Getting Started with Java Using Alice

Use Control Statements

Objectives

This lesson covers the following objectives:

- Define multiple control statements to control animation timing
- Create an animation that uses a control statement to control animation timing
- Recognize programming constructs to invoke simultaneous movement

Arguments

The arguments of a procedure may be edited or further defined to control object movement and timing.

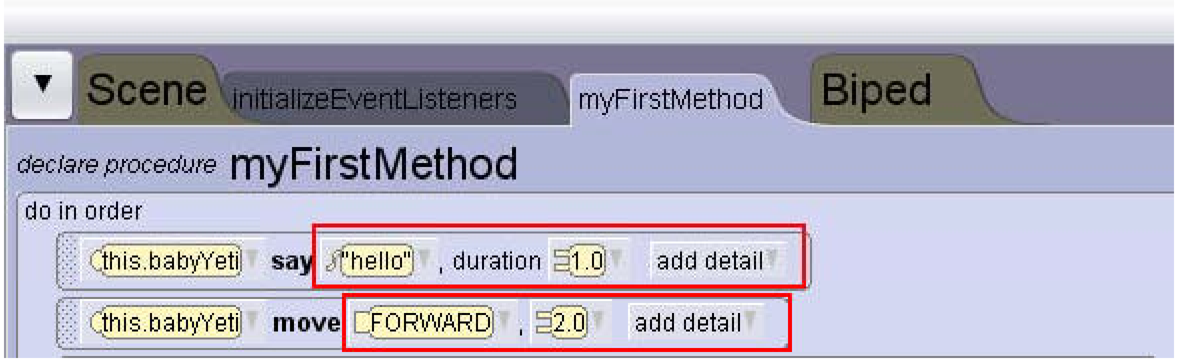
Examples of Alice 3 arguments include:

- Object
- Direction
- Direction amount
- Time duration

A computer program requires arguments to tell it how to implement the procedure.

Arguments Display

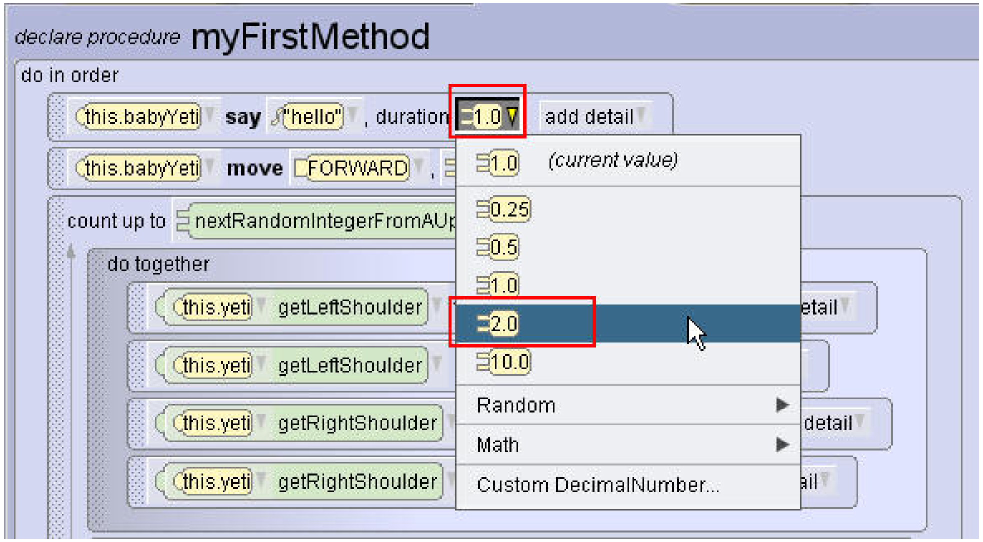
Below are examples of arguments in a procedure.



```
declare procedure myFirstMethod
do in order
    this.babyYeti say "hello" , duration 1.0 add detail
    this.babyYeti move FORWARD , 2.0 add detail
```

Steps to Edit Arguments

1. Next to the argument’s value, click the arrow to display the menu of values.
2. Select a new value. The menu indicates the current value, followed by pre-set values to choose from, followed by additional menu options to specify randomization, math calculations, or a custom decimal number.



```
declare procedure myFirstMethod
do in order
    this.babyYeti say "hello" , duration 1.0   add detail
    this.babyYeti move FORWARD ,
    count up to nextRandomIntegerFromAUp
        do together
            this.yeti getLeftShoulder                etail
            this.yeti getLeftShoulder
            this.yeti getRightShoulder               detail
            this.yeti getRightShoulder               ail
```

Selecting a Placeholder Argument

When adding a procedure to the code, you are required to select a value for each argument in the procedure.

Often times you may choose a pre-set value as a placeholder (a temporary value) which is later changed during an edit cycle.

Using a placeholder value is a common approach to creating and refining animation performance.

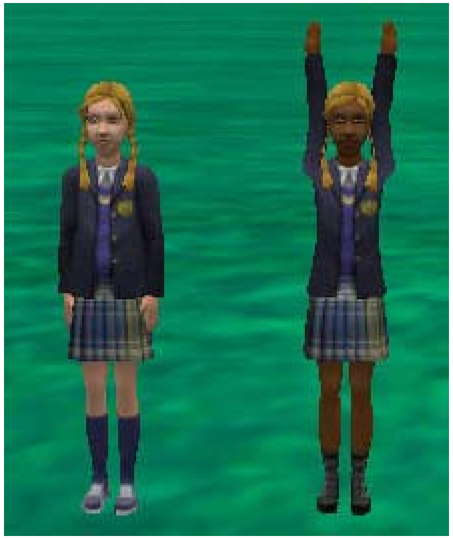
Simultaneous Movements

To create simultaneous movements for an object, use the Do together control statement.

Control Statement	Description
Do In Order	• Default control statement in Code editor. • Executes procedures in sequential order.
Do Together	•Executes procedures simultaneously. •Used for simultaneous movements such as walking and sitting motions.

Simultaneous Movements Example 1

For example, a movement executed together could be as simple as simultaneously raising both arms of a biped object from a hanging position to a straight arm overhead position.



Simultaneous Movements Example 2

Another example is a walking motion, which requires simultaneous movement of the hips and shoulders.

To create the walking motion for a biped, use:

- A series of move, roll, and turn procedures.
- Do together control statements.

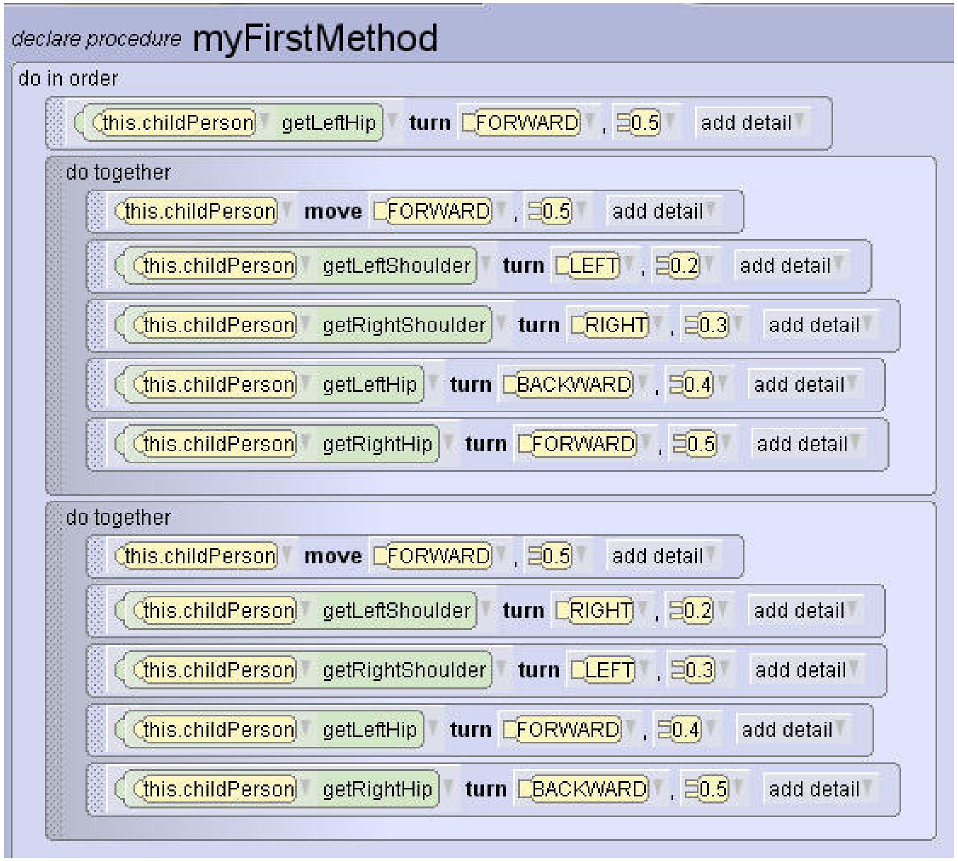
Different programming may be required for different objects because no two objects walk the same way.

Walking Textual Storyboard Example

Order of Instructions	Programming Instructions
	Left hip turns forward
Do Together	Whole body moves forward
	Left hip turns backward
	Right hip turns forward
	Left shoulder turns left
	Right shoulder turns right
Do Together	Whole body moves forward
	Left hip turns forward
	Right hip turns backward
	Left shoulder turns right
	Right shoulder turns left

Walking Motion Example

Examine this code for a simple walking motion.

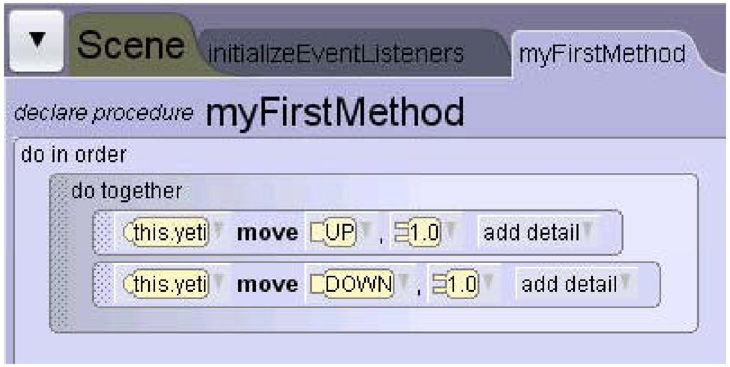


```
declare procedure myFirstMethod
do in order
    (this.childPerson getLeftHip) turn FORWARD , 0.5 add detail
    do together
        this.childPerson move FORWARD , 0.5 add detail
        (this.childPerson getLeftShoulder) turn LEFT , 0.2 add detail
        (this.childPerson getRightShoulder) turn RIGHT , 0.3 add detail
        (this.childPerson getLeftHip) turn BACKWARD , 0.4 add detail
        (this.childPerson getRightHip) turn FORWARD , 0.5 add detail
    do together
        this.childPerson move FORWARD , 0.5 add detail
        (this.childPerson getLeftShoulder) turn RIGHT , 0.2 add detail
        (this.childPerson getRightShoulder) turn LEFT , 0.3 add detail
        (this.childPerson getLeftHip) turn FORWARD , 0.4 add detail
        (this.childPerson getRightHip) turn BACKWARD , 0.5 add detail
```

When Procedures Offset Each Other

A common mistake is to include two procedures that will offset and cancel each other in a Do together construct.

For example, if you include a *move up 1 meter* procedure, followed by a *move down 1 meter* procedure in a Do together, nothing will happen. The procedures cancel each other out.



```
declare procedure myFirstMethod
do in order
    do together
        this.yeti  move  UP , 1.0  add detail
        this.yeti  move  DOWN , 1.0  add detail
```

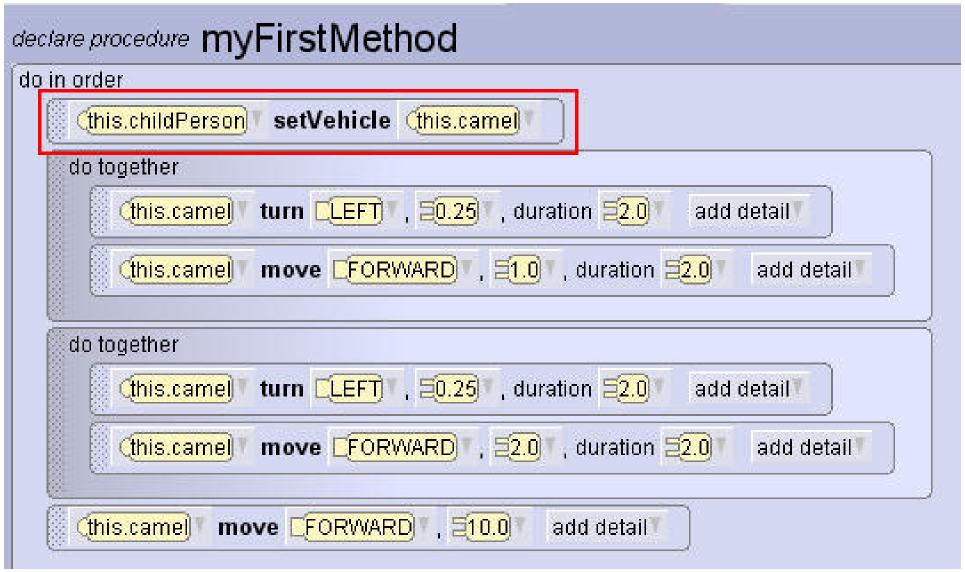
setVehicle Procedure

The setVehicle procedure employs the concept of a rider object and a vehicle object.

- The rider object is selected when the setVehicle procedure is used to specify the vehicle for the rider.
- Then, when the vehicle object is programmed to move, the rider object will automatically move with it.
- Examples:
  - Person rides a camel or horse.
  - Camera follows a helicopter to shoot the scene from the helicopter's point of view.

setVehicle Procedure Example 1

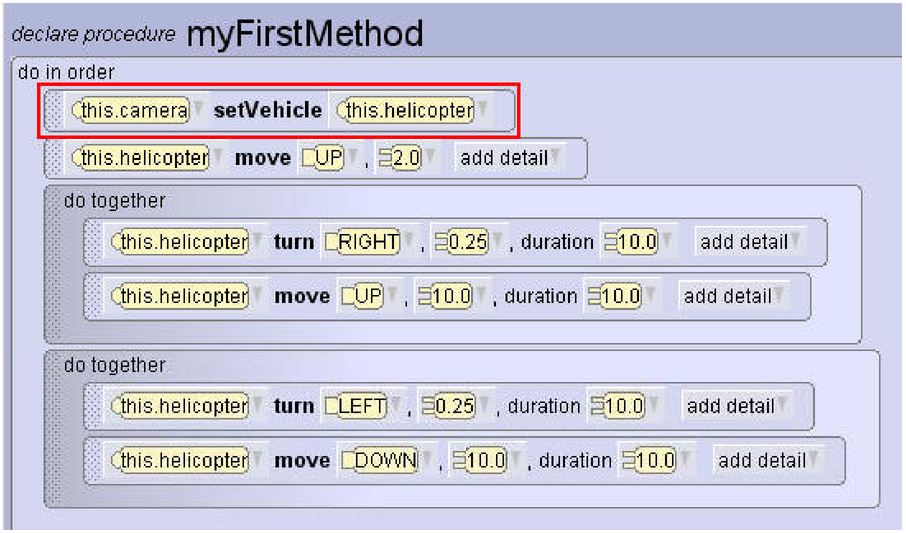
The child is positioned on the camel in the Scene editor. Then, the camel is set as the vehicle of the child in the Code editor. When the camel moves, the child stays on top and moves with the camel.



```
declare procedure myFirstMethod
do in order
    this.childPerson setVehicle this.camel
    do together
        this.camel turn LEFT , 0.25 , duration 2.0 add detail
        this.camel move FORWARD , 1.0 , duration 2.0 add detail
    do together
        this.camel turn LEFT , 0.25 , duration 2.0 add detail
        this.camel move FORWARD , 2.0 , duration 2.0 add detail
    this.camel move FORWARD , 10.0 add detail
```

setVehicle Procedure Example 2

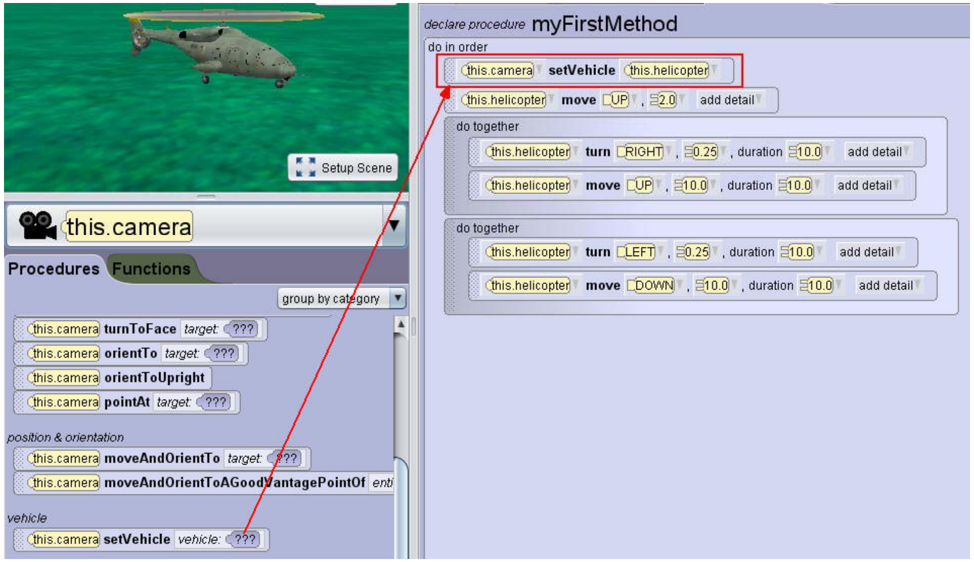
The helicopter is set as the vehicle of the camera in the Code editor. When the helicopter moves, the camera films the scene from the helicopter's perspective.



```
declare procedure myFirstMethod
do in order
    this.camera setVehicle this.helicopter
    this.helicopter move UP , 2.0 add detail
    do together
        this.helicopter turn RIGHT , 0.25 , duration 10.0 add detail
        this.helicopter move UP , 10.0 , duration 10.0 add detail
    do together
        this.helicopter turn LEFT , 0.25 , duration 10.0 add detail
        this.helicopter move DOWN , 10.0 , duration 10.0 add detail
```

Steps to Use setVehicle Procedure

1. Determine the vehicle object and the rider object.
2. In the Code editor, select the rider object from the Instance menu.
3. From the Procedures tab, drag the setVehicle procedure into the Code editor.
4. In the procedure, select the vehicle object from the menu.

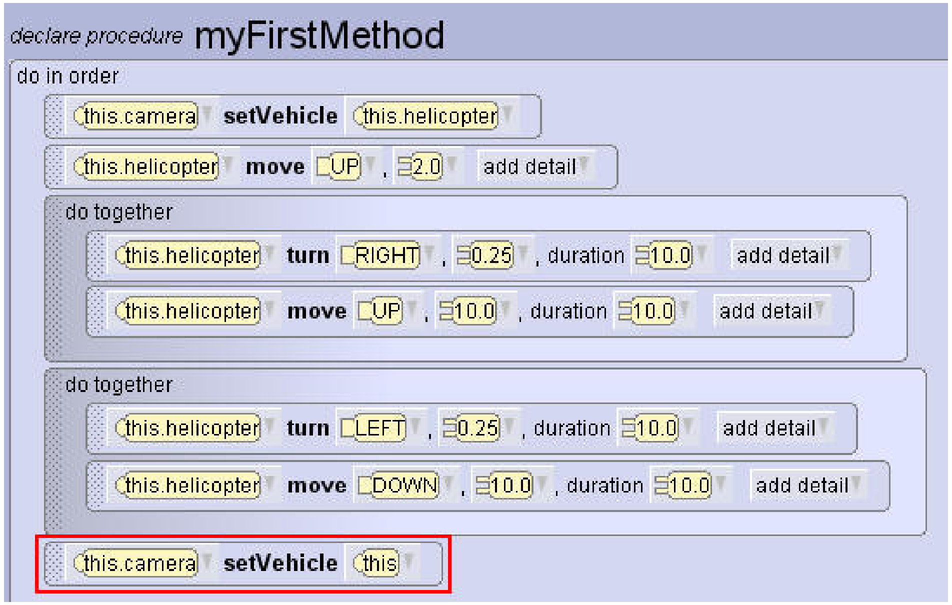


```
this.camera turnToFace target: ???
this.camera orientTo target: ???
this.camera orientToUpright
this.camera pointAt target: ???
position & orientation
this.camera moveAndOrientTo target: ???
this.camera moveAndOrientToAGoodVantagePointOf enti
vehicle
this.camera setVehicle vehicle: ???
```

```
declare procedure myFirstMethod
do in order
  this.camera setVehicle this.helicopter
  this.helicopter move UP , 2.0 add detail
  do together
    this.helicopter turn RIGHT , 0.25 , duration 10.0 add detail
    this.helicopter move UP , 10.0 , duration 10.0 add detail
  do together
    this.helicopter turn LEFT , 0.25 , duration 10.0 add detail
    this.helicopter move DOWN , 10.0 , duration 10.0 add detail
```

Steps to Stop setVehicle Procedure

1. If you want the rider object to get off the vehicle object, drag another setVehicle procedure into the Code editor at the point the rider should get off the vehicle.
2. Set the vehicle to *this*, which sets the vehicle of the rider back to the scene.



```
declare procedure myFirstMethod
do in order
    this.camera setVehicle this.helicopter
    this.helicopter move UP , 2.0 add detail
    do together
        this.helicopter turn RIGHT , 0.25 , duration 10.0 add detail
        this.helicopter move UP , 10.0 , duration 10.0 add detail
    do together
        this.helicopter turn LEFT , 0.25 , duration 10.0 add detail
        this.helicopter move DOWN , 10.0 , duration 10.0 add detail
    this.camera setVehicle this
```

Summary

In this lesson, you should have learned how to:

- Define multiple control statements to control animation timing
- Create an animation that uses a control statement to control animation timing
- Recognize programming constructs to invoke simultaneous movement