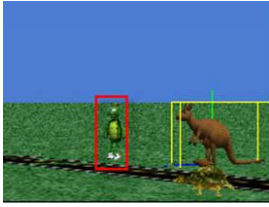


# Methods Tutorial: Part Two



By Deborah Nelson

Duke University

Professor Susan Rodger

June 16, 2008

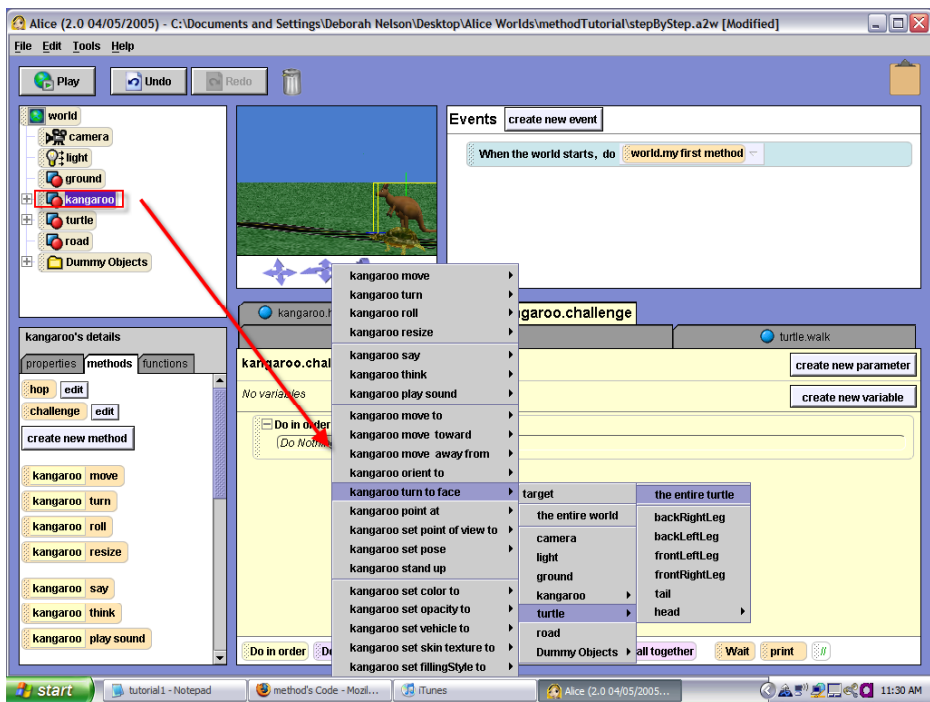
- We will now complete the world that you started in part one of the tutorial entitled "Methods." If you have not yet done Part One, you must go through that tutorial first.

## Loading the World

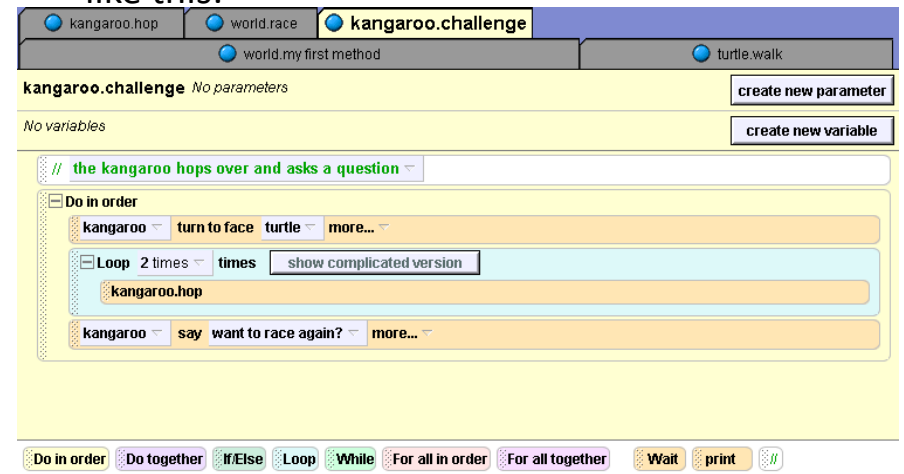
- For this tutorial, you can use your completed version from part one of the Methods tutorial. That world was entitled methodStart.a2w.
- Or you can download the version of the world with the solution from part one. Remember to save it in a directory that you can find again, and then start Alice and open the world.
- NOTE: You cannot double-click the file to open it; Windows will not know what to use, and even if you select Alice from a list of programs, the loading will fail.

## Part 1: Parameters

- Now that the kangaroo and the turtle have raced, let's make a method for the kangaroo to hop back to the turtle and challenge him to a race again.
- Click on the kangaroo, create new method, and name it 'challenge'. Drag a 'do in order' into your method.
- Then, click on kangaroo in the object area, drag it into the method and select the method 'turn to face' - select the turtle - the entire turtle.
- See the screenshot on the next slide for an illustration



- When you finish dragging all of the instructions into your method, it should look like this:



## Why use parameters

- If we save the kangaroo and put it in a world where there is no turtle, what will happen? Alice will crash because the method kangaroo.challenge refers to a non-existent object. A class-level method should not have any references to other characters or world-level methods.
- In other words, instead of referring to the turtle in the first instruction of this method, we're going to use a parameter. A parameter is a place holder.

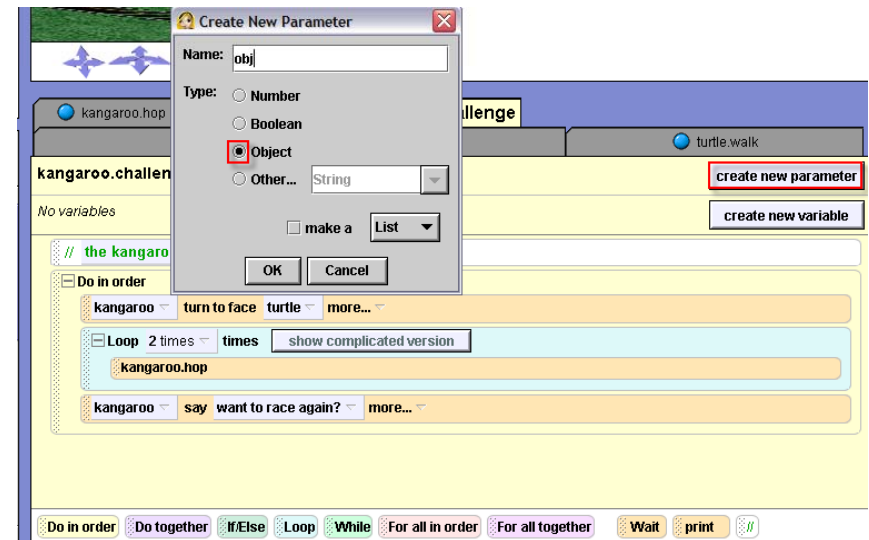
## An example scenario

- For example, in another world, you may want your kangaroo to be able to challenge a turtle or a bunny or a penguin. We could write three different methods: one for the kangaroo to challenge the turtle, another for the kangaroo to challenge the bunny and a separate one for the kangaroo to challenge the penguin.

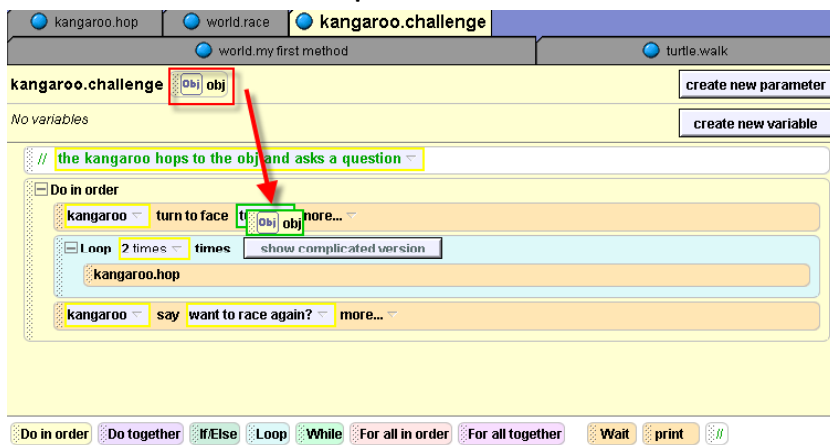
## How to create a parameter

- In this case, a parameter is going to be a placeholder for an object that the kangaroo will challenge - such as a bunny, a penguin or a turtle.
- Click on 'create new parameter' and name it 'obj'. Before you click okay, make sure you select the type 'object'
- See the screenshot on the next slide for an illustration

## How to create a parameter (cont 1)



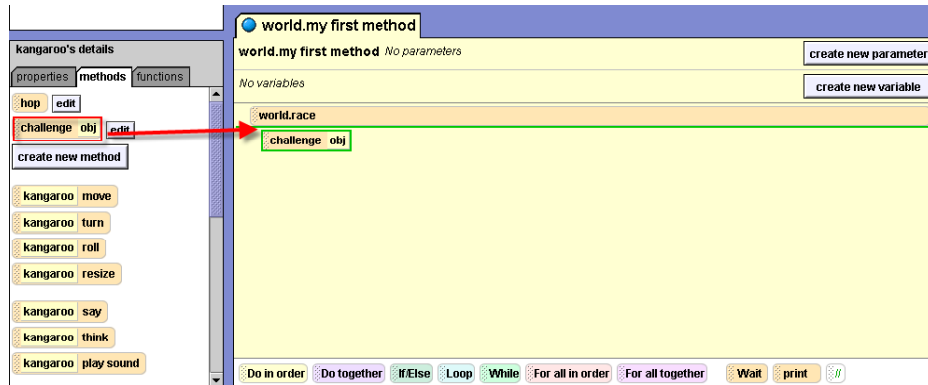
- Now, you can see that the 'obj' parameter has appeared beside the name of the method. I've highlighted it with a red box. Drag 'obj' into the method to replace 'turtle'.



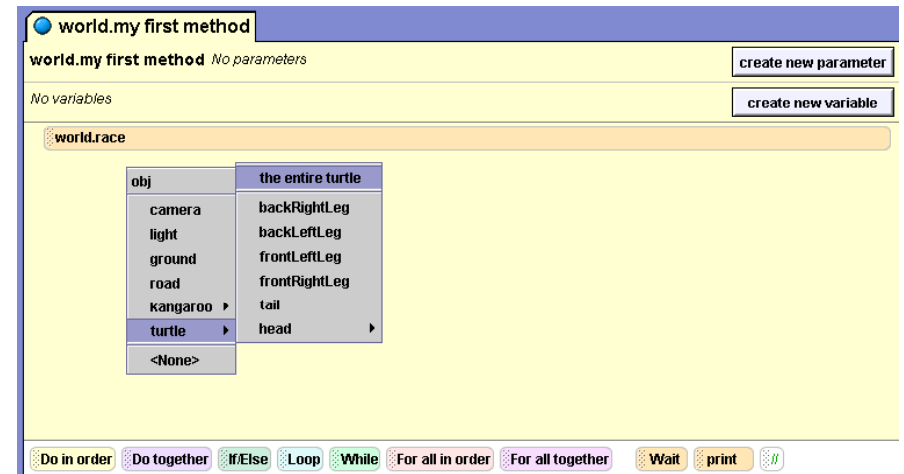
## How to call a method that has a parameter

- To test your code, drag kangaroo.challenge into world.myfirstmethod underneath the world.race method that is already there.
- When you drag kangaroo.challenge into the method, once you release your mouse you will have to select turtle, then entire turtle as the obj to replace parameter. See the screenshots on the next two slides for an illustration:

## Dragging kangaroo.challenge into world.race



- Selecting the turtle as the parameter argument:

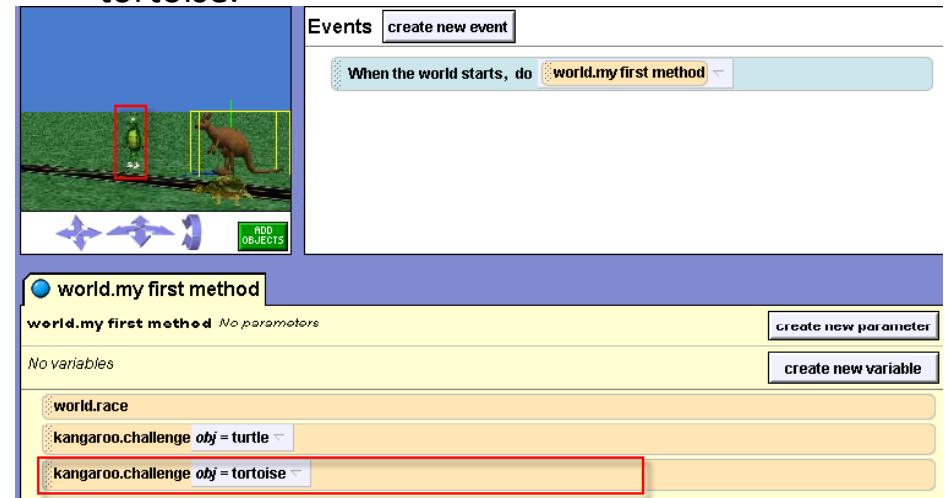


- Press the play button to test your world.

## Testing kangaroo.challenge on another object

- To reinforce your understanding of parameters, let's call the method on another object. Add the tortoise (from the Animal folder) to your world by clicking on the Add objects button. Below, I've highlighted in red where I placed the tortoise.
- Drag kangaroo.challenge into your world.myfirstmethod and select the tortoise as the parameter.
- See the screenshot on the next slide for an illustration

- Play your world. Now after the race, the kangaroo challenges the turtle and then the tortoise.



## Testing kangaroo.challenge (cont 1)

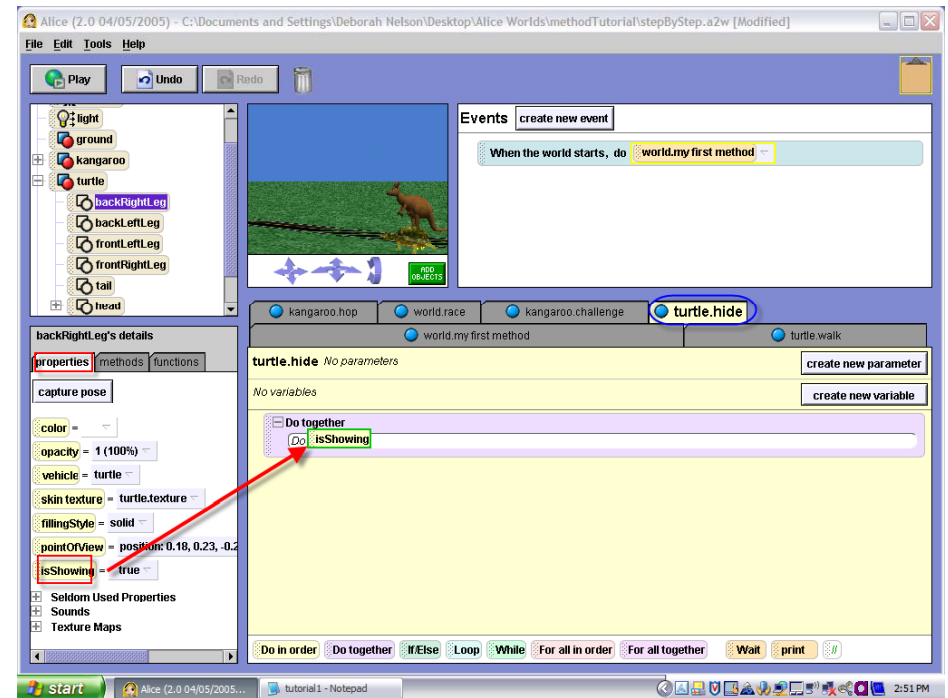
- Depending on where you placed the tortoise in your world, you may notice that having the kangaroo hop twice toward him does not look very good. Once you know how to use the built in function 'distance to' you can improve the appearance of this method. For now, don't worry about it.
- Let's finish making the rest of our world. In world.myfirstmethod, delete the second call to kangaroo.challenge for the tortoise. If you want, you can delete the entire tortoise from your world.

## Part 2: Properties

- Finally, we want to write a method to make the turtle go into his shell. Click on turtle in the object tree. Click on the method tab and create a new method named 'hide' (If you use the world given to you as a starter world, hide will have already been created, but there is no code in it).
- We are going to make all of the turtle's body parts invisible at the same time, except for his shell.
- To do this, first drag a 'do together' into the 'hide' method.

### Creating the turtle.hide method

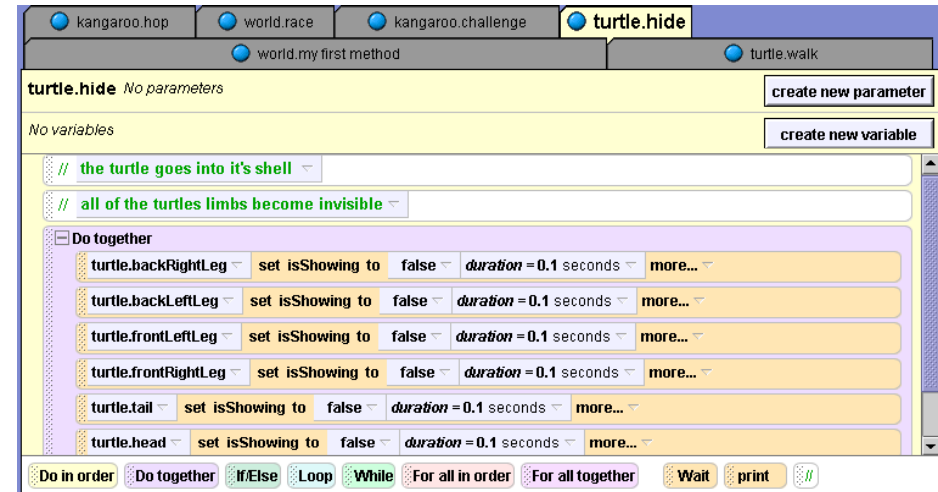
- Then, click on the + beside turtle in the object tree. Click on the 'backRightLeg'.
- In the details area: Click on the properties tab. Click on 'isShowing' and drag it into the 'do together'.
- Set the value to false. Click on the 'more' and set duration to 0.1
- See the screenshot on the next slide for an illustration



## Turtle.hide method (cont 1)

- Do the same thing for each of the body parts by clicking on each of these in the object tree - backLeftLeg, frontLeftLeg, frontRightLeg, tail and head - and dragging the isShowing property of each into the turtle.hide method.
- Your code should look like the screenshot on the following slide:

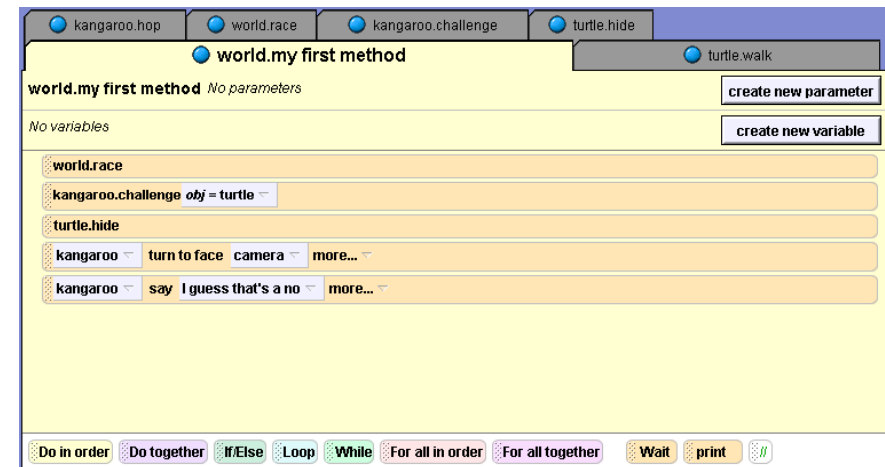
## The code for turtle.hide (cont 2)



## Turtle.hide (cont 3)

- Now drag the turtle.hide method into your world.myfirstmethod underneath kangaroo.challenge.
- If you want, you can have the kangaroo say something at the end.

- Here is what my final code in world.myfirstmethod:



- Press play to watch your entire simulation.

## Recap

- If you want to write a method in which an object interacts with another character, you can either write a world-level method or write a class-level method with parameters
- A class-level method with parameters is a good choice if you want to be able to save your object out so that it can perform your new method in different worlds.

## Recap continued

- Keep in mind that parameters are not only used in class-level methods. For example, if you have five characters in your world and you want them to all flip together, you can write one world level method with an object parameter that flips. Then in a do together, call the method for each of the objects in your world.
- You can change certain properties of an object while you're writing a method