

## الفصل السابع تحسين الواجهات بالرسم

نشرح في هذا الفصل تحسين واجهة المستخدم (النماذج والنوافذ) بالخطوط والنقاط والدوائر والصور المختلفة سواءً باستخدام الوظائف التي تقوم برسم الخطوط والدوائر والمنحنيات أو بإظهار الصور من خلال الأداة **PictureBox** ونوضح فوائده ومساوئ كل طريقة.

بانتهاه هذا الفصل سنتعرف على:

- ◆ تحسين واجهة المستخدم بالرسم
- ◆ عرض الصور على الأدوات والنماذج
- ◆ استخدام أداة مربع الصورة **PictureBox**
- ◆ تعيين خلفية الأداة
- ◆ إنشاء الرسوم أثناء التشغيل
- ◆ استخدام الأقلام وفرش الرسم والألوان
- ◆ رسم الخطوط والأشكال

## تحسين واجهة المستخدم بالرسم

واجهة المستخدم النمطية عبارة عن عدة نوافذ بها قوائم ومربعات نصوص وعناوين وأزرار أوامر وما إلى ذلك. ولكن لو اكتفينا بذلك لصارت هذه الواجهة مملة وغير جذابة. إدخال الرسوم إلى الواجهة يحسن الواجهة من النواحي التالية:

- لفت الانتباه إلى معلومات معينة على الشاشة
  - تستخدم لعرض المعلومات بطرق مختلفة (طريقة رسومية)
  - تجعل فهم وظيفة العناصر في كثير من الأحيان بديهية وسهلة
- هذه التحسينات تتم كما سنرى من خلال طريقتين مختلفتين هما:
- إضافة الصور إلى أدوات التحكم التي تدعم أساساً استخدام الصور مثل أداة مربع الصور PictureBox أو زر الأمر أو حتى النموذج نفسه.
  - استخدام وظائف الرسم المختلفة الموجودة بتصنيف الرسوم Graphics داخل

..NET

وسوف نتعرف على كلتا الطريقتين من خلال هذا الفصل.

تحتوي الإصدارات التي تسبق Visual Basic .Net مثل Visual Basic 6.0 على أدوات التحكم Line و Shape التي يمكن من خلالها رسم الخطوط والأشكال المختلفة كالمستطيلات والدوائر والمربعات ... الخ. أما Visual Basic 2008 فلم يعد يدعم أيّاً من الأدوات. وإنما يتم رسم هذه الأشكال من خلال الوظائف فقط.



## معرض الصور على الأدوات والنماذج

يدعم الكثير من أدوات التحكم الموجودة في Visual Basic 2008 بما في ذلك النماذج تضمين الصور ذات التنسيقات المختلفة وذلك من خلال خاصيتين مشتركتين بين جميع هذه الأدوات وهما الخاصية Image والخاصية BackgroundImage التي يمكنك تخصيص صورة لأيٍّ منها، حيث يمكنك استخدام أحد تنسيقات الصور الموضحة في جدول ٧-١

التالى. ونقصد بالصور **Pictures** الرسوم النقطية **Bitmaps** التي تحتوى رسوم فنية أو مخططات توضيحية أو صور فوتوغرافية. يمكن لهذه الصور أيضاً أن تحمل بعض المعلومات غير تحسين المظهر. مثال لذلك الصور التي تقوم برامج الإعداد بعرضها لتسليتك أثناء الإعداد حيث تقوم بعرض خصائص البرنامج الذي يتم إعداده.

جدول ٧-١ أنواع ملفات الرسوم التي يعرضها Visual Basic

امتداد الملف	نوعية الملف
.BMP	ملف صور نقطي <b>Windows bitmap</b>
.ICO	ملف رمز <b>Icon</b>
.WMF	ملف أوامر رسم <b>Windows Metafile</b>
.EMF	ملف أوامر رسم محسن <b>Enhanced Metafile</b>
.GIF	الاسم اختصار لعبارة <b>Graphics Interchange Format</b> و هو نوع من الملفات يستخدم بكثرة على الإنترنت و هو نقطي أيضاً
.JPG .JPEG	الاسم اختصار لعبارة <b>Joint Photographic Experts Group</b> وهي المجموعة التي ابتكرت هذا النوع من الملفات . وهو نوع يستخدم مع الملفات ذات الألوان الحقيقية ، لكن به أسلوب للضغط يوفر في حجم الملف - وهو نوع منتشر أيضاً على الإنترنت
.PNG	وهو اختصار للعبارة <b>Portable Network Graphics</b> ويعاد امتداد للتنسيق <b>GIF</b> ويجمع بين كفاءة الصورة وقلة حجم الملف كما أنه أحد التنسيقات الجديدة المستخدمة على الإنترنت

#### استخدام الخاصية *Image*

يمكنك من خلال الخاصية **Image** تعيين الصورة التي ترغب في إضافتها للأداة حيث تُدعم هذه الخاصية من قِبَل أدوات تحكم مربع الصورة **PictureBox** وزر الأمر **Button** وأداة العنوان **Label** وعنوان الارتباط **LinkLabel** ومربع الاختيار **CheckBox**

وأخيراً زر الاختيار **Radio Button**. وهذه الخاصية كثيرة الاستخدام مع أداة مربع الصورة التي تم تصميمها خصيصاً لاحتواء الصور والرسوم. أما الأدوات الأخرى فتستخدم هذه الخاصية في حالة الرغبة في تدعيم نصوصها بالصور أو الاستغناء أساساً عن النصوص. فزر الأمر الذى يقوم بإجراء أمر الطباعة على الطباعة، من الممكن أن يحتوى على صورة للطباعة بجانب أو بدون كلمة "طباعة" أو **Print**. يوضح شكل ٧-١ استخدام الخاصية **Image** مع كل أداة من أدوات التحكم التي تدعم هذه الخاصية.



شكل ٧-١ استخدام الخاصية **Image** لتضمين الصور مع أدوات التحكم.

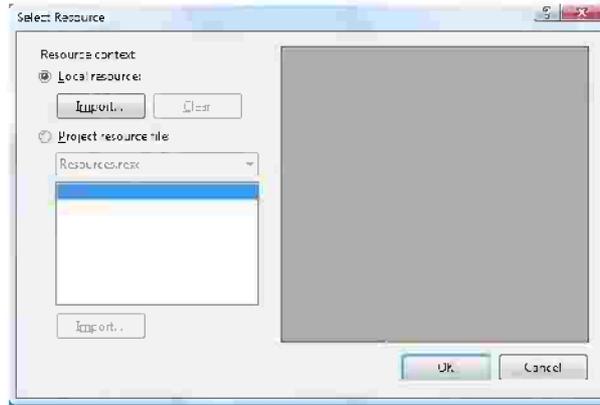
#### تعيين الخاصية **Image** أثناء التصميم

يمكنك تعيين قيمة الخاصية **Image** أثناء التصميم أو حتى أثناء التشغيل من خلال الكود.

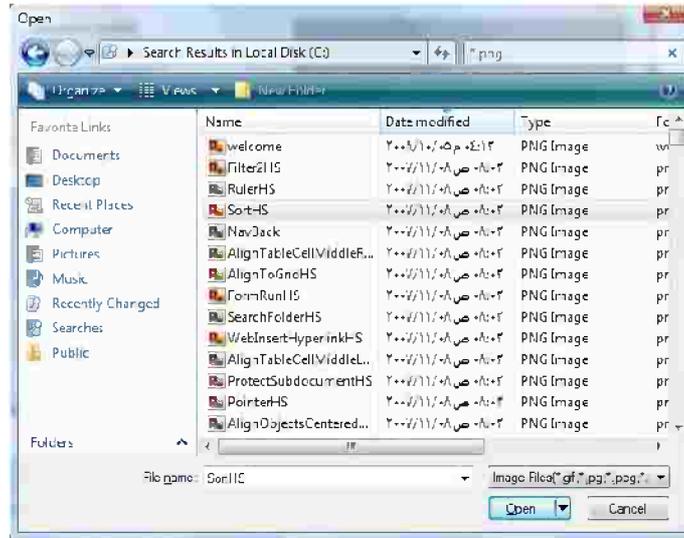
لتعيين قيمة الخاصية، أثناء التصميم، تابع معنا الخطوات الآتية:

١. اختر الأداة التي ترغب في تعيين قيمة الخاصية لها داخل نافذة التصميم.
٢. انقر الزر  المجاور للخاصية **Image** داخل المجموعة **Appearance** بمربع الخصائص، يظهر المربع الحوارى **Select Resource** (انظر شكل ٧-٢).
٣. يحتوى المربع الحوارى **Select Resource** على جزأين، الأول **Local Resource** ويتم من خلاله تحديد إحدى الصور الموجودة على الحاسب أو المتصلة بالشبكة، والثانى **Project resource file** ويتم من خلاله اختيار إحدى

الصور التي تم بالفعل إضافتها إلى المشروع من قبل. نشط زر الاختيار المناسب ثم انقر زر **Import**، يظهر المربع الحوارى **Open** المعتاد (انظر شكل ٧-٣).



شكل ٧-٢ المربع الحوارى **Select Resource**.



شكل ٧-٣ اختيار الصورة المناسبة من المربع الحوارى **Open**.

٤. اختر الصورة التي ترغب في تضمينها ثم انقر زر **Open**، يتم إضافة الصورة إلى أداة التحكم.

حينما تستخدم الخاصية **Image** مع إحدى الأدوات، يمكنك أيضاً استخدام الخاصية **ImageAlign** التي تحدد محاذاة الصورة داخل الأداة حيث يوجد تسع قيم مختلفة وهى القيم الموضحة بجدول ٧-٢ التالى.

جدول ٧-٢ قيم الخاصية **ImageAlign**

وضع الصورة	القيمة
الركن الأيسر العلوى من الأداة	TopLeft
منتصف أعلى الأداة	TopCenter
الركن الأيمن العلوى من الأداة	TopRight
منتصف الحافة اليسرى من الأداة	MiddleLeft
منتصف الأداة (القيمة الافتراضية)	MiddleCenter
منتصف الحافة اليمنى من الأداة	MiddleRight
الركن الأيسر السفلى من الأداة	BottomLeft
منتصف الحافة السفلية من الأداة	BottomCenter
الركن الأيمن السفلى من الأداة	BottomRight

إذا قمت بإضافة صورة إلى أداة تحكم تحتوى أساساً على عنوان أو نص (مثل زر الأمر **Button**)، يمكنك استخدام الخاصية **TextAlign** لتحديد محاذاة العنوان على الأداة إلى جانب استخدام الخاصية **ImageAlign** لتحديد محاذاة الصورة على الأداة.



تعيين الخاصية **Image** وقت التشغيل

كما يمكنك تعيين قيمة الخاصية **Image** أثناء التصميم من خلال اختيار إحدى الصور من المربع الحوارى **Open** أو المربع الحوارى **Select Resource**، يمكنك أيضاً تعيين

قيمتها أثناء التشغيل. فإذا أردت مثلاً تعيين قيمة الخاصية لأداة مربع الصور PictureBox1، قم باستخدام العبارة التالية:

```
PictureBox1.Image = Image.FromFile("c:\Mohamed.gif")
```

وإذا أردت في أي وقت حذف هذه الصورة، قم بتخصيص القيمة Nothing للخاصية Image كما يلي:

```
PictureBox1.Image = Nothing
```

لكن ربما تتساءل الآن عن أي الطريقتين أفضل، تعيين الصور أثناء مرحلة التصميم أم تعيينها أثناء التشغيل؟. في الواقع هناك بعض الاعتبارات التي يجب أن تضعها في خلدك قبل اتخاذ القرار. فحينما تقوم بتعيين الصورة في مرحلة التصميم، يتم تضمين الصورة برمتها داخل تطبيقك ومن ثم يتم تضمينها أثناء ترجمة هذا التطبيق. وعلى هذا كلما زاد عدد الصور كلما زاد حجم التطبيق. أما إذا قمت بتحميل الصور وقت التشغيل من خلال الكود، فلا يتم ترجمة هذه الصور داخل التطبيق كما أنك تستطيع تغيير هذه الصور كيفما شئت. ولكن كن حذراً، فإذا لم يجد البرنامج الصورة المحددة أثناء التشغيل، يقوم على الفور بإظهار رسالة الخطأ المناسبة وعليك في هذه الحالة احتواء هذا الخطأ حتى لا يؤدي إلى توقف التطبيق بالكامل.

## استخدام أداة مربع الصور PictureBox

يستخدم مربع الصورة في عرض الصور والرسوم المختلفة على النماذج، مثل الصورة الطبيعية الموجودة بشكل 7-1 السابق. ولأن هذا المربع هو حاوي الصورة، فيمكنك التحكم في موقع الصورة على النموذج ببساطة شديدة من خلال تعيين قيمة الخاصية Location الخاصة بمربع الصورة. ومن الضروري أيضاً تعيين قيمة الخاصية SizeMode التي تحدد كيفية تحجيم الصورة داخل المربع أو تحجيم المربع كي يظهر الصورة بالكامل، فغالباً ما يختلف حجم الصورة عن حجم مربع الصورة المستخدم. يوضح جدول 7-3 التالي القيم المتاحة للخاصية SizeMode وتأثير كل منها.

جدول 7-3 قيم الخاصية SizeMode داخل مربع الصورة PictureBox

التأثير	القيمة
لا يتم إعادة تحجيم الصورة أو مربع الصورة. فإذا كانت الصورة أكبر من المربع، يتم قص الصورة. أما إذا كان المربع أكبر من الصورة، فيتم وضع الصورة بالركن الأيسر العلوي من المربع.	Normal
يتم إعادة تحجيم المربع كي يتوافق مع حجم الصورة.	AutoSize
يتم إعادة تحجيم الصورة كي تتوافق مع حجم المربع.	StretchImage
إذا كان المربع أكبر من حجم الصورة، يتم توسيط الصورة داخل المربع. أما إذا كانت الصورة أكبر من المربع، فيتم توسيط الصورة داخل المربع وقص الأطراف الزائدة.	CenterImage
يتم زيادة حجم الصورة أو إنقاصه مع الحفاظ على النسبة الموجودة بحجم الصورة (نسبة الطول إلى العرض)	Zoom

تحتوي الإصدارات التي تسبق Visual Basic.Net على الخاصية Picture الخاصة بالنماذج والتي تمكّنك من وضع صورة بالركن الأيسر العلوي من النموذج دون إمكانية إعادة تحجيم الصورة. يمكنك تطبيق هذه الوظيفة داخل Visual Basic 2008 عن طريق وضع الصورة المطلوبة داخل مربع الصورة PictureBox ثم وضع المربع نفسه في المكان المناسب.



يوضح شكل ٧-٤ التالي تأثير كل قيمة من القيم الخمسة على ظهور الصورة داخل مربع الصور.



شكل ٧-٤ تأثير الخاصية SizeMode على ظهور الصورة داخل مربع الصورة.

### تعيين خلفية الأداة

يحتوي العديد من أدوات التحكم بما في ذلك النماذج نفسها على الخاصية **BackgroundImage** التي تستخدم لتعيين صورة كخلفية للأداة أو النموذج. وهذه الأدوات هي زر الأمر **Button** ومربع الاختيار **CheckBox** وزر الاختيار **RadioButton** ومربع المجموعة **GroupBox** ومربع الصورة **PictureBox** وأخيراً اللوحة **Panel**. وعادةً تستخدم صور الخلفية في تزيين مظهر الأداة أو النموذج بعكس الصورة **Image** التي تستخدم دائماً لتوضيح وظيفة الأداة (انظر شكل ٧-٥).

إذا كانت صورة الخاصية **BackgroundImage** أصغر من الأداة المستخدمة،  
تتمدد الصورة أفقياً ورأسياً كي تملأ الأداة بالكامل.





شكل ٧-٥ إضافة خلفية إلى النموذج.

يمكنك تعيين قيمة الخاصية **BackgroundImage** في مرحلة التصميم أو أثناء التشغيل كما يمكنك حذف قيمة هذه الخاصية أثناء التشغيل باتباع نفس الخطوات المستخدمة مع الخاصية **Image**.



### إنشاء الرسوم أثناء التشغيل

ناقشنا في الأجزاء السابقة من الفصل إظهار الصور الجاهزة المعدة مسبقاً من برنامج للرسم ولكن **Visual Basic** لا تقف قدرته عند هذا الحد، بل من الممكن تعديل وإنشاء هذه الرسوم بالأوامر من خلال الكود. تحتاج ذلك مثلاً عند عرض مخططات بيانية لبيانات شركة مثلاً. ولأن البيانات تتغير فيصعب إنشاء صور جاهزة لها، ويلزم أن ينشئها البرنامج أثناء تشغيله.

يتم إنشاء الرسوم في **Visual Basic** أساساً بوظائف الرسم الخاصة بالنافذة ومربع الصور بالإضافة إلى استخدام الكائن **Printer** في الطباعة على الطباعة المتاحة في **Windows**.

هناك خمس وظائف أساسية لإنشاء الرسوم وهي:

- الوظيفة **DrawLine()** وتستخدم لرسم خط مستقيم بين نقطتين.
- الوظيفة **DrawRectangle()** وتستخدم لرسم مربع أو مستطيل.

- الوظيفة `DrawEllipse()` وتستخدم لرسم دائرة أو شكل بيضاوى.
  - الوظيفة `DrawPolygon()` وتستخدم لرسم شكل مغلق يحتوى على عدد من الجوانب (مضلع).
  - الوظيفة `Clear()` وتستخدم فى إزالة الرسوم الموجودة على أحد الكائنات.
- وفيما يلى نوضح الصيغة العامة لكل وظيفة من هذه الوظائف.

#### الوظيفة `DrawLine()`

تقوم هذه الوظيفة برسم خط بين نقطتين وتحتوى على الصيغة العامة التالية:  
`control.CreateGraphics.DrawLine(Pen, x1, y1, x2, y2)`

حيث:

- `control` تعبر عن الأداة أو النموذج الذى يتم رسم الخط بداخله
- `Pen` يعبر عن القلم المستخدم فى عملية الرسم والذى يتم تحديده مسبقاً
- `x1,y1` تعبر عن إحداثى نقطة بداية الخط
- `x2,y2` تعبر عن إحداثى نقطة نهاية الخط

#### الوظيفة `DrawRectangle()`

تقوم هذه الوظيفة برسم مستطيل محدد بزواج من الإحداثيات وعرض وطول وتحتوى على الصيغة التالية:

`control.CreateGraphics.DrawRectangle(Pen, x, y, width, height)`

حيث:

- `control` تعبر عن الأداة أو النموذج الذى يتم رسم المستطيل بداخله
- `Pen` يعبر عن القلم المستخدم فى عملية الرسم والذى يتم تحديده مسبقاً
- `x,y` تعبر عن إحداثى الركن الأيسر العلوى من المستطيل
- `Width` تعبر عن عرض المستطيل
- `Height` تعبر عن ارتفاع المستطيل

### الوظيفة *DrawEllipse()*

تقوم هذه الوظيفة برسم قطع ناقص عن طريق تعريف المستطيل المحيط به والذي يحتوى على إحداثى الركن الأيسر العلوى والعرض والطول كما سبق، لذا فهي تحتوى على الصيغة العامة التالية:

**control.CreateGraphics.DrawEllipse(Pen, x, y, width, height)**

وكما ترى، تحتوى هذه الوظيفة على نفس معاملات الوظيفة السابقة.

### الوظيفة *DrawPolygon()*

تقوم هذه الوظيفة برسم مضلع باستخدام مصفوفة من النقاط التي تحدد أبعاد هذا المضلع وعدد أضلاعه وتحتوى على الصيغة العامة التالية:

**control.CreateGraphics.DrawPolygon(Pen, PointsArray)**

حيث **PointsArray** عبارة عن مصفوفة من النقاط.

### الوظيفة *Clear()*

تقوم هذه الوظيفة بمسح جميع الرسوم الموجودة على الأداة أو النموذج وإحلالها بلون الخلفية وتحتوى على الصيغة العامة التالية:

**control.CreateGraphics.Clear(Color)**

حيث يعبر **Color** عن لون الخلفية الذي ترغب في استخدامه وهو اختياري. فإذا أهملته يتم استخدام لون الخلفية الحالى للأداة.

## استخدام الأقلام وفرش الرسم والألوان

يمكنك استخدام كائنات القلم والفرشاة لرسم الخطوط والأشكال المختلفة. فالقلم هو أحد حالات التصنيف **Pen** ويستخدم لرسم الخطوط والأشكال المجوفة (أى الخالية من الألوان). بينما تعتبر الفرشاة إحدى الحالات المشتقة من التصنيف **Brush** وتستخدم في رسم الأشكال الملونة. وهناك أيضاً كائنات الألوان التي تشتق من التصنيف **Color** وتستخدم مع كلٍ من الأقلام وفرش الرسم لتوضيح اللون الذي يتم استخدامه في عملية الرسم.

### استخدام الأقلام Pens

يستخدم القلم Pen في رسم الخطوط والمنحنيات والأشكال المجوفة ويتم تعريفه كما يلي:

- يمكنك تعريف قلم أحمر اللون بسمك نقطة واحدة باستخدام العبارة التالية:

**Dim myPen As New Pen (Color.Black)**

- يمكنك تعيين سمك أكبر للقلم وليكن 5 نقاط كما يلي:

**Dim myPen As New Pen (Color.Black,5)**

- يمكنك استخدام لون إحدى فرش الرسم المعرفة من قبل كما يلي:

**Dim myPen As New Pen (myBrush,5)**

وبمجرد تعريف القلم، يمكنك استخدامه في رسم خط أو منحنى أو حتى شكل مجوف كما في المثال التالي الذى يقوم برسم قطع ناقص:

**Dim myPen As New Pen (Color.Black)**

**Dim g As Graphics = Me.CreateGraphics**

**g.DrawEllipse(myPen,20,30,10,6)**

وبمجرد تعريف القلم يمكنك التحكم في خصائصه المختلفة والتي تحدد طريقة رسمه للخطوط. فهناك الخصائص **Width** و **Color** التي تحدد مظهر الخط، والخصائص **StartCap** و **EndCap** التي تتيح لك إضافة أشكال مخصصة لبداية الخط ونهايته، والخاصية **DashStyle** التي تتيح لك الاختيار بين الأشكال المختلفة للخط هل هو مجسم **Solid** أو منقط **Dotted** أو مقطوع **Dashed** وغيرها من الخيارات الأخرى.

### حذف الأقلام

بعد الانتهاء من استخدام القلم، يكون من الأفضل حذفه لتوفير المساحة المستغلة بالذاكرة. لأداء ذلك، قم باستخدام الوظيفة **Dispose** كما يلي:

**myPen.Dispose()**

### استخدام فرش الرسم

فرش الرسم **Brushes** عبارة عن كائنات تستخدم مع كائن الرسوم **Graphics** لإنشاء الأشكال المجسمة (ذات اللون الداخلى). يوضح جدول ٧-٤ التالى الأنواع المختلفة من فرش الرسم وهى عبارة عن تصنيفات مشتقة أساساً من التصنيف الجرد **Brush**.

جدول ٧-٤ الأنواع المختلفة من فرش الرسم

الوصف	تصنيف الفرشاة
هذا النوع هو أبسط أنواع فرش الرسم ويستخدم في رسم اللون المصمت	<b>SolidBrush</b>
تشبه النوع السابق إلى حد كبير إلا أنها تتيح لك الاختيار من بين عدد من الأشكال دون التقييد باللون المصمت	<b>HatchBrush</b>
تقوم بالرسم باستخدام مادة داخلية كالصور مثلاً	<b>TextureBrush</b>
تقوم بالرسم بلونين مع استخدام تدرج الألوان من البداية إلى النهاية	<b>LinearGradientBrush</b>
تقوم بالرسم باستخدام مجموعة من الألوان المركبة متدرجة اللون وذلك من خلال مسار تقوم أنت بتحديدته	<b>PathGradientBrush</b>

وفيما يلي نوضح بعض الأمثلة على استخدام هذه الأنواع.

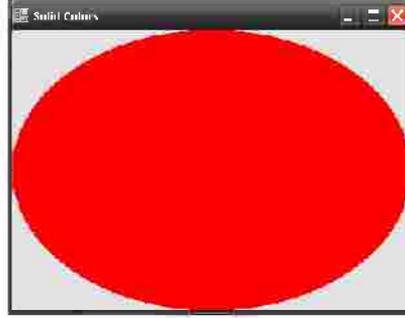
رسم الأشكال المجسمة

لرسم قطع ناقص مصمت أحمر اللون على النموذج، قم بإدخال الكود التالي داخل الحدث

**Form\_Paint**:

```
Dim myBrush As New SolidBrush(Color.Red)
Me.CreateGraphics.FillEllipse(myBrush, ClientRectangle)
```

حيث قمنا باستخدام **ClientRectangle** لتحديد مساحة الأداة المستخدمة بالكامل (النموذج في هذه الحالة)، لذا حينما تقوم بتنفيذ هذا الكود، يظهر لك قطع ناقص حمراء بحجم النموذج بالكامل (انظر شكل ٧-٦).



شكل ٧-٦ القطع الناقص بحجم النموذج.

يمتاز الحدث **Paint** بإعادة تنفيذ الكود الموجود بداخله كلما تم تغيير حجم النموذج أو تحديثه وهذا على عكس الحدث **Load** الذي يتم تنفيذه بمجرد تحميل النموذج داخل الذاكرة.



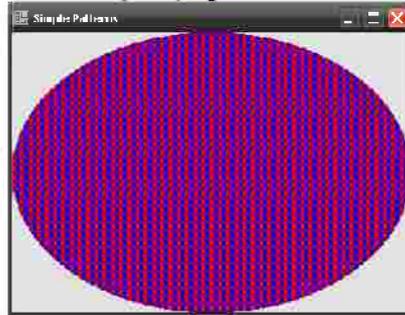
رسم الأشكال البسيطة

يمكنك استخدام الفرشاة **HatchBrush** في الرسم باستخدام عدد كبير من الأشكال وليس اللون المصمت كما في حالة الفرشاة السابقة. لرسم قطع ناقص نسيجي بمقدمة حمراء وخلفية زرقاء (انظر شكل ٧-٧)، قم بكتابة السطر التالي أعلى نافذة الكود:

```
Imports System.Drawing.Drawing2D
```

ثم قم بإدخال الكود التالي داخل الحدث **Form\_Paint**:

```
Dim myHatchBrush As New HatchBrush(HatchStyle.Plaid,  
Color.Red, Color.Blue)  
Me.CreateGraphics.FillEllipse(myHatchBrush, ClientRectangle)
```



شكل ٧-٧ رسم قطع ناقص نسيجي من خلال الفرشاة **HatchBrush**.

### رسم الأشكال المركبة

يمكنك رسم الأشكال المركبة التي تحتوي بداخلها على مادةٍ ما ولتكن صورة من الصور مثلاً باستخدام الفرشاة **TextureBrush**. لرسم القطع الناقص السابق باستخدام مادة داخلية عبارة عن صورة (انظر شكل ٧-٨)، قم بإدخال الكود التالي داخل الحدث **Paint** الخاص بالنموذج:

```
Dim myBrush As New TextureBrush(New Bitmap("c:\logo.bmp"))
Me.CreateGraphics.FillEllipse(myBrush, ClientRectangle)
```



شكل ٧-٨ الرسم باستخدام محتويات صورة.

### رسم الظلال المركبة

يمكنك استخدام الفرشاة **LinearGradientBrush** لرسم الظلال المركبة التي تتكون من لونين، بينما يمكنك استخدام الفرشاة **PathGradientBrush** لرسم الظلال الأكثر تعقيداً. لترى تأثير استخدام هذا النوع من فرش الرسم، قم بإدخال الكود التالي داخل الحدث **Paint** الخاص بالنموذج (انظر شكل ٧-٩):

```
Dim myBrush As New LinearGradientBrush(ClientRectangle,
Color.Red, Color.Yellow, LinearGradientMode.Vertical)
Me.CreateGraphics.FillEllipse(myBrush, ClientRectangle)
```



شكل ٧-٩ تدرج الألوان داخل القطر الناقص رأسياً من اللون الأحمر إلى اللون الأصفر.

حذف فرش الرسم

بعد الانتهاء من استخدام فرشاة الرسم، يكون من الأفضل حذف الفرشاة لتوفير المساحة المستغلة بالذاكرة. لأداء ذلك، قم باستخدام الوظيفة **Dispose()** كما يلي:

**myBrush.Dispose()**

استخدام الألوان

للألوان علاقة وطيدة بكل من الأقلام وفرش الرسم، فهي التي تحدد اللون المستخدم في رسم الخطوط أو الأشكال بالقلم كما أنها تحدد اللون الداخلي المستخدم مع الفرشاة. وهناك مجموعة من الألوان المعرفة مسبقاً داخل **Visual Basic 2008** والتي تزيد عن مائة لون مثل الألوان التالية:

```
Dim myColor As Color
myColor = Color.Red
myColor = Color.Aquamarine
myColor = Color.LightGoldenrodYellow
myColor = Color.PapayaWhip
myColor = Color.Tomato
```

فكل من العبارات السابقة تقوم بتخصيص لون معرف مسبقاً إلى اللون **myColor** الذي يمكنك استخدامه فيما بعد كلون جديد مع الأقلام وفرش الرسم.

### إنشاء ألوان مخصصة

فضلاً عن الألوان المعرفة داخل النظام، يمكنك تعريف ألوان مخصصة بنفسك باستخدام الوظيفة `Color.FromArgb()` والتي تتيح لك تحديد دقة الألوان الثلاثة الأساسية الأحمر Red والأخضر Green والأزرق Blue كما في الكود التالي:

```
Dim myColor As Color  
myColor = Color.FromArgb(23,56,78)
```

وهذا الكود يقوم بتعريف لون جديد رمادي مائل إلى الزرقة. وكل رقم من الأرقام الثلاثة يكون من 0 إلى 255 حيث يعبر الرقم 0 عن غياب اللون تماماً بينما يعبر الرقم 255 عن اكتماله، لذا فإن `Color.FromArgb(0,0,0)` تعنى اللون الأسود، بينما `Color.FromArgb(255,255,255)` تعنى اللون الأبيض.

يمكنك أيضاً استخدام نفس الوظيفة `FromArgb()` لتعيين شفافية اللون أو ما نطلق عليه "ألفا" Alpha وذلك بإدخال رقم من 0 إلى 255 كعامل أول قبل المعاملات الثلاثة السابقة كما يلي:

```
Dim myColor As Color  
myColor = Color.FromArgb(127,23,56,78)
```

وهذا الكود يقوم بتعريف لون رمادي مائل للزرقة وله درجة شفافية بنسبة ٦٠% . يمكنك بالطبع إضافة درجة الشفافية إلى لون موجود مسبقاً سواءً من الألوان المعرفة داخل .NET. أو التي قمت بتعريفها بنفسك كما في الكود التالي:

```
Dim myColor As Color  
myColor = Color.FromArgb(127,Color.Red)
```

### رسم الخطوط والأشكال

بعد أن تعرفنا على مفهوم الأقلام وفرش الرسم والألوان، دعنا نتعرف على كيفية رسم الخطوط والأشكال، حيث يحتوى التصنيف `Graphics` على الوظائف المستخدمة لرسم المجموعات المختلفة من الخطوط والأشكال سواءً البسيطة أو المركبة التي تتكون من ألوان مجسمة أو شفافة. كما عرفنا منذ قليل يمكنك إنشاء الخطوط والمنحنيات المفتوحة

والأشكال باستخدام التصنيف **Pen**، كما يمكنك ملء الأشكال كالمستطيلات والمضلعات باستخدام التصنيفات المشتقة من التصنيف **Brush**.

### الخطوات العامة لرسم الخطوط والأشكال المجوفة

لرسم خط أو شكل مجوف (أى لا يحتوى على ألوان داخلية)، قم بإجراء الخطوات العامة التالية:

١. قم بإنشاء مرجع للتصنيف (الكائن) **Graphics** الذى ستقوم باستخدامه فى عملية الرسم (وليكن زر **Button1** مثلاً) كما فى الكود التالى:

```
Dim g As Graphics = Button1.CreateGraphics
```

٢. قم بإنشاء حالة جديدة من التصنيف **Pen** لتحديد مواصفات القلم الذى ترغب فى استخدامه فى عملية الرسم كما فى الكود التالى الذى يقوم بتعريف قلم رسم باسم **myPen** يحتوى على اللون الأحمر وعرضه ٥ نقاط:

```
Dim myPen As New Pen(Color.Red,5)
```

٣. قم باستدعاء الوظيفة المناسبة للشكل الذى ترغب فى رسمه مستعيناً بجدول ٧-٥ وباستخدام كود مشابه للكود التالى:

جدول ٧-٥ وظائف رسم الخطوط والأشكال المجوفة

الشكل الناتج	الوظيفة
تقوم هذه الوظيفة برسم خط وتحتاج إلى تعيين إحداثيات نقطة البدء ونقطة الانتهاء وكذلك القلم المستخدم فى عملية الرسم.	<b>Graphics.DrawLine()</b>
تقوم هذه الوظيفة برسم الأشكال المعقدة وربما تحتاج إلى مصفوفة من الإحداثيات.	<b>Graphics.DrawPolygon()</b>
تقوم هذه الوظيفة برسم مستطيل وتحتاج إلى كائن أو أكثر كمعاملات.	<b>Graphics.DrawRectangle()</b>

```
g.DrawLine(myPen, 5, 5, 45, 65)
g.DrawBezier(myPen, 15, 15, 30, 30, 45, 30, 87, 20)
g.DrawEllipse(myPen, New Rectangle(33, 45, 40, 6))
g.DrawPolygon(myPen, New PointF() {New PointF(1, 1), New
PointF(20, 10), New PointF(5, 4), New PointF(100, 2), New
PointF(200, 6), New PointF(39, 45)})
```

### الخطوات العامة لرسم الأشكال الملونة

لرسم شكل ملون (أى يحتوى على ألوان داخلية)، قم بإجراء الخطوات العامة التالية:

١. قم بإنشاء مرجع للتصنيف (الكائن) **Graphics** الذى ستقوم باستخدامه فى عملية الرسم (وليكن زر **Button1** مثلاً) كما فى الكود التالى:

```
Dim g As Graphics = Button1.CreateGraphics
```

٢. قم بإنشاء حالة جديدة من تصنيف الفرشاة التى ترغب فى استخدامها لتحديد مواصفاتها (راجع جدول ٧-٤) كما فى الكود التالى الذى يقوم بتعريف فرشاة مصممة باسم **myBrush** تحتوى على اللون الأحمر:

```
Dim myBrush As New SolidBrush(Color.Red)
```

٣. قم باستدعاء الوظيفة المناسبة للشكل الذى ترغب فى رسمه مستعيناً بجدول ٧-٦ وباستخدام كود مشابه للكود التالى:

جدول ٧-٦ وظائف رسم الأشكال الملونة

الشكل الناتج	الوظيفة
تقوم هذه الوظيفة برسم الأشكال المعقدة وربما تحتاج إلى مصفوفة من الإحداثيات.	<b>Graphics.FillPolygon()</b>
تقوم هذه الوظيفة برسم مستطيل وتحتاج إلى كائن أو أكثر كمعاملات.	<b>Graphics.FillRectangle()</b>
تقوم هذه الوظيفة برسم مخطط بياني.	<b>Graphics.FillPie()</b>
تقوم هذه الوظيفة برسم مسار محدد.	<b>Graphics.FillPath()</b>

```
g.FillRectangle(myBrush, New Rectangle(33, 45, 40, 6))
```

```
g.FillPie(myBrush, New Rectangle(33, 45, 40, 6),0,90)  
g.FillPolygon(myBrush, New PointF() {New PointF(1, 1), New  
PointF(20, 10), New PointF(5, 4), New PointF(100, 2), New  
PointF(200, 6), New PointF(39, 45)})
```

