Trapping Keypresses with When characterInput and Inkey$

May 15, 2006 -
JanetTerra May 15, 2006

Liberty BASIC can trap keypresses in a graphicbox and a graphics window. You can then identify the key pressed for navigating throughout your program. The command is **When characterInput** and the special variable is **Inkey$**. The graphicbox (or graphics window) must have focus in order to sense the keypress.

```
Print #1, "When characterInput [Keypress]"
Print #1, "Setfocus"
```

The [Keypress] event handler uses Inkey$ to hold the detected character.

```
[Keypress]
key$ = Inkey$
Print key$
```

Ascii Values

When the character of the key is identified, the numerical value of that key can be obtained using the Ascii command. Using the Ascii value can often be easier to use for comparisons than the character itself. Every character has a different Ascii value.

```
Print Asc("A") ' Ascii value = 65 : A-Z = 65 - 90
Print ASC("f") ' Ascii value = 102 : a-z = 97 - 122
Print Asc("?") ' Ascii value = 63
```

Virtual Keys

The graphical representation of the "A" key is "A." Not all keys have graphical representations. Examples include the

- The **TAB** key
- The **CONTROL** key
- The **ALT** (menu) key
- The **Number Pad** keys
- The **Home** key
- The **End** key

- The **Function** keys (F1-F16)

Commonly, the Arrow keys and the Enter key are of most importance. Arrow keys can either be trapped by their Ascii values (37, 38, 39 and 40), or their corresponding _VK_Keys

- _VK_LEFT
- _VK_UP
- _VK_RGHT
- _ VK_DOWN

_VK_Keys are especially useful for those keypresses that are actually composed of two characters.

```
' Enter key has been pressed and trapped with When character Input
key$ = Inkey$
For i = 1 to 2
   Print Mid$(key$, i, 1) ' Prints 0, then 13
Next i
```

To ignore the leading special character, use

```
key$ = Right$(Inkey$, 1)
```

With keys that have graphical representations, the value of the _VK_Key is the same as the Asc(character$). Since the Right$(character$, 1) of a one character string is the character$, then the same command can be used regardless of whether the trapped key is one or two characters in length.

```
character$ = "a"
Print character$ ' Prints a
Print Right$(character$, 1) ' Also prints a
```

This demo uses the command *When characterInput* and the special variable *Inkey$* to sense and trap a keypress. Right$(Inkey$, 1) is stored in key$, and key$ is examined for relevant (or allowed) keypresses. This comparison is made using *Select Case* and *If Then Else End If* statements.

```
    WindowWidth = 600
    WindowHeight = 400
    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

    Open "Trapping Keypresses with Inkey$" for Graphics as #1
    #1, "Trapclose [EndKeypresses]"
```

```
    #1 "Font Verdana 20 Bold"
    xPos = 0 ' Starting x Coordinate
    yPos = 30
' Starting y Coordinate, y is bottom left of character, not top left
    #1 "When characterInput [keypress]"
    #1 "Color Darkblue; Backcolor White"
    #1 "Down; Setfocus"

    Wait

[keypress]
' Some keypresses send 2 characters, so only read rightmost
    key$ = Right$(Inkey$, 1)
    key = Asc(key$)
    dir = 0: keypress = 0
    Select Case
        Case key = 37 ' Left Arrow
            dir = 1
        Case key = 38 ' Up Arrow
            dir = 2
        Case key = 39 ' Right Arrow
            dir = 3
        Case key = 40 ' Down Arrow
            dir = 4
        Case (key > 64) and (key < 91) ' A - Z
            keypress = key ' Asc("A") - Asc("Z")
        Case (key > 96) and (key < 123) ' a - z
            keypress = key - 32 ' Asc("A) - Asc("Z")
        Case (key > 47) and (key < 58)
            keypress = key
        Case key = 13
            keypress = 13
    End Select
    If dir > 0 Then ' Arrow Key
        Select Case dir
            Case 1 ' Move Left
                xPos = xPos - 30
                If xPos < 0 Then
                    xPos = 0
                End If
            Case 2 ' Move Up
                yPos = yPos - 30
                If yPos < 30 Then
                    yPos = 30
                End If
            Case 3 ' Move Right
```

```
                xPos = xPos + 30
                If xPos > 570 Then
                    xPos = 570
                End If
            Case 4 ' Move Down
                yPos = yPos + 30
                If yPos > 360 Then
                    yPos = 360
                End If
        End Select
    End If
    If keypress > 47 Then ' Letter or Number Key
        #1 "Place ";xPos;" ";yPos
        #1 "\";Chr$(keypress)
    End If
    If keypress = 13 Then ' Enter Key
        Confirm "Do you want to quit?";quit$
        If quit$ = "yes" Then [EndKeypresses]
    End If
' Any keypresses not captured in above Select Cases and If Thens are i
gnored
    Wait

[EndKeypresses]
    Close #1
    End
```

For a list of Windows Virtual-Key Codes, view the MSDN Library. The values will need to be converted from hexademical to decimal for Liberty BASIC use.

Sub Event Handlers

If you prefer to use sub event handlers rather than branch label event handlers, you must include receiving variables for the window handle and the character variable.

```
Sub keypressDetected handle$, char$
```

This is the same demo as above, but with sub event handlers. Because variables cannot be seen "inside a sub," xPos and yPos must be declared as **Global** in the beginning of the code.

```
    WindowWidth = 600
    WindowHeight = 400
```

```
    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

    Open "Trapping Keypresses with Sub Events" for Graphics as #1
    #1 "Trapclose EndKeypresses"
    #1 "Font Verdana 20 Bold"
    xPos = 0 ' Starting x Coordinate
    yPos = 30
' Starting y Coordinate, y is bottom left of character, not top left
    Global xPos, yPos
    #1 "When characterInput keypressDetected"
    #1 "Color Darkblue; Backcolor White"
    #1 "Down; Setfocus"

    Wait

Sub keypressDetected handle$, char$
' Some keypresses send 2 characters, so only read rightmost
    key$ = Right$(char$, 1)
    key = Asc(key$)
    dir = 0: keypress = 0
    Select Case
        Case key = 37 ' Left Arrow
            dir = 1
        Case key = 38 ' Up Arrow
            dir = 2
        Case key = 39 ' Right Arrow
            dir = 3
        Case key = 40 ' Down Arrow
            dir = 4
        Case (key > 64) and (key < 91) ' A - Z
            keypress = key ' Asc("A") - Asc("Z")
        Case (key > 96) and (key < 123) ' a - z
            keypress = key - 32 ' Asc("A") - Asc("Z")
        Case (key > 47) and (key < 58)
            keypress = key
        Case key = 13
            keypress = 13
    End Select
    If dir > 0 Then ' Arrow Key
        Select Case dir
            Case 1 ' Move Left
                xPos = xPos - 30
                If xPos < 0 Then
                    xPos = 0
                End If
```

```
            Case 2 ' Move Up
                yPos = yPos - 30
                If yPos < 30 Then
                    yPos = 30
                End If
            Case 3 ' Move Right
                xPos = xPos + 30
                If xPos > 570 Then
                    xPos = 570
                End If
            Case 4 ' Move Down
                yPos = yPos + 30
                If yPos > 360 Then
                    yPos = 360
                End If
        End Select
    End If
    If keypress > 47 Then ' Letter or Number Key
        #1 "Place ";xPos;" ";yPos
        #1 "\";Chr$(keypress)
    End If
    If keypress = 13 Then ' Enter Key
        Confirm "Do you want to quit?";quit$
        If quit$ = "yes" Then [EndKeypresses]
    End If
' Any keypresses not captured in above Select Cases and If Thens are i
gnored
End Sub

Sub EndKeypresses handle$
    Close #1
    End
End Sub
```

Trapping Mouse Events

The When command can also be used to track mouse events, those events being button states and mouse movement. These commands are also When commands. An example of a mouse When command is

Print #main.g, "When leftButtonUp [LeftButtonClicked]"

For more information concerning the mouse When commands, see Trapping Mouse Actions and the When Commands.