

MÔN : LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài thực hành số 7.4 : Xây dựng chương trình đồng hồ có hình dạng tùy ý bằng cách lập trình động

I. Mục tiêu :

- Giúp SV làm quen với cách thức tạo form có hình dạng do ảnh bitmap qui định.

II. Nội dung :

- Lập trình động các thuộc tính liên quan của Form để nó có hình dạng do ảnh nền qui định.
- Hiện thực lại các chức năng trong các control button đã bị mất : đóng form, di dời form.

III. Chuẩn đầu ra :

- Sinh viên nắm vững và lập trình thành thạo các đoạn code để tạo form có hình dạng tùy ý.

IV. Qui trình :

1. Thực hiện lại bài thực hành 7.1 hay nhân bản thư mục chứa Project thực hành 7.1 thành Project NRF_Dongho_P.
2. Chạy VS .Net, chọn menu File.Open.Project rồi mở lại Project NRF_Dongho_P.
3. Ấn phải chuột vào mục Form1.cs trong cửa sổ Solution Explorer rồi chọn option View Designer để hiển thị lại cửa sổ thiết kế Form. Chọn Form, cửa sổ thuộc tính của nó sẽ hiển thị, duyệt tìm mục BackgroundImage và kiểm tra xem nó đã chứa ảnh nền đồng hồ chưa. Nếu chưa thì khai báo ảnh nền cho Form.
4. Tìm hàm Form1_Load và viết thêm đoạn code để thiết lập hình dạng của form theo ảnh nền như sau :

```
private void Form1_Load(object sender, EventArgs e)
{
    //tạo đối tượng quản lý assembly
    System.Reflection.Assembly myAssembly =
System.Reflection.Assembly.GetExecutingAssembly();
    //tạo đối tượng Stream miêu tả ảnh bitmap
    Stream myStream =
myAssembly.GetManifestResourceStream("Form_Dongho.Resources.DonghoTho.bmp");
    //tạo đối tượng ảnh bitmap nền
    bgimg = new Bitmap(myStream);
    //xác định màu nền cần lọc bỏ
    Color col = bgimg.GetPixel(1, 1);
    //thiết lập hình dạng Form ứng dụng theo ảnh bitmap
    this.Region = ConvertB2R(bgimg, col);
    //thiết lập ảnh nền cho Form
    this.BackgroundImage = bgimg;
    //ẩn tittle bar và đường viền của form
    this.FormBorderStyle = FormBorderStyle.None;
}
```

5. Viết thêm 2 hàm dịch vụ để xây dựng Region cho Form theo ảnh nền :

```
//hàm kiểm tra 1 pixel ảnh có trùng màu nền qui định không
bool Equal(Byte[] pbase, int idx, Color key)
{
    if (pbase[idx] != key.B) return false;
    if (pbase[idx + 1] != key.G) return false;
    if (pbase[idx + 2] != key.R) return false;
```

```

    if (pbase[idx + 3] != key.A) return false;
    return true;
}


//hàm chuyển ảnh bitmap thành Region qui định đường viền của ảnh
Region ConvertB2R(Bitmap bitmap, Color Key)
{
    GraphicsUnit unit = GraphicsUnit.Pixel;
    //xác định kích thước ảnh nền
    RectangleF boundsF = bitmap.GetBounds(ref unit);
    Rectangle bounds = new Rectangle((int)boundsF.Left, (int)boundsF.Top,
(int)boundsF.Width, (int)boundsF.Height);
    //xác định dữ liệu của ảnh nền
    BitmapData bitmapData = bitmap.LockBits(bounds, ImageLockMode.ReadOnly,
PixelFormat.Format32bppArgb);
    //xác định địa chỉ của hàng pixel đầu tiên
    IntPtr ptr = bitmapData.Scan0;
    //xác định độ dài vùng nhớ chứa toàn bộ pixel ảnh (24 bits per pixels).
    int bytes = Math.Abs(bitmapData.Stride) * bitmap.Height;
    //copy vùng chứa toàn bộ pixel ảnh vào pbase
    byte[] pbase = new byte[bytes];
    System.Runtime.InteropServices.Marshal.Copy(ptr, pbase, 0, bytes);
    //xác định số hàng, cột của ảnh bitmap
    int yMax = (int)boundsF.Height;
    int xMax = (int)boundsF.Width;
    //tạo đối tượng Path chứa các path miêu tả ảnh nền
    GraphicsPath path = new GraphicsPath();
    int isrow = 0;
    int idx = 0;
    int x, y;
    for (y = 0; y < yMax; y++)
    { //duyet xử lý từng hàng pixel
        idx = isrow; //setup offset vùng nhớ chứa pixel hiện hành
        for (x = 0; x < xMax; x++)
        {
            //nếu pixel hiện hành trùng màu nền thì bỏ qua
            if (Equal(pbase, idx, Key))
            {
                idx = idx + 4; //hiệu chỉnh offset vùng nhớ chứa pixel hiện hành
                continue;
            }
            //nhớ vị trí pixel lẻ trái vùng ảnh tìm được
            int x0 = x;
            //duyet tìm lẻ phải vùng ảnh tìm được
            while (x < xMax && (!Equal(pbase, idx, Key)))
            {
                x = x + 1;
                idx = idx + 4;
            }
            //add path chữ nhật miêu tả đoạn pixel ảnh tìm được
            path.AddRectangle(new Rectangle(x0, y, x - x0, 1));
        }
    }
}

```

```

        //hiệu chỉnh offset vùng nhớ chứa hàng pixel kế tiếp
        isrow = isrow + bitmapData.Stride;
    }
    //tạo Region được miêu tả bởi đối tượng Path vừa tìm được
    Region region = new Region(path);
    //dọn dẹp
    path.Dispose();
    bitmap.UnlockBits(bitmapData);
    //trả về region tìm được
    return region;
}

```

6. Dời về đầu file, thêm 2 lệnh using sau đây để dùng các dịch vụ về xử lý ảnh :
`using System.Drawing.Imaging;`
`using System.Drawing.Drawing2D;`
7. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Xem kết quả và đánh giá kết quả. Ta thấy hiện nay Form có hình dạng do ảnh nền qui định nhưng không còn titlebar và các button điều khiển nên không thể di dời hay đóng Form trực quan được. Ta sẽ thực hiện tiếp 1 số việc để giải quyết vấn đề này.
8. Hãy đóng Form lại (bằng cách chọn option Debug.Stop Debugging hay chọn option “Close window” trong menu pop-up của icon của ứng dụng trên thanh taskbar).
9. Ấn phải chuột vào mục Form1.cs trong cửa sổ Solution Explorer rồi chọn option View Designer để hiển thị lại cửa sổ thiết kế Form. Chọn Form, cửa sổ thuộc tính của nó sẽ hiển thị, click icon  để hiển thị danh sách các sự kiện của Form, duyệt tìm lần lượt các sự kiện MouseDown, MouseUp, MouseMove và tạo hàm xử lý sự kiện tương ứng. Viết code cụ thể cho 3 hàm vừa tạo như sau :

```

private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left) //chỉ chấp nhận ấn chuột trái
    {
        //kiểm tra việc click chuột để đóng form
        if (Math.Abs(e.X - xC) < EPSI && Math.Abs(e.Y - yC) < EPSI) Close();
        //drag chuột : lưu giữ vị trí chuột và vị trí Form tại lúc bắt đầu drag
        mouseDownPos = Control.MousePosition;
        formDownPos = Location;
        //ghi nhận trạng thái giữ chuột
        isMouseDown = true;
    }
}

private void Form1_MouseUp(object sender, MouseEventArgs e)
{
    if (e.Button == MouseButtons.Left) //chỉ chấp nhận thả chuột trái
        //ghi nhận trạng thái thả chuột
        isMouseDown = false;
}

private void Form1_MouseMove(object sender, MouseEventArgs e)
{
    if (isMouseDown)
    {

```

```

//xác định vị trí chuột hiện hành
Point mousePos = Control.MousePosition;
//tính độ dời chuột so với vị trí bắt đầu drag
int dx = mousePos.X - mouseDownPos.X;
int dy = mousePos.Y - mouseDownPos.Y;
//thiết lập vị trí mới cho Form theo yêu cầu drag
Location = new Point(formDownPos.X + dx, formDownPos.Y + dy);
}
}

```

10. Dời cursor nhập liệu về đầu class Form1 rồi viết thêm các lệnh định nghĩa các thuộc tính cần dùng để quản lý chuột như sau :

```

private Point formDownPos; //vị trí Form lúc bắt đầu drag chuột
private Point mouseDownPos; //vị trí chuột lúc bắt đầu drag chuột
private Boolean isMouseDown = false;
//tọa độ tâm của button Close form
int xC = 94;
int yC = 43;
//sai số chấp nhận so với tâm của button Close
const int EPSI = 10;

```

11. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Thử đóng các cửa sổ khác và để duy nhất Form đồng hồ hiển thị trên nền desktop. Xem kết quả hiển thị, thử drag mouse để dời Form đồng hồ. Click thử vào vị trí mất phải của thỏ (đại diện button Close) xem form có đóng lại không.