Computer Programming Fundamentals

CS 152
Professor: Leah Buechley
TAs: Melody Horn, Noah Garcia, Andrew Geyko, Juan Ormaza
Time: MWF 10:00-10:50am
https://handandmachine.cs.unm.edu/classes/CS152_Fall2021/

ASSIGNMENT 5

Due Monday 11/8 at midnight

Post questions on Piazza

WHERE WE ARE

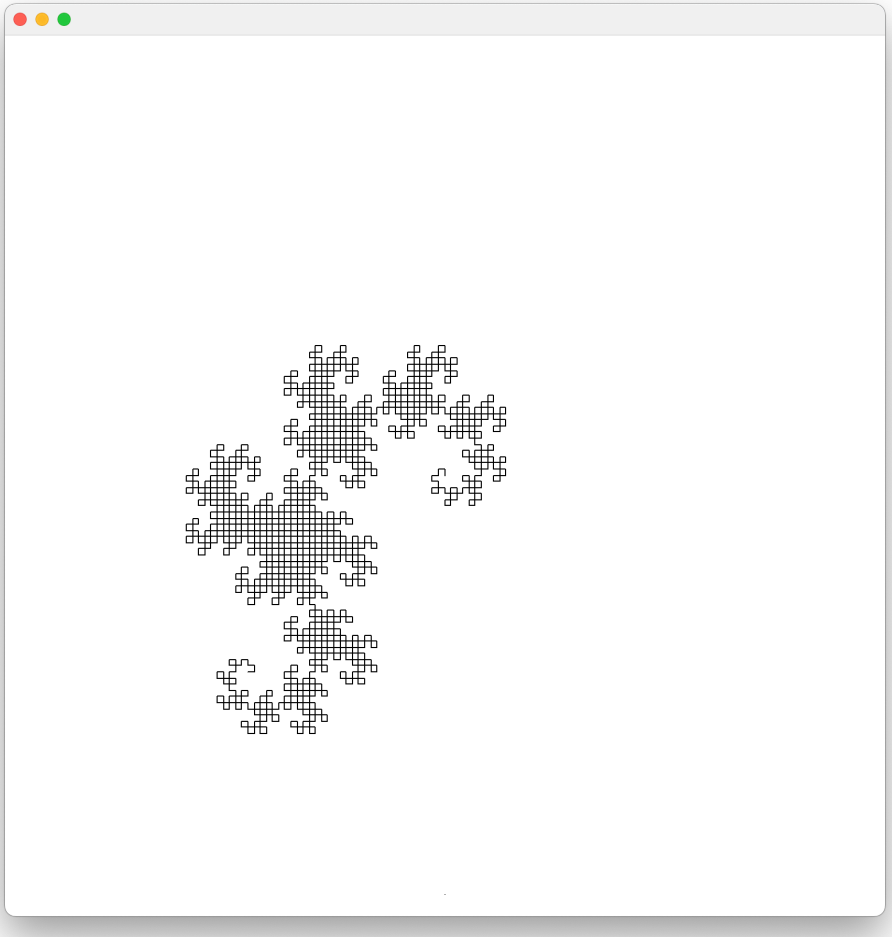
CLASSES

```java
public class LSystem {
    String startingWord, computedWord;
    String[][] rule = {{"F", "F-F++F-F"}};
    int iterations;

    LSystem () {
        startingWord = "F";
        computedWord = startingWord;
        iterations = 0;
        System.out.println("iteration " +iterations +": " +computedWord);
    }

    LSystem (String startingWord, String[][] rule) {
        this.startingWord = startingWord;
        computedWord = startingWord;
        this.rule = rule;
        iterations = 0;
        System.out.println("iteration " +iterations +": " +computedWord);
    }

    public static void main(String[] args) {
        LSystem l = new LSystem();
        for (int i=0; i<5; i++)
            l.iteration();
    }

    void iteration() {
        String tempWord = computedWord;
        computedWord = "";
        for (int i = 0;i<tempWord.length();i++){
            for(int j=0;j<rule.length;j++) {
                if (tempWord.charAt(i)==rule[j][0].charAt(0))
                    computedWord = computedWord + rule[j][1];
            }
        }
        iterations++;
        System.out.println("iteration " +iterations +": " +computedWord);
    }

    void draw (Turtle t, double size, double angle) {
        for (int i=0;i<computedWord.length();i++) {
            if (computedWord.charAt(i)=='F')
                t.forward(size);
            else if (computedWord.charAt(i)=='-')
                t.left(angle);
            else if (computedWord.charAt(i)=='+')
                t.right(angle);
            else if (computedWord.charAt(i)=='L')
                t.forward(size);
            else if (computedWord.charAt(i)=='R')
                t.forward(size);
        }
    }
}
```

```java
import java.awt.*;
import java.awt.event.MouseEvent;

public class LSystemVis extends BasicPanel{
    Turtle t;
    LSystem l;
    double size, angle;
    final String[][] dragon = {{"L","L-R-"},
                               {"R","+L+R"},
                               {"+","+"},
                               {"-","-"}};

    LSystemVis () {
        t = new Turtle(this);
        l = new LSystem("L",dragon);
        size = 100;
        angle = 90;
        l.draw(t,size,angle);
    }

    public static void main(String[] args) {
        LSystemVis lSystemVis = new LSystemVis();
        MyFrame myFrame = new MyFrame(lSystemVis);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        //we can draw here
        setBackground(Color.WHITE);
        t.drawPath(g);
    }

    @Override
    public void mousePressed(MouseEvent e) {
        size = size/1.3;
        t.clearTurtleHistory();
        t.penUp();
        t.setX(width/2);
        t.setY(height/2);
        t.penDown();
        l.iteration();
        l.draw(t,size,angle);
        repaint();
    }
}
```

COMPILE AND RUN

starting word = L
rule = dragon
angle = 90



questions?

BACK TO TREES

TURTLE PUSH AND POP

t.push(): remember current turtle state
t.pop(): return to last remembered state

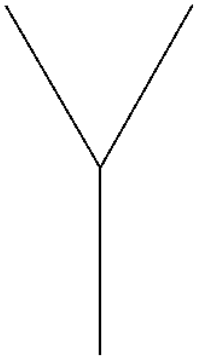
ADDING SYMBOLS & RULES
PUSH AND POP

[= push, remember current turtle state
] = pop, return to last remembered state

Turtle interpretation:
[= push
] = pop

THINKING ABOUT A TREE

using push & pop,
what is a good expression for this beginning of a tree?



F[-F][+F]

starting word = F
rules = F ⟶ F[-F][+F]

TRYING IT OUT

ADD NEW SYMBOLS TO DRAW METHOD

NEW draw METHOD

```
void draw (Turtle t, double size, double angle) {
    for (int i=0;i<computedWord.length();i++) {
        if (computedWord.charAt(i)=='F')
            t.forward(size);
        else if (computedWord.charAt(i)=='-')
            t.left(angle);
        else if (computedWord.charAt(i)=='+')
            t.right(angle);
        else if (computedWord.charAt(i)=='L')
            t.forward(size);
        else if (computedWord.charAt(i)=='R')
            t.forward(size);
        else if (computedWord.charAt(i)=='[')
            t.push();
        else if (computedWord.charAt(i)==']')
            t.pop();
    }
}
```

questions?

ADD A tree RULE

```
public class LSystemVisTest extends BasicPanel{
    Turtle t;
    LSystemTest l;
    double size, angle;
    final String[][] dragon = {{"L", "L-R-"},
            {"R", "+L+R"},
            {"+","+"},
            {"-","-"}};
    final String[][] tree = {{"F", "F[-F][+F]"},
            {"+","+"},
            {"-","-"},
            {"[","["},
            {"]","]"}};
```

CREATE A NEW L-SYSTEM WITH THIS NEW RULE

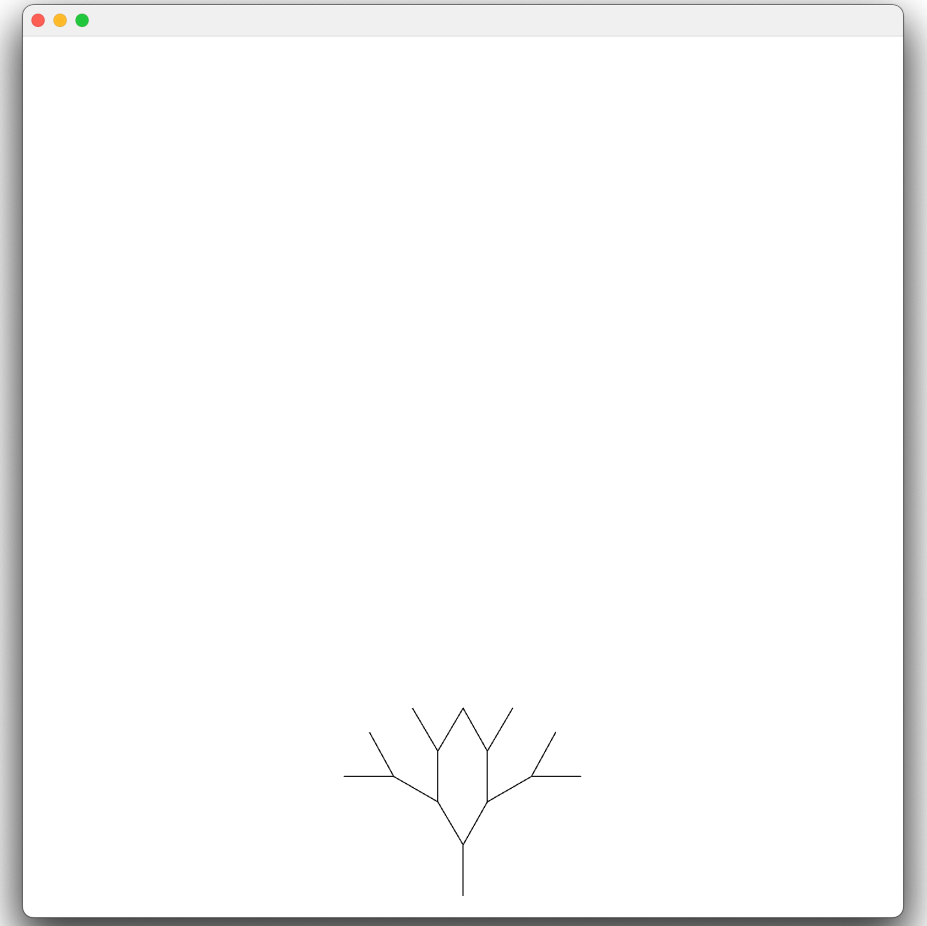
```
LSystemVis () {
    t = new Turtle(this);
    l = new LSystem("F", tree);
    size = 100;
    angle = 30;
    l.draw(t,size,angle);
}
```

starting word = F

rules = F ⟶ F[-F][+F]

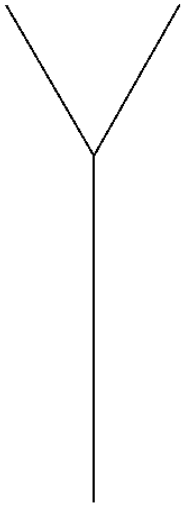
COMPILE & RUN

starting word = F
rule = tree
angle = 30



ADJUSTMENTS

what is a good expression for this (better) beginning of a tree?
trunk is taller than branches

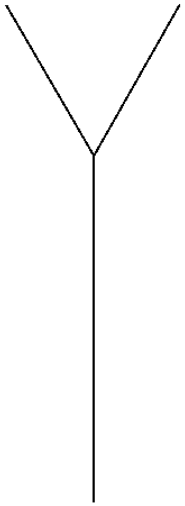


old: F[-F][+F]
new: FF[-F][+F]

ADJUSTMENTS

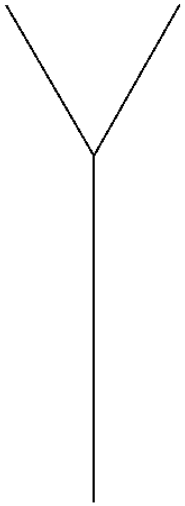
how can we write a rule that results in behavior we want?
we don't want every F to branch

old: F[-F][+F]
new: FF[-F][+F]



ADJUSTMENTS

how can we write a rule that results in behavior we want?
we don't want every F to branch
add a new symbol (X) for branching
F now for "growing" only



old: F[-F][+F]
new: FF[-F][+F]

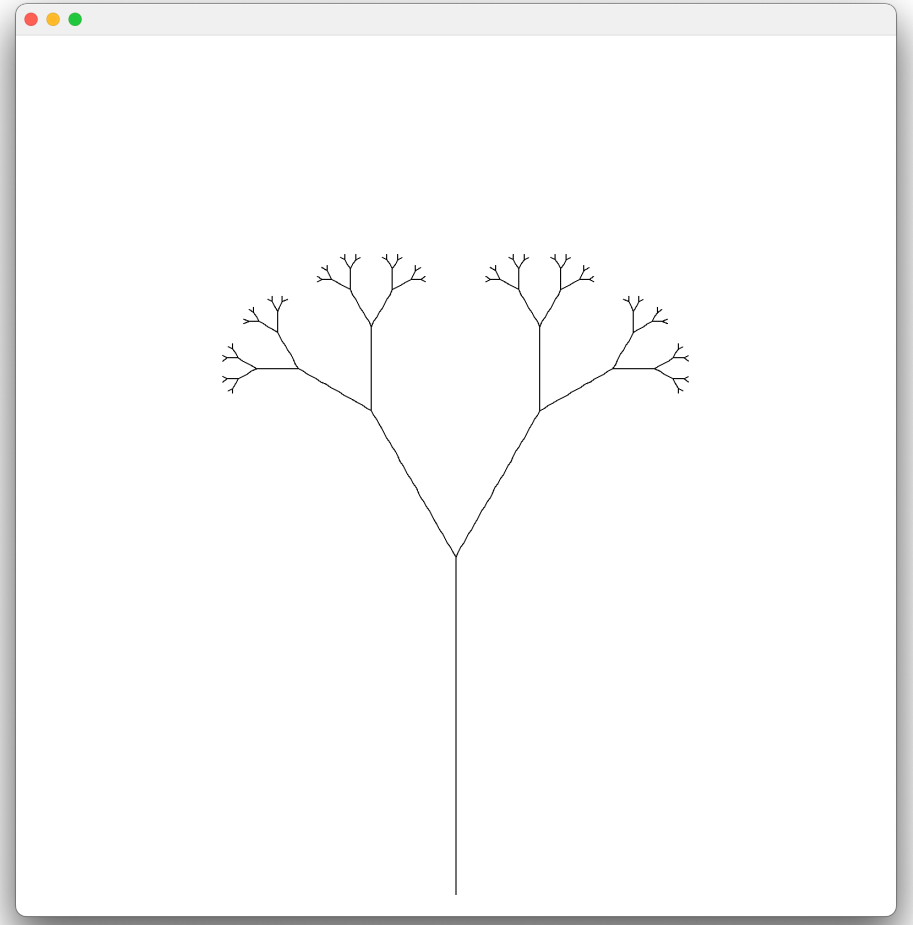
starting word = X
rules = X → F[-X][+X]
F → FF

EDIT tree RULE, ADJUST PARAMETERS

```
public class LSystemVis extends BasicPanel{
    Turtle t;
    LSystemTest l;
    double size, angle;
    final String[][] dragon = {{"L", "L-R-"},
            {"R", "+L+R"},
            {"+","+"},
            {"-","-"}};
    final String[][] tree = {{"X", "F[-X][+X]"},
            {"F","FF"},
            {"+","+"},
            {"-","-"},
            {"[","["},
            {"]","]"}};

    LSystemVisTest () {
        t = new Turtle(this);
        l = new LSystemTest("X",tree);
        size = 20;
        angle = 30;
        l.draw(t,size,angle);
    }
```

COMPILE & RUN

starting word = X
rule = tree
angle = 30
size = 30



questions?

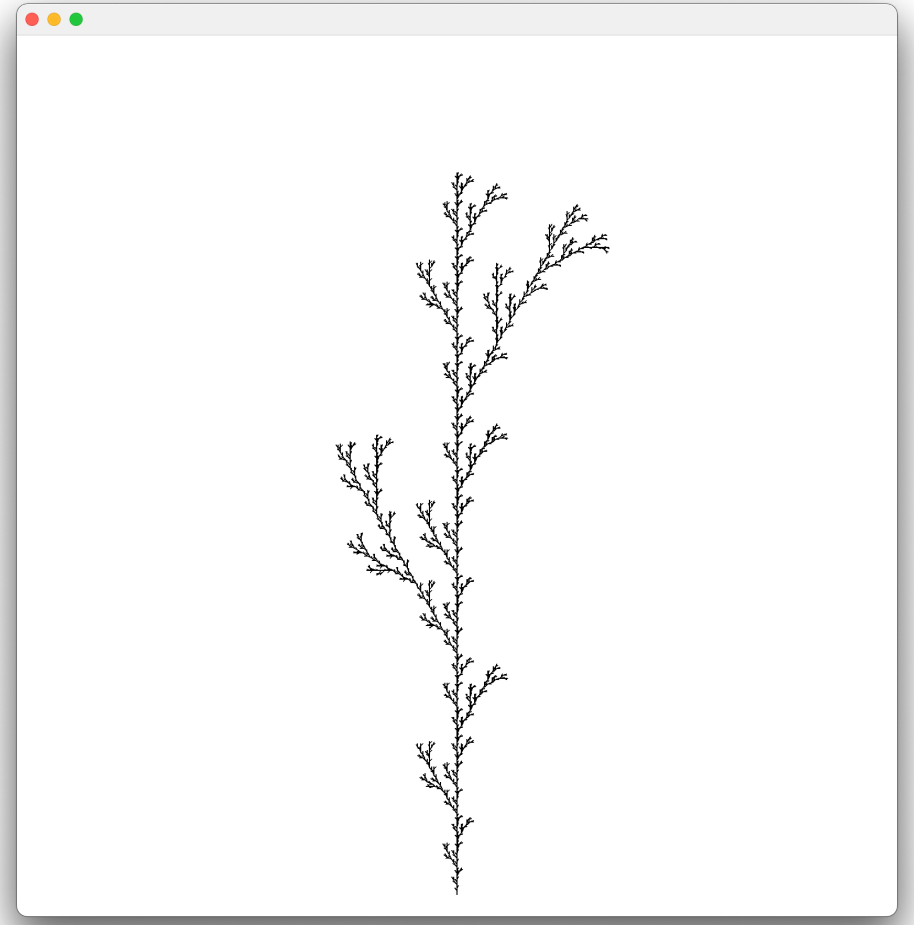
PLAYING WITH RULE & PARAMETERS

starting word = F

rules = F → F[-F]F[+F]F

angle = 30

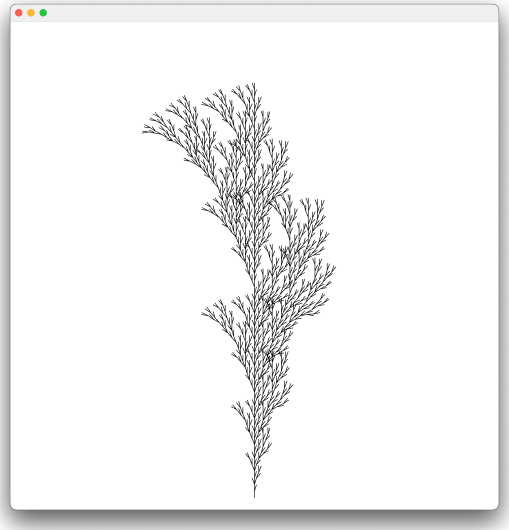
size = 10



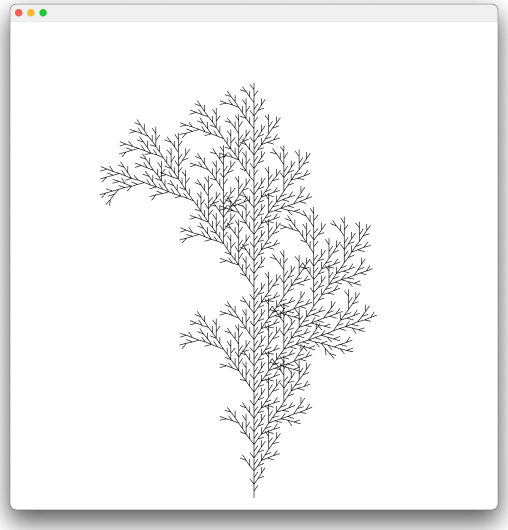
starting word = F

rules = F → F[+F]F[-F][F]

size = 40



angle = 20



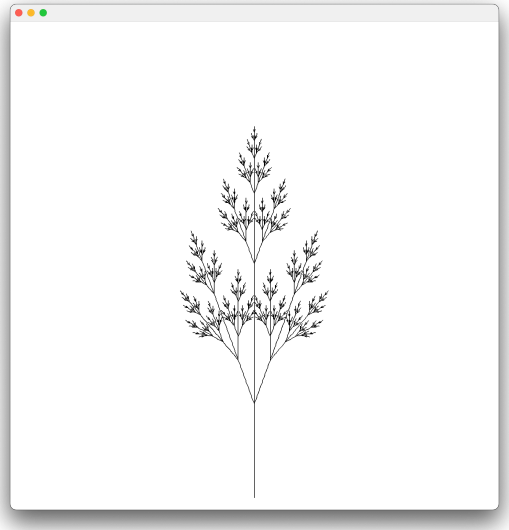
angle = 35

starting word = X

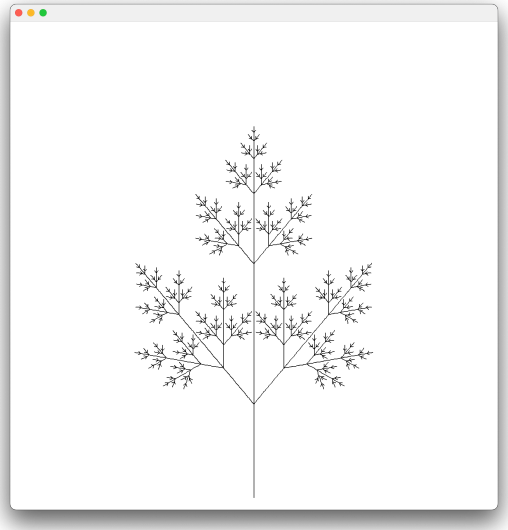
rules = X → F[+X][-X]FX

F → FF

size = 15



angle = 20



angle = 35

questions?

ADDING SOME RANDOMNESS

ADD TWO random HELPER METHODS TO LSystem CLASS

```
static double random(double lowerBound, double upperBound) {
    double result;
    result = lowerBound + Math.random()*(upperBound-lowerBound);
    return result;
}

static int randomInt(int lowerBound, int upperBound) {
    int result;
    result = lowerBound + (int)(Math.random()*(upperBound-lowerBound));
    return result;
}
```

will return numbers between lowerBound and upperBound

do numbers include lowerBound? yes

do numbers include upperBound? no

questions?

ADDING RANDOMNESS PART 1

```
void drawRandom (Turtle t, double size, double angle) {
    for (int i=0;i<computedWord.length();i++) {
        if (computedWord.charAt(i)=='F')
            t.forward(size*random(.1,1.8));
        else if (computedWord.charAt(i)=='-')
            t.left(angle*random(.5,1.5));
        else if (computedWord.charAt(i)=='+')
            t.right(angle*random(.5,1.5));
        else if (computedWord.charAt(i)=='L')
            t.forward(size);
        else if (computedWord.charAt(i)=='R')
            t.forward(size);
        else if (computedWord.charAt(i)=='[')
            t.push();
        else if (computedWord.charAt(i)==']')
            t.pop();
    }
}
```

add a new drawRandom method to LSystem class

IN LSystemVis

```
@Override
public void mousePressed(MouseEvent e) {
    size = size/1.6;
    t.clearTurtleHistory();
    t.penUp();
    //t.setX(width/2);
    //t.setY(height/2);
    t.penDown();
    l.iteration();
    l.drawRandom(t,size,angle);
    repaint();
}
```

call it in LSystemVis

RANDOMIZING LENGTH & ANGLE

starting word = X
rules = X → F[+X][-X]FX
F → FF
size = 15
angle = 30

STRUCTURE ALWAYS THE SAME

HOW CAN WE RANDOMIZE STRUCTURE? RANDOMIZING THE RULES

IN LSystemVis

```
final String[][] randomTree = {{"X", "F[+X][-X]FX", "F-[[X]+X]+F[+FX]-X", "F[-X][+X]"},
        {"F","FF", "FF", "FF"},
        {"+","+","+","+"},
        {"-","-","-","-"},
        {"[","[","[","["},
        {"]","]","]","]"}};

LSystemVisTest () {
    t = new Turtle(this);
    l = new LSystemTest("X",randomTree);
    size = 15;
    angle = 30;
    l.draw(t,size,angle);
}
```

1 2 3

add three different options for each rule

IN LSystem RANDOMLY PICK AN OPTION

```
void iterationRandom() {
    String tempWord = computedWord;
    computedWord = "";
    for (int i=0;i<tempWord.length();i++) {
        for (int j=0;j<rule.length;j++) {
            if (tempWord.charAt(i) == rule[j][0].charAt(0)) {
                computedWord = computedWord + rule[j][randomInt(0,4)];
            }
        }
    }
    iterations++;
    System.out.println("iteration " +iterations +": " +computedWord);
}
```

add a new iterationRandom method to LSystem class

IN LSystemVis

```
@Override
public void mousePressed(MouseEvent e) {
    size = size/1.3;
    t.clearTurtleHistory();
    l.iterationRandom();
    l.draw(t,size,angle);
    repaint();
}
...

@Override
public void mousePressed(MouseEvent e) {
    size = size/1.6;
    t.clearTurtleHistory();
    t.penUp();
    //t.setX(width/2);
    //t.setY(height/2);
    t.penDown();
    l.iterationRandom();
    l.draw(t,size,angle);
    repaint();
}
```

use random iteration
use non-random drawing

RANDOMIZING RULE

starting word = X
rules = randomized

size = 15
angle = 20