

الباب الثاني عشر

الرسم والألوان

مفتم

فى هذا الباب ، سوف نتجول مع دوال وأدوات الرسم المختلفة وسوف نتدرب على المهارات الآتية:

١. إعداد منطقة عرض (Display Context).
 ٢. إعداد أدوات الرسم وهى القلم (Pen) وفرشاة ملء المساحات (Brush).
 ٣. اختيار خصائص منطقة العرض (Attributes).
 ٤. استخدام دوال الرسم فى رسم الأشكال الهندسية المختلفة مثل الدوائر والخطوط والقطاعات والمضلعات.
 ٥. تفصيل أدوات رسم (فرشاة وقلم) حسب الطلب بدلا من استخدام الأدوات سابقة التعريف.
- كما نتعرف بالخطوات الأساسية لتطوير برنامج الرسم إلى الطراز النهائى باستخدام أدوات ودوال الرسم.



تذكر:

١. للاطلاع على الصيغة التفصيلية لإحدى الدوال أو الفصائل ، ضع مؤشر الفأر فوقها ثم اضغط زر اللوحة F1.
٢. لعرض الخريطة الكاملة لشجرة فصائل المؤسسة MFC ابحث فى شاشة النجدة (باستخدام البطاقة Search) عن "Hierarchy Chart".
٣. للاطلاع على الكود الكامل لأحد المشروعات ، افتح المشروع من القرص المصاحب للكتاب.

(١٢-١) الإعداد للرسم

مررنا من قبل بعملية رسم الخطوط في النافذة ، وعرفنا مفهوم منطقة العرض ، كما عرفنا أن المؤسسة MFC تمدنا بمنطقة عرض جاهزة وأدوات سابقة التعريف. ولك أن تستخدم منطقة العرض والأدوات سابقة التعرف ، أو تصنع لنفسك منطقة عرض خاصة وأدوات رسم خاصة. ونلخص هنا بسرعة الخطوات اللازمة للرسم في النافذة:

١. إعداد منطقة عرض (Display Context).
٢. أدوات رسم وهي القلم (Pen) وفرشاة ملء المساحات (Brush).
٣. اختيار خصائص منطقة العرض (Attributes).
٤. استخدام دوال الرسم.

Display Context

(١٢-٢) إعداد منطقة العرض

عندما تنشئ برنامجاً من برامج MFC فإن الدالة `OnDraw()` بفصيولة المشهد تحتوى على مؤشر منطقة العرض pDC الذى يمكنك استخدامه مباشرة ، أى:

```
void CMyProgramView::OnDraw(CDC* pDC)
{
    ... استدعاء دوال الرسم باستخدام المؤشر PDC ...
}
```

أما إذا كانت عملية الرسم تتم في نافذة خاصة لا تتحدر من فصيلة المشهد **CView** ، مثل صناديق الحوار (Dialog Boxes) فإنه من الضروري أن تنشئ المقيض **OnPaint()** (المناظر للرسالة **WM_PAINT**) ثم إنشاء منطقة عرض مشتقة من الفصيلة **CPaintDC** لاستخدامها في عمليات الرسم وإعادة الطلاء. ويتم إنشاء منطقة العرض من داخل الدالة **OnPaint()** كالآتي:

```
void CMyProgramView::OnPaint()
{
    CPaintDC PaintDC(this);
    ... PaintDC باستخدام الهدف
}
```

وسوف يلي شرح صناديق الحوار في الأبواب القادمة. ومن الجائز أيضاً أن تستدعي دوال الرسم من أى دالة أخرى خلاف **OnPaint()** أو **OnDraw()** ، وهذا يتطلب خلق منطقة عرض موروثه من الفصيلة **CClientDC** ، أى:

```
void CMyProgramView::MyDrawingFunction()
{
    CClientDC ClientDC(this);
    ... ClientDC باستخدام الهدف
}
```

(١٢-٣) إعداد أدوات الرسم

تتكون أدوات الرسم من القلم الذي يستخدم في رسم الخطوط والأشكال ومن الفرشاة التي تستخدم في عمليات الملء ، أى تلوين المساحات الداخلية. كما أن البنط المستخدم في الكتابة ينتمي أيضاً إلى أدوات الرسم ولكننا لن نتحدث عنه في هذه الفقرة. ويطلق على أدوات الرسم إجمالاً اسم أهداف GDI (GDI Objects) ، حيث يأتي الاختصار GDI من العبارة Graphical Device Interface.

أدوات الرسم سابقة التعريف

عندما تستخدم منطقة العرض فإنك - كما لاحظت من قبل - تحصل على قلم سابق التعريف يرسم باللون الأسود خطوطاً سمكها بكسلة واحدة. أما الفرشاة سابقة التعريف فلونها أبيض ، بمعنى أنها تملأ المساحات الداخلية للأشكال باللون الأبيض.

اختيار أدوات الرسم CDC::SelectStockObject()

يمكنك اختيار أداة رسم بخلاف الأداة سابقة التعريف باستخدام الدالة **SelectStockObject()** كالآتي:

```
CGdiObject* SelectStockObject(int nIndex);
```

أما البارامتر المستخدم للدالة فقد يأخذ أحد القيم الموضحة بالجدول التالي (لم نذكر هنا ثوابت البنط):

الثابت	معناه
BLACK_BRUSH	فرشاة سوداء
DKGRAY_BRUSH	فرشاة — رمادى داكن
GRAY_BRUSH	فرشاة — رمادى
LTGRAY_BRUSH	فرشاة — رمادى فاتح
NULL_BRUSH	فرشاة بلا لون (تناظر عدم ملء المساحات الداخلية)
WHITE_BRUSH	فرشاة بيضاء (اللون سابق التعريف)
BLACK_PEN	قلم أسود (اللون سابق التعريف)
NULL_PEN	قلم شفاف
WHITE_PEN	قلم أبيض

جدول (١٢-١) الاختيارات المتاحة من ثوابت القلم والفرشاة

مثال: لتحديد لون القلم الأسود والفرشاة ذات اللون الرمادى الفاتح
نستخدم العبارتين الآتيتين (من داخل الدالة (OnDraw):

```
pDC->SelectStockObject(BLACK_PEN);
pDC->SelectStockObject(LTGRAY_BRUSH);
```

كما رأينا فى هذه الفقرة أن أنواع الأقلام والفرش الجاهزة محدودة.
وبالطبع يمكنك تفصيل أدوات رسم على هواك ، وسوف نناقش
ذلك فى فقرة أخرى.

(١٢-٤) دوال الرسم

فى الفقرة التالية نقدم مجموعة من دوال الرسم التى يمكنك استخدامها مباشرة من داخل الدالة (**OnDraw()**) وهى جميعا تنتمى إلى الفصيلة **CDC**. وفى الجدول التالى نقدم ملخصا سريعا لهذه الدوال مع ربط الدوال بأداة الرسم التى تؤثر عليها (القلم أو الفرشاة).

الدالة	الشكل الناتج	الأداة التى تؤثر عليها
Arc	قوس	القلم
Chord	وتر	القلم والفرشاة
Ellipse	قطع مكافئ (بيضاوى)	القلم والفرشاة
LineTo	خط	القلم
Pie	فتيرة	القلم والفرشاة
Polyline	خط متعرج	القلم
PolyBezier	خط منحن	القلم
Polygon	مضلع مقفل	القلم والفرشاة
PolyPolygon	مضلعات متقاطعان أو متجاوران.	القلم والفرشاة
Rectangle	مستطيل	القلم والفرشاة
RoundRect	مستطيل ذو أركان مستديرة	القلم والفرشاة

جدول (١٢-٢) دوال الرسم

(١٢-٥) أمثلة على استخدام دوال الرسم

في هذه الفقرة سوف نقدم مجموعة من الأمثلة لاستخدام دوال الرسم ، وذلك بتقسيم الدوال إلى مجموعات متشابهة. وفي هذه الأمثلة ، لاحظ أن كل التعديلات المطلوبة سوف تقع في الدالة **OnDraw()** بفصيلة المشهد.

مثال (١) المستطيلات والأشكال البيضاوية (& Rectangles Ellipses)

ابدأ مشروعاً جديداً بالاسم Rectangles ، ثم أضف إليه الأجزاء الموضحة بالبنط الأسود في الشكل التالي. ويوجد هذا البرنامج على القرص المصاحب للكتاب بالدوسيه Rectangles.

```
void CRectanglesView::OnDraw(CDC* pDC)
{
    CRectanglesDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    // Select Drawing Tools:                اختيار أدوات الرسم
    // pDC->SelectStockObject(BLACK_PEN);
    // pDC->SelectStockObject(LTGRAY_BRUSH);
    //                                        شحن الإحداثيات

    int Inc=50;
    int x1=120;
    int x2=x1+3*Inc;
    int y1=50;
    int y2=y1+4*Inc;

    // Rectangle:                            رسم مستطيل
    pDC->Rectangle(x1,y1,x2,y2);
    // Round Rectangle:                      رسم مستطيل ذي أركان مستديرة
```

```

int R=50;
pDC->RoundRect(x1+Inc,y1+Inc,
                x2+Inc,y2+Inc,
                R,R);

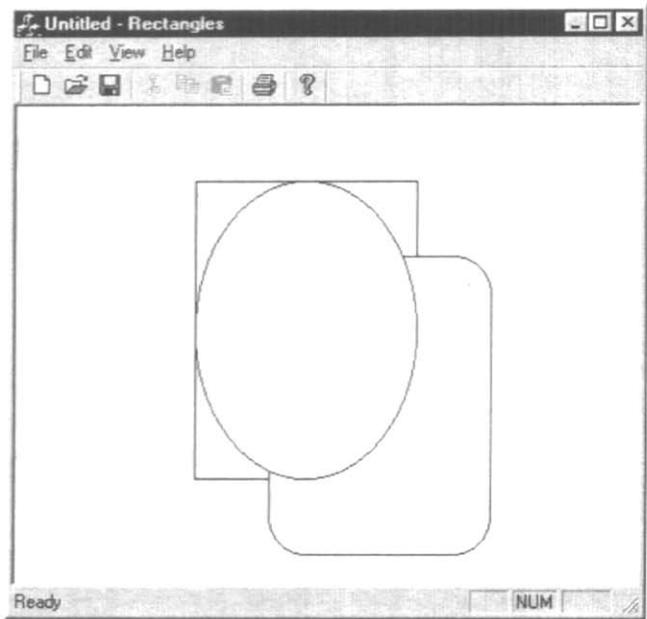
// Ellipse:
pDC->Ellipse(x1,y1,x2,y2);
}

```

رسم بيضاوي

مناقشة البرنامج:

١. عند تشغيل هذا البرنامج فإنك تحصل على الشكل التالي:



شكل (١-١٢) نتيجة تنفيذ البرنامج Rectangles

٢. يبدأ البرنامج باختيار أدوات الرسم بالعبارتين:

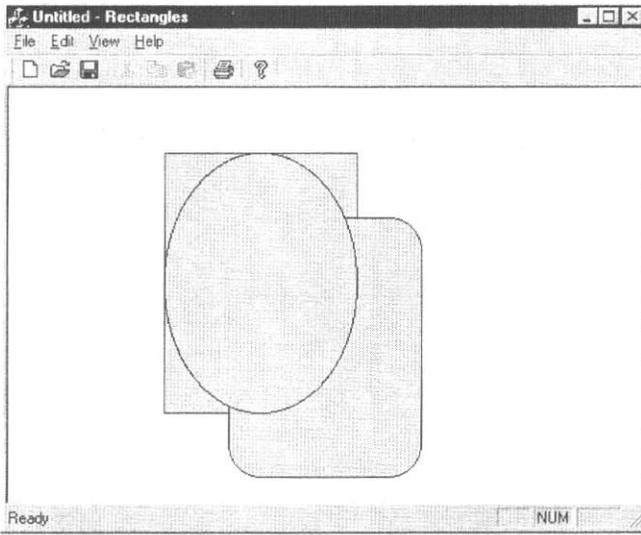
```

// pDC->SelectStockObject(BLACK_PEN);
// pDC->SelectStockObject(LTGRAY_BRUSH);

```

ومع ذلك فإننا قد وضعنا أمام هذه العبارات علامات تعليق ، أي أننا لم نستخدمها بالفعل ، وإنما قد اكتفينا بأدوات الرسم سابقة

التعريف. ولذلك نرى أن الأشكال المرسومة تستخدم القلم الأسود الذي سمكه بكسلة واحدة ، كما أن الأشكال جاءت مفرغة. جرب أن تحذف علامات التعليق ، وسوف تشاهد تغير الأشكال المرسومة إلى الشكل التالي حيث نرى أن الفرشاة قامت بملء الأشكال باللون الرمادي الفاتح.



شكل (١٢-٢) نتيجة تنفيذ البرنامج Rectangles بعد اختيار أدوات الرسم

٣. استخدمنا في هذا البرنامج الدالة **Rectangle()** لرسم المستطيل ، وهي تأخذ الصورة التالية

```
BOOL Rectangle(int x1,int y1,int x2,int y2);
```

حيث:

x1,y1: إحداثي الركن الأيسر العلوي – القيمة المستخدمة 120,50

x2,y2: إحداثي الركن الأيمن السفلي – القيمة المستخدمة 200,250

والقيم الفعلية المستخدمة هنا تقاس بالبكسل ، وهي الوحدة سابقة التعريف.

٤. استخدمنا الدالة **RoundRect()** لرسم المستطيل ذي الأركان الدائرية ، وهي تأخذ الصورة التالية:

BOOL RoundRect(int x1,int y1,int x2,int y2,int x3,int y3);

حيث:

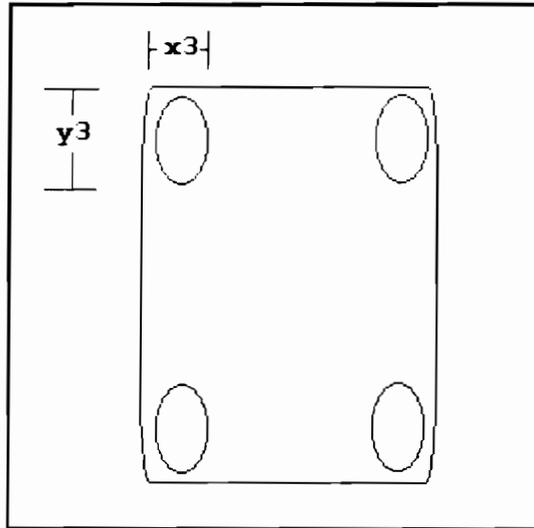
$x1,y1$: إحداثي الركن الأيسر العلوي

$x2,y2$: إحداثي الركن الأيمن السفلي

$x3$: القطر الأفقى للبيضاوى الذى يحدد تقوس ركن المستطيل

$y3$: القطر الرأسى للبيضاوى الذى يحدد تقوس ركن المستطيل

أنظر الشكل التالى للتوضيح.



شكل (١٢-٣) تحديد مقدار التقوس للأركان

وفى البرنامج السابق فإننا قد استخدمنا قيما متساوية لكل من $x3$ و $y3$ (المتغير R) فأصبح مقدار التقوس متساويا.
٥. استخدمنا الدالة $Ellipse()$ لرسم البيضاوى ، وهى تأخذ الصورة الآتية:

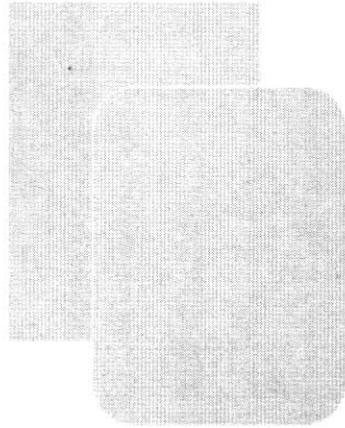
```
BOOL Ellipse(int x1,int y1,int x2,int y2);
```

وتمثل المتغيرات $x1,y1,x2,y2$ أبعاد المستطيل الوهمى الذى يقع بداخله الشكل البيضاوى.
أى:

$x1,y1$: إحداثى الركن الأيسر العلوى
 $x2,y2$: إحداثى الركن الأيمن السفلى

تدريب:

أجر التعديل اللازم على البرنامج السابق حتى تحصل على مساحات ملونة بدون الإطارات الخارجية كما بالشكل التالى:



تدريب:

مثال (٢) النقط والخطوط (Points and Lines)

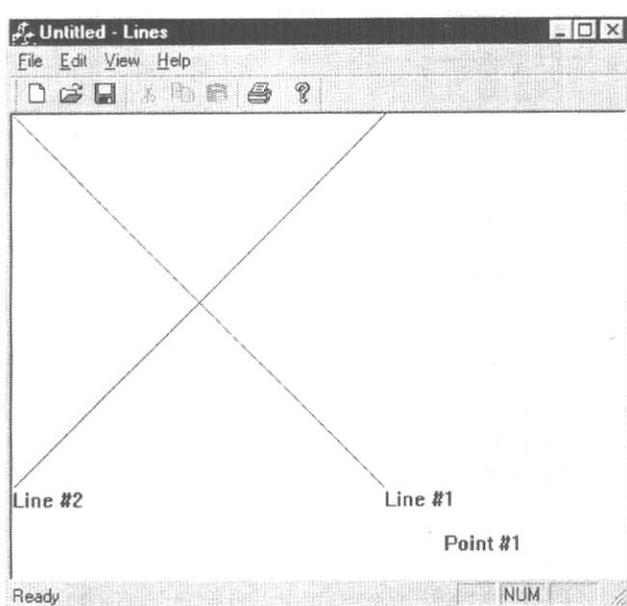
في هذا المثال نعرض دالة رسم النقطة (`SetPixelV()`) والتي يمكن استخدامها لرسم المستقيمت أيضا ، كما نقارن هذه المستقيمت بالمستقيمت الى رسمناها من قبل بالاستعانة بالدوال (`LineTo()`) و `MoveTo()`.

افتح مشروعنا جديدا بالاسم Lines ، ثم أضف الكود الموضح باللبنت الأسود في الشكل التالي. ويوجد هذا البرنامج على القرص المصاحب للكتاب بالدوسيه Lines.

```
void CLinesView::OnDraw(CDC* pDC)
{
    CLinesDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    int x=280, y=280;          إحدائي النقطة
    // 1. Draw a point:        رسم النقطة ، وكتابة العبارة Point #1 عند موقعها
    pDC->SetPixelV(x,y,RGB(255,0,0));
    pDC->TextOut(x+10, y, "Point #1");
    // 2. Draw a line of points: رسم خط مكون من مجموعة نقط
    for (x=1, y=1; x<250; x++) {
        pDC->SetPixelV(x,y,RGB(255,0,0));
        y++;
    }
    pDC->TextOut(x, y, "Line #1");  كتابة عبارة في نهاية الخط
    // 3. Using MoveTO and LineTo: رسم خط بالطريقة المعتادة
    x=250;
    y=0;
    pDC->MoveTo(x,y);
    x=x-250;
    y=y+250;
    pDC->LineTo(x,y);
    pDC->TextOut(x, y, "Line #2");  كتابة عبارة في نهاية الخط
}
```

مناقشة البرنامج:

١. عند تشغيل هذا البرنامج تحصل على الشكل التالي ، وهو
يحتوى على نقطة (Point #1) ، وخطين. الخط الأول (Line#1)
باللون الأحمر ، والخط الثانى (Line#2) باللون سابق التعريف
(الأسود):



شكل (١٢-٥) تنفيذ البرنامج – مثال (٢)

٢. العبارة التى تؤدى إلى رسم نقطة (بلون معين) على الشاشة
هى:

BOOL SetPixelV(int x, int y, COLORREF color);

حيث:

x,y: إحداثي النقطة.

color: لون النقطة وهو متغير من النمط **COLORREF** يمكن تحديده بالماكرو **RGB(R, G, B)** الذى يستخدم ثلاثة بارامترات R, G, B تمثل اللون الأحمر والأخضر والأزرق بالترتيب. وفى هذا المثال قد استخدمنا اللون الأحمر بمنح المتغير R أقصى قيمة للون ، وهى 255 .

`SetPixelV(x,y,RGB(255,0,0));`

وفيما يلى القيم المناظرة لدرجات تشبع الألوان الأساسية الثلاثة. وبتغيير درجة تشبع كل لون (باستخدام قيمة تتراوح ما بين 0 و 255) يمكنك أن تحصل على خلطة جديدة من الألوان.

B	G	R	اللون
0	0	255	الأحمر
0	255	0	الأخضر
255	0	0	الأزرق

جدول (١٢-٣) نسب الألوان الأساسية

وحتى تتمكن من توليف اللون المطلوب (بجانب المحاولة والخطأ)، فإنه من الضرورى أن تعرف أن لكل لون من الألوان الأساسية لون مقابل (عكسى). فاللون الأصفر مثلا هو اللون المقابل للأزرق وينتج عندما تكون درجة الأزرق 0 وتكون الألوان الأخرى مشبعة (255). والجدول التالى يلخص مواصفات الألوان المقابلة ، ونلاحظ

أن الأرقام الموجودة بالجدول عبارة عن "عكس" الأرقام الموجودة بالجدول السابق ، أى أن الرقم 0 يصبح 255 والعكس بالعكس.:

B	G	R	اللون
255	255	0	سايان (Cyan) – عكس الأحمر ملاحظة: اللون "سايان" عبارة عن درجة من درجات اللون السماوى.
255	0	255	ماجينتا (Magenta) – عكس الأخضر ملاحظة: اللون "ماجينتا" عبارة عن درجة من درجات اللون البنفسجى.
0	255	255	الأصفر _ عكس الأزرق

جدول (١٢-٤) نسب الألوان المقابلة للألوان الأساسية (العكسية)



للاطلاع على أنواع البيانات بالمؤسسة MFC ارجع إلى الملحق ب فى نهاية الكتاب.

٣. بجانب رسم النقطة فى موضع معين فقد كتبنا بجوارها النص "Point #1" وذلك لمجرد توضيح مكانها ، فالنقطة مجرد بكسلة واحدة وقد يتعذر رؤيتها على الشاشة. والدالة المستخدمة هنا هى `TextOut()` التى سبق شرحها فى الأبواب المتقدمة:

```
TextOut(x+10, y, "Point #1");
```

٤ . باستخدام دالة رسم النقطة ، رسمنا مستقيماً ملونا من خلال حلقة تكرارية كالآتي:

```
for (x=1, y=1; x<250; x++) {  
    pDC->SetPixelV(x,y,RGB(255,0,0));  
    y++;  
}
```

ثم كتبنا عبارة مميزة في نهاية هذا الخط:

```
pDC->TextOut(x, y, "Line #1");
```

٥ . أما الخط الثاني فقد رسمناه بالطريقة المعتادة باستخدام

الدوال **MoveTo()** و **LineTo()** ثم كتبنا في نهايته العبارة

"Line#2". ونلاحظ أن هذا الخط قد تم رسمه بالأدوات سابقة

التعريف ، أي باللون الأسود وبسمك بكسلة واحدة.

```
x=250;  
y=0;  
pDC->MoveTo(x,y);  
x=x-250;  
y=y+250;  
pDC->LineTo(x,y);  
pDC->TextOut(x, y, "Line #2");
```

مثال (٣) المضلعات (Polygons)

في هذا المثال نعرض دوال رسم الأشكال المضلعة والخطوط غير

المنتظمة باستخدام الدوال **Polygon()** و **PolyBezier()** و

PolyLine().

افتح مشروعاً جديداً بالاسم **Polygons** ، ثم أضف الكود الموضح

بالبنط الأسود في الشكل التالي. ويوجد هذا البرنامج على القرص

المصاحب للكتاب بالدوسيه **Polygons**.

```

void CPolygonsView::OnDraw(CDC* pDC)
{
    CPolygonsDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    int Start = 50, Inc = 70;
    // Polyline:
    POINT P[5];
    P[0].x = Start;
    P[0].y = Start;
    P[1].x = Start+Inc;
    P[1].y = Start+2*Inc;
    P[2].x = Start+2*Inc;
    P[2].y = Start+Inc;
    P[3].x = Start+3*Inc;
    P[3].y = Start+2*Inc;
    P[4].x = Start+4*Inc;
    P[4].y = Start+Inc;
    pDC->Polyline(P,5);
    // Irregular Curve:
    pDC->PolyBezier(P,4);
    // Polygon:
    Start=200;
    P[0].x = Start;
    P[0].y = Start;
    P[1].x = Start+Inc;
    P[1].y = Start+2*Inc;
    P[2].x = Start+2*Inc;
    P[2].y = Start+Inc;
    pDC->Polygon(P,3);
}

```

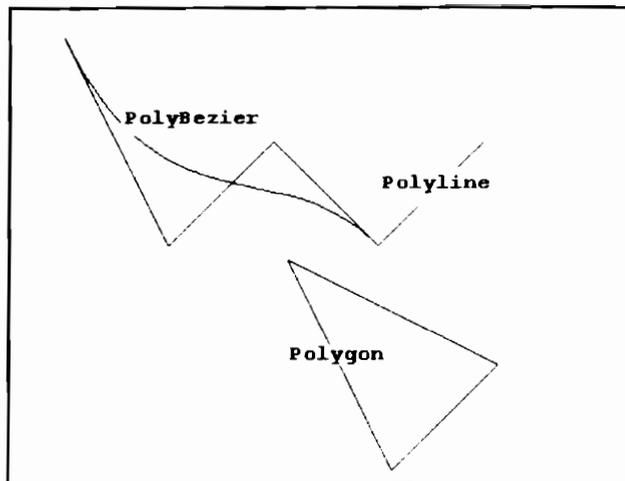
خط متعرج

الخط المنحني

مضلع مقل

المناقشة:

١. عند تشغيل هذا البرنامج تحصل على الشكل التالي (بدون التعليقات):



شكل (١٢-٦) تنفيذ البرنامج Polygons

٢. تم رسم الخط المتعرج باستخدام الدالة Polyline() التي تأخذ الصورة:

```
BOOL Polyline(LPPOINT P, int n);
```

حيث:

P: مصفوفة من النقاط تمثل إحداثيات النقاط التي يصل بينها الخط المتعرج.

n: عدد النقاط المطلوب توصيلها بالخط المتعرج. وقد استخدمنا هنا الرقم 5 ويجوز أن تستخدم أى رقم آخر طالما أنه أقل أو يساوى عدد عناصر المصفوفة.

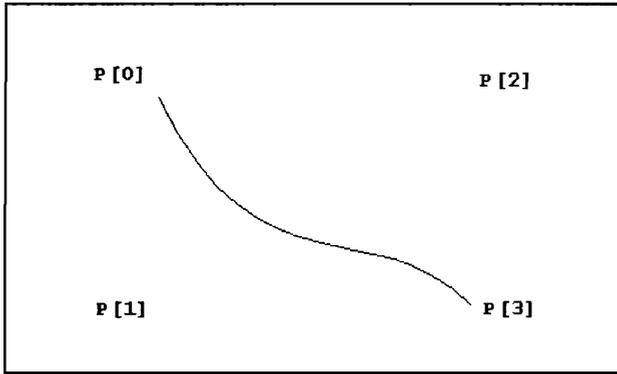
٣. تم رسم المنحنى الذي يصل بين نفس المجموعة من النقاط باستخدام الدالة PolyBezier() التي تأخذ الصورة:

```
BOOL PolyBezier(const POINT* P, int n);
```

حيث:

P: مؤشر إلى مصفوفة النقط تمثل إحداثيات النقط التي يصل بينها الخط المنحنى.

n: عدد النقط المطلوب توصيلها بالخط المنحنى. وقد استخدمنا هنا الرقم 4 وهو أقل عدد يمكن استخدامه مع هذه الدالة لرسم منحنى واحد. ويوضح الشكل التالى دور النقط الأربعة فى رسم هذا المنحنى ، حيث نرى أن المنحنى يصل بين النقطة الأولى P[0] والنقطة الرابعة P[3]. أما النقطة الثانية والثالثة فتحددان مدى تقوس المنحنى.



شكل (١٢-٧) دور النقط الأربعة فى رسم المنحنى PolyBezier

أما الرقم التالى الذى يمكن استخدامه مع هذه الدالة فهو الرقم 7 ، وهو يضيف منحنياً جديداً باستخدام ثلاث نقط جديدة ، أما النقطة الرابعة فسوف تكون النقطة المشتركة P[3].

٤. أما الشكل المضلع فقد رسمناه باستخدام الدالة Polygon() التى تأخذ الصورة:

BOOL Polygon(LPPOINT P, int n);

حيث:

P: مصفوفة من النقط تمثل إحداثيات النقط التي يصل بينها المضلع.

n: عدد النقط المطلوبة التي تمثل أركان المضلع. وقد استخدمنا هنا الرقم 3 ويجوز أن تستخدم أى رقم آخر طالما أنه أقل أو يساوى عدد عناصر المصفوفة.

تدريب:

• ارسم منحنيًا بالدالة PolyBezier() يستخدم ٧ نقط من مصفوفة النقط.

• ارسم مستطيلاً باستخدام دالة المضلع Polygon().

تدريب:

مثال (٤) القطاعات والأقواس (Arcs, chords, and Pies)

فى هذا المثال نعرض دوال رسم الأشكال الأقواس والقطاعات والفطائر باستخدام الدوال Arc() و Chord() و Pie().
افتح مشروعاً جديداً بالاسم Sectors ، ثم أضف الكود الموضح بالبنط الأسود فى الشكل التالى. ويوجد هذا البرنامج على القرص المصاحب للكتاب بالدوسيه Sectors.

```
void CSectorsView::OnDraw(CDC* pDC)
{
    CSectorsDoc* pDoc = GetDocument();
```

```

ASSERT_VALID(pDoc);
// TODO: add draw code for native data here
// Arc:
int x1=10,y1=10;
int x2=250,y2=250;
int x3=50,y3=0;
int x4=250,y4=200;
pDC->Arc(
    x1,y1,
    x2,y2,
    x3,y3,
    x4,y4);
// Chord:
int IncX=250, IncY=10;
x1=x1+IncX; y1=y1+IncY;
x2=x2+IncX; y2=y2+IncY;
x3=x3+IncX; y3=y3+IncY;
x4=x4+IncX; y4=y4+IncY;
pDC->Chord(
    x1,y1,
    x2,y2,
    x3,y3,
    x4,y4);
// Pie:
IncX=10, IncY=150;
x1=x1+IncX; y1=y1+IncY;
x2=x2+IncX; y2=y2+IncY;
x3=x3+IncX; y3=y3+IncY;
x4=x4+IncX; y4=y4+IncY;
pDC->Pie(
    x1,y1,
    x2,y2,
    x3,y3,
    x4,y4);
}

```

القوس

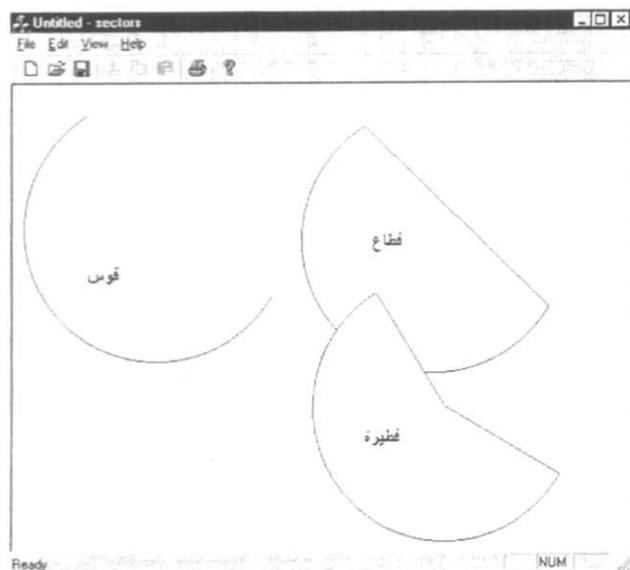
القطاع

القطيرة

المناقشة:

١. عند تشغيل هذا البرنامج تحصل على الشكل التالي (بدون

التعليقات):

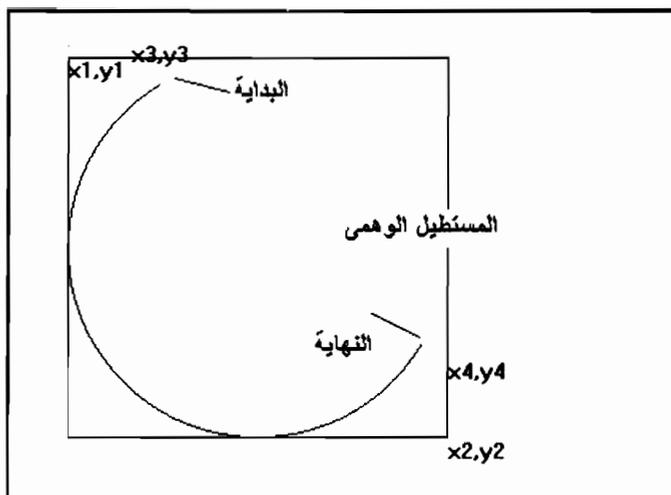


شكل (١٢-٨) تنفيذ البرنامج - مثال ٤

٢. استخدمنا هنا الدالة $Arc()$ لرسم القوس ، وهي تأخذ الصورة التالية:

```
BOOL Arc(int x1, int y1, int x2, int y2, int x3, int y3,  
int x4, int y4);
```

حيث تحدد الإحداثيات $(x1, y1)$ و $(x2, y2)$ ركني المستطيل الوهمي المحدد للقوس ، أى الركن الأيسر العلوى والركن الأيمن السفلى. كما تحدد الإحداثيات $(x3, y3)$ و $(x4, y4)$ بداية ونهاية القوس ، كما هو موضح بالشكل التالى.



شكل (١٢-٩) إحداثيات القوس Arc

٣. استخدمنا الدالة **Chord()** لرسم القطاع (جزء من دائرة) وصيغتها كالتالي:

```
BOOL Chord(int x1, int y1, int x2, int y2, int x3, int y3,  
int x4, int y4);
```

حيث تحدد الإحداثيات $(x1, y1)$ و $(x2, y2)$ ركني المستطيل الوهمي المحدد لدائرة القطاع ، أى الركن الأيسر العلوى والركن الأيمن السفلى. كما تحدد الإحداثيات $(x3, y3)$ و $(x4, y4)$ بداية ونهاية القطاع (قارن بطريقة رسم القوس **Arc()**).

٤. استخدمنا الدالة **Pie()** لرسم الفطيرة (الفطيرة عبارة عن قطاع ولكنه يأخذ شكل شطيرة خبز) ، وصيغة الدالة كالتالي:

```
BOOL Pie(int x1, int y1, int x2, int y2, int x3, int y3,  
int x4, int y4);
```

حيث تحدد الإحداثيات (x_1, y_1) و (x_2, y_2) ركني المستطيل الوهمي المحدد لدائرة الفطيرة ، أى الركن الأيسر العلوى والركن الأيمن السفلى. كما تحدد الإحداثيات (x_3, y_3) و (x_4, y_4) بداية ونهاية قوس الفطيرة (قارن بطريقة رسم القوس $(Arc())$).

تدريب:

باستخدام دوال الرسم ، أضف ما يلزم إلى البرنامج السابق (مثال ٤) لرسم المستطيلات الوهمية ومكان الإحداثيات الأربعة لكل من الفطيرة والقطاع ، بحيث تحصل على شكل مماثل للشكل (٩-١٢).

تدريب:

(١٢-٦) تفصيل أدوات الرسم

كما رأينا فى الفقرات السابقة أن أنواع الأقلام والفرش الجاهزة محدودة. وفى هذه الفقرة سوف نرى كيف يمكنك تفصيل أدوات الرسم على هوك. وهذا هو ملخص خطوات إعداد أدوات الرسم ذات المواصفات الخاصة:

١. أنشئ هدفاً من كل من فصيلتى القلم (CPen) والفرشاة (CBrush). مثال:

CPen Pen; // إعلان هدف قلم

CBrush Brush // إعلان هدف فرشاة

٢. احفظ القلم القديم والفرشاة القديمة باستخدام مؤشر إليهما:

CPen* PtrOldPen; // حفظ مؤشر إلى القلم السابق

CBrush* PtrOldBrush; // حفظ مؤشر إلى الفرشاة السابقة

٣. اشحن أهداف القلم والفرشاة باستخدام الدوال المناسبة. سيلي عرضها.

٤. أضف القلم الجديد والفرشاة الجديدة إلى منطقة العرض:

PtrOldPen = pDC->SelectObject(&Pen); // استبدال القلم القديم بالجديد

PtrOldBrush = pDC->SelectObject(&Brush); // استبدال الفرشاة القديمة بالجديدة

٥. استخدم دوال الرسم فى إخراج الرسم المطلوب - سبق عرضها.

٦. أعد القلم القديم والفرشاة القديمة إلى منطقة العرض (أو بمعنى آخر

حذف الأدوات الجديدة):

pDC->SelectObject(PtrOldPen); // استعادة القلم القديم

pDC->SelectObject(PtrOldBrush); // استعادة الفرشاة القديمة

تحديد خصائص القلم

يتم شحن القلم بالمواصفات الخاصة باستخدام الدالة

CPen::CreatePen() التي تأخذ الصورة الآتية:

BOOL CreatePen(int style, int width, COLORREF color);

حيث:

width: سمك القلم (القيمة صفر تناظر بكسلة واحدة)
color: متغير اللون. كما سبق أن أوضحنا ، فإن قيمة المتغير من النمط **COLORREF** يتم تحديدها بالماكرو **RGB(R, G, B)**.
style: طراز القلم ، وهو يأخذ أحد القيم الثابتة الموضحة بالجدول التالي:

الخاصية	طراز القلم
خط صلب – سابق التعريف	PS_SOLID
خط مكون من شرط	PS_DASH
خط منقط	PS_DOT
خط مكون من نقط وشرط متتابعة	PS_DASHDOT
خط مكون من تكرار شرط تتبعتها نقطتان	PS_DASHDOTDOT
خط شفاف	PS_NULL
يختص هذا النوع بالألوان غير النقية ولن نتعرض له في موضوعنا	PS_INSIDEFRAME

جدول (١٢-٥) طرازات القلم

مثال: قلم يرسم خطا صلبا باللون الأزرق وسمكه 5:

```
Pen.CreatePen(PS_SOLID, 5, RGB(0,0,255));
```

تحديد خصائص الفرشاة

يمكن اختيار الفرشاة وتحديد خصائصها بأحد الدوال التالية ، وهى جميعاً من أعضاء الفصيلة **CBBrush**:

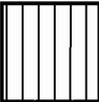
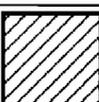
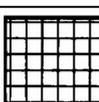
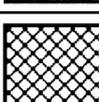
١. لملأ المساحات بلون واحد:

BOOL CreateSolidBrush(COLORREF color);

٢. لملأ المساحات المقفلة بلون مهشّر (Hatched):

BOOL CreateHatchBrush(int index, COLORREF color);

ويمثل البارامتر index نوع الشكل المستخدم فى التهشير (التظليل باستخدام الخطوط) كما بالجدول التالى. ويجوز استخدامه كرقم من 0 إلى 5 ، أو استخدام الثابت المسمى المناظر.

الرقم	اسم الثابت	الشكل
0	HS_HORIZONTAL	
1	HS_VERTICAL	
2	HS_FDIAGONAL	
3	HS_BDIAGONAL	
4	HS_CROSS	
5	HS_DUAGCROSS	

جدول (١٢-٦) طرازات فرشاة التهشير

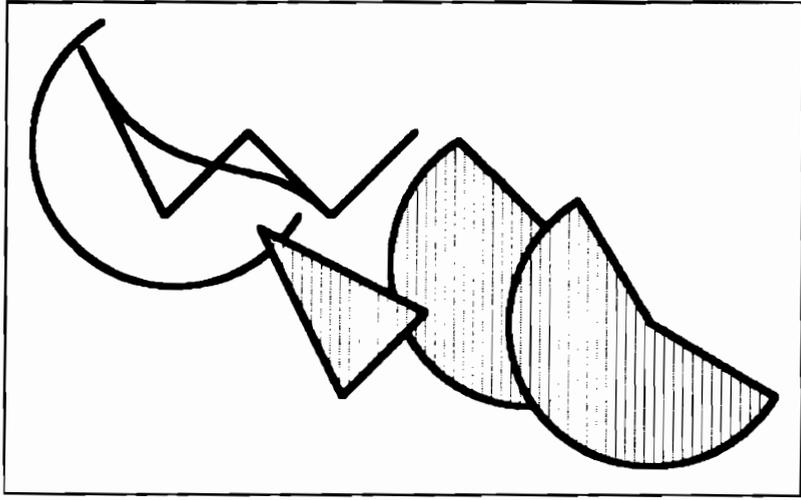
٣. يمكنك أيضاً استخدام الصور الجاهزة (BITMAP) لملأ الخلفية باستخدام الدالة:

```
BOOL CreatePatternBrush(CBitmap* pBitmap);
```

وفى هذه الصيغة فإن البارامتر المستخدم عبارة عن مؤشر إلى الصورة.

مثال (٥) مثال عام باستخدام القلم والفرشاة الخاصة

فى هذا المثال نجمع جميع مهارات الرسم معا فى شكل واحد ، كما نستخدم قلماً خاصاً لونه أزرق وسمكه خمسة بكسلات ، وفرشاة تملأ المساحات المقفلة بخطوط حمراء رأسية. كما بالشكل التالى.



شكل (١٢-١٠) نتيجة تنفيذ المثال (٥)

وفيما يلي نص الكود بالدالة **OnDraw()** ، أما البرنامج فهو موجود على القرص تحت الاسم Figures.

```
void CFiguresView::OnDraw(CDC* pDC)
{
    CFiguresDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    CPen Pen;
    CBrush Brush;
    CPen *PtrOldPen;
    CBrush *PtrOldBrush;
    // Solid blue pen, 5 pixels wide:
    Pen.CreatePen(PS_SOLID, 5, RGB(0,0,255));
    //Using red brush #1:
    Brush.CreateHatchBrush(1, RGB(255,0,0));
    // Select pen and brush:
    PtrOldPen = pDC->SelectObject(&Pen);
    PtrOldBrush = pDC->SelectObject(&Brush);
    int Start = 50, Inc = 70;
    // Polyline:
    POINT P[5];
    P[0].x = Start;
    P[0].y = Start;
    P[1].x = Start+Inc;
    P[1].y = Start+2*Inc;
    P[2].x = Start+2*Inc;
    P[2].y = Start+Inc;
    P[3].x = Start+3*Inc;
    P[3].y = Start+2*Inc;
    P[4].x = Start+4*Inc;
    P[4].y = Start+Inc;
    pDC->Polyline(P,5);
    // Irregular Curve:
    pDC->PolyBezier(P,4);
    // Sector:
    int x1=10,y1=10;
    int x2=250,y2=250;
    int x3=50,y3=0;
    int x4=250,y4=200;
    pDC-> Arc(
            x1,y1,
            x2,y2,
            x3,y3,
            x4,y4);
    // Chord:
```

```

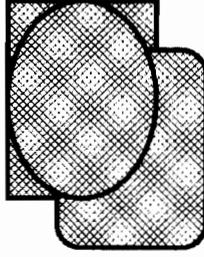
int IncX=300, IncY=100;
x1=x1+IncX; y1=y1+IncY;
x2=x2+IncX; y2=y2+IncY;
x3=x3+IncX; y3=y3+IncY;
x4=x4+IncX; y4=y4+IncY;
pDC->Chord(
        x1,y1,
        x2,y2,
        x3,y3,
        x4,y4);
// Polygon:
    Start=200;
    P[0].x = Start;
    P[0].y = Start;
    P[1].x = Start+Inc;
    P[1].y = Start+2*Inc;
    P[2].x = Start+2*Inc;
    P[2].y = Start+Inc;
    pDC->Polygon(P,3);
// Pie:
    IncX=100, IncY=50;
    x1=x1+IncX; y1=y1+IncY;
    x2=x2+IncX; y2=y2+IncY;
    x3=x3+IncX; y3=y3+IncY;
    x4=x4+IncX; y4=y4+IncY;
    pDC->Pie(
            x1,y1,
            x2,y2,
            x3,y3,
            x4,y4);
// Remove the pen and the brush:
    pDC->SelectObject(PtrOldPen);
    pDC->SelectObject(PtrOldBrush);
}

```

تدريب:

أعد كتابة الأمثلة السابقة من ١ إلى ٤ باستخدام قلما ذا سمك ٣ و لون أزرق وفرشاة تملأ المساحات المقفلة بخطوط متقطعة. توجد إجابة تمرين المستطيلات تحت الاسم Rectangles1 على القرص المصاحب. والشكل التالي يوضح نتيجة التنفيذ المطلوبة.

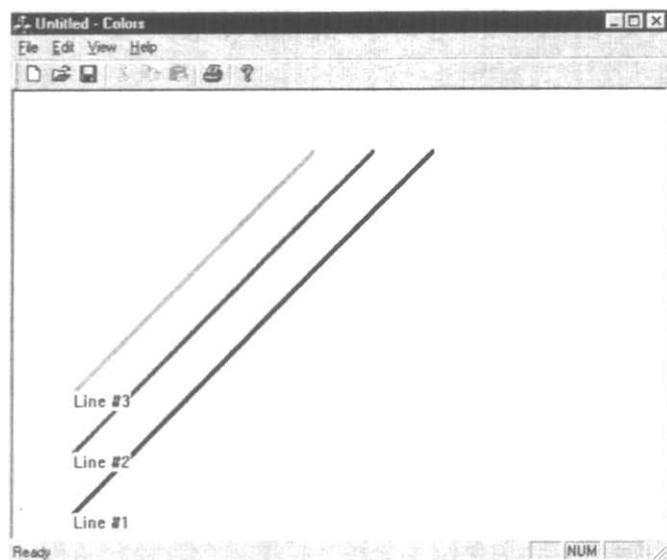
تدريب: _____



مثال (٦) استخدام أكثر من قلم واحد في نفس البرنامج

يمكنك بالطبع أن تستخدم أكثر من قلم واحد أو فرشاة واحدة ، ولكن يجب أن تتذكر أن كل أداة من أدوات الرسم تعتبر هدفاً مستقلاً يشغل من الذاكرة حيزاً ما ويلزم حذفه عند الانتهاء من استخدامه.

في المثال التالي نرسم ثلاثة خطوط ملونة كل خط بلون مختلف ، باستخدام ثلاثة أقلام مختلفة كما هو موضح بالشكل التالي. ويوجد هذا البرنامج تحت الاسم Colors على القرص المصاحب.



شكل (١٢-١٢) تنفيذ البرنامج Colors

والآتي بعد هو الكود اللازم إضافته للدالة **OnDraw()** لإنتاج هذا الرسم.

```
void CColorsView::OnDraw(CDC* pDC)
{
    CColorsDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    CPen Pen1, Pen2, Pen3;
    CPen *PtrOldPen;
    // Solid blue pen, 3 pixels wide:
    Pen1.CreatePen(PS_SOLID, 3, RGB(0,0,255));
    // Select pen:
    PtrOldPen = pDC->SelectObject(&Pen1);
    //Set x and y:
    int Shift1=300, Shift2=250, Shift3=200;
    int Start =50;
    int x, y=Start;
    // 1st Line - Blue:
    x=Shift1+Start;
    y=Start;
    pDC->MoveTo(x,y);
    x=x-Shift1;
    y=y+Shift1;
    pDC->LineTo(x,y);
}
```

```

pDC->TextOut(x, y, "Line #1");

// Remove pen from the DC:
pDC->SelectObject(PtrOldPen);
//2nd Line - Red:
Pen2.CreatePen(PS_SOLID, 3, RGB(255,0,0));
PtrOldPen = pDC->SelectObject(&Pen2);
x=Shift2+Start;
y=Start;
pDC->MoveTo(x,y);
x=x-Shift2;
y=y+Shift2;
pDC->LineTo(x,y);
pDC->TextOut(x, y, "Line #2");
// Remove:
pDC->SelectObject(PtrOldPen);
// 3rd Line - Green:
Pen3.CreatePen(PS_SOLID, 3, RGB(0,255,0));
PtrOldPen = pDC->SelectObject(&Pen3);
x=Shift3+Start;
y=Start;
pDC->MoveTo(x,y);
x=x-Shift3;
y=y+Shift3;
pDC->LineTo(x,y);
pDC->TextOut(x, y, "Line #3");
// Remove:
pDC->SelectObject(PtrOldPen);
}

```

١٢-٧) جولة أخيرة مع الإحداثيات

تتأثر نظم الإحداثيات بما يسمى بالطور الحالى للخريطة (Mapping Mode). وفيما مررنا به من تطبيقات فى مجال الرسم حتى الآن كنا نستخدم الطور سابق التعريف (المسمى MM_TEXT). وفى هذا الطور تقاس كل من الإحداثيات الطبيعية والمنطقية بوحدة البكسل (Pixel) كما أن نقطة الأصل تقع فى الركن الأيسر العلوى دائما. ولكنك لو غيرت الطور المستخدم فسوف تختلف الإحداثيات

المنطقية عن الإحداثيات الطبيعية ، سواء في وحدة القياس أو في نظام المحاور. ولكي تخصص طورا معيناً للخريطة ، استخدم الدالة `CDC::SetMapMode()` التي تأخذ الصيغة التالية:

```
virtual int SetMapMode(int nMapMode);
```

حيث يمثل البارامتر `nMapMode` دليل طور الخريطة الذي يأخذ أحد قيم الثوابت الموضحة بالجدول التالي:

اتجاه المحاور	وحدة القياس	قيمة البارامتر
يمكن تعريفه	يمكن تعريفها	MM_ANISOTROPIC
تزيد قيمة γ إلى أعلى	0.001 بوصة	MM_HIENGLISH
تزيد قيمة γ إلى أعلى	0.01 ملليمتر	MM_HIMETRIC
يمكن تعريفه	يمكن تعريفها	MM_ISOTROPIC
تزيد قيمة γ إلى أعلى	0.01 بوصة	MM_LOENGLISH
تزيد قيمة γ إلى أعلى	0.1 ملليمتر	MM_LOMETRIC
تزيد قيمة γ إلى أسفل	بكسلة (وحدة إحداثيات الجهاز)	MM_TEXT سابق التعريف
تزيد قيمة γ إلى أعلى	1/400 من البوصة أو 1/20 من النقطة	MM_TWIPS

جدول (١٢-٧) أطوار الخريطة ونظم الإحداثيات

ملاحظات:

- يتميز الطور `MM_ANISOTROPIC` بإمكانية تعريف وحدة قياس مختلفة لكل محور.

- لتعريف وحدات القياس لكل من الطورين MM_ANISOTROPIC و MM_ISOTROPIC ، تستخدم الدوال (نكتفى بعرض أسماء الدوال ، وعليك باستكمال المسيرة في استخدامها إذا شئت):

CDC::SetWindowExt()

CDC::SetViewportExt()

- تتميز الأطوار الأخرى بخلاف الطور سابق التعريف بأنه يمكن رسم الصور بحجمها الحقيقي وبصرف النظر عن خصائص الجهاز.

(١٢-٨) خطوات استكمال برنامج الرسم

بعد أن عرفنا الطرق المختلفة لرسم وتلوين وتظليل الأشكال فإن المهمة القادمة قد اتضحت معالمها. ولنبدأ بالملاح التي يمكن إضافتها إلى البرنامج:

١. إضافة مجموعة من أوامر القائمة ، وما يكافئها من الأزرار ، لرسم الأشكال المختلفة (الدوائر والخطوط والقطاعات إلى آخره).

٢. يمكنك أيضا إضافة مجموعة من أوامر القائمة والأزرار للتلوين والتظليل حسب الطلب.

٣. في إمكانك أيضا أن تضيف بعض أوامر وأزرار لتحديد سمك القلم المستخدم في الرسم على غرار ما هو موجود في البرنامج Paint أو Paintbrush.

والكود المطلوب لتحقيق ذلك قد يستغرق بعض الصفحات ، ولكن خطة العمل يمكن تلخيصها في الخطوات التالية:

١. إنشاء الأزرار وأوامر القائمة باستخدام ساحر الفصائل كما عرفنا في الأبواب السابقة ، ثم ربط الأوامر والأزرار بالمقايض المناسبة.

٢. حتى يمكن رسم أى شكل من الأشكال فلا بد أن تكون قد أعددت لذلك بإنشاء فصيلة خاصة بهذا الشكل ، على غرار ما فعلنا مع الخط المستقيم.

٣. من الأفضل أيضا تفصيل البرنامج بطريقة موجهة إلى الأهداف بمعنى أن ننشئ فصيلة أم ترثها كل الفصائل الأخرى ، ولتكن فصيلة الشكل العام (class CMyGraph) التي ترث من فصيلة المؤسسة **CObject**. من هذه الفصيلة تأتي فصيلة الخطوط (CMyLine) ، والدائرة (CMyCircle) ، والبيضاوى (CMyEllipse) ، الخ. وتحتوى كل فصيلة من هذه الفصائل على الأعضاء المميزة للفصيلة من البيانات والدوال. والآتى بعد مثال للفصيلة الأم:

```
class CMyGraph: public CObject
{
    protected:
        int m_x1, m_x2, m_y1, m_y2;
        COLORREF m_color;
        CMyGraph() {}
        DECLARE_SERIAL(CMyGraph)

    public:
        virtual void DrawIt(CDC *PDC) {}
        virtual void Serialize(CArchive& ar)
```

CRect getRect();

ونلاحظ الآتى فى الفصيلة الأم:

- تحتوى الفصيلة على البيانات الأعضاء المشتركة فى جميع الأشكال. مثل الإحداثيات الأربعة ومعلومة اللون:

```
int m_x1, m_x2, m_y1, m_y2;  
COLORREF m_color;
```

وتضيف كل فصيلة من عندها ما يزيد عن هذه البيانات من البيانات الخاصة بها (مثل سمك الخط).

- تحتوى الفصيلة أيضا على الدالة `getRect()` التى تحدد المستطيل المحتوى على الرسم. وهى تستخدم مع جميع الأشكال.

- تحتوى الفصيلة على دالة الحفظ `Serialize()`. وهى تصلح لاختزان بيانات الفصائل التى لا تحتوى على أعضاء خاصة بها (مثل السمك). فإذا كانت تحتوى على أعضاء خاصة ، فمن اللازم ركوب الدالة `Serialize()` من داخل الفصيلة المعينة.

- تحتوى الفصيلة على إعلان الدالة `DrawIt()` المسئولة عن الرسم. أما تعريف الدالة فيأتى بداخل الفصائل الأبناء حيث أنها تختلف من فصيلة إلى أخرى.

والآتى بعد مثال لأحد الفصائل الأبناء (فصيلة الدائرة) التى ترث من الفصيلة الأم:

```
class CMyCircle: public CMyGraph  
{  
protected:  
    int m_thickness;  
    CMyCircle() {}  
    DECLARE_SERIAL (CMyCircle)  
public:
```

```

CMyCircle (int x1, int y1, int x2, int y2, COLORREF
            color, int thickness);
virtual void DrawIt(CDC *PDC);
virtual void Serialize(CArchive& ar);
};

```

Bitmaps

(١٢-٩) استخدام خرائط البكسلات

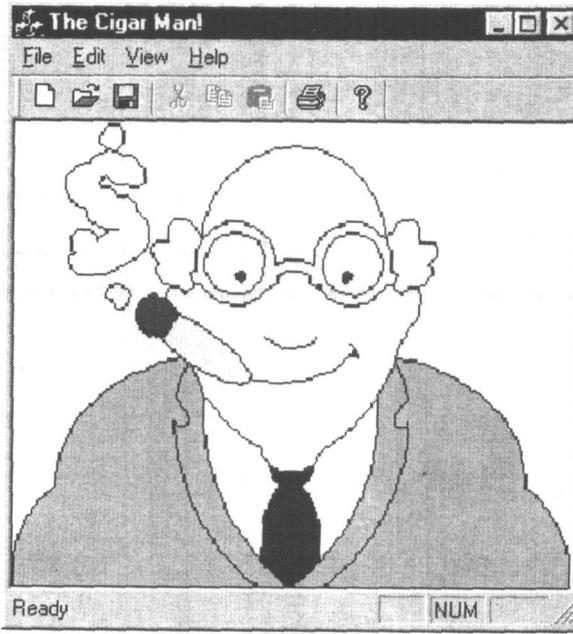
في بعض الأحيان قد يتطلب الأمر استخدام صورة أو رسم جاهز ، والتي نستخدم عليها باسم خرائط البكسلات (Bitmaps) ، وعرضها باستخدام البرنامج. ويحتوي ملف خريطة البكسلات على مجموعة من البتات (Bits) تتأثر بكسلات الصورة أو الرسم بما تحتويه من معلومات الإضاءة والألوان. وفي هذه الفقرة سوف نوضح كيفية إعداد الرسم ، تحميله في الذاكرة ، وعرضه على الشاشة من خلال المثال الموجود على القرص تحت الاسم MyBitmap.

ملاحظة:

إن موضوع معالجة خرائط البكسلات يحتوي على موضوعات فرعية كثيرة لن نتعرض لها في هذا الكتاب ، وسوف نكتفي بالإشارة إليها بحيث تستطيع استكمال الرحلة معها إذا شئت.

ملاحظة:

عند تشغيل البرنامج MyBitmap سوف يرسم الشكل التالي على شاشتك.

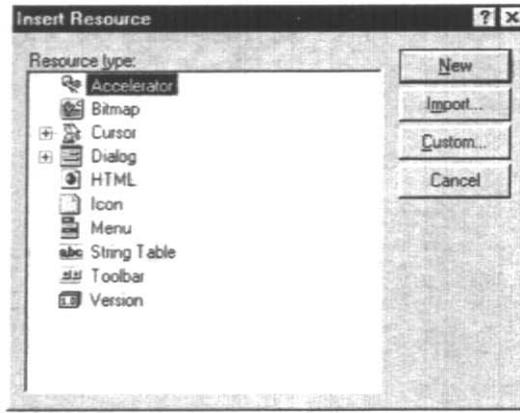


شكل (١٢-١٣) نتيجة تنفيذ البرنامج MyBitmap

كما نرى أعلى النافذة المحتوية على الشكل اسم الصورة "The Cigar Man".

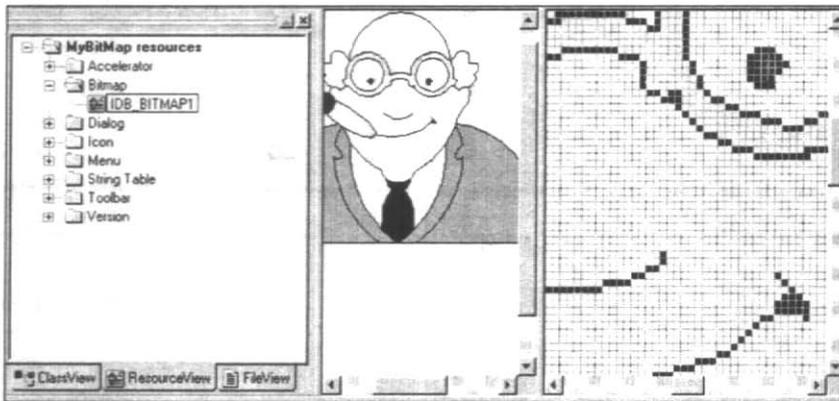
خطوات إنشاء البرنامج

١. أنشئ مشروعاً جديداً بالاسم MyBitmap.
٢. ابدأ بإضافة مكان للصورة في المشروع الجديد ، وذلك باستخدام أمر القائمة Insert-Resource ، ثم اختيار المورد (Bitmap) من نافذة الموارد الموضحة بالشكل التالي.



شكل (١٢-١٤) نافذة الموارد

٣. في مشهد الموارد ، استخدم محرر الرسومات الموضح بالشكل التالي ، لإضافة الصورة أو الرسم بأحد طريقتين:
- إما برسم الصورة المطلوبة على شبكة المربعات بالاستعانة بإمكانات المحرر (وهي عملية مماثلة لرسم الأيقونة).
 - أو بنسخ صورة جاهزة ولصقها بالأمر Paste على شبكة المربعات.



شكل (١٢-١٥) شاشة محرر الصور والرسومات

٤. أجز التعديلات الموضحة بعد على كود البرنامج.

أولاً: إعلان هدف خريطة البكسلات

أضف الإعلان التالي ، الموضح بالبنط الثقيل ، إلى ملف العناوين
MyBitMapView.h

```
class CMyBitMapView : public CView
{
protected: // create from serialization only
    CMyBitMapView();
    DECLARE_DYNCREATE(CMyBitMapView)
// new code: أضف السطور التالية:
    CBitmap m_BMP;
    int m_BMPHeight;
    int m_BMPWidth;
}
```

ويتضمن الإعلان العضو m_BMP وهو عبارة عن هدف من
الفصيلة **CBitmap** ، أما الأعضاء m_BMPHeight و m_BMPWidth
فهى متغيرات صحيحة تمثل طول وعرض الصورة.

ثانياً: تحميل معلومات الصورة فى الذاكرة

أضف الكود التالي ، الموضح بالبنط الثقيل ، إلى دالة بناء المشهد
CMyBitMapView()

```
CMyBitMapView::CMyBitMapView()
{
    // TODO: add construction code here
// new code: أضف السطور التالية:
}
```

```

BITMAP BMP;
m_BMP.LoadBitmap(IDB_BITMAP1);
m_BMP.GetObject(sizeof(BMP), &BMP);
m_BMPHeight = BMP.bmHeight;
m_BMPWidth = BMP.bmWidth;
}

```

ويتضمن هذا الكود الخطوات المنطقية التالية:

- إعلان المتغير BMP من النمط **BITMAP**. هذا النمط عبارة عن منشأ يخزن معلومات خريطة البكسلات وفقا للإعلان التالي:

```

typedef struct tagBITMAP {
int bmType;      نوع الخريطة
int bmWidth;     عرض الخريطة
int bmHeight;    ارتفاع الخريطة
int bmWidthBytes; عدد البايت بكل سطر (عدد زوجي)
BYTE bmPlanes;   عدد مستويات الصورة
BYTE bmBitsPixel; عدد البتات المخصصة لتحديد لون البكسل
LPVOID bmBits;   متغير يشير إلى مكان مصفوفة الخريطة
}

```

- يلي ذلك تحميل الصورة من بيئة الموارد باستخدام الدالة **CBitmap::LoadBitmap()**. أما بارامتر هذه الدالة فيشير إلى الصورة التي أعددناها من قبل باستخدام محرر الرسم. ورقم التعارف IDB_BITMAP1 فهو رقم يمنحه البرنامج من عنده للصورة الأولى. وإذا كانت هناك صورة تالية فيمنحها الرقم IDB_BITMAP2 وهكذا. ولك أن تغير هذا الاسم من بيئة الموارد إذا شئت ، مع ضرورة تغييره في البرنامج أيضا.

• تخزين معلومات الصورة في متغير الوعاء BMP باستخدام الدالة `CGDIBitmap::GetObject()`. ونلاحظ أن الوعاء يستخدم الدالة `sizeof()` التي تحدد عدد البايت في الصورة ، علاوة على مؤشر إلى الوعاء BMP.

• يتم أيضا تخزين أبعاد الصورة المختزنة في المنشأ BMP (أى `bmHeight` و `bmWidth`) في المتغيرين الأعضاء `M_BMPHeight` و `M_BMPWidth`. ولتلاحظ أنك عندما تكتب علامة النقطة (.) تالية لاسم المتغير BMP فإن الأستوديو يعرض أمامك معلومات المنشأ BITMAP ، ولذلك فإنك لا تحتاج إلى تذكر أسماء أعضاء المنشأ.

ثالثا: عرض الصورة

تستخدم الدالة `OnDraw()` لوضع الكود اللازم لعرض الصورة كما هو معتاد. أضف الكود الموضح بالبنط الثقيل. وقد أوضحنا وظيفة كل دالة بجوارها بين سطور الكود نفسه.

```
void CMyBitMapView::OnDraw(CDC* pDC)
{
    CMyBitMapDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    // new code:
    CDC MyDC;
    RECT MyClientRect;
    MyDC.CreateCompatibleDC (NULL);
    // أضف هذه السطور:
    // إعلان هدف منطقة العرض بالذاكرة
    // أبعاد المستطيل المحتوى على الصورة
    // إنشاء هدف منطقة عرض وتخزين الخريطة به:
```

MyDC.SelectObject (&m_BMP);

الحصول على معلومة أبعاد النافذة:

GetClientRect (&MyClientRect);

عرض الصورة بحيث تملأ مساحة النافذة:

pDC->StretchBlt

(0, 0, إحداثى الركن الأيسر العلوى للمستطيل المستقبل للصورة

MyClientRect.right, اتساع المستطيل المستقبل للصورة

MyClientRect.bottom, ارتفاع المستطيل المستقبل للصورة

&MyDC, عنوان هدف منطقة العرض

0, 0, إحداثى الركن الأيسر العلوى للمستطيل الأصلي للصورة

m_BMPWidth, اتساع المستطيل الأصلي للصورة

m_BMPHeight, ارتفاع المستطيل المستقبل للصورة

SRCCOPY); ثابت الراستر

}

في شريحة الكود السابقة استخدمنا الدوال الآتية:

• **CDC::CreateCompatibleDC (CDC* pDC);**

وهي تستخدم في خلق منطقة عرض بالذاكرة ، وتستخدم المؤشر pDC. فإذا كانت قيمة المؤشر NULL تكون منطقة العرض هي الشاشة.

• **CDC::SelectObject (CBitmap* pBitmap);**

وهي تستخدم لاختيار أو تغيير خريطة البكسلات المشار إليها بالمؤشر bitmap ، وقد استخدمناها من قبل مع أدوات الرسم (القلم والفرشاة). وهي دالة محملة تحميلاً زائداً لها أكثر من صيغة.

• **CWnd::GetClientRect (LPRECT lpRect);**

وتستخدم هذه الدالة في الحصول على أبعاد النافذة باستخدام المؤشر

•lpRect

- **CDC::StretchBlt (int x, int y, int nWidth, int nHeight, CDC* pSrcDC, int xSrc, int ySrc, int nSrcWidth, int nSrcHeight, DWORD dwRop);**

وهي تستخدم في نسخ الصورة من مصدرها إلى المستطيل المخصص لاستقبالها بالنافذة. وتقوم الدالة بضغط أو فرد الصورة حتى تتناسب المستطيل المستقبل لها. وقد تم عرض أسماء ووظائف البارامترات المستخدمة في شريحة الكود السابقة. ومن الجدير بالذكر أن البارامتر الأخير dwRop (ثابت الراستر: Raster Operation Code) قد يأخذ قيما مختلفة تناظر خصائص لونية مختلفة للصورة. أما الثابت الذي استخدمناه هنا (SRCOPY) فهو يؤدي إلى نسخ الصورة الأصلية طبق الأصل. أما بقية الثوابت فنكتفى بذكر أسمائها ، وعلينا أن نتعرف على وظائفها بالطريقة المثلى ، وهي التجربة:

BLACKNESS
DSTINVERT
MERGECOPY
MERGEPAINT
NOTSRCCOPY
NOTSRCERASE
PATCOPY
PATINVERT
PATPAINT
SRCAND .
SRCCOPY
SRCERASE
SRCINVERT
SRCPAINT
WHITENESS

جدول (١٢-٨) قيم ثابت الراستر المستخدم مع الدالة StretchBlt()

رابعا: عرض اسم الرسم بالنافذة

أما الخطوة الأخيرة فهي كتابة اسم الصورة أو الرسم فى نافذة التطبيق ، وهذا يتم بإضافة الكود التالى إلى الملف MyBitMap.cpp بداخل الدالة `.InitInstance()`.

```
BOOL CMyBitMapApp::InitInstance()
{
    AfxEnableControlContainer();
    ...
    ...
    m_pMainWnd->ShowWindow(SW_SHOW);
    m_pMainWnd->UpdateWindow();
    // new code:                أضف السطر التالى:
    m_pMainWnd->SetWindowText("The Cigar Man!");
    return TRUE;
}
```

وقد استخدمنا هنا الدالة `SetWindowText()` التى تأخذ الصورة العامة:

- `CWnd::SetWindowText(LPCTSTR lpszString);`
وبارامتر الدالة عبارة عن مؤشر طويل إلى ثابت حرفى وقد يستبدل بالحرفى نفسه كما هو موضح بشريحة الكود.

ملاحظة:

- علاوة على عرض الصورة أو الرسم من خلال برنامجك فقد تحتاج إلى هذه النوعية من الرسومات فى المناسبات الآتية:
- لوضع رسم أو صورة على زر من أزرار التحكم (باستخدام الفصيلة `CBitmapButton`).
 - لرسم علامة صح (?) مختلفة عن العلامة التقليدية بجوار أحد اختيارات القائمة (باستخدام الدالة `CMenu::SetMenuItemBitmap()`)

• لملأ مساحة ما بشبكة من الصور أو الرسومات (باستخدام الدالة

(CBrush::CreatePatternBrush())

وسوف نترك لك استكشاف هذه التطبيقات كلون من التدريب.

ملاحظة: _____



الموجز

١. في هذا الباب تجولنا مع دوال وأدوات الرسم المختلفة

وعرفة المهارات الآتية:

- إعداد منطقة عرض (Display Context).
- إعداد أدوات الرسم وهي القلم (Pen) وفرشاة ملء المساحات (Brush).
- اختيار خصائص منطقة العرض (Attributes).
- استخدام دوال الرسم.

- تفصيل أدوات رسم حسب الطلب.
- ٢. عرفنا الخطوات الأساسية لتطوير برنامج الرسم إلى الطراز النهائي باستخدام أدوات ودوال الرسم.
- ٣. عرفنا أيضا كيفية إعداد وعرض الصور والرسومات (خرائط البكسلات) من خلال البرامج.
- ٤. استخدمنا مجموعة كبيرة من الدوال والفصائل والثوابت ، وهي:

الفصائل:

CDC
CPen
CBrush
CClientDC
CPaintDC
CBitmap
CBitmapButton

الثوابت والماكرو:

- نمط الألوان **COLORREF** (يستخدم من خلال الماكرو **(RGB)**)
- منشأ خريطة البكسلات **BITMAP**
- مجموعة من أنماط البيانات بالمؤسسة **MFC** – موضحة بالملحق (ب).
- ثوابت القلم والفرشاة – الجدول (١-١٢)
- نسب الألوان الأساسية – جدول (٣-١٢)
- نسب الألوان المقابلة للألوان الأساسية – جدول (٤-١٢)
- طرازات القلم – جدول (٥-١٢)
- طرازات فرشاة التهشير – جدول (٦-١٢)
- أطوار الخريطة ونظم الإحداثيات – جدول (٧-١٢)

• القيم المختلفة لثابت الراستر – موضحة بالجدول (١٢-٨)

الدوال: (دوال الرسم مشروحة بالجدول (١٢-٢))

CWnd::OnPaint()
CView::OnDraw()
CDC::SetPixelV()
CDC::MoveTo()
CDC::LineTo()
CDC::TextOut()
CDC::Polygon()
CDC::PolyBezier()
CDC::PolyLine()
CDC::PolyPolygon()
CDC::Arc()
CDC::Chord()
CDC::Pie()
CDC::Rectangle()
CDC::RoundRect()
CDC::SelectStockObject()
CDC::SelectObject(&Pen)
CPen::CreatePen()
CBrush::CreateSolidBrush();
CBrush::CreateHatchBrush();
CBrush::CreatePatternBrush()
CDC::SetMapMode()
CDC::SetWindowExt()
CDC::SetViewportExt()
CDC::CreateCompatibleDC ()
CDC::SelectObject ()
CBitmap::LoadBitmap()
CGDIObject::GetObject()
CWnd::GetClientRect ()
CDC::StretchBlt ()
CMenu::SetMenuItemBitmap()
CWnd::SetWindowText()