Pulse works .pde

documentation

Concept

A pulsing L-system that resembles fireworks.

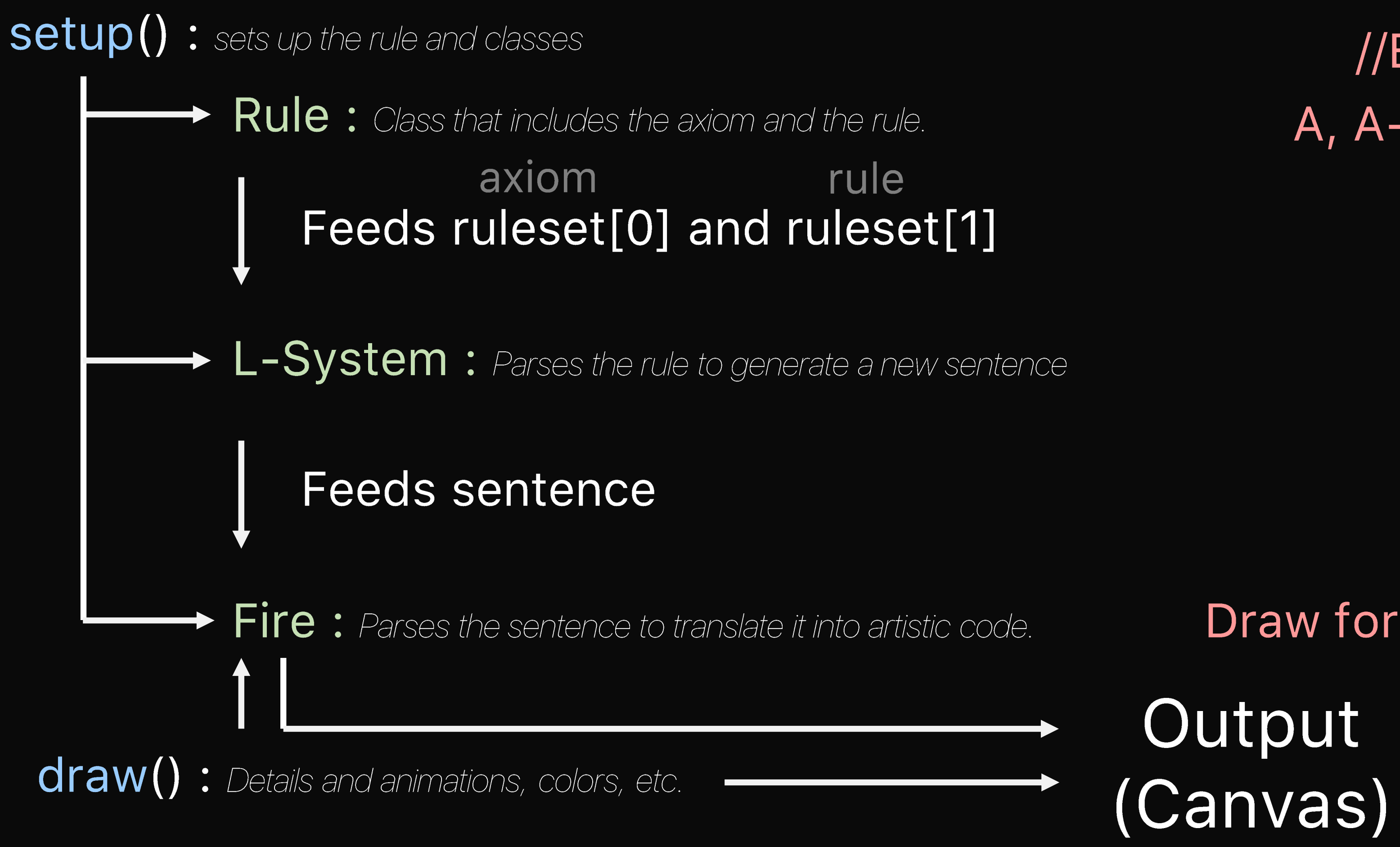
Ideas:

- A symmetrical tree L-system
- Pulsing firework animations
- Perhaps a bokeh-like filter
- 3D?

A photo of fireworks with a bokeh filter applied.

Pulse
works
.pde

Flow



setup() : *sets up the rule and classes*

Rule : *Class that includes the axiom and the rule.*

axiom rule
Feeds ruleset[0] and ruleset[1]

L-System : *Parses the rule to generate a new sentence*

Feeds sentence

Fire : *Parses the sentence to translate it into artistic code.*

draw() : *Details and animations, colors, etc.*

Output
(Canvas)

//Example//

A, A->AB & B->BA

A	Axiom
AB	Gen 1
ABBA	Gen 2
⋮	

Draw for A, skip for B, etc.

Pulse
works
.pde

Rule

The class that contains the axiom and rules.

```
Rule[] ruleset = new Rule[1];

ruleset[0] = new Rule('F', 'G[+F][-F]');
```

Therefore, the rule of this sketch is G[+F][-F], with the axiom F.

Pulse
works
.pde

L-system

Parses the rule to generate a new sentence.

```
void generate() {

  StringBuffer nextgen = new StringBuffer();

  for (int i = 0; i < sentence.length(); i++) {
   char curr = sentence.charAt(i);
   String replace = "" + curr;
   for (int j = 0; j < ruleset.length; j++) {
    char a = ruleset[j].getA();
    if (a == curr) {
     replace = ruleset[j].getB();
     break;
    }
   }
   nextgen.append(replace);
  }
  sentence = nextgen.toString();
}
```

Because we cannot edit the original string on the fly, a StringBuffer is used to make the temporary next generation.

getA() calls the axiom.

getB() calls the rule.

Therefore, the sentence becomes:

F	Axiom
G[+F][-F]	Gen 1
G[+G[+F][-F]][-G[+F][-F]]	Gen 2
⋮	

Pulse works .pde

Fire(Turtle)

Parses the sentence into artistic code.

```
void render() {

  for (int i = 0; i < todo.length(); i++) {
    char c = todo.charAt(i);

    switch(c) {
    case 'F':
      circle(0, 0, rad);
      translate(dist, 0);
      break;
    case 'G':
      translate(dist, 0);
      break;
    case '+':
      rotate(theta);
      break;

      ...
```

Therefore,

G[+F][-F]

is parsed into: (in pseudo code)

```
translate(dist);
pushMatrix();
rotate(theta);
circle();
popMatrix();
pushMatrix();
rotate(-theta);
circle();
popMatrix();
```

Pulse works .pde

void draw() Details.

Color

```
rcolor = rcolor + rCof*counter;
gcolor = gcolor + gCof*counter;
bcolor = bcolor + bCof*counter;

rcolor = random(150, 255);
gcolor = random(150, 255);
bcolor = random(150, 255);

rCof = random(-1, 1);
gCof = random(-1, 1);
bCof = random(-1, 1);

fill(rcolor, gcolor, bcolor, 10);
```

Reset Animation

```
lsys.resetTo("F", ruleset);
fire.resetTo(lsys.getSentence(), 50,
radians(random(25)), 100);
```

Filter

```
filter(BLUR, 1);
fill(rcolor, gcolor, bcolor, 10);
```

Press p to screenshot

```
void keyPressed() {
  if (key == 'p') {
    saveFrame();
  }
}
```

Pulse
works
.pde

Color changer

Changes the color (obviously).

Every axiom starts with three random values for rgb.

```
rcolor = random(150, 255);
gcolor = random(150, 255);
bcolor = random(150, 255);
```

Also, every axiom generates three coefficients for the changing rate of each value.

```
rCof = random(-1, 1);
gCof = random(-1, 1);
bCof = random(-1, 1);
```

This is used to make the changes gradual, instead of sudden.

Every generation adds the coefficient * # of generation (counter).

```
rcolor = rcolor + rCof*counter;
gcolor = gcolor + gCof*counter;
bcolor = bcolor + bCof*counter;
```

Then, apply the color.

```
fill(rcolor, gcolor, bcolor, 10);
```

Pulse works .pde

Animation Reset

Resets the animation.

Reset L-system

```
lsys.resetTo(axiom, ruleset);
```

In this case,

```
lsys.resetTo("F", ruleset);
```

Reset Fire

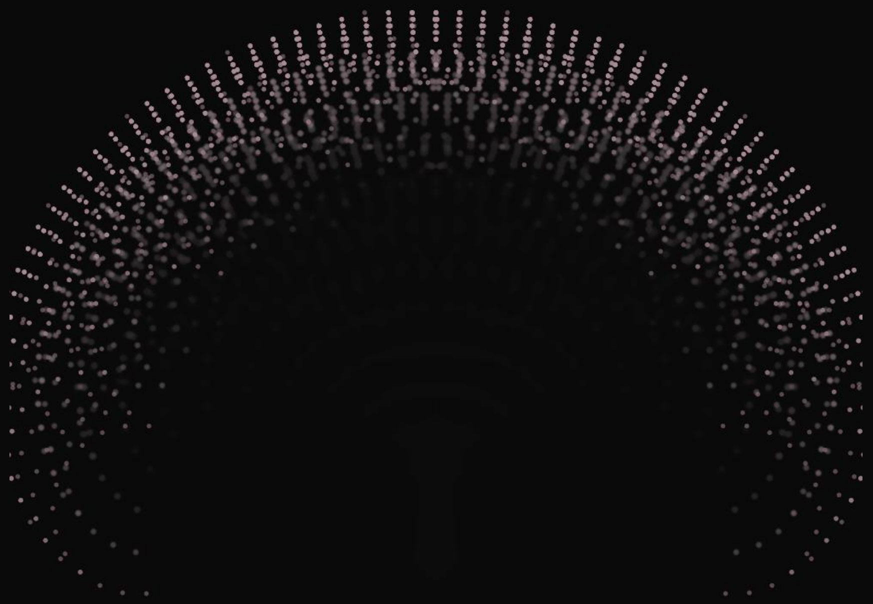
```
fire.resetTo(sentence, distance, theta, radians);
```

In this case,

```
fire.resetTo(lsys.getSentence(), 50, radians(random(25)), 100);
```

Pulse works .pde

Screenshots



Pulse
works
.pde

Screenshots

Pulse
works
.pde

Pulse works .pde