XỬ LÍ CÁC SỰ KIỆN NHẬP LIỆU

Nội dung

1. Keyboard
2. Mouse
3. Timer

Giới thiệu

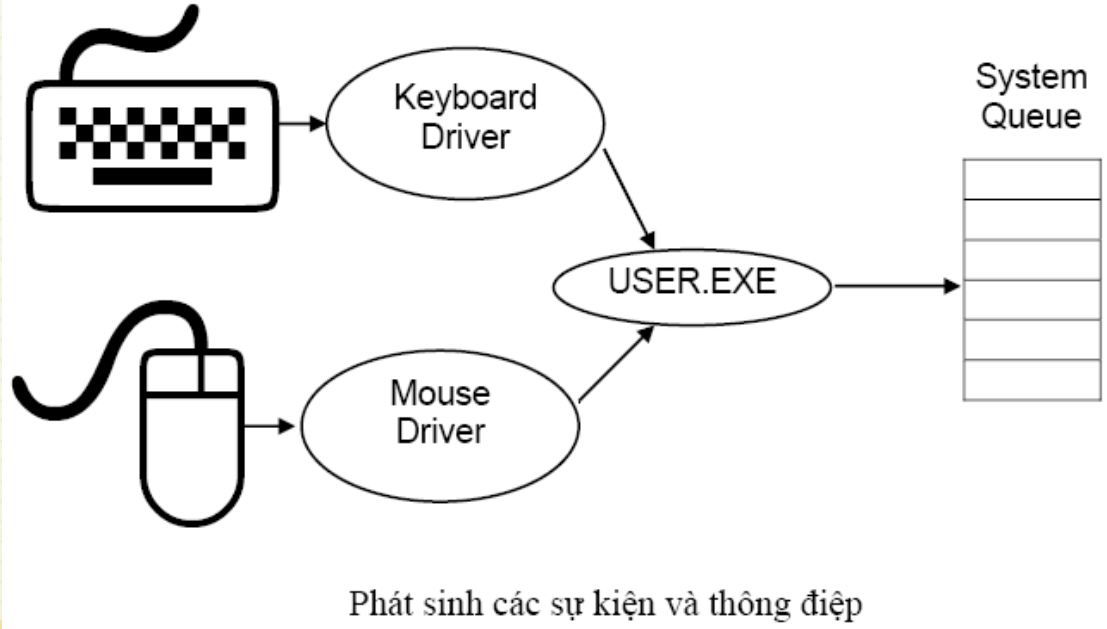
- Tìm hiểu các thông điệp được phát sinh từ bàn phím hay từ thiết bị chuột để viết các xử lý tương ứng với từng thiết bị.
- Bộ định thời gian: Windows cung cấp cơ chế này để truyền thông với ứng dụng theo định kì.
  - Ứng dụng chỉ cần khai báo một bộ định thời gian với một khoảng thời gian cho trước.
  - Khi ứng dụng hoạt động thì hệ thống sẽ truyền một tín hiệu cho ứng dụng theo từng khoảng thời gian định kì đã được khai báo.

Keyboard - Mouse

- Bàn phím và chuột là hai thiết bị nhập liệu quan trọng nhất của máy tính.
- Hầu hết các chức năng của Windows đều hỗ trợ dùng bàn phím và chuột.

Keyboard - Mouse

- Bàn phím và chuột được xử lý thông qua cơ chế thông điệp của Windows.
- Mọi sự kiện đối với bàn phím và chuột được Windows gửi đến chương trình thông qua các thông điệp.



Phát sinh các sự kiện và thông điệp

Keyboard

- Khi nhấn phím có thể xảy ra các trường hợp sau:
  - Nhấn một phím ký tự.
  - Nhấn một phím điều khiển (các phím ESC, Enter, F1-F12..).
  - Nhấn Shift hoặc Ctrl hoặc Alt hoặc một tổ hợp nào đó của ba phím này với các phím ký tự.
- Khi phím trên bàn phím được gõ, nhả hay giữ thì các thông điệp tương ứng sẽ được gửi đến cửa sổ đang được focus.

Keyboard

- Các phím được nhấn được phân thành hai nhóm chính:
  - Nhóm các phím hệ thống (system keys): là các phím được nhấn với phím Alt.
  - Nhóm các phím thường (nonsystem keys): khi phím Alt không được nhấn.
- Thường thì các phím hệ thống được Windows xử lý và dịch thành các sự kiện tương ứng.

Mã phím ảo – Virtual Keycode

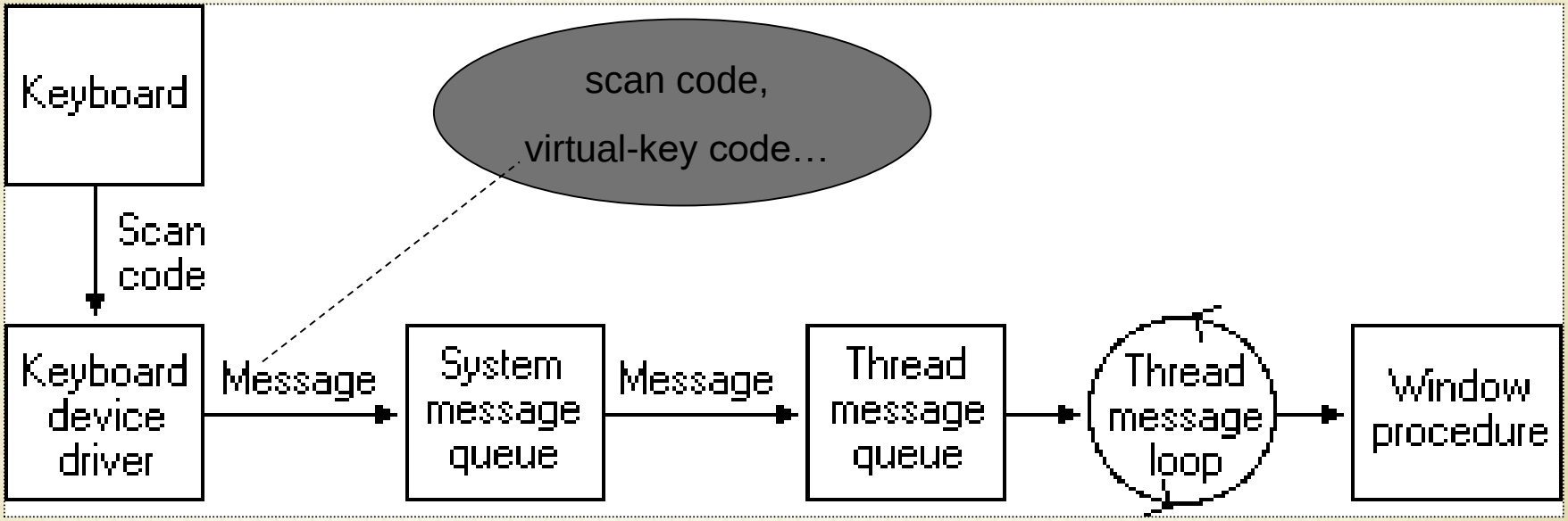
- Windows gán cho mỗi phím trên bàn phím một mã, gọi là mã phím ảo.
- Mã phím ảo là mã không phụ thuộc thiết bị, thay thế cho mã quét (scan code) phụ thuộc loại bàn phím và nhà sản xuất.
- Các mã phím ảo được định nghĩa dưới dạng các macro, bắt đầu bằng VK_.
  - Ví dụ mã phím ảo cho các phím ESC, Enter, F1 hay Alt là VK_ESCAPE, VK_RETURN, VK_F1 và VK_ALT.

Mã phím ảo – Virtual Keycode

- Cần phải phân biệt giữa ký tự nhận được khi ấn phím và mã phím ảo.
  - Ví dụ khi phím A được nhấn, thì ký có thể nhận được ký tự ‘a’ hoặc ‘A’ hoặc không, tuỳ thuộc vào trạng thái phím CAPSLOCK, các phím Shift, Alt, Ctrl có được nhấn hay không.

Xử lý sự kiện bàn phím

- Mô hình xử lý sự kiện bàn phím của Windows



Xử lý sự kiện bàn phím

- Khi người dùng nhấn hoặc nhả một phím bất kỳ từ bàn phím, các driver bàn phím sẽ nhận được mã quét (scan code) của phím tương ứng.
- Mã quét này sẽ được chuyển thành mã phím ảo (Virtual keycode) và một thông điệp bàn phím tương ứng (bao gồm cả scan code, virtual keycode và một số thông tin khác) sẽ được gửi đến cho System message queue.

Xử lý sự kiện bàn phím

- Các sự kiện bàn phím chỉ được gửi đến cho cửa số đang giữ focus hiện hành.
- Hệ thống gửi hai sự kiện bàn phím khác nhau khi người dùng nhấn phím và nhả phím

Xử lý sự kiện bàn phím

- Các phím được nhấn được chia làm 4 nhóm sau:
  - Toggle keys: Caps Lock, Num Lock, Scroll Lock
  - Shift keys: Shift, Ctrl, Alt
  - Noncharacter keys: các phím chức năng như các phím di chuyển, Pause, Delete
  - Character keys: các phím ký tự, phím số,…

Xử lý sự kiện bàn phím

- Khi nhấn hoặc thả phím:

Event	Method	Delegate	Argument
KeyDown	OnKeyDown	KeyEventHandler	KeyEventArgs
KeyUp	OnKeyUp	KeyEventHandler	KeyEventArgs

Xử lý sự kiện bàn phím

Có thể override lại các phương thức OnKeyDown và OnKeyUp

```
protected override void OnKeyDown (KeyEventArgs kea)
{
    …….
}

protected override void OnKeyUp (KeyEventArgs kea)
{
    …….
}
```

Xử lý sự kiện bàn phím

- Cũng có thể xử lý các sự kiện nhấn và thả phím trên các control bằng cách định nghĩa các phương thức tương ứng.

```
void MyKeyDownHandler(object objSender, KeyEventArgs kea)
        {…}
void MyKeyUpHandler(object objSender, KeyEventArgs kea)
        {…}
```

```
cntl.KeyDown += new KeyEventHandler (MyKeyDownHandler);
cntl.KeyUp += new KeyEventHandler (MyKeyUpHandler);
```

Xử lý sự kiện bàn phím

KeyEventArgs Propeties			
Type	Propety	Accesibility	Comments
Keys	KeyCode	get	Identifies the key
Keys	Modifiers	get	Identifies shift states
Keys	KeyData	get	Combination of KeyCode and Modifies
bool	Shift	get	Set to true if Shift key is pressed
bool	Control	get	Set to true if Ctrl key is pressed
bool	Alt	get	Set to true if Alt key is pressed
bool	Handled	get/set	Set by event handler (initially false)
int	KeyValue	get	Return KeyData in the form of an integer.

Xử lý sự kiện bàn phím

- Mỗi khi phím được nhấn hoặc thả thì phát sinh sự kiện, kèm theo một tham số KeyEventArgs có các thuộc tính như sau:
  - Keycode: Cho biết phím nào được nhấn hoặc thả, các phím này có thể bao gồm các phím Shift, Ctrl, Alt
  - Modifiers: Cho biết trạng thái của các phím Shift, Ctrl, Alt trong lúc nhấn phím hay thả phím
  - Keydata: Kết hợp giữa hai thuộc tính Keycode và Modifiers

Xử lý sự kiện bàn phím

- Ví dụ: khi người dùng nhấn phím Shift và phím D sau đó thả phím D và phím Shift sẽ phát sinh liên tiếp các sự kiện sau:

KeyEventArgs Properties and Associated Key Actions				
		Properties		
Action	Event	*KeyCode*	*Modifiers*	*KeyData*
Press Shift	*KeyDown*	*Keys.ShiftKey*	*Keys.Shift*	*Keys.Shift | Keys.ShiftKey*
Press D	*KeyDown*	*Keys.D*	*Keys.Shift*	*Keys.Shift | Keys.D*
Release D	*KeyUp*	*Keys.D*	*Keys.Shift*	*Keys.Shift | Keys.D*
Release Shift	*KeyUp*	*Keys.ShiftKey*	*Keys.None*	*Keys.ShiftKey*

Keys enumeration

- Kiểu Keys được định nghĩa để liệt kê tất cả các phím. Bảng liệt kê giá trị 26 ký tự Latin được mô tả trong bảng sau:

Keys enumeration

Keys Enumeration (letters)			
Member	**Value**	**Member**	**Value**
A	65	***N***	78
B	66	O	79
C	67	***P***	80
D	68	***Q***	81
E	69	***R***	82
F	70	***S***	83
G	71	***T***	84
H	72	***U***	85
1	73	***V***	86
J	74	*W*	87
K	75	***X***	88
L	76	***Y***	89
M	77	*Z*	90

Keys enumeration

***Keys* Enumeration (function keys)**			
Member	**Value**	**Member**	**Value**
F1	112	***F13***	124
F2	113	***F14***	125
F3	114	***F15***	126
F4	115	***F16***	127
F5	116	***F17***	128
F6	117	***F18***	129
F7	118	***F19***	130
F8	119	***F20***	131
F9	120	***F21***	132
F10	121	***F22***	133
F11	122	***F23***	134
F12	123	***F24***	135

Ví dụ

```
class MyForm:Form
   {
      public MyForm()
      {
         this.Text = "Test Keyboard";
         this.KeyDown+=new KeyEventHandler(MyForm_KeyDown);
         this.KeyUp+=new KeyEventHandler(MyForm_KeyUp);
      }
      void MyForm_KeyDown(Object sender, KeyEventArgs kea)
      {
          //if (kea.KeyCode == Keys.X) Application.Exit();
         Console.WriteLine("KeyDown");
      }
      void MyForm_KeyUp(Object sender, KeyEventArgs kea)
      {
          Console.WriteLine("KeyUp");
      }
   }
```

Sự kiện KeyPress

- Sự kiện KeyPress phát sinh khi một phím ký tự được nhấn.

```
protected override void OnKeyPress(KeyPressEventArgs kea)
    {
                    if (kea.KeyChar == 'x')
                        Close () ;
    }
```

Xử lý sự kiện chuột

- Về cơ bản Windows hỗ trợ các loại thiết bị chuột có một nút, hai và ba nút, ngoài ra Windows còn có thể dùng thiết bị khác như joystick hay bút vẽ để bắt chước thiết bị chuột.
- Các thông điệp được tạo từ chuột rất khác với thông điệp của bàn phím:
  - chuột di chuyển qua cửa sổ
  - hay kích vào trong cửa sổ,
  - thậm chí cả trong trường hợp cửa sổ không được kích hoạt hay không nhận được sự quan tâm.

Xử lý sự kiện chuột

- Các sự kiện chuột sẽ được gửi đến cho:
  - Cửa sổ hiện đang chứa con trỏ chuột.
  - Hoặc cửa sổ đang “capture” chuột.
- Có hai loại sự kiện về chuột:
  - Client area messages
    - Các sự kiện chuột xảy ra khi chuột đang ở vùng client của cửa sổ.
  - Non-client area messages
    - Các sự kiện chuột xảy ra khi chuột đang ở các vùng như border, menu bar, title bar, scroll bar, window menu, minimize button, và maximize button.

Các sự kiện cơ bản

- Có 4 sự kiện chuột cơ bản:

Control Events (Selection)			
Event	**Method**	**Delegate**	**Argument**
MouseDown	**OnMouseDown**	**MouseEventHandler**	**MouseEventArgs**
MouseUp	**OnMouseUp**	**MouseEventHandler**	**MouseEventArgs**
MouseMove	**OnMouseMove**	**MouseEventHandler**	**MouseEventArgs**
MouseWheel	**OnMouseWheel**	**MouseEventHandler**	**MouseEventArgs**

MouseEventArgs

- Lớp MouseEventArgs có 5 thuộc tính read-only

MouseEventArgs Properties			
Type	**Property**	**Accessibility**	**Description**
Int	**X**	**get**	**The horizontal position of the mouse**
Int	**Y**	**get**	**The vertical position of the mouse**
MouseButtons	**Button**	**get**	**The muose button or buttons**
Int	**Clicks**	**get**	**Returns 2 for a double-click**
int	**Delta**	**get**	**Mouse wheel movement**

Thuộc tính Button

MouseButtons Enumeration	
Member	Value
None	0x00000000
Left	0x00100000
Right	0x00200000
Middle	0x00400000
XButton1	0x00800000
XButton2	0x01000000

(mea.Button == MouseButtons.Right)

Ví dụ Test Mouse Button

```
protected override void OnMouseClick(MouseEventArgs mea)
        {
            base.OnMouseClick(mea);
            if (mea.Button == MouseButtons.Left)
                MessageBox.Show("Nhan chuot trai");
            if (mea.Button == MouseButtons.Right)
                MessageBox.Show("Nhan chuot phai");
            if (mea.Button == MouseButtons.Middle)
                MessageBox.Show("Nhan chuot giua");

        }
```

Sự kiện MouseDown

- Sự kiện MouseDown được phát sinh khi người dùng nhấn một nút của chuột.
- Để xử lý sự kiện MouseDown ta override phương thức OnMouseDown
- Ví dụ:

```
protected override void OnMouseDown(MouseEventArgs mea)
{
    MessageBox.Show("Ban vua nhan chuot " + mea.Button);
}
```

Sự kiện MouseUp

- Sự kiện MouseUp được phát sinh khi người dùng nhả một nút của chuột.
- Để xử lý sự kiện MouseUp ta override phương thức OnMouseUp
- Ví dụ:

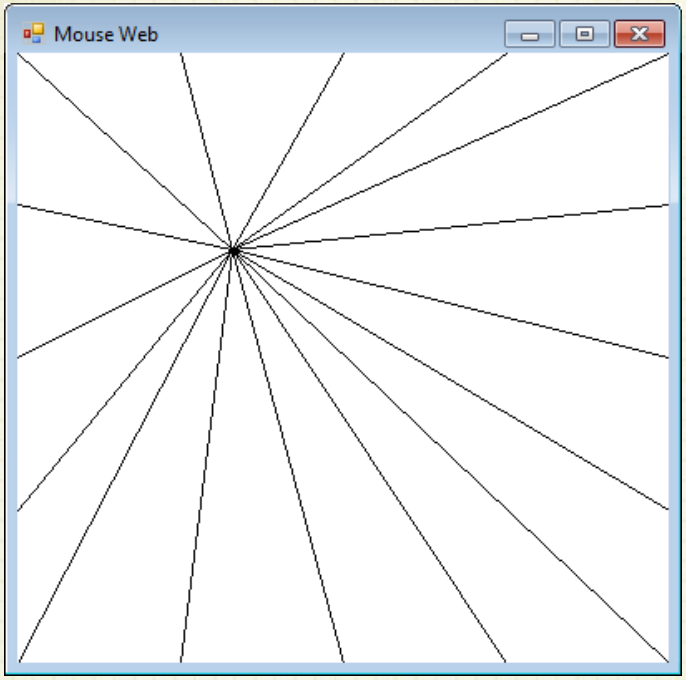
```
protected override void OnMouseUp(MouseEventArgs mea)
{
    MessageBox.Show("Ban vua nha chuot " + mea.Button);
}
```

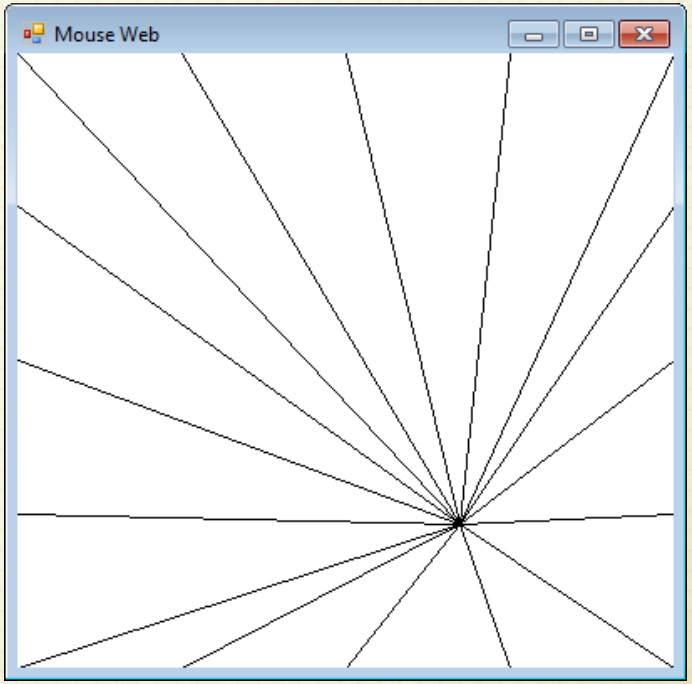
Sự kiện MouseMove

- Sự kiện MouseMove được phát sinh khi người dùng di chuyển chuột.
- Để xử lý sự kiện MouseMove ta override phương thức OnMouseMove
- Ví dụ:

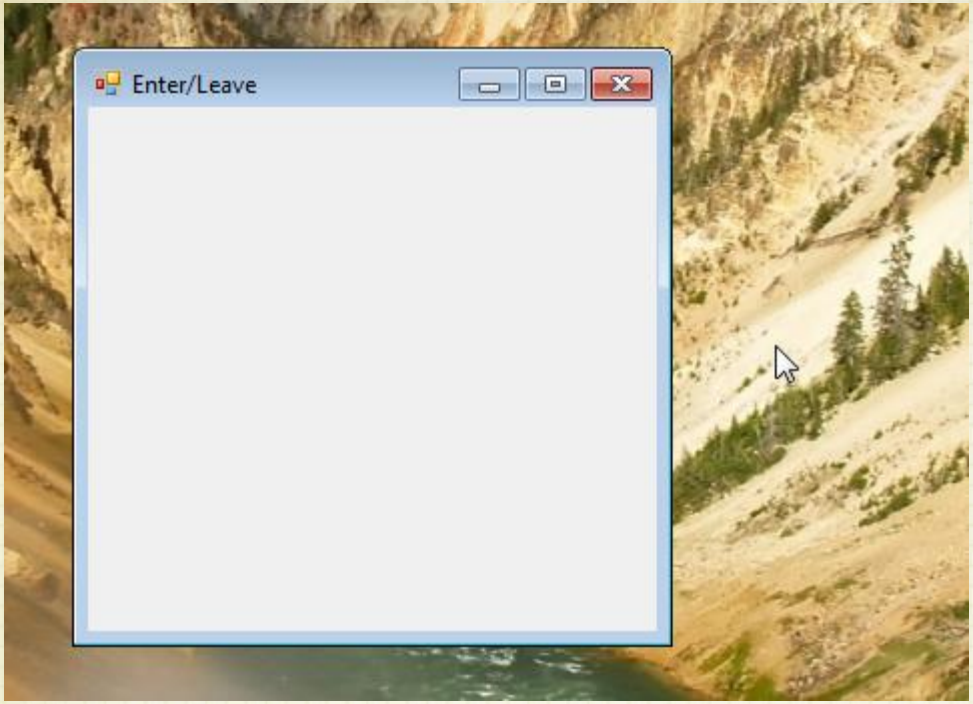
```
protected override void OnMouseMove(MouseEventArgs mea)
{
        //Ve mot duong thang tu toa do (0,0) den toa do chuot di chuyen
        Graphics g = CreateGraphics();
        Pen pen = new Pen(System.Drawing.Color.Blue);
        g.DrawLine(pen, 0, 0, mea.X, mea.Y);
}
```

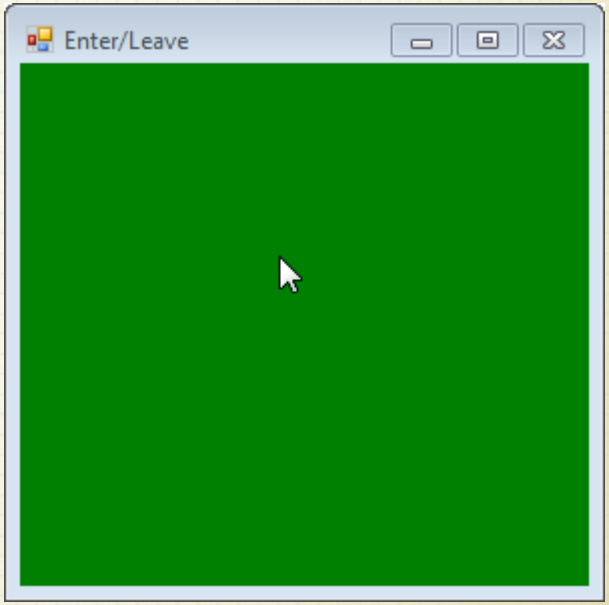
Sự kiện MouseMove





Enter – Hover - Leave





Sự kiện MouseWheel

- Sự kiện MouseWheel được phát sinh khi người dùng scroll chuột.
- Để xử lý sự kiện MouseWheel ta override phương thức OnMouseWheel
- Ví dụ:

```
protected override void OnMouseWheel(MouseEventArgs mea)
{
        if (mea.Delta>0)
                MessageBox.Show("Ban vua scroll chuot len", "Thong bao");
        else
                MessageBox.Show("Ban vua scroll chuot xuong", "Thong bao");
}
```

Sự kiện Click

- Sự kiện Click phát sinh khi một phím **bất kỳ** của chuột được nhấn.
- Sự kiện này phát sinh kèm theo tham số EventArgs, tham số này không chứa thông tin về trạng thái của nút chuột được nhấn cũng như vị trí của con trỏ chuột khi nhấn.

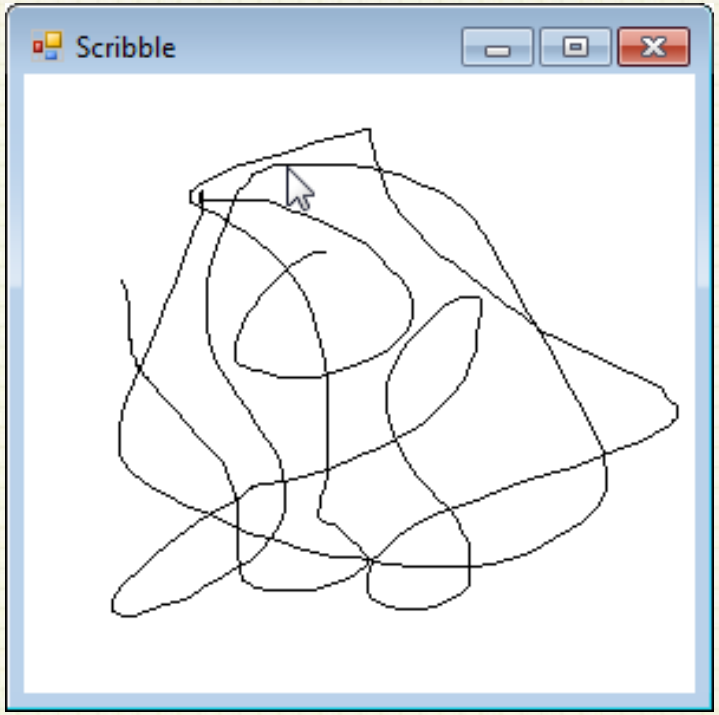
```
protected override void OnClick(EventArgs ea)
        {
                ...
        }
```

Sự kiện DoubleClick

- Khi DoubleClick sẽ phát sinh một dãy các sự kiện sau:
  - MouseDown
  - Click
  - MouseUp
  - MouseMove
  - MouseDown
  - DoubleClick
  - MouseUp
  - MouseMove

```
protected override void OnDoubleClick(EventArgs ea)
        {
                ...
        }
```

Bài tập



Timer

- Multitasking
- Quản lý và thông báo các trạng thái
- Autosave
- Demo version
- Game

Xử lý sự kiện Timer

- Theo lý thuyết thông điệp thời gian do Windows cung cấp là chính xác đến mili giây nhưng thực tế không hoàn toàn như vậy.
- Sự chính xác còn phụ thuộc vào đồng hồ của hệ thống và các hoạt động hiện thời của chương trình.

Lớp Timer

- Có thể tạo đối tượng ***Timer*** bằng cách dùng constructor mặc định như sau:

  Timer timer = new Timer();

- ***Timer*** có một sự kiện:

Timer Event			
Event	Method	Delegate	Argument
Tick	OnTick	EventHandler	EventArgs

Lớp Timer

- Chúng ta có thể định nghĩa sự kiện cho timer như sau:

```
void TimerOnTick(object obj, EventArgs ea)
{
    ….
}
```

- Đăng ký sự kiện:

```
Timer.Tick += new EventHandler(TimerOnTick)
```

Lớp Timer

- Lớp ***Timer*** có 2 thuộc tính:

Timer Properties			
Type	Property	Accessibility	Description
int	Interval	get/set	Tick time in milliseconds
bool	Enabled	get/set	Set to ***true*** if timer is running

Lớp Timer

- Các phương thức của ***Timer*** :

  void Start()

  void Stop()

Lớp Timer (Ví dụ 1)

```
using System;
using System.Drawing;
using System.Windows.Forms;
public class CloseInFive: Form
{
        public static void Main()
        {
                Application.Run(new CloseInFive());
        }
        //……
```

Lớp Timer (Ví dụ 1)

```
public class CloseInFive: Form
{
    public CloseInFive(){
        Text = "Closing in Five Minutes";
        Timer timer    = new Timer();
        timer.Interval = 5 * 60 * 1000;
        timer.Tick += new EventHandler(TimerOnTick);
        timer.Enabled  = true;
    }
    void TimerOnTick(object obj, EventArgs ea){
        Timer timer = (Timer) obj;
        timer.Stop();
        timer.Tick -= new EventHandler(TimerOnTick);
        Close () ;
    }
}
```

Lớp Timer (Ví dụ 2)

```
using System;
using System.Drawing;
using System.Windows.Forms;
class RandomRectangle: Form{
    public static void Main(){
        Application.Run(new RandomRectangle());
    }
    public RandomRectangle(){
        Text = "Random Rectangle";
        Timer timer    = new Timer();
        timer.Interval = 1;
        timer.Tick    += new EventHandler(TimerOnTick);
        timer.Start();
    }
    //……
```

Lớp Timer (Ví dụ 2)

```
    void TimerOnTick(object obj, EventArgs ea)
    {
        Random rand = new Random();
        int x1 = rand.Next(ClientSize.Width);
        int x2 = rand.Next(ClientSize.Width);
        int y1 = rand.Next(ClientSize.Height);
        int y2 = rand.Next(ClientSize.Height);
        Color color = Color.FromArgb(rand.Next(256),
        rand.Next(256), rand.Next(256));
        Graphics grfx = CreateGraphics();
        grfx.FillRectangle(new SolidBrush(color), Math.Min(x1, x2),
                Math.Min(y1, y2), Math.Abs(x2-x1), Math.Abs(y2-y1) );
        grfx.Dispose() ;
    }
}
```