Alice 3

How-to Guide

(Part 3 – Working with the Code editor)

Wanda Dann
Don Slater
Laura Paoletti
Dennis Cosgrove
Dave Culyba
Pei Tang

Cover artwork by Laura Paoletti, 2012.

Working with the Code editor

16. CODE EDITOR TABS – SCENE CLASS

The purpose of this section is to demonstrate the purpose of the default Code editor tabs, which display the Scene class and two of its procedural methods (*initializeEventListeners* and *myFirstMethod***)**. For illustration, we will use an African themed scene, as shown in Figure 16.1. The scene includes a variety of props (grasses, rocks, trees, termite mound, and a pond) as well as elephant, ostrich, and baby ostrich objects.

Figure 16.1 African scene with elephant, ostrich, and baby ostrich

When a project is first opened in Alice, the Code editor automatically displays three default tabs, as shown in Figure 16.2. The first tab is the **Scene** class document tab, which is accompanied by two procedural method tabs: *initializeEventListeners* and *myFirstMethod*. The Scene's *myFirstMethod* tab is automatically selected as the active editor tab. Each of these tabs is briefly described below.

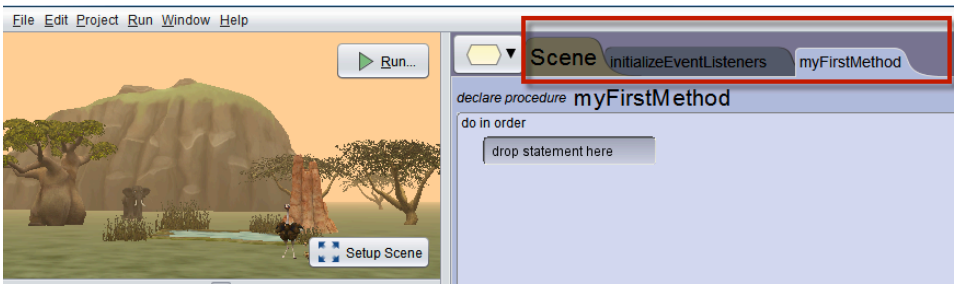

Figure 16.2 Code editor tabs for the Scene class

Scene class document tab

A **class** is a **document file** that defines how to create an instance of a specific type of object. The Scene class defines how to create a scene for a virtual world. A **class document** may also contain definitions for procedures (action methods), functions (computational methods), and properties (fields – objects or items of data, such as color or opacity). Figure 16.3 shows the Scene class document tab for the example world used in this section.

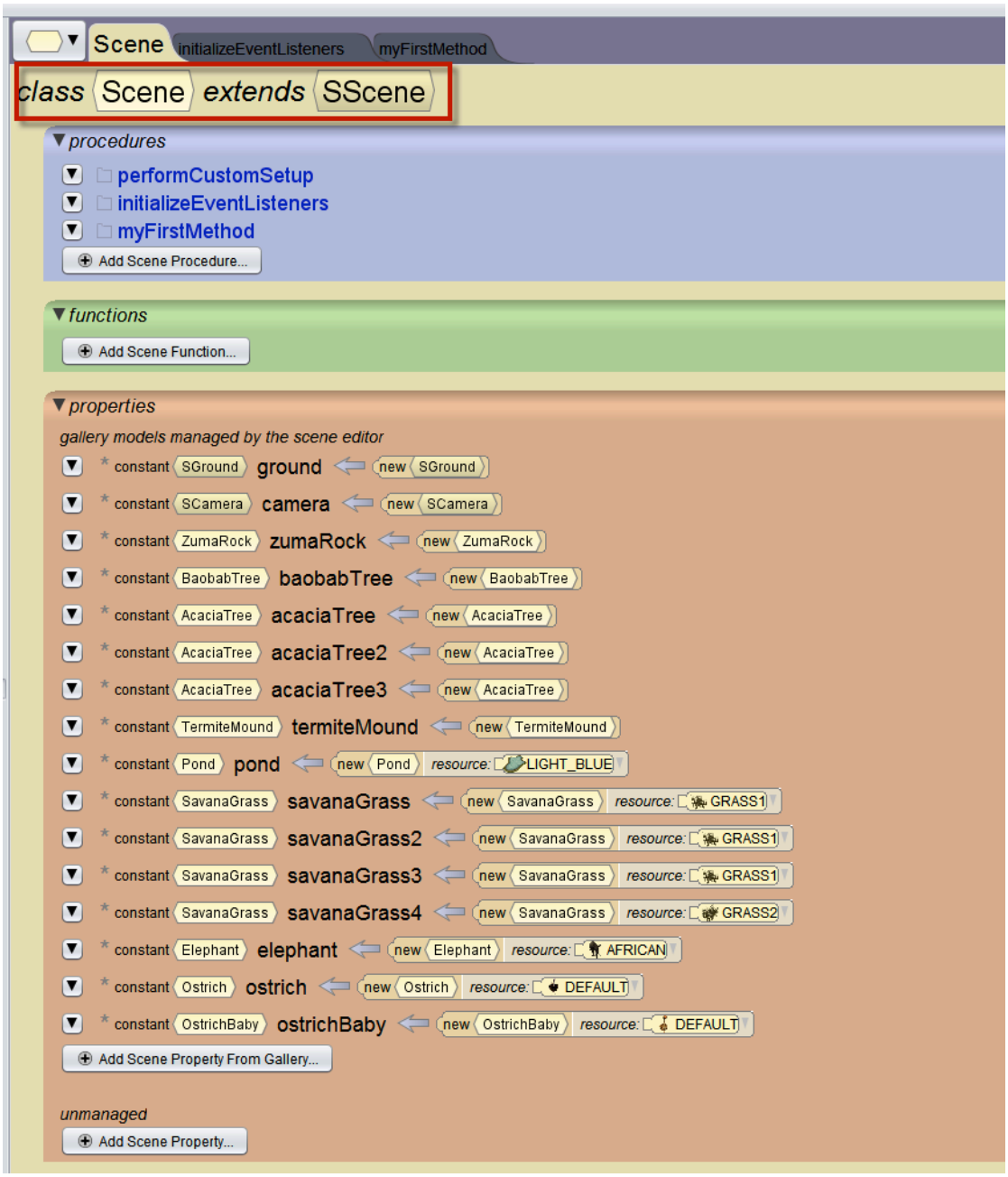

Figure 16.3 Scene class tab in this example world

initializeEventListeners tab

The *initializeEventListeners* tab displays the current content of the Scene class' *initializeEventListeners* procedure, as shown in Figure 16.4. The Scene's *initializeEventListeners* procedure is similar to an Events editor, where **listeners for specific events** may be created**.**

To explain listeners and events, let's use real world analogies. An **event** is something that happens. It may be a huge event such as a football game held in an arena or a very small event such as the blink of an eye or a mouse-click. A **listener** is an alert that tells Alice to listen for an event. A listener acts like an alarm clock. When an alarm clock is set, the clock keeps watch (listens) for a specific time to occur. When the time event occurs, the alarm clock starts a buzzer or turns on the radio to a selected channel.

Similarly, in interactive computer programs (for example, tutorials and games), we make use of events and listeners. An event can be things like a mouse click on a button, a key press on a keyboard, or a touch on the monitor. An event listener keeps watch for the event to occur and then responds to the event in some way.

By default, the Scene class' *initializeEventListeners* tab has one built-in event listener, named *addSceneActivationListener*, which tells Alice to listen for a mouse-click on the Run button. When the Run button is clicked, a runtime window is displayed and the current scene becomes active. When this event occurs, *myFirstMethod* is called (executed).

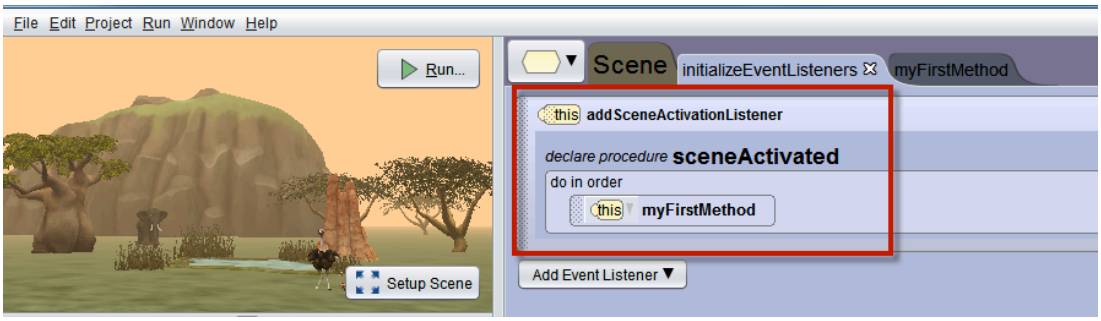

Figure 16.4 The Scene's initializeEventListeners tab

Additional event listeners may be created in the *initializeEventListeners* tab. To add a new event, click the **AddEventListener** button. A popup menu is displayed where four different kinds of event categories are displayed, as shown in Figure 16.5. The event categories are **Scene Activation/Time, Keyboard, Mouse,** and **Position/Orientation**. Each category has several options of listeners that may be selected. Figures 16.6-16.9 show the event listener options for each category, one at a time.

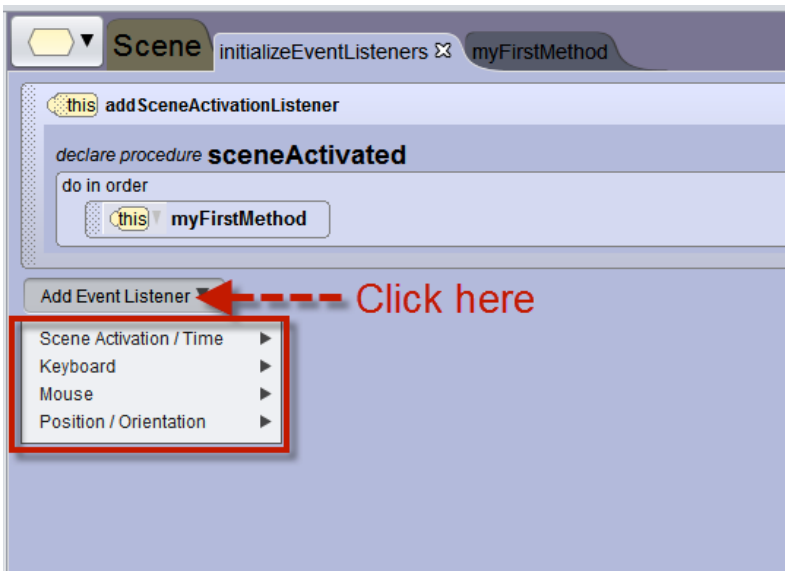

Figure 16.5 Event listener categories menu

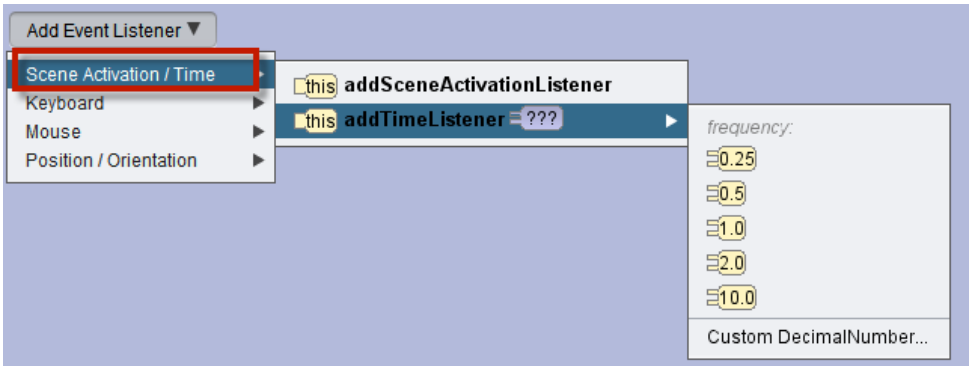

Figure 16.6 Scene Activation and Time event listeners

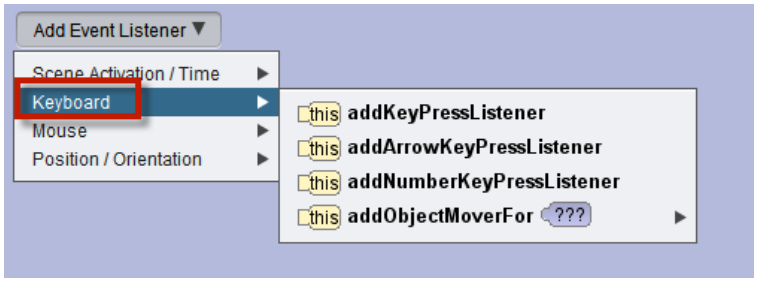

Figure 16.7 Keyboard event listeners

Figure 16.8 Mouse click event listeners

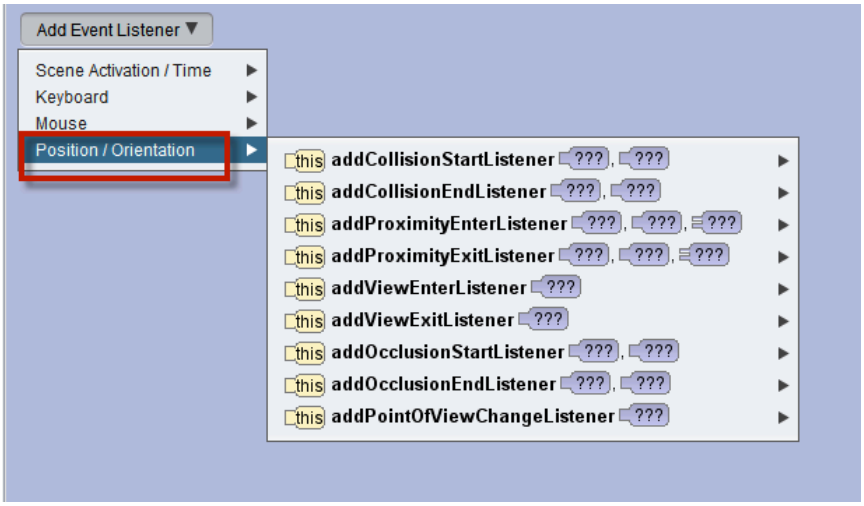

Figure 16.9 Position and Orientation event listeners

As an example of adding an event listener, suppose we want to allow the user to move objects around in the scene. As shown in Figure 16.10, select the **Mouse** event listener category and then **addDefaultModelManipulation** in the cascading menu.

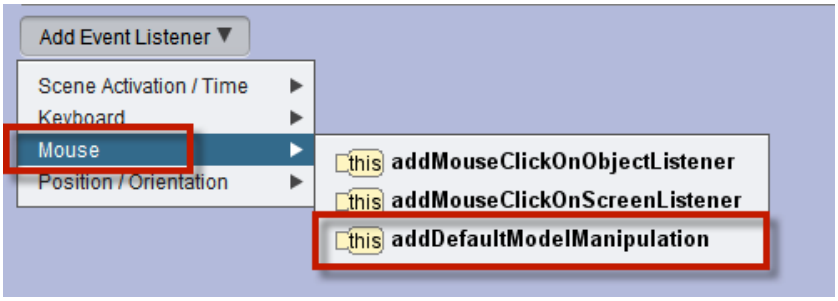

Figure 16.10 Example--Select **Mouse/addDefaultModelManipulation** *listener*

The resulting event listener can be seen in Figure 16.11. This event listener watches for a mouse click-and drag action. Any object within the scene can be pulled around the scene as the animation is running. For example, the mouse could be used to move the elephant around in the scene shown in Figure 16.11.

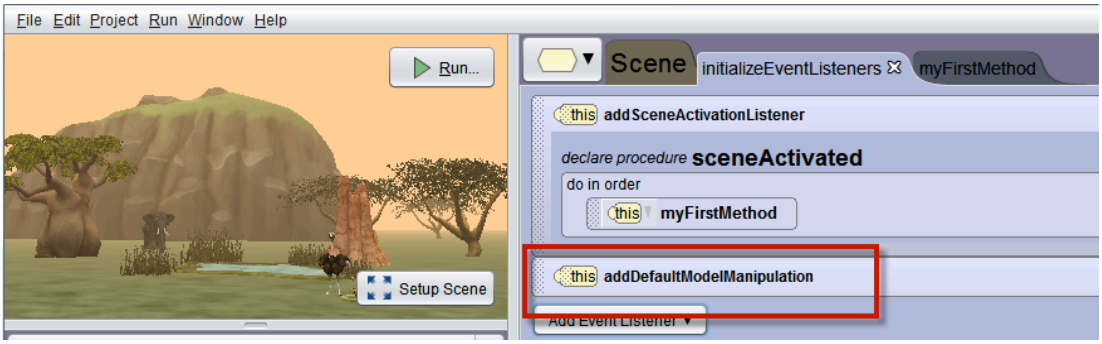

Figure 16.10 **addDefaultModelManipulation** *event listener code*

myFirstMethod tab

The *myFirstMethod* tab is where program you will likely create program statements that you expect to be performed when the Run button is clicked. Built into *myFirstMethod* is a first line of code (do in order), as shown in Figure 16.11. Do in order is a control statement that tells Alice to perform any statements in *myFirstMethod* in the order in which they listed (one at a time, one after the other). In the example shown in Figure 16.11 no further program statements have yet been added to *myFirstMethod.*

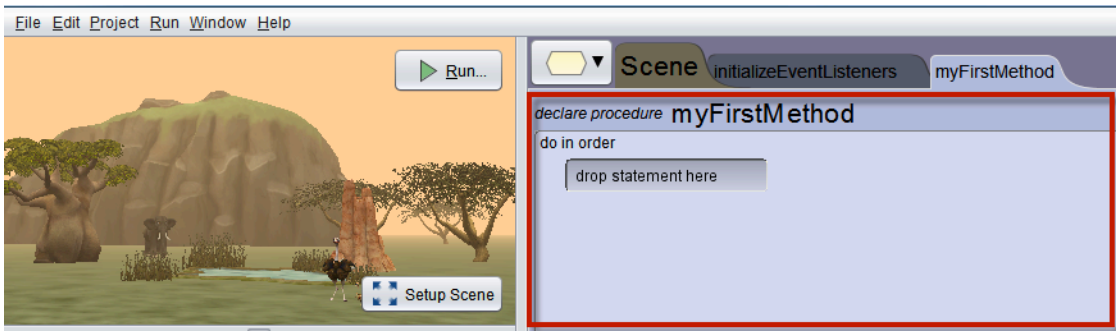

Figure 16.11 The Scene's myFirstMethod tab

17. How to work with a class (other than Scene)

As illustrated in Section 16 above, the default class in the Code editor is the Scene class. What if you want to create program code for a different class? For example, suppose you want to write code for the Elephant class to teach an elephant object how to flap its ears.

Open a Class tab (in addition to the default Scene tab)

As previously described in Part 1, Section 5 of this How-To guide, a Class tree menu button is provided (just to the left of the Scene class tab). For your convenience, selecting a class is illustrated again in Figure 17.1. Click on the class name in the tree and also in the cascading menu. In this example, the Elephant class is selected and Alice responds by opening the Elephant class tab in the editor, as shown in Figure 17.2. You may notice that the Scene class and associated procedure tabs are still available in the editor, but the Elephant class tab is also now displayed as an editor tab.

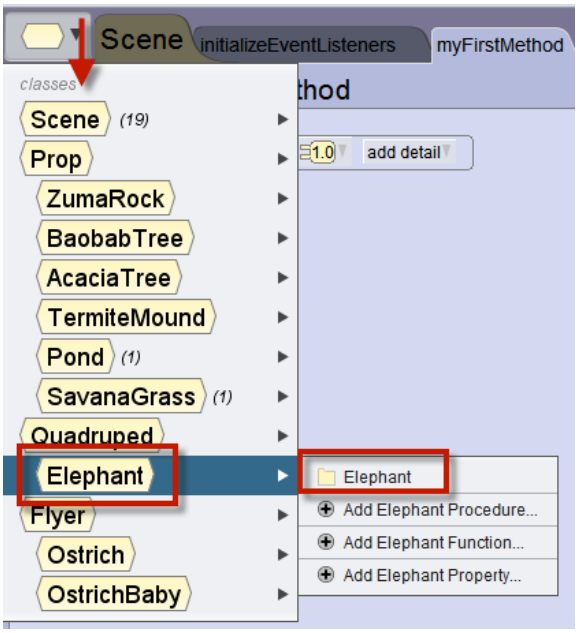

Figure 17.1 Selecting a class from the class tree menu

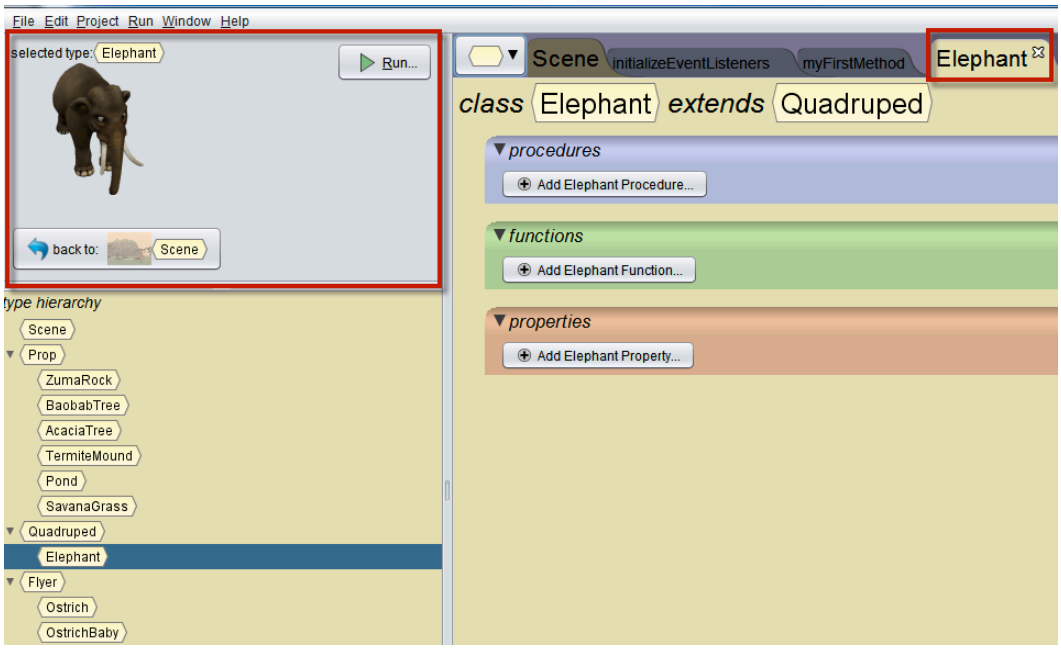

Figure 17.2 Elephant class is now actively displayed

Open a new method (procedure or function) tab

Now that the Elephant class tab is displayed, tabs can be opened for an existing Elephant class method (procedures and functions) or new methods may be created. Also, existing properties can be edited or new properties created. As an example, we will illustrate creating a new procedure. As shown in Figure 17.3, we clicked the *AddElephantProcedure* button. When the add procedure button is clicked, a dialog box pops up where the name of new procedure may be entered. In Figure 17.3, we entered the name *flapEars*.

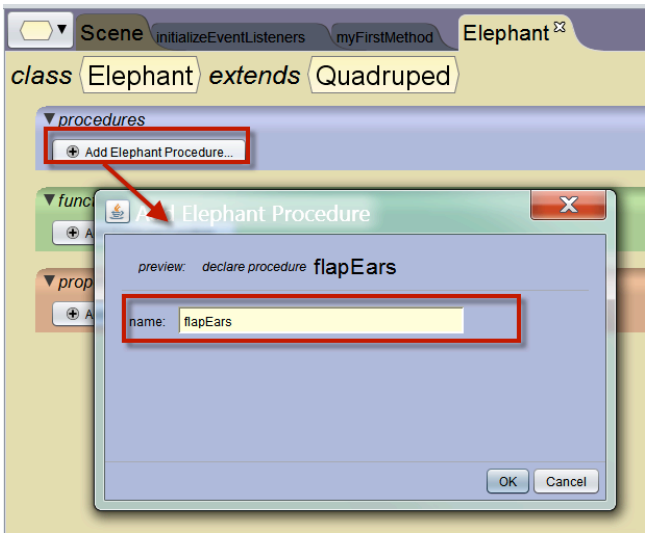

Figure 17.3 Elephant class is now actively displayed

After a name for the new procedure has been entered and the OK button is clicked, Alice opens a new tab for the new procedure, as shown in Figure 17.4. Code statements can be created in this tab to provide animation instructions for flapping an elephant's ears.

Figure 17.4 flapEars procedure tab is now active

Ordering of tabs in the Code editor

Look closely at the positioning of the tabs for classes and procedures in Figure 17.5, above. Tabs in the Code editor are always arranged such that a Class tab is immediately followed by method tab(s) for procedures and functions defined by that Class, as illustrated in Figure 18.4. The Scene class tabs are open by default. A newly opened class is always opened to the right of the Scene class's tabs.

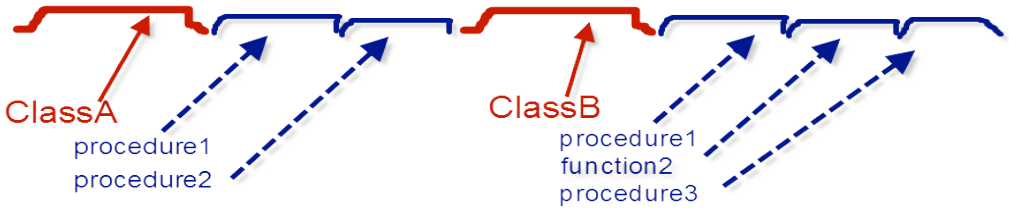

Figure 17.5 Order: Class tab is followed by its procedure & function tabs

Closing a tab

To close a tab, click on the X in the upper right of the tab, as shown in Figure 17.6. A Class document tab cannot be closed until all its procedure & function tabs have been closed.

Figure 17.6 How to close an editor tab

18. How to create program statements

The important thing to know about the Code editor is: this is where you create your program statements. Also, when the Run button is clicked, the program statements in this scene's *myFirstMethod* will be performed. So, to illustrate how to create a program statement, we will create statements in *myFirstMethod*.

Drag and drop a program statement

In our example scene, the baby ostrich is only a few hours old and is just getting her first look at the world around her. Let's create a statement to have the baby ostrich spin all the way around (one complete revolution). First, select the babyOstrich object in the object menu, as shown in Figure 18.6.

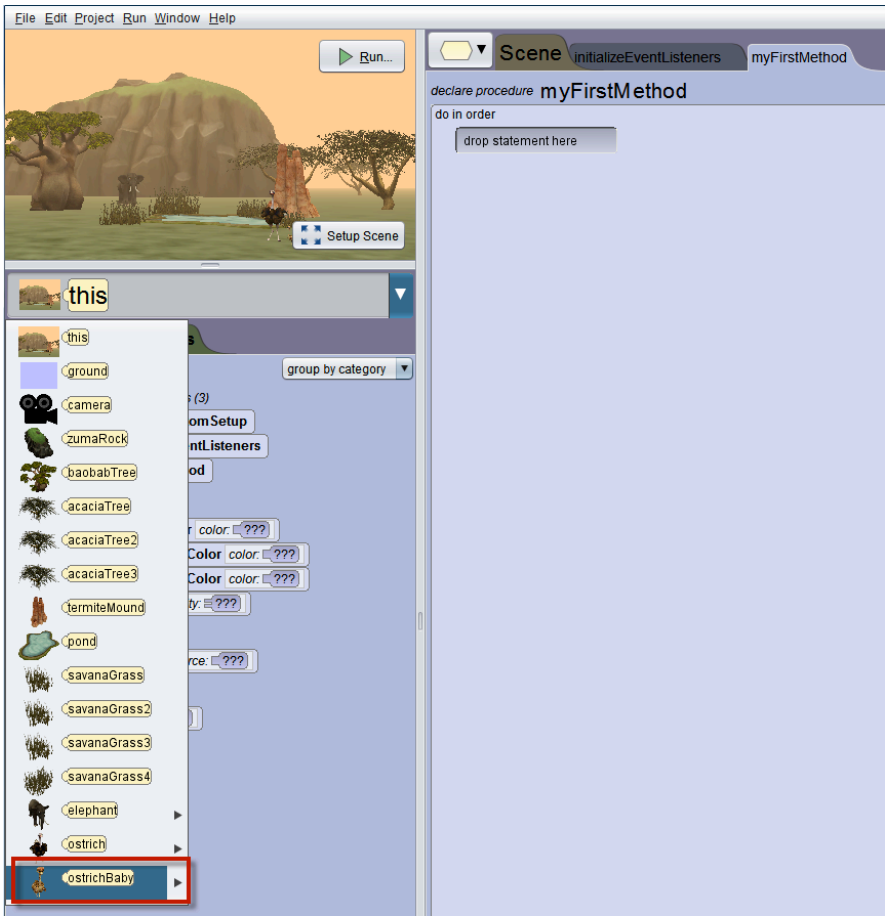

Figure 18.6 Select ostrichBaby object

Make sure that ostrichBaby is the selected object and then select the *turn* tile in the Procedures panel. Use the mouse to drag the *turn* tile into the editor tab, as shown in Figure 18.7

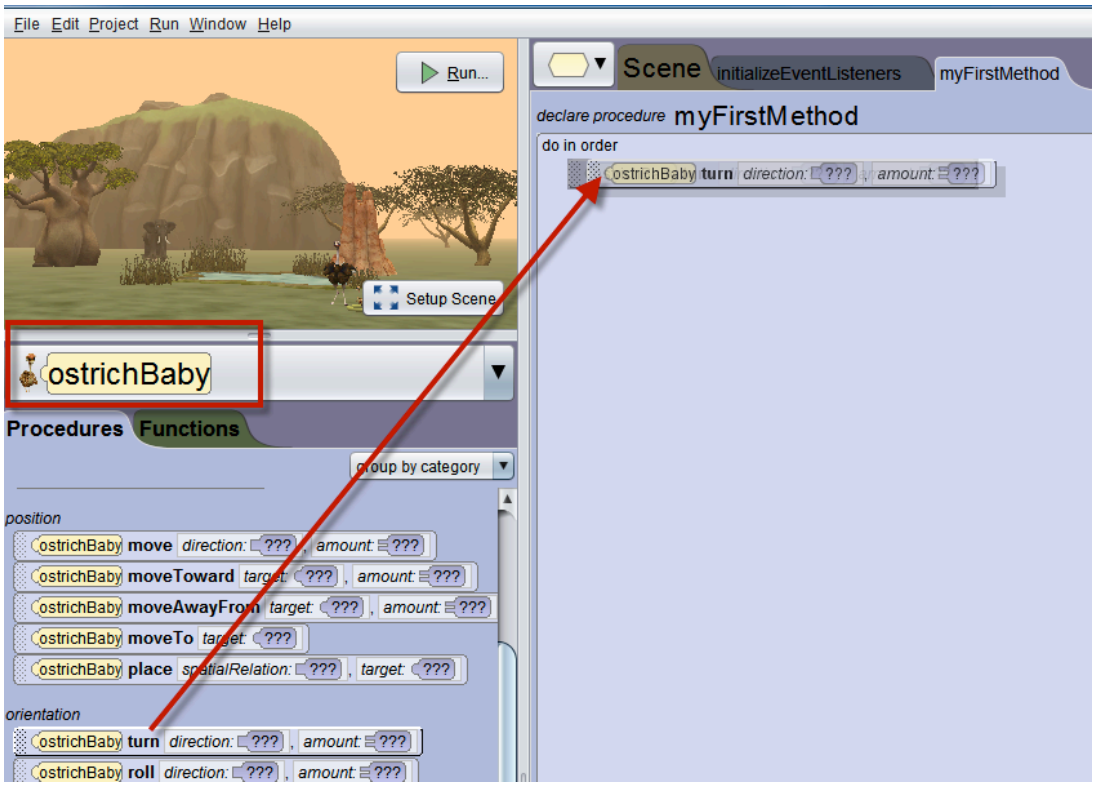

Figure 18.7 Select ostrichBaby object

When the tile is dropped into the editor, a menu pops up where the direction and the amount of the turn can be specified. In this example, we chose turn left 1.0 revolution, as shown in Figure 18.8. The resulting statement is shown in Figure 18.9.

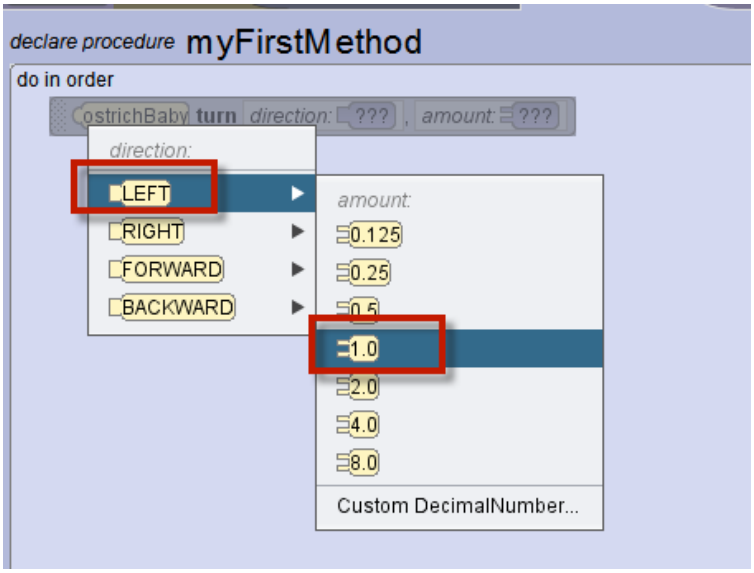

Figure 18.8 Select direction and amount

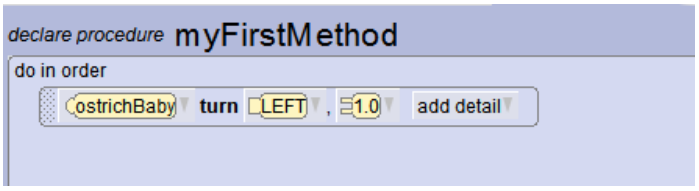

Figure 18.9 Completed statement

Run

One of the features of Alice is the program code can be run and tested without first having to write dozens of lines of code and compiling a complete project. Just click on the Run button to view the animation created when your program. A runtime window is displayed where the animation can be viewed, as shown in Figure 18.10. To view the animation again, just click the Restart button in the runtime window.

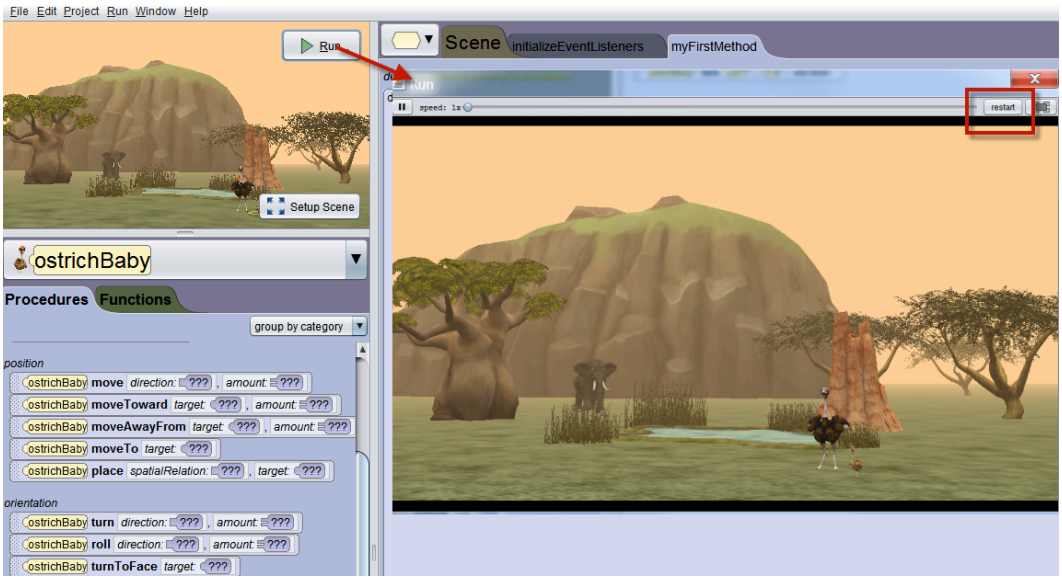

Figure 18.10 Click on Run to view runtime window

19. How to Cut, Copy, and Paste using the Clipboard

The purpose of this section is to demonstrate using the clipboard to perform cut, copy, and paste in a drag-and-drop Code editor. Cut, Copy, and Paste items appear in the Edit menu, as shown in Figure 19.1. However, these items are placeholders…allowing for possible future implementation.

Figure 19.1 Edit options in the Menu bar

Actually, the traditional cut, copy, and paste actions are useful in a text-based editor but are of limited use in a drag-and-drop editor. In Alice, a clipboard is far more useful as a way to store a single graphic tile (one statement) or a block of graphic tiles (multiple statements) for cut, copy, and paste actions.

Cut

To cut, use the mouse to drag a single graphic tile or a block of graphic tiles into the clipboard, as shown in Figure 19.2. In the example shown here, an entire *do together* block is dragged to the clipboard.

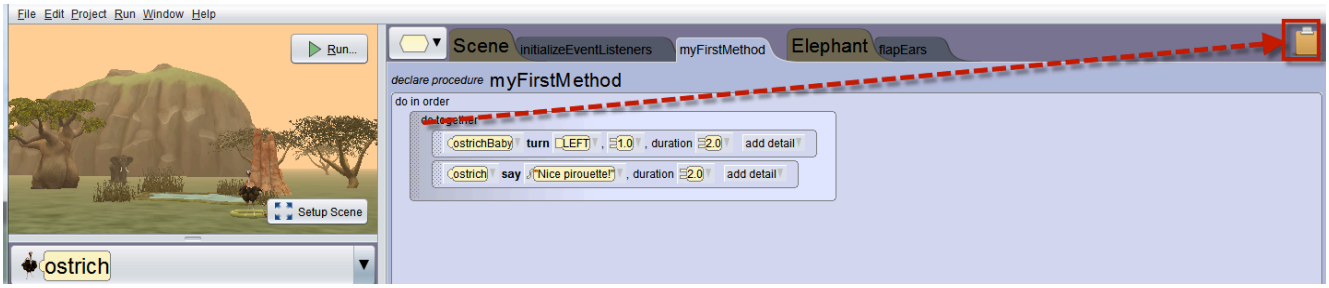

Figure 19.2 Drag code to cut to the clipboard

By default, when the mouse is released, the clipboard turns white and the block of tiles is erased from the editor, as shown in Figure 19.3. In other words, the default clipboard action is cut.

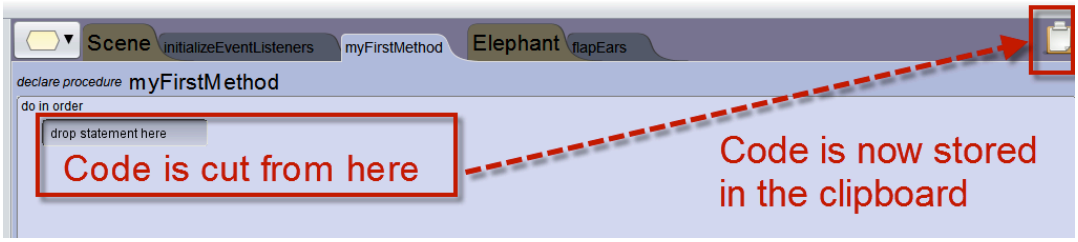

Figure 19.3 Cut removes selected code tiles

Paste (and remove)

Once code has been stored on the clipboard, it can be pasted into any open tab in the Code editor. In this example shown in Figure 19.4, we dragged the code from the clipboard back into *myFirstMethod*, but it could have been pasted onto any tab in the editor. Note, in Figure 19.4, the color of the clipboard has returned to its usual brown. By default, dragging a graphic tile out of the clipboard removes the tile from the clipboard. In this example, the clipboard is now empty.

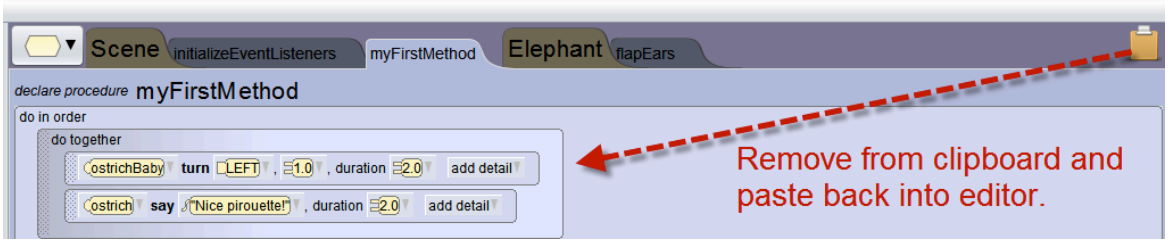

Figure 19.4 Paste the code from the clipboard into the editor

Copy

To copy code (instead of cutting), press and hold the Ctrl key (the Option key on Mac) while using the mouse to drag the code into the clipboard, as shown in Figure 19.5.

Hint: release the Ctrl (OPTION) key only after releasing the mouse button.

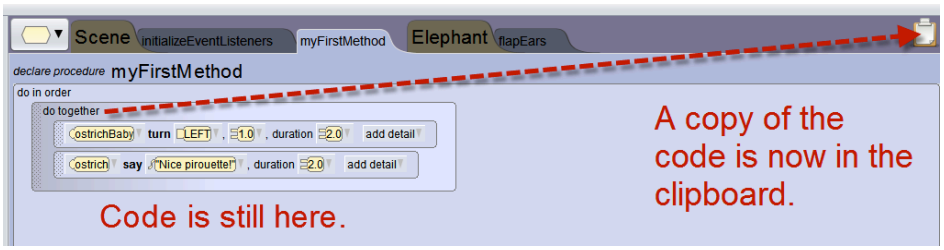

Figure 19.5 Drag with Ctrl key (Option key on Mac) held down to copy to clipboard

Paste (no remove)

To paste without removing the code from the clipboard, press and held the Ctrl (Option on the Mac) key while dragging from the clipboard into the editor, as shown in Figure 19.6. Note that the color of the clipboard has remained white. This means the clipboard still holds a copy of the tile, allowing it to be pasted more than once.

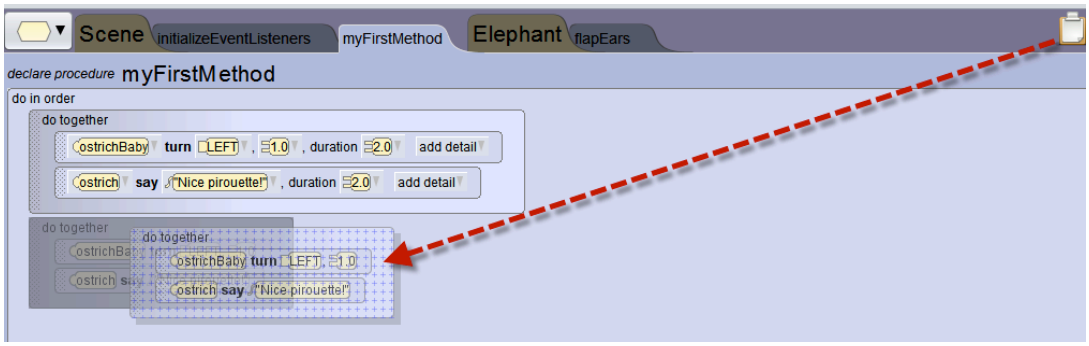

Figure 19.6 Paste with Ctrl (Option on Mac) to copy from clipboard to the code editor

Note: Cut, copy, and paste actions can result in scope errors. In the examples used here, we worked with myFirstMethod and encounter methods -- both of which belong to the Scene class. Because this scene contains all other objects in the virtual world (in this example, the dolphin and seaPlant1), we had no scope errors.

We wish to caution the reader, however, that if code is cut or copied from a method belonging to one class and then pasted into a method belonging to a different class, a scope error may occur. This is not unique to Alice. This is standard protocol for scope-enabled programming languages, whether working in a text editor or a drag-and-drop editor.

20. How to Import and Play a Sound (Audio File)

The purpose of this section is to demonstrate how to import and use audio files for creating sound effects in an Alice 3 animation.

Import with Resource Manager

One way to import a sound is to use the Resource Manager. The Resource Manager was previously introduced in Part 1, Section 3 of this How-To guide. Also, an example of using the Resource Manager to import a 2D image as a billboard was provided in Part 2, Section 6. In this section, the Resource Manager will be used to import an audio file which can then be used to play sounds in an animation program. In the Project menu, select Resource Manager, as shown in Figure 20.1. A Resource Manager dialog box is displayed, where you can select the Import Audio button.

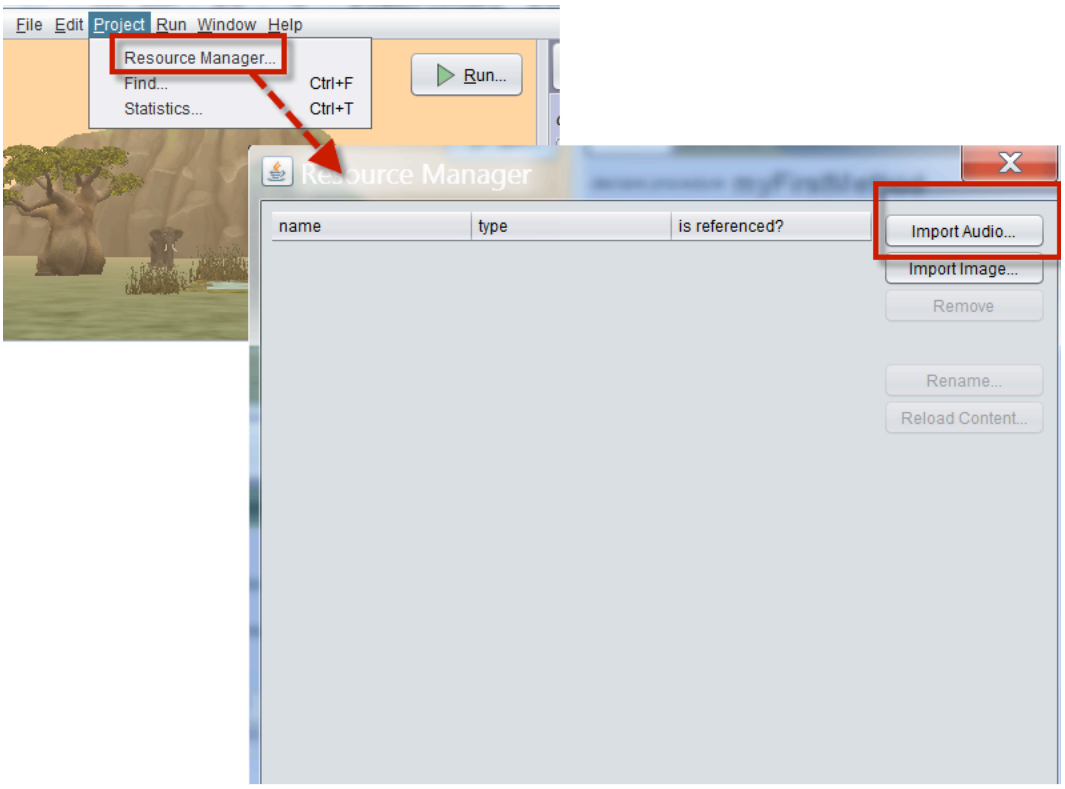

Figure 20.1Paste with Ctrl (Option on Mac) to copy from clipboard to the code editor

When the Import Audio button is clicked, a navigation window is displayed containing a Sound Gallery. The Sound Gallery is a collection of audio files, especially constructed for use in Alice projects. ***The audio files in the Alice Sound Gallery are freely provided for use in non-***

commercial, educational projects. Please note, however, that the audio files are copyrighted and may not be used for commercial purposes without prior written permission from Carnegie Mellon University.

You may browse the audio files in the Sound Gallery and select an appropriate audio file for import. In the example shown in Figure 20.2, we selected *drumroll_finish.mp3*.

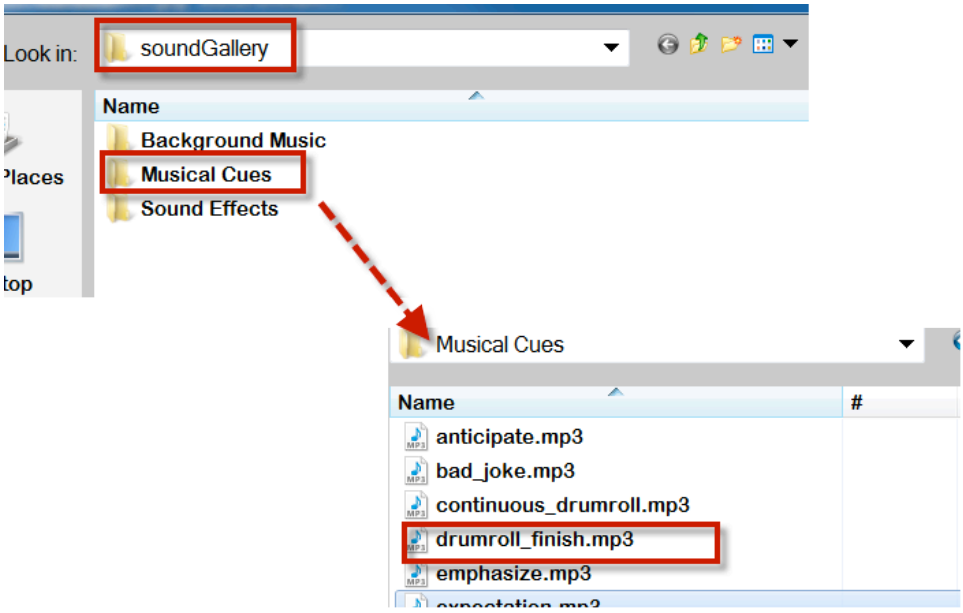

Figure 20.2 Select an audio file for import

The imported audio file is then listed in the Resource Manager, as shown in Figure 20.3. Because the file in this example has just now been imported, it is not yet being used in the program code and the Resource Manager indicates that "is referenced?" is **NO**.

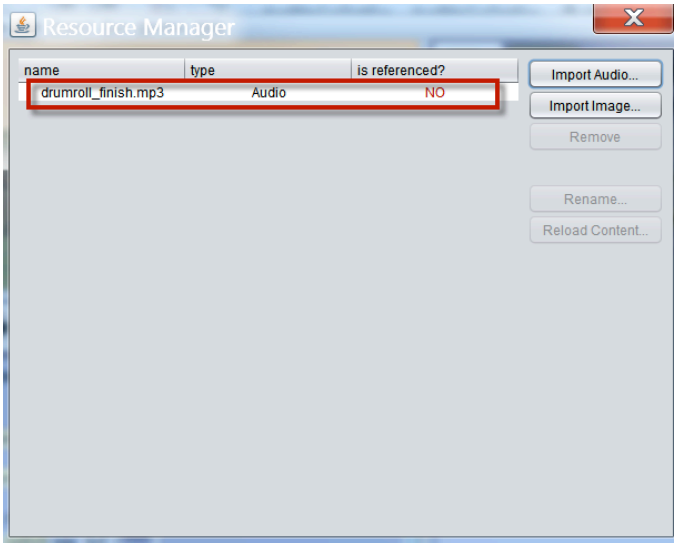

Figure 20.3 Imported file is listed in the Resource Manager

Play an audio file

Sound effects in Alice animations are created by playing an audio file. To play an audio file, drag a *playAudio* tile into the editor, as shown in Figure 20.4. In this example, we dragged *this* (the current scene's) playAudio tile into the editor. When the tile is released in the editor, a popup menu offers the option of selecting an audio file that has already been imported into the Resource Manager or to import a different audio file.

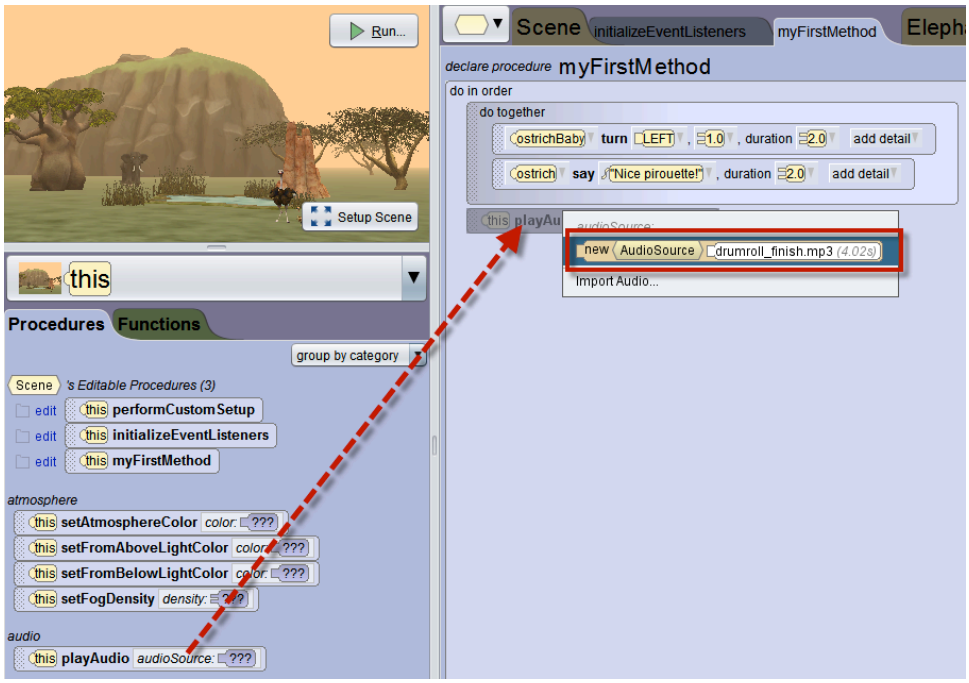

Figure 20.4 Creating a playAudio statement

As shown in Figure 20.5, the *playAudio* statement in this example was positioned immediately after the *do together* code block where the baby ostrich turns one revolution and the mother ostrich says "Nice pirouette!"

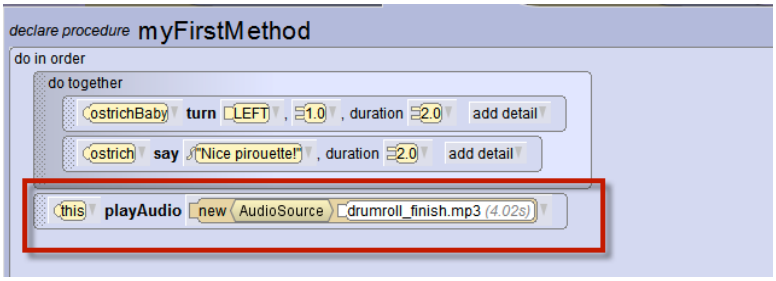

Figure 20.5 Complete playAudio statement

Modifying sound effects

In the example shown above, the sound is playing in a spot that isn't really appropriate for the context and sequence. It would be much better if the drumroll were played prior to the *do together* code block, or perhaps even within the *do together*. To modify where a sound is played during the animation sequence, just use the mouse to click-and-drag the statement to a different place in the code sequence, as shown in Figure 20.6.

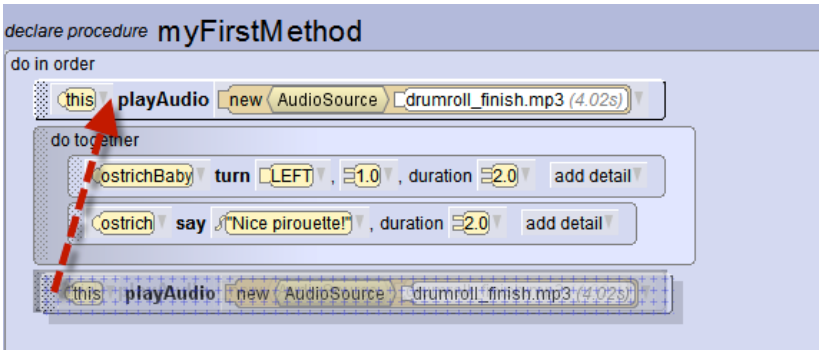

Figure 20.6 Changing sequence for playing a sound

When playing a sound, we often want to shorten the length of time it plays. For instance, in this example the drumroll is 4.02 seconds. This may seem to be a short time, but is actually much too long for this animation. To play only a portion of the sound, we can customize the start and stop points for playing the sound. Click the arrow at the end of the *playAudio* statement and then use the slides to select a start time and a stop time for a shorter length of time, as shown in Figure 20.7. In this example, the start was set at 1.0 and the stop at 0.0703 on the audio timer. As a result, the sound will now play for 1.5 seconds.

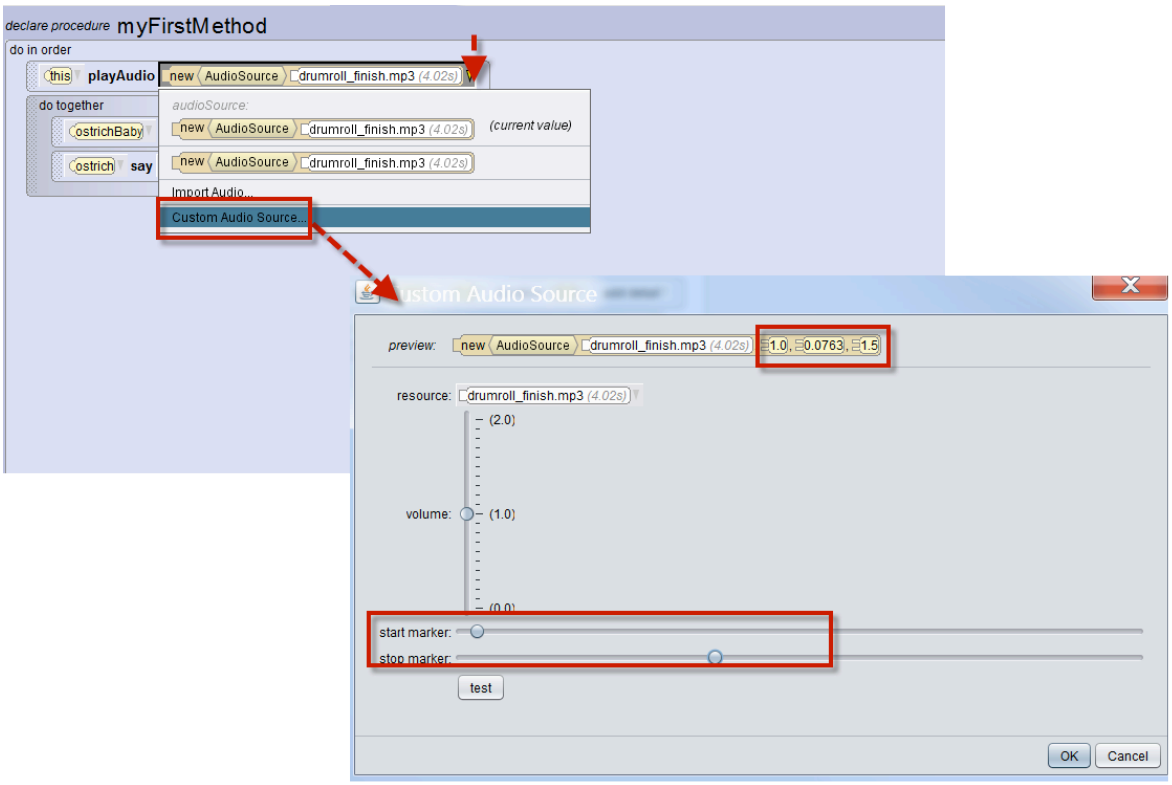

Figure 20.7 Customizing to play a shorter segment of the audio file

Suggestions for using and editing audio files

Alice is not sound/audio recording studio software. Other applications are available online that performs these actions far better than our resources can suuport. For creating your own recordings, you might consider software such as GarageBand (Apple, Inc.) or Mixcraft 6 (Acoustica) which are not free but are reasonably priced and have user's guides.

For purposes of editing existing audio files, we use and recommend Audacity, free software from Carnegie Mellon University, see:

http://www.cmu.edu/computing/software/all/audacity/

Audacity is highly effective as a tool for extracting and exporting a short audio clip for use in Alice 3. Use of a shorter audio clip can dramatically decrease the size of an Alice project and also helps adhering to the guidelines for educational "fair use." In any case, we strongly recommend that you observe copyright laws. Of particular importance is the need to guard against redistribution of any copyrighted media.

21. HOW TO EXPORT AND IMPORT A CLASS FILE

Video: Exporting and Importing a Class File

The purpose of this section is to demonstrate how to reuse Class code by exporting code written for a Class in an Alice project and then (later) importing that file in a different Alice project.

Export

To export code written for a Class in an Alice project, follow these steps:

Step 1: Open the class you wish to export. In the example shown in Figure 21.1, the Ostrich class is selected in the **Class** menu.

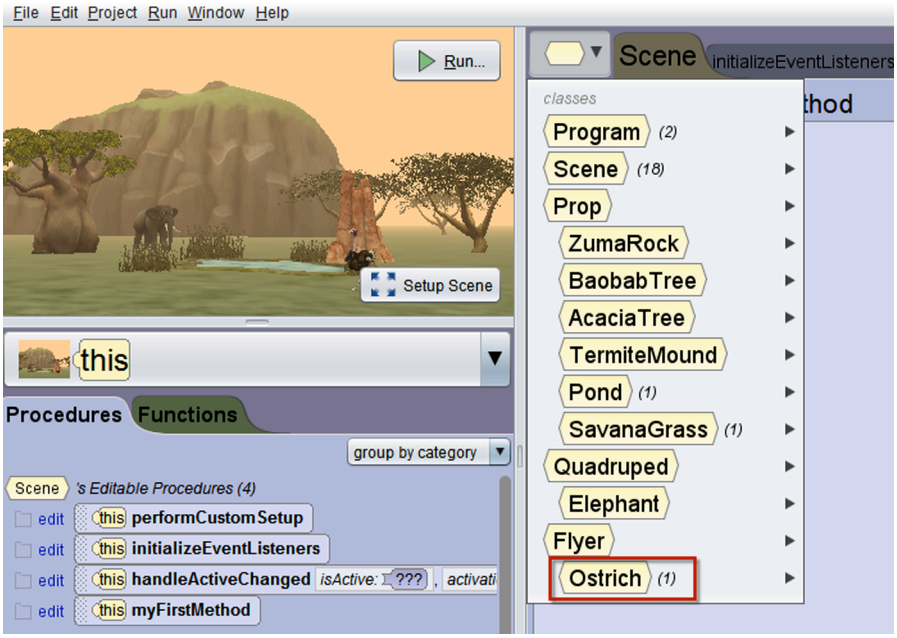

Figure 21.1 Select a class in Class tab

When a class is selected, the tab for that class should become the active tab in the Edit panel, as shown in Figure 21.2.

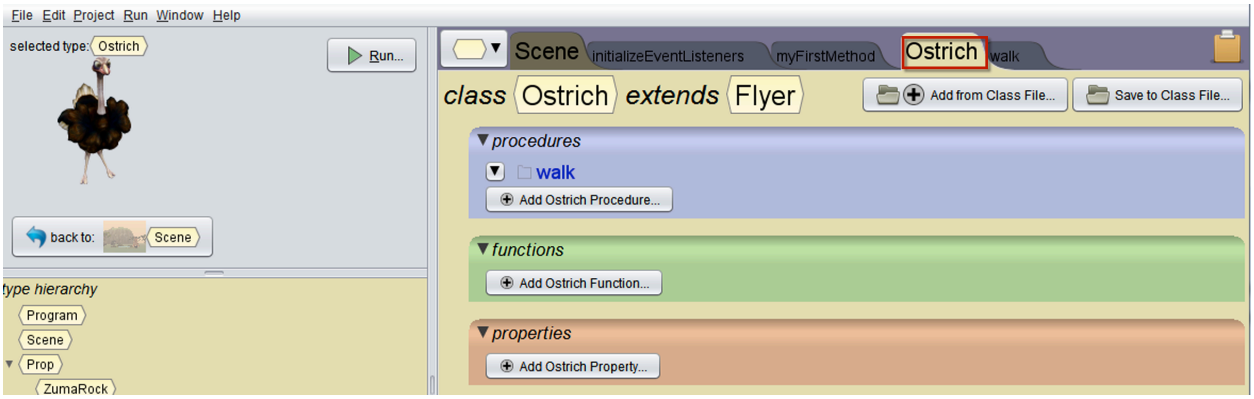

Figure 21.2 The Ostrich class is the active tab in the Edit panel

Step 2: In this example, we have defined a *walk* procedure for the *Ostrich* class. To save this code for use with Ostrich objects in another project, click the **Save to Class File** button. A save file dialog box is displayed, as shown in Figure 21.3.

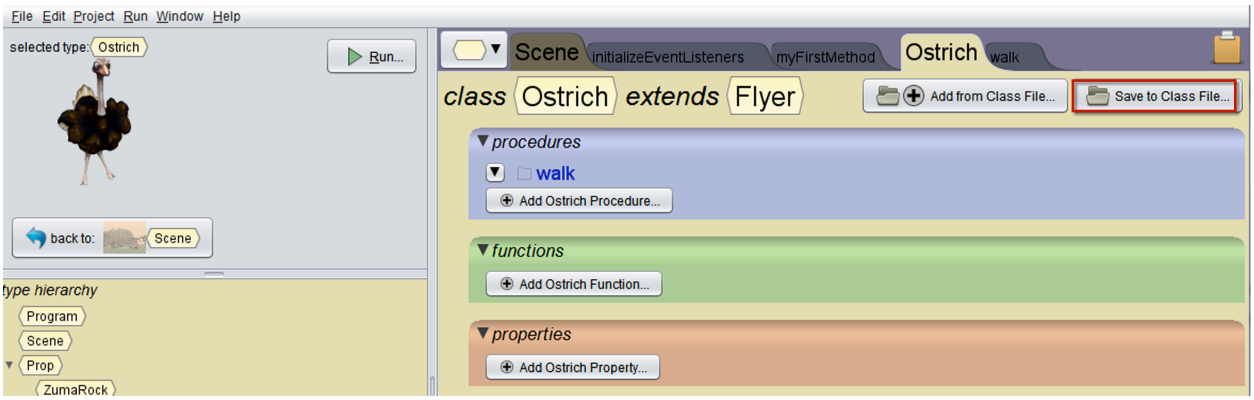

*Figure 21.3 Click the **Save to Class File** button*

Alice automatically creates a **MyClasses** folder where class files can be saved. Save the file.

Note: We recommend that you save the file in **MyClasses**, but you may select another location on your computer for storing the file. You do not need to enter the filename extension; Alice automatically adds .a3c as the file format.

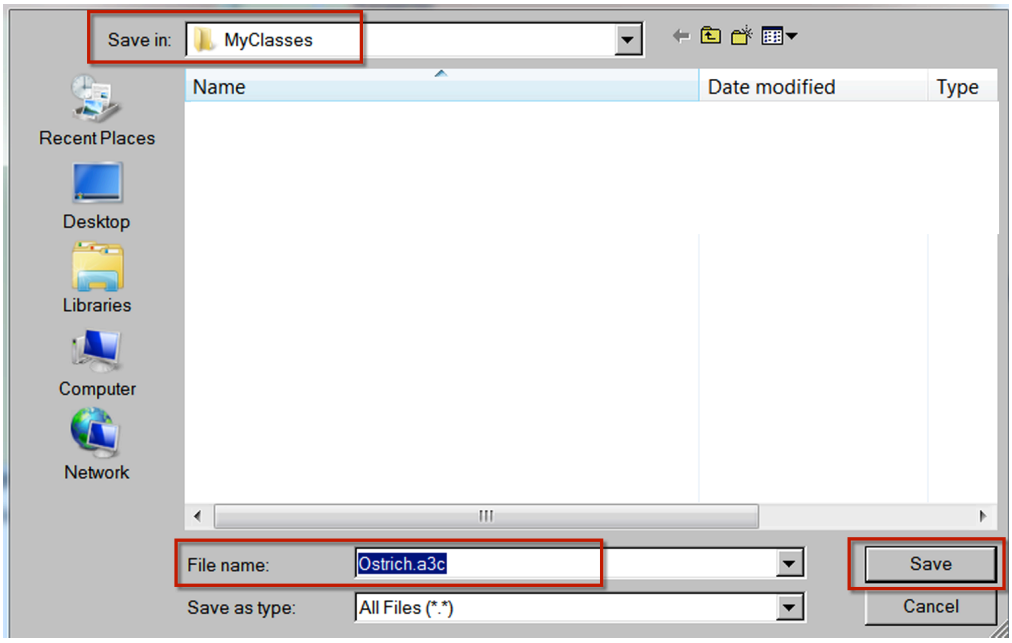

Figure 21.4 Save the class file

If the class file has been successfully saved in the **MyClasses** directory, it should appear in the Gallery tab labeled **My Classes**, as shown in Figure 21.5.

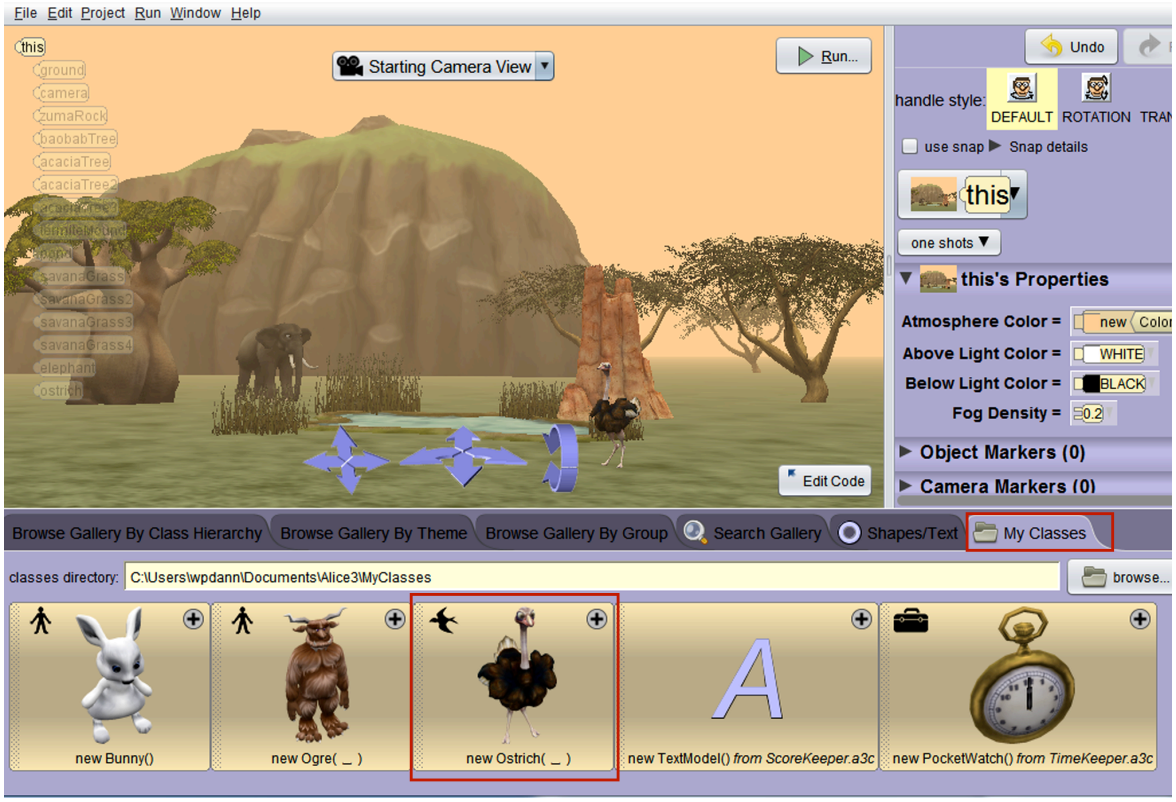

Figure 21.5 Exported classes in the My Classes directory

Import

To illustrate how to import previously exported code, we will build upon the Ostrich export example described above. We are assuming the Ostrich class, with the walk procedure, has been exported. Sometime later, we create a new Alice project that has an Ostrich object, as shown in Figure 21.6.

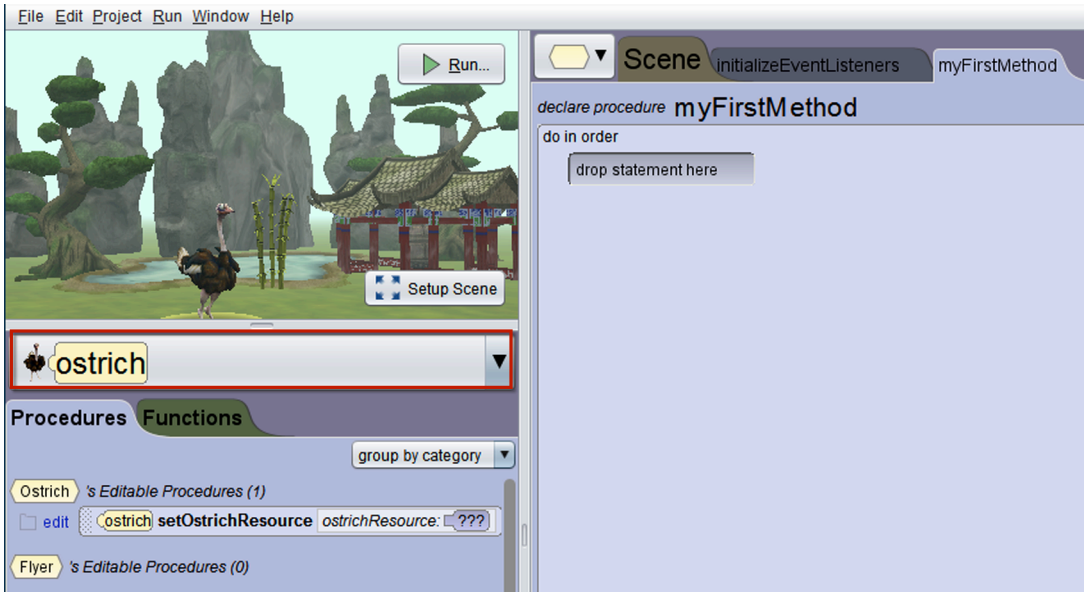

Figure 21.6 A different project with an ostrich object

Step 1. Select the class in the Class menu. You should see the tab for that class as the active tab in the Edit panel, as shown in Figure 21.7. In this example, the ostrich object was added to the scene using the **Ostrich** class in the **Flyer Gallery.** The **Ostrich** class in the **Flyer Gallery** does not have a *walk* procedure. (This is not unique -- none of the Flyer classes have a pre-defined *walk* procedure.)

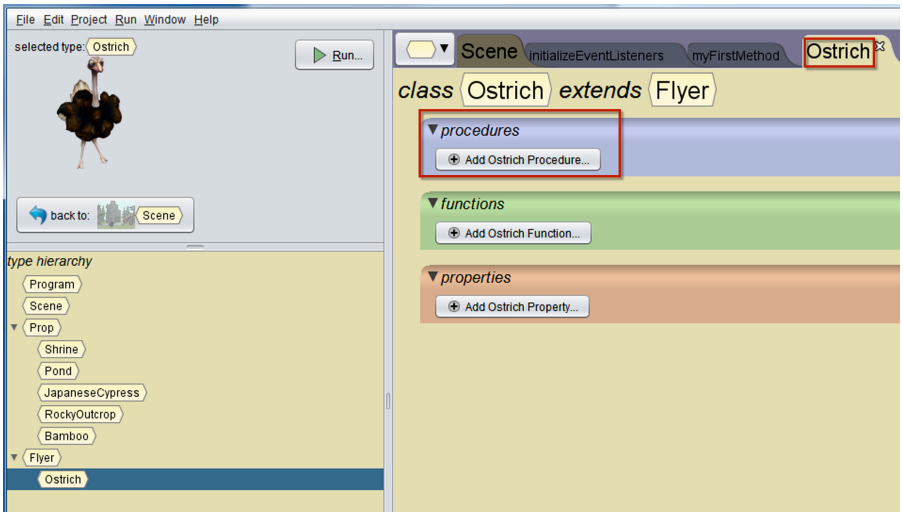

Figure 21.7 No walk procedure

Step 2. Click the **Add from Class File** button in the Class tab, as shown in Figure 21.8.

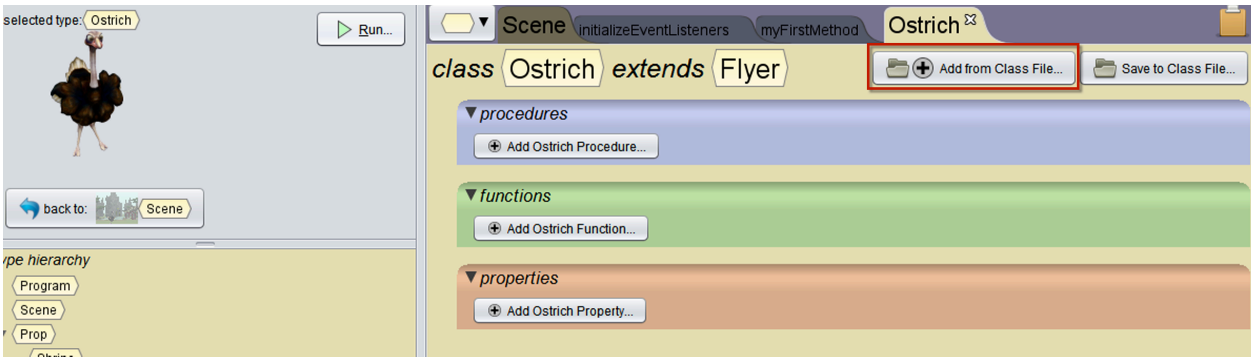

*Figure 21.8 Click the **Add from Class File** button*

A **Save File** dialog box is displayed, as illustrated in Figure 21.9. In this example, we clicked on Ostrich.a3c and then the Open button.

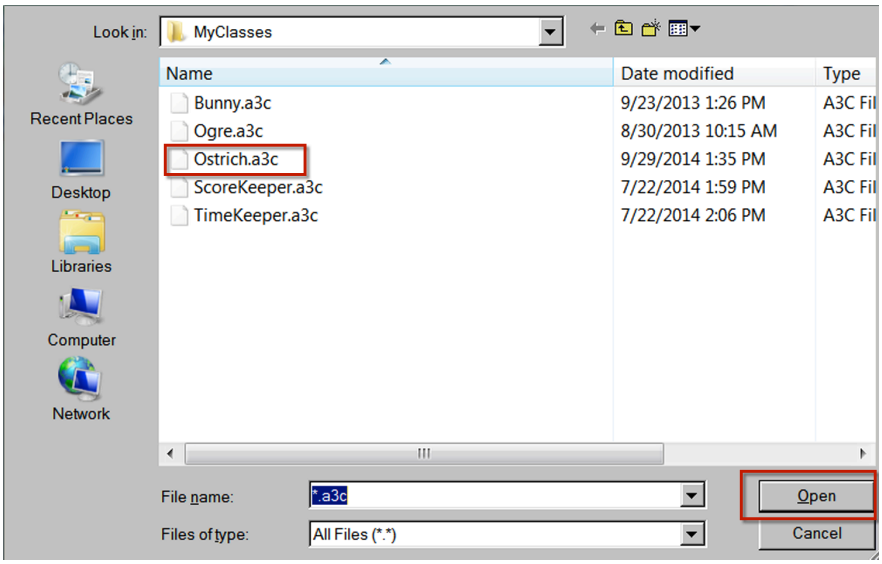

Figure 21.9 Select the desired class file and open it

A **Class file content** dialog box is displayed, where you can select the procedures, functions, or properties you wish to import. In the example shown in Figure 21.10, we selected the *walk* procedure and then clicked the **Next** button.

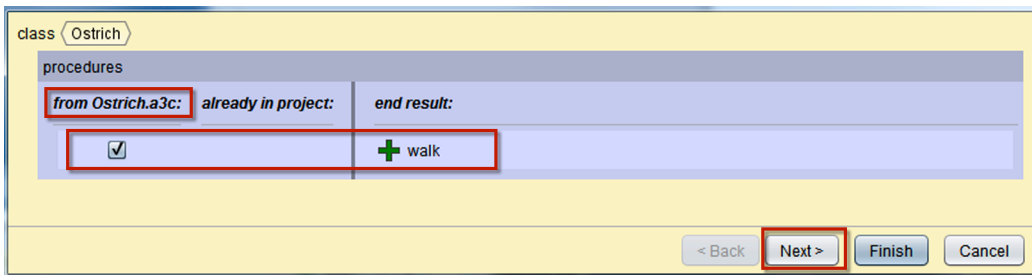

Figure 21.10 Select the class content to be imported

A **Merge** dialog box is displayed, as shown in Figure 21.11. If any conflicts exist between code already in the project and the code to be imported, this dialog box will provide options for selecting which version you wish to keep. If you wish to keep more than one version, the versions can be renamed to avoid name conflicts. When conflicts, if any, have been resolved, click the **Finish** button.

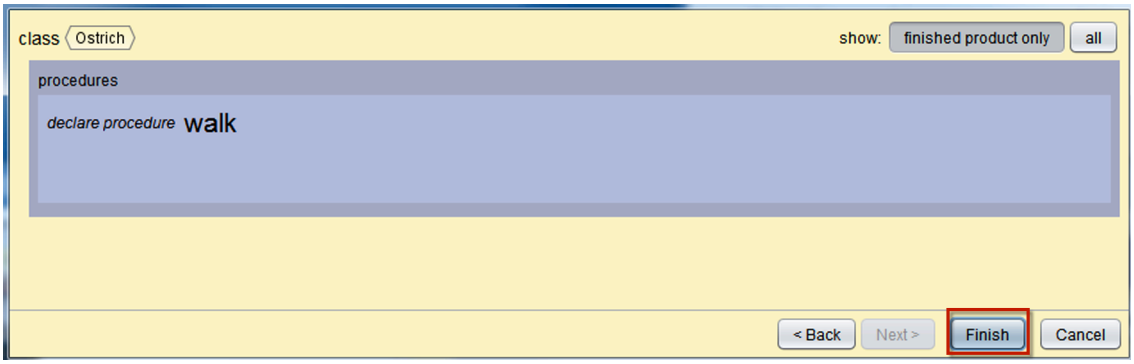

*Figure 21.11 Click **Finish***

Imported class file content will now be listed in the Class tab, as shown in Figure 21.12.

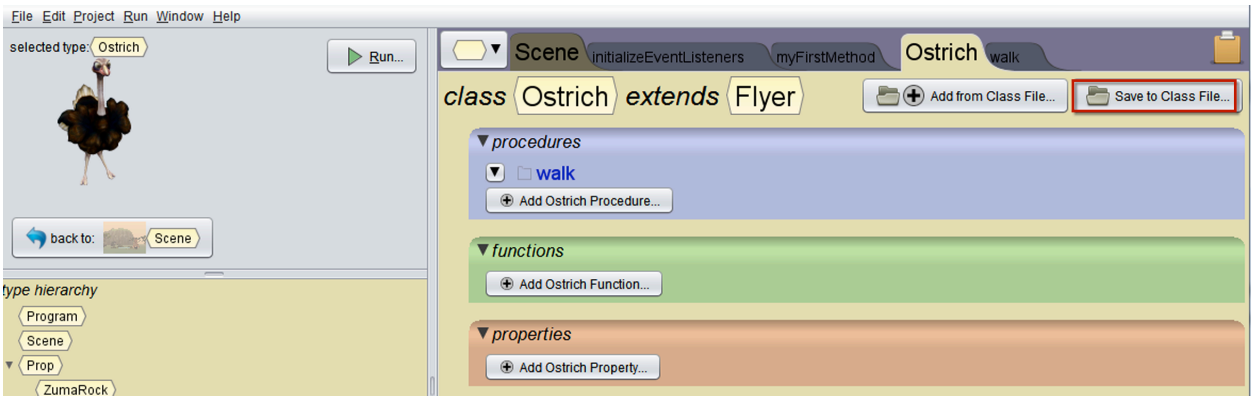

Figure 21.12 Import is complete

A detailed tutorial for Export and Import is available as a Video at

http://www.alice.org/3.1/materials_videos.php