Nonvisual Arrays and Recursion

by Chris Brown
under Prof. Susan Rodger
Duke University
June 2012
Nonvisual Arrays

• This tutorial will display how to create and use nonvisual arrays in Alice. Nonvisual arrays are collections of any object or data type that don’t necessarily have to be in order in the world as opposed to visual arrays, but they are still ordered in the array structure. We will use this to store the values of our recursive function so that we don’t have to calculate it each time we want to ask the user to solve for a specific value.
Recursion

• This presentation will also show how to use recursion, which is an advanced Computer Science programming concept. Recursion is when a function must call itself with a smaller problem in order to solve a larger one. It’s similar when a word is used in the definition of the word, but using code. Here are some images showing examples of recursion below:
Recursion

• In this tutorial, our recursive function will be Fibonacci’s sequence of numbers. In Fibonacci’s sequence, each successive number is calculated by adding the preceding Fibonacci numbers. Initially, the 0\textsuperscript{th} Fibonacci number is 0 and the 1\textsuperscript{st} Fibonacci number is 1 (0, 1, 1, 2, 3, 5, 8,…). Note here that \text{fib}(x) = the \textit{xth} Fibonacci number.

• Ex: \text{fib}(2) = \text{fib}(1) + \text{fib}(0), \text{fib}(3) = \text{fib}(2) + \text{fib}(1), …\text{fib}(n) = \text{fib}(n-1) + \text{fib}(n-2)…
Standards

CSTA Standard 5.3.B- Computer Science Concepts and Practices (CT):
Students will be able to… “3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms.”

CSTA Standard 5.3.B- (CT):
“6. Compare and contrast simple data structures and their uses (e.g., arrays and lists).”
Nonvisual Arrays and Recursion

• In this world, we will use Alice to create a world where the user will have to enter the nth number of the Fibonacci sequence as prompted by the world. We will use recursion to calculate the values of the Fibonacci series and store those values in a nonvisual array.
Getting Started

- After opening Alice, choose any environment template and open it in Alice.
Getting Started

• The only thing that we will need to add to the world is a person to ask the questions and a 3D text object to keep score. Click on Add Objects and import the MadScientist in the Local Gallery in the People section.
Getting Started

• Then, scroll all the way to the right of the Local Gallery and select “Create 3D Text”. Set the string to be “0”.
Getting Started

• Arrange the objects in the world using the move objects in the top right corner. Your world should look something like this:
Nonvisual Array

• Now we’re ready to start coding. Click “Done” on the right of your screen to go back to the method editor. To create the array, go to the “properties” tab under world’s details and create a new variable.
Nonvisual Array

- Name the variable *array* and make sure that it is of type Number. Then, at the bottom of the pop-up box, check "make a" and select Array from the menu. Don’t add any items to the array yet.
You should see that the *array* variable was created in your world under the world’s properties. Now we are ready to fill in the array with the Fibonacci numbers with a recursive function.
Function

• To create the function, go to the “functions” tab under world’s details. Click on the button to create a new function and name this function `fib` and make sure it’s a Number type function.
Function

- Now in the fib function, we want to create a parameter to pass into the function when we call it. Call this parameter num and make it a number parameter.
Recursion

- When using recursion, the first thing you need to do is make sure that there is a “base case”, or a way out so that you do not get trapped in infinite recursion. In this case, the base cases are: when $num$ is 0, $fib(0) = 0$; and when $num$ is 1, $fib(1) = 1$. 
Recursion

• At the beginning of this function, drag in an If/Else statement from the bottom of the screen into the “Do Nothing” and set the value to true for now.
Recursion

- Now, under the world’s “functions” tab, find “a==b” under the math heading. Drag this over the true value in the If/Else and set the value of a to world.fib.num and b to 1.
• Now, drag in a Return statement from the bottom of the screen to below the If. We want the function to return 1 if \( num \) is 1, so Return 1 if \( a == b \) is true.
Recursion

- In the Else section, we will need to check if the value of `num` is 0. Drag another If/Else statement inside the Else and check if `num` is 0. If `num` is 0, then we want to return 0. (In the menu for Return, you may need to select “other…” and type in 0.)
Recursion

• At this point, we have finished checking for the base cases and are ready to use recursion. Recursion will be used at the bottom of the function, where it says Return 1.
Recursion

- In the world’s functions tab, drag \textit{fib} that we created over the 1 as the Return value. Alice will display a warning dealing with recursion to make sure you know what you are doing. Make sure to click Yes. Choose any value to be \textit{num}, we will change this later.
Recursion

- Notice that inside the `world.fib` function we are calling `world.fib`. That is recursion.
- Now drag the `num` parameter over the value of `num = 1` in the final Return value.
Recursion

• Select the down arrow next to $num$, and go to "math" $\rightarrow$ "num -" $\rightarrow$ "1".
Recursion

- Now select the down arrow at the end of the return statement and select “math” → “world.fib[…] +” → “expressions” → “world.fib”
Recursion

- Finally, drag the `num` parameter over `num = 1`, and click the down arrow next to `num` and go to “math” → “num -” → “2”.

- The final Return statement should look like this:
Recursion

- That concludes our recursive function. $fib$ must call itself in order to find the sum of the previous 2 values of num. Note that there are two recursive calls to world.fib in the return statement and both values, $(num - 1)$ and $(num - 2)$, are smaller than $num$, which is the value world.fib is called with originally. Now, in world.my first method, we will fill in the array with the Fibonacci values.
world.my first method

- Click on the world.my first method tab.
- Drag a Loop into the method from the bottom of the screen and choose 10 times. This will eventually calculate the first 10 numbers of the Fibonacci series. Also click on “show complicated version”.
world.my first method

• Now, drag the array into the loop and select “set <index> to <item> in world.array”. Set the index to be index and item to world.fib, both under “expressions”.
Once you have done that, then drag the index element from the loop to pass in as the parameter for \texttt{fib}.
world.my first method

- We are now finished filling in the array, and are ready to quiz the user. Create a new Number variable called *answer*. This variable will take the user’s input.
Quiz

- Drag another loop into the method and set it to be the same length as the previous loop (in this case, 10). Then move the `answer` variable into this new loop and just set the value to 1 for now.
Now, go to the world’s functions tab and under the “ask user” heading, drag “ask user for a number” over the 1 that we set as answer’s value. This function will set the value of answer to be whatever number the user inputs when prompted while the world is running.
Quiz

• Now, under the world’s functions “string” heading, drag “a joined with b” over the question asked for the value of \textit{answer}. Let b be the default string for now.
Quiz

• Then drag “what as a string” over the default string for b and select “expressions” → “index”.
Quiz

• For the value of $a$ in the question, click on it and ask something like, “What is the value of the Fibonacci series at ”.

• Now, drag an If/Else into the second loop and select true.
Quiz

- Under the world’s functions, drag “a == b” over the “true” in the If/Else statement. Set a to be \textit{answer} under “expressions” and b to be any value for now.
Quiz

• In world’s properties, drag array over the value of b and choose index as the “ith item from array”.
Quiz

- If the user gets the answer right, we want to update the score and have the MadScientist say “That’s right!” If they get it wrong then we want to have him say, “Sorry, that’s wrong.” The *say* method is under the MadScientist methods. Click on MadScientist in the object tree and drag “madScientist say” into the If/Else and enter the string after clicking “other…”.
Quiz

• For keeping score, create a new number variable called `score` and set it to 0. Drag `score` into the If part of the If/Else and choose “increment score by 1”.
Quiz

• Now to update the score on the screen, go to the properties of 3D Text and drag “text” to the top of the second loop.

• Over the default string, go to the world functions and drag “what as a string” over it, and then select score under expressions.
Conclusion

• That concludes our world! Run the world to try it out and take the quiz to calculate the first 10 Fibonacci numbers. This world used nonvisual arrays to store all of the values and recursion to calculate the values of the Fibonacci series.
Challenge

• Try creating a world similar to this one that asks users to calculate factorials, another recursive mathematical function. What would be the base case?