واضحة)
  2. أن يكون المستقيم مستمر.
  3. أن تكون السماكة موحدة على كل المستقيم.

- المواصفات اللازمة لهذه الخوارزمية: أن تكون الخوارزمية فعالة بشكل جيد
  - لأنه يمكن استخدامها ملايين المرات فيجب أن تنفذ بأكبر سرعة ممكنه.
  - يمكن استخدام incremental method أي الاستفادة من الحسابات السابقة
  - استعمال عمليات جمع أكثر من الضرب.
  - عمل on للبكسلات الاقرب من الخط المثالي.
  - استعمال العمليات على integer كونها أسرع.
  - استعمال المعالجة التفرعية لتسريع عمل الخوارزمية.

- أول خوارزمية للرسم على الحاسب كانت للعالم Bersenham وهي line-drawing عام 1965 ثم تلتها خوارزمية الدوائر circles لنفس العالم عام 1977 .
- المضلعات يمكن رسمها باستخدام المستقيمات.
- أي شكل استطيع تقسيمه الى عدة مستقيمات.

ملاحظة: جميع الخوارزميات السابقة مستقلة عن hard ware ويمكن تنفيذها على جميع مكتبات الgraphics .

خوارزمية رسم المستقيم

نعلم أن معادلة رسم المستقيم هي:

$$y = mx + b$$

حيث m هي الميلان و b هي نقطة التقاطع مع المحور y

إذا كان لدي نقطتين من هذا المستقيم $(x_1, y_1)(x_2, y_2)$ نحسب m من هذه المعادلة

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

هناك طريقة ثانية لتمثيل المستقيم تسمى parametric form

$$y = y_1 + (y_2 - y_1)t$$

$$x = x_1 + (x_2 - x_1)t$$

حيث t هو معامل بين ال 0 و 1

حالات خاصة لرسم المستقيم:

1) خط افقي: موازي لمحور x

نبدأ بنقطة البداية ونزيد قيمة الx مع تثبيت قيمة الy ونضيئ البكسل

وسوف يكون المستقيم مثالي كوني اضيئ البكسلات بشكل افقي.

2) خط عامودي: موازي لمحور y

نبدأ بنقطة البداية ونزيد قيمة الy مع تثبيت قيمة الx ونضيئ البكسل

وسوف يكون المستقيم مثالي كوني اضيئ البكسلات بشكل عامودي.

3) خط قطري: ميلان بزاوية 45°

نبدأ بنقطة البداية ونزيد x بقيمة 1 و y بقيمة 1 ونضيئ البيكسل.

الحالة العامة:

لدي مستقيم بميل ما m

نرسمه بحيث نضيئ البكسلات الأقرب الى الخط للحصول على مستقيم مثالي.

الفكرة هنا بكيفية تحديد البكسلات الأقرب.

خوارزمية اختيار البيكسل هي التي تلعب دور في شكل المستقيم النهائي الذي نحصل عليه.

الخوارزمية NAÏVE ALGORITHM

لدي معادلة مستقيم

$$y = mx + b$$

$$y_0 = mx_0 + b$$

$$y_1 = mx_1 + b$$

$$m = \frac{(y_1 - y_0)}{(x_1 - x_0)}$$

$$b = y_0 - mx_0$$

- لكتابة الخوارزمية:

```
void line (int x0 , int yo , int x1 ,int y1 )
{
    int dx = x1 - x0;
    int dy = y1 - y0;
    setPixel(x0 , yo);   //اضيء بكس نقطة البداية
    if (dx != 0)   //اتحقق إذا وصلت لنقطة النهاية
    {
        float m =(float) dy / (float) dx;   //الميل
        float b = y0 - m*x0;   //نقطة التقاطع مع محورy
        dx = (x1 > x0) ? 1 : -1;   //النقطة التي تلي نقطة البداية
        while (x0 != x1)   //طالما لم أصل للنقطة النهائية
        {
            x0 += dx;
            yo = Round( m*x0 + b);   //int معادلة المستقيم مع تقريب للحصول على الاحداثيات بنمط
            setPixel(x0 , yo);   //اضيئ النقطة التالية التي حصلت على احداثياتها
        }
    }
}
```

- مشاكل الخوارزمية السابقة:

أنها ليست فعالة لأنه:

1. يوجد عمليات قسمة وضرب
2. النمط المستخدم هو النمط float
3. عملية round

وجميع العمليات السابقة هي عمليات مكررة.

وإذا كان الميل أكبر من 1 لن تعطي نتائج جيدة.

- نلجأ لتحسين الخوارزمية السابقة باستخدام incremental Algorithm

حيث استفيد من حسابات سابقة يمكن من خلالها حساب احداثيات النقطة اللاحقة من السابقة.

$$y_{i+1} = mx_{i+1} + b = m(x_i + D_x) + b = y_i + mD_x$$

$$if\ D_x = 1\ \ then\ y_{i+1} = y_i + m$$

$$if\ D_x = -1\ \ then\ y_{i+1} = y_i - m$$

فنحصل على المعادلتين التاليتين:

$$x_{i+1} = x_i + 1$$

$$y_{i+1} = y_i + m$$

- فيكون الكود البرمجي بالشكل:

```
void line (int x0 , int yo , int x1 ,int y1 )
{
    int dx = x1 - x0;
    int dy = y1 - y0;
    float y = y0;
    if (dx != 0)   //اتحقق إذا وصلت لنقطة النهاية
    {
        float m =(float) dy / (float) dx;   //الميل
        for (x = x0 ; x <= x1 ; x++)
        {
            setPixel(x,Round(y));
            y += m;   // تخلصت هنا من عملية الضرب
        }
    }
}
```

ولكن لا تزال الخوارزمية ليست فعالة بشكل كافي.

- يوجد طريقة أخرى باستخدام الدارات المنطقية:

لن ندخل بتفاصيلها ولكن مفهومها هو بتجزيئ ال x, y الى جزء float وجزء integer.

تبقى جميع الخوارزميات السابقة ليست فعالة بالشكل المطلوب وليست المستخدمة للرسم على الحاسب.

الخوارزمية المستخدمة تسمى خوارزمية Mid-Point Line Algorithm

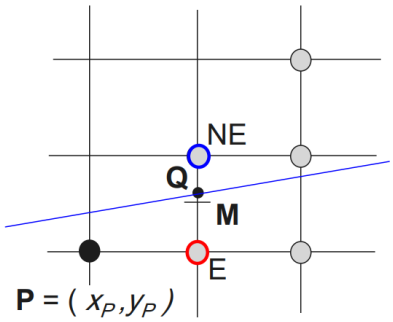
خوارزمية MID-POINT LINE ALGORITHM

هي الخوارزمية المستخدمة على الحاسب وتعد أسرع بكثير من الخوارزميات السابقة وتستخدم فقط في المستقيمات التي ميلها بين ال 1 و 0

باقي الحالات تستخدم فيها عملية التناظر.

البيكسل الذي يضيئ تكون احداثياته $P(x_p, y_p)$ ويكون من النمط integer

نأخذ النقطة الوسطى M ونرى أين سيمر المستقيم ونختار على اساسه البكسل المناسب.



معادلة المستقيم:

$$f(x) = y = m * x + b = \frac{d_y}{d_x} * x + b$$

$$f(x,y) = a * x + b * y + c = 0$$

بمقارنة المعادلتين السابقتين نستطيع حساب a, b, c

$$a = d_y$$

$$b = -d_y$$

$$c = b * d_x$$

$$F(x,y) \begin{cases} > 0 & \text{for points below the line} \\ = 0 & \text{for points on the line} \\ < 0 & \text{for points above the line} \end{cases}$$

ملاحظة: يوجد العديد من الحسابات الممكن الاطلاع عليها من السلايدات ولكنها غير مطلوبة, المطلوب فقط هو خطوات تنفيذ الخوارزمية.

- خطوات تنفيذ الخوارزمية:

1) نبدأ بتخزين النقطة (x_0, y_0)

2) نحدد أن $(x,y) = (x_0, y_0)$ كتعريف لنقطة البداية ونضيئها setpixel(x,y)

3) أحسب d

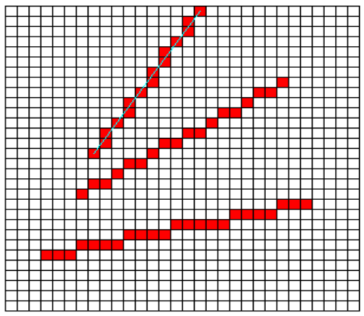
4) أضيئ النقاط بحسب d (التي سبق وذكرت الدكتورة أن طريقة حسابا غير ضرورية).

ملاحظة: الخوارزمية ككود غير مطلوب حفظها ولكن المهم أن نعرف أنه عندما تكون

$d \leq 0$ نضيئ البكسل E $d > 0$ نضيئ البكسل NE

المستقيمات التي تحسب ترسم بحسب الخوارزمية السابقة تعتبر مقبولة

مثال:



- ميزات خوارزمية Mid-Point Line Algorithm
  1. تستخدم كل الحسابات على النمط integer
  2. لا يوجد عمليات ضرب أو قسمة
  3. الخيار محدود بين بكسلين فقط أختار أحدهما
  4. ممكن استخدام التناظر للتقليل من التعقيد
  5. مناسبة للتنفيذ على ال Hard Ware
  6. مناسبة للغة Assembly

نتيجة:

- Scan Converting مهم لأنه شائع ويتكرر كثيراً في رسم الأغراض.
- كل الأشكال الهندسية يمكن ارجاعها الى مستقيمات.

خوارزمية رسم الدائرة

معادلة الدائرة:

$$x^2 + y^2 = R^2$$

$$y = \sqrt{R^2 - x^2}$$

نفترض أن الدائرة مركز الاحداثيات وان x تقع بين $-R \leq x \leq +R$

نحول للنمط integer لأن الاحداثيات دائما بالنمط integer

نضيئ البكسل الذي احداثياته (x, y) للحصول على نصف الدائرة.

```
setPixel(Rund(x),Round(y));
```

وبشكل متناظر نضيئ البيكسل المقابل لنرسم نصف الدائرة الآخر.

```
setPixel(Rund(x),Round(-y));
```

- تعتبر هذه الخوارزمية بسيطة ولكن مشكلاتها:
  1. تحوي على عملية جذر
  2. تحوي على عملية تربيع مرتين
  3. تحوي على عملية Round أربع مرات

وبالتالي تعتبر غير فعالة.

- تم تحسين هذه الخوارزمية باعتماد الاحداثيات القطبية:

حيث $x = 0^{\circ} \rightarrow 360^{\circ}$

نضيئ البيكسل :

```
setPixel(Rund(R.cos(x)),Round(R.sin(y)));
```

- ولكن أيضاً لهذه الخوارزمية مشاكل:
  1. عمليتين ضرب
  2. عمليتين Round
  3. بقيم x القريبة من R يكون هناك فراغات بين البكسلات.

- لتحسين الخوارزمية بشكل أكبر نستخدم التناظر:

نرسم النقاط الموجودة في ثمن الدائرة وبالاعتماد عليها استطيع رسم 7 نقاط متناظرة معها.

كل بكسل اضيئه استطيع اضاءة 7 بكسلات متناظرة.

ولكن حتى هذه لا تعتبر بالكفاءة المطلوبة وليست المستخدمة في الحاسب

الخوارزمية المستخدمة تسمى خوارزمية Mid-Point Circle Algorithm

خوارزمية MID-POINT CIRCLE ALGORITHM

هذه الخوارزمية أيضاً تعتمد على نقطة المنتصف

إذا مر خط الدائرة فوق نقطة المنتصف نختار البيكسل E

إذا مر خط الدائرة تحت نقطة المنتصف نختار البيكسل NE

معادلة الدائرة:

$$f(x,y) = x^2 + y^2 - R^2 = 0$$

$$F(x,y)\begin{cases} >0 & \text{for points outside the circle} \\ =0 & \text{for points on the circle} \\ <0 & \text{for points inside the circle} \end{cases}$$

$d \le 0$ نضيئ البكسل E

$d > 0$ نضيئ البكسل SE

ملاحظة: حساب ال d غير مطلوبة.

- خطوات الخوارزمية:

1. نبدأ من النقطة $(0,R)$
2. نضيئ البيكسل $(x,y) = (0,R)$
3. نحسب ال d
4. نضيئ النقاط بحسب d

ملاحظة: الفكرة دائما نقارن نقطة المنتصف مع خط الدائرة ويكون الخيار بين بيكسلين

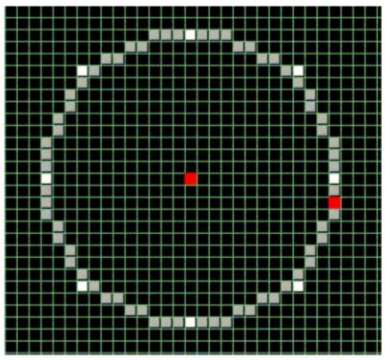
بكسل فوق E

بكسل تحت SE

بالاعتماد علة معادلة الدائرة نختار أحد البيكسلين لنضيئه.

الدائرة المرسومة بالاعتماد على الخوارزمية السابقة تعتبر مقبولة

مثال:



ملاحظة: لتحسين الدائرة نزيد الدقة (نقلل حجم البكسلات)

رسم خطوط بسماكة معينة

لرسم خطوط بسماكة معينة هناك 3 طرق:

1. Replication Pixels

ألاحظ حالة الميل $-1 < slope < +1$ نضيئ البكسلات ضمن نفس العامود.

مثلاً بيكسلين فوق البيكسل المحدد وبكسلين تحته.

أما إذا كان $slope > 1$ أو $slope < -1$ نضيئ البكسلا ضمن نفس السطر.

مثلاً بكلسيين لليسار وبكسلين لليمين من البيكسل المحدد.

ملاحظة: يمكن إضاءة البكسلات المكررة بنفس اللون أو بلون أقل كثافة.

2. Moving The Pen

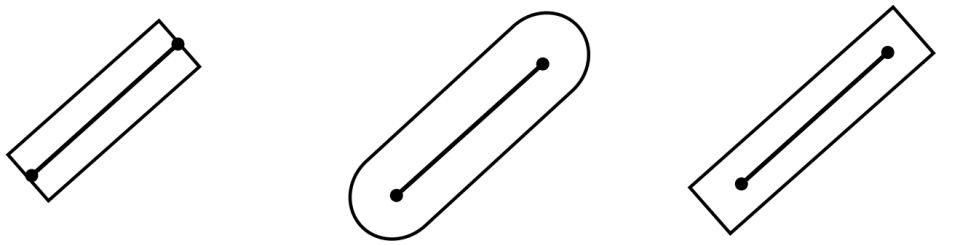
نختار مربع حول البيكسل المحدد ونضيئه كله.

ولكن مشكلتها أنه يمكن إضاءة البكسلات نفسها أكثر من مرة.

3. Filing Areas Between Boundaries

نحدد خطان يمثلان السماكة المطلوبة ونلون جميع البكسلات ضمن المسحة بين الخطين.

مشكلتها بالنهايات حيث يوجد عدة أنواع من النهايات



Butt caps Round caps Projecting square caps

- لرسم الخط المتقطع:

يتم تمثيله ثنائياً حيث اضيئ البكسلات عند ال 1 ولا اضيئ البكسلات عند ال 0 .

انتهـــــــت المحـــــــــاضـــــــــرة الثالثـــــــــة