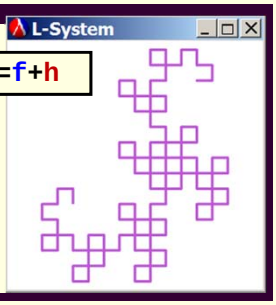
CS 152L
Computer Programming Fundamentals
Lindenmayer Systems



Instructor:
Joel Castellanos
e-mail: joel@unm.edu
Web: http://cs.unm.edu/~joel/

11/28/2017

Context-Free Grammars - Definition

A Context-Free Grammar consists of 4 parts:

A set of Variables: Symbols that can be replaced by production rules.

A set of Terminals**:** Symbols that do not appear on the predecessor side of any of the system's production rules.

An Axiom**:** A string composed of some number of variables and/or constants. The axiom is the initial state of the system.

A set of Production Rules: Which define the way variables can be replaced with combinations of terminals and other variables.

- A production consists of two strings: the predecessor and the successor.
- Being "Context-Free" requires that every predecessor be exactly one variable.

2

Example:
Nondeterministic Context-Free Grammar

1. **Axiom:** S
2. **Production Rules:** {S→aSb, S→aSXb, S→, X→b}
3. **Set of Variables:** {S, X}
4. **Set of Terminals:** {a, b}

Example Productions:

S
aSb
aaSbb
aaaSbbb
aaaabbbb

S
aSb
aaSXbb
aaaSbbbb
aaaaSXbbbbb
aaaabbbbbb

3

Context-Free Language Definitions

- A ***language*** is a set of strings.
- A language L, is said to be a ***context-free language*** (CFL) if there exists a CFG, G, that recognizes the language L.
- A ***grammar recognizes a language*** if and only if: Given a string, the grammar can determine whether that string is a member of the language.
- A ***Deterministic Context-Free Language*** (DCFL) is one in which each variable appears in the predecessor of exactly one Production Rule.

4

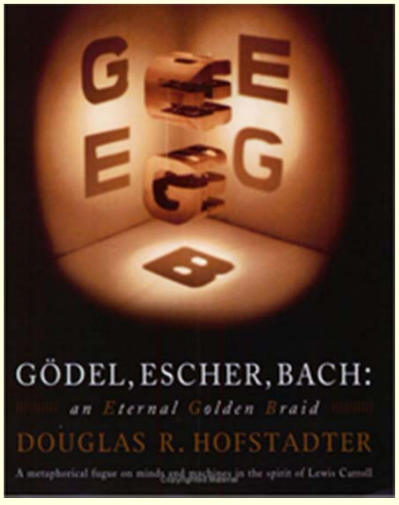
Simple Context Free Grammars

Which are deterministic?

1. $S \rightarrow abS$ $\quad S \rightarrow ab$
2. $S \rightarrow SS$ $\quad S \rightarrow ab$
3. $S \rightarrow aX$ $\quad X \rightarrow bS$ $\quad X \rightarrow a$
4. $S \rightarrow aX$ $\quad X \rightarrow bS$ $\quad X \rightarrow ab$
5. $S \rightarrow aX$ $\quad X \rightarrow bS$ $\quad S \rightarrow ab$
6. $S \rightarrow aX$ $\quad X \rightarrow bS$
7. $S \rightarrow aX$ $\quad X \rightarrow aXY$ $\quad Y \rightarrow ab$

5

CS Great Book: Gödel, Escher Bach



By
Douglas R. Hofstadter.

Pulitzer Prize winning, metaphorical Fugue on Minds and Machines in the spirit of Lewis Carroll.

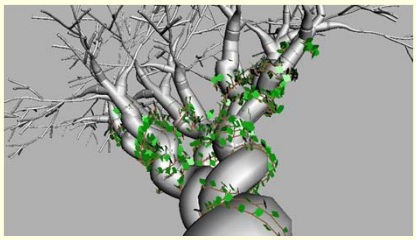
6

Lindenmayer Systems

A ***Lindenmayer system*** is a deterministic, context-free grammar where some of the symbols in the grammar's alphabet have graphical interpretations.

Lindenmayer systems can produce fractal patterns.

Lindenmayer systems are most famously used to model the growth processes of plant development.



7

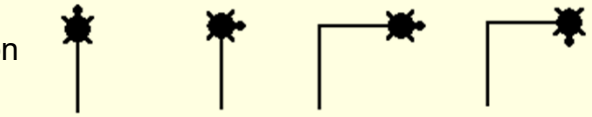
Character Draw Commands

h	Draw a straight line segment 5 pixels long in the current heading.
f	Draw a straight line segment 5 pixels long in the current heading.
g	Move (without drawing) 5 pixels in the current heading.
+	Turn the handing clockwise 90°
-	Turn the handing counter-clockwise 90°
K	Change the turtle color to BLACK
R	Change the turtle color to RED
G	Change the turtle color to GREEN
B	Change the turtle color to BLUE
C	Change the turtle color to CYAN
O	Change the turtle color to ORANGE

8

Turtle Graphics

- An imaginary ***turtle*** moves and draws with commands that are relative to its own position, such as:
  1. "Move forward 10 spaces" and
  2. "Turn left 90 degrees".
- The Turtle moves on a 2D surface and carries a pen.
- The state of the turtle has three attributes:
  1. Position
  2. Orientation
  3. Color



9

Lindenmayer System Lab

Given: User inputs a String where each character is interoperated as a draw command. All turns are right angles. Program checks for bad user input.

Lab 8: User inputs an axiom, a set of rules, and the number of times to apply those rules. The program uses this input to generate a string. Each character of the string is interoperated as a draw command as in lab 4.

Lab 8 - extra credit: The user also inputs an angle from 0 to 360 degrees. All `'+'` and `'-'` commands turn right or left by that angle. Also, the image auto scales to fit the window and the window can be resized.

10

Input Format Example #1

```
String axiom = "f"
String rules = "f=f-h; h=f+h"
int Generation = 4
```

Use JavaFX to ask user for each of these inputs.

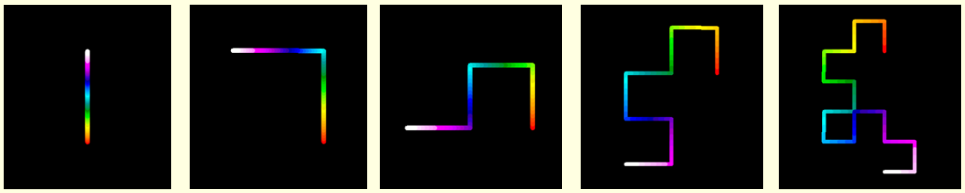
How quickly does the string grow?

generation 1: f-h

generation 2: f-h - f+h

generation 3: f-h - f+h - f-h + f+h

generation 4: f-h - f+h - f-h + f+h - f-h - f+h + f-h + f+h



Development of Given Program

- Create JavaFX window
- Add input field for String and "draw" button.
- Check input String for errors *as the user types*.
- When String is in error, display background in red, an error message and "draw" button is inactive.
- When String is ready to draw, draw button becomes active.
- Clicking draw, draws the string.

12

Input Format Example #2

```
String axiom = "f"
String rules = "f  = f -f-"
int Generation = 3
```

Ignore All Spaces in Input

What does this look like after 20 generations?

generation 1: f-f-

generation 2: f-f--f-f--

generation 3: f-f- - f-f- -- f-f- -f-f- --

Input Format Example #3

```
String axiom = "x"
String rules = "x=f+f+f+f+;f=hh-hhx--hh+h--"
int Generation = 4
```

x

f+f+f+f+

hh-hhx--hh+h--+hh-hhx--hh+h--+hh-hhx--hh+h--
+hh-hhx--hh+h--+

hh-hhf+f+f+f+.....

Requirements: Axiom Input

- TextField for user to enter axiom
- Exit program if user closes window.
- If user enters invalid input. Display error message and disable draw button.
- Valid input for axiom is a String consisting of one or more of the following characters:

  `'+'` `'-'`

  Any lowercase letter 'a' through 'z'.

  Any uppercase letter 'A' through 'Z'.

15

Requirements: Rules Input

- TextField for user to enter a set of Rules.
- If user enters invalid input. Display error message and disable draw button.
- Remove all spaces and tabs from the input rule String.
- Each rule must be delimited by a `';'`
- Each rule must have the form a=x where:

  a is any single upper or lower case letter

  x is a String consisting of one or more letters, `'+'` or `'-'`.
- No two rules may have the same character on the left of the `'='`.

16

Requirements: Generation Input

- TextField for user to enter a number of generations.
- If user enters invalid input. Display error message and disable draw button.
- The number of generations must be non-negative integer.
- The number of generations must be ≤ 25.
- If the number of generations is 0, then it means to just draw the axiom.

17

Lab Clarifications

- One Window: All input, display, drawing error messages and everything else is displayed in ONE WINDOW. All these things use the same window.
- One input textfield for axiom, one for set of rules and a third for number of generations.
- You are supporting a deterministic grammer. Thus, there are NO RULES that have the SAME variable
- You are supporting a context free gramer. Thus, every rule can only have one symbol on the left of the =.

18

Requirements: `buildFractal`

- The LSystem class must implement the method:

```
public static char[] buildFractal(
  String axiom, String rules, int generation)
```

Your program will be tested by a grading robot that calls this method. If you do not implement the interface ***exactly*** as shown, you WILL GET ZERO points for **`buildFractal`**.

- **`buildFractal`** must return a char[] containing no spaces that results from the given number of generations.
- If a given set of inputs results in a string of more than 100 million characters, your program ***may*** display an error message and exit.
- Since **`axiom`**, **`rules`** and **`generation`** are ***parameters*** to **`buildFractal`**, they must input those BEFORE **`buildFractal`** is called.

19

Overview of `buildFractal`

- The input to **`buildFractal`**, is:

```
String axiom, String rules, int generation
```

- axiom and rules should already have all spaces removed and should already have been verified to be valid. Also, there should be two class instance fields, **`char[] srcArray`** and **`char[] targetArray`** each with 100 million elements. Of course, you can name these arrays whatever you want.
- This function needs to:
  1) Break **`rules`** into an array of rules, **`String ruleList[]`**.
  2) Copy the axiom into **`targetArray`**.
  3) For each generation greater than 0, the current **`targetArray`** must become the **`srcArray`**, and **`targetArray`** must be updated by applying all rules to the current **`srcArray`**.
  4) Return the final **`targetArray`**.

20

Break rules into ruleList[]

- Breaking **`rules`** into an array of rules, **`ruleList[]`** is the first step in **`buildFractal`**.
- How to do this was shown in class using the .split() method of the String class:

```
String[] ruleList = rules.split(";");
```

21

Copy the axiom into `targetArray[]`

The second step in **`buildFractal`** should be to Copy the axiom into **`targetArray`**. To do this:

1. Loop through each character in the axiom.
2. Copy each character into **`targetArray`** starting with element 0.
3. Set class field targetEnd to one less than the length of the length of axiom.

```
for (int i=0; i<axiom.length(); i++)
{
    char c = axiom.charAt(i);
    targetArray[i] = c;
}
targetEnd = axiom.length()-1;
```

If the number of generations is 0, then the function is done. Just return **`targetArray[i]`**

22

Set `targetArray[]` to next generation

The third step in **`buildFractal`** is to update **`targetArray[]`** so that it is the next generation:

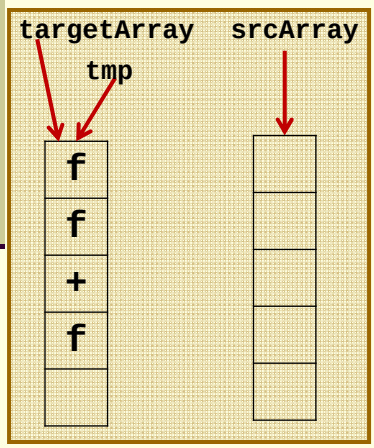
1) Swap the pointers **`srcArray[]`** and **`targetArray[]`**
2) Set **`srcEnd = targetEnd`**
3) Set **`targetEnd`** to -1 (meaning it is empty).
4) Loop through each character in **`srcArray[]`** from element 0 through element **`srcEnd`**.
5) For each character in **`srcArray[]`**, loop through all the rules in **`ruleList[]`**. If the character is the same as the first character of a rule, then copy all the characters after the = from that rule into **`targetArray[]`** (adding 1 to **`targetEnd`** each time you copy a character). If the character does not match any rules, then copy that character and add 1 to **`targetEnd`**.

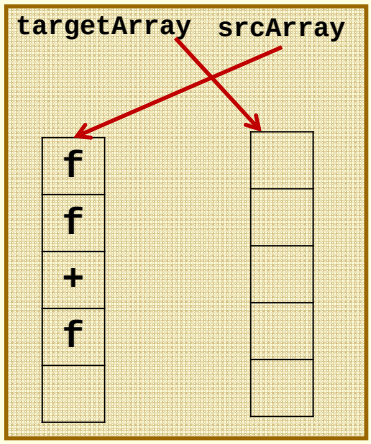
23

Swap pointers `srcArray[]` & `targetArray[]`

```
char[] tmp  = targetArray;
targetArray = srcArray;
srcArray    = tmp;
```

This does NOT create a new array.
It just creates a new ***pointer*** to an array.





24

Finishing `buildFractal`

- For each generation, repeat the steps of:
  1) Swapping the **`targetArray`** with **`srcArray`** so that the last generation's target is the source array for the next generation.
  2) set **`targetEnd`** back to -1.
  3) loop through each character in srcArray and if it is the first character of a rule, copy everything right of the '=' character in the rule to the end of **`targetArray`** (adding 1 to **`targetEnd`** for each character copied). If the character is not in the first character of any rule, then just copy the character into **`targetArray`** (adding 1 to **`targetEnd`**).
- When you have repeated this for the specified number of generations, return **`targetArray`**.

25

Requirements: Visual Display

- For each valid input, your program must display the resulting graphical interpretation of the resulting String.
- When drawing, any character that is not one of the graphics commands is ignored. For example the strings **`"f+f"`** and **`"fzzzzzzz+ZZZZf"`** have the same graphical interpretation since all the **`'z'`** and **`'Z'`** must be ignored.
- The canvas must be fully visible on the window, must have a different background color than the window background and must be 600x600 pixels.

26

Grading Rubric for Part 1(20 points total)

[3 point] Your program creates three input boxes labeled axiom, rules and number of generations.

[2 points] When the user enters an axiom with a character that is not a letter, then the text color of the axiom becomes red and an error label displays in the window saying "Illegal Character".

[8 points] When the user enters a rule set that is not legal, then the text color of the rule field becomes red and an error label displays in the window saying an appropriate message. For example, missing '=', illegal symbol, two or more rules with the same symbol on the left of the '=', a rule with more than one symbol on the left of the equal.

[5 point] When the user enters a number of generations that is illegal, the rule field becomes red and an error label displays in the window saying an appropriate message. For example, a character that is not a digit or a number that is too large.

[2 point] When none of the three input fields has an error, then the draw button is enabled, otherwise it is disabled.

27

Useful String and Character Methods

```
java.lang.Character
   public static boolean isDigit(char ch)
   public static boolean isLetter(char ch)

java.lang.String
   public char charAt(int index)
   public String[] split(String regex)
   public String trim()

java.lang.Integer
   public static Integer valueOf(String s)
```

28

Grading Rubric for Part 2 (35 points total)

[6 points] **`buildFractal`** works as specified for example #1 with generations 0 through 10.

[6 points] **`buildFractal`** works as specified for example #2 with generations 0 through 10.

[6 points] **`buildFractal`** works as specified for example #3 with generations 0 through 10.

[6 points] **`buildFractal`** works as specified for unknown #1.

[6 points] **`buildFractal`** works as specified for unknown #2.

[5 points] The visual display works for each of the above test cases. This is a free 5 points since, except for adding more colors, it is mostly unchanged from the given code

29

Grading Rubric Penalties

[-5 points]: Code does not adhere to the hallowed CS-152 coding standard: http://www.cs.unm.edu/~joel/cs152/CS-152-Lecture-05-CodeStandards.pdf. This includes indenting, class, method, and in-line comments, removing all warnings, etc. Note: all 5 points are lost if any one of the standards is severely broken. For example:

- Very Poor comments: -5.
- Poor comments: -1 through -3.
- Many indenting errors: -5.
- A few indenting errors: -1 through -3.
- Multiple breaking of naming convention: -5. For example, variables that start with an uppercase letter or fields declared final that are not all uppercase.
- Multiple warnings (other than the ignorable: Serializable...).

30

Extra Credit [+10]: Fit to Screen

As specified, generation 10 of example 1 (named the *Heighway Dragon Curve*) will fit within the 600×600 pixel drawing area.

Level 11 of the dragon curve will not fit☹

Equally unpleasant is the visual size of the first few generations☹

For extra credit, upgrade the program so that the largest dimension of each drawing, regardless of system and generation, exactly fits within 10 pixels less than the available space.

You must avoid distortion by scaling both x and y by the same amount.

Your image must be centered within the display.

31

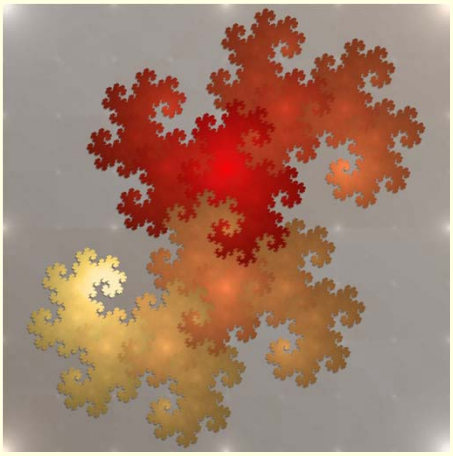
Extra Credit [+5]: Resizing Screen

- In the given version, when the user resizes the screen, the canvas and text stay the same.
- This extra credit option by itself has no effect on the drawing image size. When the user makes the window smaller/larger, a smaller/larger picture will fit.
- This extra credit option may be combined with the fit to sereen option.

32

Extra Credit [+10]: Metallic Coloring

By level 20, if drawn with each f and g no longer than one pixel, the dragon curve appears to be space filling. At this density, metallic coloring effects, if used, will be visible.

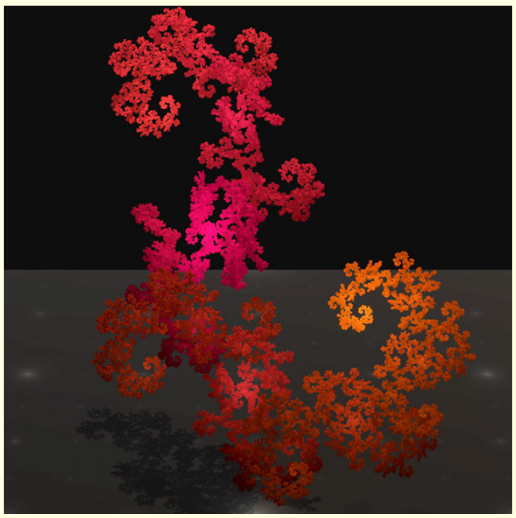


1) Load a 600x600 image of a brushed steel texture into a onto the screen
2) Merge the image of your L-system with the steel texture. This can be done by looping through each pixel of the two images, and for each pixel set the new L-system red, green and blue values as a weighted average of the r, g and b of the two images:

$$r' = \frac{w_1 * r_{lsystem} + w_2 * r_{texure}}{w_1 + w_2}$$

33

Extra Credit [+30]: Do it in 3D



Rendering by a Solkoll, a Swedish Wikipedian

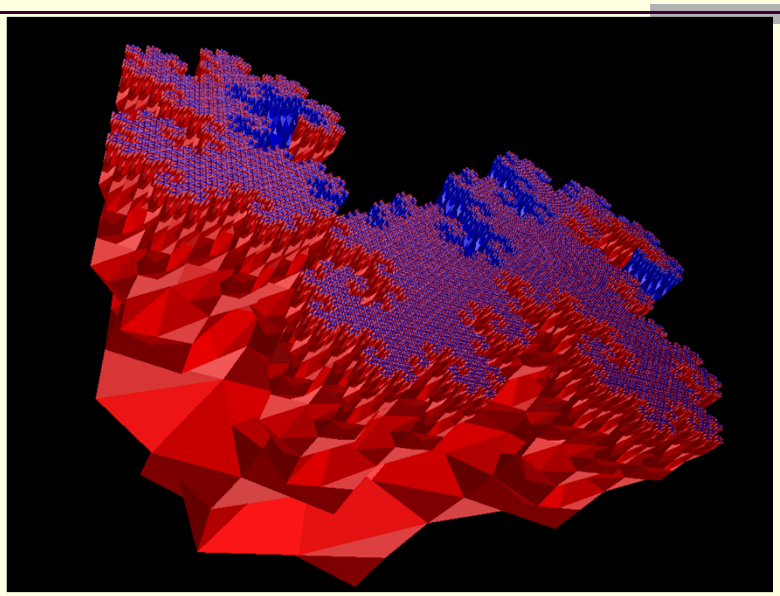
JavaFX supports 3D graphics rendering with

Ray casting, shadows, specular and defuse reflection...

Have fun and make something beautiful.

34

Dragon Curve Rendering by Geoffrey Irving



35

Implementation Suggestions

- Whenever possible, software engineers should try to remove all implementation specifications from the project requirements.
- Ideally, requirements be limited to use cases and performance.
- This is not always possible. A contractor (or teacher) may require use of a particular language (i.e. Java in CS-152) or may require a particular algorithm.

The next few slides show implementation suggestions and spoilers. They are not requirements.

36

Implementation Suggestion: Rules:

- Your program inputs the rules as a single string. This is awk to work with becasue where in the string is the left part and the right part
- so we will make a method that breakes the single string of rules into an array of rules where each element of the array is a single rule:

- String[] ruleList = cmdString.split(";");

37

Helper method: String.trim()

```
//String.trim() removes whitespace from the beginning and end of a
//    string. This includes spaces, tabs, return characters and all
//    ASCII control characters.
public class HelloWorld
{ public static void main(String[] args)
  { String str = "  white   space   " + '\t' + " ";
    System.out.println("<" + str + ">");
    str.trim();
    System.out.println("<" + str + ">");
    str = str.trim();
    System.out.println("<" + str + ">");
  }
}
```

Output: - what is going on?

```
<  white   space         >
<  white   space         >
<white   space>
```

38

Helper method: removeWhitespace

```
//First, uses String.trim() to remove all whitespace from the front and
//end of the given str.
//The String str then references the new, trimmed string.
//Then builds a new String, str2, one character at a time from str.
//Each non-blank, non-tab character from str is added to str2.
//The method returns a reference to str2.


private static String removeWhitespace(String str)
{ String str2 = "";
  for (int i=0; i<str.length(); i++)
  {
    char c = str.charAt(i);
    if (c != ' ' && c != '\t') str2 += c;
  }
  return str2;
}
```

39

Testing: removeWhitespace

```
public static void main(String[] args)
{ String x = "  I contain some   whitespace        ";
  String y = "abcdefghijklmnop";
  //The String object referenced by x is passed into
  //    removeWhitespace(x).

  //After the method returns, x is assigned the reference to a new
  //    object created by removeWhitespace(x).

  //The String object that x used to reference is no longer referenced
  //    therefore, Java marks it for garbage collection.

  x = removeWhitespace(x);
  y = removeWhitespace(y);

  //By surrounding the Strings with <> symbols, we can
  //    see if there are leading or trailing spaces.
  System.out.println("<" + x + ">");
  System.out.println("<" + y + ">");
}
```

40

Other Suggested Helper Methods

```
private String inputAxiom()
//Display dialog, check for errors, repeat until good.

private String inputRules()
//Display dialog, check for errors, repeat until good.

private int inputGeneration()
//Display dialog, check for errors, repeat until good.
```

Separating code that can be understood as a ***functional unit*** into a method helps organize your code.

41

Scaling and Centering to Fit the Screen in *x* (horizontal) direction

- Initialize **minX**, **maxX** each to the center of the canvas, 300.0.
- Loop through all the draw commands of drawing the Original Fractal starting at (300,300) and a draw distance of 5 pixels. However, do not actually draw anything, just calculate each change to x. Each time x changes:

```
if (x<minX) minX=x;
if (x>maxX) maxX=x;
```

- After the loop is done, calculate **scaleX** and **shiftX**:

```
double screenMinX = 0., sreenMaxX = 600.
screenRange       = sreenMaxX - screenMin
fractalRange      = maxX - minX
double lineLength = 5.0
scaleX            = screenRange / fractalRange
lineLength        = lineLength * scaleX
shiftX            = -minX*scaleX
```

42