




2-29-2000

KeyBoardEvent Behavior 1.0

Event id's: 200070-200079

By Matthew Peterson

matthew@pinoko.berkeley.edu

Introduction

This set of behaviors provides a framework for capturing keyboard events. To use this framework, place the main KeyBoardEvent behavior:

on the sprite that will function as your keyboard event handling sprite. Alone, the keyboardEvent behavior is useless. Other behaviors must be added to tell the KeyboardEvent behavior what keys to look out for. When one of the polled keys is pressed a custom event is executed. The ID of this custom event is set up by the user as a parameter in the keyboardEvent behavior. The precise key that is pressed is determined by checking the global variable WhichKeyN. The number in WhichKeyN will correspond to the ascii code for the key pressed. If the AsciiArray behavior is also present. A global array of length 127 will be set up to map ascii numbers into characters. See examples below.

Global Variables:
1) KeyBoardOn -- Set to TRUE to poll the keyboard. (Initially set to false)
2) WhichKeyN -- The ascii number of the key pressed. You may use the accompanying asciiArray behavior to give your scripts access to the ascii table.
3) oldKeyN -- The previous value of WhichKeyN.
4) IsCap -- Is the key pressed a capital letter or not.
5) Ascii[127] -- The standard ascii table (see appendix below)

Purpose

As wired movies become more sophistocated, it is increasing important to be able to provide a complete UI. Right now, Quicktime is lacking in its ability to handle keyboard input. The string functions are extremely limited (SetString, and AppendString). There is also no KeyPressed event. There is only a KeyPressed property that must be queried for each and every key you are interested in. If you are interested in the whole keyboard, you must loop

through and test the KeyPressed property for every key. These behaviors help you do this.

Some common uses: Entering text and numbers, Playing Midi notes, keyboard commands.

Quick Reference

When the global variable KeyBoardOn is set to TRUE, the keyboard is polled on each idle event. Which keys are tested is determined by which accessory behaviors are added. If a key is found to be pressed, the custom event (as specified by the user in the behavior's parameters) is executed. This event is only executed when there is a change in the keys being pressed. So if I press 'A' down, the custom event will be executed once, no matter how long I hold down the A key. The event is not executed upon keyUp (See Technical Notes below for information on how to capture keyUp events).

Parameters:
1) `On KeyPress Event ID:` --- Set this parameter to the ID of the custom event you want executed upon a key press.

Global Variables:
1) KeyBoardOn -- Set to TRUE to poll the keyboard. This is initially set to FALSE. Only set it to TRUE when you want to have keyboard events. Polling the keyboard takes up idle event time.
2) WhichKeyN -- The ascii number of the key pressed. You ma use the accompanying asciiArray behavior to give scripts access to the ascii table. But in general it is sufficient to know that 48='0' and 65='A'. For letters, WhichKeyN is always equal to the ascii number of the Upper Case for that letter. If you want to know if it was an Upper Case or a Lower Case letter, you must check the IsCap Global Variable (see below).

   When WhickKey = 0, this means no key is being pressed at the moment.
3) oldKeyN -- The previous value of WhichKeyN.
   You can check if oldKeyN = 0 to determine if there was no key being pressed as the current key was pressed.
4) IsCap -- IsCap is true if the shiftkey is down, or the capsLock is down, but not when both are down.
5) Ascii[127] -- The standard ascii table. This gets set up when the AsciiArray behavior is added to your sprite track. (See appendix)

NOTE: In the QuickTime Player Pro, many keys will do something to your movie. For instance, the delete key will delete the entire movie. The spacebar

will start the movie playing. The arrow keys will move forward and back frames... This makes keyboard input from the Player rather dangerous. One way around this is to present the movie, and then the edit functions of the Player are disabled.

Accessories:


LetterKeys. Add this behavior to the same sprite as the KeyboardEvent behavior. This will allow the Keyboard event behavior to poll for keyboard presses. When a letter is pressed, WhichKeyN is set the the Ascii number associated with the Upper Case of that letter. If you would like to differentiate Upper and Lower case, check the IsCap Global Variable. When TRUE, the letter is upper case, when FALSE, it is lower case. Add 32 to WhichKeyN for the Lower Case characters. The reason I used Upper Case as the default is because they are the appear first in the table, and if one wanted to make a one-to-one mapping with images, it would be easier this way.


NumberKeys. Add this behavior to the same sprite as the KeyboardEvent behavior to have number keys tested. This also polls the following symbols:
negative sign (-), comma(,), period(.), and colon(:)


ControlKeys. Add this behavior to the same sprite as the KeyboardEvent behavior to have the following control keys tested:
Tab, whickKeyN = 9
Delete, whickKeyN = 8
Return, whickKeyN = 13
Up arrow, whickKeyN = 200
Right arrow, whickKeyN = 201
Down arrow, whickKeyN = 202
Left arrow, whickKeyN = 203


AsciiArray. Add this behavior to set up the Global Variable ascii[127]. See the appendix below for a list of the values in this array.
If you want, you can modify this Behavior to reduce the size of this Array down to 91, which is the minimum number to contain numerals and Upper Case letters (You can, of course, make them lower case if you prefer). Use AppendString( ) to concatinate strings out of ascii characters.

Example, (set mystring to “Hi”):

```
GlobalVars ascii[127] mystring
```

```
AppendString(mystring, ascii[72], mystring)
AppendString(mystring, ascii[105], mystring)
```

Reserved Variable Names for this Behavior:

GlobalVariables: KeyBoardOn , WhichKeyN, oldKeyN, IsCap, Ascii

Note: Internal variables all start with “MP_” and thus should not overlap with other variables.

Technical Notes:

This behavior has its own ArcTan function, which it uses to calculate the angle that points to the mouse location.

Revision History:

2-19-2000 Version 1.0 first written.
2-29-2000 Increased performance by placing key-checking code in the same event.

Appendix (Ascii Table):

```
ascii[9] = kTabKey
ascii[10] = kLineFeedCharacter
ascii[13] = kCRLFCharacters
ascii[32] = " "
ascii[33] = "!"
ascii[34] = kDoubleQuoteCharacter
ascii[35] = "#"
ascii[36] = "$"
ascii[37] = "%"
ascii[38] = "&"
ascii[39] = "'"
ascii[40] = "("
ascii[41] = ")"
ascii[42] = "*"
ascii[43] = "+"
ascii[44] = ","
ascii[45] = "-"
ascii[46] = "."
ascii[47] = "/"
ascii[48] = "0"
ascii[49] = "1"
ascii[50] = "2"
ascii[51] = "3"
ascii[52] = "4"
ascii[53] = "5"
ascii[54] = "6"
ascii[55] = "7"
ascii[56] = "8"
ascii[57] = "9"
ascii[58] = ":"
ascii[59] = ";"
ascii[60] = "<"
ascii[61] = "="
ascii[62] = ">"
ascii[63] = "?"
ascii[64] = "@"
ascii[65] = "A"
ascii[66] = "B"
ascii[67] = "C"
ascii[68] = "D"
ascii[69] = "E"
ascii[70] = "F"
ascii[71] = "G"
ascii[72] = "H"
ascii[73] = "I"
ascii[74] = "J"
ascii[75] = "K"
ascii[76] = "L"
ascii[77] = "M"
ascii[78] = "N"
ascii[79] = "O"
ascii[80] = "P"
```

```
ascii[81] = "Q"
ascii[82] = "R"
ascii[83] = "S"
ascii[84] = "T"
ascii[85] = "U"
ascii[86] = "V"
ascii[87] = "W"
ascii[88] = "X"
ascii[89] = "Y"
ascii[90] = "Z"
ascii[91] = "["
ascii[92] = "\"
ascii[93] = "]"
ascii[94] = "^"
ascii[95] = "_"
ascii[96] = "`"
ascii[97] = "a"
ascii[98] = "b"
ascii[99] = "c"
ascii[100] = "d"
ascii[101] = "e"
ascii[102] = "f"
ascii[103] = "g"
ascii[104] = "h"
ascii[105] = "i"
ascii[106] = "j"
ascii[107] = "k"
ascii[108] = "l"
ascii[109] = "m"
ascii[110] = "n"
ascii[111] = "o"
ascii[112] = "p"
ascii[113] = "q"
ascii[114] = "r"
ascii[115] = "s"
ascii[116] = "t"
ascii[117] = "u"
ascii[118] = "v"
ascii[119] = "w"
ascii[120] = "x"
ascii[121] = "y"
ascii[122] = "z"
ascii[123] = "{"
ascii[124] = "|"
ascii[125] = "}"
ascii[126] = "~"
```