

Bài 8: Graphics – GDI+

Lương Trần Hy Hiến

FIT, HCMUP

Lập trình Windows Form với C#

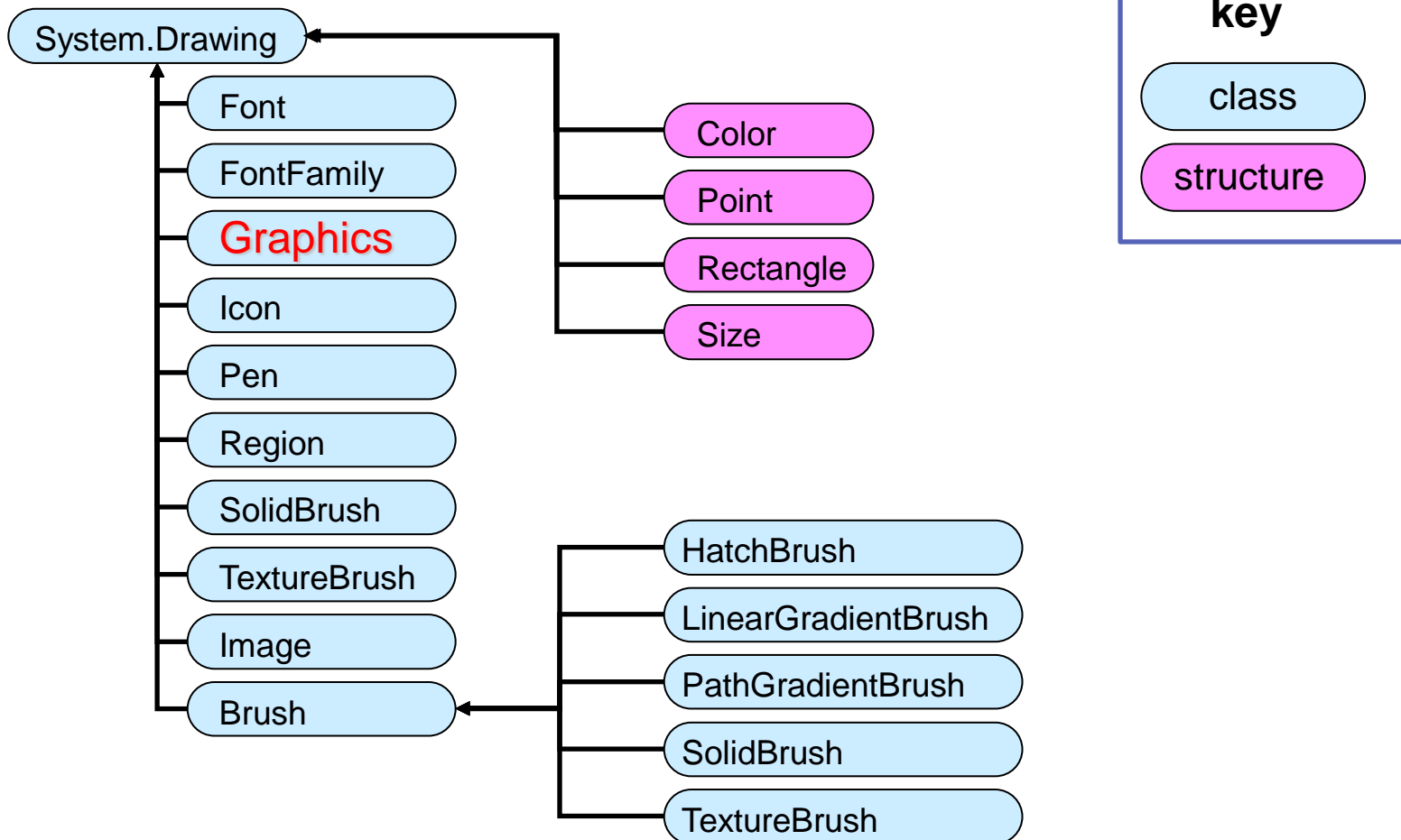
Nội dung

- **GDI+**
- **Chương trình vẽ cơ bản trên Form**
- **Lớp Graphics và hàm OnPaint()**
- **Lớp Color và Font**
- **Lớp Pen và Brush**
- **Các hàm vẽ đường thẳng, hình chữ nhật, ellipse**
- **Các hàm vẽ cung, đa giác**
- **Hiển thị ảnh**
- **Minh họa Multimedia**

Tổng quan

- Thư viện giúp “vẽ” lên màn hình hoặc máy in mà không cần quan tâm đến cấu trúc phần cứng → độc lập thiết bị
- GDI+ bao gồm 3 nhóm “dịch vụ” chính:
 - **2D vector graphics**: cho phép tạo hình từ các hình cơ bản (primitive): đường thẳng, tròn, eclipse, đường cong,...
 - **Imaging**: làm việc với các tập tin hình ảnh (bitmap, metafile)
 - **Typography**: vẽ chữ

System.Drawing



Vẽ trên Form

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    protected override void OnPaint(PaintEventArgs e)
    {
        Graphics g = e.Graphics;
        g.DrawString("Hello GDI!", Font, Brushes.Red, 20, 20);
    }
}
```

Graphics



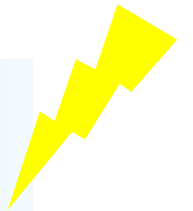
- **Lớp Graphics thể hiện**
 - “*Abstract*” drawing surface
 - Tập hợp những “*tool*” cho phép thao tác trên surface đó
- **Để lấy đối tượng Graphics**
 - Sử dụng thuộc tính Graphics được truyền cho **OnPaint()**
 - Sử dụng phương thức **CreateGraphics()** của control
 - Lấy từ đối tượng dẫn xuất từ Bitmap
- **Gọi hàm **Invalidate()** thay vì OnPaint()**

Lấy đối tượng Graphics

```
protected override void OnPaint(PaintEventArgs paintevent)
{
    Graphics graf=paintevent.Graphics;
}
```

Từ tham số PaintEventAtgs

```
private void mainForm_Paint(object sender, PaintEventArgs
    paintevent)
{
    Graphics graf=paintevent.Graphics;
}
```



Lấy đối tượng Graphics

```
private void PaintMe(Control testcontrol)
{
    Graphics graf=testcontrol.CreateGraphics();
    ...
}
```

Lấy từ control



```
protected override void OnPaint(PaintEventArgs paintevent)
{
    Bitmap bmpimage=new Bitmap("hoalan.jpg");
    Graphics graf = Graphics.FromImage (bmpimage);
    ...
}
```

Lấy từ ảnh



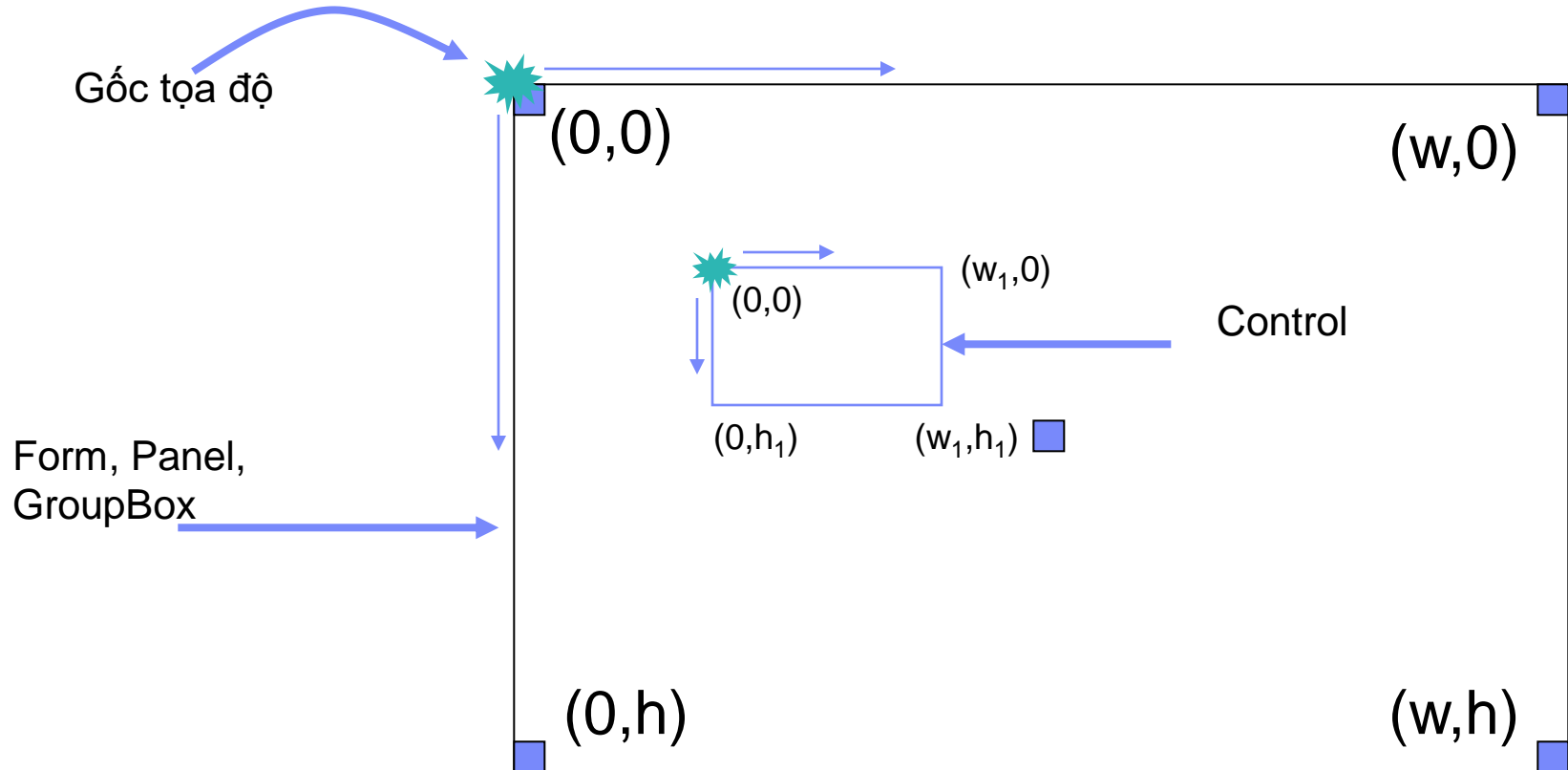
DrawString() method

- **Hiển thị text trong Graphics cụ thể**
 - Có nhiều phiên bản

```
DrawString( String text,           // Text thể hiện  
            Font f,                // Font  
            Brush b,               / Color & texture  
            float x, float y);     // vị trí góc trái trên
```

- Tham số Font và Brush không có mặc định nên phải truyền vào.

Tọa độ hệ thống



Tọa độ hệ thống

- **Graphics.PageUnit:** xác định đơn vị của bề mặt
 - GraphicsUnit.Pixel (default)
 - GraphicsUnit.Inch
 - GraphicsUnit.Milimeter
 - GraphicsUnit.Point
- **Graphics.PageScale:** tỷ lệ output
 - g.PageScale = 1f (default)

Color

- **Sử dụng màu được định nghĩa trong Color**
 - Color.Blue, Color.Red, Color.White...
- **Sử dụng màu định nghĩa cho hệ thống**
 - SystemColors.Control, SystemColors.ControlText...
- **Sử dụng màu ARGB**
 - 32 bit để thể hiện màu
 - A (alpha) thể hiện mức độ trong suốt (255 opaque)
 - RGB là Red, Green và Blue
 - Tạo màu sử dụng hàm FromArgb()
 - Color red = Color.FromArgb(255,0,0);
 - Color blue = Color.FromArgb(128, 0, 255, 0);

Tóm tắt một số hằng số Color

Hằng số trong cấu trúc Color	Giá trị RGB
Orange	255, 200, 0
Pink	255, 175, 175
Cyan (Green Blue)	0, 255, 255
Magenta (Red Blue)	255, 0, 255
Yellow	255, 255, 0
Black	0, 0, 0
White	255, 255, 255
DarkGray	64, 64, 64
Red	255, 0, 0
Green	0, 255, 0
Blue	0, 0, 255

Sử dụng **ColorDialog** để chọn màu từ bảng màu

- Tạo đối tượng **ColorDialog**

```
ColorDialog colorChooser = new  
ColorDialog();
```

- Hàm hiển thị Dialog:

```
colorChooser.ShowDialog();
```

Thuộc tính **FullOpen** của **ColorDialog**

```
public virtual bool FullOpen { set; get; }
```

- Tóm tắt:

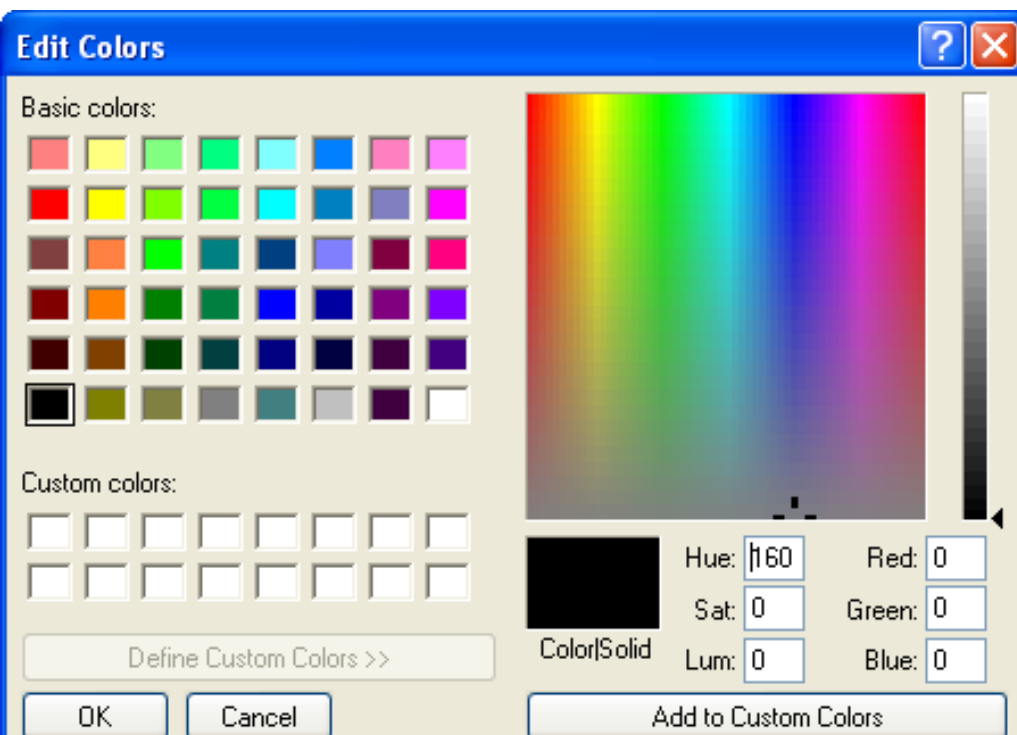
Gets or **sets** một giá trị chỉ ra rằng liệu control sử dụng để tạo ra màu riêng có hiện ra hay không khi hộp dialog được mở

- Return:

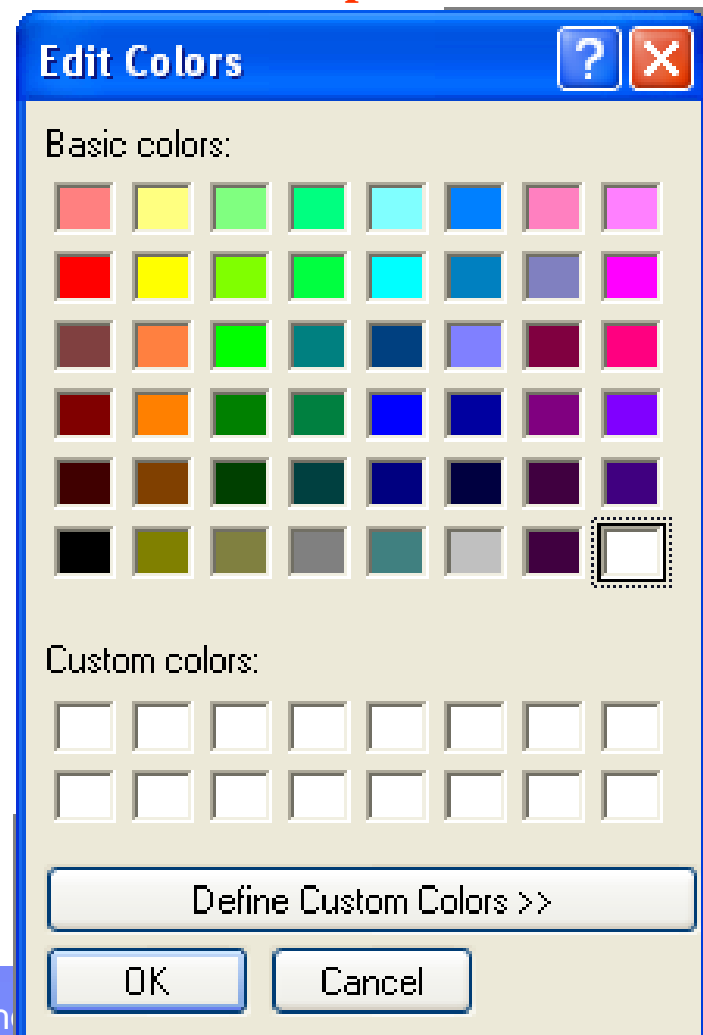
true nếu control màu riêng thì có sẵn khi hộp dialog được mở; ngược lại, **false**. Giá trị mặc định là **false**.

FullOpen

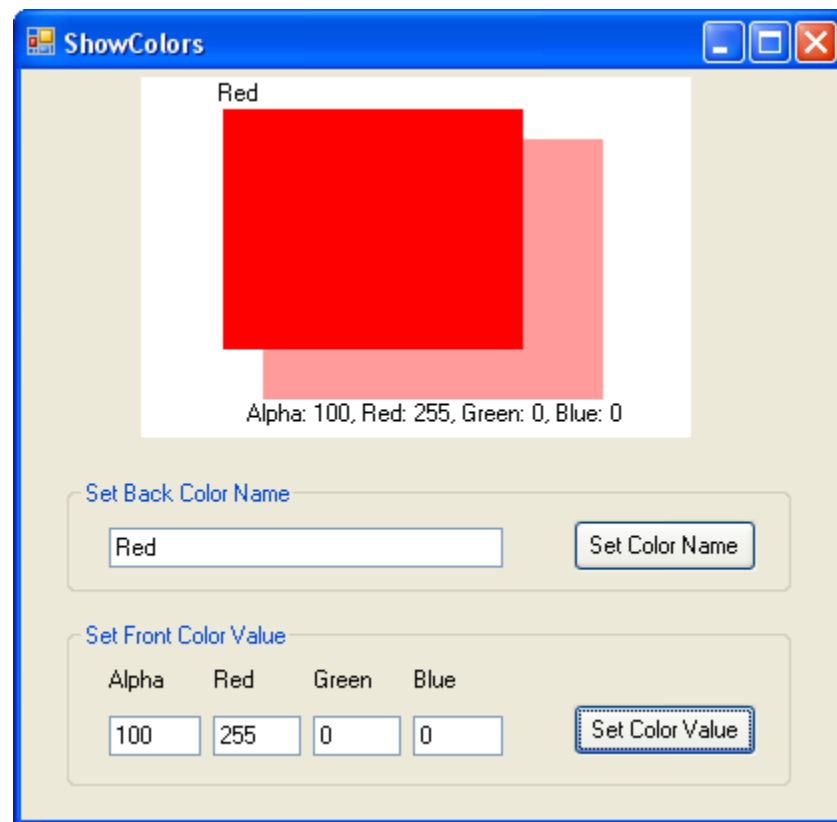
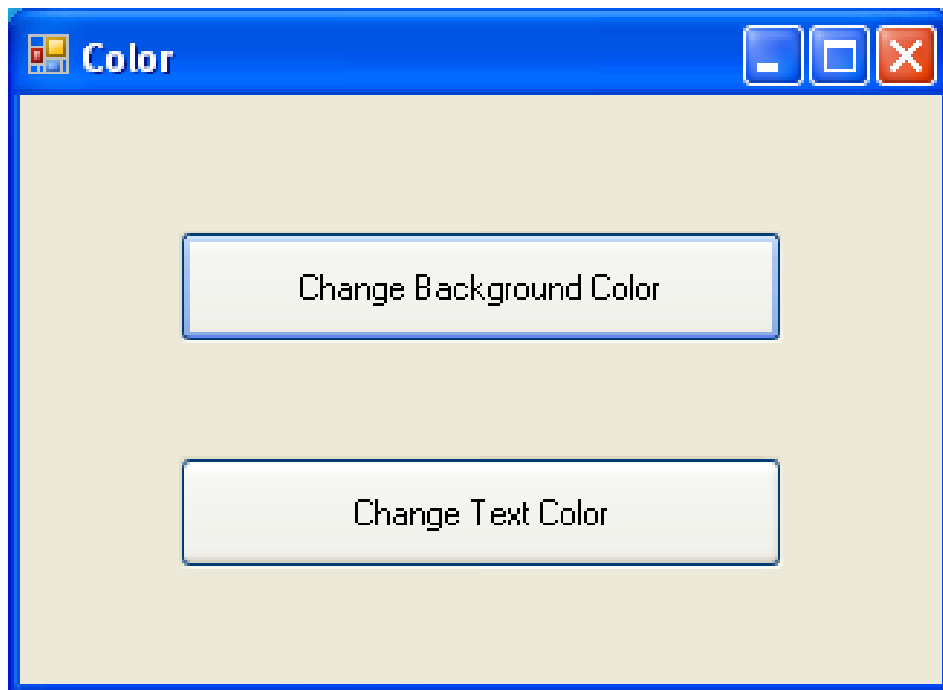
FullOpen = true



FullOpen = false



Demo



FONT & STRING

Lập trình Windows Form với C#

Font và String

- **Các khái niệm cơ bản**
- **Font**
 - Các khái niệm về Font
 - Lớp Font
 - Lớp FontFamily
- **String**
 - Một số thành phần liên quan đến chuỗi ký tự
 - Lớp StringFormat
 - Lớp TextRenderer

Các khái niệm cơ bản

- Bộ ký tự
- Trang mã
- Bộ ký tự Unicode

Bộ ký tự và trang mã

- **Bộ ký tự (character set)**

- Bộ mã bàn phím cố định mà một hệ máy tính cụ thể đang sử dụng.

- **Trang mã (code page)**

- Trang mã chứa 1 bộ các ký tự ứng với hệ bộ ký tự nào đó của một hoặc nhiều ngôn ngữ.
- Phần lớn các trang mã chứa 256 ký tự

Các khái niệm về Font

- Định nghĩa Font
- Kiểu Font và Họ Font
- Các loại Font trên HĐH Windows
- Các thông số Font
- Độ đo Font

Định nghĩa Font

- **Tập hợp hoàn chỉnh**

- các chữ cái
- các dấu câu
- các con số
- Các ký tự đặc biệt

theo một:

- kiểu loại
- trọng lượng (thường hoặc đậm nét)
- dáng bộ (thẳng hoặc nghiêng)

với kích cỡ phù hợp và có thể phân biệt khác nhau.

Định nghĩa Font (tt)

- Một font có thể được cung cấp bởi 1 hoặc nhiều trang mã
- Mỗi font thường được lưu trong 1 file font
- Mỗi font thường được đặt 1 tên.
- Ví dụ: font Times New Roman Bold,
 font Courier New Italic

Kiểu Font và họ Font

Tên Font = Họ Font + (Kiểu Font)

Font name = Font family + (Typestyle)

Ví dụ:

Họ font Times New Roman chứa 4 font khác nhau:

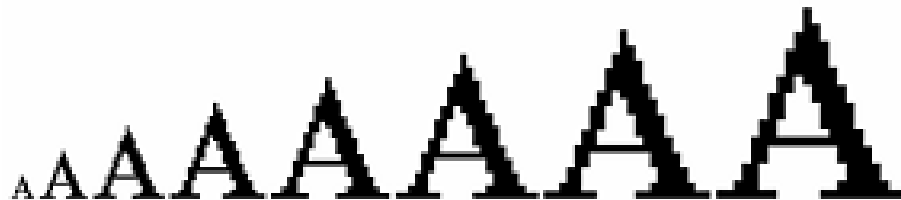
- **Times New Roman**
- *Times New Roman Italic*
- **Times New Roman Bold**
- *Times New Roman Bold Italic*

Các loại Font trên HĐH Windows

- **Bitmap Font**
- **Vector Font**
- **TrueType Font**
- **OpenType Font**

Bitmap Font

- Dùng hình ảnh để hiển thị các điểm ảnh của 1 ký tự
- Thường là những file có dạng *.fon
- Khi hiển thị ký tự có kích thước lớn thường bị nát hình



Vector Font

- Dùng các đoạn thẳng nối với nhau để hiển thị ký tự
- Thường là những file có dạng *.fnt
- Mặc dù hiển thị ký tự có kích thước lớn tốt hơn bitmap font nhưng vẫn chưa có được độ sắc nét cao



TrueType Font

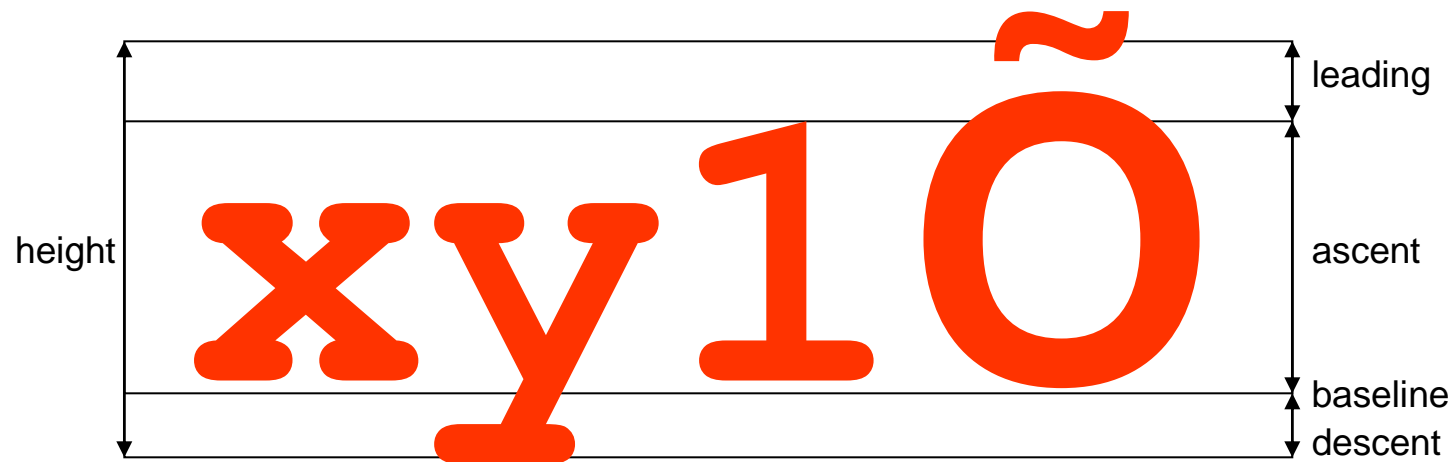
- Dùng các đoạn thẳng và đoạn cong nối với nhau để hiển thị ký tự
- Thường là những file có dạng *.ttf
- Hiển thị ký tự có kích thước lớn với độ sắc nét cao

→ *thường dùng*

OpenType Font

- Là chuẩn định dạng font kết hợp 2 loại định dạng font có sẵn: Type 1 (PostScript) font và TrueType font
- Độc lập với hệ điều hành (cross-platform)
- Hỗ trợ nhiều loại ngôn ngữ trong 1 font
- Thường là những file có dạng *.otf
- Trong Windows, TrueType Font có biểu tượng là chữ TT, còn OpenType Font là chữ O

Thông số Font (Font Metric)



Độ đo Font

Font được đo bởi nhiều độ đo:

- **pixel: phần tử nhỏ nhất của ảnh mà 1 thiết bị có thể hiển thị (màn hình, máy in)**
- **point: 1 point = 1/72 inch trong in ấn**
- **em: độ rộng của chữ M ứng với kiểu chữ đang dùng**
- **design unit: dùng để đo kích cỡ 1 họ font bằng độ đo point khi bị thay đổi kích thước**

Class Font

- **Mô tả lớp Font**
- **Các thuộc tính lớp Font**
- **Các hàm khởi tạo lớp Font**
 - Tập hợp FontStyle
 - Tập hợp GraphicsUnit
- **Các phương thức lớp Font**
- **Các ví dụ**

Mô tả lớp Font

- Dùng để xác định cách định dạng văn bản
- Bao gồm các thuộc tính cách thể hiện, kích thước kiểu dáng thước, kiểu dáng
- Không cho phép kế thừa
- Namespace: **System.Drawing**
- Assembly: **System.Drawing (in dll)**

Thuộc tính	Mô tả
Bold	Kiểm tra xem Font có được in đậm hay không.
FontFamily	Thông tin về FontFamily của Font
Height	Chiều cao của Font
Italic	Kiểm tra xem Font có được in nghiêng hay không
Name	Tên của Font
Size	Kích thước của Font, kiểu float
Strikeout	Kiểm tra Font có in gạch ngang hay không.
Underline	Kiểm tra Font có in gạch dưới hay không.

FontFamily Class

- **Mô tả lớp FontFamily**
- **Các thuộc tính lớp FontFamily**
- **Các hàm khởi tạo lớp FontFamily**
 - Tập hợp GenericFontFamilies
- **Các phương thức lớp FontFamily**
- **Các ví dụ**

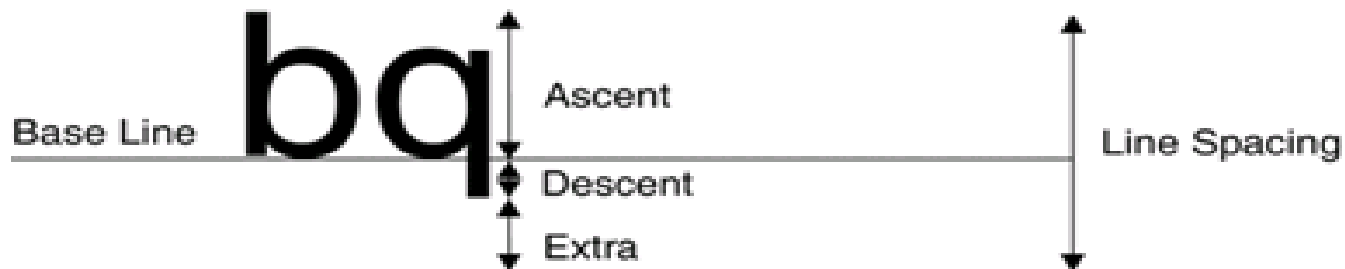
Mô tả lớp **FontFamily**

- **Xác nhận một nhóm các font được thiết kế về cách thể hiện và kiểu dáng tương tự nhau**
- **Không cho phép kế thừa**
- **Namespace: System.Drawing**
- **Assembly: System.Drawing (in dll)**

Các thuộc tính **FontFamily**

Thuộc tính	Miêu tả
Families	Trả về một mảng tất cả các font families liên kết với ngữ cảnh đồ họa hiện thời
GenericMonospace	Trả về font family đơn
GenericSansSerif	Trả về font không có chân chữ
GenericSerif	Trả về font có chân chữ
Name	Trả về tên của font families

Các phương thức **FontFamily**



Các phương thức **FontFamily**

Phương thức	Miêu tả
GetCellAscent	trả về phần chữ bên trên đường baseLine
GetCellDecent	trả về phần chữ bên dưới đường baseLine
GetEmHeigth	Trả về chiều cao
GetFamilies	Trả về mảng chứa tất cả các giá trị của font families
GetLineSpacing	Trả về khoảng cách giữa hai dòng liên tiếp
GetName	Trả về tên của font family
IsStyleAvailable	Kiểm tra giá trị style của font ví dụ: <code>FontFamily ff = new FontFamily("Arial");</code> <code>if(ff.IsStyleAvailable(FontStyle.Italic))</code> <code>//xử lý</code>

Ví dụ

❖ Khởi tạo một Font:


//tạo Font

```
Font tnwFont = new Font("Times New Roman", 12);
```

//tạo FontFamily

```
Font fa = new Font("Times New Roman", 8);
Font fb = new Font("Arial", 36, FontStyle.Bold);
Font fc = new Font(fb, FontStyle.Bold | FontStyle.Italic);
Font fd = new Font("Arial", 1, GraphicsUnit.Inch);
```

Size = 8 pixel



Size = 1 inch





Pen

- Xác định width, style, fill style
- Không cho kế thừa, nhưng tạo thể hiện được
- Trong namespace **System.Drawing**
 - `Pen p1 = new Pen(Color.Green);`
`Pen p2 = new Pen(Color.Blue, 10);`
- Sử dụng lớp Pens có 141 pen được định nghĩa trước.
 - `Pen p3 = Pens.Violet;`

Pen (tt)

- **Dùng Pen có sẵn thông qua Pens**
 - Pens.AliceBlue
 - Pens.Aqua
 - Pens.Black
 - Pens.Brown
 - ...
- **Dùng Pen do tự tạo thông qua lớp Pen**
 - new Pen(Brush)
 - new Pen(Color)
 - new Pen(Color, Width)

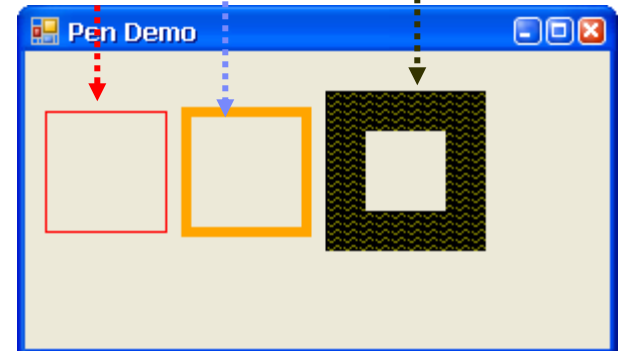
Pen (tt)

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

    Pen p1 = new Pen(Color.Red); .....
    g.DrawRectangle(p1, new Rectangle(10, 30, 60, 60));

    Pen p2 = new Pen(Color.Orange, 5); .....
    g.DrawRectangle(p2, new Rectangle(80, 30, 60, 60));

    HatchBrush hb = new HatchBrush( HatchStyle.Wave, Color.Olive);
    Pen p3 = new Pen(hb, 20); .....
    g.DrawRectangle(p3, new Rectangle(160, 30, 60, 60));
}
```





Brush

- Dùng để tô vùng bên trong của hình
- Lớp Brush là lớp Abstract nên không tạo thể hiện
- Sử dụng các lớp kế thừa sau để tạo brush
 - SolidBrush
 - LinearGradientBrush
 - TextureBrush
 - HatchBrush
- Sử dụng lớp Brushes định nghĩa trước các brush.

Brush

```
protected override void OnPaint(PaintEventArgs e)
```

```
{
```

```
    Graphics g = e.Graphics;
```

```
    SolidBrush b1 = new SolidBrush(Color.Aquamarine);  
    g.FillEllipse(b1, 40, 40, 80, 80);
```

Solid

```
    HatchBrush b2 = new HatchBrush(HatchStyle.ZigZag, Color.Red);  
    g.FillEllipse(b2, 80, 80, 80, 80);
```

Hatch

```
    Rectangle rect = new Rectangle(0, 0, 100, 100);  
    LinearGradientBrush b3 = new LinearGradientBrush(rect,  
        Color.Red, Color.Blue, LinearGradientMode.Horizontal);  
    g.FillEllipse(b3, 120, 120, 80, 80);
```

LinearGradient

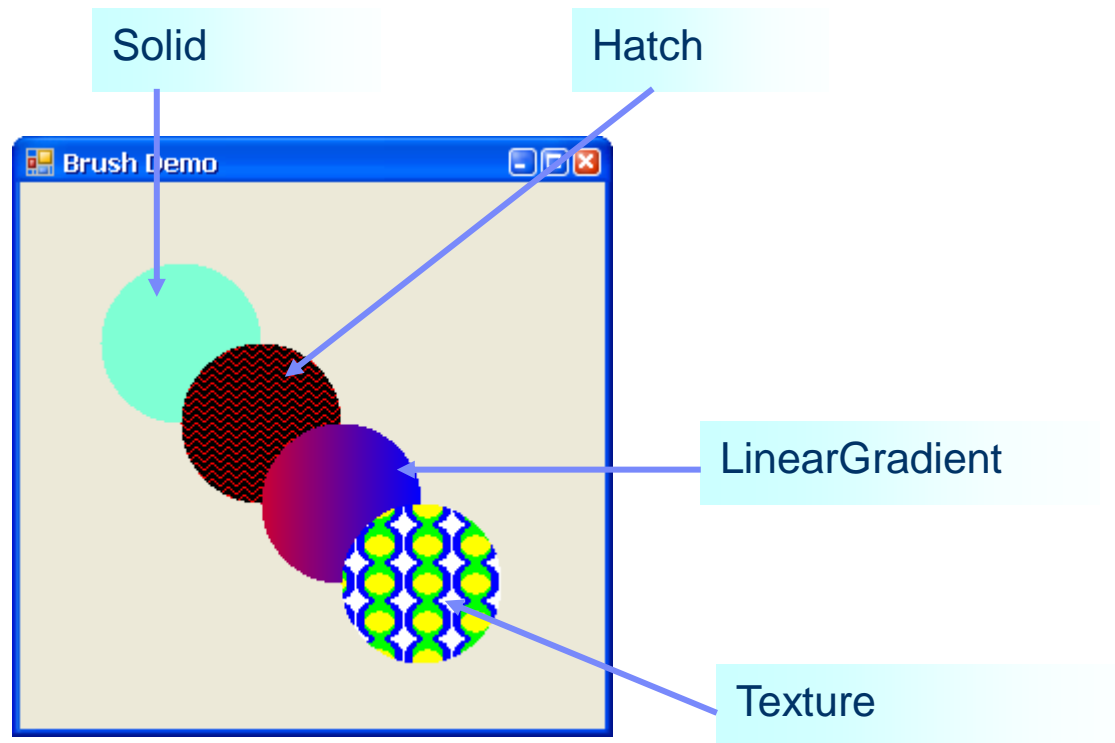
```
    Image img = Image.FromFile("bitmap1.bmp");  
    TextureBrush b4 = new TextureBrush(img);  
    g.FillEllipse(b4, 160, 160, 80, 80);
```

Texture

```
}
```

Brush

■ Demo

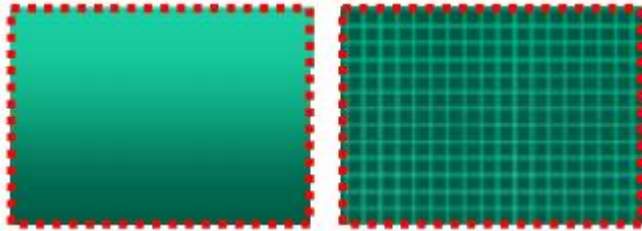


VỀ CÁC ĐƯỜNG

Lập trình Windows Form với C#

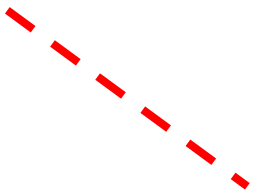
2D vector graphics

■ Pen & brush

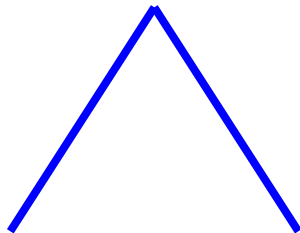


Pen, Pens, SystemPens
Brush, Brushes, SystemBrushes,
SolidBrushes, TextureBrushes,
(System.Drawing.Drawing2D)
HatchBrush, LinearGradientBrush,
PathGradientBrush

■ Polygon



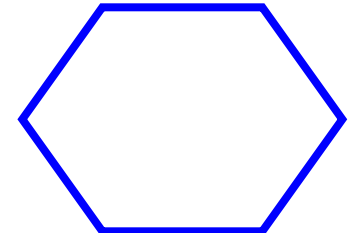
DrawLine



DrawLines



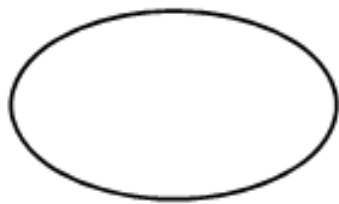
DrawRectangle
FillRectangle



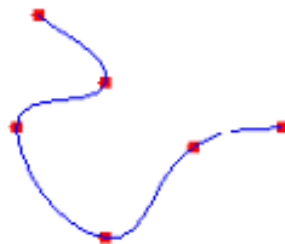
DrawPolygon
FillPolygon

2D vector graphics

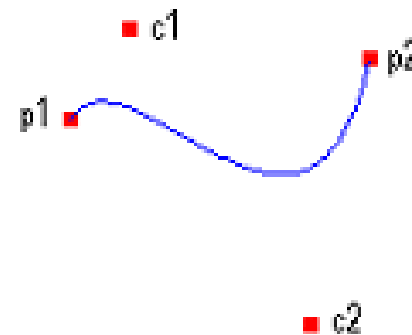
- ellipse, arc, cardinal spline, bezier spline



DrawEllipse
FillEllipse



DrawCurve
DrawClosedCurve
FillClosedCurve



DrawBezier
DrawBeziers



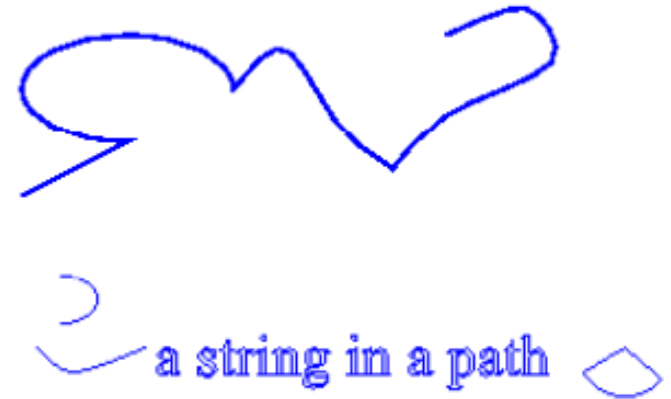
DrawPie
FillPie



DrawArc

2D vector graphics

- **Path:** kết hợp nhiều loại đường nét thành một đối tượng duy nhất. Các “nét” không nhất thiết phải liền nhau.



- **GraphicsPath (AddLine, AddCurve, ...)**
- **Graphics.DrawPath**
- **Graphics.FillPath**

Line, Rectangle, Ellipse

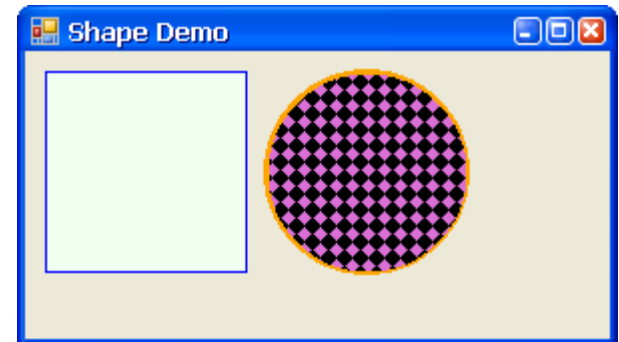
- **DrawLine**
 - (*Pen p, int x1, int y1, int x2, int y2*)
- **DrawRectangle**
 - (*Pen p, int x, int y, int width, int height*)
- **DrawEllipse**
 - (*Pen p, int x, int y, int width, int height*)
- **FillRectangle**
 - (*Brush b, int x, int y, int width, int height*)
- **FillEllipse**
 - (*Brush b, int x, int y, int width, int height*)

Line, Rectangle, Ellipse

```
protected void DemoShape(Graphics g)
{
    Rectangle r1 = new Rectangle(10, 10, 100, 100);
    Rectangle r2 = new Rectangle(11, 11, 99, 99);
    Rectangle r3 = new Rectangle(120, 10, 100, 100);
    Rectangle r4 = new Rectangle(121, 11, 99, 99);
    Pen p = new Pen(Color.Orange, 3);
    HatchBrush hb = new
        HatchBrush(HatchStyle.SolidDiamond, Color.Orchid);

    g.DrawRectangle(Pens.Blue, r1);
    g.FillRectangle(Brushes.Honeydew, r2);

    g.DrawEllipse(p, r3);
    g.FillEllipse(hb, r4);
}
```

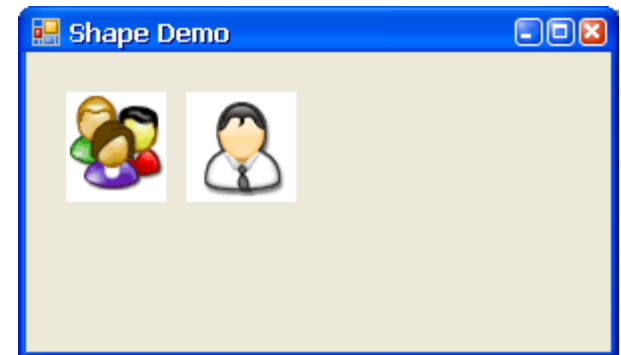


Image

- **Lớp Image hiển thị các ảnh bitmap**
 - Các dạng ảnh: *.bmp, *.gif, *.jpg, *.ico...
- **Phương thức static FromFile tạo ảnh từ file**
 - `Image img = Image.FromFile("niie.bmp");`
 - `Image img2 = Image.FromFile("niie.gif");`
- **Phương thức DrawImage xuất ảnh lên Graphics**
 - `g.DrawImage(img, 10, 10);`
 - `G.DrawImage(img2, 10, 10, 100,100);` // scale trong hình chữ nhật kích thước 100x100

Image

```
protected void DemoImage(Graphics g)
{
    //đọc ảnh từ đĩa
    Image img = Image.FromFile("people.bmp");
    g.DrawImage(img, 20, 20);
    //đọc ảnh từ embedded resource
    Image img2 = new Bitmap(GetType(), "people3.bmp");
    g.DrawImage(img2, 80, 20);
}
```



Image

```
protected void DemoImage2(Graphics g)
{
    Image img = new Bitmap(GetType(), "bluekimono.bmp");
    // lấy đối tượng Graphics của img
    Graphics Gimg = Graphics.FromImage(img);
    // tô hình ellipse đặc
    Gimg.FillEllipse(Brushes.Gold, 10, 10, 60, 60);

    // hiển thị ra Graphics của form
    g.DrawImage(img, 0, 0, this.Width, Height);
}
```

Ellipse được vẽ lên ảnh, rồi sau đó với
vẽ ảnh lên Form



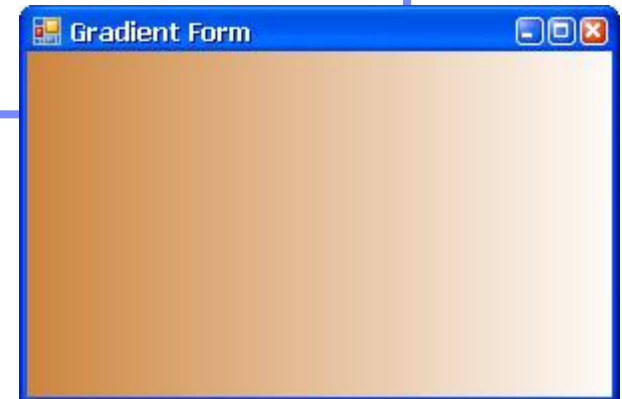
Minh họa 1

■ Custom lại nền của Form

```
protected override void OnPaintBackground(PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Rectangle rect = new Rectangle(0, 0, Width, Height);
    LinearGradientBrush b = new
        LinearGradientBrush(rect, Color.Peru, Color.White,
        LinearGradientMode.Horizontal);

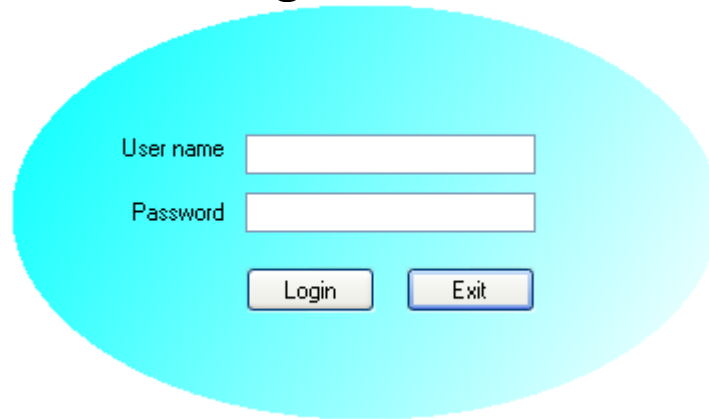
    g.FillRectangle(b, 0, 0, Width, Height);
}
```

Override phương thức OnPaintBackground của Form



Form có dạng NonRectangle

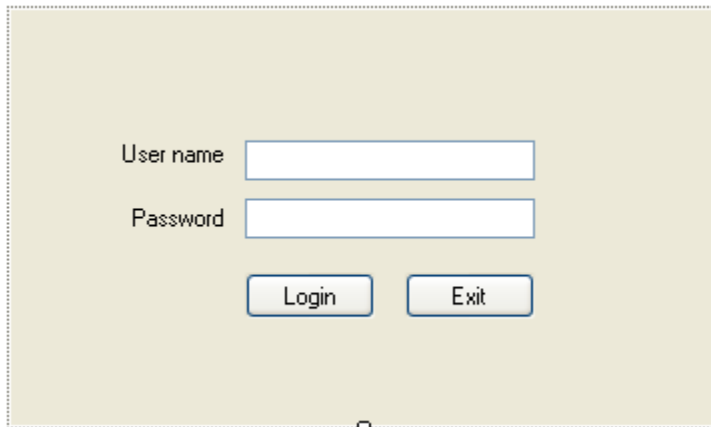
- **Tạo form có hình dạng khác hình chữ nhật**
 - Sử dụng thuộc tính TransparencyKey của Form
 - Sử dụng các hiệu ứng màu được tô



Form có dạng NonRectangle

■ Bước 1:

- Tạo ứng dụng Windows Application
- Thiết kế Form có dạng như sau

A screenshot of a Windows Form with a light beige background and no border. It contains two text boxes labeled 'User name' and 'Password', and two buttons labeled 'Login' and 'Exit'.

FormBorderStyle = none



Form có dạng NonRectangle

■ Bước 2:

- Thiết lập các thuộc tính cho Form như sau:
 - TransparencyKey = Control: màu sẽ trong suốt khi vẽ trên Form
 - FormBorderStyle = None: Form không có đường biên
- Thiết lập màu nền cho 2 Label là Transparent
 - Phần background của 2 label sẽ tiếp với nền bên dưới

Form có dạng NonRectangle

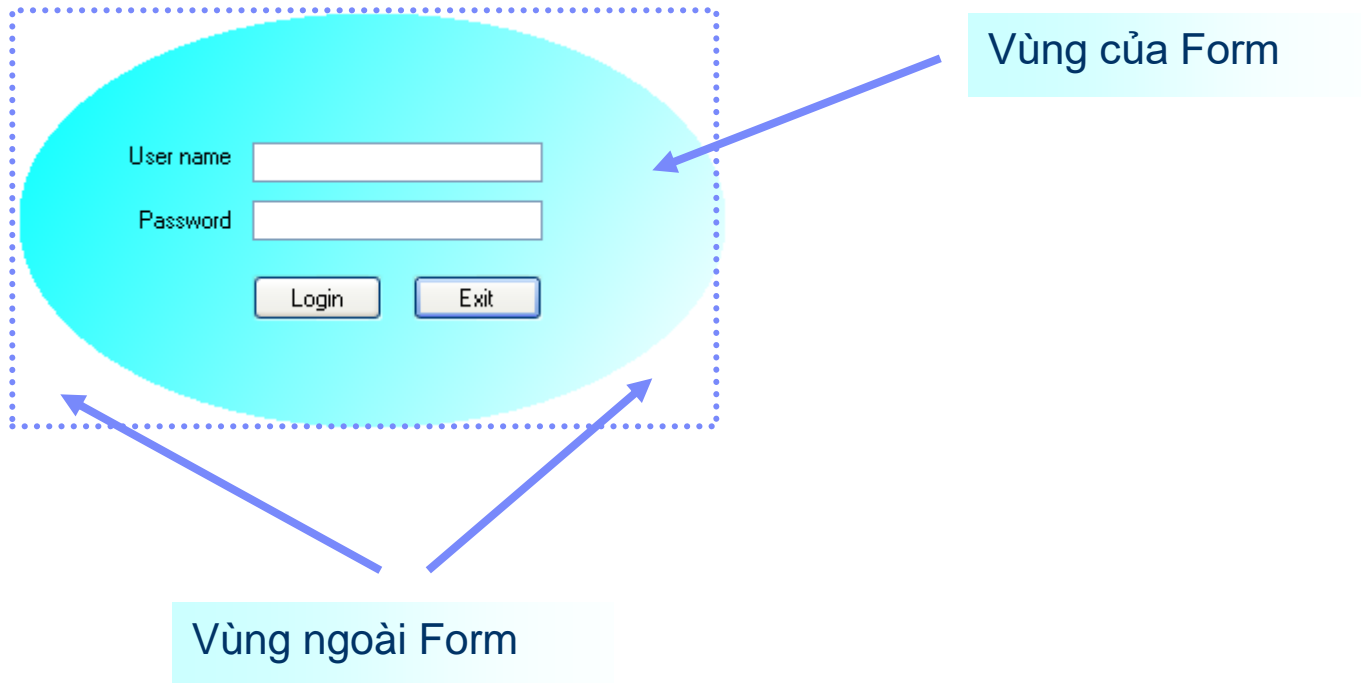
■ Bước 3: Tạo trình xử lý cho sự kiện Paint

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics myGraphic = e.Graphics;

    // khai báo các tọa độ & kích thước
    Point p = new Point(0, 0);
    Point p2 = new Point(Width, Height);
    Size size = new Size(Width, Height); // kích thước của form
    // tạo brush gradient
    LinearGradientBrush myBrush =
        new LinearGradientBrush(p, p2, Color.Cyan, Color.White);
    // khai báo hình ellipse làm hình dạng của form
    Rectangle r1 = new Rectangle(p, size);
    // tô hình ellipse với brush vừa định nghĩa
    myGraphic.FillEllipse(myBrush, r1);
}
```

Form có dạng NonRectangle

■ Demo



Multimedia

- **Tạo ứng dụng chứa Windows Media Player control cho phép**
 - Play các file video và sound theo nhiều dạng format
 - MPEG (Motion Pictures Expert Group): video
 - AVI (Audio-video Interleave): video
 - WAV (Windows Wave-file Format): audio
 - MIDI (Musical Instrument Digital Interface): audio

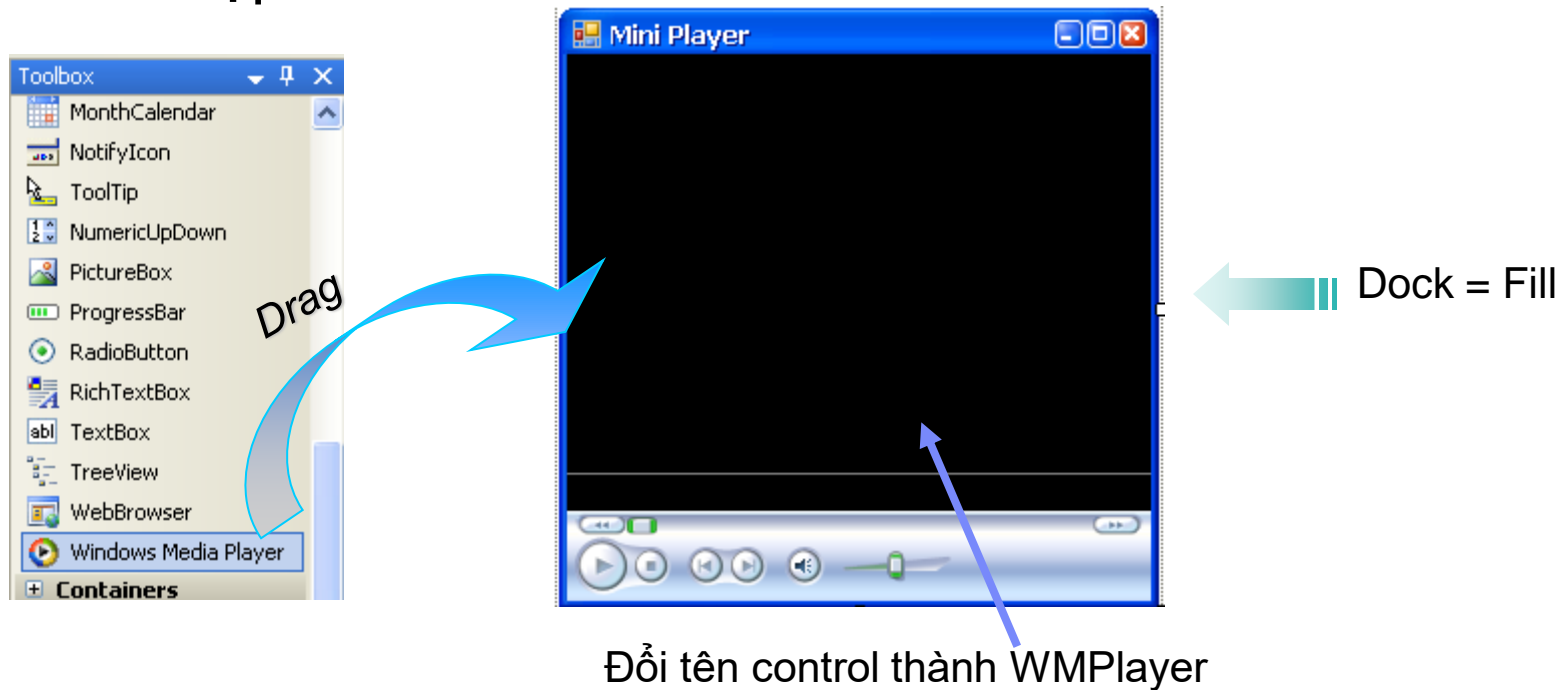
Multimedia

- **Bước 1: bổ sung Windows Media Player vào Toolbox**
 - Kích chuột phải vào Toolbox -> chọn Choose Items...
 - Trong Dialog Choose Toolbox Items chọn COM Components
 - Chọn Windows Media Player
 - Khi đó control WMP sẽ hiện ở dưới cùng của Toolbox

Multimedia

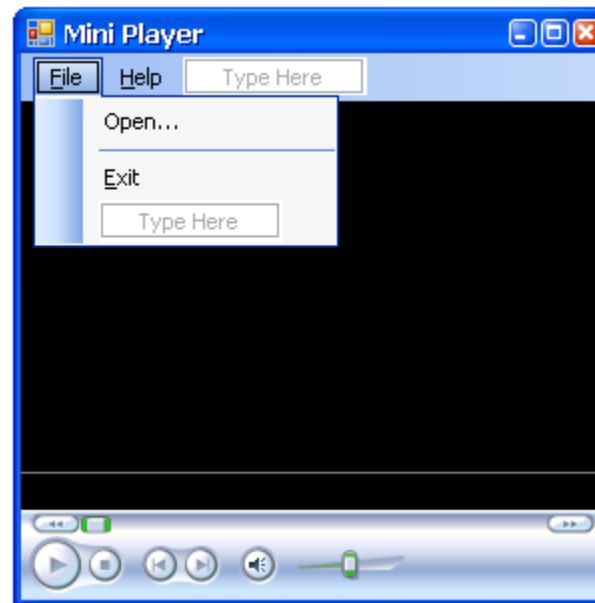
■ Bước 2: kéo Windows Media Player thả vào Form

– Thiết lập Dock = Fill



Multimedia

- **Bước 3: Tạo MenuStrip để bổ sung chức năng Open File media**



menuStrip1

Multimedia

■ Bước 4: viết trình xử lý cho MenuItem Open

```
private void openMenuItem_Click(object sender, EventArgs e)
{
    // tạo hộp thoại mở file
    OpenFileDialog dlg = new OpenFileDialog();
    // lọc hiển thị các loại file
    dlg.Filter = "AVI file|*.avi|MPEG File|*.mpeg|"+
                "WAV File|*.wav|MIDI File|*.midi";
    // Hiển thị Open Dialog
    if (dlg.ShowDialog() == DialogResult.OK)
    {
        // lấy tên file cần mở cho WMPPlayer
        WMPPlayer.URL = dlg.FileName;
    }
}
```

Multimedia

- Demo



