Simulation of Bubble Sort

Description

Sorting numbers is a classical problem in Computer Science which involves arranging a set of numbers in increasing or decreasing order. There are several popular algorithms to do this, and "Bubble Sort" is one such technique. In this program, we will simply use the bubble sort algorithm and create a graphical simulation to show how it works.

We will use an array of numbers (say 1 to 20). Associated with each number there will be a vertical bar. The pipe's length will be proportional to the number.

The program will let you shuffle the array and then sort it using the bubble sort algorithm. The bars will visually depict how the numbers move around in the array.

Scratch and CS Concepts Used

When we design this program, we will make use of the following Scratch and CS concepts. Learn these concepts if you don't know them before proceeding further.

- Algorithms
    - Using algorithms
- Animation using costumes
- Arithmetic
    - Expressions
    - Basic operators (+, -, *, /)
    - Advanced operators: mod, floor, ceiling, etc.
- Concurrency
    - Synchronization using broadcasting
- Conditional statements:
    - Conditions: YES/NO questions
    - Relational operators (=, <, >)
    - Conditionals (IF)
    - Conditionals (If-Else)
    - Conditionals (nested IF)
    - Boolean operators (and, or, not)
- Data structures – list
    - List operations
- Data types – basic
    - Integers
- Data types – strings
    - String operations (join, letter, length of)
- Looping (iteration)

    - Looping - simple (repeat, forever)
    - Looping - conditional (repeat until)
- OOP
    - Clones
- Procedures
    - Built-in
    - User defined (custom)
    - Simple
    - With inputs
- Random numbers
- Sequence
- User input
    - Text
    - Click buttons
- Variables
    - Simple
    - Properties (built-in)
    - Local/global scope

Explore the program:

If you want to play with my final program to get a feel for how it works, click the link given at the end of the article. Try not to peek at the scripts yet, since we want to design them ourselves below.

1. Click on the "Green flag": A sorted (in ascending order) array of numbers will be created. The pipes displayed on the screen will be sorted accordingly.
2. Click on the "Shuffle" button: the list will be shuffled. The pipes displayed on the screen will be shuffled accordingly.
3. Click on the "Sort" button: Start sorting the list. For each compare the pipes will be colored, for each swap the pipes will be swapped.

High Level Design

This is where we take a step back and try to get our arms around the task of writing this program. We will first understand the basic features and the flow of operation.

There are two parts to our program. The back-end (or invisible) part does the actual shuffling and sorting of numbers. The front-end consists of vertical bars that represent the numbers as they are shuffled and sorted.

For the backend, we will use the standard Bubble Sort algorithm given at https://en.wikipedia.org/wiki/Bubble_sort. Since we are sorting numbers, we will use a list to hold these numbers.

For the front-end, we will use vertical bars to represent the numbers. Since all bars are identical except their height, we will use a single "bar sprite" and clone it as needed. But, how will we vary the height of each clone? We will use a visual trick: each bar would actually have the same height (the same as Scratch vertical height), but the bar could be *vertically positioned* such that it *appears* short or tall.

We could design the back-end first, that is, create all the logic of the number list, shuffling the numbers, and running the sorting algorithm on the list, etc. We could then design the front-end, that is, create the bar sprite, create its clones (as many as the numbers in the list), and position them on the screen such that they correctly represent the numbers. Finally, we could put in code which synchronizes movement of bars concurrently with the shuffling and sorting of numbers.

Objects:

As usual, we will use object-oriented design techniques to build our program. The first step is to list the required objects. Here is our initial list:

- Back-end logic (some invisible sprite)
- Bar
- Click-buttons to "shuffle" and "sort"

Global data:

We will have a List called "L" to hold the array of numbers.

Let's now build the program feature by feature and design the objects incrementally.

Initial Version

In the initial version of the program, we will work on the following feature ideas:

- Add code to the back-end object to shuffle and sort a list of numbers.
- Add "shuffle" and "sort" click-buttons for user to request these operations.

Feature idea #1: The back-end logic

Create logic to build the list of numbers, shuffle it, and sort it using Bubble Sort.

Design:

The list L will initially be loaded with numbers from 1 to N where N is specified by the user. Obviously the list will be sorted initially. The "shuffle" algorithm below will ruin this order. Then we can pass it through the "bubble sort" algorithm to sort it again.

Create custom blocks as per the algorithms below.

Shuffle list:

(adapted Donald Knuth's algorithm)

```
I = 0
```

```
Repeat N times:
      Pick a random integer R such that 0 <= R <= N-I-1
      T = I + R
      Swap numbers between locations T+1 and I+1
End repeat
```

Bubble sort:

(N is size of number array. Assuming number array base is 1)

```
i = N + 1
Repeat until i=1
      Newi = 1
      j = 2
      Repeat until j = i
            If A[j-1] > A[j] then
                  Swap(A[j-1], A[j])
                  Newi = j
            End if
            j = j + 1
      End repeat
      i = Newi
End repeat
```

Feature idea #2: Shuffle and Sort buttons

Create buttons using which user can shuffle or sort the list.

Design:

We will use two new objects: shuffle and sort button sprites whose job simply is to send messages then they are clicked.

Save as Program Version 1

Let's save this project before continuing to the remaining ideas. Compare your program with my program below.

Solution: BubbleSort-1.sb2

How to run the program:

1. Click on the "Green flag" to initialize the list of numbers.
2. Click "Shuffle" to shuffle the list.
3. Click "Sort" to sort the list.

Final Version

In this next version of the program, we will work on the following feature ideas:

- Create vertical bars that represent each number in the list.

- Make the bars move on the screen when the numbers are moved within the list (during shuffling or sorting).
- Control the speed of movement and highlight the bars so that the user can understand how the algorithm works.

Feature idea #3: Vertical bars

Create vertical bars that represent each number in the list.

Design:

As mentioned earlier, we will use a single bar sprite and use its clones to represent the numbers. This is a vertical bar as high as the screen. It should hide somewhere so that it doesn't interfere with its clones. All its clones will be positioned on X axis. Each bar will have unique X and Y positions. Once created, the Y position of each bar remains unchanged throughout the program.

Let us try to understand what each clone needs to know and do.

Step 1: During setup arrange the bars left to right on the screen.
Arrange such that the first bar will represent the first number in the list, the second bar will represent the second number in the list, and so on. We will spread out the bars to cover most of screen.

Design:

We can achieve this by having each clone calculate its "x position", which should be straightforward. We can also have each clone calculate its height (i.e. its "y position") by looking at the number it represents. This should also be straightforward.

Step 2: Relate each clone with its associated number in the list L.
That is, how would a clone know which number it represents?

Design:

We have two options: one is to save the number itself as a "private variable" of the clone, or, save the location (aka index) of the number in list L. If you think about these options carefully, it is clear that the latter option is better, because using the index we can quickly find the number, but the other way round is not so easy.

Step 3: During shuffling and sorting, move the bars as their corresponding numbers move in the list.

Design:

After a careful study of the shuffling and sorting algorithms, it is clear that the movement of bars happens when a pair of numbers is "swapped". That means a pair of corresponding bars would be swapped on the screen.

Every time a swap happens in the list, we could save the two indexes in a couple of global variables swap1 and swap2 and send a broadcast. Every clone will then check if its own index matches one of these two. If it does, it will save the other index as its own and recalculate its new "x position".

Go ahead and write the above scripts for the "bar" object.

Feature idea #4: Delay and highlight

Since the main purpose of this program is to learn about the Bubble Sort algorithm, it would be nice to able to watch the algorithm in slow motion.

Step 1: Add the feature to control the speed of every swap.

Design:

This can achieved by having a slider variable called "delay" and using "glide" instead of "go to x y" when we move the bars.

The problem is this will also slow down the "shuffling" of bars, which we are not really interested in watching in slow motion. The easy fix for this is to set the delay to 0 temporarily during shuffling.

Step 2: Highlight the pair of bars which has been selected to be swapped.

Design:

This can be achieved by having the selected bars change their color before the swap and resetting it after.

Step 3: Vary the thickness of bars based on the total count. The bars should be thinner if the count is large and thicker if the count is small.

Design:

Since there is no way in Scratch to resize only the thickness of bars, we will use 3 separate costumes to accommodate different ranges. One is for 5 to 20, second is 20 to 50, and third is 50 to 100. The choice of the appropriate costume would be made during setup.

Save as Program Version "Final"

Congratulations! You have completed all the main features of the program. Compare your program with my program at the link below.

Solution: BubbleSort-final.sb2
Link: https://scratch.mit.edu/projects/356571544/

How to run the program:

1. Click on the “Green flag”: A sorted (in ascending order) array of numbers will be created. The bars displayed on the screen will be sorted accordingly. Set the "delay" slider appropriately.
2. Click on the “Shuffle” button: the list will be shuffled. The bars displayed on the screen will be shuffled accordingly.
3. Click on the “Sort” button: Start sorting the list. For each compare the bars will be colored, for each swap the bars will be swapped.

Author: Abhay B. Joshi (abjoshi@yahoo.com)
Last updated: 30 December 2019