Chapter 12

Keyboard & Mouse Event
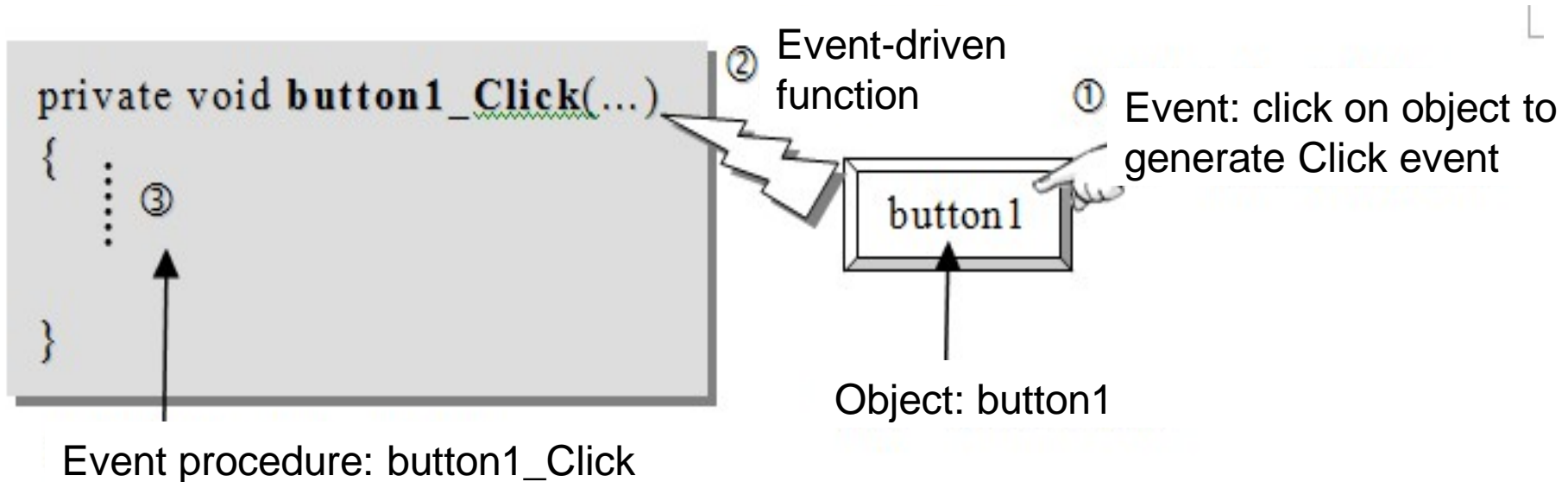
12-1 Event Handler Program Design

- **The communication between programs in Windows program design is formed by message**

- **Messages is sent by event happening**

- **When mouse is moving on the desktop**
  - ⇨ **trigger mouse event continuously**
  - ⇨ **also send message continuously**

- **Event is sent by object**
  - ⇨ **action is caused by user**
  - ⇨ **may be triggered by other program logic**
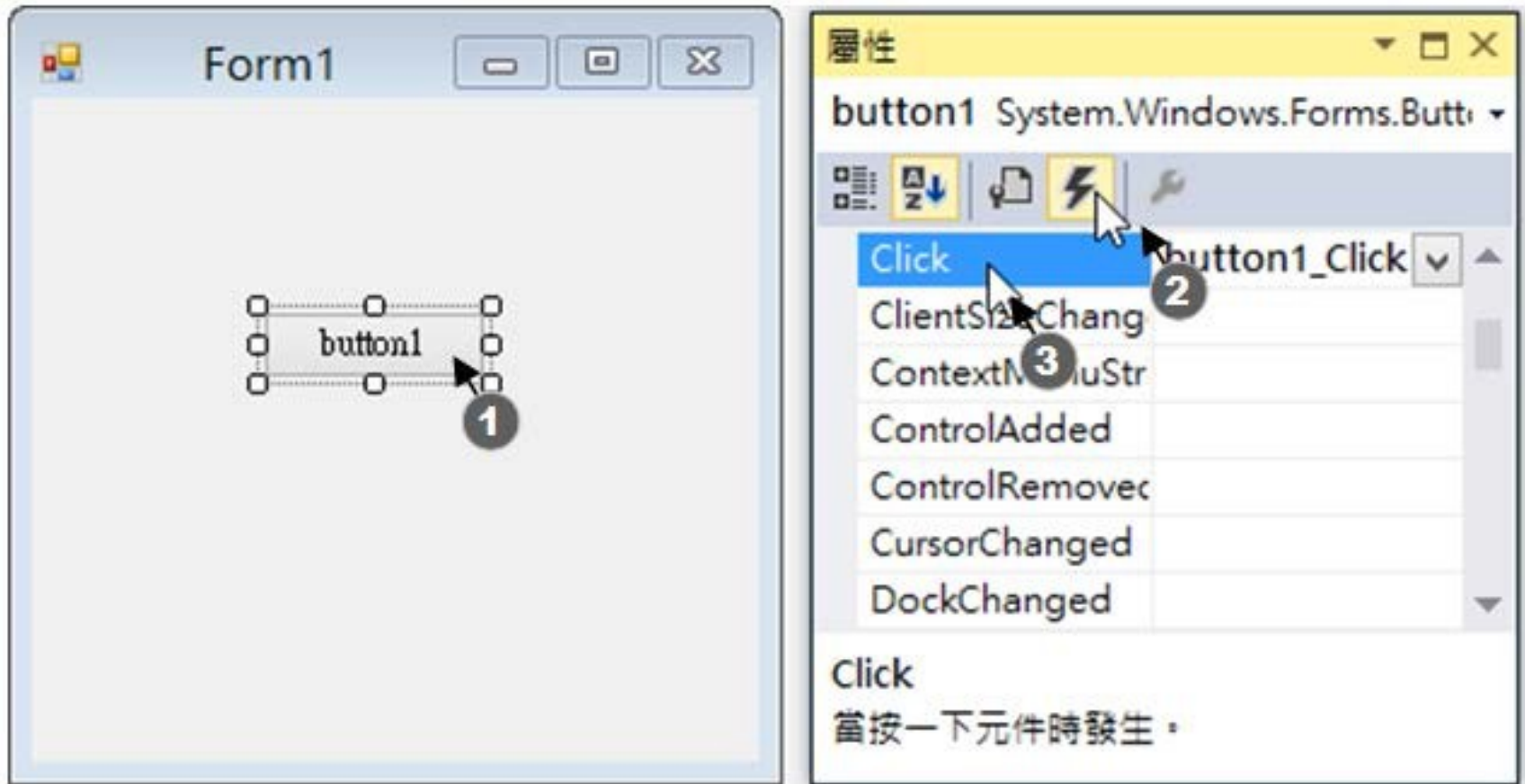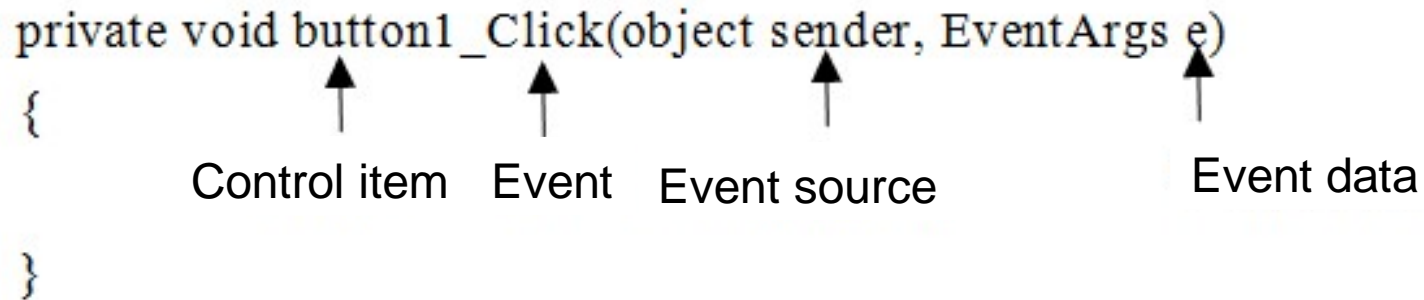
- **An event handler contains 3 elements:**
  - **1. Object**
  - **2. Event-Driven Function**
  - **3. Event**

```
private void button1_Click(...)
{
    ⋮
    ③
    ⋮
}
```

② Event-driven function

① Event: click on object to generate Click event

button1

Object: button1

Event procedure: button1_Click

Create Event in Property Window

```
private void button1_Click(object sender, EventArgs e)
{
            Control item   Event   Event source              Event data

}
```

By the steps of creating event handler, IDE generates the following source code in InitializeComponent method of Form1.Designer.cs automatically for finishing subscription
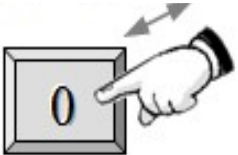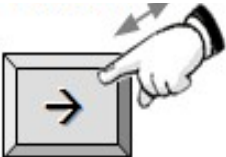
```
private void InitializeComponent()
{
    ...
    this.button1.Click += new System.EventHandler(this.button1_Click);
    ...
}
```

12-2 Keyboard Event

- **Press buttons of keyboard to trigger keyboard events**
- **Keyboard events often used:**
  - **1. KeyDown**
  - **2. KeyPress**
  - **3. KeyUp**
- **Press and release a keyboard button:**
  KeyDown ⇨ KeyPress ⇨ KeyUp

KeyDown & KeyUp Event

- **To press any key to trigger KeyUp and KeyDown events**

- **KeyPress is only triggered by pressing character keys**

- **Other special keys ⇨ use KeyDown or KeyUp**

Press char keys	Trigger: KeyDown > KeyPress > KeyUp
Press control keys	Trigger: KeyDown > KeyUp

KeyDown Event

- **Triggered when the key is pressed**

Grammar

private void controlItem_KeyDown(object sender, KeyEventArgs e)
{
      …
}

KeyUp Event
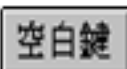
- **Triggered when the key is released**

Grammar
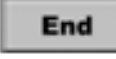
private void controlItem_KeyUp(object sender, KeyEventArgs e)
{
    …
}

Argument e	Description
e.KeyCode	Get the current key code, data type is Keys Ex: KeyCode of button A is Keys.A
e.KeyValue	Get ASCII code of current key, data type is integer Ex: KeyValue of button A is 65
e.Alt	Get whether users press Alt key or not, return true or false
e.Control	Get whether users press Ctrl key or not, return true or false
e.Shift	Get whether users press Shift key or not, return true or false
e.Modifiers	Get whether users press combination keys such as Alt, Ctrl and Shift. Ex: check whether users press Alt+A: if (e.Modifiers == Keys.Alt && e.KeyCode == Keys.A)

Keys and KeyValue Reference Table

Key	Keys Enumeration	KeyValue
0 ~ 9 Numbers	Keys.D0 ~ Keys.D9	48~57
0 ~ 9 Num pad	Keys.NumPad0 ~ Keys.NumPad9	96~105
A ~ Z	Keys.A ~ Keys.Z	65~90
Enter↵、空白鍵	Keys.Enter、Keys.Space	13、32
⇧ Shift、Ctrl	Keys.ShiftKey、Keys.ControlKey	16、17
Alt、Esc、Backspace	Keys.AltKey、Keys.Escape、Keys.Back	18、27、8

Keys and KeyValue Reference Table (cont'd)

Key	Keys Enumeration	KeyValue
← 、 →	Keys.Left 、Keys.Right	37 、39 、
↑ 、 ↓	Keys.Up 、Keys.Down	38 、40
PgUp 、 PgDn	Keys.PageUp 、Keys.PageDown	33 、34
Home 、 End	Keys.Home 、Keys.End	36 、35
Insert 、 Delete	Keys.Insert 、Keys.Delete	45 、46
F1 ~ F12	Keys.F1 ~ Keys.F12	112~123

Ex: run the designated program when Delete key is pressed

```
if (e.KeyCode == Keys.Delete)
```

  or

```
if (e.KeyValue == 46)
```

● **Set KeyPreview property true – form controls keyboard events**

Example(keyDown):

Design a program to move picture with direction keys, requirements:
1. The starting position of the picture is at (116, 116), the picture moves 1 point to the direction which is pointed by the direction key. The title bar shows "未按 Shift 鍵" hint message. The picture moves continuously if the key is pressed and hold.
2. If Shift key and direction key are pressed, the picture moves 10 points to the direction pointed by the direction key, and the title bar shows "按 Shift 鍵" hint message.
3. The picture returns to the starting position when the direction key is released, and the title bar changes to "復原"

Result:
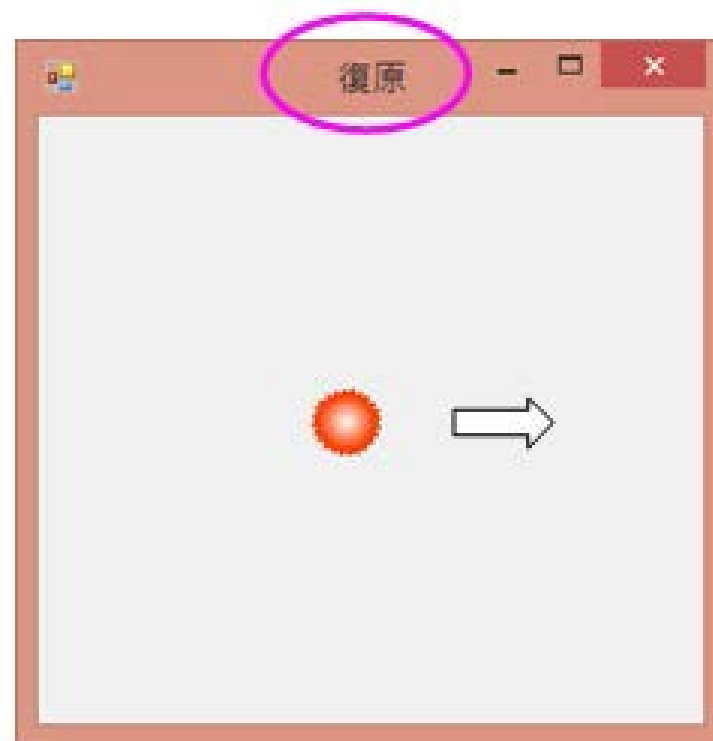
Design User Interface
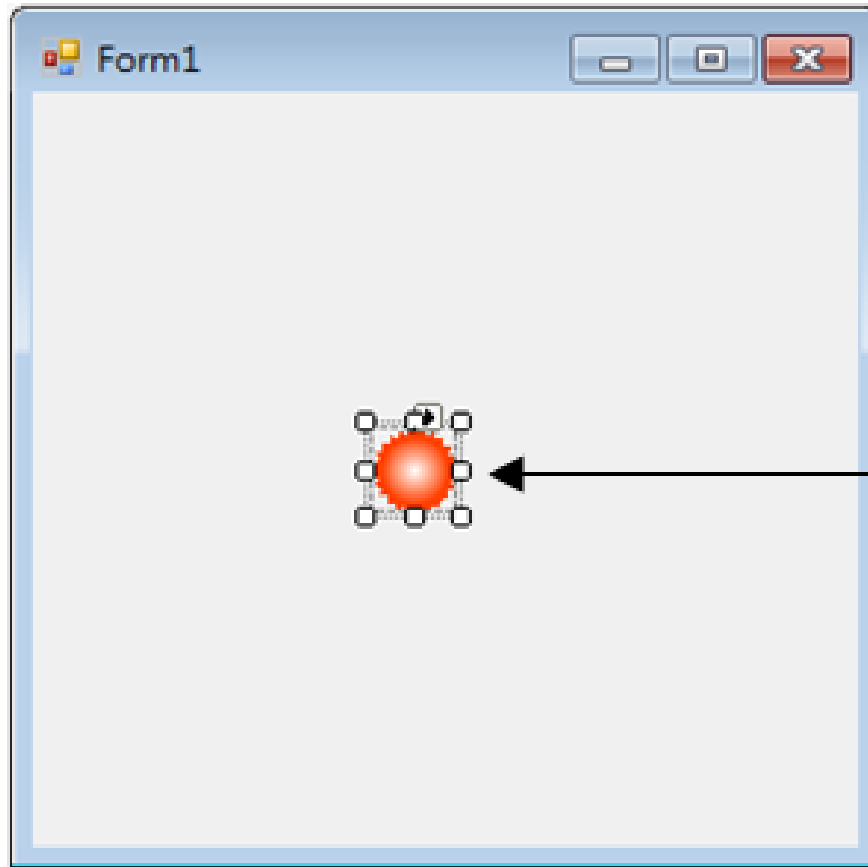
Size=300,300

Namre=picP
Location=116,116
Size=30,30
SizeMode=StretchImage

KeyPress Event

When users press keys of visible characters (KeyValue: 33 ~ 126), enter key ⏎ , space key or backspace key and release it, KeyPress event is triggered. If users press non-character keys such as Alt Alt , F1 and so on, only KeyDown and KeyUp are triggered.

Grammar

```
private void controlItem_KeyPress(object sender, KeyPressEventArgs e)
{
    .
    .
    .
    .
}
```

```
private void controlItem_KeyPress(object sender, KeyEventArgs e)
{
          …
}
```

Argument e	Description
e.KeyChar	Get input characters, data type is char
e.Handled	Set whether the event is processed or not, false: send to system to do default action, true: the event is processed, end event handler directly

```
char c = e.KeyChar;
if (c < 'A' || c > 'Z')
{
    e.Handled = true;    // inputted non-uppercase alphabet is not handled
}
```

Can translate characters to ASCII code (uppercase ASCII code: 65~90)

```
int n = (byte)e.KeyChar;  // translate inputted characters to ASCII code
if (n < 65 || n > 90)
{
    e.Handled = true;     // inputted non-uppercase alphabet is not handled

}
```

Example(atm):

Design an ATM system, requirements:
1. Users can input password when the program starts, but number of withdrawal is not available. The hint shows "請輸入晶片密碼(6~12位)後按下 Enter 鍵"
2. Users can only input numbers and backspace key. The program shows "請案數字鍵" hint message when other character keys are inputted. The maximum number of password is 12. After the password is inputted, press enter key to check password:
   ① If the password is right, focus on number of withdrawal, the password is set to read-only and the program shows "請輸入提款金額後按下 Enter 鍵" hint message
   ② If the password is incorrect, highlight inputted password and show "密碼錯誤，請再試一次" hint message

Result:

Top-left window: ATM提款機
- 密碼：*** ← Password
- 金額： ← Not available
- 請輸入晶片密碼(6~12位)後按下 Enter 鍵

Top-right window: ATM提款機
- 密碼：***
- 金額：
- 請按數字鍵 ← Non-number input

Bottom-left window: ATM提款機
- 密碼：****** ← Not available
- 金額：| ← Focused
- 請輸入提款金額後按下 Enter 鍵

Bottom-right window: ATM提款機
- 密碼：******
- 金額：12500 ← Not available
- 您的提款金額為 12500 元

Design User Interface
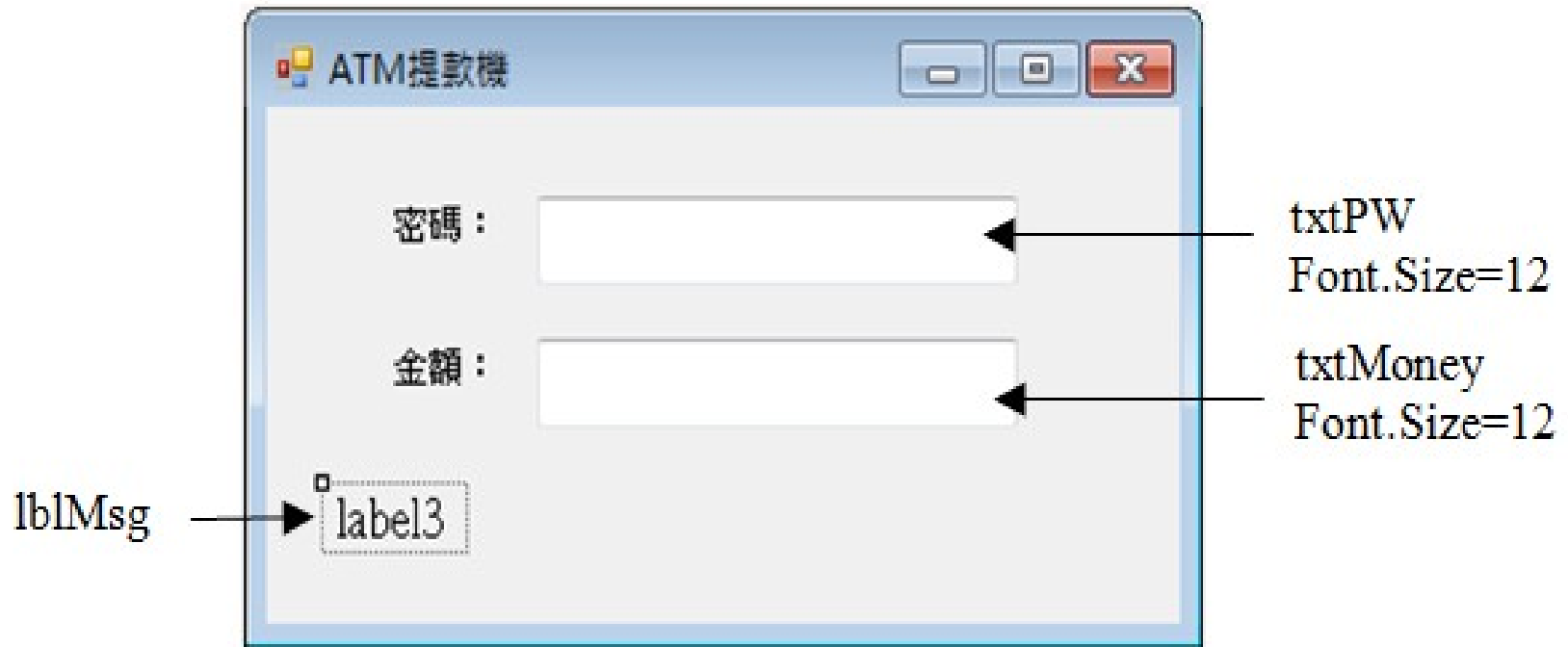
12-3 Mouse Event

Mouse is the most input peripheral. Users can use interface instinctively

Event	Description
MouseClick	Triggered by click mouse left button on the control item
Click	Triggered by click mouse left button on the control item or the control item is focused and space key is pressed
MouseDoubleClick	Triggered by double-click mouse left button on the control item
DoubleClick	Triggered when users click on control item twice

Event	Description
MouseDown	Triggered when the mouse button is pressed
MouseUp	Triggered when the pressed mouse button is release
MouseEnter	Triggered when the mouse cursor moves into the control item
MouseMove	Triggered when the mouse cursor moves within the control item
MouseHover	Triggered when the mouse cursor stays within the control item
MouseLeave	Triggered when the mouse cursor leaves control item
MouseCaptureChanged	As the mouse is clicked, the event is triggered when the mouse stops changing

Click and MouseClick Event

- **Click event is the most used event**

- **Click on control item and release left button:**
  - ⇨ **trigger Click event of control item**
  - ⇨ **MouseClick is also triggered**

- **Click and MouseClick is triggered in the same time**
  - ⇨ **but parameters are different**

● **Click and MouseClick event handler:**

Grammar
| ```
private void 控制項_MouseClick(object sender, MouseEventArgs e)
{
    ⋮
}

private void 控制項_Click(object sender, EventArgs e)
{
    ⋮
}
``` |

MouseDown and MouseUp Event

- **When user's mouse cursor is on control item:**
  - ⇨ **trigger MouseDown event by pressing mouse button**
  - ⇨ **trigger MouseUp event by releasing mouse button**

Grammar
| ```
private void 控制項_ MouseDown(object sender, MouseEventArgs e)
{
    ⋮
}
``` |

Argument e	Description
e.Button	Get which button users press, values: 1. MouseButtons.Left (left key) 2. MouseButtons.Middle (middle key) 3. MouseButtons.Right (right key) Ex: whether left key or not if (e.Button == MouseButtons.Left)
e.X	Get x position of mouse cursor
e.Y	Get y position of mouse cursor

- **e.X and e.Y property values**
  - ⇨ **stand for (X, Y) position of mouse cursor**
- **Based on top-left corner of control item**

DoubleClick and MouseDoubleClick Event

- **Rapidly double-click on the control item**
  ⇨ trigger DoubleClick and MouseDoubleClick events

- **Not all control items supports these 2 events**
  ex: Button, CheckBox and RadioButton do not support

Click once	Trigger these events: MouseDown ⇨ Click ⇨ MouseClick ⇨ MouseUp ⇨ MouseCaptureChanged
Click twice	Trigger these events: MouseDown ⇨ Click ⇨ MouseClick ⇨ MouseUp ⇨ MouseDown ⇨ DoubleClick ⇨ MouseDoubleClick ⇨ MouseUp ⇨ MouseCaptureChanged

MouseEnter, MouseMove, MouseHover and MouseLeave Events

- **When the mouse move on control item**
  ⇨ trigger MouseEnter event

- **Move within control item ⇨ trigger MouseMove continuously**

- **Cursor stays within control item ⇨ trigger MouseHover**

- **Leave control item ⇨ trigger MouseLeave**

- **When mouse cursor move into control item, stay for a while and leave, trigger these events**

MouseEnter ⇨ MouseMove ⇨ MouseHover ⇨ MouseMove ⇨ MouseLeave 事件

MouseEnter, MouseHover and MouseLeave are only triggered once, MouseMove is triggered continuously when the mouse moves

②MouseMove …　④MouseMove …

①MouseEnter　③MouseHover　⑤MouseLeave

滑鼠進入　滑鼠移動　滑鼠移動　滑鼠離開

① ② ③ ④ ⑤

滑鼠停留

Example(drag):

Design a program to drag the picture, requirements:
1. There is a closed box in the form. The box is opened when the mouse moves in
2. Press and hold box to drag it, the box is moved in the same time
3. When mouse leaves the box, it is closed.

Result:

Design User Interface

Practice 6_bonus: Keyboard Event

- **Use keyboard event to let label move, and show which key you press and it's keycode.**
- **When the mouse is being clicked, press label shows which click it is and its coordinate.**

12- 4 Sharing Event

- **An event handler can be used by many other control items**
- **Simplify source code and improve maintainability**

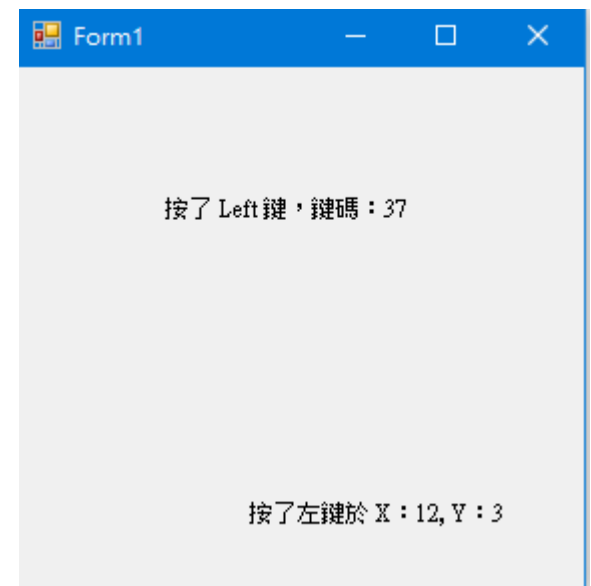# 12- 4 Sharing Event

- **Add sharing event in design phase or runtime**
- **C# allows different-class control item sharing events**
- **Different events can use the same event handler as long as number and data type of arguments are identical**
- **Can share customized event handler**

Set that CheckedChanged event of radioButton1 shares the event handler of button1's Click event

Assign Shared Event handler in Design Phase

After the steps mentioned before, the system generate these source code in InitializeComponent method of Form1.Designer.cs to finish subscription

```
private void InitializeComponent()
{
    …
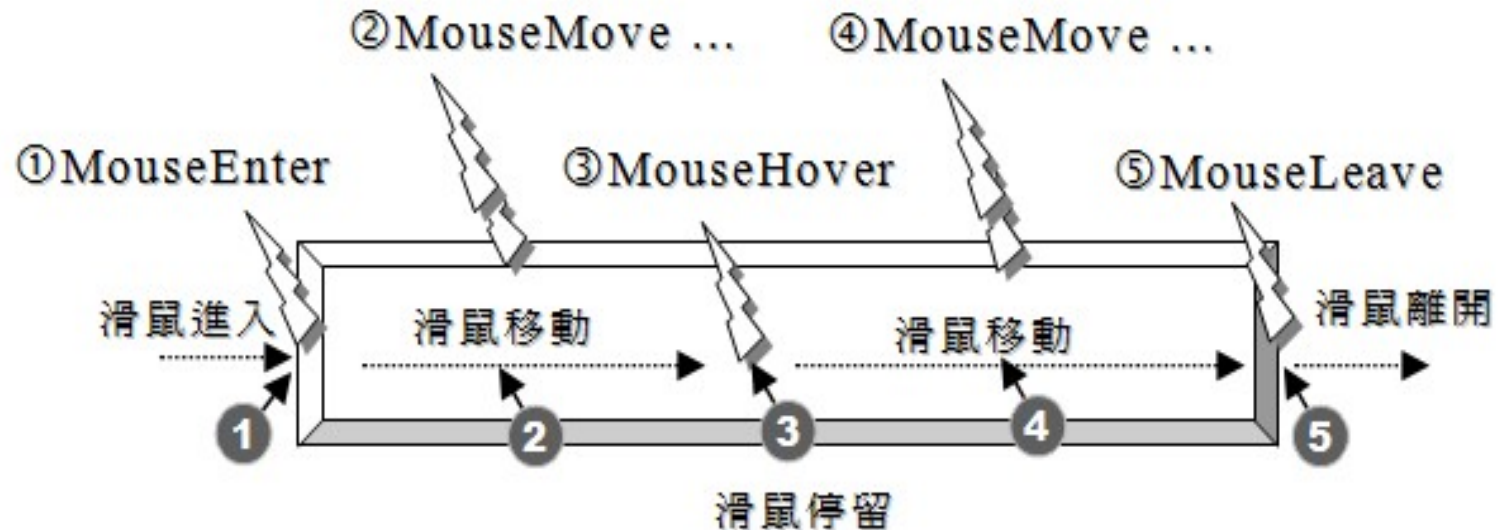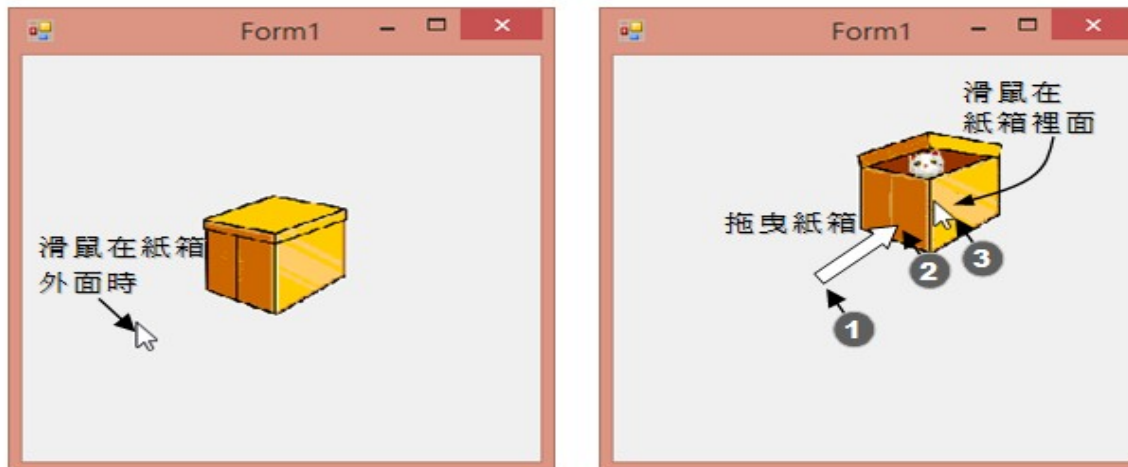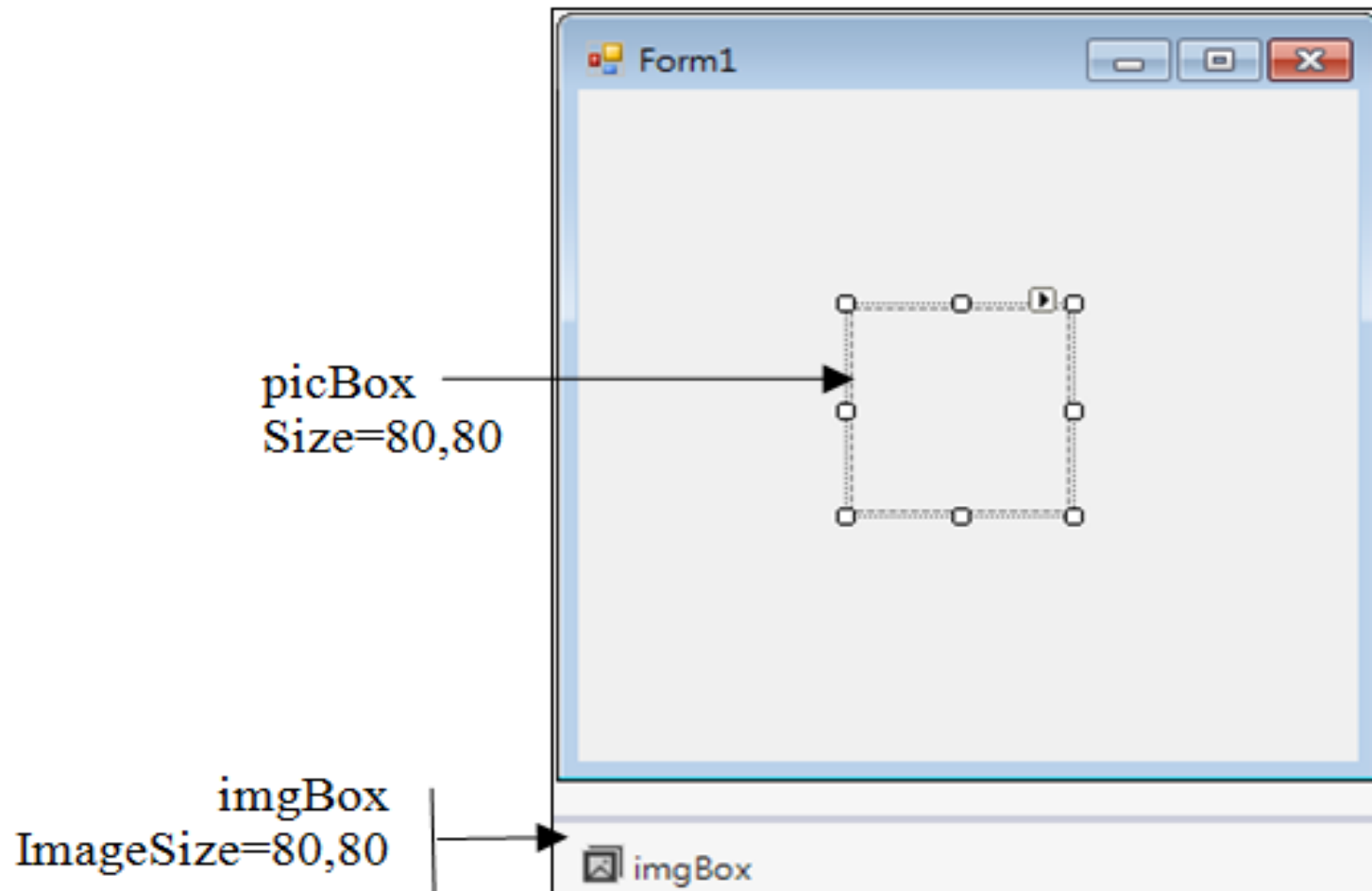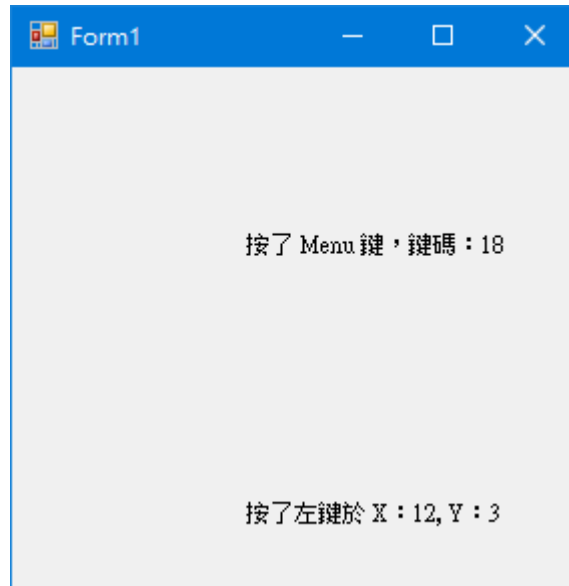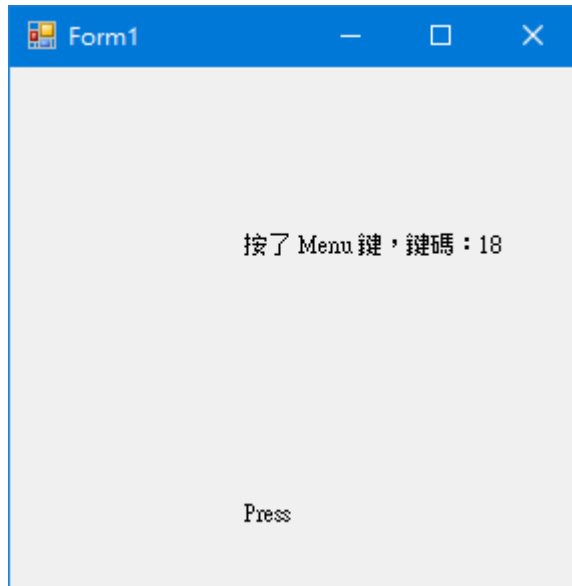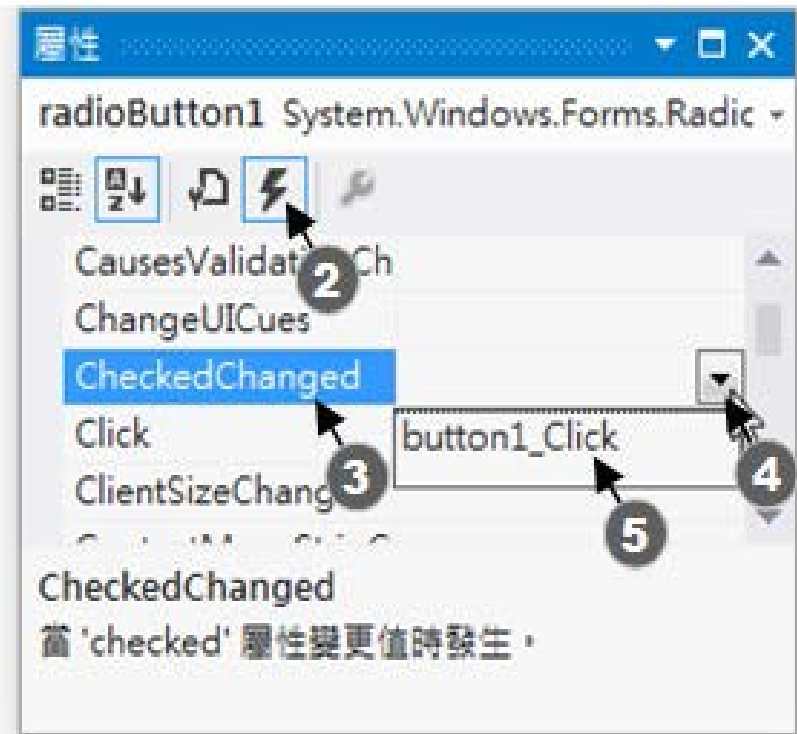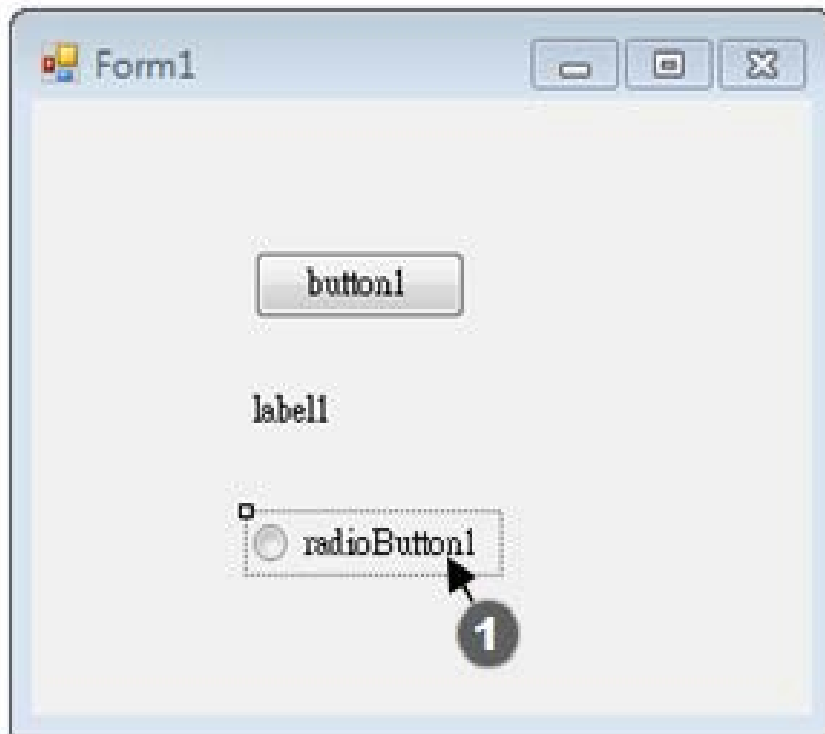
    this.label1.Click += new System.EventHandler(this.button1_Click);

    …

    this.radioButton1.CheckedChanged += new System.EventHandler(this.button1_Click);

    …

}
```

Assign Shared Event Handler in Runtime

Grammar

controlItem.event += eventHandler;
      or
controlItem.event += new EventHandler(eventHandler);

Ex: trigger Click event handler of button1 when button is clicked

```
1. button2.Click += button1_Click;
2. button2.Click += new EventHandler(button1_Click);
```

Ex: set CheckedChanged event of radioButton1 to trigger Click event handler of button1

```
1. radioButton1.CheckedChanged += button1_Click;
2. radioButton1.CheckedChanged += new EventHandler(button1_Click);
```

Ex: assign KeyPress event of textBox2 to share textBox1_KeyPress event handler

```
1. textBox2.KeyPress += textBox1_KeyPress;

2. textBox2.KeyPress += new KeyPressEventHandler(textBox1_KeyPress);
```

Ex: assign MouseDown event of textBox2 to share textBox1_MouseDown event handler

```
textBox2.MouseDown += textBox1_MouseDown;

textBox2.MouseDown += new MouseEventHandler(textBox1_MouseDown);
```

Disable sharing event

Grammar

controlItem.event -= eventHandler;
            or
controlItem.event -= new EventHandler(eventHandler);

Ex: remove button1_Click event handler shared by Click event of button2

1. button2.Click −= button1_Click;
2. button2.Click −= new EventHandler(button1_Click);

Share Customized Event Handler

Ex: Click event of button1 and Click event of button2 share myEvent event handler

```
button1.Click += myEvent;

button2.Click += myEvent;

       …

private void myEvent(object sender, EventArgs e)

{

    …

}
```

Decide which source control item is

To acquire which control item triggers event, first transform sender parameter into control item object, then use Equals method to decide.

Button button = (Button)sender means change sender into Button item.

Ex: Click event of button2 subscribe button1_Click event hander, to execute different program segment:

```
private void button1_Click(object sender, EventArgs e)
{
        Button btnClick = (Button)sender;
        if (btnClick.Equals(button1))
        {
                ...
        }
        else
        {
                ...
        }
}
```
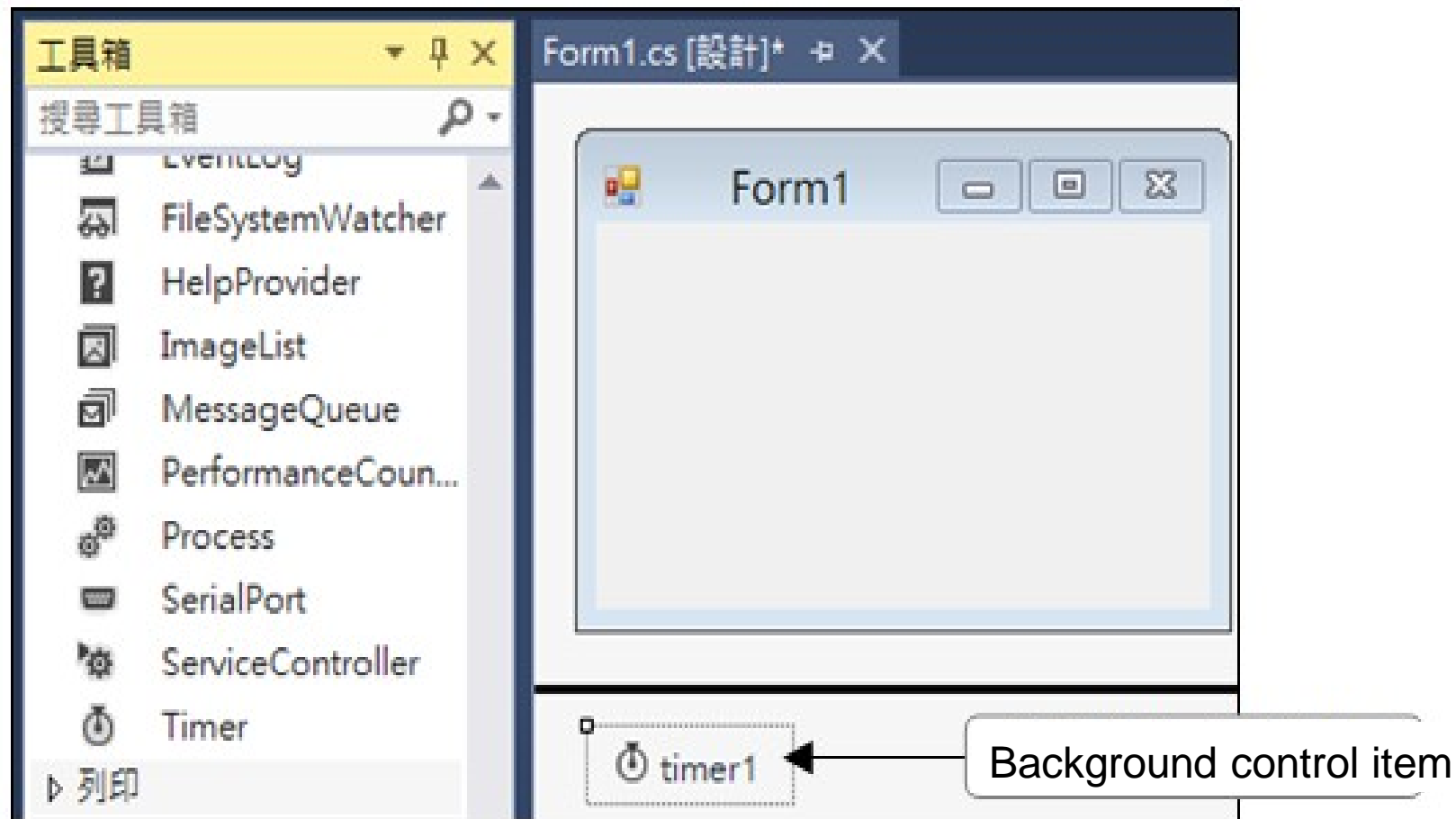
12-5 Timer Control Item

Usage

- **Change the displayed contents during a duration – creating an animation**

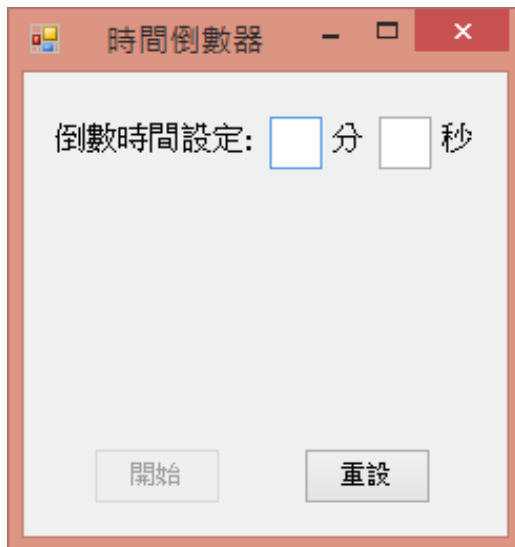- **Change the graph's position to create a movement**

Background control item

Property	Description
Enabled	False: inactivated (default) True: start counter, trigger the Tick event after the duration set by Interval property
Interval	How much time to trigger Tick event repeatedly, default: 100(0.1 sec). Do not use this timer if the Interval is 0

Event	Description
Tick	When the program is running, the property Enabled = True and the property Interval is not 0, the timer is activated. The event Tick is triggered as the timer arrives the duration set by Interval.

Example(Timer):

Design a timer program. 2 text boxes get the minute and second inputs. When the "開始" button is pressed, show the total seconds for count down in the non-border label control item. Requirements:
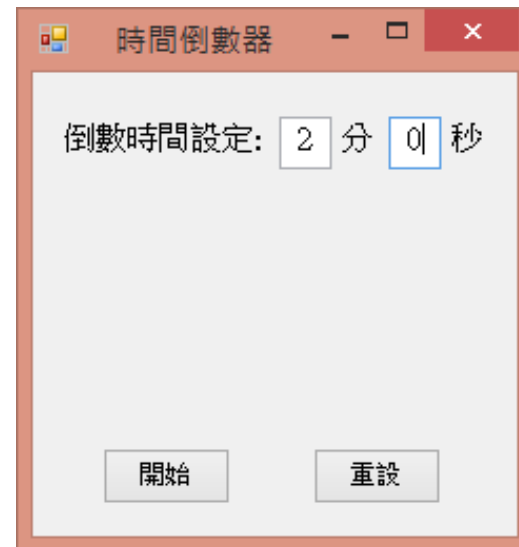
1. The minute and second inputs only allow unsigned integer and the scope is 0~59. The minute field can be blank. The input contents are right then the "開始" button is enabled, otherwise the button is disabled
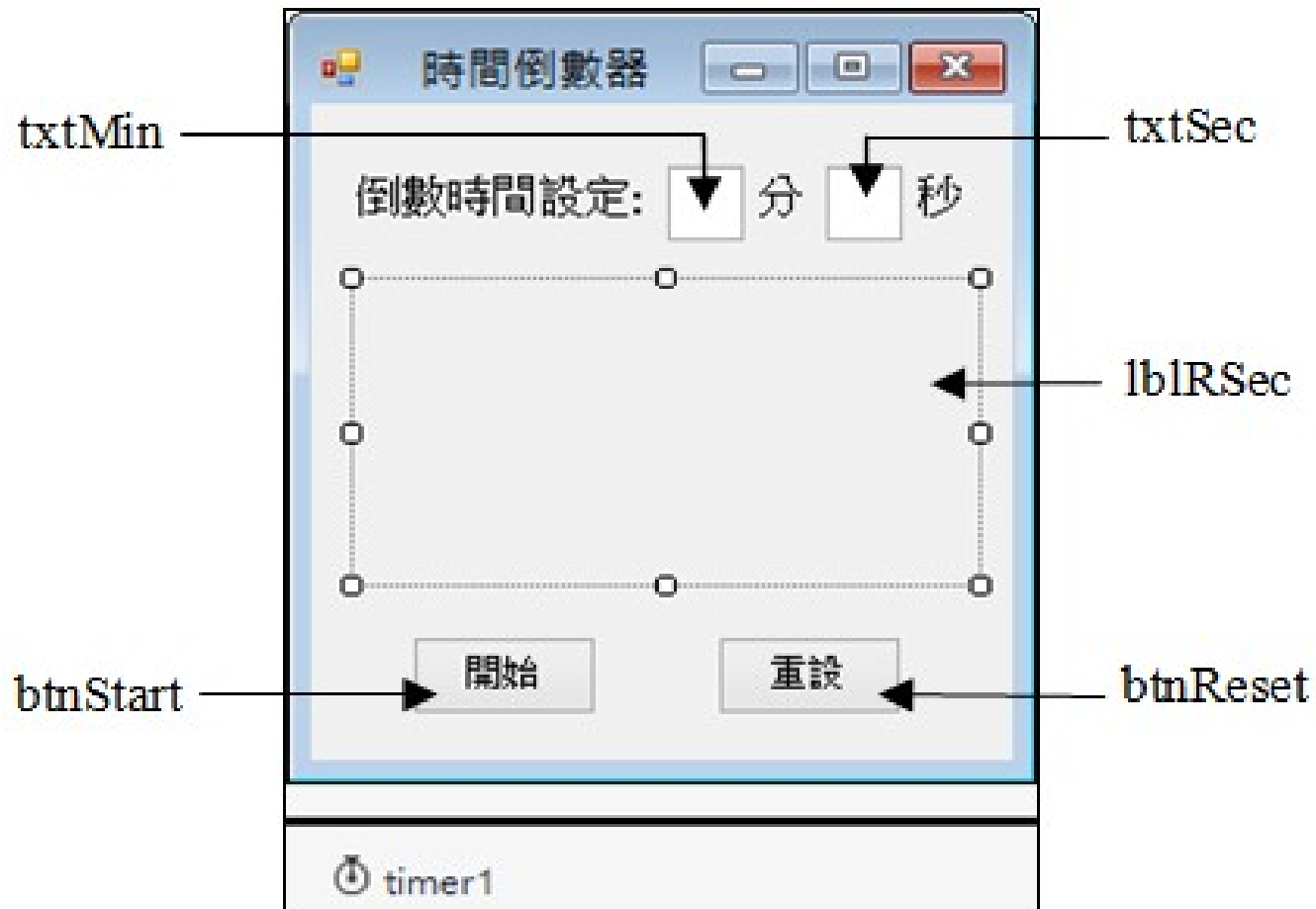
2. When the "開始" button is pressed, the label shows seconds for count down. Show the information message "時間到!" when count to 0. The text is larger and centered.

3. Button "重設" is available any time. The program returns to initial status when the button is pressed
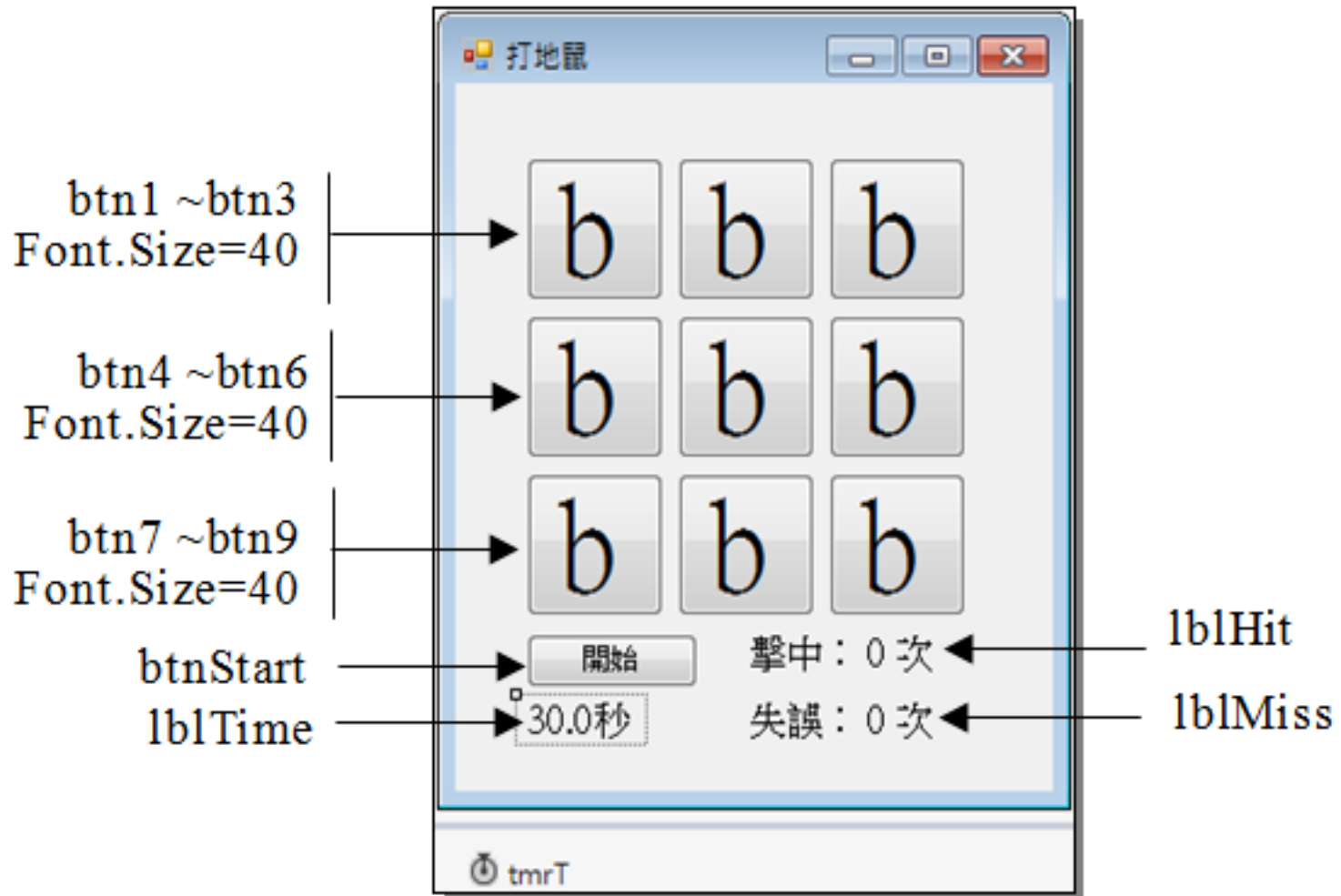
Design User Interface

Example(hit):

Design a hit-the-mode game, requirements:
1. When the program starts, there are 5 button but not available
2. Press "開始" button and the time starts counting down 30 sec, in the meanwhile, "開始" button is not available. Users hit X button and hit count pluses 1; users hit blank button and error count pluses 1.
3. If 30 sec is over, 9 buttons are not available, "開始" button can be used again

Result:

Design User Interface

The End

Take a Break   ..