VelvetPath – Layout Design System with Sketch and Paint Manipulations

Hiroaki Tobita

Interaction Laboratory,
Sony Computer Science Laboratories, Inc
tobita@csl.sony.co.jp

Abstract

We describe the VelvetPath system, a system that allows users to design layouts for visualized information by using sketch and paint manipulations. Many systems enable users to visually search and browse through information by treating the data as visualized nodes (e.g., related images and simple figures). While these layouts are pre-defined and useful for the applications considered by the designers, users cannot freely change or redesign the layouts. In contrast, users of the VelvetPath system can freely design and change layouts by simply drawing strokes. Since the information layouts are defined automatically according to the user's strokes, the user can create layouts based on his/her own needs and retrieve information using these layouts. Moreover, because all the manipulations in the system are based on simple sketching and painting interactions, the system is easy to use. These simple manipulations are useful not only for information retrieval, but also for a wide variety of applications such as 2D and 3D content creation, preparing slide presentations, and communication through a computer. In this paper, we describe our VelvetPath system and how it can be effectively applied.

Keywords:

Layout Design, Sketch and Paint Interactions, Information Retrieval, Contents Creation, Presentation.

1. Introduction

As computer hardware and software improves, it has become possible to more quickly express a wider variety of information (e.g., images, movies, documents, and web pages). In addition, many types of computer (e.g., augmented reality (AR) systems and personal digital assistants (PDAs)) have been developed to allow users to naturally interact with the information space. Although there are many opportunities to use particular information layouts to handle data sets (e.g., for information retrieval, presentation, creation, and communication), most users still use simple linear and rectangular layouts that are supported by the computer's operating system (OS) or a pre-defined layout produced by software. As data and computer types continue to proliferate, though, there will be a growing demand for effective layout techniques that enable users to interact with their computers more effectively.

While considerable research has been done to support the use of information visualization [1] for retrieving information, existing visualization systems are not designed to treat all data sets in a common way. Actually, by applying visualization techniques such as zooming and scaling [3,4,5], specialized systems can be designed to enable efficient visualization of huge amounts of data. However, such systems are not effective for simple visualization of moderate amounts of data. Particularly, in the case of 3D layouts [6,7,8], the navigation methods become as important as the layouts as the amount of data increases. Developing effective navigation methods for pre-designed layouts is more difficult than developing an effective layout. Moreover, although some systems include different layouts, these layouts are still designed based on the system designers' intention.

On the other hand, the user could use simple rectangular and linear layouts to treat sets of data. While such layouts can be used for many types of data, they are too simple to control the scale and position of each piece of information or to reflect all user requirements. Consequently, although such systems are easy to use, some limitations exist. For example, the user cannot control information parameters such as size and position depending on the user's situation and focus, and cannot add relationships between information. As a result, there is no layout system that both supports casual use and reflects the user's full range of needs.

Figure 1: *Example of using the VelvetPath system. Visualized data is automatically displayed according to the user's drawings.*

Figure 2: *VelvetPath system overview. The system has a work area and simple GUIs. Three types of pen attribute are available for drawing different types of stroke.*

Therefore, we developed the VelvetPath system not with the aim of providing a pre-made layout that is capable of meeting the needs of every user, but with the aim of allowing users to design and create their own information layouts naturally. Based on sketch and paint interface techniques, the system supports both simple rectangular and linear layouts and complex layouts in 2D and 3D spaces. Visualized information like image or document data automatically appears along the strokes drawn by the user. The user can also create 3D layouts by adding a shadow stroke in a 2D layout, as the shadow stroke is automatically defined as indicating depth in a 3D layout. Also, by adjusting the pen width and paint area, he or she can control the information size and relation rate between information. In this way, the user can control the focus area of a layout and then see information in detail by using the large size display.

Although the system allows the user to freely create and design layouts, the design process may still be difficult. However, because the user can create layouts according to his/her needs by applying sketch and paint techniques without having to engage in parameter-setting through complex graphic user interfaces (GUIs), most design difficulties can be avoided. Even a simple stroke can become a kind of layout. Because the user can also create a new layout by adding a stroke to a previous layout or one of the templates included in the system, the user can redesign and improve the layout according to his/her needs.

In addition, these design interactions are created through simple interactions and pen attributes, so these manipulations can be combined with traditional design systems and other tools. For example, in the case of 2D and 3D creations, a user can define many data and select image data for texture mapping by simply drawing a stroke and painting. The system has great potential for collaboration with other computers and software, so we intend it to be effective not only for information retrieval, but also for a wide variety of applications such as creation, presentation, and communication applications.

In this paper, we describe the VelvetPath system and how it enables users to handle information efficiently.

2. Sketch and paint interactions

A primary characteristic of the VelvetPath system is that it allows users to apply both sketch and paint manipulations to the design layout. Since the system uses the pen width and paint area to positively control the information parameters, we will describe their roles before describing the user interface.

2.1. Pen width

While only the pen trace is processed to obtain the information used to create a model or scene in traditional sketch systems [13,14,15,16], our system also uses the stroke width to control information parameters such as the visualized information size and relation rate. The user can therefore directly control these parameters by adjusting the pen width.

2.2. Paint area

The paint area is also related to information parameters such as size and position. Since a screen image is calculated through image recognition, the system recognizes the area as one big point with the width and height of the painted area. Thus, the user can also control the information size and position by how the paint area draws.

3. User interface

The VelvetPath system provides an information environment that enables the user to create original layouts to meet his/her needs and to browse the many types of data included in the original data set. Figure 2 shows the user interface of the VelvetPath system. The buttons at the bottom of the figure allow the user to choose pen attributes and draw three types of stroke (normal, shadow, and relation).

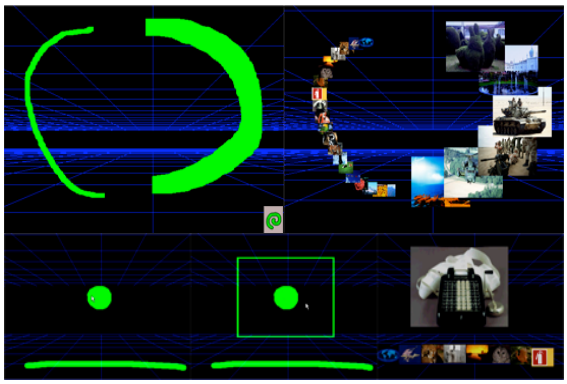

Figure 3: *Normal Stroke. The user can control the information size by selecting the appropriate pen width (top) and adjusting the paint area (bottom). The preview rectangle shows how the information size is displayed within a painted area (bottom).*

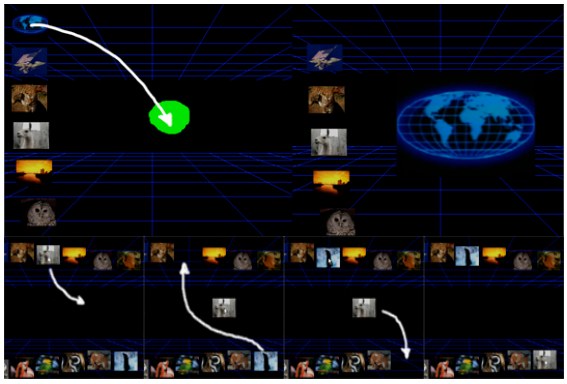

Figure 4: *Click and drag manipulations. The user can change the visualized information size by moving information onto a painted area (top) and exchanging the positions (bottom).*

The first and second buttons activate the *normal* and *shadow pens*, respectively, which are used to create a layout. The user can create a 2D layout by drawing a normal stroke and a 3D layout by adding a shadow stroke to the 2D layout. The third button is for the *relation pen*, which allows the user to add relationships between different pieces of information. The stroke color depends on the button color, so the user can use stroke colors like paint colors. Moreover, the user can control the pen width by clicking on the pen GUI area such that the pen width and GUI become thinner if the user clicks the left button, and the pen width and GUI become bolder if the user clicks the right button in either of the pen GUIs.

3.1. Normal stroke

The user can create a 2D layout by using the normal pen (displayed as a green stroke). Since the information nodes and images are automatically displayed along the user's strokes (Fig. 1), the user can create a layout somewhere in 2D or 3D space and draw strokes on pre-defined objects. In addition, as the pen width is directly related to the image size, larger images appear if the user draws a bold stroke, while smaller images appear if the user draws a thin stroke (Fig. 3 (top)). A painted area is recognized as one big point, and the size of information to be displayed on the painted area can be adjusted with a preview rectangle (Fig. 3 (bottom)). The pen width can therefore perfectly reflect the information size.

The user can control the information size by clicking the information and dragging it onto a painted area (Fig. 4 (top)). The user can also change the data position by clicking data and dragging it to the desired position (Fig. 4 (bottom)), and add a relation between separate groups of information by drawing relation strokes. These manipulations are contained within a simple text file, so the user can customize both the layout and order. First, as the system loads files from a folder, the order of the visualized information depends on default file information such as the name and time information of the files.

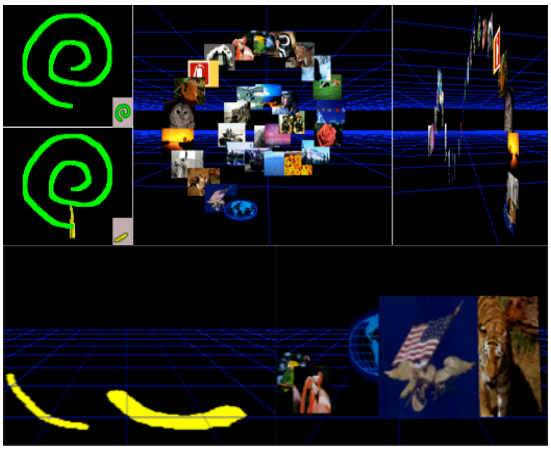

Figure 5: *Shadow stroke. The user can create a 3D layout by drawing a normal stroke (green) and adding a shadow stroke (yellow) (top). It is also possible to create a layout by using only shadow strokes (bottom).*

3.2. Shadow stroke

By adding a shadow stroke (displayed as a yellow stroke) with the shadow pen, the user can create 3D layouts. Basically, the 3D layout is created by combining normal and shadow strokes. If the user draws a shadow stroke under a normal stroke, the system translates the shadow into a depth parameter. Figure 5 (top) shows the creation of a time-oriented layout [9] by using the depth parameter as a time parameter. By simply drawing shadow strokes, the user can cause an information node to appear along the strokes on the background (Fig. 5 (bottom)).

Figure 6: *Relation Stroke. The user can create a layered layout by drawing relation strokes (left). Multiple nodes can be connected to one node at the same time (right).*

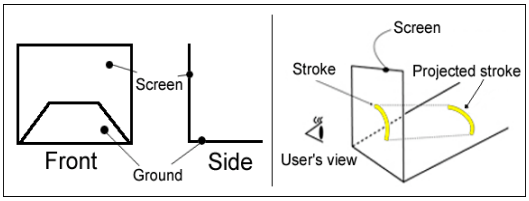

Figure 7: *Work area. The work area is divided into two areas: screen and ground (left). The shadow stroke is projected onto the ground area automatically (right).*

3.3. Relation stroke

The user can add relationships to connect focused nodes by drawing a relation stroke from one node to another. In the same way as in a layout creation, the pen width directly corresponds to a relationship value, so a user can make and control relationships between nodes (Fig. 6 (top)). In addition, by drawing a closed curve, multiple nodes can be connected to one node at the same time (Fig. 6 (bottom)). This manipulation is especially useful for grouping new data, such as digital photographs and user creations like painted images or 3D models.

4. Implementation

The work area consists of two drawing canvases: screen and ground canvases (Fig. 7 (left)). Normal and Relation strokes are drawn on the screen canvas. Shadow strokes are drawn on the ground canvas. Since the shadow stroke is projected onto the 3D scene from 2D strokes, the user can draw it as if creating a 2D drawing (Fig. 7 (right)).

When the user draws normal strokes (Fig. 8 (1)), the strokes are stored as both mouse trace data and labeling data calculated by the system (Fig. 8 (2,3)). Through template matching to the labeling data, the system recognizes what shape was drawn (for example, a line, a circle, a rectangle, or a triangle), so that a painted circle can be recognized as one big point (Fig. 8 (4)). As the system holds information regarding the painted area, it is possible to perfectly reflect the features of the painted area such as the shape and size. Next, the trace data is recalculated so that the visual information will be displayed consistently, i.e., at equal intervals along the trace (Fig. 8 (5)).

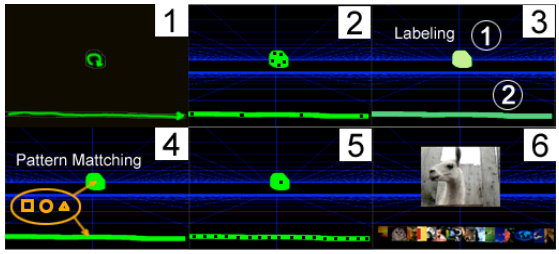

Figure 8: *Calculation of the drawing area. The painted area is recognized as one point through labeling and pattern matching of the screen image (3, 4). Next, the system recalculates the mouse trace data depending on the information size (5, 6).*

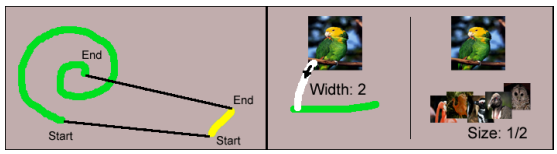

Figure 9: *Technical layout. The user can create 3D layouts by continuously drawing normal and shadow strokes (left), and he/she can add relationships between information and create a layered layout by drawing a relation stroke (right).*

To create 3D layouts, the user first draws a 2D stroke, and then draws a shadow stroke. The start and end points of the 2D layout (based on the normal stroke) are connected with the start and end points of the shadow stroke. The 2D stroke then deforms the 3D curve relative to the length of the depth stroke (Fig. 9 (left)). In the case of drawing only shadow strokes, the system calculates in the same way as for normal strokes (Fig. 8). After setting up the layout, the user can shift images from the starting point to the end point of the recalculated data. As the amount of expressed data depends on the stroke length, less data is visualized if the total length of the data is longer than the stroke. These calculations are done when the user finishes drawing a stroke.

In the case of connecting with a relation stroke, the system recognizes the starting point node as a parent node, and the connected node as a child node of the scene graph. Thus, the parameters of the child node change along with those of the parent node. By taking the stroke width into account, the system defines the size and position parameters of the child node. Figure 9 (right) shows an example of connecting an image to a predefined layout. The child nodes that are included in the layout become half the size of the parent, because they are connected with a double-sized pen. Similar sketch-based link techniques already exist [11] and are effective for directly creating relationships between information. However, our system differs in that the pen width can be used to directly control the related value.

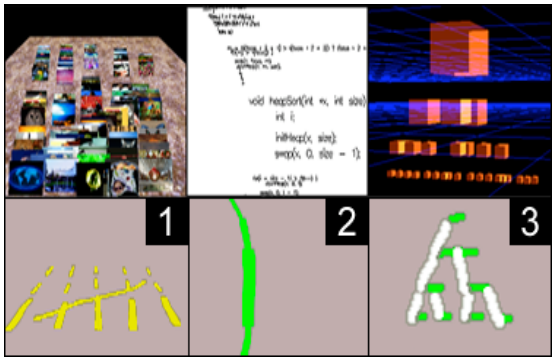

Figure 10: *Layout examples from the VelvetPath System. The figure shows examples of layout creation (top) and how they were created (bottom).*

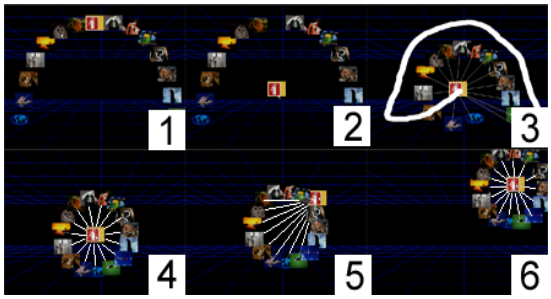

Figure 11: *Creating a graph layout. The user can create a spring model [10] by drawing relation strokes between information (3, 4). Related nodes are moved if a user clicks and drags a node (5, 6).*

5. Layout design example

Figure 10 shows examples of the VelvetPath system being used to create layouts similar to those of conventional information visualization systems (top) and how the example layouts were created (bottom). The user can create layouts by continuously drawing simple strokes and using pen attribute combinations. In Fig. 10 (1), images are placed on the background in a Data Mountain system [2] by drawing shadow strokes. Figure 10 (2) shows a text layout that depends on the pen width. Fig. 10 (3) shows a layered layout like ConeTree [6] created by using the relationships between strokes.

Figure 11 shows how a spring model can be created. In this case, the pen width of a relation stroke is not related to the information size, but to the force (e.g., attraction or repulsion) between nodes. If a user connects a node to another node with a relation stroke, the system contains a click node and a connected node as shown in Fig. 6. The system then calculates the attraction or repulsion, and displays a line between the nodes (Fig. 11 (3, 4)). These nodes are calculated according to the frame rate, so related nodes are moved if a user clicks and drags another node (Fig. 11 (5, 6)).

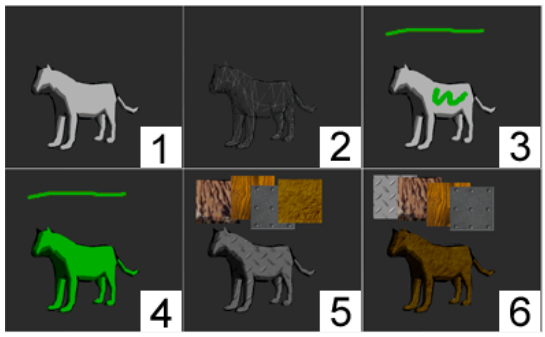

Figure 12: *Example of texture mapping. A stroke enclosing the model is drawn to define a texture, and the model is painted to map the texture (3, 4). The texture will be moved if it is on the stroke, but mapped to the model if it is on the model (5, 6).*

Figure 13: *Example of scene creation. A user can shift models and use strokes to set 3D models.*

6. VelvetPath applications

VelvetPath system has a variety of applications, such as to content creation and presentation.

6.1. Creation

The user can apply the system to 2D illustration, 3D modeling, or scene creation. As the painted area is recognized as one big point, the user can map texture data or set objects within the area.

6.1.1. Texture mapping

Figure 12 shows texture mapping to a 3D model with the VelvetPath System (Fig. 12 (1, 2)). When a stroke enclosing the model is drawn to define a texture and the model is painted to map the texture (Fig. 12 (3, 4)), the texture will be set and moved on the stroke and the painted area, so the user can quickly retrieve a desired texture for the model (Fig. 12 (5, 6)). In conventional systems, the user has to use file dialogs to import image data. However, users of our system can set data by simply drawing a stroke and painting on objects. In the same way, the user can test vertex and pixel shaders.

6.1.2. Scene layout

Likewise, a user can apply the system to 3D scene creation (Fig. 13). In this case, the user draws a stroke and a large point to define scene elements such as 3D models on the stroke. The user can then select and define elements by shifting the models.

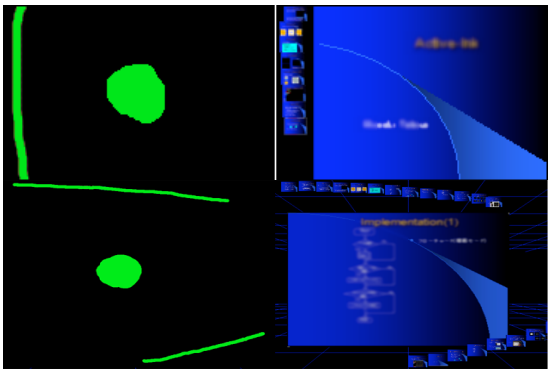

Figure 14: *Presentation with the VelvetPath system. The user can produce presentations with original layouts.*

In the case of content creation, as the user can set a layout in the same workspace as for the main creative activities, the user can rapidly test textures and models by using simple manipulations. We expect these creation methods to be particularly useful for novice content creators and children.

6.2. Presentation

A user can create a layout for a presentation by drawing strokes, and thus can define and show presentation contents by applying different layouts (Fig. 14). The layouts can be designed to show all slides with scaling, with the biggest slide corresponding to the current part of the presentation sequence. In a presentation, it is important that the audience understand the relationships between previous and current slides. With such layouts, the audiences can easily understand these relationships. Also, these techniques are available for use during question and answer sessions, as well as during the main presentation. For example, if the audience wishes to see two particular slides, the presenter can quickly and easily paint the two areas.

In small meetings as well, the VelvetPath system provides good support for displaying data. Here, the user can choose and create the most effective layout according to the situation and data type.

7. User experience

Here, we discuss the experiences of test users who used the VelvetPath system for information retrieval, and compare the use of various layouts.

7.1. Tasks

The users were asked to retrieve information (in a group of random images and similar images), and map images to a 3D model by using an original layout and a pre-defined layout. We also asked the users to rate the techniques on a scale from 1 to 5 (1 = very bad, 2 = bad, 3 = OK, 4 = good, and 5 = excellent) and to explain their choices.

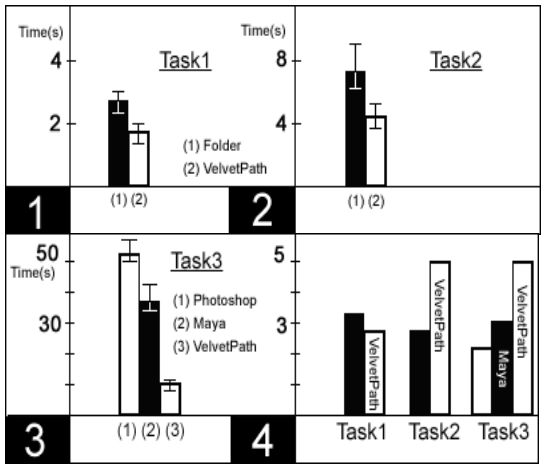

Figure 15: *Results of user testing. The results for tasks 1, 2, and 3 are shown in (1), (2), and (3), respectively. User reactions are shown in (4).*

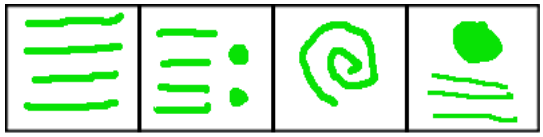

Figure 16: *Examples of layouts drawn by the users in the test.*

7.2. Subject

Six users, who were using the VelvetPath system for the first time, performed the tasks. They varied in terms of their proficiency in using computers, and none had special design skills. Before the experiments, the users were trained for five minutes on how to use VelvetPath features such as how to control the information's position and size. In all tests, the users first created original layouts and then redesigned them.

7.3. Results

Figure 15 shows the results of the user tests. The results for tasks 1 and 2 are shown in Fig. 15 (1, 2). In these cases, users retrieved information in a folder supported by the OS and the VelvetPath layout. In Fig. 15 (3), the users used the VelvetPath layout for their creative activities. In this case, the users mapped ten textures onto the same position by using Photoshop [17], Maya [12], or VelvetPath. The user reactions are shown in Fig. 15 (4). Figure 16 shows the the layout designs drawn by the.

8. Discussion

Here, we discuss the interactions and applications of the VelvetPath system based on the user experiences during the test and comments by visitors to our demonstrations and consider related work.

8.1. Interactions

Users quickly understood our system concepts and interaction methods. There was no difference between our system and the comparison system in task 1, but the users of our system could retrieve information more quickly than with the comparison layout used in task 2. Our system was more effective in task 2 because the users controlled the scale parameter by using the paint area and pen width, and this allowed them to see information in detail so that they could quickly retrieve it. Since our system seems to be superior when handling similar data sets, we think it will be effective for processing the work history in programming, mail, and creation tasks. In task 3, all users could retrieve desired images more quickly. Users of our system could test mapping data by simply drawing and painting without using the file dialog, so the manipulations were faster than in the other methods. Together, the results from tasks 2 and 3 suggest that our interactions are an effective means for information retrieval that requires detailed information and creativity. The users reacted favorably to our system throughout the testing (Fig. 15 (4)).

In the layout design, most users designed original layouts before the information retrieval. (Examples of the layouts are shown in Fig. 16.) Although none of the users created a complex layout, they could easily create simple layouts, and drew words and pictures as well as a strokes before setting the information. Our system is designed to enable a user to design simple layouts and freely redesign them. The users in the test could casually create simple or strange layouts, without needing special design skills, and design difficulties did not arise. We expect that users will be able to create more original, increasingly effective layouts by continuously redesigning existing layouts according to their needs.

Moreover, we showed through simple demonstrations that both the sketch and paint interactions could be used to directly control parameters (e.g., scale and relation). Users reacted especially favorably to the relation stroke, because this stroke made it possible to handle the user's original data set while controlling the relation rate by adjusting the pen width. Generally, the users held their original data set in a folder named related data, and each user could establish relationships between information and create simple databases by using only relation strokes. On the other hand, some users had questions related to the system's lack of effectiveness in making complex manipulations. Also, a few visitors would have preferred more complex layouts and relationships. However, our goal was to enable simple and convenient interactions that would allow casual users to freely apply the system. Complex manipulations and GUIs would increase the complexity of the layout and relations, and so, the system does not support them.

8.2. Application

Many visitors to the system demonstrations could imagine a wide variety of applications for the system and commented on the uniqueness of the potential applications. Some people suggested that the system would be useful for movie making, because the user can draw strokes to directly indicate the time axis. In movie creation, each frame's data is connected to time information, so the user can see and smoothly exchange movie contents. Also, it is possible to treat data from a group of frames through a drawn path.

8.3. Related Work

There are many sketch-based interfaces that allow users to perform 2D manipulations of 3D creations. Characteristically, all the manipulations in these systems are simple and similar to drawing a stroke on a piece of paper with a pen. Sketch [17] users can draw 3D curves by performing 2D manipulations. This system calculates a 3D curve by combining a 2D stroke and a shadow stroke. Users of Harold [18] and Tolba [19] can create flat models in a 3D space with sketch-based manipulations, effectively creating a 2.5D scene in a 3D space. Teddy [16] is another 3D modeling system with 2D manipulations. In Teddy, the user interactively draws freeform 2D strokes to specify the silhouette of an object, and the system automatically constructs a 3D polygonal surface model based on these strokes. PaintEffect [15] is another sketch-based system, used not to create 3D scenes and models, but to add accents (e.g., grass, flowers, trees, fire, etc.) to existing 3D scenes. In this system, while information is set along the user's strokes, the system cannot recognize a painted area as one area and cannot recognize variation in the pen width. Thus, users have to control information attributes by setting information parameters through a complex GUI.

9. Conclusion and future work

The VelvetPath system is an information retrieval system that integrates the simple manipulations of sketch interfaces with information visualization systems. We have described the prototype VelvetPath system and its features, such as basic and particle layouts and interaction support. Several examples were shown to demonstrate how the system can be used for a wide variety of applications.

We will consider other types of effective animation methods by using simple techniques like sketch-based manipulations, because the combination of layout and animation is important to support more natural information retrieval.

10. Acknowledgement

We thank the members of our Sony CSL Interaction Laboratory for their encouragement and helpful discussions and comments.

References

1. S. K. Card, J. D. MacKinlay, and B. Shneiderman. Readings in Information Visualization: Using Vision to Think. Morgan Kaufmann, 1999.
2. G. Robertson, M. Czerwinski, K. Larson, D. Robbins, D. Thiel, and M. van Dantzich. Data Mountain: Using spatial memory for document management. *In Proceedings of UIST '98,* pp. 153-162. 1998.
3. H. Koike. Fractal views: a fractal-based method for controlling information display. *ACM Transactions on Information Systems (TOIS)*, Vol. 13, No. 3, pp. 305-323, July 1995.
4. G. W. Furnas. Generalized fisheye views. *Proceedings of the ACM Tran. on Computer-Human Interaction,* Vol. 1, No. 2, pp. 126-160, 1994.
5. B. B. Bederson, J. D. Hollan, K. Perlin, J. Meyer, D. Bacon, and G. Furnas. Pad++: A Zoomable Graphical Sketchpad for Exploring Alternate Interface Physics. *Journal of Visual Languages and Computing,* Vol. 7, No. 1, pp. 3-31, 1996.
6. G. G. Robertson, J. D. Mackinlay and S. K. Card. Cone Trees: Animated 3D Visualization of hierarchical information. *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI '91),* pp. 189-194, 1991.
7. H. Koike and T. Takada. VisuaLinda: A Framework for Visualizing Parallel Linda Programs. *Proceedings of 1997 IEEE Symposium on Visual Languages (VL'97)*, pp. 174-180, 1997.
8. T. Takada and H. Koike. Tudumi: Information Visualization System for Monitoring and Auditing Computer Logs. *Proceedings of 6th International Conference on Information Visualization,* pp. 570-576, 2002.
9. J. Rekimoto. Time-Machine Computing: A Time-centric Approach for the Information Environment. *In Proceedings of UIST' 99*, pp. 45-54, 1999.
10. R. Davidson and D. Harel. Drawing Graphics Nicely Using Simulated Annealing. *ACM Transactions on Graphics*, Vol. 15, No. 4, pp. 301-331, 1996.
11. J. Lin, M. Thomsen, and J. A. Landay. A Visual Language for Sketching Large and Complex Interactive Designs. *In Proceedings of CHI2002*, pp. 307-314, 2002.
12. Maya and PaintEffect. http://www.aliaswavefront.com/
13. T. Igarashi, S. Matsuoka, and H. Tanaka. Teddy: A sketching interface for 3D freeform design. *In SIGGRAPH '99 Proceedings*, pp. 409-416, 1999.
14. R. C. Zeleznik, K. P. Herndon, and J. F. Hughes. An Interface for Sketching 3D Curves. *In SIGGRAPH '96 Proceedings*, pp. 163-170, 1996.
15. J. M. Cohen, J. F. Hughes, and R. C. Zeleznik. Harold: A World Made of Drawings. *In NPAR2000 (Symposium on Non-Photorealistic Animation and Rendering)*, pp. 83-90, 2000.
16. O. Tolba, J. Doresey, and L. McMillan. Sketching with Projective 2D Strokes. *In Proceedings of UIST '99*, pp. 149-157, 1999.
17. Photoshop. http://www.adobe.com/