P#2 Reed

by L-system
20201127 이창현

Idea : reed

Initial String

FL

Rules

F -> F*[+f]F[+f]

f -> F[-f]F[-f]

L -> *f

Alphabet

F : / Draw forward

+ : Turn right by 60 degree

- : Turn right by 30 degree

* : Turn right by 0.001 radian

[: Save current position and angle
] : restore position and angle stored at corresponding [

L and f : just call the rule

Process

- It start by ‘FL’
- It gradually bends to the right according to the rules
- ‘f’ is a little bit bended
- Last ‘L’ is a more little bit bended

Code

```
//Rules
String S = "FL";
String Rule1 = "F*[+f]F[+f]"; //Rule for 'F'
String Rule2 = "F[-f]F[-f]";  //Rule for 'f'
String Rule3 = "*f";          //Rule for 'L'

float angleOffset = radians(30);

void setup() {
  size(900, 600);
  stroke(0);
}

void draw() {
  background(255);
  translate(0, height);
  rotate( -(HALF_PI-radians(5)) );  //rotate -85degree
  float branchLen = map(mouseY, 0, height, 30, 0.1);
  float branchWeight = map(mouseY, 0, height, 1, 0.01);
  render(S, branchLen, branchWeight);
}

```

```
void render(String S, float branchLen, float branchWeight) {
  int strLen = S.length();
  for (int i=0; i<strLen; i++) {
    switch( S.charAt(i) ) {
    case 'F':
      strokeWeight(branchWeight);
      line(0, 0, branchLen, 0);
      translate(branchLen, 0);
      translate(branchWeight, 0);
      break;
    case '+':
      rotate( 2*angleOffset ); //rotate
      break;
    case '-':
      rotate( angleOffset );  //rotate lite bit
      break;
    case '[':
      pushMatrix();
      break;
    case ']':
      popMatrix();
      break;
    case '*':
      rotate(0.001);          //rotate little bit
      break;
    }
  }
}

```

```
String ApplyRule( String s ) {
  String result = "";
  int strLen = s.length();
  for (int i=0; i<strLen; ++i) {
    char c = s.charAt(i);
    if (c == 'F') {
      result += Rule1;
    }else if ( c == 'f'){
      result += Rule2;
    }else if ( c == 'L'){
      result += Rule3;
    } else {
      result += c;
    }
  }
  return result;
}
```

Screenshot