# Processing Assignment #2 Garden CMSC 101 / IS 101Y – Fall 2013

## Due Date

* **Processing Assignment 2: Due Thursday, September 26**
  + ***Note:*** *The late policy for Processing assignments applies to this assignment (it may be submitted up to one week late, with a 15% penalty per day or fraction of a day late). However, you are* ***strongly*** *encouraged to start early and complete the assignment on time, so that you are well prepared to start the group project!*

## Collaboration Policy

* This assignment ***must be completed individually.***You may discuss the requirements with other students, but you must write your own code, and you may not share your code with any other students, search for code on the Internet, or use solution material that was created by anyone but yourself. If you have any doubt whether what you are doing constitutes plagiarism, ***ask the course staff for guidance*.**

## Assignment Summary

The goal of this assignment is to design a simple Processing application that shows a garden layout. Most gardeners decide what to plant in their gardens during the cold winter months, so that they will be prepared for the planting season when spring arrives. Your task is to automate the planning process by showing a layout of the planned garden.

You’ll implement your garden planning application in Processing using concepts you’ve seen in class or read about in the Processing book. Your program should accept user input about the number and kind of plants. It should also have a button that, when clicked, causes the program to lay out a garden, spacing the plants relatively equally in a square.

For example, a garden with 16 carrots in a square layout would look like this:

16-carrots.tif

If the number of plants isn’t a perfect square, then you should round up to the nearest square for the layout, and “fill” the garden with plants starting at the top left. Here’s what a layout for 22 flowers would look like:

22-flowers.tif

## Requirements

Implementing the requirements in this order, and testing as you go along, will be very helpful for getting your program working correctly. It is not suggested that you simply dive in and start implementing in any order – do so at your peril!

* Be sure you include the “import” command and the two variable definitions that are included in the Provided Code at the end of this document. The “import” command is necessary for getIntInput and getTextInput to work correctly. You don’t need to understand what it means (but feel free to ask if you want to know!) The other two lines define MaxPlants, a constant for the maximum number of plants (which your code should use, but the value of which you can change) and numPlants, a variable for the actual number of plants requested.
* Use the provided getIntInput() function (see Provided Code) to input a number of plants from the user and set the value of numPlants. The popup window should include an appropriate text prompt, and you should check to be sure that the number of plants is a legal value (between 1 and MaxPlants).
* Use the provided getTextInput() function (see Provided Code) to input the name of the type of plant to be placed in the garden. You should include an appropriate text prompt, and error checking to be sure that the name of the plant is one that your program knows how to draw.
  + Clip art images of a carrot, a flower, and a tomato are posted on the website – you can use these or find or generate your own pictures.  
    http://www.csee.umbc.edu/courses/undergraduate/CMSC101/  
     Fall2013/Handouts/flower.png  
    http://www.csee.umbc.edu/courses/undergraduate/CMSC101/  
     Fall2013/Handouts/carrot.png  
    http://www.csee.umbc.edu/courses/undergraduate/CMSC101/  
     Fall2013/Handouts/tomato.png
* Write a function called drawButton() to create a button with a useful text label (e.g., “Plant carrots” if the user has asked to plant carrots). This function should be called from setup().
* When this button is clicked, your program should call a function you will write called drawLayout(). (You should check for a mouse click in the draw() function, similarly to the examples we did in class.) The drawLayout() function should generate a square layout and draw, and the selected plants (top to bottom, left to right, as shown in the examples above).
  + You may use the nearestSquare() function (Provided Code) to determine the correct size of the square layout for the number of plants selected.
  + You will need a nested pair of “for” or “while” loops to draw the appropriate number of plants. There are several different ways to design this functionality in order to “stop early” instead of filling the entire square.
* For your submission, you must save a picture of the display your program creates when you ask it to draw 28 flowers (or other plant if you use your own plant images). As in Processing Assignment 1, you should include the provided keyPressed() code in your program (see Provided Code). Run your program using the “Play” button, enter 28 at the “number of plants” prompt, and “flower” at the “type of plant” prompt. Then type the “s” key, which will save an image file called “flowers.png” into your sketch directory.

## Extra Credit

Up to 10 points of extra credit are available for adding extra features. These are up to your imagination, but some ideas include:

* Allow the user to choose a different layout (other than a square), such as a circle, rectangle, or triangle.
* Let the user draw the outline of their garden using mouse clicks, and fill this garden with plants.
* Permit the user to alternate multiple types of plants in their garden.

## Additional Reminders and Suggestions

* Always design first, then implement, following the design process we established in class.
* Always test incrementally. Once your design is ready, implement a small part of it (e.g., drawButton) in Processing, and test it thoroughly before moving to the next program component.
* Be sure to follow the class coding standards (see handout).

## Submitting Your Assignment

1. Submit your code using **Blackboard**:

* When you save your Processing sketch, name it <username>\_garden (Example: mariedj\_garden).
* Go to Blackboard and click on Course Documents.
* Upload <username>\_garden.pde into the assignment “PA2.”
  + IMPORTANT NOTE: If you have any companion files (e.g., images other than the standard provided ones), you must instead submit a .zip file that contains your entire sketch directory with the images and .pde file. If you are not sure how to generate a zip file, consult with any of the teaching fellows (how you do this depends on the type of computer you are using).

1. Turn in (in class on the due date) a **hard copy** description of your project. It should have the following elements:

* Your name, user name, and UMBC email address.
* A very short summary of your program’s capabilities, including any extra credit that you implemented.
* The names of anyone you discussed the project with. If you discussed it with no one, say so.
* A signed statement that you wrote your own code.
* A printout of your <username>\_garden.pde program.
* A picture of the garden layout produced by your program when the user specified a layout of 28 flowers (i.e., the “flowers.png” image that you created in the final step of the assignment).

## Provided Functions

You should type all of the code below into your program. You do not *have* to use these functions if you choose to implement the program in a different way, but it is highly recommended. Any changes you make to this code should be clearly documented.

import javax.swing.JOptionPane;

final int MaxPlants = 100;

int numPlants;

/// YOUR CODE SHOULD GO HERE

/// PROVIDED FUNCTIONS (any modifications should be clearly documented)

int nearestSquare (int numPlants) {

int i;

for ( i=0 ; i <= sqrt (MaxPlants) ; i++ ) {

if ( i\*i >= numPlants) {

return (i);

}

}

println ("ERROR: no nearest square for " + numPlants);

exit();

return(0);

}

int getIntInput(String query) {

return Integer.parseInt(JOptionPane.showInputDialog (query));

}

String getTextInput (String query) {

return JOptionPane.showInputDialog (query);

}

void keyPressed() {

if (key == 's') {

save("flowers.png");

}

}