PRACTICE 9: EVENTS

Making Your Program Eventful

9

In this lesson we'll explain the `GraphicsWindow` object and how to interact with the graphics window, using events like keyboard strokes or mouse clicks.

GraphicsWindow

First let's start by making the graphics window appear:

```
GraphicsWindow.Width=300
GraphicsWindow.Height=300
GraphicsWindow.Title = "Graphics Window"
```

Run the code, and you should see a white window pop up, with the title "Graphics Window" at the top. Feel free to change the width, height, or title to suit your liking.

The graphics window can be used to do draw objects and allow you to create a program that reacts to mouse clicks or key presses. We'll cover the basics of drawing shapes and some basic events.

Now let's add a shape to the graphics window:

```
GraphicsWindow.Width=300
GraphicsWindow.Height=300
GraphicsWindow.Title = "Graphics Window"

shape1 = Shapes.AddRectangle(100, 50)
Shapes.Move(shape1, 100, 125)
```

The `Shapes.AddRectangle(100, 50)` adds a rectangle of width 100 and height 50 to the screen. The function `Shapes.Move(shape1, 100, 125)` moves the rectangle to the (x, y) position (100, 125). Feel free to move around the shape or try to change its dimensions.

Keyboard Events

What do we do if we want to make something happen when someone presses the Enter key? The code looks like this:

```
GraphicsWindow.Height = 300
GraphicsWindow.Width = 300
GraphicsWindow.Title = "Graphics Window"

shape1 = Shapes.AddRectangle(100, 50)
Shapes.Move(shape1, 100, 125)
return = "Return"
GraphicsWindow.KeyDown = keydown
GraphicsWindow.KeyUp = keyup

Sub keydown
  If GraphicsWindow.LastKey = return then
    Shapes.Rotate(shape1, 90)
  EndIf
EndSub

Sub keyup
  If GraphicsWindow.LastKey = return then
    Shapes.Rotate(shape1, 0)
  EndIf
EndSub
```

The key aspect here is the idea of sub-routines. Sub-routines are a series of instructions that are executed only if a specific action is taken. In this case the subroutines are triggered if a key is pressed up or down. When this happens the series of instructions which rotate the rectangle are executed.

Mouse Events

We can also add sub-routines for mouse events.

The following code rotates the shape when the left or right mouse button is clicked – on `GraphicsWindow.MouseDown`.

When the mouse is moved and if the left button has been pressed down, the title gets updated with the current mouse position using `GraphicsWindow.MouseX` and `GraphicsWindow.MouseY` which represents the (x, y) position in the `GraphicsWindow` coordination system.

When the mouse button is released, the title gets set to the original value. The shape returns to original position as well.

```
GraphicsWindow.MouseDown = mousedown
GraphicsWindow.MouseMove = mousemove
GraphicsWindow.MouseUp = mouseup

Sub mousedown
  If Mouse.IsLeftButtonDown Then
    Shapes.Rotate(shape1, 45)
  ElseIf Mouse.IsRightButtonDown Then
    Shapes.Rotate(shape1, -45)
  EndIf
EndSub

Sub mousemove
  If Mouse.IsLeftButtonDown Then
    GraphicsWindow.Title =  "Mouse:(" +
GraphicsWindow.MouseX + ", " + GraphicsWindow.MouseY + ")"
  EndIf
EndSub

Sub mouseup
  GraphicsWindow.Title = "Graphics Window"
  Shapes.Rotate(shape1, 0)
EndSub
```

Challenge: Print Out the Keyboard Events

We've learned the `KeyUp` and `KeyDown` events. Write a program to print out the key being pressed using `GraphicsWindow.LastKey`. This is helpful for the next challenge so you know how to detect which key was pressed.

Remember how to print out text using `GraphicsWindow.DrawText(x,y,text)`? It requires the position where the text is drawn. Try printing the text on `KeyDown` event and clearing the screen on `KeyUp` event using `GraphicsWindow.Clear()`. See what happens if you don't clear the screen.

Now try using a loop to print out all the keys that have been pressed without clearing the screen. You can decide if it should stop with a certain condition, when if you want to assign a key to clear the screen.

Challenge: Draw Paths with Keyboard

Design a game that draws paths with the control of keyboard. For example, starting from a position that is not close to the edge of the screen such as position (x=100, y=100), when the left/right arrow key is pressed down, draw the path starting from current position to the left/right with a certain length (length=10). When the up/down arrow key is pressed down, draw the path from current position to up/down. That way, the player can use the four arrow keys to draw the paths from starting position to ending position.

The drawing can be done using `GraphicsWindow.DrawLine(x1,y1,x2,y2)`. This will draw a line from the coordinate (x1, y1) to the coordinate (x2, y2).

The arrow key being pressed can be detected using `GraphicsWindow.LastKey`, just like using "Return" for the return key. You can use the first challenge to test what the `LastKey` is for each of the arrow keys and any other key!

To make it more interesting, you can use your imagination to assign any key event to control the drawing to make more complicated shapes. Have fun to exploring the possibilities and see what you can come up with!

Challenge: Draw with Mouse

Make a drawing application using mouse events. The goal is that you can use the mouse to draw a picture.

Hint: Think of ways you can utilize the mouse events to control the pen or brush to draw. You can use various interactions to achieve the goal. For example, when the mouse button is release, draw a small circle or square at the current mouse position. You can

also use mouse events to draw a line between two position to make stick figure. This is similar to a mouse drag which starts from the `MouseDown` event to the `MouseUp` event, with `MouseMove` in between.

Discussion Questions

- Can you think of how subroutines are used in real life?
- Think about a video game you've played. Describe what events and subroutines are used in the game.
- What are some limitations of this event-based programming style?
- Can you think of a way to detect a key combination such as Ctrl+C?
- What events or event combinations you can think of other than the ones we've discussed?

Additional Resources

- Small Basic Curriculum: Lesson 2.1: Graphics Window
  - https://aka.ms/sbcurriculum2.1
- Small Basic Curriculum: Lesson 3.4: Events and Interactivity
  - https://aka.ms/sbcurriculum3.4
- Mouse and Keyboard Events
  - https://aka.ms/mouseandkeyboardevents
- Small Basic: Event Basics
  - https://aka.ms/eventbasics
- Small Basic Getting Started Guide: Chapter 11: Events and Interactivity
  - https://aka.ms/sbguidechapter11
- Small Basic – Mouse Position
  - https://aka.ms/sbmouseposition