Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Procedural modeling of water caustics and foamy water for cartoon animation*

Jing LIAO[†1], Jin-hui YU[†‡1], Long JIA[2]

([1]*State Key Laboratory of CAD & CG, Zhejiang University, Hangzhou 310027, China*)

([2]*School of Information Technology and Mathematical Sciences, University of Ballarat, Ballarat 3350, Australia*)

[†]E-mail: {Liaojing, jhyu}@cad.zju.edu.cn

Received June 29, 2010; Revision accepted Oct. 29, 2010; Crosschecked May 5, 2011

Abstract: We propose a method for procedural modeling and animation of cartoon water effects such as water caustics, foamy wake, and longshore currents. In our method we emulate the visual abstraction of these cartoon effects by the use of Voronoi diagrams and the motion abstraction by designing relevant controlling mechanisms corresponding to each effect. Our system enables the creation of cartoon effects with minimal intervention from the animator. Through high-level initial specification, the effects are animated procedurally in the style of hand-drawn cartoons.

Key words: Procedural modeling, Water effects, Non-photorealistic rendering, Cartoon animation
doi:10.1631/jzus.C1000228 **Document code:** A **CLC number:** TP391

1 Introduction

Abstraction in art indicates a departure from reality in the depiction of imagery. Cartoon animation as an art of abstraction has two dimensions: motion abstraction and visual abstraction (Bregler *et al*., 2002). In the context of cartoon water animations, water motion abstraction depends on various conditions such as flow rates, interactions between barriers, and wind conditions. Fig. 1 shows some hand-drawn water effects taken from commercial animations, including water caustics seen from underwater in *Ponyo on the Cliff*©Studio Ghibli (Fig. 1a), the foamy wake left behind by a moving ship in *The Legend of Nezha*©China Central Television (Fig. 1b), and longshore currents in *Ichigo Mashimaro*©Daume (Fig. 1c).

In this paper we propose an approach to modeling and rendering these cartoon water effects. Our method is based on observations that hand-drawn effects shown in Fig. 1 closely resemble Voronoi diagrams (see Section 3 for details) in structure; that is, the visual abstraction of these effects can be characterized by the Voronoi diagram. We therefore adopt the Voronoi diagram to emulate the structures of these hand-drawn effects and design relevant rendering schemes and controlling mechanisms to animate them procedurally.

(a) (b) (c)

Fig. 1 Hand-drawn water effects
(a) Water caustics; (b) Foamy wake; (c) Longshore currents

2 Related works

In realistic water animation, many techniques have been proposed during the past three decades. In a survey of techniques for realistic visualization of

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 60933007) and the Key Technologies R&D Program of China (No. 2007BAH11B02)

fluids, Adabala and Manohar (2002) addressed issues of representing fuzzy fluid geometry, capturing the dynamic behavior, and modeling the light interaction with the fluids. Iglesias (2004) made a historic survey on the most relevant computer graphics techniques developed during the 1980s and 1990s for realistic modeling, rendering, and animation of water. Although complex and visually intricate water animations can be achieved with previous techniques for realistic water animations, it is difficult to incorporate the motion abstraction and visual abstraction of cartoon water animation into these techniques.

In cartoon water animation, only a few methods have been proposed during the past few years. di Fiore *et al.* (2004) proposed a system capable of generating flames, drops, and smoke puffs with a hand-drawn style, in which hand-drawn visual elements are animated along the user-defined path. Thornton (2006) presented a particle based method to produce splash effects, by creating a modular rig composed of a series of nodes to describe the shape of the splash. In the resultant animation the particle usage is obvious and dominant. A fluid based approach was proposed by Eden *et al.* (2007) in which a 3D liquid surface simulated physically is rendered with bold lines and flat colors. In this approach the main concern is on the cartoon style rendering, and the dynamics of the fluid is realistic. Yu *et al.* (2007) used a template based method to model the cartoon water effects including flowing water, ripples, and water jet. Later on, Yu *et al.* (2008) extended the work to handle the interaction between objects and cartoon water by introducing a grid-based controlling mechanism. Although Yu's methods preserve the cartoon look well in the water animations generated, the templates they constructed are not suitable for modeling cartoon water effects with Voronoi structures. Kwatra *et al.* (2007) and Narain *et al.* (2007) synthesized textures over dynamically changing fluid surfaces. Visually the synthesized textures look different from the hand-drawn cartoon water in style. Recently, You *et al.* (2009) proposed a cartoon water shader by modifying the Phong illumination model augmented by the optical properties that ray tracing provides. Such a shader contributes only to the cartoon shading and the rendered water lacks the liveliness and energy in hand-drawn cartoon water. To date, to our knowledge, there are no studies on the water caustics and foamy water for cartoon animation.

3 Voronoi diagrams and Delaunay triangulations

Voronoi diagrams were first discussed by Peter Lejeune Dirichlet in 1850. However, it was more than half a century later in 1908 when these diagrams were written about in a paper (Voronoi, 1908), hence the name 'Voronoi diagrams'.

Given n distinct points (sites) $P=\{p_1, p_2, ..., p_n\}$ in the plane, the Voronoi polygon (cell) of a point p_i, $\mathrm{VP}(p_i)$, is defined to be the set of all points q in the plane for which p_i is among the closest points to q in P. That is,

$$\mathrm{VP}(p_i)=\{q:|p_i-q|\le|p_j-q|, i\ne j\}. \quad (1)$$

The union of the boundaries of the Voronoi polygon is called the Voronoi diagram of P, denoted by $\mathrm{VD}(P)$, as shown by solid lines in Fig. 2. A Voronoi edge is the boundary between two Voronoi polygons and a Voronoi vertex is the intersection of three or more Voronoi edges.

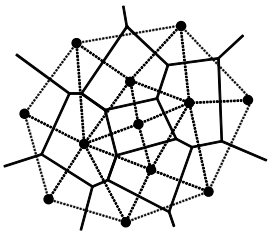

Fig. 2 Voronoi diagram (solid line) and Delaunay triangulation (dotted line)

The dual of the Voronoi diagram is a triangulation of the point set, called the Delaunay triangulation (dotted lines in Fig. 2), which is a collection of edges satisfying an empty circumcircle property: for each edge we can find a circle containing the edge's endpoints but not containing any other points in P.

There are many algorithms for constructing the Voronoi diagram (Aurenhammer, 1991), including the calculation of Voronoi diagrams using graphics hardware (Hoff *et al*., 1999). We adopt the simple algorithm by Lischinski (1994) for the incremental construction of the Delaunay triangulation and the Voronoi diagram. For implementation details of the algorithm, please refer to Lischinski (1994).

The $\mathrm{VD}(P)$ constructed above models only underlying spatial structures of the cartoon effects shown in Fig. 1. To animate these cartoon effects we

need to incorporate appropriate rendering and controlling mechanisms into VD(P). The main idea in our method is to construct time-varying textures of structure as VD(P) for varying effects shown in Fig. 1 and map the texture onto the water surface. To handle the more complex foamy wake and longshore currents, we first construct some reference curves that capture features of effects regions. We then use these curves to guide the placement of Voronoi textures on the texture image and deform relevant parts on the water surface. The texture image is finally mapped onto the deformed water surface to obtain a frame of animation. In the next three sections we describe our rendering schemes and controlling mechanisms for the water caustics seen from under water, foamy wake, and longshore currents.

4 Water caustics seen from underwater

Water caustics result from light rays reflecting or refracting from a water surface and hence focusing only in certain areas of the surface. The original hand-drawn animation in Fig. 1a represents the effect of water caustics seen from under water. In the figure curvy lines are drawn as caustic textures on the water surface, and the light scattering effect is depicted by the scattered light near the sun and a few scattering beams in the deep water.

Fig. 3 illustrates our model for water caustics. First, we generate a Voronoi diagram on a texture image to emulate the structure of water caustic textures in Fig. 1a, and render Voronoi edges with curvy lines to obtain water caustic textures. The texture image is mapped onto the flat water surface laid horizontally in the scene (Fig. 3a). Next, we project the scene composed of the water surface and the background image surface onto the view plane at a view point below the water surface (Fig. 3b). On a separate image we generate the light scattering effect (Fig. 3c) and then blend it with the former projected image on the view plane to obtain the final result (Fig. 3d). The following three subsections describe our model in detail.

4.1 Water caustic textures

We first use sites with uniform distribution to construct a Voronoi diagram VD(P) on a texture image. Next we render Voronoi edges with curvy lines. On each Voronoi edge, we take its two end points (Voronoi vertices) V_1 and V_2 as extremes and introduce two intermediate points P_1 and P_2 by the following formula:

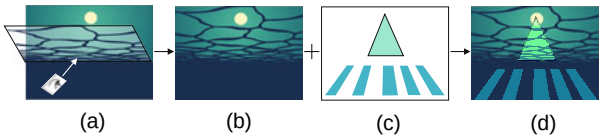

Fig. 3 Water caustics model

(a) A scene composed of the water surface and the background image surface; (b) Projection of the scene on the view plane; (c) Image of light scattering effect; (d) Blending of the projected image and light scattering effect

$$\begin{cases} P_1 = V_1 + u_1(V_2 - V_1) + \Delta, \\ P_2 = V_1 + u_2(V_2 - V_1) + \Delta, \end{cases} \tag{2}$$

where u_1 and u_2 are positional parameters satisfying $0<u_1<u_2<1$, and Δ is a small random perturbation. The four points V_1, P_1, P_2, and V_2 are interpolated with the cardinal spline (in the rest of the paper, we use 'spline' for short) to draw a curvy line between V_1 and V_2 (Fig. 4a). Each curvy line is further rendered with varying width along its length, and the line width is determined by simple sinusoidal functions, as shown by gray curves in Fig. 4b.

Note that all corners inside a Voronoi cell are sharp corners, while corners on hand-drawn water caustic textures (Fig. 1a) appear round. We simulate these round corners by adding a small arc-like curve (solid curve in Fig. 4c) near each sharp corner inside the Voronoi cell. The area surrounded by the arc-like curve and two neighboring curvy lines near the corner (dotted lines in Fig. 4c) is colored with the same color that renders the curvy lines.

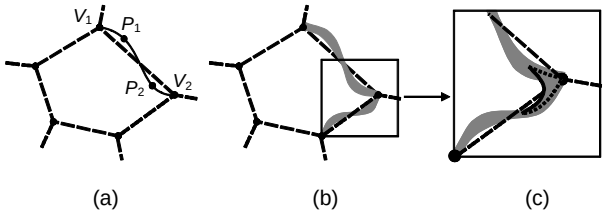

Fig. 4 Rendering of Voronoi edges

(a) Drawing a curvy line between V_1 and V_2; (b) Rendering the curvy line with varying width; (c) Generation of a round corner

4.2 Underwater light scattering effect

To model the cartoon light scattering effect shown in Fig. 1a, we take the sky as the background and draw the sun surrounded with gradient colors on a background image. Scattering beams in the deep water are modeled on a separate image ScatI by a few skeletal lines from a reference center taken from the sun on the background image to the bottom of ScatI, as indicated by dashed lines in Fig. 5a; directions of these lines are defined by θ_m (m=1, 2, ..., m_1). For each dashed line we add two additional lines (in dots) with directions defined by $\theta_m \pm \Delta$ to form the boundary of a scattering beam. Next, we use a horizontal line with height H_1 to divide the scattering beam into two parts: an upper triangle and a lower quadrilateral. All lower quadrilaterals associated with skeletal lines are taken as scattering beams in the deep water, and the scattering effects inside the quadrilaterals are obtained by filling them with brighter light color.

As for the scattered light near the sun, we select a few dashed lines of central beams. A triangle is formed by the far left and right boundary of selected scattering beams and a horizontal line with height $H_2 > H_1$ on ScatI, as indicated by the triangle in aquamarine in Fig. 5a. To render the triangle, we assign a brighter color to the selected dashed lines and a darker color to two upper sides of the triangle and midlines between dashed lines (Fig. 5b). The light scattering effect inside the triangle is obtained by linearly interpolating these two colors along each horizontal scan line inside the triangle.

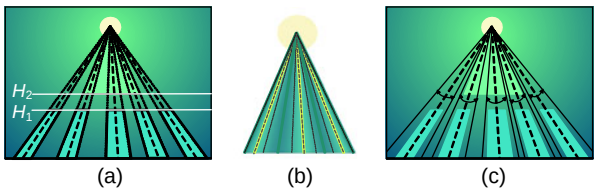

Fig. 5 Cartoon light scattering model
(a) Skeletal lines of scattering beams; (b) Upper triangle of scattering beams; (c) Swaying scattering beams

After scattering beams in the deep water and the scattered light near the sun are generated and rendered on ScatI, we project the scene composed of the water surface and the background image surface onto the view plane to obtain an image with the sun and water caustic textures seen from under water, and then blend ScatI with the projected image on the view plane to produce a frame of animation (see Fig. 10 in Section 7).

4.3 Dynamic control

For the dynamics of water caustics, we try different strategies to animate water caustic textures. At first, we add perturbations to Voronoi vertices every frame. The resultant animation of water caustic textures, however, looks jerky due to the random perturbations added. To obtain smooth movements of water caustic textures, we alternatively generate small ellipses with initial Voronoi vertices of VD(P) as reference centers and then animate Voronoi vertices along the generated ellipse paths. On each frame of animation, Voronoi vertices are moved to new positions on their ellipse paths, and curvy lines associated with Voronoi edges are generated afterward. The resultant water caustic textures animation looks smooth as shown in the accompanying video (http://www.cad.zju.edu.cn/home/jhyu/Animations/Foamywater.wmv).

During animation, the light scattering effect is further animated by swaying scattering beams to and fro horizontally (Fig. 5c), where the solid lines indicate the range that one beam's skeletal line can reach during the movement. The brightness of the light scattering near the sun is also changed periodically with time.

5 Foamy wake

Wake is the foamy path of disturbed water left behind by a moving ship. In hand-drawn animation (Fig. 1b) the wake is depicted with stylized holes distributed with structure as a Voronoi diagram. These holes are progressively larger along the path, simulating the dispersion of the wake. Foamy waves near the stern appear higher than those in other parts on the path and some forms are added near the boundary between the ship and the water.

Note that the wake is caused by the ship moving on the water, making cartoon wake animation by this method requiring at least modeling of a water wave surface, a ship, the wake behind the ship, and forms surrounding the ship, and animating them coordinately. Since the flat surface used for the water caustics is not suited for representing the water wave surface in cartoon wake animation, we choose instead a fast Fourier transformation (FFT) based method

(Mastin *et al.*, 1987) to synthesize the water surface with random waves. The ship model can be made through commercial software or downloaded from the Internet.

Fig. 6 illustrates our model for wake animation. To coordinate the movements between the ship model, foamy wake, and forms surrounding the ship in the animation, we predefine two curves, WakeR for the boundary of the wake region and ShipB for the boundary between the ship and the water surface (Fig. 6a). Both WakeR and ShipB are moved together with the ship on the water surface along the user-defined path. On each frame of animation, we map WakeR and ShipB onto a texture image as constraints to construct the wake texture inside WakeR and form textures surrounding ShipB (Fig. 6b), and then take WakeR and ShipB as the references to raise some parts on the water surface (Fig. 6c). The texture image is mapped back to the deformed water surface to obtain the final result (Fig. 6d). The following subsections describe our model in detail.

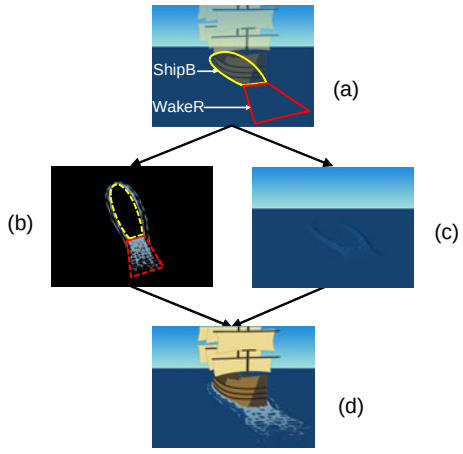

Fig. 6 Foamy wake model

(a) Specification of the wake region and the ship boundary; (b) Generation of textures for the wake and forms surrounding the ship; (c) Deformation of the water surface; (d) Mapping of texture on the deformed water surface

5.1 Wake and form textures

We choose an isosceles trapezoid on the water surface to approximate the wake region with boundary WakeR. The lengths of the top and the base are set empirically approximately 1.2 and 1.6 times the ship stern width respectively, and the height is set to half length of the ship. Our user interface (UI) allows further editing on the shape of WakeR according to the scene.

Once WakeR is obtained, we map it onto the texture image and generate a Voronoi diagram VD(P) by putting some sites with uniform distribution inside the mapped WakeR. To render a hole inside the cell VP(p_i), we pick up an intermediate point P_k along the line formed by p_i and vertices V_k of VP(p_i) with the following formula:

$$P_k = uV_k + (1-u)p_i, \quad k = 1, 2, ..., \mathrm{vn}_i, \tag{3}$$

where $u \in [0, 1]$ is a parameter that controls the position of P_k on the line, vn_i is the vertex number associated with VP(p_i) and can be determined uniquely after a VP(p_i) is constructed (Lischinski, 1994). A hole inside VP(p_i) can be drawn by interpolating intermediate points P_k in VP(p_i) with the spline, as illustrated by the solid curve drawn inside a Voronoi cell with five vertices in Fig. 7. The hole size is adjustable by varying the amplitude of u. To control the hole size varying progressively from small to large inside WakeR, we define a simple model $u=d/H$, where d is the distance from p_i to the top of the isosceles trapezoid and H is the height of the isosceles trapezoid.

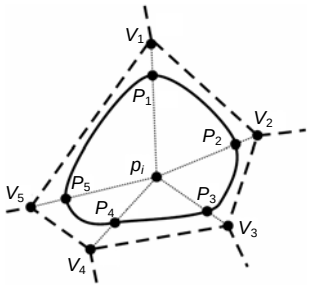

Fig. 7 Hole rendering inside a Voronoi cell VP(p_i)

When the ship model is put into the scene, a boundary is formed between the ship and the water. To avoid the slow boundary detection between the ship and the water surface in every frame in the animation, we pre-construct the curve ShipB. To do this, we cut the ship model by a plane parallel to the water surface and take a few control points interactively from the boundary between the plane and the ship model. ShipB is obtained by interpolating these points with the spline and is then mapped onto the texture image. Shapes of forms surrounding the ship are modeled by a series of sinusoidal units with variations

in amplitude and radian frequency and attached to the mapped ShipB as the form textures shown in Fig. 6b. Some curved lines indicating ripples are added near forms surrounding the ship.

5.2 Water surface deformation

Water surface deformation is to raise some parts corresponding to the wake path on the FFT water surface, simulating effects that the wake near the stern is higher than elsewhere on the water surface. In our implementation we use WakeR to deform the water surface as follows.

First we define a normalized height map $h(u, v)$ on the unit square in the u-v space with

$$h(u,v)=w_1(u)\cdot w_2(v), \tag{4}$$

where $w_1(u)$ and $w_2(v)$ are empirical height weights characterized by

$$w_1(u)=\begin{cases}\sin(2.5\pi u), & u\le 0.2,\\ 1.0, & 0.2<u<0.8,\\ \cos[2.5\pi(u-0.8)], & u\ge 0.8,\end{cases} \tag{5}$$

$$w_2(v)=\begin{cases}0.5[1+\cos(2.5\pi v)], & v\le 0.4,\\ 0.0, & v>0.4.\end{cases} \tag{6}$$

The corresponding plots of these three functions are given in Fig. 8.

Next, we map the height map onto the wake region with boundary WakeR on the FFT water surface and raise vertices of the mesh falling in WakeR by displacement mapping.

The height map near ShipB can be constructed using a simple model. Since ShipB is obtained by interpolating some control points with the spline, let it be parameterized by an arc length parameter s as ShipB(s). We define a normalized peak profile $H_m(s)$ on ShipB(s) by a series of sinusoidal units with variations in amplitude and radian frequency. Let r be a radius of influence around ShipB(s) on the height map image. The height value of a pixel p' within the influence area is set empirically to $\Delta H=H_m(s_c)\times \cos[0.5\pi|\text{ShipB}(s_c)-p'|/r]$, where ShipB($s_c$) is the closest point to p' among points on the curve. Once the height map for the ship boundary is obtained, vertices near the ship boundary on the water surface can be raised in a similar manner as in the wake region.

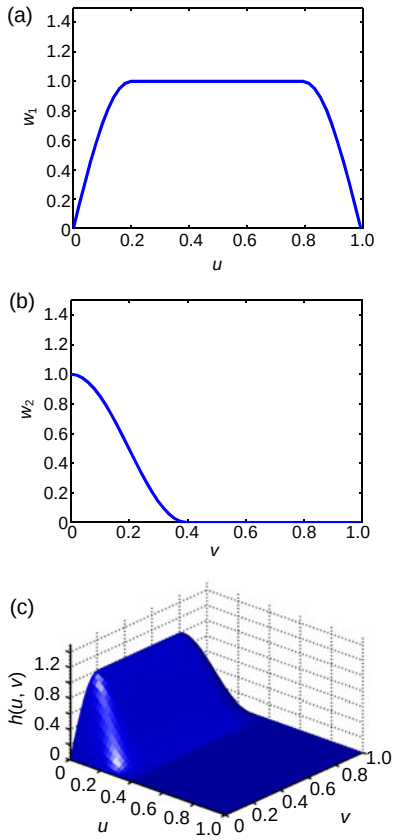

Fig. 8 Plots of functions $w_1(u)$ (a), $w_2(v)$ (b), and $h(u, v)$ (c)

After the water surface is deformed, a frame of animation can be obtained by mapping the wake and form textures onto the raised water surface (see Fig. 11 in Section 7).

5.3 Wake dynamics

Dynamic control of the wake is simple. As the ship is moved along the user-defined path in the animation, we just move all Voronoi vertices of VD(P) in the direction from the top to the base of the isosceles trapezoid (i.e., the opposite direction of the ship movement) and spread them out as moving, to simulate the process that foamy waves come out near the stern and then remain at their positions on the water surface and disperse with time.

6 Longshore currents

Longshore currents are currents of water flowing parallel to the shore. The movements of the longshore currents consist of two phases: an incoming wave

phase in which the wave crest begins to rise and reaches its maximum height, and then the wave 'breaks' by decreasing its height as it moves on the beach, forming the foamy, bubbly surface; a backwash phase in which a seaward current that results from the receding swash on the beach face, after a wave breaks, joins the seaward movement of the wave trough toward the next incoming crest. As waves repeatedly hit the shore, water moves onto the beach and then retreats in a continuous cycle.

Fig. 9 illustrates our model for cartoon longshore currents. First, we model the region of the incoming wave with two parallel sinusoidal-like curves C_f and C_b with amplitude varying stochastically in time, where C_f is the profile of the incoming wave near the beach and C_b is the boundary of foamy water behind C_f. The distance between C_f and C_b controls the width of the incoming wave. Next, we map C_f and C_b onto the texture image and generate longshore current foamy textures by rendering holes inside Voronoi cells on the incoming wave region. Hole sizes are controlled progressively larger from C_f to C_b.

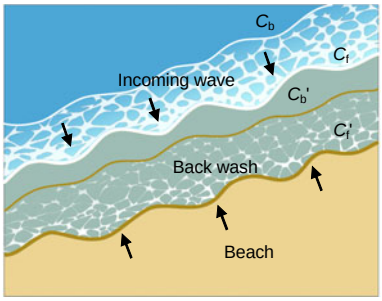

Fig. 9 Longshore currents model

To simulate height variations of incoming waves on the water mesh surface, we define the peak height on C_f in the normalized height map by an empirical formula:

$$H_m(u)=\begin{cases}u/0.7, & u\le 0.7,\\ (1.0-u)/0.3, & u>0.7,\end{cases} \tag{7}$$

where $u\in[0, 1]$ is a spatial parameter with u=0 corresponding to the position where the incoming wave is triggered, and u=1 corresponding to the predefined limit position that C_f can reach on the beach. With the peak height defined above, we can generate a height map for the incoming wave in the same manner as described earlier in Section 5.2 for the height map near ShipB. The height map is mapped onto the water surface to raise relevant vertices by displacement mapping. After the water surface is deformed, we map the longshore current texture onto the raised water surface to produce a frame of animation (see Fig. 12 in Section 7).

During the animation, the incoming wave is triggered periodically at the user-specified position in the scene. As C_f and C_b are moved towards the beach, the width of the incoming wave and the hole size are increased progressively. When C_f reaches the predefined limit position on the beach, the incoming wave stops moving and the backwash phase starts.

During the backwash phase, the two curves turn to be C_f' and C_b', which are then moved backward (while the next incoming wave is still moved towards the beach). On the backwash region, hole sizes are increased further, and the remaining foamy textures are blended with the beach texture image with time.

In the accompanying video we show two versions of longshore currents animations. One contains a single incoming wave and the other contains two succeeding incoming waves. More incoming waves can be added if desired.

7 Results

Models constructed above define cartoon water animations including water caustics, foamy wake, and longshore currents. Through high level initial specification, the defined effects are animated procedurally. Our system allows further editing on some parameters to change the appearance of defined animations. For example, the Voronoi cell size can be altered by varying the distribution density of sites p_i, as shown by the three variants of the generated animation for the water caustic effects in Fig. 10. To stage the scene properly we let some fish move under water in the

Fig. 10 Three variants of water caustics effects with 50 (a), 100 (b), and 500 (c) Voronoi cells

animation. A side-by-side comparison between hand-drawn animation (Fig. 1a) and our procedurally generated animation for the water caustic effects is given in the accompanying video.

For cartoon wake animation, the user needs to specify the path and speed of the ship, control points for ShipB, the length and width of the wake region, and the speed of the wake. Fig. 11 includes six frames of generated foamy wake animation, in which three frames in the upper row show the ship moving away. Some ripples are scattered on the water surface. To simulate the hand-drawn series in Fig. 1b, we animate the foamy wake by moving the camera up and away from the ship, as shown by the three frames in the bottom row. A side-by-side comparison between hand-drawn and computer generated animations is given in the accompanying video.

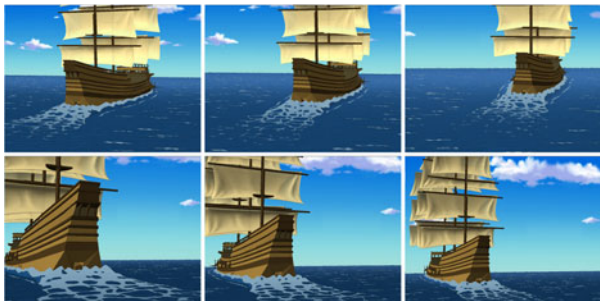

Fig. 11 Sample frames taken from two pieces of generated foamy wake animation

The three frames in the upper row show the ship moving away; in the bottom row, the foamy wake is animated by moving the camera up and away from the ship

To animate longshore currents, users are required to specify the limit position that the longshore current can reach on the beach in the scene, the speed of the longshore currents, and the distance between the two succeeding incoming waves. Fig. 12 shows two versions of longshore current animations. The upper three frames are taken from one cycle of the longshore current movements in the animation with a changing view point. The bottom three are taken from the animation with a fixed view position, simulating the hand-drawn series in Fig. 1c, as shown by a side-by-side comparison in the accompanying video.

The procedural nature of our models allows for a much faster generation of cartoon effects animations than could otherwise be created by hand-drawn series. Our models are run on a PC with 2.8 GHz CPU and 1 GB RAM. Table 1 tabulates the processing time involved in Voronoi diagram texture generation, water surface deformation, and rendering for a frame of animation. Since the number of Voronoi cells changes dynamically in the animation of longshore currents, we show only the statistics corresponding to one incoming wave in Table 1.

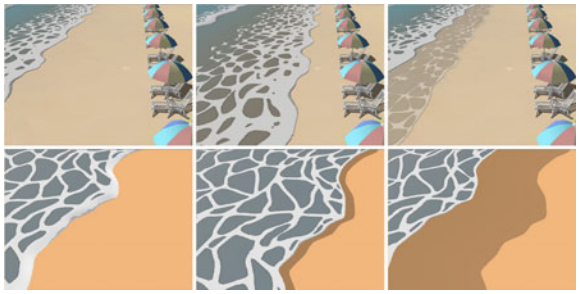

Fig. 12 Two versions of generated longshore currents animation

Upper row: taken from one cycle of the longshore current movements in the animation with a changing view point; bottom row: taken from the animation with a fixed view position, simulating the hand-drawn series in Fig. 1c

Table 1 Summary of processing time

Animation	VP number	Processing time (ms)		
		Texture	Deformation	Rendering
Water	120	0.64	0.0	32.74
Foamy	240	1.24	18.10	63.96
Longshore	405	17.38	34.32	47.82

VP: Voronoi polygon

8 Conclusions and future work

We have demonstrated a procedural system capable of generating cartoon effects ranging from water caustics to foamy water such as wake and longshore currents. The main contributions of our approach are the emulation of visual abstraction of these cartoon effects by using Voronoi diagrams and emulation of the motion abstraction by designing relevant controlling mechanisms for each effect. Our models enable the creation of cartoon effects with minimal intervention from the animator. In our approach the effects are animated procedurally in the style of cartoons. Thus, the strength of our approach to animation lies in the fact that this scheme can reduce animator's workload significantly. It turns the role of the animator from a hand-drawing slave to a high level

model controller.

Due to the diversity of motion abstraction and visual abstraction in hand-drawn cartoon effects animation, it is impossible to discover a general solution to all modeling and rendering problems encountered in cartoon effects animation. A practical goal is therefore to allow cartoon effects animation to be handled by as few separate procedures as possible, as those presented in this paper.

In the future we intend to extend our work to other cartoon effects whose visual abstraction is also similar to Voronoi diagrams, such as cracking ice, drying land, and pouring water.

References

Adabala, N., Manohar, S., 2002. Techniques for realistic visualization of fluids: a survey. *Comput. Graph. Forum*, **21**(1):65-81. [doi:10.1111/1467-8659.00566]

Aurenhammer, F., 1991. Voronoi diagrams—a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, **23**(3):345-405. [doi:10.1145/116873.116880]

Bregler, C., Loeb, L., Chuang, E., Deshpande, H., 2002. Turning to the masters: motion capturing cartoons. *ACM Trans. Graph.*, **21**(3):399-407. [doi:10.1145/566654.566595]

di Fiore, F., Claes, J., Reeth, F.V., 2004. A Framework for User Control on Stylised Animation of Gaseous Phenomena. Computer Animation and Social Agents Conf., p.171-178.

Eden, A.M., Bargteil, A.W., Goktekin, T.G., Eisinger, S.B., O′brien, J.F., 2007. A Method for Cartoon-Style Rendering of Liquid Animations. Graphics Interface Conf., p.51-55. [doi:10.1145/1268517.1268528]

Hoff, K.E.III, Keyser, J., Lin, M., Manocha, D., Culver, T., 1999. Fast Computation of Generalized Voronoi Diagrams Using Graphics Hardware. 26th Annual Conf. on Computer Graphics and Interactive Techniques, p.277-286. [doi:10.1145/311535.311567]

Iglesias, A., 2004. Computer graphics for water modeling and rendering: a survey. *Fut. Gener. Comput. Syst.*, **20**(8):1355-1374. [doi:10.1016/j.future.2004.05.026]

Kwatra, V., Adalsteinsson, D., Kim, T., Kwatra, N., Carlson, M., Lin, M., 2007. Texturing fluids. *IEEE Trans. Visual. Comput. Graph.*, **13**(5):939-952. [doi:10.1109/TVCG.2007.1044]

Lischinski, D., 1994. Incremental Delaunay Triangulation. Graphics Gems Series (Graphics Gems IV). Academic Press, p.47-59.

Mastin, G.A., Watterberg, P.A., Mareda, J.F., 1987. Fourier synthesis of ocean waves. *IEEE Comput. Graph. Appl.*, **7**(3):16-23. [doi:10.1109/MCG.1987.276961]

Narain, R., Kwatra, V., Lee, H.P., Kim, T., Carlson, M., Lin, M., 2007. Feature-Guided Dynamic Texture Synthesis on Continuous Flows. Eurographics Symp. on Rendering Conf., p.361-370.

Thornton, J.D., 2006. Directable Simulation of Stylized Water Splash Effects in 3D Space. Int. Conf. on Computer Graphics and Interactive Techniques, p.94. [doi:10.1145/1179849.1179967]

Voronoi, G.F., 1908. Nouvelles applications des paramètres continus à la théorie de formes quadratiques. *J Reine Angew. Math.*, **133**:97-102 (in France). [doi:10.1515/crll.1908.133.97]

You, M., Park, J., Choi, B., Noh, J., 2009. Cartoon animation style rendering of water. *LNCS*, **5875**:67-78. [doi:10.1007/978-3-642-10331-5_7]

Yu, J.H., Jiang, X., Yao, C., Chen, H.Y., 2007. Real-time cartoon water animation. *Comput. Animat. Virt. Worlds*, **18**(4-5):405-414. [doi:10.1002/cav.207]

Yu, J.H., Liao, J., Patterson, J., 2008. Modeling the interaction between objects and cartoon water. *Comput. Animat. Virt. Worlds*, **19**(3-4):375-385. [doi:10.1002/cav.239]