Vis Comput (2011) 27: 605–612
DOI 10.1007/s00371-011-0564-0

ORIGINAL ARTICLE

An interactive design system for pop-up cards with a physical simulation

Satoshi Iizuka · Yuki Endo · Jun Mitani · Yoshihiro Kanamori · Yukio Fukui

Published online: 20 April 2011

Abstract We present an interactive system that allows users to design original pop-up cards. A pop-up card is an interesting form of papercraft consisting of folded paper that forms a three-dimensional structure when opened. However, it is very difficult for the average person to design pop-up cards from scratch because it is necessary to understand the mechanism and determine the positions of objects so that pop-up parts do not collide with each other or protrude from the card. In the proposed system, the user interactively sets and edits primitives that are predefined in the system. The system simulates folding and opening of the pop-up card using a mass–spring model that can simply simulate the physical movement of the card. This simulation detects collisions and protrusions and illustrates the movement of the pop-up card. The results of the present study reveal that the user can design a wide range of pop-up cards using the proposed system.

S. Iizuka (✉) · Y. Endo · J. Mitani · Y. Kanamori · Y. Fukui
Department of Computer Science, University of Tsukuba, 1-1-1 Tenno-dai, Tsukuba, Ibaraki 305-8573, Japan
e-mail: iizuka@npal.cs.tsukuba.ac.jp

Y. Endo
e-mail: endo@npal.cs.tsukuba.ac.jp

J. Mitani
e-mail: mitani@cs.tsukuba.ac.jp

Y. Kanamori
e-mail: kanamori@cs.tsukuba.ac.jp

Y. Fukui
e-mail: fukui@cs.tsukuba.ac.jp

J. Mitani
JST ERATO, 1-28-1 Koishigawa, Bunkyo-ku, Tokyo 112-0002, Japan

Keywords Pop-up card · Interactive design system · Mass–spring model

1 Introduction

The pop-up card is an interesting form of papercraft in which a three-dimensional structure appears when the card is opened, and the structure is folded flat when the card is closed. Pop-up cards are roughly classified into two types: 90-degree and 180-degree cards. The 90-degree card is constructed by adding cuts and folds to a single piece of paper so that a structure pops up when the card is opened 90 degrees. Such cards are sometimes referred to as *origami architectures*. The 180-degree card is constructed from multiple pieces of paper, and a structure pops up when the card is opened 180 degrees.

Several geometric constraints must be satisfied when a pop-up card is designed so that the three-dimensional structure pops up and can be folded flat without tearing or crushing. Knowledge of the constraints and mechanisms is required, and designing a pop-up card is difficult for the average person. The pop-up mechanisms of the 180-degree card are usually more difficult to understand intuitively, as compared with the 90-degree card, because the 180-degree card is assembled from multiple pieces. Some 180-degree cards designed by artists have very complicated structures. A practical approach to the design of a 180-degree card is to combine several prepared primitives, which are guaranteed to flatten when arranged appropriately. Using this approach, we can incrementally construct complicated shapes. However, problems such as collisions between parts or the protrusion of part of the structure often occur after the cards are assembled. Therefore, pop-up cards are usually designed by professionals who have knowledge and experience.

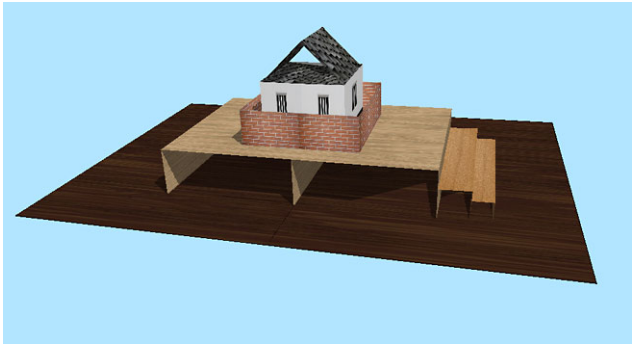


Fig. 1 An example of a 180-degrees type pop-up card designed with our system

In the present paper, we propose a design system that enables novice users to design original pop-up cards interactively using intuitive operations. The target card is a 180-degree card that is composed of several predefined primitives such as cylinders, cubes, and cones. Figure 1 shows an example target. In the proposed system, the users place primitives on a card and edit the scales, alignments, and leaning angles of these primitives using a simple interface. The alignments are adjusted automatically using the system based on its geometric constraints. With the help of the proposed system, the users can concentrate on editing without having to worry about the constraints. The detailed appearance of the primitives can be changed by mapping transparent texture images. The users can make actual pop-up cards by printing out templates of the parts generated by the system. Furthermore, the proposed system simulates the closing and opening motions of the pop-up card. The users can determine whether there are collisions between parts during closing and opening. Our system can also determine whether all of the parts fit into the card when the card is closed. This simulation helps the users avoid the trial-and-error process in making pop-up cards using actual materials.

The flow of designing a pop-up card using the proposed system is as follows:

1. Select a primitive from a pre-defined primitives list.
2. Specify the location at which the primitive is to be placed. The positions of fixed points are automatically adjusted such that they will be able to pop-up and be folded correctly.
3. Edit the positions, scales, and leaning angles of the primitive through mouse operations. The system guarantees that geometric constraints are satisfied even after these operations.
4. Confirm the simulated motion of closing and opening the card. When collisions between primitives or a protruding part of primitives are detected, the system displays a warning massage.
5. Iterate steps 1 through 4 in an arbitrary order until a desired pop-up card is obtained.
6. Print out and assemble the templates.

Okamura and Igarashi [10] proposed an interactive system for designing a 180-degree of pop-up card. The appearance of the proposed system is similar to the system they proposed. The difference lies in the method of simulating the closing and opening of the card. Their system analytically calculates the position of each vertex on primitives during the simulation. Since expressing the movements of all vertices in a primitive in analytical form is not straightforward, it is difficult to implement a wide variety of primitives on a system. Their system involves five types of primitives. On the other hand, the system proposed herein involves physical simulation based on a simple mass–spring model. The advantage of the proposed approach is the ease of adding different types of primitives to the system because movements of these primitives are simulated based on a common physical model. Moreover, primitives that have curved surfaces, such as cylinders, that are difficult to simulate analytically can be treated in the same framework. We implemented 14 types of primitives in the proposed system. Combining these multiple primitives, the user can design a greater variety of pop-up cards than is possible with existing systems.

The remainder of the present paper is organized as follows. Section 2 describes research related to the design of a pop-up card using a computer. Section 3 describes in detail the design of the structure of a pop-up card using the proposed system. Section 4 describes the physical simulation performed in order to compute the motion of a pop-up card during closing and opening. Section 5 shows examples of pop-up cards designed using the proposed system. Finally, conclusions are presented in Sect. 6.

2 Related work

Designing original pop-up cards by hand while considering the geometric constraints of the card is difficult for the average person. Several approaches to aid in the design of a pop-up card using a computer have been proposed.

2.1 90-degree card

Mitani and Suzuki [9] proposed a system for designing a 90-degree pop-up card in which a user can add vertical or horizontal polygons interactively. Hendrix and Eisenberg [5] proposed a system in which the user can design the flattened pattern of a pop-up card by adding prepared primitive patterns. In contrast to these interactive approaches, Li et al. [7] proposed a fully automated system for generating a pop-up card from an existing three-dimensional polygonal mesh model.

These methods, however, cannot be adopted for the design of a 180-degree card because they are based on the

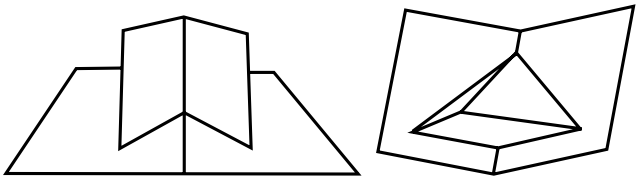


Fig. 2 V-fold (*left*) and single-slit (*right*)

geometric constraints of a 90-degree card, which is made using a single sheet. Therefore, we must consider different methods for designing a 180-degree card, which is assembled from several pieces.

2.2 180-degree card

Mitani and Suzuki [8] proposed a system for generating a 180-degree pop-up card from an existing three-dimensional polygonal mesh by composing its cross-sections in lattice form. However, this system cannot create the target pop-up cards shown in Fig. 1 because the construction is set only on the center fold line of the card. Lee et al. [6] described geometric constraints required for adding v-fold structures to a card. However, they did not implement a design system. Glassner [3, 4] described the motion of three basic primitives (v-fold and symmetric/asymmetric single-slit (Fig. 2)), and introduced analytic solutions for calculating the positions of vertices of primitives when the card is closed and opened. Based on this description, Okamura and Igarashi [10] implemented a system with an interactive user interface for designing a pop-up card, and they reported the results of a user study. Five primitives, based on v-fold and shingle-slit primitives, are prepared in their system. The user selects a primitive and places it on the card. The user then edits the shape under the constraints of the pop-up card. We implemented a user interface in the system proposed herein by referring to their system.

3 Designing pop-up cards

In this section, we describe how to design the structure of a pop-up card using the proposed system. A user first selects a primitive from 14 predefined types. The user then places the primitive on the empty card which consists of two rectangles connected by an edge along which the card is folded. The user then edits the shape of the primitive. A complicated structure is designed by adding primitives incrementally. In the following sections, these steps are described in detail.

3.1 Predefined primitives and edge types

Figure 3 shows the primitives implemented in the proposed system. When these primitives, most of which are introduced in the book "Elements Of Pop Up" [1], are aligned

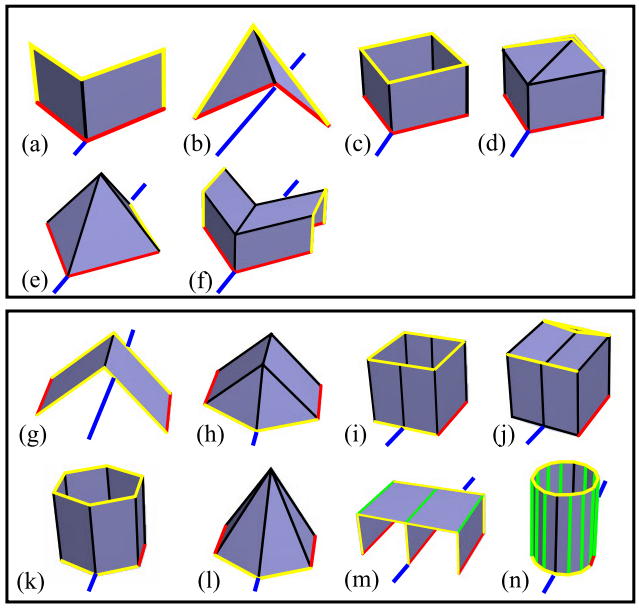


Fig. 3 Primitives implemented in the proposed system. These primitives are classified into two categories. **a** through **f** are v-fold primitives, and **g** through **n** are parallel-fold primitives

on an edge which is folded flat (blue lines in Fig. 3), they are also folded flat. Primitives (a), (c), (d), and (g) in Fig. 3 correspond to five primitives which are implemented in Okamura and Igarashi's system [10]. Primitive (g) represents two types of the primitives, the even-tent and the uneven-tent. The others are newly implemented in the proposed system.

A primitive is constructed of polygons, a set of triangles or planar quads. Each edge in a primitive belongs to one of the following types:

- *Fixed edges*: Edges fixed to the card or to a polygon of another primitive (red lines in Fig. 3).
- *Folding edges*: Edges that fold when the card is closed (black lines in Fig. 3).
- *Folding edges with elasticity*: Edges that have elasticity. This type of edges are introduced for the simulation of primitives having shapes that are defined by the elasticity of materials (green lines in Fig. 3).
- *Boundary edges*: Edges that are located on the boundaries of a primitive and not classified as fixed edges (yellow lines in Fig. 3).

When a primitive is placed on a card, its *fixed edges* act like *folding edges*. Therefore, other primitives are also placed on the *fixed edges* of previously placed primitives. We hereinafter refer to these edges on which primitives are placed as *base edges*.

Primitives can be classified into two types, namely, *v-fold* primitives (Figs. 3(a) through (f)) and *parallel-fold* primitives (Figs. 3(g) through (n)), based on how the primitives are fixed over base edges. The v-fold primitives are fixed at

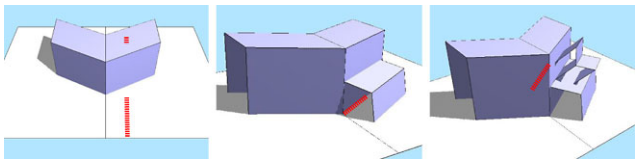


Fig. 4 Primitives are placed on base edges (shown in *red lines*)

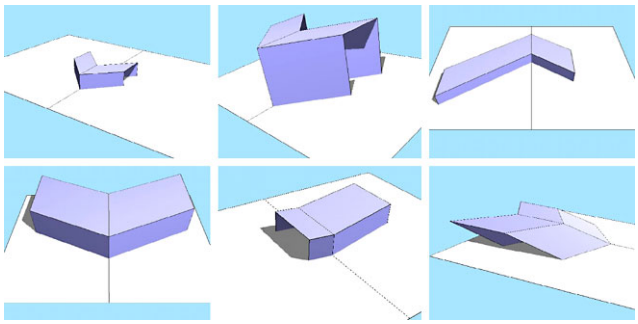


Fig. 5 A user can edit the shapes of primitives by changing the width, height, depth, and leaning angle while satisfying the constraints. These examples are obtained from primitive (f) in Fig. 3

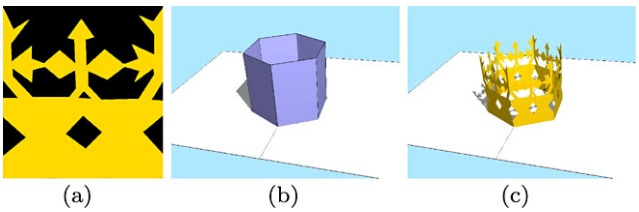


Fig. 6 The appearance of a primitive is changed by applying a transparent texture image. **a** The texture image. Transparent parts are colored black. **b** The shape of a primitive. **c** The appearance of the primitive when the texture is applied to each polygon

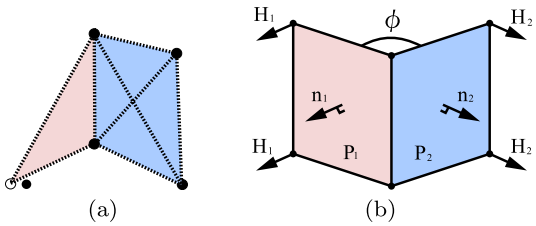


Fig. 7 Mass–spring model. **a** A triangle and a rectangle are represented by a network of springs. **b** A hinge-spring is added to an edge for which the angle must be maintained constant

two edges to polygons of the card or other primitives. These two edges are not parallel to the base edge. The parallel-fold primitives have two or more fixed parallel edges.

3.2 Placing and editing primitives

In order to place a primitive on a card, the user selects a base edge and then specifies a position on the edge. Since, at the beginning, there is only a single base edge, which connects the two rectangles of the card, the first primitive is placed onto this base edge, after which the card acquires several base edges. The user constructs a complicated structure by repeatedly selecting primitives and placing these primitives on base edges, as shown in Fig. 4.

The user can edit the shape of a primitive after placement by changing the width, height, depth, and leaning angle by dragging a mouse. Not to violate the geometric constraints, angles between the base edge and the fixed edges on the primitive are kept constant. The user interface implemented in the proposed system is basically the same as that introduced by Okamura and Igarashi [10]. Figure 5 shows some edited examples of primitive (f) in Fig. 3.

In addition, the user can place a transparent texture image on a polygon of a primitive in order to change the appearance of the polygon, as shown in Fig. 6.

4 Physical simulation of opening and closing

An important factor of the pop-up card is the motion that occurs as the three-dimensional structure appears from the closed flat card. Simulating the motion of the structure is needed in order to confirm the effect of the pop-up cards. Furthermore, the user can check for the existence of errors in the structure by means of the simulation before constructing an actual pop-up card. We performed a physical simulation based on the mass–spring model.

4.1 Computing the coordinates of vertices using the mass–spring model

We follow an approach proposed by Furuta et al. [2] in which the motion of a piece of folded paper is simulated using a mass–spring model. They used this model to simulate origami. In the present study, we use this model to simulate pop-up cards.

The shape of a primitive is represented by a network of springs. For example, three springs are placed on the edges of a triangle, and six springs are placed on edges and diagonals of a quadrangle, as shown in Fig. 7(a).

While the vertices of the *fixed edges* are fixed to the card or other primitives, other vertices move so as to maintain the lengths of the springs at their rest lengths during a simulation. As a result, the whole structure pops-up when the card opens. When the card closes, the structure flattens.

The force $\mathbf{F}_i$ acting on a vertex i, the velocity $\mathbf{v}_i$, and the coordinate $\mathbf{r}_i$ of the vertex i are defined as follows:

$$\mathbf{F}_i = \sum_j \{k(|\mathbf{r}_{ij}| - L_{ij})\mathbf{r}_{ij} - D\mathbf{v}_{ij}\}, \tag{1}$$

$$\mathbf{v}_i(t + \Delta t) = \mathbf{v}_i(t) + \frac{\mathbf{F}_i(t)}{m}\Delta t, \tag{2}$$

$$\mathbf{r}_i(t + \Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\Delta t, \tag{3}$$

where t is the simulation time, Δt is the time step, k is the spring constant, L_{ij} is the rest length of the spring between vertices i and j, $\mathbf{r}_{ij}$ is the relative coordinate of vertex i to vertex j, $\mathbf{v}_{ij}$ is the relative velocity of vertex i to vertex j, D is the damper constant, and m is the mass of a vertex. In the present implementation, we set $k = 8$, $D = 0.02$, and $m = 100$. We use the explicit Euler method to calculate the position of each vertex.

Although this model works well for almost all of the primitives, primitives (m) and (n) in Fig. 3 which pop-up due to the elasticity of paper do not move as expected. Figure 8 shows an example of the problem with primitive (m). The reason for this is that the simulation converges when the lengths of all of springs reach the rest lengths. In order to solve this problem, we employ hinge-springs which simulate the elasticity of paper. The hinge-springs are applied to edges for which the angle must be maintained.

A pair of forces $\mathbf{H}_1$ and $\mathbf{H}_2$ generated by a hinge-spring act on vertices on polygons P_1 and P_2 which are connected to the hinge-spring as shown in Fig. 7(b). The forces are described as follows:

$$\mathbf{H}_1 = k_h(\phi_0 - \phi)\mathbf{n}_1, \tag{4}$$

$$\mathbf{H}_2 = k_h(\phi_0 - \phi)\mathbf{n}_2, \tag{5}$$

where ϕ_0 and ϕ denote the rest angle and the current angle between P_1 and P_2, respectively. The spring constant of the hinge-spring is denoted as k_h, and $\mathbf{n}_1$ and $\mathbf{n}_2$ are the unit normal vectors of P_1 and P_2, respectively. We set $k_h = 0.02$.

Using the forces generated by the hinge-springs, all of the primitives listed in Fig. 3 move as expected. Figure 9 shows examples of the motion of (m) and (n) in Fig. 3. The cylindrical primitive (n) is divided into 14 quads, and hinge-springs having rest angles of 180 degrees are placed along all of the vertical edges. Although this primitive is a rough approximation of a cylinder, the primitive moves as a realistic cylinder in the physical simulation of the present study. A typical example using this primitive is shown in the bottom of Fig. 10.

Fig. 8 A primitive without hinge springs that does not pop-up appropriately

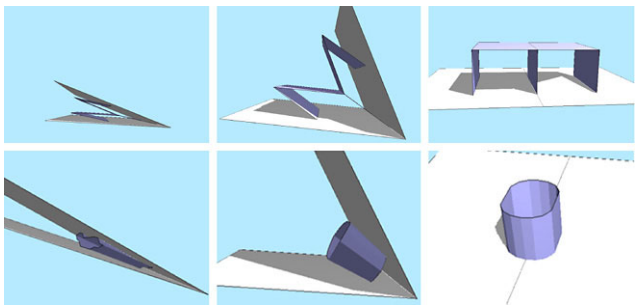


Fig. 9 The motions of primitives (m) and (n) after hinge springs are added

4.2 Error detection

The proposed system detects collisions between primitives and protrusions during a simulation. Specifically, the system determines whether the edges of a primitive penetrate the polygons of other primitives and whether the coordinates of all of the vertices are inside the card when the card is closed. The system displays a warning message when a problem is detected. The user can then correct the problematic part before constructing an actual pop-up card.

5 Results

We implemented the proposed system using C++ and OpenGL GLUT, and ran program on a PC with an Intel Core i7 620M (2.67 GHz, 4.00 GB RAM) and an NVIDIA Quadro NVS 3100M. We designed a number of pop-up cards using the proposed system and created actual cards by assembling the printed patterns. The results are shown in Fig. 10. Designing a card required approximately 10 to 20 minutes, excluding the time required to prepare texture images. The top and the middle rows in Fig. 10 show examples of a castle and a tortoise, respectively. The bottom row in Fig. 10 shows an example of the Leaning Tower of Pisa constructed from a cylinder and two v-folds. The templates of the parts are generated by the system. Flaps and alignment marks for gluing are automatically added to the templates of primitives according to their relative position. The correspondence between flaps and alignment marks are distinguished by the colors.

Figure 11 shows an example of a complicated structure designed by an author. Four frames were captured during a physical simulation. These pop-up cards contain primitives that are difficult to realize using existing analytical approaches (primitives (m) and (n) in Fig. 3).

We conducted an informal user study involving two computer science students in order to examine how these subjects would design original pop-up cards. Neither of the subjects had designed an original pop-up card prior to this study. The subjects took a 10-minute lecture before starting to design their own pop-up cards using the proposed system. Figure 12 shows two of the pop-up cards designed in this study. Designing Figs. 12(a) and (b) required approximately 15 and 30 minutes, respectively. By making actual

Fig. 10 Examples of pop-up cards designed using the proposed system. Top to bottom: a castle, a tortoise, and the Leaning Tower of Pisa. Left to right: screen shots of the proposed system, generated templates, and constructed cards

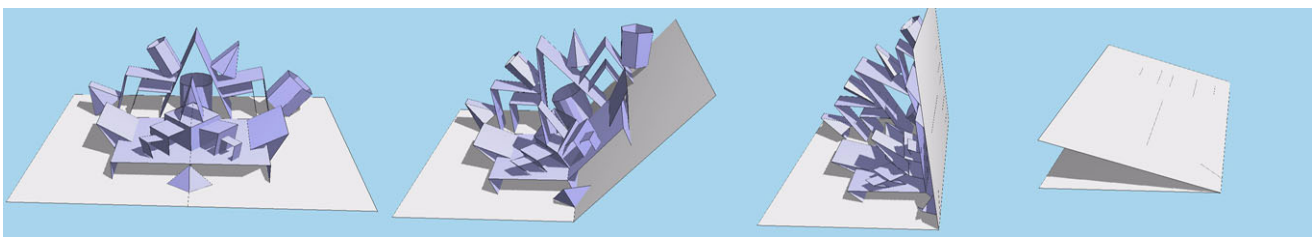


Fig. 11 Animation of a complicated pop-up card

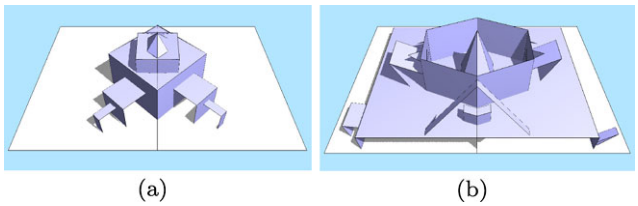


Fig. 12 Pop-up cards designed with two test users

pop-up cards, we confirmed that both of the pop-up cards were flat when closed. After the study, we received positive comments, such as "Viewing the motion of pop-up cards through the simulation is fun." and "The approach of putting primitives on a card is interesting." On the other hand, we also received the following comment: "I want to place a primitive as a bridge between two separated primitives". At present, this operation is not available using the proposed system.

6 Conclusion and future research

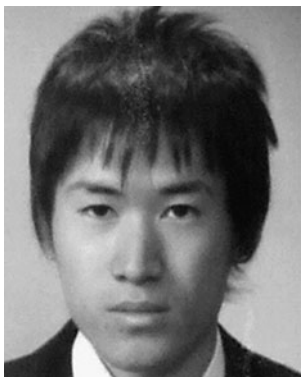
We have presented an interactive system for designing pop-up cards. Using a physical simulation based on a mass–spring model, we were able to easily prepare several primitives using the proposed system. A number of primitives, including the shapes of which change due to the elasticity of paper, are appropriately simulated by applying hinge-springs. The proposed system enables the user to easily design pop-up cards without special knowledge of pop-up card design.

On the other hand, there are limitations in the physical simulation implemented on our system. Although detailed appearances of primitives are modified by mapping transparent texture images as described in Sect. 3.2, this does not change the structures of spring networks. If large holes are added to a primitive by mapping an image, the holes will be ignored in the simulation. Further, collisions between primitives do not affect the deformation of them. In the future, in order to take full advantage of the physical simulation against the analytical approach, we intend to improve collision detection and simulate the deformation of primitives due to collisions. We will also attempt to add more flexibility to the editing operations in order to enable the design of complicated mechanisms, like those that appear in cards designed by artists.

Acknowledgements This work was supported by the Grant-in-Aid for the Program for Enhancing Systematic Education in Graduate Schools, "Program for Development of ICT Solution Architects", from the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

References

1. Elements of Pop Up: A Pop Up Book for Aspiring Paper Engineers
2. Furuta, Y., Kimoto, H., Mitani, J., Fukui, Y.: Computer model and mouse interface for interactive virtual origami operation. IPSJ J. **48**(12), 3658–3669 (2007) (in Japanese)
3. Glassner, A.: Interactive pop-up card design, part 1. IEEE Comput. Graph. Appl. **22**(1), 79–86 (2002)
4. Glassner, A.: Interactive pop-up card design, part 2. IEEE Comput. Graph. Appl. **22**(2), 74–85 (2002)
5. Hendrix, S.L., Eisenberg, M.A.: Computer-assisted pop-up design for children: computationally enriched paper engineering. Adv. Technol. Learn. **3**(2), 119–127 (2006)
6. Lee, Y.T., Tor, S.B., Soo, E.L.: Mathematical modelling and simulation of pop-up books. In: Proc. of Computers & Graphics 1996, vol. 20(1), pp. 21–31 (1996)
7. Li, X.Y., Shen, C.H., Huang, S.S., Ju, T., Hu, S.M.: Popop: automatic paper architectures from 3D models. ACM Trans. Graph. **29**(4), 111 (2010)
8. Mitani, J., Suzuki, H.: Computer aided design for 180-degree flat fold origamic architecture with lattice-type cross sections. J. Graph. Sci. Jpn. **37**, 3 (2003) (in Japanese)
9. Mitani, J., Suzuki, H.: Computer aided design for origamic architecture models with polygonal representation. In: Computer Graphics International 2004, Crete, Greece, 16–19 June, pp. 93–99 (2004)
10. Okamura, S., Igarashi, T.: An interface for assisting the design and production of pop-up card. In: Proc. of Smart Graphics 2009, pp. 68–78 (2009)



Satoshi Iizuka received his B.S degree in Computer Science from the University of Tsukuba in 2010, Japan. Since April 2010, he has been a Master's degree student in the Department of Computer Science at University of Tsukuba. His research interests center on computer graphics and include interactive simulations, image-based modeling and image editing.

Yuki Endo received his B.S. degree in Computer Science from the University of Tsukuba, Japan, in 2010. Since April 2010, he has been a Master's degree student in the Department of Computer Science at University of Tsukuba. His research interests center on computer graphics and include interactive simulations and modeling of natural phenomena.

Jun Mitani received his Ph.D. in Engineering from the University of Tokyo in 2004 for his research on designing 3D papercraft models with a computer. In 2006, he started working at University of Tsukuba. His present post is an associate professor at the Graduate School of Systems and Information Engineering in University of Tsukuba. He is studying about designing three-dimensional curved origami with a computer as part of research activities.

Yoshihiro Kanamori received his B.S., M.S., and Ph.D. degrees in Computer Science from the University of Tokyo, Japan, in 2003, 2005, and 2009, respectively. Since April 2009, he has been an assistant professor in the Department of Computer Science at University of Tsukuba. His research interests center on computer graphics and include real-time rendering, point-based graphics, and interactive simulations.

Yukio Fukui received his Ph.D. in Engineering from the University of Tokyo in 1993. He had been working at the National Institute of Bioscience and Human-Technology, M.I.T.I. and the Institute of Information Sciences and Electronics at University of Tsukuba since 1993 and 1998, respectively. He has been a professor in the Department of Computer Science at University of Tsukuba since 2004.