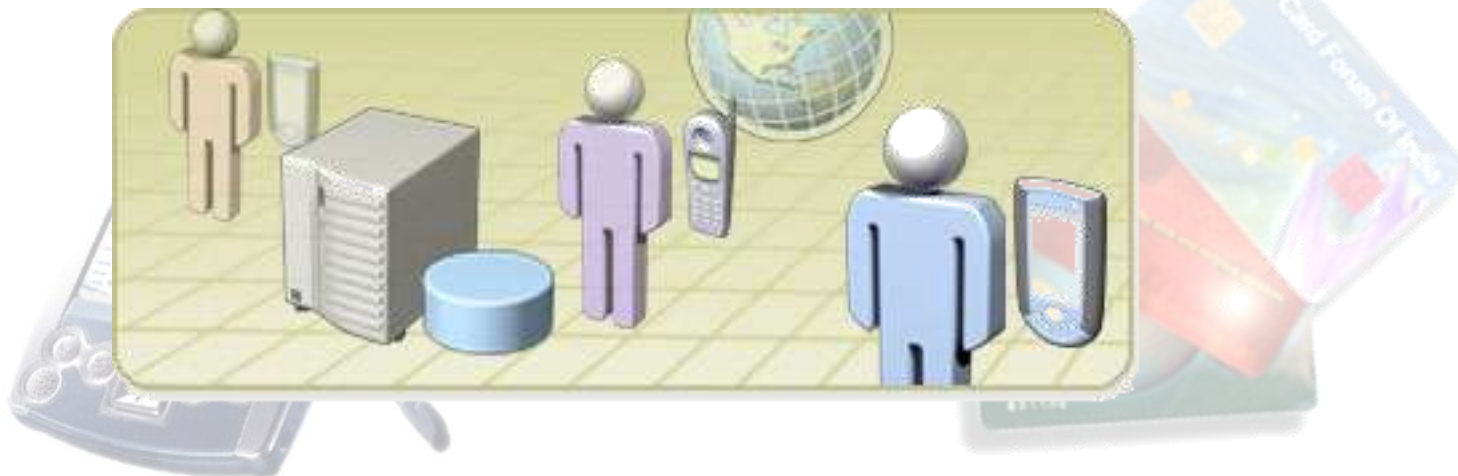


Đại học Khoa học Tự nhiên, ĐHQG-HCM
Khoa Công Nghệ Thông Tin

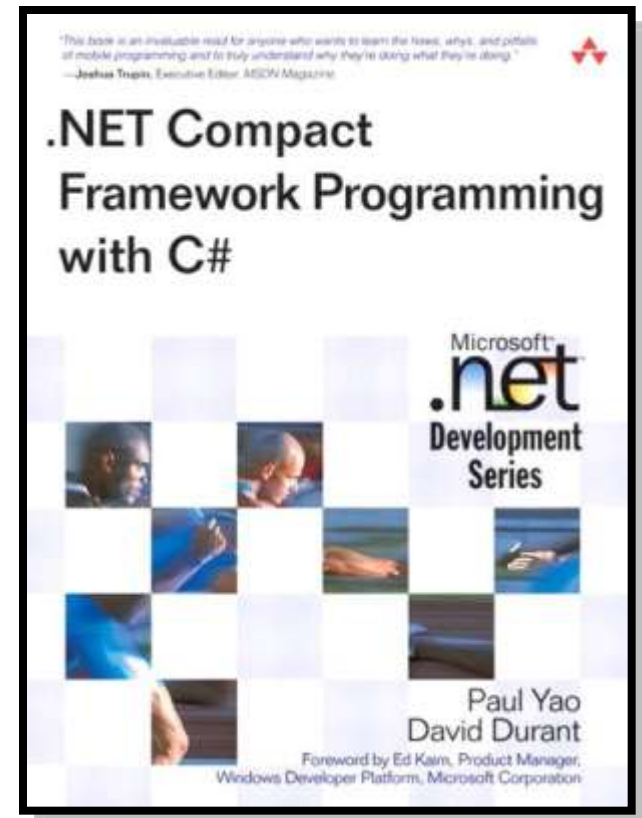
Bài 3: Thao tác đồ họa trên .Net Compact Framework

TS. Trần Minh Triết



Tham khảo

- ***.NET Compact Framework Programming with C#***, Paul Yao, David Durant (2004), Prentice Hall PTR
 - Chương 15 - .Net Compact Framework Graphics
 - Chương 16 – Text and Fonts



Đối tượng Graphics

- Cách 1: Sử dụng đối tượng **Graphics** được truyền vào trong tham số của hàm xử lý sự kiện **Paint**

```
private void FormMain_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // draw
}
```
- Cách 2: Tự tạo ra đối tượng **Graphics** (lưu ý: cần giải phóng vùng nhớ sau khi dùng xong)

```
Graphics g = CreateGraphics();
// Draw
g.Dispose();
```
- Cách 3: sử dụng phương thức tĩnh **Graphics.FromImage** để nhận được đối tượng graphics của ảnh

Xác định màu

- 3 cách để xác định màu
 - Dùng màu hệ thống (`System.Drawing.SystemColors`)
 - Dùng màu được định nghĩa sẵn
 - Dùng bộ giá trị RGB



Danh sách các màu được dùng trong hệ thống

- ActiveBorder
- ActiveCaption
- ActiveCaptionText
- AppWorkspace
- Control
- ControlDark
- ControlDarkDark
- ControlLight
- ControlLightLight
- ControlText
- Desktop
- GrayText
- Highlight
- HighlightText
- HotTrack
- InactiveBorder
- InactiveCaption
- InactiveCaptionText
- Info
- InfoText
- Menu
- MenuText
- ScrollBar
- Window
- WindowFrame
- WindowText

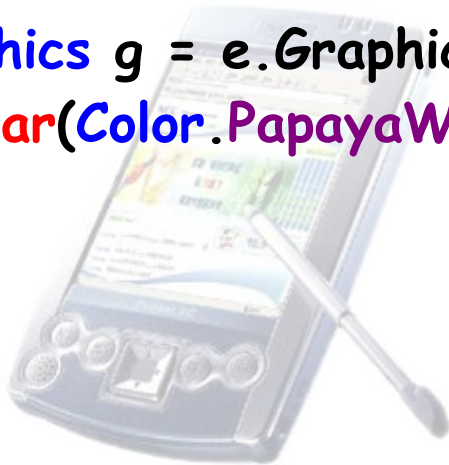


**Màu cụ thể tương ứng với mỗi hàng số
sẽ thay đổi tùy theo từng hệ thống cụ thể**

Ví dụ

```
private void FormMain_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.Clear(SystemColors.Window);
}
```

```
private void FormMain_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.Clear(Color.PapayaWhip);
}
```



Ví dụ

- Hàm **FromArgb** (không có thành phần alpha trên .Net CF)

```
public static Color FromArgb( int red, int green, int blue);
```

- Ví dụ:

```
private void FormMain_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    g.Clear(Color.FromArgb(204,204,204));
}
```



Tạo Brush

- Trên .Net CF chỉ hỗ trợ **solid brush** và **bitmap brush**

- Class: **System.Drawing.SolidBrush**

- Constructor:

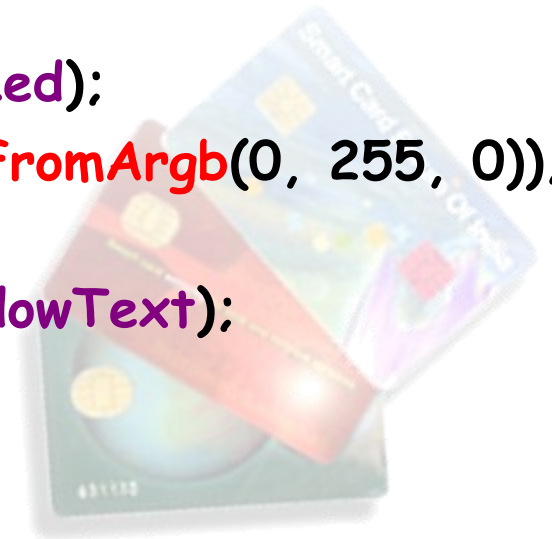
public SolidBrush(Color color);

- Ví dụ:

Brush brRed = new SolidBrush(Color.Red);

Brush brGreen = new SolidBrush(Color.FromArgb(0, 255, 0));

**Brush brWindowText =
new SolidBrush(SystemColors.WindowText);**



Tạo bitmap

- Constructor tạo bitmap rỗng
`public Bitmap (int width, int height);`



Vẽ lên Bitmap

```
private void CreateAndDraw(int x, int y)
{
    // Create a bitmap and a Graphics object for the bitmap.
    Bitmap bmpNew = new Bitmap(100,100);
    Graphics gbmp = Graphics.FromImage(bmpNew);

    // Clear the bitmap background.
    gbmp.Clear(Color.LightGray);

    // Get a Graphics object for the form.
    Graphics g = CreateGraphics();

    // Copy the bitmap to the window at (x,y) location.
    g.DrawImage(bmpNew, x, y);

    // Clean up when we are done.
    g.Dispose();
    gbmp.Dispose();
    bmpNew.Dispose();
}
```



Tạo Bitmap từ file

- Constructor tạo **Bitmap** từ file

```
public Bitmap ( string filename);
```

- Các dạng file được hỗ trợ

- Bitmap (.bmp) (1, 4, 8, hay 24 bit màu)
- JPEG (.jpg)
- GIF (.gif)
- PNG (.png)

- Ví dụ:

```
try
```

```
{ bmpNew = new Bitmap(strFileName);  
}
```

```
catch
```

```
{ MessageBox.Show("Cannot create bitmap from File: " +  
    strFileName);  
}
```



Tạo Bitmap từ Resource

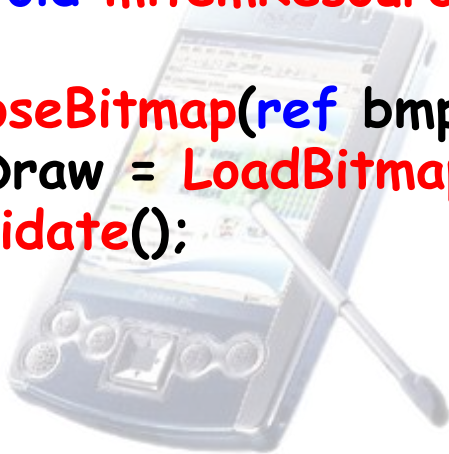
```
private Bitmap LoadBitmapResource(string strName)
{
    Assembly assembly = Assembly.GetExecutingAssembly();
    string strRes = "ShowBitmap." + strName;
    Stream stream =
        assembly.GetManifestResourceStream(strRes);
    Bitmap bmp = null;
    try
    {
        bmp = new Bitmap(stream);
    }
    catch { }
    stream.Close();
    return bmp;
}
```



Tạo Bitmap từ Resource (tt)

```
private void DisposeBitmap(ref Bitmap bmp)
{
    if (bmp != null)
    {
        bmp.Dispose();
    }
    bmp = null;
}
```

```
private void mitemResourceCup_Click(object sender, EventArgs e)
{
    DisposeBitmap(ref bmpDraw);
    bmpDraw = LoadBitmapResource("CUP.BMP");
    Invalidate();
}
```



Hiển thị Bitmap

- Các trường hợp sử dụng:
 - Hiển thị **toàn bộ** bitmap với **kích thước gốc**
 - Hiển thị **một phần** bitmap với **kích thước gốc**
 - Hiển thị **một phần** bitmap với **kích thước được thay đổi**
 - Hiển thị **một phần** bitmap với **kích thước được thay đổi** và có **vùng trong suốt**



Hiển thị toàn bộ bitmap với kích thước gốc

- **Hàm**

```
public void DrawImage( Image image, int x, int y);
```

- **Ví dụ:**

```
private void FormMain_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    int x = 10;
    int y = 10;

    g.DrawImage bmpDraw, x, y);
}
```



Hiển thị một phần bitmap với kích thước gốc

- **Hàm**

```
public void DrawImage  
( Image image, int x, int y,  
    Rectangle srcRect, GraphicsUnit srcUnit);
```

- Trên .NetCF, tham số **srcUnit** chỉ có một chọn lựa là **Pixel**



Hiển thị một phần bitmap với kích thước được thay đổi

- **Hàm:**

```
public void DrawImage  
( Image image, Rectangle destRect,  
  Rectangle srcRect, GraphicsUnit srcUnit);
```



Hiển thị một phần bitmap với kích thước được thay đổi và có vùng trong suốt

- **Hàm**

public void DrawImage

(**Image** image, **Rectangle** destRect,
int srcX, int srcY, int srcWidth, int srcHeight,
GraphicsUnit srcUnit, **ImageAttributes** imageAttr);

- Lưu ý: tham số **imgatt** cho phép xác định dải màu sẽ được vẽ trong suốt khi hiển thị ảnh

imgatt.SetColorKey
(**Color.FromArgb**(r1, g1, b1), **Color.FromArgb**(r2, g2, b2));

- Những màu có giá trị (r, g, b) thỏa $r1 \leq r \leq r2$, $g1 \leq g \leq g2$, $b1 \leq b \leq b2$ sẽ không được vẽ (tức là transparent)

Thao tác đồ họa vector

DrawEllipse	Vẽ đường viền ellipse (với viết được chọn)
DrawLine	Vẽ đoạn thẳng (với viết được chọn)
DrawPolygon	Vẽ đường biên đa giác (với viết được chọn)
DrawRectangle	Vẽ đường biên hình chữ nhật (với viết được chọn)
FillEllipse	Tô ellipse (với brush được chọn)
FillPolygon	Tô đa giác (với brush được chọn)
FillRectangle	Tô hình chữ nhật (với brush được chọn)



Tạo bút vẽ

- Class: `System.Drawing.Pen`

- Constructor:
`public Pen(Color color);`

- Ví dụ:

```
// Pen from a system color
```

```
Pen penCtrl = new Pen(SystemColors.ControlDark);
```

```
// Pen from a named color
```

```
Pen penRed = new Pen(Color.Red);
```

```
// Pen from an RGB value
```

```
Pen penBlue = new Pen(Color.FromArgb(0, 0, 255));
```



Sử dụng bút vẽ và brush

```
private void FormMain_Paint(object sender, PaintEventArgs e)
{
    Brush brText = new SolidBrush(SystemColors.WindowText);
    e.Graphics.DrawString("Simple Draw String", Font, brText,
        xDraw, yDraw);

    // Highlight origin.
    int x = (int)xDraw;
    int y = (int)yDraw;

    Pen penBlack = new Pen(Color.Black);
    e.Graphics.DrawLine(penBlack, x, y, x-8, y);
    e.Graphics.DrawLine(penBlack, x, y, x, y-8);
}
```



Xuất chuỗi ký tự

- Hàm

```
public void DrawString( string str, Font font, Brush brText,  
float x, float y);
```

- Ví dụ

```
private void FormMain_Paint(object sender, PaintEventArgs e)  
{ Brush brText = new SolidBrush(SystemColors.WindowText);  
  e.Graphics.DrawString("Simple Draw String", Font, brText,  
    xDraw, yDraw);  
  
  // Highlight origin.  
  int x = (int)xDraw;  
  int y = (int)yDraw;  
  Pen penBlack = new Pen(Color.Black);  
  e.Graphics.DrawLine(penBlack, x, y, x-8, y);  
  e.Graphics.DrawLine(penBlack, x, y, x, y-8);  
}
```


Thay đổi thuộc tính Font

- Khác với **.Net Framework**, trong **.Net Compact Framework**, việc thay đổi thuộc tính **BackColor**, **Cursor**, **Font**, và **ForeColor** trong control cha không làm thay đổi các thuộc tính này trong các control con.
- Do đó, cần tự thay đổi các thuộc tính này cho các control cần thiết



Generic Font

- Hàm khởi tạo Generic Font

`public Font (FontFamily family, float emSize, FontStyle style);`

- Ví dụ

`Font font = new Font`

`(FontFamily.GenericSansSerif, 10, FontStyle.Regular);`



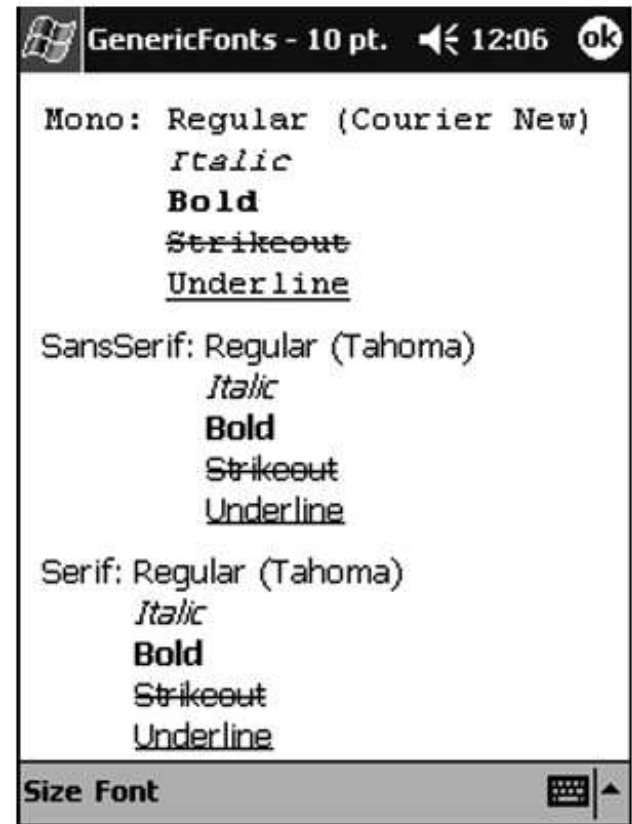
Generic Font

- **FontFamily.GenericMonospace**: sử dụng fixed-pitch font (ví dụ Courier New)
- **FontFamily.GenericSansSerif**: sử dụng variable-pitch font không có chân (ví dụ: Arial trên Windows, Tahoma trên Pocket PC)
- **FontFamily.GenericSerif**: sử dụng variable-pitch font có chân (ví dụ Times New Roman trên Windows)



Generic Font

- `FontStyle.Bold`
- `FontStyle.Italic`
- `FontStyle.Regular`
- `FontStyle.Strikeout`
- `FontStyle.Underline`



Generic Font

// Create monospace bold 10-point font.

```
Font fontMono = new Font(FontFamily.GenericMonospace, 10,  
    FontStyle.Bold);
```

// Create sans serif italic 10-point font.

```
Font fontSans = new Font(FontFamily.GenericSansSerif, 10,  
    FontStyle.Italic);
```

// Create serif italic and underlined 10-point font.

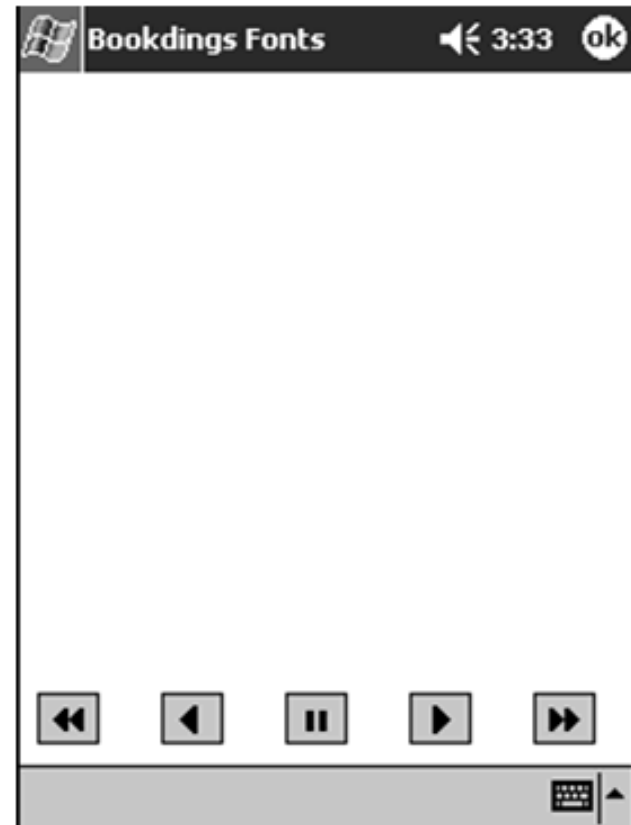
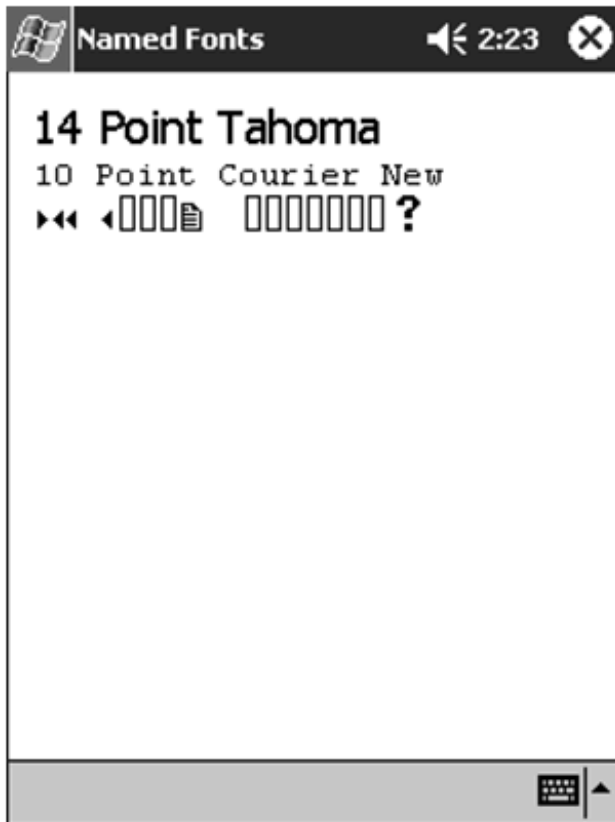
```
Font fontSerif = new Font(FontFamily.GenericSerif, 10,  
    FontStyle.Italic | FontStyle.Underline);
```



Tạo font từ tên font

- Hàm khởi tạo Named Font

`public Font(string familyName, float emSize, FontStyle style);`



Tạo font từ tên font

```
private void FormMain_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    float x = 10;    float y = 10;

    Font font1 = new Font("Tahoma", 14, FontStyle.Regular);
    Font font2 = new Font("Courier New", 10, FontStyle.Regular);
    Font font3 = new Font("Bookdings", 12, FontStyle.Regular);

    Brush brText = new SolidBrush(SystemColors.WindowText);

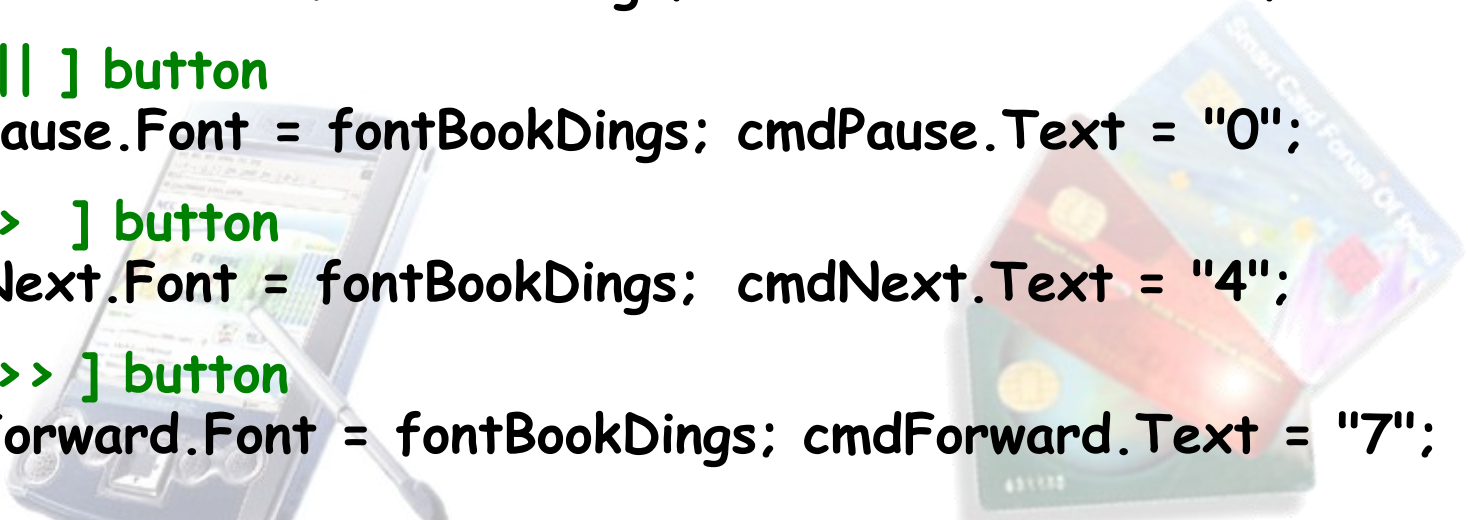
    g.DrawString("14 Point Tahoma", font1, brText, x, y);
    SizeF sizeX = g.MeasureString("X", font1);
    y += sizeX.Height;

    g.DrawString("10 Point Courier New", font2, brText, x, y);
    sizeX = g.MeasureString("X", font2);
    y += sizeX.Height;

    g.DrawString("12 Point Bookdings", font3, brText, x, y);
}
```


Tạo font từ tên font

```
private void FormMain_Load(object sender, System.EventArgs e)
{
    Font fontBookDings = new Font("BookDings", 14, FontStyle.Bold);
    // [ << ] button
    cmdRewind.Font = fontBookDings; cmdRewind.Text = "2";
    // [ < ] button
    cmdBack.Font = fontBookDings; cmdBack.Text = "3";
    // [ || ] button
    cmdPause.Font = fontBookDings; cmdPause.Text = "0";
    // [ > ] button
    cmdNext.Font = fontBookDings; cmdNext.Text = "4";
    // [ >> ] button
    cmdForward.Font = fontBookDings; cmdForward.Text = "7";
}
```



Phụ lục: Hộp thoại chọn màu (gọi WinAPI bằng P/Invoke)

```
public struct CHOOSECOLOR
{
    public int          IStructSize;
    public IntPtr       hwndOwner;
    public IntPtr       hInstance;
    public int          rgbResult;
    public IntPtr       lpCustColors;
    public int          Flags;
    public int          lCustData;
    public IntPtr       lpfnHook;
    public IntPtr       lpTemplateName;
};
```



Phụ lục:

Hộp thoại chọn màu (gọi WinAPI bằng P/Invoke)

```
public const int CC_RGBINIT = 0x00000001;  
public const int CC_FULLOPEN = 0x00000002;  
public const int CC_PREVENTFULLOPEN = 0x00000004;  
public const int CC_ENABLEHOOK = 0x00000010;  
public const int CC_ENABLETEMPLATE = 0x00000020;  
public const int CC_ENABLETEMPLATEHANDLE = 0x00000040;  
public const int CC_SOLIDCOLOR = 0x00000080;  
public const int CC_ANYCOLOR = 0x00000100;
```

```
public static int INVALID_HANDLE_VALUE = -1;  
public const int LMEM_FIXED = 0x0000;
```



Phụ lục:

Hộp thoại chọn màu (gọi WinAPI bằng P/Invoke)

```
[DllImport("coredll.dll")]
```

```
public static extern IntPtr LocalAlloc (int uFlags, int uBytes);
```

```
[DllImport("coredll.dll")]
```

```
public static extern IntPtr LocalFree (IntPtr hMem);
```

```
[DllImport("commdlg.dll")]
```

```
public static extern int ChooseColor (ref CHOOSECOLOR lpcc);
```



Phụ lục:

Hộp thoại chọn màu (gọi WinAPI bằng P/Invoke)

```
public bool Init(Control ctrlParent)
{
    // Allocate the array for initial colors.
    int cbColorData = 16 * 4;
    IntPtr ipColors = LocalAlloc(LMEM_FIXED, cbColorData);
    if (ipColors == IntPtr.Zero) return false;

    m_cc = new CHOOSECOLOR();
    m_cc.lStructSize = Marshal.SizeOf(m_cc);
    m_cc.hwndOwner = GetHwndFromControl(ctrlParent);
    m_cc.hInstance = IntPtr.Zero;
    m_cc.rgbResult = 0;
    m_cc.lpCustColors = ipColors;
    m_cc.Flags = CC_RGBINIT;
    m_cc.lCustData = 0;
    m_cc.lpfHook = IntPtr.Zero;
    m_cc.lpTemplateName = IntPtr.Zero;

    return true;
}
```



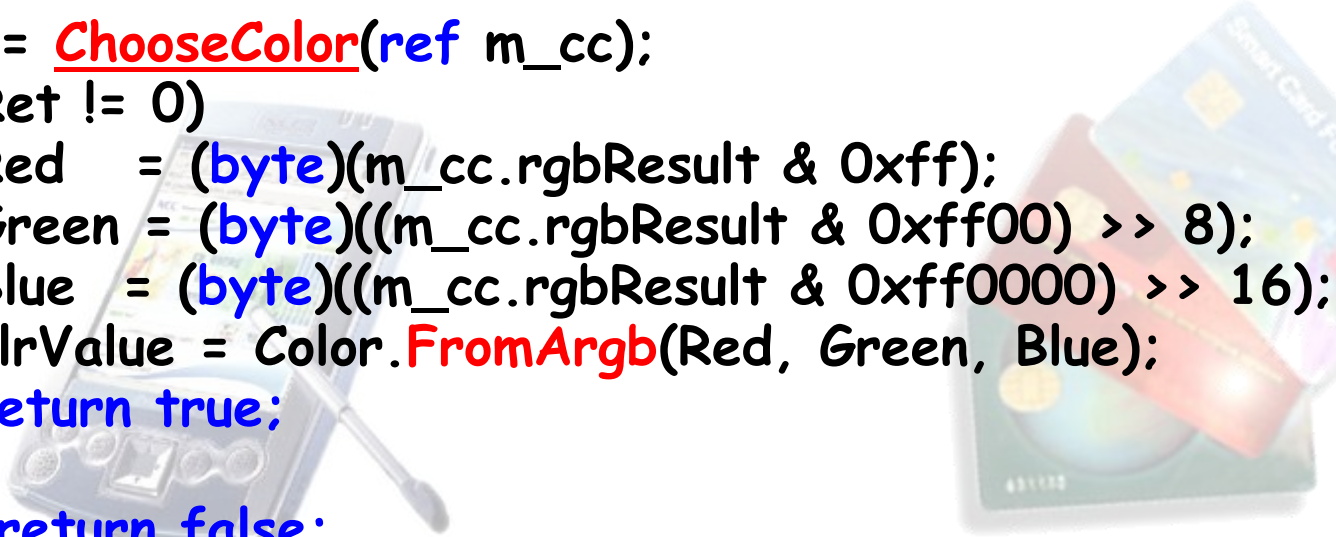
Phụ lục:

Hộp thoại chọn màu (gọi WinAPI bằng P/Invoke)

```
public bool ShowDialog (ref Color clrValue)
{
    int iRet = 0;
    byte Red = clrValue.R;
    byte Green = clrValue.G;
    byte Blue = clrValue.B;

    m_cc.rgbResult = (Blue << 16) + (Green << 8) + Red;

    iRet = ChooseColor(ref m_cc);
    if (iRet != 0)
    {
        Red = (byte)(m_cc.rgbResult & 0xff);
        Green = (byte)((m_cc.rgbResult & 0xff00) >> 8);
        Blue = (byte)((m_cc.rgbResult & 0xff0000) >> 16);
        clrValue = Color.FromArgb(Red, Green, Blue);
        return true;
    }
    else return false;
}
```



Phụ lục: Hộp thoại chọn màu (gọi WinAPI bằng P/Invoke)

//

// Focus functions

//

[DllImport("coredll.dll")]

public static extern IntPtr GetFocus ();

[DllImport("coredll.dll")]

public static extern IntPtr SetFocus (IntPtr hWnd);



Phụ lục: Hộp thoại chọn màu (gọi WinAPI bằng P/Invoke)

```
public IntPtr GetHwndFromControl (Control ctrl)
{
    IntPtr hwndControl;

    // Check whether the control has focus.
    if (ctrl.Focused)
    {
        hwndControl = GetFocus();
    }
    else
    {
        IntPtr ipFocus = GetFocus();
        ctrl.Focus();
        hwndControl = GetFocus();
        SetFocus(ipFocus);
    }
    return hwndControl;
}
```

