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

授業資料について

- 授業支援システムに資料を置いている。

2023年度後期担当科目

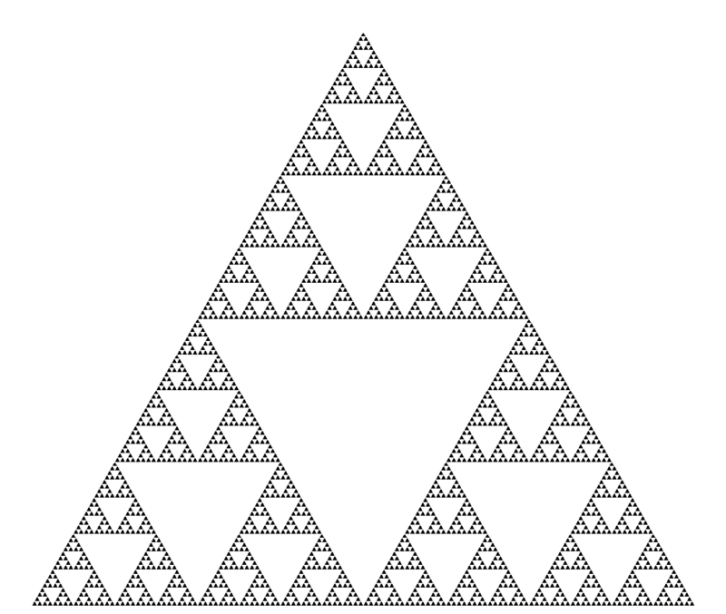
	月曜	火曜	水曜	木曜	金曜
1限		情報数学	C演習I		
2限		データベースシステム	C演習I		
3限	プログラミング言語論		（オフィスアワー）	知能情報システム論	
4限				情報セミナール	
5限					

日付	回	スライド	リンク
10/19	第1回	資料	chinou01, chinou02
10/26	第2回		
12/21	第3回		

フラクタルとは

フラクタル幾何学（1975年）
→「自己相似性」をもつ図形
一部を拡大しても全体と変わらない

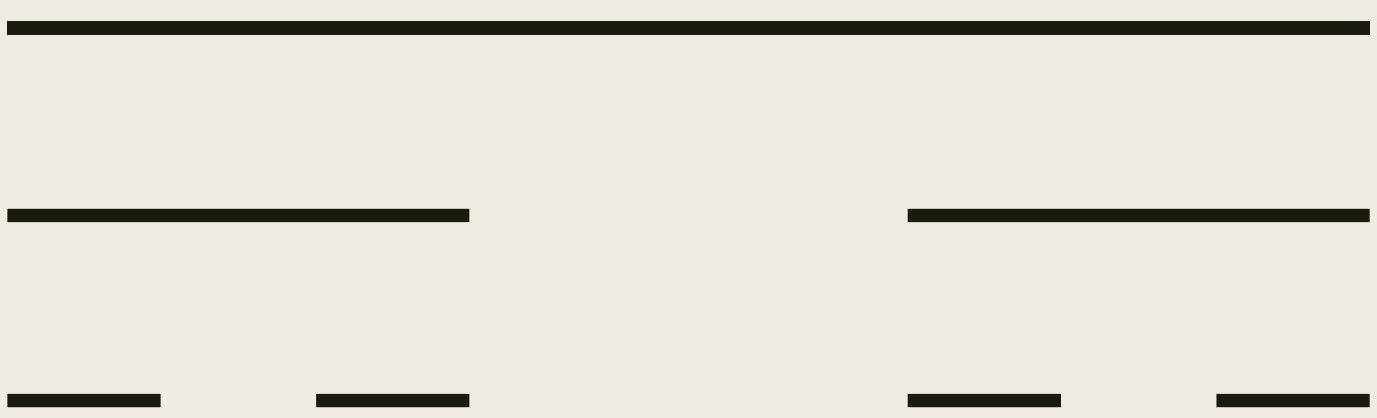
動機：
複雑な形や構造がもつ秩序や規則を調べたい
（海岸線、樹木、山、雲、株価）



マンデルブロ(1924-2010)

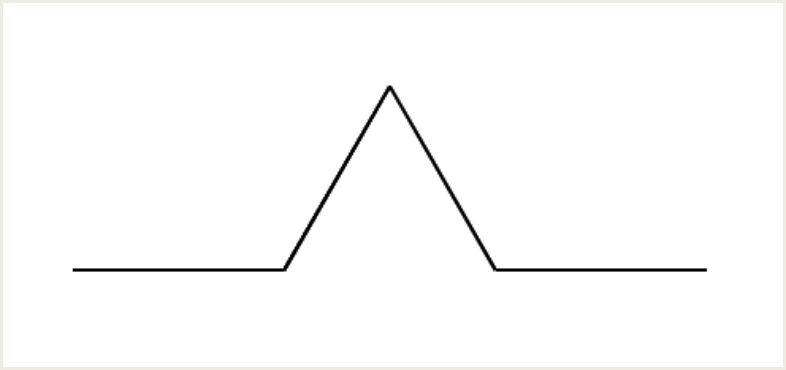
数学的なフラクタル

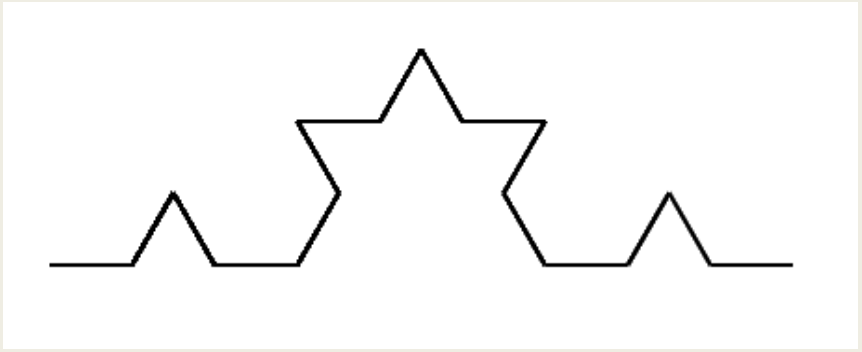
カントール集合 (Cantor Set)
①線分を3等分し中央部分を取り除く
②各線分に対して①の手続きを繰り返す

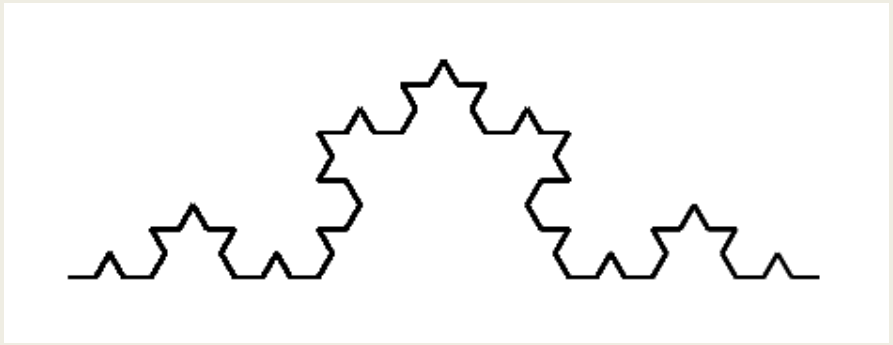


数学的なフラクタル

コッホ曲線 (Koch Curve)
①線分の中央 1/3 を折り曲げる
②各線分に対して①の手続きを繰り返す

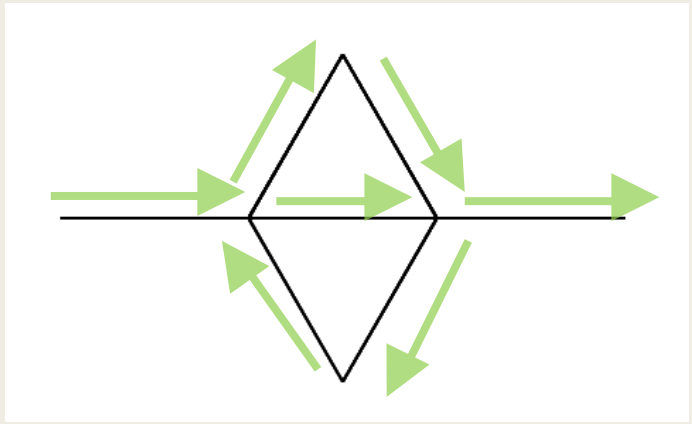


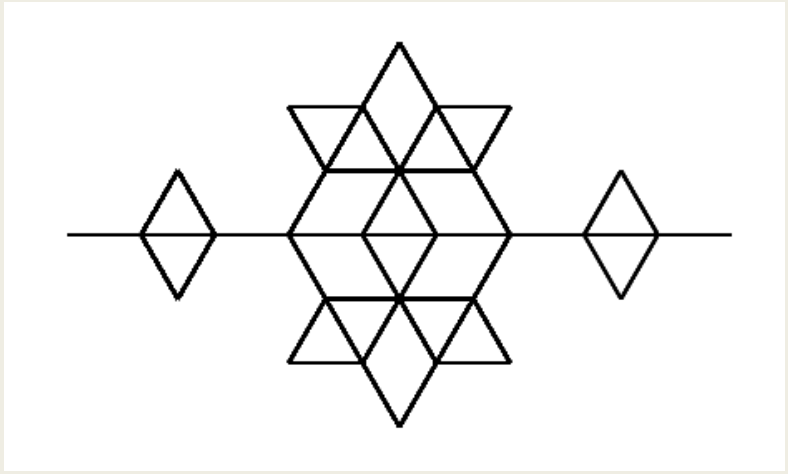


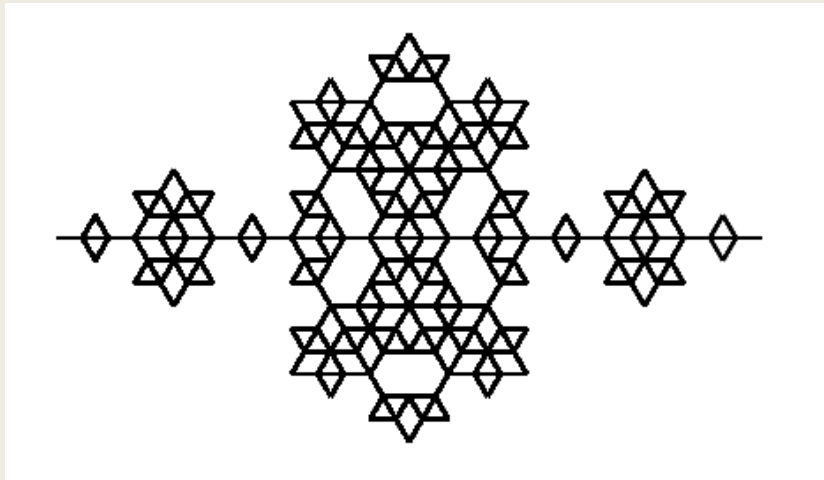


数学的なフラクタル

ペアノ曲線 (Peano Curve)
①線分を一筆書きで置き換える
②各線分に対して①の手続きを繰り返す





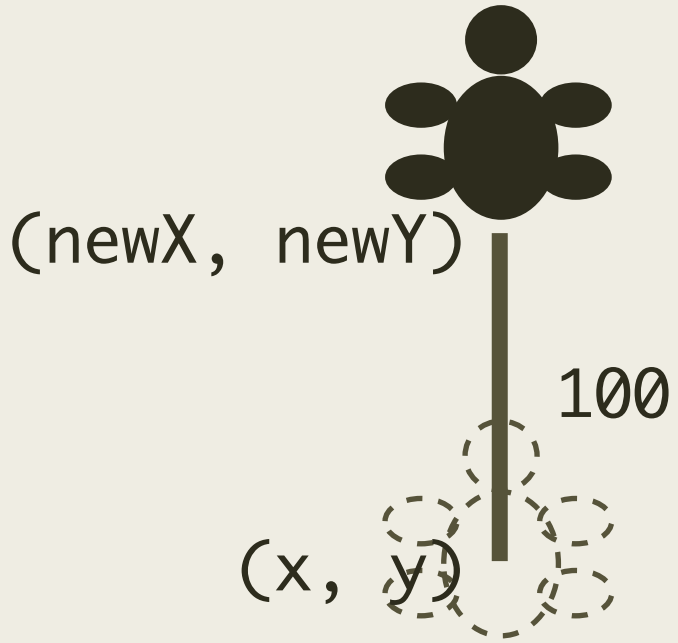


数学的なフラクタル図形をかく

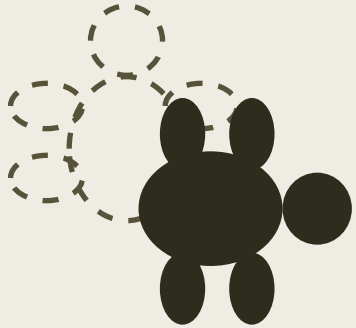
コッホ曲線をプログラムでかく

曲線の各点を計算して線を結ぶのは大変
→タートルグラフィックス(Turtle Graphics)を用いる

forward(100)



turn(-90)



```
forward(100);
turn(60);
forward(100);
turn(-120);
forward(100);
turn(60);
forward(100);
```

タートルグラフィックス

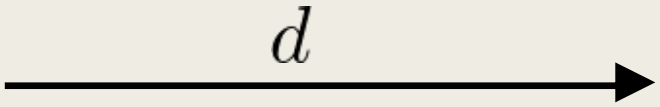
```
class Turtle{     （コンストラクタなどは省略）
   Graphics g;
   double x, y; // 亀の位置の座標を表す変数
   int a; // 亀の向きを表す変数（角度）

   void forward(double d){
      double newX, newY;
        (x,y)から a 方向に d 進んだ座標を
                                (newX, newY)とする
      g に(x,y)から(newX, newY)へ直線をひく
      x = newX; y = newY; //亀の位置を更新
   }

   void turn(angle a){
      angle += a;
   }
}
```

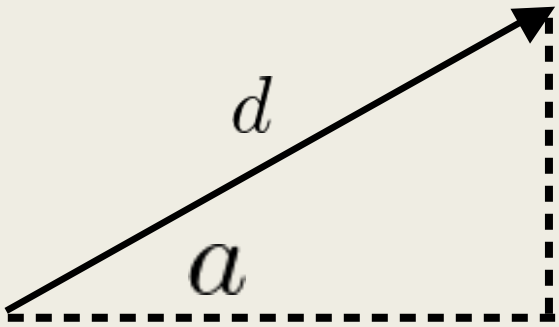
newX, newY を求める

三角関数とベクトルを使って考える



0° の方向に d 進む

$$\vec{v} = (d, 0)$$



a° の方向に d 進む

$$\mathrm{newX} = d\cos a$$

$$\mathrm{newY} = d\sin a$$

タートルグラフィックス

```
class Turtle{     （コンストラクタなどは省略）
   Graphics g;
   double x, y; // 亀の位置を表す変数
   double a; // 亀の向きを表す変数

   void forward(double d){
      double newX, newY;
      newX = d*cos(angle);
      newY = d*sin(angle);
      g に(x,y)から(newX, newY)へ直線をひく
      x = newX; y = newY; //亀の位置を更新
   }

   void turn(double a){
      angle += a;
   }
}
```

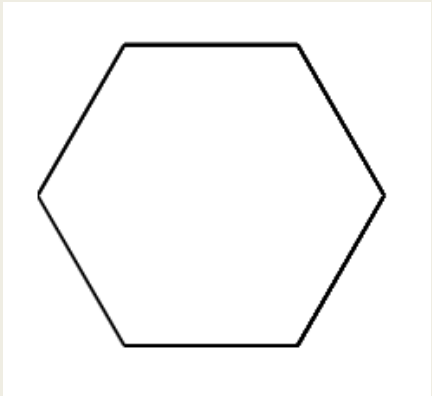
タートルグラフィックスで簡単な絵を書いてみる

次のプログラムはどのような図形をえがくか？

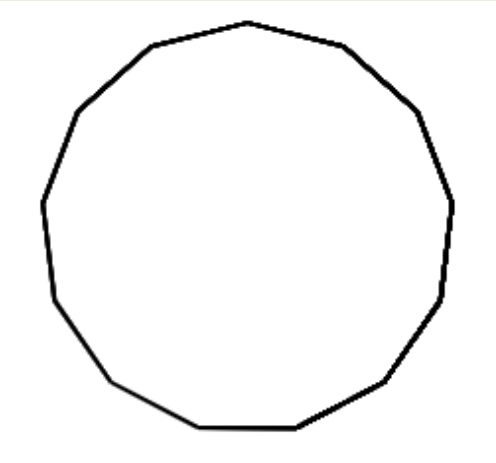
```
forward(100);
turn(60);
forward(100);
turn(60);
forward(100);
turn(60);
forward(100);
turn(60);
forward(100);
turn(60);
```

=

```
for(int i=0; i<6; i++){
    forward(100);
    turn(60);
}
```



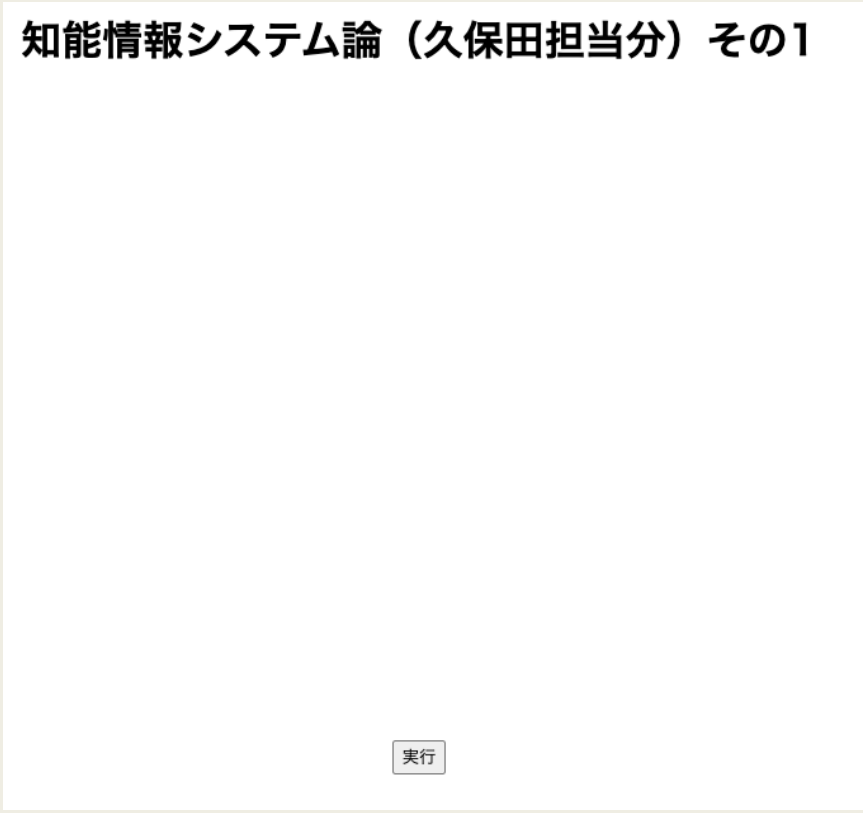
```
for(int i=0; i<13; i++){
    forward(50);
    turn(360/13);
}
```

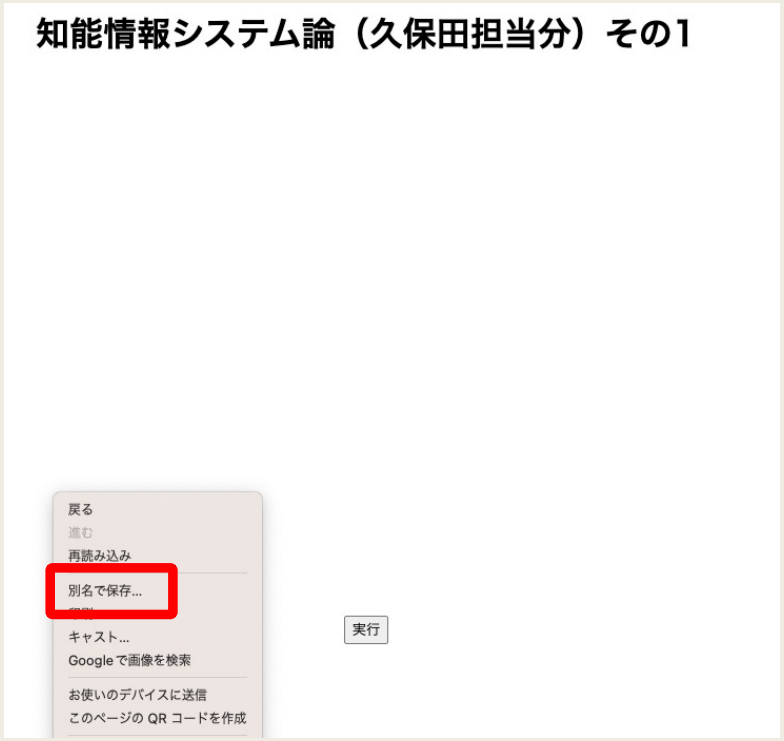


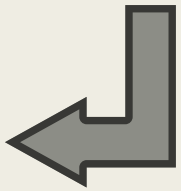
HTMLファイルのダウンロード

- 授業支援サイトに JavaScript によるタートルグラフィックスのプログラム（chinou01.html など）を用意した。

日付	回	スライド	リンク
10/19	第1回	資料	chinou01, chinou02
10/26	第2回		
12/21	第3回		



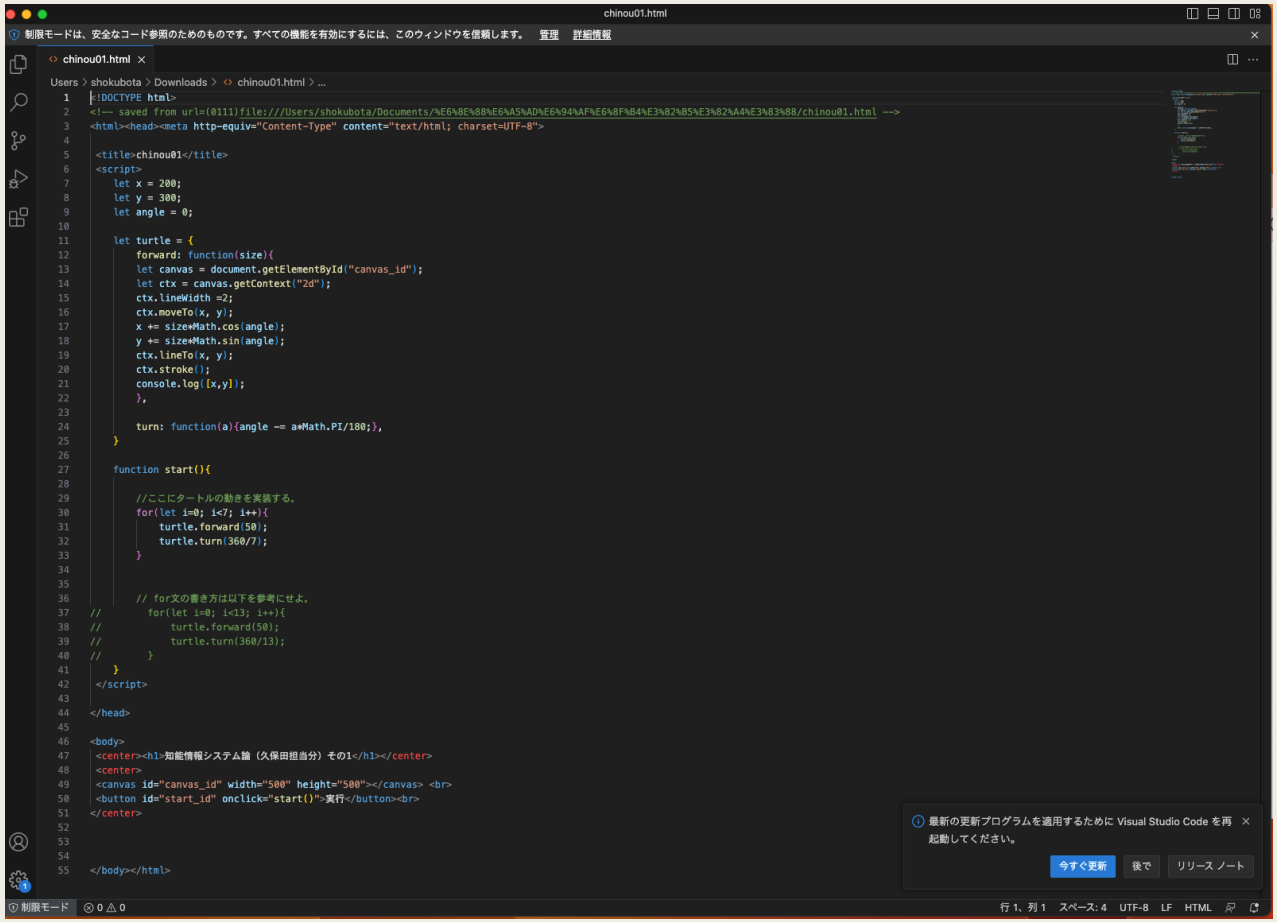




ページの端っこあたりを
右クリックして
htmlファイルを保存

HTMLファイルのダウンロード

- ダウンロードしたhtmlファイルは「ダブルクリック」で開くと学習支援サイトと同じページが開かれるはず。
- 右クリックをして「このアプリケーションで開く → 好きなコードエディタ」を選択するとソースプログラムが開かれる。



JavaScript によるタートルグラフィクス

- ダウンロードした html ファイルは JavaScript によるタートルグラフィックスのプログラムである。
- JavaScript を知らなくてもプログラムいじれるように補足。

```
42  <body>
43   <center><h1>知能情報システム論（久保田担当分）その1</h1></center>
44   <center>
45   <canvas id="canvas_id" width="500" height="500"></canvas> <br>
46   <button id="start_id" onclick="start()">実行</button><br>
47  </center>
48  </body>
```

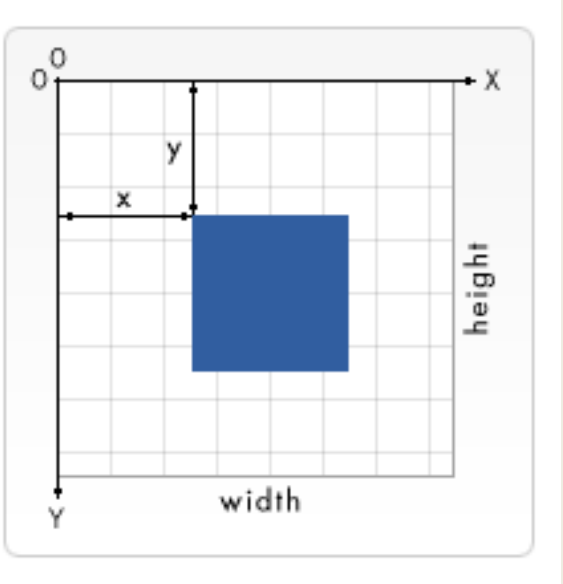
- 42行目以降は body 部（ページの概形を作る）
- 45行目で図形描画のためのエリア（canvas）を宣言している。
  - サイズは横と縦がそれぞれ500ずつ。
  - width と height の値をいじるとスペースを広くとれる。

JavaScript によるタートルグラフィクス

- 12~26行目でタートルオブジェクトを宣言。
- 13行目で forward メソッドを定義。
- 25行目で turn メソッドを定義。
- turtle.forward(100) のように使う。

```
8   let x = 150;
9   let y = 250;
10  let angle = 0;
11
12  let turtle = {
13      forward: function(size){
14      let canvas = document.getElementById("canvas_id");
15      let ctx = canvas.getContext("2d");
16      ctx.lineWidth =2;
17      ctx.moveTo(x, y);
18      x += size*Math.cos(angle);
19      y += size*Math.sin(angle);
20      ctx.lineTo(x, y);
21      ctx.stroke();
22      console.log([x,y]);
23      },
24
25      turn: function(a){angle -= a*Math.PI/180;},
26  }
```

- 8, 9行目はタートルの初期位置を表す。
  - 図形がキャンバスから外れたら初期位置の座標を変更しよう。
  - ただし、y軸は「下向き」であることに注意。
- 10行目はタートルの向きを表す。
  - 0°はx軸と同じ向きである。



JavaScript によるタートルグラフィクス

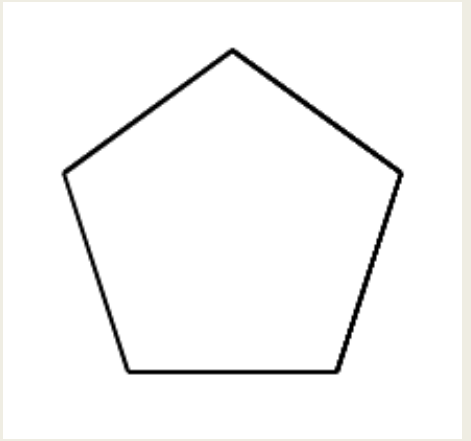
```
28      function start(){
29
30          //ここにタートルの動きを実装する。
31
32          // for文の書き方は以下を参考にせよ。
33  //          for(let i=0; i<13; i++){
34  //              turtle.forward(50);
35  //              turtle.turn(360/13);
36  //          }
37      }
```

- 28行目以降に実際のタートルの動きを入力する。
- 「実行」ボタンを押すと start 関数が発動する。

練習.
31行目に
　　turtle.forward(100);
と入力して上書き保存した後、
htmlファイルを ダブルクリック
によって開き、「実行」ボタン
を押してみよう。

[演習] 簡単な絵を書いてみる

例題.
「chinou01.html」内のプログラムを修正し、右のような正五角形を書いてみよう。for 文を使うこと。



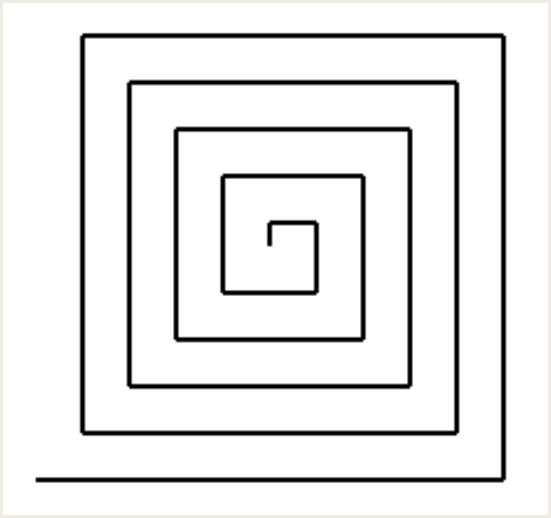
解答例.

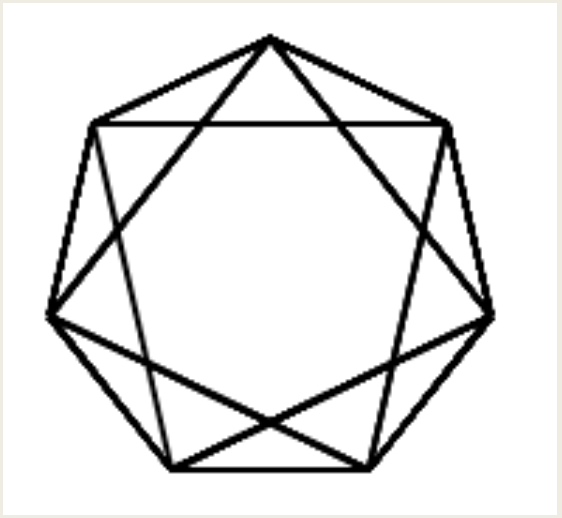
```
28      function start(){
29          //ここにタートルの動きを実装する。
30          for(let i=0; i<5; i++){
31              turtle.forward(100);
32              turtle.turn(360/5);
33          }
```

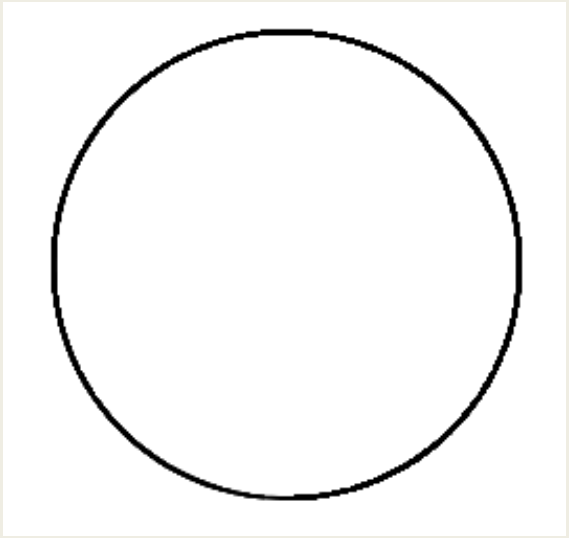
例題.
右のような図形を書くためには、どのようにすればよいだろうか？

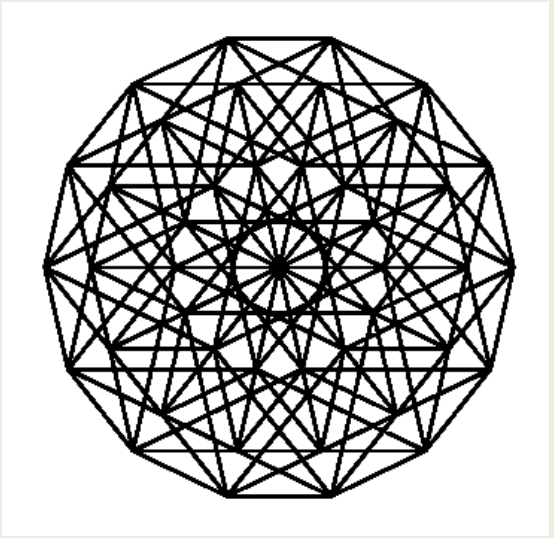
メモ欄

[演習] タートルグラフィックスで遊んでみよう





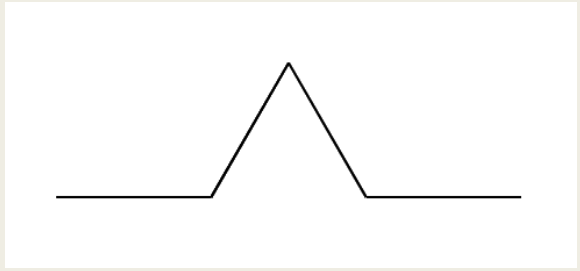


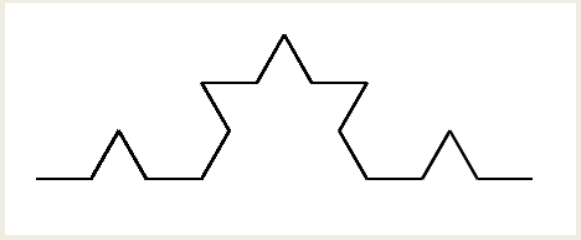


数学的なフラクタル図形をかく

- 再帰的な関数を使うとコッホ曲線がかける。
- 「chinou02.html」を使う。
- 「ベース」と「モチーフ」は後で解説。
- 32行目。ボタンを押すとstart 関数が発動。レベル数を取得して関数 go を呼び出す。
- 38行目。関数 go でフラクタルを書いている。

```
31      //ベース
32      function start(e){
33          let level = Number(e.id[0])
34          go(size, level);
35      }
36
37      //モチーフ
38      function go(size, level){
39          if(level == 0){
40              turtle.forward(size);
41          } else {
42              go(size/3, level - 1)
43              turtle.turn(60)
44              go(size/3, level - 1)
45              turtle.turn(-120);
46              go(size/3, level - 1)
47              turtle.turn(60);
48              go(size/3, level - 1)
49          }
50      }
```



変数の補足

- 再帰的な関数を使うとコッホ曲線がかける。
- 「chinou02.html」を開く。
- 「ベース」と「モチーフ」は後で解説。
- 32行目。ボタンを押すと start 関数が発動。レベル数を取得して関数 go を呼び出す。
- 38行目。関数 go でフラクタルを書いている。
- 10行目。変数 size を追加。
- 変数 x, y, angle は「chinou01.html」と同じ。

```
31      //ベース
32      function start(e){
33          let level = Number(e.id[0])
34          go(size, level);
35      }
36
37      //モチーフ
38      function go(size, level){
39          if(level == 0){
40              turtle.forward(size);
41          } else {
42              go(size/3, level - 1)
43              turtle.turn(60)
44              go(size/3, level - 1)
45              turtle.turn(-120);
46              go(size/3, level - 1)
47              turtle.turn(60);
48              go(size/3, level - 1)
49          }
50      }
```

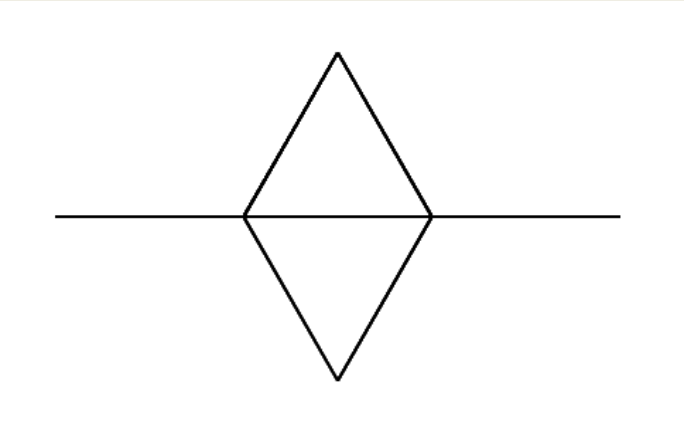
```
 8      let x = 50;
 9      let y = 150;
10      let size = 300; //基本の長さ
11      let angle = 0; //x軸の正の方向を0として、今どの角度を見ているかを表す変数
```

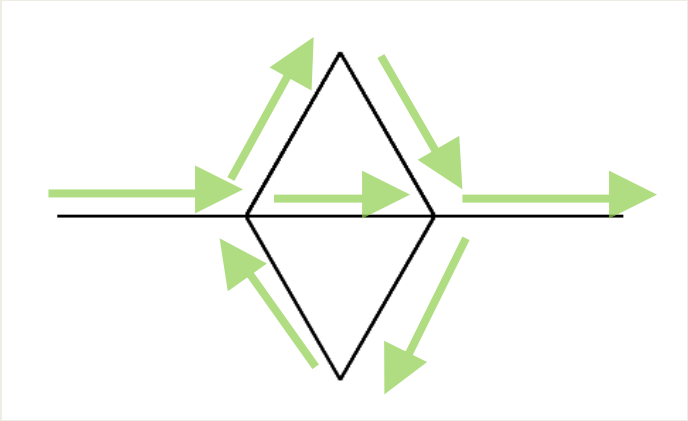
[演習]ペアノ曲線

例題.
「chinou02.html」内のプログラム（モチーフ部）を修正し、ペアノ曲線によるフラクタル図形を描いてみよう。

- level 1 で右の図形が出てくれば正解。
- 角度は±60°または± 120°

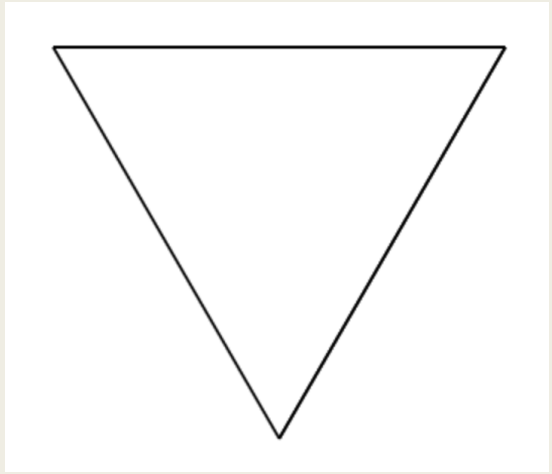
メモ欄



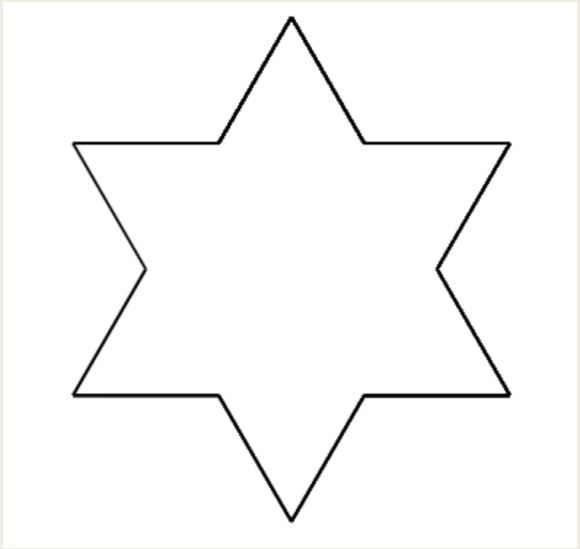


コッホ曲線をつなげてみよう（コッホ雪片）

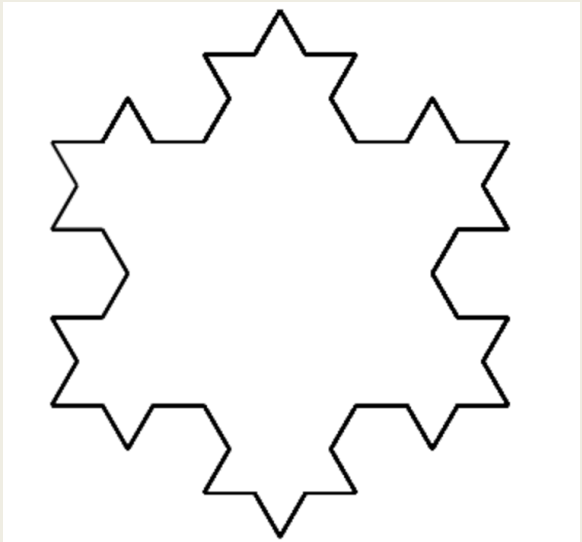
```
go(size, level);
turtle.turn(-120);
go(size, level);
turtle.turn(-120);
go(size, level);
```



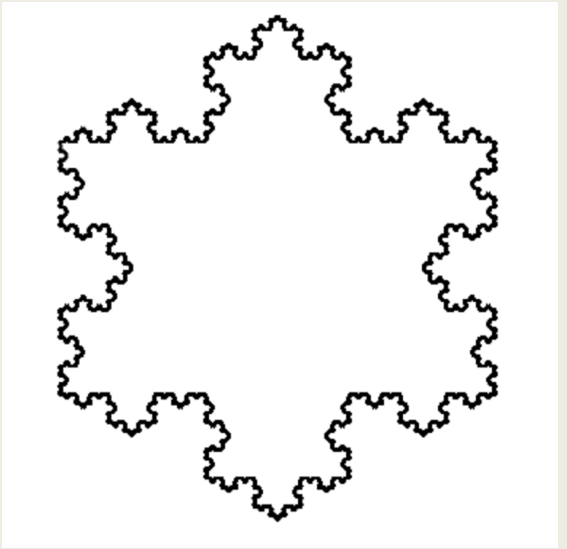
Level = 0



Level = 1



Level = 2



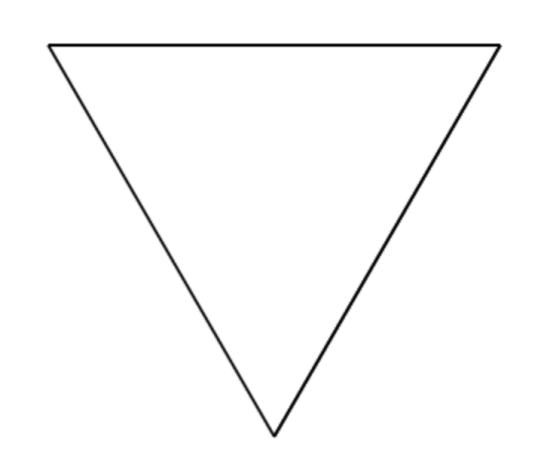
Level = 4

ベースとモチーフ

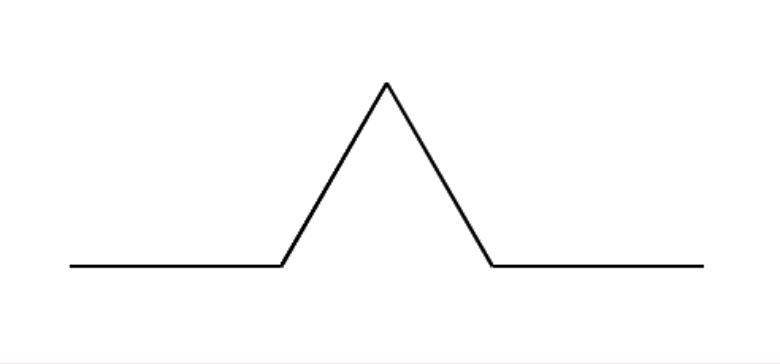
コッホ雪片は

- 三角形をベースにし、
- 各直線をコッホ曲線で置き換える

ことで得られた。



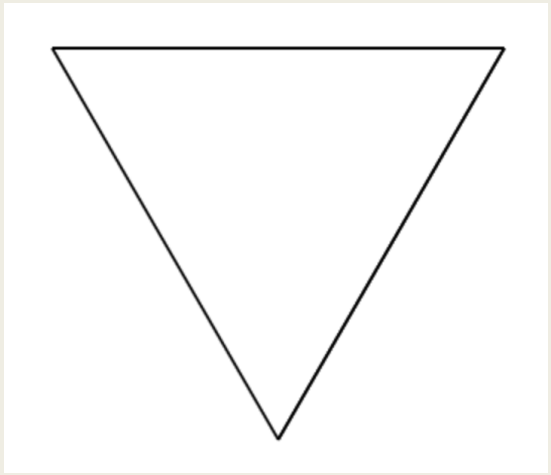
ベース



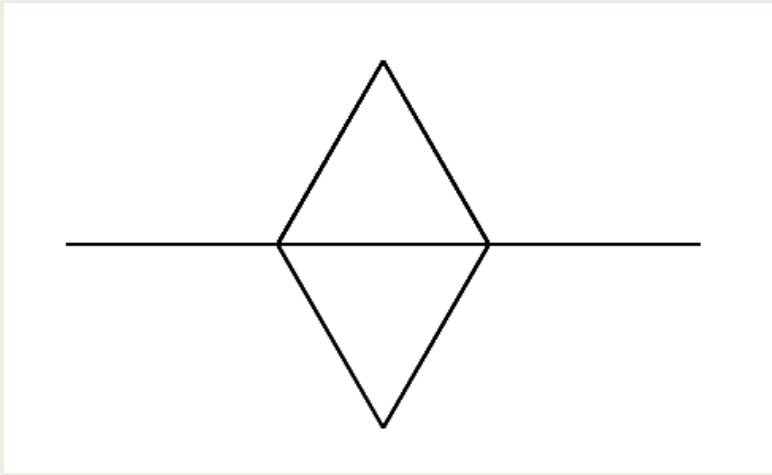
モチーフ

- ベースとモチーフを変えれば色々な図形がかける

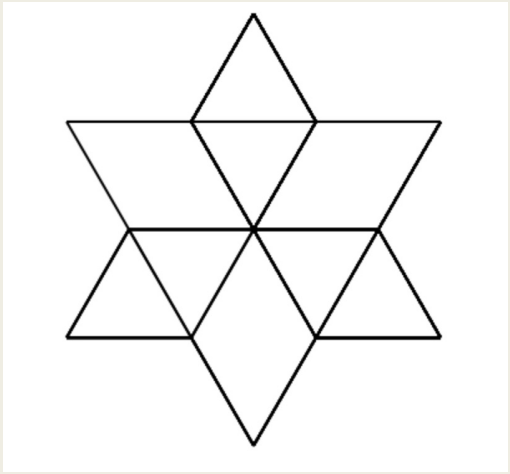
モチーフを変える



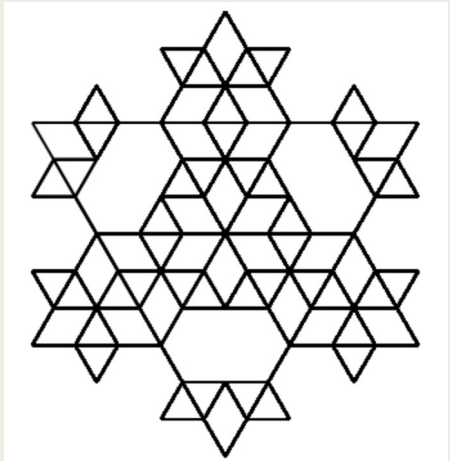
ベース
（Level = 0）



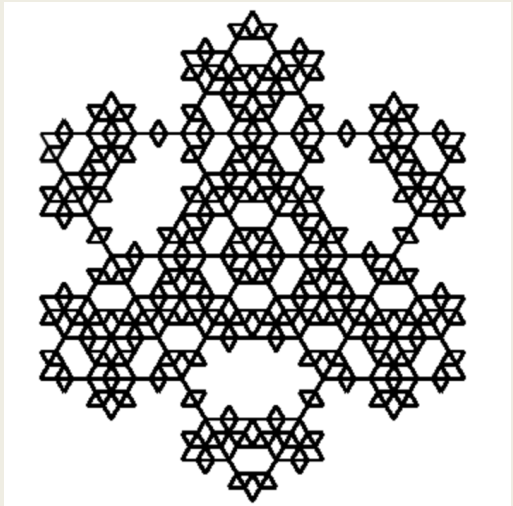
モチーフ



Level = 1



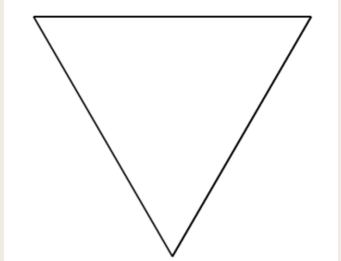
Level = 2



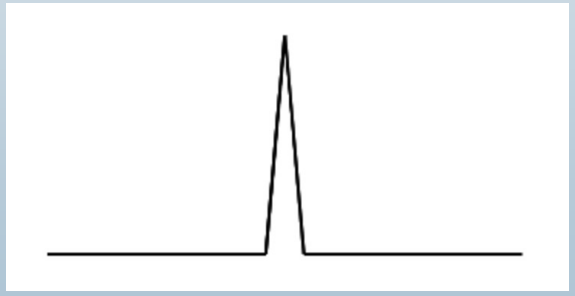
Level = 3

[演習]モチーフを考えよう

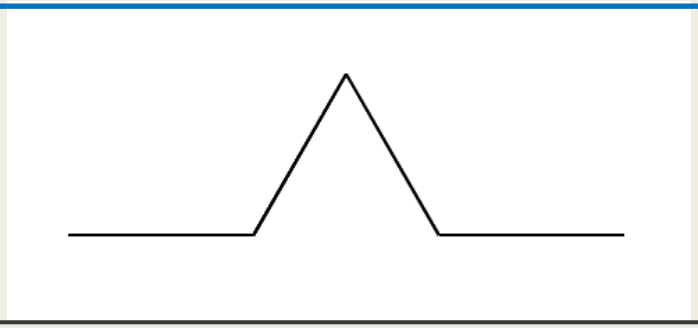
例題.
「chinou02.html」内のプログラムのベース部とモチーフ部をそれぞれ以下のように入力し、コッホ雪片を書いてみよう。次に、あなたが考えたモチーフでフラクタル図形を作成してみよう。



```
go(size, level);
turtle.turn(-120);
go(size, level);
turtle.turn(-120);
go(size, level);
```

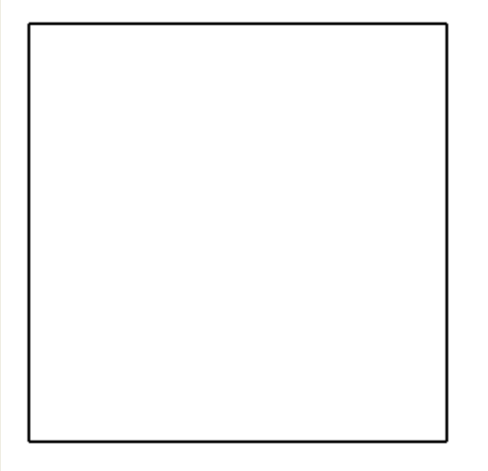


モチーフのアイディア



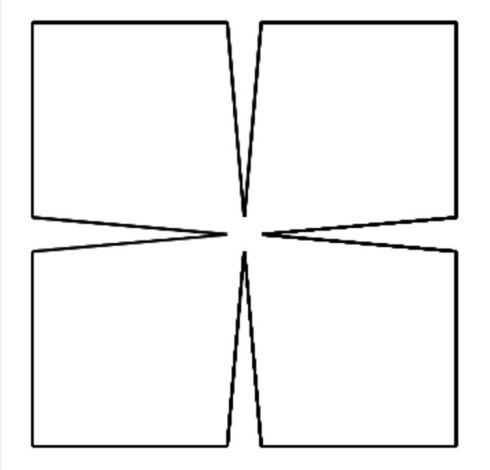
```
function go(size, level){
    if (level == 0){
        turtle.forward(size);
        return;
    } else {
        go(size/3, lebel - 1);
        turtle.turn(60);
        go(size/3, lebel - 1);
        turtle.turn(-120);
        go(size/3, lebel - 1);
        turtle.turn(60);
        go(size/3, lebel - 1);
    }
}
```

ベースも変える

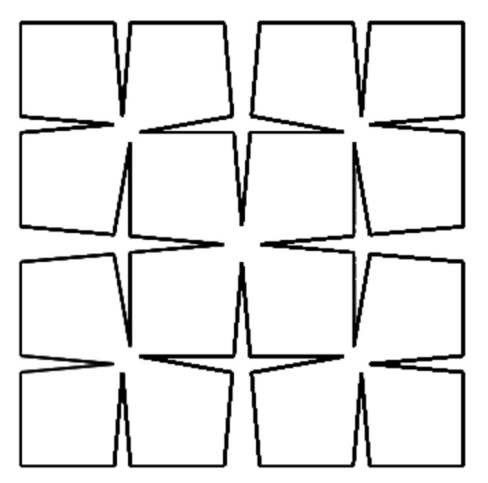


ベース
（Level = 0）

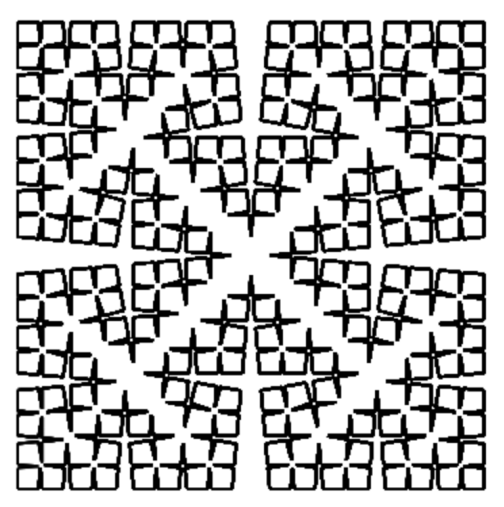
モチーフ



Level = 1

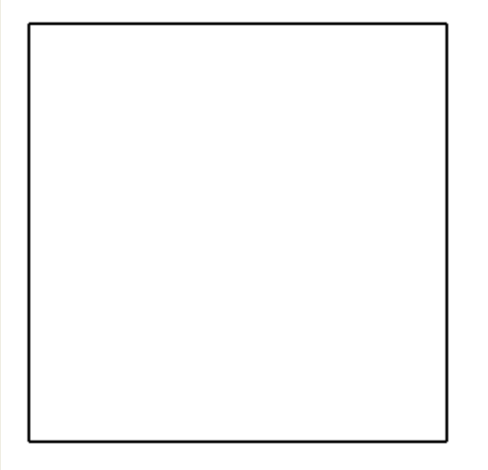


Level = 2



Level = 4

[演習]ベースとモチーフを両方考えよう



```
for(let i=0; i<4; i++){
  go(size, level);
  turn(90);
}
```

繰り返し回数と角度を調整すれば好きな正多角形をベースにできる

```
function go(size, level){
    if (level == 0){
        turtle.forward(size);
        return;
    } else {
        go(size/2.1, lebel - 1);
        turtle.turn(85);
        go(size/2.1, lebel - 1);
        turtle.turn(-170);
        go(size/2.1, lebel - 1);
        turtle.turn(85);
        go(size/2.1, lebel - 1);
    }
}
```

```
function go(size, level){
    if (level == 0){
        turtle.forward(size);
        return;
    } else {
        go(size/2.1, lebel - 1);
        turtle.turn(85);
        go(size/2.1, lebel - 1);
        turtle.turn(-170);
        go(size/2.1, lebel - 1);
        turtle.turn(85);
        go(size/2.1, lebel - 1);
    }
}
```

←モチーフの入力
すらめんどくさい

フラクタルの他の作り方は？

↓

L-system を使った方法
(形式文法を使った方法)

L-system

- L-system を使うと次のような図形がかける。