Trapping Mouse Actions and the When Commands

June 9, 2006 -
JanetTerra Jun 9, 2006

When using a Graphics Window, or a Graphicbox inside a Window, Liberty BASIC gives an easy way to detect mouse button events and mouse movements. These mouse events and movements are (from the Liberty BASIC help file) -

- leftButtonDown - the left mouse button has been pressed
- leftButtonUp - the left mouse button has been released
- leftButtonMove - the mouse moved while the left button was down
- leftButtonDouble - the left mouse button has been double-clicked

- rightButtonDown - the right mouse button has been pressed
- rightButtonUp - the right mouse button has been released
- rightButtonMove - the mouse moved while the right button was down
- rightButtonDouble - the right mouse button has been double-clicked

- middleButtonDown - the middle mouse button has been pressed
- middleButtonUp - the middle mouse button has been released
- middleButtonMove - the mouse moved while the middle button was down
- middleButtonDouble - the middle mouse button has been double-clicked

- mouseMove - the mouse moved when no button was down

The commands are case sensitive. An example of a When command is

```
Print #main.g, "When leftButtonUp [LeftClickHere]"
```

Clicking, moving or releasing the mouse buttons is what triggers these events. This first demo demonstrates trapping a Left Button Click and a Right Button Click.

```
    '
    '   Demo demonstrating
    '       When leftButtonUp
    '       When rightButtonUp

    WindowWidth = 407
    WindowHeight = 350

    UpperLeftX = Int((DisplayWidth - WindowWidth)/2)
```

```
    UpperLeftY = Int((DisplayHeight - WindowHeight)/2)

    '   Define a Menu
    Menu #main, "&Options", "E&xit", [EndDemo]
    '   Define a Graphicbox
    Graphicbox #main.g, 0, 0, 400, 300
    '   Open the Window
    Open "The Mouse When Commands" for Window as #main
    '   Add the Trapclose statement to properly end the program
    Print #main, "Trapclose [EndDemo]"
    '   Add the When leftButtonUp command
    Print #main.g, "When leftButtonUp [ButtonLeftUp]"
    Print #main.g, "When rightButtonUp [ButtonRightUp]"

    '   Place the pen in the Down position
    Print #main.g, "Down"
    '   Fill the graphicbox with Darkcyan
    Print #main.g, "Fill Darkcyan"
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Darkcyan
    Print #main.g, "Backcolor Darkcyan"
    '   Assign a position
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Left Button Up - Fills Yellow"
    Print #main.g, "\\Right Button Up - Fills Green"
    '   Preserve the graphics
    Print #main.g, "Flush"
    '   Await the user's next move
    Wait

[EndDemo]
    Close #main
    End

[ButtonLeftUp]
    '   Clear the graphicbox and clear memory
    Print #main.g, "Cls"
    '   Fill the graphicbox with the color Yellow
    Print #main.g, "Fill Yellow"
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Yellow
    Print #main.g, "Backcolor Yellow"
    '   Place the pen
```

```
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Left Button Up - Fills Yellow"
    Print #main.g, "\\Right Button Up - Fills Green"
    '   Preserve the graphics
    Print #main.g, "Flush"
    '   Await the user's next move
    Wait

[ButtonRightUp]
    '   Clear the graphicbox and clear memory
    Print #main.g, "Cls"
    '   Fill the graphicbox with the color Green
    Print #main.g, "Fill Green"
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Green
    Print #main.g, "Backcolor Green"
    '   Place the pen
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Left Button Up - Fills Yellow"
    Print #main.g, "\\Right Button Up - Fills Green"
    '   Preserve the graphics
    Print #main.g, "Flush"
    '   Await the user's next move
    Wait
```

Using the When command not only detects mouse clicks and movements, but also retrieves the x and y positions of the mouse when the event occurred. The x and y positions are stored in the special variables MouseX and MouseY. The variables MouseX and MouseY are also case sensitive. mouseX or MOUSEY will not work.

This second demo detects when the left button or right button has been depressed and released (When leftButtonUp, When rightButtonUp) and also retrieves the mouse coordinates (MouseX and MouseY).

```
    '
    '   Demo demonstrating
    '       When leftButtonUp
    '       When rightButtonUp
    '       MouseX
    '       MouseY

    WindowWidth = 407
    WindowHeight = 350
```

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

    '   Define a Menu
    Menu #main, "&Options", "E&xit", [EndDemo]
    '   Define a Graphicbox
    Graphicbox #main.g, 0, 0, 400, 300
    '   Open the Window
    Open "The Mouse When Commands" for Window as #main
    '   Add the Trapclose statement to properly end the program
    Print #main, "Trapclose [EndDemo]"
    '   Add the When leftButtonUp command
    Print #main.g, "When leftButtonUp [ButtonLeftUp]"
    Print #main.g, "When rightButtonUp [ButtonRightUp]"

    '   Place the pen in the Down position
    Print #main.g, "Down"
    '   Fill the graphicbox with Darkcyan
    Print #main.g, "Fill Darkcyan"
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Darkcyan
    Print #main.g, "Backcolor Darkcyan"
    '   Assign a position
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Left Button Up - Draw Circle"
    Print #main.g, "\\Right Button Up - Draw Square"
    '   Preserve the graphics
    Print #main.g, "Flush"
    '   Await the user's next move
    Wait

[EndDemo]
    Close #main
    End

[ButtonLeftUp]
    '   Obtain mouse x, y positions with the special
    '   variables MouseX and MouseY
    '   Note that MouseX and MouseY are CASE SENSITIVE
    '   mousex, mouseY, etc. will not work
    x = MouseX
    y = MouseY
    '   Clear the graphicbox and clear memory
```

```
    Print #main.g, "Cls"
    '   Fill the graphicbox with the color Yellow
    Print #main.g, "Fill Yellow"
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Darkblue
    Print #main.g, "Backcolor Darkblue"
    '   Place the pen at the mouse click position
    Print #main.g, "Place ";x;" ";y
    '   Draw a filled circle
    Print #main.g, "Circlefilled 20"
    '   Assign the pen color as Black
    Print #main.g, "Color Black"
    '   Assign the backcolor as Yellow
    Print #main.g, "Backcolor Yellow"
    '   Assign a position
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Left Button Up - Draw Circle"
    Print #main.g, "\\Right Button Up - Draw Square"
    '   Preserve the graphics
    Print #main.g, "Flush"
    '   Await the user's next move
    Wait

[ButtonRightUp]
    '   Obtain mouse x, y positions with the special
    '   variables MouseX and MouseY
    '   Note that MouseX and MouseY are CASE SENSITIVE
    '   mousex, mouseY, etc. will not work
    x = MouseX
    y = MouseY
    '   Clear the graphicbox and clear memory
    Print #main.g, "Cls"
    '   Fill the graphicbox with the color Green
    Print #main.g, "Fill Green"
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Darkblue
    Print #main.g, "Backcolor Darkblue"
    '   Place the pen at the mouse click position
    Print #main.g, "Place ";x;" ";y
    '   Draw a filled square
    Print #main.g, "Boxfilled ";x + 40;" ";y + 40
    '   Assign the pen color as Black
    Print #main.g, "Color Black"
```

```
'   Assign the backcolor as Green
Print #main.g, "Backcolor Green"
'   Assign a position
Print #main.g, "Place 100 50"
'   Write the directions
Print #main.g, "\\Left Button Up - Draw Circle"
Print #main.g, "\\Right Button Up - Draw Square"
'   Preserve the graphics
Print #main.g, "Flush"
'   Await the user's next move
Wait
```

The When command can track MouseX and MouseY coordinates when the mouse moves as well. This can be very advantageous when the user is clicking and dragging drawn objects. In this third demo, MouseX and MouseY are retrieved when the left button is first depressed (When leftButtonDown). This initiates the drawing of a filled circle at that spot. MouseX and MouseY represent the number of pixels from the upper left corner of the Graphics Window or Graphicbox. Moving the mouse while keeping the left button depressed (When leftButtonMove) "drags" the circle along the path of the mouse. Releasing the left button (When leftButtonUp) places the final circle drawing at MouseX and MouseY where the mouse was positioned at the time of the left button release.

```
'
'   Demo demonstrating
'       When leftButtonDown
'       When leftButtonMove
'       When leftButtonUp
'       MouseX
'       MouseY

WindowWidth = 407
WindowHeight = 350

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

'   Define a Menu
Menu #main, "&Options", "E&xit", [EndDemo]
'   Define a Graphicbox
Graphicbox #main.g, 0, 0, 400, 300
'   Open the Window
Open "The Mouse When Commands" for Window as #main
'   Add the Trapclose statement to properly end the program
Print #main, "Trapclose [EndDemo]"
'   Add the When leftButton commands
Print #main.g, "When leftButtonDown [ButtonLeftDown]"
```

```
    Print #main.g, "When leftButtonMove [ButtonLeftMove]"
    Print #main.g, "When leftButtonUp [ButtonLeftUp]"

    '   Place the pen in the Down position
    Print #main.g, "Down"
    '   Fill the graphicbox with Yellow
    Print #main.g, "Fill Yellow"
    '   Assign the pen color as Black
    Print #main.g, "Color Black"
    '   Assign the backcolor as Yellow
    Print #main.g, "Backcolor Yellow"
    '   Assign a position
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Left Button Down - Start Circle"
    Print #main.g, "\\Left Button Move - Drag Circle"
    Print #main.g, "\\Left Button Up - Place Circle"
    Wait

[EndDemo]
    Close #main
    End

[ButtonLeftDown]
    '   Obtain mouse x, y positions with the special
    '   variables MouseX and MouseY
    '   Note that MouseX and MouseY are CASE SENSITIVE
    '   mousex, mouseY, etc. will not work
    xOld = MouseX
    yOld = MouseY
    '   Clear the graphicbox and clear memory
    Print #main.g, "Cls"
    '   Fill the graphicbox with the color Yellow
    Print #main.g, "Fill Yellow"
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Darkblue
    Print #main.g, "Backcolor Darkblue"
    '   Place the pen at the mouse click position
    Print #main.g, "Place ";xOld;" ";yOld
    '   Draw a filled circle
    Print #main.g, "Circlefilled 20"
    '   Don't preserve the graphic
    Print #main.g, "Discard"
    '   Await the user's next move
    Wait
```

```
[ButtonLeftMove]
    '   Obtain mouse x, y positions with the special
    '   variables MouseX and MouseY
    '   Note that MouseX and MouseY are CASE SENSITIVE
    '   mousex, mouseY, etc. will not work
    xNew = MouseX
    yNew = MouseY
    '   Erase the previous circle by drawing in background color
    '   Assign the pen color as Yellow
    Print #main.g, "Color Yellow"
    '   Assign the backcolor as Yellow
    Print #main.g, "Backcolor Yellow"
    '   Place the pen at the old mouse position
    Print #main.g, "Place ";xOld;" ";yOld
    '   Draw a filled circle
    Print #main.g, "Circlefilled 20"
    '   Draw the new circle by in Darkblue
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Darkblue
    Print #main.g, "Backcolor Darkblue"
    '   Place the pen at the new mouse position
    Print #main.g, "Place ";xNew;" ";yNew
    '   Draw a filled circle
    Print #main.g, "Circlefilled 20"
    '   Don't preserve the graphics
    Print #main.g, "Discard"
    '   Await the user's next move
    '   Make the new x and new y the old x and old y
    xOld = xNew
    yOld = yNew
    Wait

[ButtonLeftUp]
    '   Obtain mouse x, y positions with the special
    '   variables MouseX and MouseY
    '   Note that MouseX and MouseY are CASE SENSITIVE
    '   mousex, mouseY, etc. will not work
    xFinal = MouseX
    yFinal = MouseY
    '   Clear the graphicbox and clear memory
'    Print #main.g, "Cls"
    '   Fill the graphicbox with the color Yellow
'    Print #main.g, "Fill Yellow"
    '   Assign the pen color as Darkblue
```

```
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Darkblue
    Print #main.g, "Backcolor Darkblue"
    '   Place the pen at the mouse click position
    Print #main.g, "Place ";xFinal;" ";yFinal
    '   Draw a filled circle
    Print #main.g, "Circlefilled 20"
    '   Assign the pen color as Black
    Print #main.g, "Color Black"
    '   Assign the backcolor as Yellow
    Print #main.g, "Backcolor Yellow"
    '   Assign a position
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Left Button Down - Start Circle"
    Print #main.g, "\\Left Button Move - Drag Circle"
    Print #main.g, "\\Left Button Up - Place Circle"
    '   Preserve the graphics
    Print #main.g, "Flush"
    '   Await the user's next move
    Wait
```

No Button Clicking

It may be desirable at times to track the MouseX and MouseY position of the mouse as it moves over the Graphics Window or Graphicbox. This is done with the command

```
Print #main.g, "When mouseMove"
```

This fourth demo is very similar to the third demo above, except no button clicking is required to start and stop the drawing of the filled circle.

```
    '   Demo demonstrating
    '       When mouseMove
    '       MouseX
    '       MouseY

    WindowWidth = 407
    WindowHeight = 350

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

```
    '   Define a Menu
    Menu #main, "&Options", "E&xit", [EndDemo]
    '   Define a Graphicbox
    Graphicbox #main.g, 0, 0, 400, 300
    '   Open the Window
    Open "The Mouse When Commands" for Window as #main
    '   Add the Trapclose statement to properly end the program
    Print #main, "Trapclose [EndDemo]"
    '   Add the When leftButton commands
    Print #main.g, "When mouseMove [MouseMoving]"

    '   Place the pen in the Down position
    Print #main.g, "Down"
    '   Fill the graphicbox with Yellow
    Print #main.g, "Fill Yellow"
    '   Assign the pen color as Black
    Print #main.g, "Color Black"
    '   Assign the backcolor as Yellow
    Print #main.g, "Backcolor Yellow"
    '   Assign a position
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Move Mouse Over this Area"
    '   Preserve the graphics
    Print #main.g, "Flush"
    Wait

[EndDemo]
    Close #main
    End

[MouseMoving]
    '   Obtain mouse x, y positions with the special
    '   variables MouseX and MouseY
    '   Note that MouseX and MouseY are CASE SENSITIVE
    '   mousex, mouseY, etc. will not work
    xNew = MouseX
    yNew = MouseY
    '   Erase the previous circle by drawing in background color
    '   Assign the pen color as Yellow
    Print #main.g, "Color Yellow"
    '   Assign the backcolor as Yellow
    Print #main.g, "Backcolor Yellow"
    '   Place the pen at the old mouse position
    Print #main.g, "Place ";xOld;" ";yOld
```

```
'   Draw a filled circle
Print #main.g, "Circlefilled 20"
'   Draw the new circle by in Darkblue
'   Assign the pen color as Darkblue
Print #main.g, "Color Darkblue"
'   Assign the backcolor as Darkblue
Print #main.g, "Backcolor Darkblue"
'   Place the pen at the new mouse position
Print #main.g, "Place ";xNew;" ";yNew
'   Draw a filled circle
Print #main.g, "Circlefilled 20"
'   Don't preserve the graphics
Print #main.g, "Discard"
'   Assign the color as Black"
Print #main.g, "Color Black"
'   Assign the backcolor as Yellow
Print #main.g, "Backcolor Yellow"
'   Assign a position
Print #main.g, "Place 100 50"
'   Write the directions
Print #main.g, "\\Move Mouse Over this Area"
'   Preserve the graphics
Print #main.g, "Flush"
'   Make the new x and new y the old x and old y
xOld = xNew
yOld = yNew
'   Await the user's next move
Wait
```

Using a Sub as the Event Handler

Mouse events can be sent to a Sub rather than a [BranchLabel]. When that event is sent, sub variables must be present to receive the handle of the sending graphicbox or graphicswindow and the x and y mouse coordinates. This is true even if you don't necessarily need that information. Unlike branch events, you need not use special variables to receive this information. Rather than MouseX and MouseY, any receiving variables will do.

```
'
'   Demo demonstrating
'       When leftButtonUp
'       When rightButtonUp
```

```
    Nomainwin
    WindowWidth = 407
    WindowHeight = 350

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

    '   Define a Menu
    Menu #main, "&Options", "E&xit", [EndDemo]
    '   Define a Graphicbox
    Graphicbox #main.g, 0, 0, 400, 300
    '   Open the Window
    Open "The Mouse When Commands" for Window as #main
    '   Add the Trapclose statement to properly end the program
    Print #main, "Trapclose EndDemo"
    '   Add the When leftButtonUp command
    Print #main.g, "When leftButtonUp ButtonLeftUp"

    '   Place the pen in the Down position
    Print #main.g, "Down"
    '   Fill the graphicbox with Darkcyan
    Print #main.g, "Fill Darkcyan"
    '   Assign the pen color as Darkblue
    Print #main.g, "Color Darkblue"
    '   Assign the backcolor as Darkcyan
    Print #main.g, "Backcolor Darkcyan"
    '   Assign a position
    Print #main.g, "Place 100 50"
    '   Write the directions
    Print #main.g, "\\Click Left Mouse Button"
    '   Preserve the graphics
    Print #main.g, "Flush"
    '   Await the user's next move
    Wait

Sub EndDemo handle$
    Close #handle$
End Sub

Sub ButtonLeftUp handle$, xClick, yClick
    '   handle$ receives the handle of the graphicbox
    '   xClick and yClick receive the mouse x and y coordinates
    '   Clear the graphicbox and clear memory
    Print #handle$, "Cls"
    '   Fill the graphicbox with the color Yellow
    Print #handle$, "Fill Yellow"
```

```
    '   Assign the pen color as Darkblue
    Print #handle$, "Color Darkblue"
    '   Assign the backcolor as Yellow
    Print #handle$, "Backcolor Yellow"
    '   Place the pen
    Print #handle$, "Place ";xClick;" ";yClick
    '   Write the receiving information
    Print #handle$, "\Handle = ";handle$
    Print #handle$, "\Mouse X Position = ";xClick
    Print #handle$, "\Mouse Y Position = ";yClick
    '   Preserve the graphics
    Print #handle$, "Flush"
    '   End the sub and return to previous state
End Sub
```

Trapping Keyboard Events

The When command can also be used to track keyboard events (pressing a key), but only if the Graphics Window or Graphicbox has focus. The command is

```
Print #main.g, "When characterInput[KeyWasPressed]
```

The pressed key is stored in the special variable Inkey$. It is important to remember that When characterInput can only be used with a Graphics Window or Graphicbox, and only when that window or box has focus.

For more information concerning the When characterInput command, see Capturing Keypresses with When characterInput and Inkey$.