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

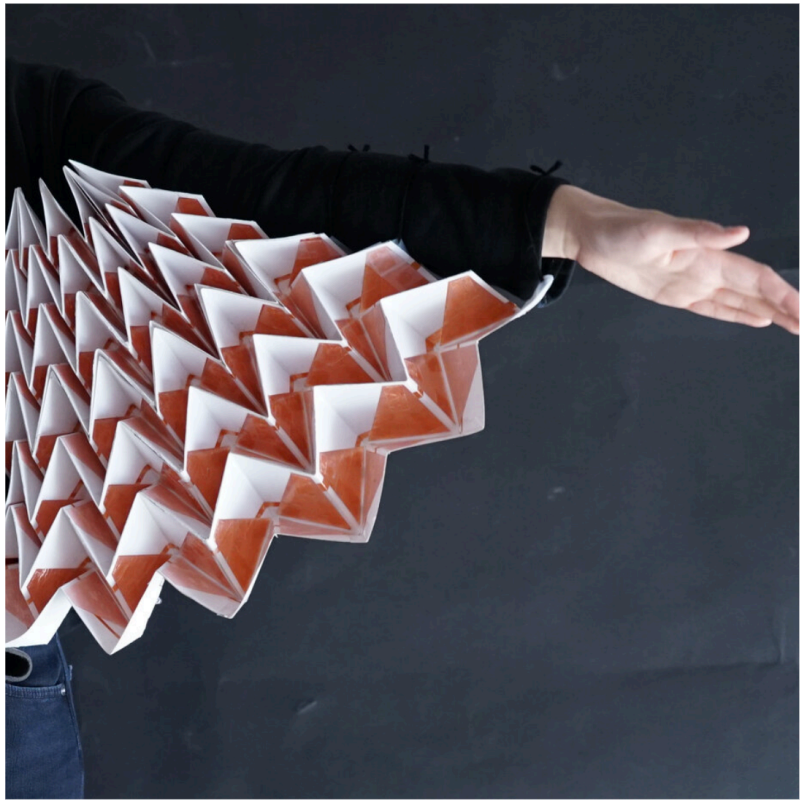
Due Monday 11/8

Start early!

email me by the end of today

if you want to work on a team

TALK TODAY: HyunJoo OH
2pm



Self-powered Paper Interfaces

Sensing Kirigami

https://www.codecraft.group/projects

questions?

OPEN IntelliJ

OPEN “Week11” PROJECT

WHERE WE LEFT OFF

ACTUALLY DRAWING A GENERATED L-SYSTEM

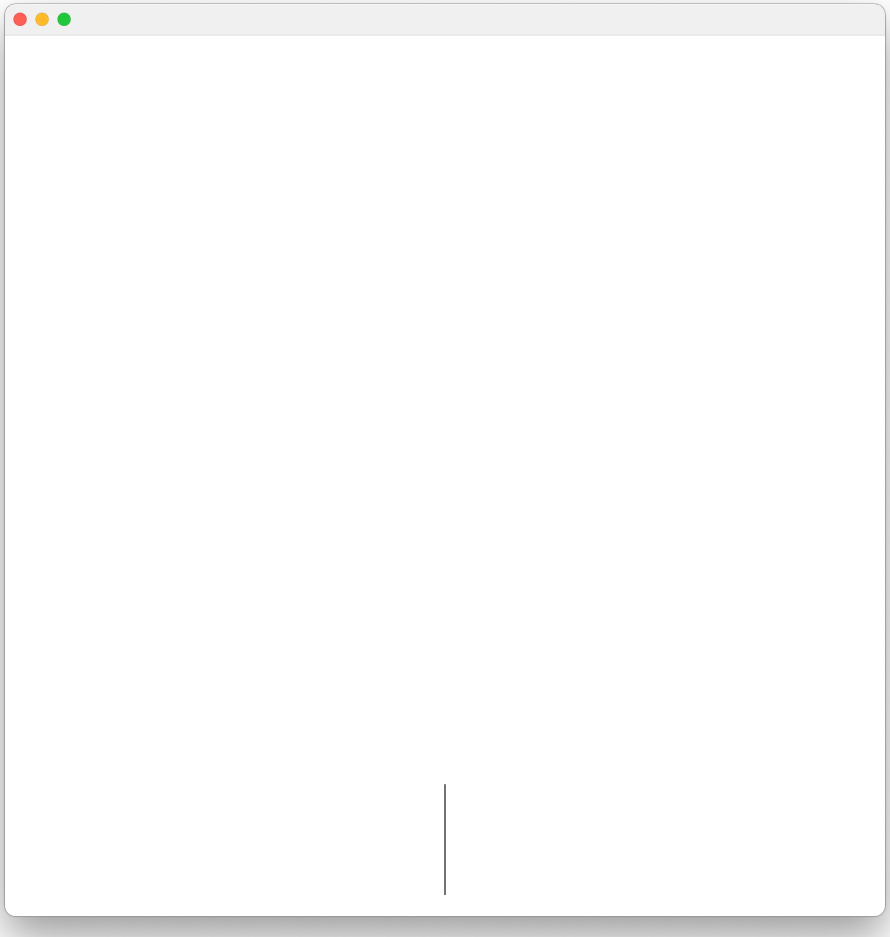
LSystemVis.java

```
public class LSystemVis extends BasicPanel {
    Turtle t;
    LSystem l;
    double size, angle;

    LSystemVis() {
        t = new Turtle(this);
        l = new LSystem();
        size = 100;
        angle = 60;
        l.draw(t,size,angle);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        setBackground(Color.WHITE);
        t.drawPath(g);
   }

}
```

COMPILE AND RUN

DRAWS STARTING WORD/ITERATION 0

```
interation 0: F
```



LET’S DRAW A NEW ITERATION EACH TIME MOUSE IS CLICKED

LSystemVis.java

```
public class LSystemVis extends BasicPanel {
    Turtle t;
    LSystem l;
    double size, angle;

    LSystemVis() {
        t = new Turtle(this);
        l = new LSystem();
        size = 100;
        angle = 60;
        l.draw(t,size,angle);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        setBackground(Color.WHITE);
        t.drawPath(g);
   }

    @Override
    public void mousePressed(MouseEvent e) {
    }
}
```

add a mousePressed method

LSystemVis.java

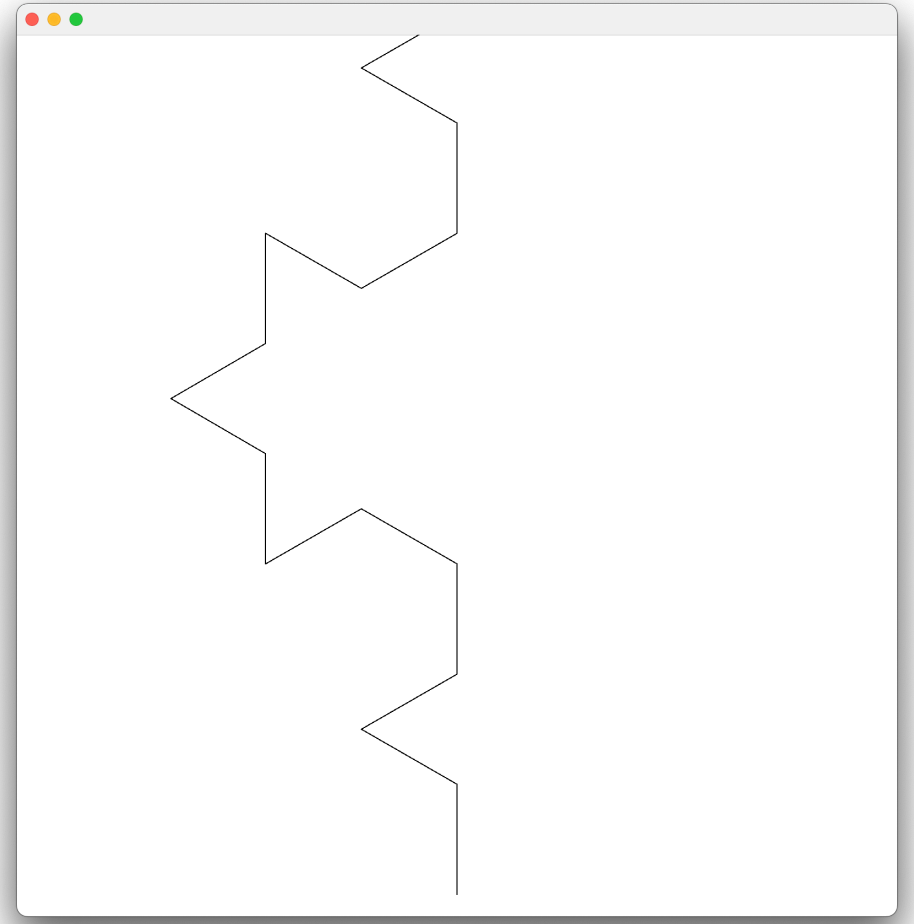
```
public class LSystemVis extends BasicPanel {
    Turtle t;
    LSystem l;
    double size, angle;

    LSystemVis() {
        t = new Turtle(this);
        l = new LSystem();
        size = 100;
        angle = 60;
        l.draw(t,size,angle);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        setBackground(Color.WHITE);
        t.drawPath(g);
   }

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

clear previous path
iterate
draw
repaint

COMPILE AND RUN

DRAWS ITERATIONS

```
interation 0: F
interation 1: F-F++F-F
interation 2: F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F
```



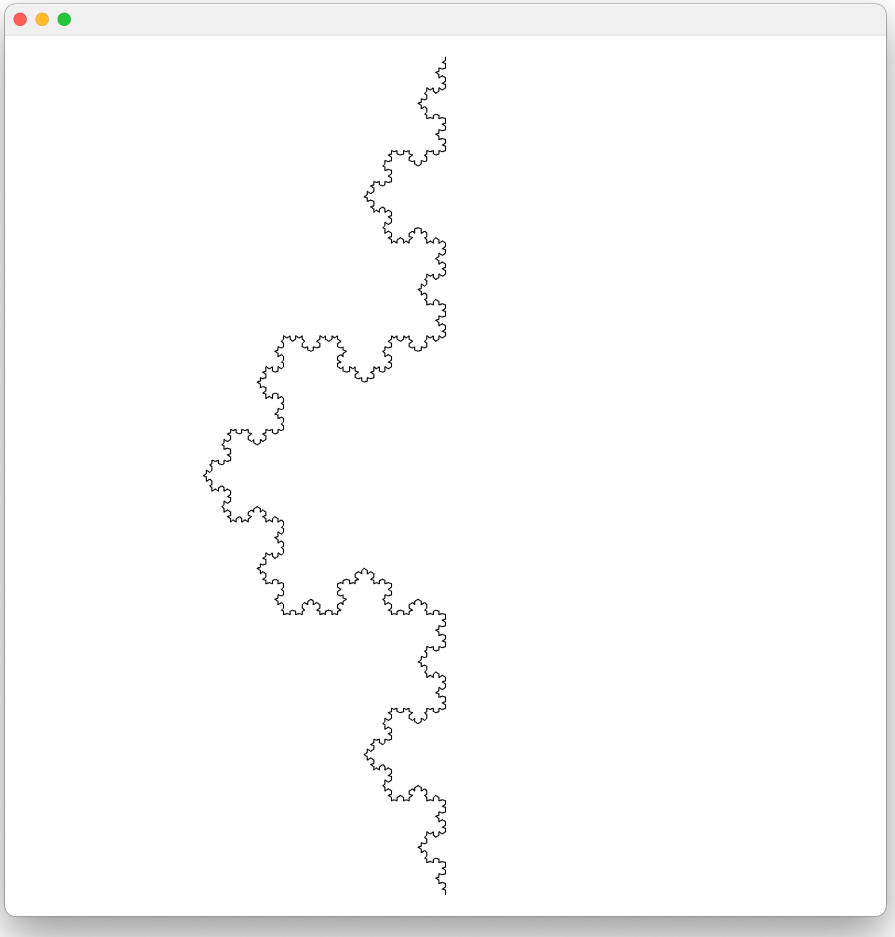
AN IMPROVEMENT

```
public void mousePressed(MouseEvent e) {
    size = size/2;
    t.clearTurtleHistory();
    l.iterate();
    l.draw(t,size,angle);
    repaint();
}
```

decrease size with each iteration

5 ITERATIONS

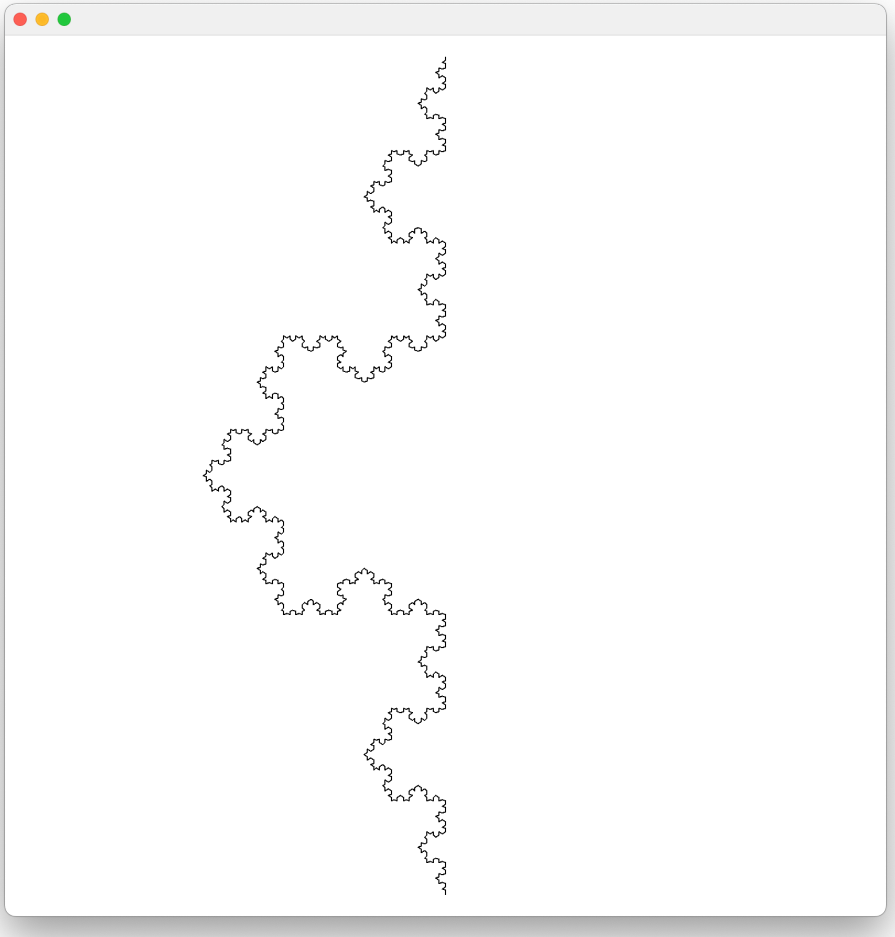
```
interation 0: F
interation 1: F-F++F-F
interation 2: F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F
interation 3: F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F
  -F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-
interation 4: F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F
  -F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-
  ++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-
  -F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F+
  ++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-
  -F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F
  -F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-
  ++F-F-F-F++F-F++F-F++F-F-F-F++F-F
interation 5: F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F
  -F++F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-
  ++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-
  -F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F+
  ++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F++F-F++F-F-F-
  -F-F-F++F-F-F-F++F-F-F-F++F-F++F-F++F-F-F-F++F-F-F-F++F-F-F-F++F-F++F
  ...
```



questions?

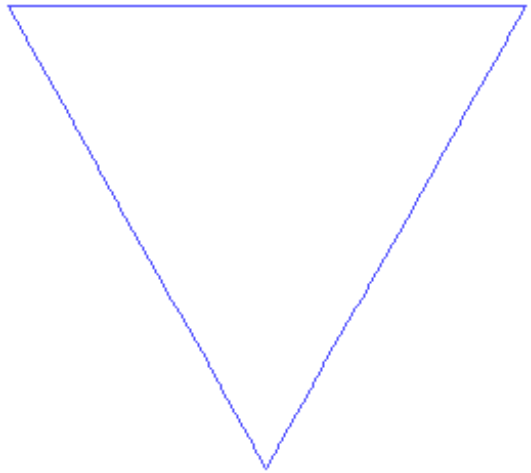
KOCH CURVE

- Named after the Swedish mathematician Helge von Koch
- One of the earliest fractals described/ discovered
- Introduced in a 1904 paper
- Koch snowflake: edit all three edges of an equilateral triangle



KOCH SNOWFLAKE

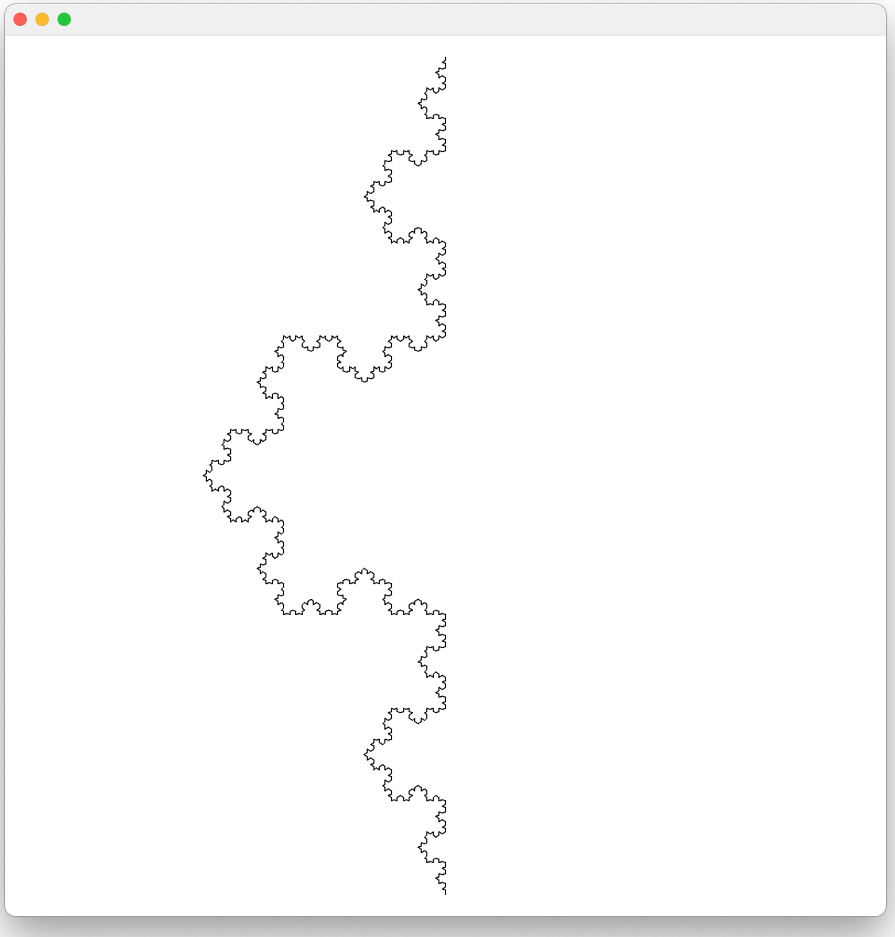
- Named after the Swedish mathematician Helge von Koch
- One of the earliest fractals described/ discovered
- Introduced in a 1904 paper
- Koch snowflake: edit all three edges of an equilateral triangle



PLAY WITH PARAMETERS

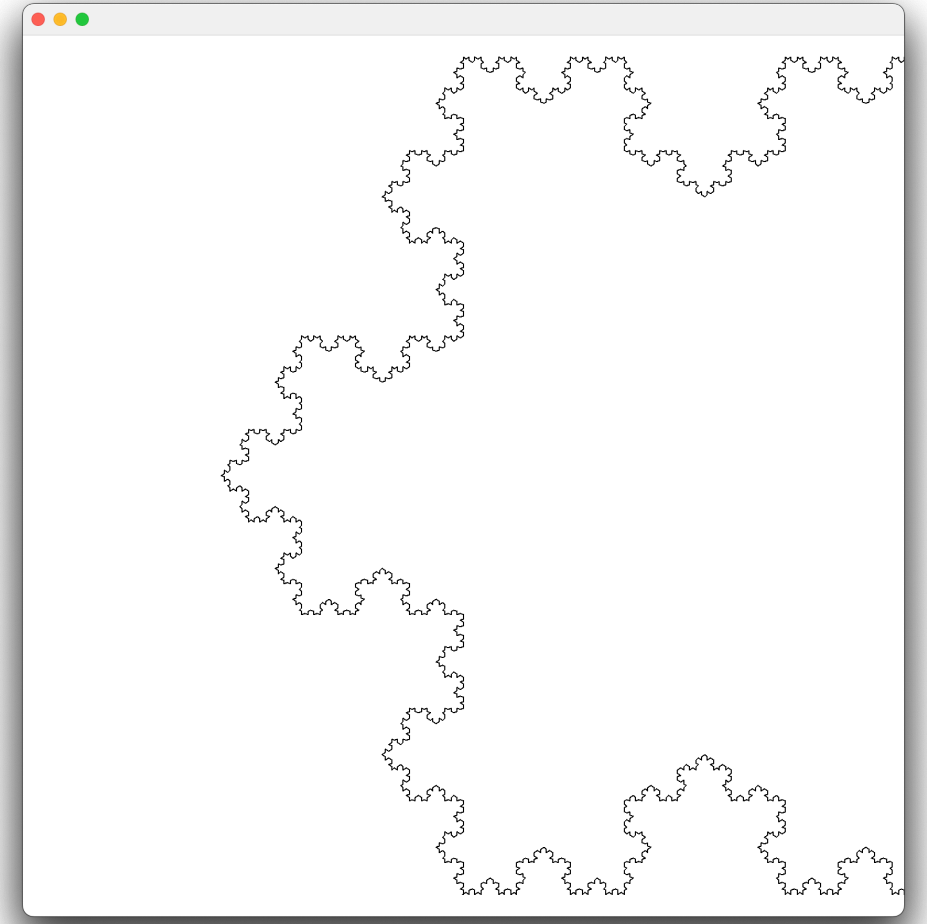
PLAYING WITH PARAMETERS

starting word = F
angle = 60



PLAYING WITH PARAMETERS

starting word = F++F++F++
angle = 60



KEEPING DRAWING ON SCREEN

```
public void mousePressed(MouseEvent e) {
    size = size/2.05;
    t.clearTurtleHistory();
    t.penUp();
    t.setX(width/2);
    t.setY(height/2);
    t.penDown();
    l.iterate();
    l.draw(t,size,angle);
    repaint();
}
```

adjust scale factor

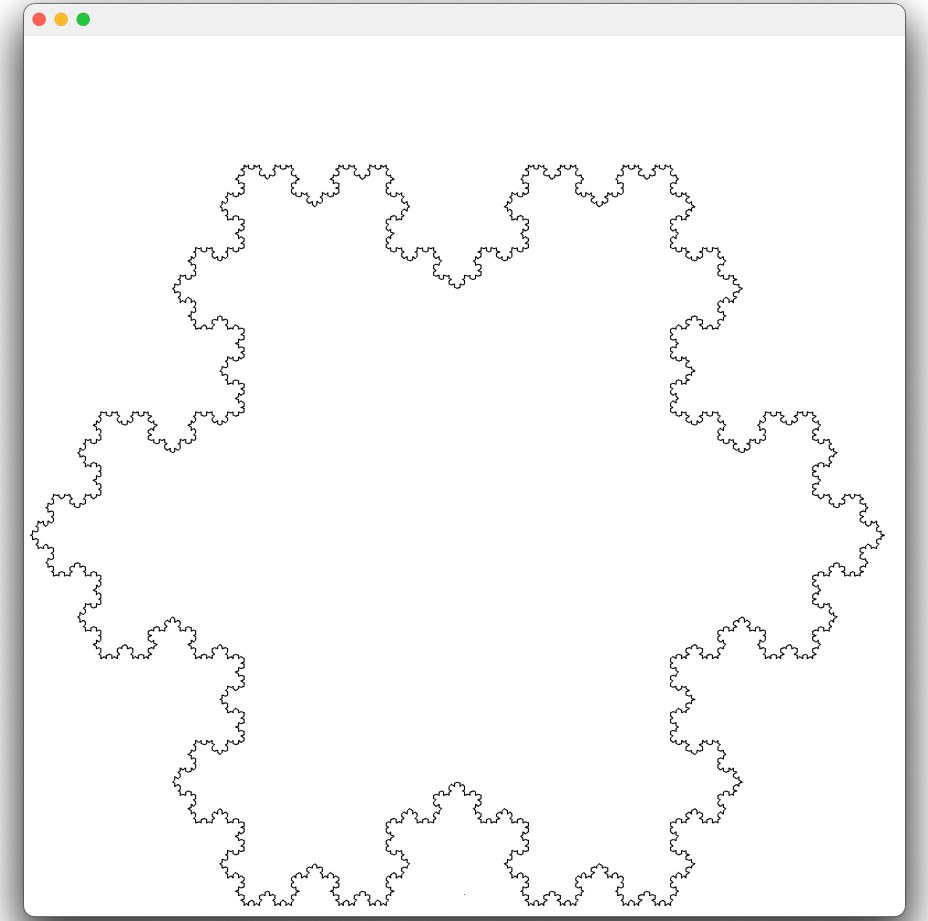
KEEPING DRAWING ON SCREEN

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

move the turtle
to a better starting position

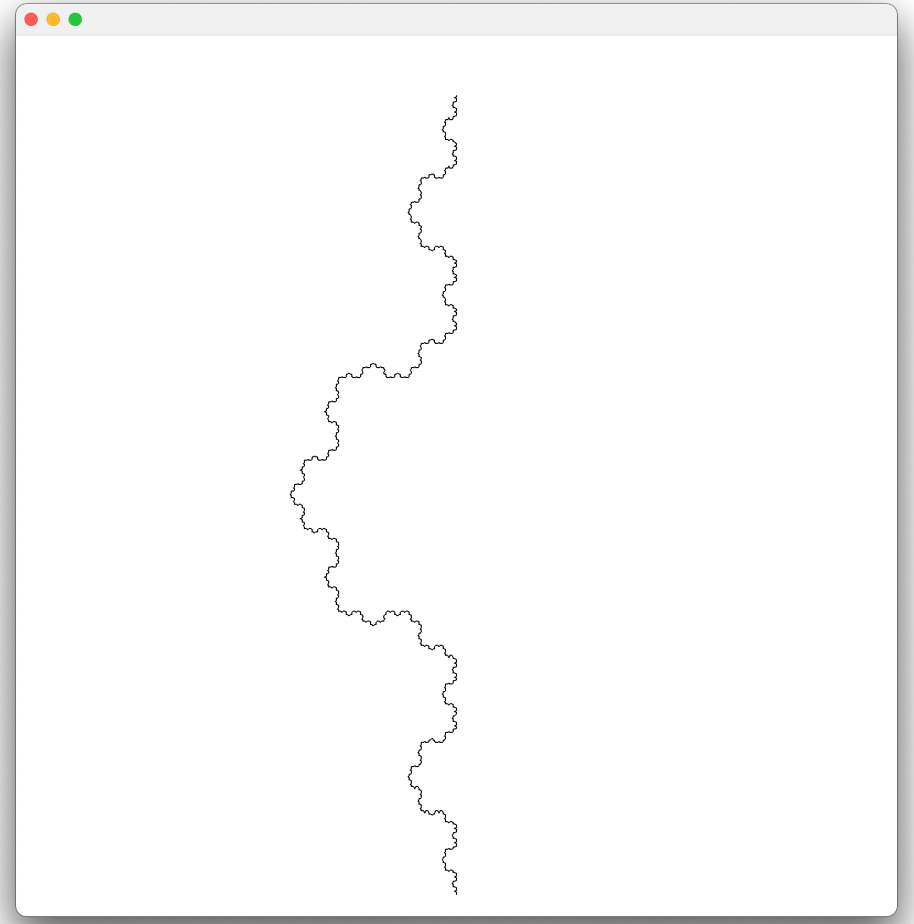
PLAYING WITH PARAMETERS

starting word = F++F++F++
angle = 60



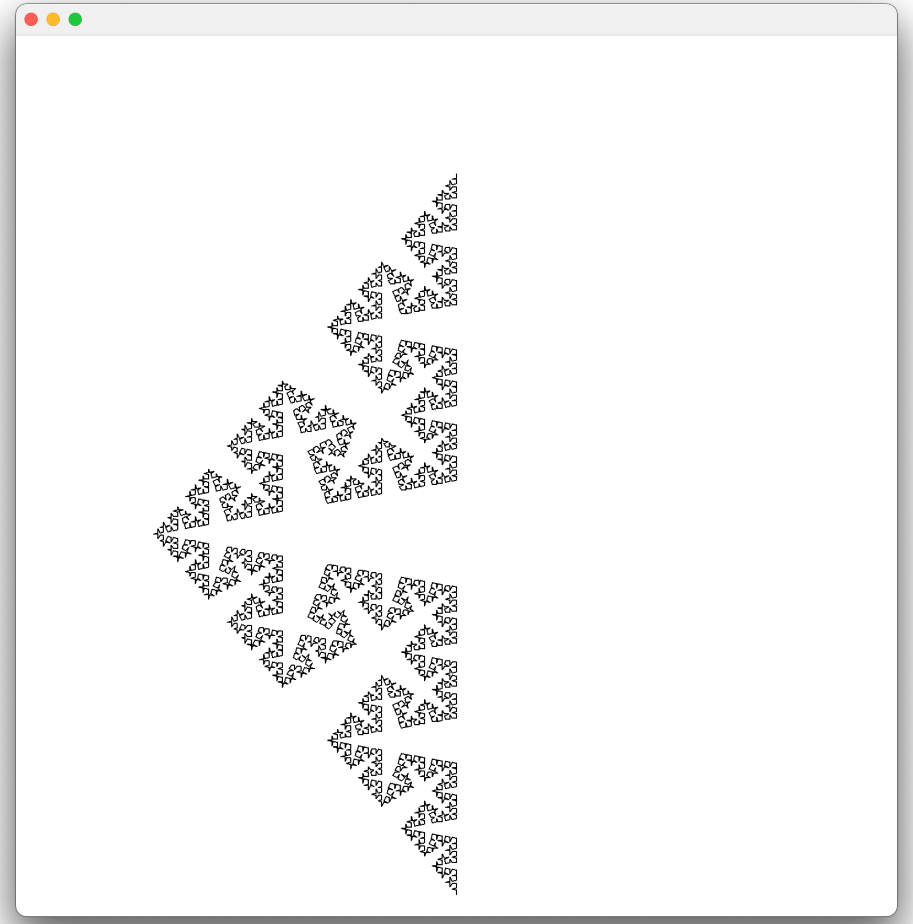
PLAYING WITH PARAMETERS

starting word = F
angle = 45



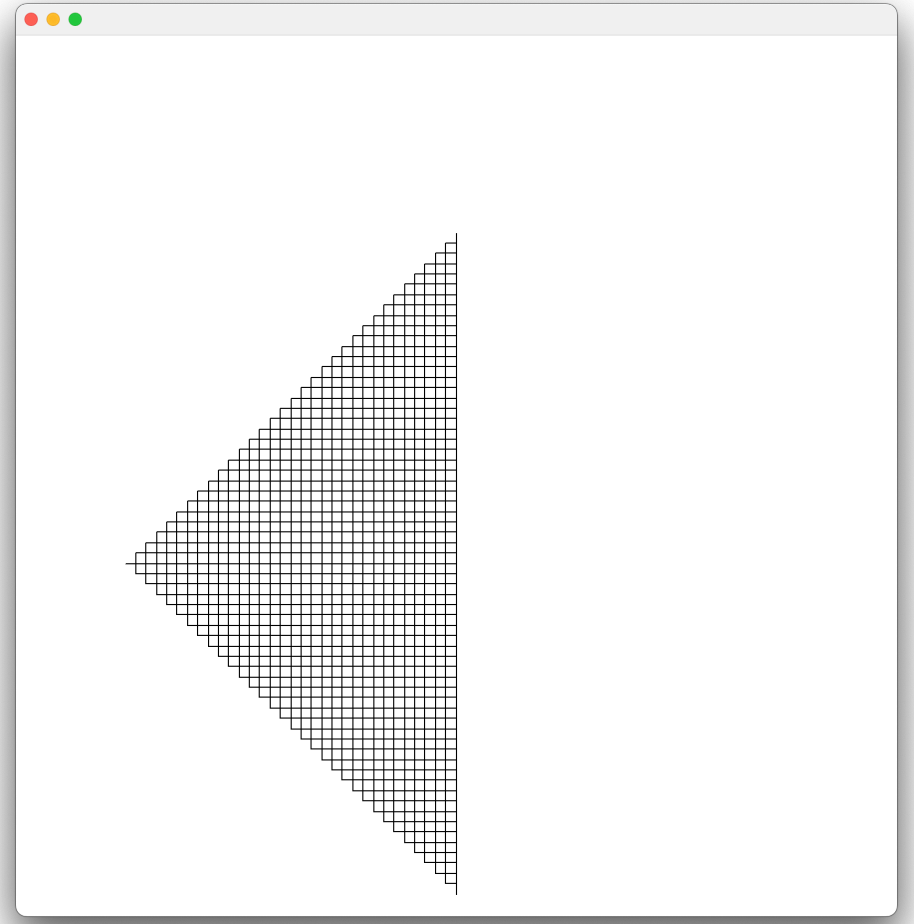
PLAYING WITH PARAMETERS

starting word = F
angle = 80



PLAYING WITH PARAMETERS

starting word = F
angle = 90



questions?

MORE COMPLEX RULES

ADDING SYMBOLS & RULES CAPTURING EDGE DIFFERENCES

L = left edges
R = right edges

starting word = L
rules = L → L-R-
R → +L+R

Turtle interpretation:
L = forward
R = forward

ADD THESE TO OUR CODE

CURRENT LSystem.java

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

    LSystem () {
        startingWord = "F";
        computedWord = "F";
        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 = tempWord.replaceAll(rule[0][0],rule[0][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);
        }
    }
}
```

ADD A NEW CONSTRUCTOR WITH STARTING WORD & RULE AS INPUT PARAMETERS

ADD A CONSTRUCTOR TO LSystem CLASS

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

ADD NEW SYMBOLS TO DRAW METHOD

CURRENT 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);
    }
}
```

Turtle interpretation:
L = forward
R = forward

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);
    }
}
```

Turtle interpretation:
L = forward
R = forward

questions?

IN LSystemVis.java
USING THE NEW CONSTRUCTOR

ADD A CONSTANT VARIABLE FOR OUR NEW L-SYSTEM RULES

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

starting word = L

rules = L → L-R-

R → +L+R

CREATE A NEW L-SYSTEM WITH THIS NEW RULE

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

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

starting word = L

rules = L → L-R-

R → +L+R

WON’T WORK YET…

COMPUTING AN ITERATION WITH MORE THAN ONE RULE

CURRENT CODE

```
void iteration() {
    String tempWord = computedWord;
    computedWord = tempWord.replaceAll(rule[0][0],rule[0][1]);
    iterations++;
    System.out.println("iteration " +iterations +": " +computedWord);
}
```

WHY WON’T THIS WORK?

CURRENT CODE

```
void iteration() {
    String tempWord = computedWord;
    computedWord = tempWord.replaceAll(rule[0][0],rule[0][1]);
    iterations++;
    System.out.println("iteration " +iterations +": " +computedWord);
}
```

only looks at first rule

we now have 2

CHECK ALL THE RULES

A FIRST ATTEMPT

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

will loop through all rules

DOES THIS WORK?

Hint: No

Why not?

WHAT'S HAPPENING

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

starting word = L

rules = L ⟶ L-R-

R ⟶ +L+R

```
iteration 0: L                          should be
iteration 1: L-+L+R-                    iteration 0: L
iteration 2: L-+L+R--+L-+L+R-++L+R-     iteration 1: L-R-
                                        iteration 2: L-R—+L+R-
```

WE'RE MIXING ITERATIONS, DOING REPLACEMENTS ON REPLACEMENTS

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

starting word = L

rules = L ⟶ L-R-

R ⟶ +L+R

```
iteration 0: L                          should be
iteration 1: L-+L+R-                    iteration 0: L
iteration 2: L-+L+R--+L-+L+R-++L+R-     iteration 1: L-R-
                                        iteration 2: L-R—+L+R-
```

questions?

FINISHING ONE ITERATION BEFORE STARTING THE NEXT ONE

A SECOND ATTEMPT: CHAR BY CHAR

```
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);
    }
}
```

loop through characters in word

A SECOND ATTEMPT: CHAR BY CHAR

```
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);
    }
}
```

loop through rules

A SECOND ATTEMPT: CHAR BY CHAR

```
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);
    }
}
```

build new word char by char

NEED TO ADD ONE MORE THING

BECAUSE WE'RE BUILDING THE WORD UP FROM SCRATCH INSTEAD OF THROGH REPLACEMENT

A SECOND ATTEMPT: CHAR BY CHAR

```
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);
    }
}
```

this has to be true for each symbol
there has to be a rule for each symbol

IN LSystemVis CLASS
NEED TO ADD + and - RULES

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

starting word = L

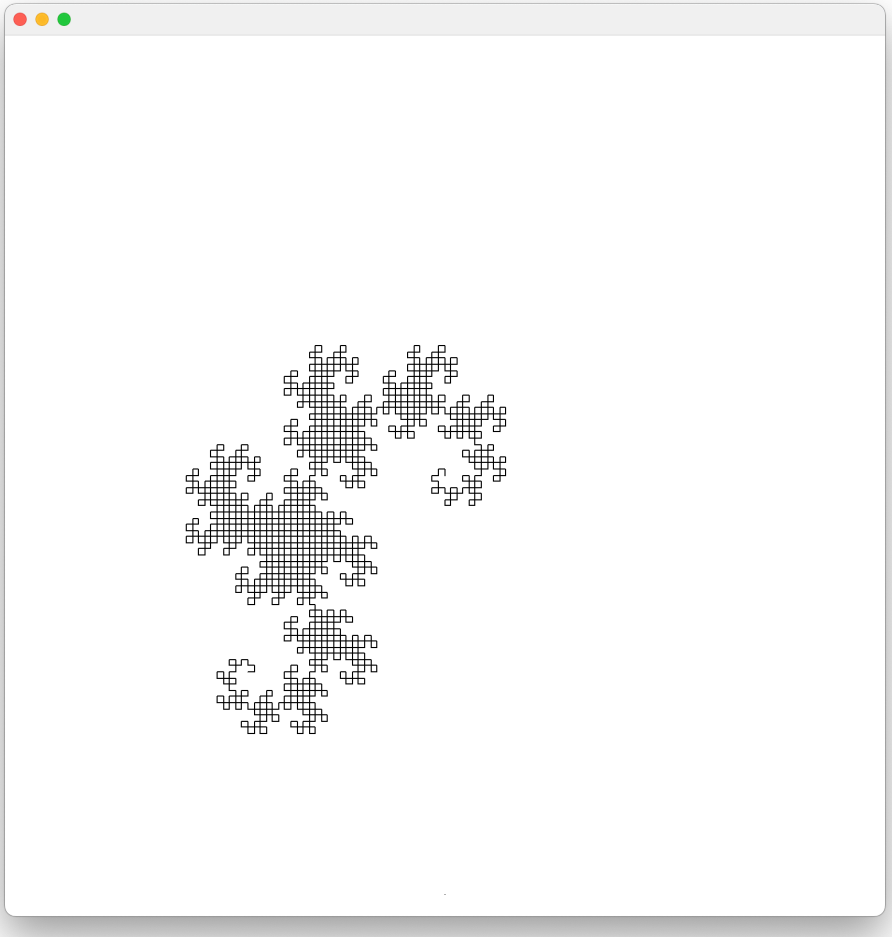
rules = L → L-R-

R → +L+R

questions?

COMPILE AND RUN

starting word = L
rule = dragon
angle = 90

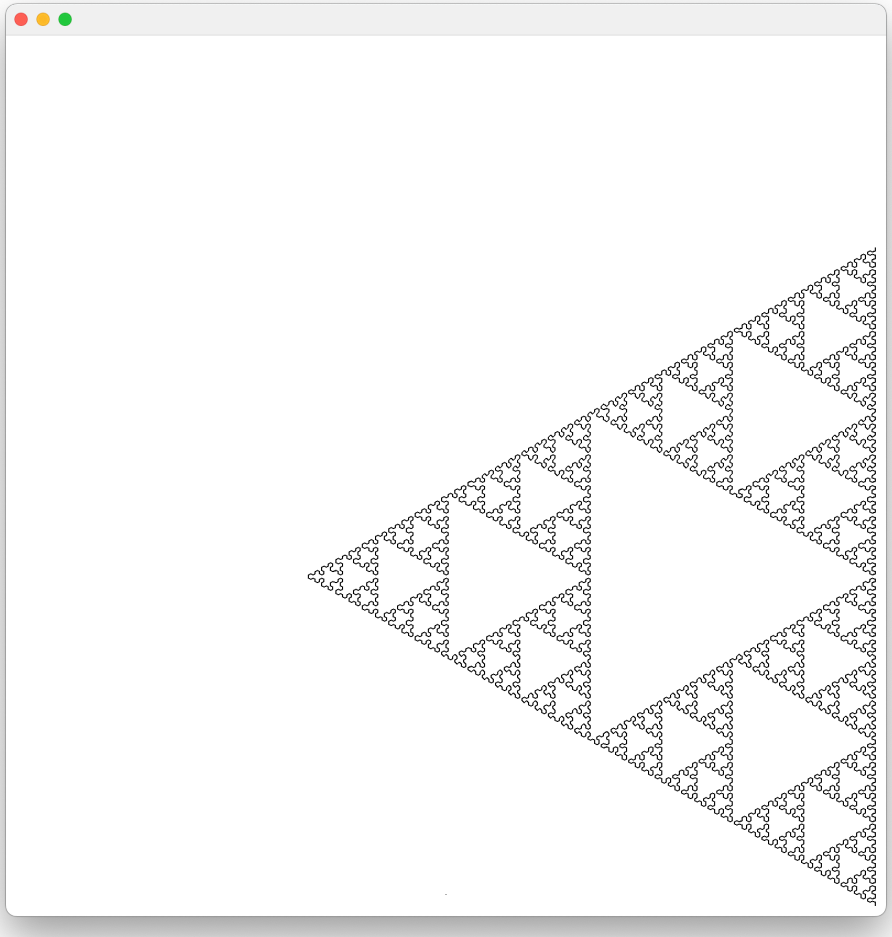


PLAY WITH RULES & PARAMETERS

ANOTHER RULE

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

starting word = L
rule = gasket
angle = 60



questions?

BACK TO BOTANICALS

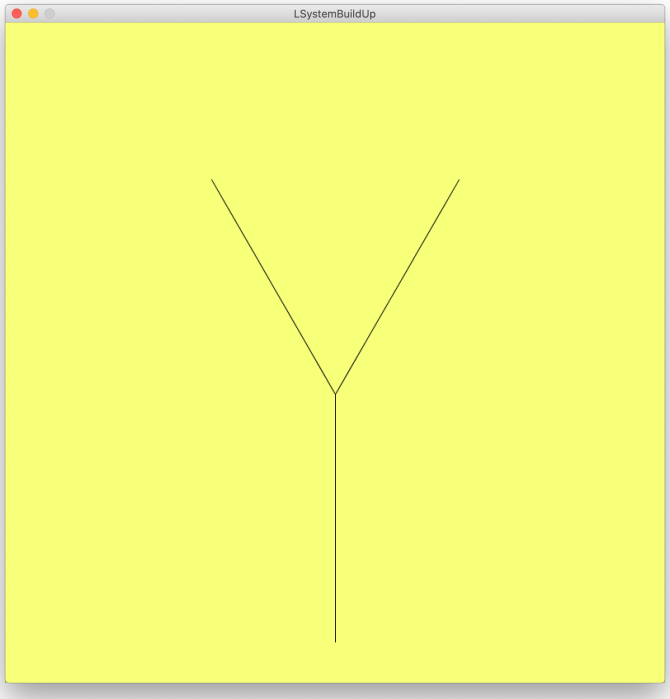
ADDING SYMBOLS & RULES
PUSH AND POP

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

Turtle interpretation:
[= push
] = pop

THINKING ABOUT A TREE

what is a good expression for this beginning of a tree?



F[-F][+F]

Thank you!

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/