**Making an Object React to Something**

We can make shapes react when someone presses a key by using *event bindings*. ***Events***are things that occur while a program is running, such as someone moving the mouse, pressing a key, or closing a window. You can tell tkinter to watch for these events and then do something in response.

When you entered the code to show the x,y coordinates when the mouse was moved, you declared the event in the function parameter.

#function to set-up x,y coordinates to show

def moved(**event**):

canvas.itemconfigure(tag, text="(%r, %r)" % (event.x, event.y))

#Call to show coordinates

**canvas.bind("<Motion>", move**d)

tag = canvas.create\_text(250, 25, text="", anchor="nw")

To begin handling events (making Python do something when an event occurs), we first create a function. The binding part comes when we tell tkinter that a particular function is bound (or associated) to a specific event; in other words, it will be automatically called by tkinter to handle that event.

For example, to make the triangle move when the enter key is pressed, we can define this function:

**def movetriangle(event):**

**canvas.move(1, 5, 0)**

The function takes a single parameter (event), which tkinter uses to send information to the function about the event. We now

tell tkinter that this function should be used for a particular event, using the bind\_all function on the canvas. The full code now looks like this:

def movetriangle(event):

canvas.move(1, 5, 0)

**canvas.bind\_all('<KeyPress-Return>', movetriangle)**

The first parameter in this function describes the event that we want tkinter to watch for. In this case, it’s called <KeyPress-Return>, Using tkinter for Better Graphics 187 which is a press of the enter or return key. We tell tkinter that the movetriangle function should be called whenever this KeyPress event occurs. Run this code, click the canvas with your mouse, and then try pressing enter on your keyboard.

How about changing the direction of the triangle depending on different key presses, such as the arrow keys? That’s no problem. We just need to change the movetriangle function to the following:

def movetriangle(event):

if event.keysym == 'Up':

canvas.move(1, 0, -3)

elif event.keysym == 'Down':

canvas.move(1, 0, 3)

elif event.keysym == 'Left':

canvas.move(1, -3, 0)

else:

canvas.move(1, 3, 0)

canvas.bind\_all('<KeyPress-Up>', movetriangle)

canvas.bind\_all('<KeyPress-Down>', movetriangle)

canvas.bind\_all('<KeyPress-Left>', movetriangle)

canvas.bind\_all('<KeyPress-Right>', movetriangle)

The event object passed to movetriangle contains several variables. One of these variables is called keysym (for key symbol), which is a string that holds the value of the actual key pressed. The line if event.keysym == 'Up': says that if the keysym variable contains the string 'Up', we should call canvas.move with the parameters (1, 0, -3), as we do in the following line. If keysym contains 'Down', as in elif event.keysym == 'Down':, we call it with the parameters (1, 0, 3), and so on.

Remember that the first parameter is the identifying number for the shape drawn on the canvas, the second is the value to add to the x (horizontal) coordinate, and the third is the value to add to the y (vertical) coordinate.

We then tell tkinter that the movetriangle function should be used to handle events from four different keys (up, down, left, and right). The following shows how the code looks at this point. When you enter this code, it will again be a lot easier if you create a new shell window by selecting **File**4**New Window**. Before running the code, save it with a meaningful filename, such as *movingtriangle.py*.

from tkinter import \*

tk = Tk()

canvas = Canvas(tk, width=400, height=400)

canvas.pack()

canvas.create\_polygon(10, 10, 10, 60, 50, 35)

def movetriangle(event):

if event.keysym == 'Up':

canvas.move(1, 0, -3)

elif event.keysym == 'Down':

canvas.move(1, 0, 3)

elif event.keysym == 'Left':

canvas.move(1, -3, 0)

else:

canvas.move(1, 3, 0)

canvas.bind\_all('<KeyPress-Up>', movetriangle)

canvas.bind\_all('<KeyPress-Down>', movetriangle)

canvas.bind\_all('<KeyPress-Left>', movetriangle)

canvas.bind\_all('<KeyPress-Right>', movetriangle)

On the first line of the movetriangle function, we check whether the keysym variable contains 'Up' at u. If it does, we move the triangle upward using the move function with the parameters 1, 0, -3 at v. The first parameter is the identifier of the triangle, the second is the amount to move to the right (we don’t want to move horizontally, so the value is 0), and the third is the amount to move downward (–3 pixels).

We then check whether keysym contains 'Down' at w, and if so, we move the triangle down (3 pixels) at x. The final check is whether the value is 'Left' at y, and if so, we move the triangle left (– 3 pixels) at z. If none of the values are matched, the final else at { moves the triangle right at |. Now the triangle should move in the direction of the pressed arrow key.

**All of this information is on pages 186-188 in the pdf or 212-214 in to browser.**