

Análisis y Diseño de Sistemas Informáticos
PED941
Ciclo 01-2024
G01T



DESAFÍO 1

Docente: Carmen Celia Morales Samayoa

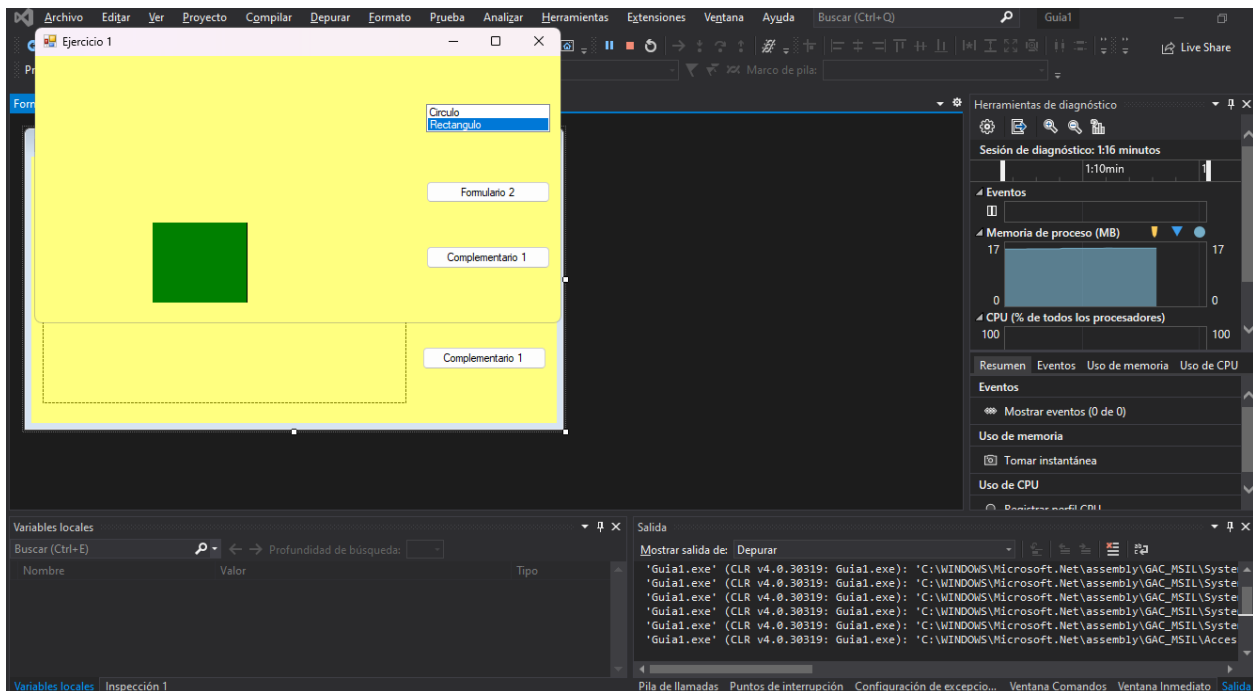
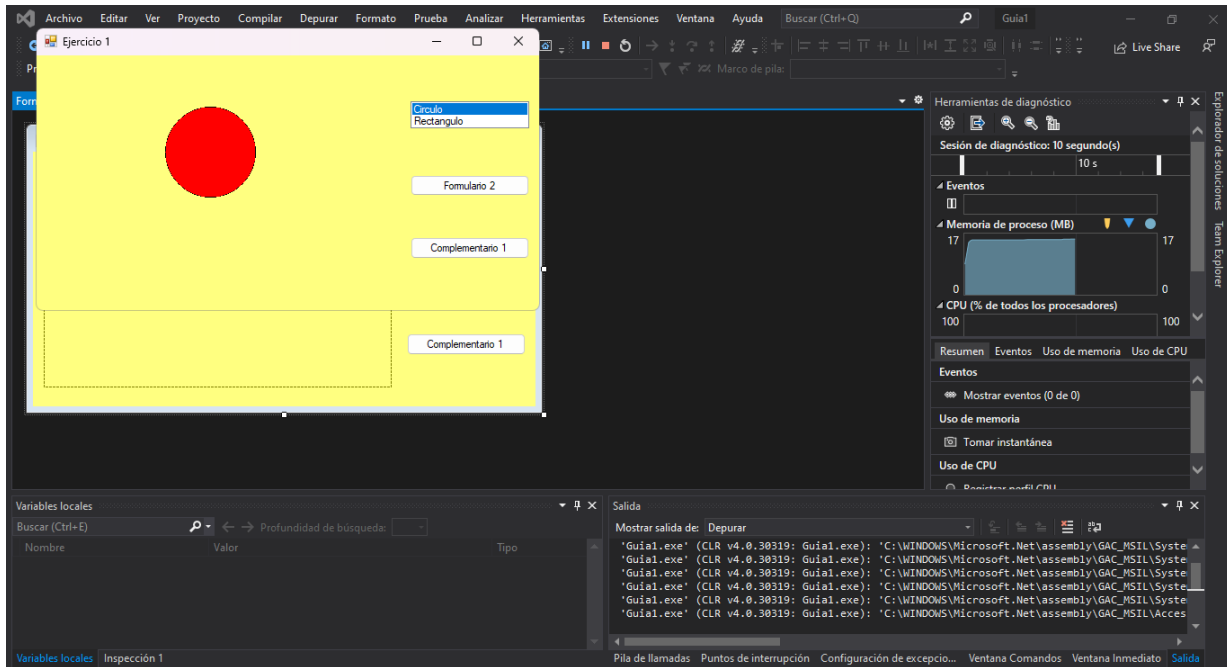
Alumno: Irvin Eduardo Gonzalez Romero

Apellidos	Nombres	Carné
González Romero	Irvin Eduardo	GR202825

REPOSITORIO: <https://github.com/Irvin-g/Guia01PED.git>

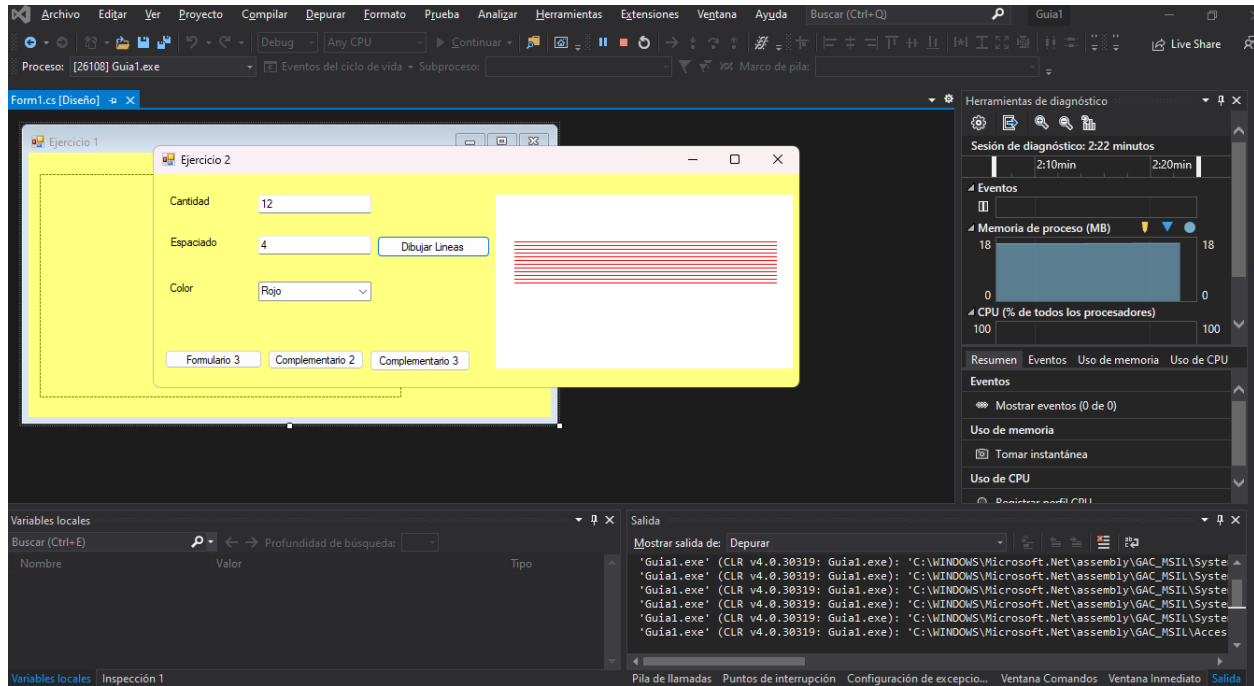
Ejemplo 1:

Solo mostrare la parte funcional ya que el código es proporcionado.



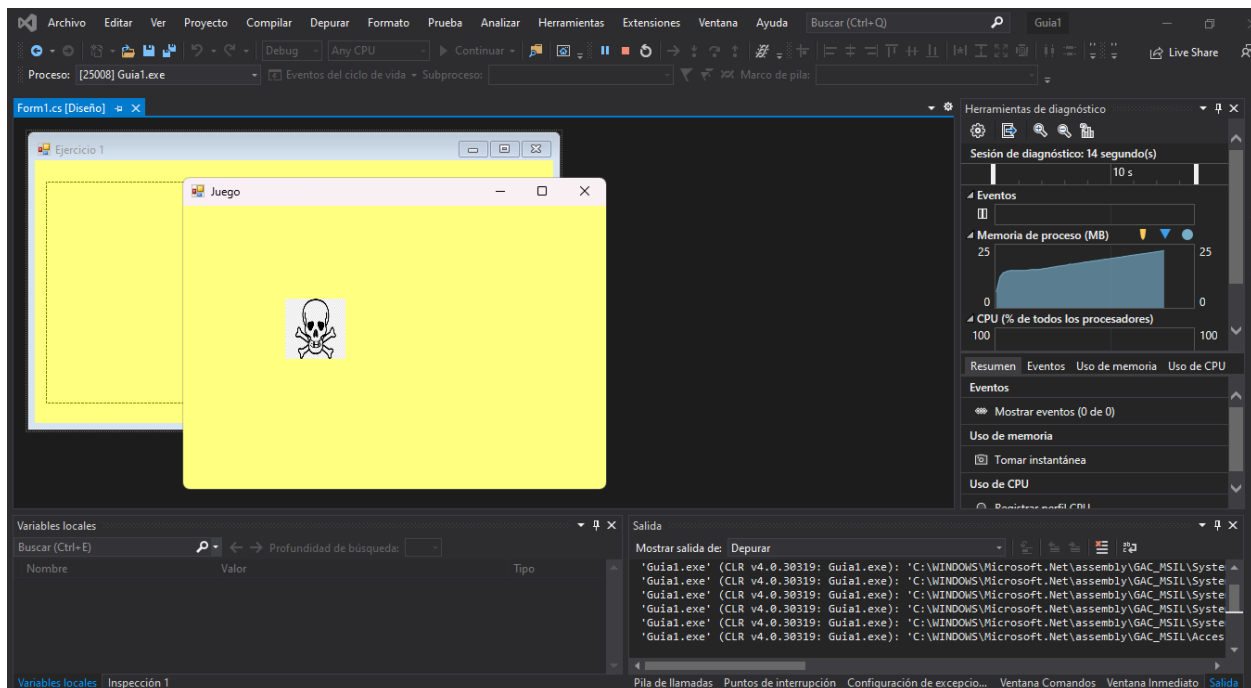
Ejemplo 2:

Solo mostrare la parte funcional ya que el código es proporcionado.



Ejemplo 3:

Solo mostrare la parte funcional ya que el código es proporcionado.



Ejercicio complementario 1:

Resumen de cambios de código.

- Color Aleatorio:

En lugar de utilizar colores predefinidos (rojo y verde), se ha cambiado a colores aleatorios utilizando la clase Random para generar valores de color RGB aleatorios.

- Tamaño Aleatorio:

Se ha introducido un tamaño aleatorio para las figuras (círculo y rectángulo) en el rango de 50 a 150.

- Manejo del Panel y Evento de Click:

El evento panel1_Paint_1 ahora utiliza el objeto Graphics proporcionado por el evento, eliminando la creación de un nuevo objeto Graphics usando panel1.CreateGraphics().

El manejo del evento panel1_MouseClick_1 ahora guarda las coordenadas del clic (e.X y e.Y) directamente en x e y, eliminando la necesidad de crear un objeto Point.

- Manejo de Cambios en el ListBox:

Se ha agregado una llamada a panel1.Invalidate() en el evento listBox1_SelectedIndexChanged para que el panel se invalide y se vuelva a pintar cuando cambie la selección en el ListBox.

Código

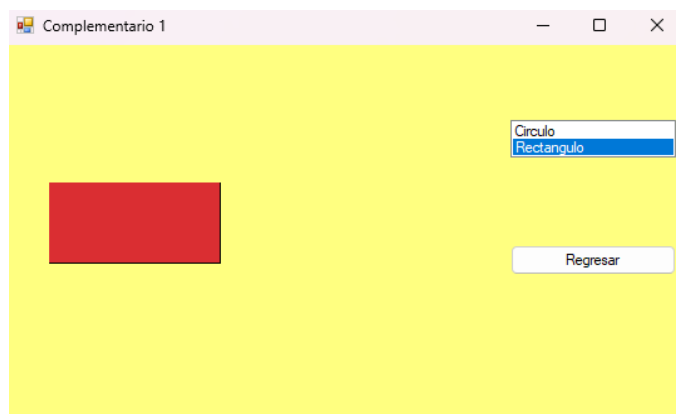
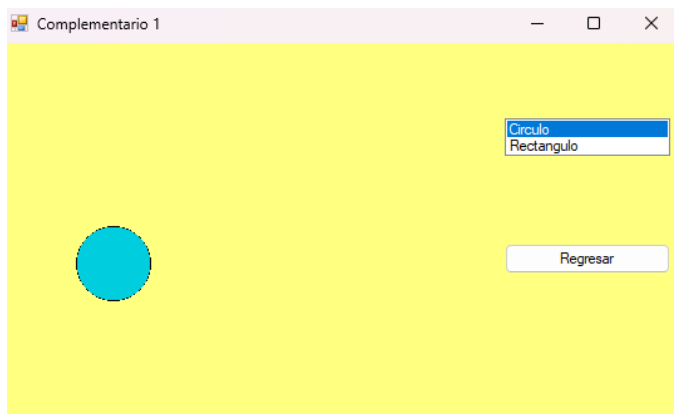
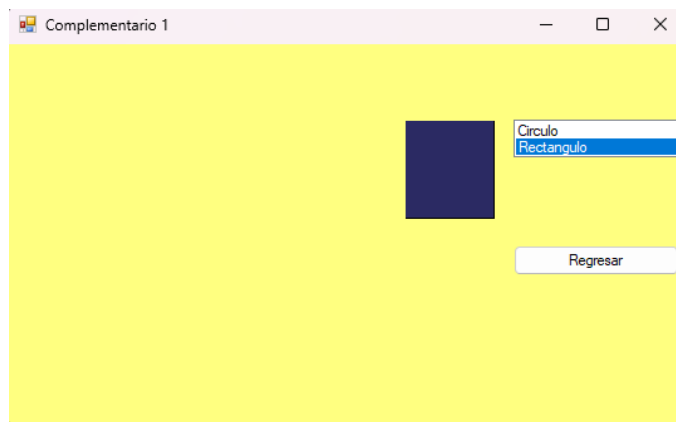
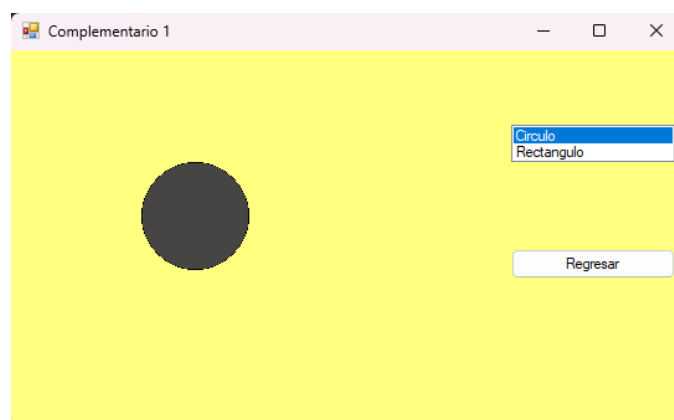
```
11 namespace Gui1
12 {
13     4 referencias
14     public partial class Form5 : Form
15     {
16
17         private Random random;
18         int x, y; //Permite determinar la ubicacion del click
19         1 referencia
20         public Form5()
21         {
22             InitializeComponent();
23             random = new Random();
24         }
25
26         1 referencia
27         private void Form5_Load(object sender, EventArgs e)
28         {
29         }
30
31
32
33         1 referencia
34         private void panel1_Paint_1(object sender, PaintEventArgs e)
35         {
36             Graphics g = e.Graphics; // Utilizamos el objeto Graphics proporcionado por el evento
37             Pen lapiz = new Pen(Color.Black); // Declaramos color del PEN a utilizar
38
39             if (listBox1.SelectedIndex == 0) // Si selecciona círculo
40             {
41                 SolidBrush sb = new SolidBrush(Color.FromArgb(random.Next(256), random.Next(256), random.Next(256))); // Brush con color aleatorio
42                 int diametro = random.Next(50, 150); // Tamaño aleatorio en el rango de 50 a 150
43                 g.DrawEllipse(lapiz, x - diametro / 2, y - diametro / 2, diametro, diametro); // Dibujar círculo con posición y dimensiones aleatorias
44                 g.FillEllipse(sb, x - diametro / 2, y - diametro / 2, diametro, diametro); // Rellenar de color el círculo dado
45             }
46         }
47     }
48 }
```

```

46     else if (listBox1.SelectedIndex == 1) // Si selecciona rectángulo
47     {
48         SolidBrush sb = new SolidBrush(Color.FromArgb(random.Next(256), random.Next(256), random.Next(256))); // Brush con color aleatorio
49         int ancho = random.Next(50, 150); // Ancho aleatorio en el rango de 50 a 150
50         int alto = random.Next(50, 150); // Alto aleatorio en el rango de 50 a 150
51         g.DrawRectangle(lapiz, x - ancho / 2, y - alto / 2, ancho, alto); // Dibujar rectángulo con posición y dimensiones aleatorias
52         g.FillRectangle(sb, x - ancho / 2, y - alto / 2, ancho, alto); // Rellenar de color el rectángulo dado
53     }
54 }
55
56 // 1 referencia
57 private void panel1_MouseClick_1(object sender, MouseEventArgs e)
58 {
59     x = e.X;
60     y = e.Y;
61     panel1.Invalidate();
62 }
63
64 // 1 referencia
65 private void button2_Click(object sender, EventArgs e)
66 {
67 }
68
69 // 0 referencias
70 private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
71 {
72     panel1.Invalidate(); // Al cambiar la selección, se invalida el panel para volver a pintar las figuras
73 }
74
75 }

```

Resultado



Ejercicio complementario 2:

Resumen de cambios de código.

- Manejo de Área de Dibujo:

En lugar de crear el objeto Graphics (area) dentro del constructor, se trasladó a los métodos específicos donde se necesita. En este caso, se creó dentro del método `btndibujar_Click` antes de realizar el dibujo. Esto evita la creación prematura del objeto Graphics y asegura que se cree cuando sea necesario.

- Parámetros de Dibujo:

Se han agregado nuevos controles (cuadros de texto) para especificar el punto de inicio (`puntoInicioX` y `puntoInicioY`) y el punto de fin (`puntoFinX` y `puntoFinY`). Estos parámetros permiten mayor flexibilidad al definir la posición inicial y final de las líneas dibujadas.

- Uso de Controles de Texto Adicionales:

Se han agregado controles de texto (`txtpuntoinicioX`, `txtpuntoinicioY`, `txtpuntofinX`, `txtpuntofinY`) para ingresar los valores de punto de inicio y punto de fin, proporcionando una interfaz más amigable para especificar estos valores.

Código

```
11 namespace Guial
12 {
13     4 referencias
14     public partial class Form6 : Form
15     {
16         Graphics area; // área de dibujo
17
18         1 referencia
19         public Form6()
20         {
21             InitializeComponent();
22             area = areadibujo.CreateGraphics(); // Establezco área de dibujo
23         }
24
25         1 referencia
26         private void Form6_Load(object sender, EventArgs e)
27         {
28         }
29
30         1 referencia
31         private void btndibujar_Click(object sender, EventArgs e)
32         {
33             Pen lapicero = new Pen(Color.Black); // color por defecto
34
35             switch (cmbcolor.SelectedIndex) // colorear dependiendo de lo seleccionado
36             {
37                 case 0: lapicero = new Pen(Color.Yellow); break;
38                 case 1: lapicero = new Pen(Color.Red); break;
39                 case 2: lapicero = new Pen(Color.Blue); break;
40                 case 3: lapicero = new Pen(Color.Black); break;
41             }
42
43             int interacciones = int.Parse(txtcantidad.Text); // cantidad de líneas a dibujar
44             int espaciado = int.Parse(txtespaciado.Text); // espaciado asignado
45
46             int puntoInicioX = int.Parse(txtpuntoinicioX.Text); // punto de inicio en el eje X
47             int puntoInicioY = int.Parse(txtpuntoinicioY.Text); // punto de inicio en el eje Y
```

```

45     int puntoInicioY = int.Parse(txtpuntoinicioY.Text); // punto de inicio en el eje Y
46     int puntoFinX = int.Parse(txtpuntofinX.Text); // punto de fin en el eje X
47     int puntoFinY = int.Parse(txtpuntofinY.Text); // punto de fin en el eje Y
48
49     area.Clear(Color.White); // limpia área a blanco
50
51     for (int i = 0; i < interacciones; i++)
52     {
53         area.DrawLine(lapicero, puntoInicioX + (espaciado * i), puntoInicioY, puntoFinX + (espaciado * i), puntoFinY);
54         // dibuja línea por línea de acuerdo al color dado, en x varia según la interacción y en y va de puntoInicioY a puntoFinY
55     }
56 }
57
58 }
59
60

```

Resultado

Complementario 2

Cantidad:

Espaciado:


Color:

Punto inicio X:

Punto fin X:

Punto inicio Y:

Punto fin Y:



Complementario 2

Cantidad:

Espaciado:

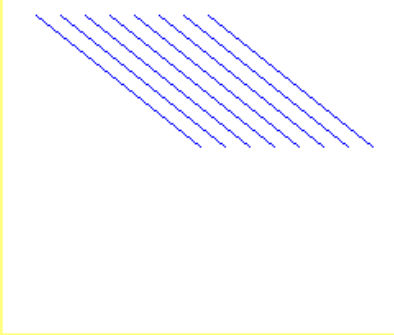
Color:

Punto inicio X:

Punto fin X:

Punto inicio Y:

Punto fin Y:



Ejercicio complementario 3:

Resumen de cambios de código.

- Manejo de Direcciones:

En lugar de solo mover la imagen hacia arriba, abajo, izquierda o derecha, ahora se maneja una serpiente que se mueve en una cuadrícula y puede cambiar de dirección con las teclas de flecha. La dirección de la serpiente (`objposicion`) se actualiza en el evento `Form7_KeyDown_1` según la tecla presionada.

- Temporizador y Movimiento Continuo:

Se ha introducido un temporizador (`timermov_Tick`) que maneja el movimiento continuo de la serpiente en la dirección actual.

- Dibujo del Juego:

El método `Form7_Paint_1` se encarga de dibujar el fondo cuadriculado, la serpiente (compuesta por segmentos verdes), y la comida (un círculo rojo). En cambio, `Form3_Paint` solo dibuja una imagen en una posición específica.

- Reinicio del Juego:

Si la serpiente toca los bordes de la ventana o choca consigo misma, se reinicia el juego en el método `ReiniciarJuego`. Además, la comida se genera en una nueva posición.

- Generación Aleatoria de Comida:

Se ha introducido el método `GenerarComida` para obtener una posición aleatoria para la comida en la cuadrícula.

- Uso de `List<Point>` para la Serpiente:

En lugar de tener variables `x` y `y` individuales, ahora la serpiente se almacena como una lista de puntos (serpiente). La cabeza de la serpiente es el primer elemento de la lista.

Código

```
11 namespace Guial
12 {
13     4 referencias
14     public partial class Form7 : Form
15     {
16         15 referencias
17         enum Posicion // Define un set de constantes que pueden ser asignados a una variable
18         {
19             Izquierda, Derecha, Arriba, Abajo
20         }
21         private int x; // Coordenada en x
22         private int y; // Coordenada en y
23         private Posicion objposicion; // Variable del enum Posicion
24         private List<Point> serpiente; // Lista para almacenar los segmentos de la serpiente
25         private Point comida; // Posición de la comida
26         private const int TamanoCelda = 20; // Tamaño de cada celda
27
28         1 referencia
29         public Form7()
30         {
31             InitializeComponent();
32             x = 50; // Iniciamos x en 50
33             y = 50; // Iniciamos y en 50
34             objposicion = Posicion.Abajo; // Por defecto definimos que se mueve hacia abajo
35
36             serpiente = new List<Point>(); // Inicializar la lista de la serpiente
37             serpiente.Add(new Point(5, 5)); // Posición inicial de la serpiente
38
39             comida = GenerarComida(); // Generar posición inicial de la comida
40
41             // Configuración de la ventana de juego
42             DoubleBuffered = true;
43             KeyPreview = true;
44             //ClientSize = new Size(400, 400);
45             //Text = "Juego de la Serpiente";
46             //BackColor = Color.Black;
47
48             // Iniciar el temporizador
```

```
46         System.Windows.Forms.Timer timermov = new System.Windows.Forms.Timer();
47         timermov.Interval = 100;
48         timermov.Tick += timermov_Tick;
49         timermov.Start();
50     }
51
52     1 referencia
53     private void timermov_Tick(object sender, EventArgs e)
54     {
55         Point cabeza = serpiente[0];
56         Point nuevaCabeza = cabeza;
57
58         switch (objposicion)
59         {
60             case Posicion.Derecha:
61                 nuevaCabeza.X += 1;
62                 break;
63             case Posicion.Izquierda:
64                 nuevaCabeza.X -= 1;
65                 break;
66             case Posicion.Arriba:
67                 nuevaCabeza.Y -= 1;
68                 break;
69             case Posicion.Abajo:
70                 nuevaCabeza.Y += 1;
71                 break;
72         }
73
74         if (nuevaCabeza == comida)
75         {
76             serpiente.Insert(0, nuevaCabeza); // Agregamos la nueva cabeza en la posición actual
77             comida = GenerarComida();
78         }
79         else
80         {
81             serpiente.Insert(0, nuevaCabeza);
82             if (serpiente.Count > 1)
83                 serpiente.RemoveAt(serpiente.Count - 1); // Quitamos la última cola si no se ha comido
```

```

83     }
84
85     if (nuevaCabeza.X < 0 || nuevaCabeza.X >= ClientSize.Width / TamanoCelda ||
86         nuevaCabeza.Y < 0 || nuevaCabeza.Y >= ClientSize.Height / TamanoCelda)
87     {
88
89         ReiniciarJuego();
90         return;
91     }
92
93     if (serpiente.Skip(1).Any(seg => seg == nuevaCabeza))
94     {
95
96         ReiniciarJuego();
97         return;
98     }
99
100     Invalidate();
101 }
102
103 3 referencias
104 private Point GenerarComida()
105 {
106     // Generar una posición aleatoria para la comida
107     Random rand = new Random();
108     int x = rand.Next(ClientSize.Width / TamanoCelda);
109     int y = rand.Next(ClientSize.Height / TamanoCelda);
110     return new Point(x, y);
111 }
112
113 2 referencias
114 private void ReiniciarJuego()
115 {
116     // Reiniciar el juego
117     serpiente.Clear();
118     serpiente.Add(new Point(5, 5));
119     objposicion = Posicion.Abajo;
120     comida = GenerarComida();
121     Invalidate();

```

```

119 }
120
121 1 referencia
122 private void Form7_Load(object sender, EventArgs e)
123 {
124 }
125
126 1 referencia
127 private void Form7_KeyDown_1(object sender, KeyEventArgs e)
128 {
129     // Cambiar la dirección de la serpiente según la tecla presionada
130     switch (e.KeyCode)
131     {
132         case Keys.Left:
133             if (objposicion != Posicion.Derecha)
134                 objposicion = Posicion.Izquierda;
135             break;
136         case Keys.Right:
137             if (objposicion != Posicion.Izquierda)
138                 objposicion = Posicion.Derecha;
139             break;
140         case Keys.Up:
141             if (objposicion != Posicion.Abajo)
142                 objposicion = Posicion.Arriba;
143             break;
144         case Keys.Down:
145             if (objposicion != Posicion.Arriba)
146                 objposicion = Posicion.Abajo;
147             break;
148     }
149 }
150
151 1 referencia
152 private void Form7_Paint_1(object sender, PaintEventArgs e)
153 {
154     // Dibujar fondo cuadrulado
155     for (int i = 0; i < ClientSize.Width / TamanoCelda; i++)

```

```

154 {
155     for (int j = 0; j < ClientSize.Height / TamanioCelda; j++)
156     {
157         if ((i + j) % 2 == 0)
158         {
159             e.Graphics.FillRectangle(Brushes.LightGray, i * TamanioCelda, j * TamanioCelda, TamanioCelda, TamanioCelda);
160         }
161     }
162 }
163
164 //e.Graphics.DrawImage(new Bitmap("error by 404.png"), x, y, 65, 65);
165
166 foreach (Point segmento in serpiente)
167 {
168     e.Graphics.FillRectangle(Brushes.Green, segmento.X * TamanioCelda, segmento.Y * TamanioCelda, TamanioCelda, TamanioCelda);
169 }
170
171 e.Graphics.FillEllipse(Brushes.Red, comida.X * TamanioCelda, comida.Y * TamanioCelda, TamanioCelda, TamanioCelda);
172 }
173 }
174 }
175

```

Resultado

Debido a que es una imagen no se puede apreciar muy bien el movimiento de la serpiente cuando come la comida que es el círculo rojo, y al tocar la esquina se reinicia

