

الفصل التاسع عشر العمل مع الخطوط

أصبحت النصوص المقروءة من العمليات الأساسية في حياتنا اليومية، سواءً كان هذا النص في كتاب أو جريدة أو حتى على شاشات الكمبيوتر أو التلفاز. لذا كان الاهتمام كبيراً بخطوط الكتابة. ومن حسن الطالع أن لغة **Visual C++** تحتوي على العديد من الدوال التي تتيح لك التحكم في الخط المعروض على الشاشة أو المكتوب في التقارير بسهولة تامة، وهو ما سنتناوله بالتفصيل في هذا الفصل من هذا الكتاب، فكن معنا من فضلك. بانتهاء هذا الفصل ستتعرف على:

- ◆ إظهار النصوص بأشكال متعددة من الألوان والأنماط
- ◆ استخدام الخطوط لتحسين تطبيقاتك

دوال كتابة النصوص

يوجد داخل لغة Visual C++ العديد من الدوال المستخدمة لكتابة النصوص، إلا أن أشهرها على الإطلاق الدالة `TextOut()` وهى أسرع وأسهل طريقة لكتابة النصوص، كما أنها دالة عضو فى بيئة العنصر وتحتوى على ثلاثة معاملات، الأول الإحداثى الأفقى `X_Coordinate` والثانى الإحداثى الرأسى `Y_Coordinate`، أما الثالث فيحتوى على النص المراد إظهاره.

للتعرف على كيفية عمل الدالة `TextOut()`، تابع معنا الخطوات الآتية:

١. استخدم معالج التطبيقات لإنشاء تطبيق أحادى الوثيقة باسم `SimpText` مع قبول الإعدادات الافتراضية للمعالج.
٢. نشط تبويب عرض التصنيفات `Class View` من نافذة عمل المشروع إذا لم يكن هو التبويب النشط.
٣. انقر التصنيف `CSimpTextView` بالجزء العلوى من التبويب لاختياره.
٤. انقر الدالة `OnDraw()` نقراً مزدوجاً من الجزء السفلى من التبويب، تظهر الدالة داخل نافذة المحرر.
٥. قم بتعديل كود الدالة كما يلى:

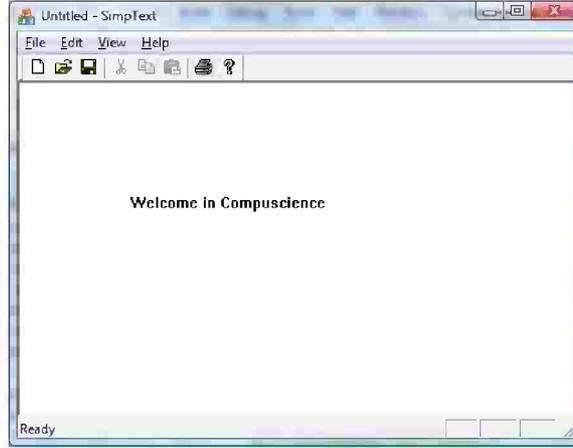
```

1. void CSimpTextView::OnDraw(CDC* pDC)
2. {
3.     CSimpTextDoc* pDoc = GetDocument();
4.     ASSERT_VALID(pDoc);
5.     // TODO: add draw code for native data here
6.     CString strText = "Welcome in Compucience";
7.     pDC->TextOut(100,100,strText);
8. }
    
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٦، تم تخزين النص المراد عرضه داخل المتغير النصى `strText`.
- فى السطر رقم ٧، تم استدعاء الدالة `TextOut()` لإظهار النص السابق. حيث تحتوى هذه الدالة على ثلاث معاملات، يحتوى الأول على الإحداثى الأفقى لنقطة

بدء الكتابة، بينما يحتوى الثانى على الإحداثى الرأسى لها، أما المعامل الثالث فيحتوى على النص المراد إظهاره.
قم ببناء التطبيق وتنفيذه، تلاحظ ظهور النص بالخط والألوان الافتراضية (خط أسود على خلفية بيضاء) قريباً من الركن الأيسر العلوى للنافذة (انظر شكل ١٩-١).



شكل ١٩-١ كتابة النص بالإعدادات الافتراضية

- يوجد العديد من الدوال المشابهة للدالة `TextOut()` والتي يمكن بيانها كما يلى:
- الدالة `TabbedTextOut()` ويمكنها إظهار مجموعة من النصوص المخزنة داخل مصفوفة بطريقة مجدولة.
 - الدالة `PolyTextOut()` ويمكنها إظهار مجموعة من النصوص باستدعائها مرة واحدة.
 - الدالة `ExtTextOut()` ويمكنك من خلالها التحكم فى طريقة عرض النص.

محاذاة النصوص

يمكنك التحكم فى محاذاة النص حول نقطة معينة باستخدام الدالة `SetTextOut()` التى تحتوى على العديد من الخيارات لتحديد المحاذاة الافتراضية للنص وكيفية وضع مؤشر الكتابة بعد إظهار النص. يحتوى جدول ١٩-١ على الخيارات التى يمكنك استخدامها مع الدالة `SetTextAlign()`.

جدول ١٩-١ خيارات محاذاة النصوص

الوصف	القيمة
محاذاة النص يمين نقطة البدء	TA_LEFT
محاذاة النص يسار نقطة البدء	TA_RIGHT
توسيط النص فوق نقطة البدء	TA_CENTER
محاذاة النص أسفل نقطة البدء	TA_TOP
محاذاة النص أعلى نقطة البدء	TA_BOTTOM
تتم محاذاة النص بحيث يقع الخط الأساسي فوق نقطة البدء	TA_BASELINE
يتم تحديث مؤشر الكتابة بعد كل استدعاء للدالة <code>TextOut()</code>	TA_UPDATECP
لا يتم تحديث مؤشر الكتابة بعد كل استدعاء للدالة <code>TextOut()</code>	TA_NOUPDATECP

تكون المحاذاة تبعاً للنقطة التي ستتم كتابة النص عندها. فمثلاً **TA-RIGHT** تعني محاذاة نقطة بدء الكتابة يمين النص المكتوب وليس العكس.



للتعرف على كيفية عمل الدالة `SetTextAlign()`، قم بتعديل كود الدالة `OnDraw()` كما يلي:

```

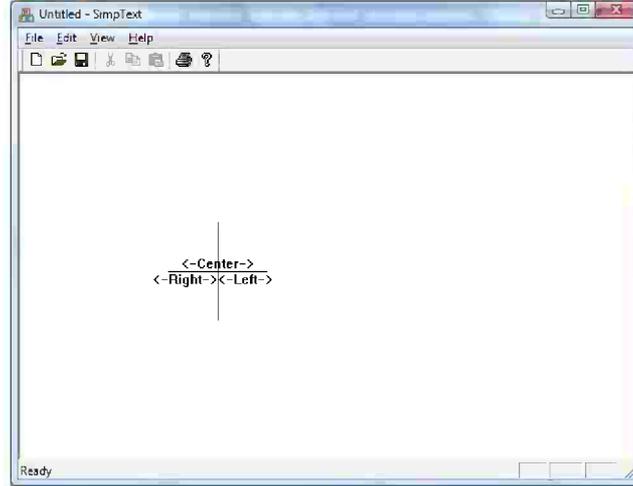
1. void CSimpTextView::OnDraw(CDC* pDC)
2. {
3.     CSimpTextDoc* pDoc = GetDocument();
4.     ASSERT_VALID(pDoc);
5.     // TODO: add draw code for native data here
6.     pDC->SetTextAlign(TA_RIGHT);
7.     pDC->TextOut(200,200,"<-Right->");
8.     pDC->SetTextAlign(TA_LEFT);
9.     pDC->TextOut(200,200,"<-Left->");
10.    pDC->SetTextAlign(TA_CENTER + TA_BOTTOM);
11.    pDC->TextOut(200,200,"<-Center->");
12.    pDC->MoveTo(150,200);
13.    pDC->LineTo(250,200);

```

```
14. pDC->MoveTo(200,150);
15. pDC->LineTo(200,250);
16. }
```

وعن هذا الكود، نوضح ما يلي:

- في السطر رقم ٦ تم استدعاء الدالة `SetTextAlign()` بتمرير المعامل `TA_RIGHT` وذلك لحاذاة النص يسار نقطة البدء التي سيتم رسمها عند الإحداثي `(200,200)`.
 - في السطر رقم ٧، تم استدعاء الدالة `TextOut()` لكتابة كلمة `<-Right->` يسار نقطة البدء.
 - في السطر رقم ٨، تم استدعاء الدالة `SetTextAlign()` مرةً أخرى بتمرير المعامل `TA_LEFT` وذلك لحاذاة النص يمين نقطة البدء.
 - في السطر رقم ٩، تم استدعاء الدالة `TextOut()` لكتابة كلمة `<-Left->` يمين نقطة البدء.
 - في السطر رقم ١٠، تم استدعاء الدالة `SetTextAlign()` بتمرير المعاملين `TA_CENTER` و `TA_BOTTOM` وذلك لحاذاة النص منتصف وأعلى نقطة البدء.
 - في السطر رقم ١١، تم استدعاء الدالة `TextOut()` لكتابة كلمة `<-Center->` منتصف وأعلى نقطة البدء.
 - في السطور من ١٢ إلى ١٥، تم استدعاء الدالتين `MoveTo()` و `lineTo()` لرسم العلامة + عند نقطة البدء.
- قم ببناء التطبيق وتنفيذه، تحصل على نتائج مشابهة لتلك الموجودة بشكل ١٩-٢.



شكل ١٩-٢ محاذاة النصوص باستخدام الدالة `TextOut()`

يمكنك استخدام الدالة `GetTextAlign()` للحصول على الإعدادات الحالية لمحاذاة النصوص وذلك باستخدام نفس معاملات الدالة `SetTextAlign()`.



تغيير ألوان النص والخلفية

اللون الافتراضي للنص هو اللون الأسود على خلفية بيضاء، فإذا أردت تغيير هذه الألوان، قم باستخدام الدالة `SetTextColor()` لتغيير لون النص أو الدالة `SetBkColor()` لتغيير لون الخلفية، حيث يحتوي كل من الدالتين على معامل واحد فقط من النوع `COLORREF`. كما يمكنك أيضاً الحصول على ألوان النص الحالية باستخدام الدالتين `GetBkColor()` و `GetTextColor()`. للتعرف على كيفية تغيير ألوان النصوص، قم بتعديل كود الدالة `OnDraw()` كما يلي:

```

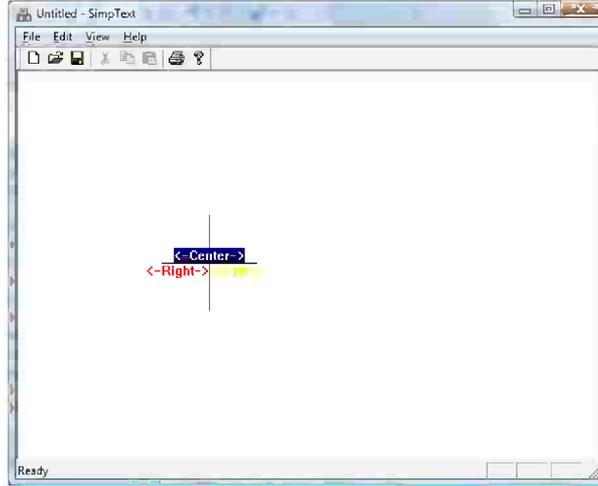
1. void CSimpTextView::OnDraw(CDC* pDC)
2. {
3.     CSimpTextDoc* pDoc = GetDocument();
4.     ASSERT_VALID(pDoc);
5.     // TODO: add draw code for native data here
6.     pDC->SetTextColor(RGB(0,0,255));
7.     pDC->SetTextAlign(TA_RIGHT);
8.     pDC->SetTextColor(RGB(255,0,0));

```

```
9. pDC->TextOut(200,200,"<-Right->");
10. pDC->SetTextAlign(TA_LEFT);
11. pDC->SetTextColor(RGB(255,255,0));
12. pDC->TextOut(200,200,"<-Left->");
13. pDC->SetTextAlign(TA_CENTER + TA_BOTTOM);
14. pDC->SetBkColor(RGB(0,0,128));
15. pDC->TextOut(200,200,"<-Center->");
16. pDC->MoveTo(150,200);
17. pDC->LineTo(250,200);
18. pDC->MoveTo(200,150);
19. pDC->LineTo(200,250);
20. }
```

وعن هذا الكود، نوضح ما يلي:

- في السطر رقم ٦، تم استدعاء الدالة `SetTextColor()` بتمرير المعامل `RGB(0,0,255)` والذي يعني تغيير لون النص من اللون الأسود الافتراضى إلى اللون الأزرق.
 - في السطر رقم ٨، تم استدعاء الدالة `SetTextColor()` بتمرير المعامل `RGB(255,0,0)` والذي يعني تغيير لون النص من اللون الأزرق إلى اللون الأحمر.
 - في السطر رقم ١١، تم استدعاء الدالة `SetTextColor()` بتمرير المعامل `RGB(255,255,0)` والذي يعني تغيير لون النص من اللون الأحمر إلى اللون الأصفر.
 - في السطر رقم ١٤، تم استدعاء الدالة `SetBkColor()` بتمرير المعامل `RGB(0,0,128)` والذي يعني تغيير خلفية النص من اللون الأبيض إلى اللون الأزرق الداكن.
- قم ببناء التطبيق وتنفيذه، تحصل على نفس الشكل السابق ولكن مع تغيير ألوان النص والخلفية (انظر شكل ١٩-٣).



شكل ١٩-٣ تغيير لون النص وخلفيته

إظهار النص المعتم والشفاف

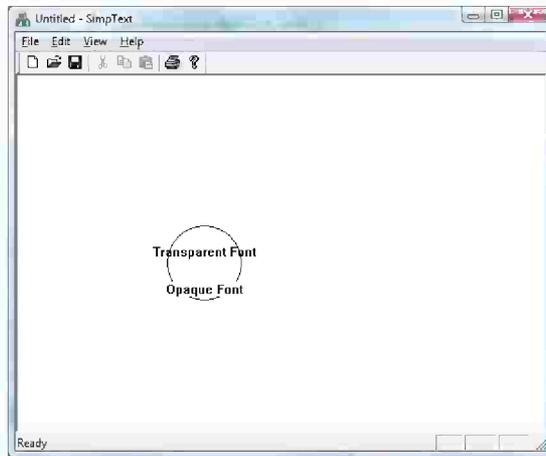
نريد أحياناً كتابة النص المعتم **Opaque** لتغطية الخلفية، كما نريد في أحيان أخرى مزج النص مع الخلفية ليكون شفافاً **Transparent**. تتيح لك الدالة **SetBkMode()** التنقل بين هذه الخيارات، حيث تحتوي هذه الدالة على معامل واحد فقط هو **OPAQUE** للنص المعتم (القيمة الافتراضية) أو **TRANSPARENT** للنص الشفاف. لاحظ أنك إذا اخترت النمط **OPAQUE**، سيتم استخدام كل من لون النص والخلفية لإظهار النص، أما إذا اخترت النمط **TRANSPARENT**، فسيتم استخدام لون النص فقط لإظهاره. للتعرف على كيفية تغيير أنماط عرض النص باستخدام الدالة **SetBkMode()**، قم بتعديل كود الدالة **OnDraw()** كما يلي:

1. `void CSimpTextView::OnDraw(CDC* pDC)`
2. `{`
3. `CSimpTextDoc* pDoc = GetDocument();`
4. `ASSERT_VALID(pDoc);`
5. `// TODO: add draw code for native data here`
6. `pDC->Ellipse(160,160,240,240);`
7. `pDC->SetTextAlign(TA_CENTER);`
8. `pDC->SetBkMode(TRANSPARENT);`
9. `pDC->TextOut(200,180,"Transparent Font");`

```
10. pDC->SetBkMode(OPAQUE);
11. pDC->TextOut(200,220,"Opaque Font");
12. }
```

وعن هذا الكود، نوضح ما يلي:

- في السطر رقم ٦ تم استدعاء الدالة **Ellipse()** لرسم دائرة بالأبعاد المحددة.
 - في السطر رقم ٧ تم استدعاء الدالة **SetTextAlign()** بتمرير المعامل **TA_CENTER** لتوسيط النص فوق نقطة البدء.
 - في السطر رقم ٨ تم استدعاء الدالة **SetBkMode()** بتمرير المعامل **TRANSPARENT**.
 - في السطر رقم ٩ تم استدعاء الدالة **TextOut()** لكتابة النص الشفاف.
 - في السطر رقم ١٠ تم استدعاء الدالة **SetBkMode()** بتمرير المعامل **OPAQUE**.
 - في السطر رقم ١١ تم استدعاء الدالة **TextOut()** لكتابة النص المعتم.
- قم ببناء التطبيق وتنفيذه، تلاحظ ظهور حافة الدائرة تحت عبارة **Transparent Font** لأنه نص شفاف وعدم ظهور حافة الدائرة تحت عبارة **Opaque Font** لأنه نص معتم (انظر شكل ١٩-٤).



شكل ١٩-٤ النص المعتم والنص الشفاف

وضع النص داخل مستطيلات

تحتاج أحياناً إلى وضع النص داخل مستطيل بحيث لا يخرج هذا النص عن حدود المستطيل حتى وإن اختلفت بعض حروفه. ولعل رسم المخططات خير مثال لذلك. لوضع نص معين داخل مستطيل، سنقوم باستخدام الدالة `ExtTextOut()` وهى إصدار مميز من الدالة الأم `TextOut()`، حيث تحتوى على ستة معاملات وذلك كما يلي:

- المعامل الأول والثاني عبارة عن الإحداثي الأفقي والرأسي للنقطة التي من عندها ستبدأ كتابة النص تماماً كما بالدالة `TextOut()`.
- المعامل الثالث يحتوى على زوج من المعرفات `Flags`. الأول هو `ETO_CLIPPED` وفي حالة استخدامه يتم التقاط النص داخل المستطيل. والثاني هو `ETO_OPAQUE` ويستخدم في حالة تطبيق النمط `OPAQUE` على النص حتى ولو كان النمط الحالى هو `Transparent`. يمكنك تغيير القيمة 0 إذا لم ترد استخدام أي من المعرفين.
- المعامل الرابع عبارة عن عنصر من النوع `CRect` ويستخدم لاحتواء النص في حالة استخدام المعامل `ETO_CLIPPED` السابق.
- المعامل الخامس هو النص المراد كتابته.
- المعامل السادس هو الذى يحتوى على مصفوفة من الحروف أو القيمة `NULL`.

للتعرف على طريقة عمل الدالة `ExtTextOut()`، قم بتعديل كود الدالة `OnDraw()` وذلك كما يلي:

```

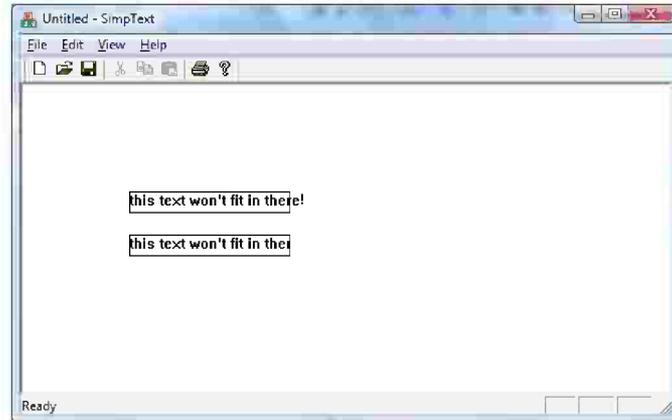
1. void CSimpTextView::OnDraw(CDC* pDC)
2. {
3.     CSimpTextDoc* pDoc = GetDocument();
4.     ASSERT_VALID(pDoc);
5.     // TODO: add draw code for native data here
6.     CRect rcClipBox(CPoint(100,100),CPoint(250,120));
7.     pDC->Rectangle(rcClipBox);
8.     pDC->SetBkMode(TRANSPARENT);
9.     pDC->ExtTextOut(100,100,0,rcClipBox,"this text won't fit
                                     in there!",NULL);

```

```
10. rcClipBox.OffsetRect(0,40);
11. pDC->Rectangle(rcClipBox);
12. pDC->ExtTextOut(100,140,ETO_CLIPPED,rcClipBox,"this
    text won't fit in there!",NULL);
13. }
```

وعن هذا الكود، نوضح ما يلي:

- في السطر رقم ٦ تم تعريف عنصر من النوع **CRect** لاستخدامه في رسم مستطيل فيما بعد.
 - في السطر رقم ٧ تم استدعاء الدالة **Rectangle()** لرسم المستطيل السابق تعريفه.
 - في السطر رقم ٩، تم استدعاء الدالة **ExtTextOut()** بدون المعامل **ETO_CLIPPED** لذلك سيظهر جزء من النص خارج حدود المستطيل.
 - في السطر رقم ١٢، تم استدعاء الدالة **ExtTextOut()** مرة أخرى باستخدام المعامل **ETO_CLIPPED** لذلك سيختفي الجزء الخارج عن حدود المستطيل.
- قم ببناء التطبيق وتنفيذه، تلاحظ ظهور مستطيلين أحدهما يخرج منه النص والآخر يقوم باحتواء النص داخله مع قطع الجزء الزائد (انظر شكل ١٩-٥).



شكل ١٩-٥ كتابة النص داخل المستطيلات

إنشاء الخطوط المتعددة

قمنا فيما سبق بإظهار الخطوط بإعدادات الخط الافتراضية الموجودة بيئة العنصر. وتتشابه الخطوط مع أقلام الرسم وفرش الألوان بتمثيلها كعناصر GDI ويمكن اختيارها داخل أو خارج بيئة العنصر. يمكنك إنشاء الخطوط وتشكيلها كما تريد باستخدام مجموعة الدوال الموجودة بنظام التشغيل Windows ومكتبة MFC كما يلي.

استخدام التصنيف CFont

يعتبر التصنيف CFont الوعاء الأساسي المستخدم عند العمل مع الخطوط، حيث يحتوى على العديد من الدوال المستخدمة لإنشاء الخطوط المختلفة.

إنشاء الخطوط باستخدام الدالة CreatePointFont()

يمكنك استخدام الدالة CreatePointFont() في إنشاء الخطوط وهي أسهل الطرق، حيث تحتوى الدالة على ثلاثة معاملات. يحتوى المعامل الأول على حجم الخط مضروباً في القيمة ١٠، بينما يحتوى المعامل الثاني على اسم الخط، أما المعامل الثالث فيحتوى على بيئة العنصر التي سيتم استخدام الخط فيها. للتعرف على طريقة استخدام الدالة CreatePointFont() في إنشاء الخطوط، قم بتعديل كود الدالة OnDraw() كما يلي:

```

1. void CSimpTextView::OnDraw(CDC* pDC)
2. {
3.     CSimpTextDoc* pDoc = GetDocument();
4.     ASSERT_VALID(pDoc);
5.     CFont fnBig;
6.     fnBig.CreatePointFont(360, "Arial", pDC);
7.     CFont* pOldFont = pDC->SelectObject(&fnBig);
8.     pDC->TextOut(50,50,"** 36 pt Arial Font **");
9.     pDC->SelectObject(pOldFont);
10. }

```

وعن هذا الكود، نوضح ما يلي:

- في السطر رقم ٥ تم إنشاء عنصر جديد ينتمى للتصنيف CFont.

- في السطر رقم ٦ تم استدعاء الدالة `CreatePointFont()` بتمرير القيمة 360 للمعامل الأول إنشاء خط حجمه ٣٦ نقطة.
- في السطر رقم ٧ تم استدعاء الدالة `SelectObject()` لاختيار الخط الجديد وتخزين عنوان الخط القديم داخل المؤشر `pOldFont`.
- في السطر رقم ٨ تم استدعاء الدالة `TextOut()` لكتابة النص بالخط الجديد بدءاً من إحداثي النقطة (50,50).

إنشاء الخطوط باستخدام الدالة `CreateFont()`

تعتبر الدالة `CreateFont()` من الدوال الفريدة المميزة داخل مكتبة MFC نظراً لاحتوائها على أربعة عشر معاملاً، يحتوى معظمها على توليفة من القيم المعقدة. يمكننا بيان هذه المعاملات والشكل العام للدالة كما يلي:

```
BOOL CreateFont(int nHeight, int nWidth, int nEscapement,
int nOrientation , int nWeight, BYTE bItalic, BYTE
bUnderline, BYTE cStrikeOut, BYTE nCharSet, BYTE
nOutPrecision, BYTE nClipPrecision, BYTE nQuality, BYTE
nPitchAndFamily, LPCTSTR lpszFacename);
```

- يحتوى المعامل الأول والثاني على ارتفاع الخط المطلوب وعرضه على الترتيب. قد يحتوى المعامل الأول `nHeight` على قيمة موجبة وحينئذٍ يتم مطابقة هذا الارتفاع مع ارتفاع الخلية لاختيار أنسب خط من قائمة الخطوط المتاحة، ومن الممكن أيضاً أن يحتوى على قيمة سالبة وحينئذٍ يتم مطابقة هذا الارتفاع مع ارتفاع الخط. والفرق بين الخلية والحرف هو أن الخلية تحتوى على بعض الفراغات أعلى وأسفل الحرف، أما الحرف فله نفس ارتفاع الخلية مع تجاهل هذه الفراغات.

عند مطابقة ارتفاعات الخطوط، يتم اختيار أكبر خط أصغر من الارتفاع الذى قمت بتحديدده.



- يحتوى المعامل الثالث `nEscapement` على زاوية النص بعيداً عن المحور الأفقى مضروبةً في القيمة ١٠.

- يحتوي المعامل الرابع **nOrientation** على زاوية الحروف بعيداً عن المحور الأفقى مضروبةً في القيمة ١٠.
- يستخدم المعامل الخامس **nWeight** لتعيين سمك الخط من 0 إلى 1000 باختصار إحدى القيم المعرفة بالجدول ١٩-٢ التالى.

جدول ١٩-٢ قيم المعامل **nWeight**

القيمة	المعرف
0	FW_DONTCARE
100	FW_THIN
200	FW_EXTRALIGHT
300	FW_LIGHT
400	FW_NORMAL
500	FW_MEDIUM
600	FW_SIMIBOLD
700	FW_BOLD
800	FW_EXTRABOLD
900	FW_HEAVY

- يستخدم المعامل السادس **bItalic** لكتابة خط مائل إذا كانت قيمته **TRUE**.
- يستخدم المعامل السابع **bUnderline** لوضع خط أسفل الكتابة إذا كانت قيمته **TRUE**.
- يستخدم المعامل الثامن **bStrikeOut** لوضع خط منتصف الكتابة إذا كانت قيمته **TRUE**.
- يستخدم المعامل التاسع **nCharSet** لتعيين مجموعة الحروف المناسبة سواءً كانت **ANSI_CHARSET** للحروف العادية أو **SYMBOL_CHARSET** للرموز.
- يستخدم المعامل العاشر **nOutPrecision** لتحديد استخدام خط معين دون الآخر. وعادةً تكون قيمة هذا المعامل **OUT_DEFAULT_PRECIS**، فإذا أردت استخدام خط من النوع **True Type** فتكون قيمته، **OUT_TT_PRECIS**.

- يستخدم المعامل الحادى عشر **nClipPrecision** لتحديد دقة قص الخط، وغالباً ما تكون قيمته **CLIP_DEFAULT_PRECIS**.
- يستخدم المعامل الثانى عشر **nQuality** لتحديد العلاقة بين درجة نقاء ظهور الحروف والمعاملات الأخرى التى قمت بتعيينها. ويمكنك استخدام إحدى القيم **DEFAULT_QUALITY** و **DRAFT_QUALITY** أو **PROOF_QUALITY**.
- يستخدم المعامل الثالث عشر **nPitchAndFamily** لتحديد معاملين فى آنٍ واحد بفصلهما بالعلامة + أو | . يحتوى المعامل الأول على درجة الانحدار **Pitch** باختيار إحدى القيم الموضحة بالجدول ١٩-٣، أما المعامل الثانى فيحتوى على اسم عائلة الخط باختيار إحدى القيم الموضحة بالجدول ١٩-٤.

جدول ١٩-٣ قيم إعدادات **Pitch**

الوصف	القيمة
الخط على أى حالة	DEFAULT_PITCH
الخط ذو الانحدار المتغير فقط	VARIABLE_PITCH
الخط ذو الانحدار الثابت فقط	FIXED_PITCH

جدول ١٩-٤ قيم إعدادات عائلات الخط

الوصف	القيمة
الخطوط الجديدة	FF_DECORATIVE
أى عائلة	FF_DONTCARE
الخطوط ذات العرض الطفيف الثابت	FF_MODERN
خطوط مفصولة بعرض طفيف متغير	FF_ROMAN
تشبه خط الكتابة باليد	FF_SCRIPT

الوصف	القيمة
يحتوى على مسافات متناسبة	FF_SWISS
من النوع True Type	TMPF_TRUETYPE

نظراً لتعدد صور الخطوط True Type، يمكنك استخدام القيمة TMPF_TRUETYPE مع إحدى القيم الأخرى لاختيار نوع معين من هذه العائلة.



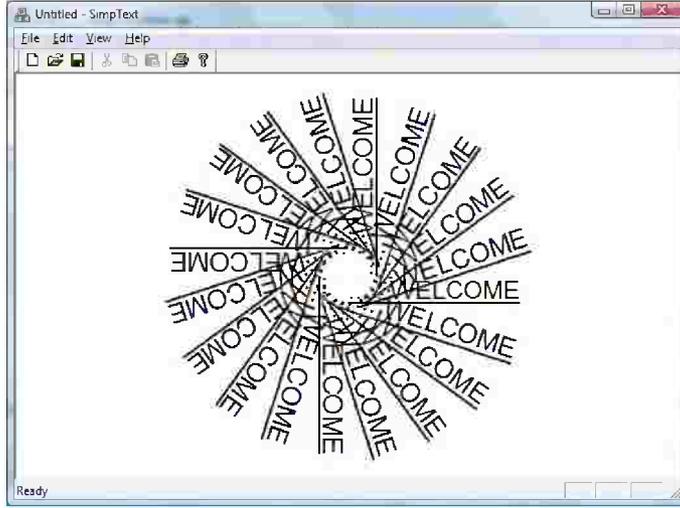
- يستخدم المعامل الرابع عشر والأخير IpszFacename لتحديد اسم خط معين من الخطوط المثبتة على جهازك مثل Times New Roman. للتعرف على طريقة استخدام الدالة CreateFont() لإنشاء الخطوط، قم بتعديل كود الدالة OnDraw() كما يلي:

```

1. void CSimpTextView::OnDraw(CDC* pDC)
2. {
3.     CSimpTextDoc* pDoc = GetDocument();
4.     ASSERT_VALID(pDoc);
5.     //TODO: add draw code for native data here
6.     CRect rcClient;
7.     GetClientRect(&rcClient);
8.     CPoint ptCenter = rcClient.CenterPoint();
9.     pDC->SetBkMode(TRANSPARENT);
10.    for(int i=0 ; i<360 ; i+=18)
11.    {
12.        CFont fnBig;
13.        fnBig.CreateFont(30,0,i*10,i*10,i/4,FALSE,TRUE,FALSE,
14.                        ANSI_CHARSET , OUT_DEFAULT_PRECIS ,
15.                        CLIP_DEFAULT_PRECIS ,PROOF_QUALITY,
16.                        DEFAULT_PITCH + FF_DONTCARE,"Arial");
17.        CFont* pOldFont = pDC->SelectObject(&fnBig);
18.        pDC->TextOut(ptCenter.x,ptCenter.y,"..... WELCOME");
19.        pDC->SelectObject(pOldFont);
20.    }
21. }

```

حيث تم استدعاء الدالة `CreateFont()` في السطر رقم ١٣ لإنشاء الخط بالموصفات الموضحة السابق شرحها، وبعد ذلك تم استدعاء الدالة `TextOut()` في السطر رقم ١٥ لكتابة النص بالخط السابق، وفي كل مرة يتم تغيير المعاملات الثالث والرابع والخامس، لذا يتم كتابة النص حول نقطة المنتصف (انظر شكل ١٩-٦).



شكل ١٩-٦ دوران النص حول المحور باستخدام الدالة `CreateFont()`

تعديد الخطوط واختيارها

تتيح لك العديد من التطبيقات العاملة تحت نظام التشغيل Windows اختيار الخط المناسب من مجموعة الخطوط المثبتة على جهازك. ولإضافة ذلك إلى تطبيقاتك، تحتاج إلى طريقة لاسترجاع هذه الخطوط. تتيح لك GDI هذه الإمكانية عن طريق مجموعة من الدوال والمربعات الحوارية كما سنرى.

استخدام الدالة `EnumFontFamilies()`

يمكنك استخدام الدالة `EnumFontFamilies()` والدالة المصاحبة `FontCallback()` للحصول على قائمة بالخطوط المتاحة في جهازك. فإذا أردت مثلاً الحصول على قائمة

بالخطوط المتاحة بنظام التشغيل وطباعة كل منها على الشاشة بإعداداته المختلفة، يتم ذلك على خطوتين:

الخطوة الأولى: إضافة الدالة المصاحبة `FontCallback()` (وهي دالة عامة لا تنتمي إلى أى تصنيف) قبل الدالة `OnDraw()` وذلك كما يلي:

```

1.  int CALLBACK FontCallback(ENUMLOGFONT FAR*
2.  lpelf,NEWTEXTMETRIC FAR* lpnt,intFontType,LPARAM
3.  lparam)
4.  {
5.      CDC* pDC = (CDC*)lparam;
6.      CFont fnEnum;
7.      fnEnum.CreateFontIndirect(&lpelf->elfLogFont);
8.      Font *pOldFont= pDC->SelectObject(pOldFont);
9.      int nYPos = pDC->GetCurrentPosition().y;
10.     pDC->TextOut(5,nYPos,CString(lpelf-> elfLogFont.
                                     IfFaceName));
11.     pDC->MoveTo(0,nYPos + lpelf->elfLogFont.IfHeight);
12.     pDC->SelectObject(pOldFont);
13.     return TRUE;
14. }

```

وعن هذا الكود، نوضح ما يلي:

- سيتم استدعاء هذه الدالة مع كل خط مثبت على جهازك بتمرير تفاصيل الخط كمعاملات للدالة.
- يحتوى المعامل الأول على المتغير `lpelf` الذى ينتمى للتركيب `ENUMLOGFONT` المعرف كما يلي:

```

typedef struct tagENUMLOGFONT
{
    LOGFONT elfLogFont;
    BCHAR   elfFullName[LF_FULLFACESIZE];
    BCHAR   elfStyle[LF_FACESIZE];
} ENUMLOGFONT;

```

حيث يحتوى التركيب على اسم الخط `elfFullName` ونمطه `elfStyle` بالإضافة إلى المتغير `elfLogFont` الذى ينتمى بدوره للتركيب `LOGFONT` الذى يحتوى

على جميع المعاملات التي يمكنك تمريرها للدالة **CreateFont()** لذا يتم استخدامه مع الدالة **CreateFontIndirect()** ويعرف كما يلي:

```
typedef struct tagLOGFONT
{
    LONG    lfHeight;
    LONG    lfWidth;
    LONG    lfEscapement;
    LONG    lfOrientation;
    LONG    lfWeight;
    BYTE    lfItalic;
    BYTE    lfUnderline;
    BYTE    lfStrikeOut;
    BYTE    lfCharSet;
    BYTE    lfOutPrecision;
    BYTE    lfClipPrecision;
    BYTE    lfQuality;
    BYTE    lfPitchAndFamily;
    TCHAR   lfFaceName[LF_FACESIZE];
} LOGFONT;
```

- يحتوى المعامل الثانى على المتغير **lpnt** الذى ينتمى للتركيب **NEWTEXTMETRICS** الذى يحتوى على تفاصيل أحجام الحروف وكيفية ظهور النص عند كتابته فى بيئة العنصر المناسبة.
- يحتوى المعامل الثالث على المتغير **FontType** المستخدم لتحديد نوع الخط هل هو **RASTER_FONTTYPE** أم **TRUETYPE_FONTTYPE** أم **DEVICE_FONTTYPE**.
- يحتوى المعامل الأخير **LParam** على مؤشر لبيئة العنصر التى سيتم استخدامها لرسم الخطوط المختلفة على الجهاز المصاحب (الشاشة فى هذه الحالة).
- فى السطر رقم ٧ تم استدعاء الدالة **CreateFontIndirect()** وهى شبيهة إلى حد كبير بالدالة **CreateFont()** إلا أنها تحتوى على تركيب به جميع المعاملات كما سنرى فيما بعد.
- فى السطر رقم ٩ تم استدعاء الدالة **GetCurrentPosition()** للحصول على

المكان الحالي للرسم.

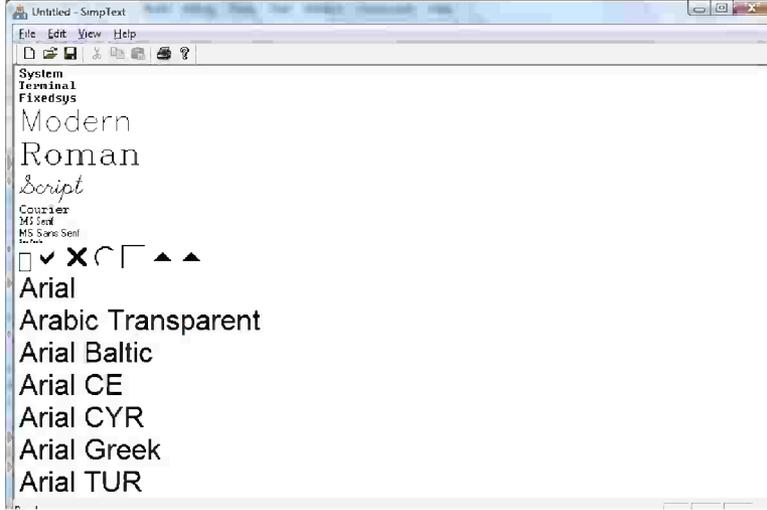
- في السطر رقم ١٠ تم استدعاء الدالة `TextOut()` بتمرير المعامل الثالث بالقيمة `IfFaceName` لإظهار اسم الخط.
- في السطر رقم ١١ يتم استدعاء الدالة `MoveTO()` بزيادة ارتفاع الخط عن المكان الحالي للكتابة تمهيداً لكتابة الخط الجديد.
- في السطر رقم ١٣ يتم إرجاع القيمة `TRUE` لإعادة استدعاء الدالة بعد ذلك ، حيث تعني القيمة `FALSE` عدم استدعاها مرةً أخرى.

الخطوة الثانية: تعديل كود الدالة `OnDraw()` لاستدعاء الدالة `EnumFontFamilies()` وذلك كما يلي:

```
void CSimpTextView::OnDraw(CDC* pDC)
{
    CSimpTextDoc* pDoc = GetDocument();
    ASSERT_VALID(pDoc);
    // TODO: add draw code for native data here
    EnumFontFamilies(pDC->GetSafeHdc(),NULL,
        (FONTENUMPROC) FontCallback,(LPARAM)pDC);
}
```

حيث تم استدعاء الدالة `EnumFontFamilies()` بتمرير أربعة معاملات. حيث يحتوي المعامل الأول على بيئة العنصر المستخدمة، بينما يحتوي المعامل الثاني على عائلة الخط المستخدم، وقد قمنا بتمرير القيمة `NULL` حتى يتم استخدام جميع عائلات الخطوط. كما يحتوي المعامل الثالث على اسم الدالة المصاحبة، بينما يحتوي المعامل الرابع والأخير على بيانات للمستخدم كبيئة العنصر مثلاً.

والآن قم ببناء التطبيق وتنفيذه، تلاحظ ظهور قائمة بأشكال الخطوط المثبتة على جهازك (انظر شكل ١٩-٧).

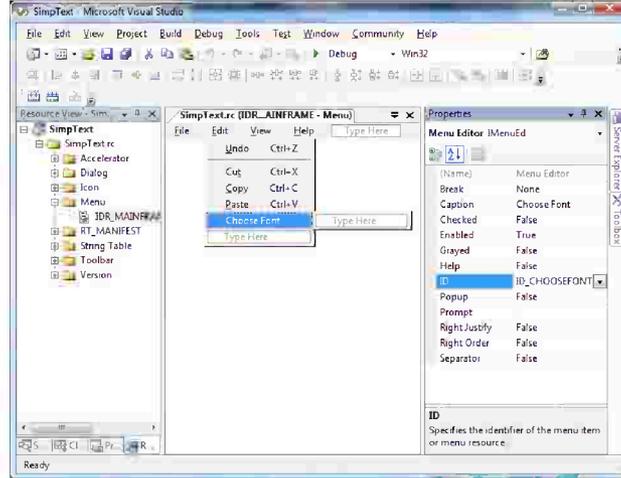


شكل ١٩-٧ قائمة الخطوط المثبتة بجهازك

استخدام المربع الحوارى Font

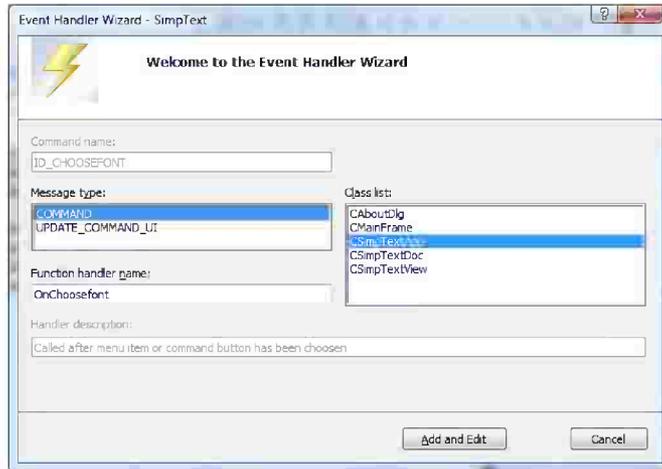
يحتوى نظام التشغيل على مربع حوارى قياسى Font يمكنك من خلاله اختيار الخطوط المثبتة فى جهازك وإعداداتها، ويمكنك استخدامه داخل تطبيقاتك وقتما شئت. لإضافة هذا المربع الحوارى إلى التطبيق الحالى، تابع معنا الخطوات الآتية:

١. قم بإضافة وسيلة لفتح المربع الحوارى ولتكن عنصر قائمة جديد باسم ID_CHOOSFONT وعنوان Choose Font إلى إحدى القوائم ولتكن قائمة Edit (انظر شكل ١٩-٨).



شكل ١٩-٨ إضافة عنصر القائمة لفتح المربع الحوارى

٢. قم بإضافة دالة احتواء رسالة عنصر القائمة الجديد (`OnChoosefont()`) باستخدام معالج الأحداث (انظر شكل ١٩-٩).



شكل ١٩-٩ إضافة دالة احتواء رسالة عنصر القائمة الجديد

٣. قم بتعديل كود الدالة (`OnChoosefont()`) كما يلي:
1. `void CSimpTextView::OnChoosefont()`
 2. `{`
 3. `// TODO: Add your command handler code here`

```
4. CFontDialog dlgChooseFont;
5. if(dlgChooseFont.DoModal() == IDOK)
6. {
7.     m_fnCustom.DeleteObject();
8.     m_fnCustom.CreateFontIndirect(dlgChooseFont.
9.     m_cf.lpLogFont);
10.     Invalidate();
11. }
```

وعن هذا الكود، نوضح ما يلي:

- في السطر رقم ٤ تم تعريف عنصر ينتمي لتصنيف المربع الحوارى `CFontDialog` هو `dlgChooseFont`.
- في السطر رقم ٥ يتم استدعاء الدالة `(DoModal)` لفتح المربع الحوارى.
- إذا قام المستخدم باختيار خط جديد ونقر زر `Ok`، يجب حذف الخط القديم باستخدام الدالة `(DeleteObject)` في السطر رقم ٧.
- في السطر رقم ٨ يتم استدعاء الدالة `(CreateFontIndirect)` لإنشاء الخط الجديد باستخدام عنصر ينتمى إلى `m_cf` ويحتوى على مؤشر للتركيب `.LOGFONT`.
- في السطر رقم ٩ يتم استدعاء الدالة `(Invalidate)` لإعادة طلاء النافذة واستدعاء الدالة `(OnDraw)`.
- ٤. قم بإضافة المتغير `m_fnCustom` إلى تصنيف العرض `CSimpTextView` كما يلي:

```
// Attributes
public:
```

```
    CFont m_fnCustom;// إضافة المتغير الجديد للتصنيف
```

```
    CSimpTextDoc* GetDocument();
```

```
    ٥. قم بتعديل كود الدالة (OnDraw) لكتابة رسالة بالخط الجديد كما يلي:
```

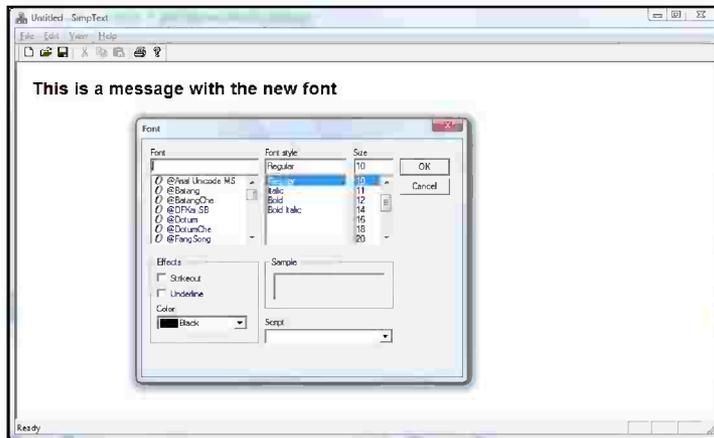
```
void CSimpTextView::OnDraw(CDC* pDC)
{
    CSimpTextDoc* pDoc = GetDocument();
```

```

ASSERT_VALID(pDoc);
// TODO: add draw code for native data here
CFont* pOldFont = pDC->SelectObject(&m_fnCustom);
pDC->TextOut(20,20,"This is a message with the new font ");
pDC->SelectObject(pOldFont);
}

```

والآن قم ببناء التطبيق وتنفيذه. اختر **Font Choose** من قائمة **Edit**، يظهر لك المربع الحوارى **Font**. قم باختيار خط جديد وغير من إعداداته ثم انقر زر **Ok**، تلاحظ ظهور الرسالة بالإعدادات الجديدة (انظر شكل ١٩-١٠).



شكل ١٩-١٠ فتح المربع الحوارى **Font** داخل التطبيق

تنسيق الخطوط

يمكنك استخدام الدالة **DrawText()** لكتابة النصوص وتنسيقها، حيث تحتوى هذه الدالة على ثلاثة معاملات. المعامل الأول متغير من النوع **CString** يحتوى على النص المراد كتابته، بينما يحتوى المعامل الثانى على إحداثيات المستطيل الذى ستتم فيه الكتابة، أما المعامل الثالث والأخير فيحتوى على مجموعة من المعرفات التى تحدد تنسيقات النص وكيفية ظهوره باختيار توليفة من القيم الموضحة بالجدول ١٩-٥ التالى.

جدول ١٩-٥ قيم الدالة DrawText()

الوصف	القيمة
محاذاة النص يساراً داخل المستطيل	DT_LEFT
محاذاة النص يميناً داخل المستطيل	DT_RIGHT
توسيط النص داخل المستطيل	DT_CENTER
توسيط النص رأسياً داخل المستطيل	DT_VCENTER
تقسيم النص إلى سطور	DT_WORDBREAK
عدم إظهار النص	DT_CALCRECT
توسيع حروف الجدولة إلى مسافات (افتراضياً ٨ مسافات)	DT_EXPANDTABS
تعطيل البادئة وإلا سيتم استخدام الحروف & لكتابة الاختصارات	DT_NOPREFIX
محاذاة النص أعلى المستطيل (مع النص ذو السطر الواحد فقط)	DT_TOP
محاذاة النص أسفل المستطيل (مع النص ذو السطر الواحد فقط)	DT_BOTTOM

قم بتعديل كود الدالة OnDraw() لاستخدام الدالة DrawText() بدلاً من الدالة TextOut() كما يلي:

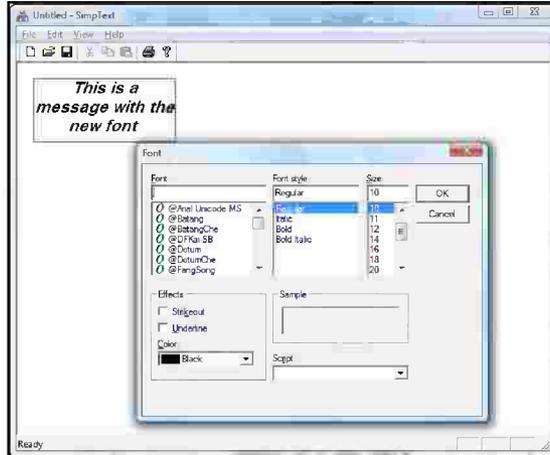
```

1. void CSimpTextView::OnDraw(CDC* pDC)
2. {
3.     CSimpTextDoc* pDoc = GetDocument();
4.     ASSERT_VALID(pDoc);
5.     // TODO: add draw code for native data here
6.     CFont* pOldFont = pDC->SelectObject(&m_fnCustom);
7.     CRect rcSmall(CPoint(20,20),CPoint(200,100));
8.     pDC->Rectangle(rcSmall);
9.     pDC->SetBkMode(TRANSPARENT);
10.    pDC->DrawText("This is a message with the new
11.    font ", rcSmall,DT_WORDBREAK + DT_CENTER);
12.    pDC->SelectObject(pOldFont);
12. }

```

وعن هذا الكود، نوضح ما يلي:

- في السطور ٧ و ٨ تم إنشاء المستطيل الذي سيتم وضع النص فيه واستخدامه من قبل الدالة **DrawText()**.
 - في السطر رقم ١٠ تم استدعاء الدالة **DrawText()** لإظهار الرسالة داخل المستطيل مع توسيط النص وتقسيمه إلى سطور.
- قم ببناء التطبيق وتنفيذه، تلاحظ ظهور النص داخل المستطيل بالتنسيقات السابقة (انظر شكل ١٩-١١).



شكل ١٩-١١ تنسيق النصوص باستخدام الدالة **DrawText()**

حذف الخطوط

مثل أقلام الرسم وفرش الألوان، يتم حذف الخطوط تلقائياً بمجرد انتهاء الدالة أو التصنيف الذي يستخدمه. إلا أنه يمكنك حذف الخط يدوياً إذا أردت عدم وجود جدوى لاستخدامه باستخدام الدالة **DeleteObject()** طالما كان الخط ما زال مختاراً داخل بيئة العنصر وخاصةً لأن الخطوط من عناصر **GDI** التي تحتل الكثير من ذاكرة الحاسب.



الباب السادس

مفاهيم متقدمة

- ٢٠ . استخدام أدوات ActiveX .
- ٢١ . تعقب الأخطاء وتصحيحها .
- ٢٢ . العمل مع الملفات .
- ٢٣ . تطبيق متكامل .