

Pengantar Pemrograman C#

Pengantar C#

Untuk mempelajari dasar-dasar pemrograman C# anda dapat mendownload tutorial C# pada website yang sudah disediakan. (check: <http://actualtraining.wordpress.com>), kategori C#.

Menggunakan Array 2D pada C#

Representasi koordinat image adalah dalam matrix, maka sebelum anda masuk kedalam topik computer graphic anda harus menguasai pembuatan matrix terlebih dahulu.

Contoh 1.1 Menginputkan Data Kedalam Matrix 2D

```
//mendeklarasikan matrix 2x2
//index array pada c# dimulai dari 0
int[,] intArr1 = new int[2, 2];

//cara untuk mengisi arraynya
intArr1[0, 0] = 12;
intArr1[0, 1] = 23;
intArr1[1, 0] = 14;
intArr1[1, 1] = -9;

//untuk menampilkan array
//for pertama adalah representasi baris
for (int i = 0; i < 2; i++)
{
    //for kedua adalah representasi kolom
    for (int j = 0; j < 2; j++)
    {
        Console.Write(intArr1[i, j] + " ");
    }
    Console.WriteLine();
}
```

Contoh 1.2 Operasi Penjumlahan Matrix

```
//untuk melakukan penjumlahan matrix, baris dan kolom matrix1 dan matrix 2 harus sama
int intBaris = 0, intKolom = 0;
Console.Write("Masukan Baris Matrix :");
intBaris = Convert.ToInt32(Console.ReadLine());
Console.Write("Masukan Banyak Kolom :");
intKolom = Convert.ToInt32(Console.ReadLine());

int[,] intMatrix1 = new int[intBaris, intKolom];
int[,] intMatrix2 = new int[intBaris,intKolom];
int[,] intMatrixHasil = new int[intBaris,intKolom];

//mengisi Matrix1
Console.WriteLine("Input Matrix1");
for (int b = 0; b < intBaris; b++)
{
    for (int k = 0; k < intKolom; k++)
    {
        Console.Write("Masukan Matrix1[" + b + "," + k + "] : ");
    }
}
```

```

        intMatrix1[b, k] = Convert.ToInt32(Console.ReadLine());
    }
}

//mengisi Matrix2
Console.WriteLine("Input Matrix2");
for (int b = 0; b < intBaris; b++)
{
    for (int k = 0; k < intKolom; k++)
    {
        Console.Write("Masukan Matrix2[" + b + "," + k + "] : ");
        intMatrix2[b, k] = Convert.ToInt32(Console.ReadLine());
    }
}

//menjumlahkan Matrix1 dan Matrix2
for (int b = 0; b < intBaris; b++)
{
    for (int k = 0; k < intKolom; k++)
    {
        intMatrixHasil[b, k] = intMatrix1[b, k] + intMatrix2[b, k];
    }
}

//menampilkan hasil penjumlahan matrix
Console.WriteLine("Hasil Penjumlahan Matrix :");
for (int b = 0; b < intBaris; b++)
{
    for (int k = 0; k < intKolom; k++)
    {
        Console.Write(intMatrixHasil[b, k] + " ");
    }
    Console.WriteLine();
}

```

Contoh 1.3 Operasi Perkalian Dua Matrix

```

//untuk perkalian matrix syaratnya adalah kolom matrix1 harus sama dengan baris
matrix2

//hasil dari perkalian matrixnya adalah baris matrix1 dan kolom matrix2
// m1[b1,k1] x m2[b2,k2] = m3[b1,k2]
int intBMat1, intKmlBm2, intKMat2;
Console.Write("Masukan baris matrix1 : ");
intBMat1 = Convert.ToInt32(Console.ReadLine());
Console.Write("Masukan kolom matrix1 dan baris matrix2 : ");
intKmlBm2 = Convert.ToInt32(Console.ReadLine());
Console.Write("Masukan kolom matrix2 : ");
intKMat2 = Convert.ToInt32(Console.ReadLine());

int[,] matrix1 = new int[intBMat1, intKmlBm2];
int[,] matrix2 = new int[intKmlBm2, intKMat2];
int[,] matrixHasil = new int[intBMat1, intKMat2];
//mengisi matrix1
for (int x = 0; x < intBMat1; x++)
{
    for (int y = 0; y < intKmlBm2; y++)
    {

```

```
        Console.WriteLine("Masukan matrix1[" + x + "," + y + "] : ");
        matrix1[x, y] = Convert.ToInt32(Console.ReadLine());
    }
}

//mengisi matrix2
for (int x = 0; x < intKmlBm2; x++)
{
    for (int y = 0; y < intKMat2; y++)
    {
        Console.WriteLine("Masukan matrix2[" + x + "," + y + "] : ");
        matrix2[x, y] = Convert.ToInt32(Console.ReadLine());
    }
}

Console.WriteLine();

//mengalikan matrix
for (int x = 0; x < intBMat1; x++)
{
    for (int y = 0; y < intKMat2; y++)
    {
        matrixHasil[x, y] = 0;
        for (int k = 0; k < intKmlBm2; k++)
        {
            matrixHasil[x, y] =
                matrixHasil[x, y] + matrix1[x, k] * matrix2[k, y];
        }
    }
}

Console.WriteLine();
//hasil perkalian matrik
for (int x = 0; x < intBMat1; x++)
{
    for (int y = 0; y < intKMat2; y++)
    {
        Console.WriteLine(matrixHasil[x, y] + " ");
    }
    Console.WriteLine();
}
}
```

GDI+ (Next-Generation Graphics Interface)

Apa itu GDI+

GDI+ adalah library pada .NET yang digunakan untuk membuat aplikasi grafis berbasis Windows dan Web yang dapat berinteraksi dengan perangkat grafis berupa printer atau monitor.

Graphical User Interface (GUI) berinteraksi dengan perangkat hardware seperti monitor, printer, atau scanner untuk menampilkan format yang dapat dibaca oleh manusia, tetapi tidak ada program yang dapat mengakses perangkat hardware tersebut secara langsung, maka dibuat user interface agar pengguna dapat berinteraksi dengan perangkat-perangkat tersebut.

Untuk membuat user interface tersebut harus digunakan third component sebagai jembatan antara program dan perangkat keras, maka dibuatlah komponen GDI+ library, dengan komponen tersebut anda dapat menampilkan output berupa text dan gambar ke perangkat hardware.

Pada aplikasi .NET anda dapat menggunakan GDI+ untuk membuat aplikasi grafis baik untuk aplikasi berbasis windows maupun aplikasi web. Library GDI+ sudah tersinstall secara default pada sistem operasi Windows XP, Vista, dan Windows 2003 yang berlokasi di class library dengan nama `Gdiplus.dll`.



Gambar Arsitektur GDI+

Apa saja fitur GDI+?

Fitur GDI+ dapat kategorikan dalam 3 fungsi utama yaitu:

- 2D vector graphic
- Imaging
- Typograh

2D Vector Graphics Programming

Vector Graphic merupakan komponen pembentuk bentuk gambar mis (kotak, lingkaran) yang dibentuk dari kumpulan titik pada sistem koordinat.

Pada .NET Framework library 2D vector graphic programming dibagi menjadi dua kategori yaitu general dan advanced. General 2D vector graphic didefinisikan dalam library `System.Drawing` namespace dan advance function didefinisikan dalam library `System.Drawing.Drawing2D` namespace.

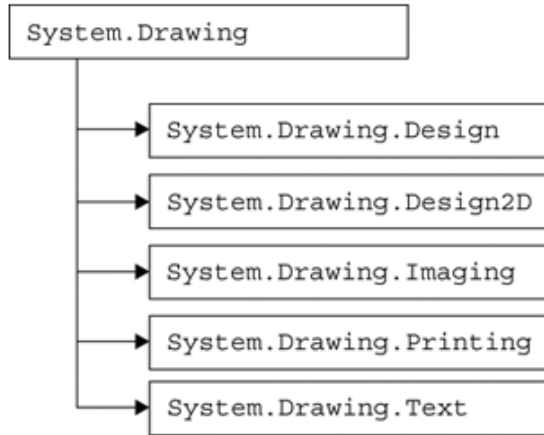
Imaging

Imaging digunakan untuk memanipulasi image. Image file yang dapat dimanipulasi misalnya .bmp, .jpg, .gif, dan .png. Fungsi-fungsi untuk memanipulasi images ada dalam Image class yang digunakan untuk create, save, dan load images.

Typography

Typography digunakan untuk mendesign dan merepresentasikan text. GDI+ menyediakan class untuk membuat dan menggunakan font.

GDI+ Namespaces dan Class dalam .NET



Class-class yang ada pada System.Drawing.Namespaces

Class	Description
Bitmap	Encapsulates a bitmap, which is an image (with its properties) stored in pixel format.
Brush	An abstract base class that cannot be instantiated directly. The Brush class provides functionality used by its derived brush classes and represents a brush graphics object. A brush is used to fill the interior of a graphical shape with a specified color.
Brushes	Represents brushes with all the standard colors. This class has a static member for each standard color. For example, Brushes.Blue represents a blue brush.
ColorConverter	Provides methods and properties to convert colors from one type to another.
ColorTranslator	Provides various methods to translate colors from one type to another.
Font	Provides members to define the format of font text, name, face, size, and styles. The Font class also provides methods to create a Font object from a window handle to a device context or window handle.
FontConverter	Provides members that convert fonts from one type to another.
FontFamily	Defines a group of typefaces having a similar basic design and certain variations in styles.
Graphics	A key class that encapsulates drawing surfaces. Among many other things, the Graphics class provides members to draw and fill graphical objects.
Icon	Represents a Windows icon. The Icon class provides members to define the size, width, and height of an icon.
IconConverter	Provides members to convert an Icon object from one type to another.
Image	Provides members to define the size, height, width, and format of an image. The Image class also provides methods to create Image objects from a file, a window handle, or a stream; and to save, rotate, and flip images. It is an abstract base class, and its functionality is used through its derived classes: Bitmap, Icon, and Metafile.
ImageAnimator	Provides methods to start and stop animation, and to update frames for an image that has time-based frames.
ImageConverter	Provides members to convert Image objects from one type to another.
ImageFormatConverter	Defines members that can be used to convert images from one format to another.
Pen	Defines a pen with a specified color and width. A pen is used to draw graphical objects such as a line, a rectangle, a curve, or an ellipse.

Class	Description
Pens	Provides static members for all the standard colors. For example, Pens.Red represents a red pen.
PointConverter	Defines members that can be used to convert Point objects from one type to another.
RectangleConverter	Defines members that can be used to convert Rectangle objects from one type to another.
Region	Represents a region in GDI+, which describes the interior of a graphics shape.
SizeConverter	Defines members that can be used to convert size from one type to another.
SolidBrush	Inherited from the Brush class. This class defines a solid brush of a single color.
StringFormat	Provides members to define text format, including alignment, trimming and line spacing, display manipulations, and OpenType features.
SystemBrushes	Defines static properties. Each property is a SolidBrush object with a Windows display element such as Highlight, HighlightText, or ActiveBorder.
SystemColors	Defines static properties of a Color structure.
SystemIcons	Defines static properties for Windows systemwide icons.
SystemPens	Defines static properties. Each property is a Pen object with the color of a Windows display element and a width of 1.
TextureBrush	Inherited from the Brush class. This class defines a brush that has an image as its texture.
ToolboxBitmapAttribute	Defines the images associated with a specified component.

Class-class yang ada dalam System.Drawing.Drawing2D

Class	Description
AdjustableArrowCap	Represents an adjustable arrow-shaped line cap. Provides members to define the properties to fill, and to set the height and width of an arrow cap.
Blend	Gradient blends are used to provide smoothness and shading to the interiors of shapes. A blend pattern contains factor and pattern arrays, which define the position and percentage of color of the starting and ending colors. The Blend class defines a blend pattern, which uses LinearGradientBrush to fill the shapes. The Factors and Positions properties represent the array of blend factors and array of positions for the gradient, respectively.
ColorBlend	Defines color blending in multicolor gradients. The Color and Position properties represent the color array and position array, respectively.
CustomLineCap	Encapsulates a custom, user-defined line cap.
GraphicsContainer	Represents the data of a graphics container. A graphics container is created by Graphics.BeginContainer followed by a call to Graphics.EndContainer.
GraphicsPath	In GDI+, a path is a series of connected lines and curves. This class provides properties to define the path's fill mode and other properties. This class also defines methods to add graphics shapes to a path. For instance, the AddArc and AddCurve methods add an arc and a curve, respectively, to the path. Wrap, Transform, Reverse, and Reset are some of the associated methods.
GraphicsPathIterator	A path can contain subpaths. This class provides the ability to find the number of subpaths and iterate through them. Count and SubpathCount return the number of points and the number of subpaths in a path, respectively.
GraphicsState	Represents the state of a Graphics object.
HatchBrush	Hatch brushes are brushes with a hatch style, a foreground color, and a background color. This class represents a hatch brush in GDI+.

Class	Description
LinearGradientBrush	Represents a brush with a linear gradient.
Matrix	Encapsulates a 3x3 matrix that represents a geometric transformation. This class defines methods for inverting, multiplying, resetting, rotating, scaling, shearing, and translating matrices.
PathData	Contains the data in the form of points and types that makes up a path. The Points property of the class represents an array of points, and the Types property represents the types of the points in a path.
PathGradientBrush	Represents a brush with a graphics path. PathGradientBrush contains methods and properties for blending, wrapping, scaling, and transformation. This class encapsulates a Brush object that fills the interior of a GraphicsPath object with a gradient.
RegionData	Represents the data stored by a Region object. The Data property of this class represents the data in the form of an array of bytes.

Class-class yang ada dalam System.Drawing.Imaging

Class	Description
BitmapData	Often we don't want to load and refresh all data of a bitmap because rendering each pixel is not only a slow process, but also consumes system resources. With the help of the BitmapData class and its LockBits and UnlockBits methods, we can lock the required data of a bitmap in memory and work with that instead of working with all the data.
ColorMap	Defines a map for converting colors. ColorMap is used by the ImageAttributes class.
ColorMatrix	Defines a 5x5 matrix that contains coordinates for the ARGB space. ColorMatrix is used by the ImageAttributes class.
ColorPalette	Defines an array of colors that make up a color palette. ColorPalette is used by the ImageAttributes class.
Encoder	Represents an encoder, which represents a globally unique identifier (GUID) that identifies the category of an image encoder parameter. Encoder is used by the EncoderParameter class.
EncoderParameter	An encoder parameter, which sets values for a particular category of an image. This class is used in the Save method with the help of EncoderParameters.
EncoderParameters	An array of EncoderParameter objects.
FrameDimension	Provides properties to get the frame dimensions of an image.
ImageAttributes	Contains information about how image colors are manipulated during rendering (for more information, see Chapter 7).
ImageCodecInfo	Retrieves information about the installed image codecs.
ImageFormat	Specifies the format of an image.
Metafile	Defines a graphic metafile, which contains graphics operations in the form of records that can be recorded (constructed) and played back (displayed).
MetafileHeader	Stores information about a metafile.
MetaHeader	Contains information about a Windows-format (WMF) metafile.
PropertyItem	Encapsulates a metadata property to be included in an image file.
WmfPlaceableFileHeader	Defines a placeable metafile.

Area Drawing

Setiap aplikasi drawing paling tidak terdiri dari tiga komponen yaitu canvas, brush, pen, dan process

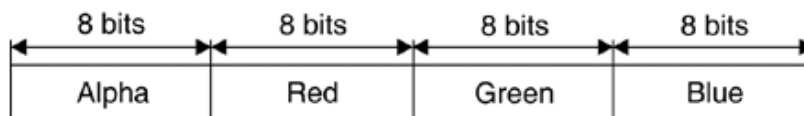
- Canvas: adalah area dimana object akan digambar, misal pada aplikasi Windows, Windows Form adalah canvasnya.
- Brush atau Pen merepresentasikan tekstur, warna, dan ukuran dari objek yang akan digambar pada canvas.
- Process mendefinisikan bagaimana object tersebut akan digambar kedalam canvas.

Setiap area drawing memiliki empat property utama yaitu: weight, height, resolution dan color depth.

- Width dan height property digunakan untuk medeskripsikan ukuran area drawing secara vertikal dan horizontal
- Resolution adalah satuan untuk mengukur output quality dari graphic object atau images dalam satuan dot per inch (dpi). Sebagai contoh resolusi 72 dpi berarti 1 inch dari area tersebut terdiri dari 72 horizontal dan 72 vertical pixel. Untuk monitor resolusi 1280x1024 berarti pada area monitor tersebut terdiri dari 1280 horizontal pixel dan 1024 vertical pixel.
- Color depth adalah jumlah warna yang digunakan untuk merepresentasikan setiap pixel.

Definisi dari pixel adalah: elemen terkecil pada proses drawing untuk menampilkan graphic object atau images pada layar. Pixel density sering direpresentasikan dengan nilai dpi (dot per inch).

Struktur color pada GDI+ mempunyai empat komponen warna yaitu: alpha, red, green, dan blue. Red green blue (RGB) komponen mewakili kombinasi warna yang akan muncul. Setiap komponen dalam RGB mempunyai 256 (28) color combination, sehingga kombinasi tiga komponen RGB tersebut mempunyai kemungkinan warna 256x256x256. Alpha komponen mewakili aspek transparan dari warna, yang akan terlihat ketika beberapa warna digabungkan.



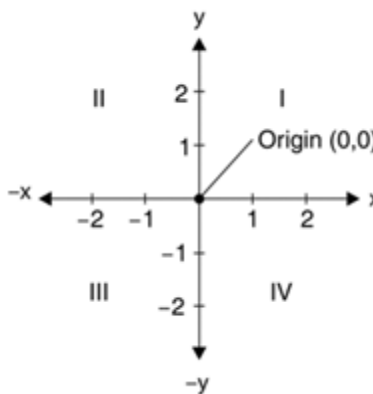
Seperti yang terlihat pada gambar diatas, area drawing paling tidak harus mensupport 24-bit color system (8-bit untuk setiap R, G, dan B komponen).

Sistem Koordinat

Memahami sistem koordinat sangat penting untuk pemrograman graphic, sistem koordinat merepresentasikan letak graphic objek pada perangkat tampilan seperti monitor dan printer.

Sistem Koordinat Kartesian

Ini adalah koordinat umum yang anda pelajari pada pelajaran matematika dasar, terdiri dari sumbu x dan y



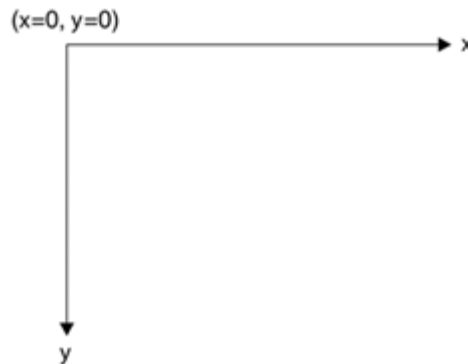
Gambar Koordinat Kartesian

Posisi sistem koordinat kartesian dibagi menjadi beberapa kuadran:

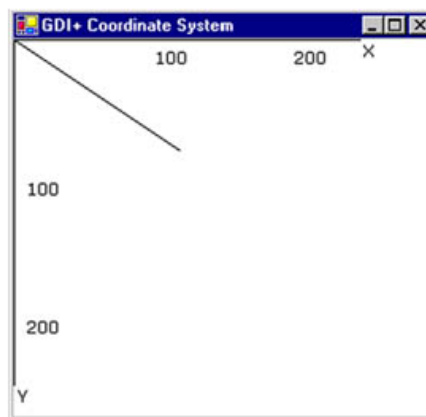
- Kuadran I: $x > 0$ dan $y > 0$
- Kuadran II: $x < 0$ dan $y > 0$
- Kuadran III: $x < 0$ dan $y < 0$
- Kuadran IV: $x > 0$ dan $y < 0$

Sistem Koordinat Standar GDI+

Tidak seperti sistem kartesian, pada perangkat tampilan seperti monitor sumbu (0,0) terletak pada pojok kiri atas.



Gambar Koordinat Display GDI+



Gambar Draw line dari titik (0,0) sampai dengan (120,80)

Aplikasi Pertama menggunakan GDI+

Sekarang saatnya anda untuk mencoba membuat aplikasi menggunakan library GDI+ menggunakan .NET Framework.

1. Pertama buat aplikasi windows form baru pada visual studio, beri nama projectnya contoh2_1
2. Untuk menggunakan GDI+ maka anda harus menambahkan referensi library kedalam program

```
using System.Drawing;
using System.Drawing.Drawing2D;
```

3. Untuk menggambar di form anda dapat menambahkan kode dalam method OnPaint.

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    //membuat instan class / objek dari class graphic
    Graphics g = e.Graphics;
}
```

4. Untuk menambahkan object Pen, Brush, serta Drawing Shape kedalam form tambahkan kode sebagai berikut

```

private void Form1_Paint(object sender, PaintEventArgs e)
{
    //membuat instan class / objek dari class graphic
    Graphics g = e.Graphics;
    //set smothing mode pada area drawing
    g.SmoothingMode = SmoothingMode.AntiAlias;
    //membuat kotak dengan width 100 dan height 100
    Rectangle rect = new Rectangle(20, 20, 100, 100);
    //membuat objek pen berwarna merah
    Pen redPen = new Pen(Color.Red, 3);
    Pen blackPen = Pens.Black;
    //membuat objek solid brush
    SolidBrush greenBrush = new SolidBrush(Color.Green);

    //menggambar shape dan line
    g.DrawRectangle(redPen, rect);
    g.FillEllipse(greenBrush, rect);
    g.DrawLine(blackPen, 0, 250, this.Width, 250);
    g.FillEllipse(Brushes.Blue, 70, 220, 30, 30);
    g.FillEllipse(Brushes.SkyBlue, 100, 210, 40, 40);
    g.FillEllipse(Brushes.Green, 140, 200, 50, 50);
    g.FillEllipse(Brushes.Yellow, 190, 190, 60, 60);
    g.FillEllipse(Brushes.Violet, 250, 180, 70, 70);
    g.FillEllipse(Brushes.Red, 320, 170, 80, 80);
}

```

5. Jalankan program dengan menekan tombol F5 untuk melihat hasilnya.

Basic GDI+ Object

Berikut ini adalah daftar property dari Color structure

Property	Description
Red, Blue, Green, Aqua, Azure, and so on	A specified color static property for almost every color.
A	Returns the alpha component value in a Color structure. We discuss alpha in color-related sections in later chapters.
R	Returns the red component value in a Color structure.
G	Returns the green component value in a Color structure.
B	Returns the blue component value in a Color structure.
IsEmpty	Indicates whether a Color structure is uninitialized.
IsKnownColor	Indicates whether a color is predefined.
IsNamedColor	Indicates whether a color is predefined.
IsSystemColor	Indicates whether a color is a system color.
Name	Returns the name of the color.

Berikut adalah daftar method dari Color structure

Method	Description
FromArgb	Creates a Color structure from the four 8-bit ARGB component (alpha-red-green-blue) values.
FromKnownColor	Creates a Color structure from the specified predefined color.
FromName	Creates a Color structure from the specified name of a predefined color.

Method	Description
GetBrightness	Returns the hue-saturation-brightness (HSB) brightness value of this Color structure.
GetHue	Returns the HSB hue value, in degrees, of this Color structure.
GetSaturation	Returns the HSB saturation value of this Color structure.
ToArgb	Returns the 32-bit ARGB value of this Color structure.
ToKnownColor	Returns the KnownColor value of this Color structure.

Menggunakan Struktur Point dan PointF

Pada GDI+ point digunakan untuk mendefinisikan koordinat titik dua dimensi yang direpresentasikan sebagai matrix (x dan y koordinat) . ada tiga macam konstruktor yang dapat digunakan untuk membuat object point yaitu:

```
Point pt1 = new Point(10);
Point pt2 = new Point( new Size(20, 20) );
Point pt3 = new Point(30, 30);
```

Struktur PointF hampir sama dengan Point hanya saja nilai sumbu x dan y-nya tidak berupa integer melainkan floating point.

```
PointF pt3 = new PointF(30.0f, 30.0f);
```

Contoh 2_2 Penggunaan Struktur Point

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

    //membuat object point
    Point pt = new Point(50, 50);
    //membuat object point kosong
    Point newPoint = Point.Empty;
    newPoint.X = 100;
    newPoint.Y = 200;

    Pen pn = new Pen(Color.Blue, 4);

    //menggambar garis dari titik pt ke newPoint
    g.DrawLine(pn, pt, newPoint);

    pn.Dispose();
    g.Dispose();
}
```

Menggunakan Struktur Rectangle

Struktur rectangle digunakan untuk membuat object rectangle di GDI+

Contoh2_3 Penggunaan Struktur Rectangle

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    // Create a Graphics object
    Graphics g = this.CreateGraphics();
    int x = 40;
    int y = 40;
    int height = 120;
```

```

int width = 120;
// Create a Point object
Point pt = new Point(80, 80);
// Create a Size object
Size sz = new Size(100, 100);
// Create a rectangle from Point and Size
Rectangle rect1 = new Rectangle(pt, sz);
// Create a rectangle from integers
Rectangle rect2 =
    new Rectangle(x, y, width, height);
// Create a rectangle from direct integers
Rectangle rect3 =
    new Rectangle(10, 10, 180, 180);
// Create pens and brushes
Pen redPen = new Pen(Color.Red, 2);
SolidBrush greenBrush =
    new SolidBrush(Color.Blue);
SolidBrush blueBrush =
    new SolidBrush(Color.Green);
// Draw and fill rectangles
g.DrawRectangle(redPen, rect3);
g.FillRectangle(blueBrush, rect2);
g.FillRectangle(greenBrush, rect1);
// Dispose of the objects
redPen.Dispose();
blueBrush.Dispose();
greenBrush.Dispose();
g.Dispose();
}

```

Method	Description
Ceiling	Converts a RectangleF object to a Rectangle object by rounding the RectangleF values to the next higher integer values.
Contains	Determines if the specified point is contained within the rectangular region of a rectangle.
FromLTRB	Creates a rectangle with the specified edge locations.
Inflate	Creates and returns an inflated copy of a rectangle.
Intersect	Replaces a rectangle with the intersection of itself and another rectangle.
IntersectsWith	Determines if a specified rectangle intersects with rect.
Offset	Adjusts the location of a specified rectangle by the specified amount.
Round	Converts a RectangleF object to a Rectangle object by rounding the RectangleF values to the nearest integer values.
Truncate	Converts a RectangleF object to a Rectangle object by truncating the RectangleF values.
Union	Returns a rectangle that contains the union of two Rectangle structures.

Graphic Class Method

Draw Method

Digunakan untuk menggambar garis, rectangle, dan ellipse

Drawing Lines

Method DrawLine digunakan untuk menggambar garis yang menghubungkan dua titik.

Method	Description
DrawArc	Draws an arc (a portion of an ellipse specified by a pair of coordinates, a width, a height, and start and end angles).
DrawBezier	Draws a Bézier curve defined by four Point structures.
DrawBeziers	Draws a series of Bézier splines from an array of Point structures.
DrawClosedCurve	Draws a closed cardinal spline defined by an array of Point structures.
DrawCurve	Draws a cardinal spline through a specified array of Point structures.
DrawEllipse	Draws an ellipse defined by a bounding rectangle specified by a pair of coordinates, a height, and a width.
DrawIcon	Draws an image represented by the specified Icon object at the specified coordinates.
DrawIconUnstretched	Draws an image represented by the specified Icon object without scaling the image.
DrawImage	Draws the specified Image object at the specified location and with the original size.
DrawImageUnscaled	Draws the specified Image object with its original size at the location specified by a coordinate pair.
DrawLine	Draws a line connecting two points specified by coordinate pairs.
DrawLines	Draws a series of line segments that connect an array of Point structures.
DrawPath	Draws a GraphicsPath object.
DrawPie	Draws a pie shape specified by a coordinate pair, a width, a height, and two radial lines.
DrawPolygon	Draws a polygon defined by an array of Point structures.
DrawRectangle	Draws a rectangle specified by a coordinate pair, a width, and a height.
DrawRectangles	Draws a series of rectangles specified by an array of Rectangle structures.
DrawString	Draws the specified text string at the specified location using the specified Brush and Font objects.

Contoh2_4 Menggambar Garis

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen redPen = new Pen(Color.Red, 1);
```

Oleh : Erick Kurniawan, S.Kom, M.Kom (<http://actualtraining.wordpress.com>)

```

    Pen bluePen = new Pen(Color.Blue, 2);
    Pen greenPen = new Pen(Color.Green, 3);
    Pen blackPen = new Pen(Color.Black, 4);
    // Draw line using float coordinates
    float x1 = 20.0F, y1 = 20.0F;
    float x2 = 200.0F, y2 = 20.0F;
    g.DrawLine(redPen, x1, y1, x2, y2);
    // Draw line using Point structure
    Point pt1 = new Point(20, 20);
    Point pt2 = new Point(20, 200);
    g.DrawLine(greenPen, pt1, pt2);
    // Draw line using PointF structure
    PointF ptf1 = new PointF(20.0F, 20.0F);
    PointF ptf2 = new PointF(200.0F, 200.0F);
    g.DrawLine(bluePen, ptf1, ptf2);
    // Draw line using integer coordinates
    int X1 = 60, Y1 = 40, X2 = 250, Y2 = 100;
    g.DrawLine(blackPen, X1, Y1, X2, Y2);
    // Dispose of objects
    redPen.Dispose();
    bluePen.Dispose();
    greenPen.Dispose();
    blackPen.Dispose();
}

```

Contoh2_5 Menggambar Garis yang Terhubung

```

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen redPen = new Pen(Color.Red, 1);
    PointF[] ptsArray = {
        new PointF( 20.0F, 20.0F),
        new PointF( 20.0F, 200.0F),
        new PointF(200.0F, 200.0F),
        new PointF(20.0F, 20.0F)};
    g.DrawLines(redPen, ptsArray);
}

```

Drawing Rectangle

Bentuk dasar selanjutnya adalah rectangle. Jika anda ingin menggambar rectangle maka yang harus diperhatikan adalah titik awal / starting point, width, dan height-nya.

Contoh2_6 Menggambar Rectangle

```

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Create pens and points
    Pen redPen = new Pen(Color.Red, 1);
    Pen bluePen = new Pen(Color.Blue, 2);
    Pen greenPen = new Pen(Color.Green, 3);
    float x = 5.0F, y = 5.0F;
    float width = 100.0F;
    float height = 200.0F;
    // Create a rectangle
    Rectangle rect = new Rectangle(20, 20, 80, 40);
}

```

```

    // Draw rectangles
    g.DrawRectangle(bluePen,
        x, y, width, height);
    g.DrawRectangle(redPen,
        60, 80, 140, 50);
    g.DrawRectangle(greenPen, rect);
    // Dispose of objects
    redPen.Dispose();
    bluePen.Dispose();
    greenPen.Dispose();
}

```

Contoh 2_7 Menggambar Rectangle Berurutan / Series Mode

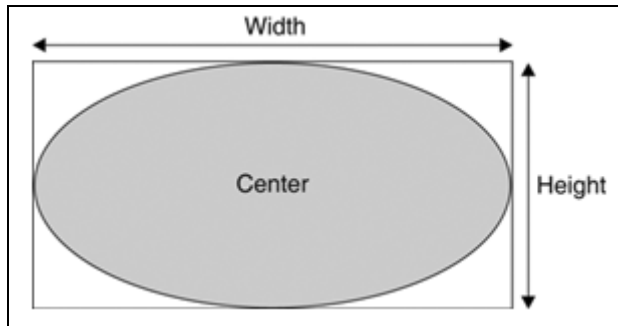
```

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen greenPen = new Pen(Color.Green, 4);
    RectangleF[] rectArray =
    {
        new RectangleF( 5.0F, 5.0F, 100.0F, 200.0F),
        new RectangleF(20.0F, 20.0F, 80.0F, 40.0F),
        new RectangleF(60.0F, 80.0F, 140.0F, 50.0F)
    };
    g.DrawRectangles(greenPen, rectArray);
    greenPen.Dispose();
}

```

Drawing Ellipses dan Circle

Bentuk ellipses adalah bentuk circular didalam bentuk rectangle. Bentuk ellipses mempunyai titik tengah / center point.



Contoh2_8 Menggambar Ellipses

```

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Create pens
    Pen redPen = new Pen(Color.Red, 6);
    Pen bluePen = new Pen(Color.Blue, 4);
    Pen greenPen = new Pen(Color.Green, 2);
    // Create a rectangle
    Rectangle rect =
        new Rectangle(80, 80, 50, 50);
    // Draw ellipses
}

```

```

        g.DrawEllipse(greenPen,
            100.0F, 100.0F, 10.0F, 10.0F);
        g.DrawEllipse(redPen, rect);
        g.DrawEllipse(bluePen, 60, 60, 90, 90);
        g.DrawEllipse(greenPen,
            40.0F, 40.0F, 130.0F, 130.0F);
        // Dispose of objects
        redPen.Dispose();
        greenPen.Dispose();
        bluePen.Dispose();
    }

```

Drawing Text

Digunakan untuk menggambar text string dalam graphic surfaces

Contoh2_9 Menggambar String

```

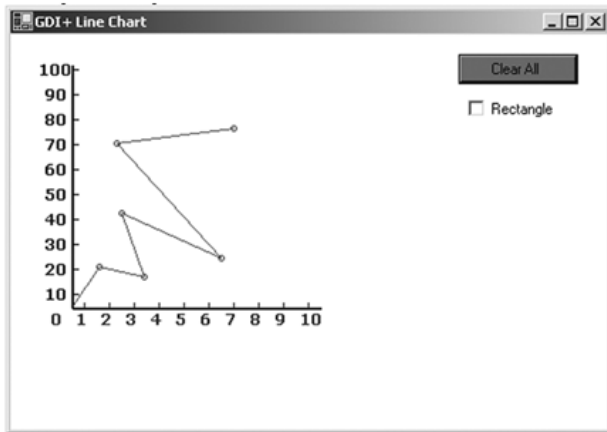
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Create brushes
    SolidBrush blueBrush = new SolidBrush(Color.Blue);
    SolidBrush redBrush = new SolidBrush(Color.Red);
    SolidBrush greenBrush = new SolidBrush(Color.Green);
    // Create a rectangle
    Rectangle rect = new Rectangle(20, 20, 200, 100);
    // The text to be drawn
    String drawString = "Hello GDI+ World!";
    // Create a Font object
    Font drawFont = new Font("Verdana", 14);
    float x = 100.0F;
    float y = 100.0F;
    // String format
    StringFormat drawFormat = new StringFormat();
    // Set string format flag to direction vertical,
    // which draws text vertically
    drawFormat.FormatFlags =
        StringFormatFlags.DirectionVertical;
    // Draw string

    g.DrawString("Drawing text",
        new Font("Tahoma", 14), greenBrush, rect);
    g.DrawString(drawString,
        new Font("Arial", 12), redBrush, 120, 140);
    g.DrawString(drawString, drawFont,
        blueBrush, x, y, drawFormat);
    // Dispose of objects
    blueBrush.Dispose();
    redBrush.Dispose();
    greenBrush.Dispose();
    drawFont.Dispose();
}

```

Membuat Line Chart Application

Setelah anda mempelajari bentuk-bentuk drawing yang bisa dibuat, maka anda akan diajak untuk membuat sebuah aplikasi yang ada pada “real world” yaitu membuat chart. Chart sangat penting untuk ilmu statistic.



Contoh2_10 Membuat Line Chart

Deklarasikan dua variabel untuk start dan end point

```
private Point startPoint = new Point(50, 217);
private Point endPoint = new Point(50, 217);
```

Untuk membuat sumbu koordinat anda harus menggambar line dan string secara manual

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Font vertFont =
        new Font("Verdana", 10, FontStyle.Bold);
    Font horzFont =
        new Font("Verdana", 10, FontStyle.Bold);
    SolidBrush vertBrush = new SolidBrush(Color.Black);
    SolidBrush horzBrush = new SolidBrush(Color.Blue);
    Pen blackPen = new Pen(Color.Black, 2);
    Pen bluePen = new Pen(Color.Blue, 2);
    // Drawing a vertical and a horizontal line
    g.DrawLine(blackPen, 50, 220, 50, 25);
    g.DrawLine(bluePen, 50, 220, 250, 220);
    // x-axis drawing
    g.DrawString("0", horzFont, horzBrush, 30, 220);
    g.DrawString("1", horzFont, horzBrush, 50, 220);
    g.DrawString("2", horzFont, horzBrush, 70, 220);
    g.DrawString("3", horzFont, horzBrush, 90, 220);
    g.DrawString("4", horzFont, horzBrush, 110, 220);
    g.DrawString("5", horzFont, horzBrush, 130, 220);
    g.DrawString("6", horzFont, horzBrush, 150, 220);
    g.DrawString("7", horzFont, horzBrush, 170, 220);
    g.DrawString("8", horzFont, horzBrush, 190, 220);
    g.DrawString("9", horzFont, horzBrush, 210, 220);
    g.DrawString("10", horzFont, horzBrush, 230, 220);
    // Drawing vertical strings
    StringFormat vertStrFormat = new StringFormat();
    vertStrFormat.FormatFlags =
        StringFormatFlags.DirectionVertical;

    g.DrawString("-", horzFont, horzBrush,
        50, 212, vertStrFormat);
    g.DrawString("-", horzFont, horzBrush,
        70, 212, vertStrFormat);
```

```

g.DrawString("-", horzFont, horzBrush,
  90, 212, vertStrFormat);
g.DrawString("-", horzFont, horzBrush,
  110, 212, vertStrFormat);
g.DrawString("-", horzFont, horzBrush,
  130, 212, vertStrFormat);
g.DrawString("-", horzFont, horzBrush,
  150, 212, vertStrFormat);
g.DrawString("-", horzFont, horzBrush,
  170, 212, vertStrFormat);
g.DrawString("-", horzFont, horzBrush,
  190, 212, vertStrFormat);
g.DrawString("-", horzFont, horzBrush,
  210, 212, vertStrFormat);
g.DrawString("-", horzFont, horzBrush,
  230, 212, vertStrFormat);
// y-axis drawing
g.DrawString("100-", vertFont, vertBrush, 20, 20);
g.DrawString("90 -", vertFont, vertBrush, 25, 40);
g.DrawString("80 -", vertFont, vertBrush, 25, 60);
g.DrawString("70 -", vertFont, vertBrush, 25, 80);
g.DrawString("60 -", vertFont, vertBrush, 25, 100);
g.DrawString("50 -", vertFont, vertBrush, 25, 120);
g.DrawString("40 -", vertFont, vertBrush, 25, 140);
g.DrawString("30 -", vertFont, vertBrush, 25, 160);
g.DrawString("20 -", vertFont, vertBrush, 25, 180);
g.DrawString("10 -", vertFont, vertBrush, 25, 200);
// Dispose of objects
vertFont.Dispose();
horzFont.Dispose();
vertBrush.Dispose();
horzBrush.Dispose();
blackPen.Dispose();
bluePen.Dispose();
}

```

Pada saat event mouse down akan digambar line yang berawal dari titik (0,0)

```

if (e.Button == MouseButton.Left)
{
    // Create a Graphics object
    Graphics g1 = this.CreateGraphics();
    // Create two pens
    Pen linePen = new Pen(Color.Green, 1);
    Pen ellipsePen = new Pen(Color.Red, 1);
    startPoint = endPoint;
    endPoint = new Point(e.X, e.Y);
    // Draw the line from the current point
    // to the new point
    g1.DrawLine(linePen, startPoint, endPoint);
    // If Rectangle check box is checked,
    // draw a rectangle to represent the point
    if (checkBox1.Checked)
    {
        g1.DrawRectangle(ellipsePen,
            e.X - 2, e.Y - 2, 4, 4);
    }
    // Draw a circle to represent the point
    else
    {

```

```

        gl.DrawEllipse(ellipsePen,
            e.X - 2, e.Y - 2, 4, 4);
    }
    // Dispose of objects
    linePen.Dispose();
    ellipsePen.Dispose();
    gl.Dispose();
}

```

Pada tombol “Clear All” masukan kode sebagai berikut, untuk menghapus gambar line chart.

```

private void btnAll_Click(object sender, EventArgs e)
{
    startPoint.X = 50;
    startPoint.Y = 217;
    endPoint.X = 50;
    endPoint.Y = 217;
    this.Invalidate(this.ClientRectangle);
}

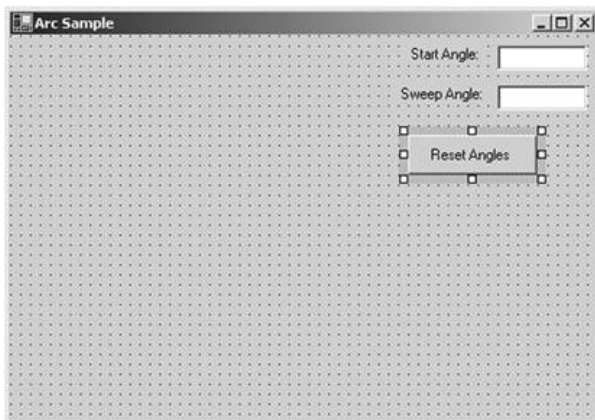
```

Drawing Arc

Arc adalah bagian dari ellipse, yang dibentuk dari area rectangle, start angle, dan sweep angle. Cara penggunaan Arc adalah sebagai berikut:

Contoh2_11 Menggambar Arc

Buat design form sebagai berikut:



Buat variabel awal untuk deklarasi awal start dan sweep angle

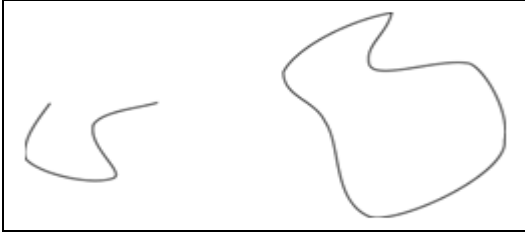
```

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    Pen redPen = new Pen(Color.Red, 3);
    Rectangle rect = new Rectangle(20, 20, 200, 200);
    g.DrawArc(redPen, rect, startAngle, sweepAngle);
    redPen.Dispose();
}
private void btnReset_Click(object sender, EventArgs e)
{
    startAngle = (float)Convert.ToDouble(txtStart.Text);
    sweepAngle = (float)Convert.ToDouble(txtSweep.Text);
    Invalidate();
}

```

Drawing Curves

Curve terdiri dari dua macam yaitu open dan close curve

**Contoh2_12 Drawing Curves**

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Create a pen
    Pen bluePen = new Pen(Color.Blue, 1);
    // Create an array of points
    PointF pt1 = new PointF(40.0F, 50.0F);
    PointF pt2 = new PointF(50.0F, 75.0F);
    PointF pt3 = new PointF(100.0F, 115.0F);
    PointF pt4 = new PointF(200.0F, 180.0F);
    PointF pt5 = new PointF(200.0F, 90.0F);
    PointF[] ptsArray =
    {
        pt1, pt2, pt3, pt4, pt5
    };
    // Draw curve
    g.DrawCurve(bluePen, ptsArray);
    // Dispose of object
    bluePen.Dispose();
}
```

Contoh2_13 Drawing Curves dengan Tension

```
private float tension = 0.5F;

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Create a pen
    Pen bluePen = new Pen(Color.Blue, 1);
    // Create an array of points
    PointF pt1 = new PointF(40.0F, 50.0F);
    PointF pt2 = new PointF(50.0F, 75.0F);
    PointF pt3 = new PointF(100.0F, 115.0F);
    PointF pt4 = new PointF(200.0F, 180.0F);
    PointF pt5 = new PointF(200.0F, 90.0F);
    PointF[] ptsArray =
    {
        pt1, pt2, pt3, pt4, pt5
    };
    // Draw curve
    g.DrawCurve(bluePen, ptsArray, tension);
    // Dispose of object
    bluePen.Dispose();
}
```

```
private void btnApply_Click(object sender, EventArgs e)
{
    tension = (float)Convert.ToDouble(txtTension.Text);
    Invalidate();
}
```

Contoh2_14 Drawing Closed Curve

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Create a pen
    Pen bluePen = new Pen(Color.Blue, 1);
    // Create an array of points
    PointF pt1 = new PointF(40.0F, 50.0F);
    PointF pt2 = new PointF(50.0F, 75.0F);
    PointF pt3 = new PointF(100.0F, 115.0F);
    PointF pt4 = new PointF(200.0F, 180.0F);
    PointF pt5 = new PointF(200.0F, 90.0F);
    PointF[] ptsArray =
    {
        pt1, pt2, pt3, pt4, pt5
    };
    // Draw curve
    g.DrawClosedCurve(bluePen, ptsArray);
    // Dispose of object
    bluePen.Dispose();
}
```

Drawing Polygon

Polygon adalah bentuk yang terdiri dari tiga atau lebih garis, misal: segitiga, dan kotak.

Contoh2_15 Drawing Polygon

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Create pens
    Pen greenPen = new Pen(Color.Green, 2);
    Pen redPen = new Pen(Color.Red, 2);
    // Create points for polygon
    PointF p1 = new PointF(40.0F, 50.0F);
    PointF p2 = new PointF(60.0F, 70.0F);
    PointF p3 = new PointF(80.0F, 34.0F);
    PointF p4 = new PointF(120.0F, 180.0F);
    PointF p5 = new PointF(200.0F, 150.0F);
    PointF[] ptsArray =
    {
        p1, p2, p3, p4, p5
    };
    // Draw polygon
    g.DrawPolygon(greenPen, ptsArray);
    // Dispose of objects
    greenPen.Dispose();
    redPen.Dispose();
}
```

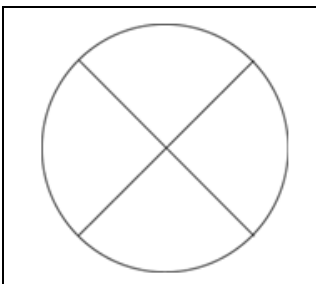
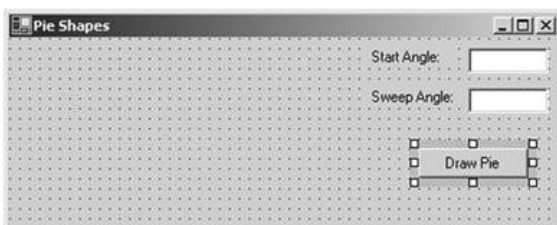
Drawing Graphic Path

Graphic Path menghubungkan beberapa objek drawing seperti line, rectable, circle, dll.

Contoh2_16 Drawing Graphic Path

```
using System.Drawing;
using System.Drawing.Drawing2D;

private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = e.Graphics;
    // Create a pen
    Pen greenPen = new Pen(Color.Green, 1);
    // Create a graphics path
    GraphicsPath path = new GraphicsPath();
    // Add a line to the path
    path.AddLine(20, 20, 103, 80);
    // Add an ellipse to the path
    path.AddEllipse(100, 50, 100, 100);
    // Add three more lines
    path.AddLine(195, 80, 300, 80);
    path.AddLine(200, 100, 300, 100);
    path.AddLine(195, 120, 300, 120);
    // Create a rectangle and call
    // AddRectangle
    Rectangle rect =
        new Rectangle(50, 150, 300, 50);
    path.AddRectangle(rect);
    // Draw path
    g.DrawPath(greenPen, path);
    // Dispose of object
    greenPen.Dispose();
}
```

Drawing Pie Shapes**Contoh2_17 Drawing Pie Chart**

```
private void btnDraw_Click(object sender, EventArgs e)
{
    // Create a Graphics object
    Graphics g = this.CreateGraphics();
    g.Clear(this.BackColor);
    // Get the current value of start and sweep
    // angles
    float startAngle =
        (float)Convert.ToDouble(textBox1.Text);
    float sweepAngle =
        (float)Convert.ToDouble(textBox2.Text);
    // Create a pen
    Pen bluePen = new Pen(Color.Blue, 1);
    // Draw pie
    g.DrawPie(bluePen, 20, 20, 100, 100,
        startAngle, sweepAngle);
    // Dispose of objects
    bluePen.Dispose();
    g.Dispose();
}
```