Tutorial #5 – Where is the mouse?

Spirit Du

Ver. 1.0, 19th September, 2007

Ver. 1.5, 3rd November, 2008

Contents

About This Document

In this tutorial, two topics are introduced: (1) how to register two mouse event handlers (a hunting dog and a ring tied on the dog) to listen the mouse events, and (2) how to interact with mouse events. Note that the usage of mouse handlers will be used in the homework.

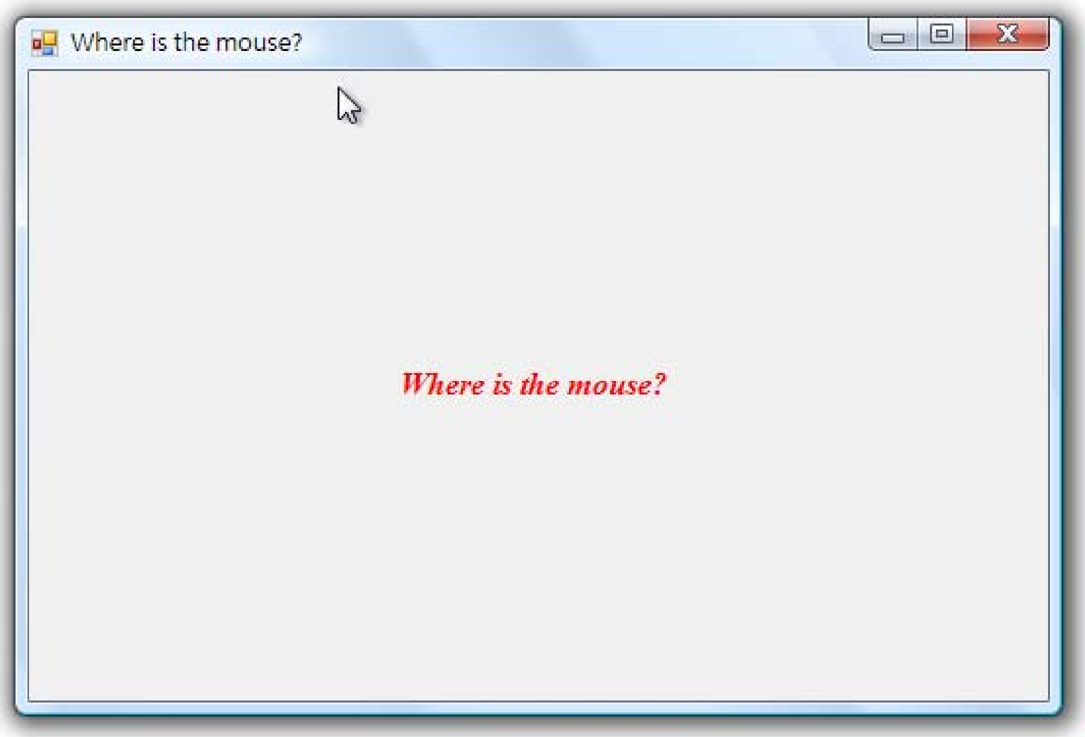

Figure 1 Interaction Application

Tutorial – Where is the mouse?

Step 1 Setup Environment

Refer to the last tutorial to setup an environment. At least, a class: MyForm which inherits Form is ready for advanced design. Please keep the setup method in mind that will be used several times because the Windows Form Designer is not used anymore in the following tutorials.

Step 2 Records

Before creating the application, we need some member variables to record information that will be used later. Please add the following codes into the proper location. Where? Try your best!

```
int _clickX = -1;
int _clickY = -1;
int _movement = 0;
bool _clicked = false;
String _message = "Where is the mouse?";
```

Step 3 Add a constructor

We use default constructor in the last tutorial to avoid confusing with the actual topic. However, creating a constructor at least is a best practice when you build object oriented programs. And the constructor is useful in later.

```
public MyForm() {
    Text = _message;
    SetClientSizeCore(640, 480);
}
```

Step 4 A Stage

In the tutorial, we need a stage: a big wild (640 * 480) within a running mouse. Create your owned overriding method; which method should be overridden? Fill the following codes in the overridden method to create the wild. Modify the entry class, and then execute the application. Click on the window to see what happened?

```
base.OnPaint(e);
Graphics g = e.Graphics;
Font font = new Font("Times New Roman", 12.0f, FontStyle.Bold);
using (font) {
    // Show the location if the mouse clicked on somewhere
    if (_clicked) {
        g.DrawEllipse(Pens.Blue, _clickX - 2, _clickY - 2, 4, 4);
        g.DrawEllipse(Pens.Blue, _clickX - 5, _clickY - 5, 10, 10);
        g.DrawEllipse(Pens.Blue, _clickX - 8, _clickY - 8, 16, 16);
        g.DrawString(_message, font, Brushes.Blue, _clickX, _clickY);
    }
    // Oops! The mouse disappeared!
    else {
        SizeF size = g.MeasureString(_message, font);
        int x = (int)(ClientSize.Width - size.Width) / 2;
        int y = (int)(ClientSize.Height - size.Height) / 2;
        g.DrawString(_message, font, Brushes.Red, x, y);
    }
}
```

Step 5 Hunter

Okay, nothing happened but just a string displayed in the center of the stage. Add a member method: MouseClicked() which is a hunter who catches the running mouse. Add the method into proper location. Then run again and click on the window to see what happed?

```
private void MouseClicked(object sender, MouseEventArgs e) {
    _clickX = e.X;
    _clickY = e.Y;
    _message = "(" + _clickX + ", " + _clickY + ")!";
    _clicked = true;
    Invalidate();
}
```

Step 6 Hunting Dog

Sorry, it is still nothing happened! Why? Because the hunter didn't go inside the wild, he cannot see any running mouse. In order to let the hunter be able to catch mice, we need a hunting dog which always barks loudly to let our hunter know where the mouse is. The dog should be put into the wild at beginning so where the hunting dog should be added? Make a guess!

```
// Listen to mouse clicked and moved events
MouseClick += new MouseEventHandler(MouseClicked);
```

Step 7 Hunter: I got you!

Run the application again and click on the window. Now, you can see that the hunter catches the mouse and show you where the mouse is like Figure 2.

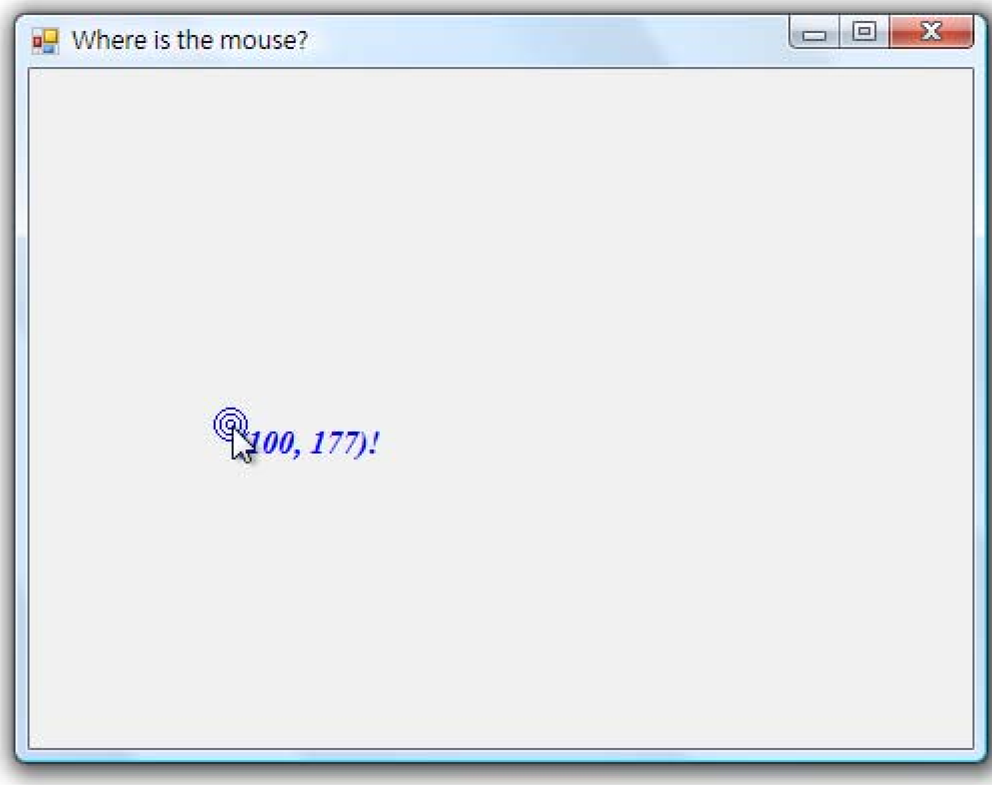

Figure 2 Hunter: I got you!

Step 8 Ring Ringing! Mouse: Hey, I'm still running...

It is not fair to the running mouse. We decide to tie a ring on the hunting dog so that the mouse can hide itself when a hunting dog is coming. Add the following codes into the proper location. Run the application and then click on the window or move the mouse in your hand to see what happened?

```
// Listen to mouse clicked and moved events
MouseMove += new MouseEventHandler(MouseMoved);
```

```
private void MouseMoved(object sender, MouseEventArgs e) {
    _movement = Math.Abs(e.X - _clickX) + Math.Abs(e.Y - _clickY);
    if (_movement > 75) {
        _clicked = false;
        _message = "Where is the mouse?";
        Invalidate();
    }
}
```

List 1 Mouse: Hey, I'm still running...

- The End -