メディアアート・プログラミング2

東京藝術大学 芸術情報センター開設科目 後期金曜4限 第8週

2023.11.24 松浦知也（matsura.tomoya@noc.geidai.ac.jp teach@matsuuratomoya.com）

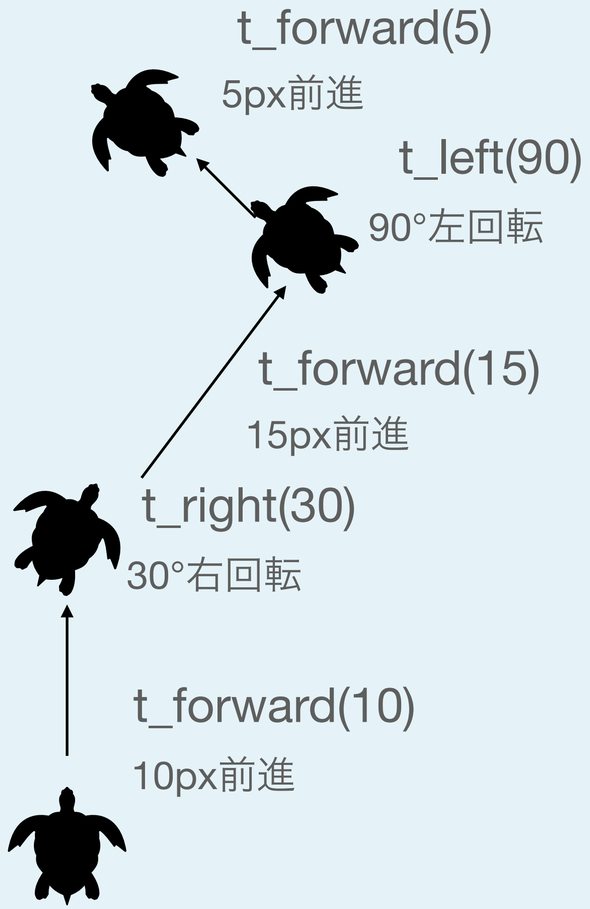
L-Systemの続き

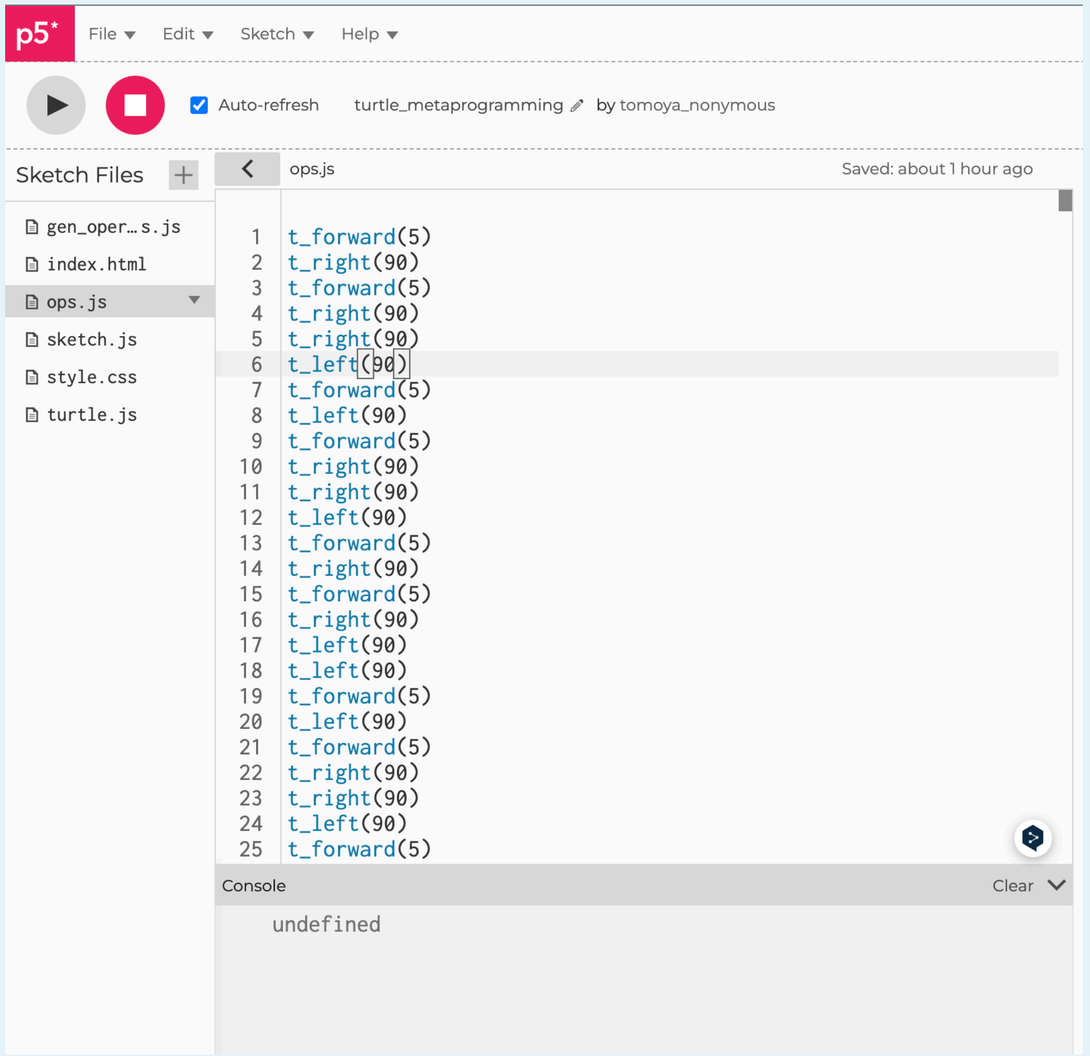
タートルグラフィックス（復習）

- 前進、右折、左折の命令のみで
  線を描くプログラム

例

```
t_forward(10);
t_right(30);
t_forward(15);
t_left(90);
t_forward(5);
```



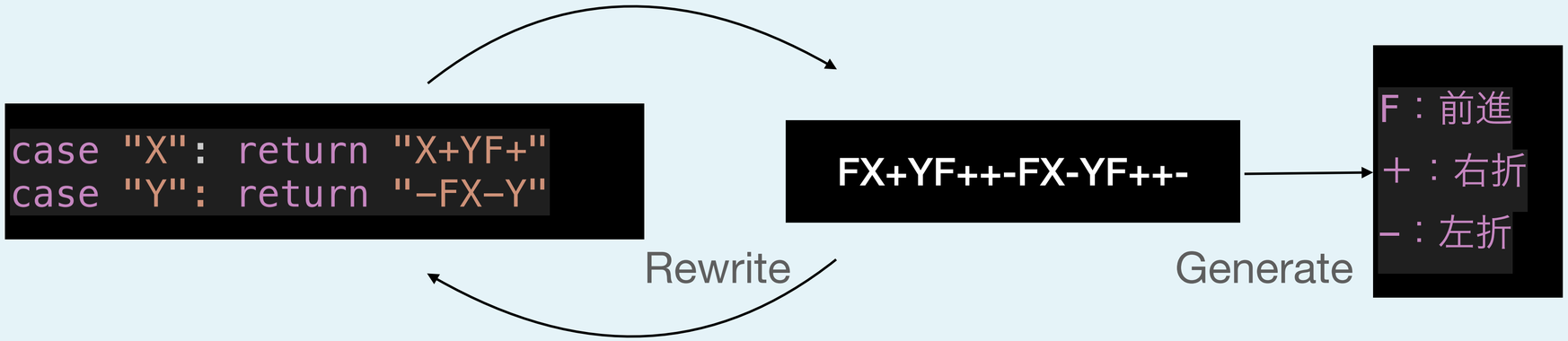


https://editor.p5js.org/tomoya_nonymous/sketches/tF5rNCHJ0

でops.jsというファイルをペーストして置き換える

L-System(復習)

- 項書き換え系にタートルグラフィックスの描画命令を加えたもの

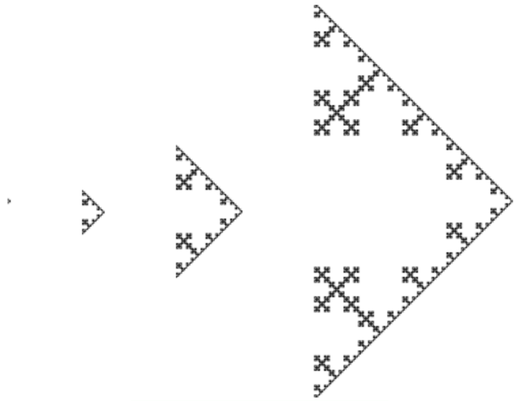


置き換えプログラムに
とってはF,+,-は意味なし

描画プログラムに
とってはX,Yは意味なし

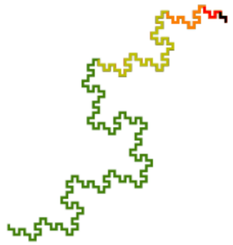
L-Systemのさまざまなルール

No.01 コッホ曲線 Koch Curve



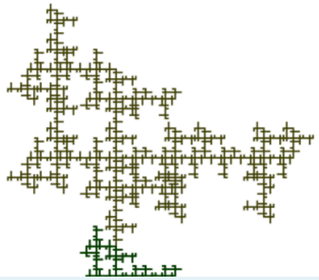
SVG File	01.svg
公理(初期)Axiom	F
ルールRule	F=F+F-F+F
角度(左,右)Angle(L,R)	90,90
ランダム化Randomize	0
標準級数Order	---

No.02 蛇足 Meandering Snake

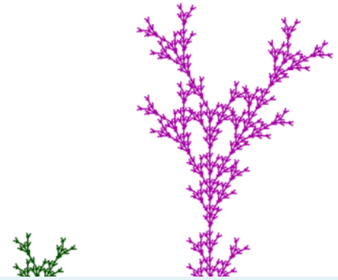


SVG File	02.svg
公理(初期)Axiom	F
ルールRule	F=F++F--F-F+F
角度(左,右)Angle(L,R)	90,90
ランダム化Randomize	0
標準級数Order	---

No.03 都市探査 Urban Charting



No.04 水草A



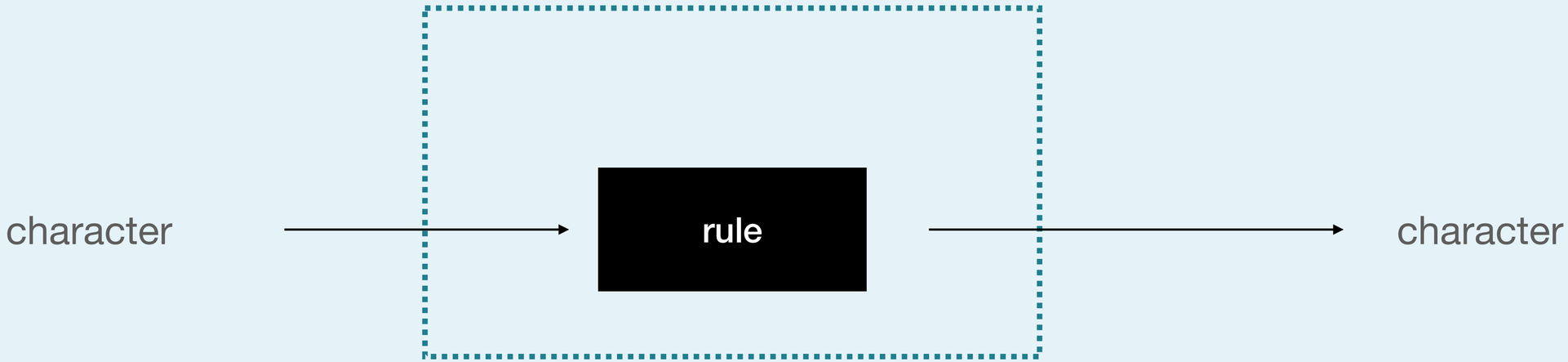
HIIRAGI COMPANY L-System Tips　https://hiiragicompany.net/ls.html

L-System
植物の置き換えルール

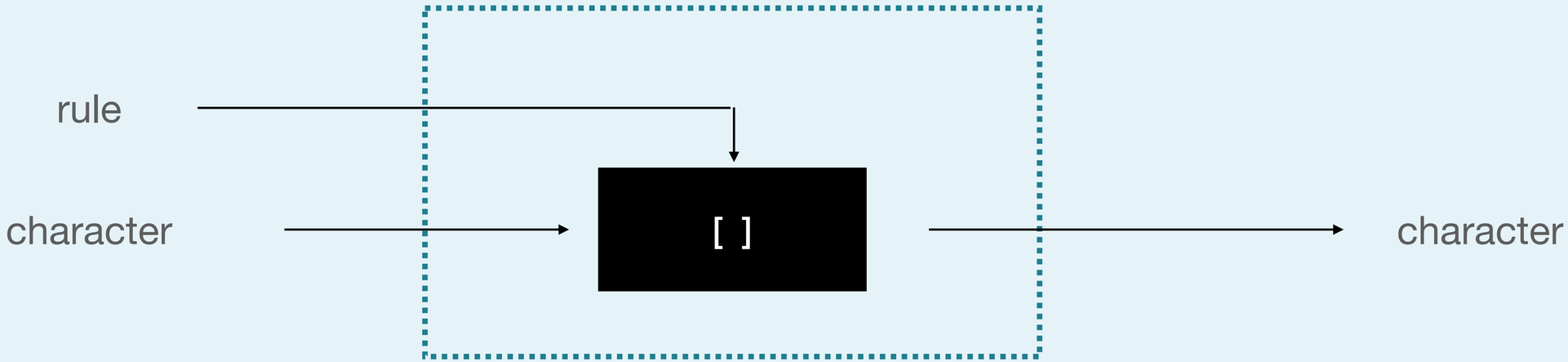
```
let plant_start = "XF"
const rule_plant = (char) => {
    switch (char) {
        case "X": return "F+[[X]-X]-F[-FX]+X"
        case "F": return "FF"
        default: return char
    }
}
```

今回はFを見つけたらFFに=置き換えるごとに直進の長さが倍増

[,]という新しい文字が登場、これは状態の保存と復帰（後述）

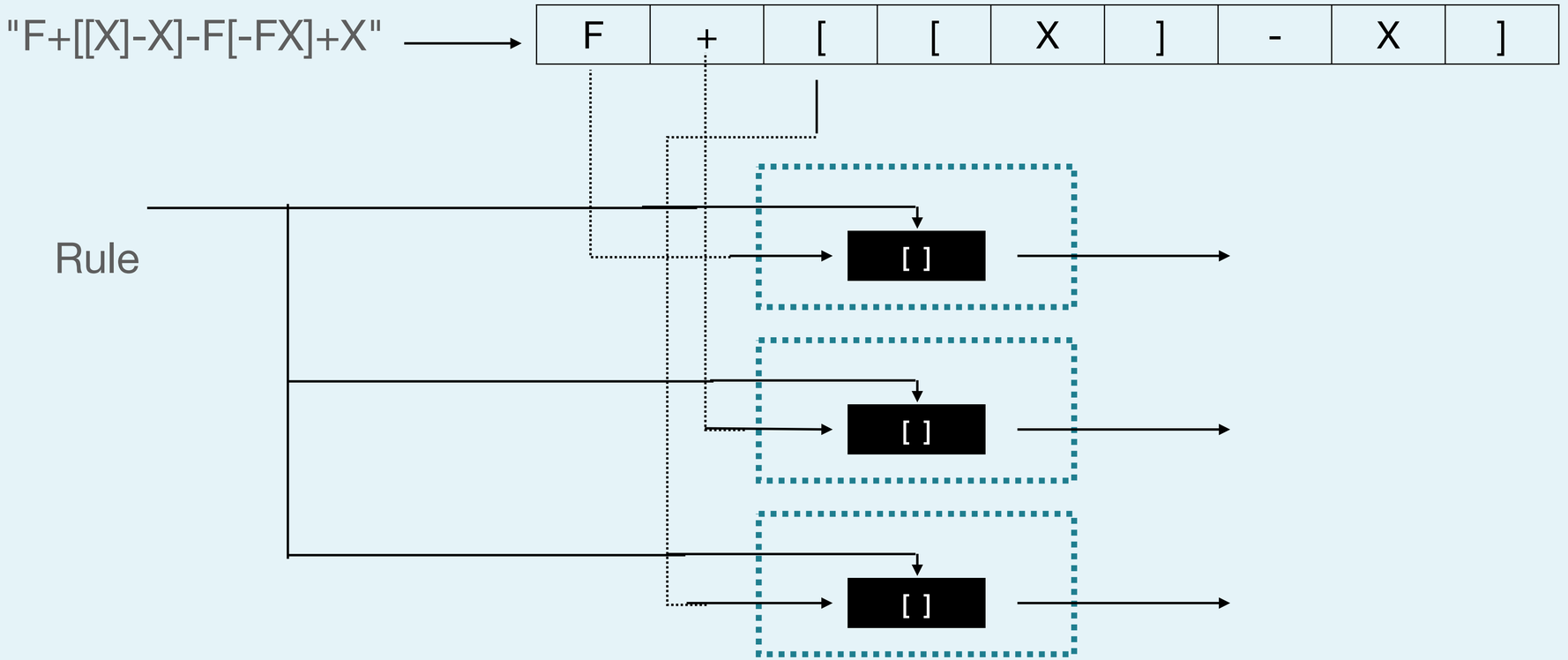

character
rule
character

```
const rewrite_char = (c, rule) => {
    return rule(c)
}
```



Ruleを外側からパラメーターとして与えてやることで、
文字列とルールの掛け合わせでいろんなパターンが作れるようになる

```
const rewrite = (str, rule) => {
    return Array.from(str).map(rule).join("")
}
```



Ruleを外側からパラメーターとして与えてやることで、

文字列とルールの掛け合わせでいろんなパターンが作れるようになる

XF

F+[[X]-X]-F[-FX]+XFF

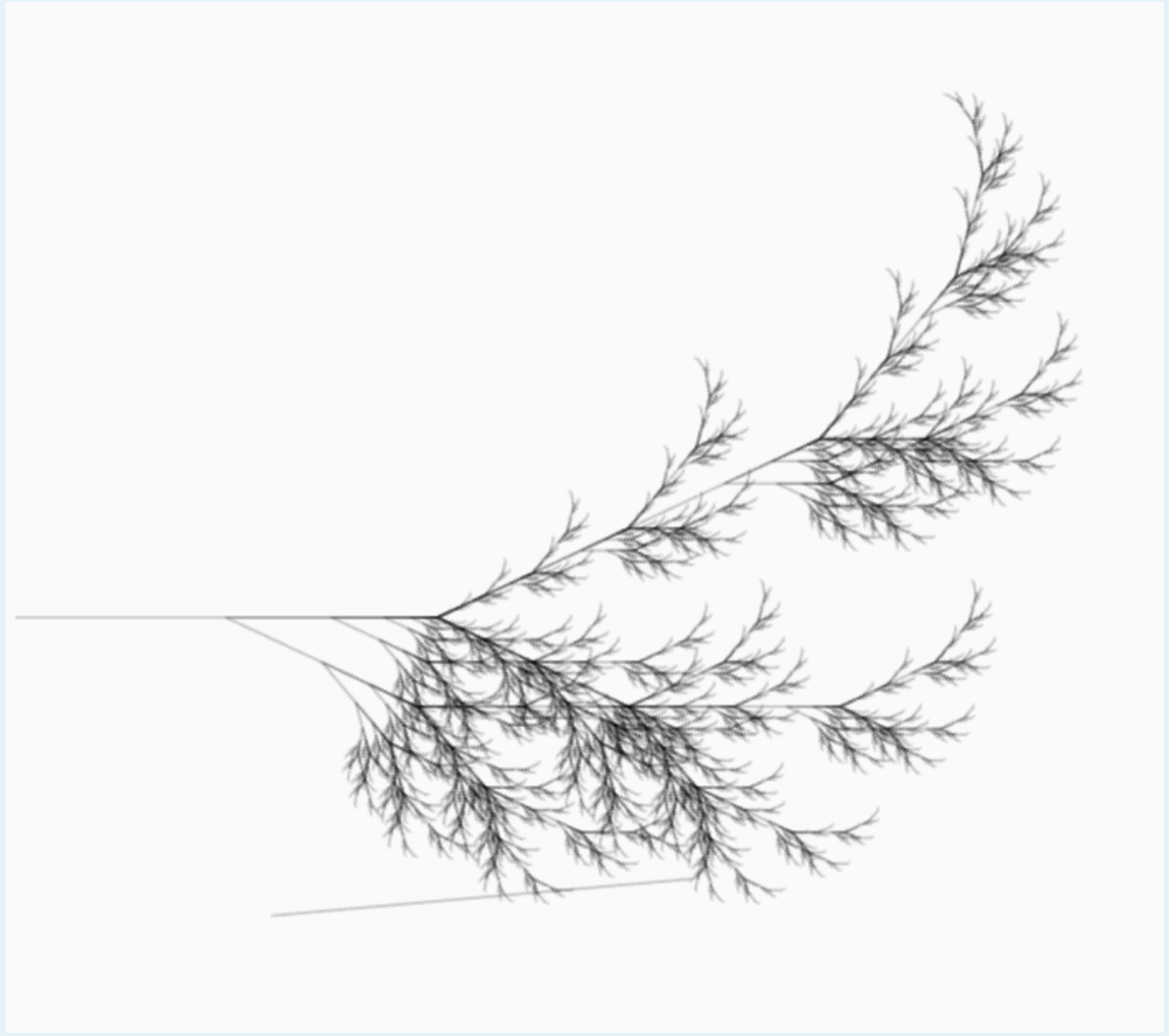
FF+[[F+[[X]-X]-F[-FX]+X]-F+[[X]-X]-F[-FX]+X]-FF[-FFF+[[X]-X]-F[-FX]+X]+F+[[X]-X]-F[-FX]+XFFFF

F

F+[[]-]-F[-F]+FF

FF+[[F+[[]-]-F[-F]+]-F+[[]-]-F[-F]+]-FF[-FFF+[[]-]-F[-F]+]+F+[[]-]-F[-F]+FFFF

Xはタートルグラフィックスに無関係なので一旦無視してみる



状態の復帰を使うことで、一筆書きじゃなくて分岐が実現できる

L-System

それぞれの文字をタートルグラフィックス操作関数に置き換える

```
const generate =(src,rule,repetition,len,angle)=>{
    let res = src;
    for (let i = 0; i < repetition; i++) {
        res = rewrite(res, rule);
    }
    res = res.replace(/F/g,`t_forward(${len})\n`)
    res = res.replace(/\+/g,`t_right(${angle})\n`)
    //角度に負の値が入ると-が混ざってエラーになるので注意
    res = res.replace(/\-/g,`t_left(${angle})\n`)
    res = res.replace(/\[/g,`t_push()\n`)
    res = res.replace(/\]/g,`t_pop()\n`)
    res = res.replace(/X/g,"")
    res = res.replace(/Y/g,"")
    return res
}
const len = 5;
const angle = 90;
const repeat = 12;
console.log(generate(src,rule_dragon,repeat,len,angle))
```

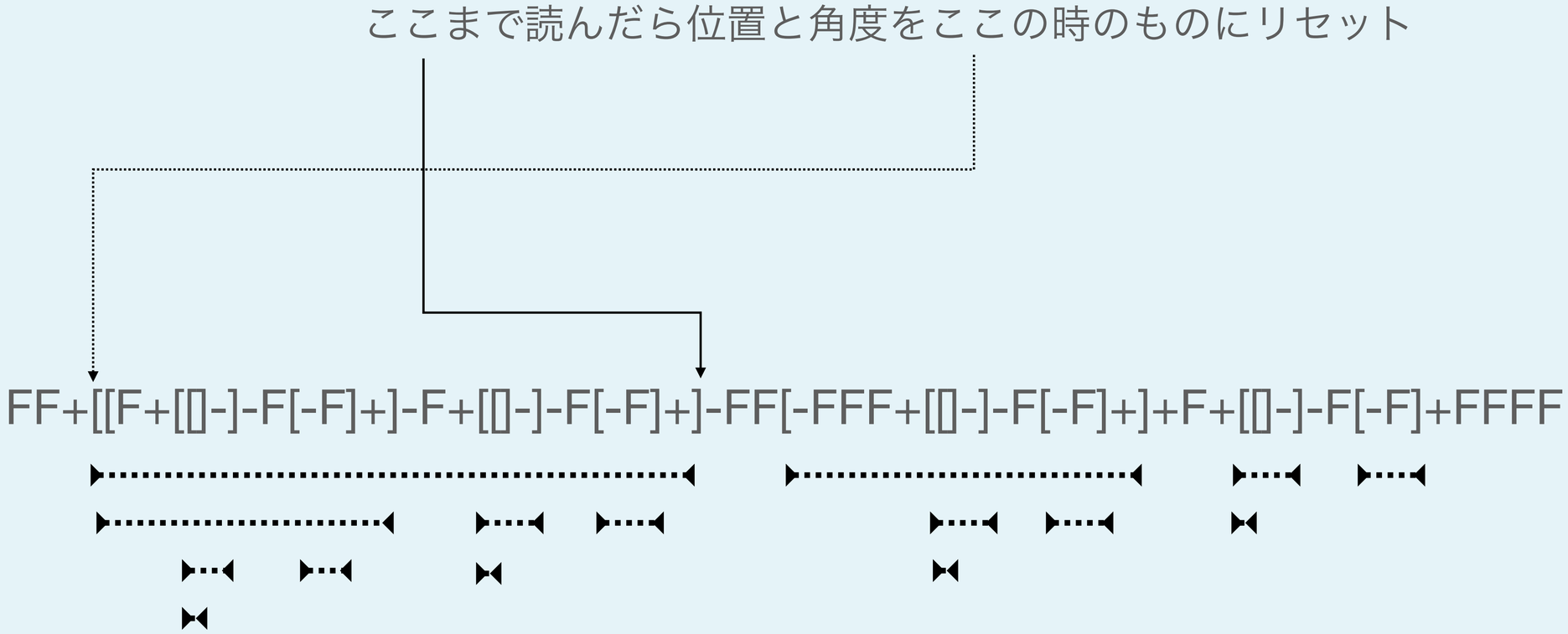
t_push()は位置と角度を記憶

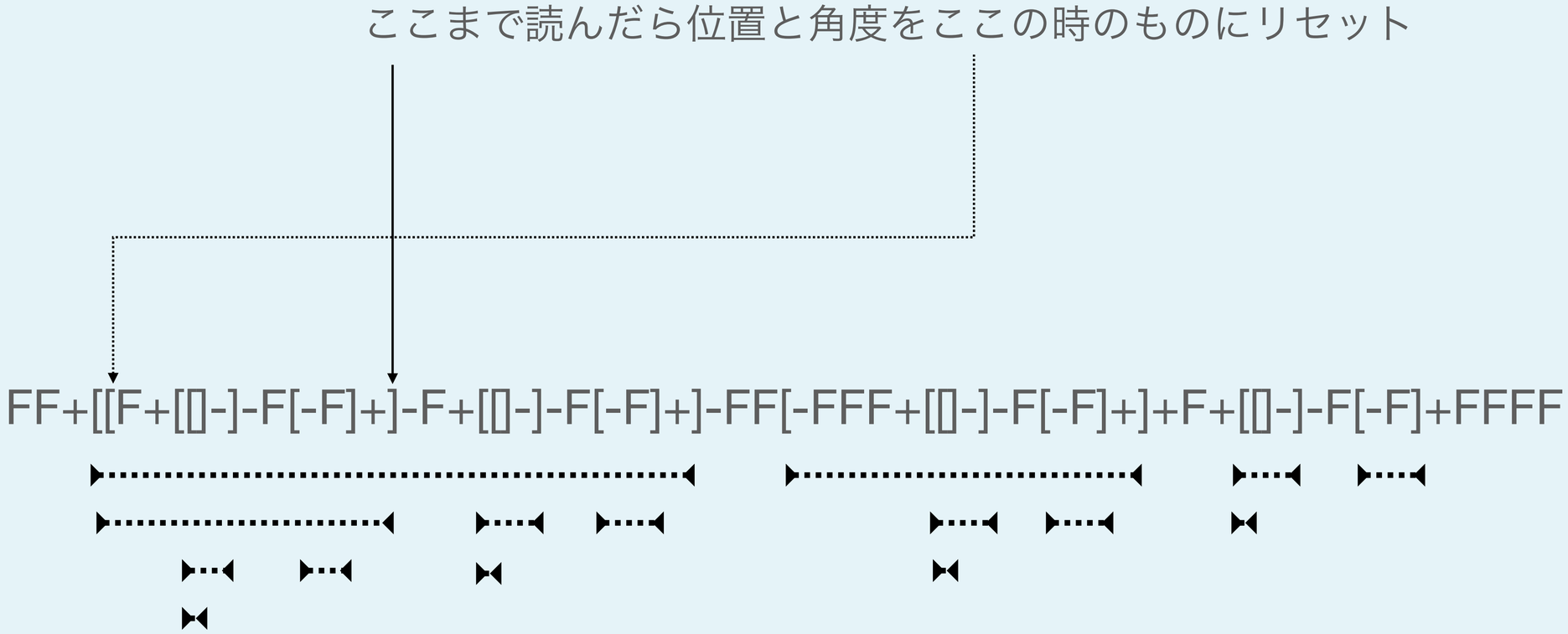
t_pop()は最新の記憶した位置と角度を取り出す

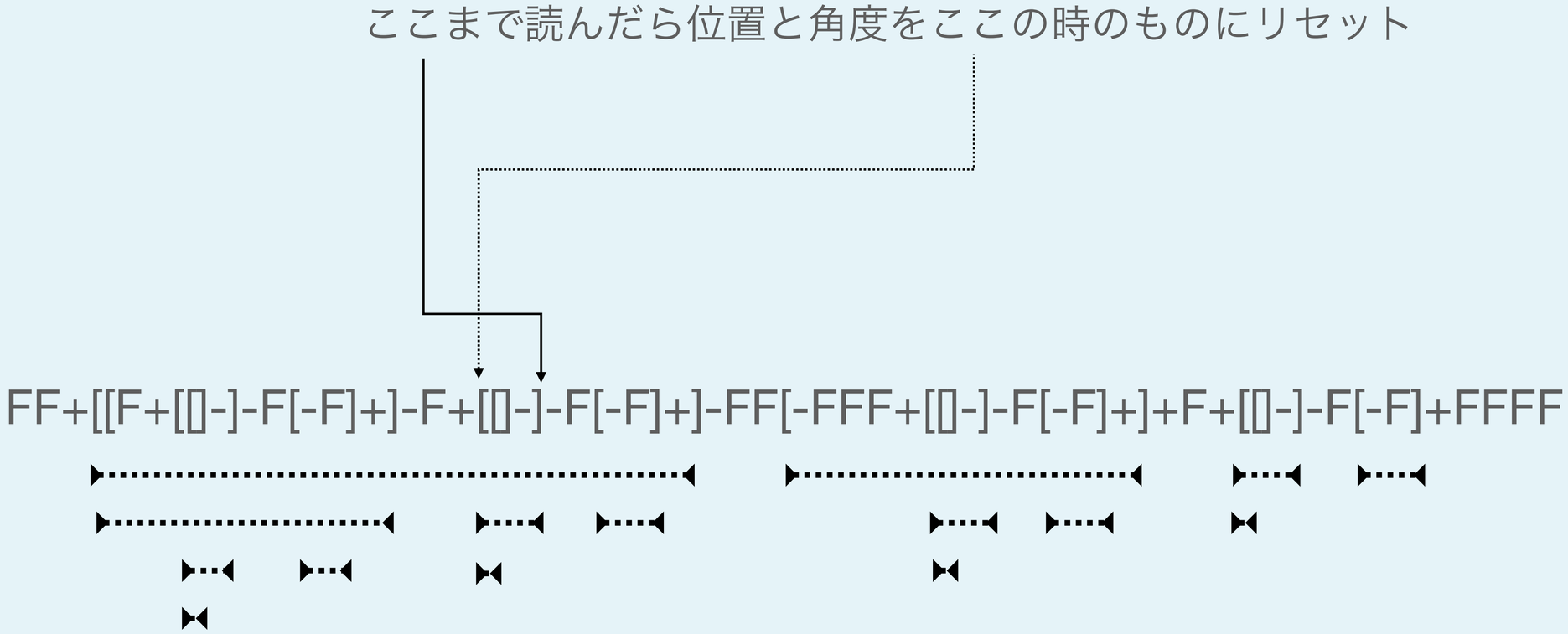
X,Yは何もせず消しているのに注目

FF+[[F+[[]-]-F[-F]+]-F+[[]-]-F[-F]+]-FF[-FFF+[[]-]-F[-F]+]+F+[[]-]-F[-F]+FFFF

rewriteの繰り返しごとに[]のネストが深くなっていく

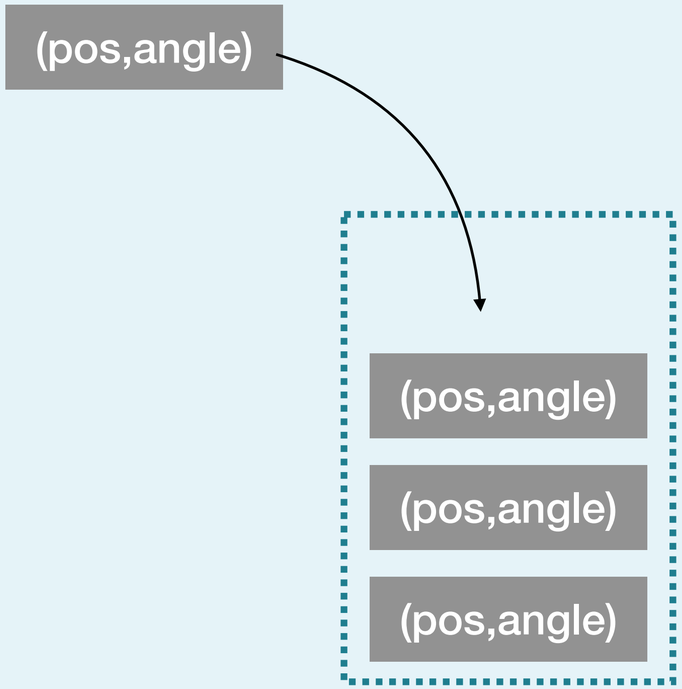

ここまで読んだら位置と角度をここの時のものにリセット
FF+[[F+[[]-]-F[-F]+]-F+[[]-]-F[-F]+]-FF[-FFF+[[]-]-F[-F]+]+F+[[]-]-F[-F]+FFFF


ここまで読んだら位置と角度をここの時のものにリセット
FF+[[F+[[]-]-F[-F]+]-F+[[]-]-F[-F]+]-FF[-FFF+[[]-]-F[-F]+]+F+[[]-]-F[-F]+FFFF

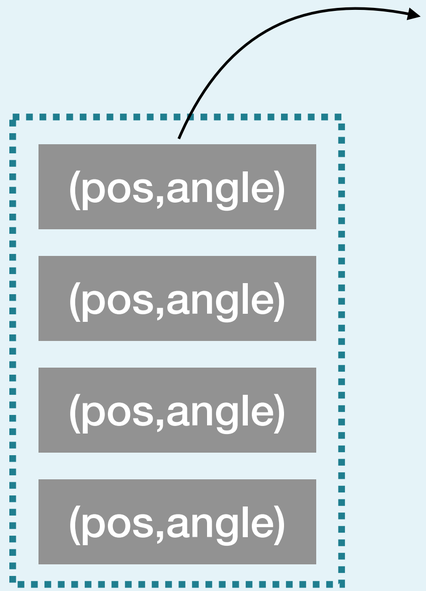

ここまで読んだら位置と角度をここの時のものにリセット
FF+[[F+[[]-]-F[-F]+]-F+[[]-]-F[-F]+]-FF[-FFF+[[]-]-F[-F]+]+F+[[]-]-F[-F]+FFFF

スタック

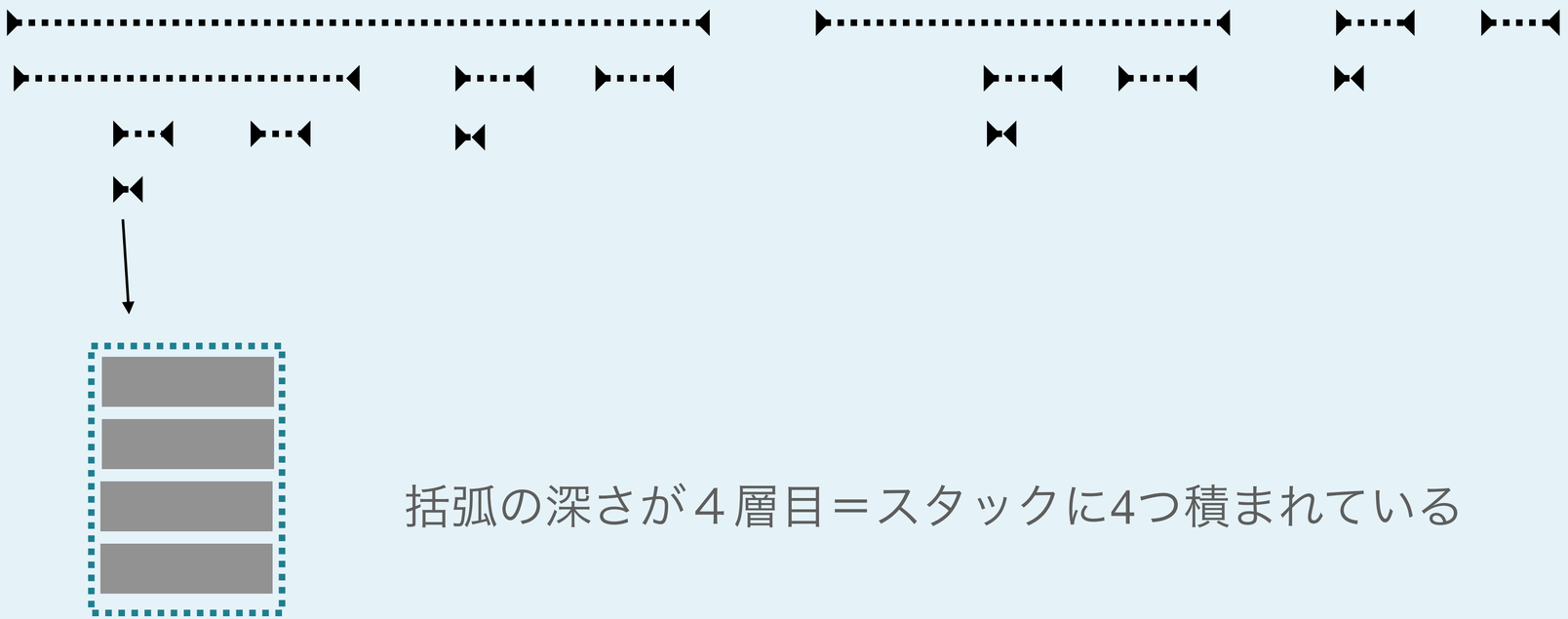
t_push()の度に一番上に積んで



t_pop()の度に一番上のものを取り出して使う
（使ったものは削除）



FF+[[F+[[]-]-F[-F]+]-F+[[]-]-F[-F]+]-FF[-FFF+[[]-]-F[-F]+]+F+[[]-]-F[-F]+FFFF



括弧の深さが４層目＝スタックに4つ積まれている

表現としてのプログラミング言語実装

難解 (Esoteric) Programming Language

Create account Log in

Main Page | Discussion | Read | View source | View history | Search Esolang

Main page
Community portal
Language list
Browse by category
Recent changes
Random page
Help

Tools

What links here
Related changes
Special pages
Printable version
Permanent link
Page information

Welcome to **Esolang**, the esoteric programming languages wiki!

Why not join us on IRC?

This wiki is dedicated to the fostering and documentation of programming languages designed to be unique, difficult to program in, or just plain weird.

For readers

You'll probably want to find out what on earth an esoteric programming language is in the first place.

Then, you might want to explore the complete list of languages, or find something more specific with the categories.

You could also visit the joke language list, which lists languages that can't even be programmed in.

Failing that, you could take a look at a random language.

You could also take a look at the list of special pages.

After getting bored, you could visit the Sandbox and have fun.

Featured language

Thue is an esoteric programming language based around the idea of a "semi-Thue system": a system which specifies strings that can be rewritten to certain other strings; a program is simply a list of search strings, and possible replacements for them. As a nondeterministic language, a program has the potential to halt if there is some way to reach an end state via applying replacements, even if rules such as "always apply the first replacement" would lead to an infinite loop. No data storage is necessary, apart from a single string that holds the entire state of the running program, although this often causes programs to run slowly due to delays in communicating information from one part of the string to another. *(more⋯)*

Previously featured: Funciton · Brainfuck · Deadfish · Emmental · *more⋯*

For creators

If you've just created a language, you can create an article for it by typing its name into the search box, assuming the name is not already taken, but be sure to take a look at the help guide first. Then you should add it to the language list (or the joke language list, as appropriate).

If you haven't got that far yet, take a look at the list of ideas for inspiration.

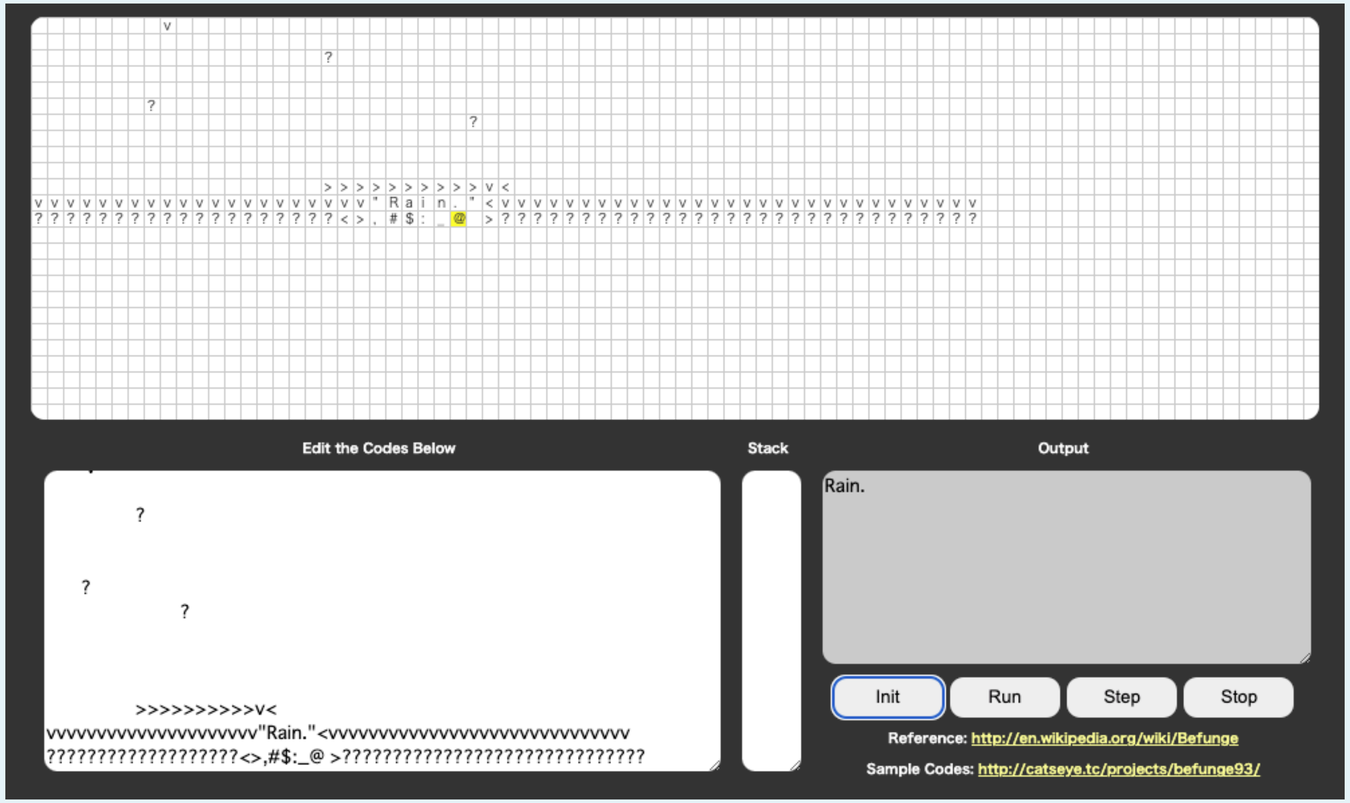
Otherwise, you could help out with a work in progress.

Meta

- Learn about this wiki
- Check out the recent changes
- View the site policies
- Download an XML dump of the wiki's content
- Discuss the wiki on the community portal's talk page
- Talk with other esolang enthusiasts in the places listed in the community portal
- Go to the main page

https://esolangs.org/wiki

Befunge



https://qiao.github.io/javascript-playground/visual-befunge93-interpreter/

久保田晃弘 "Rain"

算法詩 1

v

```
                                          >>>>>>>>>>v<
vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv"Rain."<vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
??????????????????????????????????????????<>,#$:_@ >??????????????????????????????????????
```

この算法詩 1 を実行すると、プログラムの左上からまず下に向ってプログラムの実行場所 (プログラムカウンタ) が、雨のように落下していく。プログラムカウンタが一番下の行に達すると「?」命令によってそれが左右にランダムにシフトし、再び一番上の行から落下を繰り返す。何度かプログラムカウンタが落下を移動を繰り返す内に、それが中央の「Rain.」と書かれた周辺に落ちると、「Rain.」という文字を出力して、プログラムが終了する。

v

>>>>>>>>>>v<
vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv"Rain."<vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
??????????????????????????????<>,#$:_@ >??????????????????????????????????

- 空間に？を足してみよう