```
    partial class Form1
    {
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
            this.pbLienzo = new System.Windows.Forms.PictureBox();
            this.btnIniciarPausar = new System.Windows.Forms.Button();
            this.timer1 = new System.Windows.Forms.Timer(this.components);
            this.lblVelocidad = new System.Windows.Forms.Label();
            this.tbVelocidad = new System.Windows.Forms.TrackBar();
            this.lblÁngulo = new System.Windows.Forms.Label();
            this.tbÁngulo = new System.Windows.Forms.TrackBar();
            this.btnReiniciar = new System.Windows.Forms.Button();
            this.cbMostrarTrayectoria = new System.Windows.Forms.CheckBox();
            this.lblPuntuacion = new System.Windows.Forms.Label();
            this.lblLanzamientos = new System.Windows.Forms.Label();
            this.panelControles = new System.Windows.Forms.Panel();
```

```
        this.lblInstrucciones = new System.Windows.Forms.Label();
        ((System.ComponentModel.ISupportInitialize)(this.pbLienzo)).BeginInit();
        ((System.ComponentModel.ISupportInitialize)(this.tbVelocidad)).BeginInit();
        ((System.ComponentModel.ISupportInitialize)(this.tbÁngulo)).BeginInit();
        this.panelControles.SuspendLayout();
        this.SuspendLayout();
        //
        // pbLienzo
        //
        this.pbLienzo.Anchor =
((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
        | System.Windows.Forms.AnchorStyles.Left)
        | System.Windows.Forms.AnchorStyles.Right)));
        this.pbLienzo.BackColor = System.Drawing.Color.White;
        this.pbLienzo.Location = new System.Drawing.Point(12, 12);
        this.pbLienzo.Name = "pbLienzo";
        this.pbLienzo.Size = new System.Drawing.Size(944, 450);
        this.pbLienzo.TabIndex = 0;
        this.pbLienzo.TabStop = false;
        this.pbLienzo.Paint += new
System.Windows.Forms.PaintEventHandler(this.pbLienzo_Paint);
        //
        // btnIniciarPausar
        //
        this.btnIniciarPausar.Location = new System.Drawing.Point(20, 151);
        this.btnIniciarPausar.Name = "btnIniciarPausar";
        this.btnIniciarPausar.Size = new System.Drawing.Size(120, 48);
        this.btnIniciarPausar.TabIndex = 1;
        this.btnIniciarPausar.Text = "Lanzar";
        this.btnIniciarPausar.UseVisualStyleBackColor = true;
        this.btnIniciarPausar.Click += new System.EventHandler(this.btnIniciarPausar_Click);
        //
        // timer1
        //
        this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
        //
        // lblVelocidad
        //
        this.lblVelocidad.AutoSize = true;
        this.lblVelocidad.Location = new System.Drawing.Point(16, 16);
        this.lblVelocidad.Name = "lblVelocidad";
        this.lblVelocidad.Size = new System.Drawing.Size(144, 20);
        this.lblVelocidad.TabIndex = 2;
```

```
        this.lblVelocidad.Text = "Velocidad: 5 px/tick";
        //
        // tbVelocidad
        //
        this.tbVelocidad.Location = new System.Drawing.Point(20, 39);
        this.tbVelocidad.Maximum = 15;
        this.tbVelocidad.Minimum = 1;
        this.tbVelocidad.Name = "tbVelocidad";
        this.tbVelocidad.Size = new System.Drawing.Size(242, 69);
        this.tbVelocidad.TabIndex = 3;
        this.tbVelocidad.Value = 5;
        this.tbVelocidad.ValueChanged += new
System.EventHandler(this.tbVelocidad_ValueChanged);
        //
        // lblÁngulo
        //
        this.lblÁngulo.AutoSize = true;
        this.lblÁngulo.Location = new System.Drawing.Point(16, 87);
        this.lblÁngulo.Name = "lblÁngulo";
        this.lblÁngulo.Size = new System.Drawing.Size(90, 20);
        this.lblÁngulo.TabIndex = 4;
        this.lblÁngulo.Text = "Ángulo: 45°";
        //
        // tbÁngulo
        //
        this.tbÁngulo.Location = new System.Drawing.Point(310, 38);
        this.tbÁngulo.Maximum = 90;
        this.tbÁngulo.Name = "tbÁngulo";
        this.tbÁngulo.Size = new System.Drawing.Size(242, 69);
        this.tbÁngulo.TabIndex = 5;
        this.tbÁngulo.Value = 45;
        this.tbÁngulo.ValueChanged += new
System.EventHandler(this.tbÁngulo_ValueChanged);
        //
        // btnReiniciar
        //
        this.btnReiniciar.Location = new System.Drawing.Point(146, 151);
        this.btnReiniciar.Name = "btnReiniciar";
        this.btnReiniciar.Size = new System.Drawing.Size(116, 48);
        this.btnReiniciar.TabIndex = 6;
        this.btnReiniciar.Text = "Reiniciar";
        this.btnReiniciar.UseVisualStyleBackColor = true;
        this.btnReiniciar.Click += new System.EventHandler(this.btnReiniciar_Click);
        //
```

```
        // cbMostrarTrayectoria
        //
        this.cbMostrarTrayectoria.AutoSize = true;
        this.cbMostrarTrayectoria.Checked = true;
        this.cbMostrarTrayectoria.CheckState = System.Windows.Forms.CheckState.Checked;
        this.cbMostrarTrayectoria.Location = new System.Drawing.Point(20, 205);
        this.cbMostrarTrayectoria.Name = "cbMostrarTrayectoria";
        this.cbMostrarTrayectoria.Size = new System.Drawing.Size(171, 24);
        this.cbMostrarTrayectoria.TabIndex = 7;
        this.cbMostrarTrayectoria.Text = "Mostrar Trayectoria";
        this.cbMostrarTrayectoria.UseVisualStyleBackColor = true;
        this.cbMostrarTrayectoria.CheckedChanged += new
System.EventHandler(this.cbMostrarTrayectoria_CheckedChanged);
        //
        // lblPuntuacion
        //
        this.lblPuntuacion.AutoSize = true;
        this.lblPuntuacion.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.lblPuntuacion.Location = new System.Drawing.Point(16, 242);
        this.lblPuntuacion.Name = "lblPuntuacion";
        this.lblPuntuacion.Size = new System.Drawing.Size(145, 25);
        this.lblPuntuacion.TabIndex = 8;
        this.lblPuntuacion.Text = "Puntuación: 0";
        //
        // lblLanzamientos
        //
        this.lblLanzamientos.AutoSize = true;
        this.lblLanzamientos.Font = new System.Drawing.Font("Microsoft Sans Serif", 10F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
        this.lblLanzamientos.Location = new System.Drawing.Point(16, 276);
        this.lblLanzamientos.Name = "lblLanzamientos";
        this.lblLanzamientos.Size = new System.Drawing.Size(171, 25);
        this.lblLanzamientos.TabIndex = 9;
        this.lblLanzamientos.Text = "Lanzamientos: 5";
        //
        // panelControles
        //
        this.panelControles.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bottom |
System.Windows.Forms.AnchorStyles.Left)));
        this.panelControles.BackColor = System.Drawing.Color.WhiteSmoke;
        this.panelControles.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
        this.panelControles.Controls.Add(this.lblInstrucciones);
```

```
            this.panelControles.Controls.Add(this.lblVelocidad);
            this.panelControles.Controls.Add(this.lblLanzamientos);
            this.panelControles.Controls.Add(this.tbVelocidad);
            this.panelControles.Controls.Add(this.lblPuntuacion);
            this.panelControles.Controls.Add(this.lblÁngulo);
            this.panelControles.Controls.Add(this.cbMostrarTrayectoria);
            this.panelControles.Controls.Add(this.tbÁngulo);
            this.panelControles.Controls.Add(this.btnReiniciar);
            this.panelControles.Controls.Add(this.btnIniciarPausar);
            this.panelControles.Location = new System.Drawing.Point(12, 468);
            this.panelControles.Name = "panelControles";
            this.panelControles.Size = new System.Drawing.Size(944, 331);
            this.panelControles.TabIndex = 10;
            //
            // lblInstrucciones
            //
            this.lblInstrucciones.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Right)));
            this.lblInstrucciones.BorderStyle = System.Windows.Forms.BorderStyle.FixedSingle;
            this.lblInstrucciones.Font = new System.Drawing.Font("Microsoft Sans Serif", 9F,
System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.lblInstrucciones.Location = new System.Drawing.Point(620, 16);
            this.lblInstrucciones.Name = "lblInstrucciones";
            this.lblInstrucciones.Size = new System.Drawing.Size(300, 285);
            this.lblInstrucciones.TabIndex = 10;
            this.lblInstrucciones.Text = resources.GetString("lblInstrucciones.Text");
            //
            // Form1
            //
            this.AutoScaleDimensions = new System.Drawing.SizeF(9F, 20F);
            this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
            this.ClientSize = new System.Drawing.Size(968, 811);
            this.Controls.Add(this.panelControles);
            this.Controls.Add(this.pbLienzo);
            this.MinimumSize = new System.Drawing.Size(800, 600);
            this.Name = "Form1";
            this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "Desafío del Lanzamiento Preciso";
            ((System.ComponentModel.ISupportInitialize)(this.pbLienzo)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.tbVelocidad)).EndInit();
            ((System.ComponentModel.ISupportInitialize)(this.tbÁngulo)).EndInit();
            this.panelControles.ResumeLayout(false);
            this.panelControles.PerformLayout();
```

```
            this.ResumeLayout(false);

        }

        #endregion

        private System.Windows.Forms.PictureBox pbLienzo;
        private System.Windows.Forms.Button btnIniciarPausar;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.Label lblVelocidad;
        private System.Windows.Forms.TrackBar tbVelocidad;
        private System.Windows.Forms.Label lblÁngulo;
        private System.Windows.Forms.TrackBar tbÁngulo;
        private System.Windows.Forms.Button btnReiniciar;
        private System.Windows.Forms.CheckBox cbMostrarTrayectoria;
        private System.Windows.Forms.Label lblPuntuacion;
        private System.Windows.Forms.Label lblLanzamientos;
        private System.Windows.Forms.Panel panelControles;
        private System.Windows.Forms.Label lblInstrucciones;
    }
}
    public class Círculo
    {
        public bool EnMovimiento { get; set; }
        public PointF Centro { get; set; }
        public float Radio { get; set; }
        public float VelocidadInicial { get; set; }
        public float ÁnguloLanzamiento { get; set; }
        public float VelocidadX { get; private set; }
        public float VelocidadY { get; private set; }

        public Círculo(PointF centro, float radio)
        {
            Centro = centro;
            Radio = radio;
        }

        public void ActualizarVelocidadesDesdeÁngulo()
        {
            double anguloRadianes = ÁnguloLanzamiento * Math.PI / 180;
            VelocidadX = VelocidadInicial * (float)Math.Cos(anguloRadianes);
            VelocidadY = VelocidadInicial * (float)Math.Sin(anguloRadianes);
        }
```

```
        public void ConfigurarLanzamiento(float velocidadInicial, float ánguloLanzamiento)
        {
            VelocidadInicial = velocidadInicial;
            ÁnguloLanzamiento = ánguloLanzamiento;
            ActualizarVelocidadesDesdeÁngulo();
        }

        public void ReiniciarPosición()
        {
            // Implementar lógica para reiniciar la posición del círculo
        }

        public void Mover()
        {
            // Implementar lógica para mover el círculo
        }

        public void Dibujar(Graphics g)
        {
            // Implementar lógica para dibujar el círculo
        }

        public void ManejarBordes(int anchoLienzo, int altoLienzo)
        {
            // Implementar lógica para manejar los bordes del lienzo
        }

        public Rectangle ObtenerRectangulo()
        {
            return new Rectangle((int)(Centro.X - Radio), (int)(Centro.Y - Radio), (int)(Radio * 2),
(int)(Radio * 2));
        }
    }
}
    public partial class Form1 : Form
    {
        #region Campos Privados

        private Círculo círculo;
        private bool movimientoIniciado = false;
        private List<Point> puntosTrayectoria = new List<Point>();
        private bool mostrarTrayectoria = true;
        private float velocidadInicial = 5.0f;
        private float ánguloLanzamiento = 45.0f;
```

```
private int intervaloTimer = 16;

// Nuevos campos para el juego
private List<Objetivo> objetivos = new List<Objetivo>();
private int puntuacion = 0;
private int lanzamientosRestantes = 5;
private SoundPlayer sonidoLanzamiento;
private SoundPlayer sonidoImpacto;
private Image imagenFondo;
private bool juegoTerminado = false;
private bool arrastrando = false;
private Point puntoInicio;
private Point puntoFin;

#endregion

#region Constructor y Métodos de Inicialización

public Form1()
{
    InitializeComponent();

    // Inicializar el círculo en la posición inicial
    puntoInicio = new Point(50, pbLienzo.Height - 50);
    círculo = new Círculo(puntoInicio, 15f);
    círculo.VelocidadInicial = velocidadInicial;
    círculo.ÁnguloLanzamiento = ánguloLanzamiento;
    círculo.ActualizarVelocidadesDesdeÁngulo();

    // Configurar el timer para una animación fluida
    timer1.Interval = intervaloTimer;

    // Configurar la interfaz de usuario
    ActualizarEtiquetas();

    // Configurar el trackbar de velocidad
    tbVelocidad.Minimum = 1;
    tbVelocidad.Maximum = 15;
    tbVelocidad.Value = (int)velocidadInicial;

    // Configurar el trackbar del ángulo
    tbÁngulo.Minimum = 0;
    tbÁngulo.Maximum = 90;
    tbÁngulo.Value = (int)ánguloLanzamiento;
```

```
            // Inicializar recursos del juego
            InicializarJuego();

            // Configurar eventos del mouse
            pbLienzo.MouseDown += PbLienzo_MouseDown;
            pbLienzo.MouseMove += PbLienzo_MouseMove;
            pbLienzo.MouseUp += PbLienzo_MouseUp;
        }

        /// <summary>
        /// Inicializa los recursos y elementos del juego.
        /// </summary>
        private void InicializarJuego()
        {
            // Crear objetivos con diferentes posiciones, tamaños y valores
            CrearObjetivos();

            // Cargar recursos
            try
            {
                sonidoLanzamiento = new
SoundPlayer(@"C:\Users\Matias\Documents\MATIAS\PREPA\PROGRAMACION\DesafioMovP
arabolico\Sounds\lanzamiento.wav");
                sonidoImpacto = new
SoundPlayer(@"C:\Users\Matias\Documents\MATIAS\PREPA\PROGRAMACION\DesafioMovP
arabolico\Sounds\impacto.wav");
                imagenFondo =
Image.FromFile(@"C:\Users\Matias\Documents\MATIAS\PREPA\PROGRAMACION\DesafioMo
vParabolico\fondo.jpg");
            }
            catch
            {
                // Si hay problemas cargando los recursos, continuar sin ellos
            }

            // Actualizar la etiqueta para mostrar lanzamientos restantes
            ActualizarEtiquetasJuego();

            // Cambiar texto del botón
            btnIniciarPausar.Text = "Lanzar";

            // Cambiar título del formulario
            this.Text = "Desafío del Lanzamiento Preciso";
```

```
}

/// <summary>
/// Crea los objetivos del juego con diferentes propiedades.
/// </summary>
private void CrearObjetivos()
{
    objetivos.Clear();
    Random rnd = new Random();

    // Crear entre 3 y 5 objetivos
    int numObjetivos = rnd.Next(3, 6);

    for (int i = 0; i < numObjetivos; i++)
    {
        // Posición aleatoria (evitando la zona de lanzamiento)
        int x = rnd.Next(pbLienzo.Width / 2, pbLienzo.Width - 100);
        int y = rnd.Next(50, pbLienzo.Height - 100);

        // Tamaño aleatorio
        int ancho = rnd.Next(30, 80);
        int alto = rnd.Next(30, 80);

        // Puntos basados en el tamaño (más pequeño = más puntos)
        int puntos = (int)(1000 / (ancho * alto) * 100);

        // Crear y añadir el objetivo
        Objetivo objetivo = new Objetivo(
            new Point(x, y),
            new Size(ancho, alto),
            puntos,
            @"*****\objetivo.png"
        );

        objetivos.Add(objetivo);
    }
}

/// <summary>
/// Actualiza las etiquetas relacionadas con el juego.
/// </summary>
private void ActualizarEtiquetasJuego()
{
    // Actualizar etiquetas existentes
```

```
            lblVelocidad.Text = $"Velocidad: {velocidadInicial} px/tick";
            lblÁngulo.Text = $"Ángulo: {ánguloLanzamiento}°";

            // Asumimos que tenemos nuevas etiquetas para puntuación y lanzamientos
            // Estas etiquetas debemos añadirlas al diseñador
            // lblPuntuacion.Text = $"Puntuación: {puntuacion}";
            // lblLanzamientos.Text = $"Lanzamientos: {lanzamientosRestantes}";
        }

        /// <summary>
        /// Actualiza las etiquetas de velocidad y ángulo.
        /// </summary>
        private void ActualizarEtiquetas()
        {
            lblVelocidad.Text = $"Velocidad: {velocidadInicial} px/tick";
            lblÁngulo.Text = $"Ángulo: {ánguloLanzamiento}°";
        }

        #endregion

        #region Eventos de Controles

        private void btnIniciarPausar_Click(object sender, EventArgs e)
        {
            if (juegoTerminado)
            {
                ReiniciarJuego();
                return;
            }

            if (lanzamientosRestantes <= 0)
            {
                MessageBox.Show("¡No te quedan más lanzamientos!", "Fin del juego",
MessageBoxButtons.OK, MessageBoxIcon.Information);
                return;
            }

            if (!movimientoIniciado)
            {
                // Iniciar movimiento
                movimientoIniciado = true;
                btnIniciarPausar.Text = "Pausar";
                círculo.EnMovimiento = true;
```

```
        // Reproducir sonido de lanzamiento
        if (sonidoLanzamiento != null)
        {
            try { sonidoLanzamiento.Play(); } catch { }
        }

        // Decrementar lanzamientos
        lanzamientosRestantes--;

        // Limpiar la trayectoria anterior
        puntosTrayectoria.Clear();

        // Actualizar etiquetas
        ActualizarEtiquetasJuego();
    }
    else
    {
        // Pausar movimiento
        movimientoIniciado = false;
        btnIniciarPausar.Text = "Continuar";
    }

    timer1.Enabled = movimientoIniciado;
}

private void btnReiniciar_Click(object sender, EventArgs e)
{
    ReiniciarJuego();
}

/// <summary>
/// Reinicia el juego a su estado inicial.
/// </summary>
private void ReiniciarJuego()
{
    // Detener el movimiento
    movimientoIniciado = false;
    btnIniciarPausar.Text = "Lanzar";
    timer1.Enabled = false;

    // Reiniciar la posición y velocidades del círculo
    círculo.ReiniciarPosición();
    círculo.ConfigurarLanzamiento(velocidadInicial, ánguloLanzamiento);
    círculo.EnMovimiento = false;
```

```
    // Limpiar la trayectoria
    puntosTrayectoria.Clear();

    // Reiniciar valores del juego
    puntuacion = 0;
    lanzamientosRestantes = 5;
    juegoTerminado = false;

    // Crear nuevos objetivos
    CrearObjetivos();

    // Actualizar etiquetas
    ActualizarEtiquetasJuego();

    // Redibujar el lienzo
    pbLienzo.Invalidate();
}

private void PbLienzo_MouseDown(object sender, MouseEventArgs e)
{
    if (!movimientoIniciado && !juegoTerminado)
    {
        arrastrando = true;
        puntoInicio = Point.Round(círculo.Centro); // Convert PointF to Point
        puntoFin = e.Location;

        // Calcular ángulo basado en la posición del mouse
        ActualizarAnguloDesdeRaton(e.X, e.Y);
        pbLienzo.Invalidate();
    }
}

private void PbLienzo_MouseMove(object sender, MouseEventArgs e)
{
    if (arrastrando && !movimientoIniciado && !juegoTerminado)
    {
        puntoFin = e.Location;

        // Calcular ángulo y velocidad basados en la posición del mouse
        ActualizarAnguloDesdeRaton(e.X, e.Y);
        pbLienzo.Invalidate();
    }
}
```

```
private void PbLienzo_MouseUp(object sender, MouseEventArgs e)
{
    if (arrastrando && !movimientoIniciado && !juegoTerminado)
    {
        arrastrando = false;
        puntoFin = e.Location;

        // Calcular ángulo final y lanzar
        ActualizarAnguloDesdeRaton(e.X, e.Y);

        // Iniciar el lanzamiento
        btnIniciarPausar_Click(this, EventArgs.Empty);
    }
}

/// <summary>
/// Actualiza el ángulo y velocidad basados en la posición del mouse.
/// </summary>
private void ActualizarAnguloDesdeRaton(int mouseX, int mouseY)
{
    // Calcular la diferencia en posición
    int dx = puntoInicio.X - mouseX;
    int dy = puntoInicio.Y - mouseY;

    // Evitar división por cero
    if (dx == 0) dx = 1;

    // Calcular ángulo en radianes y convertir a grados
    double angulo = Math.Atan2(dy, dx) * 180 / Math.PI;

    // Ajustar al rango 0-90 (primer cuadrante)
    angulo = (angulo < 0) ? 0 : (angulo > 90) ? 90 : angulo;

    // Calcular velocidad basada en la distancia (limitada por el máximo del trackbar)
    double distancia = Math.Sqrt(dx * dx + dy * dy);
    float velocidad = (float)Math.Min(distancia / 10, tbVelocidad.Maximum);

    // Actualizar valores
    ánguloLanzamiento = (float)angulo;
    velocidadInicial = velocidad;

    // Actualizar trackbars
    tbÁngulo.Value = (int)Math.Round(ánguloLanzamiento);
```

```
            tbVelocidad.Value = (int)Math.Round(velocidadInicial);

            // Actualizar círculo
            círculo.ConfigurarLanzamiento(velocidadInicial, ánguloLanzamiento);

            // Actualizar etiquetas
            ActualizarEtiquetas();
        }

        private void pbLienzo_Paint(object sender, PaintEventArgs e)
        {
            // Aplicar suavizado
            e.Graphics.SmoothingMode = System.Drawing.Drawing2D.SmoothingMode.AntiAlias;

            // Dibujar fondo
            if (imagenFondo != null)
            {
                e.Graphics.DrawImage(imagenFondo, 0, 0, pbLienzo.Width, pbLienzo.Height);
            }
            else
            {
                // Fondo por defecto
                e.Graphics.FillRectangle(Brushes.LightBlue, 0, 0, pbLienzo.Width, pbLienzo.Height);
            }

            // Dibujar el suelo
            int alturaSuelo = 5;
            e.Graphics.FillRectangle(Brushes.Green, 0, pbLienzo.Height - alturaSuelo,
pbLienzo.Width, alturaSuelo);

            // Dibujar la trayectoria si está habilitada
            if (mostrarTrayectoria && puntosTrayectoria.Count > 1)
            {
                for (int i = 1; i < puntosTrayectoria.Count; i++)
                {
                    e.Graphics.DrawLine(Pens.Gray, puntosTrayectoria[i - 1], puntosTrayectoria[i]);
                }
            }

            // Dibujar los objetivos
            foreach (var objetivo in objetivos)
            {
                objetivo.Dibujar(e.Graphics);
            }
```

```
        // Dibujar el círculo
        círculo.Dibujar(e.Graphics);

        // Dibujar línea de apuntado si se está arrastrando
        if (arrastrando && !movimientoIniciado)
        {
            e.Graphics.DrawLine(Pens.Red, puntoInicio, puntoFin);
        }

        // Dibujar información del juego
        string info = $"Velocidad: {velocidadInicial:F1} px/tick\nÁngulo: {ánguloLanzamiento}°\n"
+
                      $"Vx: {círculo.VelocidadX:F2}, Vy: {círculo.VelocidadY:F2}\n" +
                      $"Puntuación: {puntuacion}\nLanzamientos: {lanzamientosRestantes}";

        // Fondo para el texto
        SizeF tamaño = e.Graphics.MeasureString(info, this.Font);
        e.Graphics.FillRectangle(new SolidBrush(Color.FromArgb(180, Color.White)),
                                 10, 10, tamaño.Width + 10, tamaño.Height + 10);

        e.Graphics.DrawString(info, this.Font, Brushes.Black, 15, 15);

        // Mensaje de fin de juego
        if (juegoTerminado)
        {
            string mensaje = $"¡JUEGO TERMINADO!\nPuntuación final: {puntuacion}\nPresiona
Reiniciar para jugar de nuevo";

            // Fondo para el mensaje
            SizeF tamañoMensaje = e.Graphics.MeasureString(mensaje, new
Font(this.Font.FontFamily, 16, FontStyle.Bold));
            Rectangle rectMensaje = new Rectangle(
                (pbLienzo.Width - (int)tamañoMensaje.Width) / 2 - 20,
                (pbLienzo.Height - (int)tamañoMensaje.Height) / 2 - 20,
                (int)tamañoMensaje.Width + 40,
                (int)tamañoMensaje.Height + 40
            );

            e.Graphics.FillRectangle(new SolidBrush(Color.FromArgb(220, Color.Black)),
rectMensaje);
            e.Graphics.DrawRectangle(Pens.White, rectMensaje);

            e.Graphics.DrawString(mensaje,
```

```
                    new Font(this.Font.FontFamily, 16, FontStyle.Bold),
                    Brushes.White,
                    (pbLienzo.Width - tamañoMensaje.Width) / 2,
                    (pbLienzo.Height - tamañoMensaje.Height) / 2);
    }
}

private void timer1_Tick(object sender, EventArgs e)
{
    if (movimientoIniciado)
    {
        // Registrar la posición actual para la trayectoria
        puntosTrayectoria.Add(Point.Round(círculo.Centro));

        // Limitar el tamaño de la lista de trayectoria
        if (puntosTrayectoria.Count > 500)
            puntosTrayectoria.RemoveAt(0);

        // Mover el círculo según su velocidad
        círculo.Mover();

        // Verificar colisiones con objetivos
        VerificarColisionesConObjetivos();

        // Manejar los límites del lienzo
        círculo.ManejarBordes(pbLienzo.Width, pbLienzo.Height);

        // Verificar si el círculo se detuvo
        if (Math.Abs(círculo.VelocidadX) < 0.1f && Math.Abs(círculo.VelocidadY) < 0.1f &&
            círculo.Centro.Y >= pbLienzo.Height - círculo.Radio - 5)
        {
            movimientoIniciado = false;
            timer1.Enabled = false;
            btnIniciarPausar.Text = "Lanzar";

            // Verificar si se terminaron los lanzamientos
            if (lanzamientosRestantes <= 0)
            {
                juegoTerminado = true;
            }
        }

        // Invalidar el lienzo para redibujarlo
        pbLienzo.Invalidate();
```

```
    }
}


/// <summary>
/// Verifica si hay colisiones entre el círculo y los objetivos.
/// </summary>
private void VerificarColisionesConObjetivos()
{
    bool impacto = false;

    foreach (var objetivo in objetivos)
    {
        if (!objetivo.Impactado && objetivo.VerificarColision(círculo))
        {
            // Marcar el objetivo como impactado
            objetivo.Impactado = true;

            // Aumentar la puntuación
            puntuacion += objetivo.Puntos;

            // Reproducir sonido de impacto
            if (sonidoImpacto != null)
            {
                try { sonidoImpacto.Play(); } catch { }
            }

            // Actualizar etiquetas
            ActualizarEtiquetasJuego();

            impacto = true;
        }
    }

    // Verificar si todos los objetivos han sido impactados
    if (impacto && !objetivos.Exists(o => !o.Impactado))
    {
        // Nivel completado, crear nuevos objetivos
        lanzamientosRestantes += 3; // Bonificación de lanzamientos
        CrearObjetivos();
    }
}

private void tbVelocidad_ValueChanged(object sender, EventArgs e)
```

```
        {
            velocidadInicial = tbVelocidad.Value;
            círculo.VelocidadInicial = velocidadInicial;
            círculo.ActualizarVelocidadesDesdeÁngulo();
            ActualizarEtiquetas();
            pbLienzo.Invalidate();
        }

        private void tbÁngulo_ValueChanged(object sender, EventArgs e)
        {
            ánguloLanzamiento = tbÁngulo.Value;
            círculo.ÁnguloLanzamiento = ánguloLanzamiento;
            círculo.ActualizarVelocidadesDesdeÁngulo();
            ActualizarEtiquetas();
            pbLienzo.Invalidate();
        }

        private void cbMostrarTrayectoria_CheckedChanged(object sender, EventArgs e)
        {
            mostrarTrayectoria = cbMostrarTrayectoria.Checked;
            pbLienzo.Invalidate();
        }

        #endregion
    }
}
    public class Objetivo
    {
        public Point Posicion { get; set; }

        /// <summary>
        /// Tamaño del objetivo.
        /// </summary>
        public Size Tamaño { get; set; }

        /// <summary>
        /// Valor en puntos del objetivo.
        /// </summary>
        public int Puntos { get; set; }

        /// <summary>
        /// Indica si el objetivo ha sido impactado.
        /// </summary>
        public bool Impactado { get; set; }
```

```
        /// <summary>
        /// Imagen del objetivo.
        /// </summary>
        public Image Imagen { get; private set; }

        /// <summary>
        /// Inicializa una nueva instancia de la clase Objetivo.
        /// </summary>
        public Objetivo(Point posicion, Size tamaño, int puntos, string rutaImagen)
        {
            Posicion = posicion;
            Tamaño = tamaño;
            Puntos = puntos;
            Impactado = false;

            try
            {
                Imagen = Image.FromFile(rutaImagen);
            }
            catch
            {
                // Si no se puede cargar la imagen, usar un color sólido
                Imagen = null;
            }
        }

        /// <summary>
        /// Verifica si hay colisión con el círculo.
        /// </summary>
        public bool VerificarColision(Círculo círculo)
        {
            // Obtener el rectángulo que representa el objetivo
            Rectangle rectObjetivo = new Rectangle(Posicion, Tamaño);

            // Calcular la distancia entre el centro del círculo y el punto más cercano del rectángulo
            int closestX = Math.Max(rectObjetivo.Left, Math.Min((int)círculo.Centro.X,
rectObjetivo.Right));
            int closestY = Math.Max(rectObjetivo.Top, Math.Min((int)círculo.Centro.Y,
rectObjetivo.Bottom));

            // Calcular la distancia al cuadrado entre el centro del círculo y el punto más cercano
            int distanciaX = (int)círculo.Centro.X - closestX;
            int distanciaY = (int)círculo.Centro.Y - closestY;
```

```
            float distanciaCuadrado = (distanciaX * distanciaX) + (distanciaY * distanciaY);

            // Hay colisión si la distancia es menor que el radio al cuadrado
            return distanciaCuadrado <= (círculo.Radio * círculo.Radio);
        }

        /// <summary>
        /// Dibuja el objetivo en el lienzo.
        /// </summary>
        public void Dibujar(Graphics g)
        {
            if (!Impactado)
            {
                if (Imagen != null)
                {
                    g.DrawImage(Imagen, new Rectangle(Posicion, Tamaño));
                }
                else
                {
                    // Dibujar un rectángulo como respaldo
                    g.FillRectangle(Brushes.Orange, new Rectangle(Posicion, Tamaño));
                    g.DrawRectangle(Pens.Red, new Rectangle(Posicion, Tamaño));
                }
            }
        }
    }
}
```