Yasusi Kanada

Designing 3D-Printable Generative Art by 3D Turtle Graphics and Assembly-and-Deformation

Topic: 3D design, 3D printing

Authors:
Yasusi Kanada
Dasyn.com
Tokyo, Japan

Main References:
[1] Yasusi Kanada, *""3D Turtle Graphics" by using a 3D Printer"*, Int. Journal of Engineering Research and Applications, Vol. 5, No. 4, Part 5, pp.70-77, April 2015.
[2] Yasusi Kanada, "*Support-less Horizontal Filament-stacking by Layer-less FDM"*, International SFF Symposium 2015, August 2015.

Abstract:
3D models are usually designed by 3D modelling tools, which are not suited for generative art. This presentation proposes two methods for designing and printing generative 3D objects.

First, by using a turtle-graphics-based method, the designer decides self-motion (self-centered motion) of a turtle and print a trajectory of the turtle as a 3D object (Fig. A). The trajectory is printed using a fused-deposition-modelling (FDM) 3D printer, which is the most popular type of 3D printer.

Second, by using the assembly-and-deformation method, the designer assembles parts in a palette, each of which represents stacked filaments, applies deformations to the assembled model, and prints the resulting object by an FDM 3D printer. The designer can also map textures, characters, or pictures on the surface of the object. Various shapes can be generated by using the assembly-and-deformation method. If the initial model is a thin helix with a very low cylinder (*i.e.*, an empty cylinder with a bottom), shapes like cups, dishes, or pods with attractive brilliance can be generated, and a globe and other shapes can be generated from a helix (Fig. B).

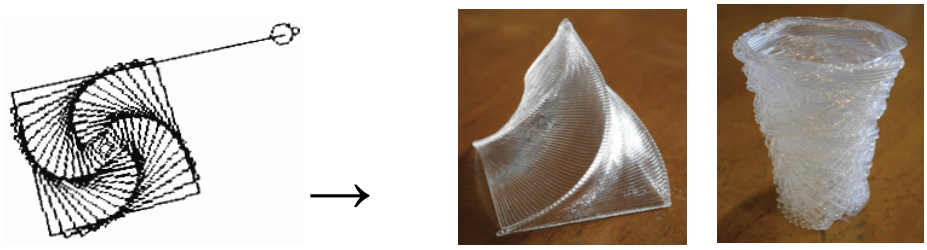


Fig. A Turtle-graphics-based prints

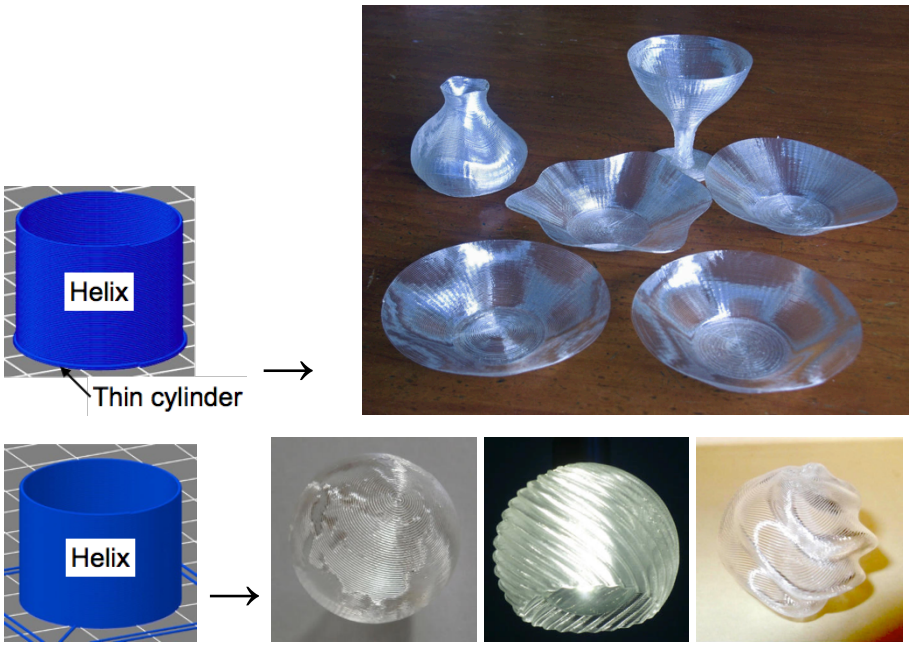


Fig. B Assembly-and-deformation (and texture-mapping) based prints

Videos:
Turtle-graphics-based method: http://youtu.be/7H5-acxQ_RE (skewed pyramid)
Assembly-and-deformation method:
http://youtu.be/5P1vaahzW98 (dish), *http://youtu.be/YWx1vqig2-o* (globe)

Contact:
yasusi@kanadas.com

Keywords:
Design, Directed 3D printing, Fused deposition modelling (FDM)

Designing 3D-Printable Generative Art by 3D Turtle Graphics and Assembly-and-Deformation

Yasusi Kanada, Ph.D.
Dasyn.com
Tokyo, Japan
e-mail: yasusi@kanadas.com

Abstract

3D models are usually designed by 3D modelling tools, which are not suited for generative art. This presentation proposes two methods for designing and printing generative 3D objects. First, by using a 3D turtle-graphics-based method, the designer decides self-motion (self-centered motion) of a turtle and print a trajectory of the turtle as a 3D object. The trajectory is printed using a fused-deposition-modelling (FDM) 3D printer, which is the most popular type of 3D printer. Second, by using the assembly-and-deformation method, the designer assembles parts on a palette, each of which represents stacked filaments, applies deformations to the assembled model, and prints the resulting object by an FDM 3D printer. The designer can also map textures, characters, or pictures on the surface of the object. Various shapes can be generated by using the assembly-and-deformation method. If the initial model is a thin helix with a very low cylinder (*i.e.*, an empty cylinder with a bottom), shapes like cups, dishes, or pods with attractive brilliance can be generated, and a globe and other shapes can be generated from a helix. Python APIs for these methods have been publicly available.

1. Introduction

2D generative objects (artwork) can be generated by using Processing [Pea 11], which is a procedural programming language. Generated objects can be shown by a computer display or realized by a printer. However, the generation process can be observed only in the virtual world. The generation process can be embodied in the real world by turtle graphics (and by Logo programming language, which was designed for children), which was developed in 1960s by Seymore Papart and his group [Pap 80]. 3D graphical generative art can be generated by the 3D functions (P3D) of Processing or by 3D turtle graphics [Ver][Tip 10]. They can create various 3D graphical shapes; however, they cannot create real-world shapes.

This study aims generatively forming real-world 3D generative design by using 3D printers; thus, 3D generative design is first explained. There are two types of 3D design tools, *i.e.*, manual-design tools and generative (or algorithmic) design tools, and they can be used for both free-form design and design by parts assembly. Manual-design tools are used for specifying parts and models. They are used for top-down design. Many commercial and free manual-design tools, which are called 3D computer-aided design (CAD) tools, such as AutoCAD, SolidWorks, or Blender, are available. Models can be created as free-form models, *i.e.*, freely designed by drawing lines or other shapes by using manual-design tools by pointing devices, or created by assembling predefined parts and specifying parameters manually. In contrast, generative-design tools draw graphics, generate models, or specify parts algorithmically using a declarative or procedural language. They can be used for bottom-up design. The users of the tools, *i.e.*, the programmers, write programs to generate models. For example, 3D graphics can be generated by using Processing [Pea 11] (P3D), and 3D models can be generated by using OpenSCAD [Wik].

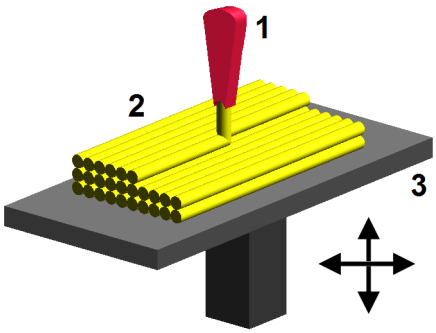
This study aims not only designing artwork by a generative method but also printing the design in a generative method. A conventional 3D-printing method can be used for printing a generative objects; that is, the design can be expressed by a static (declarative) language, *i.e.*, STL (Standard Triangulation Language or Stereo-Lithography), and the expression, *i.e.*, an STL file, can be “sliced” to layers by a so-called “slicer”, and can be printed layer-by-layer by a 3D printer. However, this complicated process may spoil the generative-ness of the original design. The author intend to print the design in a more straightforward method that reflects the generative-ness of the original design.

The rest of this paper is organized as follows. This paper first introduces popular conventional 3D printing method called FDM (fused deposition modeling) in Section 2, and then describes two methods that are suited for creating generative objects by using 3D printers. The first method, which is based on 3D turtle graphics, is described in Section 3. Designers (programmers) can directly and intuitively create 3D shapes by this method in a similar way to Processing. However, unfortunately, it is necessary for the designer to be very much careful about supporting printed filaments, otherwise they are easily drop down and the shape can easily become unprintable. In contrast, the second method described in Section 4, which is based on deformation of assembled 3D parts, can more easily managing printability. Section 5 concludes this paper.

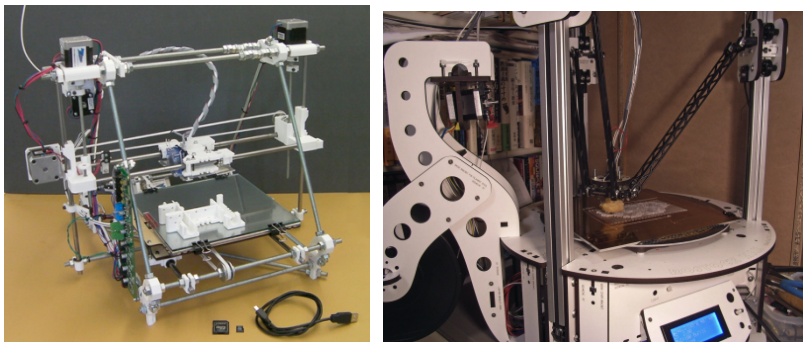
2. Conventional 3D Printing

When creating solids by using a 3D printer, conventionally, a model designed by a three-dimensional computer-aided design (3D CAD) system is horizontally sliced by using a program called a “slicer” and the resulting file is sent to the printer. Although there are various output formats for 3D design data outputted by CAD systems, the slicer usually accepts a file described by STL, which is a declarative language. STL approximates the surface shape of the model by a collection of triangles. (It cannot express the inner structure of 3D shapes.)

Although the model outputted from a CAD system is static (declarative), computer-aided manufacturing (CAM) programs for 3D printers are dynamic (procedural) because they create products operationally. There are various types of 3D printers; however, most of cheaper printers belong to the FDM type (**Fig. 1**(b)). FDM-type printers extrude melted filament (plastic) from a tip of a nozzle and solidify it (Fig. 1(a)). When using an FDM-type printer, the object to be printed is sliced horizontally and represented by G-code [Kra 00], which is a language for computer-aided manufacturing (CAM) and originally used for conventional machining tools such as milling machines. The model outputted from a CAD system in STL or other format is static (declarative). In contrast, because G-code originally expresses motion of machine tools, it is intrinsically dynamic (procedural/operational). The motion of a print head and the velocity of plastic extrusion can be specified by G-code.



(a) Principle of FDM-type 3D printing
("FDM by Zureks" by Zureks - Wikimedia Commons)



(b) Examples of FDM-type printers

Fig. 1 FDM-type 3D printers

The print head of an FDM-type printer only moves to restricted directions in conventional 3D printing; however, it can actually move freely. Because an FDM-type printer conventionally prints sliced object layer by layer, the print head does not move vertically except when transition between layers. However, by using G-code directly or by using software that generates G-code, it can be moved to arbitrary direction. Although many 3D printers are not designed to move the head toward vertical direction quickly, Delta-type printers, such as Rostock MAX (see the right photo of Fig. 1(b)), are suited for this purpose.

3. 3D Turtle Graphics Based Method

Because the head of a 3D printer can move freely to any direction, it can draw lines and curves that can be drawn by Processing. It is more similar to turtle graphics because the head moves in a similar way as turtle. A 3D printer can thus embody 3D turtle graphics exactly if there are no mechanical constraints such as gravity or resilience. This section thus

describes 2D and 3D turtle graphics first, and describes the method of 3D printing based on 3D turtle graphics, which is constrained by mechanics.

3.1 2D and 3D turtle graphics

As described in Section 1, turtle graphics was introduced by Papert in 1960s, and he also proposed a programming language called Logo. By using Logo, 2D line-art can be drawn by a trajectory of a “turtle”. This is called “turtle graphics”. The basic drawing commands of turtle graphics are the following three.

- *Forward d.* This command moves the turtle forward by distance d.
- *Turn left a.* This command turns the turtle to the left by angle a° (degrees).
- *Turn right a.* This command turns the turtle to the right by angle a°. By using these commands, the turtle can be moved to any location in the 2D space, and the trajectory can be displayed as shown in **Fig. 2**.

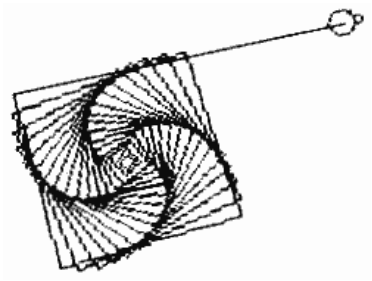


Fig. 2 2D turtle graphics

As described in the previous subsection, turtle graphics is originally two-dimensional; however, similar methods called “3D turtle graphics” were developed to draw 3D shapes (e.g., [Ver][Tip 10]). To extend turtle graphics to 3D, commands for moving up/down or for turning up/down must be added. Moreover, Bernd Paysan proposed “Dragon Graphics” [Pay 09], which is an extended 3D turtle graphics that can generate complex 3D shapes easily. However, all of them are graphics for displaying shapes by a 2D display. They cannot show the 3D trajectory of turtle.

3.2 Printing method

This subsection discusses on a method of 3D printing, which is based on 3D turtle graphics and describes a design and implementation of this method. This method is called the turtle 3D printing method.

The semantics of 3D drawing commands and G-code are similar, so the former can be translated to the latter. Because G-code is procedural, commands in G-code can draw lines in a similar way to turtle-graphics commands. However, people do not usually write G-code directly, and it is not suited for human because it is similar to assembly languages or machine languages. Fortunately, it is easy to translate 3D drawing commands to G-code; it is required only to translate forward command to a corresponding G-code command (*i.e.*, G1 command). 3D printers can thus execute commands in a similar way to turtle graphics.

However, because the coordinate of a print head is usually described by using a Descartes coordinate, they must be translated to turtle-direction-based coordinates. The technical detail is described in a previous paper [Kan 15a].

There are two alternatives for turtle coordinate: polar coordinate and cylindrical coordinate. The polar coordinate system is used for flight simulators, and the direction of turtle is decided independent of the gravity direction. Unfortunately, because it is inevitable to take gravity direction into account in 3D printing, this coordinate makes guaranteeing printability difficult. That is, when using a flight simulator (and probably when flying by an airplane), it is easy to crash the airplane because it is difficult to grasp the gravity direction. A similar situation occurs in 3D printing.

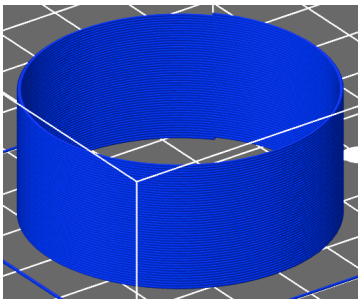
If the cylindrical coordinate system is used, the turtle always assumed to be directed horizontally. A vertical motion is described by specifying the vertical displacement, but the direction of the turtle is unchanged. Because the gravity direction is constant for the turtle, it is easier to design objects to be printed than using polar coordinate.

3.3 Examples

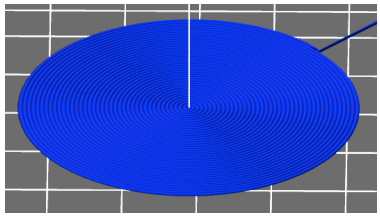
The author wrote 3D turtle graphics programs that prints shapes such as a helix, *i.e.*, thin empty cylinder, a skewed square pyramid, a 2D fractal using a program library developed by the author himself. This section first describes the program library and then describes the examples.

A program library in Python language, **`turtle.py`**, for generating G-Code for 3D printers, which is based on cylindrical coordinate system, was developed and used for describing programs to generate shapes. This library is publicly available at http://www.kanadas.com/program-e/2014/08/a_python_library_for_3d_turtle.html. This library can be used for many 3D printers, especially open-source RepRap printers [Rep]; however, unfortunately, because they are not completely compatible, the program that use the library (and maybe the library itself) has to be modified. The above programs are specialized for Rostock MAX printer.

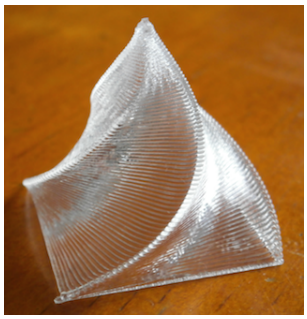
Figure 3 shows examples of shapes. Shapes such as a cylinder, disc, pyramid, and basket, shown in Fig. 3(a) to (e), were created by repeating an advance and a rotation (distance x and angle a).

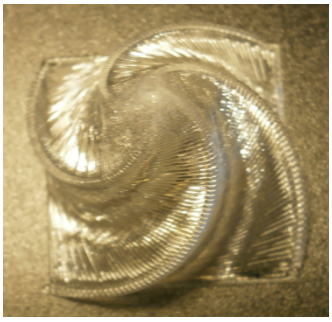


(a) Helix (Cylinder)

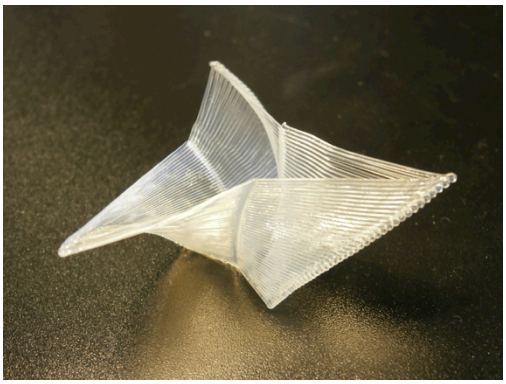


(b) Spiral (Disc)

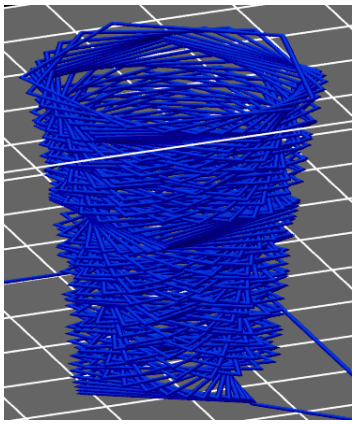




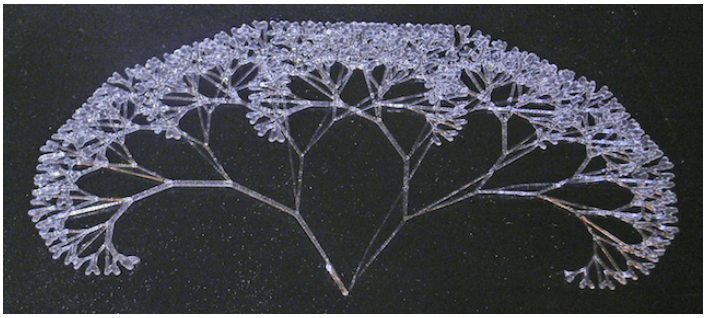
(c) Skewed square pyramid (a shrinking pattern)



(d) Example of expanding pattern



(e) A shape with less vertical overlapping of filaments



(f) 2D fractal tree

(g) 2D shape created by advances and turning expanding angles

Fig. 3 Printed results – patterns with rotation and shrinking/expanding

Figures 3(a) and (b) show models of very primitive shapes, *i.e.*, helix and disc. (These shapes are also primitives in the assembly-and-deformation method.) Figure 3(a) shows a model of an approximated helix (or cylinder). This model, which

is expressed by a Python program or a G-code (which is generated by executing the Python program), is visualized by a 3D-printing software tool called Repetier Host. The approximated helix in Fig. 3(a) consists of short straight lines drawn by the turtle (*i.e.*, print head). The direction of the lines are slightly upward, so the turtle draws a helix instead of a circle. If the amount of an advance, *d*, is decreasing or increasing, the pattern is shrinking or expanding. Figure 3(b) shows a spiral (a disc) that also consists of short lines, which is a 2D shrinking pattern. This shape also consists of short straight lines. A cone (not shown here) can be printed in the same way.

Figure 3(c) shows a skewed square pyramid, which consist of longer lines. This is a 3D shrinking pattern. The turtle moves from outer to center. This photo suggests the relationship between this shape and the 2D shape shown in Fig. 2; however, the pattern in Fig. 2 is expanding, that is, the turtle moves from center to outer. The drawing direction is reversed. The right photo in Fig. 3(c) was taken from above. A printing process was recorded as a video; it can be seen in YouTube (http://youtu.be/7H5-acxQ_RE). In contrast, an example of expanding pattern is shown in Fig. 3(d). In these patterns, filaments are stacked mostly completely, so they stay in the designed location, but never drop down.

Figure 3(e) shows a sparse pattern that the turtle does not stack filament completely, that is, there is interspace between lower and upper filaments. The left figure shows the design (drawn by Repetier Host) and the right photo shows the printed object. Originally, the vertical pitch was designed to be 0.4 mm. Limited part of the filament is supported by the filament below; however, because the pitch naturally becomes smaller in the printed result, that is, the pattern was caved in, it was redesigned to be 0.3 mm so that the macroscopic shape becomes closer to the design as shown in this figure. However, part of the filament that is not supported by the filament below still sags. If supported area is reduced, the filament sags more; however, in the case of Fig. 3(e), the print result is still close to the design. If a designed pattern is sparser, the turtle fails to shape it.

Figures 3(f) and (g) show examples of 2D patterns, which can be created in the physical space by 2D turtle graphics or drawn in a virtual space by Processing. Figure 3(d) shows a 2D fractal tree. A 2D fractal such as shown in Fig. 3(f) does not fully utilize the functions of 3D printers, *i.e.*, 3D shape generation. 3D fractal shapes are better for utilizing them; however, as far as the author knows, 3D fractal shapes require printing in the air, so they cannot be generated by turtle 3D printing; that is, no method for supporting filament in the air (without support material) is given. Figure 3(g) shows a pattern generated by multiplicatively increasing the turning angle when repeating advancing and turning.

3.4 Problems of turtle 3D printing

A supposed design and printing process for turtle 3D printing, which is described below, requires iterative design and printing process because there are many problems including dropping problem that prevent a straightforward process. A unit process consists of the following three steps: the designer (programmer) first describes the program and generating a G-code program, second verifies the G-code program by graphics, and third prints the result (sending the G-code program to a printer). However, a single iteration of this process does not usually successfully generate a correct result. The reason of failures is that there are many problems such as the following ones, which spoil the result even when the visualization of the G-code is succeeded.

Three problems are described here. An easy problem is that, because the initialization part of the G-code, **`turtle.init`**, does not initialize the printer completely, printing may fail because of incomplete initialization; however, this problem can easily be solved. Another problem is that optimum temperature may vary by the difference of filament even when the filament material is the same, e.g., it is PLA; however, the temperature can easily be optimized. The most difficult problem is caused by mechanics, especially by gravity; that is, printed filament may drop down or may cave in because the filament must be supported by previously printed filament but it fails. When it is not possible to avoid caving in, the vertical pitch may have to be changed smaller as shown in Fig. 3(e).

4. Assembly-and-Deformation Method

Because it is difficult to design objects appropriately for printing them by the 3D-turtle-graphics-based printing method as described above, the author designed one more method for designing and printing 3D objects in a generative way. This method, which is called the assembly-and-deformation method, is explained in this section.

4.1 Outline

It is easier to generate a shape, which may be complicated, by virtually deforming another (simple) shape than forming a shape by stacking a filament (or string) by turtle graphics in an ad-hoc manner. Although types of shapes that can be generated by deformation is restricted, it is easier to keep the shape printable by deformation than by turtle graphics; that is, if the original shape is printable and the deformation preserves the printability, the deformed shape is also printable. The 3D-printability concept is explained more and the preservation issues are explained in the previous paper [Kan 15b].

For example, **Fig. 4** shows examples of deformation from a virtual cup (which consists of a helix (empty cylinder) and a thin (filled) cylinder) to a variety of shapes. If a shape different from the cup is used, other types of shapes can be created by deformation. Examples of such shapes are shown in Section 4.5. The deformations are explained in the following subsections, and the technical detail is described in a previous paper [Kan 15b].

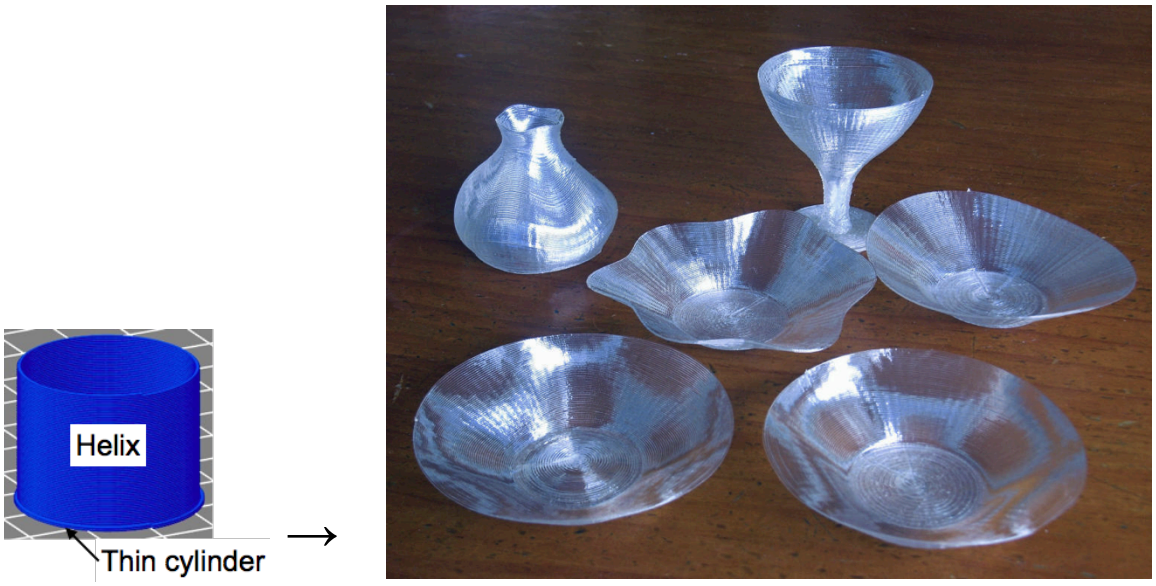


Fig. 4 Examples of shape generation by deformation

4.2 Deformation

This section briefly reviews the history of deformation in computer graphics and generative art, and describes a method of deformation for 3D printing.

4.2.1 Deformation in computer science, technology, and art

The deformation concept plays important roles in computer graphics and in generative art. In computer graphics, “free-form deformation” has been considered important for the solid modeling or surface modeling of objects with free-form surfaces [Sed 86][Coq 90], and it was used because it eases the control and rendering of 3D geometric shapes [Bar 84]. In free-form deformation, a user specifies the shape of the object using a graphical user interface (GUI).

In generative art, deformation is also an important technique and is used everywhere [Pea 11][Une 08]. However, in contrast to many other computer-graphics applications, in generative art, artistic objects are generated algorithmically, and deformations used for this purpose are also generative.

Deformation has not yet been focused on in 3D printing. Although deformation is essential in drawing graphics including generative art, and free-form deformation can be used in CAD for 3D printing, deformation is not useful in the slicing and printing steps of 3D printing. This is due to the fact that conventional 3D printing methodology does not take the attributes generated by the deformation into account because they are not (cannot be) used in slicing and printing steps. However, such attributes are important in direction-specified 3D printing.

4.2.2 Deformation method for 3D printing

This section describes a method for creating various shapes (surfaces) by using “deformation”. Basically, the shape may not have a hole on the side (unsupported part) to be printable. To preserve the printability of the object (*i.e.*, to keep the model correctly printable), deformation for 3D printing requires controlling two attributes of strings: cross section and printing velocity.

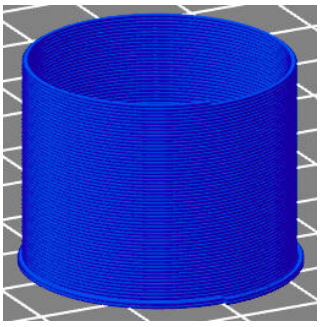
Two types of deformations are defined. One is Descartes-coordinate-based deformation and the other is cylinder-coordinate-based deformation. Both translate coordinates, cross sections, and printing speed.

A cylinder coordinate is more useful for describing a deformation, especially when axisymmetric models are deformed. The following function is defined in the library.

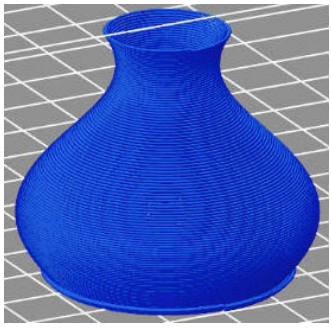
deform_cylinder($fd(r, \theta, z)$, $fc(c, r, \theta, z)$, $fv(v, r, \theta, z)$)

In this expression, function $fd(r, \theta, z)$ (*i.e.*, the first argument) maps a location (r, θ, z), which is expressed in cylinder coordinates, to a new location (r_1, θ_1, z_1). Function $fc(c, r, \theta, z)$ (*i.e.*, the second argument) maps a cross section at location (r, θ, z). Function $fv(v, r, \theta, z)$ (*i.e.*, the third argument) maps a head speed at location (r, θ, z) to a new speed. The same monotonicity conditions as deform_xyz exist for *fv* and *fc* for deform_cylinder, and functions *fd*, *fv*, and *fc* must be continuous and smooth. It would be better if the cross section could automatically be optimized; however, it is currently difficult. Therefore, the cross section must be manually specified in the method proposed in this paper.

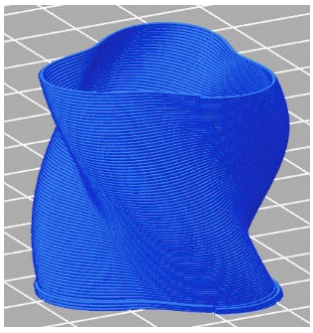
Examples of deformation are visualized in **Fig. 5**. (Repetier Host was used.) Figure 5(a) shows a cup, which consists of an empty cylinder and a disc (bottom); that is, the cup is an assembly of two direction-specified components. This cup can be transformed to a plate shown in Fig. 4. The shapes in Fig. 5(b) and (c) can be created in similar ways. Figure 5(d) shows an empty cylinder (without a bottom). This cylinder can be transformed to a sphere shown in Fig. 9(e), which preserves the filament pitch.



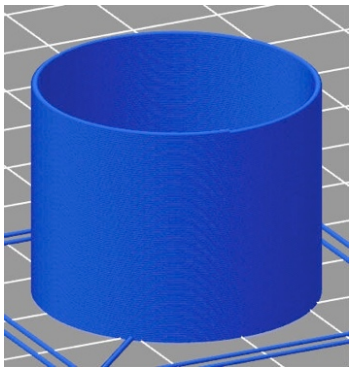
(a) Cup – original shape



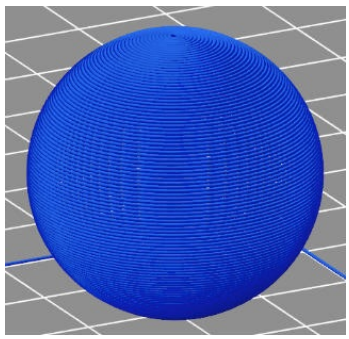
(b) Vase – symmetric deformation result



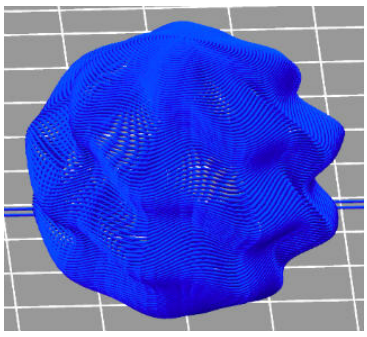
(c) Non-axisymmetric deformation results



(d) Empty cylinder – original shape



(e) Sphere – symmetric deformation result



(f) Non-axisymmetric deformation results

Fig. 5 Examples of deformation

Moreover, non-axisymmetric shapes such as those shown in Fig. 5(c) or (f), which can be generated by deforming axisymmetric shapes shown in Fig. 5(a) and (d), can be easily generated using cylinder-coordinate based deformations.

Although various shapes exist in Fig. 5, all these shapes are generated only using these trigonometric functions. However, other types of functions can of course be used.

4.3 Texture-mapping technique

This section summarizes a texture-mapping method (or modulation method) used with the proposed printing method, which is described more in a previous paper [Kan 15c] (and also in another paper [Kan 15b]). Pictures, characters, or textures can be mapped to the surface of an object printed by using this method (see Fig. 8(g) and (h)).

To map textures to the surface of an object to be printed, the cross section of the filament on the surface can be controlled to express textures. This method can generate fine structures, although it is not suited for generating large and deep structures. It can be called the extrusion modulation method; that is, the process of filament extrusion is modulated by pictures, characters, or textures.

Two methods are available for extrusion modulation. The first method is to vary the extrusion speed of filament and the second method is to vary the print-head motion speed. The second method was selected because it is better in response time. In 3D printers, the delay between the motion of the extruder that extrude the filament and the motion of filament at the nozzle, *i.e.*, the tip of the print head is large. It may take several seconds. The response is, thus, slow. Print heads of 3D printers are usually heavy so they have large inertia; however, the response of the print heads are still much better than the filament response.

A method for modulating a surface of a direction-specified model of a 3D-printed object by varying the head motion (and/or the extrusion speed) is described below. The original model is assumed to consist of short lines, which is called “strings”. This method converts the original strings S*i* to new strings S*i*’ ($1 \le i \le N$). The original model is modulated by using a bitmap and the modulated model is generated.

Figure 6 outlines the modulation process. Each string has the cross section, c*i*, as its property and the head speed, v*i*, is also specified. These values are updated by the modulation. If the corresponding bit is zero, the value of c*i* becomes c0 and the value of v*i* becomes v0. If the corresponding bit is one, the value of c*i* becomes c1 and the value of v*i* becomes v1. If v0 = v1, the cross section is controlled only by extrusion speed. This is *not* the selected method. If c1/c0 = v1/v0, the cross section is controlled only by print-head motion speed. This method is selected because the extrusion-speed control mechanism does not respond quickly; that means, it has large latency.

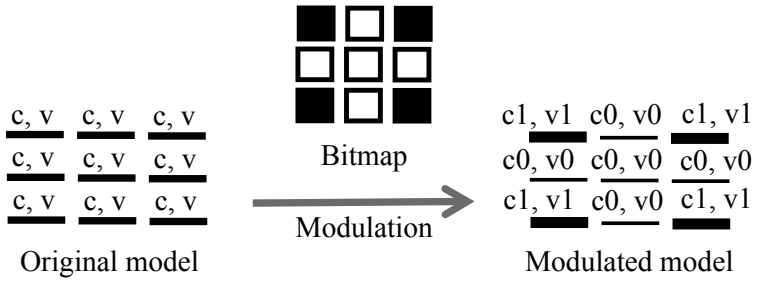


Fig. 6 Modulation by a bitmap

4.4 Reflection of light

Several types of filament used for FDM printers reflects light on the surface and brilliantly shines, and the amount and the direction of reflection can be controlled by certain techniques. This is an attractive attribute for artistic or visual-design purposes. For example, transparent PLA, especially pure PLA, is quite attractive in reflection (see Fig. 8, especially (a) and (b)); that is, reflection becomes strong in 3D-printed objects made of PLA because strings increase the surface area that reflects light. This attribute is caused not only by the surface, but it is also affected by the internal structure of an object. If the filament is not transparent, the brilliance disappears.

The reflection can be controlled by the angle of stacking filaments, filament density, or some other attribute of the printing method (**Fig. 7**). Figure 7(a) shows reflection controlled by the overhang angle. If the direction of the light source is varied,

different portions of the object more strongly reflect the light. Figure 7(b) shows reflection controlled by the filament density. Even if the direction of the light source is varied, the diffusing part never reflects light brilliantly.

(a) Reflection controlled by the overhang angle (b) Reflection controlled by the filament density

Fig. 7 Reflection control

4.5 Examples

The author designed and printed various examples. A program library, **`draw.py`**, for generating G-Code for 3D printers was developed and used. This library is publicly available at http://www.kanadas.com/program-e/2014/10/-3d_printing_library_for_parts.html. By using this library, an object designer can write a program to select parts, such as a line, a helix, a cylinder, to specify parameters of the parts, and to combine them. In the current version, deformations can only be applied to the combined model, but it will be applied to parts or partially combined model too. Similar to **`turtle.py`**, the above library is specialized for Rostock MAX; however, the same or slightly-modified program can be used for many other 3D printers. The technical detail is available in a previous paper [Kan 15b].

Pieces designed by the proposed method and printed by Rostock Max 3D-printer are shown in **Fig. 8**(a) to (h). Various plates and vases can be created by deforming a cup, as shown in Fig. 8(a) to (d); that is, a combination of an empty cylinder and a helix. Although these objects still require non-straight-forward design and printing process that includes fine adjustments of filament cross sections and printing speeds, this process is much easier for the designer than the process required for turtle 3D printing. All the samples in this section are available from a Web site (http://store.shopping.yahoo.co.jp/dasyn/).

By using the proposed printing method (which is called the helical/spiral printing method [Kan 15b]), shallow plates such as shown in Fig. 4, 8(a), or 8(b) can be created without support material, which is required for conventional 3D printing method. Because no support sticks to the surface of the plate, surface filaments can be kept smooth (to be mirror plane) and the surface can be brilliant.

The deformation function used for the plate shown in Fig. 8(a) contains cos(4θ), which generates the 4-cycle patterns, and the light-reflection control technique[1]. The brighter (reflecting) areas move while changing the light-source direction. Figure 8(b) shows a heart-shaped plate[2]. The deformation function for this plate is based on the following function that maps a circle to a heart shape.

$$fdh(x, y, z) = (x + bz\ \mathrm{sqrt}(|y| / radius), y, z)$$

This function is based on a “equations for heart-shaped curve” [Yam 07]. The appropriate range of parameter b is from 0 to 1.2. This function becomes identity function when $bz = 0$ (at the bottom (center) of the plate), and it generates sharper heart shape when bz becomes larger (at the top (perimeter)).

Figure 8(c) and (d) show taller objects. Figure 8(c) shows a vertically “swinging” vase; that is, the print head slightly moves up and down (so the filament waves vertically) when printing. Such a motion never occurs in conventional 3D printing methods. It was printed with three-cycle vertical motion. They are generated from a cup as well. Figure 8(d) shows a “wine glass”. Vases and wine glasses sometimes leak water; however, by controlling the filament cross-section and the printing speed properly, most of them do not leak water.

---

[1] A video on the printing process (http://youtu.be/5P1vaahzW98) and samples (http://store.shopping.yahoo.co.jp/dasyn/1011-04.html) are available.

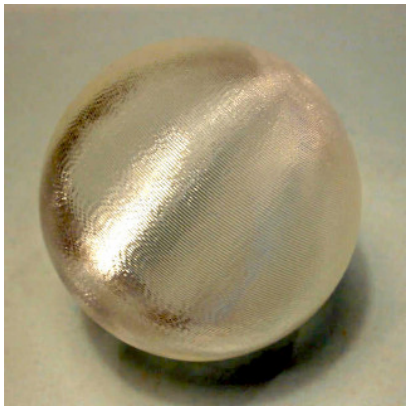
[2] A video and samples: http://youtu.be/G9x14DZYN_8, http://store.shopping.yahoo.co.jp/dasyn/3db0f5bafe.html

(a) 4-cycle plate

(b) Heart-shaped plate

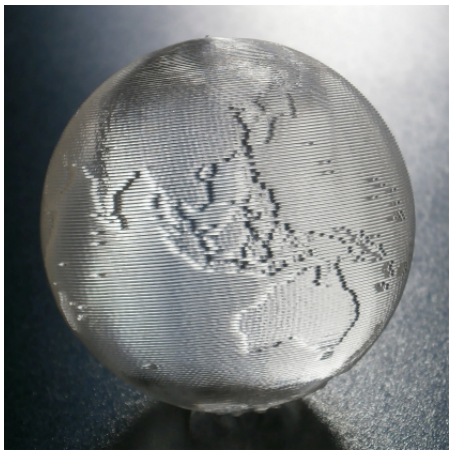
(c) Vertically-“swinging” vase

(d) Wine glass



(e) Sphere

(f) LED shade



(g) Globe

(h) Calendar

Fig. 8 Printed objects generated by various deformations

Most of the shapes of the plates and pods are approximated by 72 linear lines per round trip of the head. The turning points of the head and filament can be observed in these photos, especially in Fig. 8(a) to (c). The plates will look better if they consist of finer lines, but it will take more time to print them.

Shapes in Fig. 8(e) to (h) are created by deforming a helix (drawn in Fig. 5(d)). An example of a printed sphere is shown in Fig. 8(e)[3]. A sphere can be clean, brilliant, and strong; however, it is necessary to determine the cross section of filaments and the printing speed (*i.e.*, the head motion speed) very carefully to obtain these properties. Figure 8(f) shows a shade for an LED lamp. This is the largest object in Fig. 8; however the diameter is approximately 80 mm, and the printing time is less than 20 minutes.

[3] http://youtu.be/xr6zg0Z07HA, http://store.shopping.yahoo.co.jp/dasyn/1032-10.html

The texture-mapping method is used in the object shown in Fig. 8(g) and (h). Figure 8(g) shows a globe, *i.e.*, a sphere that a world map of 300x150 pixels is mapped to; this means the sphere consists of 150 approximate circles and each circle consists of 300 strings[4]. The map by equidistant cylindrical projection method derived from NASA was taken from a web site called "Celsia Motherload" (http://www.celestiamotherlode.net/catalog/earth.php). The diameter is 50 mm and the contrast of the land and the sea is 1.3 to 1.4. If the contrast is too strong, the printing may fail. Figure 8(h) shows part of a calendar. Each cylinder contains days in two months, so a set of calendar consists of six cylinders.

5. Conclusion

This presentation proposed two methods for designing and printing generative 3D objects, *i.e.*, a 3D turtle-graphics-based method and the assembly-and-deformation method. The former is more intuitive but easy to fail, and the latter is easier to generate printable design. Although it is yet not very easy to create an object exactly as designed by a straightforward process, various shapes can be generated by applying these methods iteratively. In the second method, the designer can also map textures, characters, or pictures on the surface of the object. If the initial model is a thin helix with a very low cylinder (*i.e.*, an empty cylinder with a bottom), shapes like cups, dishes, or pods with attractive brilliance can be generated, and a globe and other shapes can be generated from a helix. Python APIs for these methods have been publicly available; however, easier ways to use these methods will be developed.

References

[Bar 84] Barr, A. H., "Global and local deformations of solid primitives", *ACM SIGGRAPH Computer Graphics*, Vol. 18, Vol. 3, pp. 21–30, 1984.

[Coq 90] Coquillart, S., "Extended free-form deformation: a sculpturing tool for 3D geometric modeling", *ACM SIGGRAPH Computer Graphics*, Vol. 24, No. 4, pp. 187–196, 1990.

[Kan 15a] Kanada, Y., ""3D Turtle Graphics" by using a 3D Printer", *International Journal of Engineering Research and Applications*, Vol. 5, No. 4, Part 5, pp.70-77, April 2015.

[Kan 15b] Kanada, Y., "*Support-less Horizontal Filament-stacking by Layer-less FDM"*, *International Solid Free-form Fabrication Symposium 2015*, 2015-8, August 2015.

[Kan 15c] Kanada, Y., "Creating Thin Objects with Bit-mapped Pictures/Characters by FDM Helical 3D Printing", *8th International Conference on Leading Edge Manufacturing in 21st Century (LEM 21)*, October 2015.

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

[Mat] Mataerial – A Radically New 3D Printing Method, http://www.mataerial.com/.

[Pap 80] Papert, S., "Mindstorms: Children, Computers, and Powerful Ideas", Basic Books, 1980.

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

[Pea 11] Pearson, M., "Generative Art: A Practical Guide Using Processing", Manning Publishing Co., 2011.

[Rep] RepRap Wiki, http://reprap.org/

[Sed 86] Sederberg, T. W. and Parry, S. R., "Free-form deformation of solid geometric models", *ACM SIGGRAPH Computer Graphics*, Vol. 20, No. 4, pp. 151–160, 1986.

[Tip10] Tipping, S., "Cheloniidae", December 2010.

[Une 08] Unemi, T., Matsui, Y., and Bisig, D., "Identity SA 1.6: An Artistic Software that Produces a Deformed Audiovisual Reflection Based on a Visually Interactive Swarm", *2008 International Conference on Advances in Computer Entertainment Technology* (*ACE '08*), pp. 297–300, 2008.

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

[Wik] WikiBooks, "OpenSCAD User Manual", available at http://en.wikibooks.org/wiki/OpenSCAD_User_Manual

[Yam 07] Yamamoto, N., "Heart-shaped Curves", http://www.geocities.jp/nyjp07/heart/index_heart.html, 2007 (in Japanese).

[4] http://store.shopping.yahoo.co.jp/dasyn/1201-03.html, http://youtu.be/YWx1vqig2-o