CS1112 Fall 2015 Project 6 Part C **due Thursday 12/3 at 11pm**

You must work either on your own or with one partner. If you work with a partner you must first register as a group in CMS and then submit your work as a group. *Adhere to the Code of Academic Integrity.* For a group, "you" below refers to "your group." You may discuss background issues and general strategies with others, but the work that you submit must be your own. In particular, you may discuss general ideas with others but you may not work out the detailed solutions with others. It is not OK for you to see or hear another student's code and it is certainly not OK to copy code from another person or from published/Internet sources. If you feel that you cannot complete the assignment on you own, seek help from the course staff.

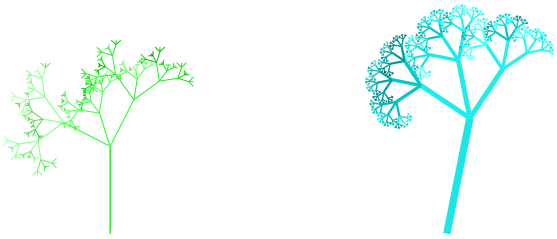
Objectives

Completing this project will solidify your understanding of structs and struct arrays (Part A), object-oriented programming (Part B), and recursion (Part C).

Parts A and B appear in separate documents.

3 Generating a Synthetic Tree

[**Before you begin, read** *Insight* **§14.1**; it will help you solve this problem.] Computer generated scenes are used frequently in games and movies. Recursive algorithms see widespread application in the generation of terrain, textured landscape (e.g. flying view of a forest), and sky. See the two "trees" below for example.



In order to "generate" a tree, we must design a pattern. Our basic tree has just one main stem (or trunk). At the top of a stem it may grow three other stems (or branches) at predetermined angles. As it "grows" the stems become shorter and thinner. To introduce variability we add random noise to the color as the tree grows.

Implement function `tree`:

```
function tree(n,x,y,stemAngle,stemLength,s,colr,branchRatio,...
              bendAngle,branchAngle)
% Draw a level n tree with the root at position (x,y) using recursion.
% The main stem of the tree has angle stemAngle, length stemLength, color
%   colr, where colr is a length 3 rgb vector, and a width that correlates
%   with n.
% The lowest level tree has n=0 and is just the main stem--no branches.
% For n>0, at the top of the main stem draw three level n-1 trees.  Each of
%   these trees has color colr+r, where r is a random value from the Normal
%   Distribution with mean zero and standard deviation s.  Assume 0<s<1.
%   The main stems of these three trees differ in the following way:
%   Center branch: stem at the angle stemAngle+bendAngle,
%                  stem length of (1-branchRatio)*stemLength
%   Side 1 branch: stem at the angle stemAngle+bendAngle-branchAngle,
%                  stem length of branchRatio*stemLength
%   Side 2 branch: stem at the angle stemAngle+bendAngle+branchAngle,
%                  stem length of branchRatio*stemLength
```

You can think of the "level" of the tree as the generation. In the diagrams above, the left tree is level 5 (or "has grown five generations") and the right tree is level 6. A level 0 tree has just the main stem and no branches. Here is an example level 1 tree:  You must use recursion, not iteration, to generate the tree. The given script `drawTree` was used to generate the level 5 tree shown above.

Stem width You decide how to determine the width of the stem in a generation (level) as long as the width correlates to `n`. For example, when drawing a level 5 tree, the first stem drawn, corresponding to `n`=5, is wider then the stems drawn for `n`=4. For your information, the default line width for `plot` is 0.5. The example below demonstrates how to change the `LineWidth` and `Color` properties of a line drawn using the `plot` built-in function:

```
width= 2.5;  % line width must be positive
colr= [1 0 .2];
plot(xpos, ypos, 'LineWidth', width, 'Color', colr)
```

Random Variability The color is modified from one generation to the next. Use "normally distributed" instead of uniformly distributed random numbers, i.e., use `randn` instead of `rand`. The statement `r=randn` assigns one random number from the standard normal distribution to variable `r`. The standard normal distribution has mean zero and standard deviation one, which says that about 70% of the time the random number would be in the range -1 to +1 (zero plus or minus one standard deviation), with values closer to zero more likely to occur. To obtain a random number from the normal distribution with mean zero and standard deviation s, scale the value returned by `randn` by s, i.e., `r = s*randn`. Recall that a color in MATLAB is a vector of length three, where each component is a real number in the range of zero to one. You may need to correct (e.g., truncate) a calculated color value so that it is in the correct range.

Experimentation and Testing During initial development of your program, change the value of `n` in `drawTree` to 1 to make sure that your code can draw a level 1 tree. Be sure to test your code for `n`=0 as well. Once you get your code correct for these basic trees, experiment with higher levels and with the "tree pattern". Design your favorite tree—modify `drawTree`!

Submit your files `tree.m` and `drawTree.m` in CMS before the deadline.