

Sketch Based Volumetric Clouds

Marc Stiver, Andrew Baker, Adam Runions, and Faramarz Samavati

University of Calgary

Abstract. Like many natural phenomenon, clouds are often modeled using procedural methods, which may be difficult for an artist to control. In this paper, a freehand sketching system is proposed to control the modeling of volumetric clouds. Input sketches are used to generate a closed mesh, defining the initial cloud volume. Sketch analysis as well as the elevation at which the cloud is drawn is used to identify the cloud type and then generate a mesh with the appropriate characteristics for the determined cloud type. The cloud volume can then be edited using Boolean operations that allow for addition and removal of material from existing clouds. The proposed modeling system provides an intuitive framework for generating individual clouds and entire cloud fields, while maintaining the interactive rates necessitated by the sketch-based paradigm.

1 Introduction

The visual aspects of clouds add a great deal to interactive outdoor scenes. The inclusion of fully volumetric clouds through the use of particle engines adds even more to a scene by providing a realistic feel. However, the variable nature of clouds makes them very difficult and time consuming to model by conventional means. Thus, a faster and more intuitive interface for modeling clouds would be beneficial. In particular, a sketch-based approach is ideal. The words sketch-based tend to bring to mind some of the more familiar sketch interfaces that have been introduced; most of which address mesh based construction and manipulation. However, in some instances meshes may be inappropriate or insufficient to supply the information required to achieve a desired effect. Particle systems, for example, are frequently used to model



Fig. 1. Using the binary cutting tool the artist is quickly able to shape a cloud as they wish

objects and phenomena that meshes are unable to portray. Due to their volumetric nature, and the wide variety of shapes they exhibit, clouds are often represented using particle systems [7,9,11,1,13].

We propose a sketch-based particle placement system for the creation and manipulation of clouds. Using our interface artists can create clouds by drawing them directly into a 3D environment. An adjustable canvas is used to determine the initial depth of the cloud. To further the natural feel of the interface we allow the artist the ability to sketch multiple strokes during the cloud creation process. The input strokes are decomposed and analyzed to determine the type (and number) of convex sections the input sketch is composed of. This, in conjunction with the altitude at which the cloud is drawn, is used to determine a cloud-type. A mesh is then created using the artist's sketch and the pre-determined cloud-type. Finally, the mesh is filled with particles to create a fully volumetric cloud. To achieve a realtime method for filling the mesh, we hash the mesh into a 2D grid which is used to facilitate real time in/out testing for the particles. Once a cloud is created the artist then has the ability to alter the clouds through the use of simple Boolean operations such as cutting and adding to a selected cloud (see Fig. 1).

The organization of the paper is as follows. In section 2 we summarize previous works related to volumetric clouds and sketch-based interfaces and provide a brief overview of cloud meteorology. Section 3 describes our method and gives an overview of the interface and the techniques that are used in the proposed cloud modeling system. Section 4 presents results, and section 5 gives our conclusion and potential future work.

2 Background and Previous Work

2.1 Volumetric Cloud Modeling

Volumetric cloud creation approaches can be separated into the categories of procedural and simulation based methods. Procedural based cloud generation provides a more viable solution for real time implementations and therefore we focus our discussion on procedural techniques.

Harris et al [4] uses a particle system to create clouds and then dynamically renders the clouds to imposters that are bill boarded to face the camera. This allows for a real time cloud environment. Visual artefacts sometimes occur when an object enters a cloud, these artefacts are reduced by using multiple layers of imposters. A similar approach is used in the system implemented by Microsoft Flight Simulator 2004; clouds at a distance are rendered to billboards in an octagonal ring and clouds near the camera are rendered using particles [16]. Groupings of cuboids are used to place the textured particles for the clouds. Both these systems work well in real-time; however neither system specifies a fast and interactive way to model clouds within their systems.

In contrast to these approaches Schpok et al, [14] demonstrate an effective cloud creation method using volume textures which are divided into layers with respect to the light vector and the view vector; spheroids are used in the

placement of the textures. Harris et al [5] demonstrates cloud dynamics by using volume textures in a similar system. These systems make use of advanced graphics hardware for improved performance and the methods used to place the textured particles are either slow for large quantities of clouds or done through simulation.

Cellular automata are frequently used in the rendering of dynamic clouds. Dobashi et al [2] uses voxels that correspond to cells of a cellular automata; each cell is set to either a 1 or a 0, and so can be expressed by Boolean operations. Liao et al [10] uses cellular automata to determine density distributions over time for each voxel within the simulation volume. These processes are generally slow and to our knowledge real time implementation is, at present, impossible on graphics hardware.

Dobashi et al [3] describe a method for cumuliform formation using fluid mechanics that takes into account physical processes to form the cloud to a user defined curve. This work is very related to our method, however fluid mechanics are not suitable for our purposes as in general they do not run in real time.

In a recent paper on cloud modeling [17], we see the creation of precisely specified cloud objects using sphere primitives. The artist is able to quickly create specific shapes using a sketch based interface. Specifically, the system employs an interface similar to that proposed by Igarashi et al. [7], which may not be ideal for clouds (this is discussed in greater detail in section 3.1). The work is on the creation of cumulus clouds and does not mention any other types of clouds or cloud scapes. The physical formation process of clouds is also not taken into account as they are more interested the creation of shapes and not the physical placement of particles.

2.2 Sketch Based Modeling

Teddy [7] is perhaps the most well known sketch based modeling system; it allows the creation of a surface by inflating regions defined by closed strokes. Strokes are inflated, using the chordal axis transform, so that portions of the mesh are elevated based on their distance from the stroke's chordal axis. This system creates models with a plush toy feel that are difficult to obtain using traditional interaction techniques.

Plushie [11] is another sketch based interface that creates a three-dimensional object from an artist's sketch by use of procedural modeling. The specific intent of Plushie however is the creation of stuffed toys and facilitates this by creating two-dimensional patches that can later be sewn together to create a plush toy.

Other procedural modeling systems include work done on more flexible objects such as clothing. Igarashi and Hughes [6] present a method of cloth manipulation where a two dimensional clothing model is placed onto a three-dimensional model, or body, through the use of artists strokes. The placement and manipulation of the cloth is controlled via interactive techniques such as surface dragging and pushpins.

Other mesh creation algorithms include of those proposed by Cherlin et al, [1] specifically, rotational and cross sectional blending. Rotational blending defines a

parametric surface which extends of surfaces of revolution. Two curves are blended together circularly about their local midpoints; the output created by this process can easily be converted to high quality meshes. Meshes generated by this technique describe certain objects well, however not all objects are suited to it (e.g. objects with variable cross-sections). Cross sectional blending permits the cross-section of the rotational blend to be modified from a circle to an arbitrary curve.

As was stated before, for the most part sketch-based systems focus on meshes as they are the most common medium for geometric modeling. Our system relies on this connection between sketch-based system and meshes for the particle placement phase. For this portion of our interface we borrow the techniques presented by Cherlin et al (rotational and cross-sectional blending), as we have found them to produce suitable meshes for most cloud types.

2.3 Cloud Meteorology

Elevation plays a key role in the formation of clouds in nature and we have found it very useful for differentiating between cloud types within our interface. High-level clouds are formed above 6,000 meters; these clouds are composed primarily of ice crystals and are typically thin when compared to other clouds. Mid-Level generally form between 2000 meters and 6000 meters, whereas low-level clouds tend to form below 2000 meters; these clouds are composed mainly of water droplets and generally have more vertical development than high-level clouds (See Fig. 2). Clouds that span multiple levels such as cumulonimbus also exist however we have yet to integrate them into our interface. Adding to the cloud classifications, we have cumulus clouds which are associated with convection, and stratus clouds that result from forced lifting of air. Cumulus are generally more full and fluffy where as stratus tend to be more thin and wispy. Both cloud types can occur at any of listed altitudes.

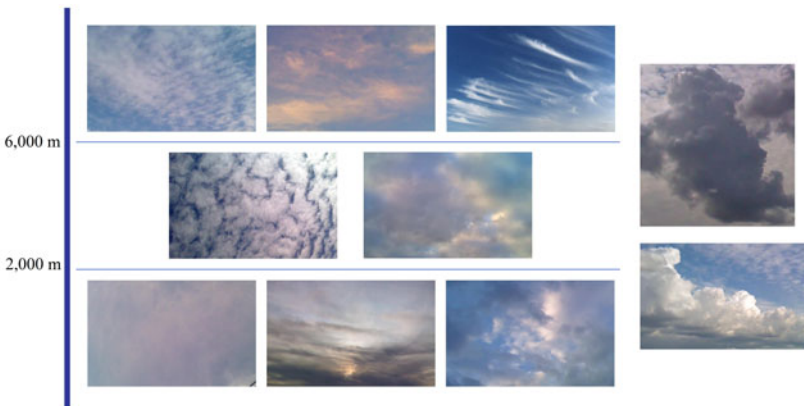


Fig. 2. The 10 different cloud types; From left to right; Top Row: Cirrocumulus, Cirrostratus, Cirrus; Middle Row: Altostratus, Altopumulus; Bottom Row: Stratus, Nimbostratus, Stratocumulus; Side (Top to bottom): Cumulonimbus, Cumulus

3 Methodology

In this section we outline the cloud creation and manipulation processes in our system, along with some of the features of our interface. A mesh is created from the artist’s input sketch, height of the sketch and drawing style (See figure 3 for the cloud creation process). The mesh is then filled with particles using an accelerator grid.

3.1 Creation of a Cloud

Clouds come in a multitude of shapes and sizes, as such creating all the different types of cloud using one methodology is difficult. In order to give our interface the ability to generate a diverse set of clouds, the creation of meshes and selection of textures changes based on how and where the artist draws their strokes. The elevations combined with the general features of the strokes are analyzed. By using three separate elevation levels and two sub-types we are able to define six of the ten recognized cloud types [8].

Two horizontal semi-transparent planes are used to allow the artist to see the level at which they are sketching the cloud (Fig. 4 displays the interface). These elevations are used to determine the clouds base-type; a low-level base-type is created by sketches that originate below both planes; sketches drawn above the lower plane will create a mid-level base-type and sketches above both planes will create a high-level base-type. The sub-type of the cloud is determined by analyzing the users input sketch and can roughly be paralleled with a subtype of cumulus or stratus.

The Canvas. As with most sketch-based interfaces we must address the problem of inferring a three dimensional object from two-dimensions. The inherent

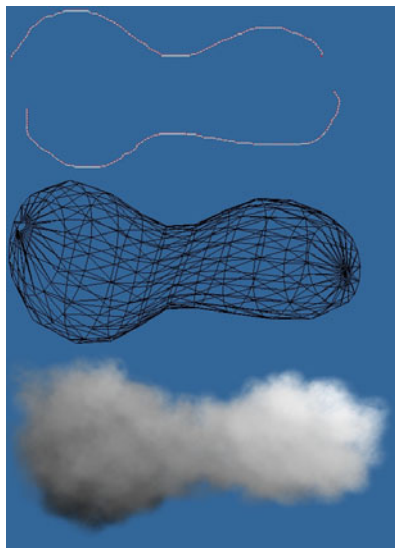


Fig. 3. The cloud creation process

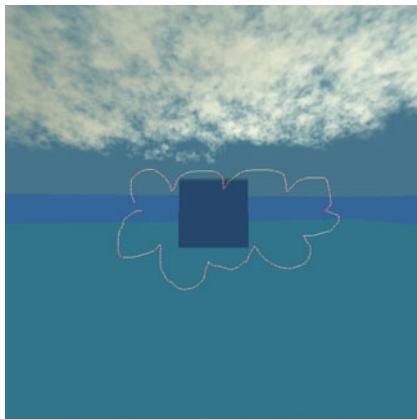


Fig. 4. The canvas is seen at the center of the image, the two horizontal planes specifying cloud height are also shown

ambiguity in the process makes it difficult to determine where the artist intends to place the object. In order to reduce this ambiguity, and enable the artist to place clouds more precisely, a sketch plane has been added to allow the artist to specify depth of placement. The sketch plane is designed to be in view at all times. When the artist sketches, the cloud is created at the depth of the canvas.

Mesh Creation. As with the placement of the cloud itself, inferring depth for the mesh is a difficult task. There are a multitude of different meshes that can be made and for the most part every sketch interface yields different outputs for the same input strokes. The type of mesh used in our application is determined by using the cloud base-type and the cloud sub-type. A cloud in our interface has a Cumulus or a Stratus sub-type as well as a high, mid or low level base-type.

Two separate mesh interpretations are used based on these given parameters; we have borrowed the rotational and cross-sectional blending techniques presented by Cherlin et al [1]. Rotational blending is used for mid and low-level clouds in our application, whereas cross-sectional blending is used for the high-level clouds. A rotational blend has a similar surface

to that of a surface of revolution; however a surface of revolution rotates about a line axis creating a uniform surface, whereas rotational blend rotates about a curved axis creating a blended surface between two curves. Any type of mesh creation can be used with our interface, however we have chosen the rotational (or circular) blending because the meshes that are created by it are similar to many low level and mid level clouds. The circular blend is a good interpretation of a cloud and we have found it to be more visually appealing than other mesh creation methods (such as inflation [7], see Fig. 5 for a comparison).

Cross-sectional blending is used for the high level clouds in our system. Similar to the rotational blend, a cross-sectional blend interpolates between two curves to form a surface. However, where the cross-section of the rotational blend would parameterize a circle, the cross-sectional blend uses predefined functions to determine cross-sections. This allows us to obtain a mesh that is thin, long and flat which turns out to be ideal for placing sparse sections of clouds as seen in



Fig. 5. The top two images display the use of Teddy style inflation within our application. Top left is the same orientation as the sketch was drawn and top right is the above view of the created cloud. The bottom two images use the same camera orientations for a similar sketch using the rotational blending algorithm. Teddy’s mesh creation tends to produce a more squashed mesh when viewed perpendicular to the canvas; our method uses a more uniform interpretation which we have found to be more appropriate for clouds.

figure 8. Additional cross-sections can be specified by the artist, thereby providing more control over the cloud volumes.

Filling The Mesh With Particles.

The cloud mesh is then filled with particles. These particles naturally create noise on the boundaries of the cloud which would otherwise require the use of a very high resolution mesh. An axis aligned bounding box is used in the initial placement of potential particles. The size of the particles used is based on the cloud type determined at the mesh creation stage. The bounding box is filled randomly with points; the density of these points is determined by the volume of the bounding box and the size of the particles. Each particle is then tested to determine whether it is within the mesh. Since the mesh may not be uniform a ray casting algorithm can be used for this purpose [15]. However, this approach does not allow for interactive rates to be achieved, which we require for our interface. Therefore we have found a more efficient approach.

Determining the particles location can ultimately be reduced to a series of triangle in and out tests by use of an accelerator grid. During the mesh creation process the triangles of the mesh are hashed into a two-dimensional grid. To make use of the accelerator grid the vertices, the maximum and minimum height, and the normal of each triangle are calculated and stored in the appropriate grid cell. The proper location is found for each triangle by projecting it onto the accelerator grid (see Fig. 6).

The position of each particle is then hashed into the accelerator grid and checked against all the triangles registered in that location. For inside/outside tests, a ray is projected from each particle, along the axis perpendicular to the accelerator grid. If there are an odd number of hits we conclude that particle is inside the mesh, otherwise the particle is removed. This optimization makes the filling process very quick, however since we reduce the problem to two dimensions some ambiguities arise in the third dimension. In particular, a particle may be above the minimum height of a triangle and below the maximum height, as the triangles exist in three dimensions. In this situation we do not know if the particle is inside the mesh or not, as a hit will be registered either way. This situation is resolved by comparing the normal of the triangle with the vector between the triangle and the particle in question as follows:

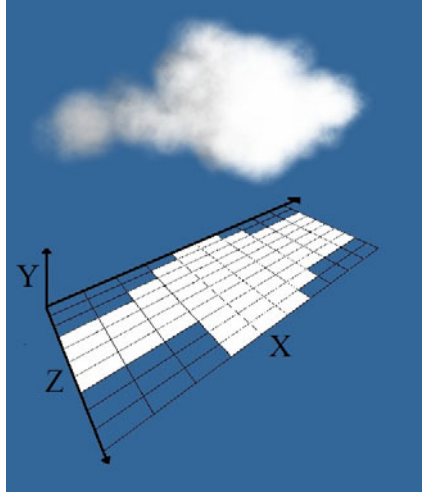


Fig. 6. The perimeter of the cloud is projected onto the accelerator grid to demonstrate where the triangles of the mesh would be hashed

$$Orientation = \begin{cases} Inside, & \text{if } M_n \cdot A_p \geq 0 \\ Outside, & \text{if } M_n \cdot A_p < 0 \end{cases} \quad (1)$$

where M_n is the normal of the triangle n and A_p is the normalized vector from the particle p to the center of the triangle n . By using this calculation ray casting is avoided entirely which permits the algorithm to be run more efficiently.

3.2 Editing

Since our end result is a grouping of particles we also define a set of particle based editing operations. The nature of a particle based structure allows us to implement some basic Boolean operations in order to manipulate the overall appearance of the cloud. Sketch-based techniques are once again used for sculpting and extending the cloud models.

Sculpting. We use a closed sketched curve (see Fig. 7, top row) for cutting away sections of the cloud. After a closed stroke has been drawn each particle within the selected cloud is temporarily projected to the screen and particles within the stroke are removed. The technique which we use for efficiently filling the mesh with particles can be easily extended to cut away particles in two-dimensions.

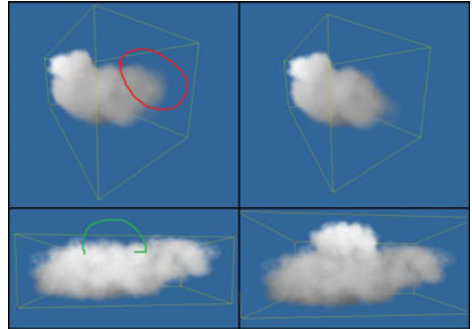


Fig. 7. The editing operations; Top Left: artist sketching a closed edit stroke. Bottom Left: artist sketching an open curve. The results of these operations are displayed in the top-right and bottom right respectively.

Extending. To add to the cloud in a given direction an extending operation has been provided to the artist; the interface determines that an extend operation is requested if an open curve is drawn on the cloud. A new mesh is then generated from the artist’s edit stroke based on the selected clouds type. This new mesh is then filled to create an addition to the pre-existing cloud. By adding to the cloud in this manner there is the possibility of creating an area of the cloud that is very dense with particles. This may cause issues, as overlapping particles may cause irregularities. To avoid this overlap, each particle of the pre-existing cloud is checked to see if it resides inside the new mesh and is deleted if this is the case. The new mesh is then filled using the accelerator grid.

The initial depth of the addition is determined by temporarily projecting the particles to the screen and comparing the start of the sketch to these projected particles. The closest particle to the start and the end of the sketch are used as depth markers for the new mesh.

4 Results

Figure 3, displays the cloud creation process from start to finish, the artist uses the canvas to select a proper location/altitude, and then proceeds to sketch their cloud. The cloud is then lit and displayed for the artist to observe and edit, all at interactive rates. The mesh used for filling is never made known to the artist.

In order to demonstrate the effectiveness of our interface we have implemented a simple one-pass light scattering algorithm. Although better lighting algorithms have been proposed [12,10,4,5,16], we have found this simplified lighting model produces reasonable results while permitting interactive frame-rates.

Figure 8 shows high level clouds created by our interface in just a few minutes. These high level clouds are created in our system using a flat and wide cross-sectional blending surface with larger, more spaced out particles. To our knowledge these types of clouds have not been attempted by any other sketch-based interface. Complex scenes such as these can be created without having to rigorously place shapes or containers.

Different particle types, based on the sub-types of the cloud, can be seen in figure 9; which also shows our technique being used on a pre-defined mesh. This allows complex forms to be used as the starting point for interactive editing (see Fig. 1).

All our algorithms run in real time on a 2GHz computer for particle groupings of less than 10,000 particles. Cloud edges can be improved by increasing the mesh resolution, however we use low resolution to facilitate speed when adding particles to the cloud. Since the mesh is used as a placement tool, we find that this sacrifice in accuracy is acceptable as the particles provide the variation that an artist may be trying to achieve (see Fig. 3). However, if more accurate clouds with respect to the sketch are required a one-time sacrifice in speed when filling the mesh would also be acceptable. The placement algorithm starts to slowdown

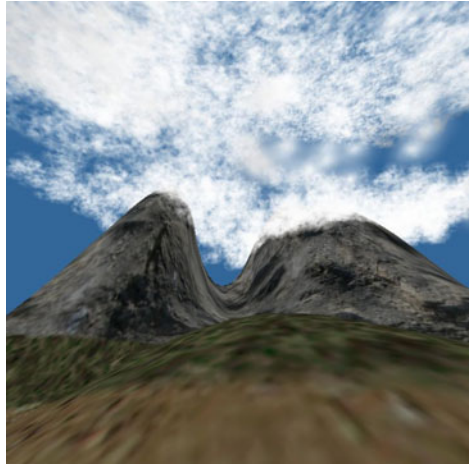
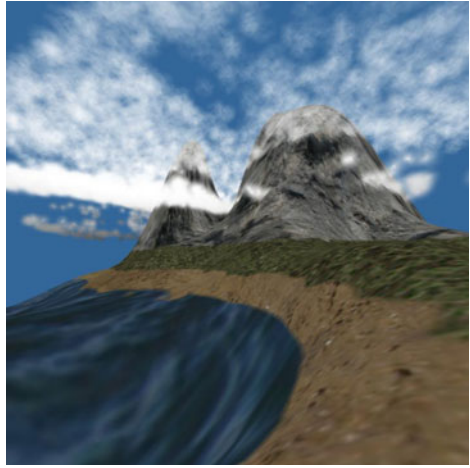


Fig. 8. Result created by our interface in a couple minutes, mostly high level clouds are displayed

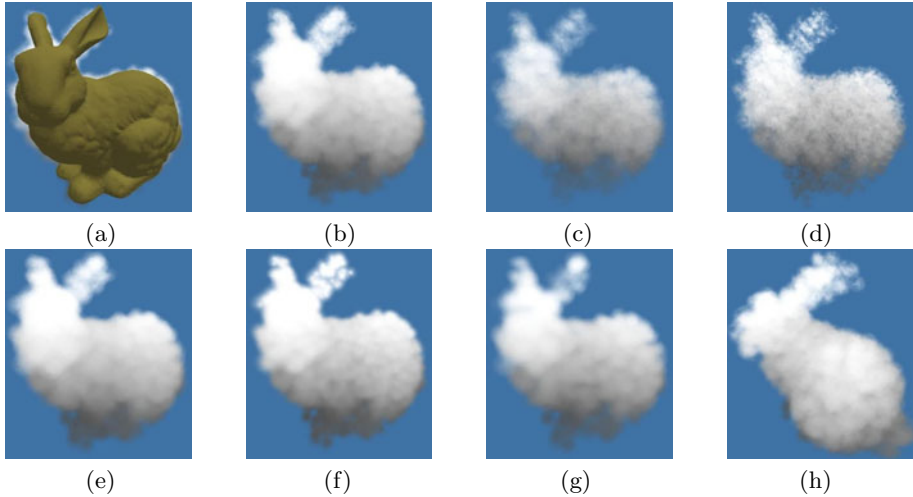


Fig. 9. (a) The Stanford bunny that has been filled with particles using our algorithm. (b-g) Different textures are used on the same particles to achieve different effects. (h) A rotated view of the formed cloud.

when the number of particles being placed is relatively large compared to the processing speed of the computer (we noticed the slowdown at around 10,000-20,000 particles on a 2.0GHZ computer). A reduction in this slow down may be accomplished by fine tuning the accelerator grid for more complex operations.

5 Conclusion

In this paper we have introduced a novel sketch based cloud modeling interface by use of particle placement and manipulation. The user draws a sketch and a mesh is then created using a rotational blending or cross-sectional blending surface. The mesh is populated with particles using an accelerator grid in conjunction with a triangle in-out test. Using this method we are able to place particles for clouds in a more efficient manner than was previously possible.

Our system is able to take a wide range of sketches as inputs, which are then interpreted based on sketch stylization and altitude. Basic Boolean operations also allow clouds to be manipulated once they have been created and placed. Such tools, to our knowledge, have not been proposed previously in the field of cloud modeling. The interface proposed by Wither et al [17] demonstrates some similarity to our interface, however we have included a methodology to create several different cloud types at interactive rates; as well as the ability to quickly define cloud fields. The content artists in games or flight simulators may benefit from the speed and flexibility of our approach when creating or manipulating a cloudy scene.

It is also worth noting that all the results presented in this paper were obtained without hardware acceleration and in general the proposed method operates in real time for reasonable sized clouds.

In our system, we emphasize an artistic interface for the creation and manipulation of clouds. The ease of using the interface provides a simple and manual yet direct approach for re-shaping and deforming the base cloud at different time points, thus providing a means for specifying the dynamics of clouds (See Fig. 10). However, a more intelligent system that supports the dynamics of cloud deformation by taking in to consideration the exact physical model may be a better approach. The challenge is how to provide a natural interface appropriate for artists to control this physical model and its parameters. Exploring sketch-based interfaces for interactive control of physically-based deformation of clouds is a topic for future work.

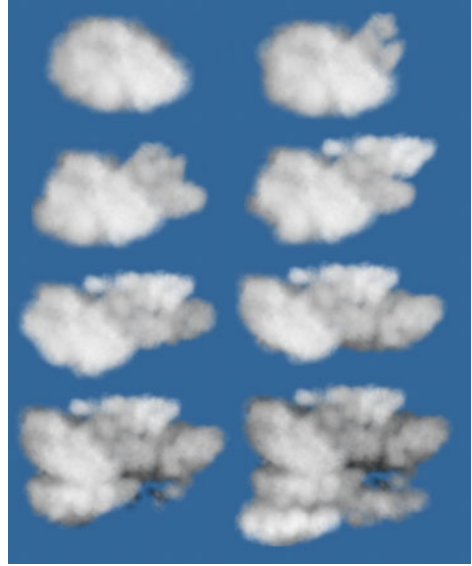


Fig. 10. A sequence of frames (left-to-right, top-to-bottom) manually created using our interface

6 Acknowledgments

We would like to thank Steve Longay for the use of his terrain in our images and Kipp Horel for helpful discussions regarding modifications to the user interface. We would also like to thank Gordon Richardson for the use of his cloud images. This research was supported in part by the National Science and Engineering Research Council of Canada and GRAND Network of Centre of Excellence of Canada.

References

1. Cherlin, J.J., Samavati, F., Sousa, M.C., Jorge, J.A.: Sketch-based modeling with few strokes. In: *SCCG 2005: Proceedings of the 21st spring conference on Computer graphics*, pp. 137–145. ACM, New York (2005)
2. Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., Nishita, T.: A simple, efficient method for realistic animation of clouds. In: *SIGGRAPH 2000: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 19–28. ACM Press/Addison-Wesley Publishing Co., New York (2000)
3. Dobashi, Y., Kusumoto, K., Nishita, T., Yamamoto, T.: Feedback control of cumuliiform cloud formation based on computational fluid dynamics. *ACM Trans. Graph.* 27(3), 1–8 (2008)

4. Harris, M.J., Lastra, A.: Real-time cloud rendering. In: Chalmers, A., Rhyne, T.-M. (eds.) EG 2001 Proceedings, vol. 20(3), pp. 76–84. Blackwell Publishing, Malden (2001)
5. Harris, M.J., Baxter, W.V., Scheuermann, T., Lastra, A.: Simulation of cloud dynamics on graphics hardware. In: HWWS 2003: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware, pp. 92–101. Eurographics Association, Aire-la-Ville (2003)
6. Igarashi, T., Hughes, J.F.: Smooth meshes for sketch-based freeform modeling. In: I3D 2003: Proceedings of the 2003, symposium on Interactive 3D graphics, pp. 139–142. ACM, New York (2003)
7. Igarashi, T., Matsuoka, S., Tanaka, H.: Teddy: a sketching interface for 3d freeform design. In: SIGGRAPH 1999: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, pp. 409–416. ACM Press/Addison-Wesley Publishing Co., New York (1999)
8. Kindersley, D.: Earth. Dorling Kindersley Ltd. (2003)
9. Levet, F., Granier, X.: Improved skeleton extraction and surface generation for sketch-based modeling. In: GI 2007: Proceedings of Graphics Interface 2007, pp. 27–33. ACM, New York (2007)
10. Liao, H.-S., Chuang, J.-H., Lin, C.-C.: Efficient rendering of dynamic clouds. In: VRCAI 2004: Proceedings of the 2004, ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry, pp. 19–25. ACM, New York (2004)
11. Mori, Y., Igarashi, T.: Plushie: an interactive design system for plush toys. In: SIGGRAPH 2007: ACM SIGGRAPH 2007 papers, p. 45. ACM, New York (2007)
12. Nishita, T., Dobashi, Y., Nakamae, E.: Display of clouds taking into account multiple anisotropic scattering and sky light. In: SIGGRAPH 1996: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 379–386. ACM, New York (1996)
13. Owada, S., Nielsen, F., Nakazawa, K., Igarashi, T.: A sketching interface for modeling the internal structures of 3d shapes. In: SIGGRAPH 2007: ACM SIGGRAPH 2007, courses, p. 38. ACM, New York (2007)
14. Schpok, J., Simons, J., Ebert, D.S., Hansen, C.: A real-time cloud modeling, rendering, and animation system. In: SCA 2003: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp. 160–166. Eurographics Association, Aire-la-Ville (2003)
15. Sealy, G., Novins, K.: Effective volume sampling of solid models using distance measures. In: CGI 1999: Proceedings of the International Conference on Computer Graphics, p. 12. IEEE Computer Society, Washington (1999)
16. Wang, N.: Realistic and fast cloud rendering. *Journal of graphics tools* 9(3), 21–40 (2004)
17. Wither, J., Bouthors, A., Cani, M.-P.: Rapid sketch modeling of clouds. In: Eurographics Workshop on Sketch-Based Interfaces and Modeling, SBM (2008)