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
yasusi@kanadas.com

概要: 3D プリンタで造形するとき，通常は 3D CAD で設計した静的 (宣言的) なモデルのデータを加工してプリンタにおくる．しかし，普及している FDM 型 3D プリンタが入力するのはプリント・ヘッドの移動とフィラメントの射出を制御する動的な手続きであり，これをより素直にプログラミング言語化ないしライブラリ化すれば，タートル・グラフィクスのような方法で 3D オブジェクトが生成できる．この「タートル 3D 印刷」のライブラリを Python によって記述し試用してみた．このライブラリは公開している．3D プリンタでは宙に印刷できないことがネックになるが，その問題をうまくクリアできれば 3D タートル・グラフィクスでえがいた図形を実物にすることができる．

キーワード: FDM 型 3D プリンタ，タートル・グラフィクス，Fused Deposition Modeling，熱溶解積層型 3D プリンタ

1. はじめに

3D プリンタを使用して 3 次元のオブジェクトを造形するとき，通常は 3D CAD で設計したモデルをスライサとよばれるソフトウェアで水平にスライスして，その結果をプリンタにおくって印刷する．CAD が出力するファイル形式はさまざまだが，スライサにおくるには STL (Standard Triangulation Language または Stereo-Lithography) という宣言的な形式のファイルが使用される．STL はモデルの表面形状を 3 角形の集合によって近似する (内部は表現できない)．

3D プリンタにはいろいろな種類があるが，安価なタイプは FDM (Fused Deposition Modeling，熱溶解積層) 型とよばれ，とかしたフィラメント (プラスティック) をノズルの先端から射出してかためる (図 1)．FDM 型のプリンタをつかうとき，スライスした結果は通常 G-code [Kra 00] という CAM 用言語で表現される．CAD が出力するモデルは静的 (宣言的) だが，G-code はもともと工作機械の刃の動作をあらわすので動的 (手続き的) である．G-code によってプリント・ヘッドの動作やプラスティックを射出する速度などを指定することができる．

G-code によるコマンドの例として 2 つをあげておく．G0 というコマンドは単純なツールの移動を指令する．たとえば

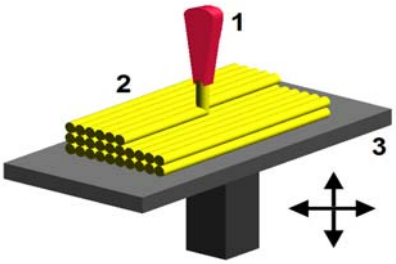


図 1　FDM 型 3D プリンタの原理
("FDM by Zureks" by Zureks - Wikimedia Commons)

G0 X0 Y0 Z0 F3600

というコマンドは分速 3600 mm で座標 (0, 0, 0) に移動することを指令する．
また，G1 というコマンドは切削型の工作機械においては加工しながらの移動を意味するが，付加型 (additive) の工作機械である 3D プリンタにおいては印刷しながらの移動を指令する．たとえば

G0 X0 Y0 Z0 F3600 E100

というコマンドを実行すると，E100 によって指定される量のフィラメントを射出しながら移動する．(フィラメントの量は指定にしたがって相対値または絶対値で指定される.)

FDM 型プリンタのプリント・ヘッドは通常はかぎられた方向にしかうごかないが，実はそれをもっと自由にうごかすことができる．FDM 型のプリンタでは通常，水平にスライスされた層ごとに印刷す

るので，層間の移動のとき以外はプリント・ヘッドが垂直方向にうごくことはない．しかし，G-code をつかえばヘッドを自由な方向に移動させることができる．たとえば，座標 (x_0, y_0, $\boldsymbol{z_0}$) にいるときにつぎのコマンドを実行すると，プリント・ヘッドは (x_1, y_1, $\boldsymbol{z_1}$) に移動する．

G0 Xx_1 Yy_1 Z$\boldsymbol{z_1}$

ただし，3D プリンタのなかには垂直移動が不得意なものが多いので注意が必要である．

2. タートル・グラフィクス

この章では (2 次元) タートル・グラフィクスとその 3 次元への拡張について説明する．

2.1 2D タートル・グラフィクス

タートル・グラフィクスは 1960 年代に Seymore Papert (パパート) らによって導入された．Papert はこどもでもつかえるようにプログラミング言語 Logo を設計した．Logo をつかうと，「カメ」の軌跡で 2 次元の線画をかかせることができる．これがタートル・グラフィクスである．

タートル・グラフィクスの基本的な描画コマンドはつぎの 3 つである．

- Forward d というコマンドにより，カメは距離 d だけ前進する．
- Turn left a というコマンドにより，カメは角度 $a°$ だけ左にまがる．
- Turn right a というコマンドにより，カメは角度 $a°$ だけ右にまがる．

これらのコマンドを使用することによって，カメを 2 次元空間のなかで自由にうごかすことができ，**図 2** の例のようにその軌跡を表示することができる．

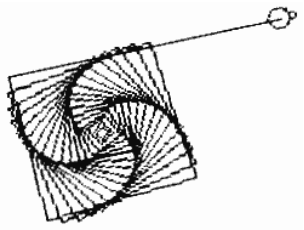


図 2　2 次元タートル・グラフィクス

2.2 3D タートル・グラフィクス

タートル・グラフィクスはもともと 2 次元のものだったが，その後，「3 次元タートル・グラフィクス」によって 3 次元の図形をえがくのにも使用されるようになった (たとえば [Ver 14] [Tip 10])．タートル・グラフィクスを 3 次元に拡張するには，基本的には forward, (turn) left, (turn) right のほかに上下方向に移動または回転するコマンド (たとえば up, down) を追加すればよい．さらに，Bernd Paysan は，複雑な 3 次元形状を容易につくりだすことができる “Dragon Graphics” [Pay 09] という拡張された 3D タートル・グラフィクスを提案している．しかし，これらはいずれも 2 次元のディスプレイで表示するためのグラフィクスであり，カメの軌跡が 3 次元で表示されるわけではない．

3. 3D 印刷による “タートル・グラフィクス”

この節では 3D タートル・グラフィクスにもとづく 3D 印刷機能を検討し，その設計・実装についてのべる．この機能をタートル 3D 印刷機能とよぶ．

3.1 概要

G-code は手続き的なので，タートル・グラフィクスと同等のコマンドを実行させることができる．もっとも，G-code を人間が直接書くことは普通はないし，アセンブリ言語のようなものなので人間が書くのに適してはいない．しかし，3 次元の描画コマンドを G-code に翻訳するのは容易である．タートル・グラフィクスの場合には，Forward コマンドを G1 コマンドに変換するだけでよい．

ただし，プリント・ヘッドの座標は通常はデカルト座標によって記述するので，それをタートル・グラフィクスにおけるように進行方向を基準にする必要がある．しかし，これもつぎのようにすることにより，容易に実現される．第 1 に，カメの方向を G-code 生成プログラムが記憶するようにする．第 2 に，forward コマンドを翻訳する際に現在の座標と方向とからつぎの座標を計算して G1 コマンドの引数とする．第 3 に，turn left および turn right コマンドを翻訳する際には，記憶している方向を修

正する．上下の動作に関しても同様だが，それについて記述するには，まず座標系を選択する必要がある．

3.2 座標系の選択と上下の移動

カメの座標系として極座標を使用する方法と円筒座標を使用する方法とがあり，上下の移動を記述する方法はそれに依存する．

極座標はフライトシミュレータと同様の座標系であり，カメの移動方向は重力の方向とは無関係にきまる．3D 印刷においては重力を考慮することが欠かせないので，これでは印刷可能性を保証するのが困難になる．つまり，フライト・シミュレータでは (あるいはたぶん実際に飛行機を操縦していても) 鉛直方向がわからなくなって墜落しがちだが，3D 印刷でも同様のことがおこる．

円筒座標においてはカメの移動方向はつねに水平であるとする．垂直方向に移動するときはその変位を記述するが，カメの方向は変化しない．この方法では重力の方向がカメに対して一定なので，極座標をつかうより印刷すべきオブジェクトがデザインしやすいとかんがえられる．

3.3 タートル 3D 印刷と 3D タートル・グラフィクスとのちがい

上記のように，3D プリンタをつかえば，3 次元タートル・グラフィクスのような方法でオブジェクトをかたちづくっていくことができるが，タートル 3D 印刷を 3 次元タートル・グラフィクスと比較すると 2 つのちがいがある．

第 1 に，タートル・グラフィクスにおいては 3 次元空間のどこにでも自由に線をひくことができるが，3D 印刷においては地上で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 (サポート) が必要であり，空中に印刷するのは困難である．

第 2 に，タートル・グラフィクスにおいては印刷速度やフィラメント射出量などの印刷パラメタを制御する必要がある (制御することができる)．タートル・グラフィクスにおいても線のふとさなどをかえることができるが，3D 印刷ではきれいに印刷するには印刷速度をおさえる必要があるところなど，タートル・グラフィクスにはない条件がある．

4. 選択肢とライブラリの設計

この章ではタートル 3D 印刷機能の提供形態に関する選択肢についてのべ，そのなかから選択して実装した Python ライブラリについてのべる．

4.1 タートル 3D 印刷機能のための選択肢

タートル 3D 印刷は手続き的に記述するのが自然であるから，Logo や他のタートル・グラフィクスと同様にプログラミング言語の一部として提供するのが自然である．それを前提とすると，2 つの選択肢がかんがえられる．

- 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．

この選択肢のうち後者を選択したが，その理由はつぎのとおりである．Logo が設計された時代にはまだプログラミング言語の数はすくなくて，拡張してタートル・グラフィクスをくみこむのが容易でないために新言語をつくるという選択がなされたのであろう．しかし，現在では拡張可能な言語は多数あり，新言語を導入するべき理由はないとかんがえられる．ひろく使用されている言語にくみこんだほうが，使用しやすいとかんがえられる．それがライブラリという方法を選択した理由である．

言語としては Python を選択したが，それは，モダンな言語のなかで Python が比較的普及しているからである．もちろん，ほかにも普及している言語はあるが，そのなかから Python を選択したのは趣味の問題でもある．

4.2 Python のためのライブラリ

著者はタートル 3D 印刷のための Python ライブラリ turtle.py を開発して，オープン・ソースで提供している (http://bit.ly/ZEyLzx or http://www. -kanadas.com/program/2014/08/-3d_3d_python.html)．ただし，まだ特定の 3D プリンタ (Rostock MAX) でしかつかえないなど，完成度がひくいこともあり，いまのところはとくに宣伝してはいない (それにしても，Google で検索してみたが容易にみつからないので，すこしくふうが必要なようだ)．

このライブラリを使用することによって G-code の

プログラムを生成することができる．すなわち，つぎのようなプログラミングが可能になる．カメの進行方向を前方とする円筒座標によってプリント・ヘッドの移動方向を記述する．forward(r, z) によって z 方向に移動しながら前進する (G-code を生成する) ことができる．

たとえば，螺旋をえがきながらフィラメントをかさねていくプログラムはつぎのように記述することができる．

```
init(FilamentDiameter,
   HeadTemperature, BedTemperature,
   CrossSection, 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)
```

init は初期化のための関数であり，あとで説明する．このプログラムは 5° ずつ 72 回まがりながら近似的な円をえがいていく 2D タートル・グラフィクスのプログラムにちかい．ち がうのはまっすぐ前進するかわりに dz (mm) ずつ上昇していくことである．フィラメントのふとさは 0.4 mm であることを仮定している．72 回の移動でちょうど 0.4 mm 上昇することによって，空中に印刷することなく，フィラメントをきちんとかさねていくことができる．実際には射出したフィラメントのふとさは 0.4 mm よりすこしふとい必要がある (射出量を調整してそのようにする)．つまり “層” のあつさが 0.4 mm ということだが，螺旋なので正確には層になっていない．そうすればフィラメントどうしがおしつけられて接着し，かんたんにははがれなくなる．

初期化関数 init によってつぎの値を調整することができる．最初の引数 FilamentDiameter はプリンタにおいて使用するフィラメントのふとさであり，通常は 1.75 mm または 3 mm である．

2 番めおよび 3 番めの引数はそれぞれ設定するべきプリント・ヘッドとプリント・ベッド (印刷台) の温度である．FDM 型のプリンタでは通常 ABS または PLA (ポリ乳酸) というプラスティックを使用するが，ABS は 240°C くらい，PLA は 200°C くらいで使用する．プリント・ベッドは PLA については加熱しない (0 を指定) でよいが，ABS については 80–110°C くらいの温度に加熱する．ABS は温度がさがると収縮してプリント・ベッドからはがれるからである．

4 番めの引数 CrossSection は射出するフィラメントの断面積を指定する．これはフィラメントの射出量を制御するためにあたえる．FDM 型 3D プリンタにおいては装着されたフィラメントのおくり量 (ながさ) によってその量を指定するが，これではフィラメントのふとさ (1.75 mm または 3.0 mm) によって射出量の指定をかえなければならない．そこで，射出されるフィラメントの断面積によって射出量を制御するようにしている．こうすればフィラメントのふとさをかえたときは init の最初の引数の値をかえるだけですむ．

最後の 3 つのパラメタは印刷を開始する位置の座標である．これらはデカルト座標によってあたえる．

上記のように印刷パラメタのうちフィラメント射出量の初期値は init によって指定することができるが，それは印刷中に変更する必要が生じる．また，印刷速度は初期値も固定されているので，最初から指定する必要が生じることもある．これらは，必要なときは setVelocity および setCrossSection という関数を使用して変更することができる．

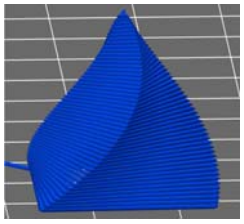
5. 実験

いくつかの図形を例として，turtle.py を使用してみた．他にもいくつかフリーのツールを使用したその開発・印刷過程と結果を報告する．

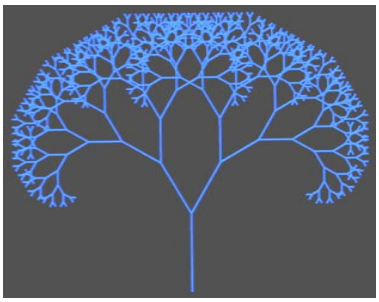
5.1 例題

円筒 (螺旋)，ねじれ四角錐，平面フラクタルなどの例題をこころみた．**図 3** にこれらをグラフィクスによって表示したものをしめす．図 3(c) のような 2 次元のフラクタルは 3D プリンタで印刷するオブジェクトとしてはいまひとつであり，フラクタル図形としてはほんとうは 3D フラクタルをためしたい．しかし，著者が知るかぎりの 3D フラクタルはみな宙に印刷する必要があるため，通常の 3D 印刷ではつくれない．

(a) 円筒 (螺旋)



(b) ねじれ四角錐



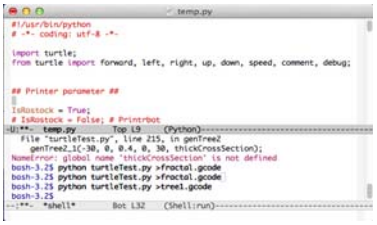
(c) フラクタル図形 (2 次元)

図 3　タートル 3D 印刷のための例題 (Repetier Host により表示)

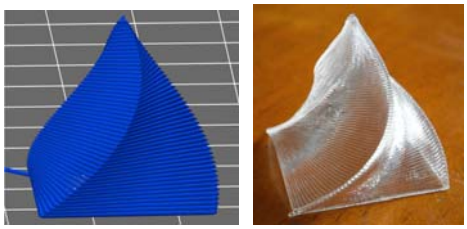
5.2　方法

基本的な手順はつぎのとおりである (図 4).

1) プログラム記述と G-code 生成
2) グラフィクスによる確認
3) 3D 印刷



1) プログラム記述



2) グラフィクスによる確認　　3) 3D 印刷

図 4　タートル 3D 印刷の手順

すなわち, まずプログラムを記述し, そのプログラムを使用して G-code を生成する. そして, その G-code をそれ用のツールで表示して形状を確認する. 成功したら印刷してみる. しかし, この手順を 1 回実行するだけで成功することはまずない. したがって, これらを成功するまでくりかえした.

1 回で成功しないのは, ツールでの表示が成功しても印刷をだめにする要因が多々あるからである. init は 3D プリンタを完全に初期化するわけではないので, 初期化わすれもおこるが, これは容易に解決できる. ABS と PLA のあいだではもちろん, 同種のプラスティックでも温度設定をかえなければならないこともあるが, これも調整は比較的容易である. やっかいなのは, フィラメントをずらしてかさねていくため, うまくかさねられずに陥没したり脱落したりすることである. その例はあとでしめす. 陥没することがさけられない形状のときは, プリント・ヘッドの上昇ピッチをさげなければならないこともある.

以下, これらのステップをよりくわしくみていく.

5.2.1　プログラムの記述と G-code 生成

プログラムを記述し実行するためには, すきな Python 用のエディタや開発環境をつかえばよい. 著者の趣味は Emacs なので, それをつかって記述しコンパイルしている (図 5). そのプログラムを実行すると標準出力から G-code がえられるので, それをファイルにいれる. とくにつけくわえることはない.

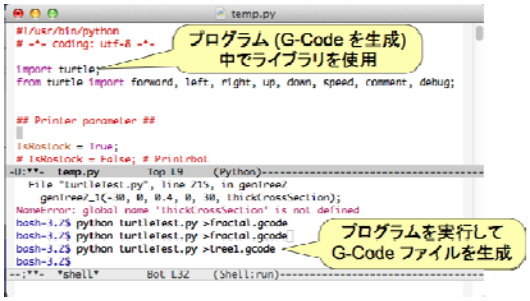


図 5　エディタ / 開発環境による記述・デバグ (Emacs)

5.2.2　グラフィクスによる作品の確認

G-code を入力してグラフィクスで表示するプログラムとして比較的便利なのは Repetier Host という 3D プリンタ用ツールである (図 6). Repetier Host は Windows, Macintosh のいずれでも動作する. 類似のツールはほかにもあるとかんがえられるが, Repetier Host は表示機能も多様であるうえ, これをつかえば確認だけでなく印刷までできる (はずである). ウィンドウの右側に表示されているのは G-code である.

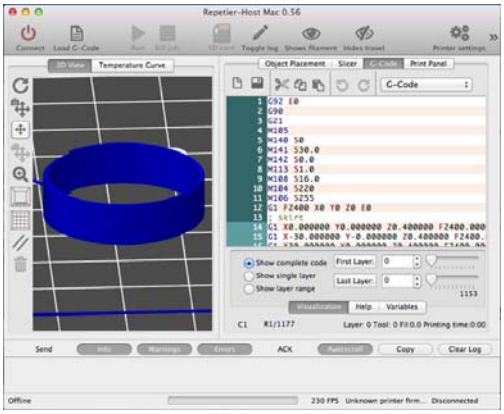


図 6　Repetier Host を使用した
グラフィクスによる作品の確認

このステップではうまく印刷できそうかどうかをツールの機能をつかって確認し, 視覚的に確認する. しかし, ツールじたいではコマンド構文がただしいかどうかなど, よわいチェックしかできない. また, ひとがみても印刷可能性が判定できないことも多い. その理由のひとつは, 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ということである.

余談だが, Repetier Host は入力された G-code にふくまれるスライスの数 (層数) をもとめようとする. タートル 3D 印刷用の G-code においては z 軸方向に自由に移動できるので奇妙なことがおこるが, それでも, 膨大な層数をかぞえるなどして, なんとか追随してくれる.

5.2.3　作品の印刷

すでに書いたように 3D 印刷のためのプログラムとしては Repetier Host がつかえるはずだが, 著者の環境 (Windows Vista) ではよく途中でとまるので印刷にはつかっていない (ただし, 最新版でどうなるかはたしかめていない). 途中でとまると印刷を最初からやりなおさなければならない. もっとも, 前記のようなオブジェクトにおいては印刷時間が数分程度なので, そうなっても印刷に何時間かかかるおおくの 3D 印刷よりは被害はすくない.

それでも, 失敗がないようにするため, Pronterface という印刷用ソフトウェアを使用している (図 7). これも Windows, Macintosh の両方で動作する. Pronterface も完全に安定しているわけではないが, 不具合はあっても印刷がとまることはまれである.

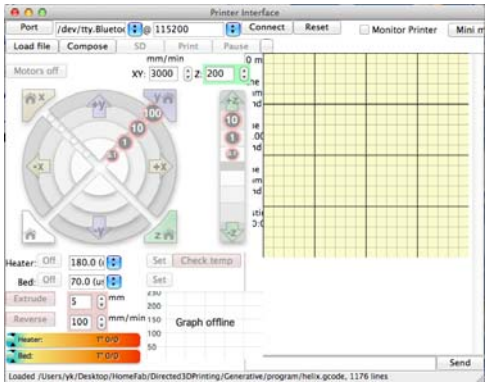


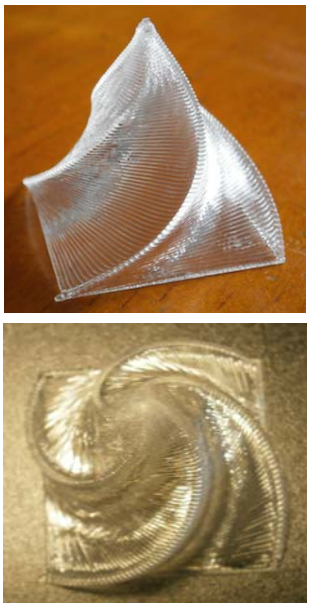
図 7　Pronterface による作品の印刷

5.3　印刷過程と結果

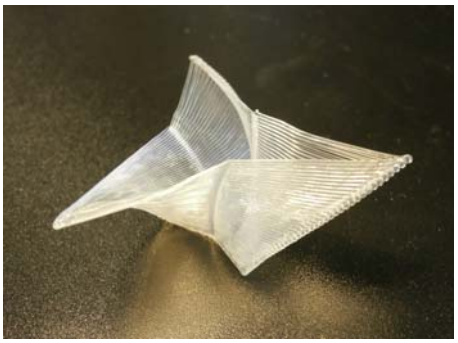
印刷のようすをビデオで撮影した. ここにのせることはできないが YouTube に投稿している (http://youtu.be/7H5-acxQ_RE).

円錐やねじれ四角錐のように前進と回転をくりかえしてつくる図形の例を図 8 にしめす. 円錐の場合はおなじパターンをくりかえしていくが, ねじれ四角錐の場合 (a) はパターンが縮小していく. (a) の 2 番めの写真は最初の写真とおなじ四角錐をうえからうつしたものだが, こうすると図 2 の平面図形との関係がよくわかる (ただし, 図 2 は拡大する方向にえがいている). 同様に拡大していくパターンもつくることができるが, その例を (b) にしめす. ただし, 接地面積がちいさくて印刷中にはがれやすいので, 注意が必要である. グラフィクス

の場合は縮小しながらかいても拡大しながらかいてもおなじ結果がえられるが，タートル 3D 印刷の場合はこのようにしたからつみあげていくので縮小と拡大とでことなる形状が生成される．



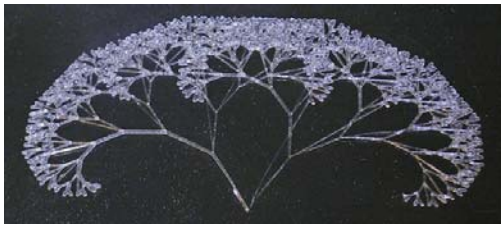
(a) ねじれ四角錐 (縮小していくパターン)



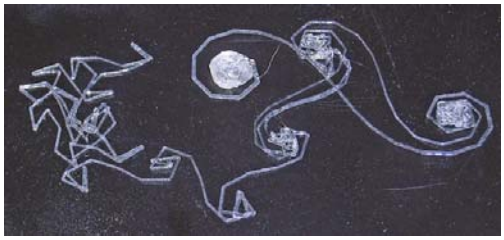
(b) 拡大していくパターンの例

図８　印刷結果 – 回転と縮小・拡大

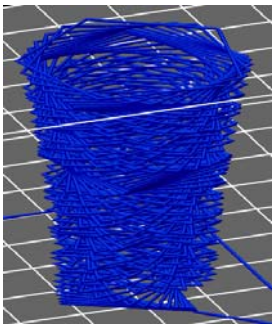
図９ はもうすこしかわったつかいかたの例である．(a) はすでにしめした平面フラクタルである．この場合は枝が分岐していくので，明示的に分岐点にもどって印刷する必要がある (プログラミング言語の環境は手続きよびだしの実行が終了すれば自動的にもとにもどるが，turtle.py ではそれができないので明示的にもどさなければならない)．もどるときに糸 (というほどほそくないが) をひいてしまっている．これをなくすには，さらにくふうが必要である．



(a) 2 次元フラクタル図形



(b) 前進と角度が拡大する回転による 2 次元図形



(c) 上下のフィラメントのかさなりがすくない図形

図９　他の印刷例

(b) は前進と回転をくりかえしていくときに回転角をだんだんおおきくしていくときにできる，著者がすきな 2 次元パターンである．プログラミング・シンポジウムの原稿でも，あまったスペースに同様

のパターンをえがいたことがある [Kan 85].

3 次元のときはきちんとつみかさねるパターンだけ, それ以外では 2 次元のパターンしかつくれないというのではつまらないので, (c) はフィラメントが完全にはつみかさならない疎な 3 次元パターンをつくってみた例である. まだグラフィクスを表示していなかったので, ここにあわせてあげる. 最初は 0.4 mm ごとにフィラメントがかさなるようにデザインしたが, それではつぶれてしまう. そこで, フィラメントのふとさは 0.4 mm のまま, "層" のたかさは 0.3 mm ごとにして, マクロにはただしい形状がえられるようにしている. それでも, 直下のフィラメントからはずれたフィラメントはすこし下に変形している. かさなっている部分をさらにへらすとさらに変形するが, この程度だとまだほぼデザインしたとおりのかたちのようにみえる.

どういう失敗がおこりうるかをしめすのも重要だとおもうので, 典型的な例をあげる. 図 9(c) はある意味では印刷に失敗した例だが, 意図的である. それに対して**図 10** の最初の写真は意図していなかったあきらかな失敗例である. つまり, 円筒をかくときにすこしずつ直径を拡大していくと, 2 番めの写真のようにある程度まではただしくつみかさなるが, 限界をこえるとつみかさならなくなる. フィラメントをちぢめる方向にちからがかかっているときは, この写真のようにフィラメントが弧をえがくかわりに一部がショートカットされて線分になってしまう.

6. 関連研究

3D プリンタは付加型加工 (additive manufacturing, AM) をおこなう工作機械とかんがえられているが, コンピュータ数値制御 (CNC) による工作機械の歴史は旋盤, フライス盤などの切削型加工からはじまっている. 1950-60 年代には切削型ツールのためのプログラミング言語がさかんに研究されていた. そのなかで代表的なものが MIT において開発された APT [Bro 63] である. タートル 3D 印刷はツールの移動や加工を手続き的に指示する点で APT にちかいが, 付加型加工を目的としている点や使用している座標系がことなっている. また, turtle.py は汎用言語のライブラリであるという点で CNC 専用言語として設計されている APT とはちがいがある (専用言語を設計した点で Logo にちかいともいえる). しかし, タートル 3D 印刷は 1960 年代以降 CAD の発展によってわすれられていた手続き型言語による機械加工のリバイバルだともいえるだろう.

図 10　印刷失敗例と成功例

7. 結論

タートル・グラフィクスにもとづいて 3D オブジェクトを生成するための「タートル 3D 印刷」のライブラリを Python によって記述し試用してみた. 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, タートル 3D 印刷の開発環境を実現することができる. 3D プリンタでは宙に印刷できないことがネックになりうるが, その問題をうまくクリアできれば 3D タートル・グラフィクスでえがいた図形を実物にすることができる.

今後の課題としては, いろいろなかたち, とくに上下のフィラメントのかさなりがすくない図形をためすこと, 現在のライブラリをつかったくふうやライブラリの拡張, 極座標をためすことなどがある.

なお, このスライドと論文のためにつぎの URLを用意している．補足などがあれば, ここに記述する． http://bit.ly/1sr1008 (http://www.kanadas.-com/papers/2014/08/3d_3.html)．また, この報告ではカメを中心とする座標系だけをあつかったが, デカルト座標をつかったほうがえがきやすい図形がある (むしろそのほうが多いとかんがえられる)．そのためのライブラリや方法もあわせて開発している．

質疑・応答

受けた質問・意見を記録したメモをなくしてしまったので再現することができない．しかし, 3 件のうち 2 件は, 指示されたとおりにプリント・ヘッドをうごかすかわりに印刷コマンドないし印刷データに適切な編集をほどこしてから印刷するほうがよいのではないか (とくに面としてあつかうのがよいのではないか) という内容だったと記憶している．こうすることで指示されたとおりに直接印刷するより印刷可能性に関する制約をへらせるのではないかということである．

タートル 3D 印刷では印刷可能であるためにきびしい制約があり, それを解消もしくは軽減することに質問・意見があつまったのだろう．発表当日はあいまいなこたえしかできなかったが, いまかんがえなおすと, 複雑なしかけをいれることはタートル・グラフィクスの直観性をそこなうようにおもう．安定な印刷のためには上下のフィラメントがかさなる必要があり, 必然的に面を形成する．しかし, それは本来, タートル・グラフィクスがめざすべきものではない．面をあつかうにはむしろデカルト座標のもとで部品をくみたててつくるほうが適していて, それはタートル・グラフィクスをめざしているこの研究とはべつものだとかんがえられる．

タートル 3D 印刷じたいを発展させるには, 3D プリンタを改良して宙に印刷できるようにするべきだろう．実際, 宙に印刷できるプリンタも開発されている．

参考文献

[Bro 63] Brown, S. A., Drayton, C. E., and Mittman, B., “A Description of the APT Language”, *Communications of the ACM*, Vol. 6, No. 11, pp. 649−658, 1963.

[Kan 85] 金田 泰, “スーパー・コンピュータによる Prolog の高速実行”, 第 26 回プログラミング・シンポジウム報告集, pp. 47−56, 1985.

[Kra 00] Kramer, T. R., Proctor, F. M., and Messina, E. “The NIST RS274NGC Interpreter - Version 3”, NISTIR 6556, August 2000.

[Pay 09] Paysan, B., ““Dragon Graphics”, Forth, OpenGL and 3D-Turtle-Graphics”, August 2009, http://bernd-paysan.de/dragongraphics-eng.pdf

[Tip 10] Tipping, S., “Cheloniidae”, December 2010, http://spencertipping.com/cheloniidae/src/-cheloniidae.pdf

[Ver 14] Verhoeff, T., “3D Flying Pipe-Laying Turtle”, Wolfram Demonstrations Project, http://demonstrations.wolfram.com/3DFlyingPipeLayingTurtle/