| Figure 1. |
| --- |

**Animation using the move method**  
  
The aim of this month's column is to develop a small project that demonstrates simple animation using the Move method. The application only consists of one form, a command button, and an arrow (at least from the user's perspective). When the command button is clicked, the movement of the arrow will be initiated, and it will travel around the perimeter of the form in an anti-clockwise direction (see Figure 1).  
  
The actual format that the animation takes is really inconsequential, that is, whether the arrow travels up or down or clockwise, etc, is not the major issue. What is important is to understand the underlying technique. This can then be used to develop whatever style of animation is best suited to the task at hand.

| Figure 2. |
| --- |

**Designing the application**  
  
Start a new standard project, and add the following controls to the project's default form: a command button, four image controls, and a timer control. Place the command button in the middle of the form. The placement of the other controls is not important because they will have their Visible property set to False (initially at least). Your completed form should resemble Figure 2.  
  
Set up the control's properties as indicated by the following table:

| **Object** | **Property** | **Setting** |
| --- | --- | --- |
| Command button | Caption | Start animation |
| Image | | Name | | --- | | Picture | | | imgArrowDown | | --- | | Arw04dn.ico | |
| Image | | Name | | --- | | Picture | | | imgArrowUp | | --- | | Arw04up.ico | |
| Image | | Name | | --- | | Picture | | | imgArrowRight | | --- | | Arw04rt.ico | |
| Image | | Name | | --- | | Picture | | | imgArrowLeft | | --- | | Arw04lt.ico | |
| Timer | | Enabled | | --- | | Interval | | | False | | --- | | 250 | |

Note that the Picture properties of the Image controls have been set with icon files from the Arrows directory that ships with Visual Basic 5.0.

**Adding the code**  
  
Add the following code to the General Declarations section of the form:  
   
Dim pointingLeft As Boolean  
Dim pointingRight As Boolean  
Dim pointingUp As Boolean  
Dim pointingDown As Boolean   
  
The four boolean variables declared here are used to keep track of the direction in which the arrow is moving: to the left, upwards, to the right, or downwards. Boolean variables can only have one of two valid settings, these being either True or False.  
  
The code in the load event of the form is used to specify the initial direction in which the arrow is to travel. This is done by setting the appropriate Boolean variable to True. In our demonstration program, that direction has arbitrarily been chosen to be towards the right of the screen. Make the right pointing arrow visible by setting its Visible property to True.  
  
Private Sub Form\_Load()  
pointingLeft = False  
pointingRight = True  
pointingUp = False  
pointingDown = False  
imgArrowRight.Visible = True  
End Sub   
  
The command button is clicked by the user to initiate the animation. The only code needed here is to enable the timer control:  
  
Private Sub Command1\_Click()  
Timer1.Enabled = True  
End Sub   
  
It is the code in the Timer event which is responsible for the animation processing. Because the Timer's Interval property has been set to 250, this is the number of milliseconds between Timer events. Once the Timer is enabled (by clicking the Start Animation button), this results in the code in the Timer event being executed every 250 milliseconds. You can experiment with the setting of the Interval property to alter the speed at which the arrow travels.

Private Sub Timer1\_Timer()

If pointingRight = True Then

imgArrowRight.Move imgArrowRight.Left + 80

If imgArrowRight.Left >= (frmArrows.Width - imgArrowRight.Width) Then

pointingRight = False

imgArrowRight.Visible = False

pointingUp = True

imgArrowUp.Move imgArrowRight.Left, imgArrowRight.Top

imgArrowUp.Visible = True

End If

End If

If pointingUp = True Then

imgArrowUp.Move imgArrowUp.Left, imgArrowUp.Top - 80

If imgArrowUp.Top <= 0 Then

pointingUp = False

imgArrowUp.Visible = False

pointingLeft = True

imgArrowLeft.Move imgArrowUp.Left, imgArrowUp.Top

imgArrowLeft.Visible = True

End If

End If

If pointingLeft = True Then

imgArrowLeft.Move imgArrowLeft.Left - 80

If imgArrowLeft.Left <= 0 Then

pointingLeft = False

imgArrowLeft.Visible = False

pointingDown = True

imgArrowDown.Move imgArrowLeft.Left, imgArrowLeft.Top

imgArrowDown.Visible = True

End If

End If

If pointingDown = True Then

imgArrowDown.Move imgArrowDown.Left, imgArrowDown.Top + 80

If imgArrowDown.Top >= (frmArrows.ScaleHeight - imgArrowDown.Height) Then

pointingDown = False

imgArrowDown.Visible = False

pointingRight = True

imgArrowRight.Move imgArrowDown.Left, imgArrowDown.Top

imgArrowRight.Visible = True

End If

End If

End Sub

The code in the Timer event is logically divided into four parts, one for each direction in which the arrow could be travelling at any point in time.  
  
The code in each part has a similar structure.  
  
The initial If statement (If pointingRight = True Then..., If pointingUp = True Then..., etc.) detects the current direction of movement of the arrow.  
  
The following line is where the Move method is used to position (move) the arrow. For example, when the arrow is moving towards the right, its position is incremented a small amount by adding 80 to its Left property. Because the code is executing every 250 milliseconds, the position is being altered by this small amount in each iteration. This is what causes the animation effect.  
  
The next If statement in each part is used to test if the arrow has reached a boundary. Once it has, the following activities are processed:

* The boolean variable for the current direction is set to False
* The arrow which has been moving is made to disappear by setting its Visible property to False
* The boolean variable for the new direction is activated by setting it to True
* The new arrow to be moved is positioned to the co-ordinates of where the last moving arrow was before it disappeared
* The new arrow is made to appear by setting its Visible property to True.

I n a similar fashion, the same sort of logic is applied to the up, left, and down pointing arrow sections of code.  
  
Once activated, the animation then continues until the program is stopped.  
  
**Enhancing the application**  
  
One easy enhancement to the program is to add a Slider control to the application. It only takes a couple of lines of code so that when the Slider's value is changed by the user, the Interval property of the Timer control is also changed to reflect the Slider's new value. This gives the user a convenient way of changing the speed at which the arrow travels.