

# Methods Tutorial: Part Two

By Deborah Nelson

Duke University

Professor Susan Rodger

June 16, 2008

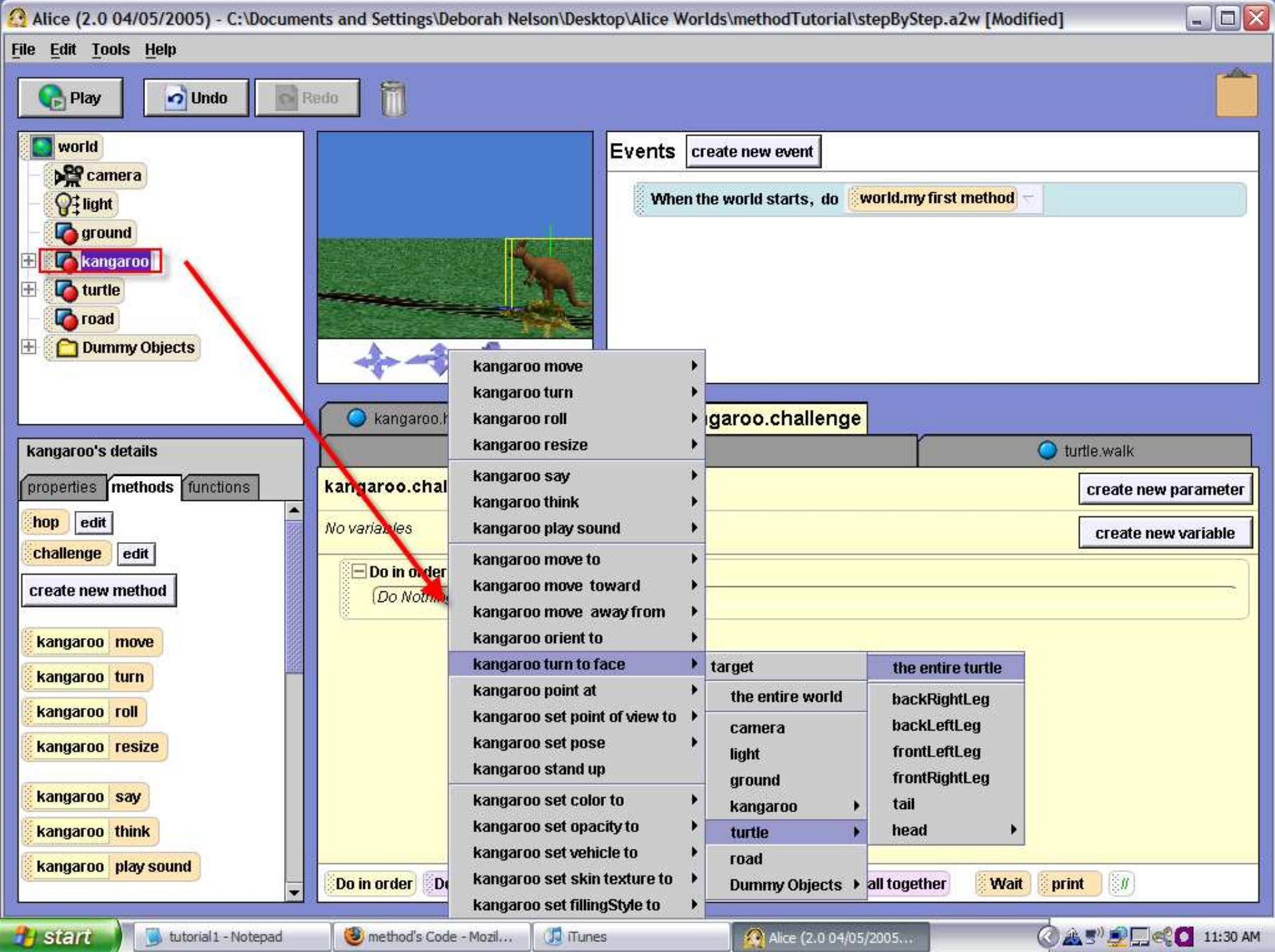
- We will now complete the world that you started in part one of the tutorial entitled "Methods." If you have not yet done Part One, you must go through that tutorial first.

# 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:

The image shows a Scratch code editor interface. At the top, there are several method tabs: 'kangaroo.hop', 'world.race', 'kangaroo.challenge' (which is selected and highlighted in yellow), and 'turtle.walk'. Below the tabs, the 'kangaroo.challenge' method is open, showing 'No parameters' and a 'create new parameter' button. Below that, it shows 'No variables' and a 'create new variable' button. The main code area contains a comment: '// the kangaroo hops over and asks a question'. Below the comment is a 'Do in order' block containing three instructions: 'kangaroo turn to face turtle more...', a 'Loop 2 times times' block containing 'kangaroo.hop', and 'kangaroo say want to race again? more...'. At the bottom of the editor, there is a palette of code blocks: 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and '//'. The 'Loop' block is highlighted in the palette.

## 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)

The image shows a Scratch workspace with a 'Create New Parameter' dialog box open. The dialog box has the following fields and options:

- Name:** A text input field containing the text 'obj'.
- Type:** A group of radio buttons with the following options:
  - Number
  - Boolean
  - Object (This option is highlighted with a red square in the image)
  - Other... String (with a dropdown arrow)
- make a List (with a dropdown arrow)
- Buttons:** 'OK' and 'Cancel' at the bottom.

The background workspace shows a script area with the following elements:

- A comment block: `// the kangaroo`
- A 'Do in order' block containing:
  - A 'kangaroo turn to face turtle more...' block.
  - A 'Loop 2 times times' block containing a 'kangaroo.hop' block.
  - A 'kangaroo say want to race again? more...' block.
- A 'challenge' block with a 'turtle.walk' block.
- Two buttons on the right: 'create new parameter' (highlighted with a red rectangle) and 'create new variable'.
- A 'No variables' label.
- A bottom toolbar with icons for 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and a comment icon.

- 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**.

The screenshot shows a Scratch-like programming environment. At the top, there are several tabs: 'kangaroo.hop', 'world.race', 'kangaroo.challenge' (selected), 'world.my first method', and 'turtle.walk'. Below the tabs, the 'kangaroo.challenge' method editor is open. It has a parameter 'Obj obj' highlighted with a red box. To the right of the parameter are buttons for 'create new parameter' and 'create new variable'. Below the parameter is the text 'No variables'. The script area contains a comment: '// the kangaroo hops to the obj and asks a question'. Below the comment is a 'Do in order' block containing three blocks: 'kangaroo turn to face to Obj obj more...', a loop with '2 times' and 'show complicated version' (containing 'kangaroo.hop'), and 'kangaroo say want to race again? more...'. A red arrow points from the 'Obj obj' parameter in the method editor to the 'Obj obj' parameter in the 'turn to face' block. At the bottom, there is a palette with various programming blocks: 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and '//'. The 'Obj obj' parameter in the 'turn to face' block is highlighted with a green box.

## 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

The image shows a programming environment with a sidebar on the left and a main workspace on the right. The sidebar is titled "kangaroo's details" and has three tabs: "properties", "methods", and "functions". The "methods" tab is active, showing a list of methods for the "kangaroo" object. The "challenge obj" method is highlighted with a red box, and a red arrow points from it to the "challenge obj" block in the workspace. The workspace is titled "world.my first method" and shows a "world.race" block with a "challenge obj" block attached to it. The workspace also has buttons for "create new parameter" and "create new variable". At the bottom of the workspace, there are several control blocks: "Do in order", "Do together", "If/Else", "Loop", "While", "For all in order", "For all together", "Wait", "print", and a comment block.

kangaroo's details

properties methods functions

hop edit

challenge obj edit

create new method

kangaroo move

kangaroo turn

kangaroo roll

kangaroo resize

kangaroo say

kangaroo think

kangaroo play sound

world.my first method

world.my first method No parameters

create new parameter

No variables

create new variable

world.race

challenge obj

Do in order Do together If/Else Loop While For all in order For all together Wait print //

- Selecting the turtle as the parameter argument:

The screenshot shows a programming environment with a method definition for 'world.my first method'. The method has no parameters and no variables. A dropdown menu is open for the parameter argument, showing a list of objects: 'camera', 'light', 'ground', 'road', 'kangaroo', 'turtle', and '<None>'. The 'turtle' object is selected, and its sub-menu is also open, showing 'backRightLeg', 'backLeftLeg', 'frontLeftLeg', 'frontRightLeg', 'tail', and 'head'. The 'head' option is highlighted. At the bottom of the interface, there are several control buttons: 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and a play button.

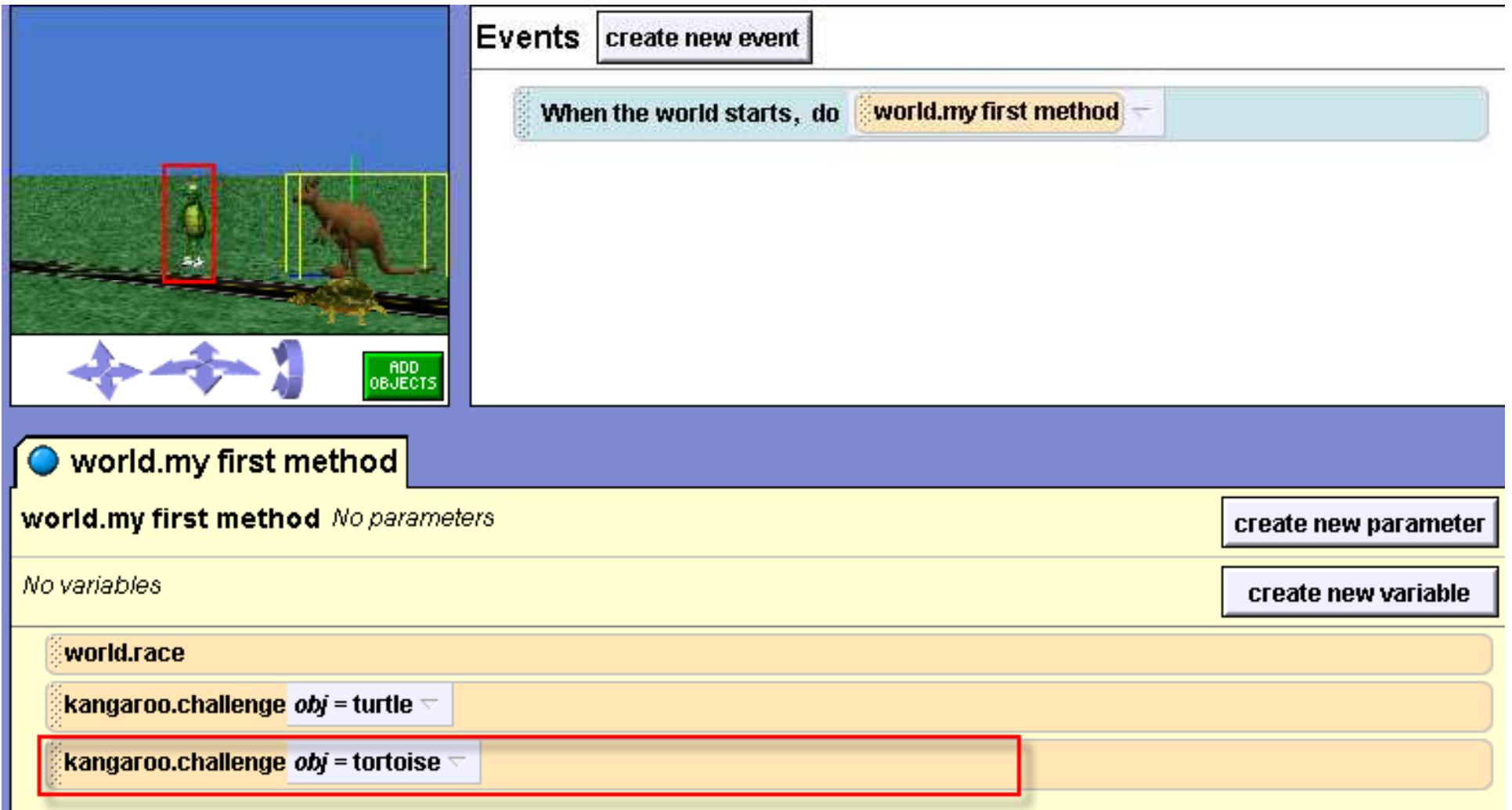
obj	the entire turtle
camera	backRightLeg
light	backLeftLeg
ground	frontLeftLeg
road	frontRightLeg
kangaroo ▶	tail
<b>turtle ▶</b>	<b>head ▶</b>
<None>	

- 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.



The screenshot shows a 3D environment with a kangaroo and a turtle on a green field. The turtle is highlighted with a red box. Below the environment are navigation arrows and an 'ADD OBJECTS' button. The 'Events' panel shows a 'When the world starts, do' event with a 'world.my first method' block. The 'Scripts' panel shows a 'world.my first method' block with 'No parameters' and 'No variables' options. The 'Blocks' panel shows three 'kangaroo.challenge' blocks, with the last one, 'kangaroo.challenge obj = tortoise', highlighted with a red box.

Events

When the world starts, do

world.my first method

world.my first method *No parameters*

*No variables*

# 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

Alice (2.0 04/05/2005) - C:\Documents and Settings\Deborah Nelson\Desktop\Alice Worlds\methodTutorial\stepByStep.a2w [Modified]

File Edit Tools Help

Play Undo Redo

light  
ground  
kangaroo  
turtle  
backRightLeg  
backLeftLeg  
frontLeftLeg  
frontRightLeg  
tail  
head

Events create new event

When the world starts, do world.my first method

kangaroo.hop world.race kangaroo.challenge **turtle.hide** turtle.walk

**backRightLeg's details**  
properties methods functions

capture pose

color =  
opacity = 1 (100%)  
vehicle = turtle  
skin texture = turtle.texture  
fillingStyle = solid  
pointOfView = position: 0.18, 0.23, -0.2  
**isShowing = true**

Seldom Used Properties  
Sounds  
Texture Maps

turtle.hide No parameters create new parameter

No variables create new variable

Do together  
Do isShowing

Do in order Do together If/Else Loop While For all in order For all together Wait print

The screenshot shows the Alice 2.0 software interface. At the top, the window title is "Alice (2.0 04/05/2005) - C:\Documents and Settings\Deborah Nelson\Desktop\Alice Worlds\methodTutorial\stepByStep.a2w [Modified]". Below the title bar is a menu bar with "File", "Edit", "Tools", and "Help". A toolbar contains "Play", "Undo", "Redo", and a trash icon. The main workspace is divided into several panels. On the left is a "Scene" panel showing a 3D environment with a kangaroo and a turtle. Below it is a "Properties" panel for the selected object, "backRightLeg". The "properties" tab is active, showing various attributes like "color", "opacity", "vehicle", "skin texture", "fillingStyle", "pointOfView", and "isShowing". The "isShowing" property is set to "true" and is highlighted with a red box. A red arrow points from this box to the "isShowing" block in the "Do together" block of the "turtle.hide" method. The "Events" panel on the right shows a single event: "When the world starts, do world.my first method". Below the workspace is a "Methods" panel with several methods listed: "kangaroo.hop", "world.race", "kangaroo.challenge", "turtle.hide" (circled in blue), and "turtle.walk". The "turtle.hide" method is selected, showing its details: "No parameters", "No variables", and a "Do together" block containing a "Do isShowing" block. At the bottom of the workspace is a "Logic" panel with various control blocks: "Do in order", "Do together", "If/Else", "Loop", "While", "For all in order", "For all together", "Wait", and "print". The Windows taskbar at the bottom shows the "start" button, the current application "Alice (2.0 04/05/2005...)", and another application "tutorial 1 - Notepad". The system clock shows "2:51 PM".

## 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)

The image shows a Scratch code editor window with several tabs: kangaroo.hop, world.race, kangaroo.challenge, turtle.hide (selected), world.my first method, and turtle.walk. The main workspace displays the code for the turtle.hide function. It starts with two green flag click events: "// the turtle goes into it's shell" and "// all of the turtles limbs become invisible". The second event is followed by a "Do together" loop containing six "set isShowing to false" blocks, each with a duration of 0.1 seconds. The blocks target turtle.backRightLeg, turtle.backLeftLeg, turtle.frontLeftLeg, turtle.frontRightLeg, turtle.tail, and turtle.head. A bottom toolbar shows various control blocks like "Do in order", "Do together", "If/Else", "Loop", "While", "For all in order", "For all together", "Wait", "print", and a comment block.

**turtle.hide** *No parameters* create new parameter

*No variables* create new variable

```
// the turtle goes into it's shell  
// all of the turtles limbs become invisible  
Do together  
  turtle.backRightLeg set isShowing to false duration = 0.1 seconds more...  
  turtle.backLeftLeg set isShowing to false duration = 0.1 seconds more...  
  turtle.frontLeftLeg set isShowing to false duration = 0.1 seconds more...  
  turtle.frontRightLeg set isShowing to false duration = 0.1 seconds more...  
  turtle.tail set isShowing to false duration = 0.1 seconds more...  
  turtle.head set isShowing to false duration = 0.1 seconds more...
```

Do in order Do together If/Else Loop While For all in order For all together Wait print //

## 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:

The image shows a Scratch code editor window with the following elements:

- Script Area:** The main area is yellow and titled "world.my first method". It contains the following code blocks:
  - `world.race`
  - `kangaroo.challenge` with a dropdown menu set to `obj = turtle`
  - `turtle.hide`
  - `kangaroo` with a dropdown menu set to `turn to face camera` and a `more...` dropdown
  - `kangaroo` with a dropdown menu set to `say I guess that's a no` and a `more...` dropdown
- Buttons:** On the right side of the script area, there are two buttons: "create new parameter" and "create new variable".
- Control Area:** At the bottom of the editor, there is a row of control blocks: "Do in order", "Do together", "If/Else", "Loop", "While", "For all in order", "For all together", "Wait", "print", and a play button.
- Background:** The background is a light blue color.

- 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