CS106 W21 - Assignment 1

Due: Friday, January 22, 11:59 PM

Note: On each assignment we will be running a program called MOSS to detect cheating.

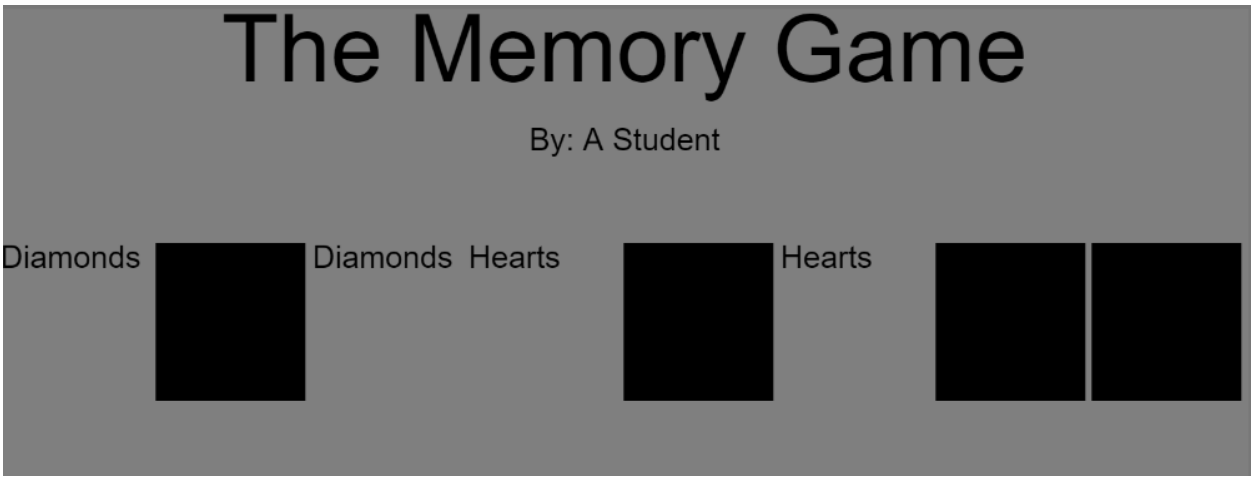
Assignment 1 is graded out of 31 marks.

In **A1_basic**, you will implement as much required functionality as you can without any enhancements. Your goal is to match the video demonstration. This is the sketch the markers will evaluate for required functionality marks. Do this first.

In **A1_enhanced**, you can extend and deviate somewhat from the required functionality to show off your programming skill and design creativity. Do this second.

You are to write a JavaScript p5 program to mimic what you see in the following video: https://vault.cs.uwaterloo.ca/s/M7YPSazPK87Tm36

There is a sketch provided in the Assignment 01 "collection" in Open Processing called "A01 TheMemoryGame_StarterCode" (https://openprocessing.org/sketch/1061553#code). Load it. Fork it. To fork you "Save as Fork". "Save as Fork" will appear after you have made modifications to the file. Save as "A1_basic_TheMemoryGame". Edit the settings for the sketch so that it is shared with "



Requirements and Grading

A1_basic_TheMemoryGame [20 marks]

Use the provided Starter Code. You do not need to modify the Starter Code, but you may if you wish. However, do not change the sketch size (800x300) and keep the background the gray colour.

Before you start, watch the video which is linked above.

1) [2 marks] Shuffle the array named “pairs”

2) [2 marks] When the game starts, the words from the array “pairs” should be displayed (using text()) but cannot be seen. They will be in random order as they were just shuffled in the previous step.

3) [2 marks] When the game starts eight rectangles should cover the words (i.e. the rectangles should cover the words that were displayed in the previous step).

4) [2 marks] The user should be able to click on one rectangle to reveal the word underneath. If the user does nothing else then nothing happens. The game just sits there.

5) [2 marks] The user should be able to click on a second rectangle to reveal a second word. At this moment, two words are revealed.

6) [2 marks] At this moment, the user should not be able to uncover any other words. If the user clicks on another rectangle then nothing should happen.

7) [2 marks] If the two reveal words are identical then nothing happens, and 2 seconds later the user can select other rectangles. If the two words are not identical, then they get covered again after 2 seconds (i.e. use a timer).

8) [2 marks] The game should then play as shown in the video with the player selecting two rectangles at a time until the game is over. When the game is over then nothing happens the game just sits there without any changes.

9) [2 marks] at any time the user can press “r” or “R” for reset and the game resets with the words being shuffled and all words covered.

10) [2 marks] The game must have a title and the student name, as shown in the video. (but replace “A Student” with your name).

[6 marks] Your enhancement

A1_enhanced_TheMemoryGame

Once you have basic functionality working for A1_basic_TheMemoryGame above, enhance the functionality or the visual design in a sketch called A1_enhanced_TheMemoryGame (to create A1_enhanced_TheMemoryGame, you'll need to fork your A1_basic_TheMemoryGame sketch.

This basic requirement must still be followed:

- It must still be a memory game where the user reveals a pair of hidden items.

You must create three enhancements, each valued at 2 marks for a total of 6 marks. Some ideas:

- Allow the user to change the number of hidden items from 8 to some other number such as the user can chose 8, 10, or 12 items.
- Change the game so that the user reveals images rather than words. That is, four pairs of images would be hidden behind the rectangles.
- Change the game so that rather than just a line of horizontal rectangles, there is a two-dimensional grid of rectangles such as a 4x4 grid.
- Make the game much more graphically pleasing by adding your own graphics and graphical effects.
- Add a help screen that explains how to play the game.
- Add a splash screen that the user sees first, and the user has to press a certain key to continue.

Note: Little or no marks will be given for simple enhancements such as making the background red. Your enhancement must truly enhance A1_basic by making use of your programming and/or visual design skills. There would be very little programming or visual design skills involved in simply changing the color of the background to red and thus an enhancement such as that would receive a grade of zero. Conversely, having a good implementation of a quality enhancement such as one of the "some ideas" above would be graded up to two marks each depending on the quality of the enhancement, for a total of six marks.

You must describe your three enhancements in comments at the top of your code (do this after the first three lines of code which are your name and student ID and a blank line). The TAs need to know what enhancements you made so they can grade each.

It's ok to not add enhancements or be extra creative if you're running out of time: a correct basic solution with excellent coding style will still achieve a grade of 25 out of 31.

Save this as A1_enhanced_TheMemoryGame.

[5 marks] Coding Style and Efficiency

Follow the course coding style for whitespace and comments. Consult the **"Code Style Guide"** on LEARN. For example:

1) [0.5] Include your name on the first line of code and your student ID number on the second line of code.
2) [0.5] Leave the third line blank.
3) [0.5] Comment your code appropriately. Avoid superfluous comments.
4) [0.5] Correctly and consistently indent your code blocks.
5) [0.5] Use correct inline spacing for variable declaration and assignment.
6) [0.5] Use good line spacing to chunk sections of your code.
7) [0.5] There are no variables that are declared or assigned, but not used.
8) [0.5] There are no unnecessary variables that are duplicates of other variables.
9) [0.5] There is no unnecessarily repeating the same code in multiple places.
10) [0.5] Semicolons were used appropriately (i.e. at the end of most lines).

Restrictions

- You may not use any functions or statements not covered in lecture or labs.
  This includes, but is not limited to:
    - No translate(), rotate(), or scale() functions.

Submitting

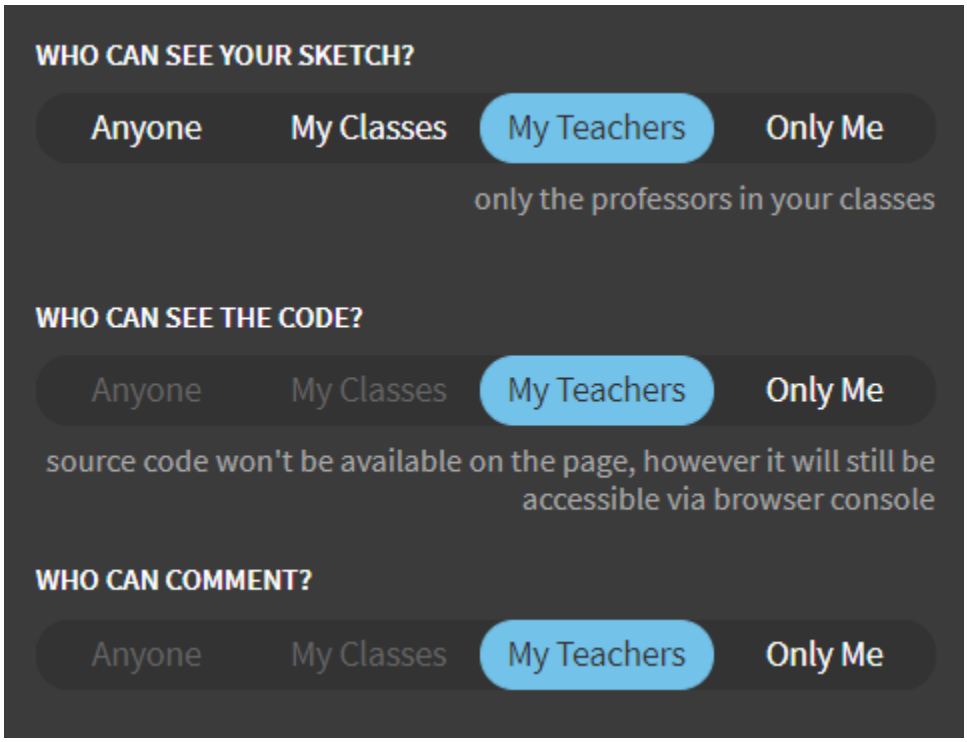
Use the template file in Word "CS106 Assignment Template" in LEARN to create your Assignment 01 submission.

Then convert your Word file to pdf. Please ensure that your URLs are hot links. The TAs need to be able to click on each link in your pdf and go directly to your sketch.

So for example, don't have a link like this: https://openprocessing.org/sketch/1050954

but rather have that link as a hot link as follows: https://openprocessing.org/sketch/1050954

Ensure that each URL you submit has its settings so that the access is as follows:



Submit that pdf file to the Assignment 01 dropbox on LEARN.

An example of how to submit a Lab is shown in the following video:
https://vault.cs.uwaterloo.ca/s/9Xx7AGsewaea773

It is your responsibility to submit to the correct dropbox with the correct file before the deadline. Otherwise you may receive a mark of 0.

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.
- 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