ORACLE Academy

Using Variables and Keyboard Controls to Manipulate Motion

What Will I Learn?

Objectives

- Understand variables and how they are used in programming
- Define the value of a variable based on a math calculation
- Use keyboard controls to manipulate an animation
- Complete an animation
- Test an animation

Why Learn It?

Purpose

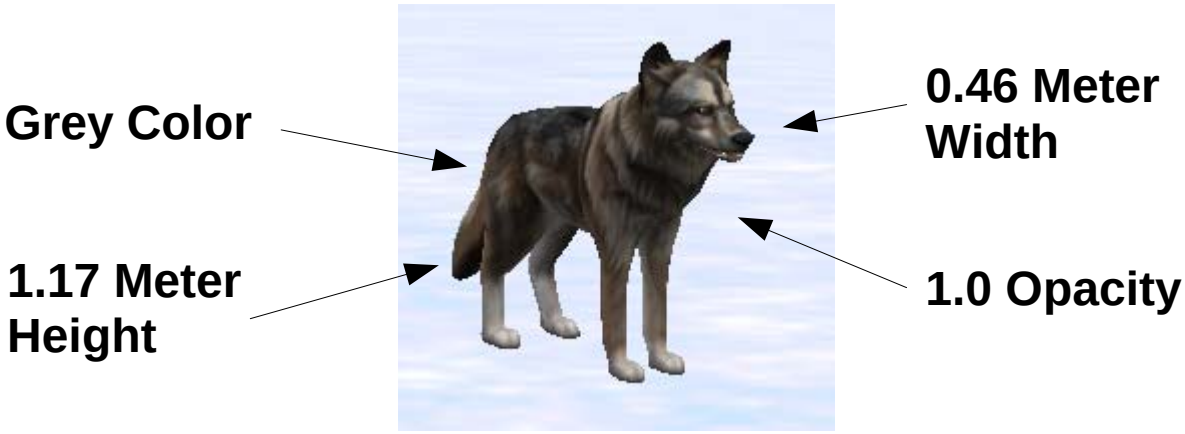
On occasion you may want to control the behavior of your objects using keys on the keyboard. For example you are directing an object through a game board and you need to use keyboard keys to move the object left, right, forward, or back. You may also want to store information about your objects to further control or influence animation movement.

Variables

There are times when we need to create a place to store information about objects in our animations. For example:

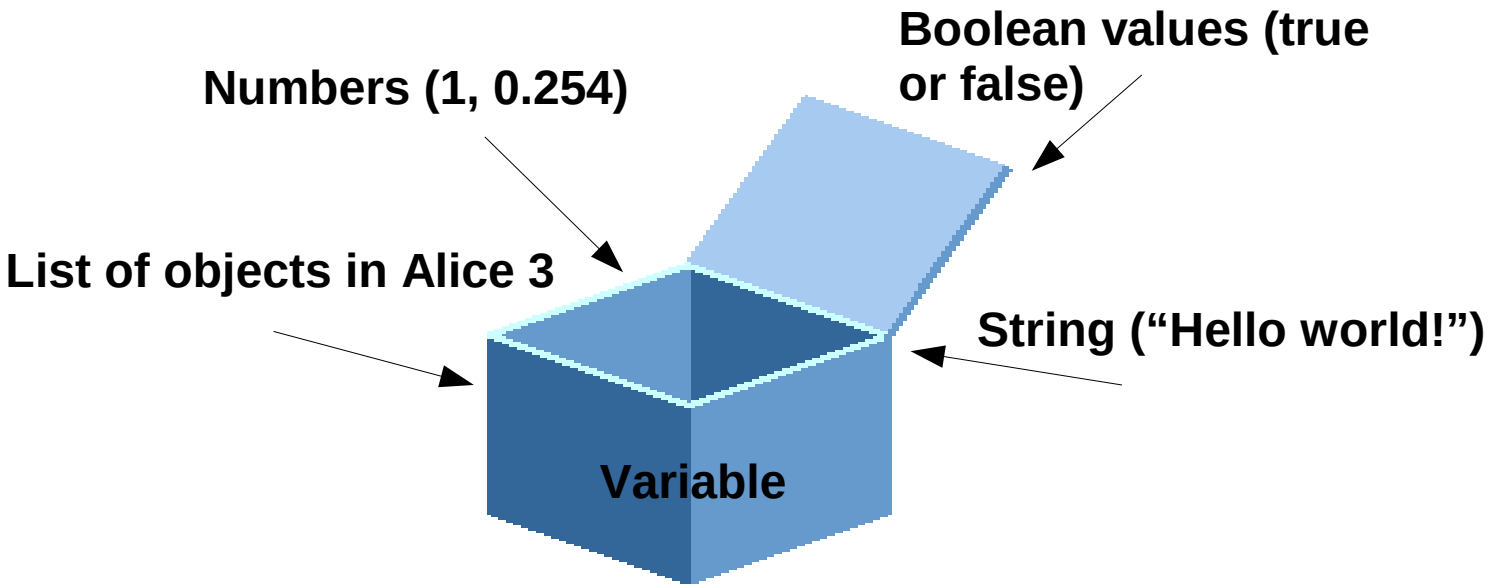
- The number of times a procedure should be executed
- An object's properties, such as its size and color

A variable is a place in memory where data of a specific type can be stored for later retrieval and use by your program. Each variable is given a unique name to make it easy to find. We can then use it to store and retrieve data.



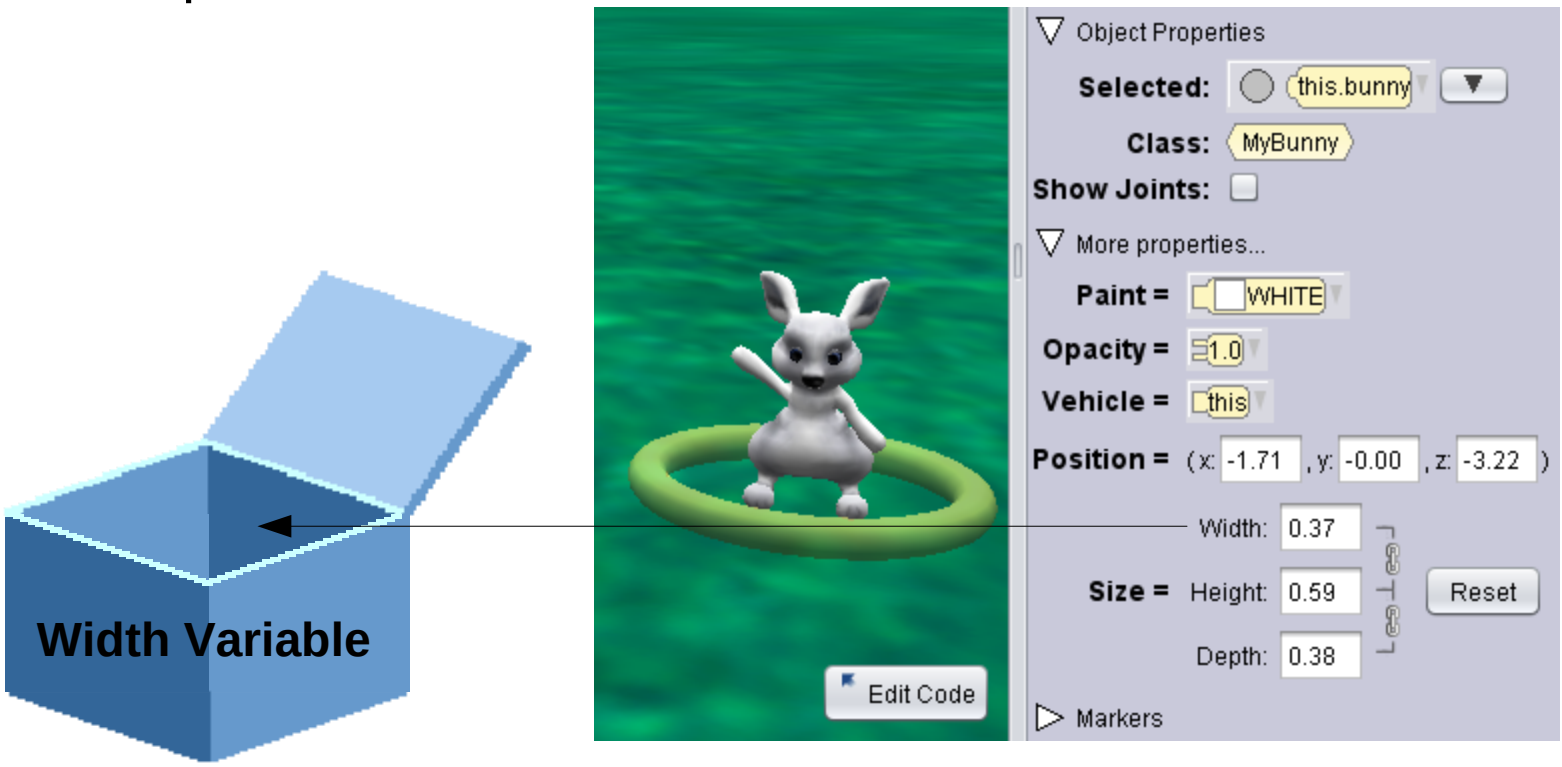
Variables (cont.)

A variable is like a container that stores a specific type of data for later retrieval and use by your program. You can declare variables by naming them and selecting the type of data to store in them.



Variables (cont.)

Object properties are also variables that store information about the object, such as an object's color, width, height, and depth.



Local Variables

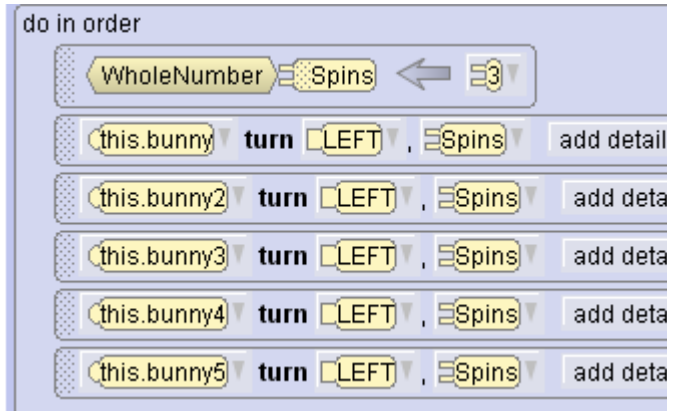
Local variables are declared (created) in your code. They are useful to declare when you need to:

- Assign the same value to multiple procedures, such as have a set of procedures move the same distance amount
- Be able to easily change that value so the change cascades to all of the procedures using it in the program

Local Variables (cont.)

For example, you may create a Spins variable that is initialized to a whole number value, 3.

This variable is then dragged into distance argument of each bunny's turn procedure so that each bunny spins around three times.



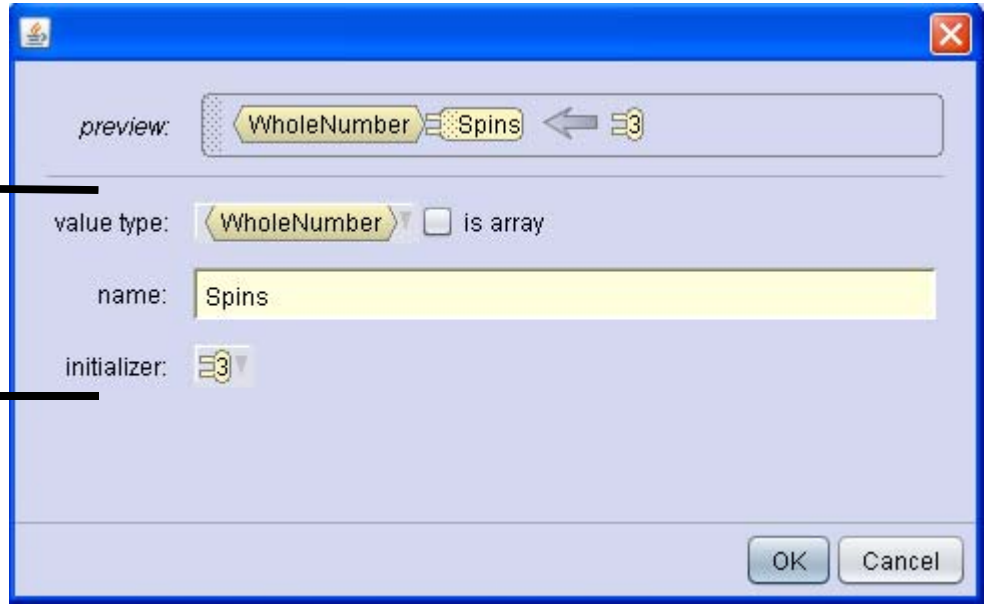
If you want the bunnies to spin more, you can change the initialized value of the Spins variable, and all of the procedures will automatically update with that value.

Local Variables (cont.)

When you create a variable you specify the variable type, the name of the variable, and the initial value for the variable.

The example below shows you that the preview section of the dialog box is indicating that you've created a variable with an whole number type, named Spins, that has an initial value of 3.

Area where you specify the variable information

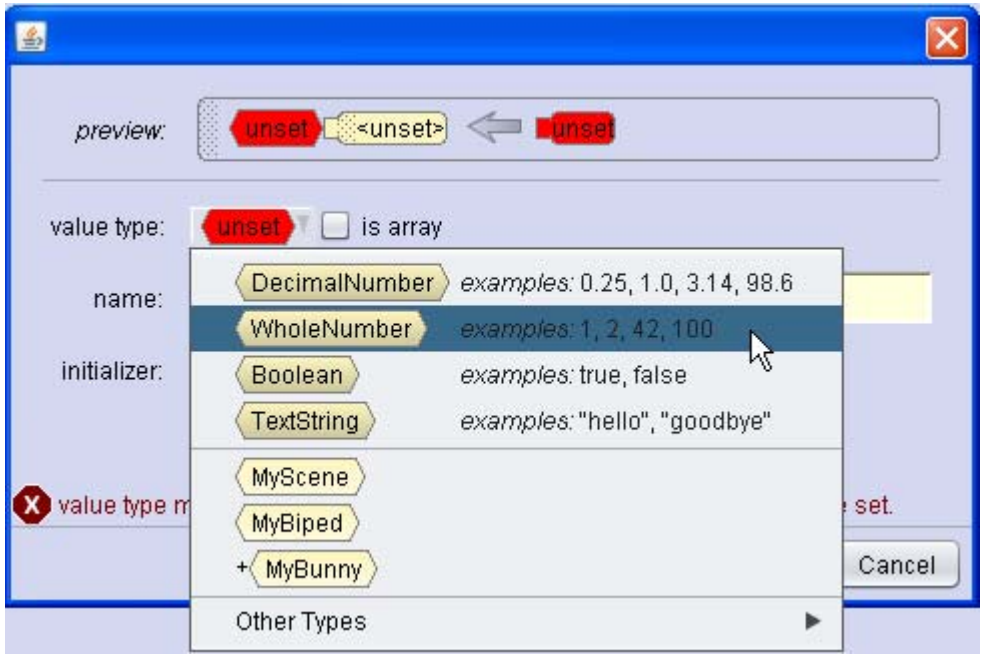


Local Variables (cont.)

Steps to create a local variable:

1. Drag the local tile into the code editor. 

2. Select the value type.



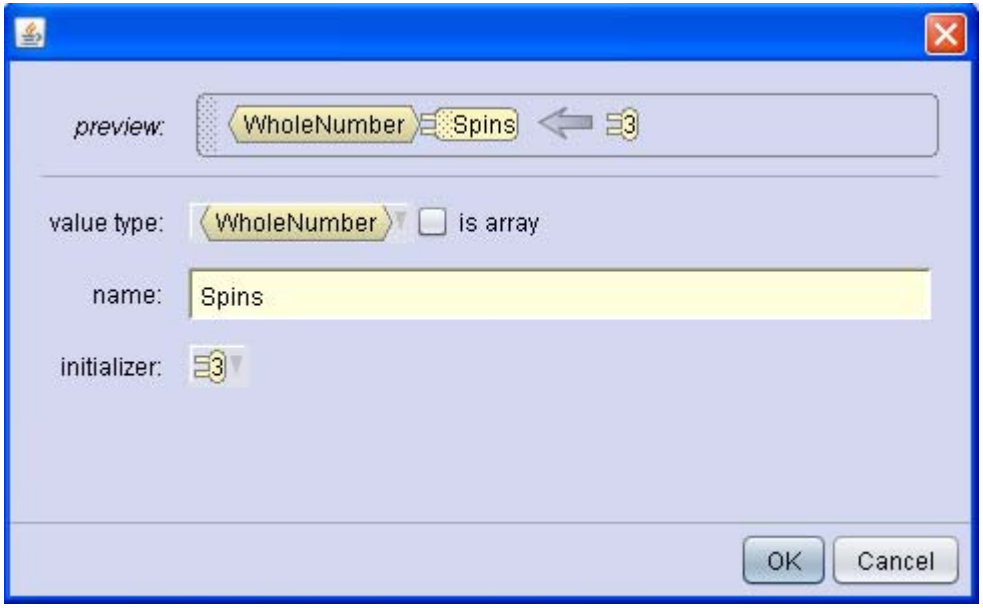
Local Variables (cont.)

Alice supports several types of data.

Data Type	Description
Decimal Number	Used to perform arithmetic and to set the value of a procedure's arguments. Examples: 0.1, 2.25, 98.6.
Whole Number	Used to perform arithmetic and to set the value of a procedure's arguments. Examples: 1, 459, 30.
Boolean	One of two values, true or false; usually data of this type is the result of tests that compare one thing to another.
Object	Any object in Alice 3 like a cat, dog, person, etc.
Classes	The classes of objects in your animation. Examples: MyBiped, MyScene, MyQuadruped.
TextString	A String of characters such as “hello” and “goodbye”.
Other	Things like sounds, colors and other special values.

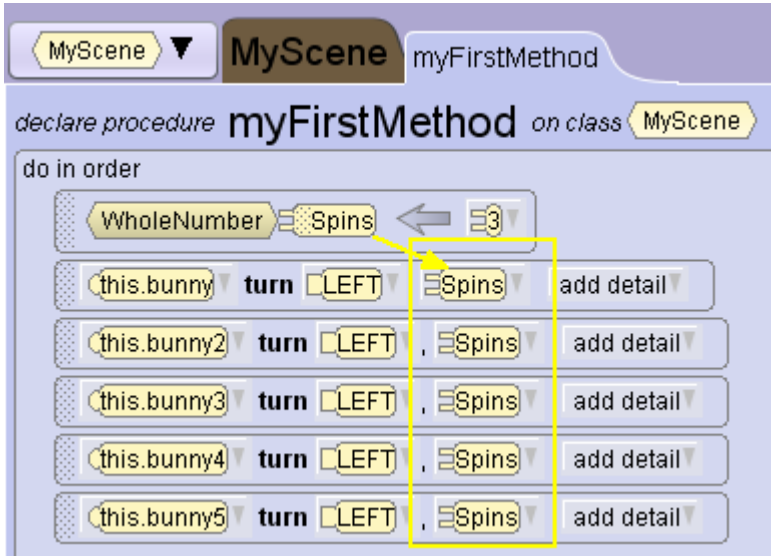
Local Variables (cont.)

3. Name the variable.
4. Select the value to initialize (set) the variable to.
5. Click OK.



Local Variables (cont.)

6. For the existing procedures in your code, drag the local variable name tile (Spins in the example below) into the distance arguments.

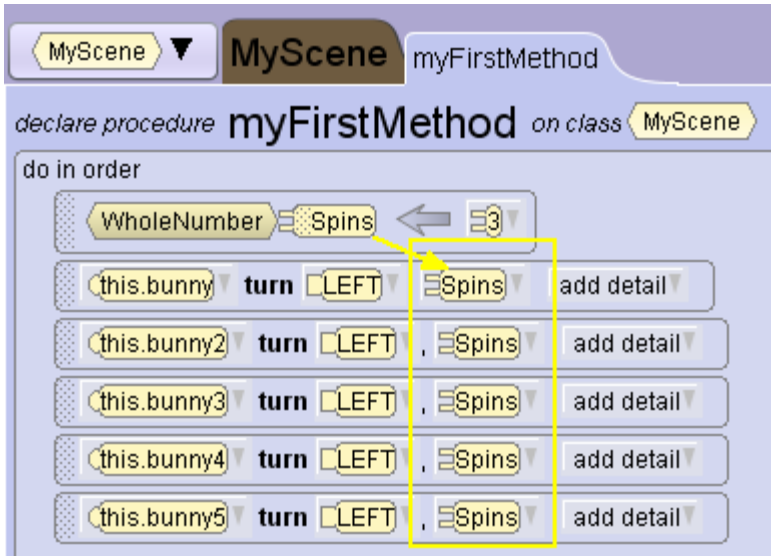


All of the bunnies will turn the distance specified in the Spins variable.

Local Variables (cont.)

You may then decide to change the distance that one of the objects move. For example, the bunny2 object should spin two more revolutions than the other bunnies.

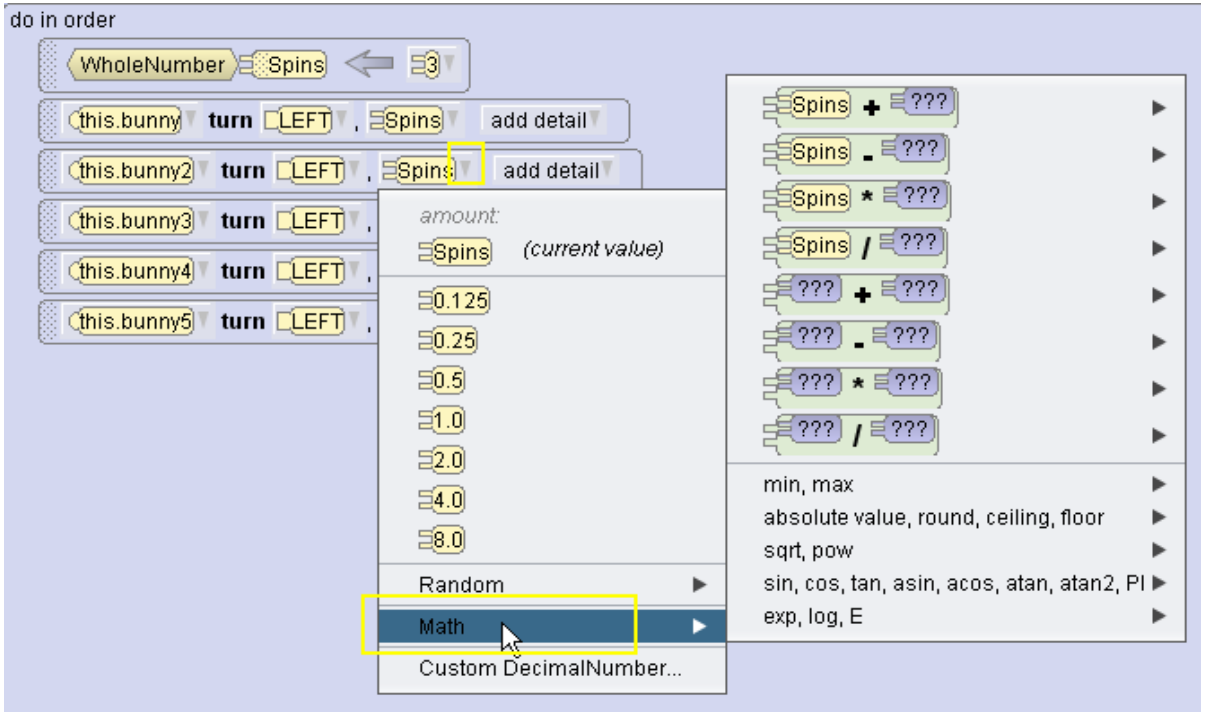
A math calculation can be used to change the value of the Spins argument for this object.



Local Variables (cont.)

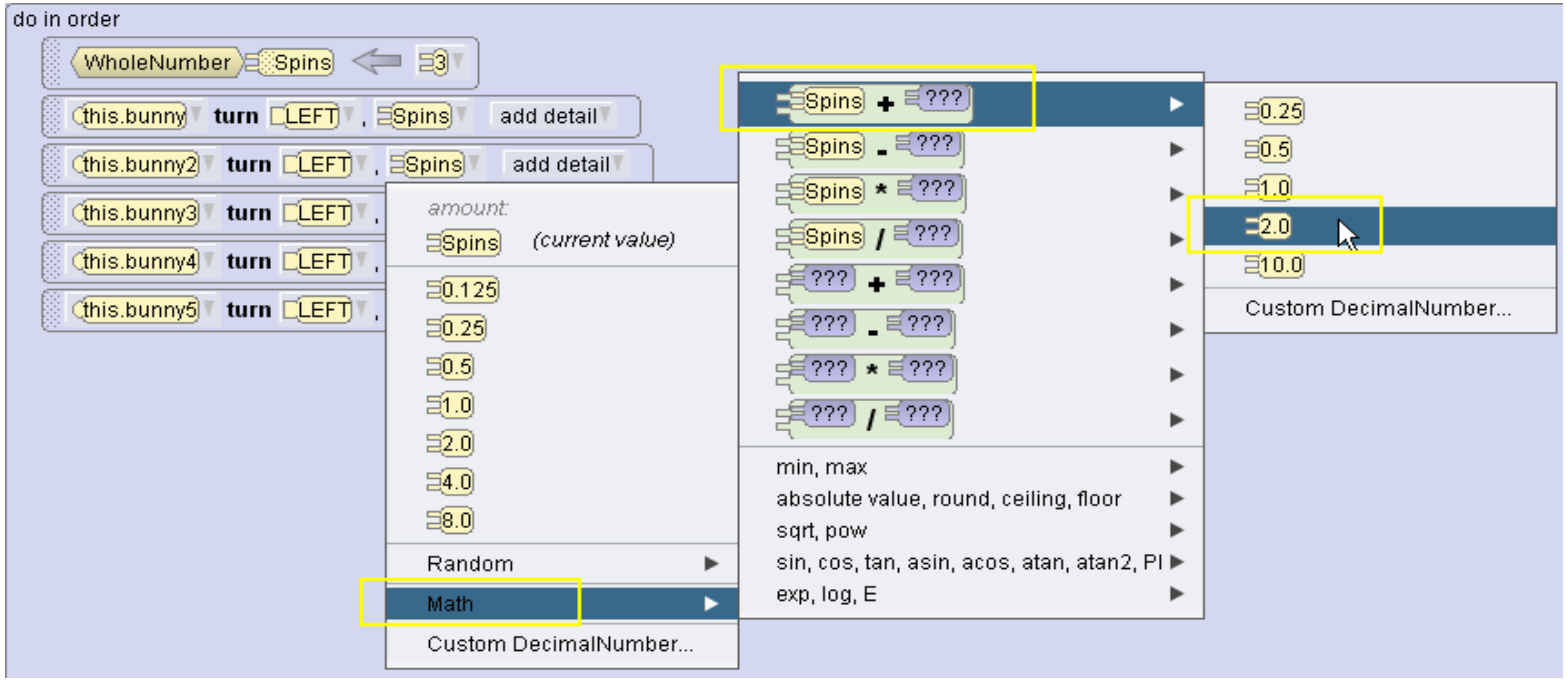
Use the following steps to change an argument using a variable and a math calculation.

1. In the selected procedure, right-click on the arrow next to the argument.
2. Select Math.



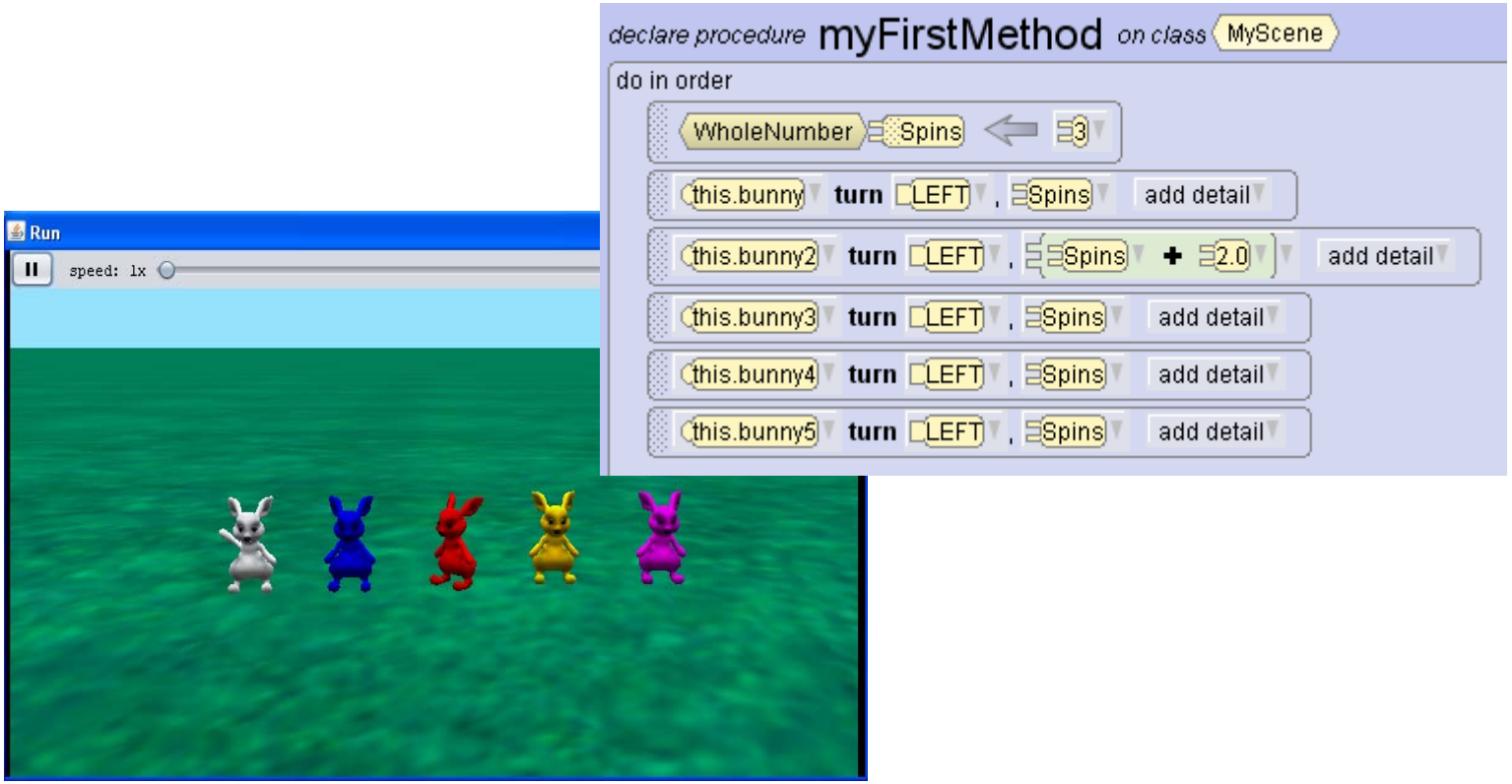
Local Variables (cont.)

3. Select the math operator.
4. Select the value for the math calculation.



Local Variables (cont.)

5. Run the animation to test. Revise the calculation as needed.



Keyboard Controls

To make your animations more interactive, you can insert keyboard controls to allow the viewer to control one or more objects in the animation.

With keyboard-controlled animations, you may:

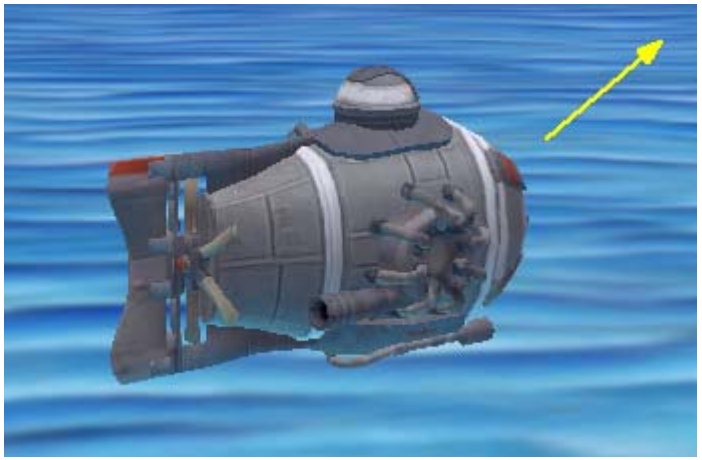
- Create animated scenes where the user controls an object to interact with other objects.
- Create animated games where the user is required to control an object to win the game.

Keyboard Controls Example

In Alice 3, you can associate procedures to keys on your keyboard. When the viewer clicks a keyboard key, the procedure assigned to the keyboard key is executed.

In programming, keystrokes, or mouse clicks, are an event. Coding the events to handle each procedure is referred to as event handling.

For example, in a submarine animation, selecting the right arrow key on your keyboard will turn the submarine right.



MyScene Activation Listeners

Event listeners are established to listen for keyboard input. Animation tasks can be associated with a keystroke. When the event listener hears keyboard input the associated animation tasks are executed.

Event listeners are created at the MyScene class level. You select the MyScene tab to add an event listener to the Scene Activation Listener. There are 4 event listener types:

1. Scene Activation / Time
2. Keyboard
3. Mouse
4. Position/Orientation

MyScene Keyboard Listener

There are 4 keyboard listener types:

1. addKeyPressListener()
2. addArrowKeyPressListener()
3. addNumberKeyPressListener()
4. AddObjectMoveFor(???)

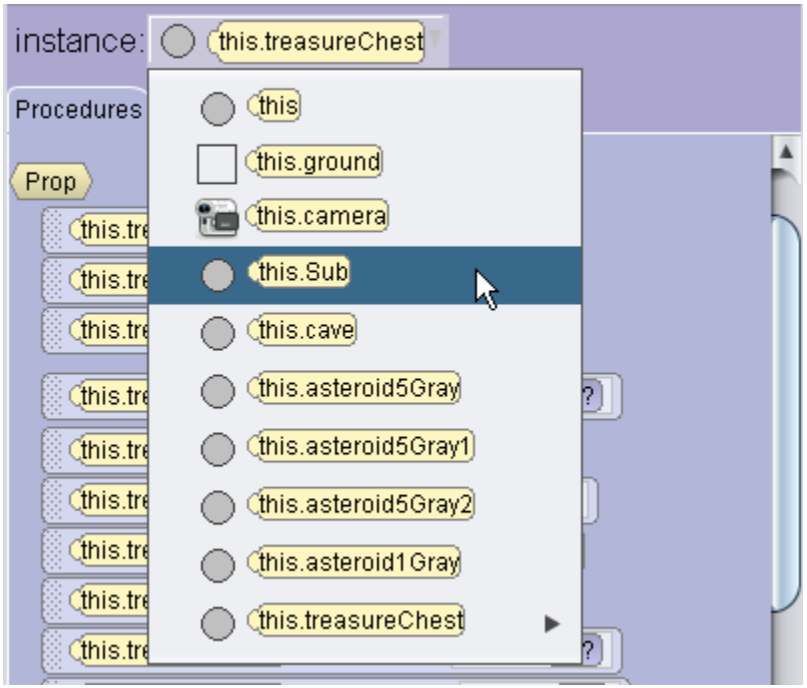
To add an addKeyPressListener() to listen for, and react to, a keyboard keystroke you specify the listener type and then add the programming arguments for the listener construct.

A listener construct requires you to specify a conditional statement that can be read as “if the true condition of the x key being pressed exists, then do something”.

Keyboard Listener Steps

Steps to assign a procedure to a keyboard listener are listed below.

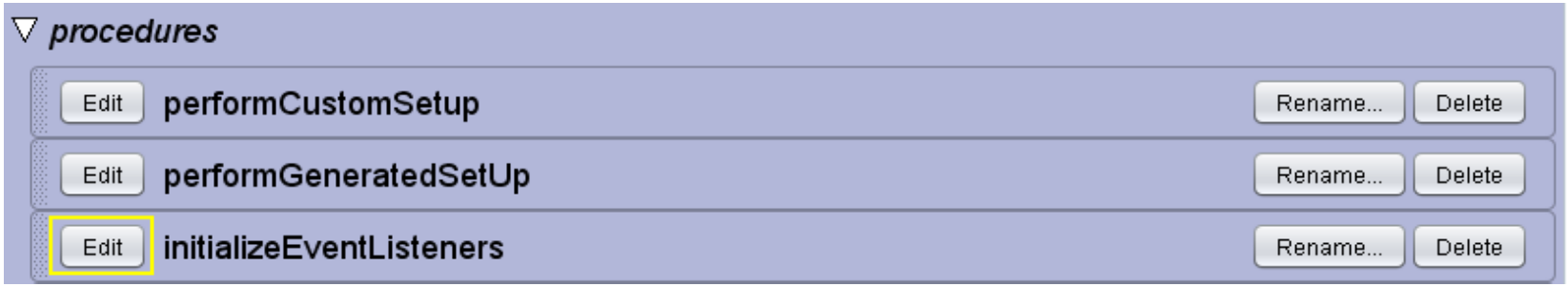
1. In the Instance menu in the code editor, select the object to control.



Keyboard Listener Steps (cont.)

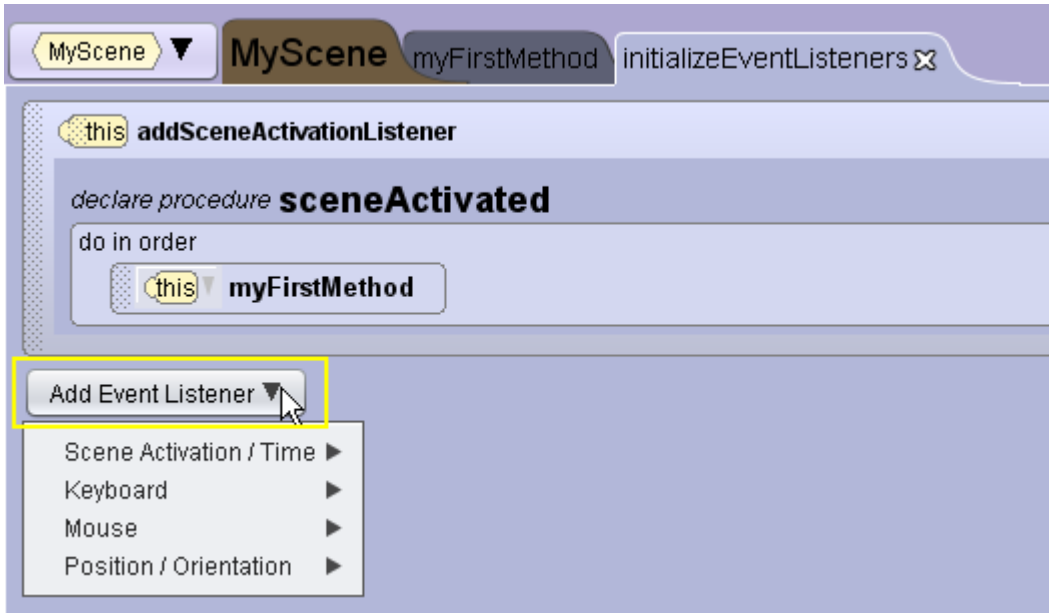
2. Click the MyScene tab.

3. From the Procedures area, click the Edit button next to InitializeEventListeners.



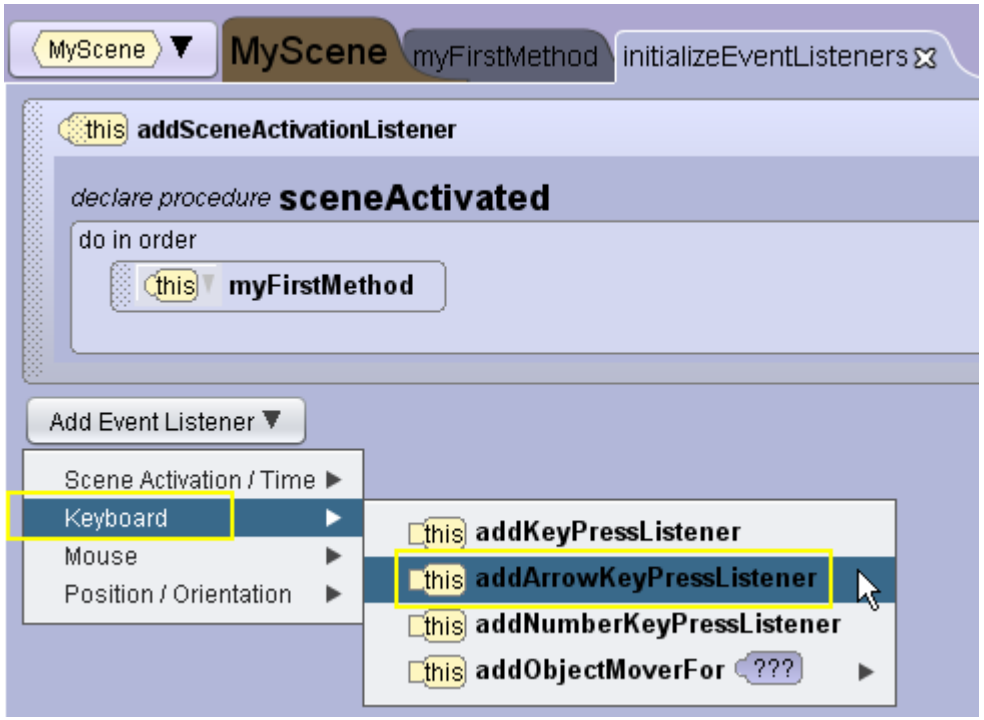
Keyboard Listener Steps (cont.)

4. In the InitializeEventListeners tab that opens, click the Add Event Listener button.



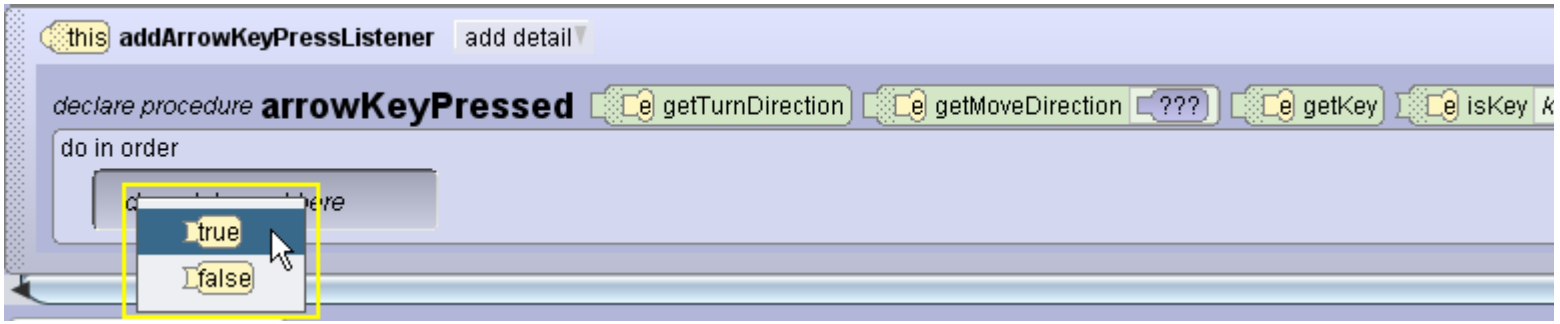
Keyboard Listener Steps (cont.)

5. Select Keyboard.
6. Select addArrowKeyPressListener.

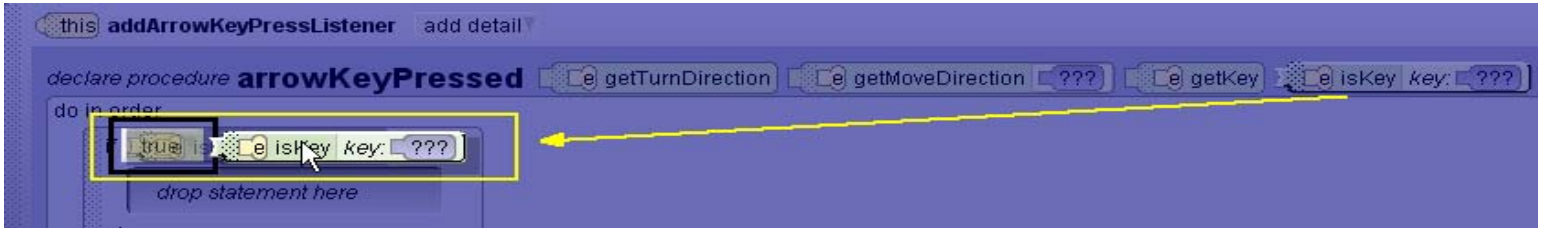


Keyboard Listener Steps (cont.)

7. Drag If control statement tile onto “drop statement here” field, and select True condition.

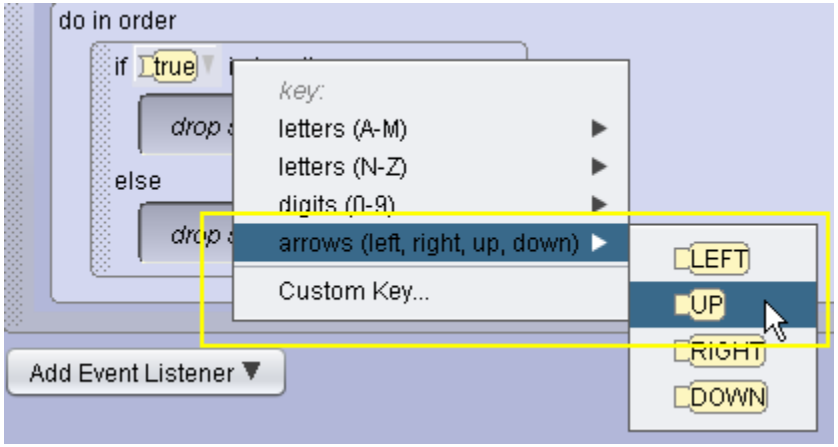


8. Drag “e isKey: ???” procedure onto True condition.



Keyboard Listener Steps (cont.)

10. Select arrows, then Up.

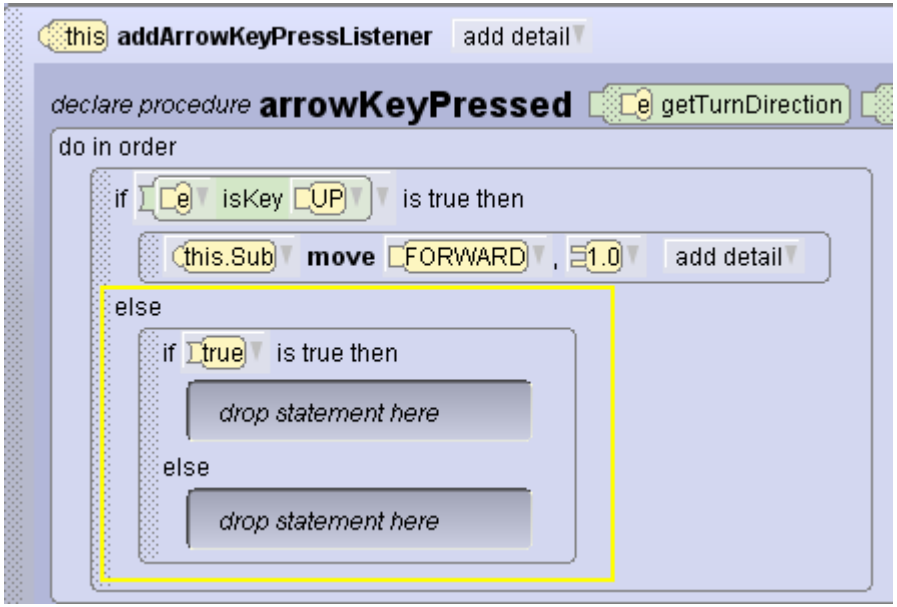


Keyboard Listener Steps (cont.)

11. Drag a move procedure into the If condition. Select Forward for direction and 1 meter for amount.

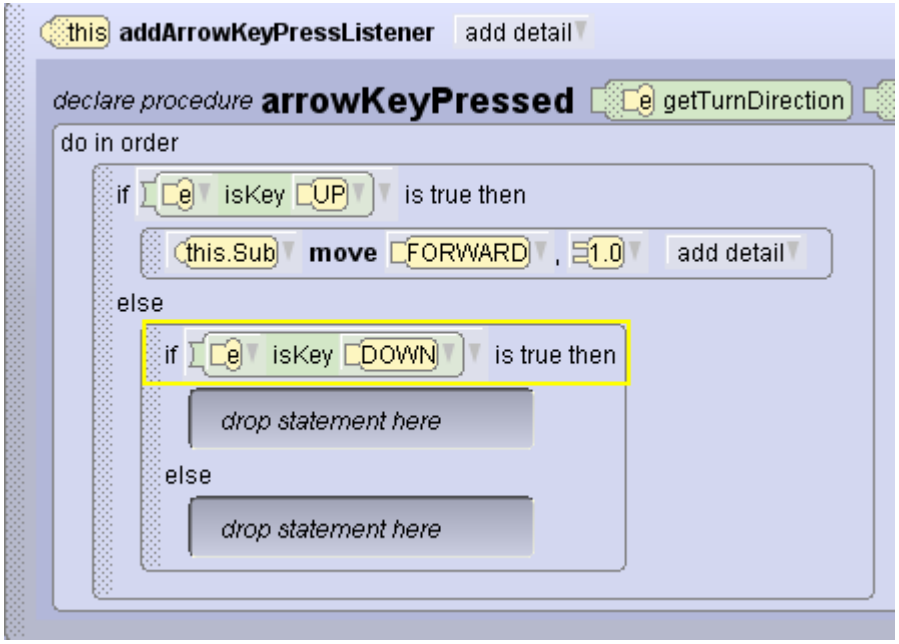
Keyboard Listener Steps (cont.)

12. Drag another If control onto the Else argument.
13. Select True condition.



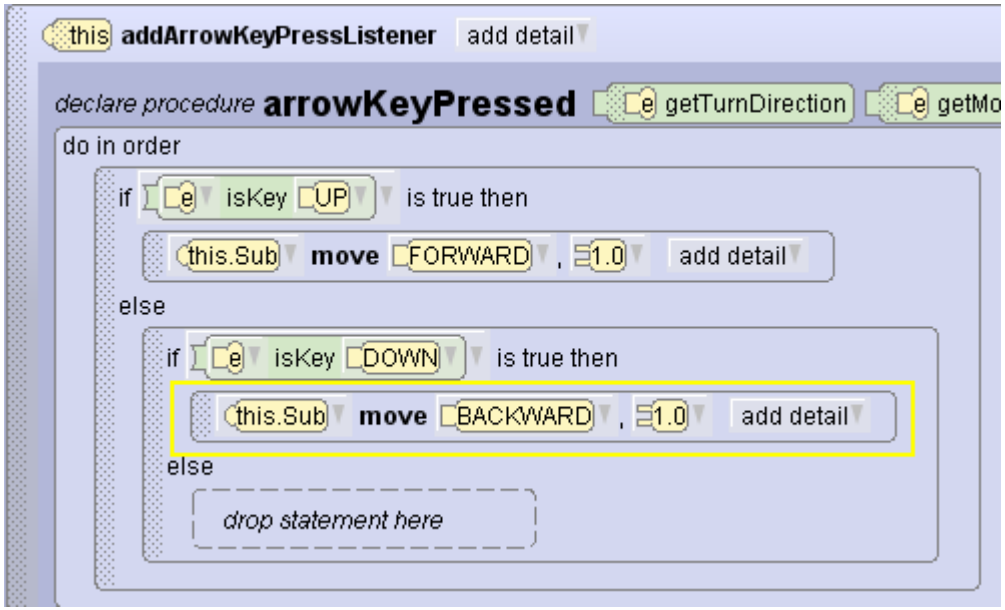
Keyboard Listener Steps (cont.)

14. Drag “e isKey: ???” procedure onto True condition.
15. Select arrows, then Down.



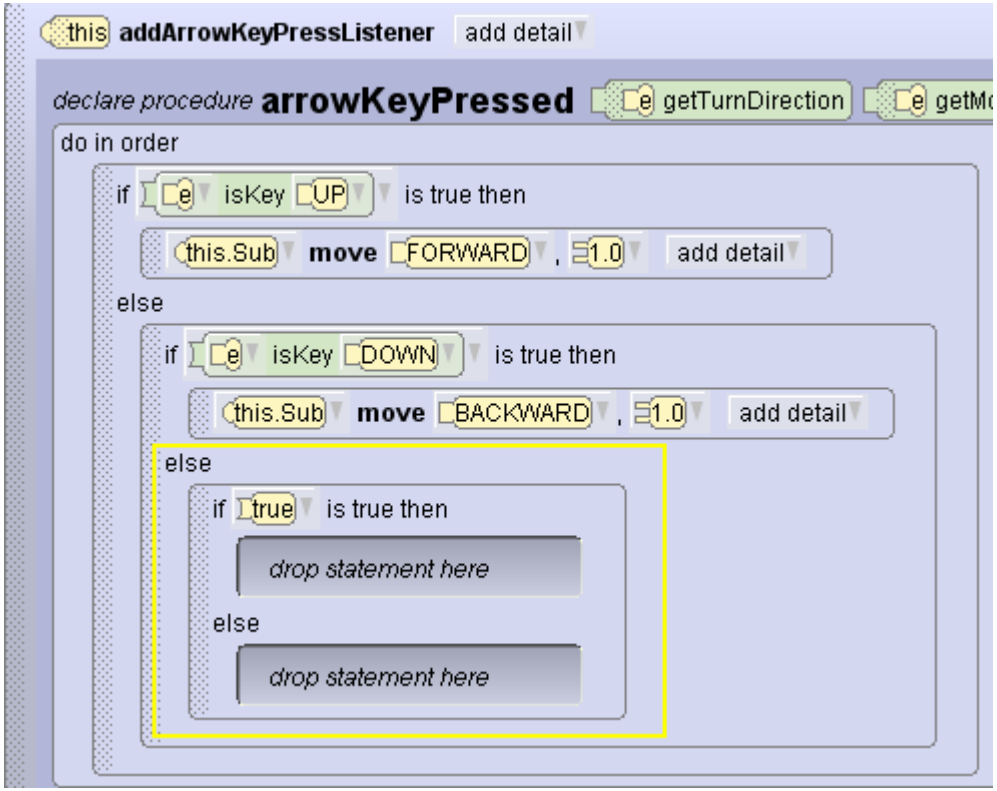
Keyboard Listener Steps (cont.)

16. Drag a move procedure into the If condition. Select Backward for direction and 1 meter for amount.



Keyboard Listener Steps (cont.)

17. Drag another If statement onto the Else argument.
18. Select True condition.

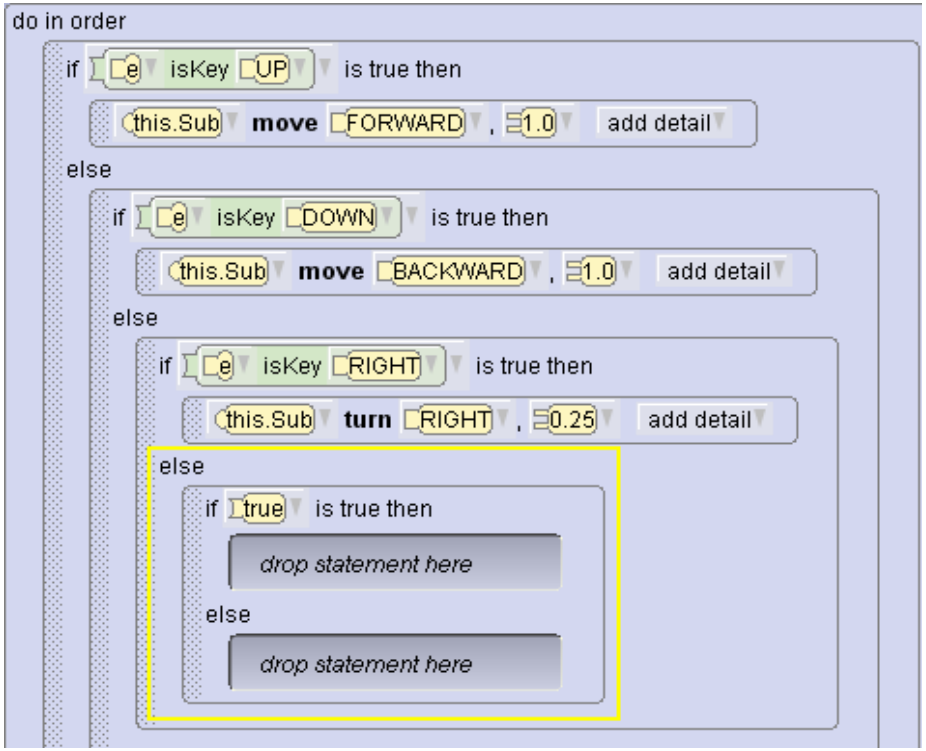


Keyboard Listener Steps (cont.)

19. Drag “e isKey: ???” onto True condition.
20. Select arrows, then Right.
21. Drag a turn procedure into the If condition. Select Right for direction and 0.25 for amount.

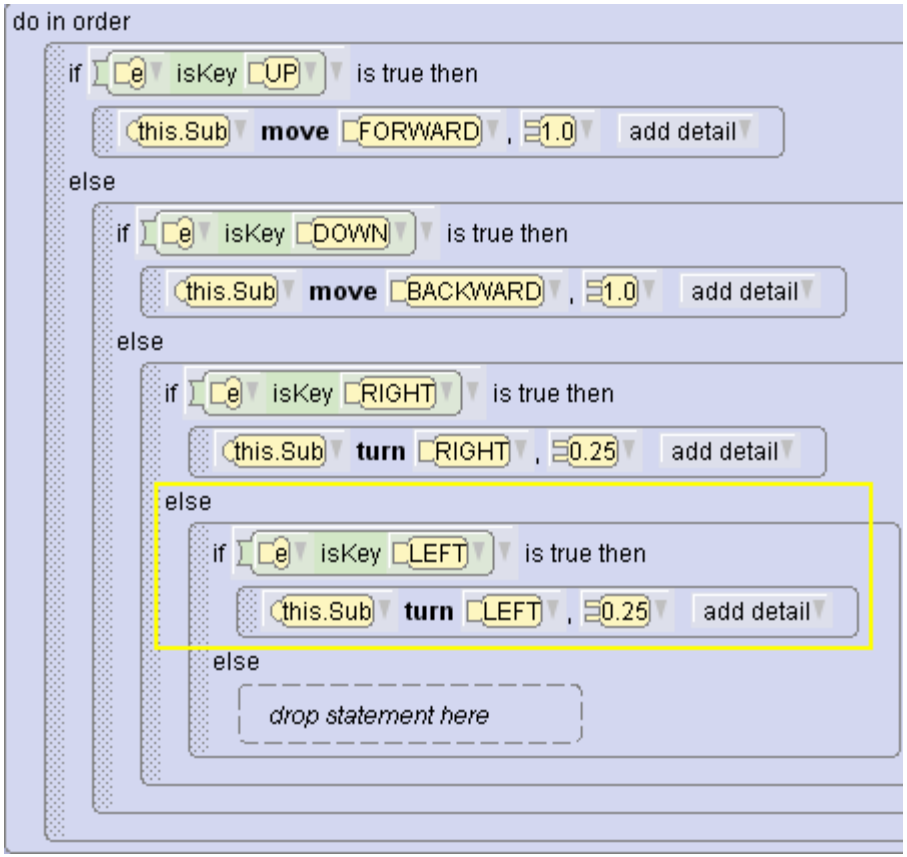
Keyboard Listener Steps (cont.)

22. Drag another If statement onto the Else argument. Select True condition.



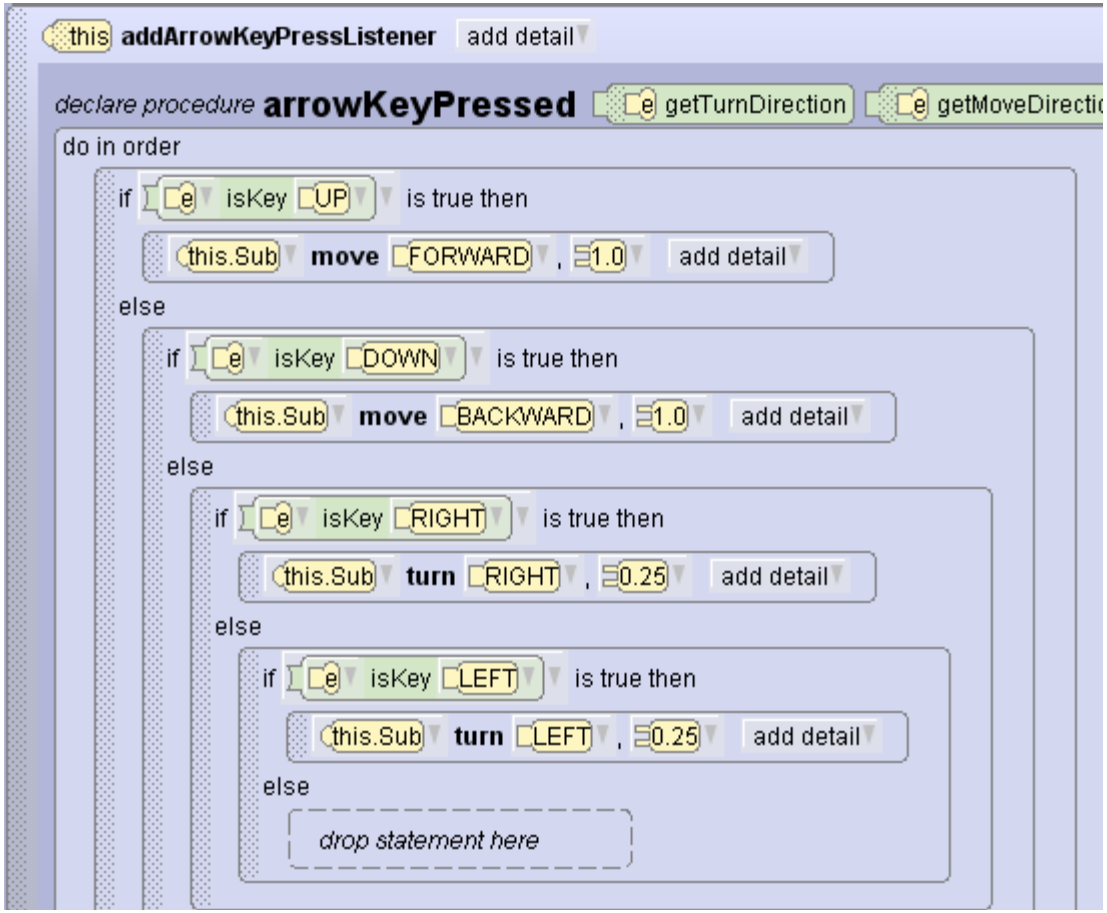
Keyboard Listener Steps (cont.)

23. Drag “e isKey: ???” onto True condition.
24. Select arrows, then Left.
25. Drag a turn procedure into the If condition. Select Left for direction and 0.25 for amount.



Keyboard Listener Steps (cont.)

The submarine is now programmed to move left, right, forward, and backward using the arrows keys on a computer keyboard.



Keyboard Listener Steps (cont.)

But what about moving up and down? Since there are only four arrow keys on a keyboard, we need to program two other keys for the up and down motion.

Steps:

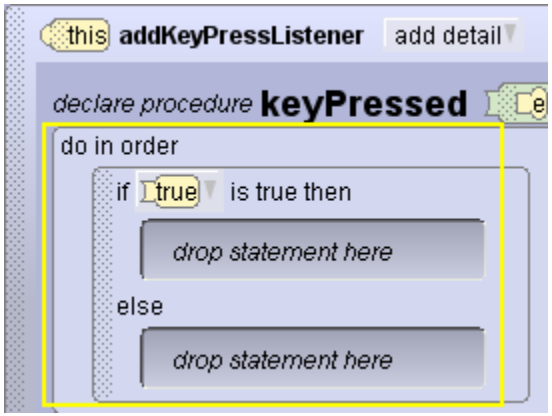
1. Click the Add Event Listener button. 

2. Select Keyboard, then addKeyPressListener.

Keyboard Listener Steps (cont.)

Steps (cont.):

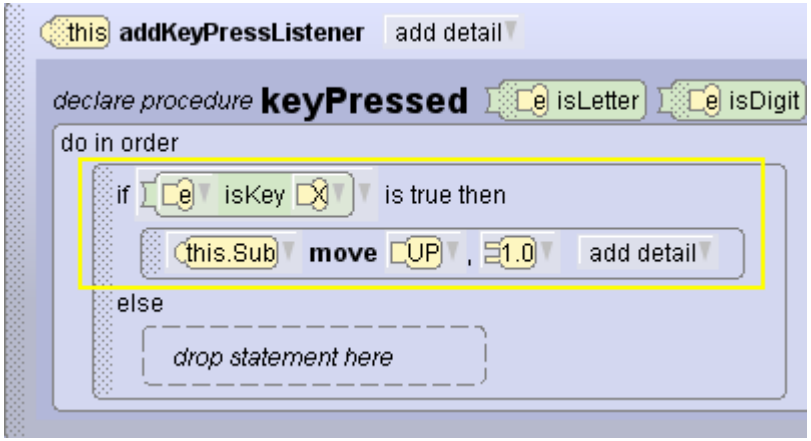
3. Drag If control statement tile onto “drop statement here” field, and select True condition.



Keyboard Controls (cont.)

Steps (cont.):

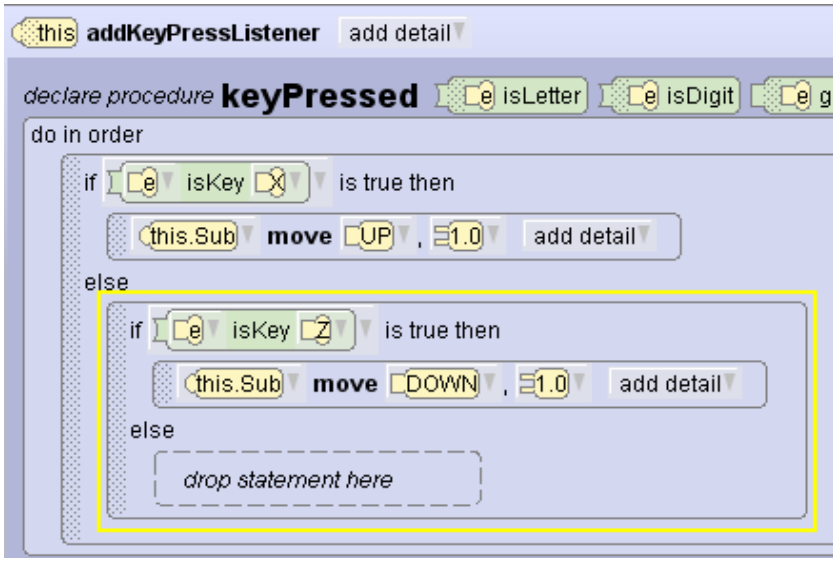
4. Drag “e isKey: ???” onto True condition.
5. Select letters N-Z, then X.
6. Drag a move procedure into the If condition. Select Up for direction and 1 meter for amount.



Keyboard Controls (cont.)

Steps (cont.):

7. Drag another If statement into the Else argument.
8. Select True condition.
9. Drag “e isKey: ???” onto True condition.
10. Select letters N-Z, then Z.
11. Drag a move procedure into the If condition. Select Down for direction and 1 meter for amount.



Keyboard Controls (cont.)

Click the Run button to run the animation, then click your cursor on the Run window. The object will stand still until the keyboard keys are used to make it move.

Animation Checklist

During the animation development process, use an animation checklist to ensure that your animation meets all animation principles.

Checklist for Animation Completion

Area	Check	☒
Scenario	Did you clearly define the scenario?	
Storyboard	Did you think-through the animation by creating a storyboard?	
Textual storyboard	Did you think-through the programming code by creating a textual storyboard?	
Program	Did you complete the programming of the animation?	
[illegible]	[illegible]	

Terminology

Key terms used in this lesson included:

Animation checklist

Event

Event handling

Keyboard controls

Variables

Summary

In this lesson, you learned how to:

- Understand variables and how they are used in programming
- Define the value of a variable based on a math calculation
- Use keyboard controls to manipulate an animation
- Complete an animation
- Test an animation

Practice

The exercises for this lesson cover the following topics:

- Using math and variables to control movement
- Manipulating movement using the keyboard
- Identifying components of a completed animation