Line up

Let's show how factorials relate to permutations by seeing how many ways we can arrange you all in a line.

By Melissa Dalis
Professor Susan Rodger
Duke University
June 2011
Overview

• This tutorial will teach you how to build a demonstration that shows all the possible permutations of 2, 3, and 4 students standing in a line, and relates this to factorials.

• Programming concepts covered include loops, billboards, events, and all of the more basic features of Alice required to understand these concepts.

• We will use an arrayVisualization to show the students standing in a line.
Adding objects

• Click Add Objects.
• Find the People section in the Local Gallery and click on the folder called Walking People.
• Add any 4 kids that you want, and also a teacher. I chose Alice, Student1, Girl, and Sparky for the kids.
• Also add some Nature to your scene from the Local Gallery. I added 4 Bonzai trees.
Adding visualizations

- Find the Visualizations folder in the Local Gallery, and click on ArrayVisualization.
- Select a kid for each of the four items. I chose aliceLiddell, student1, girl, and sparky for items 1, 2, 3, and 4.
- Add an ObjectVisualization.
Rearranging the world

• Spread out and resize the bonzai trees, or whatever other nature you added, in the background so that the scene looks more realistic.

• Move the teacher to the left and face her forward more.

• Face the ArrayVisualization with the 4 kids on it forward, and move the objectVisualization behind it.
Rolling the arms down

• Click on the + sign next to teacher in the object tree.
• Click on the + sign next to upperBody.
• Right click on leftArm, and select methods >> roll >> left >> ¼ revolution.
• Right click on rightArm, and select methods >> roll >> right >> ¼ revolution.
• Do the same for all the other people with their arms awkwardly raised.
Counting the permutations

• Find Create 3D Text in the Local Gallery and type # Permutations: 0 to change the text.
• Move the 3D text object to the top right corner.
• Rename 3D Text in the object tree to permutationsText.
• Go to permutationsText’s methods tab.
• Click create new variable, and name it counter, select Number for the type, and then select other and then 0 for the value.
Adding billboards

- If you haven’t already, download the two images from the website entitled `permutationBillboard` and `factorialBillboard`, or you can make your own.
- Click **File** at the top of the screen, and select **Make Billboard**, and click on `permutationBillboard`.
  - Do the same for `factorialBillboard`.
- Face both of them forward, make them bigger, move them to the center, and move them to the front of everything.
- It doesn’t matter which is in front.
Hiding the billboards

• The billboards should be invisible, unless the player wants to see them.
• Click on `permutationBillboard` in the object tree and find its properties tab.
• Next to `isShowing`, change `true` to `false`.
• Do the same for `factorialBillboard`.
Creating events

• Click at the right to stop adding objects.
• In the Events section in the top right corner, click on create new event.
• When the user types P, we want them to see the permutationBillboard.
• Select When a key is typed.
• Change any key to letters >> P.
• Drag a Do in order from the bottom into the Nothing.
• Click on permutationBillboard in the object tree and go to its properties tab.
• Drag isShowing into the Do in order that you just added, and select true.
• Right click on the isShowing line, select make copy, and change true to false.
• In between the isShowing lines, drag in a Wait, select other, and type in 6.
• Drag the entire **When P is typed** block onto the **clipboard** to make a copy, and then drag the **clipboard** below the block to paste it.

• To change the **When P is typed** to say **When F is typed**, click on the **P** and select **letters**, and then **F**.

• In both of the **isShowing** lines, change **permutationBillboard** to **factorialBillboard**.
• Click on **world** in the object tree and find its **properties** tab.

• Click **create new variable**, name it **startGame**, select **Boolean** for the type, and select **false** for the value.

• Back in the **Events** section, click **create new event**, and select **When a key is typed**.

• Click on **any key** and select **letters**, and then **S**.

• Drag **startGame** into the **Nothing** and select **set value**, and then **true**.
Writing world.my first method

• Find teacher in the object tree, and go to her methods tab.

• Drag teacher say into the Do Nothing in the method, select other, and type: Hi! Today we are going to learn about permutations, or a particular ordering of objects.

• Drag in another teacher say into the Do Nothing, select other, and type: Type P to learn more about permutations, F to learn about factorials, and S when you’re ready to start the game.
• We want the program to wait until the player has pressed S to start the game.

• Drag a **While** into the *Do Nothing* and select true.

• Click on **world** in the **object tree** and find its properties tab.

• Drag **startGame** into the **true**.

• Click on **startGame** and select **logic**, and then **not startGame**.
• Drag a teacher say into the bottom of the method and type: Let’s show how factorials relate to permutations by seeing how many ways we can arrange you all in a line.

• Drag in another teacher say and type: The number of ways that n people can be arranged is n!, or “n factorial”.

• At the end of each teacher say line, click more, then duration, then other, then 5 seconds.
Creating a **permute** method

- Find **world** in the **object tree** and go to its **methods** tab.
- Click **create new method** and name it **permute**.
- Find **permutationText** in the **object tree**, and go to its **methods** tab.
- Drag **counter** into the **Do Nothing** and select **set value**, and then **0**.

![Image of world.permute method creation](image-url)
• Still in `permutationText`’s properties tab, drag text into the method.
• Select other, and type in “# Permutations: ”.
• Go to the world’s functions tab.

• Drag `a joined with b` onto # Permutations: and select default string.
• Drag what as a string (from the world’s functions) onto default string onto default string and select Expressions at the bottom, and then `permutationText.counter`.
Teacher’s instructions

• Click **create new parameter** called **numPeople** and of type **Number**.

• Drag in **teacher say** into the method, and type: “Let’s see how many ways we can arrange”.

• Onto **Let’s see...** drag a joined with b from world’s functions, and select default string.

• Drag **what as a string** onto default string, and select **Expressions**, and then numPeople.

• Drag a joined with b onto numPeople as a string and type “of you.”.

• Change the **duration** to 4 seconds.
Only `numPeople` showing

- We only want `numPeople` to show in the line.
- Drag a **Loop** into the method, select **other**, and type 4.
- Drag an **If/Else** into the **Do Nothing** and select **true**.
- Find `a < b` from the world’s functions tab onto the **true** and select **expressions**, **index**, **expressions**, `numPeople`.

![Diagram showing the implementation process](image-url)
• Click on one of the kids in the **object tree**, and drag their **isShowing** property into the first **Do Nothing**, and select **true**.

• Drag another **isShowing** into the Else **Do Nothing** and select **false**.

• Find the **elements** array in **arrayVisualization**’s **properties** tab, and drag it onto where you put the kid’s name (i.e. I dragged it on top of the **aliceLiddell**).

• Select **ith item** from **array**, then **expressions**, then **index**, and do the same for the second **isShowing** line.
playerGuess

- Click **create new variable** and the top of the method and name it **playerGuess**, and select type **Number**.
- Drag **playerGuess** into the method below the Loop, and select **set value** and then **1**, for now.
- From the world’s functions tab, drag **ask user for a number** onto the **1**.
- Select **other**, and type in “How many permutations are there for the ”.
• Also from the world’s functions tab, drag in a joined with b onto what you just typed, and select default string, for now.

• Drag another a joined with b onto default string, and select default string again, for now.

• Drag in what as a string (from world’s functions) onto the first default string, and select expressions and then numPeople.

• Into the second default string type “ people standing in a line?”. 
Setting up the example

• Drag teacher say into the method and type “Let’s see if you’re correct.”
• Drag in an If/Else statement and select true.
• Drag numPeople onto the true, and select numPeople == and then 2.
• There are only 2 permutations here: person 1 person 2, or person 2 person 1.
• We need to create a method to have two people switch places.
• Under the world’s methods tab, click create new method, and name it switch.
Writing **switch**

- Create 2 parameters of type Number: `index1`, `index2`.
- Go to `objectVisualization`’s methods tab.
- Drag in `let objectVisualization = item` and select `aliceLiddell`, or another kid, for now.
- From `arrayVisualization`’s properties tab, drag elements onto the kid you selected and select `ith` item from array `>> expressions `>> `index1`.
- This will move the person in index1 to the temporary spot so that the second person has room to move.
- From `arrayVisualization`’s methods tab, drag in `let arrayVisualization [index] = item` into the method, and select `expressions >> index1 >> aliceLiddell >> the entire aliceLiddell`, for now.

- Right click on the second line and select `make copy`, and change `<None>` to `index2`.

- From `arrayVisualization`’s properties tab, drag in elements onto the first `aliceLiddell` and select `ith item from array >> expressions >> index2`.

- In the last line change `aliceLiddell` to `expressions >> objectVisualization.item`.
Here is the animation you just created

Think about how your code makes this happen.
Incrementing the counter

- From `permutationText`'s properties tab, drag counter into the method, and select `increment permutationText.counter` by 1.
- Drag in text, select `other`, and type “# Permutations: ”.
- From world’s functions, drag a joined with b onto # Permutations: and select default string.
- Drag what as a string onto default string, and select expressions, and then `permutationText.counter`.
- Change the `duration` of all the lines except the increment line to 0.5 seconds.
Using switch

• Scroll down to the If/Else statement at the end of the permute method.
• From world’s methods, drag switch into the first Do Nothing, and select 0 and 1.
• Drag in another switch and select 1 and 0.
• Drag another If/Else statement into the Do Nothing and select true.
• Drag numPeople onto the true and select numPeople ==, then other, then type 3.
• Drag a Loop into the first Do Nothing, select 3.
Testing the world

• We can now test if the first switch is correct.
• If you want to test your whole program from the start, drag in permute from world’s methods into the bottom of my first method and select 2 for numPeople.

```
world.permute numPeople = 2
```

• If you just want to test the switch, change my first method to permute in the When the world starts line of the Events tab and select 2 for numPeople.

```
When the world starts, do world.permute numPeople = 2
```
• First we want to hold the 0th person in the 0th position and switch the 1st and 2nd positions.

• And then we want to do the same thing with the blue and yellow squares in the 0 position.

• These are the only possible 6 orderings of the 3 squares.
• This is expected because $3! = 3 \times 2 \times 1 = 6$. 
• So let’s code this...
• Drag `switch` into the *Do Nothing* and select 1 and 2.
• Drag another `switch` below that and select 1 and 0.
• Drag a last *If/Else* statement into the Else’s *Do Nothing* and select true.
• Drag `numPeople` onto the true, and select `numPeople ==`, other, and 4.
4 people switching

- We need a way to show all $4!=4\times3\times2\times1=24$ permutations of 4 people standing in a line.
- We want to hold the first person in place and then make it a 3-person problem, which we’ve already solved.
- Once it’s a 3-person problem, we make it a 2-person problem, again, as we did previously.
- The next slide explicitly shows all the switches we’ll make.
  - Try to follow the pattern: switch 2/3, switch 1/2, switch 2/3, switch 1/2, switch 2/3, switch 0/3 at the end of the column.
  - Notice that at the bottom of the third column we have to switch 0 and 3 for some reason.
• Refer to the previous slide for visualization of what we code for the 4-person case.
• Let’s see if you can build some code on your own.

- Click on the arrow at the end of index_#2 == 0 line, and select logic, or, and then true.
- Repeat the last step.
• For the 2 and 3 switches, we didn’t count the order the students were standing in as a permutation because it was easy to get them back to their original positions and count that as the final permutation.

• In this case, however, it will be more complicated to get back to the original order, so we want to count the original order as the first permutation right away.

• Drag a teacher say before the Loops and type “The order you are standing is the first permutation.”

• Then copy and paste the last two lines of switch to increment the score to 1.
• Change the first `true` to `index_#2 == 2`.
• Change the second `true` to `index_#2 == 4`.

• Drag in **switch 2, 3** into the first *Do Nothing*.
  – This is because we switch the last 2 places every other time, as shown in the chart.

• Drag an **If/Else** into the first *Do Nothing* and select `true`.

• On the **true**, create this line the same way you made the last line:
• Looking down a column in the chart, we notice that, if we start counting from 0, permutation 1 and permutation 3 have places 1 and 2 switching.
• Drag in **switch** into the first *Do Nothing* and select 1, 2.
• The only other possible value for item_#2 is 5, or the last item in the column.
• Notice on the chart:
  – Columns 0, 1: Items 0 and 1 switch.
  – Column 2: Items 0 and 3 switch.
  – Column 3: Nothing happens.
• So in the Else *Do Nothing*, create this code, noticing that we are now talking about `index` and not `index_#2` since we are referring to column numbers:

```plaintext
world.switch index1 = 0  index2 = 1

Else

    If index == 2

    world.switch index1 = 0  index2 = 3

Else

    (Do Nothing)
```
Was the player correct?

- Drag in an **If/Else** statement and select **true**.
- Drag the **playerGuess** variable onto the true and select **playerGuess ==, expressions, permutationText.counter**.
- In the first **Do Nothing**, drag **teacher say**, and type **So you were correct! Good Job!**.
- Into the second **Do Nothing**, drag another **teacher say**, and type **So you were incorrect. But you should understand the correct answer now!**.
• Below the If/Else, drag in a teacher say and type “There were ”.
• Drag a joined with b on top of that and select default string, and repeat this 4 times.
• Onto the first default string, drag in what as a string and select permutationText.counter from expressions.
• Onto the second, type “ ways to arrange ”.
• Onto the third, drag in what as a string and select numPeople.
• Onto the fourth, type “ people in a line.”
• Make this long teacher say line the same way:

  teacher \say  And  joined with  permutationText.counter as a string

  joined with  is the same as  joined with  numPeople as a string

  joined with  factorial.  more...

• Change the duration on both teacher say lines to 5 seconds, or something reasonable.
Connecting my first method and permute

- Go back to my first method.
- From world’s methods tab, drag in permute and select 2.
- Drag in permute again and select 3.
- Drag in permute again and select 4.
- These will be the 2, 3, and 4-person examples, respectively.
- Drag in a teacher say and type “That’s all for today. Good job!”
Finished!!
Play the world!