الفصل الثانى عشر
الاستجابة للأحداث

نحتاج أحياناً لمعرفة مكان ووقت نقر أو تحرير زر الفأرة دون اللجوء لاستخدام أزرار التحكم. وخير مثال لذلك برنامج الرسام **Paint** المستخدم فى **Windows** والذى من خلاله نقوم برسم مجموعة من الخطوط والدوائر باستخدام أزرار الفأرة فقط حتى نحصل على لوحة رسومية متكاملة.

بانتهاء هذا الفصل، ستتعرف على:

- تحديد المهام التى يقوم البرنامج بتنفيذها بمجرد نقر زر الفأرة أو تحريره
- الاستجابة لحركة مؤشر الفأرة على الشاشة
- التعامل مع أحداث الفأرة واختبار مناطق مختارة من الشاشة

العمل مع أحداث نقر وتحرير أزرار الفأرة

تحتوى بعض الفأرات على ثلاثة أزرار، إلا أن معظمها يحتوى على زرين فقط، الأيمن والأيسر، ومع كل زر من هذه الأزرار يقوم البرنامج باستقبال حدثين، الأول هو حدث نقر الزر **ButtonDown Event** الذى يحدث بمجرد ضغط المستخدم على زر الفأرة، والآخر حدث تحرير الزر **ButtonUp Event** الذى يحدث بمجرد تحرير المستخدم لزر الفأرة.

للتعرف على كيفية ربط هذه الأحداث بالكود المناسب، قم بإنشاء تطبيق نوافذى جديد باستخدام المعالج **Application Wizard** كما تعلمت من قبل وسمه **MouseMsg** ثم قم بزيادة مساحة المربع الحوارى الأساسى **IDD_MOUSEMSG_DIALOG** من داخل نافذة محرر الموارد واحذف مربع النص لتوفير أكبر مساحة ممكنة داخل المربع الحوارى.

إنشاء دالة احتواء حدث زر الفأرة

لإضافة دالة احتواء حدث نقر زر الفأرة الأيسر، تابع معنا الخطوات الآتية:

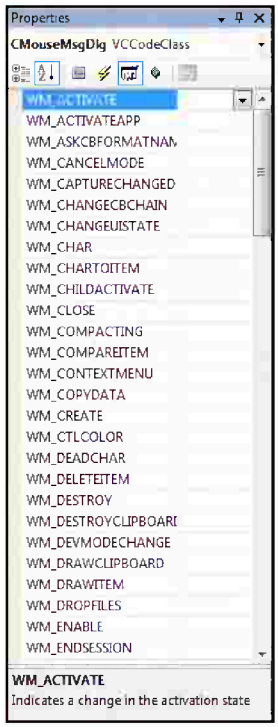
١. قم بتنشيط التبويب **Class View** من نافذة عمل المشروع إذا لم يكن هو التبويب النشط.

٢. انقر رمز التوسيع العلوى لاستعراض التصنيفات الموجودة بالمشروع.

٣. انقر التصنيف **CMouseMsgDlg** بزر الفأرة الأيمن ثم اختر **Properties** من القائمة الموضعية لإظهار مربع الخصائص إذا لم يكن ظاهراً بالفعل أو على الأقل لتنشيطه إذا لم يكن نشطاً.

٤. من مربع الخصائص انقر زر الرسائل  تظهر قائمة برسائل التصنيف التى يمكنك تعيينها (انظر شكل ١٢-١).

٥. اختر **WL_LBUTTONDOWN** من قائمة الرسائل وذلك لأننا سنقوم بكتابة دالة حدث نقر زر الفأرة الأيسر.



شكل ١٢-١ أنواع الرسائل المصاحبة للتصنيف

٦. من مربع السرد والتحرير المجاور للرسالة، اختر **OnLButtonDown <Add>** لإنشاء دالة احتواء الحدث باسم **()OnLButtonDown**، يظهر هيكل الدالة داخل نافذة الكود بالصيغة التالية تمهيداً لكتابة كود هذه الدالة:

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

        CDialog::OnLButtonDown(nFlags, point);
}
```

٧. قم بإدخال الكود التالى للدالة والذى يتسبب فى رسم علامة  كلما قام المستخدم بنقر الزر الأيسر للفأرة مع استبدال هذه العلامة بعلامة  إذا قام المستخدم بنقر الزر الأيسر مع الضغط على مفتاح Ctrl.

```
1.  void CMouseMsgDlg::OnLButtonDown(UINT nFlags,
    CPoint point)
2.  {
3.    // TODO: Add your message handler code here and/or call
            default
4.    CString strMessage;
5.    strMessage.Format("Left Button Clicked at
                                    (%d,%d)",point.x,point.y);
6.    SetWindowText(strMessage);
7.    char* pszIcon;
8.    if(nFlags & MK_CONTROL)
9.          pszIcon = IDI_HAND;
10.   else
11.         pszIcon = IDI_EXCLAMATION;

12.   CDC* pDC = GetDC();
13.   pDC->DrawIcon(point,AfxGetApp()->
                              LoadStandardIcon(pszIcon));
14.   ReleaseDC(pDC);

15.   CDialog::OnLButtonDown(nFlags, point);
16. }
```

وعن هذا الكود، نوضح ما يلى:

- تقوم الدالة OnLButtonDown() باستقبال معاملين، الأول هو nflags والذى يحتوى على قيمة توضح زر الفأرة الذى تم نقره، وما إذا كان المستخدم قد قام بضغط مفتاح Ctrl أو مفتاح Shift عند النقر أم لا وذلك طبقاً للبيانات الموضحة بجدول ١٢-١. أما المعامل الثانى point فهو من كائنات التصنيف CPoint ويحتوى على إحداثيات المكان الذى قام المستخدم بالنقر فيه تبعاً للمربع الحوارى.

- فى السطر رقم ٥، يتم تحويل إحداثيات مكان النقر إلى سلسلة من الحروف لإظهارها على شريط عنوان المربع الحوارى باستخدام الدالة **SetWindowText()** كما فى السطر رقم ٦.
- فى السطور من ٧ إلى ١١، يتم استخدام المؤشر **pszIcon** للإشارة إلى الرمز الذى سيتم رسمه طبقاً لنتيجة اختبار قيمة **nflags** مع مفتاح **Ctrl**. ويوضح جدول ١٢-٢ الرموز القياسية الموجودة بنظام التشغيل ومدلول كلٍ منها.

جدول ١٢-١ رموز المتغير **nflags**

اسم المتغير	الوصف
MK_LBUTTON	نقر الزر الأيسر للفأرة
MK_MBUTTON	نقر الزر الأوسط للفأرة
MK_RBUUTON	نقر الزر الأيمن للفأرة
MK_CONTROL	ضغط مفتاح **Ctrl** من لوحة المفاتيح
MK_SHIFT	ضغط مفتاح **Shift** من لوحة المفاتيح

جدول ١٢-٢ الرموز القياسية الموجودة بنظام التشغيل ومدلول كلٍ منها

اسم الرمز	الوصف
IDI_EXCLAMATION	ظهور علامة تعجب داخل مثلث
IDI_HAND	ظهور علامة x داخل دائرة حمراء
IDI_APPLICATION	الرمز الافتراضى للتطبيق
IDI_ASTERISK	ظهور حرف i صغير للدلالة على وجود معلومة
IDI_QUESTION	ظهور علامة استفهام

- فى السطور من ١٢ إلى ١٤ يتم استخدام مفهوم بيئة العنصر **Device Context** لرسم الرمز المناسب فى مكان النقر على المربع الحوارى.

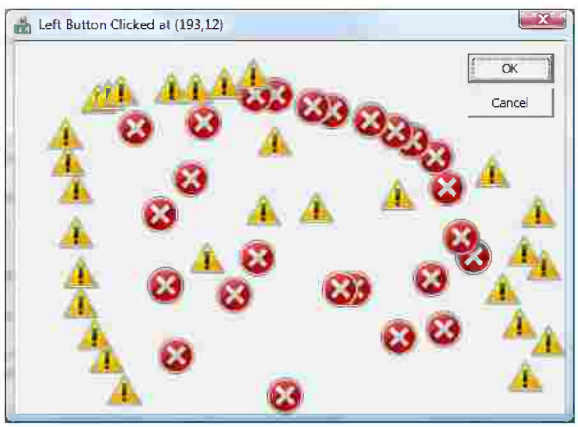
بالنظر إلى أسماء الرموز القياسية الموجودة بنظام التشغيل، نلاحظ أن هناك عدم تناسق بين أسماء بعض الرموز والرموز نفسها. فمثلاً الرمز **IDI_HAND** عبارة عن العلامة ، أى ليس له علاقة إطلاقاً باليد **Hand**. والسبب فى ذلك أن هذه الأسماء كانت مستخدمة فى أنظمة التشغيل **Windows 3.11** و **windows NT 3.51** وقد عُدلت الرموز دون أن تتغير الأسماء.

لتجربة البرنامج، تابع معنا الخطوات الآتية:

١. قم ببناء البرنامج وتنفيذه.

٢. انقر بزر الفأرة الأيسر أى مكان داخل المربع الحوارى، تلاحظ ظهور رمز علامة التعجب فى مكان النقر، بالإضافة إلى ظهور إحداثيات المكان على شريط عنوان المربع الحوارى.

٣. اضغط مفتاح **Ctrl** من لوحة المفاتيح واستمر ضاغطاً ثم انقر بزر الفأرة الأيسر أى مكان داخل المربع الحوارى، تلاحظ ظهور الرمز فى مكان النقر بدلاً من رمز علامة التعجب (انظر شكل ١٢-٢).



شكل ١٢-٢ ظهور الرموز بمجرد نقر زر الفأرة الأيسر

لاحظ جيداً أن إحداثى مكان النقر الذى يظهر على شريط عنوان المربع الحوارى يكون تبعاً للمربع الحوارى نفسه وليس للشاشة ككل، فإذا أردت أن يكون الإحداثى تبعاً للشاشة نفسها، قم بإضافة السطر **GetCursorPos(&point);** إلى بداية الدالة.

وبذلك نكون قد أدخلنا الدالة المناسبة لحدث نقر زر الفأرة الأيسر، فإذا أردت إدخال دالة تحرير الزر أيضاً، تابع معنا الخطوات الآتية:

١. قم بإنشاء دالة احتواء الرسالة **OnLButtonUp** كما سبق وذلك باختيار **WM_LBUTTONUP** من مربع الخصائص بدلاً من **WM_LBUTTONDOWN**.

٢. قم بإضافة الكود التالى للدالة **OnLButtonUp()**:

```
SetWindowText("Left Button Released");
```

والذى يتسبب فى تحول عنوان المربع الحوارى إلى العبارة **Left Button Released** بمجرد تحرير زر الفأرة الأيسر.

وبذلك يمكنك العمل فى نسق معين عندما تقوم بنقر زر الفأرة الأيسر ثم التحول إلى نسق آخر بمجرد تحرير الزر.

والآن يمكنك إن أردت إنشاء دوال احتواء كلٍ من الزر الأوسط والزر الأيمن للفأرة باتباع نفس الخطوات السابقة وطبقاً للجدول ١٢-٣ التالى.

جدول ١٢-٣ أحداث نقر أزرار الفأرة

اسم الحدث	اسم الدالة	الوصف
WM_LBUTTONDOWN	OnLButtonDown()	نقر الزر الأيسر للفأرة
WM_LBUTTONUP	OnLButtonUp()	تحرير الزر الأيسر للفأرة
WM_MBUTTONDOWN	OnMButtonDown()	نقر الزر الأوسط للفأرة
WM_MBUTTONUP	OnMButtonUp()	تحرير الزر الأوسط للفأرة
WM_RBUTTONDOWN	OnRButtonDown()	نقر الزر الأيمن للفأرة
WM_RBUTTONUP	OnRButtonUp()	تحرير الزر الأيمن للفأرة

العمل مع حدث النقر المزدوج

للنقر المزدوج معنىً خاص داخل تطبيقات النوافذ، حيث يستخدم كاختصار لتنفيذ مهام معينة، وفيه يقوم المستخدم بإرسال حدث للتطبيق بنقر زر الفأرة مرتين متتاليتين، حيث يقوم **Windows** باختبار الفترة الزمنية بين النقرتين، فإذا كانت صغيرة، قام بإرسال رسالة **WM_LBUTTONDBLCLK** فى حالة الزر الأيسر أو رسالة **WM_MBUTTONDBLCLK** فى حالة الزر الأوسط أو رسالة **WM_RBUTTONDBLCLK** فى حالة الزر الأيمن.

يمكنك إنشاء دوال احتواء النقر المزدوج باتباع نفس الخطوات المستخدمة فى البند السابق وطبقاً للجدول ١٢-٤ التالى.

جدول ١٢-٤ أحداث النقر المزدوج لأزرار الفأرة

اسم الحدث	اسم الدالة	الوصف
WM_LBUTTONDBLCLK	OnLButtonDblclk()	نقر زر الفأرة الأيسر نقراً مزدوجاً
WM_MBUTTONDBLCLK	OnMButtonDblclk()	نقر زر الفأرة الأوسط نقراً مزدوجاً
WM_RBUTTONDBLCLK	OnRButtonDblclk()	نقر زر الفأرة الأيمن نقراً مزدوجاً

قم بإنشاء الدالة **OnLButtonDblclk()** ثم أضف لها السطر التالى والذى يتسبب فى إظهار مربع رسالة كلما قمت بالنقر المزدوج لزر الفأرة الأيسر.

```
AfxMessageBox("Left Button Double Clicked");
```

والآن قم ببناء التطبيق وتنفيذه. انقر بزر الفأرة الأيسر يظهر الرمز  . انقر بزر الفأرة الأيسر أثناء ضغط مفتاح **Ctrl** بلوحة المفاتيح، يظهر الرمز  . انقر نقراً مزدوجاً بزر

الفأرة الأيسر، يظهر الرمز  كما يظهر أيضاً مربع رسالة يخبرك بالنقر المزدوج لزر الفأرة الأيسر.

يمكنك تعيين الفترة الزمنية بين النقرتين فى النقر المزدوج باستخدام الرمز من داخل لوحة التحكم **Control Panel**.

تتبع حركات مؤشر الفأرة

نحتاج أحياناً إلى إجراء عمليات معينة كلما تحرك مؤشر الفأرة. مثال ذلك برنامج الرسام **Paint** المثبت داخل **Windows**، حيث نحتاج لرسم خطوط كلما تحرك المؤشر، لذلك يقوم نظام التشغيل بإرسال رسالة **WM_MOUSEMOVE** إلى التطبيق لتنفيذ كود معين كلما تحرك المؤشر، كما تتضمن الرسالة أيضاً إحداثيات الوضع الحالى للمؤشر. قم بإنشاء دالة احتواء الرسالة من مربع الخصائص كما سبق لتحصل على الدالة **OnMouseMove()**. قم بتعديل كود الدالة كما يلى:

```
1.    void CMouseMsgDlg::OnMouseMove(UINT nFlags, CPoint
2.    point)
3.    {
4.      // TODO: Add your message handler code here and/or call
                default
5.      CString strMessage;
6.      strMessage.Format("Mouse Position =
                                    (%d,%d)",point.x,point.y);
7.      SetWindowText(strMessage);
8.      m_ptMouse = point;
9.      Invalidate();
10.     CDialog::OnMouseMove(nFlags, point);
11.   }
```

وعن هذا الكود، نوضح ما يلى:

- تستخدم الدالة **OnMouseMove()** نفس المعاملات المستخدمة من قِبَل دوال أحداث أزرار الفأرة.

- فى السطر رقم ٧ يتم استخدام الدالة **SetWindowText()** لإظهار إحداثى مؤشر الفأرة على شريط عنوان المربع الحوارى.
- فى السطر رقم ٨ يتم استخدام المتغير **m_ptMouse** لتخزين إحداثى مؤشر الفأرة.
- فى السطر رقم ٩ يتم استخدام الدالة **Invalidate()** لتعديل عنوان المربع الحوارى.

لإنشاء المتغير العضو **m_ptMouse**، تابع معنا الخطوات الآتية:

١. نشط التبويب **Class View** من نافذة عمل المشروع إذا لم يكن هو التبويب النشط.

٢. انقر رمز التوسيع العلوى لاستعراض التصنيفات الموجودة بالمشروع.

٣. انقر التصنيف **CMouseMsgDlg** بزر الفأرة الأيمن ثم اختر **Add➢Add Variable** من القائمة الموضعية، تظهر نافذة معالج إضافة متغير جديد.

٤. اكتب نوع المتغير **CPoint** فى مربع النص **Variable Type**.

٥. قم بإدخال اسم المتغير **m_ptMouse** فى مربع **Variable Name**.

٦. انقر زر **Finish** لإغلاق نافذة المعالج وإضافة المتغير الجديد إلى التطبيق.

والآن سنقوم باستخدام هذا المتغير داخل دالة الاحتواء **OnPaint()** المستخدمة لتحديث المربع الحوارى باستخدام آخر إحداثى لمؤشر الفأرة.

لرسم زوج من الأعين يتبع مؤشر الفأرة باستمرار، تابع معنا الخطوات الآتية:

١. نشط التبويب **Class View** من نافذة عمل المشروع إذا لم يكن هو التبويب النشط.

٢. انقر رمز التوسيع العلوى لاستعراض التصنيفات الموجودة بالمشروع ثم انقر رمز التوسيع المجاور للتصنيف **CMouseMsgDlg** لإظهار الدوال الأعضاء فى التصنيف بالجزء السفلى من التبويب.

٣. انقر الدالة **OnPaint()** نقراً مزدوجاً، حيث تحتوى هذه الدالة على كود رسم رمز التطبيق عند تقليصه **Minimized** حيث يتم تنفيذ الكود الموجود داخل عبارة **if** إذا أعادت الدالة **IsIconic()** القيمة **True** وإلا يتم تنفيذ الكود الموجود بالجزء **else** كما يلى:

```
1.    void CMouseMsgDlg::OnPaint()
2.    {
3.      if (IsIconic())
4.      {
5.        CPaintDC dc(this); // device context for painting
6.        SendMessage(WM_ICONERASEBKGND,
           reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);

7.        // Center icon in client rectangle
8.        int cxIcon = GetSystemMetrics(SM_CXICON);
9.        int cyIcon = GetSystemMetrics(SM_CYICON);
10.       CRect rect;
11.       GetClientRect(&rect);
12.       int x = (rect.Width() - cxIcon + 1) / 2;
13.       int y = (rect.Height() - cyIcon + 1) / 2;

14.       // Draw the icon
15.       dc.DrawIcon(x, y, m_hIcon);
16.     }
17.     else
18.     {
19.         CDialog::OnPaint();
20.     }
21.   }
```

٤. قم بإضافة الكود التالى إلى الدالة **OnPaint()** داخل عبارة **If** وبعد الشرط **else** مباشرةً كما يلى:

```
1.    else
2.    {
3.      CPaintDC dc(this);
4.      CRect rcDlg;
5.      GetClientRect(&rcDlg);
```

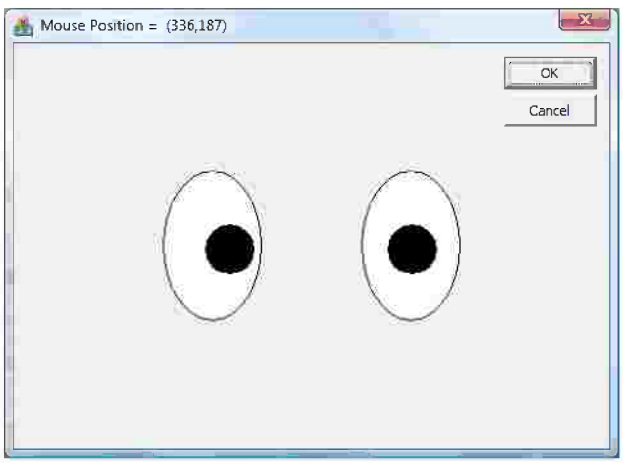
```
6.      for(int i=0;i<2;i++)
7.      {
8.        CPoint ptEye = rcDlg.CenterPoint();
9.        ptEye.x += i==0 ? -80 : +80;
10.       CRect rcEye(ptEye,ptEye);
11.       rcEye.InflateRect(40,60);
12.       dc.SelectStockObject(WHITE_BRUSH);
13.       dc.Ellipse(rcEye);
14.       rcEye.DeflateRect(20,40);
15.       CPoint ptRel = m_ptMouse – rcEye.CenterPoint();
16.       double dX =(double)ptRel.x * (rcEye.Width() /
                                        (double)rcDlg.Width());
17.       double dY = (double)ptRel.y * (rcEye.Height() /
                                        (double)rcDlg.Height());
18.       rcEye.OffsetRect(CPoint((int)dX,(int)dY));
19.       dc.SelectStockObject(BLACK_BRUSH);
20.       dc.Ellipse(rcEye);
21.     }
22.     CDialog::OnPaint();
23.   }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٣ تم إنشاء عنصر رسومى (بيئة أداة) من النوع **CPaintDC**.
- فى السطر رقم ٥ تم استدعاء الدالة **GetClientRect()** للحصول على أبعاد المربع الحوارى وتخزينها داخل المتغير **rcDlg** المعرَّف بالسطر رقم ٤، وهو عضو بالتصنيف **CRect**.
- فى السطر رقم ٦ قمنا باستخدام الدوارة **for** للقيام بجولتين، واحدة لكل عين.
- فى السطر رقم ٨ تم استدعاء الدالة **CenterPoint()** لتحديد منتصف كلٍ من العينين.
- فى السطر رقم ٩ تم تحديد مكان كلٍ من العينين ليكون ٨٠ نقطة يمين أو يسار منتصف المربع الحوارى.
- فى السطور ١٠ و١١ تم رسم كلٍ من العينين فى مكانه الصحيح.
- فى السطر رقم ١٣ تم استدعاء الدالة **Ellipse()** لرسم الجزء الأبيض من العين.

- فى السطر رقم ١٤ تم رسم بؤرة العين.
- تستخدم باقى أسطر الكود لتعيين مكان البؤر كلما تحرك مؤشر الفأرة، وسوف نتعرض لهذه الدوال فى مكان آخر من الكتاب لاحقاً إن شاء الله.

قم ببناء التطبيق وتنفيذه، تلاحظ وجود عينين داخل المربع الحوارى تتبعان حركة مؤشر الفأرة (انظر شكل ١٢-٣).



شكل ١٢-٣ تتبع العينان حركة مؤشر الفأرة

لا يُنصح عادةً بزيادة الكود الذى يتم تنفيذه عند تحرك مؤشر الفأرة وذلك لأنه كلما كثرت المهام المنوطة بالمعالج عند تحرك المؤشر، كلما زادت عمليات المعالجة وبالتالى تقل سرعة الجهاز.

تتبع حركة الفأرة على الشاشة

عندما نقوم ببناء وتشغيل التطبيق الذى بين أيدينا بعد إجراء التعديلات الأخيرة، تلاحظ تتبع الأعين لمؤشر الفأرة طالما كان المؤشر داخل المربع الحوارى، فإذا غادر المؤشر المربع الحوارى إلى الشاشة، تعطل حدث تحرك المؤشر. للتغلب على هذه المشكلة، سنقوم باستعمال الدالة **SetCapture()** لتنشيط حدث تتبع حركة الفأرة خارج المربع الحوارى، والدالة **ReleaseCapture()** لتعطيل الحدث حتى لا يعمل مع البرامج الأخرى. ولعل

أفضل مكان لوضع دالة التنشيط هذه هو دالة نقر الزر الأيسر للفأرة **()OnLButtonDown**، وأفضل مكان لوضع دالة التعطيل هو دالة تحرير الزر **()OnLButtonUp**. لتوضيح هذا الكلام عملياً، تابع معنا الخطوات الآتية:

١. قم بإضافة العبارة **;()SetCapture** لبداية الدالة **()OnLButtonDown**.

٢. قم أيضاً بإضافة العبارة **;()ReleaseCapture** لبداية الدالة **()OnLButtonUp**.

قم ببناء التطبيق وتنفيذه بعد إجراء هذه التغييرات، تلاحظ أنه عند نقر زر الفأرة الأيسر والاحتفاظ به مضغوطاً، يمكنك التحرك فى أى مكان على الشاشة مع تتبع الأعين لمؤشر الفأرة وذلك لتنشيط دالة الالتقاط بمجرد ضغط زر الفأرة الأيسر. أما إذا قمت بتحرير الزر، فلا ترى الأعين إلا داخل المربع الحوارى فقط وذلك لتعطيل دالة الالتقاط بمجرد تحرير الزر.

اختبار النقر

نحتاج عادةً لمعرفة ما إذا كان المستخدم قد قام بالنقر فى مكان معين أم لا، ويترتب على ذلك تنفيذ كود معين، وهو ما نطلق عليه اختبار النقر **Hit Test**. ففى المثال الذى بين أيدينا، إذا أردت تغيير لون العين من الأبيض إلى الرمادى بمجرد نقرها، تابع معنا الخطوات الآتية:

١. قم بإنشاء متغير جديد داخل التصنيف **CMouseMsgDlg** باسم **m_ptButton** من النوع **CPoint** لتخزين آخر مكان تم النقر فيه بزر الفأرة.

٢. قم بتخزين إحداثى آخر مكان تم النقر فيه داخل المتغير الجديد بإضافة السطر التالى داخل دالة نقر الزر الأيسر للفأرة **()OnLButtonDown** .

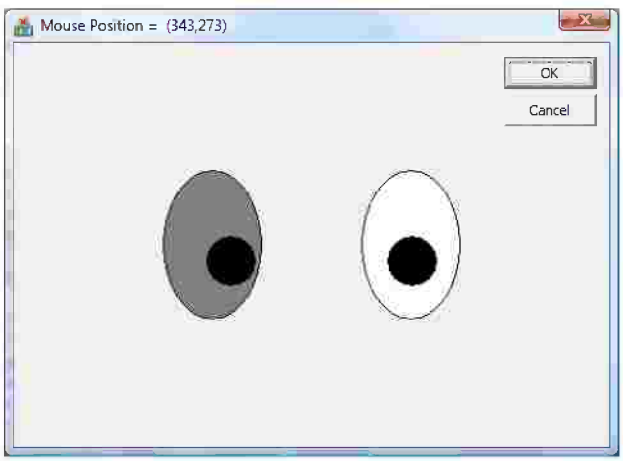
```
m_ptButton = point;
```

٣. قم باختبار المكان الذى تم النقر فيه لترى هل يقع داخل أىٍ من العينين أم لا وذلك بالتعديل الطفيف للدالة **()OnPaint** بعد سطر **;rcEye.InflateRect(40,60)** كما يلى:

```
 rcEye.InflateRect(40,60);
if (rcEye.PtInRect(m_ptButton))
    dc.SelectStockObject(GRAY_BRUSH);
else
    dc.SelectStockObject(WHITE_BRUSH);
.....
```

حيث تقوم الدالة **PtInRect()** بإرجاع القيمة **True** فى حالة ما إذا كان النقر داخل العين وبالتالى يتم صبغها باللون الرمادى وإلا سيتم صبغها باللون الأبيض.

قم ببناء التطبيق وتنفيذه مرةً أخرى بعد إجراء هذه التعديلات وانقر بزر الفأرة الأيسر أى من العينين، تلاحظ تحول العين التى تم نقرها للون الرمادى وتحول العين الأخرى إلى اللون الأبيض تلقائياً إذا كانت باللون الرمادى (انظر شكل ١٢-٤). أما إذا قمت بالنقر خارج العينين، فستلاحظ تحول العينين معاً إلى اللون الأبيض.



شكل ١٢-٤ تحول لون العين إلى الرمادى بمجرد نقرها

استخدام التصنيف CRectTracker

نحتاج أحياناً لاختيار عنصر أو أكثر عن طريق رسم مستطيل حول هذا العنصر أو هذه العناصر. يطلق على هذه العملية **Rectangle Tracking** والتى عادةً ما يتم تنفيذها باستخدام دوال التصنيف **CRectTracker** التى يمكنها تتبع حركة مؤشر الفأرة

وتمكين المستخدم من اختيار منطقة معينة تحتوى على مجموعة من العناصر بالإضافة إلى اختبار أماكن معينة لنرى هل تقع داخل المنطقة المختارة أم لا.

سنقوم فيما يلى بتغيير ألوان الأعين باستخدام عنصر ينتمى إلى التصنيف **CRectTracker** عن طريق اختيار أحد العينين أو كلاهما برسم مستطيل حولهما ثم نختبر ما إذا كان أى من العينين يقع داخل المستطيل ومن ثمَّ تغيير لونها أم لا. تابع معنا الخطوات الآتية:

١. قم بإنشاء متغير جديد عضو فى التصنيف الرئيسى **CMouseMsgDlg** باسم **m_RectTracker** من النوع **CRectTracker** كما تعلمت من قبل.

٢. قم بإعطاء المتغير القيم الابتدائية الخاصة بإحداثيات المستطيل وشكله وذلك داخل دالة إنشاء التصنيف الأساسى **CMouseMsgDlg**، والصيغة العامة لهذا المتغير هى **m_RectTracker(Crect(), Style)**. من نافذة عمل المشروع، انقر بزر الفأرة الأيسر الدالة **CMouseMsgDlg()** نقراً مزدوجاً ثم قم بتعديل كود الدالة كما يلى:

```
CMouseMsgDlg::CMouseMsgDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CMouseMsgDlg::IDD, pParent)
    ,m_RectTracker(CRect(0,0,0,0),CRectTracker::hatchedBorder
                                + CRectTracker::resizeOutside)
    ,m_ptMouse(0)
    ,m_ptButton(0)
```

بالنظر إلى الكود السابق، نلاحظ أن المتغير **m_RectTracker** يحتوى على معاملين. الأول معامل إحداثيات المستطيل وقد أعطينا له القيم صفر وذلك لأننا لا نريد رسم مستطيل لاختيار منطقة معينة حتى الآن. أما المعامل الثانى فهو نمط المستطيل ويتكون من مجموعة من الخصائص والتى يمكنك تضمينها من جدول ١٢-٥ التالى، وقد اخترنا **CRectTracker::hatchedBorder** لرسم إطار رفيع حول المنطقة المختارة، كما اخترنا أيضاً **CRectTracker::resizeOutside** لوضع أذرع التحجيم خارج المستطيل.

جدول ١٢-٥ خصائص نمط المستطيل

الخاصية	الوصف
CRectTracker::solidLine	تحديد المستطيل بخط خارجى
CRectTracker::dottedLine	تحديد المستطيل بخط خارجى منقط
CRectTracker::hatchedBorder	تحديد المستطيل بإطار خارجى رفيع
CRectTracker::hatchInside	تحديد المستطيل بملأ المنطقة من الداخل
CRectTracker::resizeInside	وضع أذرع التحجيم داخل المستطيل
CRectTracker::resizeOutside	وضع أذرع التحجيم خارج المستطيل

٣. لأن عملية الاختيار تتم بمجرد نقر الزر الأيسر للفأرة ثم تحريك المؤشر فى اتجاهٍ ما، قم بإدخال الكود التالى إلى الدالة OnLButtonDown():

```
void CmouseMsgDlg::OnLButtonDown(UINT nFlags , Cpoint point)
{
.....
m_RectTracker.TrackRubberBand(this,point,TRUE);
CDialog:: OnLButtonDown(nFlags , point);
}
```

بالنظر إلى الكود السابق، تلاحظ أننا استخدمنا الدالة TrackRubberBand() والتى تحتوى على ثلاثة معاملات، الأول عبارة عن مؤشر للنافذة التى ستحدث عليها عملية الاختيار وقد استخدمنا المؤشر this للإشارة إلى المربع الحوارى الرئيسى، بينما يحتوى المعامل الثانى على إحداثى نقطة بداية المستطيل وهو الركن الأيسر العلوى من المستطيل، أما المعامل الثالث والأخير فهو اختيارى، وقد يكون TRUE لتمكين المستطيل من التمدد يسار وفوق نقطة البداية أو FALSE لتمكين المستطيل من التمدد يمين وأسفل نقطة البداية فقط.

٤. قم بحذف الدالة SetCapture() من كود الدالة OnLButtonDown() وكذلك الدالة ReleaseCapture() من كود الدالة OnLButtonUp()

وذلك لأن التصنيف RectTracker يحتوى على ذاتية تتبع والتقاط الفأرة فى أى مكان.

٥. قم بتعديل كود اختبار النقر داخل الدالة OnPaint() وذلك لأننا سنقوم باختبار المستطيل وليس نقطة النقر نفسها، حيث يحتوى التصنيف على الدالة HitTest التى يمكنك استدعاءها بتمرير المنطقة المراد اختبارها، وذلك كما يلى:

```
rcEye.InflateRect(40,60);
if (m_RectTracker.HitTest(rcEye.CenterPoint()) !=
                                    CRectTracker ::hitNothing)
        dc.SelectStockObject(GRAY_BRUSH);
else
        dc.SelectStockObject(WHITE_BRUSH);
.....
```

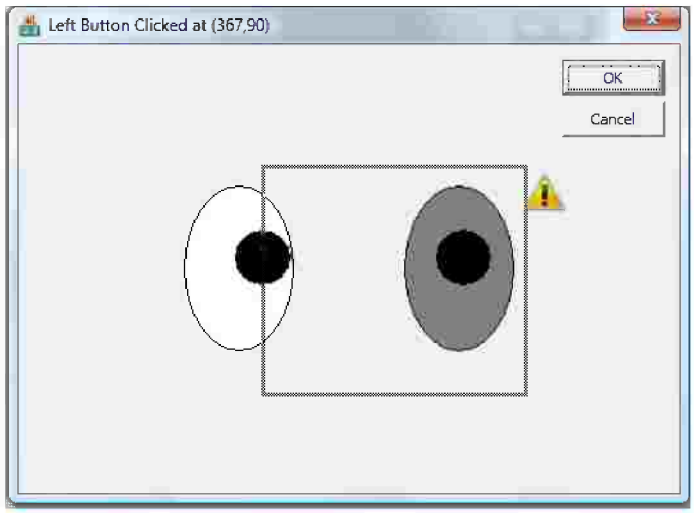
بالنظر إلى الكود السابق، تلاحظ أننا قمنا بتمرير rcEye.CenterPoint() للدالة HitTest() لنسأل هل منتصف أىٍ من العينين يقع داخل المستطيل أم لا ؟ حيث تقوم الدالة بإرجاع القيمة CrectTracker::hitNothing والتى تساوى 1- إذا كانت النقطة لا تقع داخل المستطيل. أما إذا وقعت النقطة داخل المستطيل، فسيتم إرجاع قيمة من القيم الموضحة بجدول ١٢-٦ التالى لتوضح المكان الحقيقى للنقطة داخل المنطقة المختارة.

جدول ١٢-٦ القيم المرجعة من الدالة HitTest()

القيمة المرجعة	القيمة الرقمية	المعنى
CRectTracker::hitNoting	-1	لم يتم النقر
CRectTracker::hitTopLeft	0	تم النقر بالركن الأيسر العلوى من المستطيل
CRectTracker::hitTopRight	1	تم النقر بالركن الأيمن العلوى من المستطيل

القيمة المرجعة	القيمة الرقمية	المعنى
CRectTracker::hitBottomRight	2	تم النقر بالركن الأيمن السفلى من المستطيل
CRectTracker::hitBottomLeft	3	تم النقر بالركن الأيسر السفلى من المستطيل
CRectTracker::hitTop	4	تم النقر بالحافة العلوية من المستطيل
CRectTracker::hitRight	5	تم النقر بالحافة اليمنى من المستطيل
CRectTracker::hitBottom	6	تم النقر بالحافة السفلية من المستطيل
CRectTracker::hitLeft	7	تم النقر بالحافة اليسرى من المستطيل
CRectTracker::hitMiddle	8	تم النقر فى أى مكان داخل المستطيل

والآن قم ببناء التطبيق وتنفيذه بعد إجراء التغييرات السابقة. يمكنك الآن اختيار منطقة معينة داخل المربع الحوارى بنقر زر الفأرة الأيسر والسحب فى أى اتجاه ، تلاحظ ظهور مستطيل حول المنطقة المختارة إلى أن تقوم بتحرير زر الفأرة. فإذا كان منتصف أيٍ من العينين داخل المنطقة المختارة، يتم اختيار العين ويتحول لونها إلى اللون الرمادى (انظر شكل ١٢-٥).



شكل ١٢-٥ تحول لون العين إلى الرمادى بمجرد رسم المستطيل حولها

الباب الرابع
العمل مع عناصر التطبيق

١٣ . التطبيقات أحادية الوثيقة.

١٤ . العمل مع القوائم.

١٥ . العمل مع أشرطة الأدوات وشريط المعلومات.