

الباب الحادي عشر
قضايا الانزلاق و سطر الأدوات
Scrolling and Toolbars

مفتتح

فى هذا الباب:

سوف نتعلم كيفية إضافة قضبان الانزلاق التى تمكننا من تحريك محتويات النافذة رأسياً وأفقياً عندما تزيد المحتويات عن حدود الشاشة.

كما سنعرف تفصيلات نظام الإحداثيات المستخدم فى النوافذ بنوعيه: الإحداثيات الطبيعية والإحداثيات المنطقية ، وكيفية التحويل بينهما.

سوف نتدرب أيضاً على إنشاء سطور الأدوات (Toolbar) والحالة (Status Bar) وإنشاء المقابض المسئولة عن استخدام أزرار الأدوات.



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.

(١١-١) برنامج الرسم – الطراز ٤

في هذا الباب سوف نستخدم نفس البرنامج الذي أنشأناه من قبل في الفصول السابقة MyPaintBrush لكي نمحّه إمكانية الانزلاق باستخدام قضبان الانزلاق (Scroll Bars).

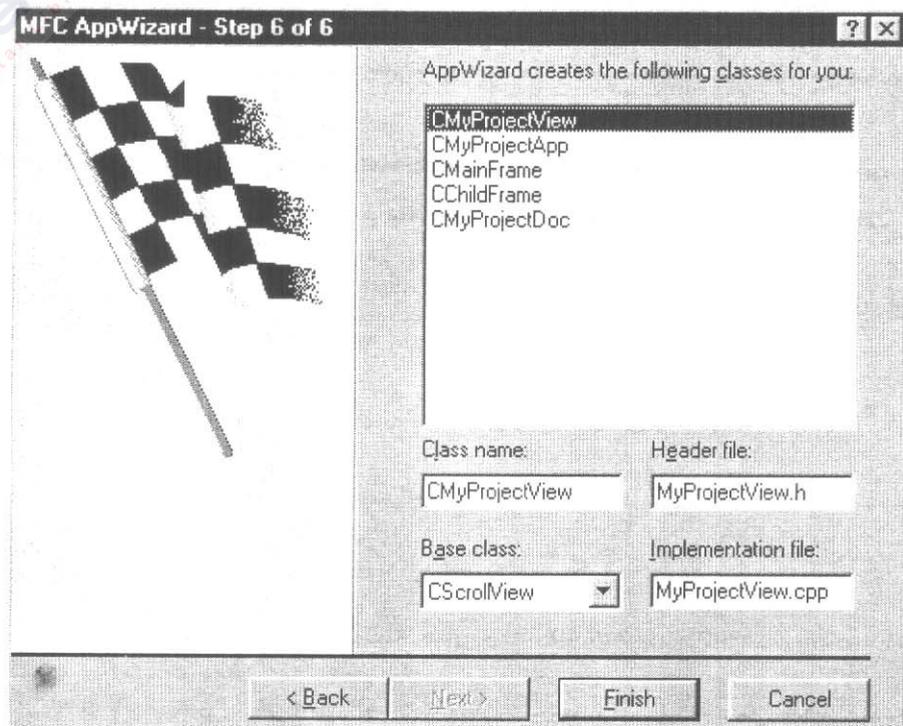
ولكي تبدأ العمل في هذا المشروع فإنك قد تستخدم أحد طريقتين:

- إما أن تستخدم نفس البرنامج الذي أنشأته في الباب العاشر وتكمل الرحلة من حيث انتهيت.

- أو تنسخ الدوسيه `\Examples\VCPP\Ch10\MyPaintBrush` الموجود على القرص المصاحب للكتاب ، إلى القرص الصلب ، وتبدأ منه العمل (ويمكنك تغيير اسمه إلى اسم جديد).

إضافة قضبان الانزلاق إلى مشروع جديد

إذا كان المشروع جديداً ، ففي إمكانك أن تمنحه خاصية قضبان الانزلاق أثناء إنشاء البرنامج باستخدام الشاشة السادسة من شاشات الساحر. وذلك باختيار الفصيلة `CScrollView` بدلا من الفصيلة `CView` كما بالشكل التالي ، حيث استخدمنا هذه الخطوة في بناء مشروع بالاسم `MyProject`. أما إضافة قضبان الانزلاق إلى مشروع موجود فهذا هو موضوعنا الأساسي في الفقرات التالية.



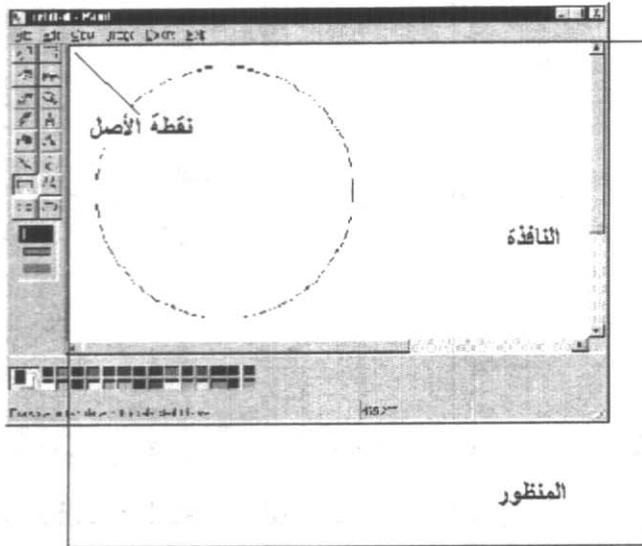
شكل (١٠-١) إضافة قضبان الانزلاق أثناء خلق المشروع

الانزلاق وعلاقته بنظام الإحداثيات

ذكرنا في الباب السابع الفرق بين الإحداثيات الطبيعية (التي قد تسمى أيضاً بإحداثيات الجهاز Device Coordinates) وبين الإحداثيات المنطقية. وموضوع الانزلاق يرتبط مباشرة بنظام الإحداثيات المستخدمة في النوافذ. ولنتصور أن نظام الإحداثيات المنطقية عبارة عن مستطيل له نقطة أصل ومحاور كالموضحة بالشكل التالي – سوف نستخدم عليه باسم المنظور (Viewport).

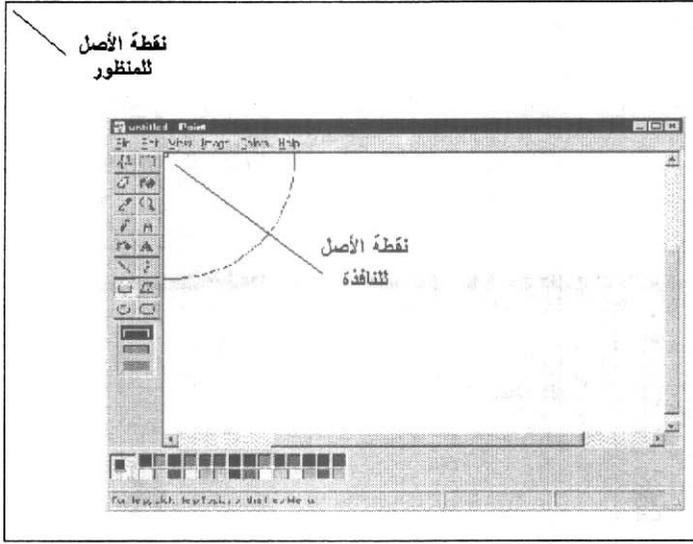
ولنفرض أنك فتحت برنامج الرسم Paint (المتضمن مع النوافذ) ثم رسمت به دائرة ما كما هو موضح بالشكل. في بداية الأمر تتطابق

كل من نقطتي الأصل للنافذة (الإحداثيات الطبيعية) والمنظور (الإحداثيات المنطقية) ، بحيث تكون نقطة الأصل المشتركة في الركن الأيسر العلوي لمنطقة العمل.



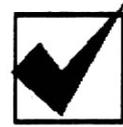
شكل (١٠-٢) الرسم على شاشة البرنامج في البداية

ولنفترض أننا حركنا المنزلق الرأسي إلى أسفل ، والمنزلق الأفقي إلى اليمين ، بحيث يصبح الرسم على شاشة البرنامج كما هو موضح بالشكل التالي. إن النتيجة التي نحصل عليها هي في الحقيقة تحريك النافذة بمحاورها بالنسبة لنقطة الأصل المنطقية. وبالرغم من أنك لا ترى إلا ربع الدائرة فقط ، ولكن بعد الدائرة عن نقطة الأصل المنطقية لم يتغير. فالدائرة موجودة في مكانها ولكن ما تراه على شاشة البرنامج هو جزء من المنظور.



شكل (٣-١١) تحريك المنظور باستخدام قضبان الانزلاق

وفي الحالة الأخيرة فإن إحداثيات مركز الدائرة بالنسبة للنافذة هي $(0,0)$ ، أما بالنسبة للمنظور فقد تكون رقما مثل $(95,100)$.
أما ما يحدث في منطق البرنامج الذي يصنع هذه الأحداث أن الدالة $OnDraw()$ المسؤولة عن الرسم وإعادة الطلاء لا تتغير ، فهي ترسم الدائرة في نفس المكان كل مرة ، منسوبا إلى المحاور المنطقية. أما عملية الانزلاق فتتولاها دوال المؤسسة MFC.



إن دوال الرسم مثل $MoveTo()$ و $LineTo()$ تستخدم الإحداثيات المنطقية ، أما ضغطات الفأر في الأماكن المختلفة على الشاشة أو النافذة فهي تستخدم الإحداثيات الطبيعية ، وهذا هو مصدر الحاجة إلى تحويل الإحداثيات.

(١١-٢) إضافة قضبان الانزلاق إلى برنامج الرسم

فى هذه الفقرة سوف نضيف قضبان الانزلاق إلى البرنامج الموجود لدينا ، وسوف نقسم العمل إلى خطوات محددة:

الخطوة الأولى: الوراثة من الفصيلة CScrollView

لتحقيق ذلك استبدل جميع تكرارات الكلمة CView بالكلمة CScrollView بكل من الملفات الآتية:

• ملف العناوين MyPaintBrushView.h

• ملف المصدر MyPaintBrushView.cpp

وتؤدى الوراثة من الفصيلة CScrollView إلى منح نافذة التطبيق خاصية قضبان الانزلاق مباشرة ، ومع ذلك فإن البرنامج يحتاج إلى إضافة بعض سطور الكود لتحويل الإحداثيات.

ملاحظة:

لاحظ أننا لا نحتاج إلى إضافة قضبان الانزلاق إلى ملف المحرر MyEditor وذلك لأنه يرث من الفصيلة CEditView التى تمنحه هذه الخصائص تلقائياً. وكما نرى بالشكل التالى أنه عند تحميل ملف محتو على عدد من السطور بما يزيد عن اتساع الشاشة فإن قضيب الانزلاق الرأسى يظهر تلقائياً ، كما أن خاصية الانتقال التلقائى إلى السطر التالى (Word Wrap) تغنى عن قضيب الانزلاق الأفقى.

```

MyE ditor.cpp - MyE ditor
File Edit Help
// CMyEditorApp

BEGIN_MESSAGE_MAP(CMyEditorApp, CWinApp)
    //{{AFX_MSG_MAP(CMyEditorApp)
    ON_COMMAND(ID_APP_ABOUT, OnAppAbout)
        // NOTE - the ClassWizard will add and remove mapping
macros here.
        // DO NOT EDIT what you see in these blocks of
generated code!
    //}}AFX_MSG_MAP
    // Standard file based document commands
    ON_COMMAND(ID_FILE_NEW, CWinApp::OnFileNew)
    ON_COMMAND(ID_FILE_OPEN, CWinApp::OnFileOpen)
    // Standard print setup command
    ON_COMMAND(ID_FILE_PRINT_SETUP,
CWinApp::OnFilePrintSetup)
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////
Ready NUM

```

شكل (٤-١١) خصائص الانزلاق لبرنامج المحرر

ملاحظة:

الخطوة الثانية: إعداد المنظور وتحويل إحداثي نقطة البداية

عند الضغط على زر الفأر الأيسر فإن معلومة الإحداثي يتم تسجيلها وتميريرها باستخدام البارامتر point إلى الدوال المختلفة. يلزم تحويل هذا الإحداثي من إحداثيات طبيعية منسوبة إلى النافذة إلى إحداثيات منطقية منسوبة إلى المنظور.

أضف الكود الموضح بالبنط الثقيل إلى الدالة OnLButtonDown() بالملف MyPaintBrushView.cpp:

```

// CMyPaintBrushView message handlers

void CMyPaintBrushView::OnLButtonDown(UINT nFlags, CPoint
point)
{

```

```
// TODO: Add your message handler code here and/or call default
// Init Variables:
// New code for coordinates conversion:           أضف السطور التالية:
CClientDC ClientDC(this);
OnPrepareDC(& ClientDC);
ClientDC.DPtoLP(&point);

//                                             نهاية الكود الجديد

m_NewPoint = point;
m_PrevPoint = point;
m_Moving =1;
SetCapture();
```

- فى هذا الكود ، تستخدم العبارة (CClientDC ClientDC(this) لخلق هدف منطقة عرض (Display Context) مرتبط بنافذة المشاهد. ويحقق المؤشر this هذا الارتباط.
 - أما العبارة التالية فهي تحتوى على الدالة:


```
CScrollView ::OnPrepareDC()
```

 ووظيفتها ضبط نقطة الأصل للمنظور ، تبعاً لموضع قضبان الانزلاق.
 - وتحتوى العبارة الثالثة على الدالة:


```
CDC::DPtoLP()
```

 وتستخدم فى التحويل من إحداثيات الجهاز (DP) إلى إحداثيات منطقية (LP) منسوبة إلى المنظور الذى تم إعداده فى الخطوة السابقة.
- ترجم البرنامج للتأكد من عدم وجود أخطاء ، ولكن لا تتوقع أن يعمل بنجاح حتى الآن.

الخطوة الثالثة: تحويل إحداثيات حركة الفأر

عندما تسحب الفأر أثناء الضغط على الزر ، فإنك ترسل معلومة الإحداثيات لنقط المسار. ومن اللازم تحويل هذه النقط أيضا إلى إحداثيات منطقية بنفس الطريقة. وفي سطور الكود التالي قد أضفنا نفس الدوال إلى الدالة OnMouseMove() بنفس الملف. وتلاحظ أن هدف منطقي العرض ClientDC كان موجودا بهذه الدالة من الأصل.

```
void CMyPaintBrushView::OnMouseMove(UINT nFlags, CPoint
point)
{
    // TODO: Add your message handler code here and/or call
default
// Add these lines:
if (m_Moving)
{
    CClientDC ClientDC(this);
// Code for Coordinate Conversion:          أضف هذه السطور
    OnPrepareDC(&ClientDC);
    ClientDC.DPtoLP(&point);
//                                          نهاية الكود الجديد
```

الخطوة الرابعة: تحويل إحداثي نهاية الخط

أما الإحداثي الأخير الذي يحتاج إلى تحويل فهو إحداثي نهاية الخط عندما تطلق الزر. أضف عبارات مماثلة بنفس الملف إلى الدالة OnLButtonUp() كما هو موضح بالشكل التالي:

```
void CMyPaintBrushView::OnLButtonUp(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call
default
// Add these lines:
```

```

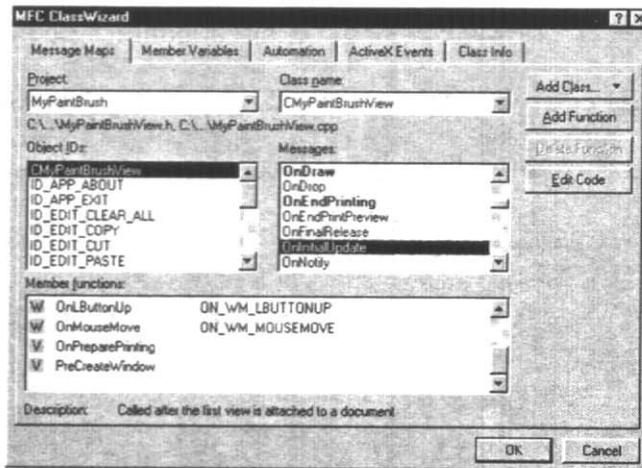
if (m_Moving)
{
    m_Moving = 0;
    ::ReleaseCapture();
    ::ClipCursor(NULL);
    CClientDC ClientDC(this);
// Code for Coordinate Conversion:           أضف هذه السطور:
    OnPrepareDC(&ClientDC);
    ClientDC.DPtoLP(&point);
//                                           نهاية الكود الجديد
    ClientDC.SetROP2(R2_NOT);

```

الخطوة الخامسة: تحديد حيز الرسم

عندما تسحب المنزلق الرأسى أو الأفقى إلى أقصى مداه ، فمن اللازم أن تتزامن هذه الحركة مع أبعاد المنظور بحيث تشاهد أقصى حدود الرسم. ولتحقيق ذلك يلزم تسجيل معلومة مساحة المنظور المطلوبة. لتحقيق ذلك اتبع الآتى:

- استخدم ساحر الفصائل (Class Wizard) لركوب الدالة `OnInitialUpdate()` كما بالشكل التالى.



شكل (١١-٥) إضافة الدالة `OnInitialUpdate()`

- اختر اسم المشروع ليكون MyPaintBrush ، و الفصيلة لتكون CMyPaintBrushView ، و رقم التعارف للهدف (Object ID) ليكون CMyPaintBrushView ، ثم اختر الدالة OnInitialUpdate() من صندوق الرسائل.
- اضغط الزر Add Function .
- اضغط الزر Edit Code لكي تنتقل إلى سطور الدالة الموضحة بالشكل التالي. أضف العبارات الموضحة بالبنط الثقيل.

```
void CMyPaintBrushView::OnInitialUpdate()
{
    CScrollView::OnInitialUpdate();

    // TODO: Add your specialized code here and/or call the base class
    أضف هذه السطور

    SIZE Size = {800, 600};
    SetScrollSizes(MM_TEXT, Size);
    نهاية الكود الجديد
}

```

وتستدعى الدالة OnInitialUpdate() عادة لتحديث المشهد قبل عرضه على الشاشة مباشرة ، سواء كان المشهد جديدا أم موجودا من الأصل. ويستخدم متغير المنشأ SIZE في تخزين أبعاد المنظور. وقد استخدمنا هنا الأرقام ٦٠٠ و ٨٠٠ ، ولكنها ليست ملزمة ، ويمكنك أن تختار أرقاما أخرى مثل ٦٤٠ و ٤٨٠.



فيما يلي إعلان المنشأ SIZE المستخدم في تحديد أبعاد المنظور:

```
typedef struct tagSIZE {  
    int cx;  
    int cy;  
} SIZE;
```

أما الدالة `CScrollView::SetScrollSizes()` فهي تستخدم لتمثيل منشأ الأبعاد علاوة على طريقة طور الخريطة (Mapping Mode) الذي يمكن تحديده بأحد الثوابت الموضحة بالجدول التالي ، وقد اخترنا منها الثابت `MM_TEXT` الذي يناظر نظام المحاور المعتاد (قطعة الأصل في الركن الأيسر العلوي) ووحدات القياس بالبكسل (Pixel).

اتجاه المحور الرأسي	وحدة القياس	طور الخريطة
إلى أسفل	1 pixel	MM_TEXT
إلى أعلى	0.01 mm	MM_HIMETRIC
إلى أعلى	1/1440 in	MM_TWIPS
إلى أعلى	0.001 in	MM_HIENGLISH
إلى أعلى	0.1 mm	MM_LOMETRIC
إلى أعلى	0.01 in	MM_LOENGLISH

جدول (١-١١) ثوابت طور الخريطة (Mapping Mode Constants)

جرب تشغيل البرنامج الآن وشاهد قضبان الانزلاق. غير أيضا مساحة المنظور إلى ٤٨٠ و ٦٤٠ ، ثم اضغط على زر تكبير النافذة وشاهد الفرق بين المنظورين.

(١١-٣) إضافة سطر الأدوات وسطر الحالة

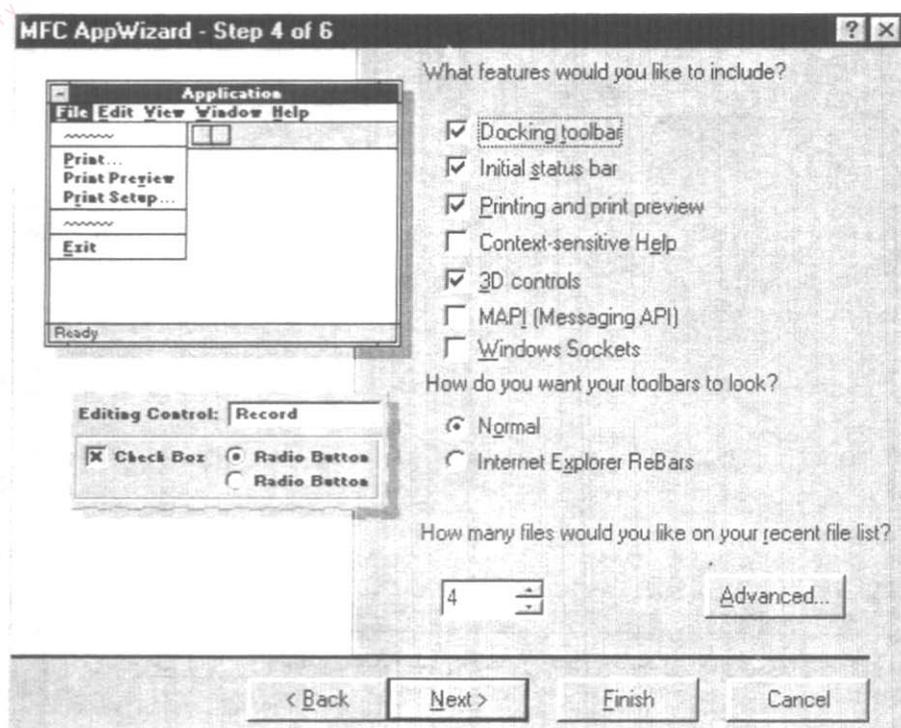
Toolbar-Status Bar

فى هذه الفقرة نتدرب على إنشاء سطر لأدوات سواء عند إنشاء برنامج جديد أو عند تطوير برنامج موجود أصلاً.

أولاً: سطر الأدوات فى برنامج جديد

عند تشغيل الساحر ، استمر فى الانتقال إلى الشاشات التالية حتى تصل إلى الشاشة الرابعة الموضحة بالشكل التالى. اختر من هذه الشاشة صناديق الاختيار التالية (إذا لم تكن هى الاختيارات سابقة التعريف):

- Docking Toolbar: سطر الأدوات العائم.
- Initial Status bar: سطر الحالة.



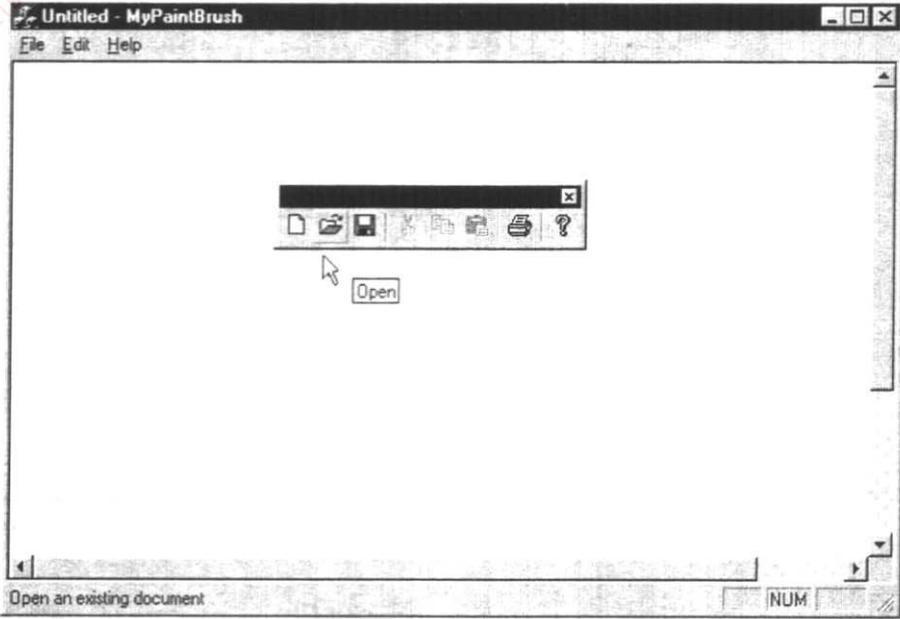
شكل (٦-١١) إنشاء سطر الأدوات و سطر الحالة في مشروع جديد

وبذلك فإن المشروع الجديد سوف يحتوى على كل من سطر الأدوات و سطر الحالة (سابقى التعريف) كما بالشكل التالى. وكما نرى بالشكل أنه عند اختيار أحد الأدوات (مثل أيقونة الملف الجديد "New" فى هذا المثال) فإن اسم الأيقونة يظهر فى صندوق خاص بجوارها ، كما أن سطر الحالة يصبح محتويا على شرح للاختيار الحالى.



شكل (٧-١١) سطر الحالة يبين الاختيار الحالي بسطر الأدوات

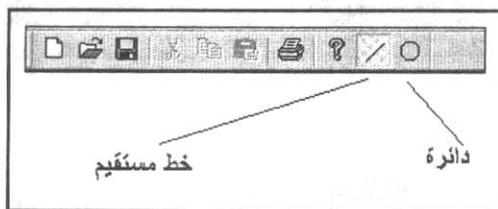
ولأن سطر الأدوات عائِم (Docked) فإنك تستطيع سحبه وتحريكه إلى أى موضع من النافذة. أنظر الشكل التالى.



شكل (١١-٨) سطر الأدوات بعد تحريكه من مكانه

ثانياً: إضافة سطر أدوات إلى برنامج موجود

سوف نستخدم برنامج الرسم MyPaintBrush فى إضافة سطر الأدوات ، وبالطبع فإننا فى هذه الحالة لن نكتفى بالأدوات سابقة التعريف (الموجودة أصلاً) بل سنضيف أيقونات جديدة ونربطها بالبرنامج. والنتيجة التى نرغب فى الوصول إليها موضحة بالشكل التالى ، وهى عبارة عن إضافة زررين من أزرار الأدوات واحد لرسم الخط المستقيم وآخر لرسم الدائرة. وبالطبع فإنك تستطيع أن تضيف أشكالاً هندسية أخرى مثل البيضاوى والمستطيل والمربع إلى آخره. وكننا سوف نكتفى بهذه الأزرار فى المثال الحالى.



شكل (٩-١١) إضافة أزرار أدوات

وهذه هي الخطوات:

الخطوة الأولى: إضافة أزرار سطر الأدوات

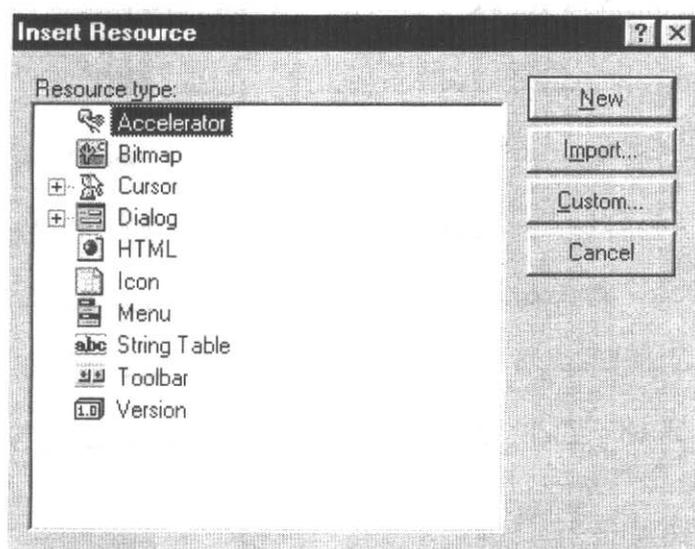
أولاً: إنشاء سطر جديد للأدوات

١. افتح المشروع MyPaintBrush.

٢. اختر أمر القائمة Insert Resource.

٣. من صندوق الحوار الذي يظهر ، اختر سطر الأدوات

(Toolbar). أنظر الشكل التالي.



شكل (١٠-١١) صندوق حوار الموارد

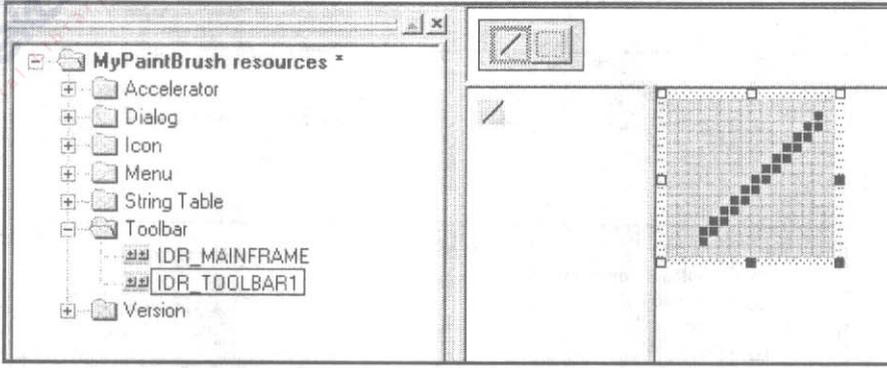
٤. يظهر في القسم الأيسر من الشاشة عنصر جديد في شجرة الموارد بالاسم IDR_TOOLBAR1. أنظر الشكل التالي.



شكل (١١-١٢) سطر الأدوات بمشهد الموارد

٥. يحتوى الشكل السابق على ثلاثة أقسام في النافذة اليمنى. واحد مخصص للرسم وهو يظهر الزر بالحجم الكبير. وآخر لمشاهدة شكل الزر بالحجم الطبيعي. أما القسم العلوى فيحتوى على جميع الأزرار التي نضيفها إلى سطر الأدوات ، ولكنه في الوقت الحالى لا يحتوى إلا على الزر الجديد المزمع إضافته. وبالطبع فإن هناك سطرًا لأدوات الرسم جاهز على استخدامه فى رسم أيقونة الزر ، وهو يظهر عائمًا فى بيئة الأستوديو.

٦. ارسم المنظر المرغوب إظهاره على الزر باستخدام المشهد المكبر. والشكل التالي يوضح رسم أيقونة الخط المستقيم. وتلاحظ أنه عندما تبدأ الرسم الجديد ، تظهر أيقونة جديدة فارغة على اليمين.



شكل (١١-١٣) رسم أيقونة الخط المستقيم

ثانياً: إضافة أزرار جديدة إلى سطر الأدوات سابق التعريف:

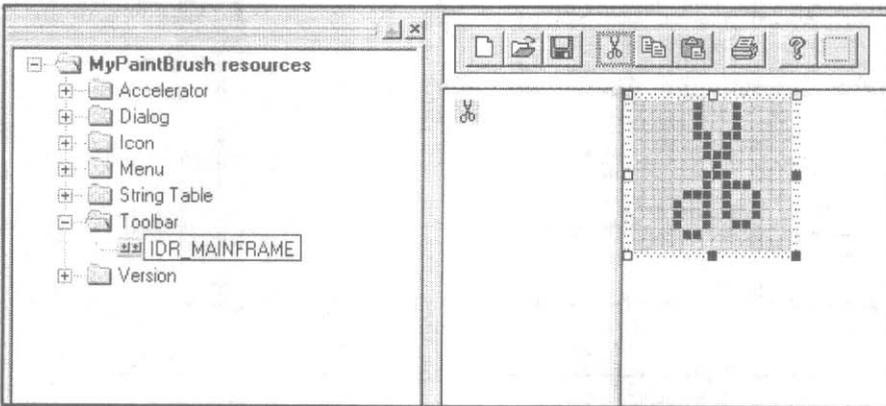
١. افتح المشروع MyPaintBrush.

٢. افتح مشهد الموارد ، واختر منه العنصر Toolbar ، ثم العنصر

IDR_MAINFRAME ، فيظهر على اليمين سطر الأدوات سابق

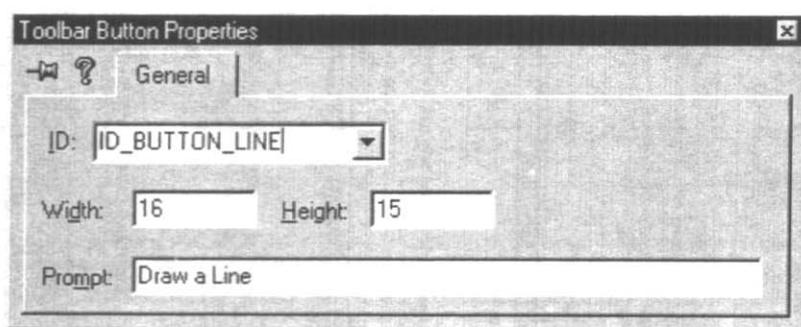
التعريف ونرى به مجموعة الأزرار المعروفة وعلى يمينها

أيقونة فارغة جاهزة على الرسم. أنظر الشكل التالي.



شكل (١١-١٤) إضافة أيقونة إلى سطر الأدوات سابق التعريف

٣. اختر الأيقونة الفارغة فيظهر المربع الفارغ في المشهد المكبر ،
وارسم أيقونة الخط المستقيم.
٤. اضغط ضغطة مزدوجة على أى نقطة خالية فى قسم الأيقونة
المكبرة ، فتظهر نافذة لخصائص الأيقونة الجارى رسمها. غير
رقم التعارف الموجود إلى ID_BUTTON_LINE. أنظر الشكل
التالى.



شكل (١١-١٥) نافذة خصائص زر من أزرار سطر الأدوات

٥. امنح وصفا ما لوظيفة كل زر باستخدام نافذة الخصائص كما
هو موضح بالشكل التالى. إن هذا الوصف (مثل Draw a Line)
يظهر فى سطر الحالة عند اختيار الزر.
٦. كرر العملية السابقة لإضافة زر الدائرة ، وامنحه الاسم
ID_BUTTON_CIRCLE ثم كرر العملية لإضافة أية أزرار جديدة
ترغب فى إضافتها مع منحها الاسم المناسب.

١. قم ببناء وتنفيذ البرنامج عند هذا الحد للتأكد من سلامة عمله ،
وسوف تشاهد سطر الأدوات وقد جدت عليه الأيقونات الجديدة ،
لكنها تظهر معتمة (غير عاملة).

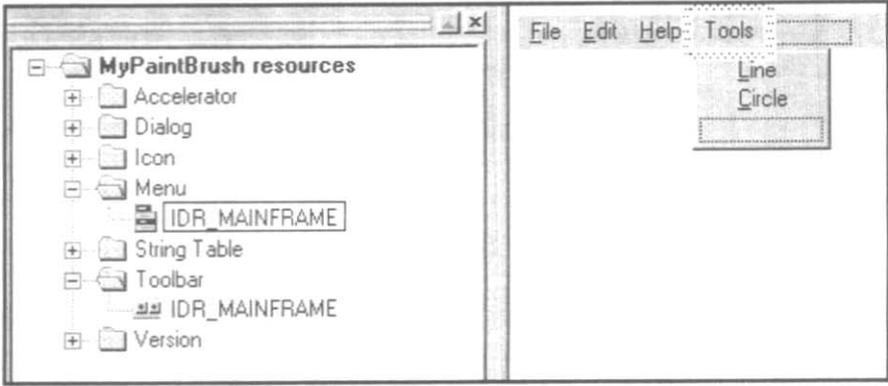
الخطوة الثانية: إضافة أوامر الرسم إلى القائمة

عادة تناظر أزرار سطر الأدوات أوامر مماثلة من أوامر القائمة ،
لأن سطر الأدوات ما هو إلا طريقة سريعة لتنفيذ المهام. فالأوامر
New و Open و Save كلها موجودة بالقائمة ، لكن وجودها بسطر
الأدوات يمنحك بديلا سريعا لتنفيذها. وفي هذه الفقرة سوف نضيف
أمرين من أوامر القائمة ، واحد لرسم الخط Line والآخر لرسم
الدائرة Circle.

٢. افتح مشهد الموارد (Resource View).

٣. اختر القائمة (Menu) ، ثم اختر منها العنصر IDR_MAINFRAME ،
فتظهر على اليمين نافذة تحرير القوائم.

٤. استخدم الطرق التي مررنا بها من قبل لإضافة أمر قائمة جديد
بالاسم Tools ، يتفرع منه الاختياران Line و Circle ، بحيث
يبدو المشهد كما في الشكل التالي.

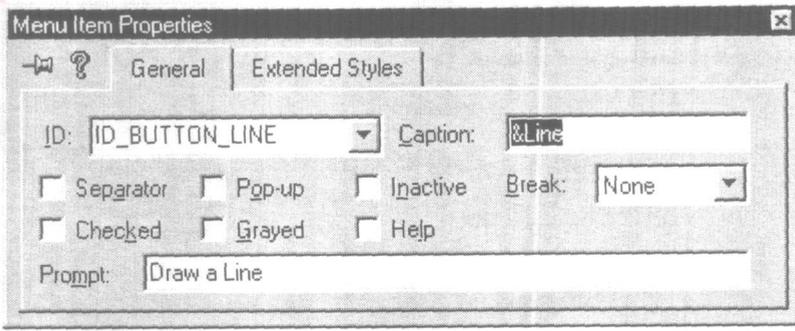


شكل (١١-١٦) المشهد النهائي لأوامر القائمة

٥. باستخدام نافذة الخصائص (التي تتوصل إليها بضغط مزدوجة على اختيار القائمة) لكل أمر من أوامر القائمة امنح الأوامر Line و Circle نفس رقم التعارف الذي منحناه للأزرار المناظرة أى:

رقم التعارف	الاختيار
ID_BUTTON_LINE	Line
ID_BUTTON_CIRCLE	Circle

٦. يمكنك أيضا أن تمنح أيضا وصفا ما لوظيفة كل زر باستخدام هذه النافذة كما هو موضح بالشكل التالي. إن هذا الوصف هو نفسه الوصف المرتبط بخصائص الزر ويجوز تعديله من أى من النافذتين.

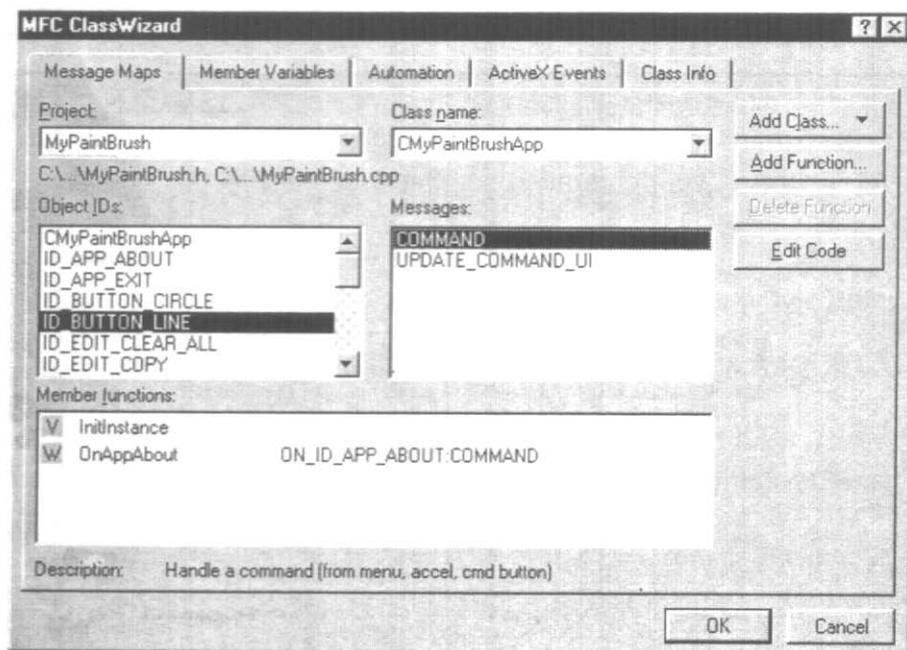


شكل (١١-١٧) نافذة خصائص أمر القائمة Line

٧. قم ببناء البرنامج وتنفيذه فتشاهد أوامر القائمة الجديدة Line و Circle وقد ظهرت بقائمة الأدوات Tools ، ولكنها تبدو معتممة (غير عاملة).

الخطوة الثالثة: إضافة المقابض

١. قم بتشغيل سحر الفصائل واختر من النافذة فصيحة التطبيق:
ID_BUTTON_LINE ، ثم اختر رقم التعارف CMyPaintBrushApp
من قسم أرقام التعارف ، كما بالشكل التالي. كما نرى بالشكل
أن هناك مقبضين لهذا الرقم هما: **COMMAND** و
UPDATE_COMMAND_UI. وهذه هي المقابض المعتادة لكل
هدف من الأهداف في بيئة النوافذ.
٢. أضف المقبضين باستخدام الأمر Add Function .
٣. كرر العمل بالنسبة لرقم تعارف الدائرة ID_BUTTON_CIRCLE .



شكل (١١-١٨) نافذة سحر الفصائل – إضافة المقابض

تؤدي إضافة هذه المقابض إلى توليد أربعة من الدوال الجديدة
بفصيلة التطبيق لكنها دوال فارغة من المحتويات. هذه الدوال
سوف تكون هي المسؤولة عن ربط الأزرار أو أوامر القائمة
بالأحداث المناظرة ، وذلك بعد أن نضيف إليها الكود المناسب ،
وهي:

• OnButtonLine()

• OnUpdateButtonLine()

• OnButtonCircle()

• OnUpdateButtonCircle()

وهذه الأوعية موضحة بالشكل التالي.

```
void CMyPaintBrushApp::OnButtonLine()           دالة زر الخط المستقيم
{
// TODO: Add your command handler code here
}

دالة تحديث زر الخط المستقيم
void CMyPaintBrushApp::OnUpdateButtonLine(CCcmdUI* pCmdUI)
{
// TODO: Add your command update UI handler code here
}

دالة زر الدائرة
void CMyPaintBrushApp::OnButtonCircle()
{
// TODO: Add your command handler code here
}

دالة تحديث زر الدائرة
void CMyPaintBrushApp::OnUpdateButtonCircle(CCcmdUI*
pCmdUI)
{
// TODO: Add your command update UI handler code here
}
```

الخطوة الرابعة: إضافة الكود إلى الدوال

قبل أن نضيف الكود إلى هذه المجموعة من أوعية الدوال ، فلنلق نظرة على ما هو موجود من الكود الذي كتبه لنا الساحر .

١. افتح ملف العناوين لفصيلة الإطار العام CMainFrame فتجد بها البيانات الأعضاء الخاصة بسطرى الحالة والأدوات ، وهى مشتقة من الفصيلتين CToolBar و CStatusBar ، وهى أعضاء جاهزة يضيفها الساحر من عنده:

```
m_wndStatusBar  
m_wndToolBar
```

وفى إمكانك أن تستخدم هذه الأعضاء أو أن تستبدلها بأعضاء جديدة من عندك. وهذا هو الكود الذى أنشأه لك الساحر:

```
class CMainFrame : public CFrameWnd  
{  
protected: // create from serialization only  
    CMainFrame();  
    DECLARE_DYNCREATE(CMainFrame)  
...  
protected: // control bar embedded members  
    CStatusBar m_wndStatusBar; // البيانات الأعضاء  
    CToolBar m_wndToolBar;
```

٢. افتح الملف MainForm.cpp ، فتجد به الدالة **OnCreate()** التى تتحكم فى خصائص سطر الأدوات ونصها كالاتى:

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)  
{
```

```
if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
    return -1;
```

خصائص سطر الأدوات:

```
if (!m_wndToolBar.CreateEx(this, TBSTYLE_FLAT,
    WS_CHILD | WS_VISIBLE | CBRS_TOP
    | CBRS_GRIPPER | CBRS_TOOLTIPS | CBRS_FLYBY |
    CBRS_SIZE_DYNAMIC) ||
    !m_wndToolBar.LoadToolBar(IDR_MAINFRAME))
{
    TRACE0("Failed to create toolbar\n");
    return -1;    // fail to create
}

```

```
if (!m_wndStatusBar.Create(this) ||
    !m_wndStatusBar.SetIndicators(indicators,
    sizeof(indicators)/sizeof(UINT)))
{
    TRACE0("Failed to create status bar\n");
    return -1;    // fail to create
}

```

احذف هذه السطور الثلاثة للحصول على سطر أدوات غير عائم (قياسي):

```
// TODO: Delete these three lines if you don't want the toolbar to
// be dockable
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndToolBar);

return 0;
}
```

٣. كما نرى في الكود السابق أن السطور الثلاثة المشار إليها (في المؤخرة) هي التي تؤدي إلى إنشاء سطر الأدوات العائم ، وبحذفها تحصل على سطر قياسي. كما أن خصائص السطر تحددها الدالة **CreateEx()** من خلال عبارة الشرطية تحتوى على الصفات المطلوبة مفصولة بالمؤثر المنطقى "|". وفى إمكانك أن تحذف بعض الصفات أو تضيف إليها. وهذا هو ملخصها:

الوظيفة	اسم الخاصية
لإنشاء زر مسطح - للاطلاع على الأنواع الأخرى اطلع على ثوابت أزرار سطر الحالة بشاشة النجدة.	TBSTYLE_FLAT
لجعل سطر الأدوات نافذة ابنة لنافذة الإطار العام.	WS_CHILD
لجعل السطر مرئياً.	WS_VISIBLE
لوضع السطر مبدئياً على قمة نافذة الإطار العام.	CBRS_TOP
لإنشاء مقبض (Gripper) على يسار السطر حتى يمكن سحبه منه.	CBRS_GRIPPER
لإظهار النص المحتوى على اسم الزر عند الإشارة إليه بالفأر.	CBRS_TOOLTIPS
لإظهار النص المحتوى على اسم الزر عند الضغط عليه بالفأر	CBRS_FLYBY
لتمكين المستخدم من تغيير شكل السطر وترتيب الأيقونات به.	CBRS_SIZE_DYNAMIC)

جدول (١١-٢) خصائص أزرار سطر الأدوات

٤. إضافة دالة بناء لأوامر الرسم: في هذه الفقرة نضيف الكود اللازم لشحن دالة التطبيق بالاختيار الحالي لزر سطر الأدوات. وهذا يتكون من جزئين:

- أولاً: إعلان عضواً جديداً يمثل متغير الزر وذلك بملف العناوين "MyPaintBrush.h". أنظر شريحة الكود التالية.

```
class CMyPaintBrushApp : public CWinApp
{
public:
    UINT m_CurrentDrawing;
    العضو الجديد
```

- ثانياً: تخصيص هذا المتغير لرقم تعارف الزر المستخدم حالياً ، وذلك بملف التطبيق "MyPaintBrushApp.cpp".

```
CMyPaintBrushApp::CMyPaintBrushApp()
{
// TODO: add construction code here,
// Place all significant initialization in InitInstance
// New code to initialize drawing:
    m_CurrentDrawing = ID_BUTTON_LINE;
    شحن العضو
}
```

٥. إضافة الكود اللازم لرسم الخط والدائرة: في شريحة الكود التالية نضيف الكود اللازم إلى الدوال الأربعة التي تؤدي إلى تشغيل الأزرار بسطر الأدوات وذلك باستخدام العضو الجديد m_CurrentDrawing وربطه برقم التعارف المعين للخط أو للدائرة. كما أن الدالة OnUpdateButtonLine() تؤدي إلى تحديث حركة الزر وإظهاره كزر مضغوط أو غير مضغوط. ولو أنك حذفته الدالة الأخيرة سوف تلاحظ عدم تغير شكل الزر عند الضغط عليه.

```

void CMyPaintBrushApp::OnButtonLine()
{
// TODO: Add your command handler code here
// New code for the Line button:
m_CurrentDrawing = ID_BUTTON_LINE;
}
void CMyPaintBrushApp::OnUpdateButtonLine(CCmdUI* pCmdUI)
{
// TODO: Add your command update UI handler code here
// New code for the Line button:
pCmdUI->SetCheck(m_CurrentDrawing ==
ID_BUTTON_LINE ? 1: 0);
}

void CMyPaintBrushApp::OnButtonCircle()
{
// TODO: Add your command handler code here
// New code for the Circle button:
m_CurrentDrawing = ID_BUTTON_CIRCLE;
}
void CMyPaintBrushApp::OnUpdateButtonCircle(CCmdUI*
pCmdUI)
{
// TODO: Add your command update UI handler code here
// New code for the Circle button:
pCmdUI->SetCheck(m_CurrentDrawing ==
ID_BUTTON_CIRCLE ? 1: 0);
}

```

لاحظ أن دالتي تحديث الزر لكل من الخط والدائرة ، تستخدمان الدالة **SetCheck()** لاختبار وضبط حالة الزر بحسب ما إذا كان مضغوطاً أم لا. وهذه الدالة تابعة للفصيلة **CCmdUI** ، وهي فصيلة مستقلة (ليست موروثه) وهي تستخدم أساساً مع المقبض

.ON_UPDATE_COMMAND_UI



إن المهمة التي أنجزناها في هذا البرنامج هي خلق أزرار سطر الأدوات التي تستجيب للضغط عليها وترتبط بمقايض معينة. ومع ذلك فإن هذا الطراز من برنامج الرسم لا يرسم إلا الخطوط المستقيمة. ولكي يؤدي زر رسم الدائرة عمله كاملاً فلا بد من إجراء نفس الخطوات التي اتبعناها مع الخط المستقيم بدءاً من إنشاء فصيلة للدائرة وتحديد كيفية رسمها ومسحها. وسوف نترك لك إكمال هذه المهمة كتدريب. ومع ذلك فإن دوال رسم الأشكال المختلفة سوف نعرضها في الأبواب القادمة.

الموجز

1. تعلمنا في هذا الباب كيفية إضافة قضبان الانزلاق التي تمنحك خاصية الانزلاق رأسياً وأفقياً عندما تزيد المحتويات عن حدود الشاشة.
2. وفي أثناء جولتنا بموضوع الانزلاق عرفنا خصائص نظام الإحداثيات المستخدم في النوافذ بنوعيه: الإحداثيات الطبيعية والإحداثيات المنطقية. وعرفنا كيفية التحويل بينهما.
3. عرفنا كيفية استخدام بيئة الموارد في إنشاء سطر أدوات جديد أو في إضافة سطر أدوات إلى السطر سابق التعريف الموجود بالعنصر IDR_MAINFRAME. كما عرفنا كيفية استخدام ساحر الفصائل في إنشاء المقايض المسئولة عن استخدام أزرار الأدوات ، وعرفنا كيفية إضافة الكود اللازم إلى هذه المقايض.

٤. عرفنا أيضاً أن أضرار سطر الحالة يمدنا بها الساحر تلقائياً وأنه لا ضرورة تدعو إلى إنشائها من جديد.
٥. استخدمنا في هذه الجولة الفصائل والدوال والثوابت الآتية:

المنشآت:

SIZE (لتخزين أبعاد المنظور)

الفصائل:

CScrollView
CClientDC
CDC
CToolBar
CScrollBar
CCmdUI
CWnd

الدوال:

CClientDC::OnPrepareDC()
CDC::DPtoLP
CView::OnInitialUpdate()
CScrollView::SetScrollSizes()
CToolbar::CreateEx()
CCmdUI::SetCheck()
CWnd::OnCreate()
CWnd::OnMouseMove()
CWnd::OnLButtonUp()