.NET GDI+

Overview

- GDI+ namespaces
- Size, Point, Rectangle, and Region
- System.Drawing.Graphics
- Examples

References

- Andrew Troelsen, “C# and the .NET Platform”, Apress, 2001

Core GDI+ Namespaces

GDI+ Namespace	Meaning
System.Drawing	This is the core GDI+ namespace, which defines numerous types for basic rendering as well as the Graphics type.
System.Drawing.Drawing2D	This namespace offers types used for more advanced 2D graphics functionality.
System.Drawing.Imaging	This namespace defines types that allow one to directly manipulate images.
System.Drawing.Printing	This namespace defines types that allow one to render images for printing.
System.Drawing.Text	This namespace allows one to manipulate collections of fonts.

Point(F)

Member	Meaning
+	These operators allow one to manipulate the underlying (x, y) point.
-	
==	
!=	
X	These properties allow one to get and set the underlying (x, y) values.
Y	
IsEmpty	This property returns true if X and Y are both set to zero.
Offset()	This method translates a given Point type by a given amount.

Rectangle(F)

Member	Meaning
== !=	These operators allow one to test whether two rectangles have identical values or not.
Inflate(), Intersect(), Union()	These static methods allow one to expand a rectangle, or create a new rectangle as a result of an intersection of union operation.
Top, Left, Bottom, Right	These properties set the dimensions of a rectangle.
Height, Width	Height and width of a rectangle.
Contains()	This method can be used to determine whether a given Point is within the bounds of a rectangle.
X, Y	These properties return the x and y coordinates of the Rectangle's upper left corner.

Size(F)

Member	Meaning
+	These operators allow one to manipulate the underlying Size type.
-	
==	
!=	
Height	These properties allow one to get and set the current dimension of a Size type.
Width	

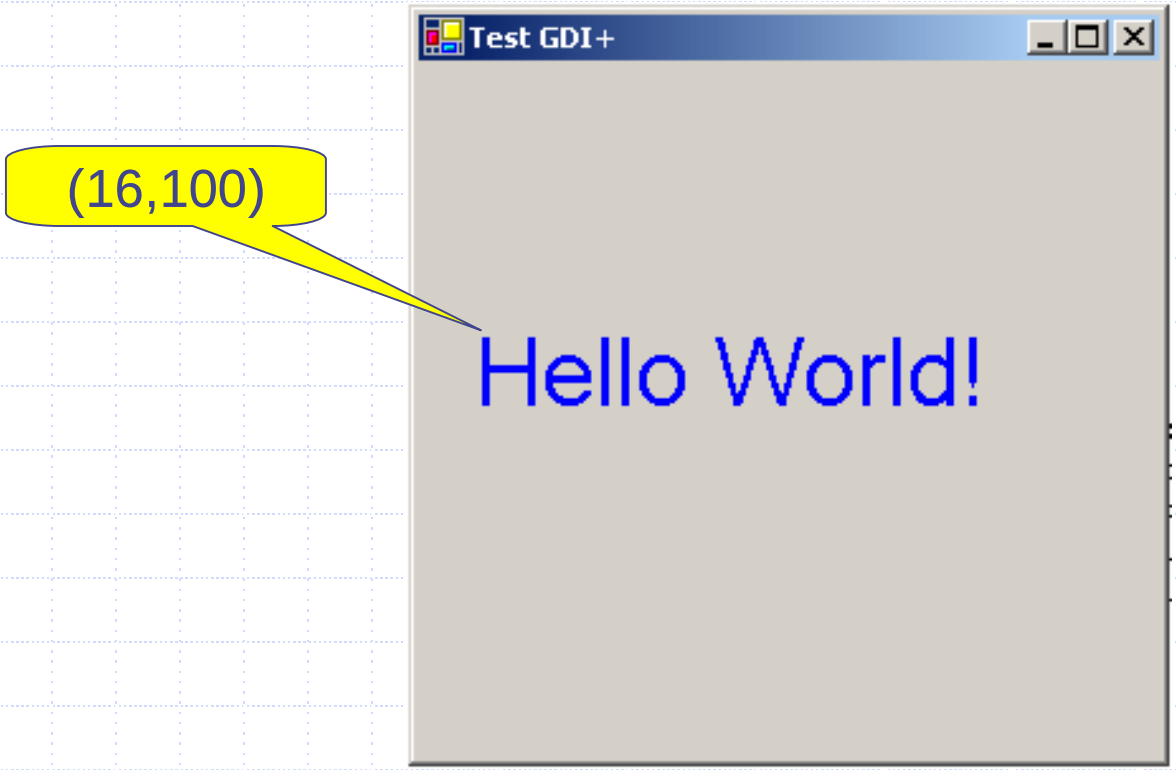
Region

Member	Meaning
Complement()	Updates this Region to the portion of the specified Graphics object that does not intersect with this Region.
Exclude()	Updates this Region to the portion of its interior that does not intersect with the specified Graphics object.
GetBounds()	Returns a RectangleF that represents a rectangular region that bounds this Region.
Intersect()	Updates this Region to the intersection of itself with the specified Graphics object.
Translate()	Offsets the coordinates of this Region by the specified amount.
Union()	Updates this Region of the union minus the intersection of itself with the specified Graphics object.

OnPaint

```
public class MainForm : Form
{
    protected override void OnPaint( PaintEventArgs e )
    {
        Graphics lGraphics = e.Graphics; // get HDC

        lGraphics.DrawString( “Hello World!”,                    // text
                              new Font( “Tacoma”, 28 ),          // font
                              new SolidBrush( Color.Blue ), // brush
                              16,                                // Point.X
                              100 );                             // Point.Y
    }
}
```

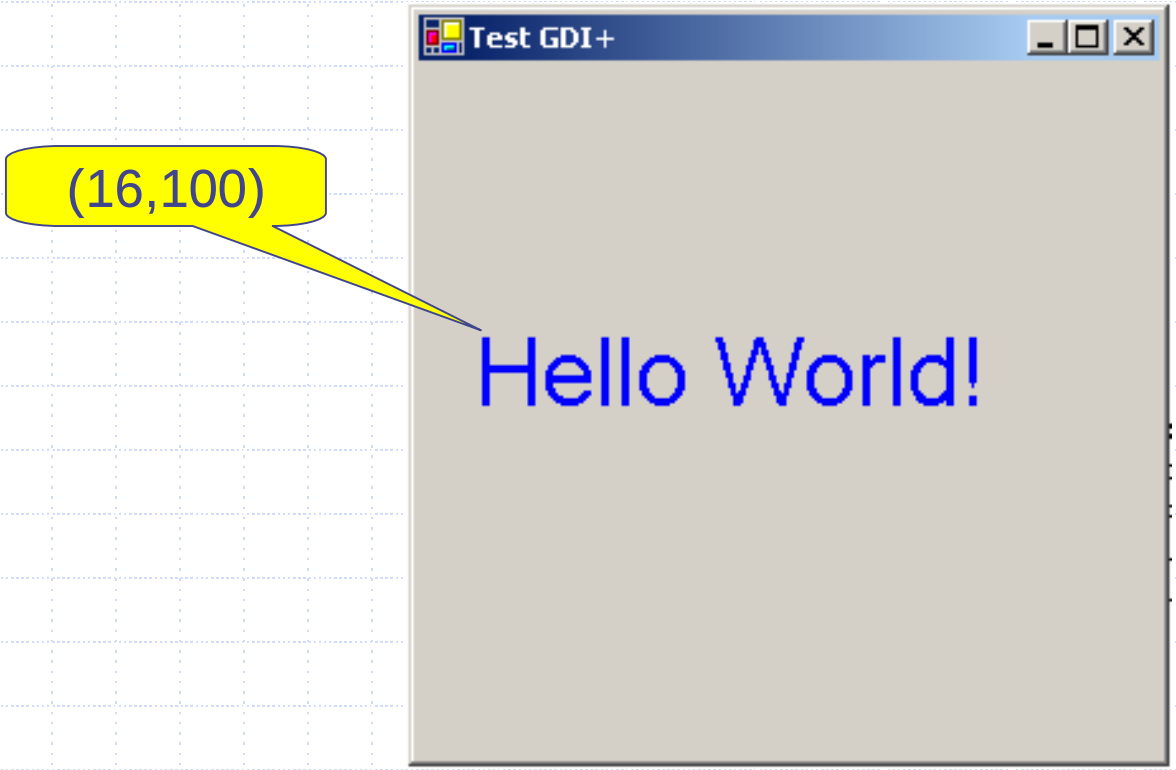
Output



Paint Event

```
public class MainForm : Form
{
    public MainForm()
    {  Paint += new PaintEventHandler( DoPaint );  }

    private void DoPaint( object sender, PaintEventArgs e )
    {
        Graphics lGraphics = e.Graphics;                    // get HDC
        lGraphics.DrawString( “Hello World!”,               // text
                              new Font( “Tacoma”, 28 ),     // font
                              new SolidBrush( Color.Blue ), // brush
                              16,                           // Point.X
                              100 );                        // Point.Y
    }
}
```

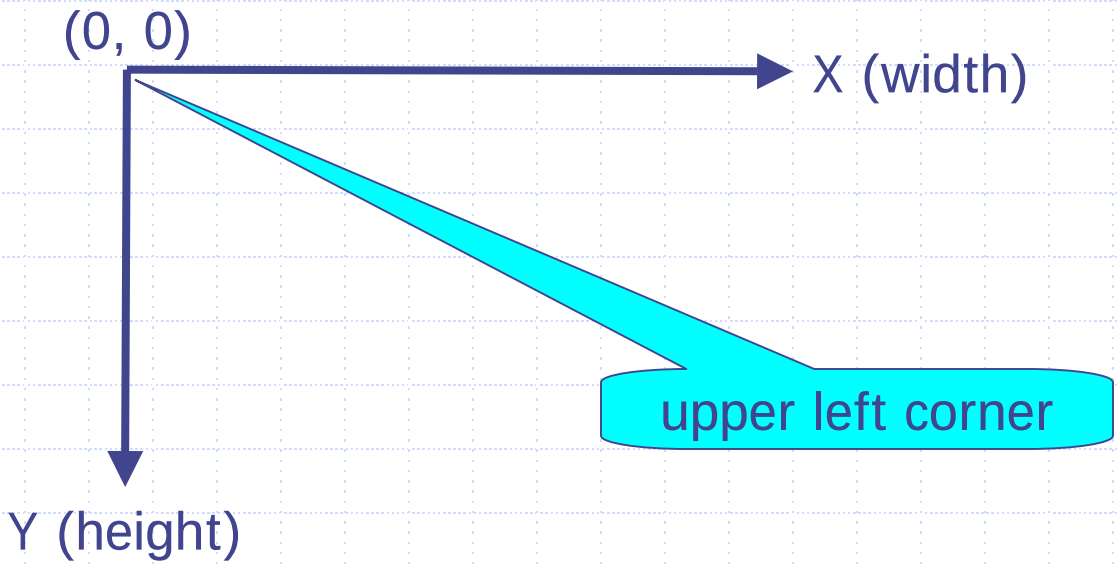
Output



Graphics

Member	Meaning
FromHDC(), FromHandle(), FromImage()	These static methods provide a way to obtain a valid Graphics object from a given image or GUI widget. These methods return a canvas for drawing.
Clear()	Fills a Graphics object with a specified color, erasing the current drawing surface.
DrawArc(), DrawEllipse(), DrawLine(), DrawLine(), DrawRectangle()	These methods (among others) are used to render a given image or geometric pattern.
FillEllipse(), FillPie(), FillRectangle()	These methods (among others) are used to fill the interior of a given geometric shape.

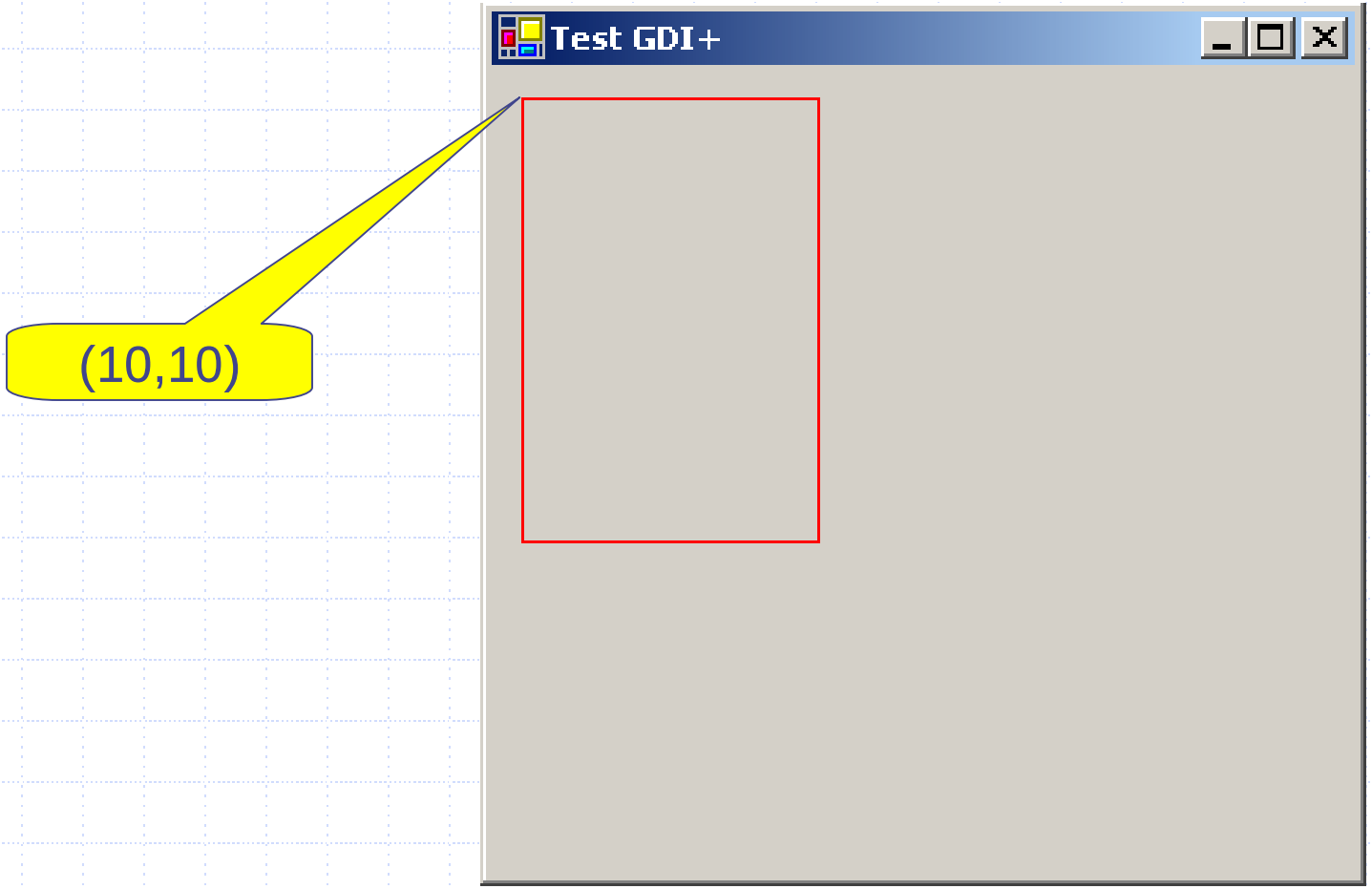
Coordinate System



DrawRectangle

```
public class MainForm : Form
{
    private void DoPaint( object sender,  PaintEventArgs e )
    {
        Graphics lGraphics = e.Graphics;                    // get HDC

        lGraphics.DrawRectangle( new Pen( Color.Red ),  // pen
                                 10,                    // X
                                 10,                    // Y
                                 100,                   // width
                                 150 );                 // height
    }
}
```

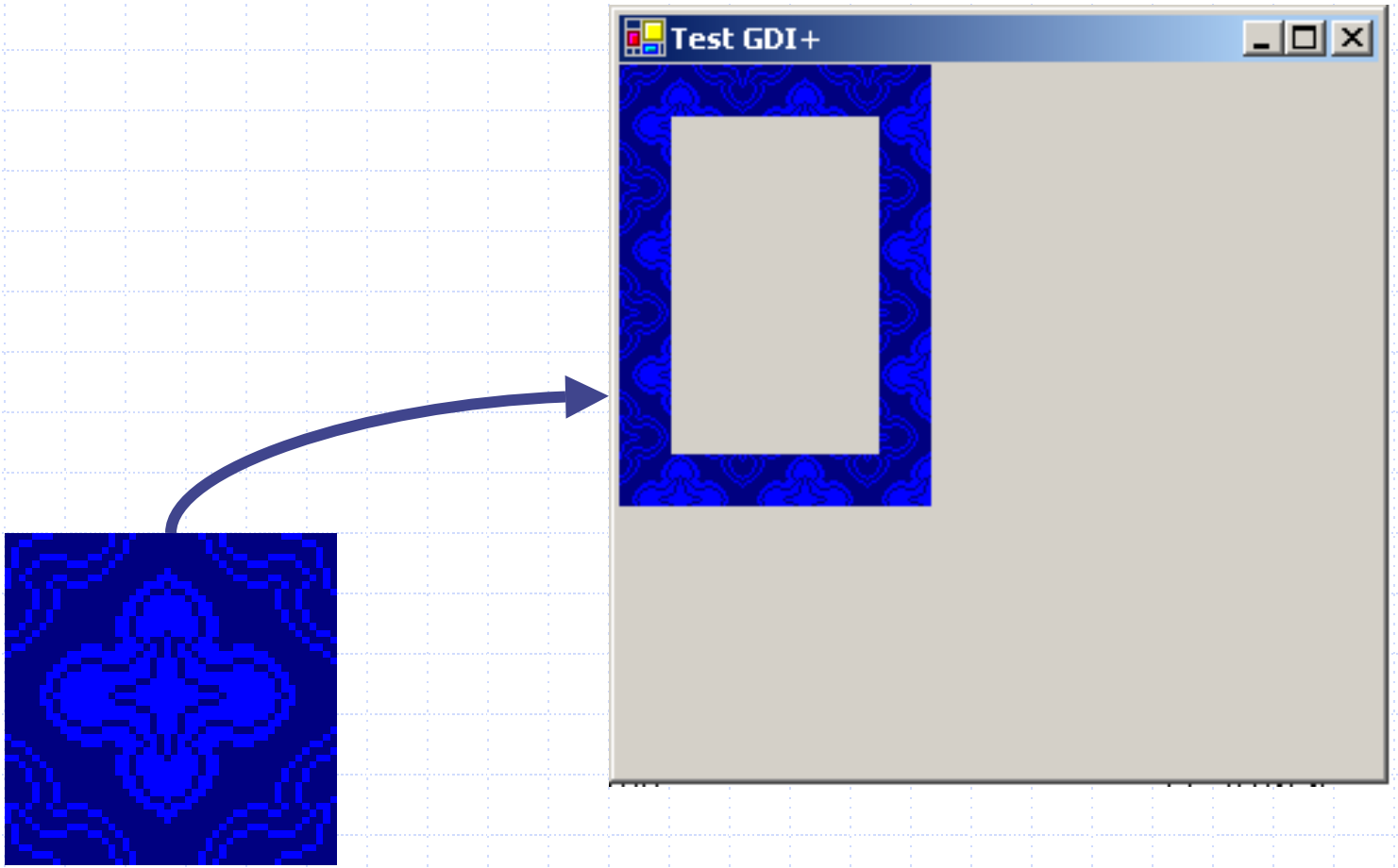
Output



Adorned DrawRectangle

```
public class MainForm : Form
{
    private void DoPaint( object sender,  PaintEventArgs e )
    {
        Graphics lGraphics = e.Graphics;                    // get HDC
        Image lImage = new Bitmap( @"c:\WINNT\Blue Lace 16.bmp" );
        Brush lBrush = new TextureBrush( lImage );

        lGraphics.DrawRectangle( new Pen( lBrush, 20.0F ),  // pen
                                 10,                        // X
                                 10,                        // Y
                                 100,                       // width
                                 150 );                     // height
    }
}
```

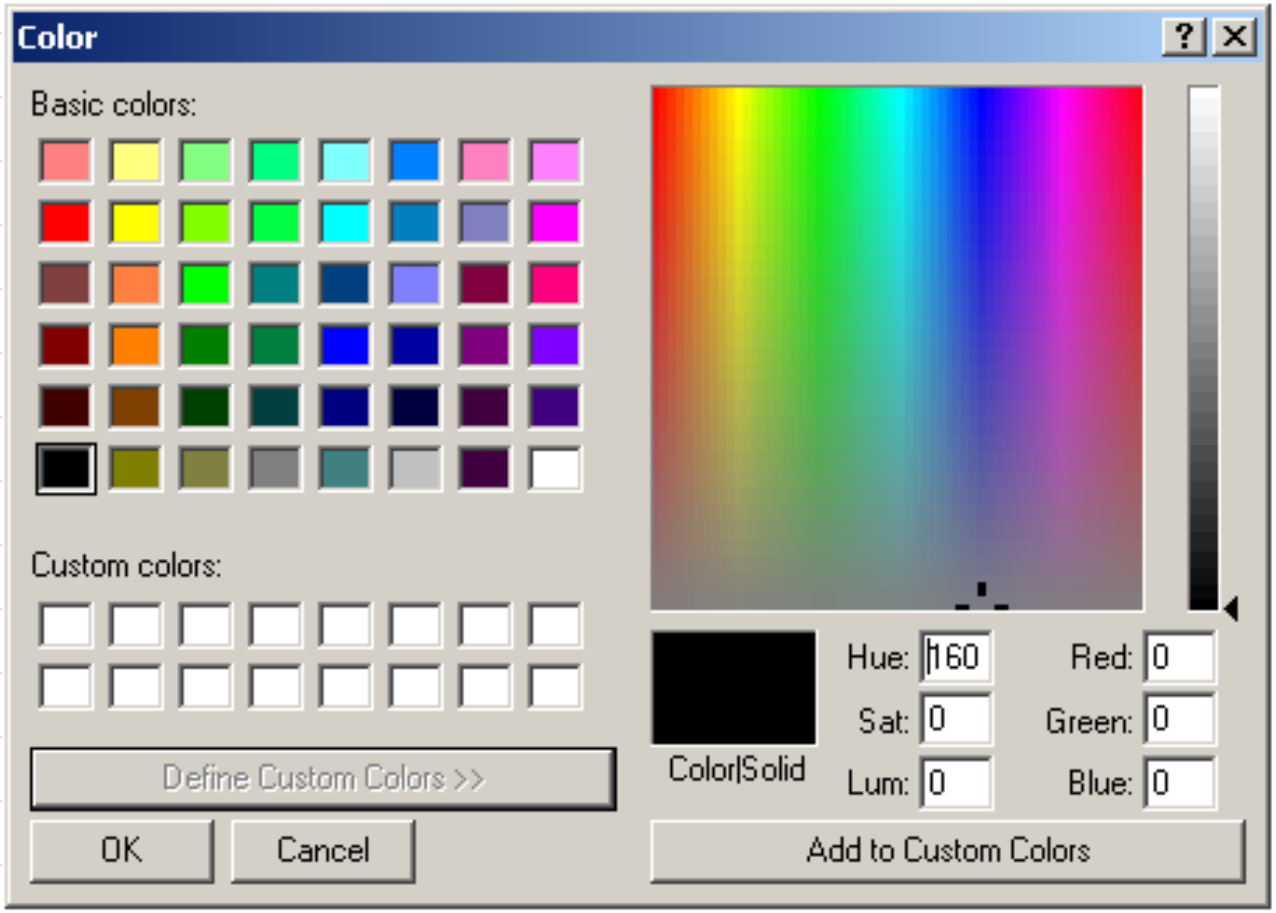
Output



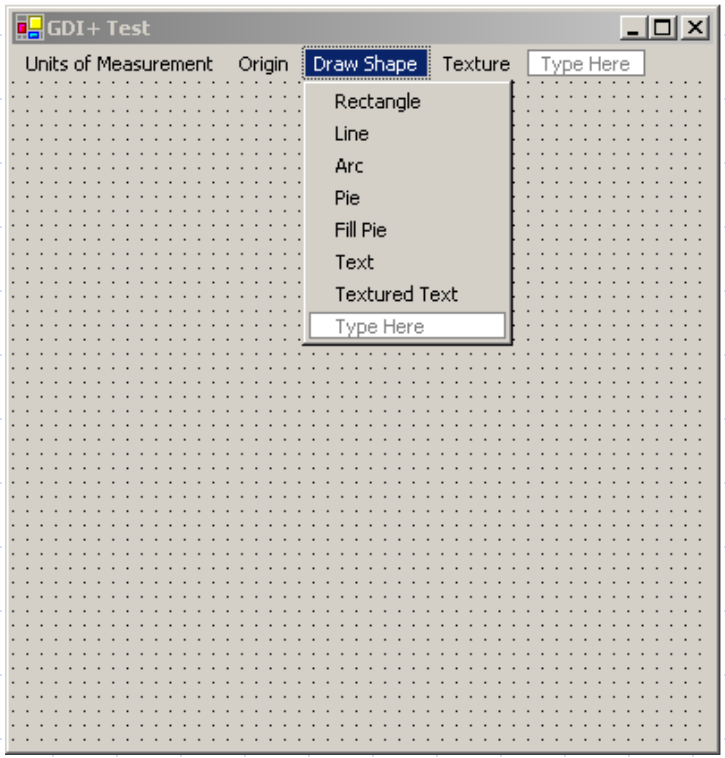
GraphicsUnit

Enumeration Value	Meaning
Display	Specifies 1/75 inch as the unit of measure.
Document	Specifies the document unit (1/300 inch) as the unit of measure.
Inch	Specifies the inch as the unit of measure.
Millimeter	Specifies the millimeter as the unit of measure.
Pixel	Specifies a device pixel as the unit of measure.
Point	Specifies a printer's point (1/72 inch) as the unit of measure.

Color Dialog



GDI+ Test

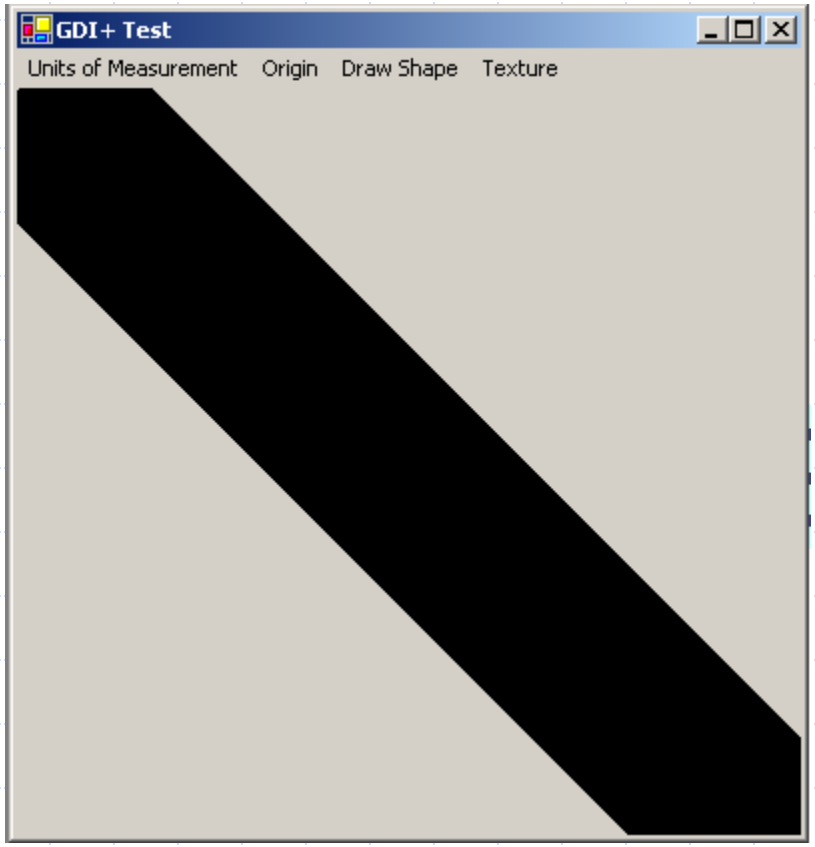


GDI+ Test illustrates some of the capabilities of Graphics.

DrawLine

```
// MenuItem (Line) Event handler
private void menuItem18_Click(object sender, System.EventArgs e)
{
    // acquire HDC using the Form's Handle property
    Graphics lGraphics = Graphics.FromHwnd( this.Handle );
    // set selected measurement
    lGraphics.PageUnit = fMeasurement;
    // create a pen using selected color
    Pen lPen = new Pen( fDefaultColor );
    // draw a line from fOrigin to (200, 200)
    lGraphics.DrawLine( lPen, fOrigin, new Point( 200, 200 ) );
}
```

Example

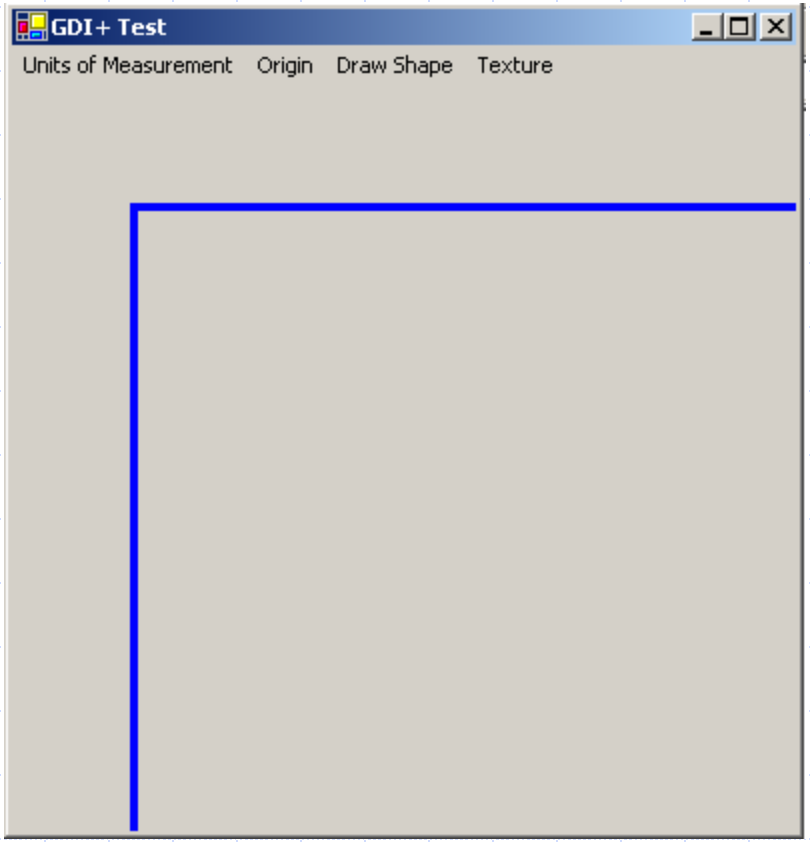


- Inch
- Origin (0, 0)
- Color.Black

DrawRectangle

```
// MenuItem (Rectangle) Event handler
private void menuItem16_Click(object sender, System.EventArgs e)
{
    // acquire HDC using the Form's Handle property
    Graphics lGraphics = Graphics.FromHwnd( this.Handle );
    // set selected measurement
    lGraphics.PageUnit = fMeasurement;
    // create a pen using selected color
    Pen lPen = new Pen( fDefaultColor );
    // draw a rectangle from fOrigin to (150, 150)
    lGraphics.DrawRectangle( lPen, fOrigin.X, fOrigin.Y, 150, 150 );
}
```

Example

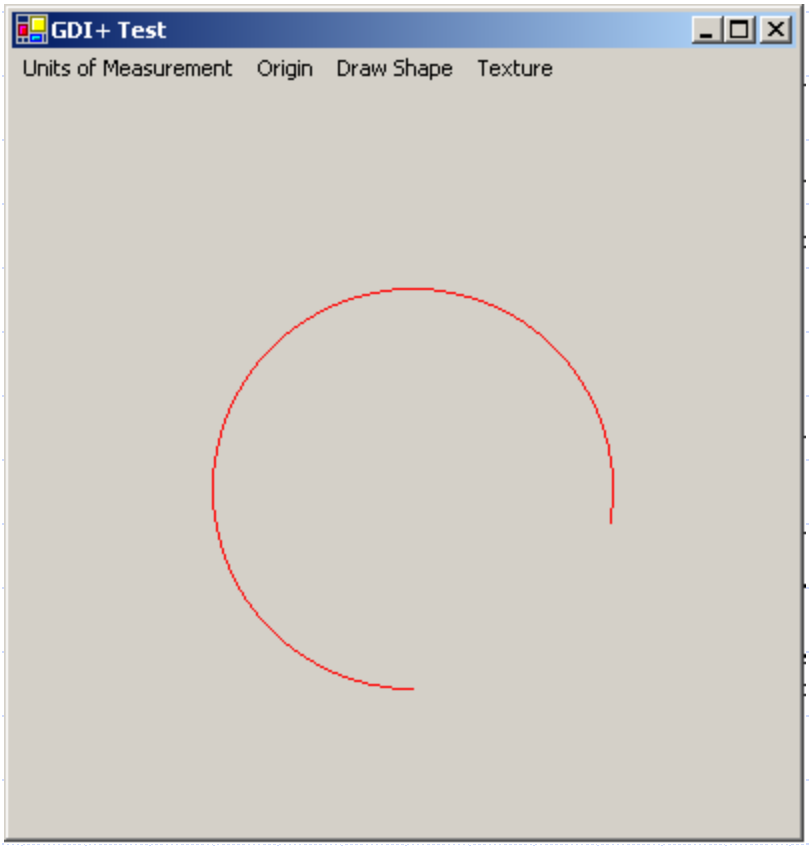


- Millimeter
- Origin (16, 16)
- Color.Blue

DrawArc

```
// MenuItem (Arc) Event handler
private void menuItem19_Click(object sender, System.EventArgs e)
{
    // acquire HDC using the Form's Handle property
    Graphics lGraphics = Graphics.FromHwnd( this.Handle );
    // set selected measurement
    lGraphics.PageUnit = fMeasurement;
    // create a pen using selected color
    Pen lPen = new Pen( fDefaultColor );
    // draw an arc from fOrigin with size (200, 200)
    lGraphics.DrawArc( lPen,
                       new Rectangle( fOrigin, new Size( 200, 200 ) ),
                       90.0F, 280.0F ); // startAngle, sweepAngle
}
```

Example

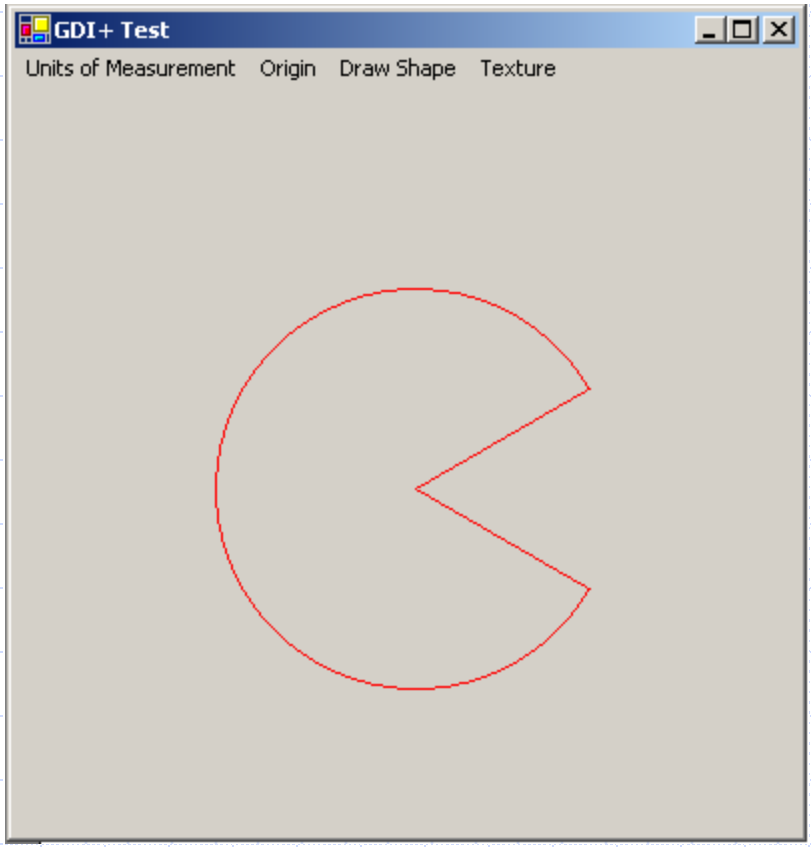


- Pixel
- Origin (100, 100)
- Color.Red

DrawPie

```
// MenuItem (Pie) Event handler
private void menuItem20_Click(object sender, System.EventArgs e)
{
    // acquire HDC using the Form's Handle property
    Graphics lGraphics = Graphics.FromHwnd( this.Handle );
    // set selected measurement
    lGraphics.PageUnit = fMeasurement;
    // create a pen using selected color
    Pen lPen = new Pen( fDefaultColor );
    // draw a pie from fOrigin with size (200, 200)
    lGraphics.DrawPie( lPen, fOrigin.X, fOrigin.Y, 200, 200,
                            30, 300 ); // startAngle, sweepAngle
}
```

Example

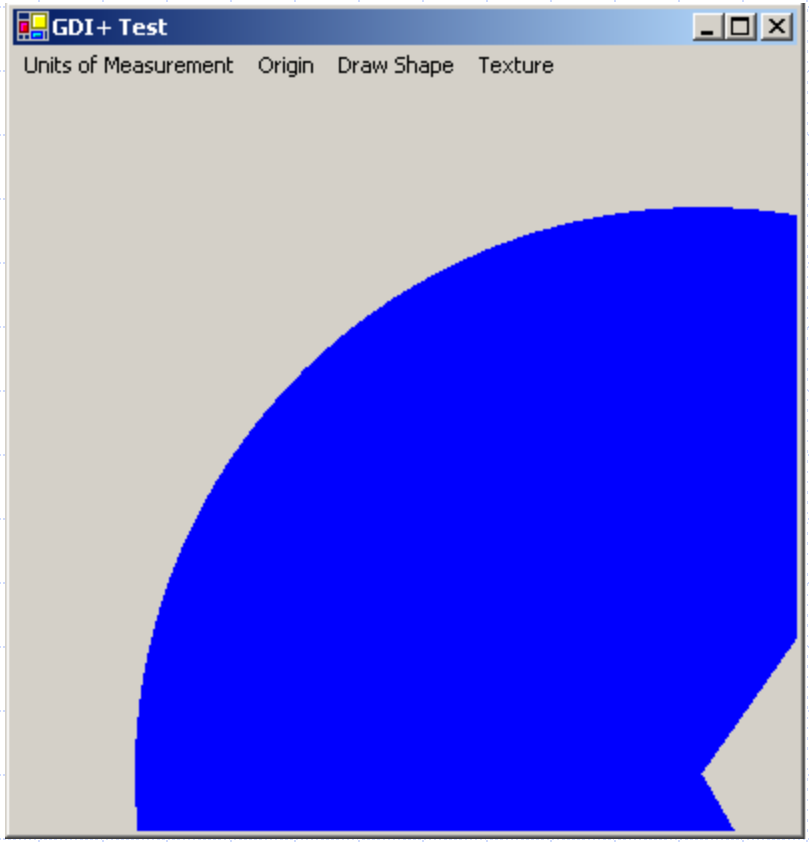


- Pixel
- Origin (100, 100)
- Color.Red

FillPie

```
// MenuItem (Fill Pie) Event handler
private void menuItem21_Click(object sender, System.EventArgs e)
{
    // acquire HDC using the Form's Handle property
    Graphics lGraphics = Graphics.FromHwnd( this.Handle );
    // set selected measurement
    lGraphics.PageUnit = fMeasurement;
    // create a brush using selected color
    Brush lBrush = new SolidBrush( fDefaultColor );
    // draw a filled pie from fOrigin with size (150, 150)
    lGraphics.FillPie( lBrush, fOrigin.X, fOrigin.Y, 150, 150,
                       60.0F, 245.0F ); // startAngle, sweepAngle
}
```

Example



- Millimeter
- Origin (16, 16)
- Color.Blue

DrawString

```
// MenuItem (Text) Event handler
private void menuItem22_Click(object sender, System.EventArgs e)
{
    // acquire HDC using the Form's Handle property
    Graphics lGraphics = Graphics.FromHwnd( this.Handle );
    // set selected measurement
    lGraphics.PageUnit = fMeasurement;
    // draw a string from fOrigin using Tacoma, 14pt
    lGraphics.DrawString( "This is a GDI+ Text string!",
                          new Font( "Tacoma", 14 ),
                          new SolidBrush( fDefaultColor ),
                          fOrigin );
}
```

Example

- Millimeter
- Origin (16, 16)
- Color.Purple

Drawstring With Texture

```
// MenuItem (Textured Text) Event handler
private void menuItem23_Click(object sender, System.EventArgs e)
{
    // acquire HDC using the Form's Handle property
    Graphics lGraphics = Graphics.FromHwnd( this.Handle );
    // set selected measurement
    lGraphics.PageUnit = fMeasurement;
    // draw a textured string from fOrigin using Tacoma, 44pt
    Image lTextureBrushImage =
        new Bitmap( @"c:\WINNT\" + fTextureName );
    Brush lBrush = new TextureBrush( lTextureBrushImage );
    lGraphics.DrawString( "This is a GDI+ Text string!",
                          new Font( "Tacoma", 44 ),
                          lBrush, fOrigin );
}
```

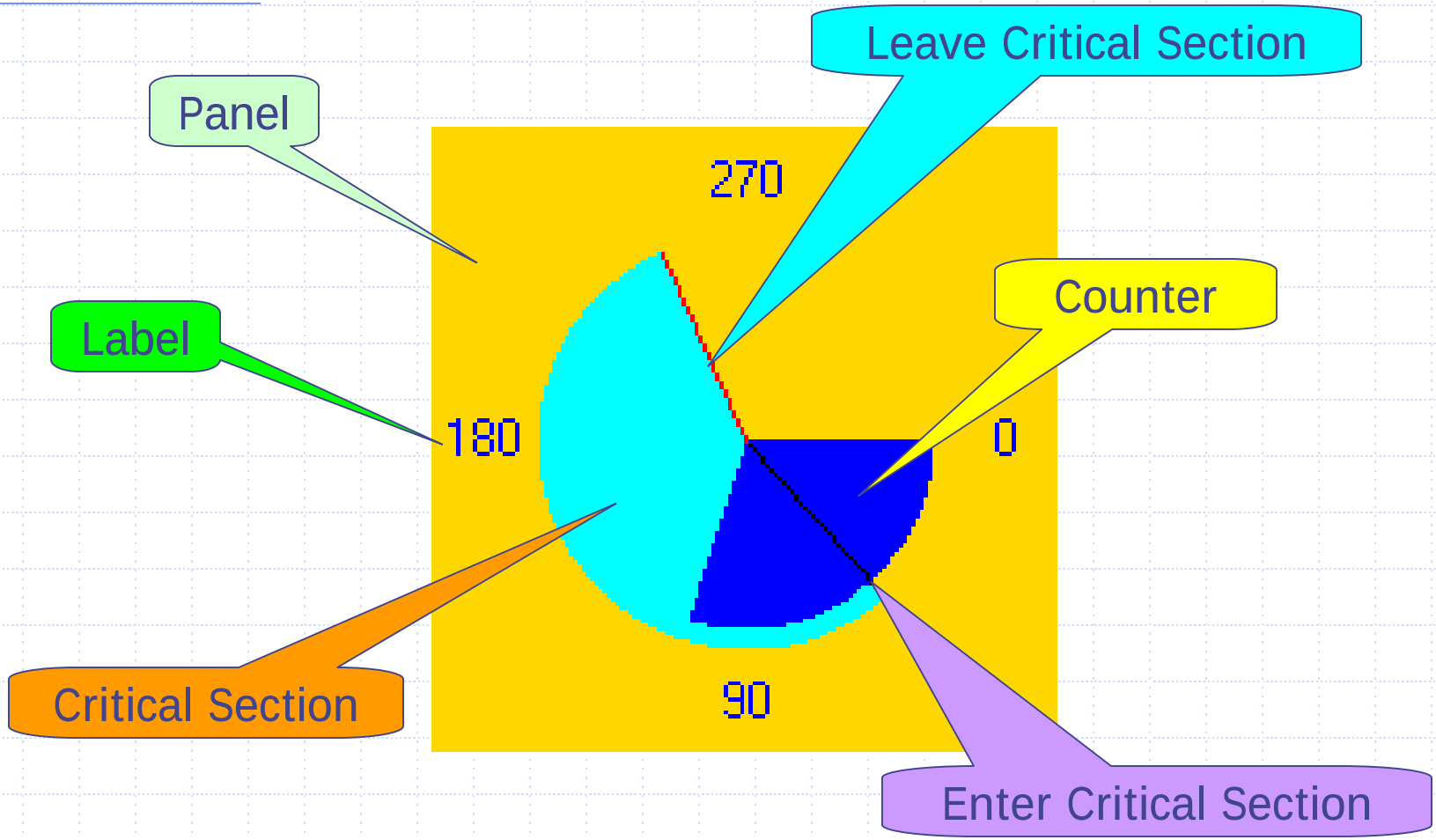
Example

- Inch
- Origin (0, 0)
- Feather

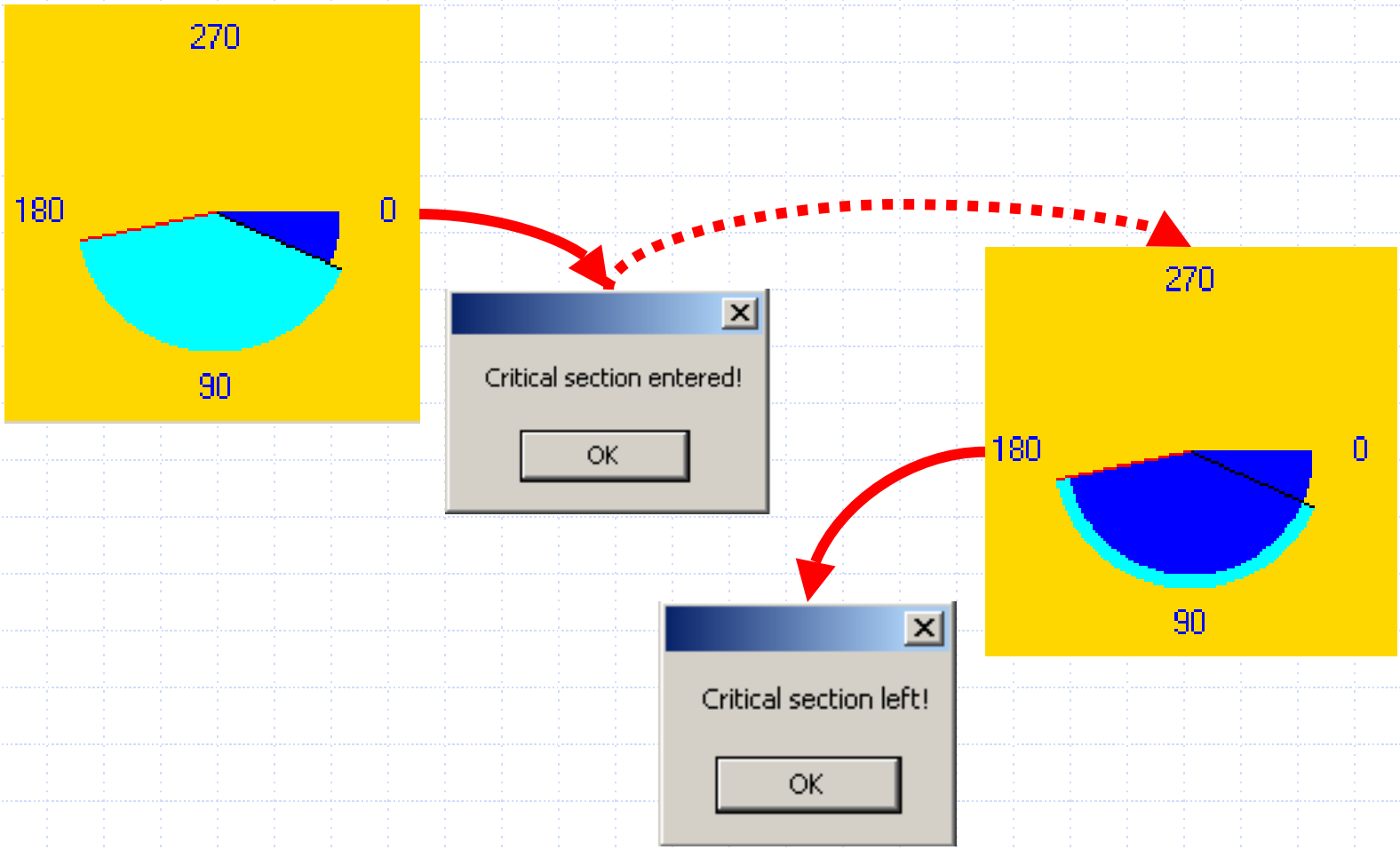
Building a User Control

- We want to define a simple user control that provides a graphical user interface to a counter with a critical region.
- The counter is represented by pie shape that grows over time.
- The critical section is also represented by a pie shape. The user should be able to change the bounds of the critical section using the mouse.
- If the counter enters the critical section, then the control must raise an “EnterSection” event. Similarly, if the counter leaves the critical section, then the control must raise an “LeaveSection” event.

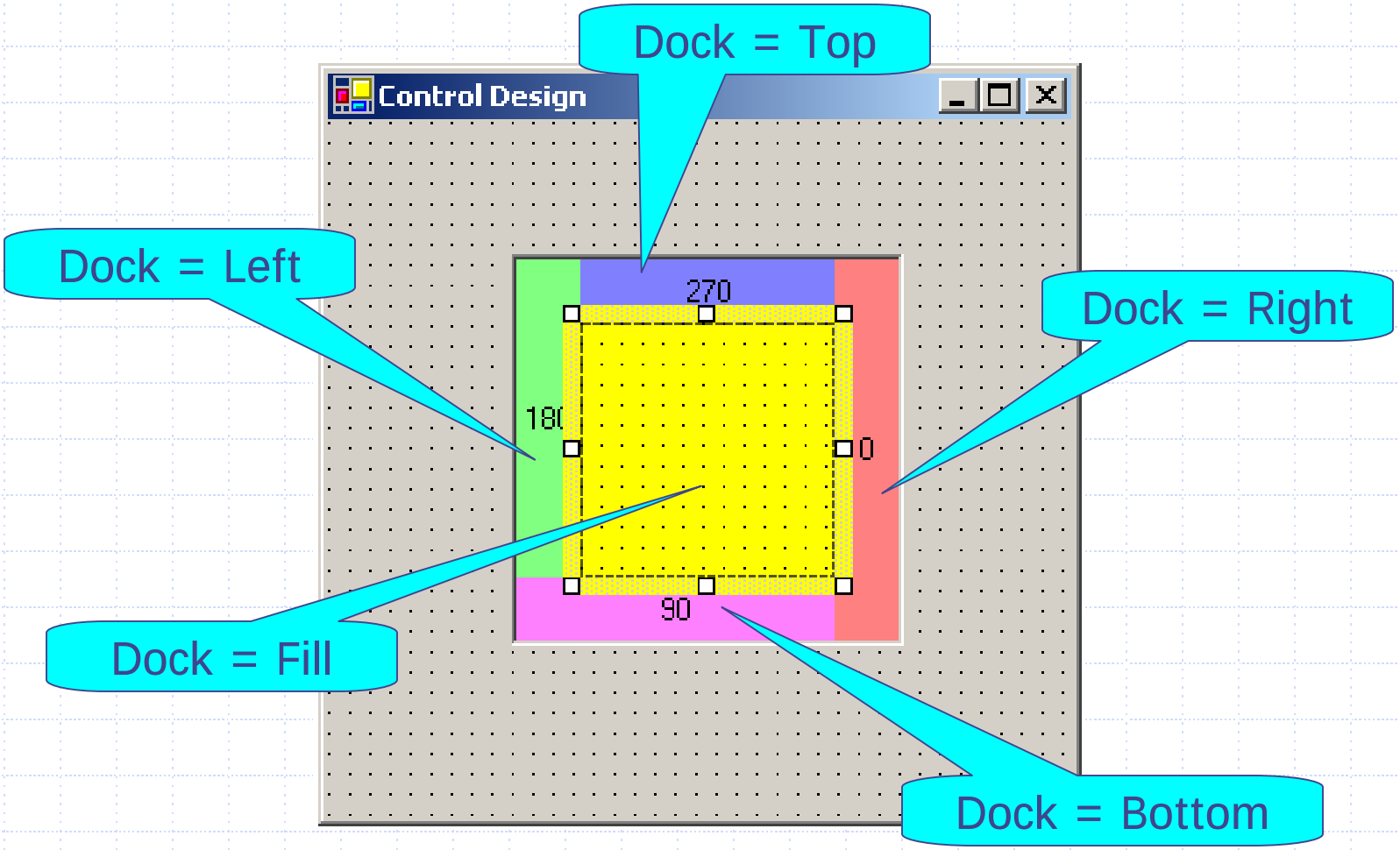
Design


Leave Critical Section
Panel
270
Counter
Label
180
0
90
Critical Section
Enter Critical Section

Runtime


270
180
0
90
Critical section entered!
OK
270
180
0
90
Critical section left!
OK

Graphical Design



ArcPanel Components

```
public class ArcPanel : System.Windows.Forms.Panel
{
    private System.Windows.Forms.Label fNorthLabel;
    private System.Windows.Forms.Label fEastLabel;
    private System.Windows.Forms.Label fSouthLabel;
    private System.Windows.Forms.Label fWestLabel;
    private System.Windows.Forms.Panel fDrawingPanel;
}
```

Component Configuration

```
private void InitializeControl()
{
    // create elements
    fNorthLabel = new Label();
    fEastLabel = new Label();
    fSouthLabel = new Label();
    fWestLabel = new Label();
    fDrawingPanel = new Panel();
    // suspend layout
    SuspendLayout();
    ...
    // resume layout
    ResumeLayout(false);
}
```

Label Configuration

```
// fEastabel
fEastLabel.Dock = DockStyle.Right;
fEastLabel.ForeColor = Color.Blue;
// use ClientRectangle as reference size
fEastLabel.Location = new Point( ClientRectangle.Width - 25, 25 );
fEastLabel.Size = new Size( 25, ClientRectangle.Height - 50 );
fEastLabel.Text = "0";
fEastLabel.TextAlign = ContentAlignment.MiddleCenter;
```

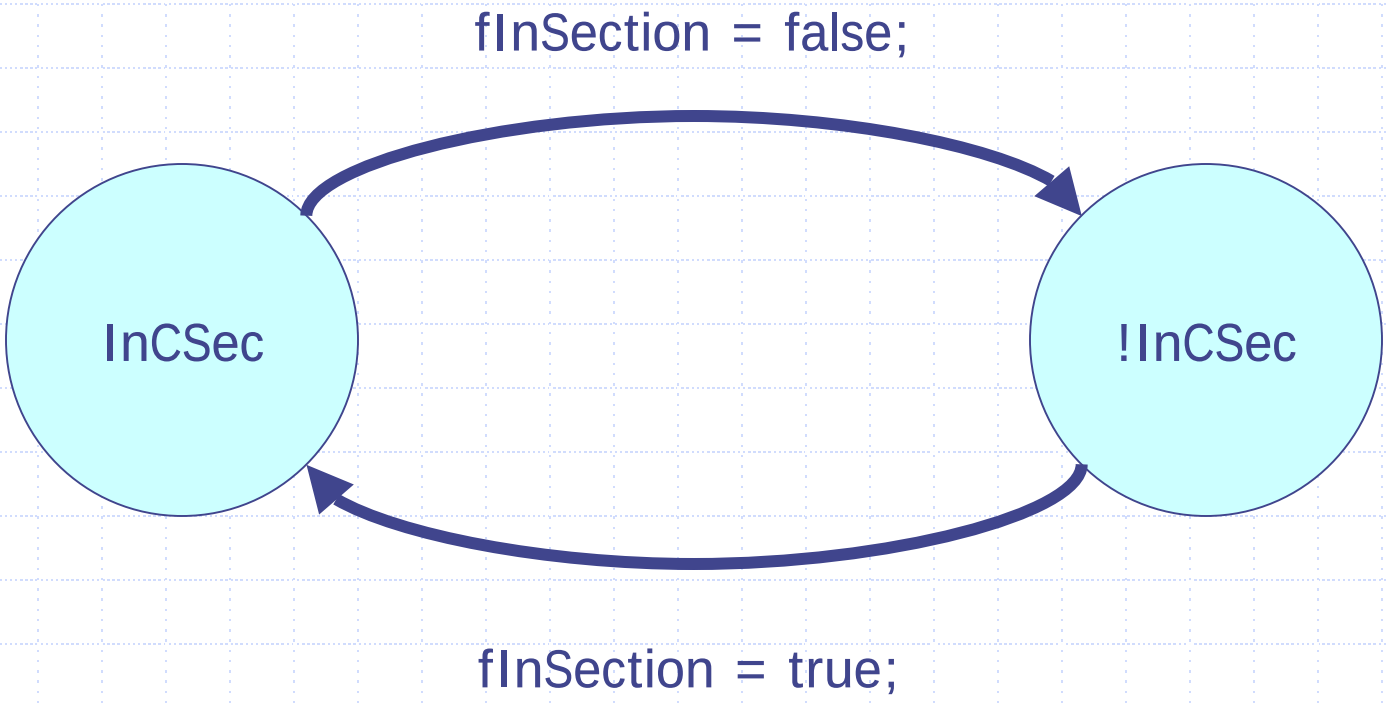
DrawingPanel Configuration

```
// place panel that hosts counter in the middle
fDrawingPanel.Dock = DockStyle.Fill;
fDrawingPanel.Location = new Point( 25, 25 );
fDrawingPanel.Size = new Size( ClientRectangle.Width - 50,
                               ClientRectangle.Height - 50 );

// register event handlers
fDrawingPanel.Paint +=                    // repaint counter
    new PaintEventHandler( HandlePaint );
fDrawingPanel.MouseDown +=                // start change critical section
    new MouseEventHandler( HandleMouseDown );
fDrawingPanel.MouseUp +=                  // stop change critical section
    new MouseEventHandler( HandleMouseUp );
fDrawingPanel.MouseMove +=                // change critical section
    new MouseEventHandler( HandleMouseMove );
```

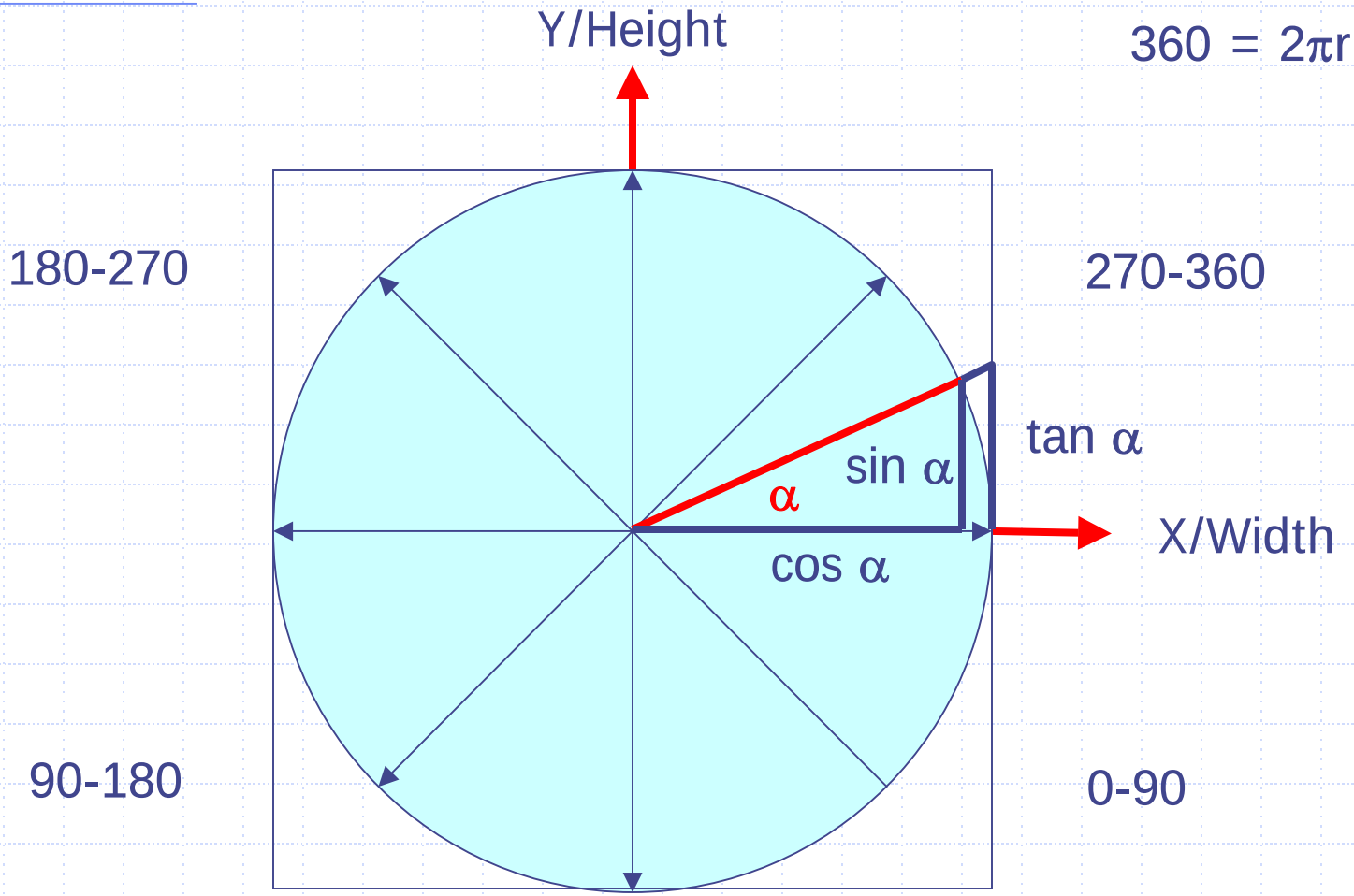
Events

```
public event EventHandler EnterSection;
public event EventHandler LeaveSection;

private void DoEnterSection() {
    if ( EnterSection != null ) {
        EnterSection( this, new EventArgs() );
        fInSection = true; }
}

private void DoLeaveSection() {
    if ( LeaveSection != null ) {
        LeaveSection( this, new EventArgs() );
        fInSection = false; }
}
```

States



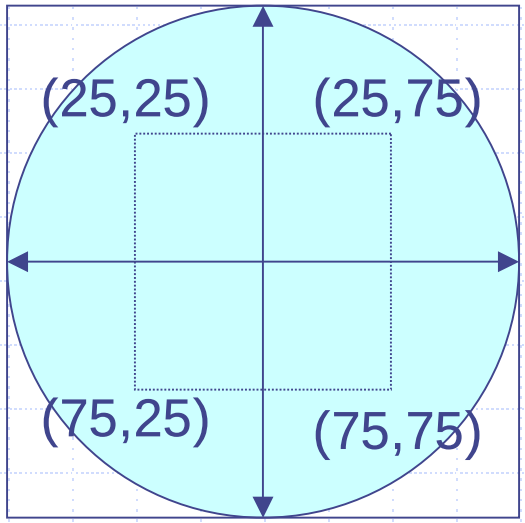
.NET Geometry



Determine Selected Angle

```
private int GetSelectedAngle( Point aPoint )
{
    // X coordinate relative to the center of the panel
    int lXDelta = aPoint.X - (fDrawingPanel.ClientSize.Width / 2);
    // Y coordinate relative to the center of the panel
    int lYDelta = (fDrawingPanel.ClientSize.Height / 2) - aPoint.Y;
    // Atans returns the angle whose tangent is the
    // quotient of two specified numbers.
    double lSelAlpha = Math.Atan2( (double)lYDelta, (double)lXDelta );
    // translate quadrants, .NET uses clockwise order
    return 360 - (((int)(lSelAlpha * 180.0D / Math.PI) + 360 ) % 360);
}
```

Selected Angle



Y/Heigth = 100

X/Width = 100

(25,75): $\Delta X = 75 - 50 = 25$, $DY = 50 - 25 = 25$, $\tan \alpha = \Delta Y / \Delta X = 1$
(25,75) = 45^O, but in .NET (25,75) is in the fourth quadrant
➔ $360 - 45 = 315^O$

HandleMouseUp

```
private void HandleMouseUp( object sender, MouseEventArgs e )
{
    // we only handle the left button
    if ( e.Button == MouseButtons.Left )
    {
        int lAngle = GetSelectedAngle( new Point( e.X, e.Y ) );
        lAngle = lAngle + STEP - (lAngle % STEP); // adjust angle
        if ( fStartSelected )
            SetStartSection( lAngle );
        if ( fStopSelected )
            SetStopSection( lAngle );
        Cursor = Cursors.Default;
    }
}
```

HandlePaint

```
private void HandlePaint( object sender, PaintEventArgs e )
{
    Graphics lGraphics = e.Graphics;
    Rectangle lClientRec = fDrawingPanel.ClientRectangle;


    ...
    // draw actual arc
    lGraphics.FillPie( new SolidBrush( fArcColor ),
                       5, 5,
                       lClientRec.Width - 10, lClientRec.Height - 10,
                       fStart, fAngle );
    ...
}
```