Smalltalk Programming
Lesson 3

For today's lesson, you will learn how to create a graphic button that uses an expression from your previous lesson. This button will be used to evaluate your expression, which will then move your *EllipseMorph*.

1. We will use an expression from Lesson 2 that moves your *EllipseMorph* to the right. You will assign that expression to a variable. Will your new variable run your expression? In your existing *Workspace* window, type the following code and then "do it" for each line:

```
moveRight := b x: b x + 10.
moveRight.
```

2. What happened when you evaluated the moveRight variable (the second line)? Did you notice anything? Try evaluating the first line again. If you did not notice anything, try evaluating the expression a few more times. What do you notice?

3. The expression in the first line of Step 1 will move your *EllipseMorph* to the right and then assign your *EllipseMorph* object (the variable *b*) to the moveRight variable. This is not exactly what we are looking for. We are wanting a variable that can run when evaluated.

4. Let's try something different. Change the expression that you typed in Step 1 to look like the line below and then "do it":

```
moveRight := [b x: b x + 10].
```

5. Did you notice anything this time? If not, that is okay. You should not have noticed anything just yet.

6. What is different in the expression in Step 4 from the first line you typed in Step 1?

7. You added [and]. In Smalltalk these create a *Block*. These []'s will make a difference, but there is one more thing needed in order for this to work.

8. In Smalltalk the []'s create a *Block*. Using a *Block* lets you delay evaluating its expression(s) until you want to. There is a lot of power with these and you will find them in a lot of Smalltalk code.

9. To have your *Block* evaluate its expression, you send to it the *value* message. In your existing *Workspace* window, type the following code and then "do it":

```
moveRight value.
```

10. What do you notice this time? Your *EllipseMorph* moved! Every time you send *value* to a *Block*, you cause it to evaluate the expression(s) inside.

11. Note that you do not need to assign a Block to a variable to evaluate its expression. You can also run the following line and get the same result. However, for our lesson, we want to use a variable that can act as a reusable code expression.

```
[b x: b x + 10] value.
```

12. You can use the variable name of your *Block* to create a simple button that can be used to evaluate its expression. Type the following code in your *Workspace*, highlight all the lines, and then “do it”:

```
s := SimpleButtonMorph new.
s label: 'Move right'.
s target: moveRight.
s actionSelector: #value.
s openInWorld.
```

13. You should see a small rectangle button in the upper left corner of your screen. Because a message was not sent to the button, letting it know where to go, that is its default opening location. Click on the button. What do you see?

14. Can you figure out what the code in Step 11 is doing by looking at it?

15. The special word *#value* that you sent is a special way to send *value* to your *Block* but it works just like Step 9. This special way will prevent *value* from being used where it should not be.

16. Save and Quit your Smalltalk image.