Marc Garcia Ferrer ID: u1973250 Grupo 2
Nicolas Daniel Rueda Araque ID: u1973233 Grupo 1

Práctica del juego de la gravedad

*Hemos decidido no usar una tabla para poner los cambios ya que teníamos muchísimos y creemos que se ve mucho mas claro y entendedor usando fotos y explicaciones tal y como lo hemos hecho.

Traducciones

- Planet por Planeta
- Velocity box por velocidad de la caja
- Results por resultados
- Start por Comenzar
- Drop por Soltar
- Reset por Reiniciar
- Earth por Tierra
- Moon por Luna
- Gravity Game por Juego de gravedad
- Earth por Tierra
- Moon por Luna
- You win por Has ganado (no existe en el código final, se sustituye por un sistema de puntos)
- Try again por Vuelve a intentarlo (no existe en el código final, se sustituye por un sistema de puntos)

Variables añadidas

int para modificar con una línea de código todas donde estas aparezcan.

```
private int boxWidth;
private int boxWidthAux;
private int textBoxWeidth, textBoxLeft;
```

La velocidad de la pelota:

```
private double ballVelocity;
```

Creamos un contador:

```
private int puntuacion;
```

Creamos un bool sentido que será igual a -1 o 1. Multiplicando la velocidad por este valor la caja se moverá hacia los dos lados.

```
private double sentido;
```

Un bool para indicar que el juego está o no pausado:

```
private bool pausado;
```

Un bool para indicar que la pelota está subiendo o bajando:

```
private bool subiendo;
```

Estandarizamos los tamaños de los botones y la ventana:

```
private int buttonWidth = 100;
private int buttonHeight = 30;
private int buttonTop = 100;
private int windowHeight =720;
private int windowWidth = 1280;
```

Guardamos las posiciones de nuestros dos “objetos”:

```
private double ballLocationInicial;
private double boxLocationInitial;
```

Creamos 3 nuevas posiciones para 3 nuevas cajas. Con estas simularemos que la caja salga por un lado y entre por el otro:

```
private double boxLocationLeft1, boxLocationLeft2, boxLocationRight;
```

El resto de variables que usaremos para las físicas:

```
private double boxAcceleration;
private double boxVelocity;
private double boxVelocityAux;
private double maxBoxVelocity;
private double maxBoxVelocityAux;
private double boxInitialVelocity;
private double boxLocation;
```

El sentido de la bola, su aceleración y un double que igualamos a 1 o -1 según el bool subiendo:

```
private double ballSentido
private double ballAcceleration;
private double subiendoValor
```

Modificación de la interfície gráfica

Cambios a la ventana del programa:

La hacemos más grande a resolución 720p:

```
private int windowHeight =720;
private int windowWidth =  1280;
```

Le cambiamos el color de fondo:

```
//Tamaño i título de la ventana
this.Height = windowHeight;
this.Width = windowWidth;
this.Text = "Juego de gravedad";
this.BackColor = Color.Lavender;
```

Caja aceleración

Creamos la etiqueta acceleration (la aceleración)

```
public class GravityGame : Form
{
    //etiqueta planeta velocidad y resultados
    private Label planetLabel;
    private Label velocityInitialLabel;
    private Label resultsLabel;
    private Label accelerationLabel;
    private Label boxActualVelocityLabel;
    private Label boxMaxVelocityLabel;

    //texto del resultado
    private TextBox resultsTextBox;
    //texto de la velocidad
    private TextBox initialVelocityTextBox;
    private TextBox accelerationTextBox;
    private TextBox actualVelocityTextBox;
    private TextBox maxBoxVelocityTextBox;
```

```
accelerationLabel = new Label();
accelerationLabel.Text = "Aceleracion (m/s^2)";
accelerationLabel.Font = new Font(accelerationLabel.Font, FontStyle.Bold);
accelerationLabel.Top = 600;
accelerationLabel.Left = 10;
accelerationLabel.Width = 75;
```

Su caja de texto:

```
//aceleracion
accelerationTextBox = new TextBox();
accelerationTextBox.Width = textBoxWeidth;
accelerationTextBox.Text = String.Format("{0}", 5.0m);
accelerationTextBox.AutoSize = true;
accelerationTextBox.Top = accelerationLabel.Top;
accelerationTextBox.Left = textBoxLeft;
```

Lo añadimos a la ventana:

```
//añadimos elementos a la GUI
this.Controls.Add(planetLabel);
this.Controls.Add(velocityInitialLabel);
this.Controls.Add(resultsLabel);
this.Controls.Add(initialVelocityTextBox);
this.Controls.Add(resultsTextBox);
this.Controls.Add(startButton);
this.Controls.Add(dropButton);
this.Controls.Add(resetButton);
this.Controls.Add(pauseButton);
this.Controls.Add(planetComboBox);
this.Controls.Add(drawingPanel);
this.Controls.Add(accelerationLabel);
this.Controls.Add(accelerationTextBox);
this.Controls.Add(boxActualVelocityLabel);
this.Controls.Add(actualVelocityTextBox);
this.Controls.Add(leftButton);
this.Controls.Add(rightButton);
this.Controls.Add(boxMaxVelocityLabel);
this.Controls.Add(maxBoxVelocityTextBox);
```

Creamos boxActualVelocity (velocidad actual de la caja) para mostrar a tiempo real a qué velocidad se mueve la caja. Seguimos los mismos pasos que antes:

```
//etiqueta planeta velocidad y resultados
private Label planetLabel;
private Label velocityInitialLabel;
private Label resultsLabel;
private Label accelerationLabel;
private Label boxActualVelocityLabel;
private Label boxMaxVelocityLabel;

//texto del resultado
private TextBox resultsTextBox;
//texto de la velocidad
private TextBox initialVelocityTextBox;
private TextBox accelerationTextBox;
private TextBox boxActualVelocityTextBox;
private TextBox maxBoxVelocityTextBox;
```

```
boxActualVelocityLabel = new Label();
boxActualVelocityLabel.Text = "Actual Box Velocity (m/s)";
boxActualVelocityLabel.Font = new Font(boxActualVelocityLabel.Font, FontStyle.Bold);
boxActualVelocityLabel.Top = 650;
boxActualVelocityLabel.Left = 10;
boxActualVelocityLabel.Width = 75;
```

```
actualVelocityTextBox = new TextBox();
actualVelocityTextBox.Width = textBoxWeidth;
actualVelocityTextBox.Text = "";
actualVelocityTextBox.AutoSize = true;
actualVelocityTextBox.Top = boxActualVelocityLabel.Top;
actualVelocityTextBox.Left = textBoxLeft;
```

```
//añadimos elementos a la GUI
this.Controls.Add(planetLabel);
this.Controls.Add(velocityInitialLabel);
this.Controls.Add(resultsLabel);
this.Controls.Add(initialVelocityTextBox);
this.Controls.Add(resultsTextBox);
this.Controls.Add(startButton);
this.Controls.Add(dropButton);
this.Controls.Add(resetButton);
this.Controls.Add(pauseButton);
this.Controls.Add(planetComboBox);
this.Controls.Add(drawingPanel);
this.Controls.Add(accelerationLabel);
this.Controls.Add(accelerationTextBox);
this.Controls.Add(boxActualVelocityLabel);
this.Controls.Add(actualVelocityTextBox);
this.Controls.Add(leftButton);
this.Controls.Add(rightButton);
this.Controls.Add(boxMaxVelocityLabel);
this.Controls.Add(maxBoxVelocityTextBox);
```

En la función ActionPerformance actualizamos el contenido de la caja cada vez que esta se ejecuta:

```
//mostramos por pantalla la velocidad actual
actualVelocityTextBox.Text = Convert.ToString(boxVelocity);
```

Reseteamos la velocidad al resetear el programa:

```
public void ResetButtonClicked(object source, EventArgs e)
{
    puntuacion = 0;
    time = 0.0;
    dropTime = 0.0;
    boxVelocity=0;
    subiendo=false;
    gameStarted = false;
    dropped = false;

    ballLocation = rd.Next(0,500);
    pausado = true;
    ballAltitude = initialAltitude;
    resultsTextBox.Text = "";
    actualVelocityTextBox.Text = Convert.ToString("0");
    //paramos el timer

    gameTimer.Stop();
    boxLocation = 0.0;

    boxWidth=boxWidthAux;

    //actualizamos la GUI
    UpdateDisplay();
}
```

Creamos boxMaxVelocity para limitar la velocidad de la caja. Si no, la velocidad sigue creciendo infinitamente y llega un momento en el que la caja está “en todas partes’ al salir y por un lado y entrar por el otro:

```
public class GravityGame : Form
{
    //etiqueta planeta velocidad y resultados
    private Label planetLabel;
    private Label velocityInitialLabel;
    private Label resultsLabel;
    private Label accelerationLabel;
    private Label boxActualVelocityLabel;
    private Label boxMaxVelocityLabel;

    //texto del resultado
    private TextBox resultsTextBox;
    //texto de la velocidad
    private TextBox initialVelocityTextBox;
    private TextBox accelerationTextBox;
    private TextBox ActualVelocityTextBox;
    private TextBox maxBoxVelocityTextBox;
```

```
    private double boxVelocity;
    private double boxVelocityAux;
    private double maxBoxVelocity;
    private double maxBoxVelocityAux;
    private double boxInitialVelocity;
    private double boxLocation;
```

```
boxMaxVelocityLabel = new Label();
boxMaxVelocityLabel.Text = "Max box velocity (m/s)";
boxMaxVelocityLabel.Font = new Font(boxMaxVelocityLabel.Font, FontStyle.Bold);
boxMaxVelocityLabel.Top = 450;
boxMaxVelocityLabel.Left = 10;

maxBoxVelocityTextBox = new TextBox();
maxBoxVelocityTextBox.Width = textBoxWeidth;
maxBoxVelocityTextBox.Text = String.Format("{0}", 15.0m);
maxBoxVelocityTextBox.AutoSize = true;
maxBoxVelocityTextBox.Top = boxMaxVelocityLabel.Top;
maxBoxVelocityTextBox.Left = textBoxLeft;

//añadimos elementos a la GUI
this.Controls.Add(planetLabel);
this.Controls.Add(velocityInitialLabel);
this.Controls.Add(resultsLabel);
this.Controls.Add(initialVelocityTextBox);
this.Controls.Add(resultsTextBox);
this.Controls.Add(startButton);
this.Controls.Add(dropButton);
this.Controls.Add(resetButton);
this.Controls.Add(pauseButton);
this.Controls.Add(planetComboBox);
this.Controls.Add(drawingPanel);
this.Controls.Add(accelerationLabel);
this.Controls.Add(accelerationTextBox);
this.Controls.Add(boxActualVelocityLabel);
this.Controls.Add(actualVelocityTextBox);
this.Controls.Add(leftButton);
this.Controls.Add(rightButton);
this.Controls.Add(boxMaxVelocityLabel);
this.Controls.Add(maxBoxVelocityTextBox);
```

Botón de pausa

Creamos un nuevo botón de pausa:

```
//botones
private Button startButton;
private Button dropButton;
private Button resetButton;
private Button pauseButton;
private Button leftButton;
private Button rightButton;
```

```
//boton pause
pauseButton = new Button();
pauseButton.Text = "Pause";
pauseButton.Height = buttonHeight;
pauseButton.Width = buttonWidth;
pauseButton.BackColor = Color.Crimson;
pauseButton.Top = buttonTop;
pauseButton.Left = resetButton.Left + pauseButton.Width + 10;
pauseButton.Click += new EventHandler(PauseButtonClicked);
```

```
//añadimos elementos a la GUI
this.Controls.Add(planetLabel);
this.Controls.Add(velocityInitialLabel);
this.Controls.Add(resultsLabel);
this.Controls.Add(initialVelocityTextBox);
this.Controls.Add(resultsTextBox);
this.Controls.Add(startButton);
this.Controls.Add(dropButton);
this.Controls.Add(resetButton);
this.Controls.Add(pauseButton);
this.Controls.Add(planetComboBox);
this.Controls.Add(drawingPanel);
this.Controls.Add(accelerationLabel);
this.Controls.Add(accelerationTextBox);
this.Controls.Add(boxActualVelocityLabel);
this.Controls.Add(actualVelocityTextBox);
this.Controls.Add(leftButton);
this.Controls.Add(rightButton);
this.Controls.Add(boxMaxVelocityLabel);
this.Controls.Add(maxBoxVelocityTextBox);
```

Y creamos el código necesario para que funcione el botón. En caso de que el juego haya comenzado y no esté pausado, detenemos el contador del juego y en caso contrario hacemos que este comience de nuevo:

```
//funcion del boton pause
public void PauseButtonClicked(object source, EventArgs e)
{
    if(gameStarted){
        pausado = !pausado;
        if(pausado){
            gameTimer.Stop();
        }
        else gameTimer.Start();
    }
}
```

Botones de izquierda y derecha

Creamos 2 botones nuevos, el de izquierda y derecha (left y right), que moverán la plataforma hacia los lados cuando sean pulsados modificando el valor de la variable sentido como ya hemos explicado antes:

```
//botones
private Button startButton;
private Button dropButton;
private Button resetButton;
private Button pauseButton;
private Button leftButton;
private Button rightButton;
```

```
//left button
leftButton = new Button();
leftButton.Text = "<";
leftButton.Height = buttonHeight;
leftButton.Width = buttonWidth;
leftButton.BackColor = Color.LightCoral;
leftButton.Top = velocityInitialLabel.Top;
leftButton.Left = (resetButton.Left + pauseButton.Width + 10)/2;
leftButton.Click += new EventHandler(ButtonLeftClicked);

//right button
rightButton = new Button();
rightButton.Text = ">";
rightButton.Height = buttonHeight;
rightButton.Width = buttonWidth;
rightButton.BackColor = Color.LightCoral;
rightButton.Top = velocityInitialLabel.Top;
rightButton.Left = (resetButton.Left + pauseButton.Width + 10);
rightButton.Click += new EventHandler(ButtonRightClicked);
```

```
//añadimos elementos a la GUI
this.Controls.Add(planetLabel);
this.Controls.Add(velocityInitialLabel);
this.Controls.Add(resultsLabel);
this.Controls.Add(initialVelocityTextBox);
this.Controls.Add(resultsTextBox);
this.Controls.Add(startButton);
this.Controls.Add(dropButton);
this.Controls.Add(resetButton);
this.Controls.Add(pauseButton);
this.Controls.Add(planetComboBox);
this.Controls.Add(drawingPanel);
this.Controls.Add(accelerationLabel);
this.Controls.Add(accelerationTextBox);
this.Controls.Add(boxActualVelocityLabel);
this.Controls.Add(actualVelocityTextBox);
this.Controls.Add(leftButton);
this.Controls.Add(rightButton);
this.Controls.Add(boxMaxVelocityLabel);
this.Controls.Add(maxBoxVelocityTextBox);
```

Ambos botones usan la variable sentido. Esta se multiplicará por la velocidad para definir en qué dirección se moverá nuestra caja:

```
public void ButtonLeftClicked(object source, EventArgs e){
    if(sentido>0){
    //velocidadpositiva=false;
    sentido=-1;

    }
}

public void ButtonRightClicked(object source, EventArgs e){
    if(sentido<0){
    //velocidadpositiva=true;
    sentido=1;

    }

}
```

Añadimos nuevos planetas

Creamos nuevos planetas con distintas gravedades y colores, añadiendolos a la ComboBox:

```
// Combobox
planetComboBox = new ComboBox();
planetComboBox.Items.Add("Tierra");
planetComboBox.Items.Add("Luna");
planetComboBox.Items.Add("Jupiter");
planetComboBox.Items.Add("Namek");
planetComboBox.Items.Add("Tatooine");
planetComboBox.Items.Add("Darwin IV");
planetComboBox.SelectedIndex = 0;
planetComboBox.Left = 80;
planetComboBox.Top = planetLabel.Top;
```

Cada planeta tiene una gravedad asignada:

```
//determinar que planeta hemos cogido y heredar su gravedad
string selectedItem = (string)planetComboBox.SelectedItem;
if (String.Equals(selectedItem, "Tierra"))
{
    g = 9.81;
}
else if (String.Equals(selectedItem, "Luna"))
{
    g = 1.624;
}
else if(String.Equals(selectedItem, "Jupiter"))
{
    g = 24.8;  //  Jupiter
}
else if(String.Equals(selectedItem, "Namek"))
{
    g = 83.2;
}
else if(String.Equals(selectedItem, "Tatooine"))
{
    g = 7.6;
}
else
{
    g = 0.6;  //  Darwin IV
}

resultsTextBox.Text = "";
```

Le añadimos un color de fondo diferente a cada planeta

```
Graphics g = drawingPanel.CreateGraphics();
int width = drawingPanel.Width - 1;
int height = drawingPanel.Height - 1;
string selectedItem = (string)planetComboBox.SelectedItem;
if (String.Equals(selectedItem, "Tierra"))
{
    drawingPanel.BackColor = Color.Red;
}
else if (String.Equals(selectedItem, "Luna"))
{
    drawingPanel.BackColor = Color.Blue;
}
else if(String.Equals(selectedItem, "Jupiter"))
{
    drawingPanel.BackColor = Color.Yellow;
}
else if(String.Equals(selectedItem, "Namek"))
{
    drawingPanel.BackColor = Color.Orange;
}
else if(String.Equals(selectedItem, "Tatooine"))
{
    drawingPanel.BackColor = Color.Gray;
}
else
{
    drawingPanel.BackColor = Color.Black;
}
```

Panel de dibujo

Modificamos el espacio de dibujo para que se ajuste mejor a los nuevos cambios:

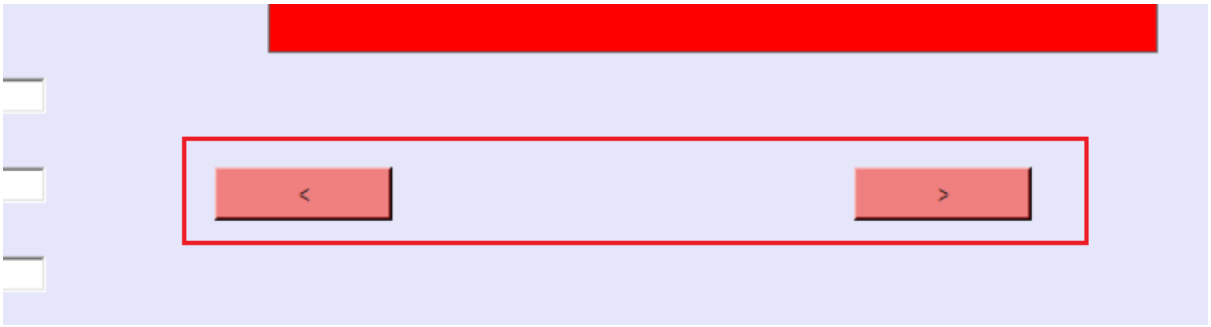
```
//espacio donde se imprime la caja y el circulo
drawingPanel = new Panel();
drawingPanel.Width = 501;
drawingPanel.Height = 251;
drawingPanel.Left = windowWidth/2 - drawingPanel.Width/2;
drawingPanel.Top = windowHeight/2 - drawingPanel.Height/2 - 50;
drawingPanel.BorderStyle = BorderStyle.FixedSingle;
```

Finalmente, modificamos ligeramente los parámetros de inicialización de la caja y la pelota para que no genere errores al pulsar el botón Start:

```
//Parametros iniciales de la caja y la pelota
boxLocation = 0.0;
boxWidth = 100;
boxWidthAux = boxWidth;
initialAltitude = 120.0;
ballAltitude = initialAltitude;
ballLocation = 210;
time = 0.0;
dropTime = 0.0;
dropped = false;
```

Dificultad

Para modificar la dificultad del juego hemos añadido dos botones para hacer que la caja se mueva hacia la izquierda o derecha como ya hemos visto anteriormente:



Al caer la pelota encima de la caja ésta rebotará hacia el cielo y volverá a generarse una nueva. El tamaño de la caja disminuirá con cada rebote:

```
//la pelota toca la plataforma
if (!subiendo&&
    ((ballAltitude <= 15.0&&ballAltitude>=5) &&
            (
            (ballLocation > boxLocation && ballLocation < boxLocation+boxWidth)||
            (ballLocation > boxLocationLeft2 && ballLocation < boxLocationLeft2+boxWidth)||
            (ballLocation > boxLocationLeft1 && ballLocation < boxLocationLeft1+boxWidth)||
            (ballLocation > boxLocationRight && ballLocation < boxLocationRight+boxWidth)
            )
        )
    )
    {
    ballSentido=-sentido;
    boxWidth-=boxWidth/8;
    subiendo=true;
    subiendoValor=1;
    //dropTime = 2;
    puntuacion++;

    resultsTextBox.Text = "Puntuacion: " + puntuacion;
}
```

Análisis y modificación de las físicas del programa

Hemos decidido suprimir el elemento *time ya que la misma función Action Performance simula este. Al hacer esto tenemos libertad para modificar la localización de la caja. Por ejemplo si queremos que la caja vuelva a su posición inicial será tan fácil como boxLocation = 0. En cambio si dentro de la fórmula boxLocation tenemos un *time la única forma sería indicar time=0. Práctica incoherente e ineficiente a no ser que usemos diferentes contadores para los diferentes aspectos del juego.

Ya que esta función se ejecuta cada 0.05 segundos decidimos dividir la velocidad entre 20 (las veces que se ejecutaría en un segundo la función) para adaptar nuestra unidad de metros/s a nuestra función.

Aceleración de la caja

- Creamos un contador que se reinicia al llegar a 20 (las veces que se ejecuta la función en un segundo). Si el contador es igual a 0 la velocidad aumenta según la aceleración (V=Vinicial+a*t), guardamos la velocidad en una variable auxiliar, dividimos la velocidad entre las veces que se ejecuta la función en un segundo y a la localización de la caja le sumamos la velocidad (o se lo restamos, según el sentido de la caja). Cuando el contador sea menor de 20 la localización de la caja será igual a la posición actual + la velocidad.

```
else{
    if(contador<20){
        boxLocation += boxVelocityAux*(sentido);
        contador++;
    }
    //Debug.Write(contador + " ");

    if(contador==20){

        //entre 20 porque esta funcion se ejecuta 20 veces en un segundo
        contador=0;
    }

    if(contador==0){
        boxVelocity=boxInitialVelocity+boxAcceleration*time;
        boxVelocityAux=boxVelocity;
        boxVelocity=boxVelocity/20;
    }
}
```

Hacemos que la caja salga por un lado y entre por el otro

- Si la caja que está a la izquierda de la original sale por la derecha y entre el ultimo pixel de la ventana y su posición hay una diferencia del tamaño de la ventana menos el ancho de la caja la posición de la caja volverá a su posición inicial menos el ancho de la caja
- si la caja original sale por la izquierda con una diferencia del primer pixel a su posición al ancho de la caja la caja volverá a su posición original menos el ancho de la caja.

```
boxLocationLeft2=boxLocation-drawingPanel.Width;
boxLocationLeft1=boxLocation-(drawingPanel.Width)*2;
boxLocationRight=boxLocation+drawingPanel.Width;
```

```
//la caja sale por un lado y aparece por el otro
if(boxLocationLeft2>=((drawingPanel.Width)*2)-boxWidth){

    boxLocation=boxLocationInitial-boxWidth;
}
if(boxLocation<=(-boxWidth)){

    boxLocation=boxLocationInitial-boxWidth;
}


if(boxVelocity>=maxBoxVelocity/20){
    boxVelocity=maxBoxVelocity;
    boxLocation += boxVelocity*(sentido);

}
```

Resto de cambios

Cambiamos el funcionamiento de StartButtonClicked:

Aplicamos la conversión indicada anteriormente a la aceleración de la pelota. Indicamos el sentido en el que se moverá horizontalmente la pelota a 0 (sin movimiento). Indicamos la localización inicial de la caja en primer pixel fuera de la pantalla (se hace para que salga por un lado y entre por el otro como ya hemos visto). Nuestras variables “subiendoValor” y “subiendo” nos indica si la pelota está subiendo o bajando. Elegimos que la pelota se genere en un sitio aleatorio y las variables velocidad inicial, maxima velocidad y aceleración se igualan al contenido de su correspondiente TextBox.

```
public void StartButtonClicked(object source, EventArgs e)
{
    ballAcceleration = g/20;
    ballSentido=0;
    contador =20;
    sentido =1;
    ballLocationInicial = ballLocation;

    boxLocationInitial = drawingPanel.Width;
    boxLocation=boxLocationInitial;
    //maxBoxVelocityLocation=0;
    //locationAux=0;

    subiendoValor=-1;
    subiendo=false;
    gameStarted = true;
    dropped = false;
    gameStarted = true;

    pausado = false;


    time = 0.0;
    dropTime = 0.0;
    puntuacion = 0;

    ballLocation = rd.Next(0,500);
    ballAltitude = initialAltitude;


boxInitialVelocity = Convert.ToDouble(initialVelocityTextBox.Text);
boxVelocityAux=boxInitialVelocity;
maxBoxVelocity = Convert.ToDouble(maxBoxVelocityTextBox.Text);
maxBoxVelocityAux = maxBoxVelocity;
boxAcceleration = Convert.ToDouble(accelerationTextBox.Text);
ballVelocity=10;
```

Detectamos que planeta ha sido seleccionado para conseguir la gravedad de este:

```
//determinar que planeta hemos cogido y heredar su gravedad
string selectedItem = (string)planetComboBox.SelectedItem;
if (String.Equals(selectedItem, "Tierra"))
{
    g = 9.81;
}
else if (String.Equals(selectedItem, "Luna"))
{
    g = 1.624;
}
else if(String.Equals(selectedItem, "Jupiter"))
{
    g = 24.8;  //  Jupiter
}
else if(String.Equals(selectedItem, "Namek"))
{
    g = 83.2;
}
else if(String.Equals(selectedItem, "Tatooine"))
{
    g = 7.6;
}
else
{
    g = 0.6;  //  Darwin IV
}

resultsTextBox.Text = "";
```

Y finalmente iniciamos el timer:

```
    gameTimer.Start();
}
```

Cambiamos la función ResetButtonClicked

Reiniciamos nuestras variables puntuación, time, droptime, boxvelocity, subiendo, gamestarted y dropped. Además la bola se vuelve a colocar en una posición aleatoria, vuelve a su latitud inicial. La caja también se recoloca en la posición 0 y su ancho vuelve a ser el inicial.

```
//botón para resetear
public void ResetButtonClicked(object source, EventArgs e)
{
    puntuacion = 0;
    time = 0.0;
    dropTime = 0.0;
    boxVelocity=0;
    subiendo=false;
    gameStarted = false;
    dropped = false;

    ballLocation = rd.Next(0,500);
    pausado = true;
    ballAltitude = initialAltitude;
    resultsTextBox.Text = "";
    actualVelocityTextBox.Text = Convert.ToString("0");
    //paramos el timer

    gameTimer.Stop();
    boxLocation = 0.0;

    boxWidth=boxWidthAux;

    //actualizamos la GUI
    UpdateDisplay();
}
```

Actualizamos el UpdateDisplay con el color de los nuevos planetas como ya hemos visto antes:

```
//valores de lo imprimido en la GUI
private void UpdateDisplay()
{


    Graphics g = drawingPanel.CreateGraphics();
    int width = drawingPanel.Width - 1;
    int height = drawingPanel.Height - 1;
    string selectedItem = (string)planetComboBox.SelectedItem;
    if (String.Equals(selectedItem, "Tierra"))
    {
        drawingPanel.BackColor = Color.Red;
    }
    else if (String.Equals(selectedItem, "Luna"))
    {
        drawingPanel.BackColor = Color.Blue;
    }
    else if(String.Equals(selectedItem, "Jupiter"))
    {
        drawingPanel.BackColor = Color.Yellow;
    }
    else if(String.Equals(selectedItem, "Namek"))
    {
        drawingPanel.BackColor = Color.Orange;
    }
    else if(String.Equals(selectedItem, "Tatooine"))
    {
        drawingPanel.BackColor = Color.Gray;
    }
    else
    {
        drawingPanel.BackColor = Color.Black;
    }
```

Cambiamos los colores de la caja y de la pelota

```
        g.Clear(drawingPanel.BackColor);

        Pen blackPen = new Pen(Color.AliceBlue, 2);
        g.DrawLine(blackPen, 0, 135, width, 135);

        //  Update the position of the box and
        //  ball on the screen.
        SolidBrush brush = new SolidBrush(Color.White);
        g.FillRectangle(brush, (int)boxLocation, 120, boxWidth, 15);
        g.FillRectangle(brush, (int)boxLocationLeft1, 120, boxWidth, 15);
        g.FillRectangle(brush, (int)boxLocationLeft2, 120, boxWidth, 15);
        g.FillRectangle(brush, (int)boxLocationRight, 120, boxWidth, 15);

        int zPosition = (int)(initialAltitude - ballAltitude);
        g.FillEllipse(brush, (int)ballLocation, zPosition, 15, 15);

        //  Clean up the Graphics object.
        g.Dispose();
    }
```

A continuación modificamos ActionPerformed:

- Si la pelota toca las paredes de la pantalla (ballLocation<o||ballLocation>drawingPanel.WIdth) “rebota” cambiando su sentido.
- Creamos 3 cajas nuevas : 2 a la izquierda y una a la derecha para simular que cuando la caja salga por un lado entre por el otro de manera fluida.
- Si la pelota ha sido soltada se activa el contador del tiempo de la pelota. Se podría eliminar ya que no lo usamos en nuestro código, pero lo mantenemos porque es útil para futuras implementaciones.
- Si la pelota ha sido soltada y su velocidad es menor que 10 su velocidad será igual a la aceleración*tiempo. Anteriormente ya hemos igualado la aceleración a la gravedad/20
- La localización horizontal de la pelota es igual a la localización actual + su velocidad.
- La altitud de la pelota es igual a su velocidad * subiendoValor (-1 o +1 según si está subiendo o bajando)

```
//esta funcion se llama cada 0.05 segundos
public void ActionPerformed(object source, EventArgs e)
{
    Debug.Write(g);


    if(ballLocation<0||ballLocation>=drawingPanel.Width){
        ballSentido=-ballSentido;
    }


    boxLocationLeft2=boxLocation-drawingPanel.Width;
    boxLocationLeft1=boxLocation-(drawingPanel.Width)*2;
    boxLocationRight=boxLocation+drawingPanel.Width;


    //el temporizador comienza a contar al principio del juego, no necesetia mas condiciones
    //cada vez que se ejcuta la funcion se suma 1 al contad
    timeIncrement = 0.05;
    time += timeIncrement;


    //el tiempo que lleva cayendo la bola solo se inicializa cuando le damos al boton de soltar
    if(dropped) {
        dropTime += timeIncrement;
        if(ballVelocity<=10){
            ballVelocity+=ballAcceleration*time;
        }
        ballLocation += ballSentido*ballVelocity;
        ballAltitude += ballVelocity*(subiendoValor);
    }
```

- Si la velocidad de la caja es igual o sobrepasa la velocidad máxima la velocidad será igual a la velocidad máxima

Mostramos la velocidad actual de manera visual.

```
//mostramos por pantalla la velocidad actual
actualVelocityTextBox.Text = Convert.ToString(boxVelocity);
```

Y finalmente pasamos al movimiento de la pelota:

Si la altitud de la pelota es menor que 15 (la altura de la caja) pero mayor que 5 (para que no haya tanto margen de error tocando la esquina de la caja) y la bola se sitúa entre una de las 4 cajas la bola rebota, el tamaño de la caja disminuye y añadimos 1 a la puntuación y actualizamos el resultado. Si no, si la bola toca el suelo el juego acaba.

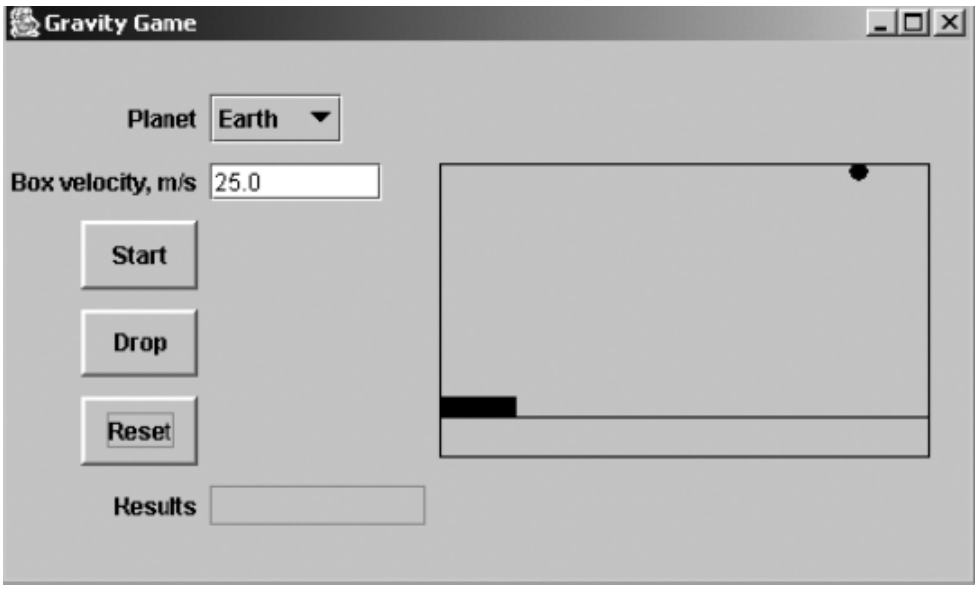
Por otro lado si la bola está subiendo y la altitud es mayor que 120 la pelota vuelve a bajar. Al perder mostramos por pantalla las veces que ha rebotado la pelota.

```
        //**********MOVIMIENTO DE LA PELOTA**********

        //la pelota toca la plataforma
        if (!subiendo&&
            ((ballAltitude <= 15.0&&ballAltitude>=5) &&
                    (
                    (ballLocation > boxLocation && ballLocation < boxLocation+boxWidth)||
                    (ballLocation > boxLocationLeft2 && ballLocation < boxLocationLeft2+boxWidth)||
                    (ballLocation > boxLocationLeft1 && ballLocation < boxLocationLeft1+boxWidth)||
                    (ballLocation > boxLocationRight && ballLocation < boxLocationRight+boxWidth)
                    )
                )
            )
            {
            ballSentido=-sentido;
            boxWidth-=boxWidth/8;
            subiendo=true;
            subiendoValor=1;
            //dropTime = 2;
            puntuacion++;

            resultsTextBox.Text = "Puntuacion: " + puntuacion;
        }


        //si la bola toca el suelo
        else if (ballAltitude <= 0.0)
        {
            //paramos el juego
            EndGame();

        }
```

```
        //la boca toca el techo
        if(subiendo&&(ballAltitude>120)) {
            subiendo=false;
            subiendoValor=-1;
        }

        UpdateDisplay();
    }

    void EndGame(){
        gameStarted = false;
            gameTimer.Stop();
            resultsTextBox.Text = "Ups! Puntos: " + puntuacion;
    }

    static void Main()
    {
        //iniciar la aplicación
        Application.Run(new GravityGame());
    }
}
```

Interfaz final del programa

La interfaz inicial del programa es:



Y el resultado final es: