Task 6.2 Rectangles and Ellipses

|  |
| --- |
| Public Class Form1  Private Sub Form1\_Paint(sender As Object, e As PaintEventArgs) Handles MyBase.Paint  Dim x As Integer = 80  Dim y As Integer = 10  Dim w As Integer = 300  Dim h As Integer = 200  Dim g As Graphics = e.Graphics  Dim myPen As New Pen(Color.Blue, 10)  g.DrawRectangle(myPen, x, y, w, h)  g.FillRectangle(Brushes.Red, x, y, w, h)  g.FillEllipse(Brushes.Yellow, x, y, w, h)  End Sub  End Class |

Task 6.2 Rectangle inside rectangles

|  |
| --- |
| Public Class Form1  Private Sub Form1\_Paint(sender As Object, e As PaintEventArgs) Handles MyBase.Paint  Dim x As Integer = 80  Dim y As Integer = 10  Dim w As Integer = 300  Dim h As Integer = 200  Dim c As Integer = 0  Dim g As Graphics = e.Graphics  Dim myPen As New Pen(Color.Blue, 10)  For x = 0 To 6 Step 1  Select Case c  Case 0  g.FillRectangle(Brushes.Blue, x, y, w, h)  Case 1  g.FillRectangle(Brushes.Red, x, y, w, h)  Case 2  g.FillRectangle(Brushes.Yellow, x, y, w, h)  Case 3  g.FillRectangle(Brushes.Green, x, y, w, h)  Case 4  g.FillRectangle(Brushes.Violet, x, y, w, h)  Case 5  g.FillRectangle(Brushes.Brown, x, y, w, h)  End Select  x = x + 5  y = y + 5  w = w - 10  h = h - 10  c = c + 1  Next  End Sub  End Class |

Task 6.3 Polygons

|  |
| --- |
| Public Class Form1  Private Sub Form1\_Paint(sender As Object, e As PaintEventArgs) Handles MyBase.Paint  Dim g As Graphics = e.Graphics  Dim myPen As New Pen(Color.Red)  Dim side As Integer = 25 '' the length of the side of a hex  Dim ShortSide As Single = Convert.ToSingle(System.Math.Sin(30 \* System.Math.PI / 180) \* side)  Dim LongSide As Single = Convert.ToSingle(System.Math.Cos(30 \* System.Math.PI / 180) \* side)  Dim shape(5) As PointF  shape(0) = (New Point(100, 100))  shape(1) = (New Point(100 + side, 100))  shape(2) = (New Point(100 + side + ShortSide, 100 + LongSide))  shape(3) = (New Point(100 + side, 100 + LongSide + LongSide))  shape(4) = (New Point(100, 100 + LongSide + LongSide))  shape(5) = (New Point(100 - ShortSide, 100 + LongSide))  g.DrawPolygon(myPen, shape)  g.FillPolygon(Brushes.Red, shape)  Dim myFont As New Font("Arial", 14, FontStyle.Regular)  Dim myBrush As Brush = Brushes.Aqua  g.DrawString("Sam", myFont, myBrush, New Point(100, 150`))  BackColor = Color.Yellow  End Sub  End Class |

Bouncing Ball Project

|  |
| --- |
| Public Class Form1  Dim rand As New Random  Dim x As Integer = 200  Dim y As Integer = 50  Dim xBall2 As Integer = 0  Dim yBall2 As Integer = 0  Dim xmove As Integer = 10  Dim ymove As Integer = 10  Dim x2move As Integer = 10  Dim y2move As Integer = 10  Dim size As Integer = 30  Dim time As Integer = 0  Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load  x = rand.Next(300) + 27  y = rand.Next(200) + 12  xBall2 = rand.Next(300) + 27  yBall2 = rand.Next(200) + 12  End Sub  Private Sub pbxDisplay\_Paint(sender As Object, e As PaintEventArgs) Handles pbxDisplay.Paint  Dim g As Graphics = e.Graphics  g.FillEllipse(Brushes.Red, x, y, size, size)  g.FillEllipse(Brushes.Red, xBall2, yBall2, size, size)  End Sub  Private Sub Timer1\_Tick(sender As Object, e As EventArgs) Handles Timer1.Tick  ''Verifies movement against the walls after the new position is set to the draw  x += xmove  y += ymove  If (y + size >= pbxDisplay.Height) Then  ymove = -ymove  ElseIf (x + size >= pbxDisplay.Width) Then  xmove = -xmove  ElseIf (x + (size - 10) <= pbxDisplay.Location.X) Then  xmove = -xmove  ElseIf (y + (size - 15) <= pbxDisplay.Location.Y) Then  ymove = -ymove  End If  Refresh()  End Sub  Private Sub Timer2\_Tick(sender As Object, e As EventArgs) Handles Timer2.Tick  ''Verifies movement against the walls after the new position is set to the draw  xBall2 += x2move  yBall2 += y2move  If (yBall2 + size >= pbxDisplay.Height) Then  y2move = -y2move  ElseIf (xBall2 + size >= pbxDisplay.Width) Then  x2move = -x2move  ElseIf (xBall2 + (size - 10) <= pbxDisplay.Location.X) Then  x2move = -x2move  ElseIf (yBall2 + (size - 15) <= pbxDisplay.Location.Y) Then  y2move = -y2move  End If  ''checks if the balls are going against each other or not  If (calcDist((x - xBall2), (y - yBall2)) <= size + 3) Then  x2move = -x2move  y2move = -y2move  xmove = -xmove  ymove = -ymove  End If  Refresh()  End Sub  Protected Overrides Function ProcessCmdKey(ByRef msg As Message, keyData As Keys) As Boolean  If (keyData = Keys.Up) Then  size += 5  Refresh()  Return True  ElseIf (keyData = Keys.Down) Then  size -= 5  Refresh()  Return True  ElseIf (keyData.ToString() = "C") Then  time += 1  If (time > 10) Then  time = 0  End If  Select Case time  Case 0  pbxDisplay.BackColor = Color.Aqua  Case 1  pbxDisplay.BackColor = Color.Yellow  Case 2  pbxDisplay.BackColor = Color.Violet  Case 3  pbxDisplay.BackColor = Color.Blue  Case 4  pbxDisplay.BackColor = Color.Green  Case 5  pbxDisplay.BackColor = Color.Gold  Case 6  pbxDisplay.BackColor = Color.Gray  Case 7  pbxDisplay.BackColor = Color.Ivory  Case 8  pbxDisplay.BackColor = Color.LemonChiffon  Case 9  pbxDisplay.BackColor = Color.Chocolate  Case 10  pbxDisplay.BackColor = Color.SlateBlue  End Select  Return True  End If  End Function  Private Function calcDist(a As Integer, b As Integer) As Double  Return Math.Sqrt((Math.Pow(a, 2) + Math.Pow(b, 2)))  End Function  Private Sub btnStart\_Click(sender As Object, e As EventArgs) Handles btnStart.Click  Timer1.Enabled = True  Timer2.Enabled = True  End Sub  Private Sub btnStop\_Click(sender As Object, e As EventArgs) Handles btnStop.Click  Timer1.Enabled = False  Timer2.Enabled = False  End Sub  Private Sub btnQuit\_Click(sender As Object, e As EventArgs) Handles btnQuit.Click  Application.Exit()  End Sub  End Class |