

Teaching Characters to Walk: Learning Methods, Part 1



By Deborah Nelson

Duke University

Under the direction of
Professor Susan Rodger

June 9, 2008

Overview

- This tutorial will teach you how to make class-level and world-level methods.
- It will do this by showing you how to make a turtle walk, and a kangaroo hop, so that they are racing each other. You will also learn how to use loops.
- By learning this, you will be able to go on to create your own world and class-level methods.

Loading the World

- Open Alice. Then open the file `methodStart.a2w` from the Student Classwork drive (mcmillan/ALICE)
- Save it to your documents.
- **NOTE:** You cannot double-click the file to open.
- Windows will not know what to use, and even if you select Alice from a list of programs, the loading will fail.

Loading (cont 1)

- First: After you have opened the file, and set up your world, go into the "Layout" mode by clicking on the green button **Add Objects** (toward the middle of screen).
- Click **more controls**. Click **drop a dummy at the camera**. Rename the dummy **originalPosition**. To leave the layout mode, click **done**.
- Look at the screenshot on the next page for an illustration.

Play Undo Redo

- world
 - camera
 - light
 - ground
 - kangaroo
 - turtle
 - road
 - Dummy Objects
 - Dummy



Dummy

- methods
- rename
- Camera get a good look at this
- capture pose
- delete
- save object...

single view quad view

Move Objects Freely

affect subparts

aspect ratio: 4/3

lens angle:

drop dummy at camera

drop dummy at selected object

move camera to dummy: <None>

fewer controls <<



Home > Local Gallery > City

Search Gallery

<p>Class ParkingMeter</p> <p>your computer</p>	<p>Class Road</p> <p>on your computer</p>	<p>Class Skyscraper1</p> <p>on your computer</p>	<p>Class Skyscraper2</p> <p>on your computer</p>	<p>Class Stadium</p> <p>on your computer</p>	<p>Class Stadium</p> <p>on your computer</p>
--	---	--	--	--	--

Why drop a dummy?

- This is something you should always do when you make a world in case you need to return the camera to this view later.
- If you don't understand dummies, look at the camera control tutorial again.

Part 1: Methods

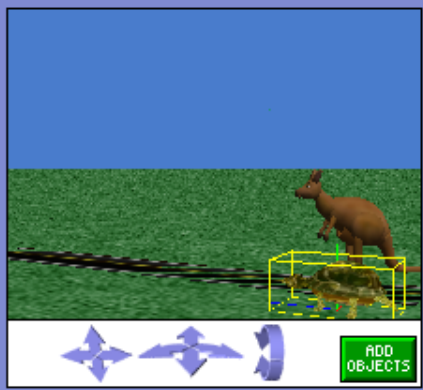
- A method is a sequence of instructions or behaviors that will be carried out when requested.
- Remember, built in methods are basic instructions every character already knows how to perform.
- You can use them to create new methods so that the characters can do more.
- The two types of methods are class-level and world-level.

Part 2: Class-level methods

- A class-level method defines the behavior for a single character.
- **Step 1: *How to create a class-level method***
 - 1) In our example world, click on the **turtle** in the object tree (upper left panel).
 - 2) In turtle's details (lower left panel) click the **methods** tab and then the button **create new method**. Name it **walk**.
- See the screenshot on the next slide for an illustration.

Play
 Undo
 Redo

world
 camera
 light
 ground
 kangaroo
turtle
 road
 Dummy Objects



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

turtle's details
 properties **methods** functions
create new method
 turtle move
 turtle turn
 turtle roll
 turtle resize
 turtle say
 turtle think
 turtle play sound
 turtle move to
 turtle move toward

New Method
 Name:
 OK Cancel

world.my first meth
 world.my first method No
 No variables
 (Do Nothing)
 create new parameter
 create new variable

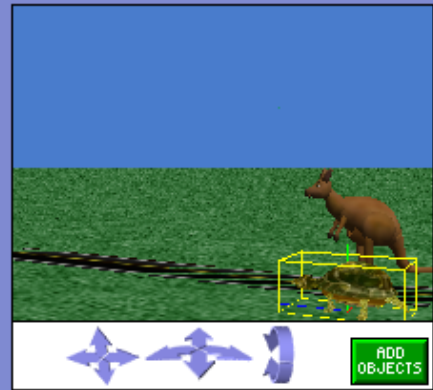
Do in order Do together IfElse Loop While For all in order For all together Wait print

Step 2: *How to write a method*

- We want to move the turtle's legs back and forth as the turtle moves forward.
- First, drag the control statement **Do in order**, from the bottom of the window, into the editor.
- This is the default setting for Alice, meaning that the instructions will be carried out in order, one after the other.
- See the screenshot on the next slide for an illustration.

Play
 Undo
 Redo

- world
- camera
- light
- ground
- kangaroo
- turtle
- road
- Dummy Objects



Events create new event

When the world starts, do world.my first method

turtle's details

properties | **methods** | functions

walk edit

create new method

- turtle move
- turtle turn
- turtle roll
- turtle resize
- turtle say
- turtle think
- turtle play sound
- turtle move to

world.my first method turtle.walk

turtle.walk *No parameters* create new parameter

No variables create new variable

(Do Nothing)

Do in order

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

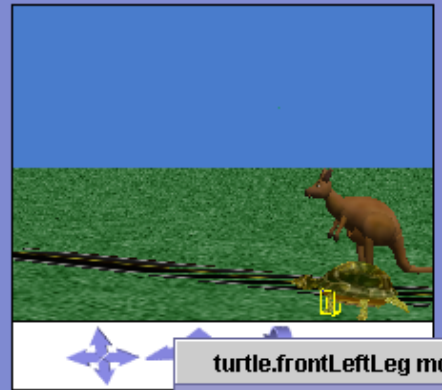


Writing a method (cont 1)

- Next, in the object tree, click on the **+** sign beside **turtle** to see the different body parts.
- Drag the **frontLeftLeg** into the method. Select **turn**, and then **backward**. Choose **other** and type in **0.1**.
- See the screenshot on the next slide for an illustration.



- light
- ground
- kangaroo
- turtle
 - backRightLeg
 - backLeftLeg
 - frontLeftLeg
 - frontRightLeg
 - tail
 - head



Events create new event

When the world starts, do world.my first method

- frontLeftLeg's details
- properties | **methods** | functions
- frontLeftLeg move
 - frontLeftLeg turn
 - frontLeftLeg roll
 - frontLeftLeg resize
 - frontLeftLeg say
 - frontLeftLeg think
 - frontLeftLeg play sound
 - frontLeftLeg move to
 - frontLeftLeg move toward
 - frontLeftLeg move away from

turtle.walk

No variables

Do in order

Do No

Do in order

- turtle.frontLeftLeg move
- turtle.frontLeftLeg turn
- turtle.frontLeftLeg roll
- turtle.frontLeftLeg resize
- turtle.frontLeftLeg say
- turtle.frontLeftLeg think
- turtle.frontLeftLeg play sound
- turtle.frontLeftLeg move to
- turtle.frontLeftLeg move toward
- turtle.frontLeftLeg move away from
- turtle.frontLeftLeg orient to
- turtle.frontLeftLeg turn to face
- turtle.frontLeftLeg point at
- turtle.frontLeftLeg set point of view to
- turtle.frontLeftLeg set pose
- turtle.frontLeftLeg stand up
- turtle.frontLeftLeg set color to
- turtle.frontLeftLeg set opacity to
- turtle.frontLeftLeg set vehicle to
- turtle.frontLeftLeg set skin texture to
- turtle.frontLeftLeg set fillingStyle to

direction

- left
- right
- forward
- backward

amount

- 1/4 revolution
- 1/2 revolution
- 1 revolution (all the way around)
- 2 revolutions
- other...

create new parameter

create new variable

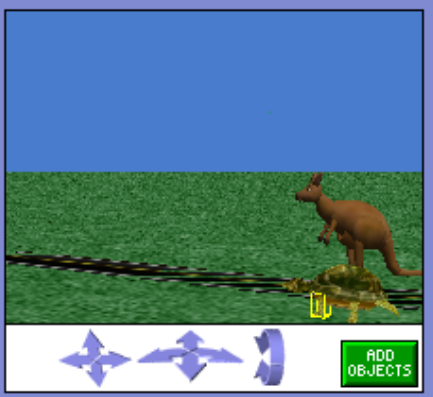
all in order | For all together | Wait | print

Writing a method (cont 2)

- Finally, click **more** at the end of the turn command and choose **duration**, then **.25 seconds**.
- The default time for an action to be performed is 1 second. We are changing it to 0.25 so that it will happen faster.
- Next, we want the turtle to move forward at the same time that the back leg goes forward.
- So drag in the *control statement* **Do together**
- See the screenshot on the next slide for an illustration.

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

frontLeftLeg's details

properties | **methods** | functions

- frontLeftLeg move
- frontLeftLeg turn
- frontLeftLeg roll
- frontLeftLeg resize
- frontLeftLeg say
- frontLeftLeg think
- frontLeftLeg play sound
- frontLeftLeg move to
- frontLeftLeg move toward
- frontLeftLeg move away from

world.my first method
turtle.walk
create new parameter

turtle.walk *No parameters*

No variables create new variable

Do in order

- turtle.frontLeftLeg turn backward 0.1 revolutions duration = 0.25 seconds more...
- Do together
 - Do Nothing

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

Writing a method (cont 3)

- Finish dragging and dropping the instructions until your method looks like this:

The screenshot shows a programming environment with two tabs: 'world.my first method' and 'turtle.walk'. The 'turtle.walk' tab is active and shows a method definition with 'No parameters' and 'No variables'. Below this, there is a 'Do in order' block containing four instructions:

- `turtle.frontLeftLeg` turn backward 0.1 revolutions duration = 0.25 seconds
- Do together** block containing:
 - `turtle` move forward 0.3 meters duration = 0.25 seconds
 - `turtle.backLeftLeg` turn forward 0.1 revolutions duration = 0.25 seconds
- `turtle.frontLeftLeg` turn forward 0.1 revolutions duration = 0.25 seconds
- `turtle.backLeftLeg` turn backward 0.1 revolutions duration = 0.25 seconds

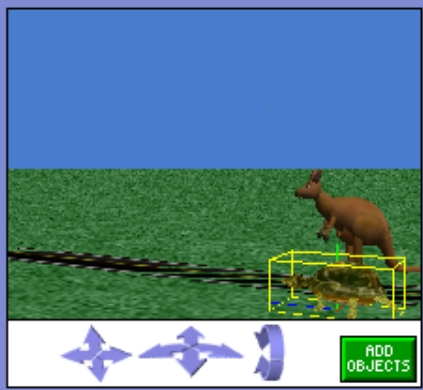
At the bottom of the interface, there is a toolbar with various control blocks: 'Do in order', 'Do together', 'If/Else', 'Loop', 'While', 'For all in order', 'For all together', 'Wait', 'print', and a comment block.

Writing a method (cont 4)

- Finally, add a comment to your method to tell someone reading your code what it does.
- Your comment can say: "Move the turtle's legs back and forth".
- This comment is not Alice code.
- It is simply an explanation for someone trying to understand your code. Alice ignores the comments when it plays your world.
- See the screenshot on the next slide for an illustration.

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

turtle's details

properties **methods** functions

walk edit

create new method

- turtle move
- turtle turn
- turtle roll
- turtle resize
- turtle say
- turtle think
- turtle play sound
- turtle move to

world.my first method **turtle.walk**

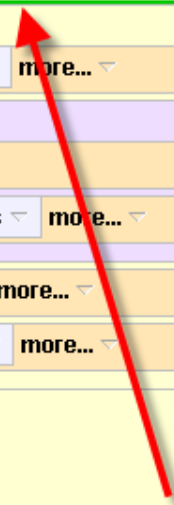
turtle.walk No parameters create new parameter

No variables create new variable

Do in order

- turtle.frontLeftLeg turn backward 0.1 revolutions duration = 0.25 seconds more...
- Do together
 - turtle move forward 0.3 meters duration = 0.25 seconds more...
 - turtle.backLeftLeg turn forward 0.1 revolutions duration = 0.25 seconds more...
 - turtle.frontLeftLeg turn forward 0.1 revolutions duration = 0.25 seconds more...
 - turtle.backLeftLeg turn backward 0.1 revolutions duration = 0.25 seconds more...

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

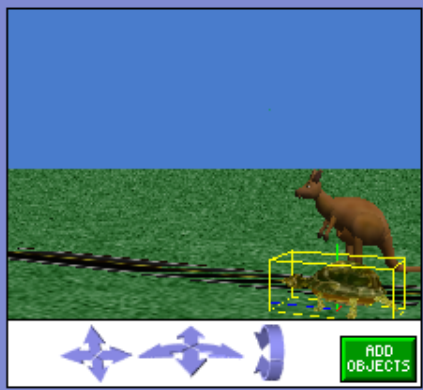


Step 3: *To call your method*

- Click on `world.myfirstmethod` to get back to it and then drag `turtle.walk` into it.
- Push the `play` button to watch the turtle walk.
- See the screenshot on the next slide for an illustration.

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

turtle's details

properties **methods** functions

walk edit

create new method

- turtle move
- turtle turn
- turtle roll
- turtle resize
- turtle say
- turtle think
- turtle play sound
- turtle move to

world.my first method turtle.walk

world.my first method No parameters create new parameter

No variables create new variable

Do walk

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

Step 4: *Writing a method for Kangaroo*

- Now we will write a class-level method for the kangaroo to make it hop.
- In the object tree, click on **kangaroo**. In kangaroo's details, click the button **create new method** and name it **hop**.
- To write the method, drag and drop the instructions listed on the following slides.

Kangaroo.hop method (cont 1)

- Your first step is to drag a **Do together** into the method.
- To find the **lowerleg** of the kangaroo, you click the **+** sign beside the **leg** tab. When you finish, your method should look like the screenshot on the next slide.

Part One of code for kangaroo.hop

The image shows the Scratch code editor interface for a script named "kangaroo.hop". The script is currently empty, with the text "No parameters" and "No variables" displayed. The main workspace contains a "Do together" block, which is expanded to show a "Do in order" block. This "Do in order" block contains two "Do together" blocks. Each "Do together" block contains two "move" blocks for the "kangaroo" object. The first "Do together" block contains "move up 0.5 meters" and "move forward 0.5 meters", both with a duration of 0.25 seconds. The second "Do together" block contains "move down 0.5 meters" and "move forward 0.5 meters", both with a duration of 0.25 seconds. Below the "Do in order" block, the beginning of another "Do in order" block is visible, starting with "kangaroo right 12 revolutions" and "turn forward 0.12 revolutions", both with a duration of 0.25 seconds. The bottom of the interface shows a palette of Scratch blocks, including "Do in order", "Do together", "If/Else", "Loop", "While", "For all in order", "For all together", "Wait", "print", and "comment".

world.my first method turtle.walk **kangaroo.hop**

kangaroo.hop *No parameters* create new parameter

No variables create new variable

Do together

Do in order

Do together

kangaroo ▾ move up ▾ 0.5 meters ▾ *duration = 0.25* seconds ▾ more... ▾

kangaroo ▾ move forward ▾ 0.5 meters ▾ *duration = 0.25* seconds ▾ more... ▾

Do together

kangaroo ▾ move down ▾ 0.5 meters ▾ *duration = 0.25* seconds ▾ more... ▾

kangaroo ▾ move forward ▾ 0.5 meters ▾ *duration = 0.25* seconds ▾ more... ▾

Do in order

kangaroo right 12 revolutions ▾ turn forward ▾ 0.12 revolutions ▾ *duration = 0.25* seconds ▾ more... ▾

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

Part two of code for kangaroo.hop

- Do in order

kangaroo.rightLeg.lowerLeg ▾ turn forward ▾ 0.12 revolutions ▾ *duration* = 0.25 seconds ▾ more... ▾

kangaroo.rightLeg.lowerLeg ▾ turn backward ▾ 0.12 revolutions ▾ *duration* = 0.25 seconds ▾ more... ▾

- Do in order

kangaroo.leftLeg.lowerLeg ▾ turn forward ▾ 0.12 revolutions ▾ *duration* = 0.25 seconds ▾ more... ▾

kangaroo.leftLeg.lowerLeg ▾ turn backward ▾ 0.12 revolutions ▾ *duration* = 0.25 seconds ▾ more... ▾

Kangaroo.hop method (cont 2)

- Remember to call your method to test it.
- Click on `world.myfirstmethod` to get back to it and then delete what you have there.
- Drag `kangaroo.hop` into it.
- Push the `play` button to watch the kangaroo hop.

Part 3: World-level methods

- A world-level method has characters that interact with each other.
- If you need to write a method that references more than one object, use a world-level method.

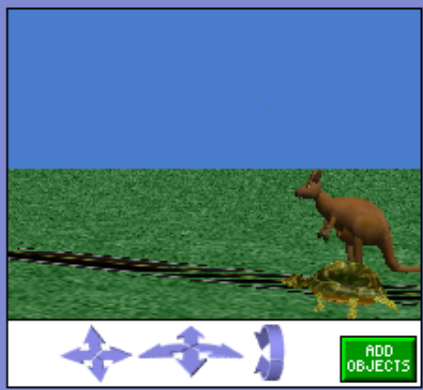
Step 5: *Writing a world-level method*

- In our example, we are going to write a method so that the turtle and the kangaroo race each other.
- Click on **world** in the object tree. Click on the **methods** tab and the button **create new method**. Name it **race**.

Play
 Undo
 Redo

world

- camera
- light
- ground
- +
- kangaroo
- +
- turtle
- +
- road
- +
- Dummy Objects



Events

When the world starts, do **world.my first method**

world's details

properties **methods** functions

my first method

world.my first meth

world.my first method No

No variables

turtle.walk

New Method

Name:

Writing a world-level method (cont 1)

- First, we want the turtle to have a conversation with the kangaroo.
- The first step is to drag a **Do in order** into the method.
- Then, click on **turtle** in the object tree and drag it into the method.
- Put the following code on the next slide into this method:

The code for world.race

world.my first method

turtle.walk

kangaroo.hop

world.race

world.race *No parameters*

create new parameter

No variables

create new variable

// the turtle and the kangaroo race each other

Do in order

turtle turn right 0.25 revolutions more...

turtle say I'll race you more...

kangaroo say ok. Let's go more...

turtle turn left 0.25 revolutions more...

Do in order

Do together

If/Else

Loop

While

For all in order

For all together

Wait

print

//

Writing a world-level method (cont 2)

- Since we want the turtle and the kangaroo to move together, drag the control statement **Do together** into the race method.
- Click on **turtle** in the object area and click on the **methods** tab.

- Then drag **turtle.walk** into the **Do together**.

The screenshot displays the Alice software interface. On the left, the 'turtle's details' panel is open, showing the 'methods' tab. The 'walk' method is highlighted with a red box, and a red arrow points from it to the 'Do together' block in the script editor. The script editor shows a 'world.race' script with the following steps:

- turtle turn right 0.25 revolutions more...
- turtle say I'll race you more...
- kangaroo say ok. Let's go more...
- turtle turn left 0.25 revolutions more...
- Do together** block containing the **walk** method.

The bottom of the screen shows the Windows taskbar with the following open applications: start, tutorial1 - Notepad, method's Code - Mozil..., iTunes, Alice (2.0 04/05/2005...), and the system tray with the time 11:21 AM.

Writing a world level method (cont 3)

- Click on **kangaroo** in the object tree.
- Drag **kangaroo.hop** into the **Do together** under **turtle.walk**.
- Click on **world.myFirstMethod**. Delete what you have there.

Play your world

- Click on `world` in the object area and drag `world.race` into `myFirstMethod`. Push `play`.
- The characters will only move once, which doesn't look like much of a race.
- If we repeatedly call the `hop` and `walk` methods in `world.race`, it will look more like a race.
- Instead of dragging the instructions over multiple times, let's use a loop.

Step 6: *Loops*

- One of the control statements is **Loop**.
- A Loop will call the method that you drop in it as many times as you ask it to.
- Drag **Loop** from the bottom of the window into the **world.race** method right above the **Do together**. Select **other** and type in **4**. This means the loop will call whatever you put in it 4 times.
- See the screenshot on the next slide for an illustration

Dragging the loop into your code

The screenshot shows a visual programming interface with several tabs at the top: "world.my first method", "turtle.walk", "kangaroo.hop", and "world.race". The "world.race" tab is active, showing "No parameters" and "No variables" with buttons for "create new parameter" and "create new variable".

The main workspace contains a "Do in order" block with the following steps:

- turtle turn right 0.25 revolutions more...
- turtle say I'll race you more...
- kangaroo say ok. Let's go more...
- turtle turn left 0.25 revolutions more...
- Do together block containing:
 - turtle.walk
 - kangaroo.hop

A red arrow points from the "Loop" menu to the "0.25 revolutions" field in the fourth step of the "Do in order" block. The "Loop" menu is open, showing options: "end", "1 time", "2 times", "5 times", "10 times", "infinity times", "0.12 times", "0.3 times", "0.1 times", and "other...".

At the bottom, there is a toolbar with buttons for "Do in order", "Do together", "If/Else", "Loop", "While", "all together", "Wait", "print", and a green flag icon.

The system tray at the bottom shows "method's Code - Mozil...", "iTunes", "Alice (2.0 04/05/2005...", and the time "11:24 AM".

Making the loop

- Drag the **Do together** statement (with **kangaroo.hop** and **turtle.walk**) into the Loop statement.
- Your code should look like this now (see next slide).

The completed code for world.race

The image shows a Scratch code editor window with several tabs: kangaroo.hop, world.race (selected), kangaroo.challenge, turtle.hide, world.my first method, and turtle.walk. The code for 'world.race' is as follows:

```
// the turtle and the kangaroo race each other
```

- Do in order
 - turtle turn right 0.25 revolutions more...
 - turtle say I'll race you more...
 - kangaroo say ok. Let's go more...
 - turtle turn left 0.25 revolutions more...
 - Loop 4 times times show complicated version
 - Do together
 - turtle.walk
 - kangaroo.hop

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 a comment block (//).

Play your world

- When you push **play**, the kangaroo will win the race.
- Congratulations on creating one of your first methods. Save this world. We will add to it again in a later tutorial.

Recap

- This tutorial introduced class-level and world-level methods.
- If there is only one character in a method, it should be a class-level method.
- If there is more than one character involved, write a world-level method.
- To repeat an action, use a loop.