Smoother Animation with When characterInput

January 1, 2008 -
JanetTerra

Liberty BASIC graphicboxes and the `When` commands make it easy for Liberty BASIC programmers to detect user input. The majority of `When` commands detect mouse movement and action:

- When leftButtonDown
- When leftButtonUp
- When leftButtonMove
- When leftButtonDouble
- When rightButtonDown
- When rightButtonUp
- When rightButtonMove
- When rightButtonDouble
- When middleButtonDown
- When middleButtonUp
- When middleButtonMove
- When middleButtonDouble
- When mouseMove

One `When` command detects keyboard input:

- When characterInput

The `When` commands only work in a graphicbox (or graphics window). The `When characterInput` command only works when the graphicbox is the control with focus. The `When` is *case insensitive*, meaning `WHEN` works as well as `when` or even `wHeN`. The second portion of the command is *case sensitive*. `When CharacterInput` will not throw an error, but neither will it cause keypresses to be detected.

When characterInput and Inkey$

Traditionally, the Liberty BASIC language associates the native `When characterInput` with `Inkey$`. Like many of Liberty BASIC's special variables, the `Inkey$` variable is *case sensitive*. `inkey$` will not be recognized. From the Liberty BASIC Help File:

- Description:
- This special variable holds either a single typed character or multiple characters including a

Windows virtual keycode. Notice that because Inkey$ is a variable, it is case sensitive. Remember that at this time, only the graphics window and graphicbox controls can scan for keyboard input. The virtual keycodes are standard Windows constants, and include arrow keys, function keys, the ALT, SHIFT, and CTRL keys, etc.

For a more indepth tutorial using `When characterInput` and `Inkey$`, read Trapping Keypresses with When characterInput and Inkey$.

Moving a Sprite with When characterInput and Inkey$

For most purposes, capturing keypresses with a graphicbox, `When characterInput`, and `Inkey$` works quite well. Occassionally, especially with arcade-type games, there is a slight hesitation before true movement begins. The following short program demonstrates this hesitation. Using the Left and Right arrow keys, move the blue circle left and right. Holding the arrow key down keeps the circle moving in that direction.

```
    WindowWidth = 800
    WindowHeight = 200
    UpperLeftX = Int((DisplayWidth - WindowWidth) /2)
    UpperLeftY = Int((DisplayHeight - WindowHeight) /2)
    Graphicbox #demo.g, 20, 20, 750, 100
    Statictext #demo.st, "", 250, 140, 300, 20
    Open "Left and Right" for Window as #demo
    #demo, "Trapclose [XbyTrap]"
    #demo, "Font Verdana 10 Bold"
    #demo.st, "Press Left / Right Arrow Keys to Move"

' Make a sprite to move
    #demo.g, "Down; Color Black; Backcolor Black"
    #demo.g, "Place 22 22; Circlefilled 20"
    #demo.g, "Place 0 45; Boxfilled 45 89"
    #demo.g, "Color Darkblue; Backcolor Blue"
    #demo.g, "Place 22 66; Circlefilled 20"
    #demo.g, "Getbmp discBMP 0 0 45 89"
    #demo.g, "Addsprite sphere discBMP"
    xSphere = 380
    #demo.g, "Spritexy sphere ";xSphere;" 40"
    #demo.g, "Drawsprites"

' Issue When characterInput command
    #demo.g, "When characterInput [MoveSprite]"
    #demo.g, "Setfocus"
Wait
```

```
[XbyTrap]
    Unloadbmp "discBMP"
    Close #demo
End

[MoveSprite]
    key = Asc(Right$(Inkey$, 1))
    Select Case key
        Case _VK_LEFT
            xSphere = Max(10, xSphere - 5)
        Case _VK_RIGHT
            xSphere = Min(700, xSphere + 5)
    End Select
    #demo.g, "Spritexy sphere ";xSphere;" 40"
    #demo.g, "Drawsprites"
Wait
```

The GetAsyncKeyState Call

The MSDN Library has this to say about the `GetAsyncKeyState` call:

- The GetAsyncKeyState function determines whether a key is up or down at the time the function is called, and whether the key was pressed after a previous call to GetAsyncKeyState.

The `GetAsyncKeyState` checks one key value, then returns either a negative number if the key associated with that value is depressed, or a zero if the key associated with that value is not depressed. The following code uses the `When characterInput` command to detect keyboard input, but then checks for specific keypresses. The specific keypresses are the Left Arrow Key (Ascii value equals 37, Virtual constant key equals _VK_LEFT) and the Right Arrow Key (Ascii value equals 39, Virtual constant key equals _VK_RIGHT). Once again, move the blue circle to the left and to the right with the keyboard arrow keys.

```
    WindowWidth = 800
    WindowHeight = 200
    UpperLeftX = Int((DisplayWidth - WindowWidth) /2)
    UpperLeftY = Int((DisplayHeight - WindowHeight) /2)
    Graphicbox #demo.g, 20, 20, 750, 100
    Statictext #demo.st, "", 250, 140, 300, 20
    Open "Left and Right" for Window as #demo
    #demo, "Trapclose [XbyTrap]"
    #demo, "Font Verdana 10 Bold"
    #demo.st, "Press Left / Right Arrow Keys to Move"
```

```
' Make a sprite to move
    #demo.g, "Down; Color Black; Backcolor Black"
    #demo.g, "Place 22 22; Circlefilled 20"
    #demo.g, "Place 0 45; Boxfilled 45 89"
    #demo.g, "Color Darkblue; Backcolor Blue"
    #demo.g, "Place 22 66; Circlefilled 20"
    #demo.g, "Getbmp discBMP 0 0 45 89"
    #demo.g, "Addsprite sphere discBMP"
    xSphere = 380
    #demo.g, "Spritexy sphere ";xSphere;" 40"
    #demo.g, "Drawsprites"

' Issue When characterInput command
    #demo.g, "When characterInput [MoveSprite]"
    #demo.g, "Setfocus"
    currentKey = 0
Wait

[XbyTrap]
    Unloadbmp "discBMP"
    Close #demo
End

[MoveSprite]
    key = _VK_LEFT
    While key = _VK_LEFT
        CallDLL #user32, "GetAsyncKeyState", _
        _VK_LEFT as Long, _
        result as Long
        If result < 0 Then
            key = _VK_LEFT
            xSphere = Max(10, xSphere - 5)
            #demo.g, "Spritexy sphere ";xSphere;" 40"
            #demo.g, "Drawsprites"
        Else
            key = 0
        End If
        CallDLL #kernel32, "Sleep", 20 as Long, result as Void
    Wend
    key = _VK_RIGHT
    While key = _VK_RIGHT
        CallDLL #user32, "GetAsyncKeyState", _
        _VK_RIGHT as Long, _
        result as Long
        If result < 0 Then
```

```
            key = _VK_RIGHT
            xSphere = Min(700, xSphere + 5)
            #demo.g, "Spritexy sphere ";xSphere;" 40"
            #demo.g, "Drawsprites"
        Else
            key = 0
        End If
        CallDLL #kernel32, "Sleep", 20 as Long, result as Void
    Wend
Wait
```

Limitations of GetAsyncKeyState

Unlike `Inkey$`, `GetAsyncKeyState` will not decipher the key pressed. `GetAsyncKeyState` will only tell you if a **specific key** is depressed. Looping through each keyboard character may not be the most efficient way to determine what, if any, key has been pressed. When only a few keys need to be searched, though, and a smooth animated display is required, `GetAsyncKeyState` may offer a better alternative to the native `Inkey$`.