Cartoon Animation Style Rendering of Water

Mi You[1], Jinho Park[2], Byungkuk Choi[1] and Junyong Noh[1]

[1]Graduate School of Culture Technology, KAIST, Republic of Korea
[2]Namseoul University, Republic of Korea

Abstract. We present a cartoon animation style rendering method for water animation. In an effort to capture and represent crucial features of water observed in traditional cartoon animation, we propose a Cartoon Water Shader. The proposed rendering method is a modified Phong illumination model augmented by the optical properties that ray tracing provides. We also devise a metric that automatically changes between refraction and reflection based on the angle between the normal vector of the water surface and the camera direction. An essential characteristic in cartoon water animation is the use of flow lines. We produce water flow regions with a Water Flow Shader. Assuming that an input to our system is a result of an existing fluid simulation, the input mesh contains proper geometric properties. The water flow lines can be recovered by computing the curvature from the input geometry, through which ridges and valleys are easily identified.

1 Introduction

With the recent development of computer graphics technology, manually created traditional animation is increasingly being replaced by computer-based cartoon style rendering. However, one problem with non-photorealistic rendering (NPR), in contrast with photorealistic rendering (PR), is excessive sacrifice of the details of individual materials and objects. For instance, a considerable amount of recent research on cartoon shading has focused on rather simple opaque objects [9][10][23]. Transparency, a common characteristic of water, is meanwhile often ignored. This treatment of water as an opaque object fails to meet the standard set by the results created by traditional cartoon artists.

In this work we present methods to draw cartoon style images that properly represent the particular characteristics of water. Cartoon water has abstract optical features such as transparency, reflection, and refraction. Our approach incorporates those features via a *Cartoon Water Shader*. Fig. 1 shows the optical features of water. Unlike previous methods that deal with only ambient and diffuse components [9] (see Fig. 6(a)), the proposed shader also accounts for specular components, which can represent the reflection or refraction effect using ray tracing. Moreover, traditional cartoon animation often shows a timely change between reflection and transparent refraction depending on the position of the viewpoint with respect to the surface of the water. We construct a similar mechanism for automatic selection of appropriate effects based on the angle between the normal vector of the water surface and the camera direction.

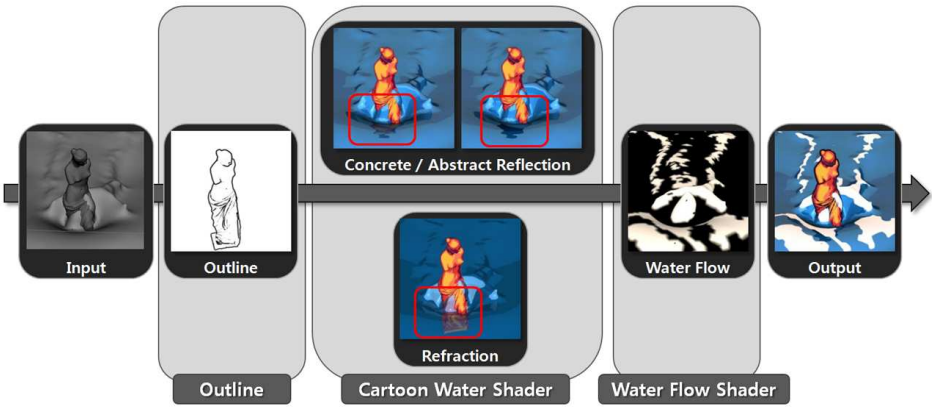


Fig. 1: An overview of our system. From the 3D input geometry, the bold outline is extracted. 'Cartoon Water Shader' is applied to the water input. The user can select from concrete reflection, abstract reflection or refraction. The difference of those effects is shown in the red box. 'Water Flow Shader' extracts the flow lines. These 3 components are combined to produce an output.

When depicting an object that moves across water, animators typically draw explicit flow lines to signify the motion waves generated by the movement of the object. In our approach, we produce water flow lines with a *Water Flow Shader*. The flow regions are geometric features and can be efficiently extracted by the computation of the curvatures on the water surface. Ridge and valley regions determined by the curvature computations represent the water flow regions, as shown in Fig. 1. This water flow region is incorporated into the result of the cartoon water shader.

2 Related Work

Diverse research in cartoon rendering has been reported to date. Todo et al. [23] provided user flexibility by adding localized light and shade using a paintbrush metaphor. Anjyo et al. [1] also proposed an intuitive and direct manipulation method for animating light and shade. In the latter, they mainly focus on the treatment of highlights. Mitchell et al. [17] applied their real-time shading method to commercial games. Barla et al. [2] suggested X-Toon, a shader that expands conventional cartoon shading. Various NPR techniques, including cartoon shading, pencil sketch shading, and stylistic inking, have also been developed [14]. As in cartoon rendering emphasizing abstraction, Gooch et al. [11] proposed a different style of abstraction that relies on interactive technical illustration. The abstraction is applied in [5], in which they develop a soft and abstract style of shadows similar to that seen in fine art.

Cartoon-style rendering of fluids from physical simulation has also attracted recent attention. McGuire and Fein [16] introduced a cartoon-style rendering

method for animated smoke particles. Selle et al. [21] proposed a cartoon style rendering technique for smoke simulation. Advected marker particles are rendered as texture-mapped 2D stencils. Instead of relying on a physical simulation, Yu et al. [24] present a template-based approach. They classify cartoon water into different types, and templates of water shapes are designed from the specified types of water. Running water is drawn in a Chinese painting style [25]. From the input videos, they generate the painting structure and the brush stroke.

One of the main characteristics of cartoon animation compared to recent 3D animation rendered by photorealistic rendering methods is the use of bold lines in the object boundary. There are several object-based approaches. Finding ridges and valleys from the geometry can help draw lines that describe the shape. Lee et al. [15] drew lines along tone boundaries around the thin dark areas in the shaded image with a GPU-based algorithm. Decarlo [6] developed a new type of line, a suggestive contour, which is determined by the zero crossings of the radial curvature. Curvature estimation plays an important role in extracting salient lines from the geometry. Judd et al. [12] estimated the view-dependent ridges and valleys from the curvature. Chandra and Sivaswamy [3] analyzed curvature based ridges and valleys represented in digital images. View-independent ridge and valley detection has also been proposed [19]. Meanwhile, demarcating curves, another new class of view-independent curves, are defined on the slopes between ridges and valleys [13].

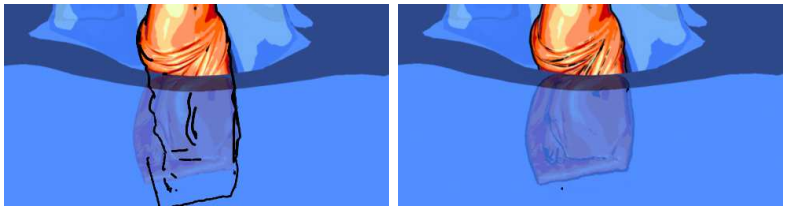
Our approach is most similar to the work in [9], which involves a cartoon style rendering of liquid animation. They utilize a bold outline, constant color, and oriented texture on the liquid surface. Although their method recovers many of the important features that water possess, they fail to address the artistic side of cartoon rendering such as abstract optical features and water flow lines observed in traditional cartoon animation. The proposed approach therefore tackles this.

3 Methods

The input to our system is a surface from a three-dimensional physically based fluid simulation. The surface mesh inherently contains the geometry information, which is useful in extracting water specific features for cartoon-style rendering. Commercial software, RealFlow is used for the generation of water simulation. However, any particle-based [4] or 3D grid-based solver [22] should work for our purpose.

3.1 Line Drawing

In traditional animation, lines are drawn mostly around the boundaries of objects to distinguish them from the background. Having the same aim as in traditional animation, we find the silhouette from the input meshes to draw desirable lines. We use the method suggested by [20]. Here, two identical polygons are combined into a set with slightly different scales, and each of the two polygons has a



(a) An image-based line drawing (b) An object-based line drawing

Fig. 2: Visual comparison of (a) and (b).(b) generates suitably distorted refractions when combined with ray tracing. (a) contains visual artifacts.

different culled face; the first is a front-facing polygon utilized for drawing the object itself, and the second is a back-facing polygon utilized for drawing the outlines. The scale difference between the two polygons creates the border lines around the object, as shown in [18][20]. As our line drawing is based on objects, our line objects also easily generate distorted refractions when combined with ray tracing (Fig. 2).

3.2 Cartoon Water Shader

Modified Phong for Cartoon Style Adjusting the Lambertian illumination model [14] effectively generates cartoon style shading for scenes with opaque objects. Their cartoon shading equation consists of two main terms, ambient and diffuse terms, which return the largest value between $\overline{L} \cdot \overline{n}$ and 0. This value is used for dividing two colors as texture coordinates. Their cartoon shading equation is:

$$I_{cs} = I_a k_a O_d + I_d[k_d Max(\overline{L} \cdot \overline{n}, 0)] \tag{1}$$

Here, $\overline{L}$ is the normalized direction of light source and $\overline{n}$ is the normalized direction of surface normal.

For opaque objects, their equation is sufficient to create a cartoon style image. However, water exhibits three peculiar characteristics: transparency, refraction, and reflection. Successful incorporation of these features helps convey realism, even in cartoon-style water animation.

To create transparently refractive effects, we adapt the Phong illumination model, modify $\overline{L} \cdot \overline{n}$ to O_d divided by 3 colors and add ray-tracing terms. The final illumination of the Cartoon Water Shader, I_{cws}, is obtained as follows:

$$I_{cws} = I_a k_a O_d + I_d[k_d O_d + k_s O_s(\overline{R} \cdot \overline{V})^n] + k_s I_r + k_t I_t \tag{2}$$

Here, I_a and I_d are the intensity of the ambient and diffuse light. k_a, k_d, and k_s are the ambient, diffuse, and specular coefficient, respectively. O_s is an object's specular color. $\overline{R}$ is the normalized direction of reflection and $\overline{V}$ is the normalized viewpoint direction. $k_s I_r$ and $k_t I_t$ are the intensity of the reflected and transmitted ray for ray tracing. Fig. 3 (a) shows the refraction of Venus under water.

To create a cartoon style shader, the object color is simplified into three levels: bright, medium, and dark. We evaluated different numbers of levels and determined that three levels produced the most convincing results, as too many levels approached photorealistic rendering and too few levels resulted in an indistinguishable appearance. The level is determined by the angle between the surface normal direction and the light direction, as follows:

$$O_d = \begin{cases} \text{Bright Color} & \text{if } \overline{L} \cdot \overline{n} > T_{bc} \\ \text{Medium Color} & \text{if } \overline{L} \cdot \overline{n} > T_{mc} \text{ and } \overline{L} \cdot \overline{n} \leq T_{bc} \\ \text{Dark Color} & \text{Otherwise} \end{cases} \tag{3}$$

Here, T_{bc} and T_{mc} are the threshold for a bright color and a medium color, respectively.

We distinguish two types of reflection. The first is concrete reflection, which shows an ordinary reflection effect commonly observed in the real world. The second is abstract reflection. The latter is also considered important in the context of cartoon animation. We allow both types of reflection in order to serve the artist's intentions depending on the situation. Rendering the two types of reflection is similar to the line drawing method described in Section 3.1. Concrete reflection is generated by setting the front-facing polygons as the reflection target of an object. In contrast, abstract reflection is produced by setting the back-facing polygons as the reflection target. Fig. 3 (b) and (c) show both effects reflected from Venus on a water surface.

(a) Refraction under water (b) Concrete reflection (c) Abstract reflection

Fig. 3: Refraction effect of Venus (a) and reflection effect (b)(c) in the Cartoon Water Shader.

Automatic Control of Refraction and Reflection Effect In traditional animation, for simplicity artists tend to employ either a reflection or refraction effect for a given scene, unlike the real world where both effects coexist. Artists utilize the refraction effect when the distance between the camera and the main character and the angle created by the two are small. Otherwise, the reflection effect is employed.

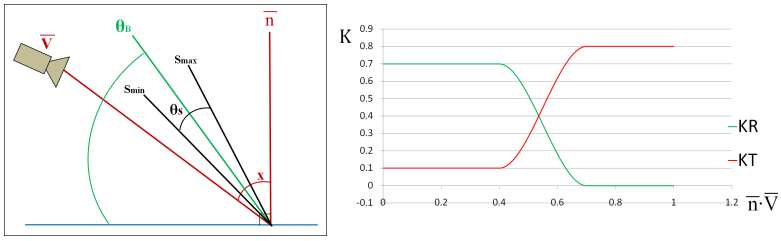
Dynamic camera movements in a 3D scene may result in frequent transitions between the reflection effect and refraction effect causing unwanted flickering.

Moreover, it would be very time consuming to determine a desirable effect according to the dynamic camera movements. Therefore, a simple but effective interpolation function that automatically determines when to apply reflection or refraction is proposed here. As transparency directly influences the visibility of refraction, we utilize transparency in the following equations.

$$k_R = \begin{cases} k_{Rmax} & \textbf{if} \quad x < \cos s_{max} \\ k_{Rmin} & \textbf{if} \quad x > \cos s_{min} \\ f(\overline{n} \cdot \overline{V}, k_{Rmax}, k_{Rmin}) & \textbf{Otherwise} \end{cases} \tag{4}$$

$$k_T = \begin{cases} k_{Tmin} & \textbf{if} \quad x < \cos s_{max} \\ k_{Tmax} & \textbf{if} \quad x > \cos s_{min} \\ f(\overline{n} \cdot \overline{V}, k_{Tmax}, k_{Tmin}) & \textbf{Otherwise} \end{cases} \tag{5}$$

Here, $f(\overline{n} \cdot \overline{V}, k_{Rmax}, k_{Rmin})$ and $f(\overline{n} \cdot \overline{V}, k_{Tmax}, k_{Tmin})$ are cubic polynomial interpolation functions. $\overline{n}$ is the normal direction and $\overline{V}$ is the viewpoint direction, respectively. k_{Rmax}, k_{Rmin}, k_{Tmax}, and k_{Tmin} are the maximum reflection, the minimum reflection, the maximum transparency, and the minimum transparency coefficient, respectively. In Fig. 4(a), θ_B is the angle where the critical change occurs and θ_S is the angle interval where interpolation happens. A user can specify k_{Rmax}, k_{Rmin}, k_{Tmax}, k_{Tmin}, θ_B, and θ_S at the key camera positions to produce desirable reflection and refraction effects. k_R and k_T are then computed automatically along the scene compositions. Fig. 4(b) shows each parameter of the control function and the shift of the reflection and transparency coefficients according to $\overline{n} \cdot \overline{V}$.



(a) The parameters used in control function

(b) The shift of the reflection and transparency coefficients according to $\overline{n} \cdot \overline{V}$

Fig. 4: The parameters and the behavior of the of control function

Although utilizing well-known physics such as the Fresnel reflection and Brewster's angle can provide a viable solution, the primary goal of our shader is to render a scene in non-photorealistic style. This cubic polynomial equation works well for our purpose.

3.3 Water Flow Shader

In traditional animation, special lines are drawn in the neighborhood of objects to effectively show the interaction between water and objects. In our simulation results, those shapes represent the flowing motions of water and traces of the objects. To emphasize these motions and traces, we introduce a Water Flow Shader. With the Water Flow Shader, the critical regions from flowing motions and traces are determined using the geometrical properties from input meshes. As the shader takes into account important geometric features, our results faithfully reproduce physically persuasive motions and appearances.

Estimation of Geometrical Properties We compute both the Gaussian curvature and the mean curvature at every vertex of the triangular mesh to identify the critical regions on the mesh surface, as the curvature on the region formed by the movement of objects has a higher value than other flat regions. Using a discretization of the Gauss-Bonet theorem, we approximate the Gaussian curvature as described in [8]. The equation is as follows:

$$K_i = \frac{1}{A}\left(2\pi - \sum_j \theta_j\right) \tag{6}$$

Here, A is the Voronoi area around X_i, and θ_j is an angle connected with X_i and X_j.

The mean curvature is approximated using a discretization of the Laplace-Beltrami operator also known as the mean curvature normal operator. A detailed explanation can be found in the literature [8][7]. The equation is given as follows:

$$H_i = \frac{1}{4A}\sum_j (\cot\alpha_{ij} + \cot\beta_{ij})(X_i - X_j)\cdot\hat{n_i} \tag{7}$$

Here, A is the Voronoi area. α_{ij} and β_{ij} are angles adjacent to the specified vertices, and X_j is a neighbor vertex connected to X_i. $\hat{n_i}$ is a normal vector at X_i. Both approximations are applied to the one-ring neighborhood of the vertex.

From the recovered Gaussian curvature and mean curvature, we specify a ridge area connected by the vertices in which the absolute value of the Gaussian curvature is larger than the threshold and the mean curvature is greater than zero. Although our approach to calculate the curvatures and a rendering style are different, our concept of specifying the regions is similar to the principle of demarcating curves [13].

Enhancement After determining the ridge and valley region, we apply an enhancement algorithm. As the computation of ridges and valleys heavily depends on the input meshes, bad geometry in the meshes can adversely affect the result. For example, if the mesh was created by a particle simulation that represents the details of the fluid well but introduces jiggling surfaces, our method will

(a) A fountain in a forest (b) A boat sailing on the sea (c) An image created by an artist (d) An image created by an artist

Fig. 5: The images rendered by our system and the images used in the experiments. Total rendering time of (a) is 8.0 sec(C.W.S: 8.0 / W.F.S: n/a / number of vertices: 75406) and that of (b) is 1330.0 sec(C.W.S: 5.0 / W.F.S: 1325.0 / number of vertices: 55091).

draw lines erroneously. Through application of the *median filter* to the one-ring neighbor of each vertex, we can effectively suppress these errors.

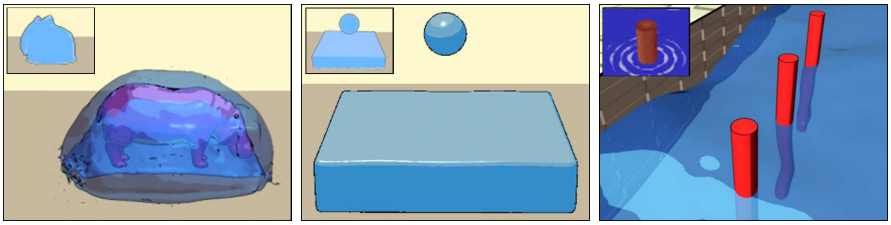
In traditional animation, special regions to be lined on the surface of water are opaque and monochromic. To generate these images, we divide curvature values into three parts. The first is for ridges, the second for valleys, and the third is the part between ridges and valleys. We paint ridge or valley parts as shaded and the other parts as transparent. As a result, the water flow region is separated from the other parts clearly. As the water flow region can be efficiently represented by simple shading, a simple Lambertian illumination model is sufficient for rendering.

4 Results and Analysis

To demonstrate the effectiveness of our system, the Cartoon Water Shader and the Water Flow Shader were created as a Maya plug-in (version 2009). The plug-in system was implemented in C++ with the Maya API. Moreover, to maximize the efficiency of our plug-in, we provide a user interface.

The images in Fig. 5 show the results rendered by our system. Images (a) and (b) clearly demonstrate that our system can produce high quality cartoon animation water rendering. Additional results are available in demo clips(see http://143.248.249.138/isvc09/paper66/). Rendering times of (a) and (b) are recorded on an Intel Core 2 Quad Q6600 machine with 4GB memory, using a single thread implementation. As the computation time of the Water Flow Shader depends on the number of vertices with the curvature approximations and the enhancement procedure, much more time is required to render the scene via the Water Flow Shader than with the Cartoon Water Shader(see Fig. 5).

To verify the visual enhancement of our method over previous approaches, we compare similar scenes with results rendered by earlier methods. In addition,



(a) A hippopotamus covered by water

(b) A ball of liquid

(c) Immersed poles near an embankment

Fig. 6: Visual comparison with two earlier approaches. Insets are the images from the previous methods.

an evaluation was conducted to verify that our results are visually equivalent to those created by an artist.

4.1 Visual Comparison with Previous Approaches

Among the studies on cartoon rendering of fluids, two prominent studies [9][24] were selected, as their main subject is similar to ours.

Eden A. M. et al. successfully recovered many crucial visual cues generated by the movement of a fluid [9]. However, their method is absent of optical features (Fig. 6 (a)). In contrast, our method shows a refracted view of a hippopotamus inside water. As we render the shading of an object after calculating the surface normal and the light information, our method can flexibly cope with the light position. With our method, the middle image in Fig. 6 is shaded following the light direction. In Eden A. M. et al. [9], lighting information is not considered.

As an approach of Yu j. et al.[24] is template-based, their images are similar to the drawing of an artist. However, the water types that they can represent are limited. With this method it is difficult to express the relationship of the shading of water, which is affected by the environment, with reflections of nearby objects on the water or transparent refractive shapes under the water. On the other hand, our image showing immersed poles near an embankment (Fig. 6); accurately reflects all objects near the water. These reflective images are changed following the variation of the mesh and the light.

4.2 Evaluation

An experiment was conducted to confirm that our results are visually equivalent to those created by an artist.

Participants 30 participants took part in the evaluation. Among the participants, 15 were experts who work in the field of art and graphics and are considered to have excellent observation capability of images. The remaining participants were non-specialists who were nevertheless familiar with cartoon animation.

Material To collect the images for comparison, a 3D artist was asked to create scenes similar to those in a feature animation. We chose two concepts: a fountain in a forest and a boat sailing on the sea. These selected concepts were visualized as a 3D scene by a 3D artist. Based on the images rendered by the Lambertian illumination model, a cartoon animation artist composed water in the style of cartoon animation. Our instructions were to represent the artist's sense inside the outline of water and to follow the motion of the water mesh. However, we did not provide instructions concerning artistic flavors such as the properties of shading, the color to use, or which points the water flow lines should represent. These restrictions allowed the artist to draw freely, tapping into her own ingenuity. All of the images for the test were organized into 12 drawings per second, as in the style of limited animation usually employed in traditional Japanese animation. Images in the experiments were scaled to 640×480, which is the same scale used for the demo clip.

Analysis P-values for the two-sided hypothesis testing problem were computed using an independent-samples t-test.

Procedure Each participant was informed that the only variation between each set of video clips was the shading of water. Participants observed four video clips two times each in a random order. Two clips depicted 'a fountain in a forest'; the first was created by our method, and the second was drawn by the artist. The other two clips depicted clips of 'a boat sailing on the sea'; the first was created by our method, and the second was drawn by the artist. After observation of four video clips, the participants were asked to rate the images from 1(Bad) to 5(Good). The first question was to evaluate the aesthetic impression of the video clips according to the participant's own standards. The second question was to rank the clips according to whether they offered a rich depiction of the water.

Results and Discussion For the first question, participants answered that the aesthetic impression of Fig. 5 (a) and Fig. 5 (c) were similar to each other as $Mean_{artist} = 3.17$, $Mean_{ours} = 3.07$. However, in the set of 'a boat sailing on the sea' clips, the video clip generated by our method in Fig. 5 (b) was rated to lend a higher aesthetic impression than the results by the artist ($p < 0.01$)($Mean_{artist} = 3.03$ vs $Mean_{ours} = 4.03$).

For the second question, most participants answered that Fig. 5 (a) and Fig. 5 (c) have similar values, implying that both clips exhibit a rich depiction of the water, as $Mean_{artist} = 3.20$, $Mean_{ours} = 3.21$. In comparison of Fig. 5 (b) and Fig. 5 (d), $M_{ours} = 4.20$ is better than $Mean_{artist} = 3.80$. In conclusion, the evaluation demonstrates that our results match those created by an artist in terms of visual equivalence.

5 Conclusion and Discussion

The contribution of our system includes the proposed Cartoon Water Shader which can convey the essential components of water -transparency, refraction, and reflection- from an artistic point of view using a modified Phong illumination model. Automatic change between refraction and reflection, frequently

observed effects in traditional cartoon animation, is also offered in the shader. Furthermore, utilizing the information of geometrical properties, Water Flow Shader is specifically designed for rendering water flow observed in traditional cartoon animation. To demonstrate the advantages of our method, we evaluated rendering results in comparison with those yielded by previous approaches and by an artist.

Although the proposed method can represent many characteristics of cartoon animation style rendering of water, there remain issues that must be addressed in the future. As we assume that the input meshes do not contain spray or bubbles, our current system cannot reproduce them. Another limitation involves the regions that cannot be described by the geometry properties alone in the Water Flow Shader. Although our system provides some thresholds for adjusting the effects to meet the artist's demands, exaggerated expressions of water flow lines are difficult to produce. We believe that these problems should be investigated in the future for a greater audience reception.

References

1. Anjyo K. I., Wemler S., Baxter W.: Tweakable light and shade for cartoon animation. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2006), ACM, pp. 133–139.
2. Barla P., Thollot J., Markosian L.: X-toon: an extended toon shader. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2006), ACM, pp. 127–132.
3. Chandra S., Sivaswamy J.: An analysis of curvature based ridge and valley detection. In *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on* (Los Alamitos, CA, USA, 2006), IEEE Computer Society Press, p. 2.
4. Desbrun M., Cani M.-P.: Smoothed particles: A new paradigm for animating highly deformable bodies. In *Eurographics Workshop on Computer Animation and Simulation (EGCAS)* (Aug 1996), Boulic R., Hegron G., (Eds.), Springer-Verlag, pp. 61–76. Published under the name Marie-Paule Gascuel.
5. DeCoro C., Cole F., Finkelstein A., Rusinkiewicz S.: Stylized shadows. In *NPAR '07: Proceedings of the 5th international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2007), ACM, pp. 77–83.
6. DeCarlo D., Finkelstein A., Rusinkiewicz S., Santella A.: Suggestive contours for conveying shape. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), ACM, pp. 848–855.
7. Desbrun M., Meyer M., Schröder P., Barr A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 317–324.
8. Desbrun M., Meyer M., Schroder P., Barr A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. Springer-Verlag, pp. 35–57.
9. Eden A. M., Bargteil A. W., Goktekin T. G., Eisinger S. B., O'Brien J. F.: A method for cartoon-style rendering of liquid animations. In *GI '07: Proceedings of Graphics Interface 2007* (New York, NY, USA, 2007), ACM, pp. 51–55.

10. Gooch A., Gooch B., Shirley P., Cohen E.: A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM, pp. 447–452.
11. Gooch B., Sloan P.-P. J., Gooch A., Shirley P., Riesenfeld R.: Interactive technical illustration. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 1999), ACM, pp. 31–38.
12. Judd T., Durand F., Adelson E.: Apparent ridges for line drawing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 19.
13. Kolomenkin M., Shimshoni I., Tal A.: Demarcating curves for shape illustration. In *SIGGRAPH Asia '08: ACM SIGGRAPH Asia 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–9.
14. Lake A., Marshall C., Harris M., Blackstein M.: Stylized rendering techniques for scalable real-time 3d animation. In *NPAR '00: Proceedings of the 1st international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2000), ACM, pp. 13–20.
15. Lee Y., Markosian L., Lee S., Hughes J. F.: Line drawings via abstracted shading. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 18.
16. McGuire M., Fein A.: Real-time rendering of cartoon smoke and clouds. In *NPAR '06: Proceedings of the 4th international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2006), ACM, pp. 21–26.
17. Mitchell J. L., Francke M., Eng D.: Illustrative rendering in team fortress 2. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses* (New York, NY, USA, 2007), ACM, pp. 19–32.
18. Markosian L., Kowalski M. A., Goldstein D., Trychin S. J., Hughes J. F., Bourdev L. D.: Real-time nonphotorealistic rendering. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 415–420.
19. Ohtake Y., Belyaev A., Seidel H.-P.: Ridge-valley lines on meshes via implicit surface fitting. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers* (New York, NY, USA, 2004), ACM, pp. 609–612.
20. Raskar R., Cohen M.: Image precision silhouette edges. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics* (New York, NY, USA, 1999), ACM, pp. 135–140.
21. Selle A., Mohr A., Chenney S.: Cartoon rendering of smoke animations. In *NPAR '04: Proceedings of the 3rd international symposium on Non-photorealistic animation and rendering* (New York, NY, USA, 2004), ACM, pp. 57–60.
22. Stam J.: Stable fluids. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 121–128.
23. Todo H., Anjyo K.-I., Baxter W., Igarashi T.: Locally controllable stylized shading. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 17.
24. Yu J., Jiang X., Chen H., Yao C.: Real-time cartoon water animation. *Comput. Animat. Virtual Worlds 18*, 4-5 (2007), 405–414.
25. Zhang S., Chen T., Zhang Y., Hu S., Martin R.: Video-based running water animation in chinese painting style. *Science in China Series F: Information Sciences 52*, 2 (February 2009), 162–171.