Schemeで 3次元図形を 造ろう!

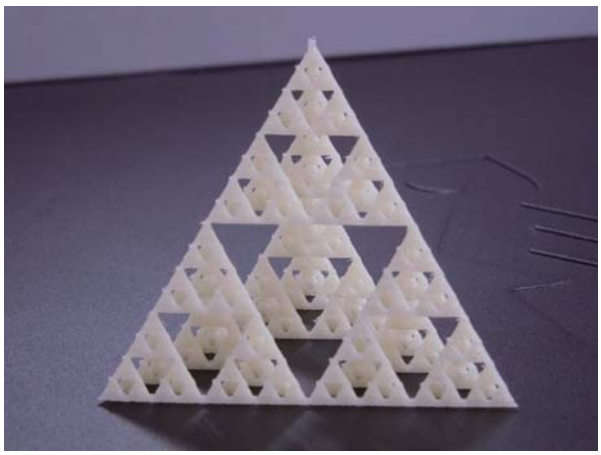
資料

京都大学 奥乃研究室 M1 古川 孝太郎

2013年12月10日 (火)

随意課題: 3次元painter作成 (1/2)

- 3次元painterを定義してプログラムを提出



造形された Sierpinski Tetrahedron

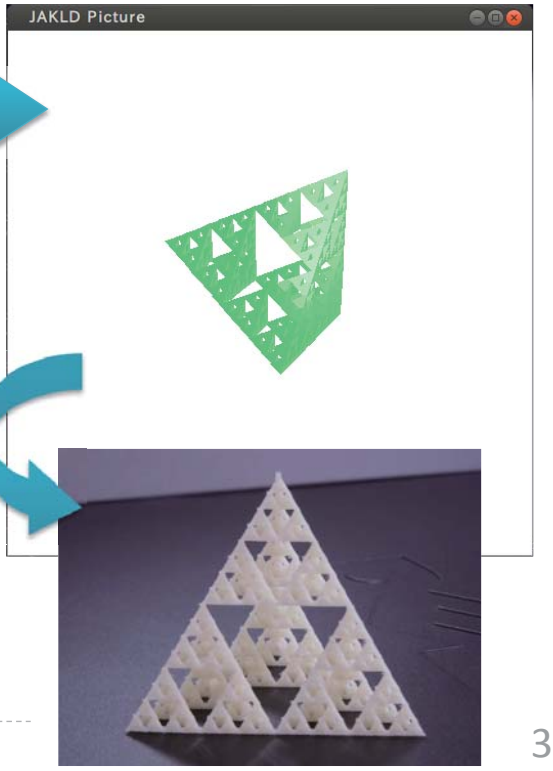
※ このスライドは ,
配布プログラムは https://github.com/vi-iv/jakld-3dcg にあります

随意課題: 3次元painter作成 (2/2)

- 配布プログラム (JAKLD-3DCG) を利用

```
(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))
```



※ Scheme 処理系は JAKLD のみ想定

目次

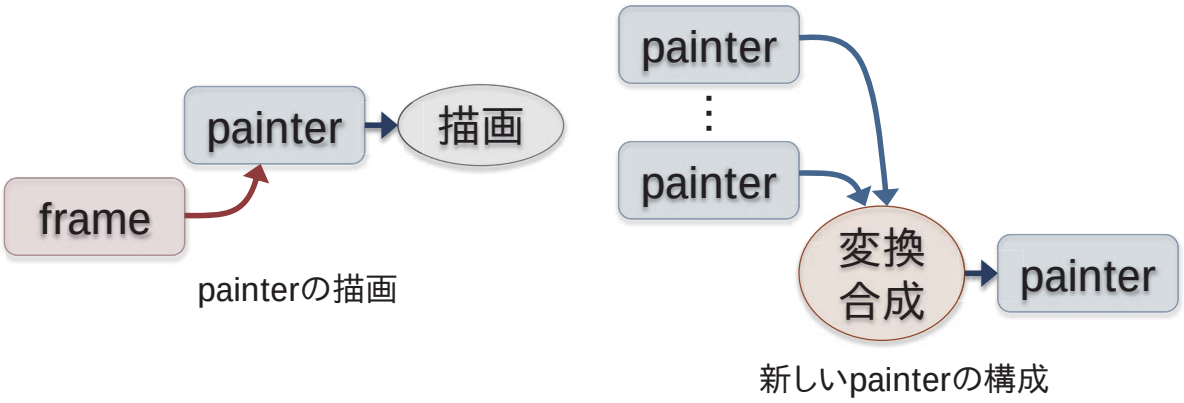
1. 2次元の図形言語のおさらい
2. 図形言語の3次元拡張の概要
3. 3次元Painterの定義方法
4. フラクタル図形の作り方
5. 課題説明と注意
6. 図形言語の3次元拡張の詳細
7. 補遺:3次元プリンタの紹介
8. 参考文献

この範囲を見れば課題はできる！

少し複雑な図形を作るとき参照

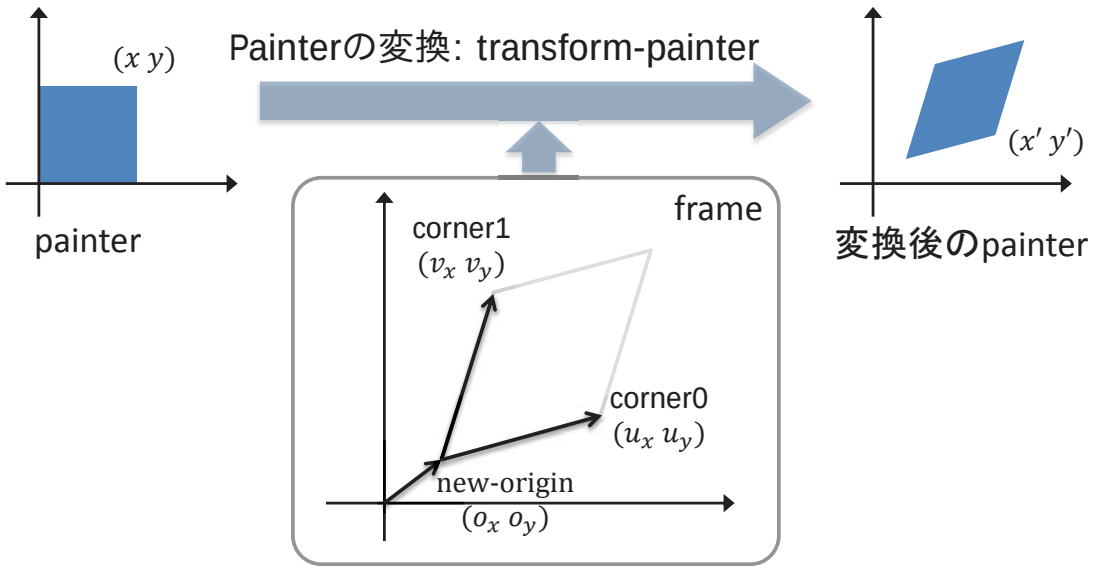
2次元の図形言語のおさらい (1/3)

- painter
  - 評価すると描画される図形
  - painterの階層
- frame
  - painterの描画される平行四辺形の領域
  - 原点と平行四辺形の2辺で表現



2次元の図形言語のおさらい (2/3)

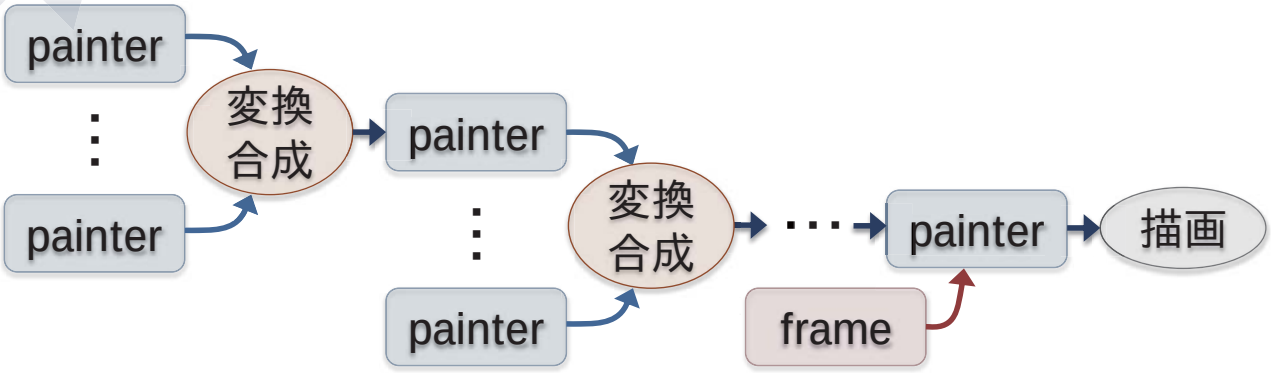
- transform-painter: 線形変換+平行移動



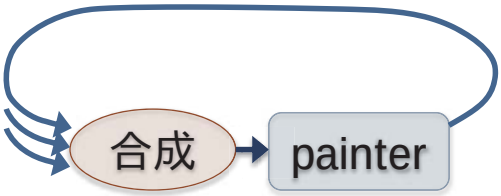
- $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} u_x - o_x & v_x - o_x \\ u_y - o_y & v_y - o_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} o_x \\ o_y \end{bmatrix}$

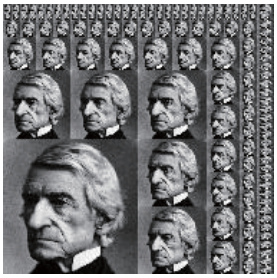
2次元の図形言語のおさらい (3/3)

◆ painterの階層構造



◆ 再帰構造



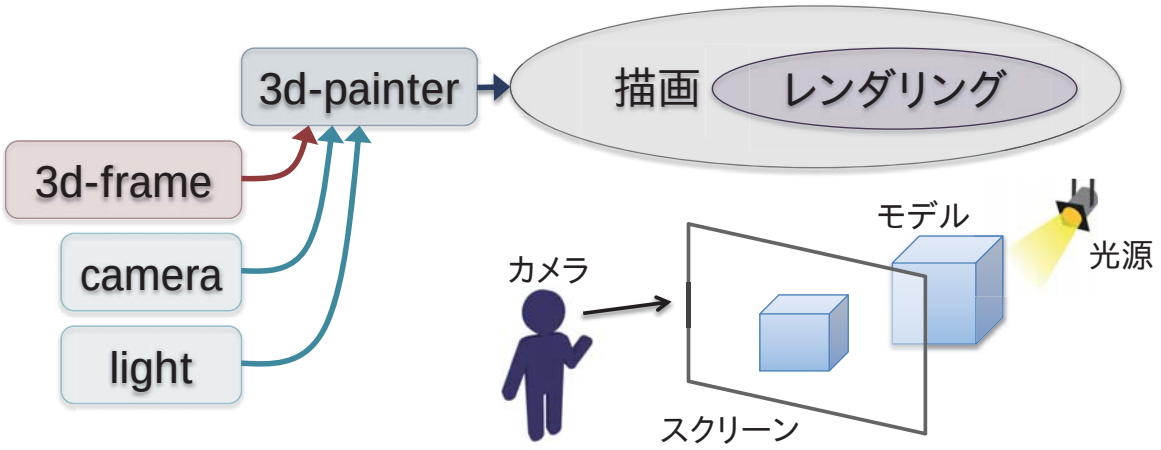


再帰的に構成されたペインタ:square-limit

図形言語の3次元拡張の概要 (1/2)

◆ 3次元painterの導入
- 評価して描画, 変換, 合成などは2次元と同じ
- 3次元frameも導入

◆ 描画には3次元→2次元の変換が必要 (レンダリング)
- レンダリング時にカメラ (観察者), 光源が必要



図形言語の3次元拡張の概要 (2/2)

- 3次元painterの定義方法
  - ①多面体 ②frameによる変換 ③合成 の組合せ

```
(painter:transfrom
 (painter:polyhedron
  attribute0
  (list (list 0.0 0.0 0.0) (list 10.0 0.0 0.0) … ) ; 頂点
  (list (list 0 1 2) (list 0 2 3) … ))             ; 面
 (make-3d-frame (list 5.0 0.0 0.0)
                (list 1.0 0.0 0.0)
                (list 0.0 1.0 0.0)
                (list 0.0 0.0 1.0))
```

  - これだけでは不便
    - ラッパーとして ④基本的な図形 ⑤基本的な変換

```
(painter:translate (list 5.0 0.0 0.0)
                   (painter:cube attribute0 (list 10 10 10))
```


3次元painterの定義方法

- 基礎となる手続き
  - painter:polyhedron :①多面体
  - painter:transform :②frameによる変形
  - painter:union :③合成
- ④基本的な図形 (painter:polyhedron上で定義可能)
  - painter:cube :直方体
  - painter:sphere :球体
  - painter:cylinder :円柱, 円錐
  - painter:revolution :回転体
- ⑤基本的な変換 (painter:transform上で定義可能)
  - painter:translate :並行移動
  - painter:scale :各軸についての拡大, 縮小
  - painter:rotate :任意軸周りの回転

3次元painterの定義方法:多面体作成

◆ (painter:polyhedron <attribute>
<points> <triangles>)

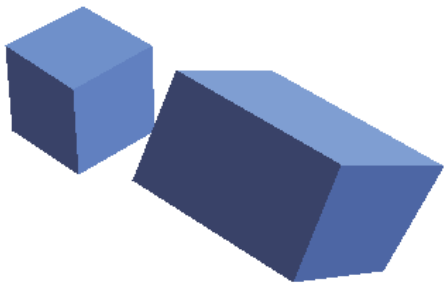
- attribute :表面属性(図形の色や性質を決定)
  - (make-attribute …)
    - 「詳細」参照
- points :多面体の頂点座標の集合
  - (list (list x y z) …)
    - x, y, z: 数値
- triangles :表面の三角形ポリゴンの集合を表す
  points内の要素の番号
  - (list (list a b c) …)
    - a, b, c: 整数値

3次元painterの定義方法:変換

◆ (painter:transform <painter> <frame>)

- painter :変換対象のpainter
- frame :変換に用いるframe
  - (make-3d-frame …)
    - 「詳細」参照

```
(show transform-001)
```



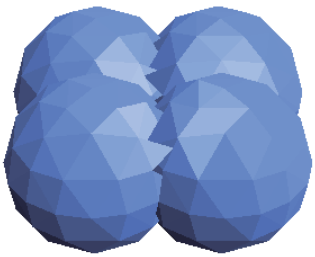
※ painter:transformのラッパーであるいくつかの基本的な変換手続き(後述)を使用すれば,
frameとpainter:transformを陽に用いなくとも図形は変換可能

3次元painterの定義方法:合成

- (painter:union <painter> …)
  - painter :合成対象のpainter

```
(show compounded-001) ; mod/trivial-shape.scm (以下同ファイル)
```

```
(show compounded-002)
```

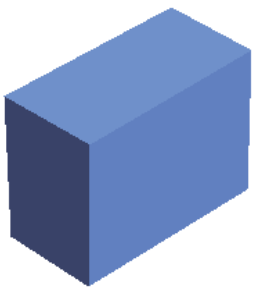


基本的な図形と変換:直方体

- (painter:cube <attribute> <size>)
  - attribute :表面属性
  - size :辺の長さ
    - (list x y z)
      - x, y, z: 数値

```
(show cube-001)
```

```
(show cube-002)
```



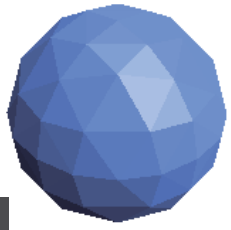
基本的な図形と変換:球体

◆ (painter:sphere <attribute>
<radius> <resolution>)

- attribute :表面属性
- radius :半径
  - 数値
- resolution :ポリゴン分割の細かさ
  - 整数値

(show sphere-002)

(show sphere-003)

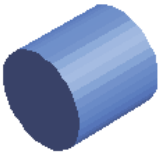


基本的な図形と変換:円柱, 円錐

◆ (painter:cylinder <attribute> <height>
<top-radius>
<bottom-radius>
<resolution>)

- attribute :表面属性
- height :高さ
  - 数値
- top-radius :上底の半径
  - 数値
- bottom-radius :下底の半径
  - 数値
- resolution :ポリゴン分割の細かさ
  - 整数値

(show cylinder-001)

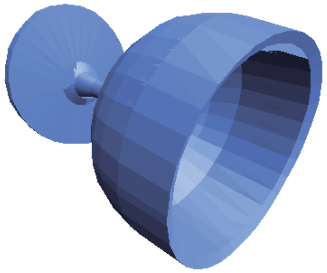


基本的な図形と変換:x軸回転体

◆ (painter:revolution <attribute> <points>
<resolution>)

- attribute :表面属性
- points :xy平面上の多角形の頂点
  - (list (list x y z) …)
    - x, y, z: 数値
- resolution :ポリゴン分割の細かさ
  - 整数値

(show glass-001)

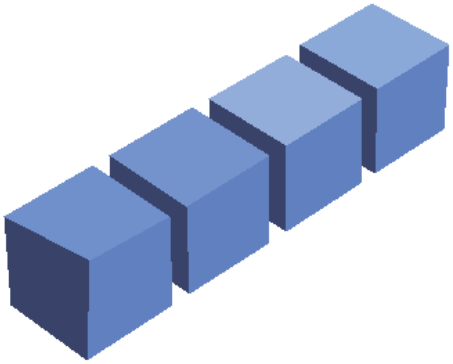


基本的な図形と変換:平行移動

◆ (painter:translate <vector> <painter>)

- vector :変位
  - (list x y z)
    - x, y, z: 数値
- painter :変換対象のpainter

(show translate-001)

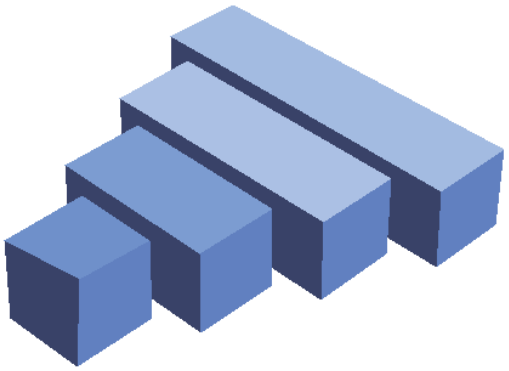


基本的な図形と変換:拡大･縮小

◆ (painter:scale <scale> <painter>)

- scale　　:各軸方向の拡大･縮小倍率
  - (list (list x y z) …)
    - x, y, z: 数値
- painter　:変換対象のpainter

(show scale-001)

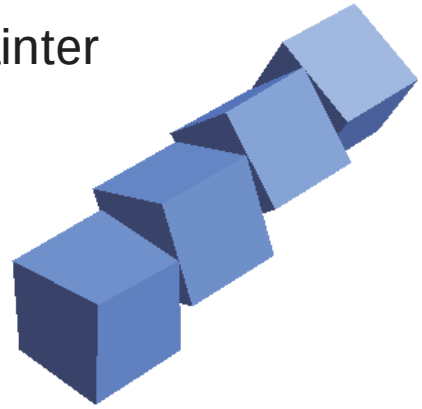


基本的な図形と変換:回転

◆ (painter:rotate <degree> <axis> <painter>)

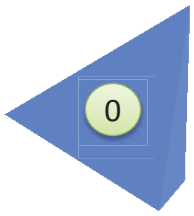
- degree　　:角度 (360°)
  - 数値
- axis　　:軸
  - (list x y z)
    - x, y, z: 数値
- painter　:変換対象のpainter

(show rotate-001)



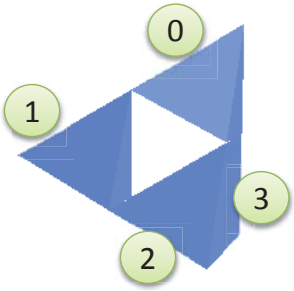
フラクタル図形の作り方

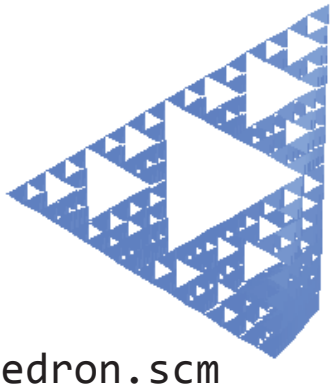
1. 最小単位に注目して図形を定義



   a. 原点
   b. 大きさ

2. 親子の位置関係に注目して再帰手続きを定義





3. 完成
   サンプル:mod/sierpinski-tetrahedron.scm

課題説明と注意 (1/4)

課題:気の利いた3次元painterを定義して提出

手順

1. 配布プログラムのダウンロード

```
~$ git clone https://github.com/vi-iv/jakld-3dcg
```

2. ディレクトリ移動

```
~$ cd ./jakld-3dcg/src
```

3. JAKLDを起動し, ロード

```
~$ jakld
> (load "load.scm")
```

4. サンプル (../mod/*.scmで定義) を実行し確認

```
> (show cube-001)
> (show sphere-001)
> (show sierpinski-tetrahedron-002)
```

課題説明と注意 (2/4)

5. モデルを定義

※ 一部の引数・変数は定義済みのものを利用可

- attribute:表面の色や性質を決定する引数
  - attribute-001, -002
- *camera*:カメラを表現する大域変数
  - camera-001
- *lights*:光源を表現する大域変数
  - lights-001

```
> (define my-painter
    (painter:union
     (painter:cube attribute-001 (list 1 20 1))
     (painter:sphere attribute-001 10 3)))
```

※ 造形されるモデルは単色(アイボリー)

課題説明と注意 (3/4)

6. 描画して確認

```
> (show my-painter)
> (redraw) ; 直前のpainterを再描画
```

7. データをエクスポート

```
> (export "my-painter.scm")          ; S式
> (export "my-painter.scad" 'scad)   ; SCADスクリプト
```

8. メールで提出
   - 作成したプログラムの*.scmファイルとその説明
   - 描画した画像 *.png | *.jpg
   - エクスポートしたデータの*.scadファイル
9. 何らかのフィードバック

※ エクスポートされたSCADスクリプトファイルはOpenSCADなどのソフトウェアで確認可能
※ インポートは手続きimportにより, エクスポートされたS式ファイルからのみ可能

課題説明と注意 (4/4)

注意点

※ 図形のサイズは 50x50x50 の範囲内（単位[mm]）

※ 図形に [mm] よりも細い・薄い部分を作らない

※ 図形に閉じた空間を持たせない

※ 分離するものは一つ一つが細かくなり過ぎないように

※ 必ず描画されることを確認するように

※ 説明不足，バグ，質問は下記まで連絡してください

kfurukaw[at]kuis.kyoto-u.ac.jp

図形言語の3次元拡張の詳細 (1/5)

◆ カメラ (観察者) の定義方法
- 位置
- 方向
- 視界の体積

```
(define camera-001
  (make-camera (list 0 0 10)    ; 位置
               (list 0 0 -1)    ; 方向
               (list 50 50 50)) ; 視界

(set! *camera* camera-001)
```

◆ 光源の定義方法
- 種類 (平行or点)
- 位置 (点のみ)
- 方向 (平行のみ)
- 強度

```
(define light-001
  (make-parallel-light
   (list 1 1 1)                    ; 方向
   (make-intensity 0.5 0.5 0.5))) ; 強度

(set! *lights*
      (cons light-001 *lights*))
```

※ カメラ・照明は大域変数に設定しておいたものがレンダリング時に参照される
光源は複数定義可能

図形言語の3次元拡張の詳細 (2/5)

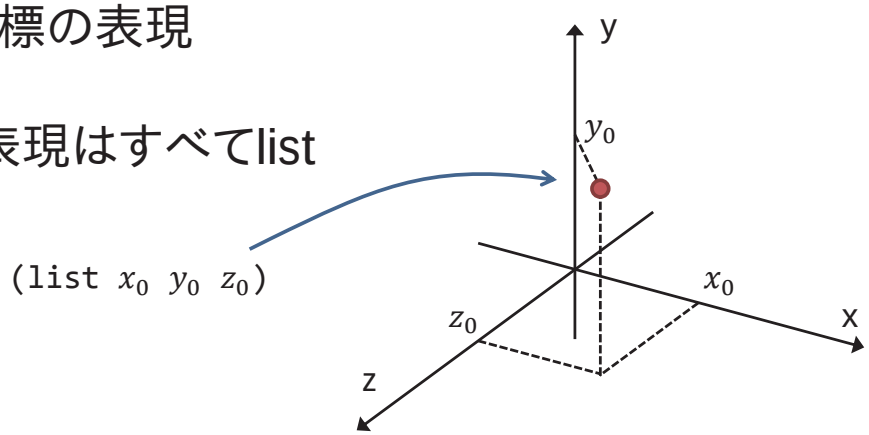
- 3次元frame
  - 平行六面体 (cf. 2次元では平行四辺形) への変換

- 3次元frameの定義方法
  - 原点
  - x, y, z軸の変換先 (の方向ベクトル)

```
(define 3d-frame-001
  (make-3d-frame (list 0.0 0.0 0.0)  ; 原点の変換先
                 (list 2.0 0.0 0.0)  ; x軸〃
                 (list 0.0 2.0 0.0)  ; y軸〃
                 (list 0.0 0.0 2.0)) ; z軸〃
```

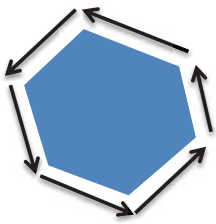

図形言語の3次元拡張の詳細(3/5)

- 座標系と座標の表現
  - 右手系
  - 座標の表現はすべてlist



  - 造形時のスケールは [mm]

- ポリゴンの裏表
  - 頂点が時計回りに並ぶ側が表



図形言語の3次元拡張の詳細(4/5)

- 表面属性
  - ポリゴンの表面の色，反射の程度などを決定
  - 配布プログラムはPhongの反射モデルに依拠

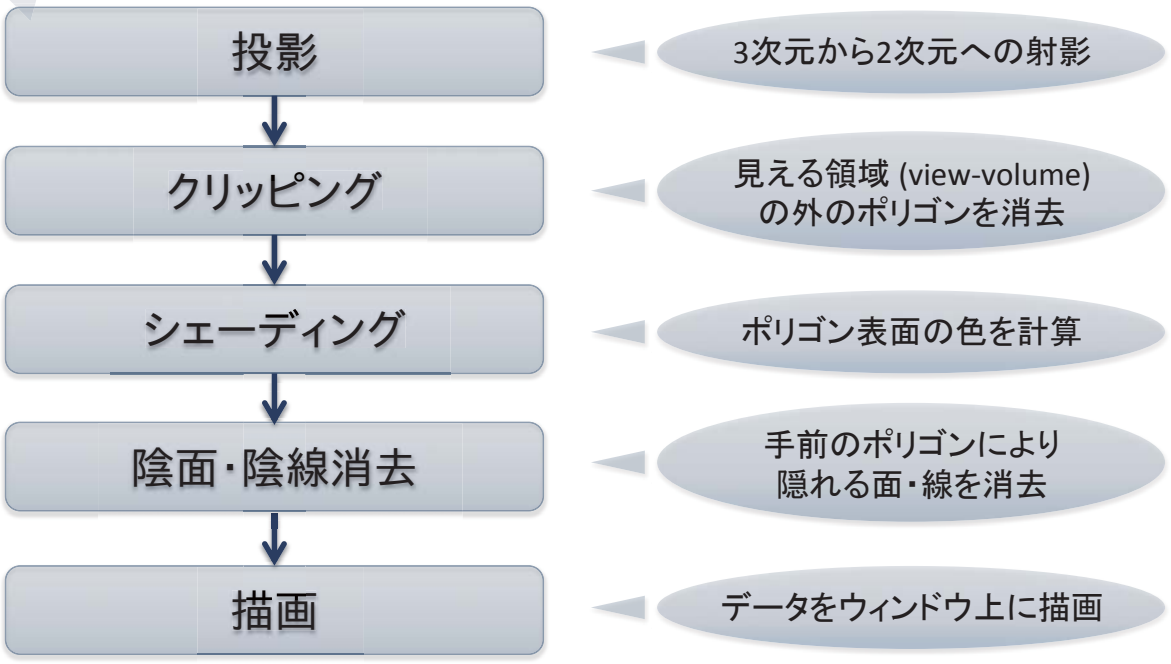
- 表面属性の定義方法

```
(define attribute-001
  (make-attribute (hex->color #x000000) ; 投影後の色（何でもよい）
                  (hex->color #x4169e1) ; 環境反射成分
                  (hex->color #x4169e1) ; 拡散反射成分
                  (hex->color #xffffff) ; 鏡面反射成分
                  (hex->color #x000000) ; 発光成分
                  3))
```

  - 環境反射成分と拡散反射成分が重要

図形言語の3次元拡張の詳細(5/5)

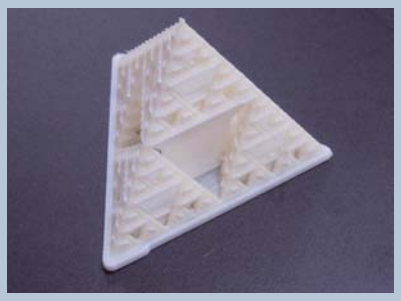
- レンダリングとは



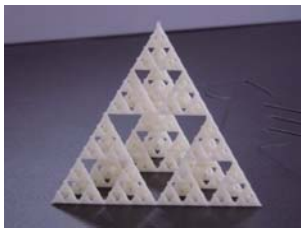
補遺:3次元プリンタの紹介 (1/2)

◆ uPrint SE (Stratasys Ltd.)

- 熱溶解積層型
- 硬化ABS樹脂で造形
  - モデル材+サポート材(穴埋め)



- サポート材は薬剤で溶かして除去

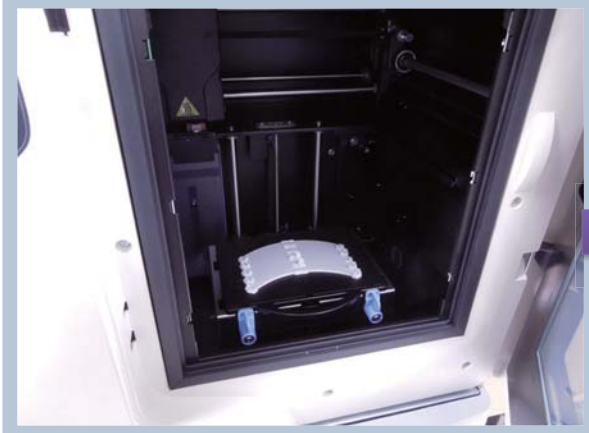


- Sierpinski tetrahedron: 4時間
- Menger sponge: 9時間

補遺:3次元プリンタの紹介 (2/2)

◆ 造形の手順

造形



- ABS樹脂を積層
- 自動でパス生成

洗浄

- サポート材を溶解

参考文献

- Structure and Interpretation of Computer Programs,
  http://mitpress.mit.edu/sicp/full-text/book/book.html
- Javaアプリケーション組み込み用のLispドライバ:JAKLD,
  http://www.yuasa.kuis.kyoto-u.ac.jp/~yuasa/jakld/index-j.html
- Wolfram|Alpha Examples: Fractals,
  http://www.wolframalpha.com/examples/Fractals.html
- Rapid Prototyping Web Page,
  http://www.georgehart.com/rp/rp.html