3D プリンタによる
“3 次元タートル・グラフィクス”

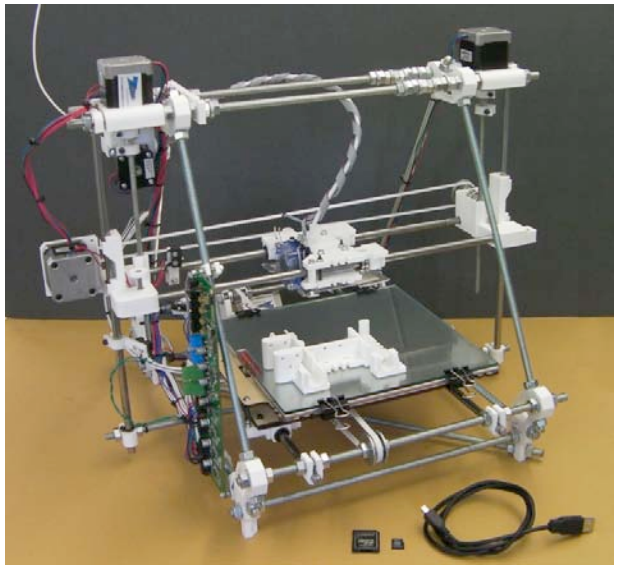
金田　泰
Dasyn.com

はじめに: 通常の 3D オブジェクト設計と印刷

► FDM 型 3D プリンタ

- 3D プリンタにはいろいろな種類があるが，普及している安価なタイプは **FDM 型** (fused deposition modeling 型) とよばれる.
- とかしたプラスティックをノズルの先端から射出してかためる.

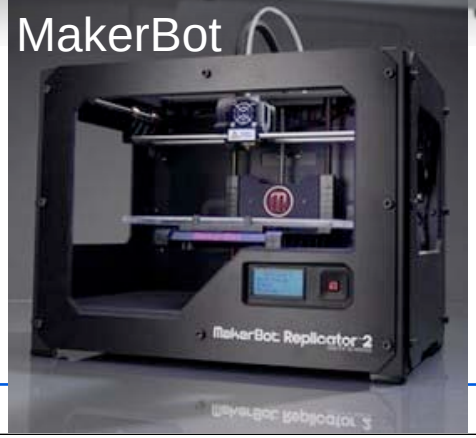
Reprap



http://www.3dfuture.com.au/2012/04/reprap-full-kit-available-for-780/

Stratasys FDM®

MakerBot



Rostock MAX

はじめに: 通常の 3D オブジェクト設計と印刷 (つづき)

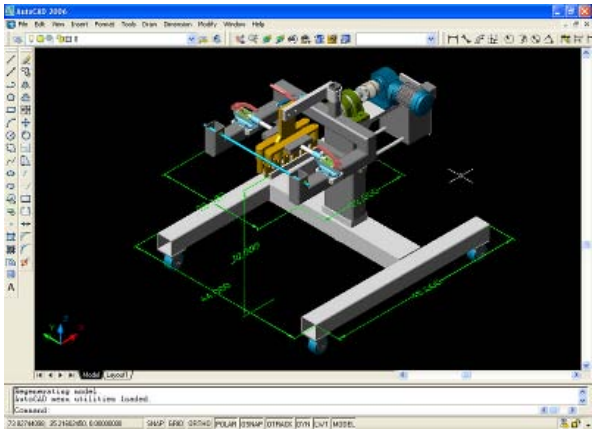
▶ 3D オブジェクトの設計・印刷過程の概要

- Step 1: 3D CAD ツールによる設計と STL ファイル出力 (変換)
  - STL = Standard Triangulation Language or Stereo-Lithography
- Step 2: スライサー (ソフトウェア) による層への分解
- Step 3: 3D プリンタによる層ごとの印刷

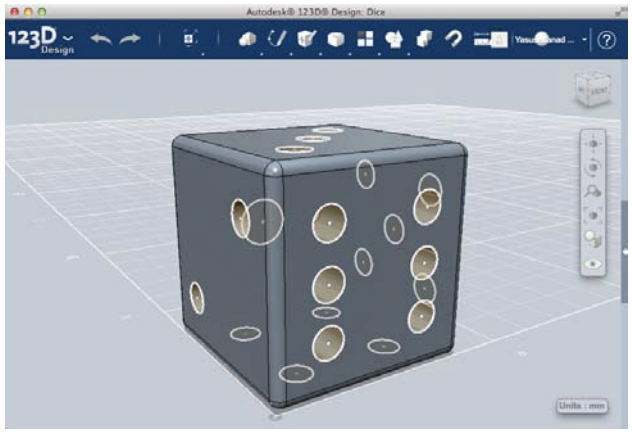
▶ 以下この過程をより詳細にみる.

[Step 1] 3D オブジェクトの設計と STL への変換

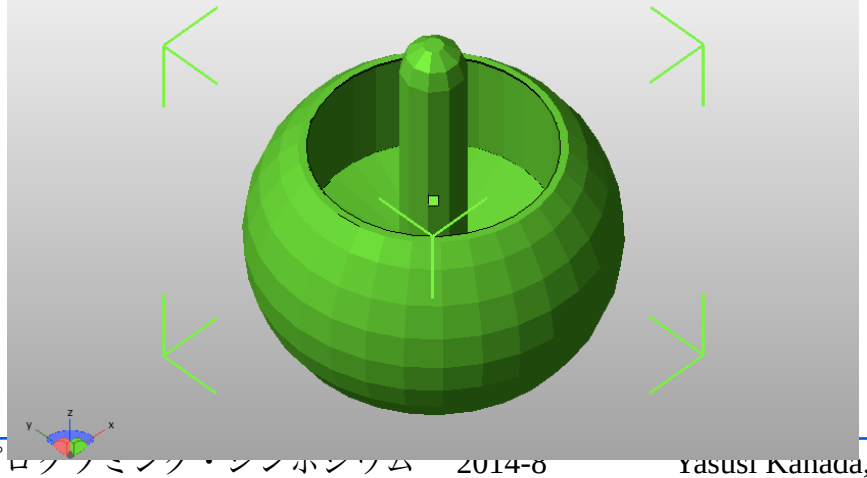
- 通常は 3D CAD を使用して設計し，ベンダ依存の 3D モデル表現用の宣言的言語によって表現する.



Autodesk
(CAD software)



- 3D モデルを標準の宣言的な言語 STL に変換する.
  - STL はオブジェクトの表面形状を三角形の集合で近似的に表現.



```
solid tippetop_1piece_v05-repaired
 facet normal -0.988014 0.103855 -0.114202
  outer loop
   vertex -16.500000 0.000000 -7.935401
   vertex -16.139400 3.430540 -7.935400
   vertex -15.786800 3.355581 -11.054070
  endloop
 endfacet
 facet normal -0.988010 0.103854 0.114240
  outer loop
   vertex -16.500000 0.000000 -7.935401
   vertex -16.139400 0.000000 -4.816731
   vertex -16.139400 3.430540 -7.935400
  endloop
 endfacet
 ...
endsolid tippetop_1piece_v05-repaired
```

[Step 2] スライスと G-Code

- **スライス結果は通常，手続き的な CAM 用言語 G-Code によって表現される.**
  - G-Code はもともと工作機械のツール (刃など) の動作をプログラムするものである.
  - G-Code はプリント・ヘッドの移動やプラスティックを射出する速度などを指定する.
- **コマンドの例**
  - G0: 単純なツールの移動
    - G0 X0 Y0 Z0 F3600: 分速 3600 mm で座標 (0, 0, 0) に移動する.
  - G1: 加工 (印刷) しながらのツールの移動
    - G0 X0 Y0 Z0 F3600 E100: E100 によって指定されるフィラメントを射出しながら移動する.

[Step 3] 3D オブジェクトの印刷

- **プリント・ヘッドは層間移動のとき以外は水平にだけ移動する.**
  - 通常は水平にスライスされた層ごとに印刷するから.
- **しかし，G-Code をつかえばヘッドを自由な方向に移動できる.**

G0 X1 Y1 **Z1**

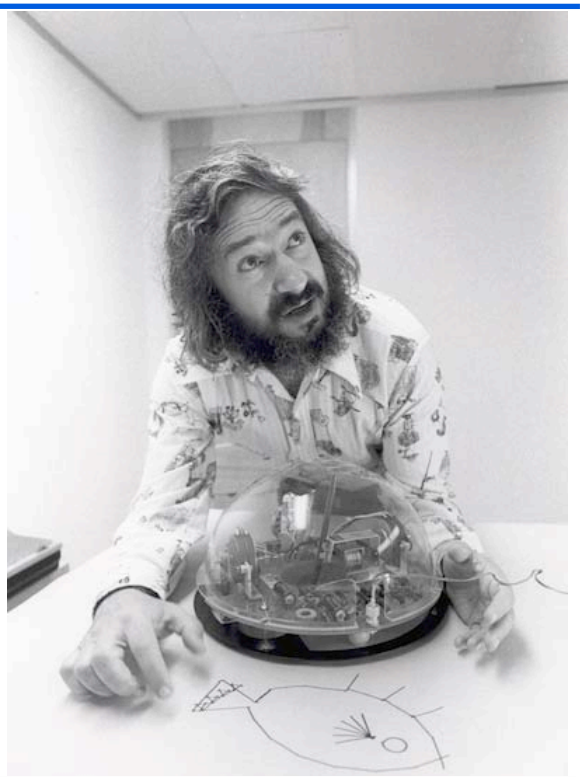
(X0, Y0, **Z0**) → (X1, Y1, **Z1**)

  - ただし，3D プリンタのなかには垂直移動の不得意なものが多い.

タートル・グラフィクス

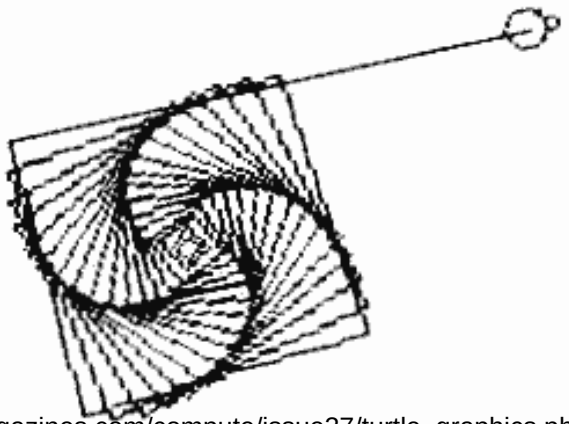
▶ Seymore Papert と Logo

- Papert はこどもむけのプログラミング言語 Logo を設計した.
- Logo をつかって「カメ」の軌跡で 2 次元の線画をかかせることができる — (2 次元) タートル・グラフィクス



▶ タートル・グラフィクスの描画コマンド

- Forward d: 距離 d だけ前進する.
- Left a: 角度 a だけ左にまがる.
- Right a: 角度 a だけ右にまがる.

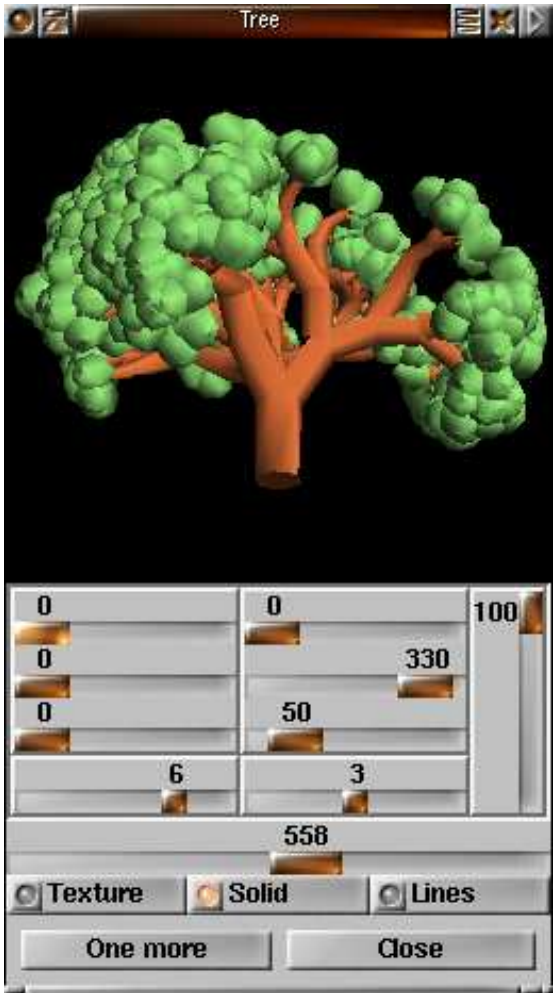


http://www.atarimagazines.com/compute/issue37/turtle_graphics.php

3 次元タートル・グラフィクス

- 基本
  - forward, left, right
  - **up**, **down**

- 拡張された 3D タートル・グラフィクス
  - Bernd Paysan, “Dragon Graphics”



Paysan, B., ““Dragon Graphics” Forth, OpenGL and 3D-Turtle-Graphics”, 2009, http://bigforth.sourceforge.net/dragongraphics-eng.pdf

3D 印刷による “タートル・グラフィクス”

- **描画コマンドを G-Code に翻訳する**
  - Forward → G1 (印刷しながらの移動)
- **カメの座標をデカルト座標に変換する**
  - カメの方向を G-Code 生成プログラムが記憶しておく.
  - Forward → 現在の座標と方向とからつぎの座標を計算する.
  - Left, Right → 方向を修正する.
- **カメの座標系の選択**
  - 極座標
    - カメの方向は 3 次元
    - フライトシミュレータの座標系
    - 印刷可能性を保証するのが困難: 重力の方向がわからなくなる.
  - 円筒座標
    - カメの方向は水平とする.
    - 垂直方向の変位を記述する.
    - この方法のほうがデザインしやすい (?)

3D 印刷と 3D グラフィクスとのちがい

- **3D 印刷で空中に印刷するのは困難**
  - 3D グラフィクスでは 3 次元空間のどこにでも自由に線をひくことができる.
  - 地上で 3D 印刷するかぎりは射出されたフィラメントをささえるものが必要なので空中印刷は困難.
- **印刷速度やフィラメント射出量を制御する必要がある (制御できる).**
  - 3D グラフィクスでも線のふとさなどをかえることはできる.
  - きれいに 3D 印刷するには 3D グラフィクスでは不要なくふうが必要になる.
    - たとえば印刷速度をおさえる必要がある.

選択肢とライブラリの開発

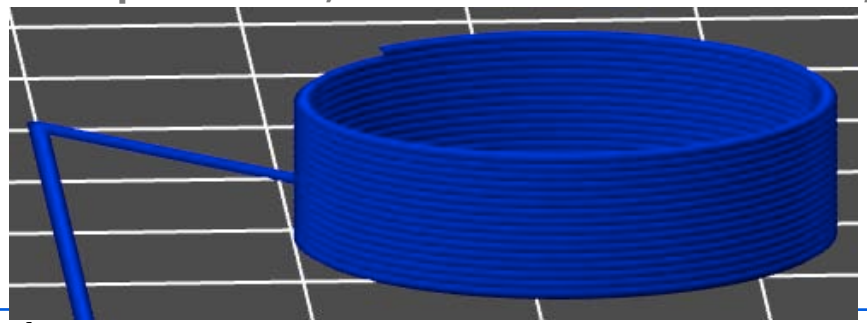
▶ 選択肢

- Logo のような言語を設計する.
- 既存の言語のライブラリを開発する
  — このほうが容易かつ普及しやすい.

▶ Python のライブラリ turtle.py を開発した

- 円筒座標を使用する.
- forward(r, z) によって，z 方向に移動しながら前進できる.
- オープンソースで提供中: Rostock MAX プリンタ用であり，他のプリンタでは多少の修正が必要.
- 例: 螺旋

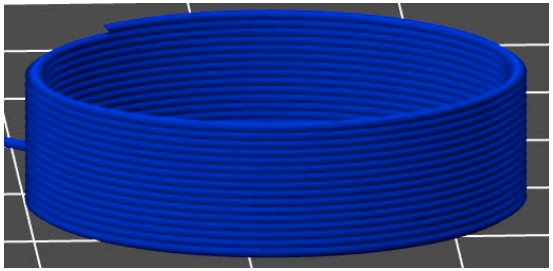
```
turtle.init(FilamentDiameter,
            HeadTemperature, BedTemperature, defaultCrossSection,
            x0, y0, 0.4);

dz = 0.4 / 72;
for j in range(0, 16):
    for i in range(0, 72):
        forward(1, dz);
        left(5);
```



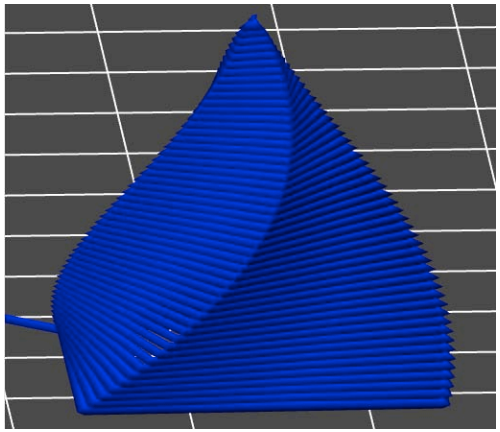
実験: 方法

▶ つぎのような例題をこころみた

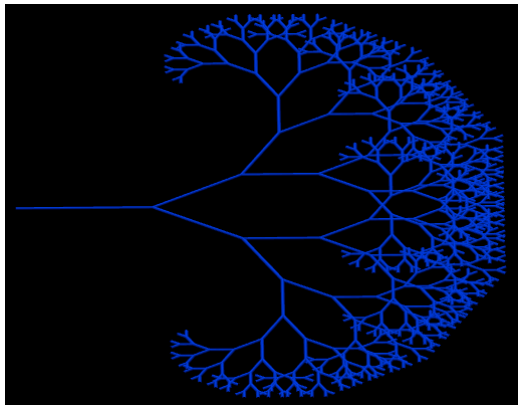
- 円筒 (螺旋)



- ねじれ四角錐



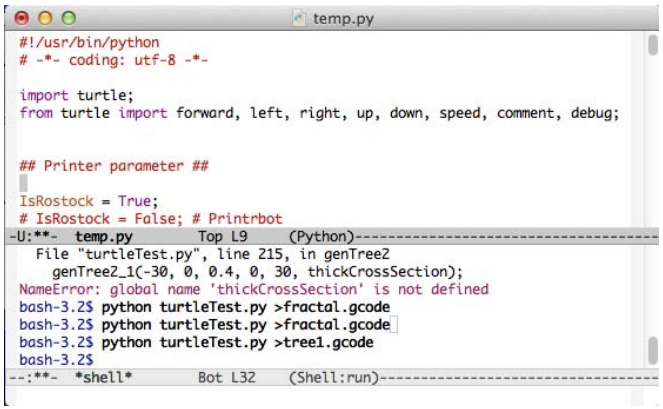
- 平面フラクタル

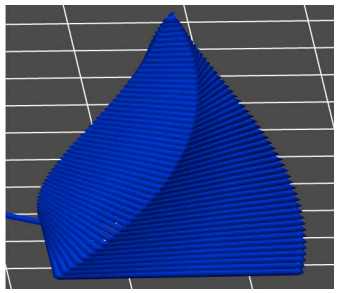


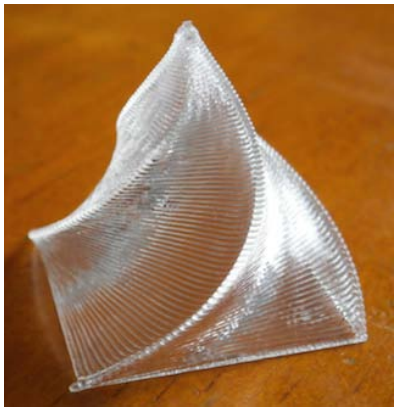
3 次元フラクタルは困難

▶ つぎのような手順を成功するまでくりかえした:

- プログラム記述
- グラフィクスによる確認
- 3D 印刷

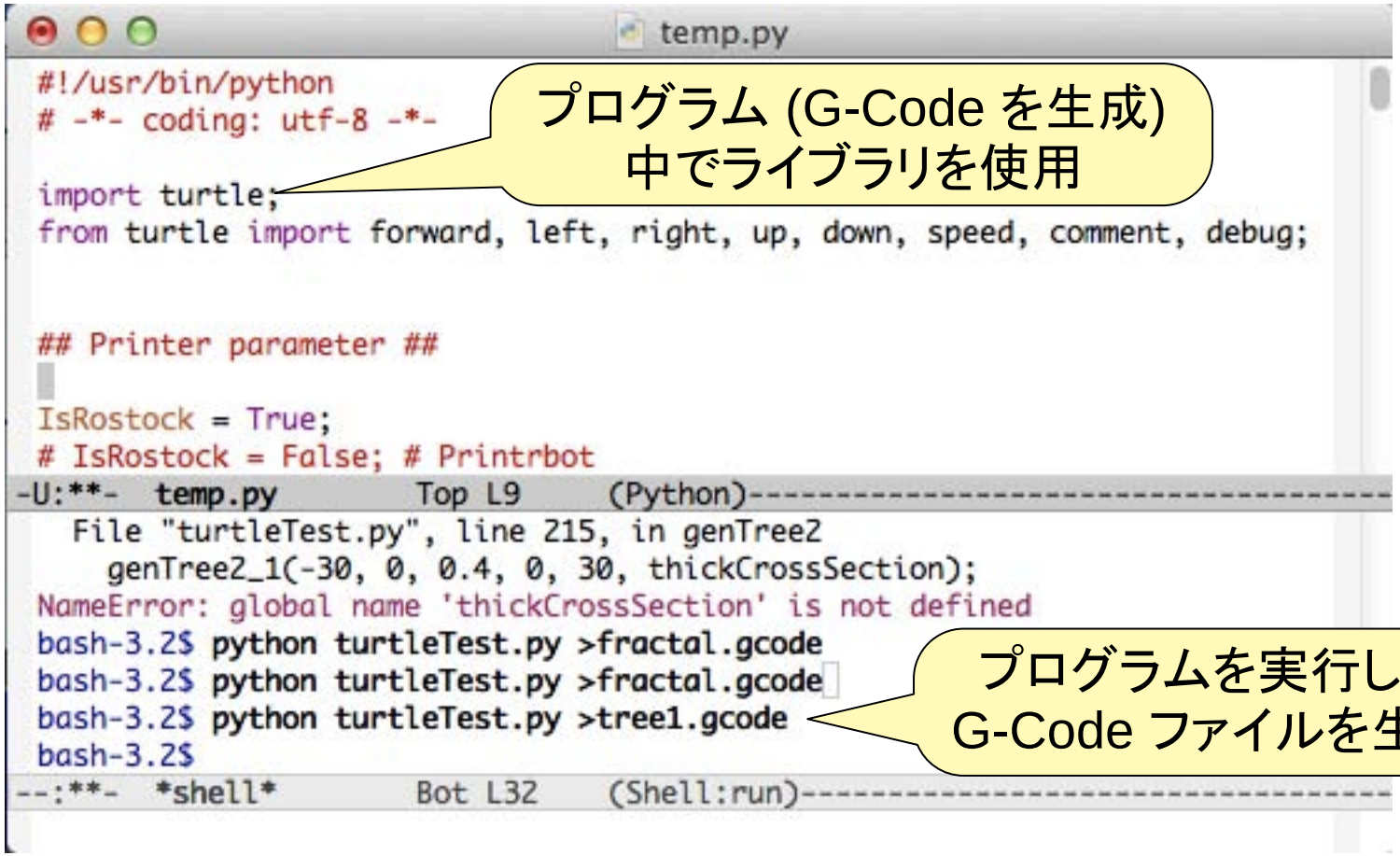






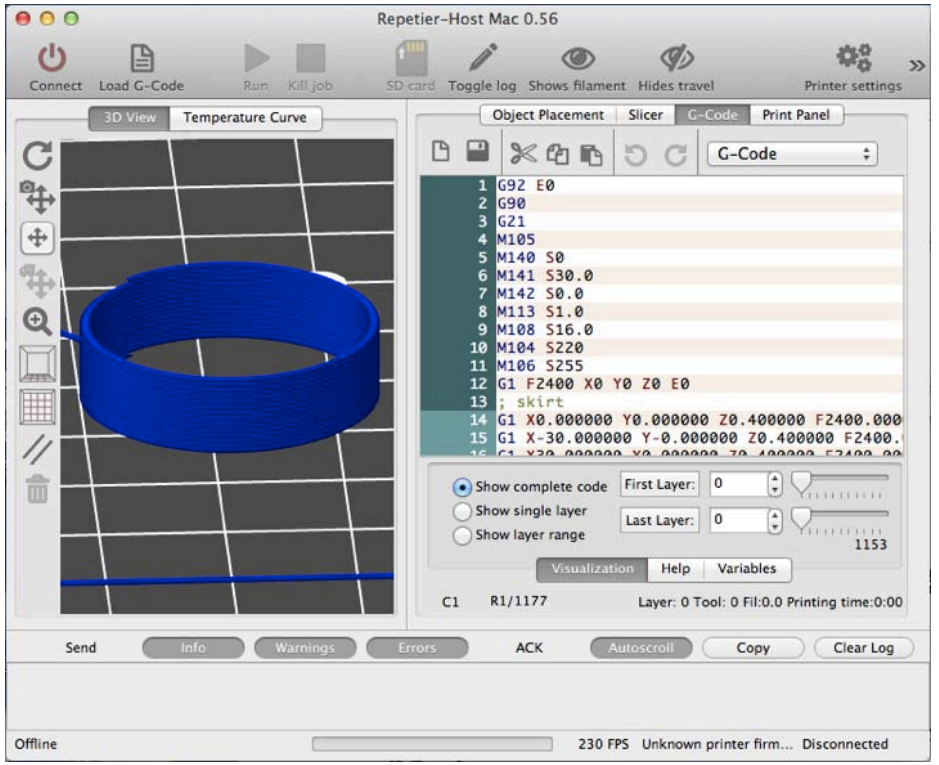
プログラムの記述

► エディタ / 開発環境による記述・デバグ



グラフィクスによる作品の確認

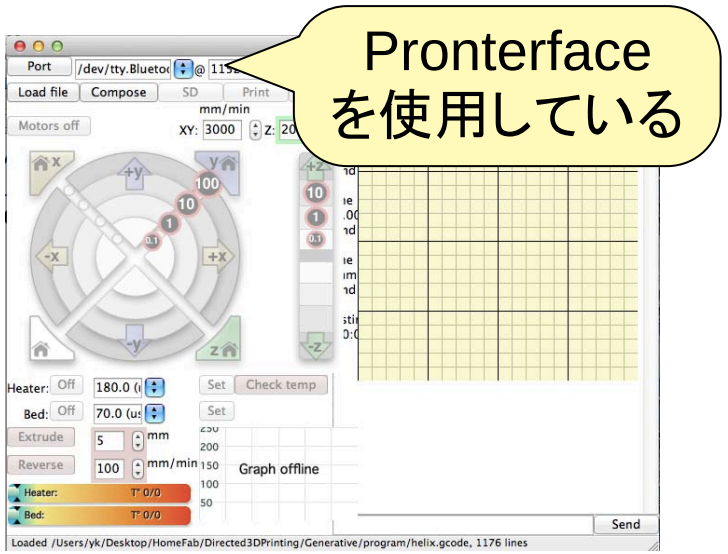
- **Repetier-Host という 3D プリンタ用ツールを使用して G-Code をグラフィック表示する.**

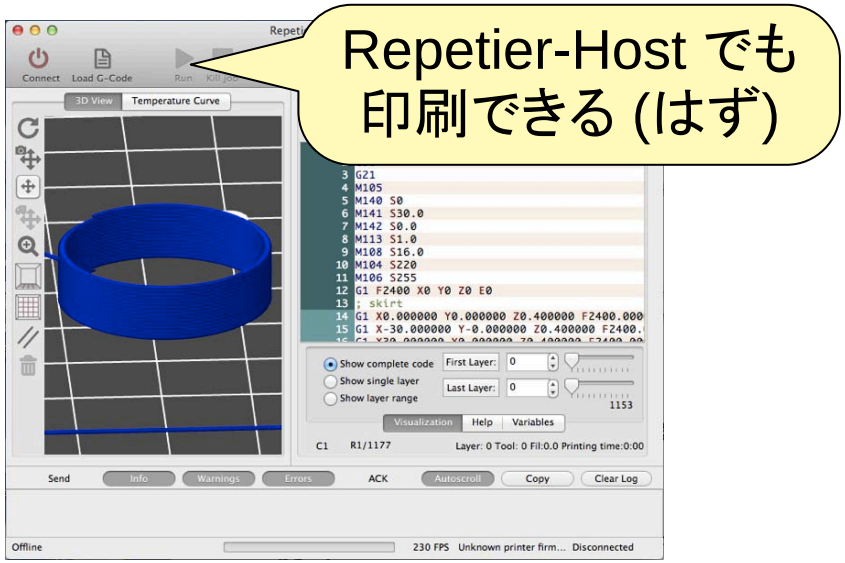


- **うまく印刷できそうかどうかを視覚的に確認する.**
  - ツールじたいではよわいチェックしかできない (コマンド構文がただしいかどうかなど).
  - ただし, ひとがみても印刷可能性が判定できないことも多い.
    - 印刷プロセスは動的だがグラフィック表示は静的だから.

作品の印刷

- 印刷用ソフトウェアで G-Code ファイルを指定して印刷する.

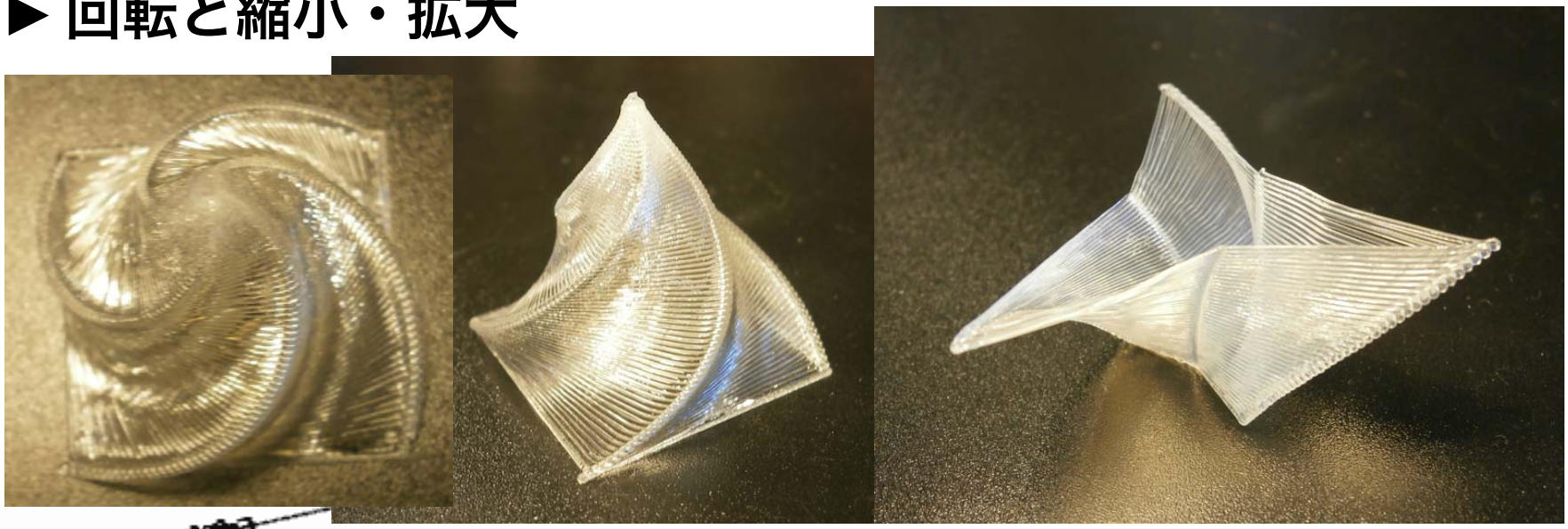


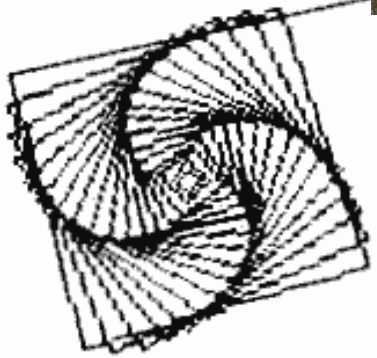


作品の印刷過程 (例)

作品 (印刷結果)

► 回転と縮小・拡大

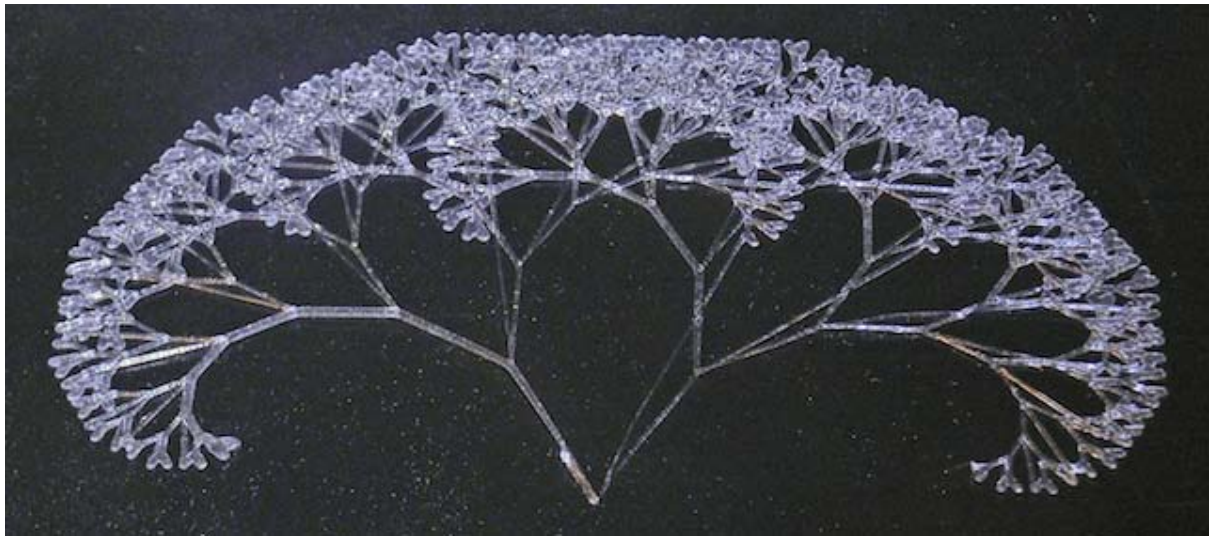




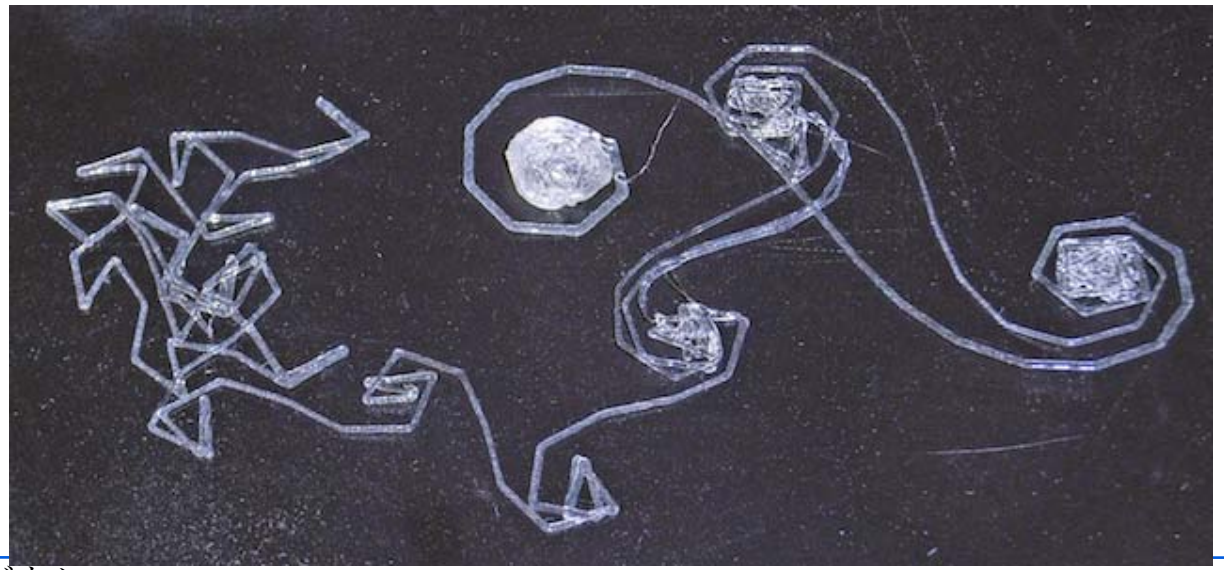
失敗作

作品 (印刷結果, つづき)

► 2 次元フラクタル図形

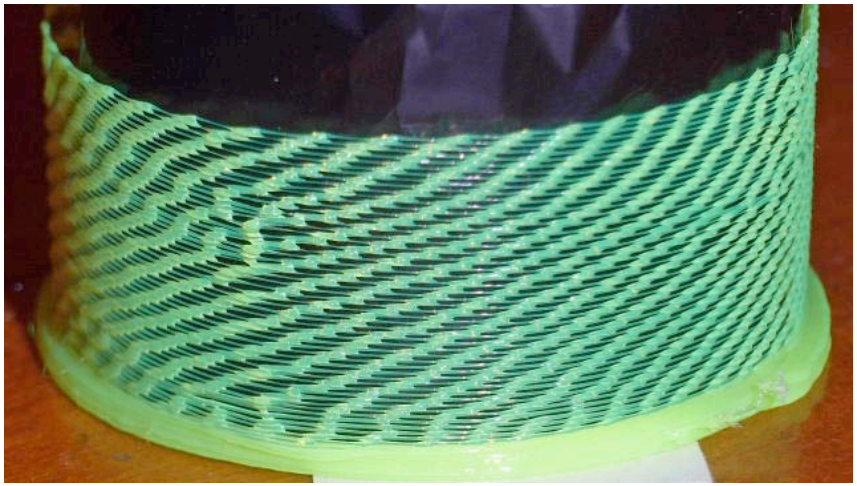


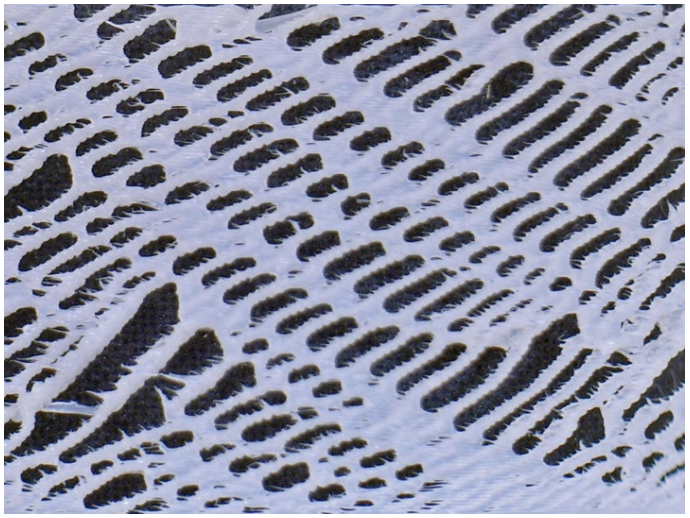
► その他
(2 次元図形)



おまけ: タートル・グラフィクス以外の作品

▶ 自己組織的 3D 印刷





Kanada, Y., “3D Printing and Simulation of Naturally-Randomized Cellular-Automata “, *19th International Symposium on Artificial Life and Robotics* (*AROB 2014*), 2014-1

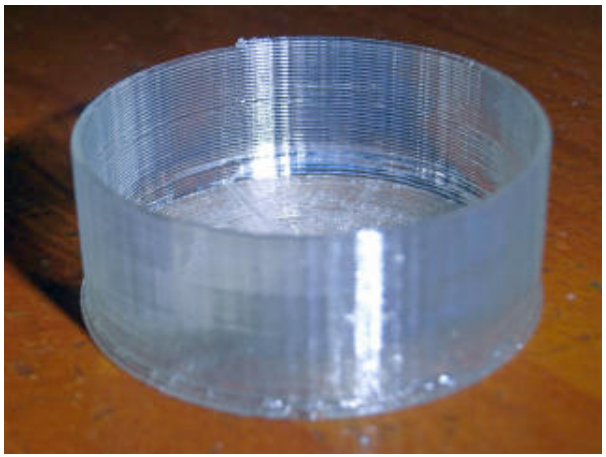
Kanada, “FDM 3D-printing as Asynchronous Cellular Automata”, Y., *8th International Workshop on Natural Computing* (*IWNC8*), 2014-3.

おまけ: タートル・グラフィクス以外の作品 (つづき)

- 印刷方向を指定した印刷

Kanada, Y., “Method of Designing, Partitioning, and Printing 3D Objects with Specified Printing Direction”, *2014 International Symposium on Flexible Automation (ISFA 2014)*, 2014-7.

- 印刷方向指定と変形による造形



まとめ

▶ 結論

- 3 次元タートル・グラフィクスのための Python ライブラリを開発した.
- このライブラリと G-Code ツール，3D プリンタ等をくみあわせれば 3D 印刷によるタートル・グラフィクス環境が実現できる.

▶ 今後の課題

- いろいろなかたち・支持法をためすこと.
  - 現在のライブラリをつかったくふう
  - ライブラリの拡張
- 極座標をためすこと.

- このスライドと論文のための URL:
  http://bit.ly/1sr1008 (http://www.kanadas.com/papers/2014/08/3d_3.html)