

## الفصل الثامن عشر

### الاستجابة لأحداث

نحتاج أحياناً لمعرفة مكان ووقت نقر أو تحرير زر الفأرة دون اللجوء لاستخدام أزرار التحكم. وخير مثال لذلك برنامج الرسام **Paint** المستخدم في **Windows** والذي 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 التي يقوم البرنامج بتنفيذها بمجرد نقر زر الفأرة أو تحريره
- ◆ الاستجابة لحركة مؤشر الفأرة على الشاشة
- ◆ التعامل مع أحداث الفأرة واختيار مناطق مختارة من الشاشة

## العمل مع أحداث نقر وتحريك أزرار الفأرة

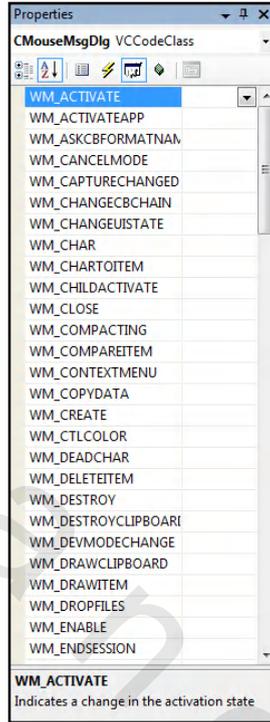
تحتوي بعض الفأرات على ثلاثة أزرار، إلا أن معظمها يحتوي على زرین فقط، الأيمن والأيسر، ومع كل زر من هذه الأزرار يقوم البرنامج باستقبال حدثين، الأول هو حدث نقر الزر **ButtonDown Event** الذي يحدث بمجرد ضغط المستخدم على زر الفأرة، والآخر حدث تحريك الزر **ButtonUp Event** الذي يحدث بمجرد تحريك المستخدم لزر الفأرة.

للتعرف على كيفية ربط هذه الأحداث بالكود المناسب، قم بإنشاء تطبيق نوافذ 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 **CMouseMoveDlg** بزر الفأرة الأيمن ثم اختر **Properties** من القائمة الموضوعية 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 التالي للدالة والذي يتسبب في 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,
2.   CPoint point)
3. {
4.   // TODO: Add your message handler code here and/or call
5.   default
6.   CString strMessage;
7.   strMessage.Format("Left Button Clicked at
8.   (%d,%d)",point.x,point.y);
9.   SetWindowText(strMessage);
10.  char* pszIcon;
11.  if(nFlags & MK_CONTROL)
12.      pszIcon = IDI_HAND;
13.  else
14.      pszIcon = IDI_EXCLAMATION;
15.
16.  CDC* pDC = GetDC();
17.  pDC->DrawIcon(point,AfxGetApp()->
18.      LoadStandardIcon(pszIcon));
19.  ReleaseDC(pDC);
20.
21.  CDialog::OnLButtonDown(nFlags, point);
22. }

```

وعن هذا الكود، نوضح ما يلي:

- تقوم الدالة OnLButtonDown() باستقبال معاملين، الأول هو nflags والذي يحتوي على قيمة توضح زر الفأرة الذي 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 الثاني point فهو من كائنات التصنيف CPoint ويحتوي على إحداثيات المكان الذي قام المستخدم بالنقر فيه تبعاً للمربع الحوارى.

- في السطر رقم ٥، يتم تحويل إحداثيات مكان النقر إلى سلسلة من الحروف لإظهارها على شريط عنوان المربع الحوارى باستخدام الدالة `SetWindowText()` كما في السطر رقم ٦.
- في 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 كل منها.

جدول ١٢-١ رموز المتغير `nFlags`

الوصف	اسم المتغير
نقر الزر الأيسر للفأرة	<code>MK_LBUTTONDOWN</code>
نقر الزر الأوسط للفأرة	<code>MK_MBUTTONDOWN</code>
نقر الزر الأيمن للفأرة	<code>MK_RBUTTONDOWN</code>
ضغط مفتاح <code>Ctrl</code> من لوحة المفاتيح	<code>MK_CONTROL</code>
ضغط مفتاح <code>Shift</code> من لوحة المفاتيح	<code>MK_SHIFT</code>

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

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

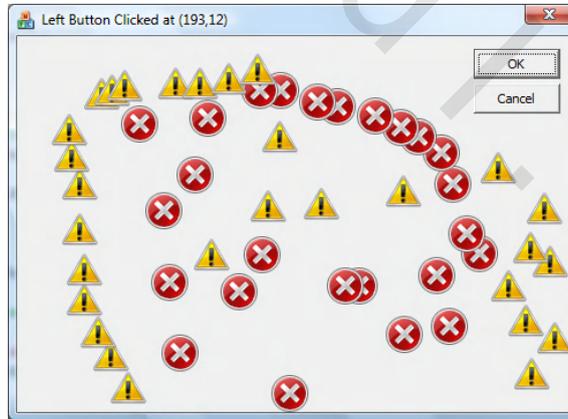
- في السطور من ١٢ إلى ١٤ يتم استخدام مفهوم بيئة العنصر `Device Context` لرسم الرمز المناسب في مكان النقر على المربع الحوارى.

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 في ذلك أن هذه الأسماء كانت مستخدمة في أنظمة التشغيل `Windows 3.11` و `windows NT` و `3.51` وقد عدلت 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  في مكان النقر، بالإضافة إلى ظهور إحدائيات 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  في مكان النقر بدلاً من رمز علامة التعجب (انظر شكل ١٢-٢).



شكل ١٢-٢ ظهور الرموز بمجرد نقر زر الفأرة الأيسر

لاحظ جيداً أن إحداثي مكان النقر الذي 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:

1. قم بإنشاء دالة احتواء الرسالة `OnLButtonUp` كما سبق وذلك باختيار `WM_LBUTTONDOWN` من مربع الخصائص بدلاً من `WM_LBUTTONDOWN`.
2. 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 كل من الزر الأوسط والزر الأيمن للفأرة باتباع نفس الخطوات السابقة وطبقاً للجدول ١٢-٣ التالى.

جدول ١٢-٣ أحداث نقر أزرار الفأرة

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

## العمل مع حدث النقر المزدوج

للنقر المزدوج معنى 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\_LBUTTONDOWNBLCLK في حالة النقر الأيسر أو رسالة WM\_MBUTTONDOWNBLCLK في حالة النقر الأوسط أو رسالة WM\_RBUTTONDOWNBLCLK في حالة النقر الأيمن.

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

جدول ١٢-٤ أحداث النقر المزدوج لأزرار الفأرة

الوصف	اسم الدالة	اسم الحدث
نقر زر الفأرة الأيسر نقرًا مزدوجًا	OnLButtonDownBlcIk()	WM_LBUTTONDOWNBLCLK
نقر زر الفأرة الأوسط نقرًا مزدوجًا	OnMButtonDownBlcIk()	WM_MBUTTONDOWNBLCLK
نقر زر الفأرة الأيمن نقرًا مزدوجًا	OnRButtonDownBlcIk()	WM_RBUTTONDOWNBLCLK

قم بإنشاء الدالة OnLButtonDownBlcIk() 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 نقرًا مزدوجًا بزر

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



يمكنك تعيين الفترة الزمنية بين النقرتين في 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 الحالي للمؤشر. قم  
بإنشاء دالة احتواء الرسالة من مربع الخصائص كما سبق لتحصل على الدالة  
(OnMouseMove). قم بتعديل كود الدالة كما يلي:

```
1. void CMouseMsgDlg::OnMouseMove(UINT nFlags, CPoint  
2. point)  
3. {  
4. // TODO: Add your message handler code here and/or call  
5. default  
6. CString strMessage;  
7. strMessage.Format("Mouse Position =  
8. (%d,%d)",point.x,point.y);  
9. SetWindowText(strMessage);  
10. m_ptMouse = point;  
11. Invalidate();  
12. CDialog::OnMouseMove(nFlags, point);  
13. }
```

وعن هذا الكود، نوضح ما يلي:

- تستخدم الدالة OnMouseMove() نفس المعاملات المستخدمة من قبل 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 **CMsgDlg** 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 **CMsgDlg** لإظهار الدوال الأعضاء فى التصنيف بالجزء السفلى من التبويب.

٣. انقر الدالة `OnPaint()` نقراً مزدوجاً، حيث تحتوي 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 يلي:

```
1. void CMouseMsgDlg::OnPaint()
2. {
3.     if (IsIconic())
4.     {
5.         CPaintDC dc(this); // device context for painting
6.         SendMessage(WM_ICONERASEBKGND,
7.             reinterpret_cast<WPARAM>(dc.GetSafeHdc()), 0);
8.
9.         // Center icon in client rectangle
10.        int cxlcon = GetSystemMetrics(SM_CXICON);
11.        int cylcon = GetSystemMetrics(SM_CYICON);
12.        CRect rect;
13.        GetClientRect(&rect);
14.        int x = (rect.Width() - cxlcon + 1) / 2;
15.        int y = (rect.Height() - cylcon + 1) / 2;
16.
17.        // Draw the icon
18.        dc.DrawIcon(x, y, m_hIcon);
19.    }
20.    else
21.    {
22.        CDialog::OnPaint();
23.    }
24. }
```

٤. قم بإضافة الكود التالي إلى الدالة `OnPaint()` داخل عبارة `If` وبعد الشرط `else` مباشرةً كما يلي:

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

```

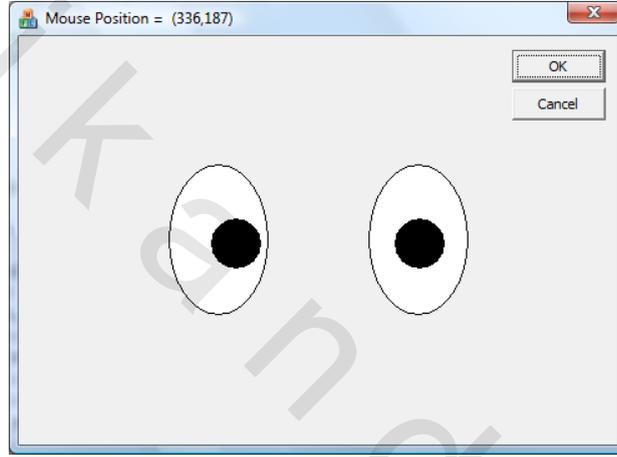
6.   for(int i=0;i<2;i++)
7.   {
8.       CPoint ptEye = rcDlg.CenterPoint();
9.       ptEye.x += i==0 ? -80 : +80;
10.      CRect rcEye(ptEye,ptEye);
11.      rcEye.InflateRect(40,60);
12.      dc.SelectStockObject(WHITE_BRUSH);
13.      dc.Ellipse(rcEye);
14.      rcEye.DeflateRect(20,40);
15.      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 يلي:

- في السطر رقم ٣ تم إنشاء عنصر رسومي (بيئة أداة) من النوع **CPointDC**.
- في السطر رقم ٥ تم استدعاء الدالة **GetClientRect()** للحصول على أبعاد المربع الحوارى وتخزينها داخل المتغير **rcDlg** المعرف بالسطر رقم ٤، وهو عضو بالتصنيف **CRect**.
- في السطر رقم ٦ قمنا باستخدام الدوارة **for** للقيام بجولتين، واحدة لكل عين.
- في السطر رقم ٨ تم استدعاء الدالة **CenterPoint()** لتحديد منتصف كلٍ من العينين.
- في السطر رقم ٩ تم تحديد مكان كلٍ من العينين ليكون ٨٠ نقطة يمين أو يسار منتصف المربع الحوارى.
- في السطور ١٠ و ١١ تم رسم كلٍ من العينين في مكانه الصحيح.
- في السطر رقم ١٣ تم استدعاء الدالة **Ellipse()** لرسم الجزء الأبيض من العين.

- في 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 وبالتالي 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 في أى مكان على الشاشة مع تتبع الأعين المؤشر 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.

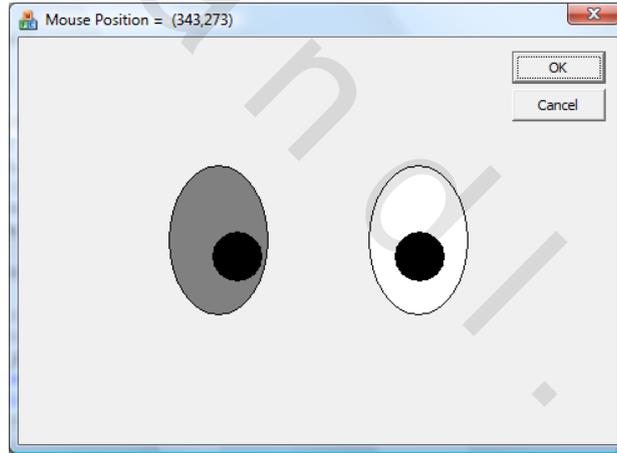
#### اختبار النقر

نحتاج عادةً معرفة ما إذا كان المستخدم قد قام بالنقر في مكان معين أم لا، ويترتب على ذلك تنفيذ كود معين، وهو ما نطلق عليه اختبار النقر `Hit Test`. ففي 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 `CMsgDlg` 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()`).
  ٣. قم باختبار المكان الذى تم النقر فيه لترى هل يقع داخل أى من العينين أم لا وذلك بالتعديل الطفيف للدالة (`OnPaint()`) بعد سطر `m_ptButton = point;`
- كما يلي: `rcEye.InflateRect(40,60);`

```
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` في حالة ما إذا كان النقر داخل العين وبالتالي يتم صبغها باللون الرمادي وإلا سيتم صبغها باللون الأبيض. قم ببناء التطبيق وتنفيذه مرة أخرى بعد إجراء هذه التعديلات وانقر بزر الفأرة الأيسر أي من العينين، تلاحظ تحول العين التي تم نقرها للون الرمادي وتحول العين الأخرى إلى اللون الأبيض تلقائياً إذا كانت باللون الرمادي (انظر شكل ١٢-٤). أما إذا قمت بالنقر خارج العينين، فستلاحظ تحول العينين معاً إلى اللون الأبيض.



شكل ١٢-٤ تحول لون العين إلى الرمادي 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` والتي عادةً ما يتم تنفيذها باستخدام دوال التصنيف `CRectTracker` التي يمكنها تتبع حركة مؤشر الفأرة

وتمكن 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 ثم تغيير لونها أم لا. تابع معنا الخطوات الآتية:

١. قم بإنشاء متغير جديد عضو فى التصنيف الرئيسى **CMsgDlg** 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 **CMsgDlg**، والصيغة العامة لهذا المتغير هي **m\_RectTracker(Crect(), Style)**. من نافذة عمل المشروع، انقر بزر الفأرة الأيسر الدالة **CMsgDlg()** نقرأ مزدوجاً ثم قم بتعديل كود الدالة كما يلى:

```
CMsgDlg::CMsgDlg(CWnd* pParent /*=NULL*/)
: CDialog(CMsgDlg::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 التى 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

الوصف	الخاصية
تحديد المستطيل بخط خارجي	<b>CRectTracker::solidLine</b>
تحديد المستطيل بخط خارجي منقط	<b>CRectTracker::dottedLine</b>
تحديد المستطيل بإطار خارجي رفيع	<b>CRectTracker::hatchedBorder</b>
تحديد المستطيل بملاً المنطقة من الداخل	<b>CRectTracker::hatchInside</b>
وضع أذرع التحجيم داخل المستطيل	<b>CRectTracker::resizeInside</b>
وضع أذرع التحجيم خارج المستطيل	<b>CRectTracker::resizeOutside</b>

٣. لأن عملية الاختيار تتم بمجرد نقر الزر الأيسر للفأرة ثم تحريك المؤشر في اتجاه ما،

قم بإدخال الكود التالي 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()**) والتي تحتوي على ثلاثة معاملات، الأول عبارة عن مؤشر للنافذة التي 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 اختياري، 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 يلي:

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

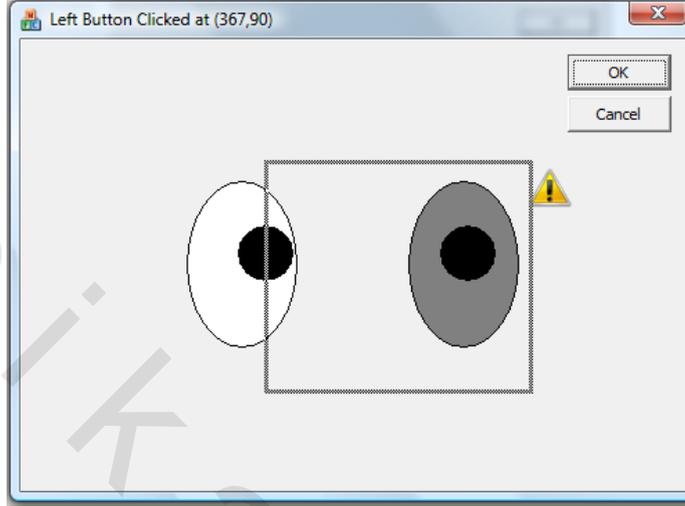
.....  
بالنظر إلى الكود السابق، تلاحظ أننا قمنا بتمرير rcEye.CenterPoint() للدالة HitTest() لسؤال هل منتصف أى من العينين يقع داخل المستطيل أم لا ؟ حيث تقوم الدالة بإرجاع القيمة CrectTracker::hitNothing والتى تساوى -1 إذا كانت النقطة لا تقع داخل المستطيل. أما إذا وقعت النقطة داخل المستطيل، فسيتم إرجاع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

جدول ١٢-٦ القيم المرجعة من الدالة HitTest()

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

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

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 أى من العينين داخل المنطقة المختارة، يتم اختيار العين ويتحول لونها إلى اللون الرمادى (انظر شكل ١٢-٥).



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



## الباب الرابع العمل مع عناصر التطبيق

١٣ . التطبيقات أحادية الوثيقة.

١٤ . العمل مع القوائم.

١٥ . العمل مع أشربة الأدوات وشريط المعلومات.