Getting Started with Java Using Alice

Add Rotation and Randomization

Objectives

This lesson covers the following objectives:

- Correlate storyboard statements with program execution tasks
- Add a control statement to the Code editor
- Use random numbers to randomize motion

Program Development Using a Top-Down Approach

Just as a homebuilder uses a set of blueprints and follows a series of steps, a programmer uses a plan and follows a series of steps to build a program.

Homebuilders work with a purpose and set specific goals as they build—the rooms are energy efficient, the home meets building codes, etc. Programmer's also work with a purpose and set goals, because without them, success cannot be measured.

Another term for the goal or purpose of an animation is a high-level scenario—a story with a purpose.

Steps to Use a Top-Down Approach to Programming

1. Define the high-level scenario (the purpose/goal of the program).
2. Document the actions step-by-step in a textual storyboard. This helps to gain a thorough understanding of the actions that need to be programmed.
3. Create a table to align the storyboard steps to the programming instructions.
4. Review the table during the animation's development to ensure the animation meets specifications.
5. Revise the table as necessary.

Textual Storyboard Example

Program the following actions in order:

Chicken walks by.

Cat turns to look at the chicken.

Cat says, “Dinner time!”

Chicken says, “Oh no!”

Chicken turns to right.

Program the following actions together:

Chicken walks away quickly.

Cat walks away quickly.

Align Storyboard to Programming Instructions

Order of Instructions	Storyboard Action	Programming Instructions
Do in order		
	Chicken walks by.	Chicken moves forward 2 meters.
	Cat looks at the chicken.	Cat turns to face chicken.
	Cat says, “Dinner time!”	Cat says “Dinner time!”
	Chicken says, “Oh no!”	Chicken says, “Oh no!”
	Chicken and cat turn right.	Chicken turns right 0.5 meters. Cat turns right 0.125 meters.
Do together	Cat chases chicken.	Chicken moves forward 2 meters. Cat moves forward 2 meters.

Object Movement

Object movement is egocentric: Objects move based on the direction they face.

An object can move in six directions:

- Up
- Down
- Forward
- Backward
- Right
- Left

Examples of Rotation Procedures

Procedure	Description
Turn	Rotates an object left, right, forward, or backward on its center point. Left and right turn on the object's vertical axis; forward and backward turn on the object's horizontal axis.
Roll	Rolls an object left or right on its center point using the object's horizontal axis.

Comparison of Turn and Roll Procedures

The object turning left and right on its center point using a vertical axis.


The object rolling left and right on its center point using a horizontal axis.


The object turning forward on its center point using a horizontal axis.

(The hare was already facing left. Turning forward caused it to lean forward)

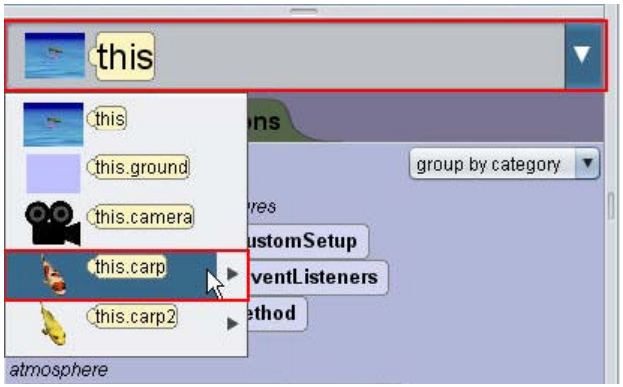
Sub-Part Rotation Example

Some objects have moveable sub-parts that can turn and roll. The pocket watch has moveable hour and minute hands that can roll on the clock's center point. The key to successful rotation is knowing the center point of an object.

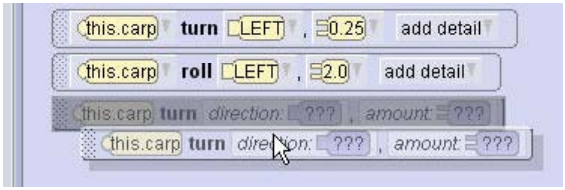


Steps to Program an Object to Rotate

1. Select the instance of the object to rotate.



2. Drag a turn or roll procedure to the Code editor.

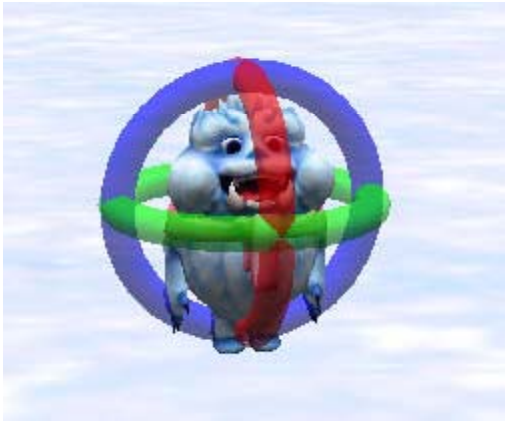


3. Set the direction argument.
4. Set the amount argument (1.0 = one full turn or roll).

Sub-Part Rotation

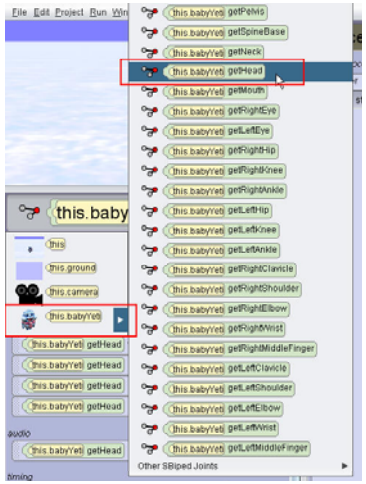
Some objects have moveable sub-parts.

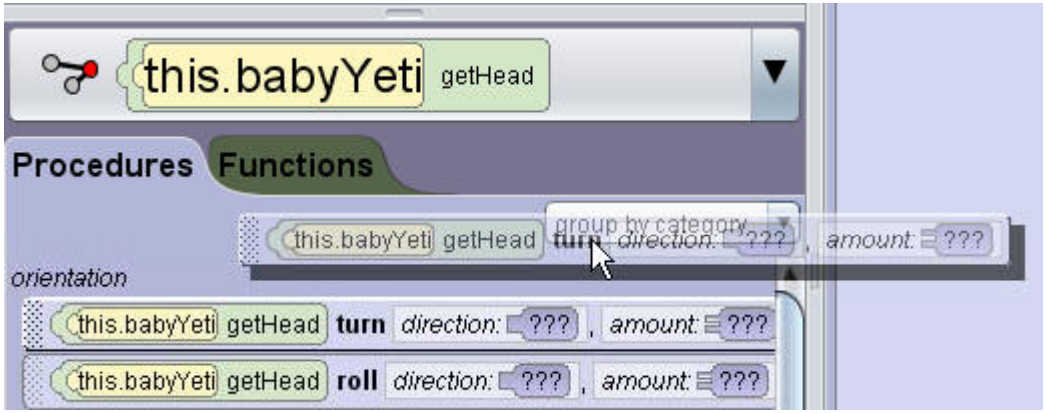
- For example, the baby Yeti's head can turn or roll.
- Rotation can be applied to an entire object, or select sub-parts of the object.
- An object's head displays rings that show its range of motion.



Steps to Program an Object's Sub-Part to Rotate

1. Select the instance of the object sub-part to rotate.
2. Drag a turn or roll procedure to the Code editor.





3. Select the direction value for the direction argument.
4. Select a rotation value for the amount argument (1.0 = one full turn or roll).

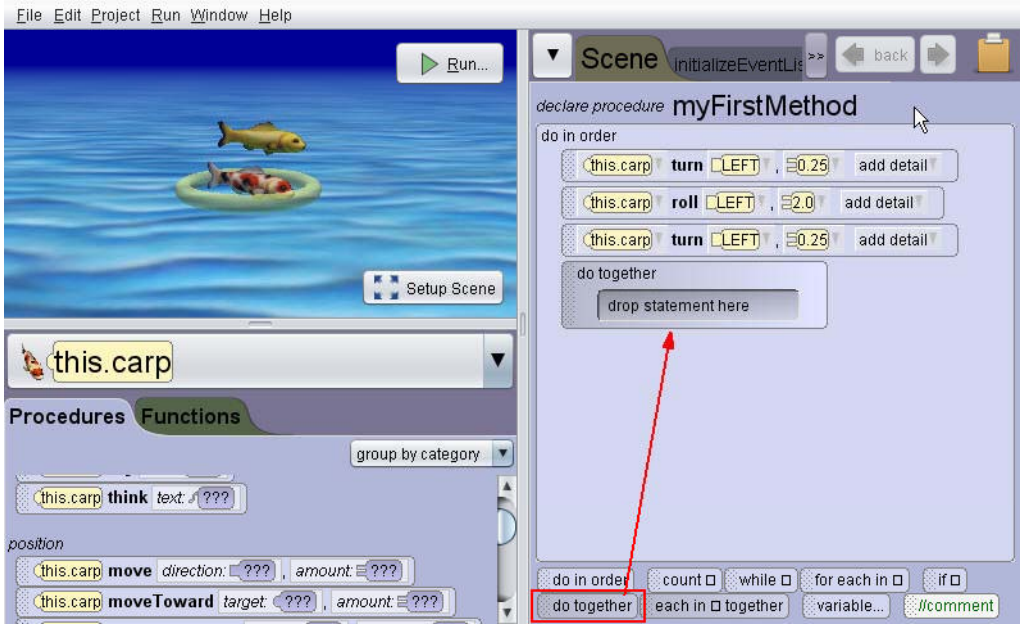
Control Statements

Control statements define how programming statements are executed in the program. myFirstMethod is created with a default Do in order control statement. Within it, all programming statements execute sequentially by default.

Procedure	Description
Do in order	Execute the statements contained within the control statement in sequential order.
Do together	Execute the statements contained within the control statement simultaneously.
Count	Execute the statements contained within the control statement a specific number of times.
While	Execute the statements contained within the control statement while a specified condition is true.

Control Statements in the Code Editor

Control statements are located at the bottom of the Code editor. They represent blocks of code into which individual statements are grouped. Drag control tiles into the Code editor, and then add statements to them.



Add Procedures to a Control Statement

Drag new or existing programming statements into the Control statement. In the example illustrated below, the carp will simultaneously move forward, turn left, and roll to the right.

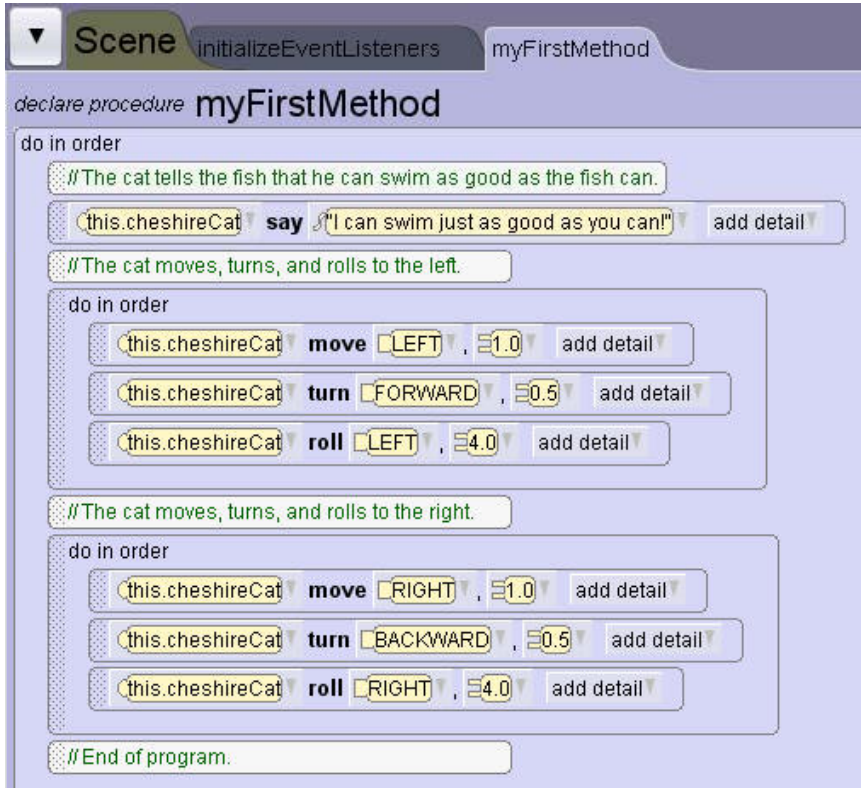
Tip: Create Programming Blocks

Even though myFirstMethod contains a default Do in order control statement, you can always add other Do in order tiles to the programming area. This allows you to keep your programming instructions together in organized blocks.

Blocks can be easily copied to the clipboard or moved around in a large program. Nested Do in order control statements do not impact the performance of the animation; they make program-editing easier for the programmer.

Nested Do In Order Control Statements

Nested statements add visual structure to a program, much like an outline brings structure to a report.



Here, two Do in order statements are nested inside a third Do in order statement.

Nesting is the process of putting one thing inside of another, like eggs lying next to each other in a nest.

Random Numbers

Random numbers are numbers generated by the computer with no predictable pattern to their sequence. Random numbers are generated within a given range of numbers.

Computers may require random numbers for:

- Security: for example, randomly generated passwords.
- Simulation: for example, earth science modeling (e.g., erosion over time).

Examples of Random Numbers

Random numbers can be generated within a wide range of numbers, both positive and negative, and may include whole numbers and fractional parts of a number.
Examples of random numbers:

- 15674
- -6934022
- 0.371
- -89.763

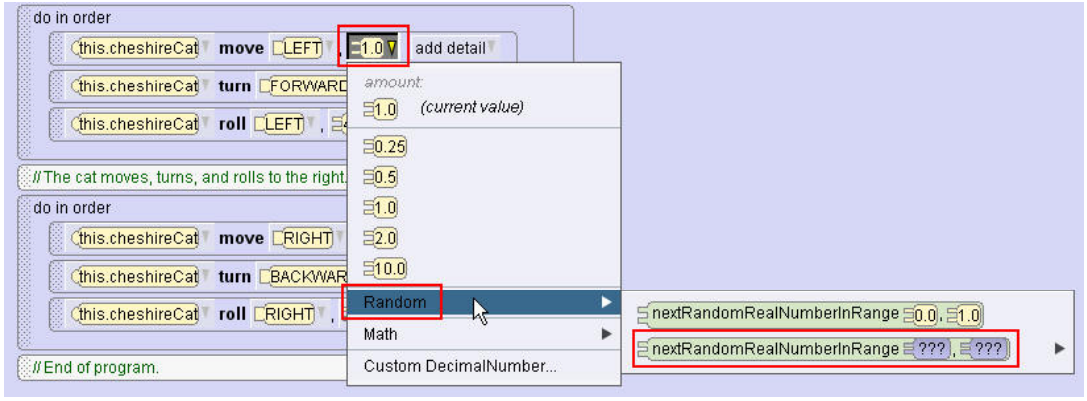
Random Numbers in Animations

Animals do not move in straight geometric lines. They change direction slightly as they walk, swim, and fly.

Random numbers may be utilized in the distance argument of a method so that the object moves in a less predictable, more life-like manner. Random numbers are often used when creating games that require unpredictable behavior.

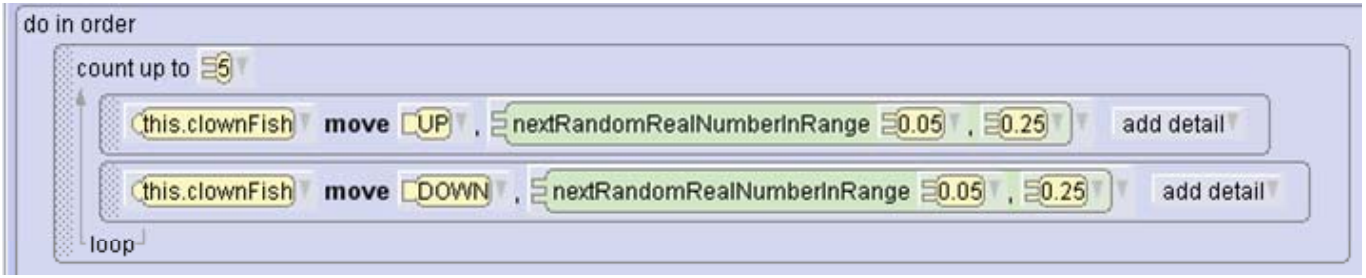
Steps to Use Random Numbers

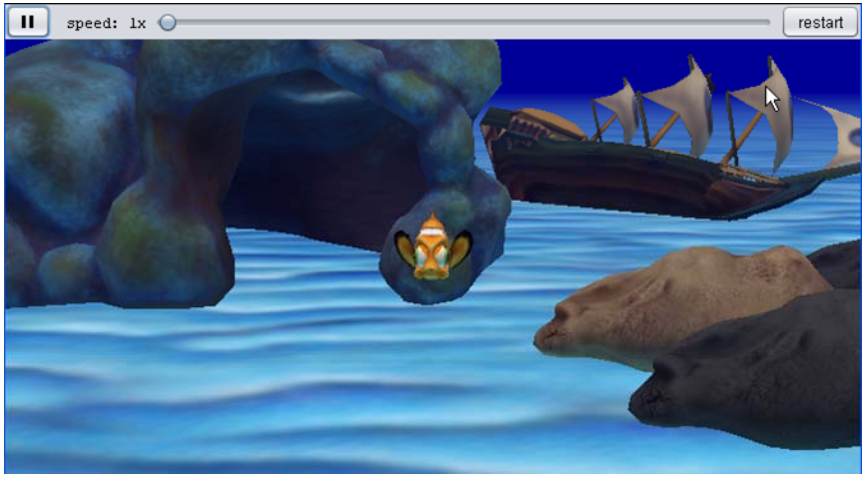
1. For any numerical argument, one of the options is a random number. Select the argument from the drop-down list.
2. Select Random.
3. Select nextRandomRealNumberInRange.
4. Select the low and high values.
5. Run the animation. Alice 3 will randomly generate a value within the selected range when the statement is executed.



Random Number Animation Example

The fish moves a random distance up and down at run-time.





Summary

In this lesson, you should have learned how to:

- Correlate storyboard statements with program execution tasks
- Add a control statement to the Code editor
- Use random numbers to randomize motion