Marc Garcia Ferrer ID: u1973250
Nicolas Daniel Rueda Araque ID: u1973233

Práctica del saco

El objetivo de la práctica es estudiar la física del lanzamiento de objetos y ubicar convenientemente las ecuaciones en un juego. También se proponen distintos cambios a la interfaz del juego.

Cambios a la interfície gráfica:

Cambio 1 Lo primero que hacemos es crear dos variables que contengan el tamaño de la ventana.	int windowdHeight = 720; int windowWith = 1280;

Cambio 2
Hacemos que el tamaño del DrawingPanel sea la mitad de la ventana y también actualizamos su posición.
	drawingPanel = new Panel();
	drawingPanel.Width = windowWith/2;
	drawingPanel.Height = windowdHeight/2;
	drawingPanel.Left = drawingPanel.Width/2;
	drawingPanel.Top = drawingPanel.Height/2;
	drawingPanel.BorderStyle = BorderStyle.FixedSingle;
	
	drawingPanel = new Panel();
	drawingPanel.Width = windowWith/2;
	drawingPanel.Height = windowdHeight/2;
	drawingPanel.Left = drawingPanel.Width/2;
	drawingPanel.Top = drawingPanel.Height/2;
	drawingPanel.BorderStyle = BorderStyle.FixedSingle;

Cambio 3 Le cambiamos el nombre al juego.	this.Text = "Basket Game";

Cambio 4
Cambiamos el color de los botones “fire” y “reset”.
	fireButton = new Button();
	fireButton.Text = "Fire";
	fireButton.Height = buttonHeight;
	fireButton.Width = buttonWidth;
	fireButton.Top = alphaLabel.Top+separation;
	fireButton.Left = buttonLeft;
	fireButton.BackColor = Color.Yellow;
	fireButton.Click += new EventHandler(FireButtonClicked);
	
	resetButton = new Button();
	resetButton.Text = "Reset";
	resetButton.Height = buttonHeight;
	resetButton.Width = buttonWidth;
	resetButton.Top = fireButton.Top+separation;
	resetButton.Left = buttonLeft;

	resetButton.BackColor = Color.Orange; resetButton.Click += new EventHandler(ResetButtonClicked);
Cambio 5 Cambiamos el color de la ventana.	this.BackColor = Color.Lavender;

Cambio 11
Creamos una variable para guardar la imagen del fondo
Cambiamos la posición de la línea horizonte (sustituimos todo lo del Update Display que contenga su valor).
Cambiamos la imagen al inicio, al pulsar lanzar y al resetear.
	//imagen de fondo
	private Image wallpaper;
	
	//cambiamos la imagen
	wallpaper = Image.FromFile("tierra.png")
	g.DrawImage(wallpaper, 0, zLocation, drawingPanel.Width, drawingPanel.Height+40);
	
	//cambiamos la línea del horizonte
	float horizontLine;
	UpdateDisplay{
	horizontLine = drawingPanel.Height/2;

Cambio 12
Cambiamos las imágenes de playerIcon.
	public void FireButtonClicked(object source, EventArgs e)
	{
	
	playerIcon = Image.FromFile("tiuquetiraelsac2.png");
	
	public void ResetButtonClicked(object source, EventArgs e)
	{
	
	playerIcon = Image.FromFile("tiuquetiraelsac.png");

Cambios a las físicas:

Cambio 4 Modificamos la variable gameTimer.Interval para que el juego se actualize mas rápido y fluidamente.	gameTimer.Interval = 050;
Cambio 5 Eliminamos: this.timeTotal = time; Le sumamos el incremento de tiempo al tiempo.	timeTotal+=dt;

Cambio 6
Creamos una variable alpha de tipo double y creamos su Label, su TextBox y la
	alphaLabel = new Label();
	alphaLabel.Text = "Alpha";
	alphaLabel.Font = new Font(vzLabel.Font, FontStyle.Bold);
	alphaLabel.Top = separation+vzLabel.Top;

ponemos con el control para que funcione.	alphaLabel.Left = 10;
	alphaLabel.Width = 180;
	
	alphaTextBox = new TextBox();
	alphaTextBox.Width = 50;
	alphaTextBox.Text = String.Format("{0}", 2.0m);
	alphaTextBox.Top = alphaLabel.Top;
	alphaTextBox.Left = 200;
	///////////////
	fireButton = new Button();
	fireButton.Text = "Fire";
	fireButton.Height = buttonHeight;
	fireButton.Width = buttonWidth;
	fireButton.Top = alphaLabel.Top+separation;
	fireButton.Left = buttonLeft;
	fireButton.Click += new EventHandler(FireButtonClicked);
	
	resetButton = new Button();
	resetButton.Text = "Reset";
	resetButton.Height = buttonHeight;
	resetButton.Width = buttonWidth;
	resetButton.Top = fireButton.Top+separation;
	resetButton.Left = buttonLeft;
	resetButton.Click += new EventHandler(ResetButtonClicked);

Cambio 7 variable alpha = valor introducido	alpha = Convert.ToDouble(alphaTextBox.Text);

Cambio 9
Añadimos nuevos planetas al juego.
	private Label planetLabel;
	private ComboBox planetComboBox;
	
	planetLabel = new Label();
	planetLabel.Text = "Planeta";
	planetLabel.Font = new Font(planetLabel.Font, FontStyle.Bold);
	planetLabel.Top = 300;
	planetLabel.Left = 10;
	planetLabel.Width = 50;
	
	planetComboBox = new ComboBox();
	planetComboBox.Items.Add("Tierra");
	planetComboBox.Items.Add("Luna");
	planetComboBox.Items.Add("Sol");
	planetComboBox.Items.Add("Jupiter");
	planetComboBox.SelectedIndex = 0;
	planetComboBox.Left = 80;
	planetComboBox.Top =300;
	
	this.Controls.Add(planetComboBox);
	this.Controls.Add(planetLabel);

Cambio 10
Le asignamos una variable “gravedad” a cada planeta. y un fondo diferente
	public double gravedad;
	
	public void FireButtonClicked(object source, EventArgs e)
	{
	selectedItem = (string)planetComboBox.SelectedItem;
	if (String.Equals(selectedItem, "Tierra"))
	{
	wallpaper = Image.FromFile("tierra.png");
	gravedad = 9.81;
	}
	else if (String.Equals(selectedItem, "Luna"))
	{
	wallpaper = Image.FromFile("luna.png");
	gravedad = 1.624;
	}
	else if(String.Equals(selectedItem, "Sol"))
	{
	wallpaper = Image.FromFile("sol.png");
	gravedad = 274;
	}
	else
	{
	wallpaper = Image.FromFile("jupiter.png");
	gravedad = 24.79;
	}

Cambio 11
A la función UpdateLocationAndVelocity le pasamos 4 valores.
 	
Eliminamos variables innecesarias.
 	
calculamos el cos y sin de alpha.
 	
Aplicamos las formulas del tiro parabólico a partir del modulo de la vlocidad y alpha. Actualizamos los valores del saco/pelota
	public void ActionPerformed(object source, EventArgs e)
	{
	bBall.UpdateLocationAndVelocity(timeIncrement, alpha, gravedad);
	
	public void UpdateLocationAndVelocity(double dt, double alpha, double g)
	{
	double x0, z0, modulo_v, angulo, x, vx,vz,z;
	
	x0 = this.posX;
	z0 = this.posZ;
	vx0 = this.velX;
	vz0 = this.velZ;
	timeTotal+=dt;
	
	double cos = Math.Cos(alpha * (Math.PI / 180.0));
	double sin = Math.Sin(alpha * (Math.PI / 180.0));
	
	vx = modulo_v*cos(angulo);
	vz = modulo_v*sin(angulo);
	
	x= x0+vx*cos*timeTotal;
	z = z0+vz*sin*timeTotal-(g/2)*timeTotal*timeTotal;
	
	this.posX = x;
	this.posZ = z;

	this.velX = vx;
	this.velZ = vz;
	
	}
	}

Cambio 12
Actualizamos la interfaz para que pida al usuario la velocidad en módulo y grados.
	vLabel = new Label();
	vLabel.Text = "Velocidad (modulo)";
	vLabel.Font = new Font(vLabel.Font, FontStyle.Bold);
	vLabel.Top = separation;
	vLabel.Left = 10;
	vLabel.Width = 180;
	
	alphaLabel = new Label();
	alphaLabel.Text = "Alpha (grados)";
	alphaLabel.Font = new Font(vLabel.Font, FontStyle.Bold);
	alphaLabel.Top = separation+vLabel.Top;
	alphaLabel.Left = 10;
	alphaLabel.Width = 180;

Cambio 13
Añadimos un label y textbox de resultado para indicar si ganamos o perdemos.
	private Label resultadoLabel;
	private TextBox resultadoTextBook;
	
	resultadoLabel = new Label();
	resultadoLabel.Text = "Resultado: ";
	resultadoLabel.Font = new Font(resultadoLabel.Font, FontStyle.Bold);
	resultadoLabel.Top = 200;
	resultadoLabel.Left = 10;
	resultadoLabel.Width = 70;
	
	resultadoTextBook = new TextBox();
	resultadoTextBook.Width = 150;
	resultadoTextBook.Text = " ";
	resultadoTextBook.Top = resultadoLabel.Top;
	resultadoTextBook.Left = 80;
	
	this.Controls.Add(resultadoLabel);
	this.Controls.Add(resultadoTextBook);

Cambio 14 Añadimos una variable que da posiciones aleatorias.	Random rand = new Random();

Cambio 15
Actualizamos el UpdateDisplay
	public int zPosition;
	public int xPosition;
	Graphics g;
	public void UpdateDisplay()
	{
	g = drawingPanel.CreateGraphics();
	Pen blackPen = new Pen(Color.Black, 2);

	int width = drawingPanel.Width - 1;
	int height = drawingPanel.Height - 1;
	SolidBrush brush = new SolidBrush(Color.Black);
	
	int zLocation = 125 - 67;
	double x = bBall.GetX();
	double z = bBall.GetZ();

Cambio 16
Dibujamos la pelota de color naranja en vez de usar un saco y la pintamos.
	SolidBrush orangeBrush = new SolidBrush(Color.Orange);
	xPosition = (int)(100.0 * x);
	double deltaZ = z - 1.25;
	zPosition = (int)(125 - 100.0 * deltaZ);
	
	g.FillEllipse(orangeBrush,xPosition, zPosition, 15, 15);

Cambio 17 Pintamos el fondo.	g.DrawImage(wallpaper, 0, zLocation, drawingPanel.Width, drawingPanel.Height+40); //if(draw){
Cambio 18 Pintamos al personaje.	g.DrawImage(playerIcon, 7, horizontLine-140, 150, 160); //}

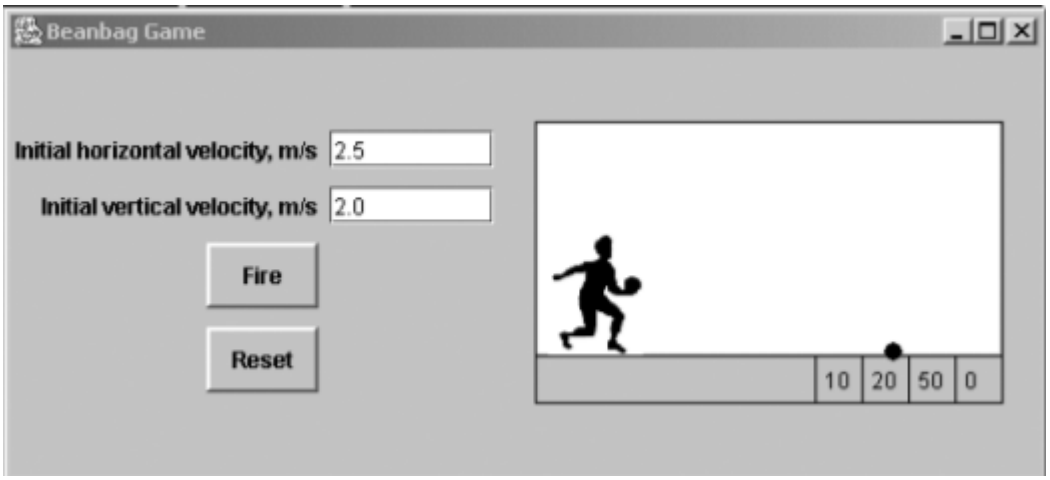
Cambio 19
Dibujamos una caja negra que aparece en posiciones aleatorias de la línea del horizonte. Esta caja la usaremos para indicar si el jugador gana o pierde.
	Rectangle rect = new Rectangle(rectX, rectY, rectWidth, rectHeight);
	//Rectangle rect = new Rectangle((int)horizontLine, (int)horizontLine-50, 50, 50);
	g.DrawRectangle(blackPen, rect);
	
	//g.Dispose();
	
	}

Cambio 20
Finalmente actualizamos la acción “ActionPerformed” y llamamos a la acción que actualiza la posición y la velocidad de la pelota, UpdateDisplay y usamos un “if” para detectar si la pelota acaba dentro de la caja choca con ella y si es así el resultado será “Has perdido". En caso contrario, habremos ganado.
	public void ActionPerformed(object source, EventArgs e)
	{
	double timeIncrement = 0.05;
	bBall.UpdateLocationAndVelocity(timeIncrement, alpha, v, gravedad);
	
	UpdateDisplay();
	
	System.Diagnostics.Debug.WriteLine(xPosition);
	
	if ((zPosition<= rectY+rectHeight)&&(zPosition>= rectY)&&((xPosition >= rectX)&&(xPosition <= rectX+rectWidth))){
	resultadoTextBook.Text="Has perdido";
	gameTimer.Stop();
	}
	
	else if(zPosition>horizontLine){
	resultadoTextBook.Text="Has ganado";
	
	gameTimer.Stop();
	}
	}

	static void Main()
	{
	Application.Run(new BeanBag());
	}

Interfície final del programa

Esta era la interfície original:



Y esta es la final: