الفصل الثامن عشر
استخدام أقلام الرسم
وفرش الألوان

تستخدم أقلام الرسم لرسم الخطوط فضلاً عن رسم حدود الأشكال الملونة، بينما تستخدم فرش الألوان لملء الأشكال بالألوان المختلفة. وفى هذا الفصل سنتعرف على كيفية إنشاء واستخدام كلٍ من أقلام الرسم وفرش الألوان.

بانتهاء هذا الفصل ستتعرف على:

- ***استخدام أقلام الرسم لرسم الخطوط والأشكال بأنماط متعددة.***
- ***استخدام فرش الألوان لرسم الأشكال الملونة.***
- ***استخدام دوال الرسم لرسم الأشكال المركبة.***

obeikandl.com

العمل مع أقلام الرسم

تعد أقلام الرسم أحد العناصر الأساسية للواجهة الرسومية لأى عنصر GDI، كما تعتبر من العناصر الأساسية المشتركة فى تكوين نظام التشغيل. وقبل أن تقوم برسم أى شئ، يجب أن تقوم أولاً بإنشاء واختيار أقلام الرسم التى تتناسب مع المهمة التى بين يديك.

استخدام التصنيف CPen

تحتوى مكتبة MFC على التصنيف CPen المستخدم لتعريف وإنشاء قلم رسم جديد. لإنشاء قلم رسم جديد، يجب أن تقوم أولاً بتعريفه واستهلاله ببعض القيم الابتدائية. فمثلاً يمكنك استخدام الكود التالى لإنشاء قلم مصمت أحمر اللون:

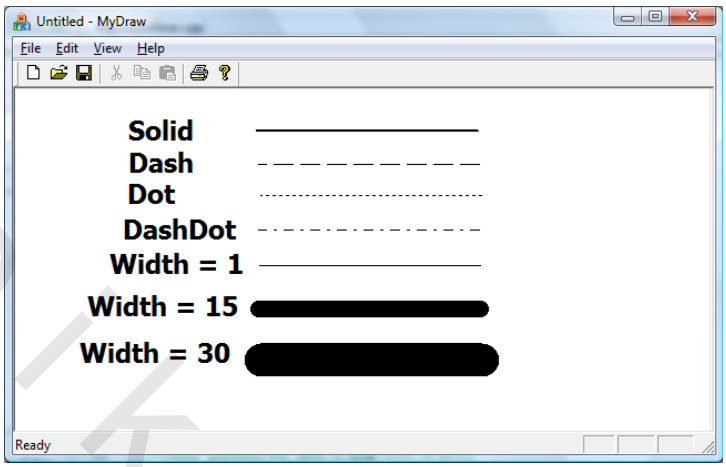
```
CPen PenRed(PS_SOLID,3,RGB(255,0,0));
```

حيث يحتوى المعامل الأول على نوع القلم وهو مصمت فى هذه الحالة، ويمكنك تغييره باختيار قيمة أخرى من القيم الموضحة بالجدول ١٨-١ التالى.

جدول ١٨-١ أنواع قلم الرسم

نوع القلم	*شكل الخطوط المرسومة*
PS_SOLID	خطوط مصمتة بسيطة
PS_DASH	خطوط متقطعة
PS_DOT	خطوط منقطة
PS_DASHDOT	خطوط متقطعة ومنقطة

تعمل الأنواع المنقطة PS_DOT والمتقطعة PS_DASH مع الأقلام التى سمكها يساوى ١، فإذا زاد السمك عن ١، ينتج تلقائياً خط مصمت (انظر شكل ١٨-١).



شكل ١٨-١ أنواع الخطوط وأحجامها

يحتوى المعامل الثانى على سمك القلم. كما يحتوى المعامل الثالث على لون الخط المرسوم بواسطة القلم عن طريق استخدام المختزل **RGB** وهو من النوع **COLORREF** ويحتوى على ثلاثة معاملات لدرجة سطوع الألوان الأحمر والأخضر والأزرق على الترتيب، وقيمة كلٍ من هذه المعاملات تتراوح بين صفر و ٢٥٥. حيث تعنى القيمة صفر عدم وجود اللون بينما تعنى القيمة ٢٥٥ وجود اللون كاملاً، وبذلك يمكنك تعيين ما يقرب من ١٦.٧ مليون لون باستخدام مزيج من قيم هذه الألوان (طالما كان كارت الشاشة المثبت بجهازك يدعم هذه الألوان). فمثلاً إذا أردت إنشاء قلم أخضر داكن مصمت ذو سمك نقطة واحدة، يمكنك استخدام الكود التالى:

```
CPen penSolidDrkGreen(PS_SOLID,1,RGB(0,128,0));
```

وإذا أردت إنشاء قلم أصفر مصمت ذو سمك ٣٠ نقطة، يمكنك استخدام الكود التالى:

```
CPen penSolidYellow(PS_SOLID,30,RGB(255,255,0));
```

وإذا أردت إنشاء قلم أحمر منقط بسمك نقطة واحدة، يمكنك استخدام الكود التالى:

```
CPen penDotRed(PS_DOT,1,RGB(255,0,0));
```

أما إذا أردت إنشاء قلم أزرق متقطع بسمك نقطة واحدة، فقم باستخدام الكود التالى:

```
CPen penDotBlue(PS_DASH,1,RGB(0,0,255));
```

استخدام أقلام الرسم الموجودة

لست فى حاجة دائماً لتعيين تفاصيل قلم الرسم الذى تود إنشاءه، وذلك لأن نظام التشغيل يحتوى على مجموعة من أقلام الرسم المخزَّنة والمعرَّفة مسبقاً. يمكنك استخدام هذه الأقلام باستدعاء الدالة **CreateStockObject()** بتمرير أحد القيم الموضحة بجـــدول ١٨-٢ التالى.

جدول ١٨-٢ الأقلام المعرفة داخل نظام التشغيل

اسم القلم	الخطوط المرسومة
BLACK_PEN	رسم خطوط سوداء
WHITE_PEN	رسم خطوط بيضاء
NULL_PEN	رسم خطوط بلون الخلفية

يستخدم المعامل **NULL_PEN** للرسم بنفس لون الخلفية لذلك فهو لا يظهـــر على الشاشة ويستخدم لرسم الأشكال الملونة التى لا تحتوى على إطار خارجى.

اختيار أقلام الرسم

قبل أن تقوم باستخدام أقلام الرسم التى قمت بإنشائها، يجب أن تقوم أولاً باختيار القلـــم الذى تريد استخدامه. فمن الممكن أن تحتوى بيئة العنصر على العديد من الأقلام، لذا يجب أن تحدد أى هذه الأقلام ترغب فى استخدامه باستدعاء الدالة **SelectObject()**. للتعرف على طريقة اختيار قلم جديد واسترجاع القلم القديم، قم بإنشاء تطبيق أحـــادى الوثيقـــة باسم **MyDraw** ثم قم بتعديل كود الدالة **OnDraw()** كما يلى:

```
1.    void CMyDrawView::OnDraw(CDC* pDC)
2.    {
3.      CMyDrawDoc* pDoc = GetDocument();
4.      ASSERT_VALID(pDoc);
5.      // TODO: add draw code for native data here
6.      CPen penSolid(PS_SOLID,5,RGB(0,200,0));
7.      CPen *pOldPen = NULL;
8.      pOldPen = pDC->SelectObject(&penSolid);
```

```
9.      // Draw with the green pen
10.     pDC->SelectObject(pOldPen);
11.  }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٨ تم استدعاء الدالة **SelectObject()** بتمرير مؤشر للقلم المراد اختياره، وحينئذٍ تقوم الدالة باختيار القلم الجديد وتخزين عنوان القلم القديم داخل المؤشر **pOldPen**.
- فى السطر رقم ٩ يتم استخدام قلم الرسم المحدد لأداء أىٍ من الرسومات المختلفة.
- فى السطر رقم ١٠ يتم استرجاع عنوان القلم القديم باستدعاء الدالة **SelectObject()** مرةً أخرى مع تمرير المؤشر **pOldPen**.

نريد أحياناً استخدام قلمين، وفى هذه الحالة يجب التبديل بينهما باستخدام الدالة **SelectObject()** كما فى الكود التالى:

```
1.    void CMyDrawView::OnDraw(CDC* pDC)
2.    {
3.      CMyDrawDoc* pDoc = GetDocument();
4.      ASSERT_VALID(pDoc);
5.      // TODO: add draw code for native data here
6.      CPen penGreen(PS_SOLID,5,RGB(0,255,0));
7.      CPen penBlue(PS_SOLID,3,RGB(0,0,255));
8.      CPen *pOldPen = NULL;
9.      pOldPen = pDC->SelectObject(&penGreen);
10.     //Draw with the green pen
11.     pDC->SelectObject(&penBlue);
12.     //Draw with the Blue pen
13.     pDC->SelectObject(pOldPen);
14.   }
```

حذف أقلام الرسم

عندما تنتهى من استخدام أىٍ من أقلام الرسم داخل بيئة العنصر **DC**، يجب أن تقوم بحذف هذا القلم لتحرير الموارد المرتبطة به، وهذا ما يقوم به التصنيف **CPen** نيابةً عنك. أما إذا أردت استخدام نفس القلم بإعدادات أخرى، فيجب أن تقوم بحذف القلم يدوياً باستخدام

الدالة **()DeleteObject** ثم إعادة استخدام القلم بالإعدادات الجديدة باستدعاء الدالة **()CreatPen**. للتعرف على طريقة حذف الأقلام وإعادة استخدامها، قم بتعديل كود الدالة **()OnDraw** كما يلى:

```
1.    void CMyDrawView::OnDraw(CDC* pDC)
2.    {
3.      CMyDrawDoc* pDoc = GetDocument();
4.      ASSERT_VALID(pDoc);
5.      // TODO: add draw code for native data here
6.      CPen penMainDraw(PS_DASH,1,RGB(128,255,255));
7.      CPen *pOldPen = NULL;
8.      pOldPen = pDC->SelectObject(&penMainDraw);
9.      //Draw with the dashed pen
10.     pDC->SelectObject(pOldPen);
11.     penMainDraw.DeleteObject();
12.     penMainDraw.CreatePen(PS_SOLID,5,RGB(200,20,5));
13.     pOldPen = pDC->SelectObject(&penMainDraw);
14.     //Draw with the solid pen
15.     pDC->SelectObject(pOldPen);
16.   }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٦ قمنا بتعريف قلم متقطع باسم **penMainDraw**.
- فى السطور من ٨ إلى ١٠ قمنا باختيار القلم واستخدامه.
- فى السطر رقم ١١ تم حذف القلم باستدعاء الدالة **()DeleteObject** وذلك لأننا نريد استخدامه مرةً أخرى بإعدادات مختلفة.
- فى السطر رقم ١٢ قمنا باستدعاء الدالة **()CreatePen** لإعادة استخدام القلم **penMainDraw** ولكن بإعدادات مختلفة.
- فى السطور من ١٣ إلى ١٥ تم استخدام القلم ثم استرجاع القلم القديم.
- لاحظ أنه بمجرد انتهاء الدالة يتم حذف القلم **penMainDraw** تلقائياً.

عند استدعاء الدالة **DeleteObject()** أثناء اختيار القلم داخل بيئة العنصر، يتم تدمير عنصر **GDI** وبالتالى ستحصل على نتائج غير متوقعة بمجرد استخدام هذا القلم. لذا لا تنسى استبدال القلم بآخر قبل أن تقوم بعملية الحذف.

رسم الخطوط والأشكال باستخدام الأقلام

يتم دائماً استخدام الأقلام للرسم داخل بيئة العنصر، حيث يمكنك استخدام دوال الرسم المختلفة داخل أى بيئة عنصر صحيحة. وفيما يلى سنتعرف على بعض أساسيات الرسم باستخدام دوال أقلام الرسم.

تغيير مكان المؤشر

تحتوى بيئة العنصر دائماً على إحداثيات المكان الحالى لمؤشر القلم وهو المكان الذى سنبدأ منه الرسم بمجرد استدعاء أىٍ من دوال الرسم المختلفة. يمكنك تغيير هذا المكان باستدعاء الدالة **MoveTo()** التى تحتوى على معاملين يمثلان الإحداثى الأفقى والرأسى للمكان الجديد. إذا قمت بفتح الدالة **OnDraw()** داخل نافذة المحرر دون إضافة أىٍ من أسطر الكود ، يظهر كود الدالة كما يلى:

```
void CMyDrawView::OnDraw(CDC* pDC)
{
	CMyDrawDoc* pDoc = GetDocument();
	ASSERT_VALID(pDoc);
	// TODO: add draw code for native data here
}
```

حيث قام معالج التطبيقات بتمرير مؤشر للدالة يشير إلى بيئة العنصر التى تحتوى على تفاصيل نافذة العرض الأساسية للتطبيق أحادى الوثيقة. ثم بعد ذلك يشير المؤشر **pDoc** إلى الوثيقة.

إذا قمت الآن ببناء التطبيق وتنفيذه، ستحصل على نافذة عرض صماء. فإذا أردت البدء فى إضافة الرسوم للوثيقة، قم بتغيير مكان المؤشر باستدعاء الدالة **MoveTo()** كما يلى:

```
void CMyDrawView::OnDraw(CDC* pDC)
{
	CMyDrawDoc* pDoc = GetDocument();
```


obeikandl.com

```
        ASSERT_VALID(pDoc);
        // TODO: add draw code for native data here
        pDC->MoveTo(50,100);
}
```

حيث قمنا باستدعاء الدالة **MoveTo()** لتغيير مكان المؤشر إلى النقطة (٥٠ ، ١٠٠).

رسم الخطوط

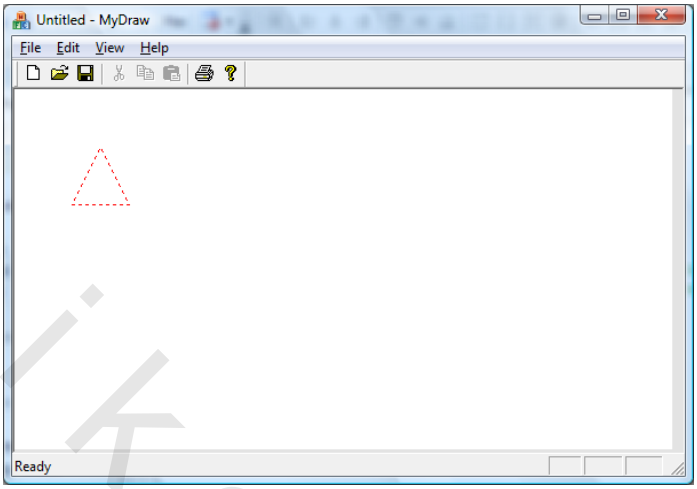
يمكنك استخدام الدالة **LineTo()** لرسم خط من مكان المؤشر الحالى إلى المكان المحـــدد داخل الدالة، حيث تحتوى هذه الدالة على نفس معاملات الدالـــة **MoveTo()**. لرســـم مثلث من الخطوط المنقطة، قم بتعديل كود الدالة **OnDraw()** كما يلى:

```
1.    void CMyDrawView::OnDraw(CDC* pDC)
2.    {
3.      CMyDrawDoc* pDoc = GetDocument();
4.      ASSERT_VALID(pDoc);
5.      //TODO: add draw code for native data here
6.      CPen penRed(PS_DOT,1,RGB(255,0,0));
7.      CPen *pOldPen = NULL;
8.      pOldPen = pDC->SelectObject(&penRed);
9.      pDC->MoveTo(50,100);
10.     pDC->LineTo(100,100);
11.     pDC->LineTo(75,50);
12.     pDC->LineTo(50,100);
13.     pDC->SelectObject(pOldPen);
14.   }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ١٠ تم استدعاء الدالة **LineTo()** لرسم خط أحمر منقط من نقطة البدء (٥٠ ، ١٠٠) إلى النقطة (١٠٠ ، ١٠٠).
- فى السطور ١١ و١٢ تم رسم بقية أضلاع المثلث باستخدام الدالة **LineTo()** كما سبق.

والآن قم ببناء التطبيق وتنفيذه، تحصل على مثلث منقط (انظر شكل ١٨-٢).



شكل ١٨-٢ رسم مثلث منقط باستخدام الدوال MoveTo() و LineTo()

الرسم بإحداثيات النقطة

يمكنك استخدام التصنيف CPoint لتمرير متغيراته إلى دوال الرسم بدلاً من تمرير قيم الإحداثيات الأفقية والرأسية صراحةً. كما يمكنك استخدام الدالة GetCurrentPosition() للتعرف على المكان الحالي لمؤشر الرسم. يمكنك أيضاً استخدام التصنيف CRect للتعرف على إحداثيات نافذة العرض وخصائصها المختلفة. كي نتعرف سوياً عن قرب على طريقة أداء هذه المهام، قم بتعديل كود الدالة OnDraw() كما يلي:

```
1.    void CMyDrawView::OnDraw(CDC* pDC)
2.    {
3.      CMyDrawDoc* pDoc = GetDocument();
4.      ASSERT_VALID(pDoc);
5.      // TODO: add draw code for native data here
6.      CPen penRed(PS_DOT,1,RGB(255,0,0));
7.      CPen *pOldPen = NULL;
8.      pOldPen = pDC->SelectObject(&penRed);
9.      CRect rcClient;
10.     GetClientRect(&rcClient);
11.     for (int x=0;x<rcClient.Width();x+=5)
12.       for (int y=0;y<rcClient.Height();y+=5)
```

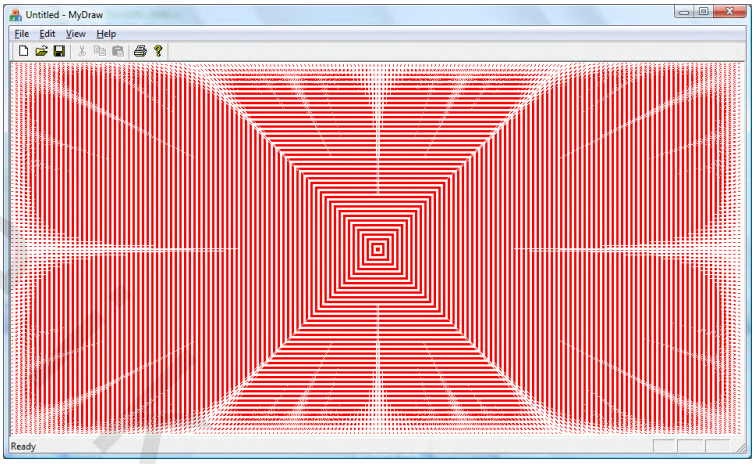
```
13.       {
14.           CPoint ptNew(x,y);
15.           pDC->MoveTo(rcClient.CenterPoint());
16.           pDC->LineTo(ptNew);
17.       }
18.     pDC->SelectObject(pOldPen);
19.   }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطور ٩ و ١٠ تم تعريف المؤشر **rcClient** الذى يحتوى على معلومات نافذة العرض باستخدام الدالة **GetClientRect()**.
- فى السطور ١١ و ١٢ تم استخدام دوارتين **for** لرسم مجموعة الخطوط باستخدام إحداثيات طول وعرض النافذة.
- فى السطر رقم ١٤ تم استخدام التصنيف **CPoint** لتعريف عنصر يحتوى على الإحداثى الأفقى والرأسى معاً.
- فى السطر رقم ١٥ تم استدعاء الدالة **MoveTo()** بتمرير الدالة **CenterPoint()** لنقل مؤشر الرسم إلى منتصف الوثيقة.
- فى السطر رقم ١٦ تم رسم خط إلى الإحداثى المخزَّن داخل العنصر **ptNew** والذى يتغير مع كل دورة.

قم ببناء التطبيق وتنفيذه. تظهر لك نافذة العرض كما فى شكل ١٨-٣.

obeikandl.com



شكل ١٨-٣ رسم الخطوط باستخدام إحداثيات النقطة

رسم الدائرة والقطع الناقص

تعتبر الدائرة قطع ناقص يتساوى فيه العرض مع الارتفاع، لذا يتم رسم كلٍ منهما باستخدام نفس الدالة **()Ellipse**. فى حالة رسم الدائرة، يتم تمرير عنصر ينتمى للتصنيف **CRect** إلى الدالة. أما فى حالة رسم قطع ناقص فيتم تمرير أربعة معاملات، الأول والثانى عبارة عن الإحداثى الأفقى والرأسى للركن الأيسر العلوى، أما الثالث والرابع فيمثلان الإحداثى الأفقى والرأسى للركن الأيمن السفلى من القطع الناقص. للتعرف على كيفية استخدام الدالة **()Ellipse** لرسم الدائرة، قم بتعديل كود الدالة **()OnDraw** كما يلى:

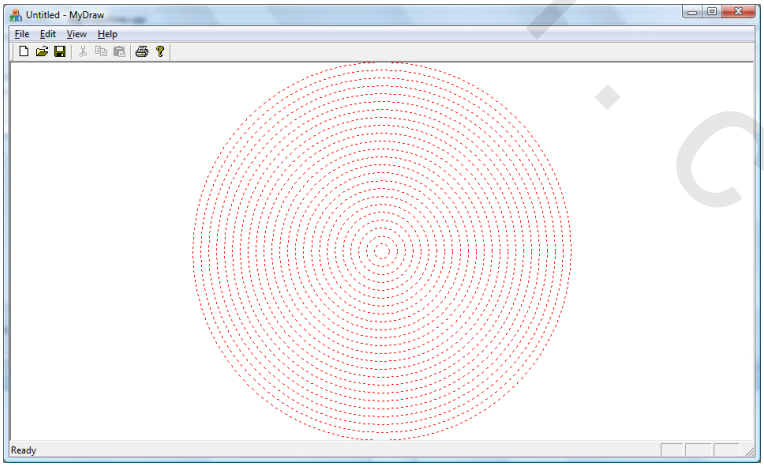
```
1.    void CMyDrawView::OnDraw(CDC* pDC)
2.    {
3.      CMyDrawDoc* pDoc = GetDocument();
4.      ASSERT_VALID(pDoc);
5.      // TODO: add draw code for native data here
6.      CPen penRed(PS_DOT,1,RGB(255,0,0));
7.      CPen *pOldPen = NULL;
8.      pOldPen = pDC->SelectObject(&penRed);
9.      pDC->SelectStockObject(NULL_BRUSH);
10.     CRect rcClient;
11.     GetClientRect(&rcClient);
12.     CRect rcEllipse(rcClient.CenterPoint() ,
13.     rcClient.CenterPoint());
```

```
14.     while(rcEllipse.Width() < rcClient.Width() &&
15.     rcEllipse.Height()  < rcClient.Height())
16.     {
17.       rcEllipse.InflateRect(10,10);
18.       pDC->Ellipse(rcEllipse);
19.     }
20.     pDC->SelectObject(pOldPen);
21.   }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٩ تم استدعاء الدالة **SelectStockObject()** مع تمرير المعامل **NULL_BRUSH** حتى لا يتم تلوين أيٍ من الدوائر وإنما يتم رسم حدودها فقط.
- فى السطر رقم ١٢ تم تعريف المستطيل المستخدم لرسم الدائرة وتعيين قيمه الابتدائية بمنتصف نافذة العرض.
- فى السطر رقم ١٤ تم استخدام الدوارة **while** لرسم مجموعة من الدوائر داخل حيز نافذة العرض.
- فى السطر رقم ١٧ تم استدعاء الدالة **InflateRect()** لوضع فاصل بين كل دائرة والأخرى.

قم ببناء التطبيق وتنفيذه، تلاحظ رسم مجموعة من الدوائر المتداخلة (انظر شكل ١٨-٤).



شكل ١٨-٤ رسم مجموعة من الدوائر المتداخلة باستخدام الدالة **Ellipse()**

رسم المنحنيات

يمكنك رسم المنحنيات باستخدام الدالة **()PolyBezier** التى تحمل بين طياتها الكثير من المعانى.فكلمة **Bezier** نسبةً إلى عالم الرياضيات الذى قام بتقليص المنحنيات إلى ما يسمى **Cubic Spline**، حيث تعبر كلمة **Spline** عن خط المنحنى وكلمة **Cubic** تعنى أن هناك أربعة نقاط لرسم المنحنى، واحدة لبداية المنحنى وأخرى لنهايته، بالإضافة إلى نقطتى تحكم يتم جذب الخط من عندهما لتكوين الانحناء. أما كلمة **Poly** فتعنى إمكانية رسم أكثر من منحنى مع بعضهما البعض. وعلى ذلك يتم تمرير مصفوفة تحتوى على أربعة نقاط على الأقل من النوع **CPoint** إلى الدالة **()PolyBezier** وكذلك تمرير عدد هذه النقاط.

للتعرف على طريقة استخدام الدالة **()PolyBezier** لرسم المنحنيات، قم بتعديل كود الدالة **()OnDraw** كما يلى:

```
1.    void CMyDrawView::OnDraw(CDC* pDC)
2.    {
3.      CMyDrawDoc* pDoc = GetDocument();
4.      ASSERT_VALID(pDoc);
5.      // TODO: add draw code for native data here
6.      CPen penRed(PS_DOT,1,RGB(255,0,0));
7.      CPen *pOldPen = NULL;
8.      pOldPen = pDC->SelectObject(&penRed);
9.      pDC->SelectStockObject(NULL_BRUSH);
10.     CRect rcClient;
11.     GetClientRect(&rcClient);
12.     CPoint ptBezier[4];
13.     CPen penBlue(PS_SOLID,1,RGB(0,0,255));
14.     pDC->SelectObject(&penBlue);
15.     for(int i=0; i<4;i++)
16.     {
17.       ptBezier[i] = CPoint(rand()%rcClient.Width() ,
                                        rand()%rcClient.Height());
18.        CString str;
19.        str.Format("%d.",i+1);
20.        pDC->TextOut(ptBezier[i].x,ptBezier[i].y,str);
21.        CRect rcDot(ptBezier[i],ptBezier[i]);
22.         rcDot.InflateRect(2,2);
```

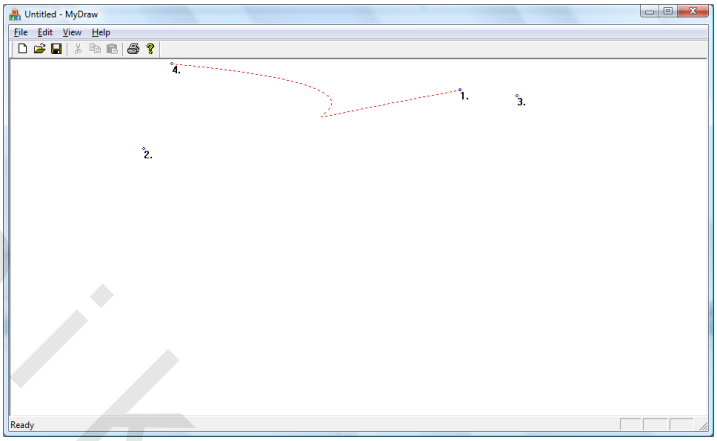
```
23.          pDC->Ellipse(rcDot);
24.        }
25.      pDC->SelectObject(&penRed);
26.      pDC->PolyBezier(ptBezier,4);
27.      pDC->SelectObject(pOldPen);
28.   }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ١٢ تم تعريف المصفوفة **ptBezier[]** التى تحتوى على الأربعة نقاط المستخدمة فى رسم المنحنى.
- فى السطور من ١٧ إلى ٢٠ داخل الدوارة **for** تم تعيين أربعة نقاط عشوائية داخل مصفوفة النقاط.
- فى السطر رقم ٢٠ تم استدعاء الدالة **()TextOut** لكتابة رقم كل نقطة (سنتعرض لهذه الدالة فى الفصل القادم إن شاء الله).
- فى السطر رقم ٢٣ تم استدعاء الدالة **()Ellipse** لرسم نقطة صغيرة زرقاء فوق كلٍ من الأرقام الأربعة.
- فى السطر رقم ٢٦ تم استدعاء **()PolyBezier** لرسم المنحنى المطلوب من النقطة ١ إلى النقطة ٤ على أن يتم قلب المنحنى عند النقطتين ٢و٣.

قم ببناء التطبيق وتنفيذه، تحصل على منحنى مختلف الأبعاد فى كل مرة تقوم فيها بتغيير وضع أو حجم النافذة أو إعادة تنشيطها (انظر شكل ١٨-٥).

obeikandl.com



شكل ١٨-٥ رسم منحنى باستخدام الدالة **Polyezier()**

رسم المضلعات

يمكنك رسم المضلعات باستخدام الدالة **PolyLine()** التى تتشابه كثيراً مع الدالة **PolyBezier()** فى أنها تحتوى على نفس المعاملات تقريباً، إلا أنها تقوم برسم خطوط مستقيمة بين النقاط بدلاً من رسم منحنى. للتعرف على طريقة استخدام الدالة **PolyLine()** لرسم المضلعات، قم باستبدال الدالة **PolyBezier()** بالدالة **PolyLine()** داخل الدالة **OnDraw()**، لتحصل على خطوط مستقيمة بدلاً من المنحنى. فإذا أردت رسم مضلع، يجب أن يكون إحداثى أول نقطة مساوياً لإحداثى آخر نقطة.

العمل مع فرش الألوان

تستخدم أقلام الرسم لرسم الخطوط وحدود الأشكال المختلفة، فإذا أردت تلوين هذه الأشكال، يجب أن تقوم باستخدام فرش الألوان. وغالباً ما تستخدم أقلام الرسم وفرش الألوان معاً لإنشاء الرسوم الجذابة عالية الجودة، وهذا ما سنتعرف عليه فى الأجزاء المتبقية من هذا الفصل.

استخدام التصنيف Cbrush

تحتوى مكتبة MFC على التصنيف CBrush المستخدم لتعريف فرشاة ألوان جديدة مصمتة اللون أو متقطعة أو تعبر عن صورة نقطية أو شكل من الأشكال. لإنشاء فرشاة ألوان مصمتة صفراء اللون مثلاً، قم باستخدام الكود التالى:

```
CBrush brYellow(RGB(192,192,0));
```

أما إذا أردت إنشاء فرشاة ألوان متقطعة، فيمكنك استخدام الكود التالى:

```
CBrush brYellowHatch(HS_DIAGCROSS,RGB(192,192,0));
```

حيث يعبر المعامل الأول عن نوع التقطع، ويمكنك اختيار النوع المناسب من الجدول ١٨-٣ التالى.

جدول ١٨-٣ أنواع فرش الألوان

النوع	التأثير
HS_CROSS	تقاطعات أفقية ورأسية
HS_DIAGCROSS	تقاطعات قطرية
HS_HORIZONTAL	خطوط أفقية
HS_VERTICAL	خطوط رأسية
HS_BDIAGONAL	خطوط قطرية (من الركن الأيسر العلوى إلى الركن الأيمن السفلى)
HS_FDIAGONAL	خطوط قطرية (من الركن الأيسر السفلى إلى الركن الأيمن العلوى)

تلوين خلفية النافذة

يمكنك استخدام فرش الألوان لتغيير لون خلفية النافذة باستخدام رسالة مسح خلفية النافذة WM_ERASEBKGND. لإضافة دالة احتواء تغيير لون خلفية النافذة بمجرد مسحها، تابع معنا الخطوات الآتية:

١. نشط التبويب Class View ثم اختر التصنيف CMyDrawView لتنشيطه.

obeikandl.com

٢. انقر زر الرسائل من شريط أدوات مربع الخصائص، تظهر قائمة بالرسائل المصاحبة للتصنيف.

٣. اختر الرسالة **WM_ERASEBKGND** ثم اختر **<Add> OnEraseBkgnd** من القائمة المصاحبة، يتم إضافة هيكل الدالة **()OnEraseBkgnd** إلى نافذة كود التصنيف **CMyDrawView**.

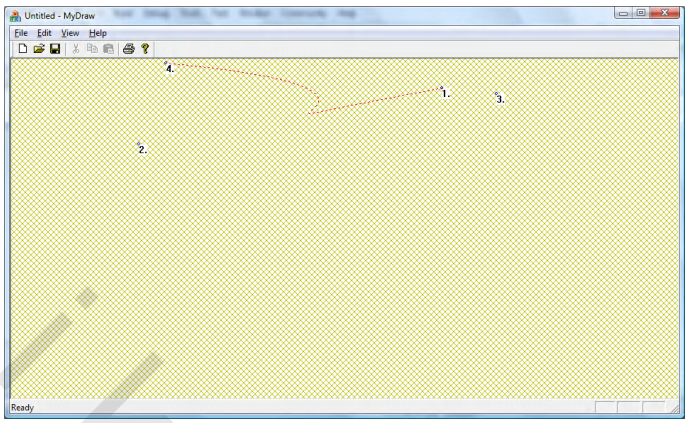
٤. يمكنك الآن استخدام الدالة لإضفاء لون معين على خلفية النافذة. فمثلاً إذا أردت تلوين خلفية النافذة باللون الأصفر المتقطع، قم بإدخال الكود التالى للدالة:

```
1.  BOOL CMyDrawView::OnEraseBkgnd(CDC* pDC)
2.  {
3.    // TODO: Add your message handler code here and/or call
            default
4.    CBrush brYellowHatch(HS_DIAGCROSS,RGB(192,192,0));
5.    CRect rcClient;
6.    GetClientRect(&rcClient);
7.    pDC->FillRect(rcClient,&brYellowHatch);
8.    return TRUE;
9.  }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٤ تم تعريف فرشاة ألوان صفراء متقطعة.
- فى السطور رقم ٥ و٦ تم الحصول على أبعاد النافذة.
- فى السطر رقم ٧ تم استدعاء الدالة **()FillRect** لإضفاء اللون على خلفية النافذة.
- فى السطر رقم ٨ يتم إرجاع القيمة **TRUE** للدلالة على نجاح تلوين خلفية النافذة.

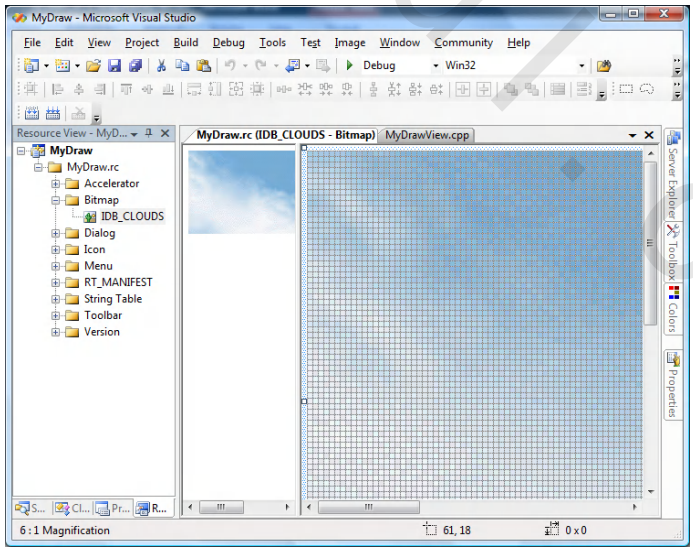
والآن قم ببناء التطبيق وتنفيذه، تحصل على خلفية صفراء متقطعة (انظر شكل ١٨-٦).



شكل ١٨-٦ تلوين خلفية النافذة

إنشاء فرش الألوان من الصور والأشكال

يمكنك إنشاء فرش ألوان منبثقة من صور أو أشكال. قم بإنشاء صورة نقطية جديدة داخل محرر الموارد أو إحضار صورة موجودة مسبقاً كما تعلمت من قبل (انظر شكل ١٨-٧) ثم قم بتعديل كود الدالة **OnEraseBkgnd()** لإنشاء فرشاة الألوان من الصورة النقطية الجديدة وذلك كما يلي:



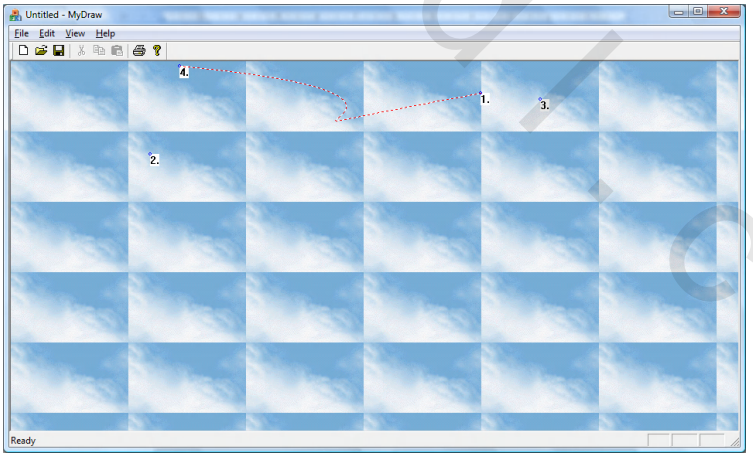
شكل ١٨-٧ إحضار صورة جديدة للتطبيق

```
1.    BOOL CMyDrawView::OnEraseBkgnd(CDC* pDC)
2.    {
3.      // TODO: Add your message handler code here and/or call
                default
4.      CBitmap bmCludes;
5.      bmCludes.LoadBitmap(IDB_CLUDES);
6.      CBrush brCludes(&bmCludes);
7.      CRect rcClient;
8.      GetClientRect(&rcClient);
9.      pDC->FillRect(rcClient,&brCludes);
10.     return TRUE;
11.   }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٤ تم تعريف متغير صورة نقطية جديد.
- فى السطر رقم ٥ تم استدعاء الدالة **LoadBitmap()** لتحميل الصورة التى قمنا بإنشائها إلى المتغير **bmCludes**.
- فى السطر رقم ٦ تم إنشاء فرشاة ألوان بلون الصورة الجديدة.

قم ببناء التطبيق وتنفيذه، تلاحظ تلوين خلفية النافذة بلون الصورة (انظر شكل ١٨-٨).



شكل ١٨-٨ تلوين خلفية النافذة بلون الصورة

استخدام فرش الألوان المخزَّنة

كما ذكرنا مع أقلام الرسم المخزَّنة، هناك العديد من فرش الألوان المعرفة مسبقاً والمخزَّنة داخل نظام التشغيل والتى يمكن بيانها بالجدول ١٨-٤ التالى.

جدول ١٨-٤ أنواع الفرش المخزَّنة داخل نظام التشغيل

اسم الفرشاة	*التأثير*
BLACK_BRUSH	طلاء أسود
WHITE_BRUSH	طلاء أبيض
DKGRAY_BRUSH	طلاء رمادى غامق
LTGRAY_BRUSH	طلاق رمادى فاتح
GRAY_BRUSH	طلاء رمادى
NULL_BRUSH	طلاء بلون الخلفية

يتم تمرير إحدى هذه القيم للدالة **SelectStockObject()** كما رأينا مع المعامل **NULL_BRUSH** قبل ذلك. يمكنك أيضاً استخدام الدالة **CreateSysColorBrush()** لتخصيص أحد ألوان النظام المختلفة لفرشاة الألوان والتى يمكن توضيحها بالجدول ١٨-٥ التالى.

جدول ١٨-٥ أنواع ألوان النظام

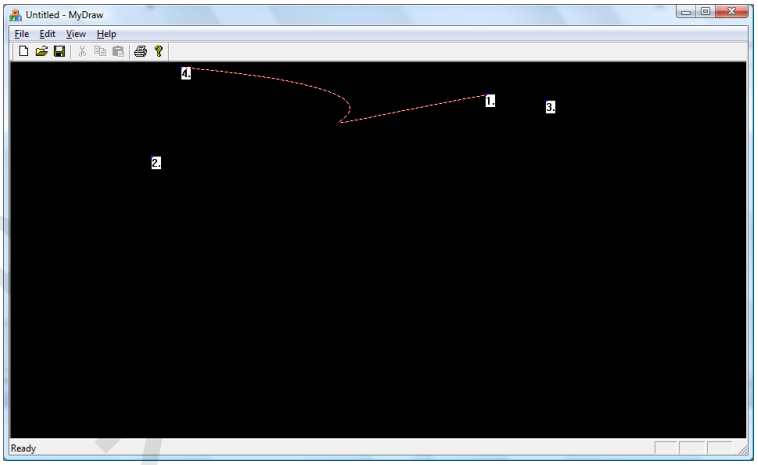
اسم اللون	*التأثير*
COLOR_DESKTOP	لون خلفية سطح المكتب
COLOR_BTNTEXT	لون نص الأزرار
COLOR_GRAYTEXT	لون رمادى معطَّل
COLOR_HIGHLIGHTTEXT	لون نص العناصر المختارة
COLOR_HIGHLIGHT	لون خلفية العناصر المختارة
COLOR_ACTIVEBORDER	لون إطار النافذة النشطة

اسم اللون	التأثير
COLOR_INACTIVEBORDER	لون إطار النافذة الغير نشطة
COLOR_INFOBK	لون خلفية التلميحات
COLOR_INFOTEXT	لون نص التلميحات
COLOR_WINDOW	لون خلفية النافذة
COLOR_WINDOWFRAME	لون إطار النافذة
COLOR_WINDOWTEXT	لون النص الموجود بالنافذة
COLOR_3DDKSHADOW	لون ظل الأزرار
COLOR_3DFACE	لون واجهة الأزرار
COLOR_3DHILIGHT	لون واجهة الأزرار المختارة
COLOR_3DLIGHT	لون إطار الزر

والآن قم بتغيير كود الدالة **OnEraseBkgnd()** لإنشاء فرشاة ألوان بلون سطح المكتب كما يلى:

```
1.    BOOL CMyDrawView::OnEraseBkgnd(CDC* pDC)
2.    {
3.      // TODO: Add your message handler code here and/or call
              default
4.      CBrush brDesktop;
5.      brDesktop.CreateSysColorBrush(COLOR_DESKTOP);
6.      CRect rcClient;
7.      GetClientRect(&rcClient);
8.      pDC->FillRect(rcClient,&brDesktop);
9.      return TRUE;
10.   }
```

قم ببناء التطبيق وتنفيذه، تلاحظ تلون النافذة بنفس لون سطح المكتب (انظر شكل ١٨-٩).



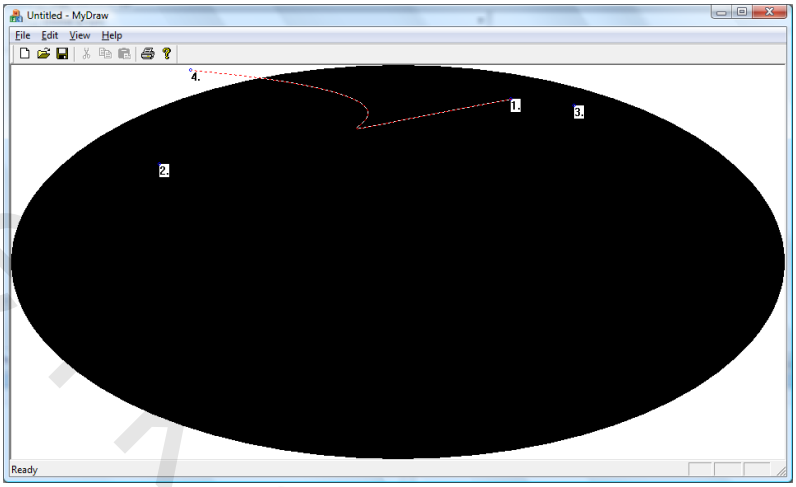
شكل ١٨-٩ تلوين النافذة بنفس لون سطح المكتب

اختيار فرش الألوان

قبل أن تقوم باستخدام فرش الألوان التى قمت بإنشائها، يجب أن تقوم أولاً باختيار فرشاة الألوان التى تريد استخدامها باستخدام الدالة **SelectObject()** كما كنا نفعل مع أقلام الرسم. للتعرف على طريقة اختيار فرشاة ألوان جديدة واسترجاع الفرشاة القديمة، قـــم بتعديل كود الدالة **OnEraseBkgnd()** كما يلى:

```
BOOL CMyDrawView::OnEraseBkgnd(CDC* pDC)
{
// TODO: Add your message handler code here and/or call
default
CBrush brDesktop;
brDesktop.CreateSysColorBrush(COLOR_DESKTOP);
CRect rcClient;
GetClientRect(&rcClient);
CBrush* pOldBrush = pDC->SelectObject(&brDesktop);
pDC->Ellipse(rcClient);
pDC->SelectObject(pOldBrush);
return TRUE;
}
```

ولأننا استخدمنا قطع ناقص كخلفية للنافذة بلون سطح المكتب، فستحصل على أشـــكال غريبة بمجرد تشغيل التطبيق (انظر شكل ١٨-١٠).



شكل ١٨-١٠ رسم خلفية النافذة كقطع ناقص

حذف فرش الألوان

عندما تنتهي من استخدام أيٍ من فرش الألوان داخل بيئة العنصر DC، يجب أن تقوم بحذف هذه الفرشاة لتحرير الموارد المرتبطة بها، وهذا ما يقوم به التصنيف CPen نيابةً عنك. أما إذا أردت استخدام نفس الفرشاة بإعدادات أخرى، فيجب أن تقوم بحذفها يدوياً باستخدام الدالة DeleteObject() ثم إعادة استخدامها بإعدادات جديدة باستدعاء أيٍ من دوال الإنشاء التالية:

- الدالة CreateSolidBrush() لإنشاء فرشاة ألوان مصمتة.
- الدالة CreateHatchBrush() لإنشاء فرشاة ألوان متقطعة.
- الدالة CreatePaternBrush() لإنشاء فرشاة ألوان من صورة نقطية أو شكل من الأشكال.
- الدالة CreateSysColorBrush() لإنشاء فرشاة ألوان من أحد ألوان النظام.

عند حذف فرشاة ألوان صورة نقطية، يجب ألا تنسى حذف الصورة نفسها لأن عملية الحذف لا تتم تلقائياً.

رسم الأشكال الملونة باستخدام الفرش

تستخدم دوال الرسم فرش الألوان لرسم الأشكال الملونة مثل المضلعات والأوتار والمستطيلات، وهذا ما سنقوم بتوضيحه فى الأجزاء التالية.

رسم المستطيلات

يمكنك استخدام الدوال **Rectangle()** و**RoundRect()** لرسم الأشكال المستطيلة. تحتوى الدالة **Rectangle()** على نفس معاملات الدالة **Ellipse()** التى تعرضنا لها فيما سبق، أما الدالة **RoundRect()** فتحتوى على معامل آخر من النوع **CPoint** أو معاملين صحيحين لتحديد عرض القطع الناقص الذى سيتم رسمه عند أركان المستطيل لنحصل على مستطيل محاط من الخارج.

قم بتعديل كود الدالة **OnEraseBkgnd()** لاستدعاء الدالة **Rectangle()** بدلاً من الدالة **Ellipse()** لرسم مستطيل بدلاً من القطع الناقص كما فى السطر التالى:

```
pDC->Rectangle(rcClient);
```

تتطلب الدالة **FillRect()** مؤشر لفرشاة الألوان المراد استخدامها، أما الدالة **Rectangle()** فتستخدم فرشاة الألوان الحالية.

للتعرف على طريقة رسم المستطيلات المحاطة، قم بتعديل كود الدالة **OnDraw()** كما يلى:

```
1.  void CMyDrawView::OnDraw(CDC* pDC)
2.  {
3.    CMyDrawDoc* pDoc = GetDocument();
4.    ASSERT_VALID(pDoc);
5.    //TODO: add draw code for native data here
6.    CPen penRed(PS_SOLID,5,RGB(255,0,0));
7.    CPen *pOldPen = NULL;
8.    pOldPen = pDC->SelectObject(&penRed);
9.    CRect rcClient;
10.   GetClientRect(&rcClient);
11.   CBrush brBlue(RGB(0,0,255));
```

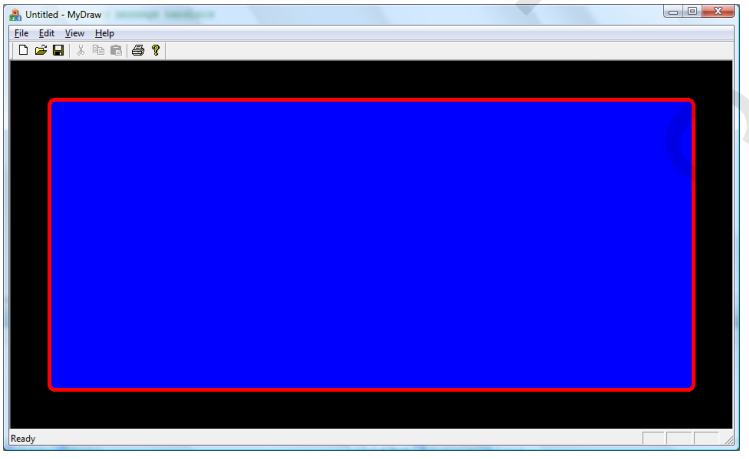

obeikandl.com

```
12.     CBrush *pOldBrush = NULL;
13.     pOldBrush = pDC->SelectObject(&brBlue);
14.     rcClient.DeflateRect(50,50);
15.     pDC->RoundRect(rcClient,CPoint(15,15));
16.     pDC->SelectObject(pOldBrush);
17.     pDC->SelectObject(pOldPen);
18. }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٦ تم استبدال القلم الأحمر المنقط بقلم أحمر مصمت حتى يظهر الحد الخارجى بوضوح.
- فى السطر رقم ١١ تم تعريف فرشاة ألوان زرقاء وفى السطر رقم ١٣ تم اختيارها داخل بيئة العنصر.
- فى السطر رقم ١٤ تم استدعاء الدالة **DeflateRect()** لإزاحة المستطيل داخل النافذة بمقدار ٥٠ نقطة أفقى و ٥٠ نقطة رأسى.
- فى السطر رقم ١٥ تم استدعاء الدالة **RoundRect()** لرسم مستطيل محاط بقطع ناقص بعرض ١٥ نقطة.

قم ببناء التطبيق وتنفيذه، تحصل على مستطيل أزرق اللون محاط بحد أحمر اللون على خلفية بلون سطح المكتب (انظر شكل ١٨-١١).



شكل ١٨-١١ نتيجة تنفيذ الكود السابق

رسم الدوائر الملونة والقطع الناقص

لا تختلف طريقة رسم الدوائر الملونة والقطع الناقص الملون عن مثيلاتها التى بدون ألوان إلا فى تعيين واختيار فرشاة ألوان بدلاً من NULL_BRUSH قبل استدعاء دالة الرسم المناسبة. قم بإضافة الكود التالى بعد سطر استدعاء الدالة RoundRect() لرسم القطع الناقص.

```
rcClient.DeflateRect(25,25);
pDC->Ellipse(rcClient);
```

قم ببناء التطبيق وتنفيذه، تلاحظ ظهور قطع ناقص صغير داخل المستطيل المحاط.

رسم الأوتار

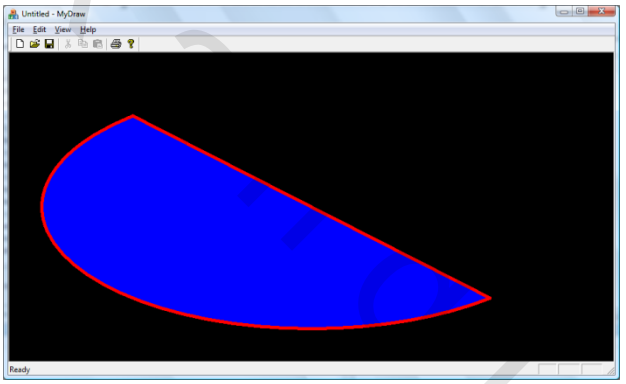
عند رسم قطع ناقص بداخله خط مستقيم ثم تقسيمه إلى جزأين، فإن كل جزء يسمى وتر Chord. للتعرف على طريقة استخدام الدالة Chord() لرسم الأوتار، قم بتعديل كود الدالة OnDraw() كما يلى:

```
1.  void CMyDrawView::OnDraw(CDC* pDC)
2.  {
3.    CMyDrawDoc* pDoc = GetDocument();
4.    ASSERT_VALID(pDoc);
5.    // TODO: add draw code for native data here
6.    CPen penRed(PS_SOLID,5,RGB(255,0,0));
7.    CPen *pOldPen = NULL;
8.    pOldPen = pDC->SelectObject(&penRed);
9.    CRect rcClient;
10.   GetClientRect(&rcClient);
11.   CBrush brBlue(RGB(0,0,255));
12.   CBrush *pOldBrush = NULL;
13.   pOldBrush = pDC->SelectObject(&brBlue);
14.   CRect rcChord = rcClient;
15.   rcChord.DeflateRect(50,50);
16.   pDC->Chord(rcChord,rcClient.TopLeft(),
17.   rcClient.BottomRight());
18.   pDC->SelectObject(pOldBrush);
19.   pDC->SelectObject(pOldPen);
20. }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ١٤ تم تعريف المتغير **rcChord** من النوع **CRect** ليحتوى على المنطقة التى سيتم استخدامها لرسم الوتر وتم إعطاؤه النافذة الأساسية كقيمة ابتدائية.
- فى السطور ١٦ و١٧ تم استدعاء الدالة **()Chord** لرسم الوتر، حيث تحتوى الدالة على ثلاثة معاملات. الأول عبارة عن المنطقة التى سيتم رسمها والثانى والثالث يحتويان على إحداثى الركن الأيسر العلوى والركن الأيمن السفلى على الترتيب.

قم ببناء التطبيق وتنفيذه، تحصل على وتر داخل حيز النافذة (انظر شكل ١٨-١٢).



شكل ١٨-١٢استخدام الدالة **()Chord** لرسم الأوتار

رسم المضلعات

تستخدم الدالة **()Polygon** فى رسم المضلعات، حيث تحتوى على نفس المعاملات المستخدمة مع الدوال **()PolyLine** و**()PolyBezier**، إلا أنك لست فى حاجة لأن تجعل إحداثى نقطة البداية مساوياً لإحداثى نقطة النهاية لأن ذلك يتم تلقائياً.

عند العمل مع الدالة **()Polygon** يجب تحديد نمط الملأ باستخدام الدالة **()SetPolyFillMode** التى تأخذ القيمة **ALTERNATE** لملء المضلع بخطوط من اليسار إلى اليمين، أو القيمة **WINDING**.

يمكنك استرجاع النمط الحالى للملء باستدعاء الدالة **GetPolyFillMode()** التى تقوم بإرجاع القيمة **ALTERNATE** أو **WINDING** تبعاً للنمط الحالى.

للتعرف على طريقة رسم المضلعات باستخدام الدالة **Polygon()**، قم بتعديل كود الدالة **OnDraw()** كما يلى:

```
1.    void CMyDrawView::OnDraw(CDC* pDC)
2.    {
3.      CMyDrawDoc* pDoc = GetDocument();
4.      ASSERT_VALID(pDoc);
5.      // TODO: add draw code for native data here
6.      CPen penRed(PS_SOLID,5,RGB(255,0,0));
7.      CPen *pOldPen = NULL;
8.      pOldPen = pDC->SelectObject(&penRed);
9.      CRect rcClient;
10.     GetClientRect(&rcClient);
11.     CBrush brBlue(RGB(0,0,255));
12.     CBrush *pOldBrush = NULL;
13.     pOldBrush = pDC->SelectObject(&brBlue);
14.     for(int w=0;w<2;w++)
15.     {
16.       const int nPoints = 5;
17.       double nAngle = (720.0/57.295)/(double)nPoints;
18.       int xOffset = (w ? 1:-1)*rcClient.Width()/4;
19.       pDC->SetPolyFillMode(w ? ALTERNATE :WINDING);
20.       CPoint ptPolyAr[nPoints];
21.       for(int i=0;i<nPoints;i++)
22.       {
23.         ptPolyAr[i].x = xOffset + (long)(sin((double)i *
                                                  nAngle) * 100.0);
24.         ptPolyAr[i].y = xOffset + (long)(cos((double)i *
                                                  nAngle) * 100.0);
25.         ptPolyAr[i] += rcClient.CenterPoint() ;
26.       }
27.       pDC->Polygon(ptPolyAr,nPoints);
28.     }
29.     pDC->SelectObject(pOldBrush);
30.     pDC->SelectObject(pOldPen);
31.   }
```

لا تنسى إضافة السطر #include "math.h" فى أول الملف كـــى يســـتطيع البرنامج التعرف على الدوال sin() و cos().

قم ببناء التطبيق وتنفيذه، تقوم الدالة برسم نجمتين باستخدام الدالة Polygon()، الأولى فى اليسار تعمل بالنمط WINDING والأخرى فى اليمين تعمل بالنمط ALTERNATE (انظر شكل ١٨-١٣).

شكل ١٨-١٣رسم المضلعات باستخدام الدالة Polygon()