COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2011; **22**:81–89
Published online 12 April 2011 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.400

SPECIAL ISSUE PAPER

Modeling ocean waves and interaction between objects and ocean water for cartoon animation

Jing Liao[1], Jinhui Yu[1]* and John Patterson[2]

[1] State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China
[2] Dept of Computing Science, University of Glasgow, Glasgow G12 8QQ, UK

ABSTRACT

We present a scheme for the animation of ocean waves in a recognizable drawn animation cartoon style. This consists principally of two parts: the first part is the ocean surface generated by a procedural model which governs the dynamics of the ocean surface and rendered with stylized whitewater forms over the ocean wave crests; the second part is water effects caused by interactions between the ocean water and obstacle objects, such as waves crashing against obstacles-like rocks or boats and forms surrounding objects, and those effects are simulated with different hierarchical models. The ocean surface can be used either alone or in conjunction with effects caused by the interactions according to the scene. With minimum user intervention, i.e., specification of a few parameters, our model is able to generate cartoon ocean wave animations using these procedural methods, as shown by examples given in the paper. Copyright © 2011 John Wiley & Sons, Ltd.

Supporting information may be found in the online version of this article.

KEYWORDS
effects; model; ocean water; cartoon animation

***Correspondence**
Jinhui Yu, State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310027, China.
E-mail: jhyu@cad.zju.edu.cn

1. INTRODUCTION

In cartoon animation, ocean waves can be depicted in a range of stylized forms, as shown by the two example frames in Figure 1. Here, the left image is of a ship moving on the sea surface, and the right image shows a crashing wave. We can see from Figure 1 that animating cartoon water is far from easy by hand, and the work load can be reduced significantly if we can generate cartoon water forms by computational means.

In realistic ocean water simulations, Fournier and Reeves [1] simulated a train of trochoids based on a mix of Gerstner and Biese swell models. Peachey [2] proposed a similar idea with fewer refinements. Later on Ts'o and Barsky [3], Gonzato and Saec [4] have proposed more precise ways to solve the propagation. Mastin *et al.* [5] produced a height field that has the same spectrum as the ocean surface. Tessendorf [6] showed that dispersive propagation can be managed in the frequency domain and the resulting field can be modified to yield trochoid waves. A hybrid approach is used by Thon *et al.* [7] in which the spectrum is synthesized using a spectral approach and used to control the trochoids generated by a Gerstner model. For a comprehensive survey of realistic water animation please refer to [8,9].

Unfortunately, methods for realistic ocean animations are unable to represent cartoon water forms seen from Figure 1. In recent years, a few procedural models have been proposed for cartoon water animation. Di Fiore *et al.* [10] simulated a garden hose by utilizing hand-drawn drops and animating them along the 3D trajectory. Thornton [11] used a modular rig composed of a series of nodes to describe the shape of hand-drawn splash, and the splash effects are generated by constraining particle emitters to these nodes. Eden *et al.* [12] take as input a liquid surface obtained from a 3D physically based liquid simulation system, and render this with bold outlines and constant colors to highlight near-silhouettes and shallow areas. Yu *et al.* [13] propose a template based approach in which some stylized water forms are placed on templates and animated along a path. Later on, they [14] use a path grid to deal with the interaction between obstacle objects and the cartoon water.

In this paper we present a framework for modeling cartoon ocean water in 3D, as illustrated in Figure 2. Our framework mainly consist of two parts: the first part is the

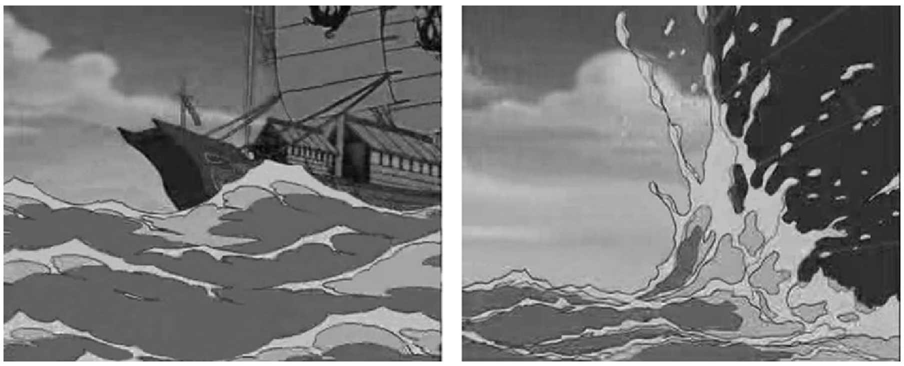

Figure 1. Hand-drawn ocean waves.

ocean surface rendered with stylized whitewater forms over the ocean wave crests, this includes generation and deformation of ocean surface, mapping wave peak lines on a texture plane on which the whitewater forms are constructed over the peak lines, and the texture image containing all whitewater forms is mapped onto the deformed ocean surface; the second part is the water effects caused by the interaction between ocean water and obstacle objects, such as forms surrounding the objects and waves crashing against obstacles such as rocks or boats.

2. OCEAN SURFACE

2.1. Ocean Surface Modeling

In cartoon series, stylized shapes are drawn as whitewater over the wave crests (Figure 1). In order to guide the placement of whitewater forms over wave peaks of computer generated ocean surface, we adopt the Gerstner wave model given in Ref. [6] from which the wave peak lines can be calculated straightforwardly.

In the Gerstner wave model points on the surface of the water go through a circular motion as a wave passes by. If a point on the undisturbed surface is labeled as $\mathbf{x}_0 = (x_0, z_0)$ and the undisturbed height is $y_0 = 0$, then the point on the surface is displaced at time t to

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_0 + \sum_{i=1}^{n} (\mathbf{k}_i/k_i)A_i \sin(\mathbf{k}_i \cdot \mathbf{x}_0 - \omega_i t + \varphi_i) \\ y &= \sum_{i=1}^{n} A_i \cos(\mathbf{k}_i \cdot \mathbf{x}_0 - \omega_i t + \varphi_i) \end{aligned} \tag{1}$$

here, n is the number of different sine functions with different amplitude A_i, the vector $\mathbf{k}_i$ is horizontal vector (called as the wave vector) pointing in the direction of travel of the wave, and have magnitude k_i related to the length of the wave (λ_i) by $k_i = 2\pi/\lambda_i$, ω_i is the frequency that, in deep water where the bottom may be ignored, satisfies the relationship $\omega_i^2(k_i) = gk_i$, where g is the gravitational constant, nominally 9.8 m/seconds^2. φ_i is the phase.

In order to simulate the underlying ocean surface in Figure 1, we choose from experiments $n = 4$ in Equation (1), and among the four components, one is

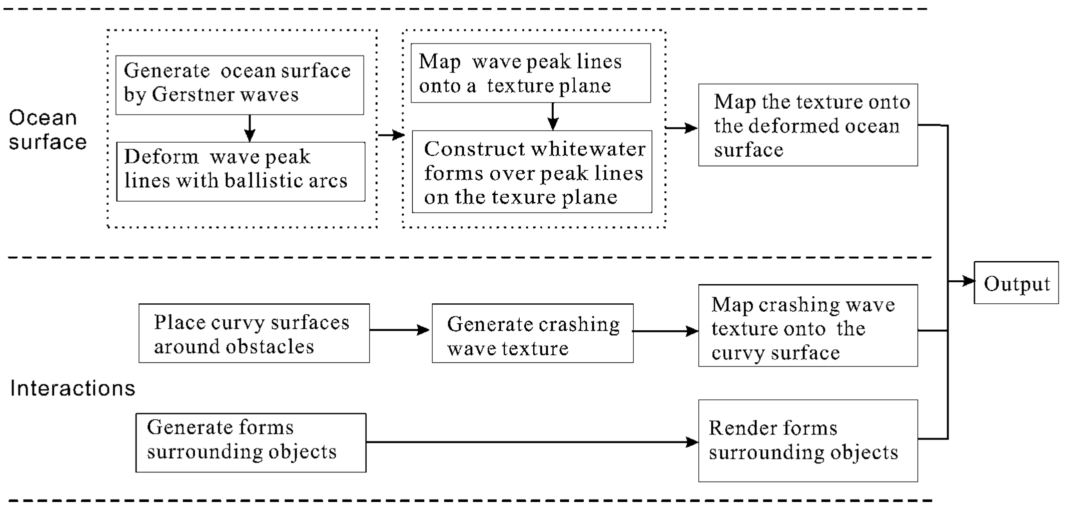
Ocean surface
Generate ocean surface by Gerstner waves
Deform wave peak lines with ballistic arcs
Map wave peak lines onto a texture plane
Construct whitewater forms over peak lines on the texure plane
Map the texture onto the deformed ocean surface
Output
Interactions
Place curvy surfaces around obstacles
Generate crashing wave texture
Map crashing wave texture onto the curvy surface
Generate forms surrounding objects
Render forms surrounding objects

Figure 2. System overview.

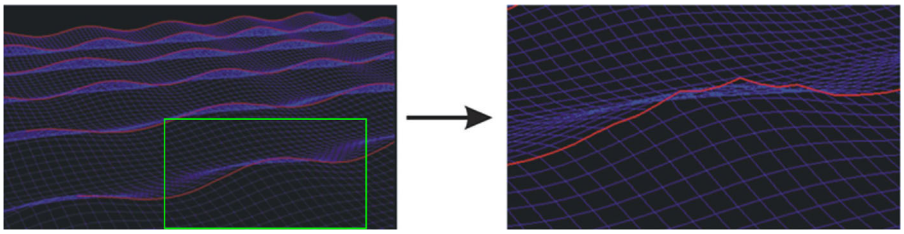

Figure 3. Wave peak lines of the main component (left) and a deformed peak line (right).

taken as a main component with amplitude three times bigger and wave length 1.5–2 times longer than other three components. A resultant surface is shown by blue grid on the left of Figure 3.

2.2. Stylized Deformation Over Wave Peaks

To simulate the stylized top profile of the whitewater as in Figure 1, we should deform wave peak lines on the smooth ocean surface generated by Equation (1). This involves the abstraction of wave peak lines, modeling of ballistic arcs and the deformation of wave peak lines with ballistic arcs, as described in the following three paragraphs.

2.2.1. Abstraction of Wave Peak Lines.

From Equation (1) the surface of the main component is defined by

$$\begin{aligned}\mathbf{x} &= \mathbf{x}_0 + (\mathbf{k}_m/k_m)A_m \sin(\mathbf{k}_m \cdot \mathbf{x}_0 - \omega_m t + \varphi_m)\\ y &= A_m \cos(\mathbf{k}_m \cdot x_0 - \omega_m t + \varphi_m)\end{aligned} \tag{2}$$

where the subscript m indicates the main component. The points on peak lines of the main component satisfy $y = A_m$ in Equation (2), which in turn requires

$$\mathbf{k}_m \cdot \mathbf{x}_0 - \omega_m t + \varphi_m = 2j\pi \tag{3}$$

where j is an integer that corresponds to each peak line. By solving Equation (3) we can get coordinates of peak lines, which are shown by red curves on the left of Figure 3.

2.2.2. Modeling of Ballistic Arcs.

Here, we adopt the simple stochastic model given in Ref. [14] to generate ballistic arcs:

$$H_k = c + \mathrm{rand}(v) \tag{4}$$

where H_k is the k-th key sample point of the top profile, c is a constant which is the mean magnitude of H_k and rand(v) is a random variable distributed about zero with a standard deviation of v. Two neighboring points as defined by Equation (4), say H_k and H_{k+1}, are then picked up and a new point H_{in} in between them is added, the magnitude of H_{in} being bounded in the range $(0.4 \sim 0.6)\min(H_k, H_{k+1})$. Three points H_k, H_{in}, and H_{k+1} are interpolated with the spline to obtain a ballistic arc BARC$_k$, as indicated by the solid curve in the upper of Figure 4.

2.2.3. Deformation of Wave Peak Lines.

Careful observation on Figure 1 reveals that ballistic arcs are relatively higher on wave crests than those on troughs, thus in our model we cannot add ballistic arcs modeled above directly onto wave peak lines because this results in the same height of ballistic arcs on both peaks and troughs on wave peak lines. Our solution to this problem is to use the height information of ocean waves to weight the magnitude of samples H_k in Equation (4), as indicated by the lower of Figure 4.

We first map the heights of peak line into the range [0,1], as shown by the dash line in the lower of Figure 4. And then we pick up some points $\mathbf{x}_k$ with an interval d along wave

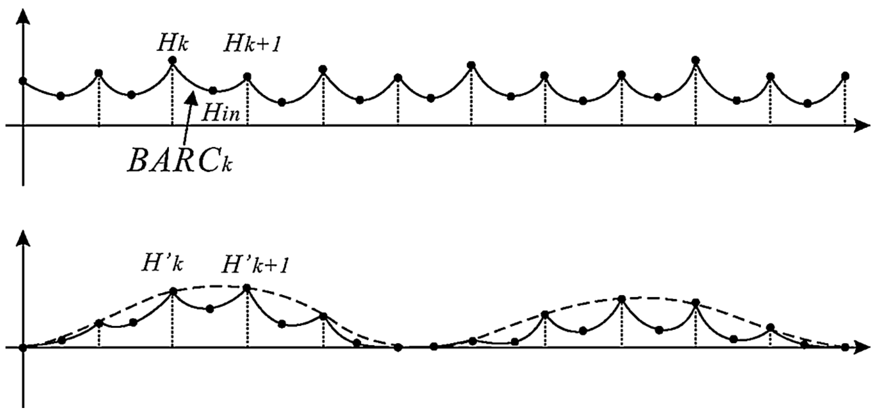

Figure 4. Ballistic arcs (upper) and intermediate top profile of wave peak line with weighted ballistic arcs (lower).

peak lines of the main component in Equation (1), and use the mapped height H'_k on the peak lines to generate the weighted ballistic arcs to form the intermediate top profile.

Next we add the intermediate top profile over original wave peak lines. In order to control the raised height over wave peak lines properly, we scale the intermediate top profile from experiments by a factor $\Delta H = 0.1A_m$. The finally deformed peak line over the wave crest is shown on the right of Figure 3.

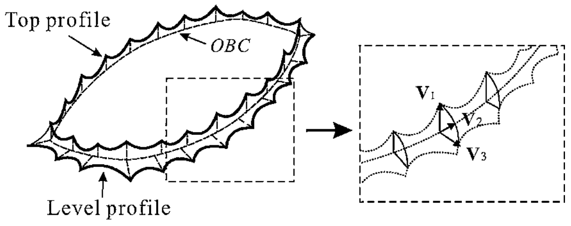

Figure 6. The top and level profiles with ballistic arcs attached to OBC (left) and the surface construction of forms surrounding objects (right).

2.3. Whitewater Forms and Ocean Surface Rendering

Whitewater forms over the ocean wave crests can be constructed with two group of ballistic arcs above and below the horizontal reference axis, as shown on Figure 5(a). During the rendering phase, wave peak lines of the main component are first mapped onto a texture domain, on which they form some straight lines. Each line is then broken into a few segments stochastically, as illustrated by red lines on Figure 5(b). Whitewater forms are drawn on those line segments, some additional scattered forms such as small (short and thin) whitewater forms and leaf shapes are added to indicate water textures between peak lines. The texture image is finally mapped back onto the ocean surface, as shown on Figure 5(c).

3. FORMS SURROUNDING OBJECTS

In the case of objects such as boats on the ocean surface, some time-varying forms may be created along the border between the object and the water surface. To simulate these time-varying forms we predefine a curve that approximates the bounder between the object and the water surface by use of a few control points interpolated with the spline on a plane parallel to the ocean surface, and such curve is denoted as OBC. Next, we lay a rig with a number of columns upward on OBC and let the height of the upper node of each column vary stochastically between the predefined upper and lower limits, then we add a small ballistic arc BARC_k between every two neighboring upper nodes to form the top profile of forms. The level profile (seen from above the water surface) is generated in a similar manner, with the only difference that the directions of columns on the rig are outward and parallel to the ocean surface, as illustrated by Figure 6 (left).

In order to create the volumes of forms surrounding objects, we take each corresponding pair between the top and level profiles as two vectors $\mathbf{V}_1$ and $\mathbf{V}_3$ starting from a point on OBC, and then add another vector $\mathbf{V}_2$ in between them with magnitude $|\mathbf{V}_2| = (0.6 \sim 0.8)(|\mathbf{V}_1| + |\mathbf{V}_3|)$. The end-points of three vectors are interpolated with the spline to generate a curve that approximates the side profile of forms surrounding objects, Figure 6 (right) shows three such curves along OBC with certain intervals. The entire surface of forms surrounding objects can be obtained by interpolating every corresponding pair between the top and level profiles along OBC in the same fashion.

Animation of forms surrounding objects can be achieved by just changing heights and positions of columns on the rig stochastically.

4. CRASHING WAVE EFFECTS

A huge arching spray may arise when wave crash against obstacles-like rocks or boats, and a hand drawn series of stylized crashing wave effects are shown in Figure 7. We can see from the figure that the crashing wave is mainly composed of two parts: a base "sheet" near the water surface and droplets above the sheet. Also, they have histories as follows: the base sheet begins to raise, then its

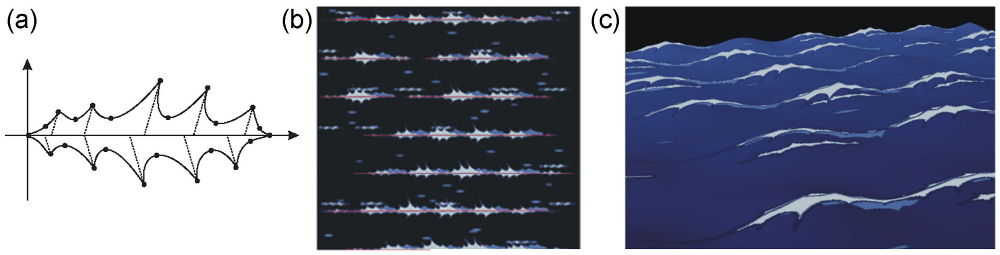

Figure 5. Whitewater forms.

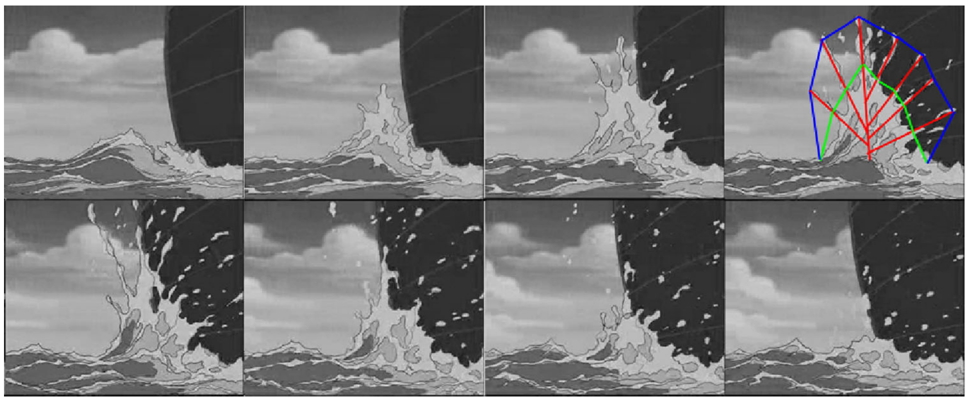

Figure 7. Hand drawn series for a crashing wave effect.

dynamic silhouette moves upward, with some longer droplets anticipating the leading edge and then splitting apart, and finally the sheet collapses with droplets falling downward and disappearing in a few frames.

In order to model the hand drawn crashing wave, we draw some lines on the series along the center of longer droplets, and line segments to approximate the silhouette of the base sheet and the outer profile formed by droplets, as shown in Figure 7. Those lines suggest that hand drawn crashing wave can be simulated with some simple models: a tree-like structure (red lines in Figure 7) for trajectories of droplets which remains unchanged during the lifetime of the crashing wave, the dynamic silhouette of the base sheet can be approximated by an arc-like curve (green lines in Figure 7), and the outer profile formed by droplets is of a bigger arc-like form (blue lines in Figure 7). In our implementation, we start from modeling the outer profile of the crashing wave, as detailed next.

The outer profile can be modeled by use of a pentagonal polygon as the skeleton (Figure 8(a)). The pentagonal polygon can be constructed by first taking coordinates (x_c, y_c) as the reference point which determines the position of the base sheet, the two side vertices (x_l, y_l), (x_r, y_r) on the bottom and the top vertex (x_t, y_t) of the pentagon polygon can be determined by the parameters of height H, width W with a simple model, as depicted in Figure 8(a).

Furthermore, additional two intermediate vertices (x'_l, y'_l) and (x'_r, y'_r) can be calculated as follows:

$$\begin{aligned} fa &= 0.2 \sim 0.3, \\ x'_l &= x_t + fa * (x_l - x_t) - \Delta x, \\ y'_l &= y_t + fa * (y_l - y_t), \\ x'_r &= x_t + fa * (x_r - x_t) + \Delta x, \\ y'_r &= y_t + fa * (y_r - y_t) \end{aligned} \tag{5}$$

where fa is a position parameter which controls the position of the intermediate point, Δx is a parameter related to the width of the skeleton. These five points are then interpolated with the spline to get the outer profile of the crashing wave.

The tree-like structure is modeled as follows. First we divide the height $fa_1\ H$ into m small segments, where $fa_1 = 0.55 \sim 0.75$ is a parameter which controls the position of the highest branching point on the central vertical axis, we then sample m points from upper left part of the outer profile and connect them to the corresponding points on divided segments on the central vertical axis to get branches for the left part of the tree structure. The right part of the tree structure can be generated in a similar manner (see Figure 8(b)). Each branch is labeled as B_i in a clockwise order.

The silhouette of the base sheet is generated by use of a small pentagonal polygon skeleton with height BH less than H in a manner similar to that used in the outer profile of the crashing wave. We interpolate five vertices of this pentagonal polygon with spline to get an arc-like curve, as shown by the dash line in Figure 8(c). For each neighboring branch pair B_i and B_{i+1} we calculate two intersection points IPt_i and IPt_{i+1} between the arc and B_i and B_{i+1}, and then add perturbations to IPt_i and IPt_{i+1} to simulate irregular shape of the silhouette, ballistic arcs are finally added to the line segment between IPt_i and IPt_{i+1} to obtain the silhouette of the base sheet, the number of ballistic arcs inserted varies from 1 to 2 randomly.

Droplets above the base sheet can be modeled by first adding a few short bars across the segment between IPt_i and the outer end point of branch B_i with equal distance, lengths of those bars are weighted with a sinusoidal function, the final shape of the droplet can be obtained by interpolating end points of short bars on two sides of B_i with the spline (Figure 8(d)). Droplets begin to split when they move further away from the base sheet, and then begin to fall and disappear in a few frames. Their dynamic control is straightforward.

Holes torn inside the base sheet. Modeling holes inside the base sheet is more difficult because those holes have varying shapes, we define different types of skeletons to match different hole shapes by use of a few control points as indicated with dots in Figure 9, the hole shapes can be determined by adding additional control points (circles in Figure 9) by some simple models associated with skeletons. While detailed description of their presence and behavior is omitted here.

In summary, the control mechanism for the crashing wave effect can be described with the following procedure:

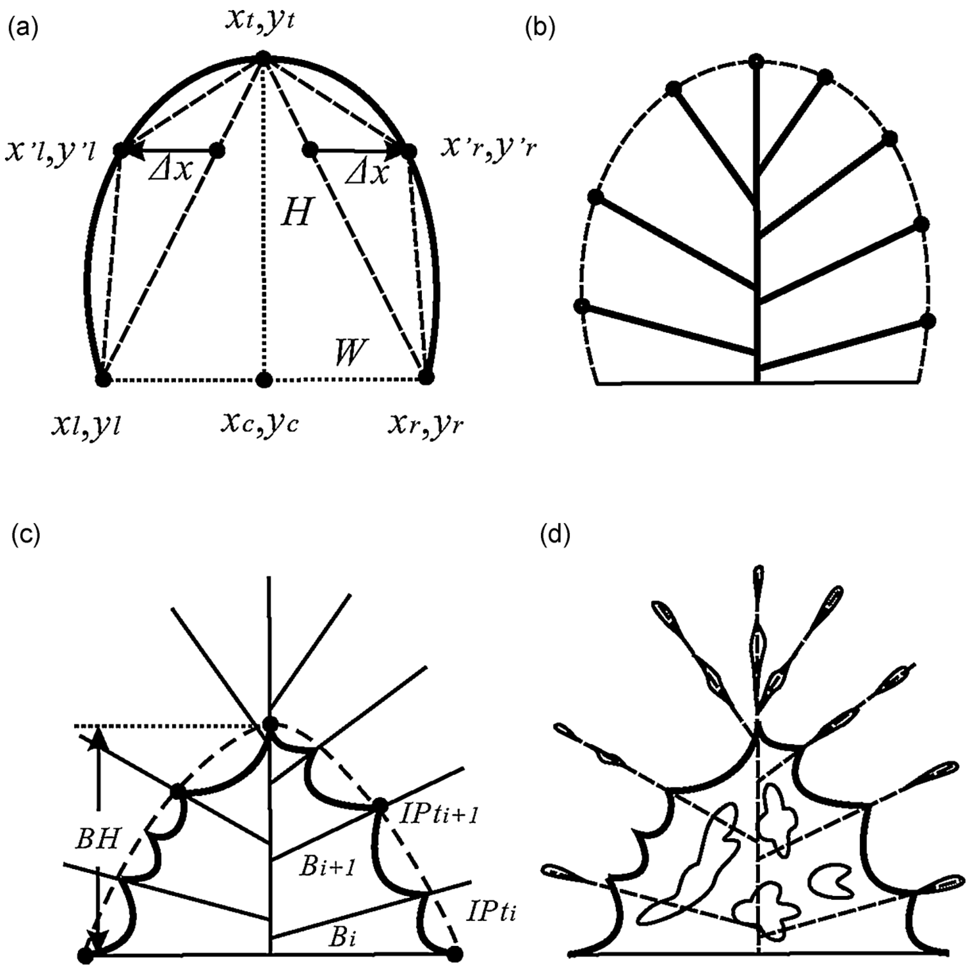

Figure 8. Process of modeling a crashing wave effect: (a) the arc-like outer profile, (b) the tree-like structure, (c) the silhouette of the base sheet, and (d) the base sheet with holes and droplets added.

```
CrashingWave($x_c$, $y_c$, $H$, $W$) {
Generate the outer profile with $x_c$, $y_c$, $H$, $W$;
Generate the tree-like structure;
For each frame of animation do {
If in the raising phase then update $BH$ by adding an increment $\Delta H$ to it;
Else if in the descending phase then update $BH$ by subtracting $\Delta H$ from it;
Generate the silhouette of the base sheet using the updated $BH$;
If in the raising phase then add some droplets along branch $B_i$ above the base sheet stochastically;
Else if in the descending phase then move droplets downward with their size decreasing, delete droplets when their sizes are smaller than the predefined threshold;
```

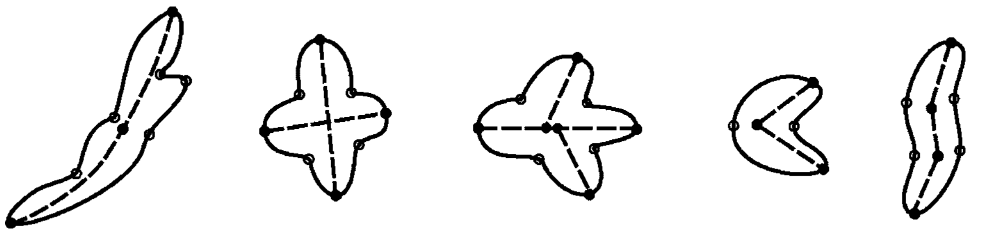

Figure 9. Holes torn inside the base sheet.

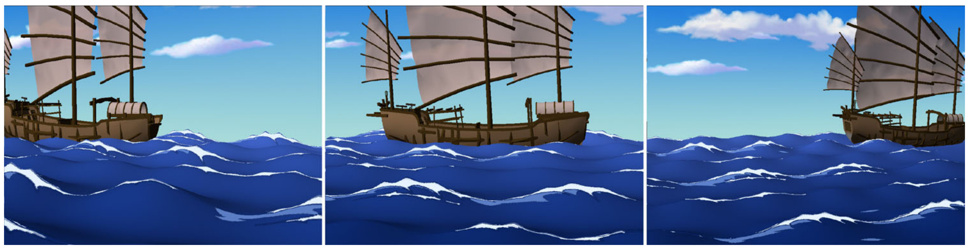
Figure 10. A boat moving on the ocean surface.

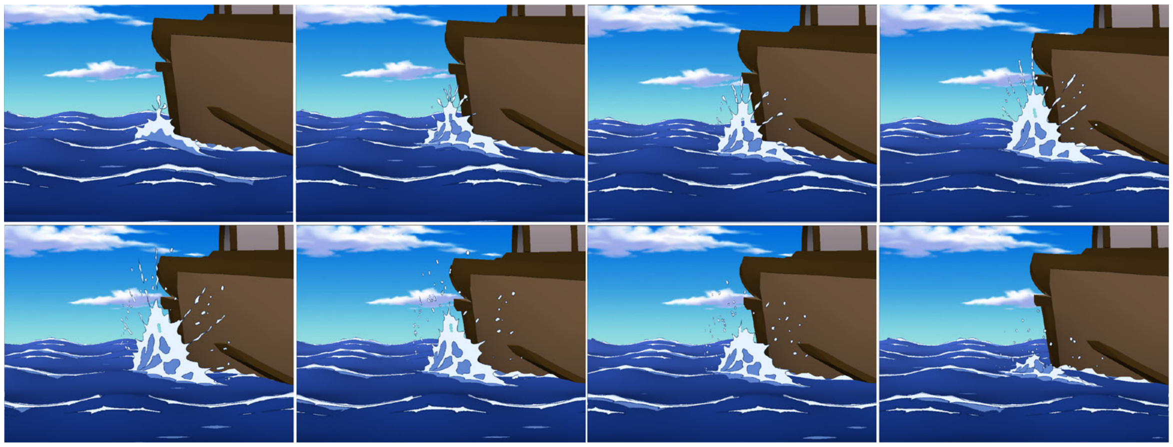
Figure 11. Waves crashing against a moving object.

```
    Place holes inside the base sheet;
    }
    }
```

To animate the crashing wave effects, we first place a curvy surface near the obstacle object. In the animation phase, once the crashing wave is trigged, the crashing wave model mentioned above generates a series of texture images as time t increases, which are then mapped onto the curvy surface near obstacles.

5. RESULTS

To demonstrate the usability of our system, we have created four animations of ocean waves under varying conditions.

The first example is a boat moving over the ocean surface. In the animation we also let the boat sway lightly when it moves from left to right, and the scene is rotated 60° to simulate the camera movement, as shown by a strip of three frames in Figure 10.

The next example is waves crashing against the boat. Ocean waves are moved from left to right and a wave peak is raised progressively in height when it approaches the boat. When the distance between the raised peak and the boat is less than the predefined threshold, the crashing wave effect is triggered and then animated, as shown by 8 frames sampled from the animation in Figure 11.

In the third example we show a different version of our animation for crashing waves. This time we put two rigid objects such as a light house and reef in the water. Ocean waves are animated toward the viewers and the crashing wave effect is triggered when the distance between wave peak lines and the obstacles is smaller than the predefined threshold. Additional forms are put surrounding the bottom of the light house as well as the reef near the viewer. Figure 12 shows a frame of the animation.

Above animations are generated at speeds ranging from 2 to 5 FPS on a 2.8GHz Pentium PC with 1024MB of RAM. The most influence factors of the frame rate are number of components summed in Equation (1), the size of the sea surface grid, resolution of texture image as well as

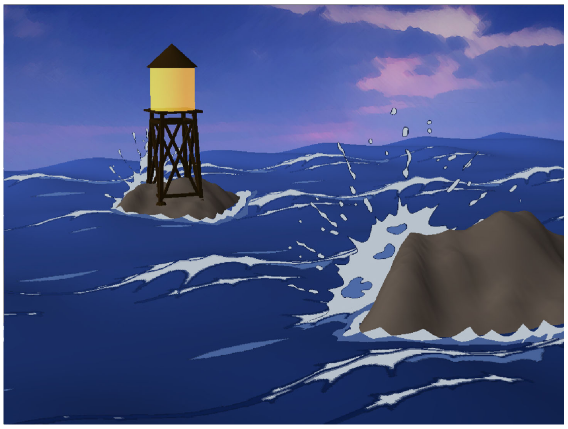
Figure 12. Waves crashing against still objects.

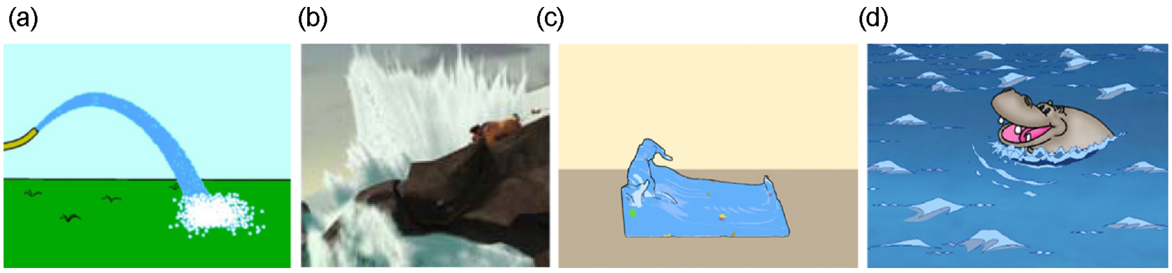

Figure 13. Four examples of other recent work. (a) The splash in Ref. [10]; (b) The garden hose in Ref. [11]; (c) Cartoon liquid animation in Ref. [12]; (d) A moving hippopotamus in Ref. [14].

the number of obstacle objects involved in the interactions with water. In our implementation, the sea surface grid size is set to 256×256 nodes and the resolution of the texture image is 1024×1024 pixels.

Finally we include other recent works of stylized water animation for comparison. The splash using particles (Figure 13(a)), the garden hose (Figure 13(b)), cartoon rendering style for liquid animations (Figure 13(c)), and a moving hippopotamus (Figure 13(d)) are taken from Refs. [10–12] and [14], respectively. In the first two examples particles are used to represent the water, and the generated effects look different from those hand-drawn water effects (Figure 1) in visual style. In the third example the fluid surface is rendered with lines and flat colors, and in the last example the generated cartoon water forms are much less dynamic than those in Figure 1.

6. CONCLUSIONS AND FUTURE WORK

In this paper we present a theme of modeling ocean waves for cartoon animation. Our work is the first attempt to model cartoon ocean waves and animate them procedurally, all water forms are constructed by use of control points, and thus they are scalable and deformable according to the ocean surface as well as obstacles in the scene.

The limitation of our method is that the appearance of whitewater and crashing waves depend on the point of view. This is because the shapes used in the model have stylized structures which are not isotropic in 3D. Nevertheless, this is not a serious problem because in cartoon animation they are often depicted with most expressive content seen from certain view angles and our model seems adequate for the task at hand.

In future work we intend to model more carton water animation including the sea surface as it approaches the littoral, giant breaking waves, etc.

ACKNOWLEDGEMENTS

This work is supported by the State Key Program of National Natural Science Foundation of China (No. 60933007), the Key Technologies R&D Program of China (No. 2007BAH11B02).

REFERENCES

1. Fournier A, Reeves WT. A simple model of ocean waves. In *Proceeding of SIGGRAPH '86*, Vol. 20, 1986; 75–84.
2. Peachey DR. Modeling waves and surf. In *Proceedings of SIGGRAPH'86*, Vol. 20, 1986; 65–74.
3. Ts'o P, Barsky B. Modeling and rendering waves: wave-tracing using beta-splines and reflective texture mapping. *ACM TOG* 1987; **6**(3): 191–214.
4. Gonzato M, Saec BL. On modeling and rendering ocean scenes. *The Journal of Visualization and Computer Animation* 2000; **11**(1): 27–37.
5. Mastin GA, Watterberg PA, Mareda JF. Fourier synthesis of ocean waves. *IEEE Computer graphics and Applications* 1987; **7**(3): 16–23.
6. Tessendorf J. Simulating ocean water. *Siggraph 2001 Course 47*, 2001.
7. Thon S, Ghazanfarpour D. A semi-physical model of running waters. In *Proceedings of Eurographics'01*, 2001; 53–59.
8. Adabala L, Manohar S. Techniques for realistic visualization of fluids: a survey. *Computer Graphics Forum* 2002; **21**: 65–81.
9. Iglesias A. Computer graphics for water modeling and rendering: a survey. *Computer Graphics Forum* 2004; **20**: 1355–1374.
10. Di Fiore F, Claes J, Van Reeth F. A framework for user control on stylized animation of gaseous phenomena. In *Proceedings of Computer Animation and Social Agents'04*, 2004; 171–178.
11. Thornton JD. Directable simulation of stylized water splash effects in 3d space. In *Proceedings of SIGGRAPH'06 (Sketches & applications)*, 2006.
12. Eden AM, Bargteil AW, Goktekin TG, Eisinger SB, O'Brien JF. A method for cartoon-style rendering of liquid animations. Proceedings of Graphics Interface '07, Vol. 31, 2007; 51–55.
13. Yu J-H, Jiang X-N, Yao C, Chen H-Y. Real-time cartoon water animation. *Computer Animation and Virtual Worlds* 2007; **18**: 405–414.
14. Yu J-H, Liao J, Patterson JW. Modeling the interaction between object and cartoon water. *Computer Animation and Virtual Worlds* 2008; **19**(3–4): 375–385.

AUTHORS' BIOGRAPHIES

Jing Liao is a graduate student in the state key lab of CAD&CG, Zhejiang University, China. Her research interests include computer animation and non-photorealistic rendering.

Jinhui Yu is a professor of computer science at Zhejiang University, China. He received his PhD in computer graphics from the University of Glasgow in 1999. His research interests include stylized computer animation and computer graphics art.

John Patterson is a Research Fellow in Computing Science at Glasgow University, specializing in drawn animation and the extension of cartoon technology to image manipulation and film effects technology generally. His academic collaborations include the MTRC at Bath University, Rainbow group at Cambridge University the EDM, University of Hasselt (Belgium) and the State Key lab. of CAD & CG Hangzhou (PR China). His industrial collaborations include work with Cambridge Animation Systems, NFTS CREATEC, Anthropics, Androme (Belgium), and AmaK Studios (France). He was project coordinator 1994–1996 for the ANIMAX project in the Computer CARTOON program and for the IST project CUSTODIEV (2002–2005).