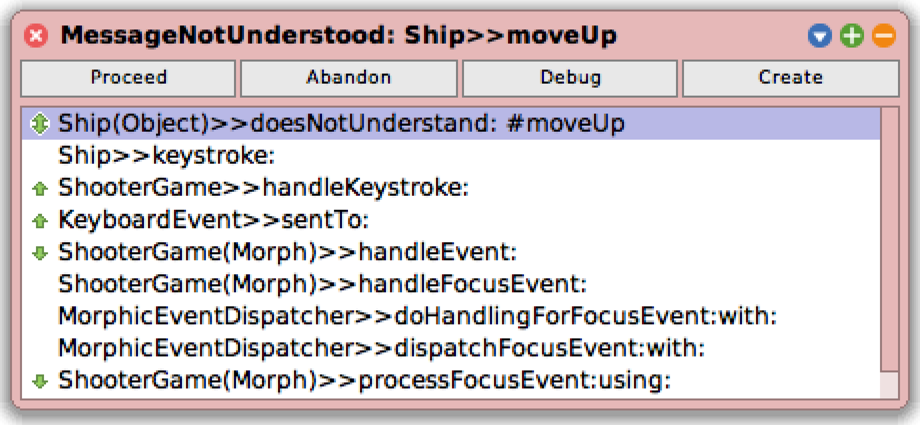
Smalltalk Programming
Lesson 12

In your last lesson you added code to handle keyboard events that could be received and used for the movement of your ship. Today you will code the movement of your ship.

1. In this lesson you will learn another way to enter your code. In most programming languages, when an error occurs within a program the program will abruptly end with an error. In Smalltalk, the program does not end when an error occurs. Smalltalk will popup a debugger window that displays when an error happens. Instead of only showing you the error, the debugger also provides a place where you can fix the error as well. Once you have made the change(s) to fix the error, your program can continue running as if nothing happened! Some programmers like to use the debugger a lot when coding. Today you will be using the debugger to enter your code.

2. Look at the code in `Ship>>keystroke:` (that is, the `keystroke:` method in the `Ship` class) that you entered in the last lesson. Notice that the methods `moveLeft`, `moveRight`, `moveUp`, and `moveDown` have not actually been created yet.

3. Move your mouse cursor over your game window. Press the up arrow key *only once*. The predebugging window will popup as seen in Figure 1 below.



```
Ship(Object)>>doesNotUnderstand: #moveUp
Ship>>keystroke:
ShooterGame>>handleKeystroke:
KeyboardEvent>>sentTo:
ShooterGame(Morph)>>handleEvent:
ShooterGame(Morph)>>handleFocusEvent:
MorphicEventDispatcher>>doHandlingForFocusEvent:with:
MorphicEventDispatcher>>dispatchFocusEvent:with:
ShooterGame(Morph)>>processFocusEvent:using:
```

Figure 1: Predebugger window

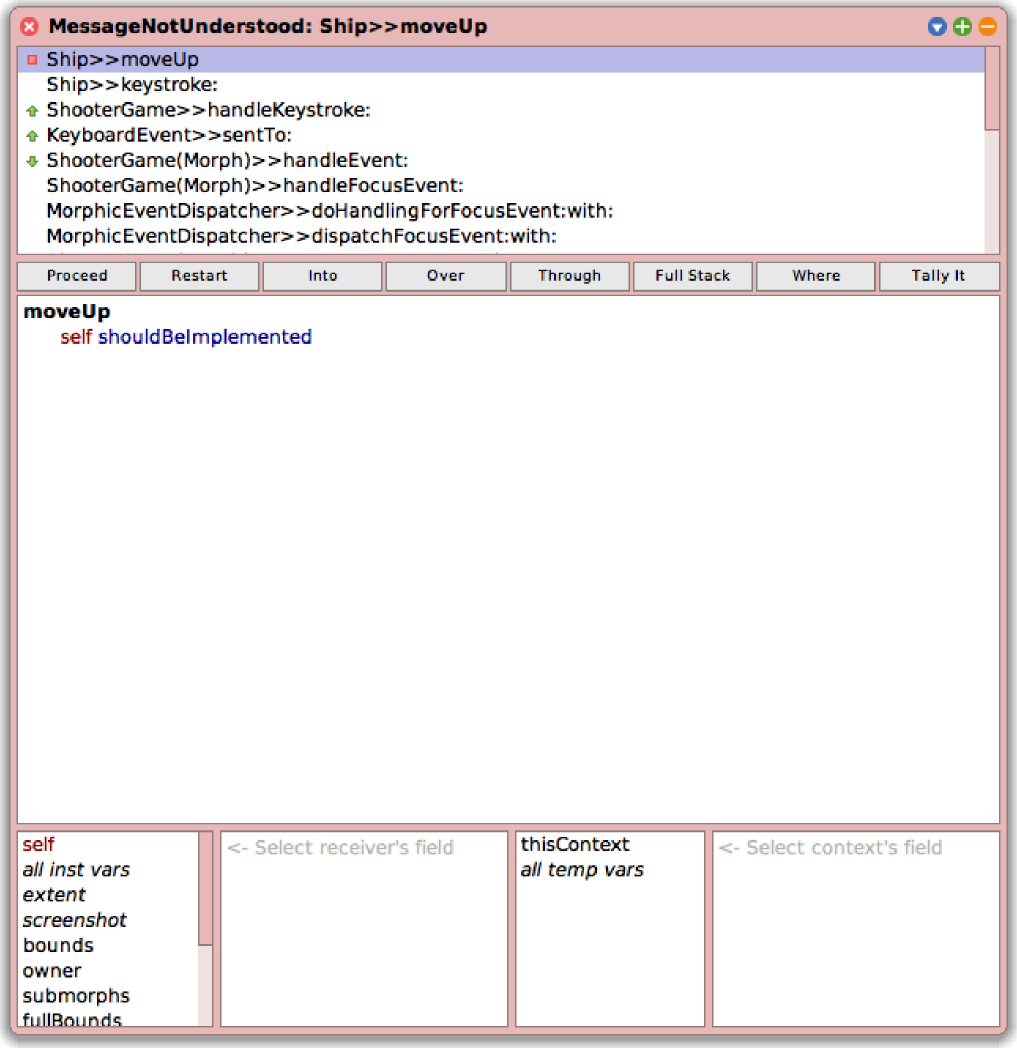
4. Read the error message at the top of your predebugger window. What does it say? What might it mean?

5. The Smalltalk predebugger says that it does not know what `moveUp` in your `Ship` class is. This is because you have not yet added the code for the `moveUp` method. Your predebugger window gives you the option to create the method.

6. Select the “Create” button to use the debugger window, then enter the code for your `moveUp` method. You will be asked to confirm where you want this new method to go. Select “Ship” and then press “Choose.”

7. You will also be asked to select a category for your method. Type the letters “un”, select “as yet unclassified”, and then press “Choose or Add.” Confirm by pressing the “as yet unclassified” button. The method category does not affect anything with your code. These are also called “Protocols.” Protocols are used for convenient groupings of your methods which help to describe what your method does. You do not need to consider them right now.

8. You will now see the debugging window like the one below in Figure 2.



```
Ship>>moveUp
Ship>>keystroke:
ShooterGame>>handleKeystroke:
KeyboardEvent>>sentTo:
ShooterGame(Morph)>>handleEvent:
ShooterGame(Morph)>>handleFocusEvent:
MorphicEventDispatcher>>doHandlingForFocusEvent:with:
MorphicEventDispatcher>>dispatchFocusEvent:with:
```

```
moveUp
    self shouldBeImplemented
```

Figure 2: Debugger window

9. We will not discuss everything that can be done in the debugging window in this lesson. However, with this window you can explore, test, and code. This is a very powerful tool that allows you to accomplish much. It is not just a tool for when something wrong happens.

10. For now, you will only need to learn how to enter new code within the debugger. Entering your code will be very similar to adding code in the System Browser.

11. Type and save the following code in your debugger window. You will notice that the debugger has already added the method name for you.

```
moveUp

    self y: self y + 5.
    owner changed
```

12. After you have saved your code changes, press the “Proceed” button. This tells the debugger to continue running your program – as if an error did not even happen! Try to watch your ship when you do this.

13. You might not have noticed it, but your ship moved up once! Not only did your program not crash on you, it still handled the up arrow key being pressed, allowed you to correct an error, and then continue running as if no problem happened.

14. Press your up arrow key a few times to see your code change at work. Do not try the other arrow keys just yet. You will do those next.

15. Now follow the same process to use the debugger to create the other 3 ship movement methods. Remember to only press the arrow key *once*. If you press your arrow key more than once the predebugger will open the debugger window and you will then not have a “Create” button to use.

```
moveDown

    self y: self y - 5.
    owner changed
```

```
moveLeft

    self x: self x - 5.
    owner changed
```

moveRight

```
    self x: self x + 5.
    owner changed
```

16. Try out your code changes.

17. What do you notice about the ship and your shooter game? What else needs to be done?

18. Save and Quit your Smalltalk image.