Internet Basic Developer Network

Newsletter 4

INTRODUCTION

This brief newsletter presents some follow-up information to our recent announcement about Comet hyperlinks. We've been hearing from Comet developers who want to implement this feature right away and we're very pleased to see this kind of enthusiasm.

One of the first questions we heard about Comet hyperlinks was, "How can I create a screen that contains hyperlinks but doesn't contain an INPUT statement?" There are certainly applications for this type of display, such as menu programs, a Comet screen that mimics a web page, or programs that use hyperlinks to immediately update the screen display (think point-of-sale workstation and you'll get the idea).

The answer comes with Comet2002 Build 311 and, in particular, with a new statement in the Internet Basic language: EVENTWAIT.

The EVENTWAIT statement causes your program to pause indefinitely while waiting for an event to occur (such as a mouse click on a hyperlink). Thus, your program might look something like this:

```
Print (Hyperlink=...)  ! display hyperlink
Print (Hyperlink=...)  ! display hyperlink
Print (Hyperlink=...)  ! display hyperlink

EventSub GetEvent,Event$,Source$

EVENTWAIT

GetEvent:
.
.
.
EventSub GetEvent,Event$,Source$
Return
```

As you can see, this program starts by displaying several Comet hyperlinks, and then sets the EventSub trap. The program then executes the EVENTWAIT statement, which makes the program pause. Unlike WAIT and INPUT (0) "", the EVENTWAIT statement does not display an INPUT field on the screen. That makes it the perfect choice for an application that contains numerous hyperlinks but doesn’t contain an INPUT statement.

When the user clicks on one of the hyperlinks, this program jumps to the GetEvent subroutine, where it determines what action to take. At the end of the subroutine, the

EventSub trap is reset. Finally, the Return statement takes the program back to the EVENTWAIT statement, where the program waits for the next event.

The obvious question is, "How do you end this program?"

The answer is to add a Return statement to the event processing subroutine that goes someplace other than the EVENTWAIT statement. For example:

```
Print (Hyperlink="Exit","QUIT",1)

EventSub GetEvent,Event$,Source$

EVENTWAIT
STOP

GetEvent:

If Event$ =  "QUIT" Then
EventSub  ! clear the EventSub trap
Return
Endif
```

When the user clicks on the "Exit" hyperlink, the program branches to the event processing routine. In this case, the EventSub statement is written without parameters, which turns off event processing. When this Return statement is executed, the program goes to the statement following the EVENTWAIT statement, which is the STOP statement.

SAMPLE PROGRAM

The sample program attached to this message demonstrates several of the relatively new Comet technologies. First, the user interaction with the main screen is entirely via hyperlinks. Each time the user clicks on a link, new information is calculated and displayed. An "Exit" link stops the program.

Second, this program uses EVENTWAIT to pause while waiting for a mouse click.

Third, several of the hyperlinks trigger data entry routines. These routines are implemented with extended windows -- the (CreateWindowEx) and (DeleteWindowEx) mnemonics.

We hope this sample program fuels your imagination with ideas on how to apply Comet hyperlinks and extended windows in your applications.