# بيانيات الحاسوب
# Computer Graphics

جامعة المنارة
MANARA UNIVERSITY

Dr.-Eng. Samer Sulaiman

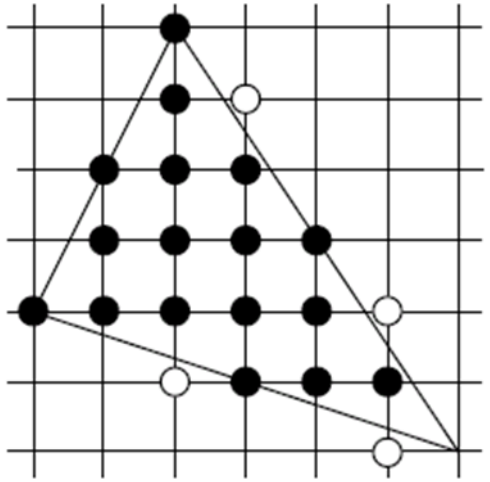
2022-2023

# مفردات المنهاج

- **أساسيات بيانيات الحاسوب**
  - مقدمة: مفاهيم أساسية
  - التحويلات ثنائية البعد 2D
  - التحويلات ثلاثية البعد 3D
- **خوارزميات بيانيات الحاسوب**
  - الخوارزميات الهندسية
  - **الخوارزميات النقطية الأساسية**
  - خوارزميات إزالة الأسطح المخفية
- **نماذج بيانيات الحاسوب**
  - الألوان ونماذج الألوان
  - المنحنيات والأسطح: النمذجة الهندسية
  - نماذج الإضاءة والتظليل
  - عناصر التركيب والنمذجة الإجرائية

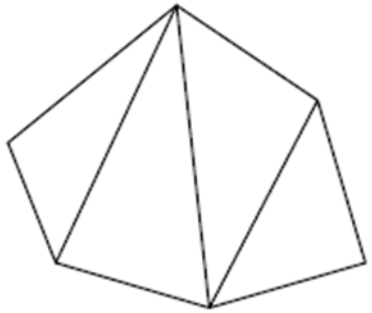
# خوارزميات بيانيات الحاسوب

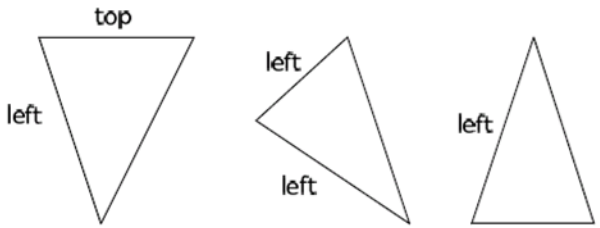
- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
  - ملء المثلث:
    - تتكون جميع الكائنات في رسومات الكمبيوتر ثلاثية الأبعاد من مثلثات ، لذا يتم استخدام تنقيط المثلث كثيرًا
    - كقاعدة عامة ، تدعم أجهزة الرسومات فقط المثلثات أو المضلعات المحدبة التي يمكن تحليلها بسهولة إلى مثلثات
    - أبسط الطرق لتنقيط المثلث:
      - تنقيط حوافها
      - ايجاد إحداثيات X القصوى (امتداد) لكل إحداثي Y
      - ملء كل واحد
    - يمكن أن ينتج عن استخدام كود تنقيط المستقيم التقليدي وحدات بكسل يمكن أن تكون بعضها خارج المثلث
      - وجود مثل هذه البكسلات أمر غير مرغوب فيه
        - التضارب مع تنقيط المثلثات المجاورة
        - يمكن أن تتسبب وحدات البكسل المتداخلة في حدوث تشوهات

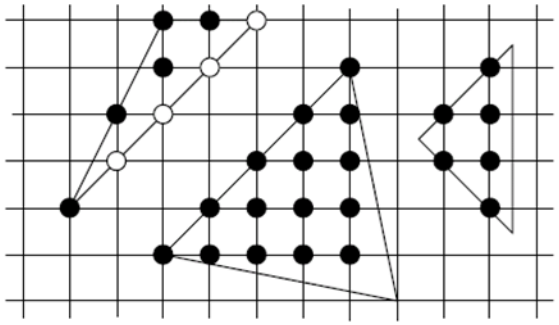


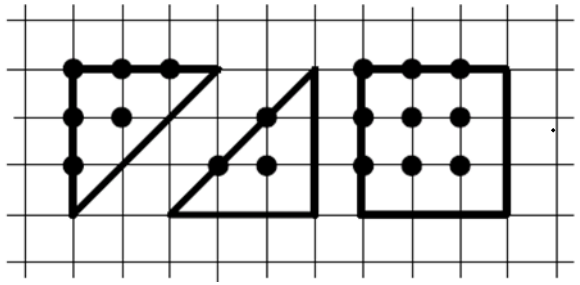
# خوارزميات بيانيات الحاسوب

- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
  - ملء المثلث:
    - تظهر مشكلة أخرى عند تنقيط شبكة مثلثات
      - يمكن أن يشترك مثلث واحد في الرؤوس المشتركة مع مثلثات أخرى
      - يجب ألا يؤدي تحويل مثل هذه الشبكة إلى ثقوب في مجموعة البكسل الناتجة
      - غير مرغوب فيه جدًا أن يتواجد بكسلات مكررة
        - عندما يتم إنتاج نفس البكسل عن طريق تنقيط العديد من المثلثات غير المتداخلة
    - الحل:
      - باستخدام قاعدة أعلى اليسار
        - يتم إنتاج البكسل إذا كان مركزه يقع داخل المثلث أو يقع على الحافة العلوية أو اليسرى
        - الحافة العلوية: هي الحافة الأفقية فوق كل حواف المثلث الأخرى
        - الحافة اليسرى: حافة غير أفقية على الجانب الأيسر من المثلث
        - يمكن أن يحتوي المثلث على حافة واحدة أو حافتين على اليسار







# خوارزميات بيانيات الحاسوب

- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
  - ملء المثلث:
    - الحل:
      - استخدام خوارزمية بسيطة ولكنها فعالة غالبًا ما تستخدمها أجهزة الرسومات
      - لنفترض أن $Ax + By + C = 0$ معادلة المستقيم الذي يمر عبر إحدى حواف المثلث
        - تقسيم المستوي إلى قسمين ، موجب (حيث $Ax + By + C > 0$) وسالب (حيث $Ax + By + C < 0$)
      - نقوم دائمًا بعمل المثلث بحيث يقع في نصف المستوى الموجب
        - إذا كان يقع في نصف المستوى السالب، يمكن تحويل المثلث إلى نصف المستوى الموجب بالمعادلة $(-A)x+(-B)y+(-C) = 0$
      - إذا كان لدينا مثل هذه المعادلات الخطية للحواف الثلاثة
        - المثلث نفسه هو تقاطع بين أنصاف المستويات الثلاثة المقابلة
      - إذا كانت لدينا نقطة $(x0, y0)$ ونريد التحقق مما إذا كانت تقع داخل المثلث
        - يمكن ببساطة استبدلها في معادلات الأسطر الثلاثة والتحقق مما إذا كنا سنحصل على ثلاثة أعداد موجبة (غير سالبة)
      - وبالتالي يمكن اعتماد المعاجلات التالية والتي تصف بشكل كامل المثلث

$$\begin{cases} f_1(x,y) = A_1x + B_1y + C_1 \geq 0, \\ f_2(x,y) = A_2x + B_2y + C_2 \geq 0, \\ f_3(x,y) = A_3x + B_3y + C_3 \geq 0. \end{cases}$$

        - إيجاد المربع المحيط بالمثلث $x_{min}$، $y_{min}$، $x_{max}$، $y_{max}$
        - لكل نقطة في هذا المربع ، يمكن التحقق مما إذا كانت تقع داخل المثلث
        - استخدام المعادلات أعلاه على أساس التوابع الخطية

# خوارزميات بيانيات الحاسوب

- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
  - ملء المثلث:
    - الحل:
      - استخدام خوارزمية بسيطة ولكنها فعالة غالبًا ما تستخدمها أجهزة الرسومات
      - الكود البرمجي للخوارزمية:
        - الحاجة إلى تابع تحسب معاملات معادلات المستقيم من نقطتين يمر بهما
        - اعتماد نقطة أخرى (ثالثة)- النقطة التي يجب أن تكون في نصف المستوى الموجب
        - تحدد هذه النقاط الثلاث بشكل فريد نصف المستوى (وهنا يجب أن يكون الموجب)
        - .

```
inline void buildPlane(
        const point& p1, const point& p2,
        const point& p3,
        int& a, int& b, int&c )
{
      // find line equations from p1 and p2
  a = p2.y - p1.y;
  b = p1.x - p2.x;
  c = p1.x*p2.y - p1.y*p2.x;

      // check whether p3 is in positive
  if ( a*p3.x+b*p3.y+c < 0 )
  {
    a = -a;
    b = -b;
    c = -c;
```

# خوارزميات بيانيات الحاسوب

- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
- ملء المثلث:
- الحل:
  - استخدام خوارزمية بسيطة ولكنها فعالة غالبًا ما تستخدمها أجهزة الرسومات
  - الكود البرمجي للخوارزمية:

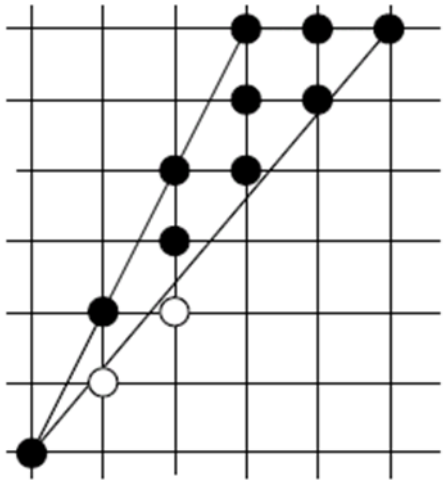
```
  }
}
```

Then the rasterizing code will look as shown below.

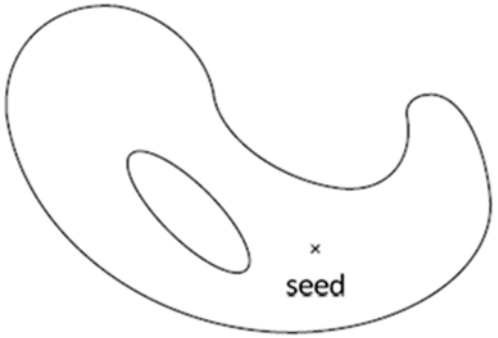
```
void rasterizeTriangle( point p [] )
{
  int xMin = p[0].x;
  int yMin = p[0].y;
  int xMax = p[0].x;
  int yMax = p[0].y;
  int a [3], b[3], c[3];

      // find bounding box
  for ( int i = 1; i < 3; i++ )
  {
    if ( p [i].x < xMin )
      xMin = p[i].x;
    else
    if ( p [i].x > xMax )
      xMax = p[i].x;

    if ( p [i].y < yMin )
      yMin = p[i].y;
    else
    if ( p [i].y > yMax )
      yMax = p[i].y;
  }

      // build line equations
  buildEquation(p[0], p[1], p[2], a[0], b[0], c[2]);
  buildEquation(p[0], p[2], p[1], a[1], b[1], c[1]);
  buildEquation(p[1], p[2], p[0], a[2], b[2], c[2]);

      // find functions at lower-left corner
  int d0 = a[0]*xMin+b[0]*yMin+c[0];
  int d1 = a[1]*xMin+b[1]*yMin+c[1];
  int d2 = a[2]*xMin+b[2]*yMin+c[2];

      // check points
  for ( int y = yMin; y <= yMax; y++ )
  {
    int f0 = d0,
        f1 = d1,
        f2 = d2:
        d0 += b[0];
        d1 += b[1];
        d2 += b[2];

        for ( int x = xMin; x <= xMax; x++ )
        {
          if ( f0 >= 0 && f1 >= 0 && f2 >= 0 )
            putPixel( x, y );

          f0 += a[0];
          f1 += a[1];
          f2 += a[2];
        }
      }
    }
```

# خوارزميات بيانيات الحاسوب

- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
  - ملء المثلث:
    - الحل:
      - استخدام خوارزمية بسيطة ولكنها فعالة غالبًا ما تستخدمها أجهزة الرسومات
      - في بعض الأحيان، قد تؤدي الحاجة إلى تضمين وحدات بكسل فقط داخل المثلث إلى حدوث عيوب غير مرغوب فيها
      - مما يؤدي إلى فقدان وحدات البكسل وعدم توصيل تمثيل نقطي للمثلث
      - ناتج عن زوايا رفيعة



- خوارزمية الملء (التعبئة) بالغمر Flood fill:
  - أكثر شيوعًا في الرسومات ثنائية الأبعاد
  - عادة ما يسمى بالغمر أو ملء الحدود
  - يستخدم في الحالة التي يكون فيها الشكل (المنطقة) المراد ملؤه معطى بلون حدوده ومعرفة نقطة معينة (نقطة البداية) داخل المنطقة
  - يستخدم للمساحة المعقدة التي بها ثقوب ، ولكن يجب أن تكون جميع وحدات البكسل الخاصة بها قابلة للوصول من بكسل نقطة البداية



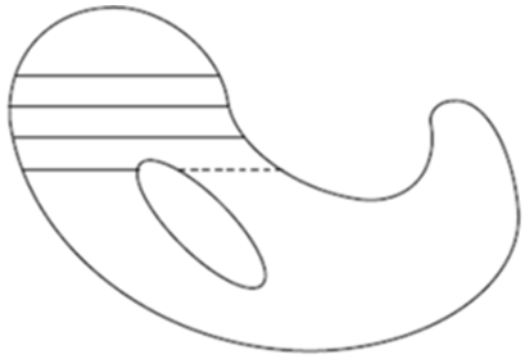
# خوارزميات بيانيات الحاسوب

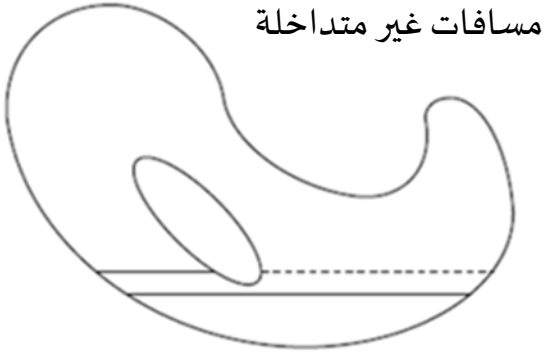
- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
  - خوارزمية الملء (التعبئة) بالغمر Flood fill:
    - يمكن كتابة أبسط خوارزمية تعبئة الحدود في بضعة أسطر من التعليمات البرمجية

```
void boundaryFill ( int x, int y, int borderColor,
                    int fillColor )
{
  int c = getPixel ( x, y );

  if ( (c != borderColor) && (c != fillColor) )
  {
    putPixel     ( x, y, fillColor );
    boundaryFill ( x - 1, y, borderColor, fillColor );
    boundaryFill ( x + 1, y, borderColor, fillColor );
    boundaryFill ( x, y - 1, borderColor, fillColor );
    boundaryFill ( x, y + 1, borderColor, fillColor );
  }
}
```

# خوارزميات بيانيات الحاسوب

- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
  - خوارزمية الملء (التعبئة) بالغمر Flood fill:
    - على الرغم من بساطتها، يمكن لهذه الخوارزمية التعامل مع منطقة معقدة للغاية
      - غير فعال للغاية ويؤدي إلى تكرار عميق للغاية
      - مناسب للاستخدام في المساحات الصغيرة فقط
    - بناء خوارزمية فعالة لملء الحدود boundary filling
      - يجب استخدام مبدأ التماسك coherence
        - إذا كان هناك بعض البكسل (x, y) داخل المنطقة المراد تعبئتها
        - من المحتمل أن تكون وحدات البكسل المجاورة لها داخل المنطقة أيضًا
    - ملء المنطقة بمسافات أفقية
      - بالنسبة للنقطة الحالية (x, y) داخل المنطقة ، نجد أقصى امتداد (xl, y)–(xr, y)، والذي يقع بالكامل داخل المنطقة ويحتوي على هذه النقطة
      - بعد أن يتم ملء الامتداد الحالي ، ننتقل إلى الأسفل
        - يتم فحص جميع وحدات البكسل الموجودة في السطر التالي والتي يمكن أن تؤدي إلى تكوين عدة مسافات غير متداخلة
      - لا ينبغي أن تنتقل إلى الأسفل فقط ، بل يجب أن تنتقل إلى الأعلى أيضًا
        - يكفي فحص وحدات البكسل فوق النطاق الحالي والتي لم تكن في النطاق السابق
    - تعتبر هذه الخوارزمية عودية لكن مستوى العودية ليس عميقًا جدًا





# خوارزميات بيانيات الحاسوب

- خوارزميات البيانات النقطية الأساسية Basic raster algorithms :
  - خوارزمية الملء (التعبئة) بالغمر Flood fill:
    - وظيفة: شرح الكود البرمجي للخوارزمية السابقة والمعطى كما يلي:

```
void boundaryFill ( int x, int y, int borderColor,
                    int fillColor )
{
  lineFill ( x, y, 1, x, x, borderColor, fillColor );
}

int lineFill (
    int x, int y, int dir, int prevXl, int prevXr,
    int borderColor, int fillColor )
{
  int xl = x;
  int xr = x;
  int c;

  do
     c = getPixel ( --xl, y );
  while ( (c != borderColor) && (c != fillColor) );

  do
     c = getPixel ( ++xl, y );
  while ( (c != borderColor) && (c != fillColor) );

  xl++;
  xr--;

  drawLine ( xl, y, xr, y );    // fill the span

  for ( x = xl; x <= xr; x++ )
  {
    c = getPixel ( x, y + dir );
    if ( (c != borderColor) && (c != fillColor) )
      x = lineFill ( x, y + dir, dir, xl, xr,
                     borderColor, fillColor );
  }

  for ( x = xl; x < prevXl; x++ )
  {
    c = getPixel ( x, y - dir );
    if ( (c != borderColor) && (c != fillColor) )
      x = lineFill ( x, y - dir, -dir, xl, xr,
                     borderColor, fillColor );
  }

  for ( x = prevXr; x < xr; x++ )
  {
    c = getPixel ( x, y - dir );
    if ( (c != borderColor) && (c != fillColor) )
      x = lineFill ( x, y - dir, -dir, xl, xr,
                     borderColor, fillColor );
  }

  return xr;
}
```