

Using Placeholders to Simplify your Methods: Learning Methods, Part 2

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** in the object tree, then click **create new method** in the methods tab, and name it **challenge**. Drag a **Do in order** into your new method.
- Then find the kangaroo's **turn to face** method and drag it into the method editor. Then select **turtle**, and **the entire turtle**.
- See the screenshot on the next slide for an illustration.

Play

Undo

Redo

world

camera

light

ground

road

+

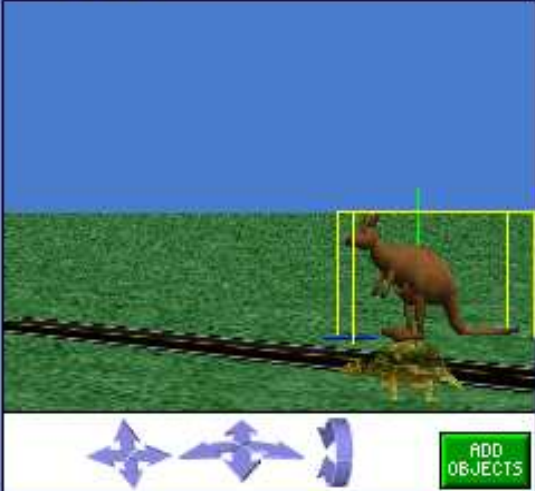
kangaroo

+

turtle

+

Dummy Objects



ADD OBJECTS

world.my first method

kangaroo.challenge

Events

create new event

When the world starts, do world.my first method

kangaroo's details

properties

methods

functions

kangaroo move to

kangaroo move toward

kangaroo move away from

kangaroo orient to

kangaroo turn to face

kangaroo point at

kangaroo set point of view to

kangaroo set pose

kangaroo stand up

kangaroo.challenge No parameters

No variables

Do in order

Do Nothing

target

the entire world

camera

light

ground

road

kangaroo

turtle

Dummy Objects

the entire turtle

backRightLeg

backLeftLeg

frontLeftLeg

frontRightLeg

tail

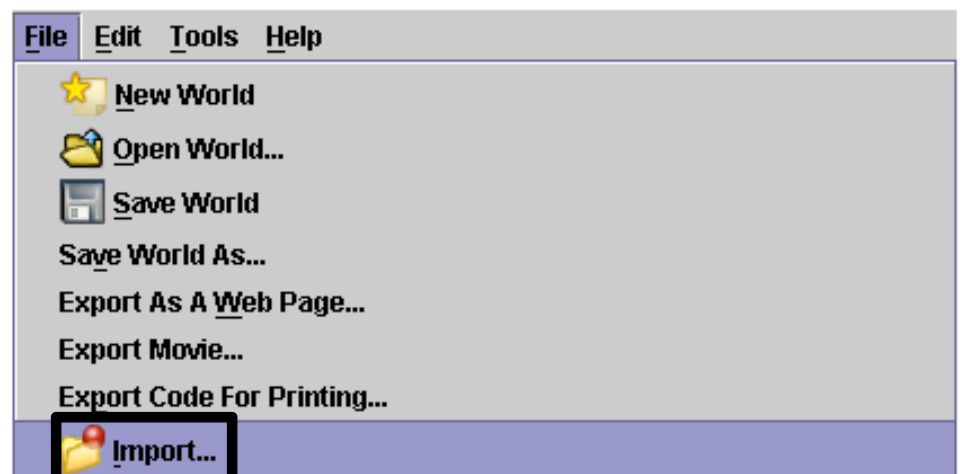
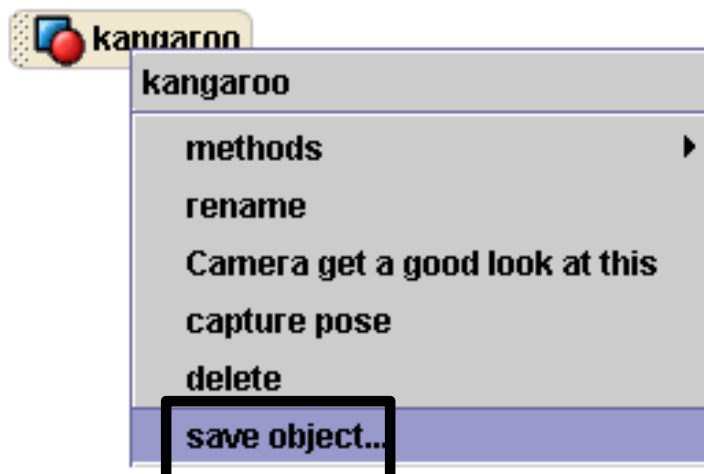
head

- Add a loop where the kangaroo executes **hop** two times. Then add code where he asks the turtle if he wants to race again. Your code will end up like this:

The image shows a Scratch script editor interface. At the top, there are several script blocks: 'kangaroo.hop', 'world.race', 'kangaroo.challenge' (selected), 'world.my first method', and 'turtle.walk'. Below these, the script for 'kangaroo.challenge' is displayed. It starts with a comment: '// the kangaroo hops over and asks a question'. This is followed by a 'Do in order' block containing three sub-blocks: 1. 'kangaroo' block with 'turn to face' and 'turtle' dropdowns, and a 'more...' dropdown. 2. A 'Loop' block set to '2 times' with a 'times' dropdown and a 'show complicated version' button. Inside the loop is a 'kangaroo.hop' block. 3. A 'kangaroo' block with 'say' and 'want to race again?' dropdowns, and a 'more...' dropdown. To the right of the script area are two buttons: 'create new parameter' and 'create new variable'. At the bottom, there is a palette of script 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.

Saving Alice Objects

- In Alice, you can save an individual object, along with any new methods you have written for it. This allows you to be able to use the same character in more than one world without having to teach it new methods over and over.
- To do this, you can right-click on any object in the object tree and then choose **save object**.
- To import the character into a new world, just go to **File**, then **Import**, and find the object where you saved it.



Why use parameters

- If we save the kangaroo object and then 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 turtle that isn't there in the new world! 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.

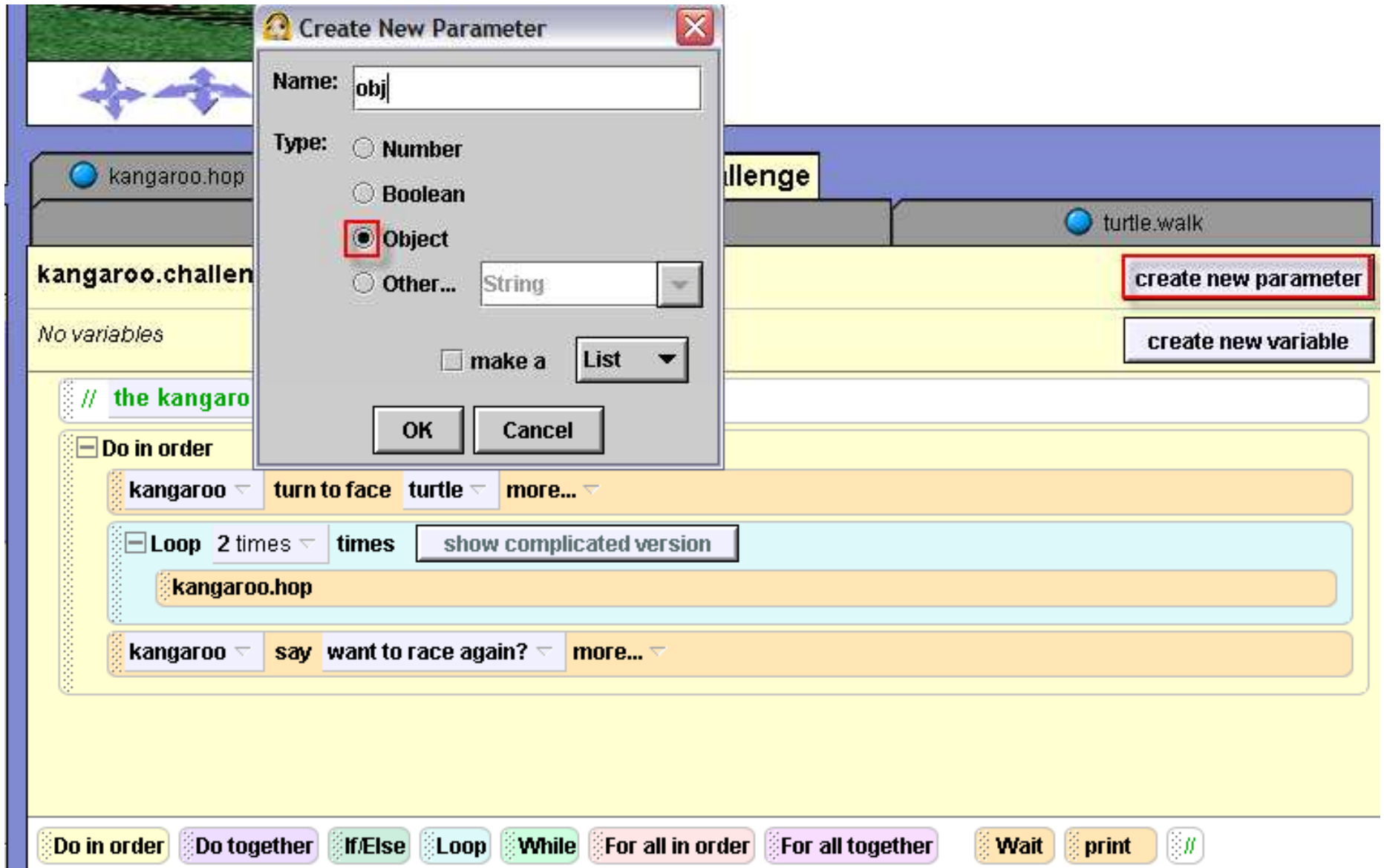
An example scenario

- 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**. With a parameter, we could do this very easily.

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 the **create new parameter** button in the method editor, and name it **obj**. Select the type **object**, and then click **OK**.
- 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 the word **turtle**.

The image shows a Scratch code editor interface. At the top, there are tabs for 'kangaroo.hop', 'world.race', and 'kangaroo.challenge' (which is selected). Below the tabs, there are buttons for 'world.my first method' and 'turtle.walk'. The main workspace is titled 'kangaroo.challenge' and contains a parameter '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, there is a comment '// the kangaroo hops to the obj and asks a question'. The code block is a 'Do in order' loop containing three blocks: 'kangaroo turn to face to obj more...', 'Loop 2 times times show complicated version kangaroo.hop', and 'kangaroo say want to race again? more...'. A red arrow points from the 'obj' parameter to the 'obj' in the 'turn to face' block. The bottom of the screen shows a palette with various 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.

How to call a method that has a parameter

- To test your code, drag `kangaroo.challenge` into `world.my first method` 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 `the entire turtle` as the object. 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 left sidebar and a main workspace.

Left Sidebar:

- kangaroo's details**
 - properties** (selected)
 - methods**
 - functions**
- hop** (block) with an **edit** button.
- challenge obj** (block) with an **edit** button. A red arrow points from this block to the workspace.
- create new method** button.
- kangaroo** (object) with methods: **move**, **turn**, **roll**, **resize**, **say**, **think**, **play sound**.

Main Workspace:

- world.my first method** (blue header bar).
- world.my first method** *No parameters* (yellow bar) with a **create new parameter** button.
- No variables** (yellow bar) with a **create new variable** button.
- world.race** (orange bar) containing a **challenge obj** (green-bordered block).

Bottom Bar:

- Control blocks: **Do in order**, **Do together**, **If/Else**, **Loop**, **While**, **For all in order**, **For all together**, **Wait**, **print**, and a comment block (//).

- Selecting the turtle as the parameter argument:

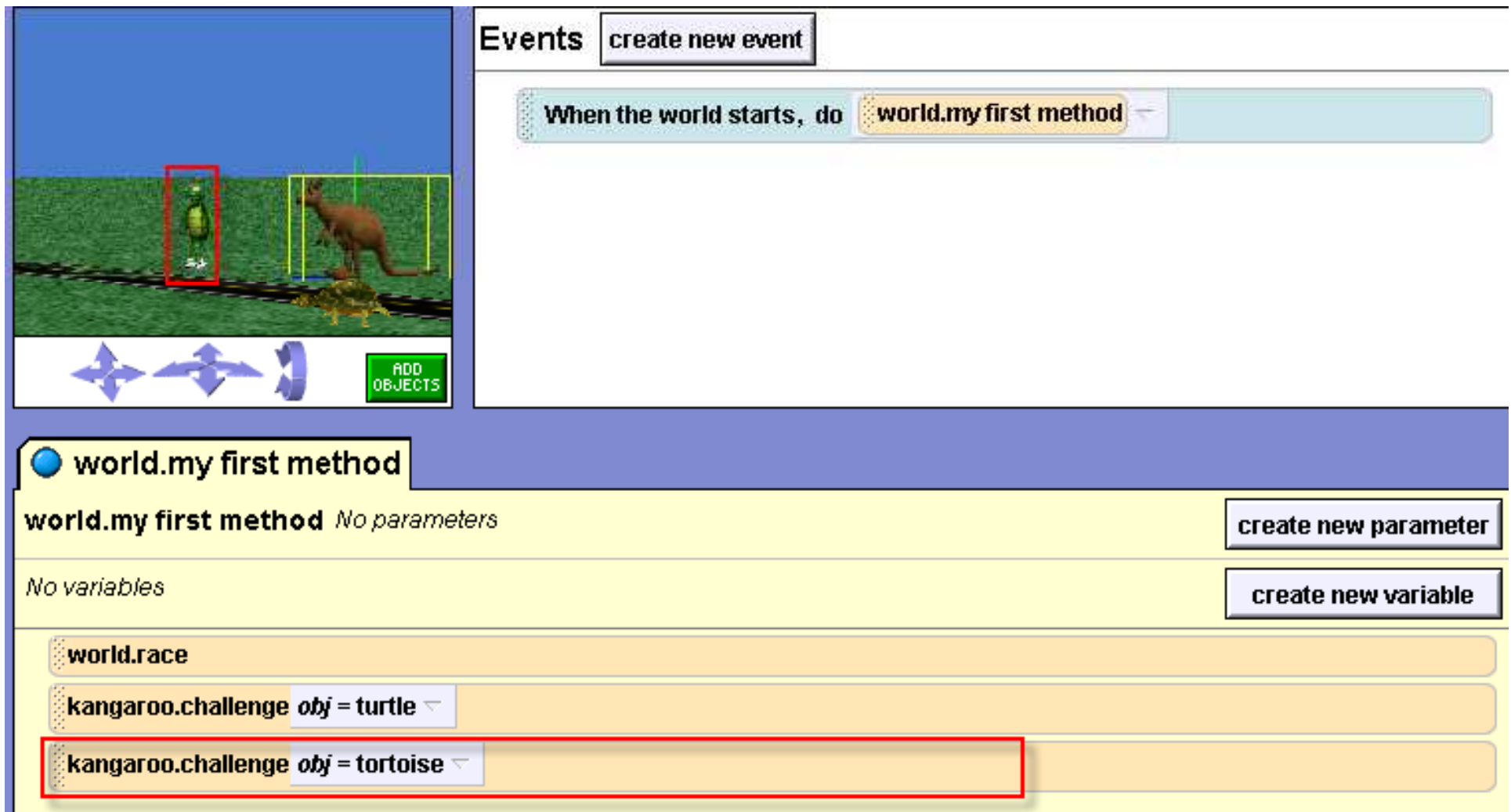
The screenshot shows the Scratch 'world.my first method' editor. The title bar is 'world.my first method'. Below it, the text 'world.my first method No parameters' is displayed, with a 'create new parameter' button to the right. Below that, the text 'No variables' is displayed, with a 'create new variable' button to the right. The main area contains a block labeled 'world.race'. Below this block, a menu is open showing the 'obj' column with options: camera, light, ground, road, kangaroo, turtle, and <None>. The 'turtle' option is selected. To the right of the 'obj' column, a list of attributes for 'the entire turtle' is shown: backRightLeg, backLeftLeg, frontLeftLeg, frontRightLeg, tail, and head. The 'head' attribute is selected. At the bottom of the editor, a row of control blocks is visible: 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and a green flag icon.

- 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.
- Drag **kangaroo.challenge** into your **world.my first method** 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 image shows a Scratch-like interface. On the left is a stage with a green field and a blue sky. A turtle is on the left, highlighted with a red box, and a kangaroo is on the right, highlighted with a yellow box. Below the stage are navigation arrows and an 'ADD OBJECTS' button. On the right is the 'Events' panel with a 'create new event' button. Below that is a script block: 'When the world starts, do world.my first method'. At the bottom is the 'Scripts' panel for 'world.my first method'. It shows 'No parameters' and 'No variables' buttons. Below these are three script blocks: 'world.race', 'kangaroo.challenge obj = turtle', and 'kangaroo.challenge obj = tortoise'. The last block is highlighted with a red box.

Events **create new event**

When the world starts, do world.my first method

world.my first method

world.my first method *No parameters* **create new parameter**

No variables **create new variable**

world.race

kangaroo.challenge *obj = turtle*

kangaroo.challenge *obj = 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.my first method**, 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 **methods** 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 + sign 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**. This will make the **backRightLeg** invisible. Click on **more...** on that line of code and set **duration** to **0.1**.
- See the screenshot on the next slide for an illustration.

Play Undo Redo

light
ground
kangaroo
turtle
backRightLeg
backLeftLeg
frontLeftLeg
frontRightLeg
tail
head

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

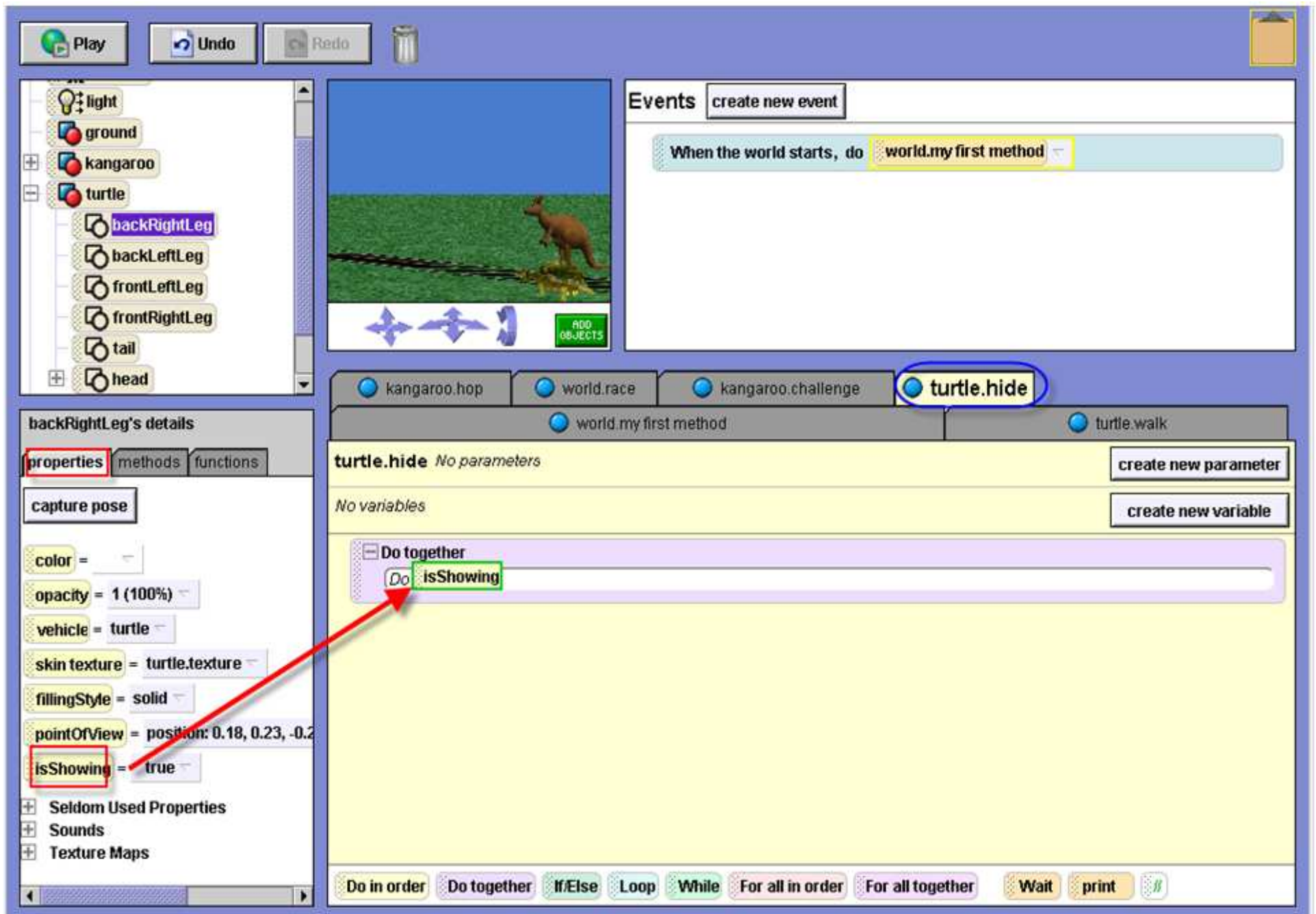
Events create new event
When the world starts, do world.my first method

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

turtle.hide No parameters
No variables
create new parameter
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



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 *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.my first method` underneath `kangaroo.challenge`.
- If you want, you can have the kangaroo say something at the end.

- Here is my final code in **world.my first method**:

The image shows a Scratch code editor window with several tabs at the top: kangaroo.hop, world.race, kangaroo.challenge, turtle.hide, and turtle.walk. The 'world.my first method' tab is selected and highlighted in yellow. Below the tab, the text 'world.my first method No parameters' is displayed, along with a 'create new parameter' button. Below that, the text 'No variables' is shown, with a 'create new variable' button. The main area of the script contains five code blocks: a 'world.race' block, a 'kangaroo.challenge' block with a dropdown menu set to 'obj = turtle', a 'turtle.hide' block, a 'kangaroo' block with a dropdown menu set to 'turn to face' and a 'camera' dropdown menu set to 'more...', and a 'kangaroo' block with a dropdown menu set to 'say' and a text field containing 'I guess that's a no' followed by a 'more...' dropdown menu. At the bottom of the editor, there is a palette of control blocks: 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and a green flag icon.

- Press play to watch your entire animation.

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.