芸術科学会論文誌 Vol.11, No.4, pp.108 – 117

Stroke history management for digital sculpting

Ryota Takeuchi[1)] **Taichi Watanabe**[2)]
Masanori Kakimoto[3)] **Koji Mikami**[2)] **Kunio Kondo**[2)]

1)2)3) School of Media Science, Tokyo University of Technology.

1) s2rita (at) nifty.com
2) {earth, mikami, kondo} (at) media.teu.ac.jp
3) kakimotoms (at) stf.teu.ac.jp

Abstract
Digital sculpting is a modeling method to make a detailed forming shape like sculpture. Recently, digital sculpting tools are becoming prevailing technology on 3DCG contents production. However, most of the sculpting operations are irreversible, because sculpting tools overwrite the information on shape directly. Therefore, the history operation of the undo redo etc. is often achieved by the snap shot of the shape, and the operation is restricted by the limit of the memory capacity. This paper proposes a new digital sculpting system based on the history of stroke input. This system retains the CSG model with an implicit function set. As a result, each operation is able to be defined as reversible command in our method. Moreover, in our method, every command history item is made re-targetable by maintaining stroke-trails and camera information. It allows users to recover any past state or to cancel a command. Our experimental results indicate the effectiveness of the proposed system.

1 Introduction

Digital sculpting [1][2] is a modeling method to make a detailed forming shape like sculpture. Recently, digital sculpting tools are becoming prevailing technology. Digital sculpting is formed by operating on the displacement of the voxel and the surface of the polygon while traditional modeling tools operate on polygons or parametric surfaces. Therefore, CG artist's sensibility is reflected easily and intuitively on the shape, and creating a lot of cases where digital sculpting is adopted in 3DCG contents and the game work in recent years. The clay metaphor is one of the main characteristics of digital sculpting. By carving and dishing up the virtual clay, the users can operate on the shape of the model surfaces. In Z-Brush [3] of the Pixologic Co. that is a typical digital sculpting, shapes are edited by giving displacement and subdividing polygons according to the input from the pen tablet. When the high polygon model is generated with digital sculpting, the displacement map to the low polygon model [4] is made and used for the target contents. With this method, however, the user cannot form a shape when its topology greatly changes. On the other hand, voxel based digital sculpting techniques [5][6][7] are proposed. They enabled the users to carry out forming and deforming operations without any topological constraints. The 3D-Coat [8] of the PILGWAY Co. is offering a voxel-based modeling environment in the practical application. However, these operations simply overwrite the shape data being edited and thus the previous data are lost. Giving invertibility can do nothing but take the snap shot from this in the editing in digital sculpting at an interval time constant to be difficult, and to achieve the undo redo, and there is a limit in the operation record that can be maintained.

On the other hand, there are history-based methods [9][10] in conventional modeling systems. Their approach is to model the shape by accumulating the user commands, each of that describes the shape distortion or carving with primitive shape or parametric surface description. The advantage of history-based modeler is that the user can freely go back and forth in the past forming process. It is easy to recycle a part of the editing process. A history-based modeler using Euler operators [11] can define the reverse operation of each shape distortion or shape forming process. Since Euler operations cannot treat topological changes, they cannot be directly applied to digital sculpting. We paid attention to the fact that when the digital sculpting user forms a solid shape each input stroke is associated with a projection from the plane defined by the camera position and the line of sight. If we could maintain the camera information and the strokes on the screen, it would be possible to reproduce the forming operations. Based on this idea, we would realize unrestricted undo and redo operations, maintaining all operation records from the beginning of the shape modeling process. Moreover, in our system the stroke inputs should be applicable to the shape that is different from the original target shape. We should be able to skip a specific process or to branch the history from a certain point as well as to trace the history in the same straight line, i.e., tree structured history information is preferred. As a result, it should become possible to use the history of the stroke more freely.

To achieve the above goal, our system uses Constructive Solid Geometry (CSG) [12][13], and it models objects with a set of implicit functions. As a result, each forming operation is expressible by the Boolean operation [14][15] of implicit functions, and thus it is possible to define it as a reversible command. The number of implicit functions will increase in this approach whenever the stroke is given as input. We solved this problem by clustering realm of implicit functions, and achieved an interactive response speed. We have implemented an interface for tracing the modeling operation along a tree structure. In addition to conventional undo/redo operations, our interface can control, on arbitrary commands, such operations as moving, copying, pasting, branching, and cancelling. When the shape at the time of the command input and re-application ahead had changed, our system deals with the problem by introducing a treatment that we call retargeting. By projecting the stroke from the input plane onto the other plane, we could make the best use of the operator's intention of the input stroke. We have conducted a user experiment and the results indicate the superior effectiveness of our proposed system.

2 Digital sculpting by implicit functions

This section explains the method for achieving of digital sculpting by a set of implicit functions. First of all, the data structure of the working shape is described. Moreover, we describe the technique for maintaining the sculpting operation as a command and the "re-targeting" processing.

2.1 Data structure for shape representation

In our modeling system, the working shape is represented by the set operation of implicit function-based primitives. A primitive consists of the combination of the following two elements.

- A surface side judging function.
- Attribute(positive or negative).

The region P_i where each primitive influences is defined as follows.

$$P_i = \{\mathbf{P} \in \mathbf{R}^3 \mid f_i(\mathbf{P}) \leq 0\}, \tag{1}$$

Where, f_i is an arbitrary implicit-function for judgment of the side of the surface shape. For example, if the implicit surface is a sphere with radius r, the implicit function f becomes the following equation $f_i(\mathbf{P}) = |\mathbf{P}|^2 - r^2$. If a shape is a closed volume to arbitrary three dimension coordinates, the shape can be each considered as a primitive. Each primitive has an attribute of positive or negative. A positive primitive is a region that should be merged as a part of the working shape, and a negative primitive is a region where the shape should be carved off. This corresponds, respectively, to an addition and subtraction of the Boolean operation in CSG model. These primitives are used as dishing up and carving operations of the shape in digital sculpting.

The set operation that expresses the entire shape is provided as follows by the use of the primitives. A shape model S_n that composes of as many primitives as n is defined by the following recurring formula.

$$\begin{cases} S_1 = P_1 \\ S_i = S_{i-1} \bigoplus P_i \quad (i = 2 \ldots n), \end{cases} \tag{2}$$

Where the binary operator $\bigoplus$ is a conditional set operator that shows

$$A \bigoplus B = \begin{cases} A \cup B & (B \text{ is positive.}) \\ A - B = A \cap \overline{B} & (B \text{ is negative.}), \end{cases} \tag{3}$$

with regard to given two sets A and B.

In order to realize intuitive user operations, we should make the surface of the working shape visible in an interactive frame rate. In general, however, the process to extract an isosurface from a set of implicit surfaces requires additional computation time. Thus we use a point-base surface expressed by a point set [16][17][18] separately generated for the purpose of visualizing the surface shape. As a result, our system can skip the computation of the connection relationship among the vertices and thus it can present quick and intuitive editing result in response to the user's input operation. When a new primitive is added, its point-set V_i representing the shaped surface is obtained by using the following expression.

$$V_i = \{\mathbf{V} \in \mathbf{R}^3 \mid f_i(\mathbf{V}) = 0\} \tag{4}$$

The system evaluates each vertex of point-set V_i according to the function of the whole shape S_n, and decides whether to assume a part of whole shape or to annul it. When the input primitive is positive, vertex outside of whole shape is left. When the input primitive is negative, vertex in the whole shape is left. Otherwise, the vertex is discarded.

When the number of primitives composing the whole shape S_n increases, it costs more to evaluate the side judgment function $S_n(\mathbf{P})$. To solve this problem, our system maintains a list of implicit functions for each clustered area, and evaluates the functions from the tail of the list. This tail-first evaluation method lightens up the evaluation cost increases for the implicit functions because the whole shape S_n is defined by the recurring formula described in Equation (2). To determine whether certain coordinates reside inside or outside the shape S_n, the system should evaluate the primitive functions in the list in the forward order. When it finds the last primitive that contains the coordinates, the side is determined by the attribute of the primitive. The determination could be made quicker by checking the list in the reverse order. Given arbitrary three dimension coordinates $\mathbf{P}$, our system actually carries out the side judgment from the tail

of the primitive list. When it first encounters the primitive that contains P, it checks the sign of the attribute of the primitive. If the attribute is positive, the point P is regarded to be inside the whole shape, and vice versa. The system regards P as to be outside of the whole shape if P was not contained in any primitives in the list that the relevant area maintains.

Figures 1 and 2 demonstrate the changes of the point set and the surface of the whole shape in case a new primitive is added. Figure 1 shows a dishing up operation example and Figure 2 shows a carving operation example.

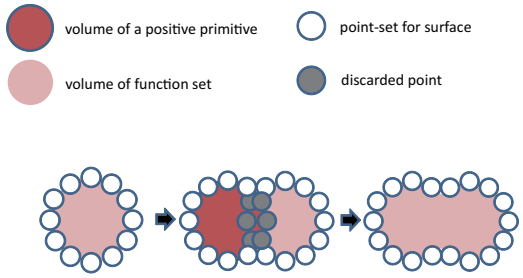


Figure 1: The state of data structure on dishing up.

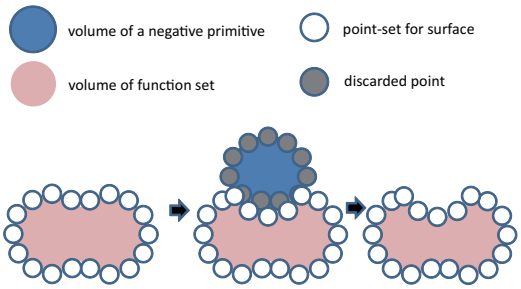


Figure 2: The state of data structure on carving.

2.2 Input stroke history management and the retargeting process

Since the working shape is expressed by a list of implicit functions, a sculpting operation that is either carving or dishing up can be achieved by adding a new implicit function. Our system assumes that a general pointing device is used for the input in digital sculpting. A stroke is defined by a trajectory of coordinates that starts when the mouse button or the tablet pen is pressed and ends when they are released. As a result, the vita information for one stroke can be recorded by maintaining indices of the implicit functions added during the corresponding stroke dragging from among the implicit function lists that compose the whole shape. Since the whole shape is represented as a series of set operations of the implicit function list, undo can be achieved by deletion from the list and redo by restoring an item to the list.

Undo and redo as usual history operations are feasible by either adding to or deleting from the implicit function list as described previously. It would make the digital sculpting very convenient for the users if the system allows them to cancel or to re-apply an arbitrary operation. To achieve this, we propose a method for applying a stroke to a situation different from that when the stroke input was made. Our underlying idea in this paper is that we could (can) obtain a result that values the user intention infused in the stroke that he/she had drawn if we process a changed shape assuming the same stroke was reapplied through an appropriate screen. We call this process retargeting.

In order to achieve the retargeting process, our system maintains the information of the camera position at the time of a stroke input and adds it to the history item for the stroke. When a stroke was given, the system records the coordinates of the stroke in the screen coordinate system, the camera position, the camera vector, and the camera up-vector. Using these pieces of information, our system processes re-targeting of a given stroke by the following flow.

1. Replace the current viewpoint with the viewpoint when the user made the stroke input.
2. Draw the working shape enabling the depth buffer writing.
3. Clear the output stroke.
4. For each stroke point:
   (a) Cast a ray from the viewpoint toward the screen-space stroke point.
   (b) Find the intersection between the ray and the surface of the working shape by reading the depth buffer value at the screen point.

(c) Append to the output stroke the intersection point in both object and screen spaces.

5. Record the camera information and add it to the output stroke.

Figure 3 illustrates the process done when the stroke on the screen is given as input. The purple line in Figure 4 indicates the trajectory of the stroke actually input on the screen, and Figure 5 shows the appearance applied to shape.

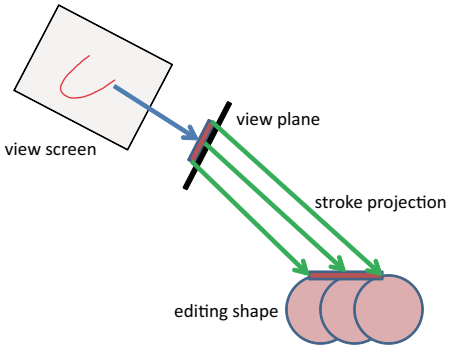


Figure 3: The projection of stroke.

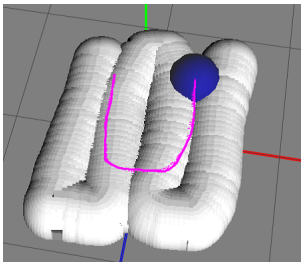


Figure 4: An input stroke example.

Figure 6 presents the process when the stroke input in the above-mentioned Figure 3 is applied with shape changed by a history operation. Figure 7 demonstrates the appearance in that the stroke input in Figure 4 is overlapped with shape that the state has changed. Figure 8 depicts the appearance applied to a shape by re-targeting.

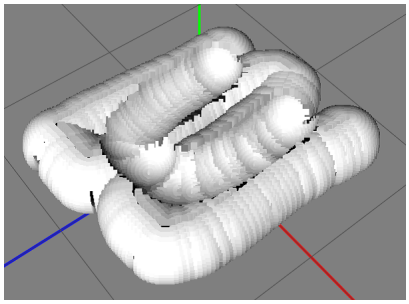


Figure 5: The result of the stroke in Figure 4.

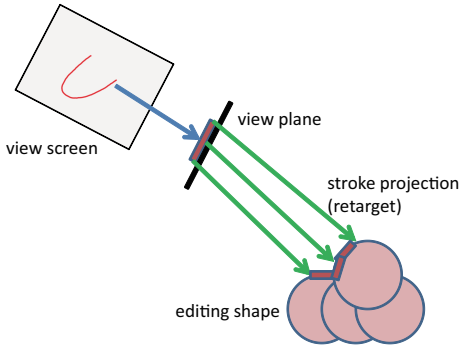


Figure 6: Re-targeting of stroke.

3 Implementation of the history-tree interface for digital sculpting

We implemented our proposed method described in the previous section as a simple digital sculpting system. Operations related to the history management can be carried out through a visual interface. This is called a history tree interface. Figure 9 shows visualized images of typical tree structure patterns used in the interface.

The examples in Figure 9 are the results when the operation is accumulated in the same way as a common historical management. A white icon indicates the initial state, and a red or blue icon shows dishing up or carving shape, respectively. When the history is rewound by an undo operation, the cursor drawn as a yellow frame moves one step back to the left, and when the user cancels an operation, it is displayed as a translucent square. The record of the history

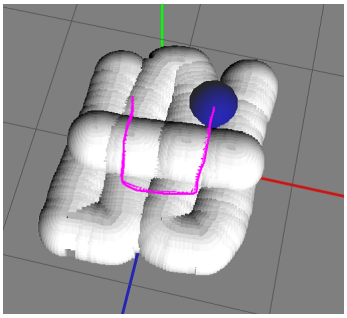


Figure 7: Overlapping of the previous stroke.

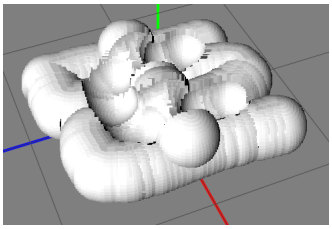


Figure 8: The result of re-targeting.

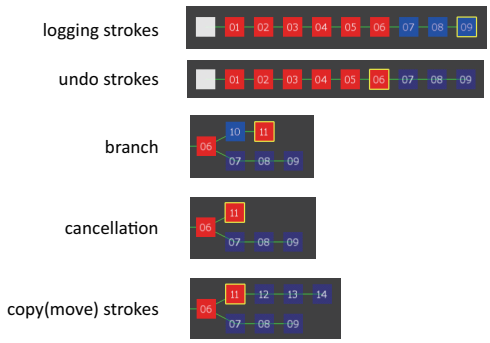


Figure 9: The view of history-tree.

is branched when a new operation is done from the state rewound by undo and the instructional information is accumulated. When some operations in the middle of the history are not needed any more, the user can selectively cancel appropriate operations. Operations recorded after the cancellation is applied again by the re-targeting processing. In order to apply the operation done on another divergence to the current state, the user can copy or move the tree by selecting and dragging the icon. An operation that is copied or moved becomes a re-targetable operation, and can be applied again in a redo operation. The user can control the functions achieved in our method by the above-mentioned interface.

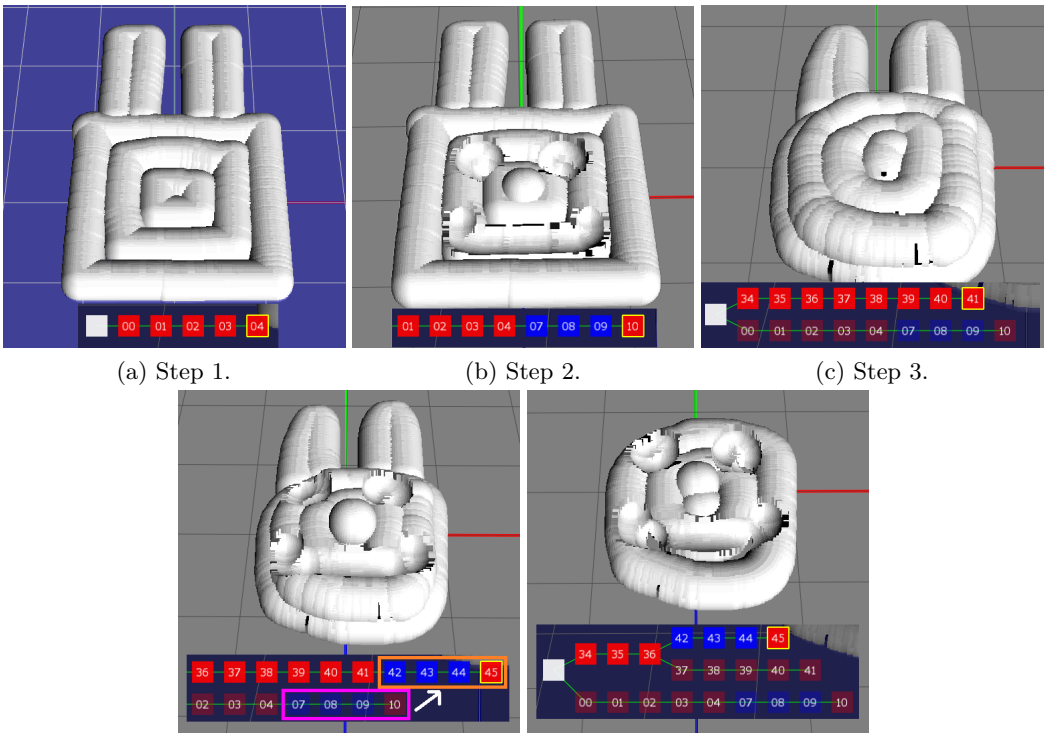
4 Verification

The program is written in C++, and FK Toolkit system [19] based on OpenGL was used as a graphics system. The Cg language of the NVIDIA Co. was used for a part of processing.

We verified our system by checking whether the user can achieve the aimed operations using the functions described in the previous sections. The re-targeting processing was verified in two scenarios. One is canceling an operation in the middle of a tree, and the other is applying a copied operation to another tree. We have achieved the expected sculpting operations along the strokes in both cases. The following series of figures demonstrate the process of sculpting an animal face model. Figure 10(a) presents a foundation shape imitating a rabbit-like animal head. Figure 10(b) is a rendering result in that the eyes and the mouth are carved off and the nose is dished up for the shape shown in Figure 10(a). Figure 10(c) indicates that the user has undone all the past operations and has made a new series of dish-up operations. Note that the history diverged into the upper branch (34-41) and the past operations (00-10) were memorized. Figure 10(d) demonstrates re-targeting the past carving and dishing up operations that have been undone in Figure 10(c). Please note that the stroke is applied to the shape with a changed state appropriately by re-targeting. Figure 10(e) presents a result of automatic re-targeting the carving operations in the case that the user canceled a series of preceding operations in the middle of the history. It is understood that the carving operations are reflected onto the shape even when the application order is changed.

In the current implementation, the camera parameters when re-targeting was processed were assumed to be the fixed for the remainder of the input. When shape changes greatly including its layout, the camera parameter should be changeable at any time to achieve more flexible re-targeting processing.

We also confirmed an increase in the processing load according to an increase in the number of implicit functions. We confirmed that the performance was 60fps for drawing and processing a



(a) Step 1. (b) Step 2. (c) Step 3.

(d) Step 4. (e) Step 5.

Figure 10: A exmaple of modeling sequence.

shape with approximately 280K points after entering 3,426 implicit functions with 108 strokes. We could say that our system runs sufficiently fast in a practical and realistic situation. Table 1 describes our experimental environment.

Table 1: The enviroment of testing computer.

CPU	AMD Phenom(tm) II X6 1090T 3.2GHz
RAM	4GB
GPU	NVIDIA GeForce GTX 470

In the current implementation, we used a depth buffer value in projecting to the working shape destination while in re-targeting. When a redo is operated, this requires an image draw execution and thus some time lags occur until the result is reflected in the operation. We forecast its improvement by replacing the depth buffer operation with some geometry calculations.

In our method, we achieved the historical management using smaller amount of memory than snapshot methods by maintaining all input stroke data. We conducted the comparative verification of memory capacity usage. Figure 11 presents the shape made with this system for this verification. To make the final shape, we took eight strokes and 556 implicit functions. Table 2 and Figure 12 compare the amounts of memory consumption needed for the edit histories of the shape between the method of recording the entire input strokes and the method of taking the entire snapshot of the shape.

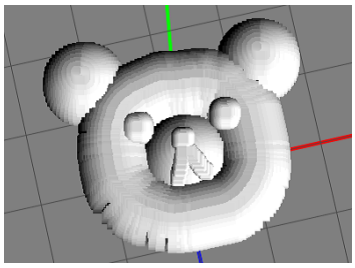


Figure 11: A test modeling shape.

As shown in the table, the stroke-based method significantly reduces the memory usage. While the difference of the memory re-

Table 2: A comparison of memory usages

step(s)	Stroke-based method(bytes)	Snapshot-based method(bytes)
1	5,752	21,792
2	6,068	43,632
3	6,384	65,520
4	6,700	87,456
5	7,016	109,440
6	7,332	131,472
7	8,788	158,112
8	9,104	184,800

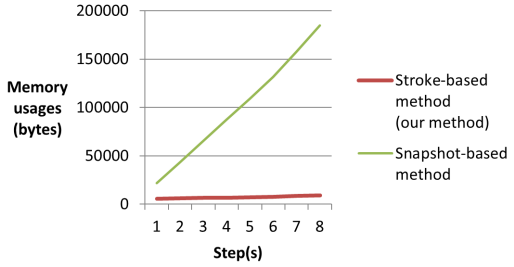


Figure 12: A comparison graph of memory usages.

quirement between the two methods was modest in the early stage of the editing process, we confirmed that our stroke-based method can manage the history with relatively less memory capacity compared to the snapshot method as the edit operation proceeds and the shape becomes more complex. Some practical applications record a limited set of snapshot history items to save the memory usage. In such a case, it becomes impossible to realize a copy and paste operation of the edit history or a branch of history. Therefore, we can say that the stroke-based historical management system is also more useful in respect of recycling the input data.

5 Conclusion

This paper proposed a digital sculpting system based on the historical management with the input stroke data, and the experimental implementation has shown the effectiveness of the concept. Our technique enabled unlimited undo-redo operations those are restricted in conventional modelers. Moreover, our system provides the cancellation of the operation and the diverging of history. As a result, the efficiency of editing in digital sculpting has been improved with our method. The system can reuse the user input strokes with the re-targeting processing for sculpting an alternative shape model that is different from the original model for the strokes.

Future work includes increasing the variations of primitive shapes. Currently the sphere shape is the only primitive available for our system. Usages of arbitrary convex hull or parametric surfaces as a primitive are preferred. Moreover, this system would also endure editing operations of a more complex, detailed shape if we employ clustering processing, that is effective for the point-set that composes the surface figure with the realm of implicit function.

References

[1] Ronald N. Perry and Sarah F. Frisken. Kizamu: a system for sculpting digital characters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '01, pp. 47–56, 2001.

[2] Roland Hess. *The Essential Blender: Guide to 3D Creation with the Open Source Suite Blender*. No Starch Press, San Francisco, CA, USA, 2007.

[3] Eric Keller. *Introducing ZBrush*. Wiley, Hoboken, NJ, 2008.

[4] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pp. 115–122, 1998.

[5] J. Andreas Bærentzen. Volume sculpting: Intuitive, interactive 3D shape modelling. In *IMM VIsionday, 2001*, May 2001.

[6] Ralf Böonning and Heinrich Müuller. Interactive sculpturing and visualization of unbounded voxel volumes. In *Proceedings of the seventh ACM symposium on Solid modeling and applications*, SMA '02, pp. 212–219, 2002.

[7] Anthony Prior. "on-the-fly" voxelization for 6 degrees-of-freedom haptic virtual sculpting. In *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications*, VRCIA '06, pp. 263–270, 2006.

[8] PILGWAY Co. 3d-coat. http://3d-coat.com/, accessed August 10, 2012.

[9] P.R. Wilson. Euler formulas and geometric modeling. *IEEE Computer Graphics and Applications*, Vol. 5, No. 8, pp. 24–36, August 1985.

[10] M. Mantyla and R. Sulonen. Gwb: A solid modeler with euler operators. *IEEE Computer Graphics and Applications*, Vol. 2, No. 7, pp. 17–31, 1982.

[11] H. Toriya, T. Satoh, K. Ueda, and H. Chiyokura. Undo and redo operations for solid modeling. *IEEE Computer Graphics and Applications*, Vol. 6, No. 4, pp. 35–42, April 1986.

[12] Aristides G. Requicha. Representations for rigid solids: Theory, methods, and systems. *ACM Comput. Surv.*, Vol. 12, No. 4, pp. 437–464, December 1980.

[13] A.A.G. Requicha and H.B. Voelcker. Boolean operations in solid modeling: Boundary evaluation and merging algorithms. *Proceedings of the IEEE*, Vol. 73, No. 1, pp. 30–44, January 1985.

[14] Bart Adams and Philip Dutré. Interactive boolean operations on surfel-bounded solids. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pp. 651–656, 2003.

[15] Mark Pauly, Richard Keiser, Leif P. Kobbelt, and Markus Gross. Shape modeling with point-sampled geometry. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pp. 641–650, 2003.

[16] Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar, and Markus Gross. Surfels: surface elements as rendering primitives. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pp. 335–342, 2000.

[17] Matthias Zwicker, Mark Pauly, Oliver Knoll, and Markus Gross. Pointshop 3d: an interactive system for point-based surface editing. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pp. 322–329, 2002.

[18] Ryota Takeuchi, Taichi Watanabe, and Soji Yamakawa. Sketch-based solid prototype modeling system with dual data structure of point-set surfaces and voxels. *International Journal of CAD/CAM*, Vol. 11, No. 1, 2011.

[19] Fine Kernel Project. Fine kernel toolkit system. http://fktoolkit.sourceforge.jp/, accessed August 10, 2012.

Ryota Takeuchi

Graduated in 2004 from School of Media Sceince, Tokyo University of Technology, and completed his master's degree in 2006 at Graduate School of Bionics,Computer and Media Sciences, Tokyo University of Technology. He research interests are computer graphics, CAD, games.

Taichi Watanabe

Taichi WATANABE is a lecturer of the School of Media Science, Tokyo University of Technology from 1999. Graduated in 1994 from Faculty of Environmental Information, Keio University, and completed his master's degree in 1996 at Graduate School of Media and Governance, Keio University. He research interests are computer graphics, CAD, games.

Masanori Kakimoto
Dr. Info. Sci. & Tech.
(University of Tokyo, 2005)

Masanori Kakimoto received his B.E. and Dr. from the University of Tokyo in 1982 and 2005, respectively. From 1982 to 1993 he was with Fujitsu Laboratories Ltd. He was a visiting researcher in the Engineering Computer Graphics Lab. at Brigham Young University from 1989 to 1990. From 1995 to 2011 he worked at SGI Japan Ltd. and from 2011 to 2012 at Silicon Studio Corp., where he developed computer graphics applications. From April 2012 he is a professor at the School of Media Science, Tokyo University of Technology. His research interest includes visual computing and image media, and their applications to industries. Kakimoto is a member of ACM SIGGRAPH, IPSJ, IIEEJ and the Japanese Society of Ophthalmological Optics. He is currently Chair of SIG on Computer Graphics and CAD of IPSJ.

Koji Mikami
Ph.D.(Keio University, 2008)

Graduated in 1995 from Faculty of Environmental Information, Keio University. He worked Nissho Iwai Corporation and MK Company as a producer. In 1998, together with Mitsuru Kaneko, he established “Creative Lab” at the Katayanagi Advanced Research Lab of Tokyo University of Technology (TUT), where research on next generation visual content production and the digitization of the animation production process is conducted. Currently, he is an associate professor at the School of Media Science, TUT.

His research interests are computer graphics, animation, game. He received the IPSJ Education Award in 2012. He is Director of The Society for Art and Sceince, Advisory bord of CEDEC.

Kunio Kondo
Dr.Eng.(University of Tokyo 1988)

Kunio KONDO is a professor,Vice Dean in the School of Media Science, Tokyo University of Technology. He received his Dr.Eng from the University of Tokyo in 1988 and Bachelor from Nagoya Institute of Technology in 1978. He was Associate professor of Department of Information and Computer Sciences, Saitama University from 1989 to 2007, Lecturer of Tokyo Polytechnic University from 1988 to 1989 and Technical staff of Nagoya Unieversity from 1973 to 1988, a part-time lecturer of Tokyo University from 1991 to 2007, Aichi Prefectural University of Fine Arts and Music from 1989 to 1999, Kyushu Institute of Design from 2002 to 2010.

His research interests are computer graphics, animation, game, and interactive modelling. He received the IPSJ the Anniversary Best Paper Award in 1985, JSGS Research Award in 1985, and JSGS Best Paper Award in 2011.

He is President Elect-2012 of The Institure og Image Electronics engineers of Japan, and He was former President of The Society for Art and Sceince, former Vice President of Japan Society of Graphic Science, and Chair of SIG on Computer Graphics and CAD of Information Processing Society of Japan, Board member of Asia Digital Art and Design Association.