Java Fundamentals
2-11: Keyboard Controls Project

This project will be the final project that covers the remainder of the Alice course.

You will need the file underTheSea_start.a3p that can be found on Oracle Academy Member Hub.

Lesson Objectives:

- Create an opening Sequence
- Use keyboard controls to manipulate an animation
- Save your Class file
- Using the starter tab
- Add an existing class file to an animation

Instructions:

1. Open Alice 3 on your computer.
2. Either using the My Projects tab or the File System tab, browse for and open the Fish_10.a3p file.

   For this lesson you are going to look at saving a class file from one program and transferring it to another.
   You are also going to learn how to include interactivity into your animations through the use of keyboard and mouse events as well as automating the process of collision detection.
3. You have created multiple transferrable procedures within the Fish class in this animation. These procedures can be saved to a file so that you can transfer them into another application.

   Click on the classes button and select Fish and then Fish. You will notice beside Fish on the main menu the number 6 in brackets. This is the number of procedures stored at that level.
   This is the template for the class that details any procedures, functions or properties that exist for that class.
4. Click on the Save to Class File button.
5. In the MyClasses folder name your file myFish.a3c
6. Using the file menu, select open and browse for the underTheSea_start.a3p file.
7. Using Save as name the file underTheSea.a3p.
8. The underTheSea file has a readymade scene that shows a fuller underwater environment. When using a file created by someone else it is always a good idea to familiarise yourself with the contents of the file. Using the object list in the scene editor complete the following diagram by writing the names of the classes in the corresponding box in the table.

Animation Contents List

Ex: scubaHelmet

Now that you are familiar with the contents you can return to the code editor interface!

You can see there is no code in this animation yet. You are going to add the myFish.a3c class from your previous animation so that you can make your fish swim.

9. Click on the classes Button and choose Fish and then Fish
10. From the Fish template click on the Add from Class File button.
11. Choose the myFish.a3c file that you saved earlier in this exercise.

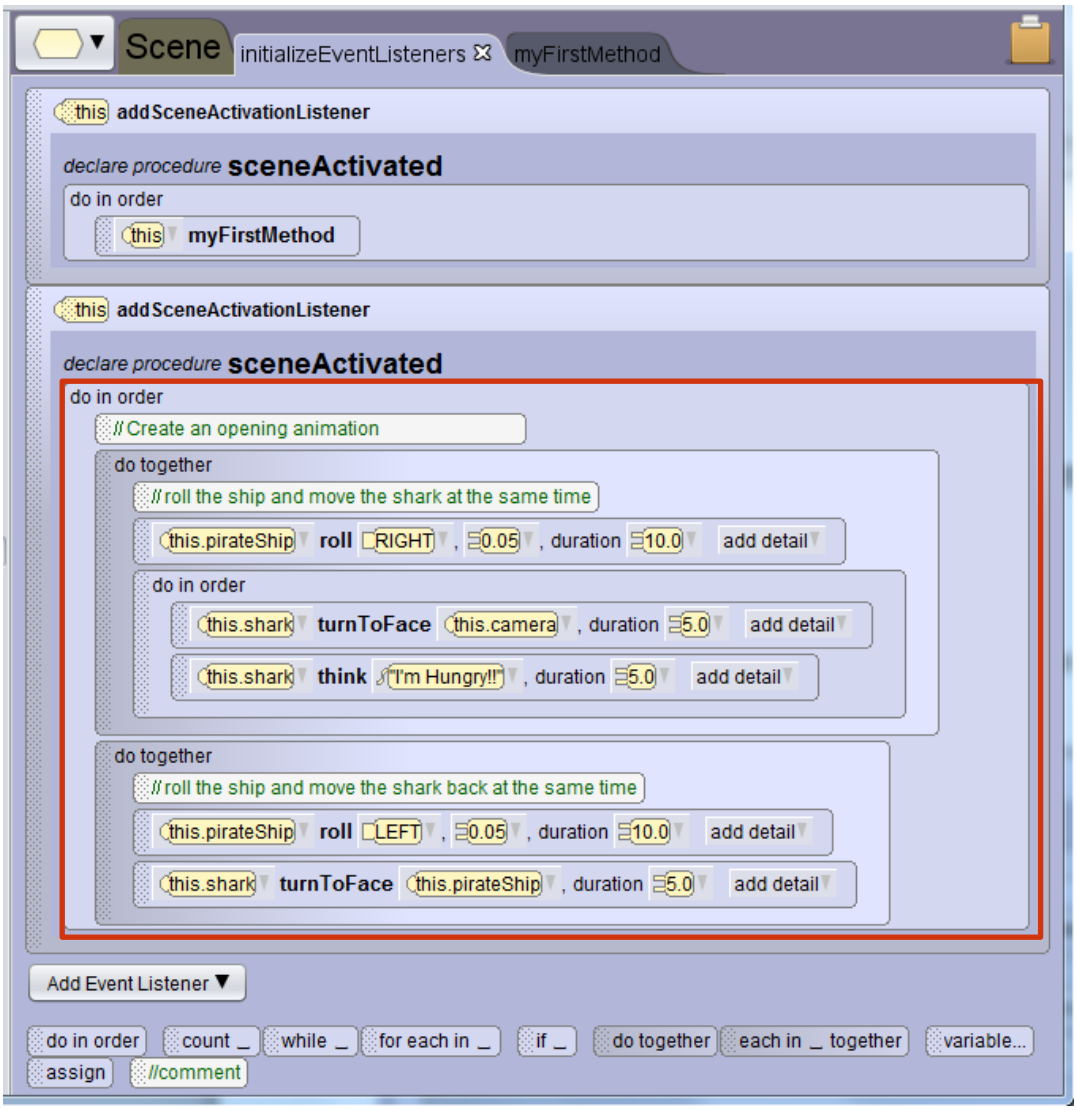
This gives you a list of procedures that can be copied into your animation. If you didn't want all of them you can deselect that procedure by un-ticking the box. If you click on the next button you get to see the code for the procedures. Click Finish to add all of the procedures into the animation.

12. You now have a list of all of the procedures you previously created for the Fish class in this animation.
13. Now that you have your pre-defined procedures available to you, you can implement them in the code for this animation.

    In the code editor drag and drop a while statement using true as the argument value.
14. You want multiple things to happen at once so drag a do together into the while loop.

    A sample textual storyboard for this part would be:
    The Clown fish should bob up and down inside the helmet.
    The Clown fish should move its tail.
    The Blue Tang fish should swim 6 metres.
    The Pajama fish should swim 4 metres.
    The Shark should move its tail.
    All of these actions should be carried out together.
15. Implement the steps from the textual storyboard in Alice by dragging the appropriate procedures into your do together statement.
16. Save your program.

    Okay let's look at using EventListeners. An event is something that happens, a mouse click, a key press the start of an application etc. The first EventListener that you are going to use is the sceneActivated one which executes when we first start the animation.
17. To create a scene activated handler click on the initialize event listeners tab.
18. Click on the add event listener button and from the drop down list choose scene/activation/time and then choose addSceneActivationListener
19. Okay let's add the procedures that we need to make the animation.



Timing is important when programming the animations. You can see that all parts of the first do together statement will take 10 seconds to execute ensuring that the shark and the ship complete their animations at the same time. For the second part of the statement the shark will finish before the ship.

20. Run and test that your animations and scene activated listener are working correctly.

21. Okay now you can create event Handlers for keyboard and mouse.

    You are going to add the listener for key presses so that we **manually move the submarine**.

22. Click add Event Listener, choose keyboard and then addObjectMoverFor. Select submarine for the object.

    This creates an event listener that allows you to control the submarine with the arrow keys.

23. Run and test the program by moving the submarine. Remember there is no object collision built into the submarine!!

Moving the Shark

24. The user should be able to click on the shark and get it to carry out an action. To do this we need to use a mouse listener. Click on AddEventListener then mouse and the addMouseClickOnObjectListener.

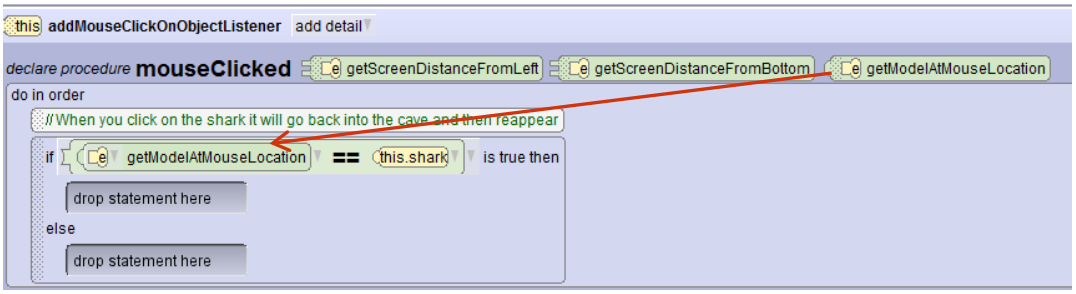
25. When you have the listener procedure you need to add an if statement to deal with the event when it happens. Choose true as the placeholder value.

26. Click on the true variable in the if statement and choose relational SThing (==) from the drop down menu.

27. Again use a random value for a placeholder for the first part of the relational operator but choose shark as the second.

28. This gives us the following procedure

29. Now we need to drag the getModelAtMouseLocation operator onto the submarine variable. Giving us the following



30. Now our expression checks if the mouse click was clicked on the shark

31. Now add the code to move the shark back 2 metres for a duration of 3 seconds the shark will then wait for 10 seconds before it comes forward again in 2 seconds.

32. Run and test the program by clicking on the shark.

Treasure Chest

33. For this object you want to consider a more complicated set of actions.

    If you click the treasure chest once an action will happen and if you click it again a different action will need to occur. If you just have a lid open procedure every time it is clicked it will open further until it does a complete circle through the chest.

34. In the case of the treasure chest the lid will open or the lid will close.

35. Create another onMouseClickOnObjectListener object.

36. Drop an if statement in.

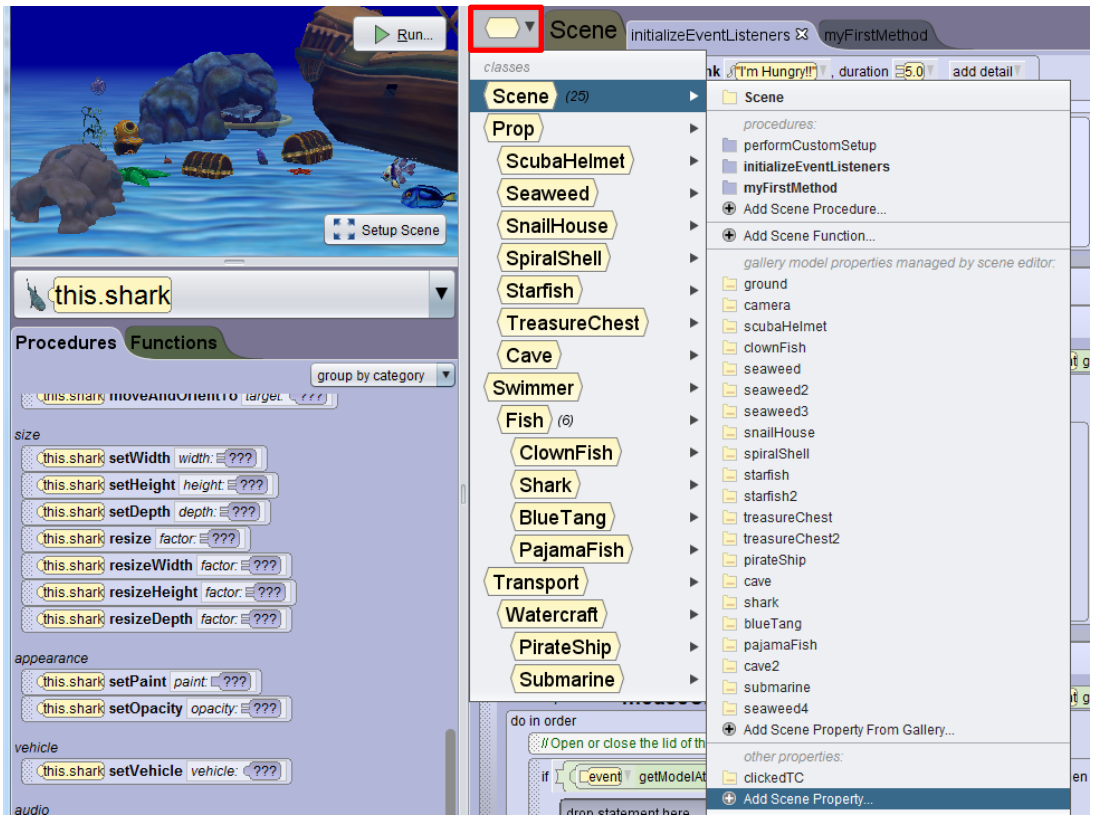
37. Choose the relational expression as you did before with the shark.

38. Choose a random object for the first half of the expression and the treasureChest2 for the second part.

39. Drag the getModelAtMouseLocation on to the placeholder

40. To keep track of whether or not we have already clicked the treasure chest we need to create a global variable that is accessible from anywhere in the code.

41. To do this click on the scene drop down and choose scene then add property



42. Then choose variable, Boolean, clickedTC as the name and choose false as the initial value and click ok.

43. Go back to the event handlers

44. In our treasure chest procedure drop another if statement in.

45. Choose this.clickedTC as the Boolean statement.

46. The if statement has to not only move the desired object but also change the value of the Boolean variable.

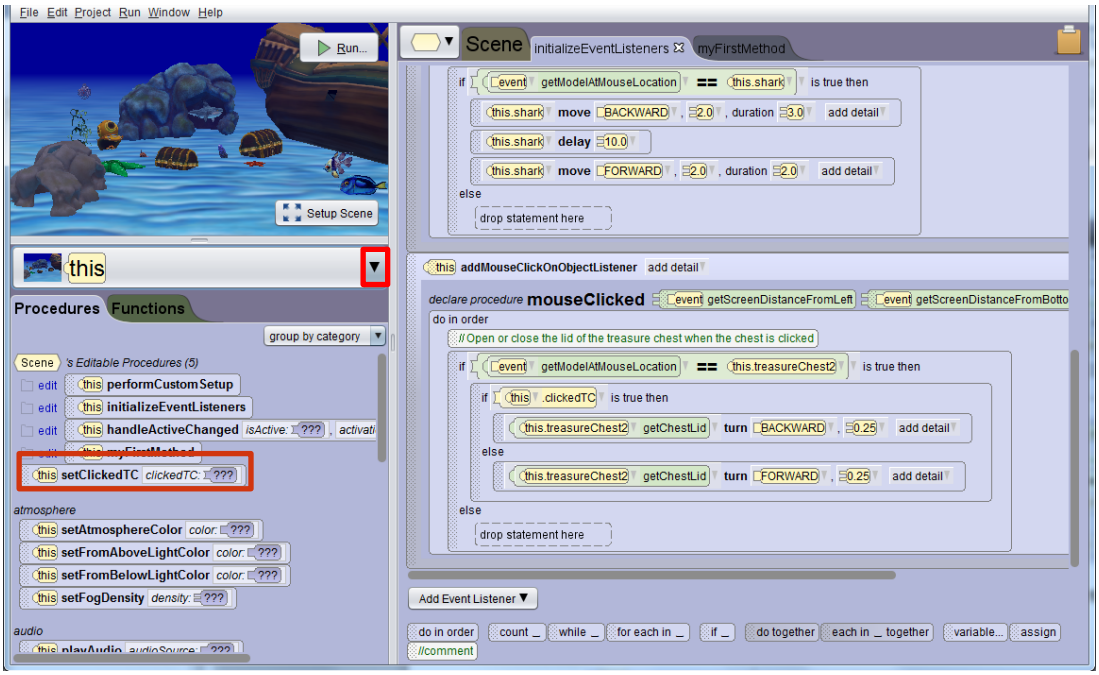
47. We want to open the chest lid so choose that from the object list (Use the prop object and then the joints arrow to access the chest lid)

48. Add a turn statement using backwards and 0.25 as the argument values.

49. Do the opposite to this in the else statement.

50. You now have to set the value of the variable. It is initialised as false to show that the chest has not been clicked. When the user clicks the chest the variable should be set to true and the lid should open. When they click on it again the variable should be set to false and the lid should close.

    To access the procedure to change the value of our global variable (clickedTC) choose **this** from the object list.



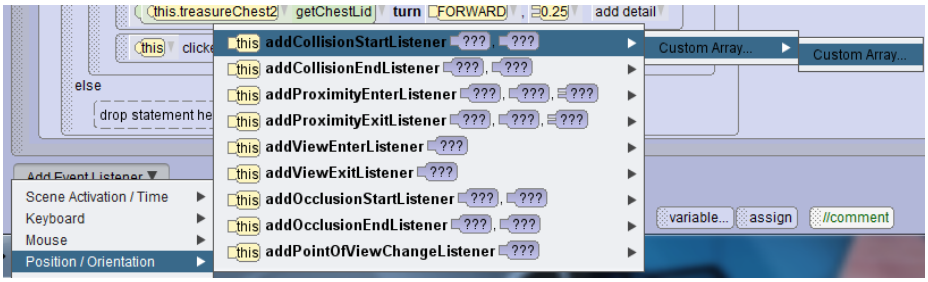
51. Drag the setClicked method underneath the turn statements so that your finished code should look like this.

Collisions

52. Okay the final listener event for this project deals with collisions between arrays (multiple) of objects.

53. You will make the sea snail move about the ocean floor and if it collides with another object it will turn around and move off in the new direction.

    To do this create a collisionStartListener

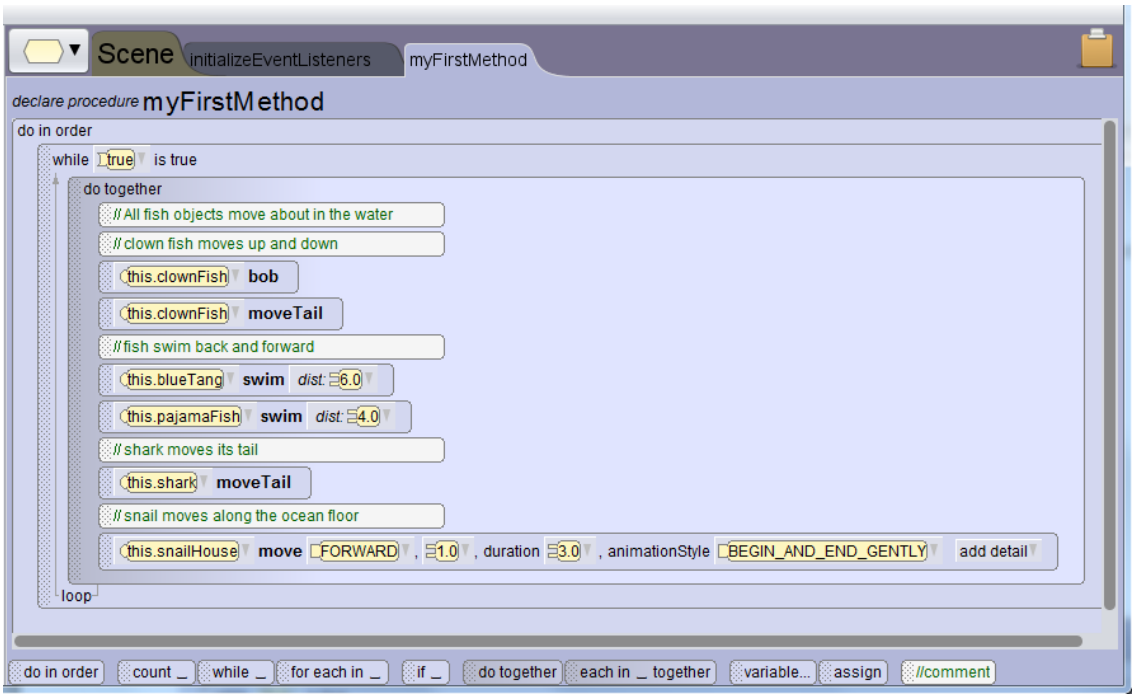


54. For the first element of your custom array add the snailHouse as that is the object that will be doing the colliding.
55. Click OK
56. For the second Array add all the objects that the snailHouse may collide with and then click ok.
57. Add a turn statement that makes the snailHouse turn right by 0.3
58. Add a comment to your code block explaining its purpose.
59. Finally go back to myFirstMethod and add the code to make the snailHouse move.

    The snail should:

    - move forward 1,
    - duration should be 3
    - Animation style should be begin and end gently

Your finished myFirstMethod should now look like this:



You can see that your driver class (myFirstMethod) has next to no code in it. It simply calls other procedures to do all the work.

This is what you should aim for in your programming!!

60. Save your program.

Optional Exercises

Try to add the additional options to the underTheSea animation.

- *When you click on the scuba helmet it will tip forward/back*
- *When you click on the red starfish make it go under the ocean floor for 5 seconds*
- *Add collision detection for the submarine*
- *Anything else that you want to add to make the animation more engaging!!*

61. Exit Alice 3.