THE JOURNAL OF VISUALIZATION AND COMPUTER ANIMATION, VOL. 7: 25–42 (1996)

An Origami Playing Simulator in the Virtual Space

SHIN-YA MIYAZAKI
School of Computer and Cognitive Sciences, Chukyo University, 101 Tokodate, Kaizu-cho, Toyota, Aichi, Japan

TAKAMI YASUDA AND SHIGEKI YOKOI
School of Informatics and Sciences, Nagoya University, Nagoya, Japan

AND

JUN-ICHIRO TORIWAKI
Department of Information Engineering, Faculty of Engineering, Nagoya University, Nagoya, Japan

SUMMARY

A virtual manipulation system for origami (paper folding) is described. A piece of paper defined in a computer can be deformed interactively by picking and moving a corner vertex of a paper face on a graphic screen using a mouse. Three kinds of folding operations and a curving operation transform the paper into a three-dimensional figure made of flat or curved surfaces. The roundness of the curved surfaces is calculated in real-time by minimizing an elastic energy function. The simulated paper has traditional Japanese decoration applied as a texture to give a more realistic appearance. It is rendered with shading in real-time.

KEY WORDS: virtual manipulation; interactive simulation; paper folding; real-time interaction; natural user interface; virtual reality

INTRODUCTION

Improvements in graphics hardware have brought human–machine communication to the point where various computer models can be manipulated interactively using a graphical display as if they were real models. VR technology has great potential for application in various fields such as design, entertainment, education, training and medicine. Over the past few years, a considerable number of developments in hardware have been made for 3D sensing, stereopsis, and force feedback as VR key technologies.[1–7] On the other hand, only a few attempts have ever been made to develop software to manipulate 3D models, except for simple examples such as presentation of moving rigid objects. We have been developing a virtual manipulation system for paper folding[8] as an example of handling non-rigid materials. To this end we have designed a natural user interface and a computer model of paper for the system.

CCC 1049–8907/96/010025–18

Received 29 July 1994
Revised 23 March 1995

In this paper, we present an advanced paper manipulation system which enables us to fold or to curve naturally a piece of paper defined by geometrical and physical models through a simple mouse interface. Folding operations are performed only by picking and moving a corner vertex of the paper face on a graphic screen while changing the viewing direction freely. We provide a *curving* operation to curve the paper in addition to three kinds of basic plane folding operations called *bending*, *folding up* and *tucking in*. The *curving* operation is important for realizing a wider class of folding shapes compared to the original plane folding system. The cross-sectional shape of the curved face is determined in real-time using an elastic model of the paper constrained by the floor plane. Successive application of these operations transforms the paper into a complex figure including plane and curved surfaces. A paper face has colourful textures, such as *yuzen* often used in Japanese traditional origami. We developed a new function by which texture images selected from a pop-up menu can be mapped onto either side of the paper.

Related work

Several research projects on computer processing of origami have been reported. Agui *et al.*[9] developed a method to generate an animation of the paper folding process from several key frame images. However, they did not develop a way to represent the state of a paper sheet accurately. They did not include connectivity or positional relations between overlapped paper faces, since their application did not need it. Their application only required a method for interpolation between two images. Uchida and Itoh[10] reported a method to derive a sequence of folding processes from the unfolded state of the paper with fold-lines. Although they produced a data structure to represent folded paper sheets, this scheme could not be used for interactive manipulation. Komori *et al.*[11] developed a fundamental interactive manipulation system for paper folding for the first time, and Konno and Itahashi[12] summarized the types of folding and described interactive instructions for performing them on a personal computer. However, they neither developed natural operations for folding paper nor an effective geometrical model for a complex multiply folded state. The mathematical theory of origami operations is another field of study in which many reports have been published,[13] but computer simulation of generating origami work has not been discussed.

On the other hand, there are many reports on the use of geometrical and physically-based modelling for animating deformable objects such as rubber, cloth and paper.[14–20] Their results prove the validity and effectiveness of modelling based on physics for the description of the shape and motion of deformable objects. However, it is not easy to apply these to interactive systems because of the immense computation time required. To achieve interactive simulation with physical models, it is necessary to simplify the model so that it can be computed in real-time and to develop software that describes exactly the natural interface between an input device and the model.

INTERACTION

This section presents basic folding operations and some functions implemented in our system to cope with actual folding manipulation. In a folding operation, one of the corner vertices of a paper face is picked and moved according to a mouse drag. Vertex

positions are automatically corrected for each manipulation in a few cases, as is explained later.

Folding operations

Three types of folding operations with a crease and a curving operation are provided in the system.

1. Bending and folding up

Each folding operation is defined by dividing a face viewed on the screen into two parts along a fold-line and rotating one of them around the line. The operation is called *bending* when all of the folded faces are rotated simultaneously in the same direction (Plate 1(a)). In particular, the operation where the rotation angle is 180° is called *folding up* (Plate 1(b)). Therefore, after a *folding up* operation, all the vertices are in the same plane. We have to pay atention to the connectivity or overlap relations between the faces to realize consistent paper folding processes.

2. Tucking in

The third type of folding operation, called *tucking in*, folds faces to the inside direction bounded by a certain face (Plate 1(c)). In the present system, *tucking in* is limited to a symmetrical operation shown in Plate 1(c). Selection of *folding up* or *tucking in* is specified by a button. More complex folding can be realized by combining more than one *tucking in* and *folding up*. Figure 1 shows two stages in folding a paper crane.

3. Curving

The *curving* operation makes a face bend along a smooth continuously curved surface (Plate 1(d)). The shape of the curved face is decided using a simple dynamics model simulating elastic forces in the paper constrained by the position of the floor.

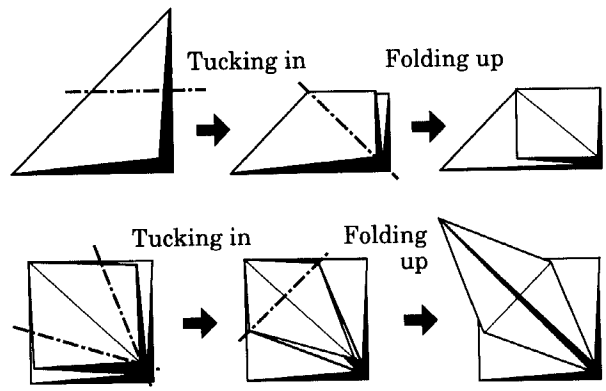


Figure 1. Complex folding realized by combining simple folding. These are two stages in folding a paper crane

The resulting curved face consists of a round part and a flat part on the floor. The *curving* operation can be selected from the other folding stages with a button.

Basic operations with the mouse

In the system a conventional 2D mouse is used as the input device to pick and move a vertex of the face. A folding operation is performed by dragging the mouse in three stages: picking a vertex, moving the vertex and releasing the vertex. The mouse cursor specifies the vertex position on the screen and two buttons control the third dimension, depth from the screen.

1. Picking a vertex

When we push the middle button of the mouse, the vertex nearest to the cursor on the screen is selected as the picked vertex to be moved and the cursor is fixed just on the picked vertex (Figure 2(a)). If there is more than one such vertex, the vertex belonging to the face nearest to the viewer is selected.

2. Moving the picked vertex

While the middle button is being pushed, the state of the folded paper is renewed and displayed automatically every moment based on the current position of the picked vertex (Figure 2(b)). As this updating of the paper state is performed in a very short time, the operator can transform the paper by moving the picked vertex directly, monitoring smooth modification of the paper and feeling as if he were deforming a paper sheet continuously. During the moving operation, if the left button on the mouse is pushed, the picked vertex moves away along the viewing line, while the right button moves the vertex towards the viewer.

3. Releasing the picked vertex

When the button is released, the position of the picked vertex is fixed and the current state of the folded paper is recorded to finish the folding process. The mouse cursor is moved freely, independent of any vertex, afterwards (Figure 2(c)).

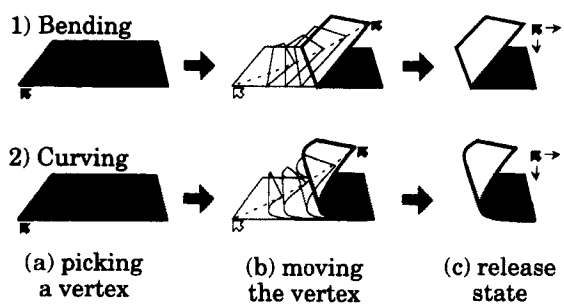


Figure 2. Three stages in a folding operation. They are performed by dragging the mouse. The shape of the face is modified smoothly according to the movement of the mouse

Correcting vertex position after movement

It is difficult and tedious work to put the picked vertex onto the position of another vertex exactly or to superimpose one edge on another edge accurately in the virtual space using only the mouse. In our system, if the picked vertex comes close enough to the operator's intended position, the position of the vertex is automatically corrected in the following four cases:

(a) matching two vertices
(b) superimposing two edges
(c) when the fold-line goes through two vertices
(d) when the rotation angle takes a special value such as 90°.

These corrections make the operator feel as though corresponding vertices or edges attract each other. As a result, the operator can fold a sheet of virtual paper more rapidly and accurately than folding a real one. In the *curving* operation, without creases, automatic corrections (a) and (b) are available.

Texture mapping and shading

Traditional Japanese origami paper has beautiful colour patterns and textured images printed on both sides. The Silicon Graphics 'Crimson' workstation with the Reality engine performs both texture mapping and Gouraud shading in real-time. This combination significantly improves the realism of synthesized origami products. We developed a new function to change the texture mapped onto a face. A suitable texture can be selected from a list of previously stored digital images using a pop-up menu. Our data structure and method of manipulation are chosen to perform consistent texture mapping in association with paper folding operations. For example, texture patterns must be continuous at any of the boundaries between faces. The vertices of the face are projected onto the 2D texture images. We also record which side of the paper faces the viewer. Every vertex keeps the 2D co-ordinates that represent the position on the original texture image, as well as the 3D co-ordinates to represent the position in 3D space. The 2D co-ordinates are considered to be the texture co-ordinates (Figure 3). When a new vertex is generated by the edge division process, the texture co-ordinates of the vertex are derived from the texture co-ordinates of the end-vertices of the edge.

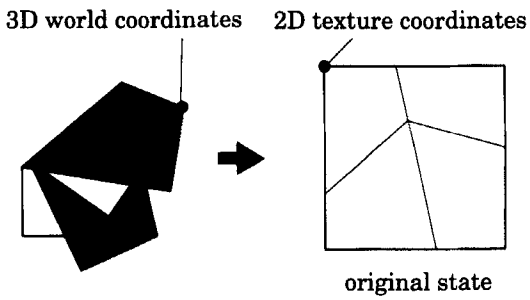


Figure 3. Two kinds of co-ordinates of the vertices. The 2D co-ordinates on the original texture image correspond to the texture co-ordinates

Changing viewpoint

The system provides the functions to rotate a viewing direction around X, Y and Z axes and to move the viewpoint along X axis or equivalently to rotate and move an object if the object is located at the origin and the viewpoint is located on the X axis. Each operation is controlled with two keys on the keyboard. With the combination of them, the folded paper in the 3D space is observed from an arbitrary direction.

FOLDING WITHOUT CURVED FACES

This section describes the data structure to determine the state of the folded paper exactly. The folded paper consists of two kinds of face, flat and curved. These faces are segmented by the fold-line or the boundary between the flat part and the round one. In this section, we will describe the data structure and its renewal for folded paper consisting of flat faces only. The data structure for curved faces is discussed in the next section.

Folding line and rotation angle

In a folding operation with creases, the rotation axis (the fold-line) and the rotation angle of the moving part of the paper must be decided according to the place to which the picked vertex is moved. The fold-line is defined as the intersection of the folded face and the 'equidistant' plane which consists of all the points equidistant from the positions of the picked vertex before and after the movement. The rotation angle is defined to be twice the angle between these two planes (Figure 4).

Data structure

Overview

The data in the system consists of three layers: the face layer, the edge layer and the vertex layer. Together they represent a complex surface where faces share edges and edges share faces and vertices:

1. Face layer. The face layer is composed of the face-cell tree and the face-cell look-

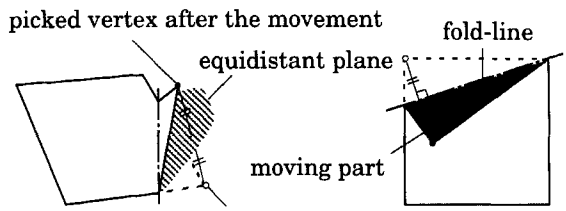


Figure 4. The fold-line and the rotation angle of the moving part of the paper

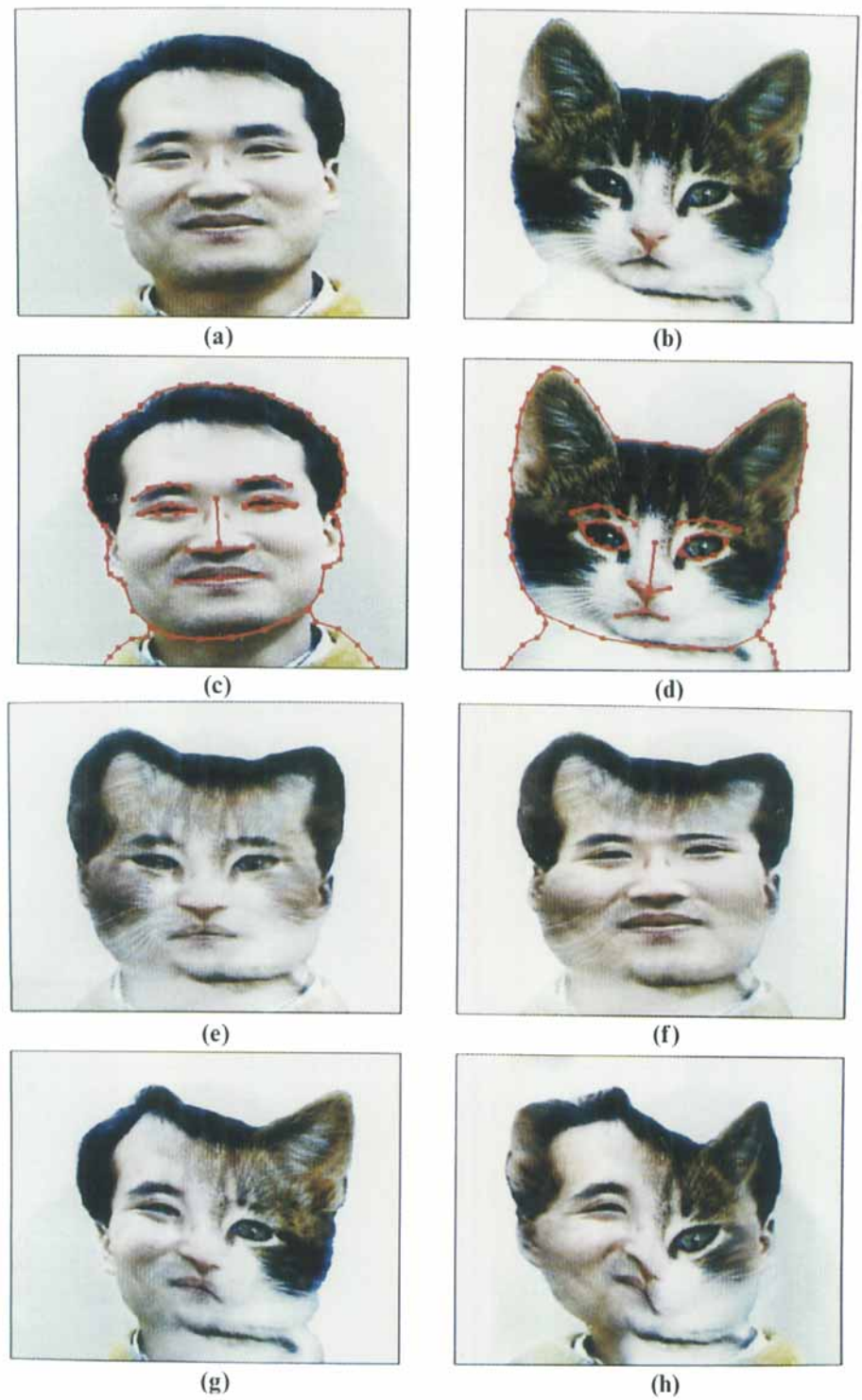


Plate 1 (Lee et al.). A metamorphosis example: from a person to a cat

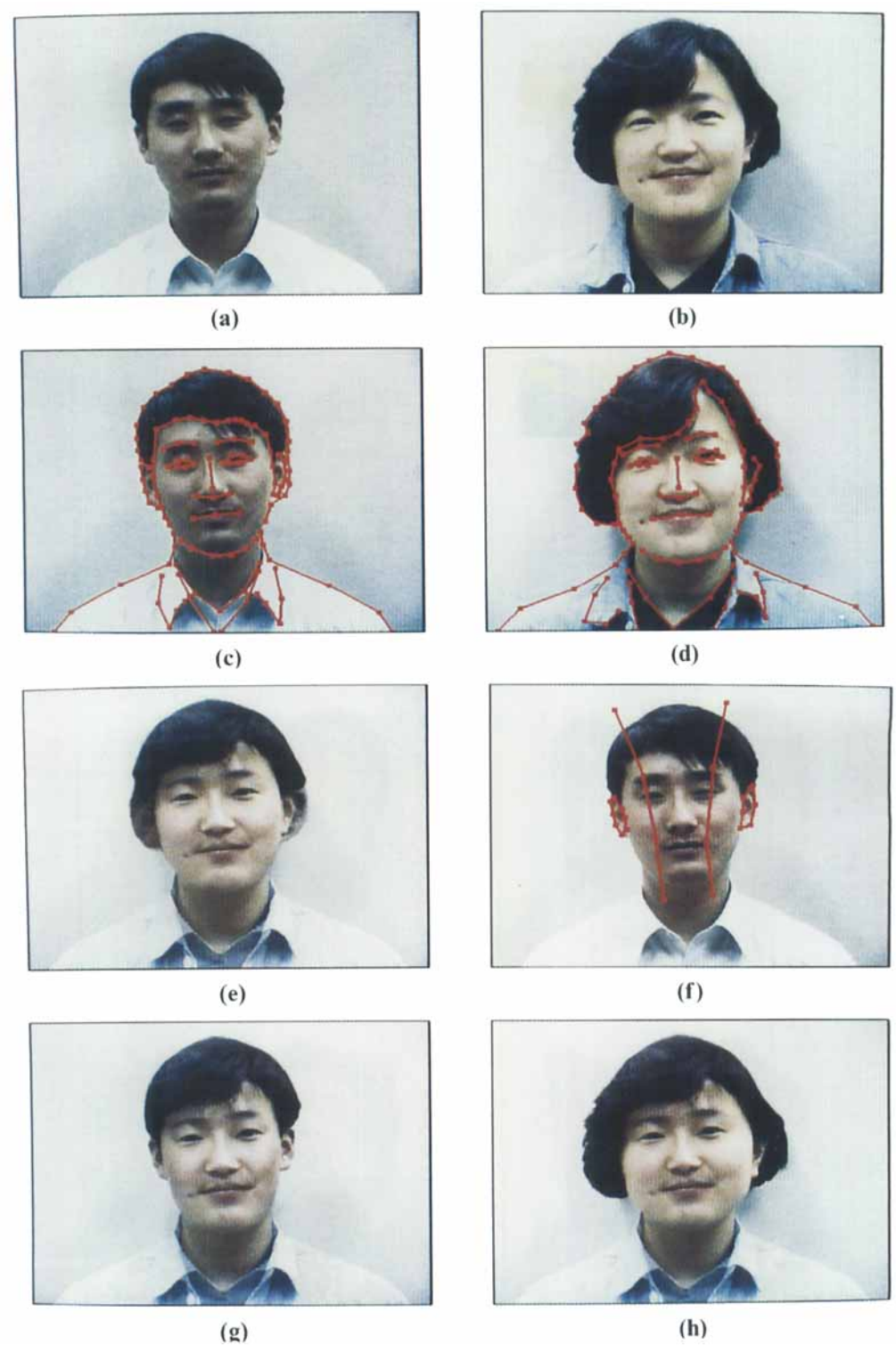


Plate 2 (Lee et al.). Transition control for overcoming an inconsistency between features

(a) Bending

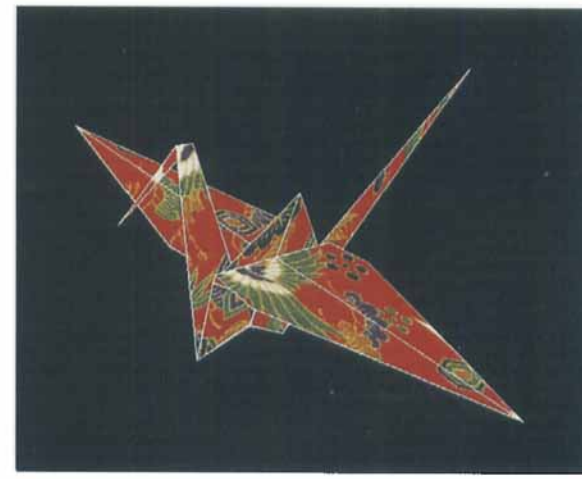
(b) Folding up

(c) Tucking in

(d) Curving

Plate 1 (Miyazaki et al.). Three types of folding operation with a crease and a curving operation. A curved face is approximated by a set of ribbon-like polygons

(a) Plane



(b) Crane

(c) Penguin

(d) Tray

Plate 2 (Miyazaki et al.). Examples of Origami work. (c) and (d) contain curved faces

up table (Figure 5(a)). The face-cell tree has a binary tree structure and represents the face division. The face-cell look-up table keeps the order of faces in a pile and is used to search quickly for a given face-cell from the face-cell tree list.

2. Edge layer. The edge layer includes edge-cell trees (Figure 5(b)). Edge-cell trees are used for recording edge division processes and have almost the same structure as the face-cell tree. When a new edge is generated in a face division process, however, the number of edge-cell trees is increased by one.
3. Vertex layer. The vertex layer consists of vertex-cell linear lists and the vertex-

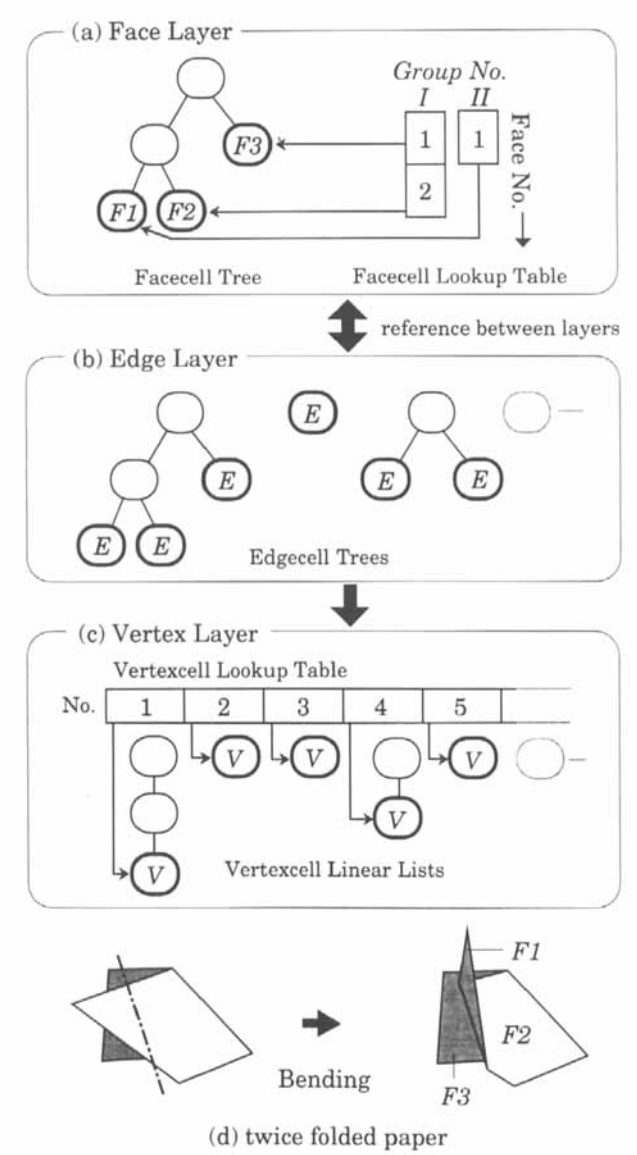


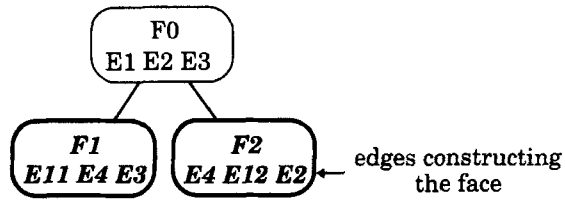
Figure 5. The data structure for a multiply folded state. Binary trees and linear lists of cells represent the states of faces, edges and vertices. The whole data (a), (b) and (c) represents the folded state (d)

cell look-up table (Figure 5(c)). Spatial co-ordinates of vertices are stored in the linear list of vertex-cells. The vertex-cell look-up table keeps the leaves of lists and the current state of a vertex. It is also used to search quickly for a vertex in the vertex list.

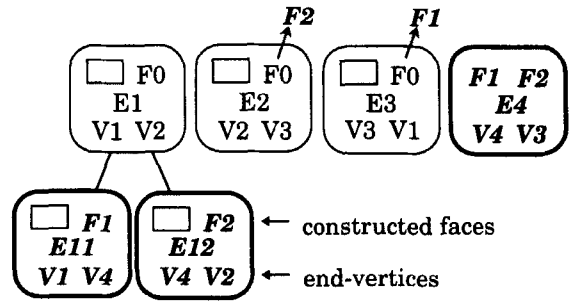
Each data cell is connected to related cells in the different layers by pointers. The structure can be searched quickly during the list renewal and the face rendering processes. For example, a face-cell has an array of pointers to all edge-cells constructing the face (Figure 6(a)). An edge-cell also has one or two pointers to face-cells including the edge and two vertex numbers of the end-vertices (Figure 6(b)).

Binary tree structure

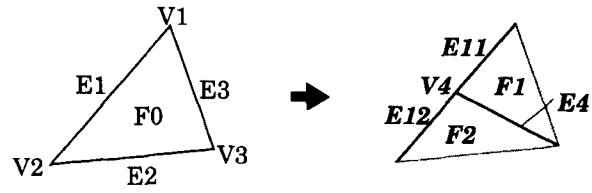
Since each face may be divided into two parts after a folding operation, the binary tree structure is suitable for the description of face information. In the structure, a node of a tree corresponds to a face (a face-cell). When a face is divided into two faces, two branch nodes to denote them are generated under the parent node. The set of leaves



(a) generated facecells and information



(b) generated edgecells and information



(c) a divided triangular face

Figure 6. Renewal of trees according to the face division

in the tree represents the current state of the folded paper, and the whole tree has all information about face division processes by the folding operations.

Face group and face stack

All faces may not be on the same plane after the *bending* operation is applied. They are classified into a few groups (*face group*) by the plane in which they lie. Each face is stored in the face stack of the corresponding group, in which all faces lie on the same plane and are given consecutive numbers according to the face order (*face number*). In Figure 5(d), for example, only *F1* (the vertical face) of the three faces constitutes a group by itself. If a pair of the group number and the face number is given, the face-cell look-up table finds the corresponding face-cell quickly from the face-cell tree list.

Classifying faces

All divided faces in folding are searched through the following two stages. The face layer and edge layer are modified only for those divided faces.

1. Searching moved faces

While the selected face in the picking process is moved, some of other faces may have to be moved with it. In the case of folding in Figure 7(a), F2 is moved in association with the movement of F1 due to the connectivity of the faces. Only F3 is not moved (a fixed face in this operation). Here the moved faces are extracted as follows:

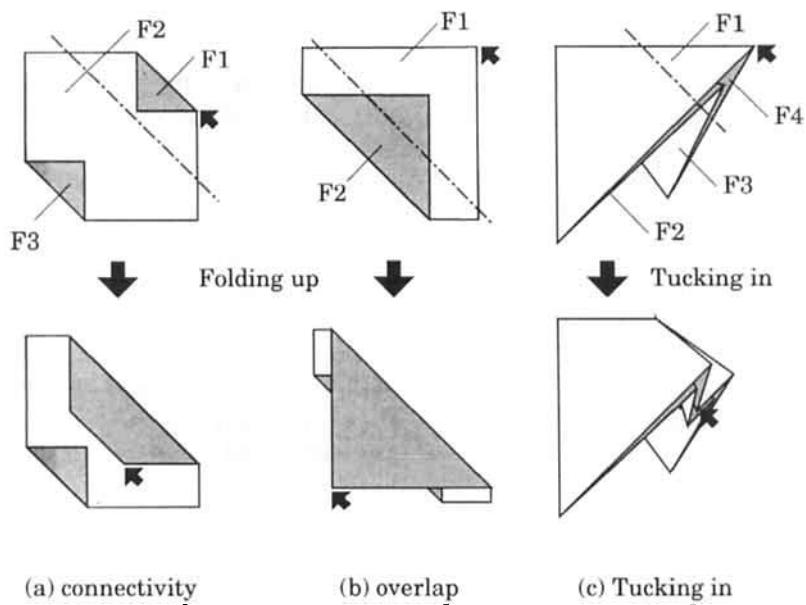


Figure 7. Three kinds of faces in a folding operation. Some faces are moved, and some of them are still divided

(a) The selected face is obviously a moved face (F1 in Figures 7(a), (b), and (c)).
(b) If there is a face connected with the moving part of any moved face, it also becomes a moved face (F2 in Figure 7(a) and F4 in Figure 7(c)).
(c) In the case of *bending* and *folding up*, if there is a face overlapped with the moving part of any moved face in the rotating side, it becomes a moved face (F2 in Figure 7(b)).
(d) In the case of *tucking in*, atin first, two outside faces to be moved are decided by the procedures (a) and (b) (F1 and F4 in Figure 7(c)). Then the other moved faces between them are found by face overlap (F2 and F3 in Figure 7(c)).
(e) Other faces are fixed faces (F3 in Figure 7(a)).

2. Classifying moved faces

All moved faces are divided into two classes according to the location of the face relative to the fold-line. One class contains a face if it lies across the fold-line. In this case it is divided into two faces (F2 in Figure 7(a) and all faces in Figures 7(b) and (c)). The other class contains faces lying on one side of the fold-line. These are not divided (F1 in Figure 7(a)). The classification is performed by examining the locations of face vertices relative to the fold-line.

List renewal

The lists in the data structure are mofidied by folding operations. Face-cells and edge-cells are simultaneously renewed when the corresponding faces are divided. Renewal procedures of the vertices' positions and the face order are independent of each other.

1. Face-cells and edge-cells

Let us explain the modification procedure for data cells using the example of a divided triangular face in Figure 6. The faces F0 has three edges (E1, E2, E3) and three vertices (V1, V2, V3). F0 is divided into two faces, *F1* and *F2*, and the edge E1 is divided into two edges, *E11* and *E12* by the fold-line. Here, a new edge *E4* and a new vertex *V4* are generated. The structure is modified as follows:

(a) For the division of face F0, new face-cells *F1* and *F2* (children of F0) and a new edge-cell *E4* (the root of a new edge-cell tree) are generated.
(b) For the division of edge E1, a new vertex-cell *V4* is produced and added to the vertex-cell look-up table, and the location of *V4* is calculated. Then, the new edge-cells *E11* and *E12* (children of E1) are created, and the end-vertices of them are fixed to (V1, *V4*) and (V2, *V4*), respectively. The end-vertices of *E4* are given as (V3, *V4*). *E11* and *E4* are registered as the edges associated with *F1*, and *E4* and *E12* as those with *F2*.
(c) The edge E2 is registered to *F2*. The face belonging to E2 is modified from F0 to *F2*.
(d) The edge E3 is registered to *F1*. The face F0 including E3 is modified to *F1*.

A face with more than four edges is processed in the same manner. The above procedures are applied to all divided faces in multiple folding.

2. Vertex-cells

Vertex-cells are renewed independently of face-cells and edge-cells. A new vertex-cell is generated for each moved vertex and the new location of the vertex is calculated. The vertex-cell look-up table picks up these new vertex-cells afterwards. The state of the vertex layer before renewal is maintained until new vertices are generated by the edge division.

3. Face-cell look-up table

A new face-cell look-up table is created from the table of the face group being manipulated. A folding operation is applied to faces on the same plane. The leaf face-cells of the group are registered to the new table in accordance with the new order of the face stack. Figure 8 shows procedures for ordering new faces in each folding. In *bending*, a set of all moved faces is reordered and a new face group is constructed. In *folding up*, the order of a group of faces to be folded is reversed and piled on the list of other faces. In *tucking in*, the similar procedure in *folding up* is applied by changing the order of faces.

FOLDING WITH CURVED FACES

The basic operation in origami is plane folding in the sense that the surface produced by folding is always a flat plane, and using only such operations we can create a wide variety of forms. However, forms with curved surfaces often show beautiful shapes. In fact, good pieces of hand-made origami work sometimes contain curved surfaces generated naturally. Therefore, we developed a function to fold a sheet with curved surfaces. A model taking into account the physical properties of paper must be developed to implement realistic curving operations. The operation in the case of plane folding was realized by defining the fold-line as the half-way line between a selected vertex and its moved position. The curving operation defines a cylindrical shape or a conic face constrained by a floor surface and the selected vertex. If we employ the cylindrical shape for the curving face, the shape is determined uniquely. When an operator selects this curving function during paper folding, the system gives the paper a cylindrical shape according to the vertex movement.

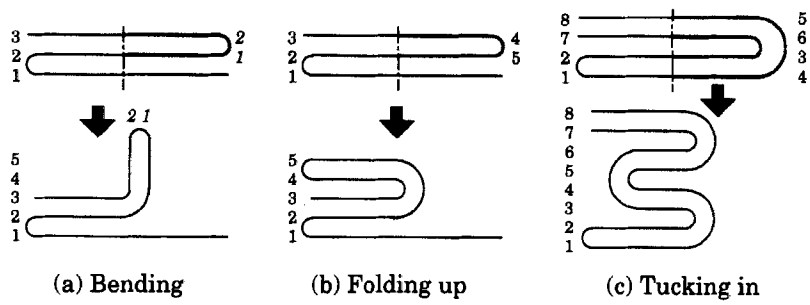


Figure 8. Renewal of the face order. The bold line represents the moving part of the faces

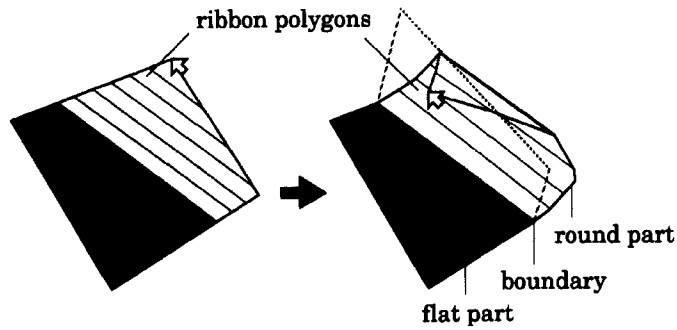


Figure 9. The round part and flat part of a curved face. A set of ribbon polygons represents the round part. The dotted line represents the operation bending *along the boundary noted above*

Round part of the curved face

For the purpose of dividing faces while preserving connectivity, *curving* is treated as a new type of *bending* operation with the crease on the boundary between the flat part and the round part (Figure 9). We approximate the round face with a set of ribbon-like polygons. Ribbon polygons are generated by dividing the face to be curved by lines parallel to the fold-line in *bending*, where the selected vertex is put in the same position (Figure 10). Therefore, the direction of the parallel lines is changed continuously according to the movement of the selected vertex. In the curving operation, the data structure is processed as follows (Figure 11).

The face is divided at the boundary in the curving operation and the associated list is updated. The face-cell corresponding to the round part is given a flag of curve and a sequence of vertices in the ribbon polygons is added. In the same way, the edge-cell constituting a curve is also given the curve flag and the sequence of vertices is also obtained. This process is executed only once for an edge. The round part of the curved face by itself constitutes a face group with the attribute of the curved face, and is not modified by any folding operations afterwards.

In this way, the curving operation is treated as one of basic operations for paper folding. Thus we can cope with the curving function without major changes to the system structure or the data structure.

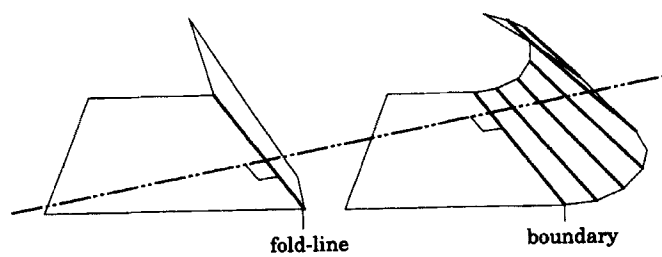


Figure 10. Bending *and* curving *whose selected vertices are put in the same position. The lines dividing the curved face are parallel to the fold-line in* bending

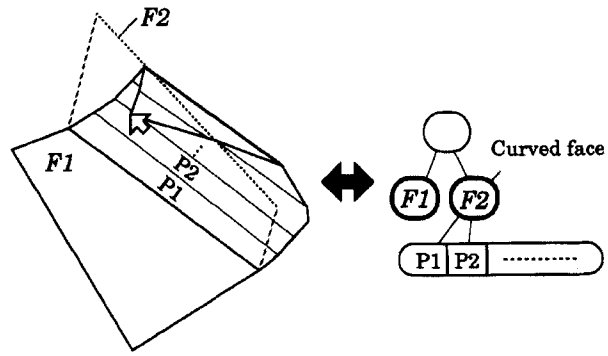


Figure 11. The data structure for curved faces. Curving *is treated as* bending *with the crease on the boundary between the round part and the flat part. Data cells for curved faces and curved edges also include the sequence of vertex co-ordinates of ribbon polygons*

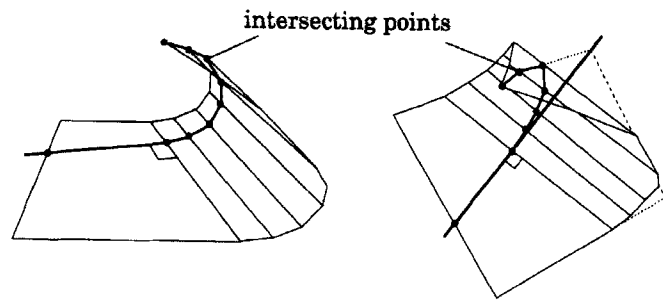


Figure 12. The bend line determining the shape of the round part

Cross-sectional shape

The round part made by ribbon polygons is determined by its cross-sectional shape represented by a curve (a bend line) on the two-dimensional plane (Figure 12).

1. Energy function

It is a common method to derive a functional form describing a surface and to obtain a stable state by minimizing the function. In the stable state, intersecting points along the borders of ribbons and the bend line should be arranged with a constant interval L which is equal to the width of the ribbon polygon (Figure 13). In addition,

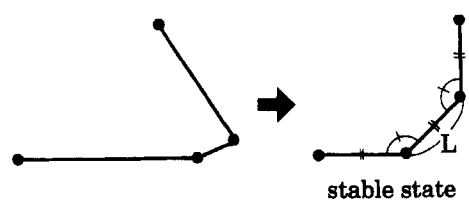


Figure 13. The stable state to minimize the energy function

the angles between adjacent ribbon polygons should be equal. The shape of the bend line is determined so that it minimizes the following energy function:

$$E = \alpha E_d + \beta E_a$$

where E_d is the sum of the squares of the errors in vertex distance from the optimal distance L (Figure 14(a)),

$$E_d = \sum_i (d_i - L)^2$$

and E_a is the sum of the squares of the difference between neighbouring vertex angles (Figure 14(b)),

$$E_a = \sum_i (a_i - a_{i+1})^2$$

To decrease E_d, if the vertex distance is smaller than the optical distance, corresponding vertices are moved to increase the distance. To reduce E_a, vertex positions are shifted in the direction shown in Figure 14(b). The amount of modification is in proportion to the above mentioned errors or differences. Suitable values of the constants α and β are determined experimentally. In implementing the system on a graphics workstation, this modification is repeated during the refreshment of the monitor screen. When the number of variable vertices is 15, for example, this iteration is performed 100 times in 33 ms.

2. *Floor constraint*

If we curve a sheet of paper on a flat table, it has a flat part because of the physical constraint of the table. In the calculation of the cross-sectional shape, this requirement is satisfied by adding the constraint that no vertex should break through the floor. In each iteration, if a vertex goes through the floor, it is returned to the boundary. This constrained minimization procedure results in a balanced shape of the curved face on the floor, and the boundary between the flat and the round parts is determined.

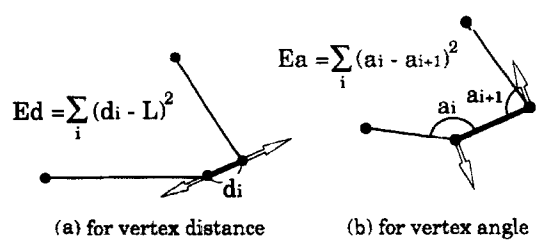


Figure 14. The energy function and modifying vertex positions to decrease it

INTERACTIVITY

The virtual manipulation system for deformable objects such as pieces of paper must operate in real-time. In this section we discuss the real-time interaction and give actual processing times. During a vertex move, a rendering process and a computation process are executed alternately. The cycle time needs to be less than about 100 ms for the movement of an object to appear smooth.

Figure 15(a) shows results in the case of a face being folded in half repeatedly. Computational time consists of renewal time including the correcting time explained earlier. All of those times were measured independently. Dotted lines show the actual time when these processes are simultaneously executed and it is slightly different from

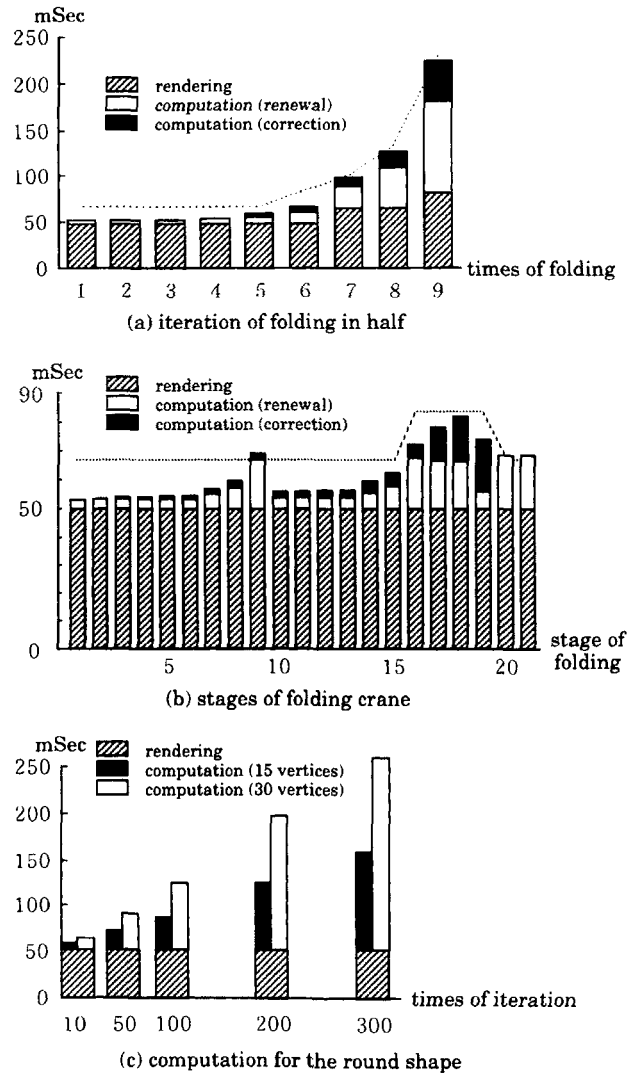


Figure 15. Computational time and rendering time in several cases

the total of each component because the timing of rendering is adjusted to the refreshment of the screen. For the same reason, the rendering time always requires more than 50 ms. The computation time of renewal and correction of faces depends on the number of faces. On the other hand, the rendering time is almost constant through folding processes. The number of faces increases as the paper is folded; it reaches 512 ($=2^9$) after folding nine times, and the processing time rises to 250 ms in this case. In real folding, however, it is impossible to fold so many times because of the thickness of a paper sheet.

Figure 15(b) shows processing time through all stages in folding a crane as a practical example of folding work including various operations. It took only 70 ms on the average and 83 ms at the maximum, because a paper crane has only about eighty faces in the complete form. The interaction is performed almost in real-time for completing this shape.

The curving operation needs additional computation time to obtain the shape of a round face shown in Figure 15(c). This is done once in a cycle regardless of the number of faces moved. This time increases in proportion to the number of vertices representing the cross-sectional round shape and the number of iterations needed to converge the energy function. We set a limit of 15 vertices and 50 iterations in the current system to keep the operation time less than 100 ms.

Our system is currently implemented in about 6000 lines of C program on a super graphics workstation, a Silicon Graphics IRIS Crimson with the Reality Engine. We have made various pieces of origami such as plane, crane, penguin and tray interactively by using the system. Several examples are shown in Plate 2.

CONCLUSIONS

In this paper, we have presented a system manipulating a virtual sheet of paper, or virtual paper folding ('origami') as a fundamental study of software development in virtual reality. This shows the potential of natural man–machine interfaces using real-time computer graphics. For concise representation of a folded paper sheet with flat and curved faces, we employed the binary tree data structure and a physically based model. The round shape of the curved face is derived in real-time based on the simple physical model. Texture mapping and shading allow us to interact with superior quality virtual paper.

Our current folding operations cover only a part of classical origami because they do not include complex folding such as inserting the tip of a face into the slit between other faces or puffing up a paper balloon. Simultaneous manipulation of more than one point is necesary to realize such complex folding. A conic surface generation function is also necessary to extend the curving function. 3D manipulation using only the mouse is not too difficult because the depth is changed independently of the movement in the other dimensions. However, more advanced input devices such as a 3D digitizer may be better for paper folding. The thickness of a sheet is not considered in the system. This has both advantages and disadvantages. We can fold a virtual sheet of paper as many times as we like without taking care of thickness. On the other hand, there are types of fold that use the thickness of a sheet positively in actual origami. We do not employ collision detection because it consumes a lot of processing time, and ignoring collisions does not affect the folding process severely. To develop more advanced sys-

tems, however, we will need further study including simplifying the physical model, devising an efficient scheme of interaction, deriving a faster polygon rendering algorithm and so on. We are planning to develop more realistic manipulation such as cutting paper or tearing paper employing a more sophisticated model of paper manipulation in the next stage.

ACKNOWLEDGEMENTS

We would like to thank Professor Daniel Thalmann and Professor Nadia Magnenat Thalmann for advising us to write this paper. We also thank Professor Teruo Fukumura in Chukyo University and colleagues in Toriwaki Laboratory of Nagoya University for supporting this work. Special thanks to Professor Geoff Wyvill in University of Otago for proof-reading this paper. This research was supported in part by the Grant-in-Aid for Encouragement of Young Scientists.

REFERENCES

1. M. Krueger, *Artificial Reality*, 2nd edn., Addison-Wesley, Reading, Mass., 1990
2. C. Ware, 'Using hand position for virtual object placement', *The Visual Computer*, **6**, (5), 245–253 (1990).
3. H. Iwata, 'Artificial reality with force-feedback: development of desktop virtual space with compact master manipulator', *Computer Graphics*, **24**, (4), 165–170 (1990).
4. T. Galyean and J. Hughes, 'Sculpting: an interactive volumetric modeling technique', *Computer Graphics*, **25**, (4), 267–274 (1991).
5. M. Sato, Y. Hirata and H. Kawarada, 'Space interface device for artificial reality—SPIDAR', *IEICE* (in Jananese), **J74-D-II**, (7), 887–894 (1991).
6. S. Tachi and T. Maeda, 'Tele-existence robot simulator with a sensation of virtual reality', *IEICE* (in Japanese), **J75-D-II**, (2), 179–189 (1992).
7. D. Sturman and D. Zelter, 'A survey of glove-based input', *Computer Graphics and Applications*, **14**, (1), 30–39 (1994).
8. S. Miyazaki, T. Yasuda, S. Yokoi and J. Toriwaki, 'Interactive manipulation of origami in 3D virtual space', IPSJ (in Japanese), **34**, (9), 1994–2001 (1993).
9. T. Agui, M. Takeda and M. Nakajima, 'Conditional key frame animation', *IPSJ* (in Japanese), *CG Technical Report*, 1981, pp. 1–2.
10. T. Uchida and H. Itoh, 'Knowledge representation of origami and its implementation', *IPSJ* (in Japanese), **32**, (12), 1566–1573 (1991).
11. A. Komori, T. Yasuda, S. Yokoi and J. Toriwaki, 'Interactive simulation system for paper folding', *IPSJ* (in Japanese), *Graphics and CAD Technical Report*, 1991, pp. 50–59.
12. A. Konno and S. Itahashi, 'Origami processing by personal computer', *IPSJ* (in Japanese), *National Convention*, 1992, pp. 2–353.
13. Special issue on Origami, 1 and 2, *Symmetry: Culture and Science*, **5**, (1) and (2) (1994).
14. D. Terzopoulos, J. Platt, A. Barr and K. Fleisher, 'Elastically deformable models', *Computer Graphics*, **21**, (4), 205–214 (1987).
15. D. Haumann and R. Parent, 'The behavioral test-bed: obtaining complex behavior from simple rules', *The Visual Computer*, **4**, 332–347 (1988).
16. A. Witkin and W. Welch, 'Fast animation and control of nonrigid structures', *Computer Graphics*, **24**, (4), 243–252 (1990).
17. A. Norton, G. Turk, B. Bacon, J. Gerth and P. Sweeney, 'Animation of fracture by physical modeling', *The Visual Computer*, **7**, 210–219 (1991).
18. M. Carignan, Y. Yang, N. Thalmann and D. Thalmann, 'Dressing animated synthetic actors with complex deformable clothes', *Computer Graphics*, **26**, (2), 99–104 (1992).
19. D. Baraff and A. Witken, 'Dynamic simulation of non-penetrating flexible bodies', *Computer Graphics*, **26**, (2), 303–312 (1992).

20. G. Wyvill and D. McRobie, 'Local and global control of cao en surfaces', *Communicating with Virtual Worlds* (Proc. CG International'93), 1993, pp. 216–227.
21. Y. Kergosien, H. Gotoda and T. Kunii, 'Bending and creasing virtual paper', *IEEE Computer Graphics & Applications,* **14**, (1), 40–48 (1994).