# 知能情報システム論

第1回　フラクタルと再帰関数
第2回　フラクタルとLシステム
第3回　~~ドラゴン曲線~~
　　フラクタルとアフィン変換

# [復習]L-systems

- Lシステムとは、大雑把に言えば 非終端記号 と 終端記号 を区別しない形式文法（文献にも依るので注意）。

定義.
Lシステム G = <N, P, S> は次の3要素からなる：
N: 記号の有限集合
P: 書き換え規則の有限集合
S: 開始記号

- すべての記号 $x \in N$ がただひとつの書き換え規則をもつとき Lシステム は D0L-システム (deterministic context-free L-system) と呼ばれる。

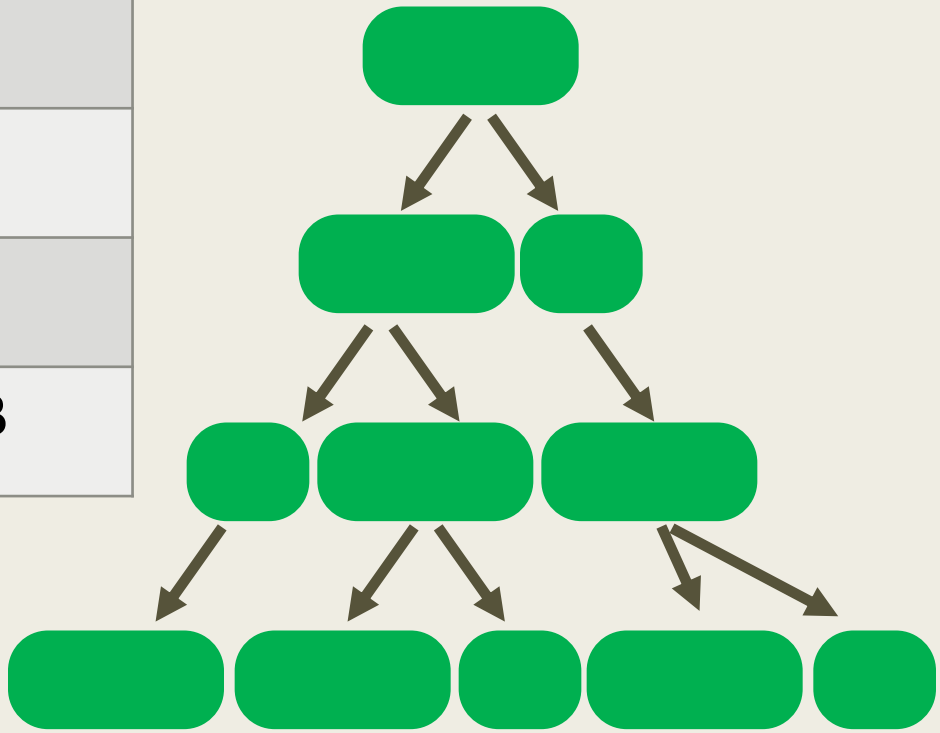
例3. $G_3$ = <N, P, S> を次で定める。

N = {A, B}, S = A, P = {A → AB, B → A}

Lシステムでは、各記号に対して毎回書き換え規則が適用される。

| 時刻（Pの適用回数） | 記号列 |
|---|---|
| n = 0 | A |
| n = 1 | AB |
| n = 2 | ABA |
| n = 3 | ABAAB |

$G_3$は deterministic なので時刻に対して記号列がただひとつ定まる。



# [復習]L-system を用いたフラクタル

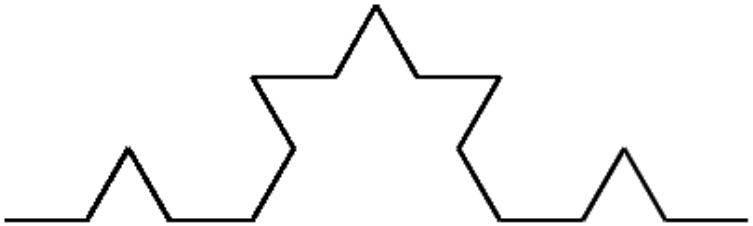
例5. (コッホ曲線)

$G_5$ = <N, P, S> を次で定める。

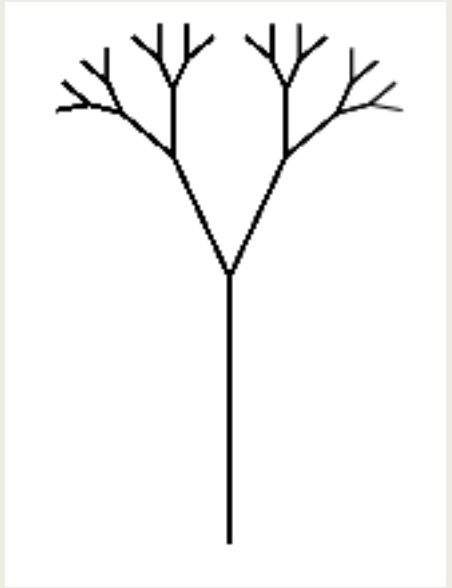
N = {F, +, -}, S = F, P = {F → F+F--F+F}

各記号に対する操作の対応は次の通り

- F はタートルによる直線の描画
- + はタートルを反時計回りに 60° 回転
- - はタートルを反時計回りに -60° 回転
  （時計回りに 60° 回転）



# [復習]その他の例（木）



例6.

$G_6$ = <N, P, S> を次で定める。

N = {F, G, +, -, [, ], |, s, t}, S = F,

P = {F →s|[+F][-F]t}

各記号に対して以下の操作を定義する。

- F は設定長だけタートルを進め直線を描く
- G はタートルを進めるだけ（線は描かない）
- + はタートルを反時計回りに 25° 回転
- - はタートルを反時計回りに -25° 回転
- [ はタートルの位置と方向をスタックに保存
- ] はタートルの位置と方向をスタックから取り出す
- | は書き換えの回数に応じた長さだけタートル進め、直線を描く
- s, t は書き換え回数を読み取るためのメモ

例11.

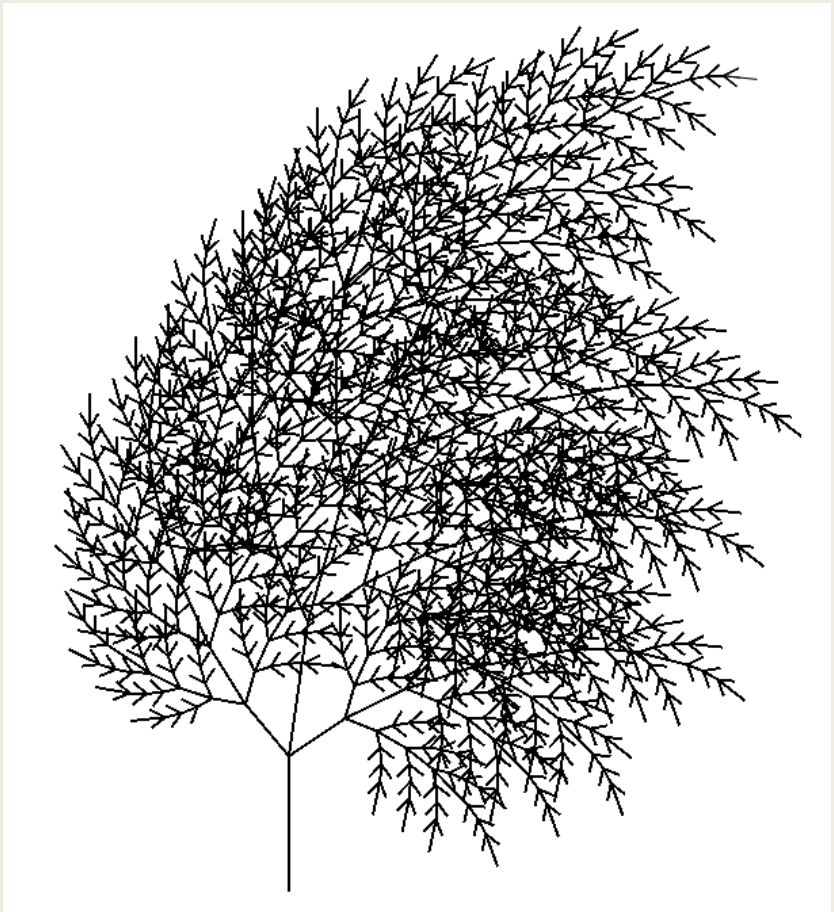
$G_{11}$ = <V, P, S> を次で定める。

N = {F, G, +, -, [, ], |, s, t},

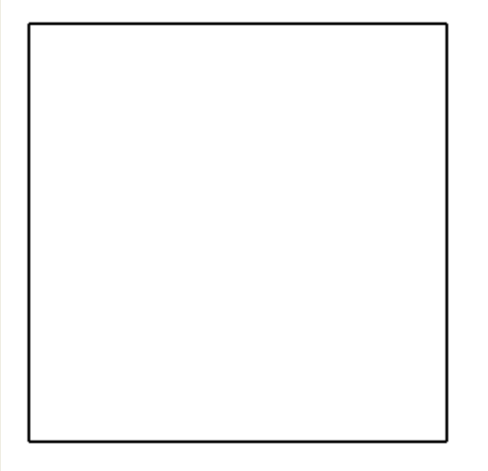
S = F,

P = {F → s|[++++++F][-------F]-|[++++F][------F]-|[+++F][-----F]-|Ft}

ただし、角度を8°とする。



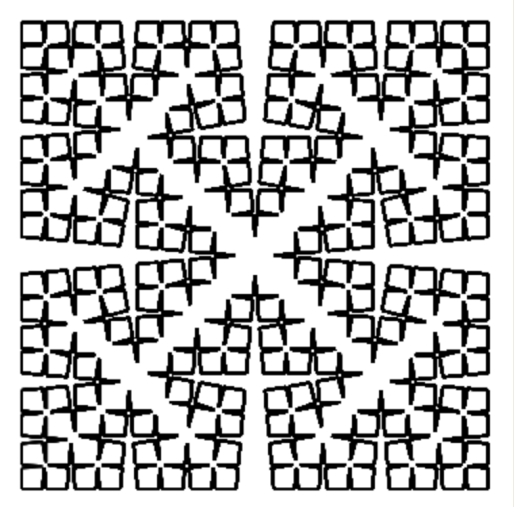
# ベースとモチーフの考え方をL-systemに応用



ベース

モチーフ

- 再帰関数を用いたフラクタルでは「ベース」と「モチーフ」を用意してフラクタルを描いた。
- L-system でも「ベース」と「モチーフ」の考え方を使えないか？



Level = 4

# ベースとモチーフの考え方をL-systemに応用

- 開始記号が「ベース」
- 書き換え規則の集合Pが「モチーフ」に対応する。

例12.

$G_{12}$ = <V, P, S>

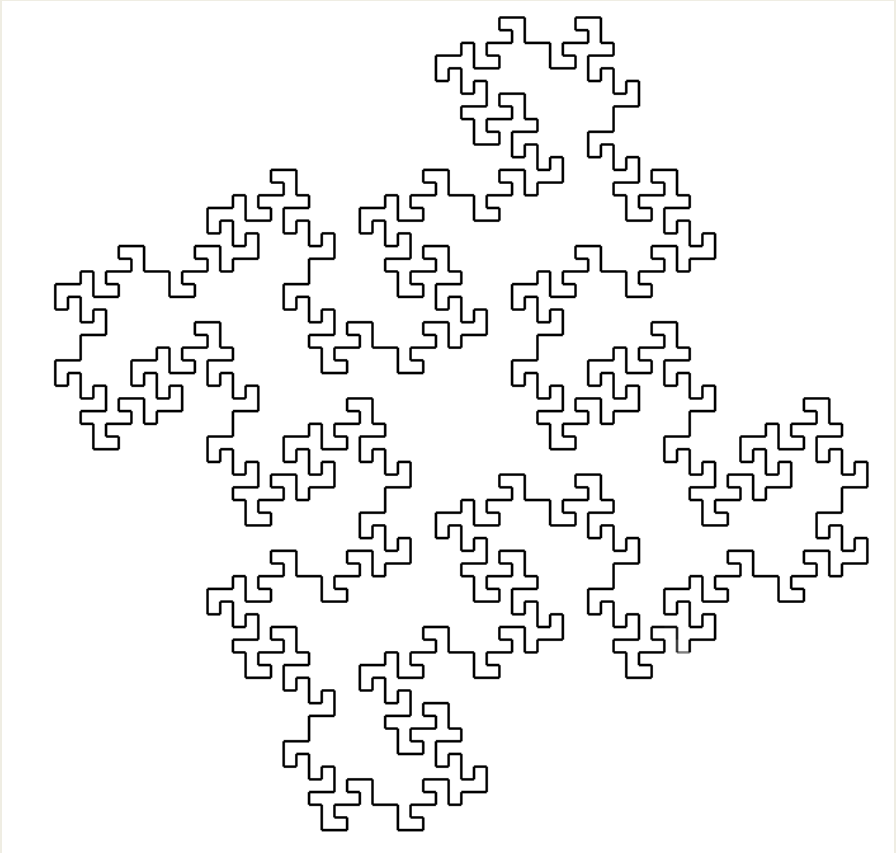
N = {F, G, +, -, [, ], |, s, t},

S = F+F+F+F,

P = {F → F-FF+FF+F+F-F-FF+F+F-F-FF-FF+F}

例題.

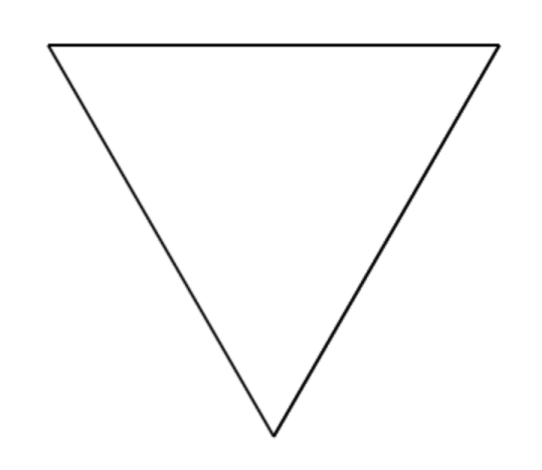
「chinou03.html」内のプログラムをいじり、右のフラクタル図形を出力しよう。SとPだけでなく、角度も調整する必要がある。



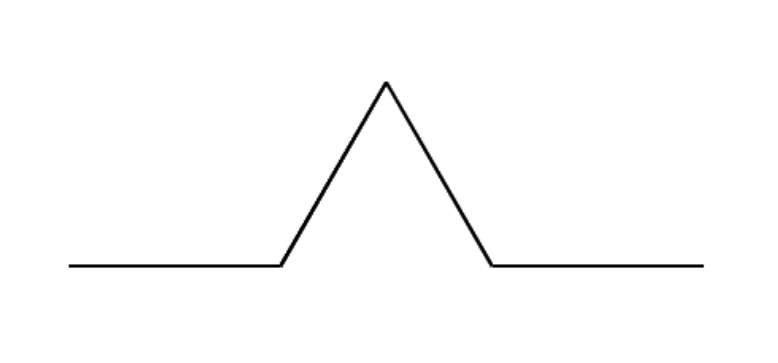
時刻 = 2

# [演習]コッホ雪片

問題.
コッホ雪片は次のベースとモチーフで与えられる。コッホ雪片を実現する開始記号Sと書き換え規則の集合Pを求めよ。ただし、記号「+」「-」の角度をそれぞれ60°, -60° とする。



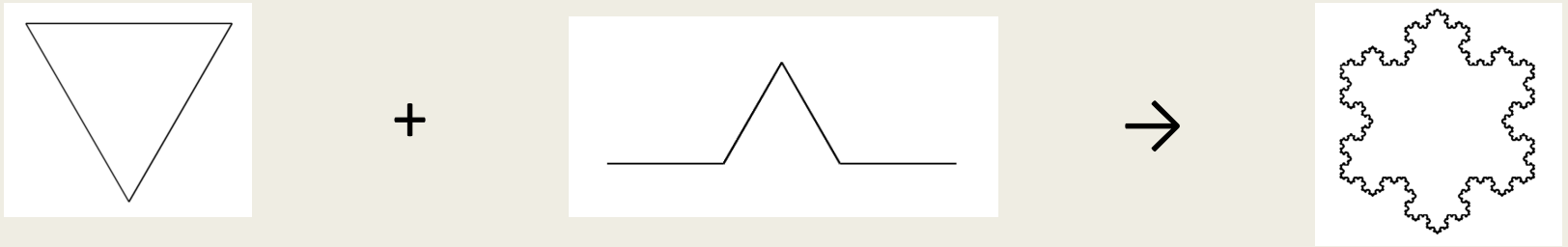
ベース

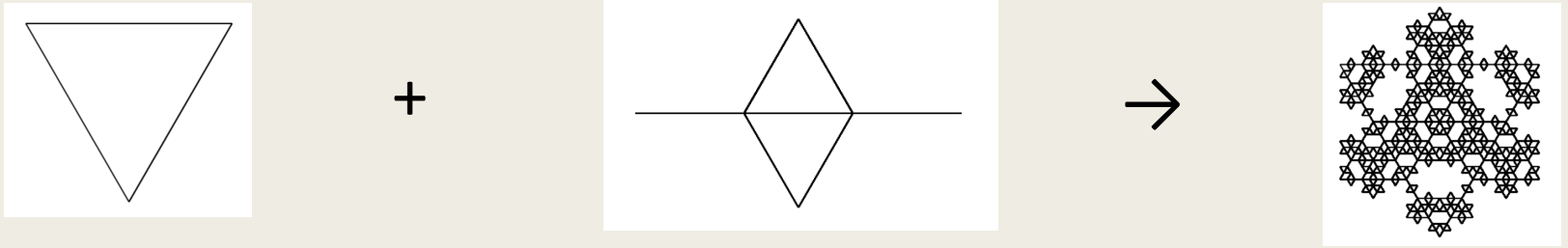


モチーフ

# [演習]

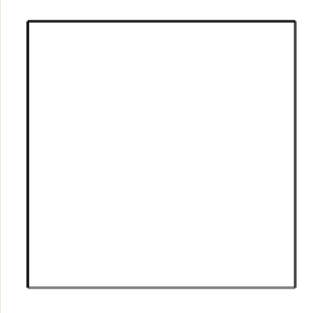
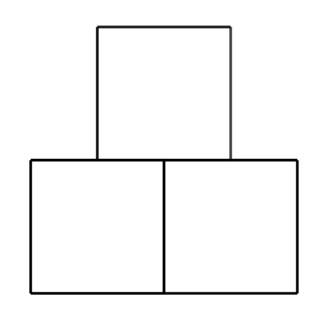
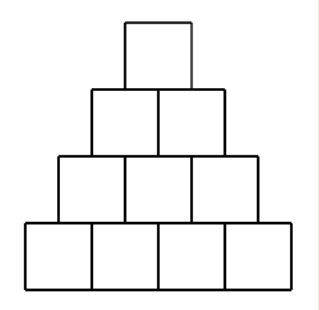
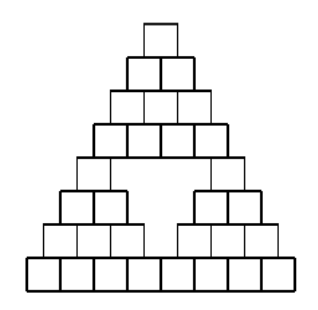
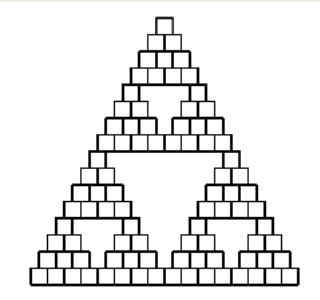
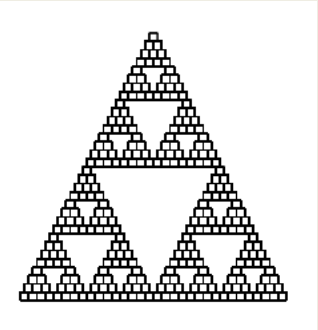
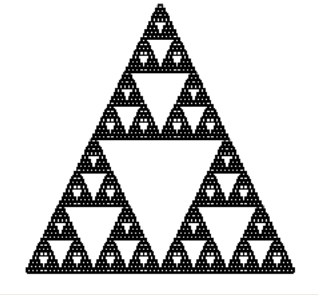
例題.
「chinou03.html」内のプログラムをいじり、「ベース」と「モチーフ」の考え方で作られるフラクタル図形（第1回で作成したフラクタル図形）を書いてみよう。





# アフィン変換

- 図形の1次変換＋平行移動を繰り返すことによってもフラクタル図形をかくことができる。

- 行列とベクトルの演算によって実現できる。

# 線形代数（線形数学I）の復習

- 行列の和　→　成分ごとの和

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 6 & 8 \\ 10 & 12 \end{bmatrix}$$

- 行列の積

　→　左の行列の第 i 行と右の行列の第 j 列の内積

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} = \begin{bmatrix} 1\cdot 5 + 2\cdot 7 & 1\cdot 6 + 2\cdot 8 \\ 3\cdot 5 + 4\cdot 7 & 3\cdot 6 + 4\cdot 8 \end{bmatrix}$$

$$= \begin{bmatrix} 19 & 22 \\ 43 & 50 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x + 2y \\ 3x + 4y \end{bmatrix}$$

# 線形代数（線形数学II？）の復習

■ Aを2次の正方行列とする。

写像$f_A : \mathbb{R}^2 \to \mathbb{R}^2$を以下で定める。

$$f_A(\boldsymbol{x}) = A\boldsymbol{x}$$

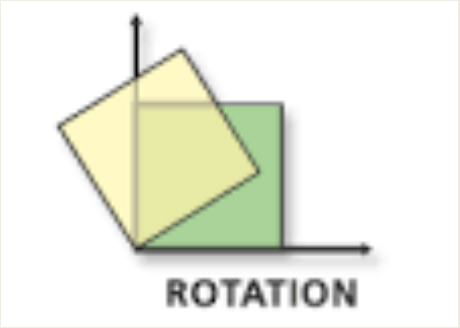
$f_A$を一次変換（線形変換）という。

$$\begin{bmatrix} \alpha & 0 \\ 0 & \beta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \alpha x \\ \beta y \end{bmatrix}$$ （拡大・縮小）

# 線形代数（線形数学II？）の復習

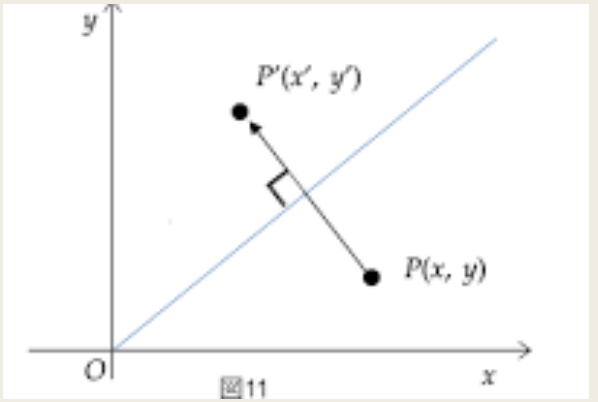
$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\sin\theta \\ x\sin\theta + y\cos\theta \end{bmatrix}$$

（原点まわりに θ 回転）




$$\begin{bmatrix} \cos 2\theta & \sin 2\theta \\ \sin 2\theta & -\cos 2\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} x\cos 2\theta + y\sin 2\theta \\ x\sin 2\theta - y\cos 2\theta \end{bmatrix}$$

（直線 y = (tanθ)x に関する折り返し）



# アフィン変換

- 1次変換と平行移動の組み合わせで記述できる写像をアフィン変換という。
- つまり…

写像$\Phi : \mathbb{R}^2 \to \mathbb{R}^2$がアフィン変換とは

$\exists A$: 2次正方行列, $\exists \boldsymbol{b} \in \mathbb{R}^2$ s.t. $\Phi(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{b}$

例.
次の対応はアフィン変換である。

$$\Phi_1 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\Phi_2 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1/2 \\ 0 \end{bmatrix}$$

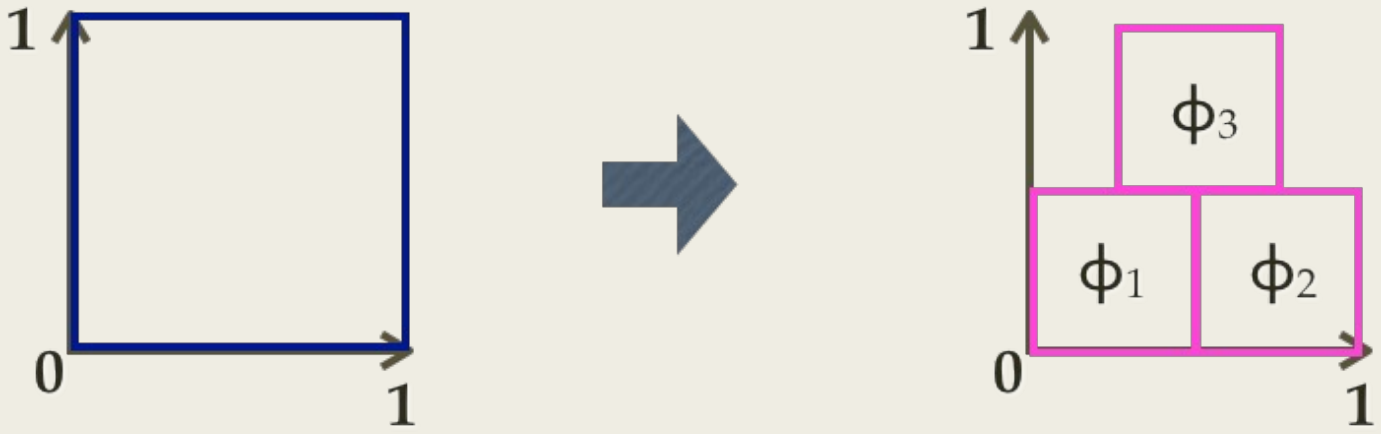
$$\Phi_3 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1/4 \\ 1/2 \end{bmatrix}$$

# アフィン変換

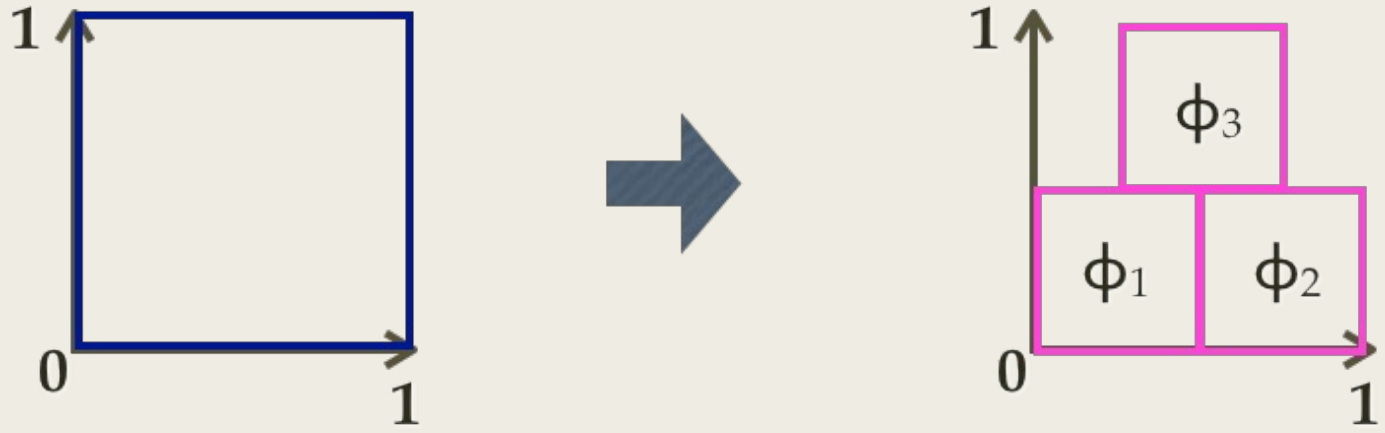
$$\Phi_1 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

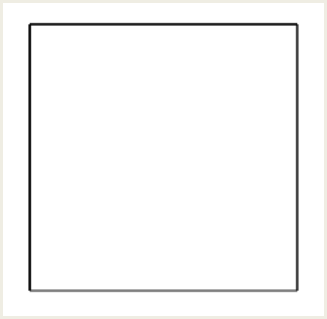
$$\Phi_2 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1/2 \\ 0 \end{bmatrix}$$

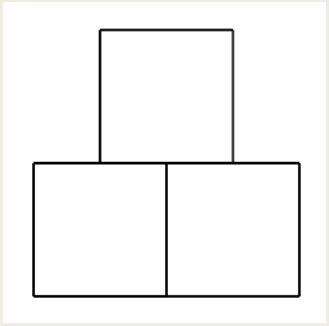
$$\Phi_3 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1/4 \\ 1/2 \end{bmatrix}$$

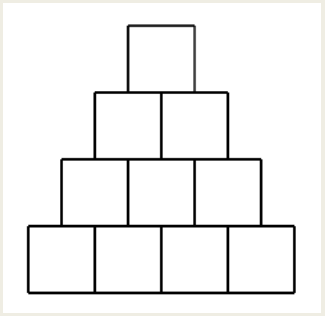


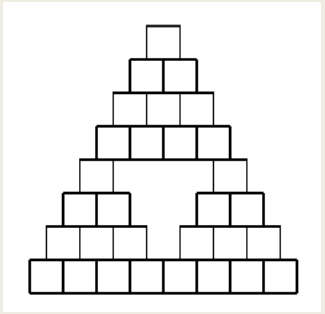
# アフィン変換

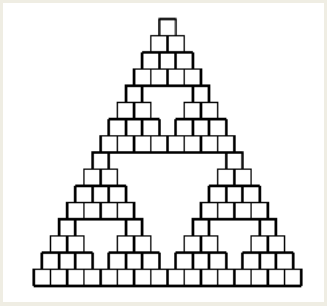


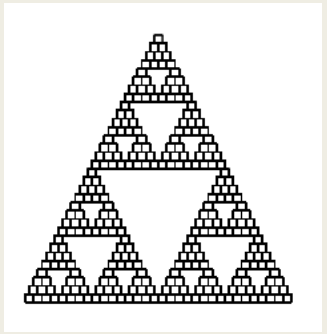



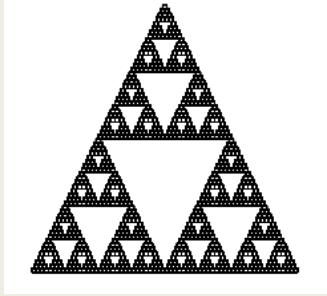












# アフィン変換を実装

- 長さ6の配列として表現...

$$\Phi_1 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$ [0.5, 0, 0, 0.5, 0, 0]

$$\Phi_2 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1/2 \\ 0 \end{bmatrix}$$ [0.5, 0, 0, 0.5, 0.5, 0]

$$\Phi_3 : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 1/4 \\ 1/2 \end{bmatrix}$$ [0.5, 0, 0, 0.5, 0.25, 0.5]

としたいところだが...

- 長さの都合で配列の4番目と5番目の要素は size（基本となる長さ）をかける。

[0.5, 0, 0, 0.5, 0, 0]

[0.5, 0, 0, 0.5, 0.5*size, 0]

[0.5, 0, 0, 0.5, 0.25*size, 0.5*size]

# アフィン変換を実装

$$\Phi : \begin{bmatrix} x \\ y \end{bmatrix} \mapsto \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} = \begin{bmatrix} ax + by + e \\ cx + dy + f \end{bmatrix}$$

```
63      //アフィン変換を行う関数
64      function affine(Phi, Point){ //Phi はアフィン変換を表す長さ6の配列。Point は座標、つまり長さ2の配列。
65          let newX = Phi[0]*Point[0] + Phi[1]*Point[1] + Phi[4];
66          let newY = Phi[2]*Point[0] + Phi[3]*Point[1] + Phi[5];
67          return [newX, newY];
68      }
```

- アフィン変換の実装自体は以上。
- 続いて、四角形に対してアフィン変換を繰り返し適用してフラクタル図形をかく。

```
81      //level = 0 の場合の4点の座標
82      let P1 = [0,0];
83      let P2 = [0,size];
84      let P3 = [size, size];
85      let P4 = [size,0];
```

# アフィン変換によるフラクタル図形

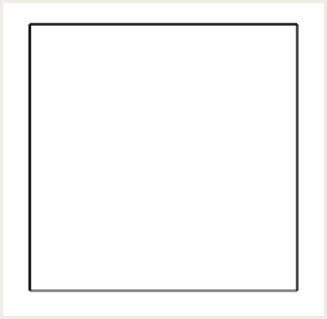
```
87      function start(e){
88          let level = Number(e.id[0]);
89          go(P1, P2, P3, P4, level);
90      }
91
92      function go(P1, P2, P3, P4, level){
93          if(level == 0){
94              turtle.draw_rectangle(P1, P2, P3, P4);
95          } else {
96              for(let i=0; i<Phi.length; i++){
97              go(affine(Phi[i], P1), affine(Phi[i], P2), affine(Phi[i], P3), affine(Phi[i], P4), level-1);
98              }
99          }
```

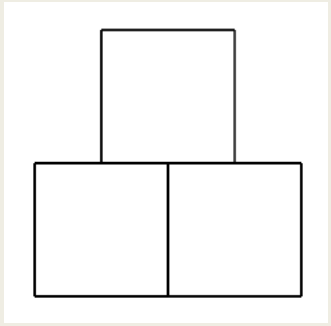
- ボタンをクリックすると start 関数が発動。
- 88行目、押したボタンに応じたレベル数を取得。
- 89行目、go関数を呼び出す。
- 93, 94行目、levelが0のときは四角形をかくだけ。
- draw_rectangle は引数の4頂点を通る四角形をかくメソッド。
- 95行目以降、level が0より大きい場合、4頂点にアフィン変換を施したあと、その4頂点に対してレベルをひとつ落として再び go 関数を（再帰的に）呼び出す。

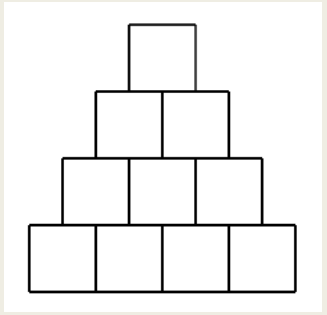
# シェルピンスキーのギャスケット

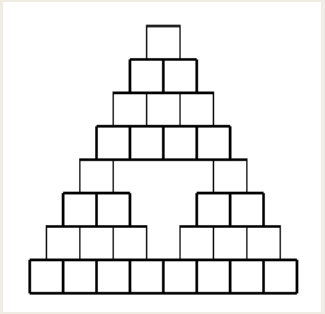
| a | b | c | d | e | f |
|---|---|---|---|---|---|
| 0.5 | 0 | 0 | 0.5 | 0 | 0 |
| 0.5 | 0 | 0 | 0.5 | 0.5 | 0 |
| 0.5 | 0 | 0 | 0.5 | 0.25 | 0.5 |

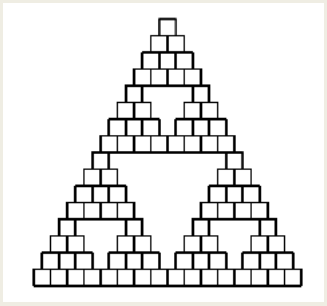
長さ6の配列を要素に持つ配列を用意する。この例ではアフィン変換を表す写像は3つ。

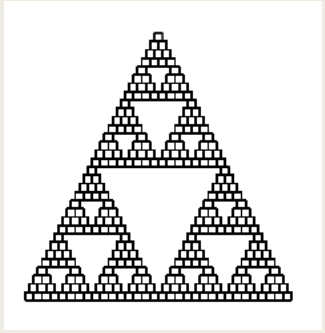


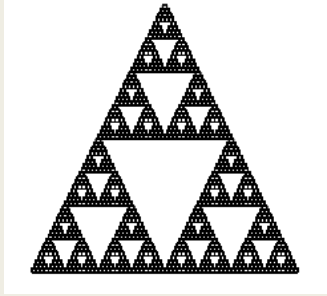






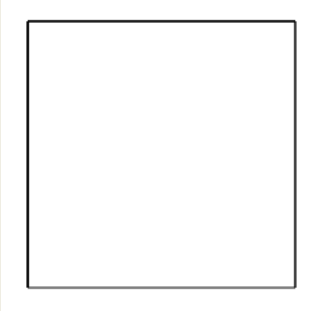


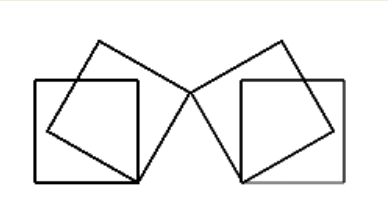


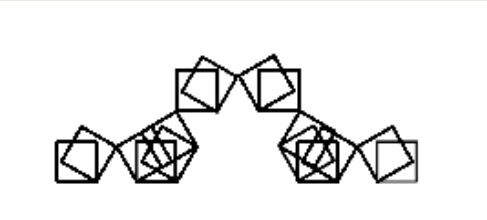


# コッホ曲線

| a | b | c | d | e | f |
|---|---|---|---|---|---|
| 0.33 | 0 | 0 | 0.33 | 0 | 0 |
| 0.166 | -0.29 | 0.29 | 0.166 | 0.33 | 0 |
| 0.166 | 0.29 | -0.29 | 0.166 | 0.5 | 0.29 |
| 0.33 | 0 | 0 | 0.33 | 0.66 | 0 |

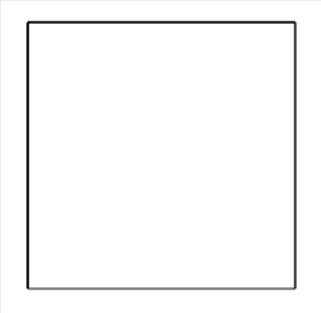


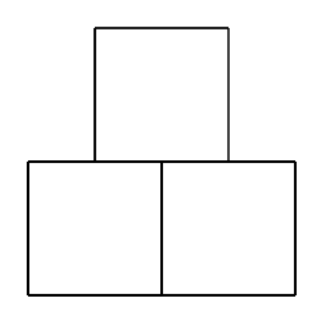


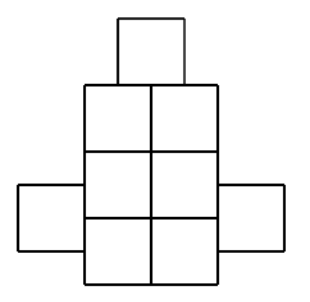


# Crystallike structure

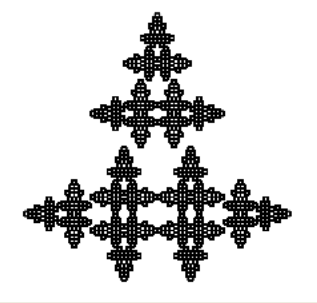
| a | b | c | d | e | f |
|---|---|---|---|---|---|
| 0 | -0.5 | 0.5 | 0 | 0.5 | 0 |
| 0 | 0.5 | -0.5 | 0 | 0.5 | 0.5 |
| 0.5 | 0 | 0 | 0.5 | 0.25 | 0.5 |







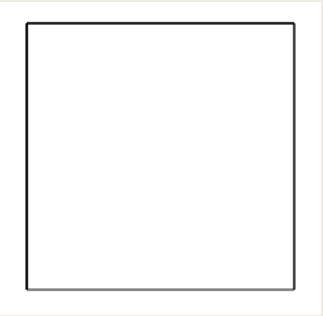
...

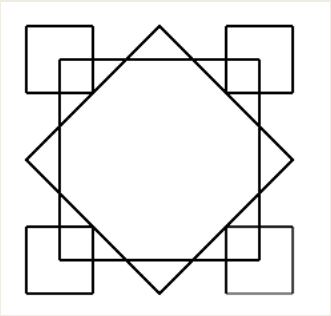


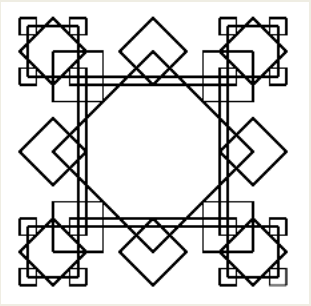
level 6

# Snowflake-like fractal

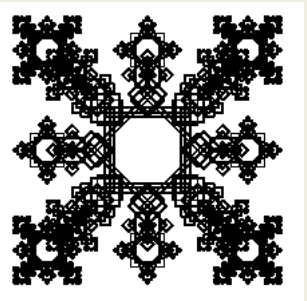
| a | b | c | d | e | f |
|---|---|---|---|---|---|
| 0.75 | 0 | 0 | 0.75 | 0.125 | 0.125 |
| 0.5 | -0.5 | 0.5 | 0.5 | 0.5 | 0 |
| 0.25 | 0 | 0 | 0.25 | 0 | 0.75 |
| 0.25 | 0 | 0 | 0.25 | 0.75 | 0.75 |
| 0.25 | 0 | 0 | 0.25 | 0 | 0 |
| 0.25 | 0 | 0 | 0.25 | 0.75 | 0 |







...

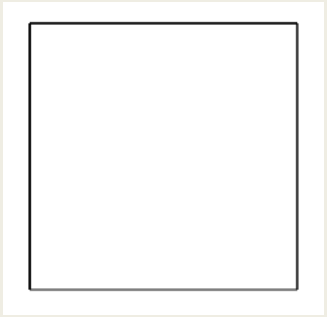


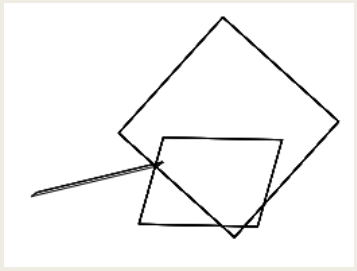
level 4

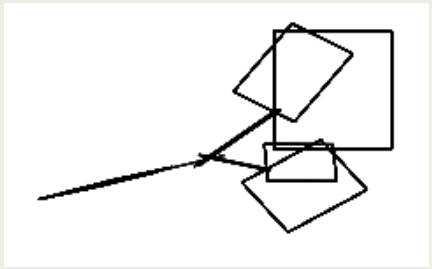
アフィン写像が6個なので重い。JavaScript では level 4 が限界。

# Twig （小枝）

| a | b | c | d | e | f |
|---|---|---|---|---|---|
| 0.387 | 0.43 | 0.43 | -0.387 | 0.256 | 0.522 |
| 0.441 | -0.091 | -0.009 | -0.322 | 0.4219 | 0.5059 |
| -0.468 | 0.02 | -0.113 | 0.015 | 0.4 | 0.4 |







…

level 6

# [演習]

## 演習問題.

「chinou04.html」内のプログラムをいじってアフィン変換によるフラクタル図形を書いてみよう。

- 前のスライドにあるフラクタル図形であなたが好きなものをひとつ選び、アフィン変換の情報（長さ6の配列の配列）を入力してみよう。
- その際、配列の（0番目から数えて）4, 5番目の数字は *size にするのを忘れないこと。

```
70      //アフィン変換を行う行列とベクトルの配列（長さ6の配列で表現）
71      let Phi = [[0.5, 0, 0, 0.5, 0, 0], [0.5, 0, 0, 0.5, 0.5*size, 0],
72                 [0.5, 0, 0, 0.5, 0.25*size, 0.5*size]];  //シェルピンスキーのギャスケット
```

## 発展的な課題.

- 自分で試行錯誤しながら色々なアフィン変換を試してみよう。
- 最初の図形を三角形、五角形、直線などに変更したらどうなるか？（適宜、プログラムを変更する必要がある。）