POLYBUTTON XFCN

The polybutton XFCN provides a mechanism that HyperCard scriptors can use to add polygon-shaped buttons to their HyperCard stacks. These buttons must be transparent, but they do autohilite when they are clicked on. It is possible to have card-layer polybuttons and background-layer polybuttons.

USING THE POLYBUTTON XFCN

The polybutton XFCN is designed to gain control whenever a mouseDown event is sent to a HyperCard card or a HyperCard background that contains polybuttons. In order for this to happen, an 'on mouseDown' handler must be added to the script of every card and every background that contains polybuttons. The script should read as follows:

```
on mouseDown
      if polybutton(buttonDefs) is "pass" then pass mouseDown
end mouseDown
```

Note: "buttonDefs" should be replaced with your polybutton definition list. Methods for doing this are discussed in more detail below.

The polybutton XFCN will return the string "pass" if no polybutton was clicked on; this signals your script to pass the mouseDown on to the next layer. Alternately, if you wish to process mouseDown events that are send to the HyperCard card or background, you could use a script that looks something like this:

```
on mouseDown
      if polybutton(buttonDefs) is "pass" then
            -- Do custom stuff here
            pass mouseDown
      end if
end mouseDown
```

In both scripts, the variable "buttonDefs" is used to indicate that the argument to the polybutton XFCN is a list of polygon definitions. The format of these definitions is described in the next section, and it is

all in human-readable ASCII text. You may place the polybutton definition list anywhere you wish. A hidden text field is one possible solution, but it is recommended that you store the definition list in the script of the card or background that the polybuttons are found in. If you do this, the invocation of the polybutton XFCN will look like this:

```
if polybutton(the script of this card) ...
```

or

```
if polybutton(the script of this bg) ...
```

The polybutton XFCN automatically ignores anything that does not look like a polybutton definition, so your scripts will not confuse it. Polybutton definitions are preceded by "-- <", so they will look like comments to HyperCard. (If you are not storing the definition list in a script, the first minus is optional.)

It is desirable to store the button definition lists in the script of the card or the background because this is where the polyedit XFCN will look for them when it is invoked.

WRITING POLYBUTTON SCRIPTS

Every polybutton must have a name, and this name cannot contain spaces. The name of the polybutton is used to generate a HyperTalk message that is sent to the card or background when the polybutton is clicked on. Therefore, if you had a polybutton named "dogCow", its handler would look like this:

```
on dogCow
      -- This script is executed whenever the polybutton
      -- "dogCow" is clicked
      play "moof"
end dogCow
```

Also, polybuttons may be assigned parameters. These parameters are passed to the polybutton handler when the polybutton is clicked on. This feature is very useful if you have many polybuttons that all do very similar things; you can assign the same name to every polybutton and give each one a different parameter. For example, if you have a large number of polybuttons that all bring the user to a different card when they are clicked on, you could name all of the

polybuttons "funnyShape" and make their parameter the name of the card that they go to. The script might look like this:

```
on funnyShape whatCard
      lock screen
      go to card whatCard
      unlock screen with dissolve
end funnyShape
```

This is much simpler than typing in a different script for every polybutton.

POLYBUTTON DEFINITION LISTS

It is recommended that the polyedit XFCN be used to create and edit polybutton definition lists. For those interested, the format of polybutton definition lists is described below.

The format of a polybutton definition is:

```
-- <polybuttonName param1 param2 ...> polyButtonID enclosingRect
-- offsetVector lineVector1 lineVector2 lineVector3 ...
```

The polybutton XFCN does not care about line breaks and whitespace, but the polyedit XFCN is less forgiving in this respect. Always make each polybutton definition two lines long.

The usage of the polybutton name and parameters was described in the section "Writing Polybutton Scripts". The polybutton ID is currently unused; it is included to make polybuttons more like HyperCard buttons. (Someday it may be possible to refer to a polybutton by ID.) It should be a unique integer, analogous to a HyperCard button ID.

The polybutton's enclosing rectangle is the smallest rectangle that completely surrounds the polybutton. If a mouseDown event occurs outside of this rectangle, the polybutton XFCN does not bother to calculate whether or not the click was inside the polygon. Therefore, if the enclosing rectangle is too small, your polybutton will not behave correctly. If it is too large, some time might be wasted, but no other ill effects will be noticed. The enclosing rectangle is expressed as: "left,top,right,bottom", where each coordinate is an integer.

The second line of a polybutton definition is a series of relative vectors that define the shape of the polybutton. Each vector is expressed in the form "x-offset,y-offset". Vectors are separated by whitespace. The first vector is the offset from the upper left hand corner of the enclosing rectangle to the first node of the polybutton. Each subsequent vector is a relative offset to the next node from the location of the node before it. It is not necessary to close the loop that defines the polybutton: the last line that connects the first and last nodes is implicitly assumed to exist even if it is not defined.

An example polybutton definition follows:

```
-- <shapePolybutton pentagon> 2 122,186,210,270
-- 78,35 -32,38 -46,-19 5,-49 48,-12 25,42
```

Editing polybuttons is much easier if the polyedit XFCN is used. See the documentation on this XFCN for more information.

Send comments and bug reports to:

Greg Anderson
greggor@apple.com