الباب السابع

الرسم فى النافذة

مفتتح

فى هذا الباب نصمم تطبيقاً نوافذياً كاملاً ، يمنحنا المهارات الآتية:

١. كيفية الاستجابة لرسائل الفأر.

٢. كيفية برمجة مقابض الرسائل بالاستعانة بساحر الفصائل (Class Wizard) الذى يعمل معك جنباً إلى جنب فى تصميم دوال المقابض.

٣. تأكيد مفاهيم منطقة العرض (Display Context) وما يلزمها من تجهيزات.

٤. الرسم على الشاشة باستخدام الفأر بالاستعانة بحركات الفأر الثلاثة: الضغط ــ السحب ــ الإطلاق.

كما نستخدم مجموعة كبيرة من الدوال والثوابت وأرقام التعارف التى تقربنا من بيئة العمل النوافذية.

تذكر:

١. للاطلاع على الصيغة التفصيلية لإحدى الدوال أو الفصائل ، ضع مؤشر الفأر فوقها ثم اضغط زر اللوحة F1.

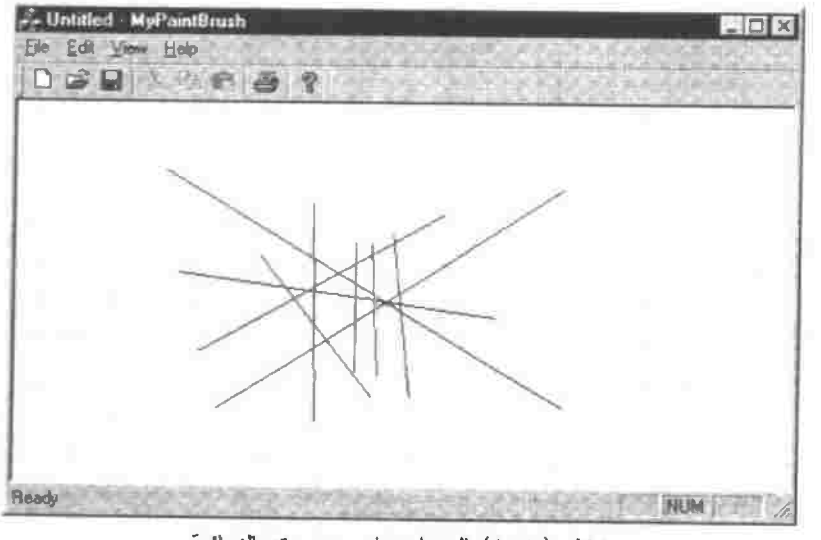
٢. لعرض الخريطة الكاملة لشجرة فصائل المؤسسة MFC ابحث فى شاشة النجدة (باستخدام البطاقة Search) عن "Hierarchy Chart".

٣. للاطلاع على الكود الكامل لأحد المشروعات ، افتح المشروع من القرص المصاحب للكتاب.

(٧-١) برنامج رسم ــ الطراز ١

سوف نبدأ رحلتنا مع برمجة النوافذ ببرنامج للرسم ، نبدأه من الصفر حتى نصل به إلى برنامج يشبه البرنامج النوافذى Paint الذى يأتى مع نظام التشغيل بدوسيه الإكسسوار. وهذه ملامح الطراز الأول من برنامج الرسم:

- عندما تقوم بتشغيل البرنامج فإنك تستطيع أن تسحب الفأر على النافذة ، مع الضغط على الزر ، فترسم خطوطا مستقيمة فى منطقة العميل كالموضحة بالشكل التالى.
- لا يتضمن هذا البرنامج إمكانية إعادة الطلاء ، بمعنى أنك لو أتلفت محتويات النافذة بأى طريقة من الطرق (مثل تصغير النافذة ، أو إخفائها خلف نافذة تطبيق آخر ، أو تغيير مساحتها) فإن المحتويات لا يعاد رسمها.



شكل (٧-١) البرنامج فى صورته النهائية

ملاحظة:

يوجد المشروع على القرص المصاحب بالملف:
.\ExamplesVCPP\Ch07\MyPaintBrush
وإذا كنت قد نسخت محتويات القرص على القرص الصلب فإنه سيوجد على القرص C بنفس الممر. ويمكنك مشاهدة أداء البرنامج بتشغيل الملف MyPaintBrush.exe مباشرة من الدوسيه الفرعى Debug لهذا المشروع. أما إذا أردت أن تبدأ المشروع خطوة بخطوة فلتضع مشروعك فى أى دوسيه آخر. ولكنه من الأفضل أن تستخدم الاسم MyPaintBrush ، لأن الأستوديو يستخدم اسم المشروع فى بناء الدوال.

ملاحظة:

(٧-٢) إنشاء هيكل المشروع MyPaintBrush

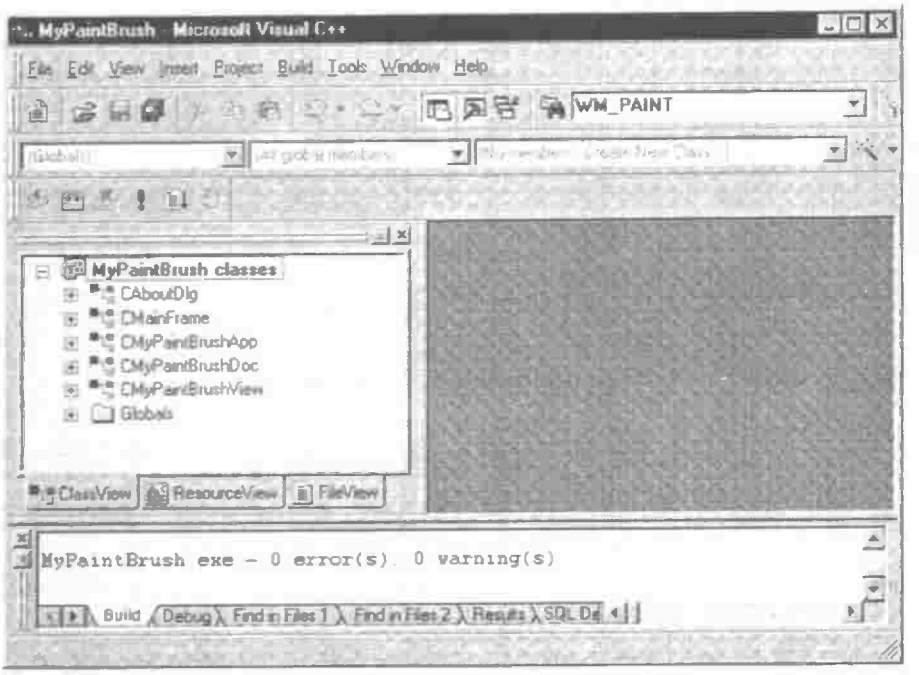
١. استخدام الساحر (MFC AppWizard exe) لبدء مشروع جديد ، وامنحه الاسم MyPaintBrush ، مع تحديد القرص واسم الممر المناسب للمشروع بالصندوق الذى يحمل العنوان Location.

٢. من شاشة الخطوة الأولى من خطوات المشروع (Step 1) اختر المشروع ذا الوثيقة الواحدة (Single Document) ، ثم اضغط الزر Finish.

٣. قم ببناء المشروع وتنفيذه مبدئياً للتحقق من سلامة الخطوات فتحصل على نافذة خالية من المحتويات. أغلق النافذة استعداداً لاستكمال الخطوات.

٤. يوضح الشكل التالى شاشة الأستوديو وبها فصائل المشروع.

لاحظ أنه يمكنك إغلاق المشروع والعودة إليه فى أى وقت لاحق باستخدام أمر قائمة الملفات Open Workspace. تذكر أن تحفظ الملفات من وقت إلى آخر باستخدام أمر القائمة Save All.



شكل (٧-٢) ملفات المشروع MyPaintBrush.dsp

(٧-٣) معالجة الرسائل فى عملية رسم الخطوط

من المهم أن نعرف مقدما أنواع الرسائل المتوقع الاستجابة لها فى البرنامج ، وكيفية الاستجابة لها ، لأن هذا يمثل منطق البرنامج.

عندما ترسم خطا واحدا باستخدام الفأر فإنه من المتوقع أن تستجيب للرسائل الآتية:

الحدث	اسم رسالة النوافذ
الضغط (المستمر) على الزر الأيسر	**WM_LBUTTONDOWN**
سحب الفأر (أثناء ضغط الزر)	**WM_MOUSEMOVE**
إطلاق الزر الأيسر	**WM_LBUTTONUP**

جدول (٧-١) الرسائل المطلوب الاستجابة لها عند رسم الخطوط

وعندما تستجيب لحدث الضغط على الزر الأيسر للفأر فعليك باتخاذ الإجراءات التالية:

- الحصول على منطقة عرض (Display Context).
- تحديد إحداثيات موضع الفأر.

أما عند سحب الفأر فعليك بتطبيق المنطق التالى:

- يتم رسم خط يصل ما بين الموضع الحالى للفأر والموضع السابق. وبالتحديد فإن الخط الذى يرسم فى هذه الحالة خط مؤقت ، لأنك لو حركت نقطة النهاية فإن الخط المرسوم يتحرك معك. أو بمعنى آخر يختفى ويظهر بدلا منه خط جديد يصل ما بين النقطتين الجديدتين. أى أن عملية تحريك نقطة النهاية تنطوى على عدة عمليات رسم ومسح متكررة.
- يلزم أيضا تحديد المنطقة التى ترسم فيها الخطوط حتى لا تخرج عن حدود النافذة.

- من اللازم أثناء هذه الأحداث أن ينحصر اهتمام نظام التشغيل فى النافذة الحالية بحيث يتم استقبال الرسائل المتتابعة بواسطة النافذة الجارى الرسم بها.

أما إذا أطلقت الزر الأيسر فإن الخط المرسوم يصبح نهائيا ويتم إتاحة منطقة العرض (أى تسليمها لنظام لتشغيل).

وفيما يلى سوف نكتب الكود اللازم لتحقيق هذه المهمة.

(٧-٤) تعريف البيانات الأعضاء بفصيلة المشهد

١. أضف البيانات الأعضاء الموضحة بالبنط الثقيل فى الشكل التالى إلى ملف إعلان فصيلة المشهد CMyPaintBrushView. (اضغط ضغطة مزدوجة على اسم الفصيلة فيظهر ملف العناوين فى القسم الأيمن من نافذة الأستوديو).

```
class CMyPaintBrushView : public CView
{
protected: // create from serialization only
// Add these members:
        HCURSOR m_HairCross;
        CPoint m_PrevPoint;
        CPoint m_NewPoint;
        int m_Moving;
//
        CMyPaintBrushView();
        DECLARE_DYNCREATE(CMyPaintBrushView)

// Attributes
```

وسوف نوضح استخدام هذه الأعضاء فى الفقرات التالية ، وهى بصفة عامة تمثل:

- m_HairCross: مؤشر الفأر

- m_PrevPoint: الموقع السابق للنقطة
- m_NewPoint: الموقع الجديد للنقطة
- m_Moving: متغير صحيح يدل على حركة الفأر من عدمها

وكما هو معتاد فإن البيانات الأعضاء تتميز بالبادئة _m كعرف عام. أما أنماط البيانات الجديدة التى ظهرت هنا فهى:

- **HCURSOR**: مقبض لمؤشر الفأر.
- **CPoint**: فصيلة النقطة وهى مماثلة لمنشأ النوافذ **POINT** وهو يحتوى على أعضاء الإحداثيات x, y.

فيما يلى إعلان منشأ النقطة POINT:

```
typedef struct tagPOINT {
        LONG x;
        LONG y;
} POINT;
```

٢.اشحن البيانات الأعضاء الآتية بقيمها الابتدائية فى دالة بناء المشهد CMyPaintBrushView():

```
CMyPaintBrushView::CMyPaintBrushView()
{
  // TODO: add construction code here
// Initialize members:
  m_Moving=0;
  m_HairCross=AfxGetApp()->LoadStandardCursor(IDC_CROSS);
}
```

والقيمة الصفرية للمتغير m_Moving تعنى أن الفأر ليس فى حالة حركة ، أما القيمة غير الصفرية (مثل ١) فسوف تستخدم للدلالة على أن الفأر فى حالة حركة.

تحميل مؤشر الفأر LoadStandardCursor()

تنتمى الدالة **AfxGetApp()** إلى الدوال العامة للمؤسسة MFC ، وسوف نلاحظ أن جميع الدوال العامة بالمؤسسة تبدأ بالحروف Afx. وتستخدم هذه الدالة فى الحصول على مؤشر إلى فصيلة التطبيق CMyPaintBrushApp (وهى ترث من الفصيلة **CWinApp**). وباستخدام المؤشر يمكن التوصل إلى الدالة **CWinApp::LoadStandardCursor()** التى تستخدم فى الحصول على شكل معين لمؤشر الفأر. وفى هذا التطبيق استخدمنا رقم التعارف **IDC_CROSS** للحصول على المؤشر الذى يستخدم عادة فى الرسم والذى يأخذ شكل العلامة +. أما الأشكال الأخرى لمؤشر الفأر فهى موضحة بالجدول التالى.

الثابت (رقم التعارف)	شكل المؤشر الناتج
IDC_ARROW	السهم المعتاد
IDC_IBEAM	مؤشر الكتابة الرأسى
IDC_WAIT	الساعة الرملية
IDC_CROSS	مؤشر الرسم (العلامة +)
IDC_UPARROW	سهم رأسى
IDC_SIZE	غير مستخدم حاليا ويستبدل بالقيمة IDC_SIZEALL
IDC_SIZEALL	سهم ذو أربعة رءوس
IDC_ICON	غير مستخدم حاليا ويستبدل بالقيمة IDC_ARROW
IDC_SIZENWSE	سهم ذو رأسين ، واحدة تتجه إلى الشمال الغربى (NW) ، والأخرى إلى الجنوب الشرقى (SE)

IDC_SIZENESW	سهم ذو رأسين ، واحدة تتجه إلى الشمال الشرقى (NE) ، والأخرى إلى الجنوب الغربى (SW)
IDC_SIZEWE	سهم أفقى ذو رأسين
IDC_SIZENS	سهم رأسى ذو رأسين

جدول (٧-٢) أرقام التعارف لمؤشر الفأر

(٧-٥) استخدام ساحر الرسائل Class Wizard

كما قدمنا ، أن الاستجابة للرسائل تمثل المهمة الرئيسية المطلوب تحقيقها فى البرنامج. ولكى نبرمج مقابض الرسائل فإننا سوف نستخدم ساحراً جديداً يسمى ساحر الفصائل Class Wizard (بخلاف الساحر الذى استخدمناه فى بداية التطبيق والذى يطلق عليه ساحر التطبيقات App Wizard).

الاستجابة لرسالة الضغط على الزر الأيسر WM_LBUTTONDOWN

لإعداد مقبض الاستجابة للضغط على الزر الأيسر اتبع الآتى:

١. استخدم أمر قائمة المشهد View.

٢. من قائمة المشهد ، اختر الأمر Class Wizard (أو استخدم الأزرار Ctrl+W). سوف ترى على الشاشة صندوق الحوار MFC ClassWizard الموضح بالشكل التالى.

٣. اختر البطاقة Message Maps (عادة تكون هذه البطاقة مفتوحة لأنها البطاقة الأولى فى الصندوق).

٧. من قائمة أرقام التعارف (Object IDs:) اختر هدف المشهد (إن لم يكن هو الهدف المختار) ، ومن قائمة الرسائل (Messages:) اختر الرسالة **WM_LBUTTONDOWN**. سوف تلاحظ أثناء التجول فى هذا الصندوق أن بعض الرسائل مكتوبة بالبنط الثقيل ، وهذا معناه أن لها مقابض تم تعريفها من قبل. وعندما تنتهى من تعريف مقبض هذه الرسالة سوف تلاحظ ظهور الاسم بالبنط الثقيل.

٨. اضغط الزر Add Function ، فيؤدى ذلك إلى توليد دالة بالاسم **OnLButtonDown()** وتظهر فى صندوق الدوال الأعضاء (Member functions:) ، كما نرى أسفل الصندوق وصفاً مختصرا لها. عند هذه النقطة ، يكون الساحر قد أضاف الدالة الجديدة إلى فصيلة المشهد وملف العناوين الخاص بها ، كما أنه يضيفها إلى خريطة الرسائل. كما نلاحظ أن اسم الرسالة الموجود بالصندوق Messages: قد اصبح مكتوبا بالبنط الثقيل. أنظر الشكل التالى.

شكل (٧-٤) صندوق حوار ساحر الفصائل بعد إضافة الدالة OnLButtonDown()

٩.إن الدالة التى أضافها الساحر هى مجرد مقبض سابق التعريف ، ولكى نستخدمها فى برنامج الرسم ، علينا أن نضيف إليها الكود اللازم. من نفس النافذة ، اضغط الزر Edit Code فينقلك الساحر إلى الدالة الجديدة **OnLButtonDown()** بنافذة الأستوديو. أضف الكود الموضح بالبنط الثقيل فى شريحة البرنامج التالية:

```
void CMyPaintBrushView::OnLButtonDown(UINT nFlags, CPoint
point)
{
// TODO: Add your message handler code here and/or call default
                                        أضف هذه السطور:

// شحن المتغيرات
        m_NewPoint = point;
        m_PrevPoint = point;
        m_Moving =1;
// حصر منطقة إدخال البيانات على النافذة
```

```
        SetCapture();
// تحديد نطاق حركة الفأر
        RECT Rect;
        GetClientRect (&Rect);
        ClientToScreen (&Rect);
        ::ClipCursor (&Rect);
        ::SetCursor(m_HairCross);
                                                نهاية الكود الجديد
// End of code. The rest is added by MFC
        CView::OnLButtonDown(nFlags, point);
}
```

١٠. قم ببناء المشروع للتأكد من عدم وجود أخطـــاء ، ولكـــن لا تنفذ المشروع فى الوقت الحالى ـــ سيلى التعليق على ذلك.

فيما يلي نشرح معني سطور الكود التي أضفناها إلي دالة المقبض:

نلاحظ أن دالة المقبض التى أضافها الساحر كانت تحتـــوى علـــى عبارة واحدة تستدعى فيها الدالة **OnLButtonDown()** من فصيلـــة الأساس **CView** أى:

```
CView::OnLButtonDown(nFlags, point);
```

ولذلك فلو أنك لم تكتب أى كود بداخل جسم الدالة فإن هذه الدالـــة تقوم بالمهام سابقة التعريف للاستجابة للحدث. وبصفة عامة فإننـــا نبقى على هذه الدالة كما هى ، ونضيف إليها ما نشاء من الملامـــح الخاصة. وتحتوى دالة المقبض على بـــارامترين ، همـــا nFlags و point. أما البارامتر point فيمثل إحداثيات موقع مؤشر الفأر الحالى بالنسبة للركن العلوى الأيسر من النـــافذة (point.y, point.x). أمـــا البارامتر nFlags فهو ثابت يمثل حالة أزرار الفأر ، ويصطلح عليـه

باسم القناع (Mask). والقناع عبارة عن مجموعة من البتـــات (bits) يؤدى كل منها إلى ضبط خاصية معينة ، كما هو موضح بـــالجدول التالى. وسوف نلاحظ أن جميع دوال مقابض رسائل الفأر ســـوف تحتوى على هذين البارامترين.

القناع	معنى البت
MK_CONTROL	الزر Ctrl مضغوط.
MK_LBUTTON	الزر الأيسر مضغوط.
MK_MBUTTON	الزر الأوسط مضغوط.
MK_LBUTTON	الزر الأيمن مضغوط.
MK_SHIFT	الزر Shift مضغوط.

جدول (٧-٣) جدول أقنعة البارامتر nFlags

تحتوى الدالة أيضا على مجموعة من عبارات التخصيـــص التـــى تشحن النقط m_NewPoint ، m_PrevPoint بالإحداثيات الحالية لموقع الفأر. كما تشحن المتغير m_Moving بقيمة غـــير صفريـــة تمـــهيدا لتحريك الفأر.

- أما الدالة **CWnd::SetCapture()** ، فهى تؤدى إلـــى قصـــر منطقة إدخال البيانات على النافذة الحالية ، بمعنـــى أن جميـــع رسائل النوافذ يتم توجيهها إلى نافذة المشهد.

أما مجموعة الدوال التالية (وهى تابعة للفصيلة **CWnd**) فهى تـــؤدى إلى حصر حركة الفأر بداخل حدود النافذة (المستطيل **Rect**) بحيـــث لا يمتد الرسم خارجها:

- الدالة **CWnd::GetClientRect()** ـــ تؤدى إلى إيجـــاد قيمـــة الإحداثيات الحالية للنافذة.
- الدالة **CWnd::ClientToScreen()** ـــ تؤدى إلى التحويل مـــن إحداثيات النافذة إلى إحداثيات الشاشة (سيلى شرح أنواع ونظـــم الإحداثيات فى نهاية الباب).
- الدالة **::ClipCursor()** ـــ تؤدى إلى حصر حركة الفأر بداخل حدود النافذة. وهذه الدالة ليست من دوال المؤسسة MFC ، فـــهى من دوال الوصلة البينية API ، ولذلك فهى لا تتبع أية فصيلـــــة. ونلاحظ هنا أننا استخدمنا مؤثر النطاق (::) مع الدالة. وبـــللرغم من أن هذا المؤثر ليس لازما فى هذا الاستدعاء ، لكننا ســـوف نستخدمه مع دوال الوصلة API لتمييزهـــا عـــن دوال مؤسســـة الفصائل.
- الدالة **::SetCursor()** ـــ تستخدم لتغيير شكل المؤشر إلـــى العلامة + عند ضغط زر الفأر.

لو أنك حاولت تشغيل البرنامج عند هذا الحد ، فإنك سترى مؤشر الفأر محصورا بداخل النافذة بحيث لا تستطيع التوصل إلى زر إغلاق نـــافذة التطبيق ، كما لن تستطيع التوصل إلى أوامـــر القائمـــة لإنـــهاء التطبيـــق. والمخرج الوحيد فى هذه الحالة هـــو اســـتخدام الأزرار Ctrl+Alt+Delete لإنهاء البرنامج.

نضيف فى هذه الفقرة مقبضا لمعالجة الرسالة **WM_MOUSEMOVE** التى ترسلها النوافذ عند تحريك الفأر (أثناء ضغط الزر):

١. استخدم أمر قائمة المشهد View.

٢. من قائمة المشهد ، اختر الأمر Class Wizard ، سوف ترى على الشاشة صندوق الحوار MFC ClassWizard كما جاء فى الفقرة السابقة.

٣. اختر البطاقة Message Maps إن لم تكن هى البطاقة المختارة.

٤. من صندوق قائمة الفصائل (Class name:) اختر فصيلة المشهد CMyPaintBrushView.

٥. من قائمة أرقام التعارف (Object IDs:) اختر هدف المشهد (إن لم يكن هو الهدف المختار) ، ومن قائمة الرسائل (Messages:) اختر الرسالة **WM_MOUSEMOVE**.

٦. اضغط الزر Add Function ، فيؤدى ذلك إلى توليد دالة بالاسم **OnLMouseMove()** وتظهر فى صندوق الدوال الأعضاء (Member functions:). عند هذه النقطة ، يكون الساحر قد أضاف الدالة الجديدة إلى فصيلة المشهد وملف العناوين الخاص بها ، كما أنه يضيفها إلى خريطة الرسائل. كما نلاحظ أن اسم الرسالة الموجود بالصندوق Messages: قد أصبح مكتوبا بالبنط الثقيل.

٧. اضغط الزر Edit Code فينقلك الساحر إلى الدالة الجديدة **OnLMouseMove()** بنافذة الأستوديو. أضف الكود الموضح بالبنط الثقيل فى شريحة البرنامج التالية:

```
void CMyPaintBrushView::OnMouseMove(UINT nFlags, CPoint point)
{
// TODO: Add your message handler code here and/or call default
// Add these lines:                                    أضف هذه السطور
        if (m_Moving)
        {
                CClientDC ClientDC(this);
                ClientDC.SetROP2(R2_NOT);
                ClientDC.MoveTo(m_NewPoint);
                ClientDC.LineTo(m_PrevPoint);
                ClientDC.MoveTo(m_NewPoint);
                ClientDC.LineTo(point);
                m_PrevPoint = point;
        }
// End of code:                                        نهاية الكود الجديد
        CView::OnMouseMove(nFlags, point);
}
```

قم ببناء المشروع للتأكد من عدم وجود أخطاء ، ولكن لا تنفذ المشروع فى الوقت الحالى ، فإذا أردت تجربته ، فإن المخرج الوحيد هو الأزرار Ctrl+Alt+Delete. (عند تجربة البرنامج فسوف تلاحظ أن الفأر يرسم خطوطا عند تحريكه ، ولكن ضغط أو إطلاق الزر لا قيمة له ، لأننا لم نعد مقبضا لرسالة إطلاق الزر).

فيما يلي نشرح معنى سطور الكود التى أضفناها إلى دالة المقبض:

إن الكود الذى أضفناه هنا قد جاء بداخل العبارة الشرطية if (m_Moving) بمعنى أنه "إذا كان الفأر متحركا". وتتضمن عمليات المعالجة ما يلى:

- الحصول على منطقة للعـــرض (Device Context) بنـافذة التطبيق باستخدام الهدف ClientDC التـابع للفصيلـة **CClientDC** (وهذه الفصيلة مشتقة من الفصيلة **CDC** المشتقة مـن **CObject**) وهى تتميز بأنها تتولى تلقائيا بناء وهدم منطقة العرض.
- استدعاء الدالة **CDC::SetROP2()** التى تؤدى إلـى إكسـاب منطقة العرض خاصية عكس الألوان أثناء الرسم. بمعنى أنك لو رسمت الخط لأول مرة فإنه يظهر على الشاشة ، ولـو أعـدت رسمه مرة أخرى فإنه يختفى. وهذا هو الأسلوب المتبـع فـى عمليات الرسم والمسح المتكررة المصاحبة لحركة الفأر. لاحـظ البارامتر **R2_NOT** المستخدم مع هذه الدالة. إن هذا البـارامتر هو مصدر هذه الخاصية. وعندما يتطلب الأمر الخروج من هذا الطور سوف نستخدم بارامترا آخر.
- مسح الخط الذى سبق رسمه (إذا كان هناك خط).
- رسم خط من الموقع الجديد (m_NewPoint) إلى الموقع الحالى (point). لاحظ أن الموقع الجديد هو إحداثى المؤشر عن الضغط على الزر.
- حفظ الموقع الحالى للمؤشر فى المتغير m_PrevPoint. لاحـظ أن الموقع الحالى يتغير كلما حركت الفأر قبل إطلاق الزر.

وفى هذه العملية فإن الدالة **CDC::MoveTo()** تستخدم فى تحديد بداية الخط ، بينما تستخدم الدالة **CDC::LineTo()** فى تحديد نهاية الخـط. ولذلك فإننا نستخدم هاتين الدالتين معا مرة للمسح ، ومرة للرسـم. ومع ذلك فإن الخط الذى نحصل عليه حتـى الآن خـط مؤقـت.

فلنستجب الآن لرسالة إطلاق الزر لكى نرى كيف يستقر الخط على الشاشة.

الاستجابة لرسالة إطلاق الزر الأيسر WM_LBUTTONUP

لتخصيص مقبض للرسالة **WM_LBUTTONUP** اتبع الخطوات التالية:

- استخدم أمر قائمة المشهد View.
- من قائمة المشهد ، اختر الأمر Class Wizard.
- اختر البطاقة Message Maps إن لم تكن هى البطاقة المختارة.
- من صندوق قائمة الفصائل (Class name:) اختر فصيلة المشهد CMyPaintBrushView.
- من قائمة أرقام التعارف (Object IDs:) اختر هدف المشهد (إن لم يكن هو الهدف المختار) ، ومن قائمة الرسائل (Messages:) اختر الرسالة **WM_LBUTTONUP**.
- اضغط الزر Add Function ، فيؤدى ذلك إلى توليد دالة بالاسم **OnLButtonUp()** وتظهر فى صندوق الدوال الأعضاء (Member functions:). عند هذه النقطة ، يكون الساحر قد أضاف الدالة الجديدة إلى فصيلة المشهد وملف العناوين الخاص بها ، كما أنه يضيفها إلى خريطة الرسائل. كما نلاحظ أن اسم الرسالة الموجود بالصندوق Messages قد اصبح مكتوبا بالبنط الثقيل.

- اضغط الزر Edit Code فينقلك الساحر إلى الدالـــة الجديـدة **OnLButtonUp()** بنافذة الأستوديو. أضف الكود الموضح بالبنط الثقيل فى شريحة البرنامج التالية:

```
void CMyPaintBrushView::OnLButtonUp(UINT nFlags, CPoint point)
{
// TODO: Add your message handler code here and/or call default
                                                    أضف هذه السطور
        if (m_Moving)
        {
                m_Moving = 0;
                ::ReleaseCapture();
                ::ClipCursor(NULL);
                CClientDC ClientDC(this);
                ClientDC.SetROP2(R2_NOT);
                ClientDC.MoveTo(m_NewPoint);
                ClientDC.LineTo(m_PrevPoint);
                ClientDC.SetROP2(R2_COPYPEN);
                ClientDC.MoveTo(m_NewPoint);
                ClientDC.LineTo(point);
        }
                                                    نهاية الكود الجديد
        CView::OnLButtonUp(nFlags, point);
}
```

قم ببناء المشروع وتنفيذه فتحصل على النتيجة النهائية للمشروع.

فيما يلي نشرح معنى سطور الكود التي أضفناها إلى دالة المقبض:

إن منطق الكود الذى أضفناه يقـــع بداخـــل عبـــارة شـــرطية (if (m_Moving)) تقول "إذا كان الفأر متحركا". ويتضمن الكود العمليات الآتية:

- إنهاء الحركة بتخصيص القيمة صفر للمتغير m_Moving.

- فك الحصار عن منطقة إدخال البيانات باستخدام الدالة **::ReleaseCapture()** ، وهى الدالة المقابلة للدالة **::SetCapture()**.
- فك الحصار عن مؤشر الفأر باستخدام الدالة **::ClipCursor()** ، وهى نفسها الدالة التى استخدمناها فى حصر المؤشر ، ولكننا نلاحظ هذه المرة استخدام البارامتر NULL. وبهذا يصبح الفأر طليقا ويمكنه التجول خارج النافذة.
- الحصول على منطقة للعرض (Device Context) بإعلان الهدف ClientDC.
- استخدام الدالة **CDC::SetROP2()** مع كل من الدالتين **CPoint::MoveTo()** و **CPoint::LineTo()** لمسح الخط السابق ، بنفس الأسلوب المستخدم فى معالجة رسالة تحريك الفأر.
- استخدام الدالة **CDC::SetROP2()** مع تمرير البارامتر **R2_COPYPEN** لتغيير طور الرسم إلى الطور العادى. ثم استخدام كل من الدالتين **CPoint::MoveTo()** و **CPoint::LineTo()** لرسم الخط النهائى من النقطة الجديدة (m_NewPoint) إلى الموقع الحالى (point).

(٧-٦) رسائل أخرى لجهاز الفأر

فى الجدول التالى نقدم مجموعة رسائل الفأر التى ترسلها النوافذ والتى يمكنك معالجتها سواء فى تطوير هذا البرنامج أو فى أية برامج أخرى تستخدم الفأر.

الرسالة	الحدث المناظر
WM_LBUTTONDOWN	**الضغط على الزر الأيسر.**
WM_LBUTTONUP	**إطلاق الزر الأيسر.**
WM_RBUTTONDOWN	الضغط على الزر الأيمن.
WM_RBUTTONUP	إطلاق الزر الأيمن.
WM_LBUTTONDBCLK	ضغطة مزدوجة على الزر الأيسر.
WM_RBUTTONDBCLK	ضغطة مزدوجة على الزر الأيمن.
WM_MOUSEMOVE	**تحريك الفأر.**
WM_MOUSEWHEEL	تحريك عجلة الفأر (العجلة عبارة عن زر ثالث يوجد ما بين الزرين الأيمن والأيسر ، وهو موجود أساسا فى الفأر طراز ميكروسوفت).

جدول (٧-٤) رسائل الفأر

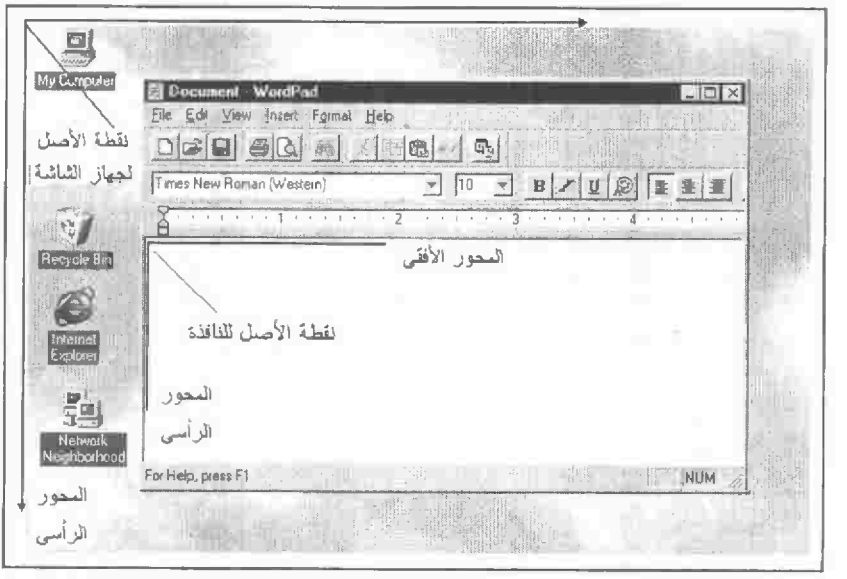
(٧-٧) نظم الإحداثيات

يوجد نوعان من الإحداثيات:

- الإحداثيات الطبيعية (Physical Coordinates).
- الإحداثيات المنطقية (Logical Coordinates).

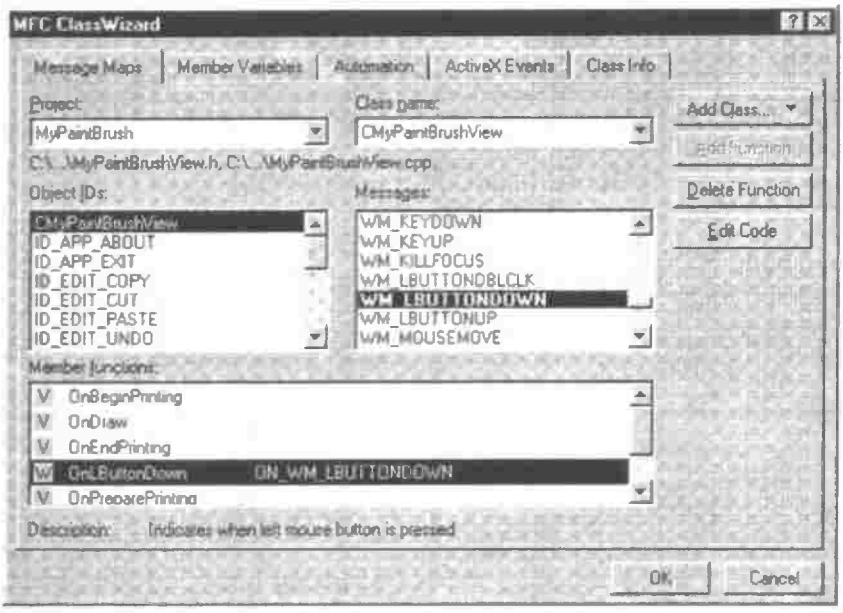
أما الإحداثيات الطبيعية فهى منسوبة إلى جهاز الشاشة أو إلى النافذة كما هو موضح بالشكل التالى ، كما نرى أن نقطة الأصل بأى منهما هى الركن الأيسر العلوى للصفحة أو للشاشة.

وأما الإحداثيات المنطقية فهى منسوبة إلى أجهزة افتراضيـــة ذات مضمون معين مثل صفحة الكتابة أو صفحة الرسم. (ســوف يلــى شرحها بالتفصيل فى الأبواب القادمة حيث يتطلب العمل فى بعــض البرامج التحويل من الإحداثيات الطبيعية إلى الإحداثيات المنطقية).



شكل (٧-٥) إحداثيات الشاشة وإحداثيات النافذة ونقطة الأصل بكل منها

ومن الجدير بالذكر أن منطقة العــرض (Device context) تحتــوى على وحدة قياس الإحداثيات (مثل البوصة أو البكسلة) علاوة علــى أدوات الرسم أو الكتابة. وتعتبر وحدة البكسلة (Pixel) هــى وحــدة القياس سابقة التعريف لكل من النافذة و الشاشة.



شكل (٧-٤) صندوق حوار ساحر الفصائل بعد إضافة الدالة OnLButtonDown()

٩.إن الدالة التى أضافها الساحر هى مجرد مقبض سابق التعريف ، ولكى نستخدمها فى برنامج الرسم ، علينا أن نضيف إليها الكود اللازم. من نفس النافذة ، اضغط الزر Edit Code فينقلك الساحر إلى الدالة الجديدة **OnLButtonDown()** بنافذة الأستوديو. أضف الكود الموضح بالبنط الثقيل فى شريحة البرنامج التالية:

```
void CMyPaintBrushView::OnLButtonDown(UINT nFlags, CPoint
point)
{
// TODO: Add your message handler code here and/or call default
                                        أضف هذه السطور :

// شحن المتغيرات
        m_NewPoint = point;
        m_PrevPoint = point;
        m_Moving =1;
// حصر منطقة إدخال البيانات على النافذة
```

أننا لم نهتم بصيغات الدوال المختلفة واكتفينا بذكر أسمائها والفصائل التى تنتمى إليها. وليس هذا تقصير منا ، فصيغة الدالة تظهر على شاشتك بمجرد كتابة اسمها فى بيئة الأستوديو ، وقد رأينا أن هذه الطريقة تقصر الطريق وتجعلنا نركز على ما هو أهم.

- فى هذا الباب عرفنا كيفية معالجة بعض الرسائل الخاصة بالفأر من خلال خدمة ساحر الفصائل.
- كما عرفنا كيفية استخدام منطقة العرض (Dispaly Context) فى الرسم باستخدام الفصيلة **CClientDC** التى تتولى عملية الحصول على منطقة العرض وإتاحتها تلقائيا ، وهذا يتعذر تحقيقه من خلال دوال الوصلة API. أو باستخدام اللغات الأخرى.
- عرفنا أيضا فى هذا الباب مجموعة من الأنماط والثوابت والدوال:

١. رسائل الفأر:

WM_LBUTTONDOWN
WM_MOUSEMOVE
WM_LBUTTONUP

كما عرضنا مجموعة من رسائل الفأر التى يمكنك التعامل معها فى برامجك ، وهى جميعا موضحة بالجدول (٧-٤).

٢. الأنماط:

HCURSOR
POINT (أو الفصيلة **CPoint**)
RECT

٣. ثوابت أرقام التعارف لمؤشر الفأر (مثل **IDC_CROSS**) ، وهى جميعا معرفة بالجدول (٧-٢).

٤. أقنعة البارامتر nFlags لتحديد وضع أزرار الفأر ، وهى جميعا معرفة بالجدول (٧-٣).

٥. ثوابت الرسم المستخدمة فى تحويل طور الرسم ما بين الألوان المعكوسة أو الرسم العادى:

R2_COPYPEN
R2_NOT

٦. دوال المؤسسة MFC:

الدالة العامة **AfxGetApp()**

CWinApp::LoadStandardCursor()
CWnd::SetCapture
CWnd::GetClientRect()
CWnd::ClientToScreen()
CDC::SetROP2()
CDC::MoveTo()
CDC::LineTo()
CView::OnLButtonDown(nFlags, point)
CView::OnMouseMove(nFlags, point)
CView::OnLButtonUp(nFlags, point)

٧. دوال الوصلة API:

::ClipCursor ();
::SetCursor();
::ReleaseCapture()