

Facultad:	Ingeniería
Escuela:	Computación
Asignatura:	Programación con Estructuras de Datos

Tema: Librería gráfica en C#.

Competencia

- Desarrolla sistemas de información informáticos mediante la integración de principios matemáticos, ciencia computacional y prácticas de ingeniería, considerando estándares de calidad y mejores prácticas validadas por la industria del software.

Materiales y Equipo

- Guía Número 2
- Computadora con programa Microsoft Visual C#.

Introducción Teórica

A menudo es necesario personalizar el aspecto de nuestras aplicaciones, para ello .NET Framework nos suministra el espacio de nombres **System.Drawing** en el que podemos encontrar clases con las funcionalidades necesarias para crear y manipular aspectos gráficos, imágenes, efectos personalizados y controles personalizados.

Características comunes de las clases de Drawing:

- ☐ Clases que representan elementos habituales de dibujo.
- ☐ Estructuras que se usan para dibujar líneas, figuras e imágenes.
- ☐ Enumeraciones que representan los colores y estilos habituales

.NET Framework suministra el espacio de nombre System.Drawing para habilitar la creación y manipulación de objetos gráficos, con clases, estructuras y enumeraciones que se usan para desarrollar una interfaz bidimensional (2D)

Los tipos de este espacio de nombres implementan un wrapper (envoltorio) agrupando los objetos de la librería del interfaz gráfico de dispositivo no manejado (GDI +)

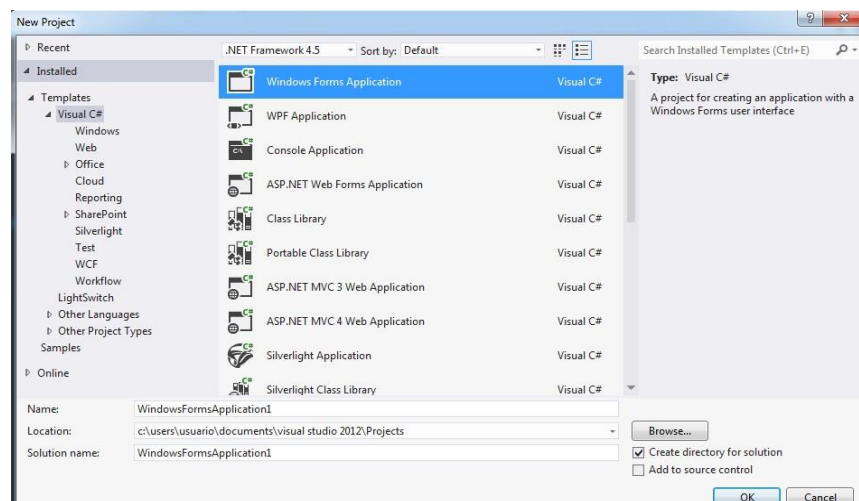
Nombre	Tipo	Descripción
Bitmap	Clase	Para trabajar con imágenes basadas en píxeles
Brush	Clase	Para rellenar el interior de los elementos gráficos
Font	Clase	Formato que se aplica a los textos (tamaño, tipo y color)
FontFamily	Clase	Grupo de fuentes con aspecto o estilo similar.
Graphics	Clase	La superficie en la que dibujamos
Icon	Clase	Un icono que usaremos en la aplicación.
Pen	Clase	Para dibujar líneas y curvas
SolidBrush	Clase	Un pincel de color único para rellenar figuras
TextureBrush	Clase	Un pincel que rellena las figuras con una imagen.
Color	Estructura	Un color Alpha (RedGreenBlue)
Point	Estructura	Dos enteros que definen un punto
PointF	Estructura	Dos valores de coma flotante para definir un punto
Rectangle	Estructura	Cuatro enteros, superior izquierdo e inferior derecha, para definir rectángulo
RectangleF	Estructura	Cuatro coordenadas de coma flotante para definir un rectángulo
Size	Estructura	Tamaño de un objeto (alto y ancho), enteros
SizeF	Estructura	Tamaño de objeto (alto y ancho), como flotante
FontStyle	Enumeración	Estilos aplicables a las fuentes.

Procedimiento

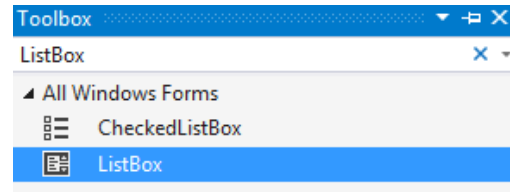
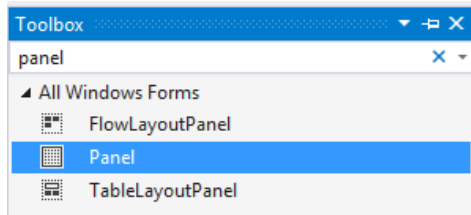
Necesariamente en todos los ejercicios debe asegurarse que exista el espacio de nombres System.Drawing (sino debemos agregarlo nosotros).

EJEMPLO No. 1: Creación elipses y rectángulos.

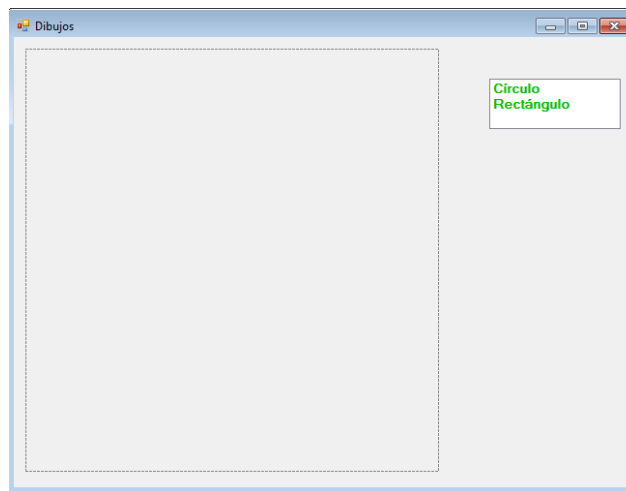
1. Para la creación del Proyecto debe elegir uno de tipo **Windows Form**



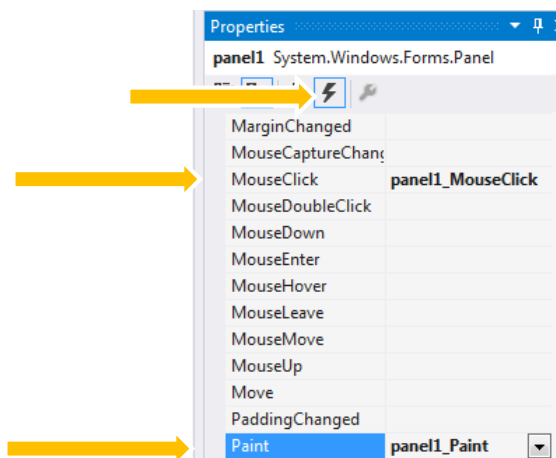
2. Puede elegir el nombre que prefiera para el proyecto, solo recuérdelo y la ubicación que le dará.
3. Una vez dentro del formulario seleccione dos herramientas del toolbox: **Panel y ListBox**



4. Con esos dos elementos deberán de construir un formulario similar al mostrado en la imagen (pueden personalizarlo según su estilo, solo recuerden que el panel será el área donde sí se puede dibujar)



5. Las dos opciones disponibles en el ListBox serán círculo y rectángulo (pueden agregarlo desde el código o desde las opciones disponibles en el área de diseño)
6. Una vez realizado esto, vamos a activar los eventos para el panel: **MouseClick** y **Paint** (para activar solo dar doble clic sobre la celda al lado del nombre)



7. Una vez realizado pasamos al código, declararemos dos variables que nos ayudarán para determinar posición del clic del mouse

```
public partial class Form1 : Form
{
    int x, y; //variables globales que permiten determinar ubicación del clic

    public Form1()
    {
        InitializeComponent();
    }
}
```

Nos da información sobre los eventos con el mouse

8. Para el evento MouseClick tendremos el siguiente código

```
private void panel1_MouseClick(object sender, MouseEventArgs e)
{
    Point punto = new Point(e.X, e.Y);
    x = punto.X;
    y = punto.Y;
    panel1.Invalidate();
}
```

Tomamos las coordenadas y las almacenamos en las variables globales

Creamos una instancia de la estructura Point, esta nos tomará las coordenadas X y Y del parámetro e

9. Para el evento Paint tendremos lo siguiente:

```
private void panel1_Paint(object sender, PaintEventArgs e)
{
    Graphics g = panel1.CreateGraphics(); //establece zona para dibujo
    Pen lapiz = new Pen(Color.Black); //declaro color de pen a utilizar

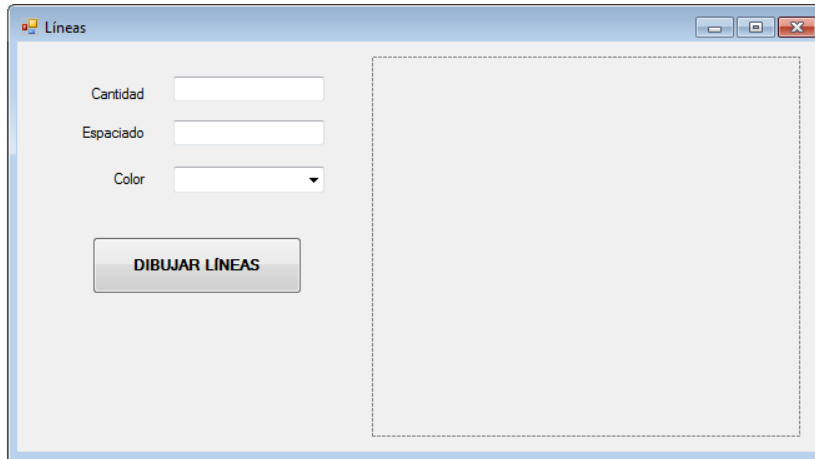
    if(listBox1.SelectedIndex==0) //si selecciona círculo
    {
        SolidBrush sb = new SolidBrush(Color.Red); //brush en rojo
        g.DrawEllipse(lapiz, x - 50, y - 50, 100, 100); /* dibujará elipse
        utilizando ese pen, con la posición en x y y menos 50 y dimensiones 100)*/
        g.FillEllipse(sb, x - 50, y - 50, 100, 100); //rellena de color la elipse dada
    }

    if (listBox1.SelectedIndex == 1) //si selecciona rectángulo
    {
        SolidBrush sb = new SolidBrush(Color.Green); //brush en verde
        g.DrawRectangle(lapiz, x - 50, y - 50, 100, 100); /* dibujará rectángulo
        utilizando ese pen, con la posición en x y y menos 50 y dimensiones 100)*/
        g.FillRectangle(sb, x - 50, y - 50, 100, 100); //rellena de color la elipse dada
    }
}
```

10. Corra la aplicación y haga clic sobre el área de dibujo. Observe qué hace el sistema

EJEMPLO No. 2: Creación de líneas

1. Abrir un nuevo proyecto de Windows Form en C#, cree un formulario que luzca similar a la siguiente imagen



Para ello vamos a requerir los siguientes elementos

Herramienta	Nombre	Texto
3 Label	Label1 Label2 Label3	Cantidad Espaciado Color
2 TextBox	txtcantidad txtespaciado	[vacío] [vacío]
1 ComboBox	cmbcolor	[valores de ítem : amarillo, rojo, azul y negro]
1 Button	btndibujar	Dibujar Líneas
1 PictureBox	areadibujo	

2. Una vez construida la interfaz sugerida procedemos a trabajar el código <F7> (no olvidar activar el evento Click del botón)

```
public partial class Form1 : Form
{
    Graphics area; //Área de trabajo

    public Form1()
    {
        InitializeComponent();
        area = areadibujo.CreateGraphics(); //establezco el área de dibujo para el picturebox
    }
}
```

3. Dentro del evento Click del botón vamos a programar lo siguiente:

```
private void btndibujar_Click(object sender, EventArgs e)
{
    Pen lapicero = new Pen(Color.Black); //color por defecto

    switch(cmbcolor.SelectedIndex) //colorear dependiendo de lo seleccionado en ComboBox
    {
        case 0: lapicero = new Pen(Color.Yellow); break; //selección amarilla
        case 1: lapicero = new Pen(Color.Red); break; //selección roja
        case 2: lapicero = new Pen(Color.Blue); break; //selección azul
        case 3: lapicero = new Pen(Color.Black); break; //selección negra
    }

    int iteraciones = int.Parse(txtcantidad.Text); //cantidad de líneas a dibujar
    int espacio = int.Parse(txtespaciado.Text); //espaciado asignado (en pixeles)

    area.Clear(Color.White); //limpia área a blanco, para que líneas no se vean una sobre otra

    int puntoinicio = 50; //inicio en un valor de y = 50

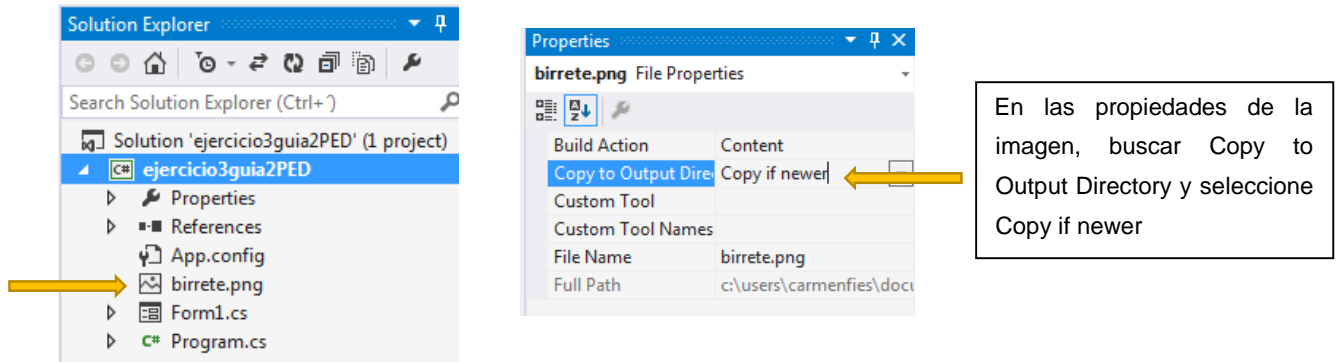
    for(int i=0; i<iteraciones; i++)
    {
        area.DrawLine(lapicero, 20, puntoinicio + (espacio * i), 300, puntoinicio + (espacio * i));
        //dibuja línea por línea de acuerdo al color dado, en x van de 20 a 300 y en y varía según iteración
    }
}
```

4. Ejecute el programa y observe qué acción realiza



EJEMPLO No. 3: Utilización de Bitmap

1. Abra un nuevo proyecto de Windows Form en C# (nómbrelo como usted prefiera)
2. Una vez cargada la ventana o formulario solamente cambie el texto del mismo a “Juego” u otro que usted prefiera.
3. Agregar al proyecto una imagen (de preferencia png)
En el explorador de soluciones, dar clic derecho sobre el nombre del proyecto → Add (Agregar) → Existing Item
4. Busca en su computadora una imagen para incluir en el proyecto y da clic en aceptar y la verá en el listado de elementos del proyecto



5. Agregaré la herramienta timer (en la guía se ha cambiado el nombre a timermov), colocar el intervalo en 100 y enabled en True. También se activa el evento Tick.
6. Activar dos eventos para el formulario (Form): “Paint” y “KeyDown”.
7. Ir al código del proyecto con F7
8. Dentro del código encontraremos esto:

```
public partial class Form1 : Form
{
    enum Posicion //define un set de constantes que pueden ser asignados a una variable
    {
        izquierda,derecha,arriba, abajo
    }

    private int x; //coordenada en x
    private int y; //coordenada en y
    private Posicion objposicion; // variable del enum Posicion

    public Form1()
    {
        InitializeComponent();
        x = 50; //inicializa en x = 50
        y = 50; //inicializa en y = 50
        objposicion = Posicion.abajo; //por defecto definimos que se mueve hacia abajo
    }
}
```

9. En el evento Paint del formulario se tiene el código:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    e.Graphics.DrawImage(new Bitmap("birrete.png"),x,y,65,65);
    //Se dibuja la imagen agregada al proyecto y se establece el punto inicial y tamaño
}
```

10. En el evento Tick del timermov:

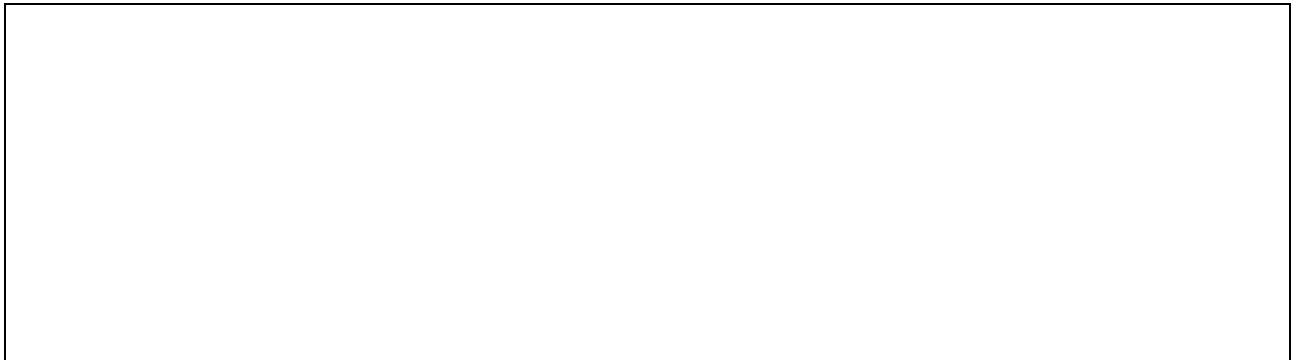
```
private void timermov_Tick(object sender, EventArgs e)
{
    if(objposicion ==Posicion.derecha)
    { x += 10; } //desplazarse 10 pixeles a la derecha
    else if(objposicion == Posicion.izquierda)
    { x -= 10; } //desplazarse 10 pixeles a la izquierda
    else if(objposicion == Posicion.arriba)
    { y -= 10; } //10 pixeles hacia arriba
    else if (objposicion == Posicion.abajo)
    { y += 10; } //10 pixeles hacia abajo

    Invalidate(); //invalida la superficie del control y hace que se vuelva a dibujar el control.
}
```

11. En el evento KeyDown del formulario:

```
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Left) //si se presiona la tecla flecha izquierda
    {
        objposicion = Posicion.izquierda;
    }
    else if (e.KeyCode == Keys.Right) //si se presiona la tecla flecha derecha
    {
        objposicion = Posicion.derecha;
    }
    else if (e.KeyCode == Keys.Up) //si se presiona la tecla flecha arriba
    {
        objposicion = Posicion.arriba;
    }
    else if (e.KeyCode == Keys.Down) //si se presiona la tecla flecha abajo
    {
        objposicion = Posicion.abajo;
    }
}
```

12. Ejecute el programa y anote los resultados



Desarrollo de habilidades

Ejercicio No. 1

Modifique el primer ejemplo de forma que dibuje figuras con colores diferentes y tamaños diferentes en cada clic

Ejercicio No. 2

Modifique el segundo ejercicio de forma que el usuario pueda decidir en qué punto inicia y termina una línea, es decir que las líneas pueden quedar inclinadas, verticales u horizontales.

Investigación complementaria

Investigación N° 1

Investigar sobre System.Threading y cómo poder utilizarla en conjunto con System.Drawing.