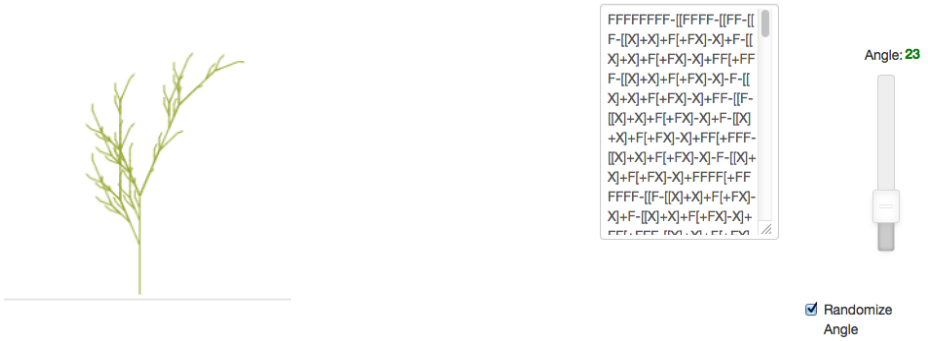
FIT3094 AI, ALife and Virtual Environments, by Alan Dorin.

Practical sheet: L-System plants

In this laboratory session you will implement a simple L-system parser that generates a string from axiom and production rules that you will invent. The string that is emitted by your software can be copied-and-pasted into a graphical interpreter that turns it into instructions for a "turtle" to produce a picture of your own virtual plants!



Before you begin this exercise, you must have prepared by reading through the iBook, *Biological Bits*, section 4.2 Lindenmayer Systems and reviewed the lecture materials on *Growing Realistic Plants and Forests*[1]. The practical exercise builds upon this background knowledge.

The widget for accepting L-system strings is provided online for you[2]. It accepts characters from an alphabet and interprets them visually as follows.

L-system alphabet and turtle graphics interpretation

C	Draw a circle
F	Draw a line forwards
X	Don't draw anything (see question 6 below)
+	Turn right by a specified angle
-	Turn left by a specified angle
[	Push the current turtle position onto the stack.
]	Pop the most recently saved turtle position off the stack.

Open the widget in your web browser and copy-and-paste the sample strings from the Appendix of this document into its text window to try it out. Your software, described below, will generate strings like these by applying your own production rules.

Notes:

- The widget is not designed as a text-editor. It is far easier to generate or type strings elsewhere and paste them into the widget window than it is to try to type them into the widget directly.
- If the widget fails to draw the string you have loaded, try to reload the web-page and re-paste your text if necessary.
- The widget will only draw strings with matching [and]s.

---

[1] Plants lecture: http://www.csse.monash.edu.au/~cema/courses/FIT3094/lecturePDFs/lecture5b_plants.pdf

[2] L-System graphical (turtle) interpreter: http://marvl.infotech.monash.edu/~sakie1/LSystems/draw.html

FIT3094 AI, ALife and Virtual Environments, by Alan Dorin.

1) Design a class to store an L-system string. The C++ class `std::string` should be sufficient as a data-member type for your class.

2) Design a class to store a single Production rule. It will need to include a data member for the predecessor module and a data member for its successor. It will need a member function that determines whether or not a supplied character matches its predecessor and returns a boolean value accordingly. It will need another function that returns its successor when requested.

3) Design a class to implement an L-system. It will need to store an axiom (the initial L-system string) and a set of production rules. It will need a method to apply the production rules to the axiom and then to the sequence of strings generated up to a specified number of iterations. The class will need to save the results of these rule applications in an L-system string data-member At least one string will be needed to store the current state of the string. More will be needed if you want to save all intermediate strings as they are generated after each time step. The class will need a method to print the resulting strings out to the terminal in plain text.

4) Implement the classes you have designed (1-3) and try these samples to test your code.

Axiom: F
Production rule: F → FF-[-F+F+F]+[+F-F-F]

Here is another sample called a *Koch Curve* (use a branching angle of 60 degrees to display this one).

Axiom: F
Production rule: F → F+F- -F+F

Use these rules to generate strings and copy-and-paste the output into the turtle widget.

5) Invent your own L-system rules, try using the C character as well as the F and see what you can come up with.

6) Extension. The online interpreter also accepts a character X as part of its alphabet, but it doesn't draw anything when it receives an X. The X can be used in a grammar all the same, for instance like this:

Axiom: X
Production rules: F → FF
X → F-[[X]+X]+F[+FX]-X for depth < 4
X → F-[[X]+X]+F[+FC]-X for depth >= 4

See what you can invent using F, X and C. Consider implementing a delay mechanism where one rule is applied up to a certain number of replacements, and then a different rule comes into play.

Appendix 1: Sample L-system strings.

Copy-and-paste these (one at a time) into the widget's text box to see how it behaves.

```
FF-[-F+F+F]+[+F-F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-
[-F+F+F]+[+F-F-F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]FF-[-
F+F+F]+[+F-F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-[-F+F
+F]+[+F-F-F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]-[-FF-[-F
+F+F]+[+F-F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-[-F+F
+F]+[+F-F-F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]+FF-[-F+F
+F]+[+F-F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+
[+F-F-F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]+FF-[-F+F+F]+
[+F-F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-
F-F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]]+[+FF-[-F+F+F]+
[+F-F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-
F-F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]-FF-[-F+F+F]+[+F-
F-F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-
F]]+[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]-FF-[-F+F+F]+[+F-F-
F]FF-[-F+F+F]+[+F-F-F]-[-FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]+FF-[-F+F+F]+[+F-F-F]]+
[+FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]-FF-[-F+F+F]+[+F-F-F]]]
```

```
FFFFFFFF-[[FFFF-[[FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-
F-[[X]+X]+F[+FC]-X]+FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-
X]-F-[[X]+X]+F[+FC]-X]+FFFF[+FFFFFF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-
[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]-FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]
+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]+FFFF-[[FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]
+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]+FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]
+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]+FFFF[+FFFFFF-[[F-[[X]+X]
+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]-FF-[[F-[[X]
+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]
+FFFFFFFF[+FFFFFFFFFFFF-[[FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]+X]
+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]+FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]
+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]+FFFF[+FFFFFF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]
+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]-FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-
X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]-FFFF-[[FF-[[F-[[X]+X]+F[+FC]-X]+F-[[X]
+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]+FF-[[F-[[X]+X]+F[+FC]-X]+F-
[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]+FFFF[+FFFFFF-[[F-[[X]+X]
+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X]-FF-[[F-[[X]
+X]+F[+FC]-X]+F-[[X]+X]+F[+FC]-X]+FF[+FFF-[[X]+X]+F[+FC]-X]-F-[[X]+X]+F[+FC]-X
```