Shorter of two objects and changing color

Functions, events and setting the color in sequence and randomly

Susan Rodger, Duke University
July 2009
What this tutorial shows

• You’ve used built-in functions. Now you will write your own functions.

• NOTES: Not all objects can change color. The chicken and snowman can change color, which is why we use them. The bunny cannot change color, the object was just designed that way.
Parts

• Part 1: Create a “world” function that returns the shorter of two objects – make it a world function because it uses two different objects

• Part 2: Create a chicken function that returns a color (cycling through 4 colors) and use the function to change the color of the chicken

• Part 3: Create a snowman function to returns a random color and use the function to change the color of the snowman randomly

• Part 4: Create a snowman function to return the color as a word and use the function to have the snowman say what color it is.
Part 1: Start Alice world with sand and add in chicken, snowman, bunny, and joey
Create World function “ObjectThatIsShorter” – select Type as Object
The new function appears – notice the “Obj” – means this function returns an object.
What this function will do

- A function computes an answer and returns the answer.
- The function ObjectThatIsShorter will compare the chicken and the snowman in height and the answer is the one that is shorter. The function returns the shorter object.
- Since the function type is “object” the function must return an “object”
Drag up if/else
Compare chicken and snowman’s heights – from world functions, drag over the “a<b” into the “true” and enter 1’s
Now click on Chicken, then functions, and drag “Chicken’s height” over twice twice.

Click on the second “Chicken” and change to “snowman”
Compute the answer

• If the chicken’s height is less than the snowman’s height, then we know the answer is the Chicken. The “answer” is put in the line after the “if” and before the “else” (see the next slide)

• To return the answer, drag up “Return” from the bottom of the window (see the next slide)
Drag up “return” and select Chicken
If Chicken is taller than snowman

• Then the answer (or shortest object) is snowman

• After the “else” return snowman
• Note that the answer is either the “chicken” or the “snowman”, but not both. Only one of the two is returned.
Now try out the function, use it where there is an object. In `myFirstMethod`, put this code...
Replace “chicken” with new function – in World Functions, drag over the function “objectThatIsShorter” (that returns an object) over Chicken.

`objectThatIsShorter` can be used in place of any object.
Click Play

- Only the Chicken and snowman’s height are compared and the one that is shorter (the chicken) says “I’m shorter!”

- Does your function really work? Resize the chicken so it is larger than the snowman and then play again. This time the snowman should say “I’m shorter!”
Adding Flexibility

• We wrote the function to compare the chicken and snowman’s heights.

• For Example, notice that with the “move” command, it is flexible in that you get to pick the direction and the distance to move.

• To make this function more flexible, we will add two parameters (choices, so you can pick the two objects to compare).
Click on “create new parameter”
Enter the name “object1” and select type “Object”
Add another parameter named “object2” of type “Object”
Click and drag “object1” over both Chickens

<table>
<thead>
<tr>
<th>world.objectThatIsShorter</th>
<th>Obj object1, Obj object2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Chicken</td>
<td></td>
</tr>
<tr>
<td>Else</td>
<td>Return snowman</td>
</tr>
<tr>
<td>Return &lt;None&gt;</td>
<td></td>
</tr>
</tbody>
</table>
Click and drag object2 on top of both of the snowman’s
Now let’s Test the method

• Back in myfirstmethod, note that now you have to choose two objects to compare in the function objectThatIsShorter.

• Select two animals and Play.

• Then select two different animals and play again
Resize objects to make some taller, and click Play again
Part 2: Change the color of the chicken from “no color” to “blue” to “red” to “green” and cycle through again
Create Chicken Function “changeColor”
Click on Chicken, then functions, then “create new function”, type “changeColor” as name, and type “Color”
The new function appears – notice the color wheel – this function returns a new color
Returning new Chicken color - Idea

• If the chicken color is “no color” then we want to return the new color “blue”
• Else, if the chicken color is “blue”, then we want to return the new color “red”
• Else, if the chicken color is “red”, then we want to return the new color “green”
• Else if the chicken color is “green”, then we want to return the new color “no color”
First Drag up the if/else. Then select “Chicken”, “properties”. Then drag over the color property into “true” and select “Chicken.color == no color”
The “no color” when dropped looks like white. Now drag up “return”, drop in after the if, and select “Blue”
Under the “else” part, drag in another if/else, and repeat the second if (if the color is blue, return red)
Resulting in:
Now continue with two more colors

• If the chicken color is red, return green
• Else, return “no color”
Here is the final code... with three nested if’s
Now, let’s use the new function Chicken.changeColor. When we click on a Chicken we will change its color to the new color returned from this function. First create an event.
First, select Chicken to click on, then drag over color property and under expressions, pick the color returned by the function chicken.changeColor
Here is the final event

<table>
<thead>
<tr>
<th>Events</th>
<th>create new event</th>
</tr>
</thead>
<tbody>
<tr>
<td>When the world starts, do</td>
<td>world.my first method</td>
</tr>
<tr>
<td>When is clicked on</td>
<td>Chicken</td>
</tr>
<tr>
<td>do</td>
<td>Chicken set color to</td>
</tr>
<tr>
<td>more...</td>
<td></td>
</tr>
</tbody>
</table>
Now click Play and then click on the chicken several times.
Part 3

• Now we will write a function to return a random color and use that color to change the color of the snowman.
Click on Snowman, functions, then create new function, with name `changeColorRandom` and type color.
How to change color randomly

• Generate a random number in Alice between 0 and 1
• If the number is between 0 and .25, return color “red”
• Else, if the number is between .25 and .50, return “blue”
• Else if the number is between .50 and .75, return “green”
• Else if the number is between .75 and 1, return “no color”
First, create a “local variable” called colorNumber to store the random number, this is like a “property” for the function. Make sure to select Type: number.
1. The local variable appears
2. Drag it down to the code and set its value to 1.
Select “world”, then “functions” to find “random number”, and drag it over the “1”
Click on “more” to add minimum set to 0 and maximum set to 1.
colorNumber stores the random number generated so we can refer to it.
Drag up an if/else
Check colorNumber for red case:
If colorNumber > 0 and colorNumber <= 0.25 from “world”, “functions”, drag over a>b, then fill in with “colorNumber > 0”
Click on the last white down arrow, select “logic”, ... “and”, and “true”
Drag in the “a<=b” and then fill in values resulting in:
colorNumber > 0 and colorNumber<= 0.25
Drag up the return, and select the color red
Drag another if/else into the else, and drag in code for the “blue” case.
Add “green” and “no color” cases
Create and event for clicking on snowman, then drag snowman’s color property over.
Click “Play” and click on the snowman and he randomly selects one of the four colors.
How do you know if it is working correctly? Drag up print to see the colorNumber value. Select Object, camera
Drag colorNumber over “camera”.

```
123 colorNumber  =  1

colorNumber  set value to  randomNumber

print  colorNumber
```
Click on down white arrow and type in text: “Value of colorNumber is “
Now play, click on snowman and you can see the value of colorNumbers to see if the numbers match the color ranges

Value of colorNumber is 0.46906740019566306
Value of colorNumber is 0.5947262670111089
Value of colorNumber is 0.2635782597487416
Part 4

• Write a function to return the current color of the snowman, as a word. We will use the word to have the snowman say what color he is.
Create getNameOfColor

- Click on “snowman”, then “functions” tab, then “create new function”. Enter the name “getNameOfColor” and select type “String”
Resulting function ... Note return type is “ABC” for a word (or String)
Drag up an if-else

- Then click on “snowman”, then properties tab. Drag over the “color” into the “true” and select “snowman’s color == red”
Result is:

```plaintext
snowman.getNameOfColor

snowman.getNameOfColor  No parameters

No variables

If snowman\.color ==

Do Nothing

Else

Do Nothing

Return default string
```
Drag up “Return” tab from the bottom and enter the value “red”
Add additional if’s for the other three color cases – here is the Blue case
Here is the code after adding the green and white (default) cases

```plaintext
If snowman.color == red
    Return red

Else
    If snowman.color == blue
        Return blue
    Else
        If snowman.color == green
            Return green
        Else
            Return white
```

Now let’s use the new function

• Let’s change the event

• To this:

• By copying the “snowman set color” line to the clipboard, then drop in a “do in order”, paste the “snowman set color back in”
Now drag in “snowman say”

- And have the snowman say “I’m the color “ and make sure you put a blank after the word color.
Join in the name of the color

• Click on “world” “functions” and drag over the “a joined with b”, and drop that on what the snowman is saying, select “expressions” and the “getNameOfColor” function
Resulting code. Now Play and click on snowman, he tells you his color!

I'm the color red

I'm the color green