An Interactive Design System for Sphericon-based Geometric Toys using Conical Voxels

Masaki Hirose[1], Jun Mitani[1,2], Yoshihiro Kanamori[1], and Yukio Fukui[1]

[1] Graduate School of Systems and Information Engineering, University of Tsukuba, 1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573, Japan
[2] JST/ERATO,Frontier Koishikawa Bldg., 7F 1-28-1, Koishikawa, Bunkyo-ku, Tokyo 112-0002, Japan
{hirose,mitani,kanamori,fukui}@npal.cs.tsukuba.ac.jp

Abstract. In this paper, we focus on a unique solid, named a "sphericon", which has geometric properties that cause it to roll down a slope while swinging from side to side. We propose an interactive system for designing 3D objects with the same geometric characteristics as a sphericon. For high system efficiency, we used a conical voxel representation for defining these objects. The system allows the user to concentrate on the design itself while the system ensuring that the geometrical constraints of a sphericon are satisfied. The user can also preview the rolling motion of the object. To evaluate the effectiveness of the proposed system, we fabricated the designed models using a 3D printer, and confirmed that they rolled as smoothly as a standard sphericon.

Keywords: user interface, geometric modeling, sphericon, conical voxel

1 Introduction

Of the many kinds of toys available in the world, one group is sometimes referred to as "geometric toys", since they have geometrically interesting shapes or movements [1]. These toys are considered to be useful to evoke children's interest in geometry and physics, and they can be found both in shops and science museums. In this paper, we focus on a unique solid, named a "sphericon", which has interesting geometric characteristics such that it rolls down a slope swinging from side to side. It is difficult to predict how it will roll at first glance and it is entertaining to watch it move. A toy based on the sphericon has been available for some time in the U.S. [2]. The shape of the sphericon is generated by the following steps (see Fig.1). First, a square is rotated by 180 degrees around its diagonal. As a result, two cones with apex angles of 90 degrees are obtained and are connected at their base circles. One half of this 3D shape is then rotated by 90 degrees about an axis perpendicular to the original rotation axis.

A sphericon has the geometrical property that its center of gravity remains a fixed height above the surface over which it rolls, similar to the case for a

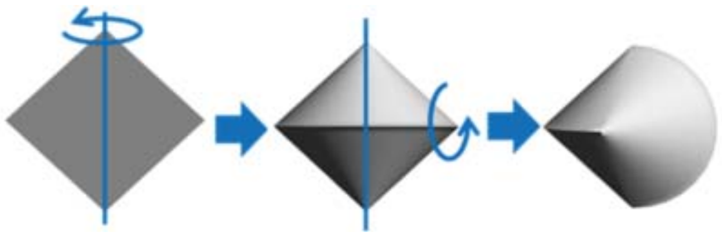


Fig. 1. Generation of a sphericon

sphere or a horizontally placed cylinder. When a sphere rolls, only a single point on its surface touches the floor. On the other hand, when a sphericon rolls, a generating line touches the floor, as is the case for a cylinder. This is because the surface of a sphericon consists of two conical sectors. This means that a sphericon will roll smoothly even if parts of its surface are sculpted away, so long as at least two points remain on a given generating line. Based on this fact, we propose an interactive system for designing objects that roll in the same way as a sphericon. The design process starts with a true sphericon, which is then sculpted to obtain new objects.The designed object should satisfy the following three geometric constraints.

Center of gravity constraint: The center of gravity of the object should remain at a fixed height above the floor on which it rolls.
Tangency constraint: The shape should touch the floor along a generating line. Although some parts of the line may be removed, at least two points on any generating line must be retained.
Topology constraint: The object should comprise a single part so that it will not break apart.

In this paper, we propose a conical voxel representation for defining the object, which is well suited to represent sphericon-like objects. We implemented two different user interfaces so that the user can interactively edit an object while satisfying the above three constraints. The first allows the user to directly add and remove cells. In the second, the user paints on a plane map obtained by flattening the surface of a sphericon. In both methods, The system allows the user to concentrate on the design itself while the system ensuring that the geometrical constraints of a sphericon are satisfied. The user can also preview the rolling motion of the object. Finally, the designed object can be physically constructed using a 3D printer and its rolling behavior can be enjoyed.

The main innovations of the present research are the use of conical voxels to represent the designed object and the development of the user interfaces described above.

In this paper, we introduce related work in Section 2 and describe the details of our method in Section 3. In Section 4, we describe the results, and finally present our conclusions and ideas for future work in Section 5.

2 Related Work

2.1 Designing a geometric shape for real-world objects

A recent trend in geometric modeling in the field of computer graphics has been the modeling of not only freeform virtual shapes but also the shapes of real-world objects. For instance, Mitani and Suzuki [3] and Shatz et al. [4] proposed methods for modifying 3D geometries to produce papercraft toys. Li et al. [5] proposed a method for generating geometries for pop-up card geometries from existing 3D data, in which a 3D structure pops up when the card is opened 90 degrees. In these studies, real-world objects are constructed from printed patterns, and original shapes are modified so that they satisfy geometrical constraints such as "the shape must comprise multiple pieces which can be flattened without distortion" or "the shape must pop up without collisions occurring". Using a computer to ensure that the target shape satisfies the required geometrical constraints is a useful approach, especially when the constraints are severe or the user is a novice.

When designing 3D real-world objects, we sometimes have to consider not only geometrical problems but also the physical characteristics of the object itself. Mori and Igarashi [6] proposed a system for designing stuffed toys made with stretchy fabric. The system simulates material stretching based on the pressure generated from the inner cotton. The user can interactively design new stuffed toys while viewing the results of the simulation. Furuta et al. [7] proposed a system for designing kinetic art, which takes account of the motion of objects. The behavior of rigid objects in the system is simulated, and the predicted scene is displayed so that the user can interactively design kinetic artwork while observing the results of the motion. These studies have shown that combining an object design system with motion simulation represents a powerful tool for designing real-world objects.

In this paper, we also propose interfaces with which the user can easily design objects that satisfy the geometrical constraints required for rolling on a floor like a sphericon. Further, we employ a motion simulation for previewing the rolling behavior before actually constructing the objects.

2.2 Interactive volume data editing

Surface models are commonly used in computer graphics to represents 3D models. However, a volumetric data representation, involving both density and weight, is sometimes needed to model solid real-world objects. One typical volumetric representation method is based on voxels, which are small cubes that build up the 3D shape of the object. Recently, in the field of industrial modeling, a voxel based topology optimization approach is often used to determine the shape of machine components [8]. In this approach, shape optimization is carried out by minimizing the volume and mass of the component while maintaining the necessary strength. However, since its purpose is not to make a shape reflecting the intent of the designer, this approach is not appropriate for an interactive editing system. Galyean et al. [9] proposed a voxel-based interactive shape editing system that combined editing tools with a 3D input device. Perry et al. [10] proposed a digital clay system in which the user can design a shape in a similar way to clay modeling.

In a study of the generation of geometrically interesting shapes based on a voxel representation, Mitra et al. [11] proposed a system which automatically generates a cubic object which projects three distinct shadow images, which are specified by the user, on plain surfaces when the solid is lit from three different directions. Further, the user can edit the shape of the object while keeping the shadow images not to be changed. This system is based on a similar concept to ours, in that while editing operations are being carried out by the user, the computer ensures that the required geometric constraints are being satisfied. However, the constraints applied are different. While Mitra et al. required that the object projected target images as shadows, our requirement is that the object rolls on a floor as smoothly as a sphericon. In addition, the models generated by our system must be single components, which was not a requirement of Mitra et al. Further, while Mitra et al. employed a standard voxel representation, we propose a new conical voxel representation.

2.3 Large objects based on a sphericon

Muramatsu constructed large human-sized objects from stainless steel pipes based on the geometrical characteristics of a sphericon [12]. The height of the center of gravity of each object was constant, and they rolled smoothly. The objects were manually designed based on the geometrical constraints. In the system proposed in the present study, such objects could be designed without the need for the user to consider specific geometrical constraints. In addition, Muramatsu could not see how the designed objects would roll in advance, whereas in our system this is made possible by integrating a motion simulation.

3 Proposed method

3.1 Conical voxel representation

A sphericon is composed of four half-cones whose apex angle is 90 degrees, as shown in Fig.2 (center). We henceforth refer to these half-cones as "units". To represent the position of any point in a unit, we introduce a coordinate system based on θ, r and h, as shown in Fig.2 (left). The origin O of the coordinate system is the center of the base circle of the original cone. O also represents the center of gravity of the sphericon. A point P in a sphericon is then represented by four parameters (i, θ, r, h), where i is an integer with a value from 0 to 3 that specifies the unit in which P is located. θ is the angle around the axis of the cone between the base generating line L and the generating line on which P lies, and takes values from 0 to π. r is the distance between P and the cone

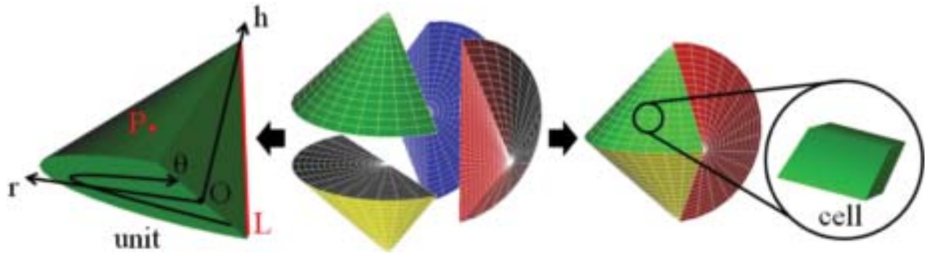


Fig. 2. Structure of a sphericon in a conical voxel representation

axis, divided by the radius of the base circle of the cone, and takes values from 0 to 1. h is the height of P above the base circle, divided by the height of the cone, and also takes values from 0 to 1.

In the proposed system, each unit is subdivided based on the coordinates θ, r and h into smaller regions referred to as "cells", as shown in Fig.2 (right). If the numbers of subdivisions along the coordinates θ, r, and h are denoted as N_θ, N_r, and N_h, respectively, the total number of cells in a sphericon is then given by $4 \times N_\theta \times N_r \times N_h$.

The position of each cell is then described by a combination of four index values (i, I_θ, I_r, I_h), where $i = 0, 1, 2, 3$, $0 \leq I_\theta \leq N_\theta - 1$, $0 \leq I_r \leq N_r - 1$, and $0 \leq I_h \leq N_h - 1$. In addition, each cell is assigned a binary value of 0 or 1. An object is defined as a set of cells which value is 1.

In this paper, we refer to this representation as a conical voxel representation. Unlike the general cubic voxel representation, it has the advantage of being able to smoothly represent the surface of a sphericon.

3.2 Geometrical constraints

The designed object must satisfy the three geometrical conditions described in Section 1. We will now show how this can be achieved using the conical voxel representation.

Center of gravity constraint When the center of gravity of the edited object is located at the same position as that of the original sphericon (i.e., $\theta = r = h = 0$), the object rolls smoothly without vibration of the center of gravity. This condition is always satisfied when all four units have the same shape because the units are located symmetrically around the center of the sphericon. Thus, in our system, the condition is imposed that all units must have the same shape. Although it is possible to satisfy the center of gravity constraint even for units with different shapes, we introduced this additional constraint in order to make the interface simpler. When a unit is edited, its shape is duplicated in the other units, leading to overall symmetry.

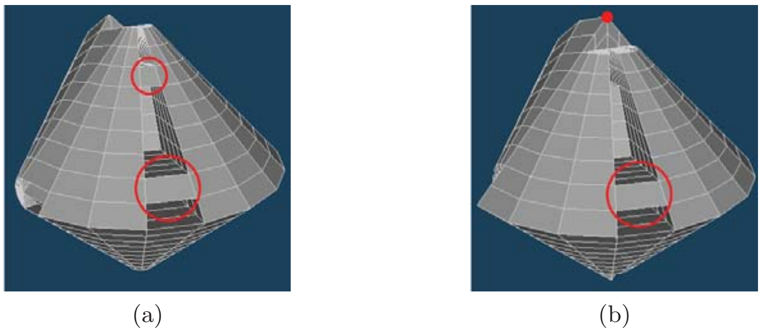


Fig. 3. (a) Two cells, surrounded by red circles, exist on a ruling line. (b) A single cell, surrounded by a red circle, exists on a ruling line, and one exists at the apex of the unit.

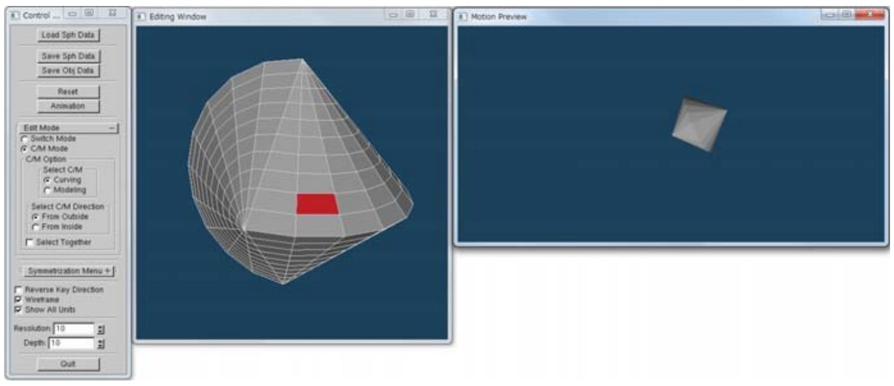
Tangency constraint Here we define the ruling line, $RulingLine(R_i, R_\theta)$, as a set of cells $Cell(i, I_\theta, I_r, I_h)$ such that $i = R_i, I_\theta = R_\theta, I_r = N_r - 1$, and $0 \leq I_h \leq N_h - 1$. When a sphericon is represented as a set of cells, it touches the floor along one of its ruling lines. Even if some cells in the ruling line are removed, the sphericon still rolls smoothly if one of the following constraints is satisfied.
(1) At least two cells exist on a ruling line (Fig.3 (a)).
(2) At least one cell exists on a ruling line and one exists at the apex of the unit (Fig.3 (b)).

Topology constraint The object should comprise a single part. This constraint is simply satisfied when all cells whose value is 1 are traversed without passing cells whose value is 0.

3.3 Implemented System

We implemented a system which has two types of user interfaces with which the user can interactively design an object which satisfies the geometrical constraints described in previous subsection. The system also has the ability to display a 3D animation of the designed solid rolling on a floor. Details of the system are described in the following.

Direct editing One of interfaces we propose is a direct-editing interface with which the user can directly add or remove cells. Adding a cell means setting its value to 1, whereas removing a cell means setting it to 0. The user edits a model starting with a sphericon which is initially prepared by the system. Based on the geometrical constraints, editable cells are limited by the system. When a cell appears red, it can be removed. When it is green, the user can add a cell to that position. Finally, when it is black, the user cannot add or remove that cell. The user repeats the operations of adding and removing cells under these limitations.



Switch Mode [✓ C/M Mode]
[✓ Curving] Modeling
[✓ From Outside] From Inside
Select Together
Reverse Key Direction
[✓ Wireframe]
[✓ Show All Units]
Resolution: 10
Depth: 10

Fig. 4. Direct-editing system, consisting of a control panel (left), a window for editing (center) and a window for previewing the motion of the object (right).

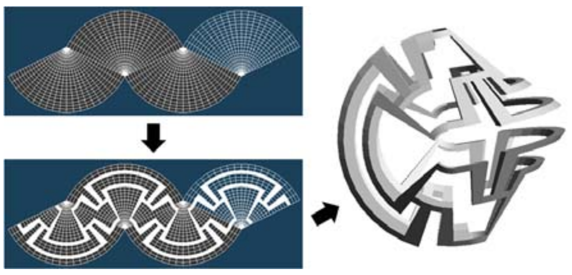
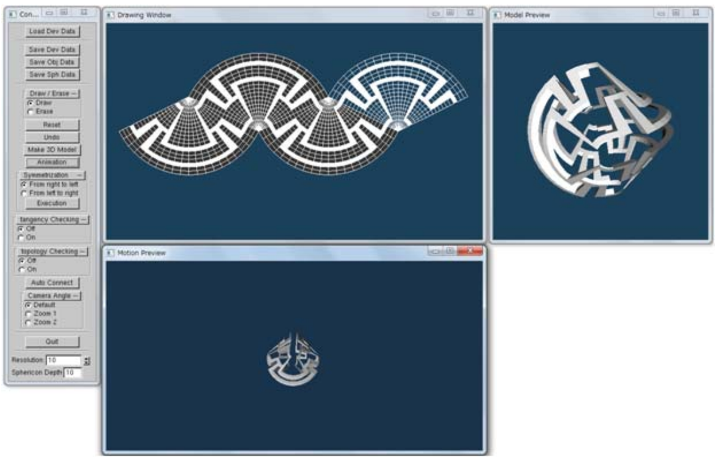


Fig. 5. The user paints a pattern on the flattened surface of a sphericon (left). The pattern is then applied to the external cells of the sphericon (right).

Fig.4 shows a screenshot of the direct-editing interface. The resolution of the conical voxels is adjustable through the control panel. The user selects a single cell or multiple cells using a keyboard. Although the topology constraint is satisfied by the system for each unit, the connectivity between different units must be confirmed by the user. Because it is not easy to evaluate whether two cells are actually in contact with each other when they are located in different units (e.g., the red and green units in Fig.2).

Pattern painting on flattened surface of a sphericon Because the surface of a sphericon is composed of four conical subsurfaces, it is flattened into a plane, as shown in Fig.5. This flattened surface corresponds to the locus produced when the sphericon rolls. In the second interface, a pattern is painted on this flattened surface, and is then applied to the external cells of the sphericon. The cells shown in Fig.5 correspond to external cells $Cell(i, I_\theta, I_r, I_h)$ where $I_r = N_r - 1$. The cells with $I_r < N_r - 1$ in the sphericon are initially removed.

The pattern is painted using a mouse on a flattened sphericon surface prepared by the system. Based on the center of gravity constraint described earlier,



Draw / Erase: [✓ Draw] Erase
Symmetrization: [✓ From right to left] From left to right
Tangency Checking: [✓ Off] On
Topology Checking: [✓ Off] On
Camera Angle: [✓ Default] Zoom 1 Zoom 2
Resolution: 10
Sphericon Depth: 10

Fig. 6. Pattern-painting system, which consists of a control panel (left), a window for painting (center top), a window for previewing the resulting object (right), and a window for previewing the motion of the object (center bottom).

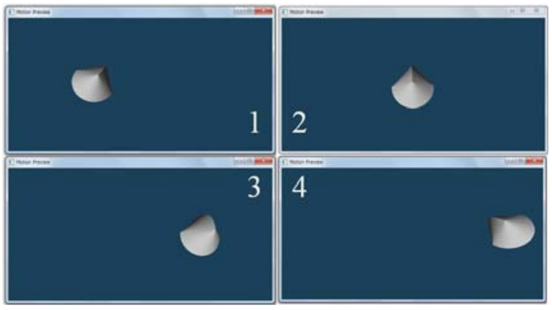


Fig. 7. Animation of a rolling sphericon.

the editable area is limited to one of the four units, and the pattern is automatically duplicated on the remaining units. When the user presses the "Tangency Checking On" button, the system notifies the user if the tangency constraint is not satisfied. The cells which need to be painted in order to make the designed object satisfy the topology constraint are automatically painted by the system. The user interface is shown in Fig.6.

Generating animation of a rolling sphericon Since the aim of designing sphericon-like objects is to enjoy watching their rolling motion, we included a motion simulation to our system. The motion data is generated by simulating the physical behavior of the original sphericon using the PhysX physics engine [13] and then applying this to the designed object.

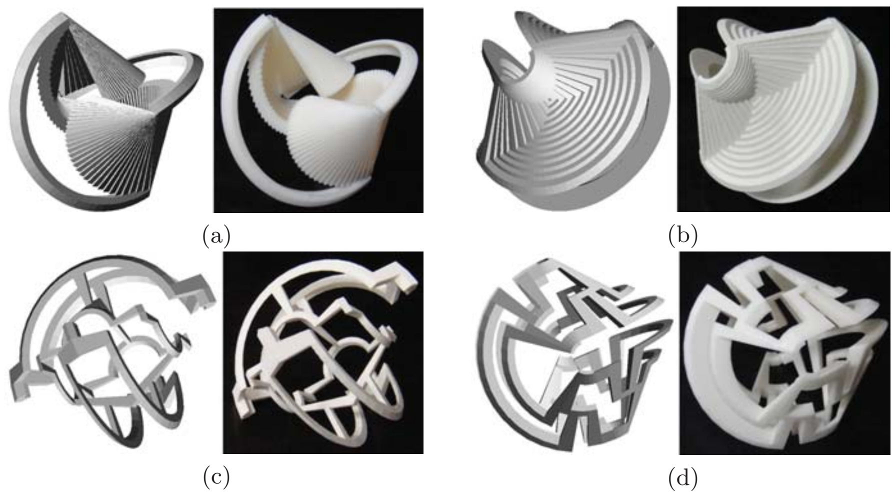


Fig. 8. CG images of 3D models and photos of real objects printed by a 3D printer.(a,b,c) Designed using the direct-editing interface. (d) Designed using the pattern-painting interface.

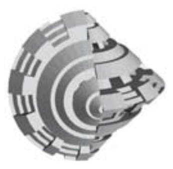
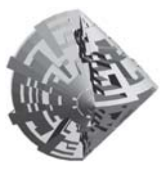
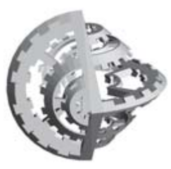
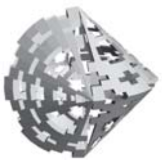
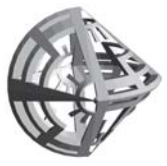
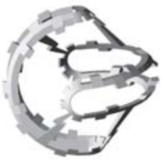
4 Results

4.1 Obtained 3D objects

We implemented the system proposed in this paper using c++ on a PC (CPU: Intel Core i7 2.80 GHz × 4, Memory: 4 GB, GPU: NVIDIA Quadro FX 580), and attempted to design new geometric toys which behaved like sphericons. We then fabricated the models using a 3D printer by exporting the geometries to OBJ formatted files to evaluate the results. Fig.8 shows four examples of designed 3D models and the final printed objects. All were confirmed to roll as well as a sphericon. The voxel resolution of each unit used during the design phase of the objects shown in Fig.8 are (a) $41 \times 41 \times 42$, (b) $41 \times 41 \times 42$, (c) $21 \times 21 \times 21$, (d) $20 \times 20 \times 10$.

User study We carried out a study to evaluate the response of users to the proposed system. The subjects were three university students in the Department of Computer Science. Table 1 shows the 3D models designed by the subjects and the time required to design each. We received positive comments such as "It was fun to interactively design 3D models. The system would be suitable for children", and "I could design a geometrically interesting shape easily, because the system assisted me". On the other hand, we received the following negative comments. "It is difficult to paint the intended patterns on the flatten pattern of a sphericon since the shape of cells is not a square".

Table 1. 3D models designed by the subjects and the time required to design each.

Interface	subject 1	subject 2	subject 3
Direct editing The resolution of each unit is $20 \times 20 \times 20$	 46 min	 13 min	 46 min
Painting on a development The resolution of each unit is $20 \times 20 \times 10$	 12 min	 27 min	 23 min

5 Conclusion and future work

We proposed an interactive system for designing objects which have the same geometrical characteristics as a sphericon. For high system efficiency, we used a conical voxel representation for defining these objects. We developed two different user interfaces to allow the user to interactively edit the object. We also implemented a preview feature to confirm the motion of the object by using a motion simulation. To evaluate our system, we printed out designed models using a 3D printer, and confirmed that they could roll as smoothly as a sphericon, confirming that sphericon-like objects can be successfully designed.

However, there are also limitations to the proposed system. A major limitation is that designed objects are limited to symmetrical shapes in order to obey the center of gravity constraint. It is possible to design dissymmetrical shapes by adjusting the center of gravity by adding additional cells after the object is designed. To evaluate this approach, we temporarily implemented this method in our system. Some cells are automatically added so that the center of gravity of the object moves to the appropriate position. Although the center of gravity constraint is fulfilled by adding some cells, as shown in Fig.9, the results are so far not very satisfactory. A strategy is necessary to determine which cells to add to adjust the center of gravity without causing large changes in the appearance of the object.

Even if a designed object satisfies the tangency constraint and has two external cells on a generating line, it will not balance if those cells are located too close to each other. Precise physical simulations are required to identify such problems before producing real objects.

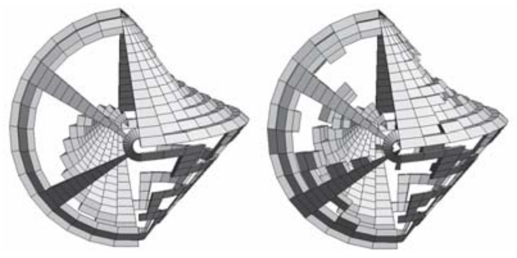


Fig. 9. Object whose center of gravity constraint is not satisfied (left). Result of adding cells to adjust the center of gravity (right).

One of drawbacks of using a conical voxel representation is that cells near the apex of a cone tend to become too small to edit. On the other hand, cells near the bottom are sometimes too large to apply to detailed shapes. In addition, it is difficult to represent smooth freeform surfaces.

In future, for the practical implementation of such a system in designing children's toys, it will be necessary to consider strength and safety. In addition, the addition of coloring methods to the simulation would be useful for designing colorful toys.

Although some future work still remains, we believe that the results shown in this paper demonstrate the efficiency of an interactive design system in which a computer ensures that certain geometrical constraints are satisfied.

References

1. Sakane, I.: The history of play. The Asahi Shimbun Company, Japan (in Japanese) (1977)
2. Toys From Times Past, `http://www.toysfromtimespast.com/toys/sphericonpins2.htm`
3. Mitani, J., Suzuki, H.: Making Papercraft Toys from Meshes using Strip-based Approximate Unfolding. ACM Transactions on Graphics 23(3), 259–263 (2004)
4. Shatz, I., Tal, A., Leifman, G.: Paper Craft Models from Meshes. The Visual Computer: International Journal of Computer Graphics 22(9), 825–834 (2006)
5. Li, X.Y., Shen, C.H., Huang, S.S., Ju, T., Hu, S.M.: Popup: Automatic Paper Architectures from 3D Models. ACM Transactions on Graphics 29(4), article No. 111 (2010)
6. Mori, Y., Igarashi, T.: Plushie: An Interactive Design System for Plush Toys. ACM Transactions on Graphics 26(3), Article No. 45 (2007)
7. Furuta, Y., Mitani, J., Igarashi, T., Fukui, Y.: Kinetic Art Design System Comprising Rigid Body Simulation. Computer-Aided Design and Applications, CAD in the Arts Special Issues 7(4), 533–546 (2010)
8. Bendsoe, M.P., Sigmund, O.: Topology Optimization. Springer (2003)
9. Galyean, T.A. Hughes, J.F.: Sculpting: an interactive volumetric modeling technique. SIGGRAPH Computer Graphics 25(4), 267–274 (1991)
10. Perry, R.N., Frisken, S.F.: Kizamu: a system for sculpting digital characters. Proceedings of the 28th annual conference on Computer graphics and interactive techniques, 47–56 (2001)

11. Mitra, N.J., Pauly, M.: Shadow Art. ACM Transactions on Graphics 28(5), article No. 156 (2009)
12. Muramatsu, T.: Solid geometric objects that maintain a constant height when moved. Japan Society for Graphics Science, Journal of Graphic Science of Japan 40(4), 11–16 (in Japanese) (2006)
13. PHYSX, `http://www.nvidia.com/object/physx_new.html`