Chapter 11: Mouse Control – Moving Things Around.

This chapter will show you how to make your program respond to a mouse. There are two different ways to use the mouse: tracking mode and clicking mode. Both are discussed with sample programs.

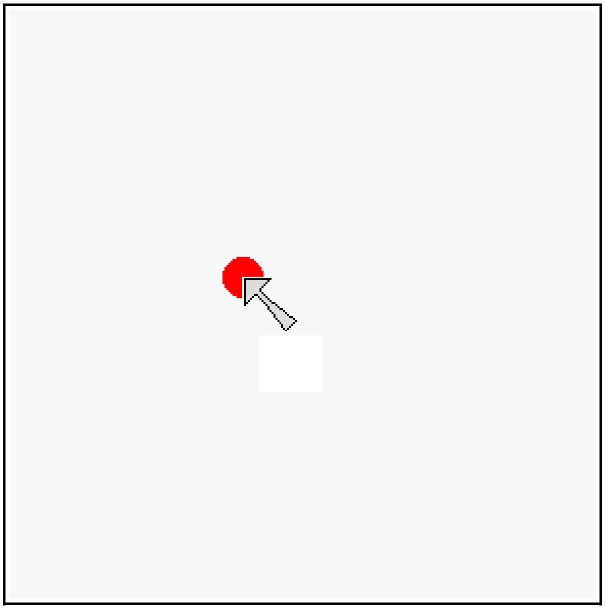
Tracking Mode:

In mouse tracking mode, there are three numeric functions (**mousex**, **mousey**, and **mouseb**) that will return the coordinates of the mouse pointer over the graphics output area. If the mouse is not over the graphics display area then the mouse movements will not be recorded (the last location will be returned).

```
# mousetrack.kbs
# track the mouse with a circle

print "Move the mouse around the graphics window."
print "Click left mouse button to quit."

fastgraphics

# do it over and over until the user clicks left
while mouseb <> MOUSEBUTTON_LEFT
     # erase screen
     clg
     # draw new ball
     color red
     circle mousex, mousey, 10
     refresh
end while

```

```
print "all done."
end
```

Program 70: Mouse Tracking



Sample Output 70: Mouse Tracking

New Concept

```
mousex or mousex()
mousey or mousey()
mouseb or mouseb()
```

The three mouse functions will return the current location of the mouse as it is moved over the graphics display area. Any mouse motions outside the graphics display area are not recorded, but the last known coordinates will be returned.

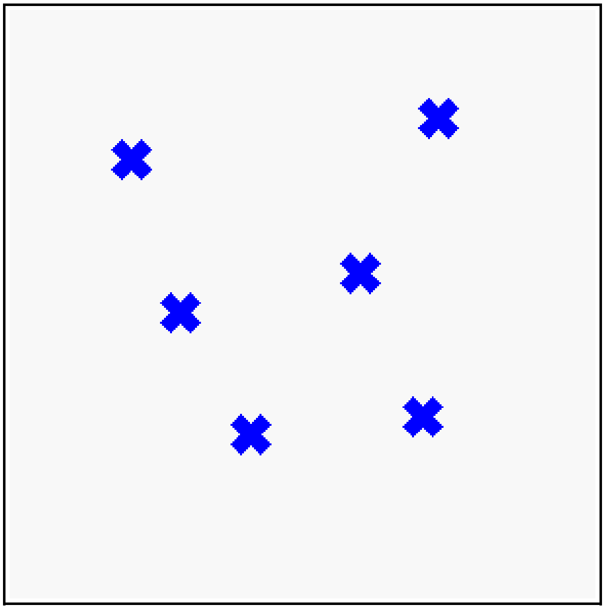
<table>
<tr><td><b>mousex</b></td><td>Returns the x coordinate of the mouse pointer position. Ranges from 0 to <b>graphwidth</b> -1.</td></tr>
<tr><td><b>mousey</b></td><td>Returns the y coordinate of the mouse pointer position. Ranges from 0 to <b>graphheight</b> -1.</td></tr>
<tr><td><b>mouseb</b></td><td>
<table>
<tr><td>0 or
MOUSEBUTTON_NONE</td><td>Returns this value when no mouse button is being pressed.</td></tr>
<tr><td>1 or
MOUSEBUTTON_LEFT</td><td>Returns this value when the "left" mouse button is being pressed.</td></tr>
<tr><td>2 or
MOUSEBUTTON_RIGHT</td><td>Returns this value when the "right" mouse button is being pressed.</td></tr>
<tr><td>4 or
MOUSEBUTTON_CENTER</td><td>Returns this value when the "center" mouse button is being pressed.</td></tr>
</table>
If multiple mouse buttons are being pressed at the same time then the value returned will be the button values added together.</td></tr>
</table>

Clicking Mode:

The second mode for mouse control is called "Clicking Mode". In clicking mode, the mouse location and the button (or combination of buttons) are stored when the click happens. Once a click is processed by the program a *clickclear* command can be executed to reset the click, so the next one can be recorded.

```
# mouseclick.kbs
# X marks the spot where you click

print "Move the mouse around the graphics window"
print "click left mouse button to mark your spot"
print "click right mouse button to stop."
clg
clickclear
while clickb <> MOUSEBUTTON_RIGHT
     # clear out last click and
     # wait for the user to click a button
     clickclear
     while clickb = MOUSEBUTTON_NONE
          pause .01
     end while
     #
     color blue
     stamp clickx, clicky, 5, {-1,-2, 0,-1, 1,-2, 2,-1, 1,0, 2,1, 1,2, 0,1, -1,2, -2,1, -1,0, -2,-1}
end while
print "all done."
end
```

Program 71: Mouse Clicking



Sample Output 71: Mouse Clicking

New Concept

```
clickx  or clickx()
clicky  or clicky()
clickb or clickb()
```

The values of the three click functions are updated each time a mouse button is clicked when the pointer is on the graphics output area. The last location of the mouse when the last click was received are available from these three functions.

New Concept

```
clickclear
```

The **clickclear** statement resets the **clickx**, **clicky**, and **clickb** functions to zero so that a new click will register when **clickb** <> 0.

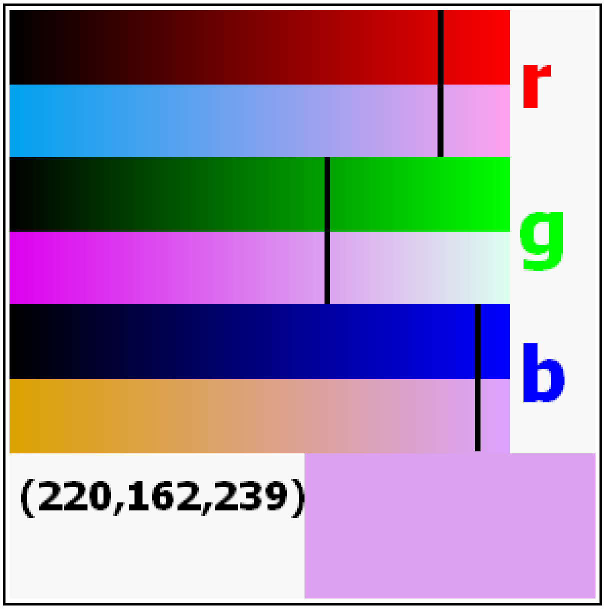
Big Program

The big program this chapter uses the mouse to move color sliders so that we can see all 16,777,216 different colors on the screen.

```
# colorchooser.kbs
fastgraphics

print "colorchooser - find a color"
print "click and drag red, green and blue sliders"

# variables to store the color parts
r = 128
g = 128
b = 128

call display(r,g,b)

while true
     # wait for click
     while mouseb = 0
          pause .01
          end while
          # change color sliders
          # the red slider y range is 0 >= red < 75
          if mousey < 75 then
               r = mousex
               if r > 255 then r = 255
          end if
          # the green slider y range is 75 >= red < 150
          if mousey >= 75 and mousey < 150 then
               g = mousex
               if g > 255 then g = 255
          end if
          # the blue slider y range is 150 >= red < 225
          if mousey >= 150 and mousey < 225 then
               b = mousex
               if b > 255 then b = 255
          end if
          call display(r,g,b)
     end while
     end

     subroutine colorline(r,g,b,x,y)
          # draw part of the color bar the color r,g,b from x,y to x,y+37
          color rgb(r, g, b)
          line x, y, x, y+37
     end subroutine

     subroutine redsliderbar(r,g,b)
          # draw the red bar from 0,0 to 255,74
          font "Tahoma", 30, 100
          color rgb(255, 0, 0)
          text 260, 0, "r"
          for t = 0 to 255
               # red and red hues
               call colorline(t, 0, 0, t, 0)
               call colorline(t, g, b, t, 38)
          next t
          color black
     rect r-1, 0, 3, 75
end subroutine

subroutine greensliderbar(r,g,b)
     # draw thegreen bar from 0,75 to 255,149
     font "Tahoma", 30, 100
     color rgb(0, 255, 0)
     text 260, 75, "g"
     for t = 0 to 255
          # green and green hues
          call colorline(0, t, 0, t, 75)
          call colorline(r, t, b, t, 113)
     next t
     # slider
     color black
     rect g-1, 75, 3, 75
end subroutine

subroutine bluesliderbar(r,g,b)
     # draw the blue bar from 0,150 to 255,224
     font "Tahoma", 30, 100
     color rgb(0, 0, 255)
     text 260, 150, "b"
     for t = 0 to 255
          # blue and blue hues
          call colorline(0, 0, t, t, 150)
          call colorline(r, g, t, t, 188)
     next t
     # slider
     color black
     rect b-1, 150, 3, 75
end subroutine

subroutine display(r, g, b)
     clg
     call redsliderbar(r,g,b)
     call greensliderbar(r,g,b)
     call bluesliderbar(r,g,b)
     # draw swatch
```

```
          color rgb(r,g,b)
          rect 151,226,150,75
          refresh
          # draw the RGB values
               color black
          font "Tahoma", 13, 100
          text 5, 235, "(" + r + "," + g + "," + b + ")"
end subroutine
```

Program 72: Big Program - Color Chooser



```
(220,162,239)
```

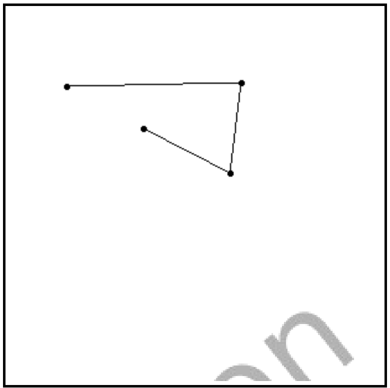
Sample Output 72: Big Program - Color Chooser

Exercises:

abc **Word Search**	r f m t x v t x n j j a a o h k s f o u n c e y u t c l e c b e x l e s h i y l k n z m c s e w l i c t m o r k u b k c i e z u n i c o g k l r p s g s g i m y c j i e h w l h l m c x l x m f z a t c center, clickb, clickclear, clickx, clicky, left, mouseb, mousex, mousey, right

Problems	1. Create a program that will draw a series of connected lines and display the points on the screen as the lines are drawn. When the left button of the mouse is clicked draw a small circle, print the coordinates, draw a line to the previous coordinates (if not the first point), and remember the point so that it can be the start of the next line. Repeat this until the user clicks stop.

```
46,62
187,59
178,132
108,96
```



2. Create a program that will allow the user to use the mouse like a paintbrush. When the user has the left button depressed then plot a point at that location. To make the line wider you may draw a circle with a radius of 2 or 3.

For extra skill when the user presses the right button make the pen color a random color

3. Use the smiling face subroutine from Problem 1 in the subroutines chapter to make a mouse drawing program with the smile. When the user clicks on a point of the screen draw a face there.