ActiveInk

Hiroaki Tobita and Jun Rekimoto

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

Abstract

The ActiveInk system integrates the advantages of real world painting techniques with computer graphics (CG) effects such as natural phenomenon animations (e.g., water, fire, snow, and clouds), attributes (e.g., rubber, cloth, and land), surface materials (e.g., texture effects, metal, and glass), and so on. Most conventional paint systems mainly allow users to set a simple and static color. Also, they require users to control many parameters if the user applies complex effects. However, the ActiveInk system treats many behaviors as separate behavior inks (e.g., water, cloud, and cloth ink), so a user can add effects by selecting a behavior ink and painting it onto objects to realize CG effects. Moreover, the system has a palette area that is similar in function to an actual painter's palette, so the user can create a new ink by mixing different types of behavior ink and can control the behavior in the palette area directly. In this paper, we describe a prototype of the ActiveInk system, explain how it allows CG effects to be applied through simple and easy manipulations, and discuss its implementation.

Keywords:
3D painting, Real-world manipulations, Behaviour ink, Palette interaction.

1. Introduction

There are now many opportunities to watch movies and games that include Computer Graphics (CG) and their effects. In such an environment, some people will naturally want to create such movies and games by themselves. Many software products and research systems are available to support these users' desires. However, these systems require users to manipulate complex GUIs and many parameters regarding 3D geometry. As a result, people cannot create desired-3D CG and their effects easily and rapidly.

Conventional CG creation systems require a user to control many types of GUIs and parameters, even when the user adds effects similar to existing effects, so the user has to learn complex GUIs before creating a desired scene and model. These problems are a serious barrier to beginners who want to create 3D CG, so simpler and easier interaction techniques are needed.

Sketch and paint systems [7,14,15,16] for 3D CG creation that allow simple and easy manipulations have been developed. Characteristically, these systems use real-world techniques such as drawing and painting for 3D CG creation. In such systems, the user's drawings are automatically projected onto a 3D world, so difficulties related to 3D CG, such as geometry and parameter setting, are avoided. Thus, users of these sketch systems can create a 3D scene and model through 2D drawing in a way that is similar to drawing a picture on a piece of paper in the real world. Also, users of paint systems can paint a color onto 3D modes as if 2D painting. However, as these systems are mainly designed for simple and rapid 3D CG creation, they are too simple to efficiently support creation.

To realize a system that is both simple and practical for creative activities, we sought to integrate the advantages of real world painting techniques (e.g., an ink metaphor and palette manipulation) with CG creation, because integration based on real-world activities would make the system easy to use without requiring the user to have extensive knowledge regarding 3D CG. The ActiveInk system has two important features: ink metaphors to support a wide variety of expressions, and palette manipulation to avoid complex parameter setting. Our approach for the system implementation is not to increase the number of GUIs and parameters to achieve new behavior, but to enable new behavior through the same paint interactions. Thus, by using the ink metaphor and palette interaction, a user can use a wide variety of behaviors by means of simple manipulations such as selecting an ink and painting with it, despite the use of various algorithms in the background.

Figure 1: *System Overview. The ActiveInk system is divided into three areas: the ink (1), palette (2), and work (3) areas. Some GUIs are included.*

In the ink area, the system treats a wide variety of behaviors (e.g., such as animation, attribute, surface material, and texture effects) as inks. Calculations of the ink behaviors are based on a painted area. In addition, there are similar inks to choose from in the area. For example, a user who wants to use a cloud ink can select from among different types of cloud ink by considering the desired scene. Moreover, our system has a palette area, so that a user can mix different types of ink to create new or strange inks by painting an ink onto a painted area, and can control the behavior conditions (e.g., animation, color, speed, and size) within the palette area. Since the user can use the palette area in a way similar to how a painter uses a real palette, the user does not have to handle complex parameters or GUIs during the creative process. Also, the user can create different types of scene and model by controlling the mix color in the palette area; thus, for example, he can create a sunset by drawing an orange cloud that is a mix of cloud ink and orange color.

In this paper, we describe a prototype of the ActiveInk system, explain how it allows CG effects to be applied through simple and easy manipulations, and discuss its implementation.

2. Previous Work

Dobashi [1] has developed a system that simulates a realistic cloud by using cellular automata. The cloud simulation is calculated using Boolean parameters and visualized with metaball techniques, so a smooth and realistic cloud animation is produced. There are systems to produce fire simulations by using cellular automata with random noise. In addition, there are water and cloth simulations based on a spring model realized through mechanical simulations [17]. There are also many systems to produce natural phenomena and attributes with a particle system [4,5]. The effects created by particle systems have already been used in conventional systems, so a wide variety of effects can be realized to create scenes, games, and movies. With a particle system, a user can set detailed information concerning, for example, water flow, smoke, or hair by using the particles [2,8,9]. Moreover, there are effective systems that combine cellular automata with a particle system. Muraoka's system [3] uses such a combination effectively to visualize snow. These systems mainly focus on achieving realistic CG expressions, rather than focusing on the interaction techniques.

Figure 2: *Examples of using ActiveInk. A user selects an ink and draws with the ink (cloud ink (top) and water ink (bottom)).*

There are also many sketch-based interfaces that allow users to perform 2D manipulations of 3D creations. Characteristically, as a user's drawings are projected onto a 3D world automatically, all the manipulations in these systems are simple and similar to drawing a stroke on a paper with a pen [14,15,16]. Actually, these systems are effective from the viewpoint of simple interactions, so even beginners and children can easily interact with 3D CG. However, as there is no design support such as template objects and scenes, users have to consider all scenes and models themselves. Thus, creation results depend on the users' design skills.

In addition, 3D paint systems have been developed. Chameleon [7] is a 3D paint system that uses an effective means of calculation for UV mapping. Users of this system can draw detailed strokes that are not related to the UV position of texture. Lengyel [13] has reported a system that can be used to realize real-time fur over arbitrary surfaces. By using lapped textures [12], a user can achieve real-time modification of viewing and lighting conditions, as well as local control over hair color, length, and direction. Paint Effects [6] is often used to create parts of 3D CG scenes, movies, and games. While it is also a paint-oriented tool focused on simple and easy creation with a brush metaphor, many parameters are needed to add detailed effects. Maya [6] has many effects that can be used for modeling and scene creation, but a user has to manipulate different types of GUIs and complex parameters to create a designed scene, even when the user adds effects of a type similar to existing effects.

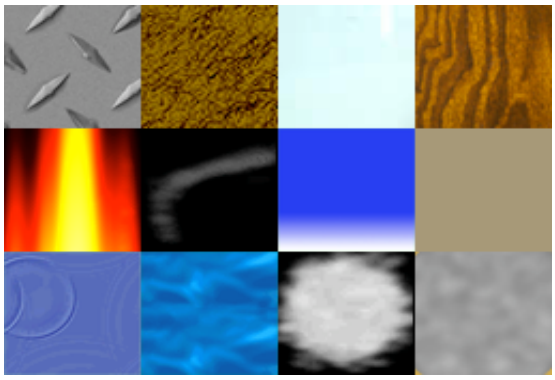

Figure 3: *Behavior Inks. (1) metal ink, (2) ground ink, (3) glass ink, (4) wood-grain ink, (5) fire ink, (6) smoke ink, (7) sky ink, (8) blank area, (9, 10) different water inks, (11,12) different cloud inks.*

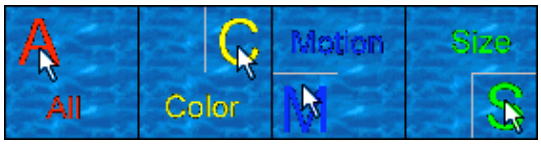

Figure 4: *Ink elements. Each behavior ink is divided into four areas: motion, color, size, and all. Support information appears if the mouse cursor moves over an area.*

3. ActiveInk system overview

The ActiveInk system is divided into three areas: the ink, palette, and work areas (Fig. 1). The ink area holds both the behavior ink and normal colors. In the ink area, each behavior ink has a specific behavior and is animated if it supports animation. In the palette area, a user can mix different types of ink and can control the ink behavior. In the work area, the user can use the ink to create the designed scene. Moreover, the ActiveInk system includes a simple GUI to select pen attributes.

3.1. Ink Area

In the ink area, there are two types of ink: behavior and normal color inks. Behavior ink is used to add behaviors such as animation and attributes, and normal color ink is used to paint with color. Both inks are used for the palette and work areas.

By using the up and down GUI buttons at the bottom of the ink area, a user can shift the ink area to show and select other inks. In the ink area, the animation ink is animated, so a user can select an ink directly through the animation. Examples of the inks found in the ink area are shown in Fig. 3. Inks that have the same name but a different behavior are also included in the area. In addition, as each ink has a default behavior that is reflected immediately after painting, the user can directly recognize the effects without a rendering and preview window. Figure 2 shows examples of drawing with cloud and water inks. The user first selects the cloud ink in the ink area, and then paints with it in the work area. The cloud behavior automatically appears and is animated within the painted area (Fig. 2(top)). Also, the user can erase a painted area by selecting the eraser GUI. While each ink behavior has default parameters, the user can control these parameters by drawing in the palette area.

Each behavior ink is divided into four areas: motion, color, size, and all ("all" means motion, color, and size), so the user can select an ink element by selecting the click area of an ink icon. Support information to click is displayed if the mouse cursor moves over the area (Fig. 4). In addition, the user can add a new ink created in the palette area as a new behavior ink by drawing the new ink in the blank area within the ink area (Fig. 3(8)).

Our main focus is to realize unique paint interactions that can support a wide variety of expression simply and easily, and here we do not consider the most effective algorithm to realize the behavior inks.

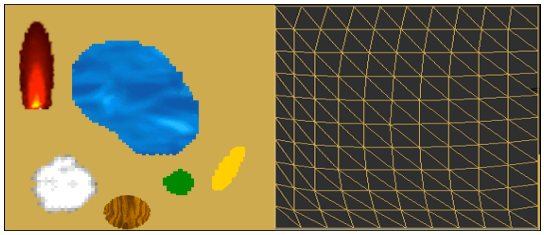

Figure 5: *Palette area. A user can mix different types of ink and control the ink condition in an area. The palette area is constructed as a 3D model.*

3.2. Palette Area

The palette of the ActiveInk system allows a user to create a new ink by mixing different types of ink, and to control the behavior condition. Moreover, the palette area is constructed using a 3D model and is reflective to light and a camera, so the ink behavior is reflected in the same way as it will be when used in the work area (Fig. 5).

3.2.1. Mixing

To mix inks, a user first applies a behavior ink in the palette area and another ink in the painted area (Fig. 6), so mixing manipulations are done simply by painting. The system then calculates a new behavior that has the features of both inks. Simple examples of mixing between normal colors are shown in Fig. 6. The user can also create strange inks (such as a cloud-fire or rubber-ground ink) by mixing inks (Fig. 7). Moreover, a user can mix behavior and normal color inks to change the default color of a behavior ink and thus create, for example, blue or green fire using the color parameters of pixels.

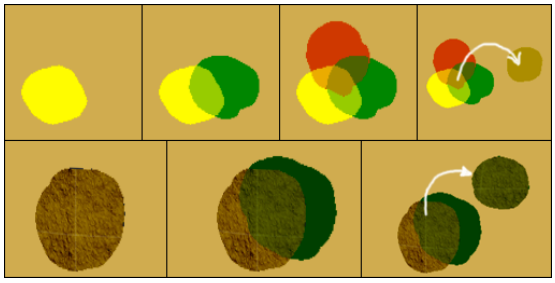

Figure 6: *Mixing ink in the palette area. A user can retrieve ink by mixing (top). Metal and cloud inks are mixed with a normal color.*

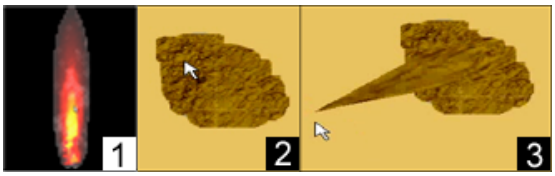

Figure 7: *Examples of mixing. A user can create a new ink by mixing different types of ink; for example, a cloud-fire ink (1) and a rubber-ground ink (2, 3).*

Figure 8: *Example of diluting ink behavior. A user can control the behavior of a painted area by mixing the ink.*

3.2.2. Diluting and Inspissating

Moreover, a user can control ink behavior by drawing the same ink for inspissating and diluting. Although each behavior ink has a default behavior, the default behavior might not be adequate for the user's design. The system allows the user to control these behaviors through mixing.

In our system, three mouse buttons (left, middle, and right) are attached to ink selections in the palette area. The left button is used for diluting, and the right button is used for inspissating. Thus, if a user clicks an ink with the left button and paints the ink onto a painted area, the behaviors of the painted area are changed along with the ink features. The middle button is used to absorb ink in the palette area.

An example of inspissating in this way is shown in Fig. 8. In this example, a user paints the same ink onto a painted area to dilute size. Inspissating manipulation is thus similar in function to the use of water and white ink in water-color painting.

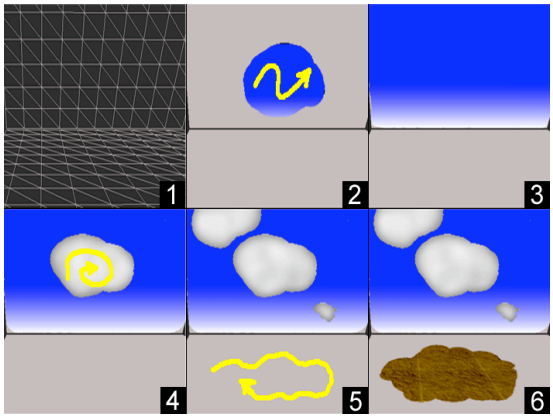

Figure 9: *An example of painting a behavior ink onto a 3D scene. A user first draws with sky and cloud inks in the sky (2, 3, 4), and then draws with a ground ink and a mixing ink to create the ground (5, 6).*

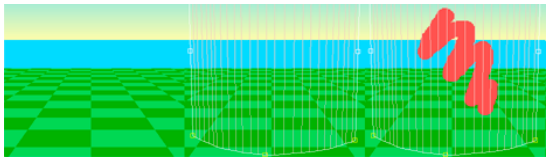

Figure 10: *A canvas creation. A canvas appears along a user's stroke. The user can then paint with a behavior ink on the canvas.*

3.3. Work Area

In the work area, a user can apply ink to objects by selecting an ink from the ink area or clicking a painted area in the palette area. The user can also change behavior vectors by using GUIs.

3.3.1. Painting in a 3D scene

Figure 9 shows an example of painting a behavior ink onto a simple 3D scene constructed using sky and ground objects. The user first draws with a sky ink on the top object and then draws with a cloud ink on that object (Fig. 9 (1, 2)). Next, the user draws on the bottom object using a ground ink (Fig. 9 (3, 4)). By mixing inks between a ground ink and a normal color, the user can add accents to the ground. A cloud ink has a default animation, so the sky object is animated in the painted area.

3.3.2. Add Canvas

By using sketch techniques, the user can create a canvas for a 3D scene for painting. If the user draws a stroke on a pre-defined object, the canvas automatically appears along the user's stroke (Fig. 10). The user can then paint with a behavior ink onto the canvas. The canvas is especially effective for painting with fire and fog inks.

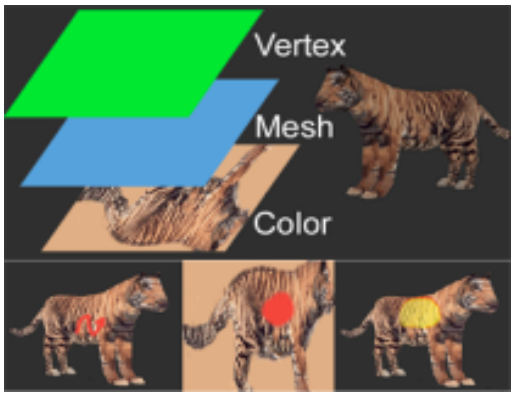

Figure 11: *Paint layers. 3D models in our system have three layers: color, mesh, and vertex (top). The layers are calculated when a user paints (bottom).*

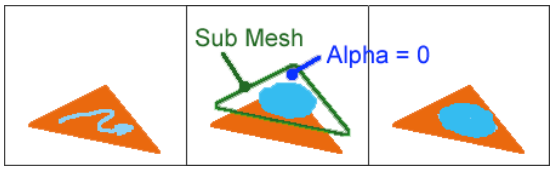

Figure 12: *Vertex shader ink. the system attaches a patch and sets an alpha value to a non-painted area.*

4. Implementation

In this section, we describe the implementation, focusing on the ink behavior and mixing in our system.

4.1. Ink behavior

Typically, the behavior ink is divided into four types based on the implementation: pixel, mesh, particle, and shader inks. To combine different types of ink implementation, the palette and work areas have three layers: color, mesh, and vector layers. (Fig. 11 (top)).

Pixel inks are based on cellular automata and texture effects. Behaviors of mesh inks are realized by patches that construct a 3D model. Calculations for the vertex shader inks depend on the patches used to construct models. Particle inks are achieved by using a particle system. Because these inks basically behave within a painted area according to their default behaviors, which depend on parameters predefined by the system, the painted area automatically behaves in a certain way. Behavior ink is implemented by connecting the user's painted area to these techniques (Fig. 11 (bottom)).

If a user draws with a vertex shader ink, the system receives click patch information and sets the same patch size for the clicked patch. The system then reflects the user's paint operation. As a non-painted area of the patch area is set to an alpha value, a vertex shader is reflected only within the painted area. Thus, only the painted area is displayed in the work area (Fig. 12).

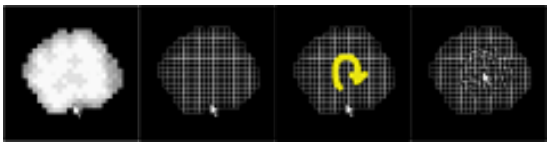

Figure 13: *Example of mixing between differet types of inks. Mixing between cloud (pixel) and water (mesh) inks uses different layers.*

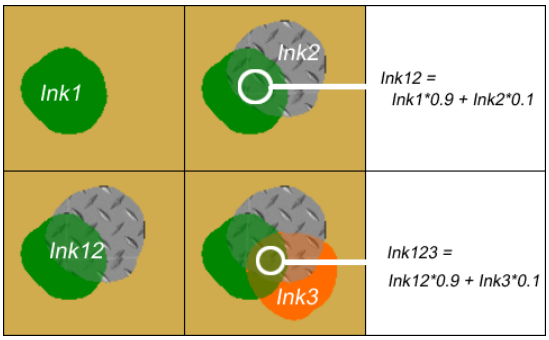

Figure 14: *Mixing. The mixing area has 90% previous ink and 10% current ink features.*

4.2. Mixing

In mixing, the features of both the previous and current ink behaviors must be considered, so we use the ink calculation results for mixing. By mixing inks that use different types of layer (for example, by mixing pixel and mesh inks), the system can easily create a new ink because of the separate layers (Fig. 13).

When inks having the same features are mixed, such as two pixel inks or two mesh inks, the ink calculation results are used as a new ink parameter for the mixing. In our system, the mixing area has 90% previous ink and 10% current ink features, so the final result for an area is based on the first ink feature with the second ink feature added. Figure 14 shows examples of mixing between pixel-based inks.

In the same way as for mixing, the inspissating and diluting manipulations are also based on painting. Continuously painting with the same ink on a painted area strengthens the ink feature (e.g., size, color, or animation) within the painted area. Painting with the same ink causes the system to increase the ink parameter used for the ink calculation. As a result, the user can directly control the parameters. As the user can select the ink element according to the click position in an ink area, the user can change ink behavior characteristics such as speed, size, and color. In the same way, if the user paints with the same ink by clicking with the right button of the mouse, the ink parameters of the painted area will decrease. In this case, the user can also select ink parameters by clicking an ink area position.

Figure 15: *Example of painting for a 3D model. These models are painted using a simple behavior ink. In model painting, a vertex shader ink is effective for changing the surface materials. These examples are expressed by using toon, metal, membrane, and fur inks implemented by vertex shader.*

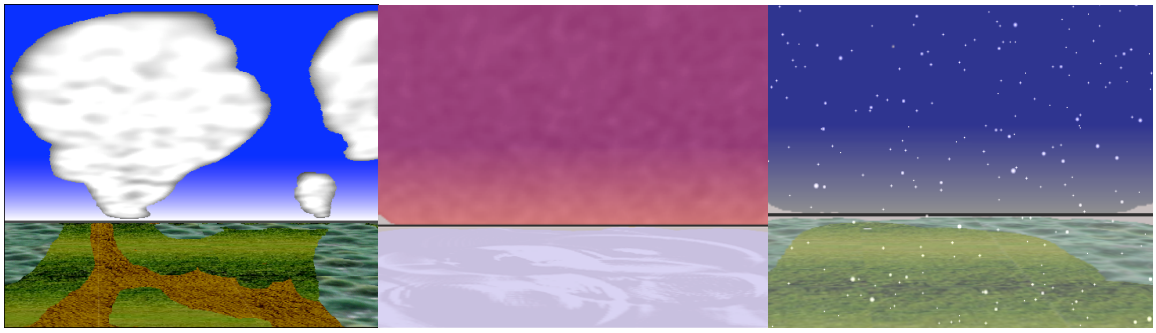

Figure 16: *Examples of painting for a 3D scene. A user can create other types of scenes by mixing inks, and so can create a sunset cloud by mixing between a cloud ink and an orange color (2), or create a scene by using a snow ink and mixing between a ground ink and the white color (3).*

5. Examples of painting

In this section, we describe three examples of painting activities (e.g., painting with a 3D model and a 3D scene) with the ActiveInk system.

5.1. Model

Figure 15 shows examples from the painting of a 3D tiger model. A user can express many types of surface by selecting behavior inks and painting them onto the model. These examples are expressed by using toon, metal, membrane, and fur inks implemented by vertex shader.

5.2. Scene

Scene creations with the system are shown in Fig. 16. A user can create other types of scenes by mixing inks. In the example shown, the user creates a sunset cloud by mixing between a cloud ink and an orange color. In these examples, mixing between a sky ink and normal color is an effective way to express other types of scene.

6. Discussion

Formal user studies regarding our system have not yet been done. However, we have demonstrated the system and received comments from visitors, so here we describe the system's possibilities and limitations in light of the reactions and comments from such users. We also consider applications for the system.

6.1. Advantages and limitations

Our concept of integrating real-world painting techniques with CG creation is clearly understood by most visitors. A primary characteristic of our system is that it enables users to add animation and material by simply selecting a behavior ink and painting with it – rather than by manipulating parameters – and most users have liked the system's simple interactions. Also, as each behavior ink has a default behavior in our system, most of the users could quickly paint a pre-defined model and add elements to a scene by using the interactions. In addition, there have been many requests to add particular inks that the users want to apply, such as star, rainbow, and fountain inks. The system mainly supports four types of ink (pixel, mesh, particle, and shader inks) that allow users to express a wide variety of behaviors, and it is possible to add other types of ink, so adding new inks in a future implementation should not be difficult. Ink-metaphor-supported behaviors have

great potential and can be enhanced by adding other types of behavior ink to enable a wide range of expression; for example, we could create wood, city-scenery, sunset, and season inks. With such inks, the necessary user manipulations would still be ink selection and painting, thus the system manipulation will remain simple and easy to handle.

We also received useful comments regarding the mixing methods. In our first prototype system, the mixing methods directly used the averaged data from calculation results. When users expressed a desire to use only water motions or a fire color, we developed a system that allows users to select parameters according to the click position in the ink area. Visitors who were interested in creating a strange ink that was a mixture of fire and water inks could understand the mixing methods that would allow them to control behavior and create a new ink after a simple demonstration. We also think it is useful to create technical inks (e.g., diluting, inspissating, detail, and rough inks) for mixing. In actual watercolor painting, a painter uses white ink or water for dilution and adds more of the same ink for inspissation, so the system supports these manipulations as a way to free the user from parameter setting.

There are certain limitations in our system. The mixing techniques are not effective in terms of the calculation speed, so more effective mixing techniques are needed. Also, in conventional paint systems, data is treated as simple bitmap data, so a user can use it for other systems; however, the ActiveInk system data is treated as an original four layers and data, so a user cannot use it in other systems. In addition, the ink calculations become increasingly heavy as the number of patches rises. It is also necessary to develop a system that supports time information for each painted area for movie creation.

6.2. Applications

In addition, the system can be easily combined with an augmented reality (AR) or virtual reality (VR) system that includes many types of input and output devices.

For AR, it can be used with a wide display (e.g., a wall-type computer display) and PDA combination [10]. In this case, the user would use a PDA palette that has ink and palette areas, and would be able to draw on a wide wall-type display (Fig. 17); in this way, the user can interact with the creative space more naturally and freely.

For VR, our system can be applied within a 3D shared virtual world. As the capabilities of network environments are further developed, 3D shared virtual world systems have become more common [18,19,20]. However, the system interactions are still quite limited (e.g., to browsing or chatting with other users). In such environments, users would be able to paint 3D models and construct a virtual world by using the ActiveInk system; that is, the user would be able to change the scene by painting and communicating with other users through their own creation. Behavior inks are reflected within a painted area automatically, so it is possible to create a rich world and add elements. We think these creative activities will encourage communication with other users, for example in chat networks.

Figure 17: *Application by combining our system with AR system: A user can use a PDA palette that has an ink and palette area, and can draw on a wall-type display.*

7. Conclusion and future work

The ActiveInk system is a simple and convenient system that allows the user to apply painting techniques in CG creation. System users can add effects by selecting behavior inks (e.g., natural phenomena, attributes, surface materials, and textures) from the ink area and painting the inks onto objects in the work area. A user can also mix different types of ink to create a unique ink in the palette area. In addition, users can control the animation speed and conditions by mixing inks. As a result, a user can quickly and easily paint 3D models and scenes through simple manipulations without dealing with difficulties such as complex parameters and complicated GUIs.

We plan to make the ActiveInk system compatible with other input devices, and will enable simpler and more direct interaction [23]. Moreover, we will improve the ActiveInk system by adding inks such as sound inks and additional manipulations. Much research has been done on such sound-effects application [21,22], and we are eager to incorporate many of the findings into a more advanced version of the ActiveInk system. In addition, user testing will also be necessary to further develop the system and make it more suitable for practical use, so we plan to have both novice users and professional designers use the system in their creative activities.

8. Acknowledgement

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

References

1. Y. Dobashi, K. Kaneda, H. Yamashita, T. Okita, T. Nishita. “A Simple, Efficient Method for Realistic Animation of Clouds”, *In SIGGRAPH ‘2000 Proceedings*, pp.19-28, 2000.
2. P. Fearing. “Computer Modeling of Fallen Snow”, *In SIGGRAPH ‘2000 Proceedings*, pp.37-46, 2000.
3. K. Muraoka and N. Chiba. “A Visual Simulation Of Melting Snow”, Journal of the Institute of Image Electronics Engineers of Japan, 27(4): 327-338, 1998.
4. K. Sims. “Particle Animation and Rendering Using Data Parallel Computation” Computer Graphics, Vol.24, No.4, SIGGRAPH’94, pp.405-513 (1990).
5. Particle system API http://www.cs.unc.edu/~davemc/Particle
6. Maya and Paint Effect http://www.alias-wavefront.com
7. T. Igarashi and D. Cosgrove. "Adaptive Unwrapping for Interactive Texture Painting", *Symposium on Interactive 3D Graphics 2001*, pp.209-216, 2001.
8. R. Fedkiw, J. Stam, and H. W. Jensen. “Visual Simulation of Smoke”, *In SIGGRAPH ‘2001 Proceedings*, pp.15-22, 2001.
9. B. J. Meier. “Painterly Rendering for Animation”, *In SIGGRAPH ’96 Proceedings*, pp.477-484, 1996.
10. J. Rekimoto, "Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments", *Proceedings of UIST'97*, pp. 31-39, 1997.
11. J. Rekimoto, "Time-Machine Computing: A Time-centric Approach for the Information Environment", *Proceedings of UIST'99*, 1999.
12. E. Praun, A. Finkelstein, and H. Hoppe. “Lapped Textures”, *In SIGGRAPH ‘2000 Proceedings*, pp. 465-470, 2000.
13. J. Lengyel, E. Praun, A. Finkelstein, and H. Hoppe. “Real-Time Fur over Arbitrary Surfaces”, *Symposium on Interactive 3D Graphics 2001*, pages 227-232.
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. Zelezni, “Harold: A World Made of Drawings”, In NPAR2000, pp.149-157, 1999.
16. T. Igarashi, S. Matsuoka, and H. Tanaka. “Teddy: A sketching interface for 3D freeform design”, *In SIGGRAPH ‘99 Proceedings*, pp.409-416, 1999.
17. D. Baraff and A. Witkin “Large Steps in Cloth Simulation”, *In SIGGRAPH ‘98 Proceedings*, pp.43-54, 1998.
18. K. Matsuda, Y. Honda, and R. Lea. Virtual Society: Multi-user Interactive Shared Space on WWW, *Proceedings of the 6th International Conference on Artificial Reality and Tele-Existence (ICAT ’96)*, Tokyo Japan, pp.83-95, 1996.
19. A. Druin, J. Stewart, D. Proft, B.B. Bederson, and J.D. Hollan. KidPad: A Design Collaboration between Children, Technologists, and Educators, In *Proceedings of ACM Conference on Human Factors in Computing Systems* (CHI 1997), pp. 463-470.
20. C. Cruz-Neira, D.J. Sandin, T.A. DeFanti, R.V. Kenyon, and J.C. Hart. "The CAVE: Audio Visual Experience Automatic Virtual Environment," Communications of the ACM, Vol. 35, No. 6, June 1992, pp. 65-72.
21. J.F. O’Brien, P.R. Cook, and G. Essl, “Synthesizing Sounds from Physically Based Motion”, *In Siggraph 2001 Proceedings*, pp. 529-536, 2001.
22. K. Doel, P.G. Kry, and D.K. Pai, “FoleyAutomatic: Physically-based Sound Effects for Interactive Simulation and Animation, *In Siggraph 2001 Proceedings*, pp. 537-544, 2001.
23. I. Poupyrev, S. Maruyama, and J. Rekimoto. "Ambient touch: designing tactile interfaces for handheld devices”, *Proceedings of UIST2002*, pp. 51-60, 2002.