CS 106 Winter 2017

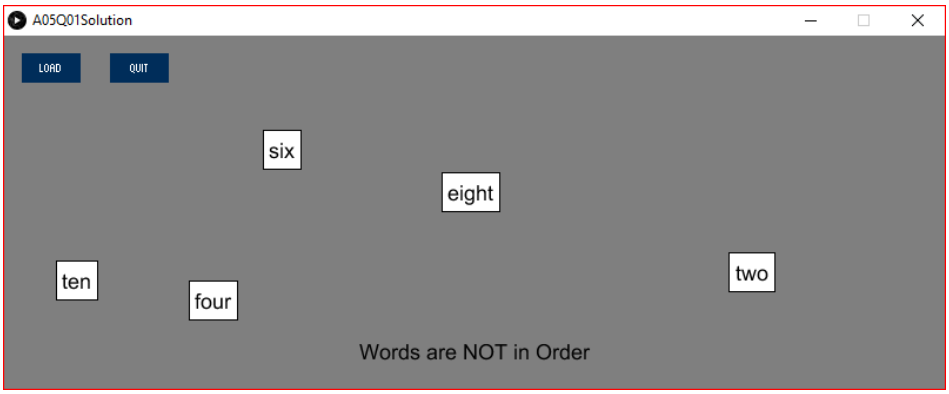
Assignment 05: User Interfaces

Due: Friday, February 8th, 11:59pm

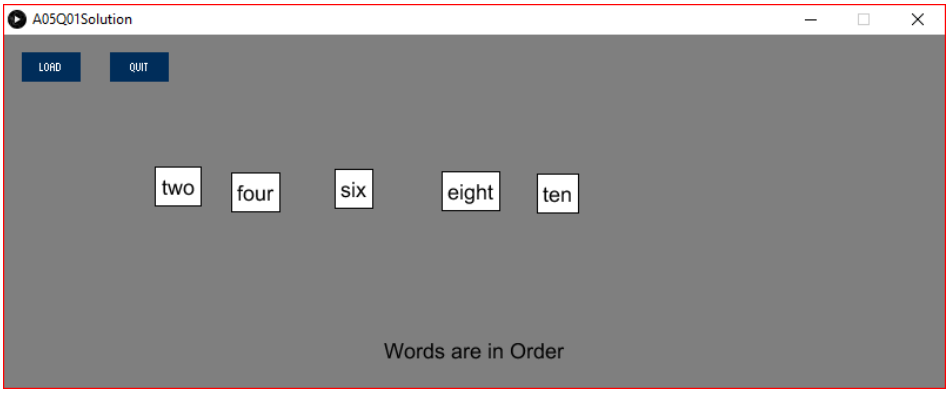
This assignment about "User Interfaces". This topic includes Direct Manipulation and ControlP5 and was covered in class on the week of February 4th. If you need to refresh yourself, review the slides that were presented in class and the demo code examples that were provided. They are both on LEARN.

This question uses both ControlP5 and Direct Manipulation. You place numbers at random locations on the sketch window. The users task is to arrange the numbers in order from left to right. See the YouTube video. https://youtu.be/mcXc8TUQOSs

The original screen may look like the following:



After you arrange the numbers in order from left to right it may look like the following.



To begin, load the provided starter code A05Q01. You'll notice that the `A05Q01` sketch contains two tabs. The first is the main sketch program, which has just a bit of code written for you. The second is a helper class called `Word`, which holds one word and knows how to measure the rectangle that the word will be drawn inside of (its "tile"). Read through the class, and try to get some sense of how to use the features that are provided (and what features you might add to it). With that starter code, we suggest you proceed as follows:

1. Add the ControlP5 library to the sketch. You'll need the `import` directive, a global variable, and a statement that initializes the global variable inside of `setup()`. Note that the `BusyBox` sample sketch demonstrates the use of ControlP5.

2. When your program starts it should read in the text file Numbers.txt (i.e. loadStrings()). Save the

```
one two three four five
two four six eight ten
zero one two three four five six
```

   array returned by loadStrings() in a global variable.

3. There is a "Load" button. Pressing the button causes a random line to be chosen as the current line (i.e. it will be either the first line, or the second line, or the third line). That line needs to be broken apart so that each word is in an array (i.e. splitTokens()). After doing splitTokens() you will have a global array of type String with the elements {"one", "two", "three", "four", "five"}, or {"two", "four", "six", "eight", "ten"}, or {"zero", "one", "two", "three", "four", "five", "six"}.

4. Add a global variable to the sketch to hold an array of the `Word` objects corresponding to the tiles of the numbers from the current line that we are using (i.e. {"one", "two", "three", "four", "five"}, or {"two", "four", "six", "eight", "ten"}, or {"zero", "one", "two", "three", "four", "five", "six"} . When the sketch starts, the array should be initialized to a new array with zero elements in it. See the `DirectManipCircleArray` sample sketch for inspiration. Then when the user presses "Load" the array of Word objects gets filled with the words from the current line.

5. Add code to draw all the `Word`s in the array. This will certainly involve a loop in the main `draw()` function. As for drawing each individual `Word`, you can put all the code for that in `draw()` if you want, but a more elegant approach is to add a new `draw()` method to the `Word` class and then call that method from the sketch's main `draw()` function. (You won't be able to see this working at first, because you need to populate the sketch with some words. That'll happen in the next step.) The `Word` class gives you the (*x*, *y*) at which to draw a magnet's text, and it provides helper methods to allow you to figure out the rectangle to draw around the word.

6. Add `mousePressed()`, `mouseDragged()`, and `mouseReleased()` hooks to the sketch that support direct manipulation of all number tiles currently visible. Make sure to hit test the tiles in the opposite order that you draw them. You may want to implement hit testing as a method inside the `Word` class. Again, the `DirectManipCircleArray` sample sketch could be a useful source of inspiration for this step.

7. Put a message at the bottom of the screen to indicate whether the numbers are in order from left to right.

8. Put a "Quit" button. If the user presses "Quit" the program ends (see the quit() function).

Requirements and Grading

QUESTION ONE: (14 marks)

[10 marks] Correctness

- The sketch must meet the following requirements:
    - The “Load” button must load a line of words/numbers and create an array of Word objects.
    - The words/numbers must display at random locations on the sketch window.
    - The user must be able to move the words/numbers using Direct Manipulation.
    - The text at the bottom of the sketch window must indicate whether the words/numbers are in order from left to right.
    - The “Quit” button must end the program.

[2 marks] Coding Style

- Comment your code appropriately. Avoid superfluous comments.
- Correctly and consistently indent your code blocks.
- Use correct inline spacing in function calls, function definitions, and variable declaration and assignment.
- Use good line spacing to chunk sections of your code.
- Pay special attention to inline spacing for your conditional statements
- One or more marks may be deducted for solutions that have obvious inefficiencies.
- Variables that are declared or assigned, but not used.
- Unnecessarily repeating the same code in multiple places.

[2 marks] Visual Design and Creativity

- Higher marks will be given to sketches with extra details for creativity and artistic appeal.

Submitting

Create a folder "A05_username", but replace "username" with your UW id. So if your email is "jac926@edu.uwaterloo.ca" you would create a folder "A05_ jac926".

SAVE your sketches in that folder as "A05Q01_username". Again, replace username with your UW id.

Zip your "A05_username" folder (with "username" replaced by your UW id) and submit it the correct assignment dropbox.

It is your responsibility to submit to the correct dropbox with the correct files before the deadline. Otherwise you will have marks deducted.

Academic Integrity

All assignments in CS106 are done individually. Group work and sharing of code is not allowed.

Detecting Plagiarism:

- We monitor Reddit, File Trading Sites, past year CS106 assignments, etc.
- We use Measure Of Software Similarity (MOSS)
    - automatic system for determining the similarity of code

Discipline

Discipline (Policy 71)

- https://uwaterloo.ca/secretariat-general-counsel/policies-procedures-guidelines/policy-71