# Toon Rendering of Water Animation

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. 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 [She07]. 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 [TABI07] [GGSC98] [EBG*07]. 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*. Figure 1 shows the optical features of water. Unlike previous methods that deal with only ambient and diffuse components [EBG*07], the proposed shader also accounts for specular components, which can represent the reflection or refraction effect using ray tracing.

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 Figure 1. This water flow region is incorporated into the result of the cartoon water shader.

## 2. Method

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.

### 2.1. Line Drawing

For the artist, the line has been the major component in the recognition and representation of objects in the world [Gom95]. 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 [RC99]. Here, two identical polygons are combined into a set with slightly different scales, and each of the two polygons has a different culled face; the first is a front-facing polygon utilized for drawing the object itself, and the second is a back-

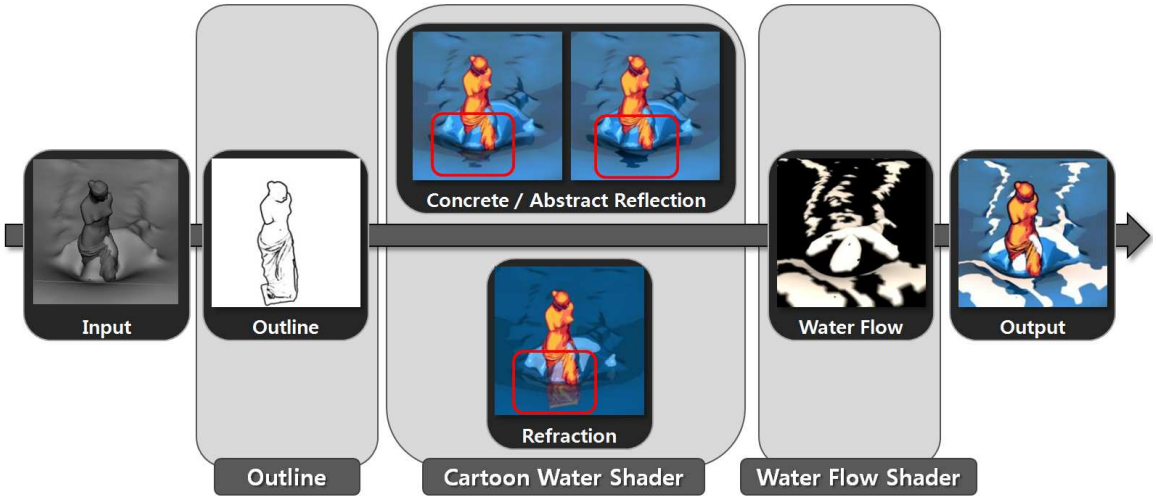


**Figure 1:** *An overview of our system. From the 3D input geometry, the bold outline is extracted. 'Cartoon Water Shader' is applied to the water input. The user can select from concrete reflection, abstract reflection or refraction. The difference of those effects is shown in the red box. 'Water Flow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

facing polygon utilized for drawing the outlines. The scale difference between the two polygons creates the border lines around the object, as shown in [RC99] [MKG*97]. As our line drawing is based on objects, our line objects also easily generate distorted refractions when combined with ray tracing.

### 2.2. Cartoon Water Shader

#### 2.2.1. Modified Phong for Cartoon Style

Adjusting the Lambertian illumination model [LMHB00] 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:

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

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:

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

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. 2 (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:

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

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 2.1. Concrete reflection is generated by setting the front-facing polygons as the reflection target of an object. In contrast, abstract reflec-

tion is produced by setting the back-facing polygons as the reflection target. Fig. 2 (b) and (c) show both effects reflected from Venus on a water surface.

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

**Figure 2:** *Refraction effect of Venus (a) and reflection effect (b)(c) in the Cartoon Water Shader.*

### 2.3. Water Flow Shader

In traditional animation, special lines are drawn in the neighborhood of objects to effectively show the interaction between water and objects. The interactions between objects and the surface of water create certain shapes such as wiggles and concentric circles. 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. In the following, a sufficiently wide bounding box that contains vertices with displacements is first selected for efficient computation.

#### 2.3.1. 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 [DMSB02]. The equation is as follows:

$$K_i = \frac{1}{A}\left(2\pi - \sum_j \theta_j\right) \tag{4}$$

Here, $A$ is the Voronoi area around $X_i$, and $\theta_j$ 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 [DMSB99] [DMSB02]. The equation is given as follows:

$$H_i = \frac{1}{4A}\sum_j (\cot\alpha_{ij} + \cot\beta_{ij})(X_i - X_j)\cdot\hat{n}_i \tag{5}$$

Here, $A$ is the Voronoi area. $\alpha_{ij}$ and $\beta_{ij}$ 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. On the other hand, the valley is the area in which the absolute value of the Gaussian curvature is also larger than the threshold and the mean curvature is smaller 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 [KST08].

## 3. Result

The images in Figure 3 show the results rendered by our system. In image (a), the water dog shows refraction of bones and muscles inside its body. Images (b) and (c) clearly demonstrate that our system can produce high quality cartoon animation water rendering.

| | # of vertices | C.W.S | W.F.S | Total |
|---|---|---|---|---|
| (a) | 59988 | 3.0 | n/a | 3.0 |
| (b) | 75406 | 8.0 | n/a | 8.0 |
| (c) | 55091 | 5.0 | 1325 | 1330.0 |

**Table 1:** *Columns C.W.S and W.F.S show the rendering time (in seconds) using the Cartoon Water Shader and the Water Flow Shader, respectively.*

Table 1 shows the rendering time for the results in Figure 3. All data 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, much more time is required to render the scene via the Water Flow Shader than with the Cartoon Water Shader.

## 4. Conclusion

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. Furthermore, utilizing the information of geometrical properties, Water Flow Shader is specifically designed for rendering water flow observed in traditional cartoon animation.

## 5. Acknowledgements

We would like to thank all the members of the KAIST Visual Media Laboratory for their help with useful comments.

**(a)** *A semi transparent water dog* **(b)** *A fountain in a forest* **(c)** *A boat sailing on the sea*

**Figure 3:** *The images rendered by our system*

This work was supported by IT R&D program of MIC/IITA (ITAA1100070400010001000300200, Software Development for Digital Creature).

## References

[DMSB99] 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.

[DMSB02] DESBRUN M., MEYER M., SCHRODER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. Springer-Verlag, pp. 35–57.

[EBG*07] 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.

[GGSC98] 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.

[Gom95] GOMBRICH E.: *The Story of Art*. Phaidon Press, 1995.

[KST08] 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.

[LMHB00] 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.

[MKG*97] 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.

[RC99] 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.

[She07] SHEN L. F.: What is "computer animation"?: examining technological advancements and cultural aesthetics of japanese animation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 educators program* (New York, NY, USA, 2007), ACM, p. 23.

[TABI07] 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.