**Computer Programming - Preventing Movement Through a Wall**

Create a project called *WallCollisions.*

Lots of things come into play with this process. Let's image we have a very simple maze. Look at the form below. We have 4 objects: 3 Labels and 1 Picturebox.

Label1

Mario (Picturebox) Label2

Label3

We want to make it so Mario cannot move through any of these walls. We need the following to accomplish this efficiently:

1. An array of all labels used as walls in the maze. This will help us minimize the amount of code drastically.

2. A string variable to keep track of the direction mario is moving. We need to know this so we can move him back in the direction he came, if he collides with one of the walls.

3. Collision detection - To sense if mario hits any of the walls in our maze.

**Variables**

Dim direction As String - This keeps track of the direction mario is moving

Dim wall(100) As Label - This stores all of the walls used in the maze

Dim x As Integer - Used in our For..Next loop

**Events**

 Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

 wall(1) = Label1

 wall(2) = Label2

 wall(3) = label3

 End Sub

This code simply gives the labels in our maze another name which we can use more easily in our collision detection subroutine.

You should continue for every label that is used in your maze. The numbers do not have to match for it to work. For example, you could use wall(4) = label5. Just be sure to include all wall labels.

Private Sub Form1\_KeyDown(ByVal sender As Object, ByVal e As System.Windows.Forms.KeyEventArgs) Handles Me.KeyDown

 If e.KeyCode = Keys.W Then

 mario.Location = New Point(mario.Location.X, mario.Location.Y - 8)

 direction = "up"

 End If

 If e.KeyCode = Keys.S Then

 mario.Location = New Point(mario.Location.X, mario.Location.Y + 8)

 direction = "down"

 End If

 If e.KeyCode = Keys.A Then

 mario.Location = New Point(mario.Location.X - 8, mario.Location.Y)

 direction = "left"

 End If

 If e.KeyCode = Keys.D Then

 mario.Location = New Point(mario.Location.X + 8, mario.Location.Y)

 direction = "right"

 End If

 checkwallcollisions()

 End Sub

This code simply moves the mario picturebox 8 pixels either up, down , left or right based on the key that was pressed (W, S, A or D, respectively). It also sets the direction variable and calls the checkwallcollisions subroutine, every time a button is pushed.

Private Sub checkwallcollisions()

 For Me.x = 1 To 3 3 represents how many walls you have in your maze.

 If mario.Bounds.IntersectsWith(wall(x).Bounds) Then

 If direction = "up" Then

 mario.Location = New Point(mario.Location.X, mario.Location.Y + 8)

 End If

 If direction = "down" Then

 mario.Location = New Point(mario.Location.X, mario.Location.Y - 8)

 End If

 If direction = "right" Then

 mario.Location = New Point(mario.Location.X - 8, mario.Location.Y)

 End If

 If direction = "left" Then

 mario.Location = New Point(mario.Location.X + 8, mario.Location.Y)

 End If

 End If

 Next

 End Sub

This subroutine checks for every wall in your maze, wall(1) to wall(3) in the above example, to see if mario collides with it. If he does, the program then checks to see what direction mario was moving when the collision happened and moves him back in the opposite direction and to his original location.

To whoever is playing the game, this happens so fast, you can't even detect the movement!

Changed needed for your program.

1. Change the number and names of labels in Form1\_Load. Also look out for gaps if you deleted labels.

2. Change the range in the For..Next statement to reflect the number of walls you had in your maze.