Java Fundamentals
2-10: Variables
Project

This project will progress with you throughout the course. After each lesson there will be more to add until it builds into a complete animation that you can upload to YouTube or export as a local animation file.

Lesson Objectives:

- Use random numbers to randomize motion
- Understand variables
- Understand how variables are used in programming
- Viewing Alice code as Java Code on the side

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_9.a3p file.
3. Using the Save As command from the file menu, rename the file to Fish_10.a3p.
4. If you are not already in the code editor use the Edit Code button to go to the code editor.

   For this lesson you are going to look at variables and random behaviour in your code.
   You currently have a code block in the myFirstMethod that makes all of the fish shake their heads at the same time.
   It only does this once though and it would be good if the fish would shake their heads a random number of times when the code is run.
   It makes your animations much more appealing if they exhibit random behaviour where appropriate.
5. Drag a count statement just above this do together statement and choose 3 as the placeholder value.
6. Drag the do together statement into the count loop.
7. This would make the fish shake their heads three times every time you run the code. To make the value random click on the arrow beside the placeholder value, then choose random and choose the random option that allows you to select the inclusive lower and upper value that you want to use as your random range. Choose 1 and 3 as the argument values.

   This code will allow the fish to shake their heads a minimum of once and a maximum of three times.
8. Test your code by running it multiple times. Count the number of times the fish shake their heads.
9. The next thing to look at is making your procedures more useful by adding variables to your code. You have already used this concept through the arguments that you have been using within your procedures.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Click on the swim tab. If it is not there then click on the classes button and then FISH and click on swim.

10. The swim class is currently very restricted because it only allows a fish to swim a set length (1.2).

    To change this you can add a variable to the top of the code block.

11. Drag the variable statement from the bottom of the screen and place it at the top of the code. Remember the green line shows you where it will be placed when you drop it. This will be at the very top of the do in order statement.

12. Select variable, choose DecimalNumber as the data type and give it the name distance. Set the initial value to 0 through the customDecimalNumber menu option.

13. This will give you a declared and initialised variable in your code named distance with an initial value of 0.0.

14. Save your program.

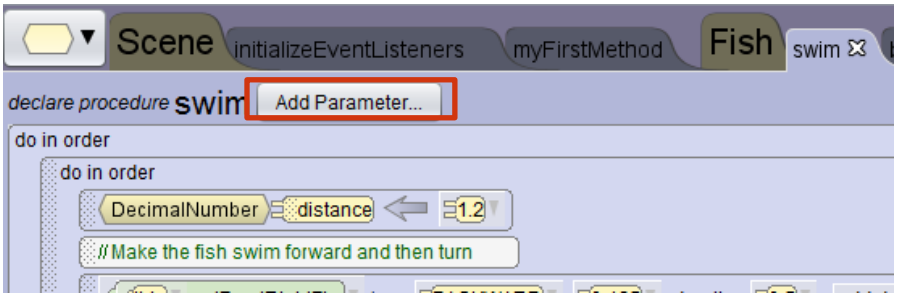
15. To assign the variable to the value, drag the variable name (distance) over the current placeholder values for 1.2. You will have to do this twice for both forward move statements for the Clown fish.

16. Run your code and test it.

17. The clown fish doesn't go anywhere because we have set the distance to 0. Change the value to 1.2 and re-test the code.

    You can see the benefit of using variables here. If you want to change the distance the fish swims then you only need to change the value in one place (at the variable level) and the changes are executed throughout the procedure.
    This can be taken one step further by passing a parameter to the procedure. A parameter is information passed from one procedure to another. You can create code that allows each fish to specify how far they want to swim when they call the swim procedure.

18. Click on the add Parameter button



19. You are going to use this instead of the distance variable. It needs to be a DecimalNumber to work in the code block and needs a name not already used in the code. Name the parameter dist and then tick the box to say that you understand you will need to update the invocations (calls) to this procedure. Click ok.

20. Delete the line of code that created the distance variable.

    The instances of the variable that existed in the code are now displayed in red.

21. To assign the value of the parameter to work within the code drag the name of the parameter (dist) to the red locations. In your code block.

22. Click on the myFirstMethod tab

23. The swim method call now has red value for the parameter within the procedure.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

24. Use the customDecimalNumber menu to change this to a value of 1.2.
25. Run your code to test it.

    You now have a procedure that will allow you to let any fish swim. The Blue Tang fish currently swims backwards and forwards. You can now use the swim procedure to make it look more realistic.
26. Drag a swim procedure from the Blue Tang to the top of the code. Use the same expression to make the fish swim 3 times its own length.
27. Disable the existing code that makes the Blue Tang fish move and run and test the code.
28. Delete the disabled code if you are happy with the results of your test.
29. The swim procedure was made for the clown fish where you could only see one side at a time. If you are going to use it for other fish then you will need to move both fins at the same time and also the tail. Adapt the swim procedure to match the following:

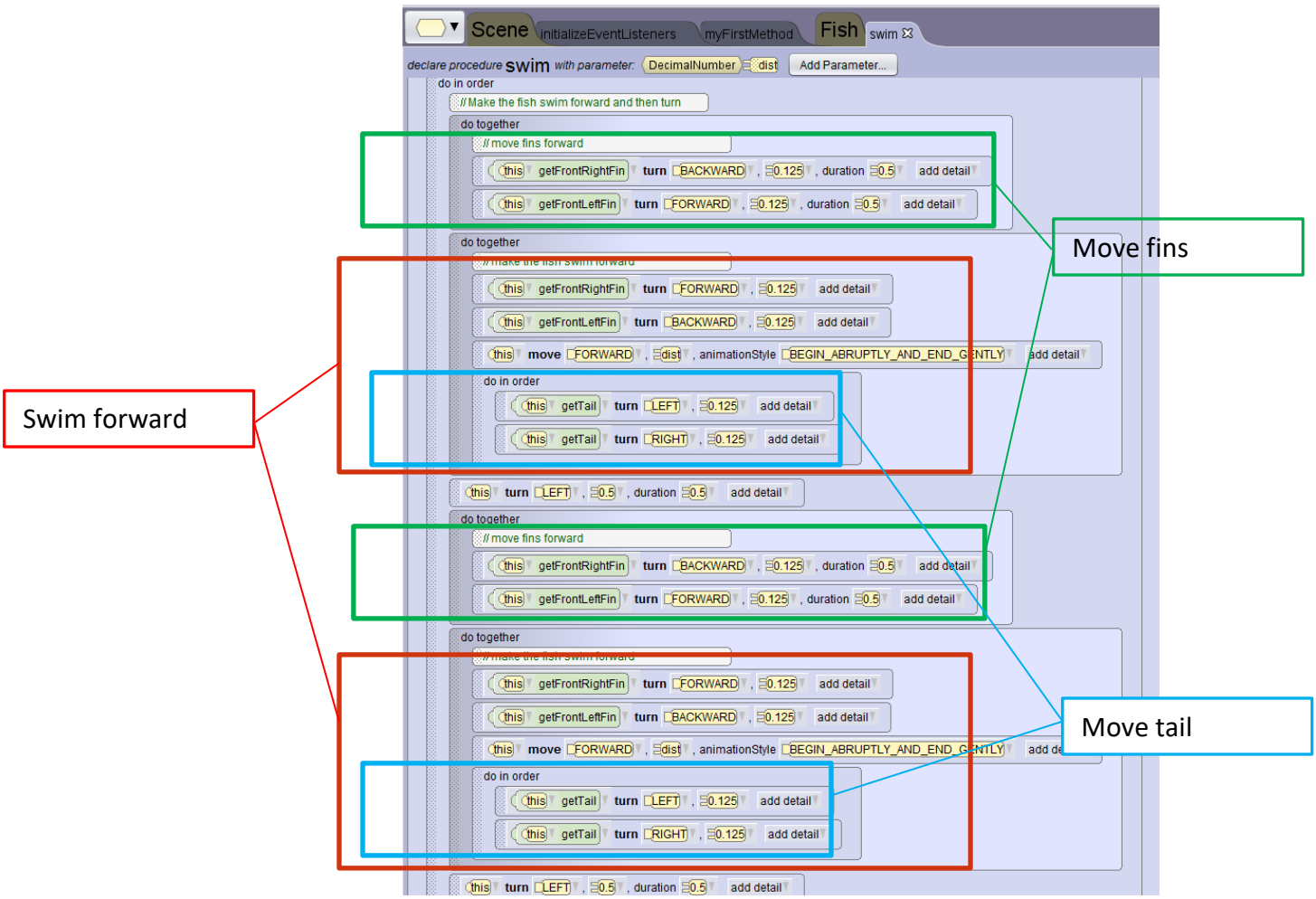
Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

30. Add additional comments to the code to explain what each section does.

The next section of this exercise is about cleaning up the code and removing any duplication of code where possible. If you look at the swim procedure you can see that the same code is there multiple times. This is something you should strive to avoid as a programmer.

What you are going to do is identify the repeating code and remove it to its own procedure.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

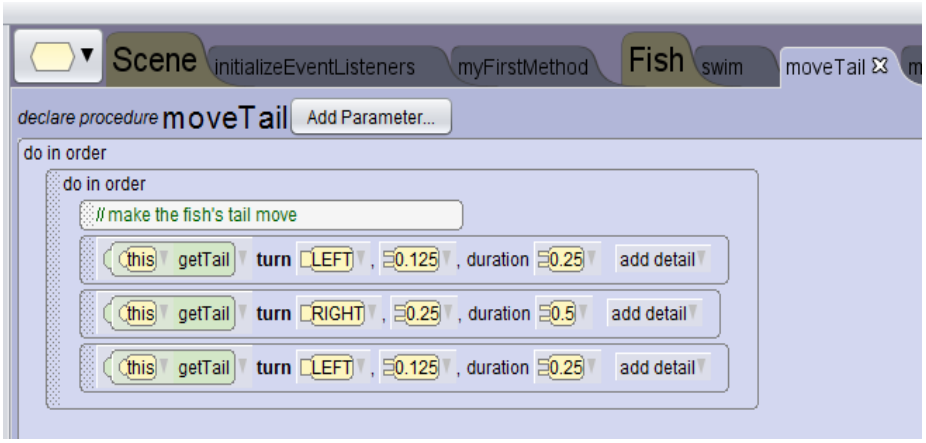


31. You can see that there are three parts to this code; Move fins forward, move tail and swim forward. These can all be removed and placed in their own procedures.

32. Using the classes button create a new procedure at the FISH level called moveTail.

33. Drag one of the do in order statement that controls the tail onto the clipboard.

34. Click on the moveTail tab and drag the code into it. This does a simple move that you can improve simply by creating a flow of events similar to the shakeHead procedure. Amend the code in the moveTail procedure to match the following:

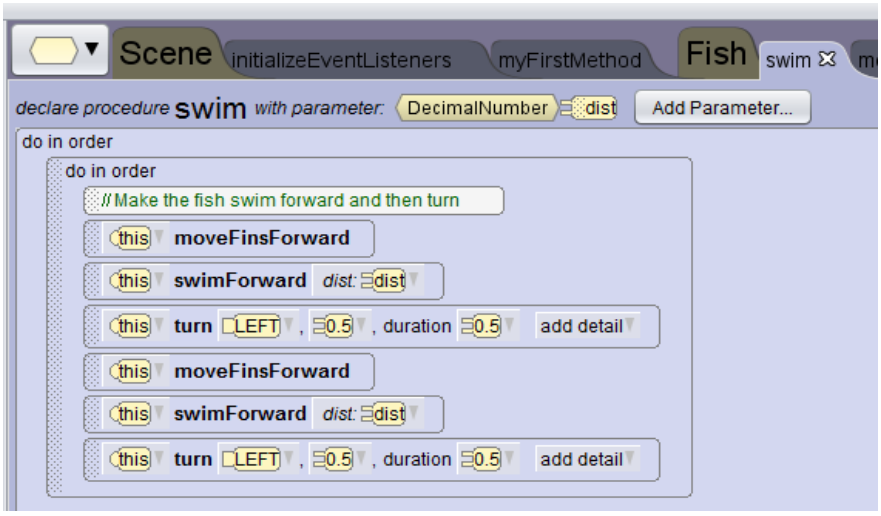


35. Click back on the swim tab.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

36. Remove the other code block that moved the tail and replace them with moveTail procedures
37. Create another FISH procedure and name it moveFinsForward
38. Drag one of the code blocks into the clipboard that moves the fins forward.
39. Click on the moveFinsForward tab and drag the code from the clipboard into the code editor.
40. Click on the swim tab and remove the code for moving the fins forward and replace anywhere that calls for the fins to be moved forward with the new moveFinsForward procedure.
41. Create another FISH procedure and name it swimForward
42. Drag one of the do together code blocks into the clipboard that moves the fish forward.
43. Click on the swimForward tab and drag the code from the clipboard into the code editor.
44. This one is slightly different because it requires a parameter so make a dist parameter here exactly as you did before. Drag the parameter name over the red label to complete the procedure.
45. Click on the swim tab and again replace any instances of a swim code block with the new swimForward procedure.

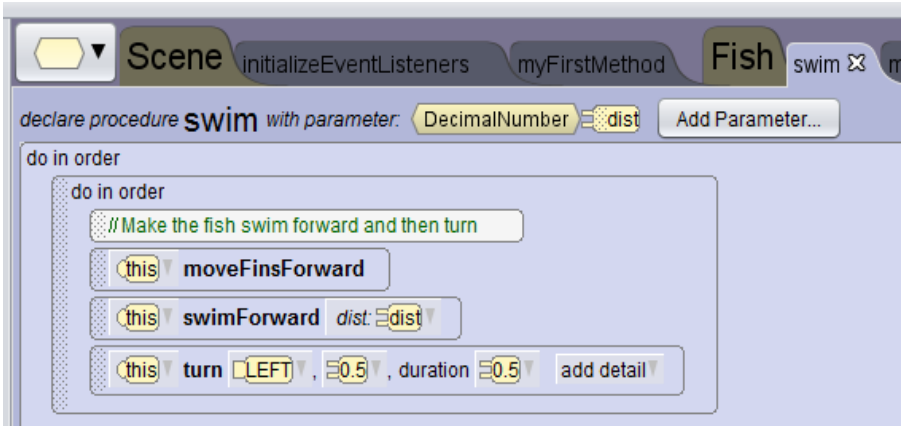
    When you drag it onto the code editor you will be asked to provide a value for the parameter. Choose dist from the list.
46. This leaves you with a swim driver class that puts all of the elements of swimming together from the different procedures available to it.



It also means that you have much more flexibility with what you can do with a fish.
One last thing to notice here is that the swim procedure makes the fish swim, turn around and swim again before making a final turn. What you can do here is reduce this code further by deleting the repeating code again.

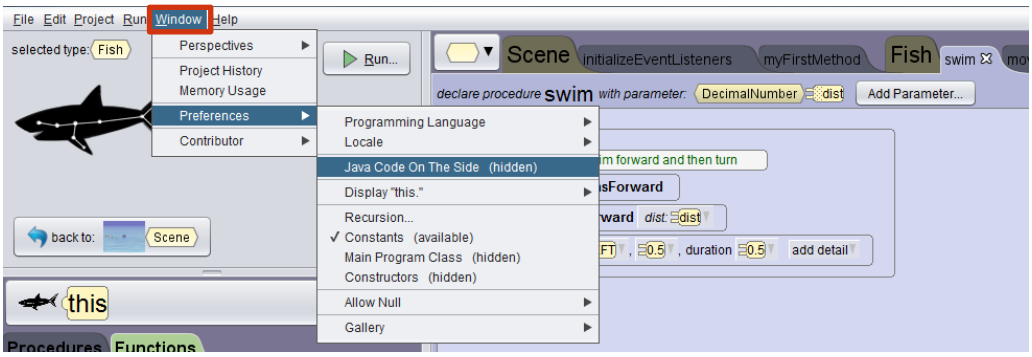
47. Delete the code lines until your swim procedure matches this:

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

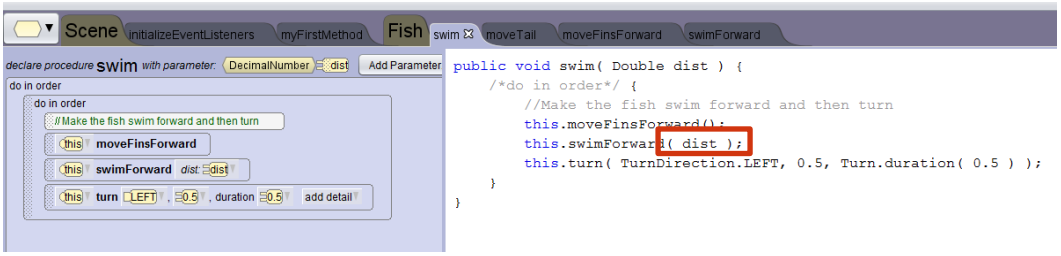


Much less here than when we started and much easier to read and understand. The difference here is that the fish will only carry out a single swim operation.

48. If you want to see how this code would look in Java then you can turn on the java code on the side from the window menu option.



It can be good to look at this code before you progress onto Greenfoot and Eclipse as it gives you an idea of how Java code looks and works.



You can see that the parameter dist is passed using brackets. This is how parameters are passed in Java. If you look at the turn method you can see that when passing multiple parameters they are separated by commas. Useful coding tips can be picked up by studying the Java code on the side option.

49. If you want to turn Java code on the side off just follow the same steps as turning it on.

50. To fix the fact that the fish only swims once and you need it to swim twice for the animation to work properly click on the myFirstMethod tab.

51. Now that the procedure only does a single swim you can make the code call it twice by using a count statement and choosing 2 as the argument. Place it above the swim procedure call and then drag the swim call in to it.

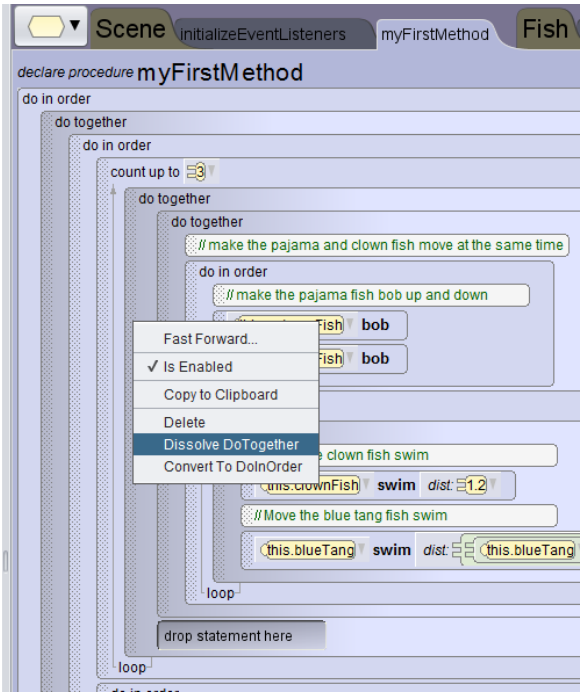
52. Make the Blue Tang fish move in the same way by dragging the Blue Tang swim procedure into the count loop statement beside the clown fish swim procedure.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

53. Change the value for the blue tang fish from 3 to 2 to stop it swimming off of the screen.

54. Add a do together statement so that both fish will swim at the same time.

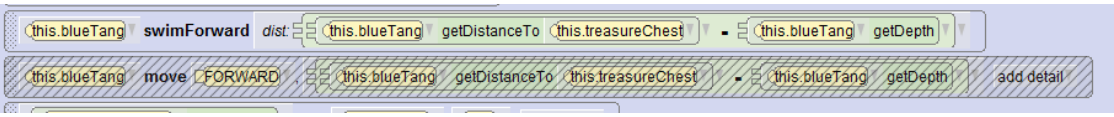
55. You can see that there are now two do together statements here one of which is now redundant. You can dissolve this by right clicking the outside do together statement and choosing dissolve from the menu.



56. The Blue Tang fish shakes its head and then says “no more swimming today!” It would be better if these actions were carried out together. In Alice you can easily convert between statements. Right click this do in order statement and choose convert to do together.

57. You have created multiple procedures that control how a fish moves in the water. You can now use these to make the fish’s movement more realistic.

    Any fish’s move forward procedure can now be replaced with a swimForward procedure to have the fins and tail move while the fish is swimming. Disable all fish move forward procedures and replace them with swimForward procedures using the same arguments.



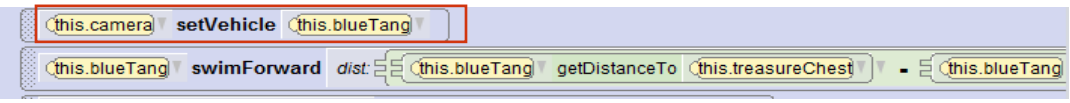
58. Delete all the disabled procedures when you have tested their swimForward replacements.

    Okay just a few extra cinematic additions before the project is finished and ready to be uploaded to YouTube.

59. When the Blue Tang fish moves towards the treasure chest I want the camera to move away at the same time. To do this you simply set the vehicle of the camera equal to the Blue Tang.

    Drag a setVehicle procedure from the Camera class onto the code editor before the Blue Tang fish moves towards the treasure chest.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

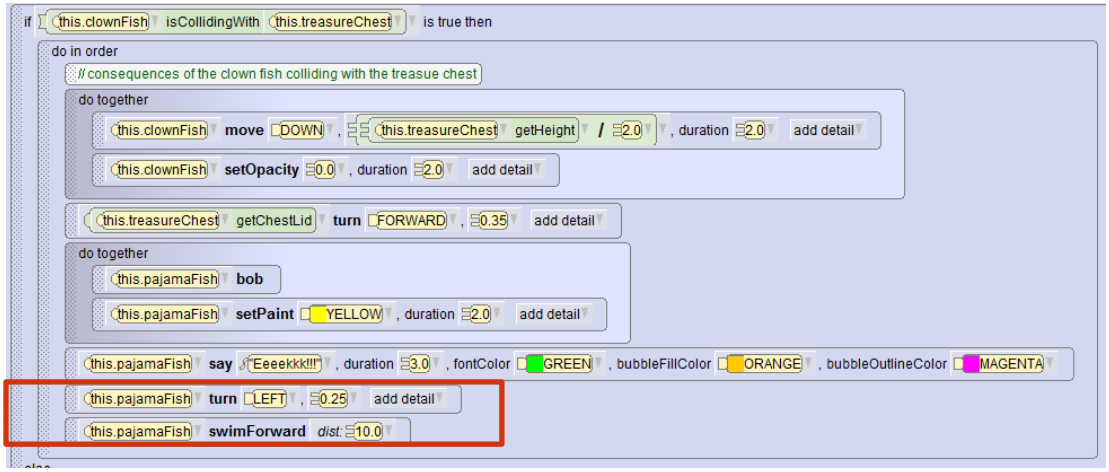


60. The camera should stay in its position before the Blue Tang swims away

    Use the CTRL + mouse drag to copy the camera set vehicle code and drop it above the turn code. Change the argument value to this. This releases the camera from the Blue Tang.

61. I want the camera to move in closer to the treasure chest while the Clown fish and Pajama fish are swimming towards it. Drag a moveToward statement into the do together code using the treasure chest and 2 as the arguments.

62. Instead of the Pajama fish just floating there at the end add a turn left procedure and a swimForward procedure using 10 as the argument to the end of the if statement.



    Okay second last element for this project. The treasure chest should wait a few seconds before returning back under the sea bed. While the chest is moving down the camera should pull away from the scene.

63. Add a do in order statement after the if statement.

64. Place a delay statement from the treasure chest into the do in order. Use 4 as the argument.

65. Add a do together statement under the delay statement.

66. Drag a move procedure from the treasure chest moving it down by the height of itself (use a function) and specify 5 as the duration.

    Drag a moveAwayFrom procedure from the camera and choose treasure chest and 5 as the arguments. Set the duration to 7.

67. Run and test your animation.

68. One final thing. Add some sound when the clown fish disappears. Choose the clown fish, drag a play audio procedure into the do together statement where the Clown fish disappears.

    Choose import audio, musical cues, and then intense as your audio.

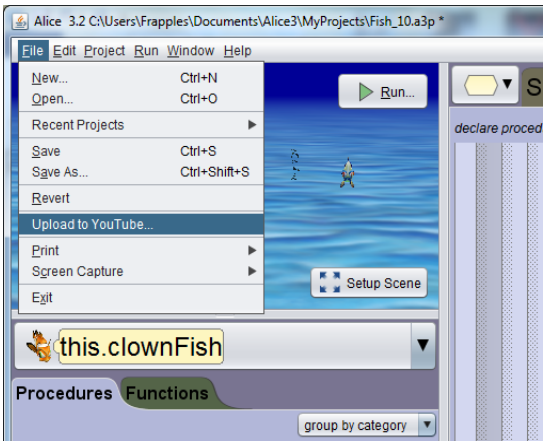
69. Test your animation.

70. Save your program.

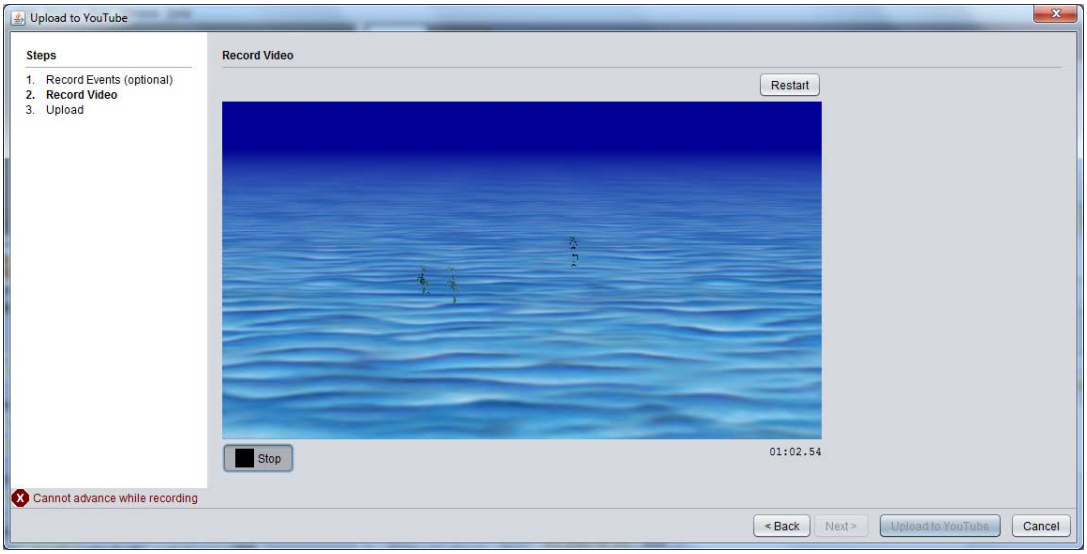
Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

71. Now that you have a complete animation you can upload your finished product to YouTube or export the file locally to your machine.

72. Click on File and then Upload to YouTube



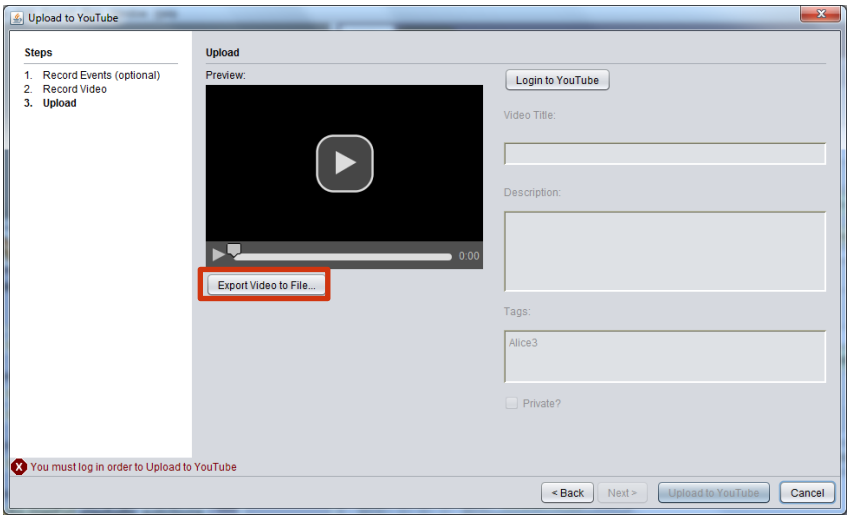
73. Press the record button to begin recording your animation. Click Stop once the camera has stopped moving at the end.



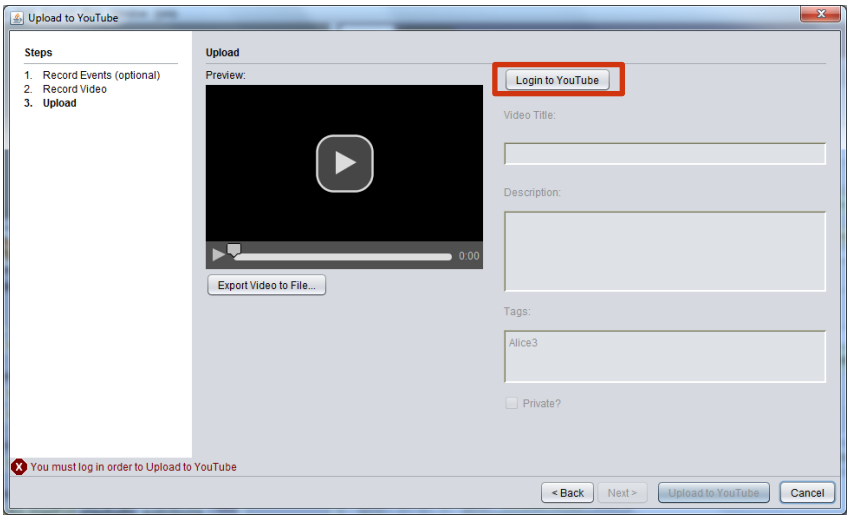
74. Click next

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

75. If you click Export Video to File you can create a local copy of your animation that you can play in a media application.



76. If you click log in to YouTube you can enter your YouTube details, give your file a description and some tags to help find it online and then upload your animation in one easy stage.



77. Save your program.

You have now completed your first complete animation and upload in Alice 3!!

78. Exit Alice 3.

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.