



TEMA4.PROGRAMACIÓN ORIENTADA A OBJETOS

INTRODUCCIÓN

La plataforma .NET Framework dispone de una librería especial para el tratamiento de gráficos GDI+.

Con GDI puede manipular cualquier tipo de imagen y trabajar con formas en las aplicaciones de C#-

Para trabajar de manera adecuada deberá emplear cuatro objetos principales Graphics, Pen, Brush y Color.

Acceso a la librería GDI

Para poder tener acceso a sus métodos y propiedades hacemos uso de la directiva: System.Drawing.

using System.Drawing;

Cómo trabajar con gráficos

Al trabajar con GDI+ en .NET, el principal objeto con el que se va a trabajar es el objeto Graphics. Este

Objeto nos va a permitir realizar el trazado de formas, trabajar con imágenes o mostrar texto. Para poder utilizarlo debemos crear primero un contenedor del gráfico a través de un Objeto de la Clase Image.



Contenedor del gráfico (Clase Image), área donde vamos a poder dibujar.

Clase Image

Sintaxis

```
Image area_imagen;
```

```
area_imagen = new Bitmap(contenedor_imagen.Width, contenedor_imagen.Height);
```

o bien

```
area_imagen = new Bitmap("ruta de la imagen");
```

*El contenedor de imagen es generalmente un PictureBox

Declaración del objeto de la clase Graphics

```
Graphics objeto_grafico;  
objeto_grafico=Graphics.FromImage(area_imagen)
```

Con este código usted está creando un objeto Graphics a partir de una archivo de imagen de mapa de bits del sistema de archivos.

Con la clase **Pen** usted puede trazar líneas y curvas sobre la superficie de gráficos. Para inicializar un objeto de la clase Pen utilizamos la siguiente sintaxis:



```
Pen pluma;  
pluma = new Pen(Color.Color_a_utilizar);
```

El color a utilizar es el nombre del color en inglés, por ejemplo: red, green, blue, yellow, etc.

Finalmente para mostrar el gráfico resultante en el formulario lo hacemos a través del pictureBox, que nos sirve como contenedor de imágenes.

```
PictureBox.Image = area_imagen;
```

Principales figuras que se pueden trazar por medio de un objeto de la clase Graphics:

Punto (X, Y) ●

El objeto punto nos va a permitir almacenar las coordenadas de un punto en el área de dibujo, para declarar un punto utilizamos un objeto de la clase Point:

```
Point P=new Point(X,Y); //X y Y pueden ser enteros o flotantes
```

Línea

(X1, Y1) ————— (X2, Y2)


```
objeto_grafico.DrawLine(pluma,X1,Y1,X2,Y2)
```

También podemos hacer uso de la clase Point para almacenar los datos de un punto del gráfico:

```
Point P1,P2;  
P1 = new Point(X1, Y1);  
P2 = new Point(X2, Y2);  
objeto_grafico.DrawLine(pluma,P1,P2);
```

Rectángulo

(X1, Y1) Width (entero o flotante)



Height (entero o flotante)

```
Objeto_grafico.DrawRectangle(pluma,X1,Y1,Widht,Height)
```

Ejemplo:

```
int X1=10, Y1=10, Wi=30, He=60;  
objeto_grafico.DrawRectangle(pluma,X1,Y1,Wi,He);
```

También podemos hacer uso de un objeto de la clase Rectangle para generar la figura:

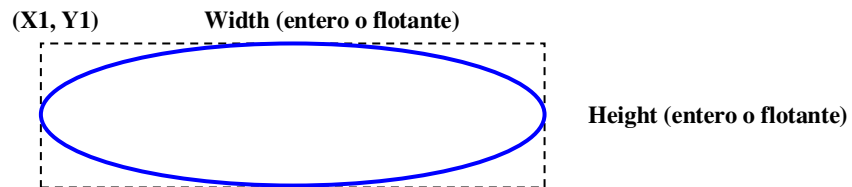
```
Rectangle Rectangulo;
```



TEMA4.PROGRAMACIÓN ORIENTADA A OBJETOS

```
Rectangulo=new Rectangle(50,50,80,10);  
//Constructor Rectangle(X1,Y1,Width,Height)  
mi_grafico.DrawRectangle(pluma,Rectangulo);
```

Elipse



```
Objeto_grafico.DrawEllipse(pluma,X1,Y1,Widht,Height)
```

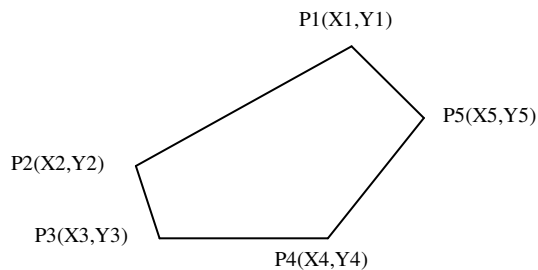
Ejemplo:

```
int X1=10, Y1=10, Wi=30, He=60;  
objeto_grafico. DrawEllipse(pluma,X1,Y1,Wi,He);
```

También podemos hacer uso de un objeto de la clase Rectangle para generar la figura:

```
Rectangle Rectangulo;  
Rectangulo=new Rectangle(50,50,80,10);  
//Constructor Rectangle(X1,Y1,Width,Height)  
mi_grafico.DrawEllipse(pluma,Rectangulo);
```

Polígono



```
objeto_grafico.DrawPolygon(pluma, Puntos[]);
```

Ejemplo:

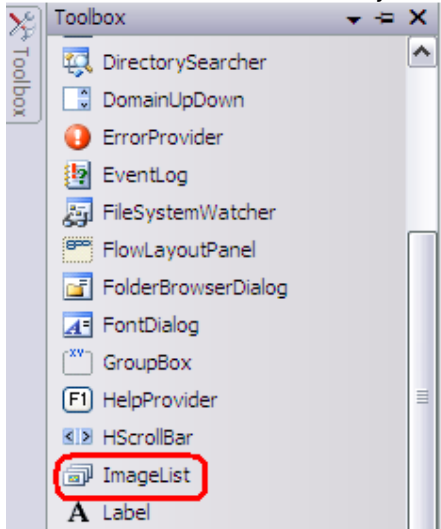
```
Point[] Puntos;  
Puntos=new Point[4];  
Puntos[0]=new Point(15,120);  
Puntos[1]=new Point(125,140);  
Puntos[2]=new Point(110,130);  
Puntos[3]=new Point(160,180);  
mi_grafico.DrawPolygon(pluma, Puntos);
```



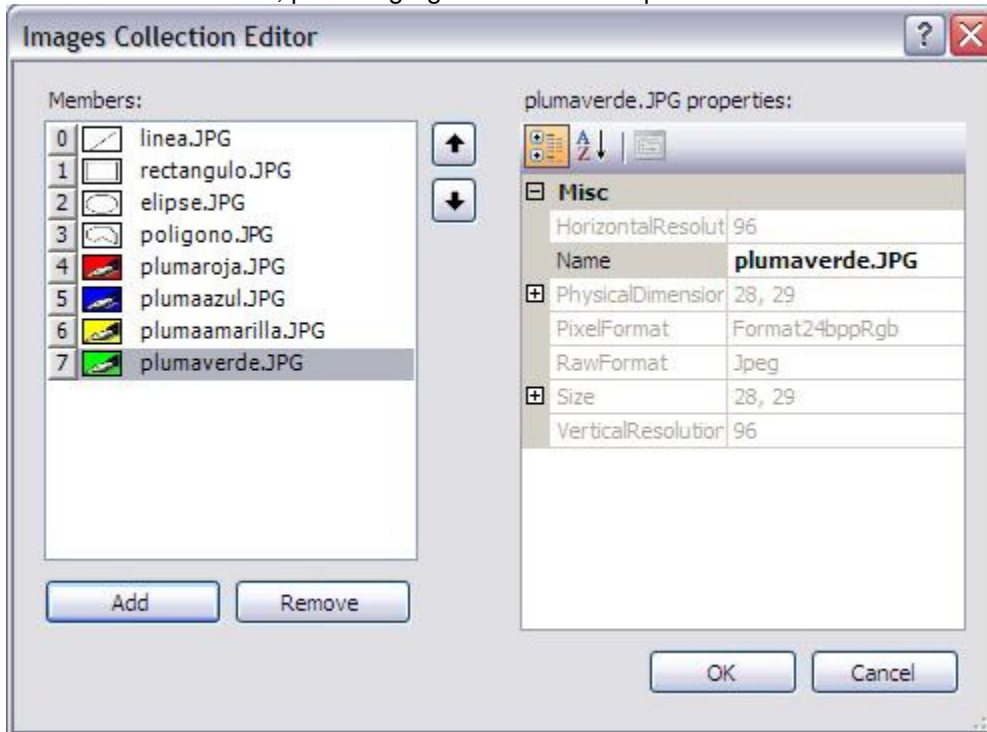
EJERCICIOS DE LA PRÁCTICA

La siguiente práctica tiene como objetivo utilizar las clases contenidas en la librería GDI+ para dibujar figuras geométricas.

- 1.Crea una nueva solución y llámala **Practica23_paint** y llama a tu proyecto **paint**. Cambia el nombre de tu formulario a frmPaint
2. Inserta en el formulario un objeto de la clase imageList (**toolBox > All Windows forms**)



3. Selecciona la propiedad **Images** y dale un clic en **(Collection)**. En la ventana que se despliega da clic en el botón **Add**, para ir agregando los iconos que utilizarás en tu formulario.





TEMA4.PROGRAMACIÓN ORIENTADA A OBJETOS

4. Posteriormente adiciona un toolStrip e inserta 4 botones, un separador y cuatro botones más.

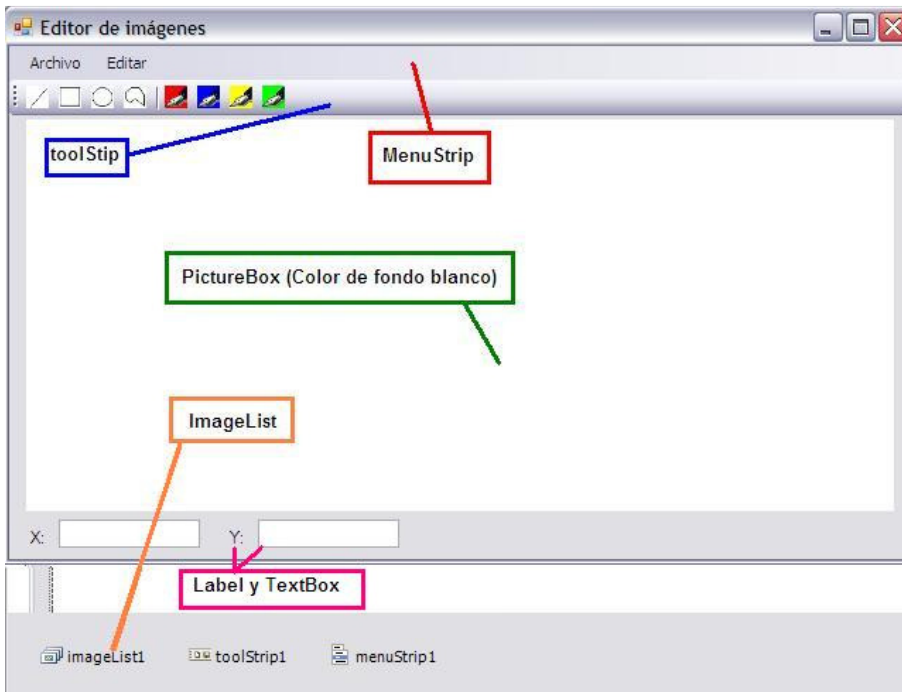


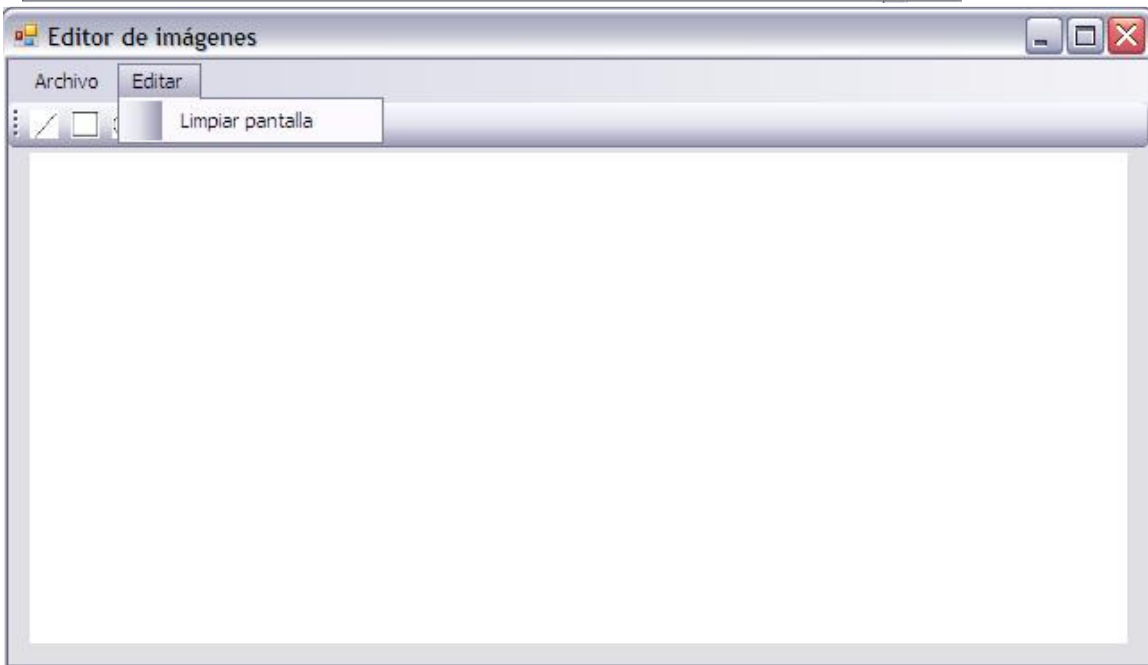
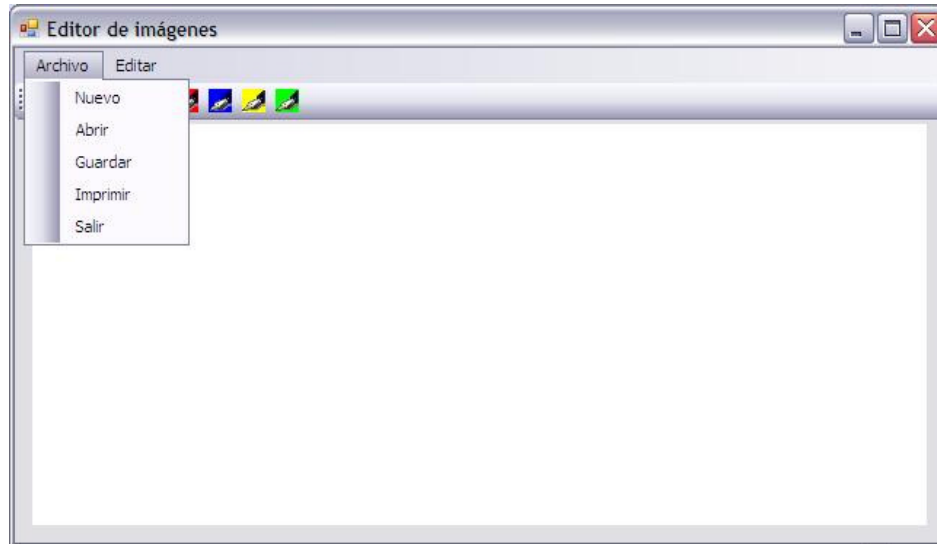
↑
Separador

5. Dale doble clic sobre cualquier parte del formulario, se generará el **evento Load**, y en el evento adiciona el siguiente código:

```
// asignar lista de imágenes a barra de herramientas  
this.toolStrip1.ImageList = this.imageList1;  
  
// asignar imágenes de lista a controles  
// de la barra de herramientas  
this.toolStripButton1.ImageIndex = 0;  
this.toolStripButton2.ImageIndex = 1;  
this.toolStripButton3.ImageIndex = 2;  
this.toolStripButton4.ImageIndex = 3;  
this.toolStripButton5.ImageIndex = 4;  
this.toolStripButton6.ImageIndex = 5;  
this.toolStripButton7.ImageIndex = 6;  
this.toolStripButton8.ImageIndex = 7;
```

6. Finalmente diseña el formulario para que quede de la siguiente manera:







Edición del formulario frmPaint

7. Edita el formulario frmPaint.Designer.cs, en las propiedades del control PictureBox y pon el siguiente

código:

```
this.pictureBox1.MouseMove += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseMove);
this.pictureBox1.MouseDown += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseDown);
this.pictureBox1.MouseUp += new
System.Windows.Forms.MouseEventHandler(this.pictureBox1_MouseUp);
```

8. Edita el formulario frmPaint.cs y escribe el siguiente código:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Text;
using System.Windows.Forms;
using System.Drawing;
using System.Drawing.Imaging;

namespace Practica22_mi_paint
{
    public partial class Form2 : Form
    {
        static Pen pluma;
        static Graphics dibujo;
        static Image imagen;
        static Point P1;
        static Point P2;
        private int figura = 0;
        private bool dibuja = false;
        private int X1, Y1, X2, Y2;

        public Form2()
        {
            InitializeComponent();
            //Creo mi imagen como Bitmap
            imagen = new Bitmap(pictureBox1.Width, pictureBox1.Height);
            //La imagen se va a guardar en un pictureBox
            pictureBox1.Image = imagen;
            dibujo = Graphics.FromImage(imagen);
            pluma = new Pen(Color.Blue);
        }
        private void pictureBox1_MouseMove(object sender, MouseEventArgs e)
        {
            textBox1.Text = Convert.ToString(e.X);
            textBox2.Text = Convert.ToString(e.Y);
        }
    }
}
```



```
private void pictureBox1_MouseDown(object sender, MouseEventArgs e)
{
    dibuja = true;
    X1 = e.X;
    Y1 = e.Y;
}

private void pictureBox1_MouseUp(object sender, MouseEventArgs e)
{
    if (dibuja == true)
    {
        X2 = e.X;
        Y2 = e.Y;
        P1 = new Point(X1, Y1);
        P2 = new Point(X2, Y2);
        switch (figura)
        {
            case 1://Linea
                dibujo.DrawLine(pluma, P1, P2);
                pictureBox1.Image = imagen;
                break;
            default:
                break;
        }
        dibuja = false;
    }
}

private void toolStripButton1_Click(object sender, EventArgs e)
{
    figura = 1;//Linea
}
private void toolStripButton2_Click(object sender, EventArgs e)
{
    figura = 2;//Rectangulo
}
private void toolStripButton3_Click(object sender, EventArgs e)
{
    figura = 3;//Ovalo
}

private void toolStripButton4_Click(object sender, EventArgs e)
{
    figura = 4;//Poligono
}

private void toolStripButton5_Click(object sender, EventArgs e)
{
    pluma = new Pen(Color.Red); //Pluma color rojo
}
```



TEMA4.PROGRAMACIÓN ORIENTADA A OBJETOS

```
private void toolStripButton6_Click(object sender, EventArgs e)
{
    pluma = new Pen(Color.Blue); //Pluma color azul
}

private void toolStripButton7_Click(object sender, EventArgs e)
{
    pluma = new Pen(Color.Yellow); //Pluma color amarillo
}

private void toolStripButton8_Click(object sender, EventArgs e)
{
    pluma = new Pen(Color.Green); //Pluma color verde
}

private void salirToolStripMenuItem_Click(object sender, EventArgs e)
{
    this.Close(); //Cerrar aplicacion
}

private void nuevoToolStripMenuItem_Click(object sender, EventArgs e)
{
    dibujo.Clear(Color.White); //Limpiar el picture box coloreandolo a blanco
    pictureBox1.Image = imagen;
}

} //Fin de clase
} //Fin de nombre de espacio
```

9. Adiciona los comentarios necesarios y compila.

10. Completa el código para que dibuje el rectángulo, ovalo y polígono.

11. Edita el formulario para que guarde la imagen en un archivo. Para guardar la imagen utiliza tu objeto Imagen en su método Save, como se muestra a continuación:

```
imagen.Save("miDibujo.png", ImageFormat.Png);
```

12. Edita el botón abrir para muestre en el pictureBox una imagen guardada.

Para mostrar una imagen en el pictureBox debes utilizar el método Load. Ejemplo:

```
pictureBox1.Load("miDibujo.png");
```

(Se recomienda usar el openFileDialog y SaveFileDialog para hacer más flexible la aplicación).