

# 随意課題 (≠1/4) 3D図形言語を利用して独創的な3Dモデルを作成せよ

**提出物：**作成した3Dモデルの**ソースコード**，**画像**，**説明**，**アンケート**

**提出先：**kfurukaw@kuis.kyoto-u.ac.jp (音声メディア分野研究室 M2 古川)

※ 3Dモデルの提出作品はTAが3Dプリンタで造型して提出者に返却します！

※ アンケート (配布ファイル中の questionnaire.pdf) に御協力ください！

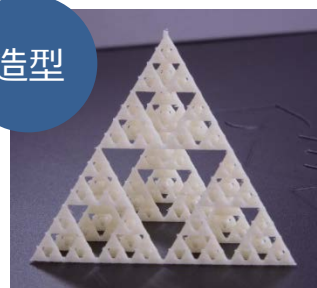
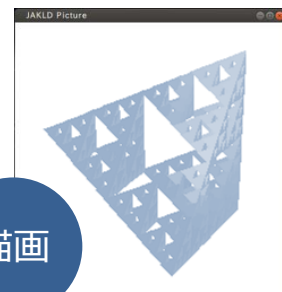
```
(define (painter:tetrahedron attribute origin size)
  (let ((s0 (/ 1.0 2.0))
        (s1 (/ 1.0 (* 2.0 (sqrt 2.0)))))
    (let ((ps (list (list s0 0.0 s1)
                    (list (- s0) 0.0 s1)
                    (list 0.0 (- s0) (- s1))
                    (list 0.0 (- s0) (- s1)))))
      (ts (list (list 0 1 3)
                (list 0 2 1)
                (list 0 3 2)
                (list 1 2 3)))
        (painter:polyhedron
         attribute
         (map (lambda (p) (add (scl size p) origin)) ps)
         ts)))

(define (sierpinski-tetrahedron attribute size max-count)
  (define (sierpinski-tetrahedron% origin size counter)
    (if (<= counter 0)
        (painter:tetrahedron attribute origin (* 1.3 size))
        (let ((s0 (/ 1.0 4.0))
              (s1 (/ 1.0 (* 4.0 (sqrt 2.0)))))
          (let ((o0 (add (scl size (list s0 0.0 s1)) origin))
                (o1 (add (scl size (list (- s0) 0.0 s1)) origin))
                (o2 (add (scl size (list 0.0 s0 (- s1)) origin))
                (o3 (add (scl size (list 0.0 s0 (- s1)) origin))
                    (s/2 (/ size 2))))
            (painter:union
             (sierpinski-tetrahedron% o0 s/2 (1- counter))
             (sierpinski-tetrahedron% o1 s/2 (1- counter))
             (sierpinski-tetrahedron% o2 s/2 (1- counter))
             (sierpinski-tetrahedron% o3 s/2 (1- counter))))
          (sierpinski-tetrahedron% (list 0.0 0.0 0.0) size max-count)))
    (sierpinski-tetrahedron% (list 0.0 0.0 0.0) size max-count))
```

Scheme  
コード

描画

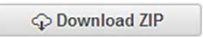
造型



## 目次（基本編）

- ① 3D図形言語のダウンロードとインストール
- ② 基本的な形状を持つ3Dモデルの作成
- ③ 変換や複合化による複雑な3Dモデルの作成
- ④ 3Dモデルのエクスポート
- ⑤ その他、3Dモデルを描画する際の設定など

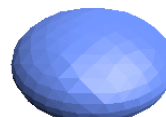
### ① 3D図形言語のダウンロードとインストール

1. Github (<https://github.com/vi-iv/jakld-3dcg>) にアクセス
  2. 右下の  ボタンからファイルをダウンロード
  3. ダウンロードしたファイルを展開し src ディレクトリへ移動
  4. Scheme処理系上で (load "load.scm") と (start-picture) を評価
- 3Dモデルの作例は doc や mod に収録，下式で読込可

```
(load "../mod/〇〇.scm")
(load "../doc/practice-〇.scm")
```



\*cube\*



\*disk1\*



\*union\*

### ② 基本的な形状を持つ3Dモデルの作成 (doc/practice-2.scm)

- 3Dモデルは3Dペインタ (cf. 図形言語のペインタ) として作成
- 3Dペインタは手続き painter:〇〇 により構成，show により描画可能

例：直方体の作成

```
(define *cube*
  (painter:cube *attribute*
    '(10 15 20)))
```

- ← \*cube\* という3Dペインタを定義
- ← \*attribute\* は色等を表す既存の変数
- ← 直方体各辺の長さをリストで指定

```
(show *cube*)
```

← show により \*cube\* を描画

- 基本図形は直方体 cube, 球体 sphere, 円柱 cylinder, 回転体 revolution 等
- 多面体 polyhedron 上で新たに定義可能

### ③ 変換や複合化による複雑な3Dモデルの作成 (doc/practice-3.scm)

- 個々の図形の変換や, 複数の図形の和を取って複雑な図形を構成
- 変換は平行移動 translate, 拡大縮小 scale, 回転 rotate 等, 和は union
- 3Dフレームの適用 transform (アフィン変換に相当) 上で新たに定義可能

例2: 球体を潰した図形を作成

```
(define *disk1*  
  (painter:scale '(1 0.5 1)  
    (painter:sphere *attribute*  
      10  
      2)))
```

- ← \*disk1\* という3Dペインタを定義
- ← x,y,z 各軸方向の倍率をリストで指定
- ← \*attribute\* は色等を表す既存の変数
- ← 球体の半径を数値で指定
- ← 球体の多面体による近似の度合いを数値で指定

例3: 球体を潰した図形の和を作成

```
(define *union*  
  (painter:union *disk1* *disk2*))
```

- ← \*union\* という3Dペインタを定義
- ← \*disk1\* と \*disk2\* を合成

### ④ 3Dモデルのエクスポート (doc/practice-4.scm)

- 作品はOpenSCADスクリプト形式へ変換してエクスポート可能
- 手続き export により直前に描画した3Dペインタをエクスポート

例4: 例3のモデルを出力

```
(export "union.scad"  
  'scad)
```

- ← union.scad というファイル名で保存
- ← OpenSCADスクリプト形式であることを指定

- 出力した作品はOpenSCADによっても確認可能

### ⑤ その他, 3Dモデルを描画する際の設定など (doc/practice-5.scm)

- 3Dモデルの色, 描画時の視点と光源を設定可能

例5: 3Dモデルの色の指定

```
(define *red*  
  (make-simple-attribute #xff0000))  
  
(show (painter:cube *red*  
  '(5 5 5)))
```

- ← \*red\* という色を定義
- ← 色をカラーコードの16進数 #xff0000 で指定
- ← \*attribute\* の代わりに \*red\* を引数
- ← 立方体の3Dペインタを構成し描画

例6: 視点 (位置, 方向, 視界) の指定

```
(set! *camera*  
  (make-camera '(0 0 20)  
    '(0 0 -1)  
    '(80 80 100)))
```

- ← 大域変数 \*camera\* に設定して反映
- ← 視点の位置を (0,0,20) と指定
- ← 視線の方向を (0,0,-1) と指定
- ← 視界を各辺長 80,80,100 の直方体として指定

例7: 光源 (方向) の指定

```
(set! *lights*  
  (list (make-light '(1 0 0))))
```

- ← 大域変数 \*lights\* に設定して反映
- ← 光線の方向を (1,0,0) と指定

※ 締切までに提出された方に研究協力への謝礼を進呈!

## 目次（応用編）

- ① 新しい基本図形の構成手続きの定義
- ② 新しい変換手続きの定義
- ③ 再帰構造を持つフラクタル図形の作成
- ④ 作例集

## ① 新しい基本図形の構成手続きの定義 (mod/sierpinski-tetrahedron.scm)

- 複合的でない図形は painter:polyhedron により直接作成
- painter:polyhedron の引数は頂点と面の情報

## 例8：正四面体の構成手続きの定義

```
(define (painter:tetrahedron attribute origin size)
  (let ((s0 (/ 1.0 2.0))
        (s1 (/ 1.0 (* 2.0 (sqrt 2.0)))))
    (let ((ps (list (list s0 0.0 s1)
                    (list (- s0) 0.0 s1)
                    (list 0.0 s0 (- s1))
                    (list 0.0 (- s0) (- s1))))
          (ts (list (list 0 1 3)
                    (list 0 2 1)
                    (list 0 3 2)
                    (list 1 2 3))))
      (painter:polyhedron
       attribute
       (map (lambda (p) (add (scl size p) origin)) ps)
       ts)))

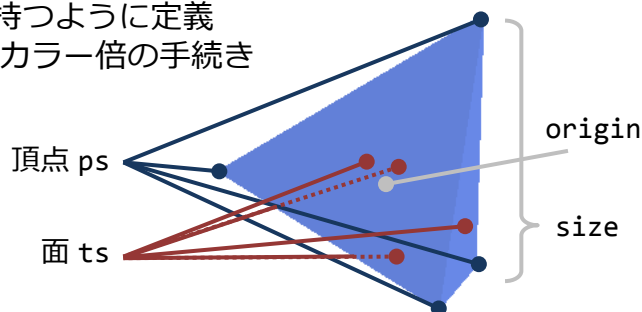
(show (painter:tetrahedron *attribute* '(0 0 0) 10))
```

4つの頂点の座標を表すリスト

4つの面を表す頂点番号のリスト

各頂点の座標について  
size 拡大, origin 平行移動

- 中心の位置 origin と大きさ size を持つように定義
- なお add, sub, scl は加算, 減算, スカラー倍の手続き



## ② 新しい変換手続きの定義

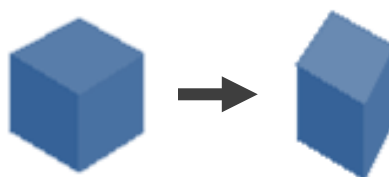
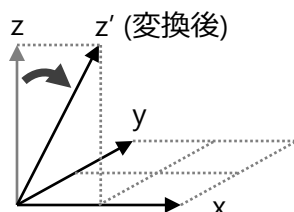
- 独自の変換手続きは3Dフレームとその適用手続き painter:transform を用いて定義
- 3Dフレームはアフィン変換行列に相当する情報

## 例9：剪断 (skew) の定義

```
(define *frame*
  (make-3d-frame '(0 0 0)
                 '(1 0 0)
                 '(0 1 0)
                 '(0.5 0 1)))

(define (painter:skew painter)
  (painter:transform painter *frame*))
```

原点(0,0,0),及び  
x軸(1,0,0),y軸(0,1,0),z軸(0,0,1)  
の変換先の座標を指定



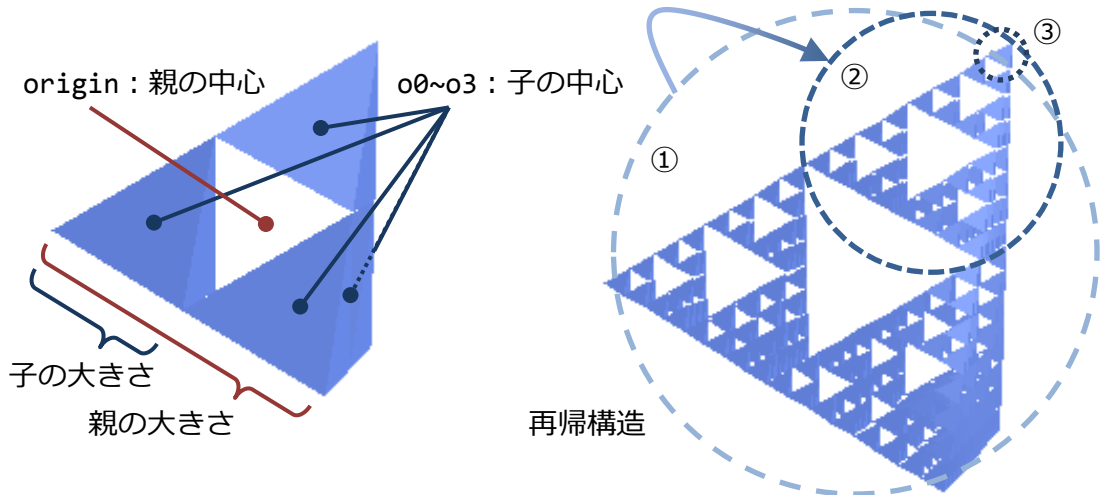
### ③ 再帰構造を持つフラクタル図形の作成 (mod/sierpinski-tetrahedron.scm)

- 3Dペインタの再帰呼出しによりフラクタル図形を作成可能

#### 例9 : Sierpinski Tetrahedron の作成

```
(define (painter:sierpinski-tetrahedron attribute size max-count)
  (define (sierpinski-tetrahedron% origin size counter)
    (if (<= counter 0)
      (painter:tetrahedron attribute origin (* 1.3 size))
      (let ((s0 (/ 1.0 4.0))
            (s1 (/ 1.0 (* 4.0 (sqrt 2.0)))))
        (let ((o0 (add (sc1 size (list s0 0.0 s1)) origin))
              (o1 (add (sc1 size (list (- s0) 0.0 s1)) origin))
              (o2 (add (sc1 size (list 0.0 s0 (- s1)) origin))
                    (o3 (add (sc1 size (list 0.0 (- s0) (- s1)) origin))
                      (s/2 (/ size 2))))
          (painter:union
            (sierpinski-tetrahedron% o0 s/2 (1- counter))
            (sierpinski-tetrahedron% o1 s/2 (1- counter))
            (sierpinski-tetrahedron% o2 s/2 (1- counter))
            (sierpinski-tetrahedron% o3 s/2 (1- counter))))))
    (sierpinski-tetrahedron% (list 0.0 0.0 0.0) size max-count))

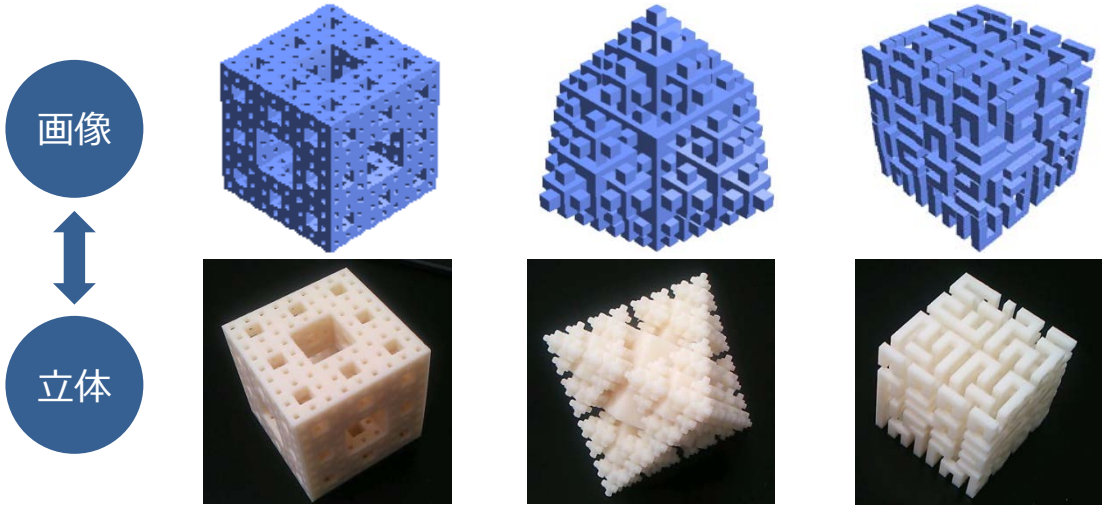
(show (painter:sierpinski-tetrahedron *attribute* 10 3))
```



- なお (\* 1.3 size) は重複を持たせて点で接さないようにする造形上の工夫

### ④ 作例集

- 作例と昨年度の講義の提出作品



Menger Sponge

オリジナル作品 (柊井君)

Hilbert Curve (豊島君)