CSC 355 PROJECT 3 ANIMATION OF SORTING ALGORITHMS

PROFESSOR GODFREY C. MUGANDA

Use WPF on the .NET platform to write a program that is capable of animating the working of a selection of sorting algorithms.

Upon start-up, the program should display a window with a menu that has two menu items:

(1) *Array Set Up*: this menu item will result in the display of a dialog box that asks the user to enter the size of an array. When the size of the array is created, the dialog box will generate an random array of type double that has the specified number of entries and make that available to the program. (Each array entry x will be in the range $0 \leq x < 1.0$). Initially, this is the only menu item enabled. When the dialog box returns, the *Animation Type* menu, described below, will become enabled, and this menu item will become disabled.

(2) *Animation Type*: This menu item has a submenu that drops down and has two menu items:

   (a) *Single Algorithm Animation*: This menu item has a submenu with the following menu items

      (i) *Insertion Sort*: this menu item will run an animation of the Insertion sort algorithm on the previously-created array.

      (ii) *Quicksort*: this menu item will run an animation of the Quicksort algorithm on the previously-created array.

   (b) *Multiple Algorithm Animation:*This menu item also has a submenu with the following menu items

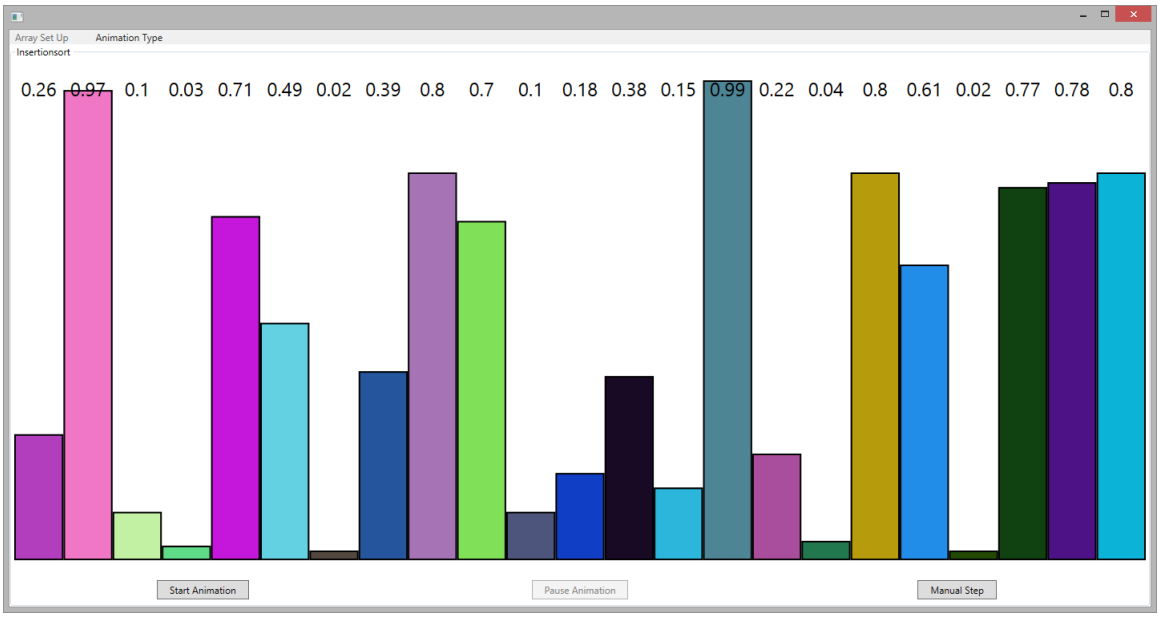
      (i) *Insertion Sort*

      (ii) *Quicksort*

      The difference between *Single Algorithm Animation* and *Multiple Algorithm Animation* is that the submenu on the latter has checkable menu items. Beyond making these menu items being checkable, you are not responsible for implementing the multiple animation.

1. Animation View and Control

Selecting an algorithm to animate will create a visual representation of the array to be sorted. This visual representation should consist of an array of rectangles, with a rectangle for each array entry. The array of rectangles should be placed in a *Canvas* panel, with the kth rectangle being in the k position from the left edge of the canvas, and the height of the rectangle being the height of the canvas multiplied by the $k$the entry of the array. The rectangles will all have the same width, which will be the width of the canvas divided by the length of the array. This is shown in the following figure



In this user interface, you will notice that the in addition to the canvas displaying the a view of the array to be sorted, there are three buttons at the bottom to control the animation:

(1) *Start Animation:* This starts, restarts a timer that controls the operation of the animation. Each timer tick will cause a single step of the animation to be performed.

(2) *Pause Animation:* Stops the timer that controls the animation.

(3) *Manual Step:* Each click of this button will cause a single step of the animation to be performed.

By a single step, we mean the execution of code that involves neither repetition (a loop) nor a call to a recursive method.

Notice that the Animation View is a group box that must display the name of the algorithm being animated.

The display of the values of the array entries at the top is optional. If you do implement it, you should use *TextBlock* objects.

Finally, you must put the group box that contains the Animation View inside of a *Viewbox* object.

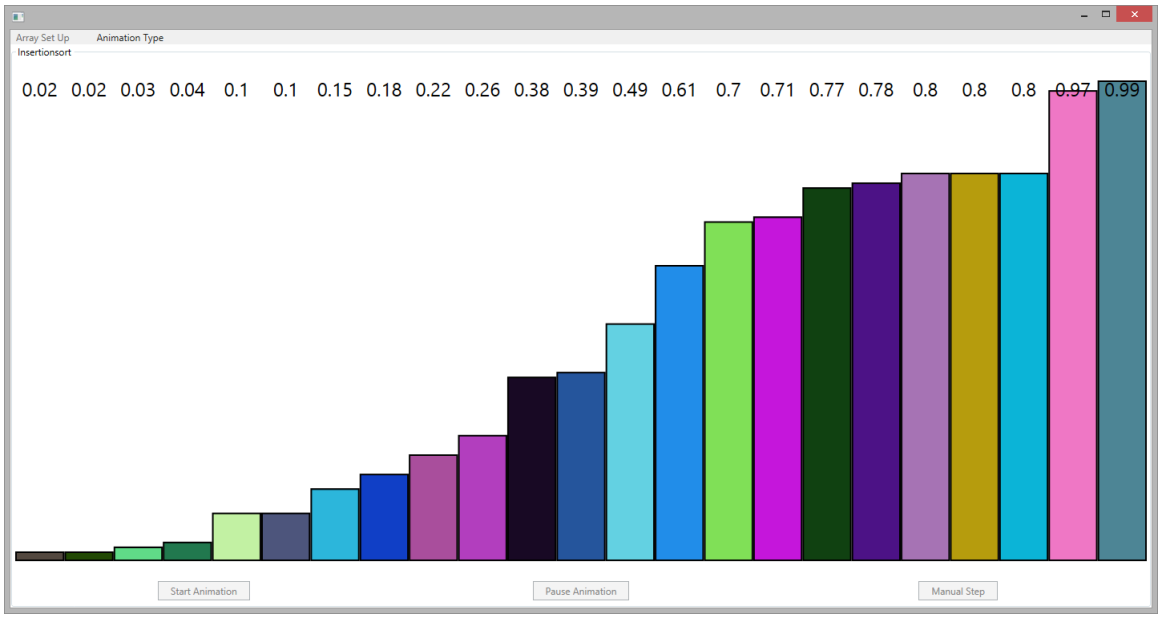
The following usability conditions on the three control buttons should be enforced: Initially the, *Pause Animation* button is disabled and the other two buttons are enabled. Whenever the animation is running on the timer, the *Start Animation* and *Manual Step* buttons must be disabled, so that only the *Pause Animation* is enabled. Whenever the animation is paused, the *Pause Animation* button is disabled and the other two buttons are enabled.

All three buttons become disabled once the animation is completed (the array has been sorted).

2. Operation of the Animation

Both algorithms being animated are based on swapping of pairs of elements. This will be displayed by by making the corresponding rectangles appear to swap positions.

Here is a screen shot of the sorted array at the end of the animation:



3. Due Date

This is due Monday of Week 6.