

## MÔN : LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

### Bài thực hành số 11.3 : Xây dựng Owner-drawn control và ứng dụng dùng nó

#### I. Mục tiêu :

- Giúp SV làm quen với cách thức xây dựng 1 Owner-drawn control giải quyết chức năng đặc thù của mình bằng cách thừa kế điều khiển đã có sẵn có cùng tính năng nhưng giao diện chưa phù hợp với nhu cầu riêng.

#### II. Nội dung :

- Xây dựng button HeartButton, nó là Button nhưng bộ mặt không phải là hình chữ nhật có đường viền thông thường mà là một trái tim màu đỏ tươi.

#### III. Chuẩn đầu ra :

- Sinh viên nắm vững và xây dựng thành thạo các Owner-drawn Control có chức năng đặc thù cũng như xây dựng ứng dụng dùng lại các Owner-drawn Control.

#### IV. Qui trình :

##### IV.1 Xây dựng Owner-drawn control

- Chạy VS .Net, chọn menu File.Open.Project để hiển thị cửa sổ duyệt file. Duyệt và tìm file \*.sln quản lý Project "Windows Control Library" có sẵn từ bài thực hành 9.2 để mở lại Project này.
- Quan sát cây Project, chúng ta đã thấy có mục LoginControl.cs quản lý user control đã xây dựng từ bài thực hành 9.1, mục MyTextBox.cs quản lý inherited control đã xây dựng từ bài thực hành 9.2. Ấn phải chuột vào gốc của cây Project trong cửa sổ "Solution Explorer", chọn option Add.User Control để tạo mới 1 User Control, nhập tên là HeartButton.cs rồi click button Add để tạo mới nó.
- Lúc này control mới chỉ là 1 vùng hình chữ nhật trống. Dời chuột về mục HeartButton.cs trong cửa sổ Project, ấn phải chuột trên nó rồi chọn option "View Code" để hiển thị cửa sổ soạn mã nguồn cho HeartButton control.
- Hiệu chỉnh lại lệnh định nghĩa class HeartButton để thừa kế class Button (thay vì UserControl như mặc định). Nội dung chi tiết của class HeartButton như sau :

```
namespace MyUserControls {
    public partial class HeartButton : Button {
        public HeartButton() {
            InitializeComponent();
        }

        //hàm kiểm tra 1 pixel ảnh có trùng màu 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 được qui định bởi đường viền của ảnh
        Region ConvertB2R(Bitmap bitmap, Color Key)
        {
            // Dim modeFlag As Boolean = (mode = TransparencyMode.ColorKeyOpaque)
```

```

GraphicsUnit unit = GraphicsUnit.Pixel;
RectangleF boundsF = bitmap.GetBounds(ref unit);
Rectangle bounds = new Rectangle((int)boundsF.Left, (int)boundsF.Top,
(int)boundsF.Width, (int)boundsF.Height);
//get access to the raw bits of the image
BitmapData bitmapData = bitmap.LockBits(bounds, ImageLockMode.ReadOnly,
PixelFormat.Format32bppArgb);
//Get the address of the first line.
IntPtr ptr = bitmapData.Scan0;
//Declare an array to hold the bytes of the bitmap.
//This code is specific to a bitmap with 24 bits per pixels.
int bytes = Math.Abs(bitmapData.Stride) * bitmap.Height;
byte[] pbase = new byte[bytes];
//Copy the RGB values into the array.
System.Runtime.InteropServices.Marshal.Copy(ptr, pbase, 0, bytes);
//avoid property accessors in the for
int yMax = (int)boundsF.Height;
int xMax = (int)boundsF.Width;
//to store all the little rectangles in
GraphicsPath path = new GraphicsPath();
int isrow = 0;
int idx = 0;
int x, y;
for (y = 0; y <= yMax - 1; y++)
{
    idx = isrow;
    for (x = 0; x <= xMax - 1; x++)
    {
        //is this transparent? if ((yes, just go on with the loop
        if (Equal(pbase, idx, Key))
        {
            idx = idx + 4;
            continue;
        }
        //store where the scan starts
        int x0 = x;
        //not transparent - scan until we find the next transparent byte
        while (x < xMax && (!Equal(pbase, idx, Key)))
        {
            x = x + 1;
            idx = idx + 4;
        }

        //add the rectangle we have found to the path
        path.AddRectangle(new Rectangle(x0, y, x - x0, 1));
    }
    //jump to the next line
    isrow = isrow + bitmapData.Stride;
}
//now create the region from all the rectangles
Region region = new Region(path);
//clean up

```

```

    path.Dispose();
    bitmap.UnlockBits(bitmapData);
    return region;
}

//tác vụ vẽ bộ mặt giao diện của Button
protected override void OnPaint(PaintEventArgs e) {
    //tạo đối tượng image gốc chứa ảnh trái tim màu đỏ
    Bitmap bm = (Bitmap)Bitmap.FromFile("d:\\bgbutton.bmp");
    //tạo đối tượng image chứa ảnh trái tim màu đỏ theo kích thước của Control
    Bitmap newBitmap = new Bitmap(ClientSize.Width, ClientSize.Height);
    Graphics g = Graphics.FromImage(newBitmap);
    g.DrawImage(bm, new Rectangle(0, 0, newBitmap.Width, newBitmap.Height), new
Rectangle(1, 1, bm.Width-2, bm.Height-2), GraphicsUnit.Pixel);
    //xác định đối tượng vẽ của Button
    g = e.Graphics;
    //vẽ image gốc theo chế độ zoom vào button
    g.DrawImage(newBitmap, 0, 0);
    Color col = newBitmap.GetPixel(1, 1);
    this.Region = ConvertB2R(newBitmap, col);
    //this.BackgroundImage = bm;
    //this.FormBorderStyle = FormBorderStyle.None;
    //định nghĩa đối tượng miêu tả cách thức hiển thị chuỗi
    StringFormat format1 = new StringFormat(StringFormatFlags.NoClip);
    format1.Alignment = StringAlignment.Center;
    //vẽ chuỗi caption của button
    g.DrawString(this.Text, this.Font, Brushes.White, this.Width / 2, this.Height / 3,
format1);
}
}
}


```

5. Chọn menu Build.Build Solution để dịch và tạo file thư viện chứa các user control. Nếu có lỗi thì sửa và dịch lại. Lưu ý khi máy báo lỗi ở hàng lệnh `this.AutoScaleMode = ...` thì hãy chú thích hàng lệnh này hay xóa nó luôn cũng được.
6. Nếu dịch thành công, file thư viện có tên là `MyUserControls.dll` sẽ được tạo ra trong thư mục con Debug (hay Release tùy chế độ dịch) trong thư mục chứa Project. Ta nên copy file này vào thư mục chung chứa các file thư viện để sau này dùng tiện lợi hơn.

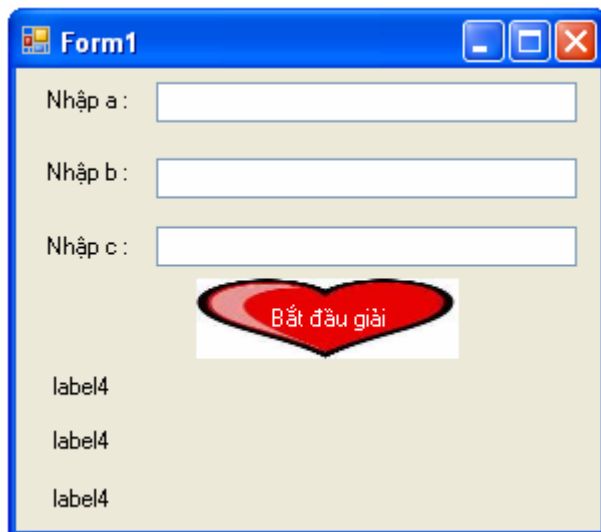
## IV.2 Xây dựng ứng dụng dùng Owner-drawn Control



Trong chương 5, chúng ta đã xây dựng chương trình giải phương trình bậc 2 dùng giao diện đồ họa trực quan. Chúng ta đã dùng 3 textbox để nhập liệu và 1 button để khởi động việc giải phương trình và hiển thị kết quả. Bây giờ chúng ta hãy hiệu chỉnh lại chương trình giải phương trình bậc 2 để dùng 3 đối tượng `MyTextBox` và 1 đối tượng `HeartButton`.

1. Nhân bản thư mục `VCGPTB2` (có được trong bài thực hành 5.1) thành thư mục mới tên là `UseHeartButton`.
2. Chạy VS .Net, chọn menu `File.Open.Project` để hiển thị cửa sổ duyệt file. Duyệt và tìm file `VCGPTB2.sln` trong thư mục `UseHeartButton` để mở lại Project này.
3. Dời chuột về mục `Form1.cs` của cây Project trong cửa sổ "Solution Explorer", ấn kép chuột vào nó để hiển thị lại cửa sổ thiết kế trực quan cho Form chương trình.
4. Chọn và xóa 3 textbox và button có sẵn.

5. Nếu cửa sổ ToolBox chưa hiển thị chi tiết, chọn menu View.Toolbox để hiển thị nó (thường nằm ở bên trái màn hình). Click chuột vào button  (Auto Hide) nằm ở góc trên phải cửa sổ Toolbox để chuyển nó về chế độ hiển thị thường trực. Dời chuột về nhóm Components trong cửa sổ Toolbox, ấn phải chuột trên nó để hiển thị menu lệnh, chọn option "Choose Items". Khi cửa sổ "Choose Toolbox Items" hiển thị, click chuột vào button Browse để hiển thị cửa sổ duyệt tìm file, hãy duyệt tìm đến thư mục chứa file MyUserControls.dll vừa xây dựng được trong phần đầu của bài thực hành, chọn file dll rồi click button OK để "add" các usercontrol trong thư viện này vào nhóm Components của cửa sổ Toolbox của Project ứng dụng.
6. Duyệt tìm phần tử MyTextBox (thường ở cuối nhóm Components), chọn nó, dời chuột về bên phải Label "Nhập a:" và vẽ nó với kích thước mong muốn. Vào cửa sổ thuộc tính của đối tượng MyTextBox vừa vẽ, đặt thuộc tính (Name) = txtA, thuộc tính ValidateFor = NumReal để nó hoạt động ở chế độ nhập số thực.
7. Lặp lại bước 6 hai lần để tạo thêm 2 đối tượng MyTextBox khác, các MyTextBox tạo mới lần lượt có thuộc tính (Name) = txtB, txtC còn thuộc tính ValidateFor = NumReal.  
Đối với các đối tượng giống nhau, ta có thể dùng kỹ thuật Copy-Paste để nhân bản vô tính chúng cho dễ dàng.
8. Duyệt tìm phần tử HeartButton (thường ở cuối nhóm Components), chọn nó, dời chuột về ngay dưới MyTextBox txtC và vẽ nó với kích thước mong muốn. Vào cửa sổ thuộc tính của đối tượng HeartButton vừa vẽ, đặt thuộc tính (Name) = btnStart, thuộc tính Text = "Bắt đầu giải".

Sau khi thiết kế xong, form ứng dụng có dạng sau :



9. Chọn lại button để hiển thị cửa sổ thuộc tính của nó, click icon  để hiển thị danh sách các sự kiện của đối tượng HeartButton, duyệt tìm sự kiện quan tâm (Click), click chuột vào dấu  bên phải sự kiện Click để máy hiển thị danh sách các hàm xử lý đã có, chọn lại hàm xử lý có tên là btnStart().
10. Chọn menu Debug.Start Debugging để dịch và chạy ứng dụng. Hãy thử nhập từng bộ ba (a,b,c) của phương trình bậc 2 rồi ấn button "Bắt đầu giải" để giải và xem kết quả.