Tutorial – Drawing & Mouse events

Spirit Du

Ver. 1.0, 18th September, 2007

Ver. 2.0, 3rd November, 2008

Ver. 3.0, 21th November, 2022

Contents

About This Document

In tutorial 1, two families of methods are introduced: Graphics.FillXXX() family and Graphics.DrawXXX() family. These two families of methods are the base of GUI. Moreover, no Windows Form designer is used that means everything is coded by the programmer's hand. A funny picture like Figure 1 is created.

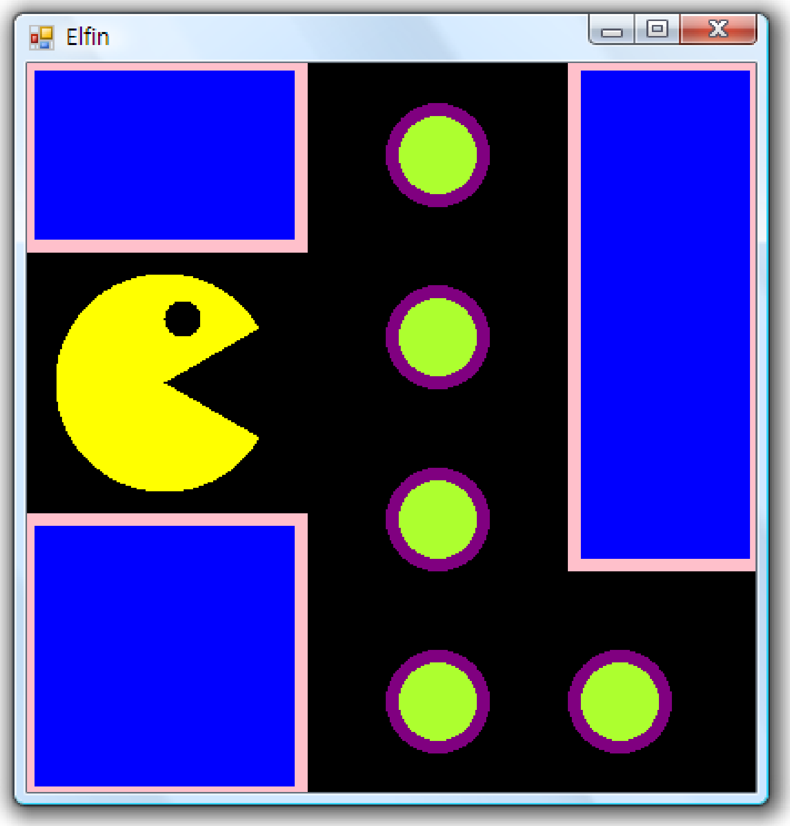

Figure 1 Elfin Game Map

In tutorial 2, 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.

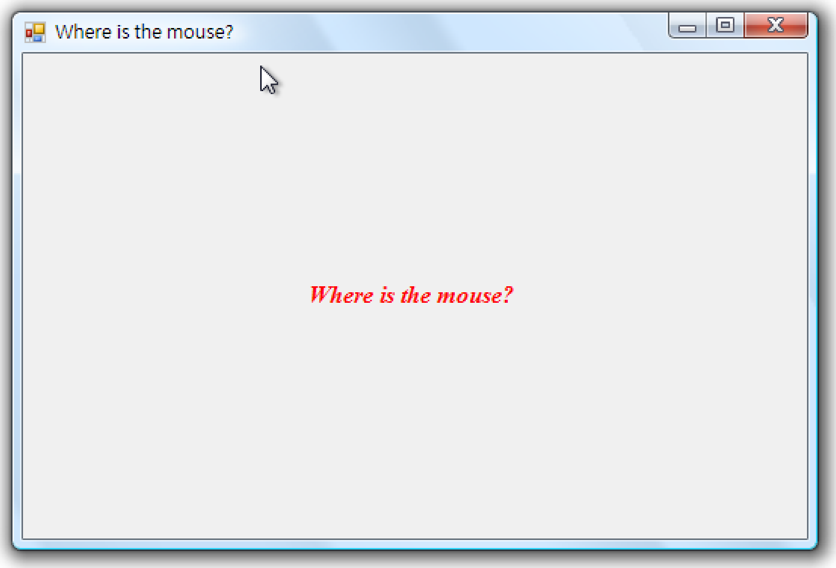

Figure 2 Interaction Application

Tutorial 1 – Elfin

Step 1 Setup Environment

Create a C# Windows Form App (.NET Framework) project named "Drawing". Add a new class: ElfinForm that inherits the Form class into the project with the following code.

```
using System.Windows.Forms;
using System.Drawing;
namespace Tutorial {
    class ElfinForm : Form {
        protected int WINDOW_WIDTH = 400;
        protected int WINDOW_HEIGHT = 400;
        protected int BALL_SIZE = 50;
        public ElfinForm() {
            SetClientSizeCore(WINDOW_WIDTH, WINDOW_HEIGHT);
            BackColor = Color.Black;
            Text = "Elfin";
        }
    }
}
```

Modify the Program.cs with the following code.

```
using System.Windows.Forms;

namespace Tutorial {
    class Program {
        static void Main(string[] args) {
            Form form = new ElfinForm();
            Application.Run(form);
        }
    }
}
```

Step 2 Fill Shapes with Brush

The outline of a Form is able to modify by overriding the OnPaint() method. When you get Graphics instance from PaintEventArgs object, you can do many things though its member methods, for examples, to fill shapes. You may find that when you want to fill shapes, you need a brush. However, you do not need to create any brush; you can find many brushes predefined in Brushes class.

```
Graphics.FillRectangle(Brush brush, int x, int y, int width, int height)
Graphics.FillEllipse(Brush brush, int x, int y, int width, int height)
Graphics.FillPie(Brush brush, int x, int y, int width, int height, int startAngle, int sweepAngle)
```

Figure 3 The FillXXX() family

Step 3 Make the Picture

Now, add a member method which override the OnPaint() method, and then we use those methods described in Step 2 to draw three blocks to identify the road, five circles as gems on the road. Then, we use FillPie() and FillEllipse() methods to draw the classic eflin.

```
protected override void OnPaint(PaintEventArgs e) {
    base.OnPaint(e);
    Graphics g = e.Graphics;

    // Three blue blocks
    g.FillRectangle(Brushes.Blue, 0, 0, 150, 100);
    g.FillRectangle(Brushes.Blue, 0, 250, 150, 150);
    g.FillRectangle(Brushes.Blue, 300, 0, 100, 275);

    // The elfin
    g.FillPie(Brushes.Yellow, 15, 115, 120, 120, 30.0f, 300.0f);
    g.FillEllipse(Brushes.Black, 75, 130, 20, 20);
    // Five balls
    g.FillEllipse(Brushes.GreenYellow, 200, 25, BALL_SIZE, BALL_SIZE);
    g.FillEllipse(Brushes.GreenYellow, 200, 125, BALL_SIZE, BALL_SIZE);
    g.FillEllipse(Brushes.GreenYellow, 200, 225, BALL_SIZE, BALL_SIZE);
    g.FillEllipse(Brushes.GreenYellow, 200, 325, BALL_SIZE, BALL_SIZE);
    g.FillEllipse(Brushes.GreenYellow, 300, 325, BALL_SIZE, BALL_SIZE);
}
```

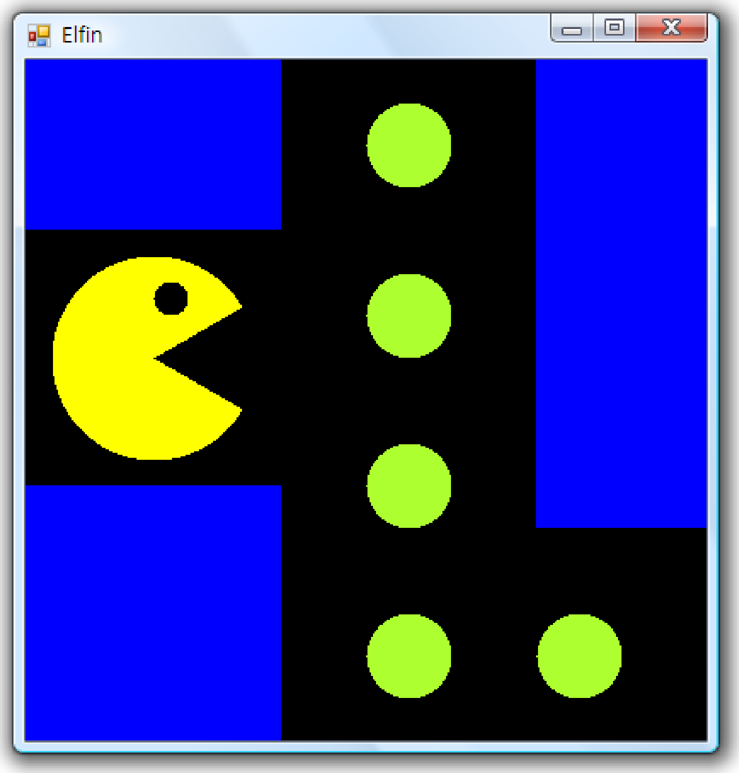

Figure 4 Semi-finished good

Step 4 Draw vs. Fill

Beside of FillXXX() methods which always draw shapes with background color, there are still many DrawXXX() methods which just sketch the outline of shape. There are different to FillXXX methods; they use a pen to sketch, not a brush. There is no FillLine() method because it is meaningless!

```
Graphics.DrawRectangle(Pen pen, int x, int y, int width, int height)
Graphics.DrawEllipse(Pen pen, int x, int y, int width, int height)
Graphics.DrawPie(Pen pen, int x, int y, int width, int height, int startAngle, int sweepAngle)
Graphics.DrawLine(Pen pen, int x1, int y1, int x2, int y2);
```

Figure 5 The DrawXXX() family

Step 5 A Thick Pen

In order to sketch the thick wall around the block, we create a thick pen instead of the system defined pen. Pen class has four constructors to create a pen depending on your needs. Here is a simple way to create a pen with specific color and thickness.

```
// Create a purple thick pen
Pen thickPen = new Pen(Color.Purple, 7.0f);
```

Step 6 Walls and Shells

At the last step, we use the thick pen as a parameter pass to DrawXXX method and draw walls and the shells of the gems. You should put the following codes after the codes of Step 3 in OnPaint() method.

```
// Give a border to each ball
g.DrawEllipse(thickPen, 200, 25, 50, 50);
g.DrawEllipse(thickPen, 200, 125, 50, 50);
g.DrawEllipse(thickPen, 200, 225, 50, 50);
g.DrawEllipse(thickPen, 200, 325, 50, 50);
g.DrawEllipse(thickPen, 300, 325, 50, 50);

// Change the color of pan to pink
thickPen.Color = Color.Pink;

// Create walls
g.DrawRectangle(thickPen, 0, 0, 150, 100);
g.DrawRectangle(thickPen, 0, 250, 150, 150);
g.DrawRectangle(thickPen, 300, 0, 100, 275);
```

- The Tutorial 1 End –

Tutorial 2 – Where is the mouse?

Step 1 Setup Environment

Refer to the last tutorial to setup an environment. Add a class: “MouseForm” which inherits Form is ready for advanced design. And then, replace ElfinForm to MouseForm in the Program.cs to open MouseForm.

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 MouseForm() {
    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 .

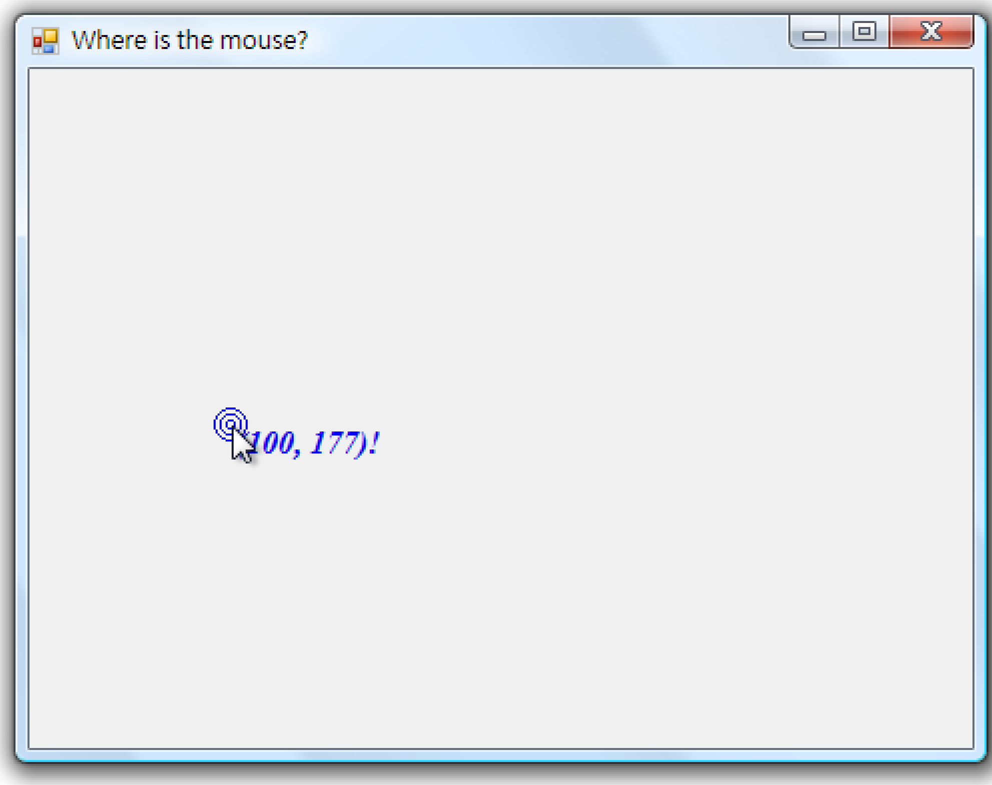

Figure 6 Hunter: I got you!

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();
    }
}
```

- The Tutorial 2 End -