OPIM 3220  Handout-9

Programming Keyboard Events

Visual Basic provides a great deal of flexibility in handling keyboard keys. VB defines three special keyboard events: **KeyPress**, **KeyDown** and **KeyUp**. The **KeyPress** event is normally used for input validation in textboxes because it recognizes only standard keys (letters, numbers, backspace etc.). The **KeyDown** event is used for more complicated applications like games because it recognizes all the keys on the keyboard (including, arrow keys, ctrl, alt etc.).

**KeyPress**

The **KeyPress** event is usually used with textboxes to perform input validation i.e. to ensure that either only text or numbers are entered into a textbox. Create a new application and add a textbox to the form. The **KeyChar** property of the e argument indicates what key was pressed. Write the following code:

```
Me.Text = e.KeyChar
```

You can also get the ASCII code for the character by the following code:

```
Me.Text = AscW(e.KeyChar)
```

Now add two textboxes and name them **txttextonly** and **txtintonly**. For the **txttextonly** textbox, we want to allow only text i.e. a to z, A to Z, spacebar and backspace. If any other key is pressed, we would like to cancel it. To cancel a KeyPress, we use **e.Handled = True**.

Similarly, **txtintonly** should allow only numbers 0 to 9, and backspace.

We can also create more complicated validation schemes. For example, suppose we want to allow only the following to be entered into a textbox (name: **Textbox3**):

- First Character: 1 to 5
- Second Character: a to k

Also, we will allow only two characters to be entered.

**KeyDown**

Start a new project. Double click on the form to open its code window and look at the **KeyDown** event. This event is activated when a user presses a key on the keyboard. The argument **e** provides information regarding which key was hit. Each key on the keyboard has a corresponding **KeyCode**. Write the following code in the **KeyDown** event of the form

```
Me.Text = e.KeyCode
```

Experiment with it and write down the **KeyCode** values for
The `e` parameter also supports three properties called `Control`, `Alt` and `Shift`. These indicate whether any of these special keys were also hit. Write the following code:

```vbnet
Me.Text = e.KeyCode
If e.Control = True Then
    Me.Text = Me.Text & " Control Key Was Also pressed "
End If
```

Keyboard based games are usually written behind the `KeyDown` event. We will use the above concepts to create a new application. Place a picture box control on the form. Name it `picobj` and place the picture of one of the arrows. Add an `imagelist` control. Add the 4 arrows to the imagelist in the following order: `left, up, right, down`. We can then refer to each arrow using `Me.ImageList1.Images(0), Me.ImageList1.Images(1), Me.ImageList1.Images(2), Me.ImageList1.Images(3)`.

Determine the `keycode` for `up, down, left` and `right` mouse keys. Write the code to move the images appropriately. Each image should move by a `distance` value either horizontally or vertically. You can imagine the distance value to be a measure of speed. Write code to adjust the speed with `ctrl` key. Now modify the program to make it more interesting. Add another picture box control called `picmonster`, place it near the center of the form and add any picture to it.

Now the game we will design works this way. *You are supposed to move the picobj to the top of the form to win. But if you come too close to the picmonster object you lose and the game is over.*

Now let us animate the `picmonster` image. It will be more interesting if the animated image is nonstationary and keeps moving. Try to make the animated object chase the arrow.