Procedural Generation Assignment #6: Grammars and L-Systems

[Due 04/20/2020 before the start of class]

The goal of this assignment is to give you a chance to implement and experiment with two applications of Grammars for procedural generation, both for generating text and for generating visuals (in this case, plants).

To experiment with an implementation of grammars for text generation, check out tracery:
http://tracery.io/

To experiment with an implementation of L-Systems for visual generation, check out this demo:
http://www.kevs3d.co.uk/dev/lsystems/#

For a brief introduction to L-Systems and some example rules to try out, check out this page:
http://paulbourke.net/fractals/lsys/

To start the assignment, download the template project from this address:
https://github.com/badtetris/ProcGenAssignment6

1. Open the scene named "Task1". Complete the function named "applyRules" in Grammar.cs to complete the implementation of a generative grammar. Keep in mind whether or not the "expandInParrallel" value is set and make sure that rules are chosen randomly if there are multiple rules that match a symbol. Test out your grammar implementation on the existing text generator. You're free to change the example text and rules to whatever you want.

[2 points]

2. Open the scene named "Task2". This task makes use of your work from Task1, but now, instead of generating text with your grammar, you'll be generating an image. Complete the "displayPlantFromString" function in PlantDisplay.cs to implement the missing drawing instructions. Test out your implementation on the existing plant generator. Feel free to change the grammar rules to whatever you want to test out different patterns.

[2 points]

3. Open the scene named “Task3”. For the final two points, expand or mod the existing Grammar generator from Task1 and/or Task2 in some way and save your mod to this scene. How you mod it is up to you, but your goal should be to make something neat. Here are some **suggestions**:

- Make another type of generator that uses your grammar implementation. For instance, can you think of a way to generate a “dungeon” by first generating a string from your grammar?
- Make a game that takes advantage of your generated plants from task 2, possibly even using their structure for something specific to the game’s mechanics.
- Make a game that takes advantage of the generated text from task 1, possible even using input from the player to help generate the text.

[2 points]
[Total: 6 points]