A Procedural Model for Interactive Animation of Breaking Ocean Waves

Stefan Jeschke
University of Rostock, CS Dept., Institute for Computer Graphics, Albert-Einstein-Str. 21, 18051, Rostock, Germany
Stefan.Jeschke@informatik.uni-rostock.de

Hermann Birkholz
University of Rostock, CS Dept., Institute for Computer Graphics, Albert-Einstein-Str. 21, 18051, Rostock, Germany
HB01@informatik.uni-rostock.de

Heidrun Schmann
University of Rostock, CS Dept., Institute for Computer Graphics, Albert-Einstein-Str. 21, 18051, Rostock, Germany
Schumann@informatik.uni-rostock.de

ABSTRACT

This paper presents a procedural model for breaking ocean waves that is intended to be used for interactive visualization. The movement as well as the appearance of the waves is modelled by a set of functions in dependence of time and space. This continuous surface description allowes it to calculate all properties of every point (including foam) on the ocean surface at every time without any information from previous time steps. By using an adaptive sampling sheme for rendering, the frame rate of the animation only depends on the screen resolution rather than on the model size. The model is quite simple, easy to implement, fast to compute and provides a visual appealing interactive animation of infinite large ocean coast scenes. On the other hand it provides only limited flexibility due to its procedural character. For achieving more realistic scene appearance, it may also easily be combined with models for deep-water waves presented in the past.

Keywords

interactive, rendering, procedural modelling, ocean modelling, animation, water waves

1. INTRODUCTION

The modelling and visualization of ocean scenes has been a challenge in computer graphics for a long time. This paper focusses on the special case of interactive (this means at least 5 frames per second) visualization of plunged breaking waves in infinite large ocean coast scenes. The goal is a simple to implement and fast to compute model for producing high output frame rates and reasonable image quality. Figure 1 shows an output image generated by using this model (in combination with wave ripples modelled with sinusoids).

The basic principle implemented here for achieving these goals is a procedural model. This means, every point in the ocean scene is described by some simple formulas in dependence of time and space. The ocean animation is then restricted to a continuously changing parametrization for the formulas.

WSCG POSTERS Proceedings
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

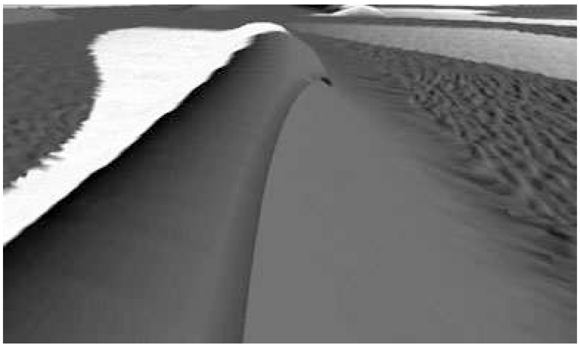


Figure 1: Output image generated with the presented method and wave ripples (sinusoids).

The advantages of a procedural approach are a continuous surface description in time and space so that the location and appearance of every point in the scene can always be calculated without using information from previous time steps. Note especially, that even foam generated by the breaking waves will also be computed without recomputation of values obtained in previous time steps (in contrast to particle systems). Furthermore, the model can easily be used in combination with wave models for deep-water waves and fine rippling waves to enhance the realism.

On the other hand, a procedural model has the limitation that the oceans appearance and behaviour is completely defined by the author (such as wave refraction on the cost). This implies a reparametrization of the model for changing environment conditions (for instance a different coastline). Because a reparametrization done always by hand is not desireable, future work should focus on this to provide a more flexible use of the model.

For displaying the ocean surface, a polygon mesh is generated by using the image space sensitive sampling method presented by Hinsinger [Hin02]. By using this rendering method, the output frame rate only depends on the screen area covered by the ocean field (i.e. the number of sampling points). Efficient use of current graphics hardware is provided due to the use of polygon strips.

Since the model is easy to implement and provides fast output image computation, its main applications are virtual environments and computer games (also because simple collision detection is possible) as well as multimedia applications (think about a fly over an infinite large ocean surf scene).

2. RELATED WORK

Early approaches in the field of ocean modelling and visualizing by Max [Max81], Fournier [Fou86], Peachey [Pea86] and Tso [Tso87] were able to produce fairly realistic results for relatively quiet ocean surfaces (also called "deep-water waves") but plunged breaking waves could not be modelled correctly due to the sinusoidal assumption in the parametric surface and/or the use of a high field wave representation. A good introduction and overview of that work is given in the SIGGRAPH 2001 course notes [Tes01].

More recent work by Jensen [Jen01] uses different wave modelling approaches for different levels of detail for interactive deep-water animation. There was also presented a texture-based method for rendering foam (that is also used in this paper) and show clever use of current graphics hardware to achive more realism.

Extensive use of programmable graphics hardware was also done by Schneider [Sch01]. Here it was used for displacement, transformation and lighting calculations of a height field water surface for realizing effects such as refraction, reflection and the Fresnel term.

Smith [Smi02] used in his diploma thesis surface markers to track a wave surface for interactive animating curling and breaking (including plunged) waves arriving at a coast.

The most closely related work to this paper was made by Hinsinger [Hin02]. Here, procedural waves are used to model an infinite large deep-water surface. The surface is rendered by using an adaptive sampling method that completely decouples the output frame rate from the size of the ocean scene. This paper can be seen as an extension of that work for handling breaking waves and the resulting foam.

3. A PROCEDURAL WAVE FIELD

For further descriptions the basic coordinate setup illustrated in figure 2 is used (z is pointing up). The intial assumption for the procedural model is that all waves are straightly running towards the beach.

For realistic wave behaviour, the phenomenon of *wave refraction* is modelled. This includes a slowing down of the wave when arriving the beach as well as a beach alignment.

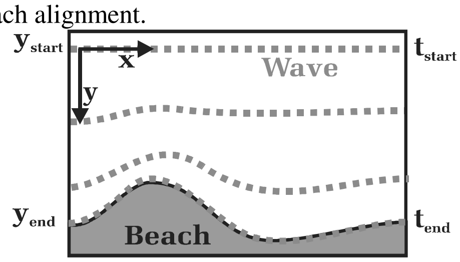


Figure 2: Setup for the procedural wave field

At first, a parameter s running from 0 to 1 over the wave's life time is defined by using the time of birth (t_{start}) and dead (t_{end}) of the wave and the current time ($t_{current}$):

$$s = \left(\frac{t_{current} - t_{start}}{t_{end} - t_{start}} \right)^r$$

Here r defines the amount of deceleration and alignment to the coastline over time. The desired y position ($y_{current}$) for the current time ($t_{current}$) is then obtained by using:

$$y_{current} = (1-s)y_{start}(x) + s\, y_{end}(x)$$

y_{start} is here a constant function so that the waves start as straight lines, whereas other functions are of couse also possible. y_{end} defines the coastline. It can be defined by using a function in dependence of x (for instance a superimposition of sinus functions) or by using cubic interpolation of sampling points. Slightly varying values for r and/or y_{end} let every wave run a bit on its own which gives a more natural look.

The second phenomenon modelled here is the *wave breaking*. Normally, a wave begins to break at several points and then successively breaks over its whole width. This is modelled here by using a simple function $t_{break}(x)$ that defines the time the wave breakes for every point in x direction (for instance also a superimposition of sinus functions). Again, using slighly varying values for the function parameters lets every wave look unique.

3.1 Modelling Foam

The foam modelled here is produced by breaking waves when the water from the top crashes into the water at the bottom. Afterwards, foam slowly disappears when the milliards of small bubbles disappear.

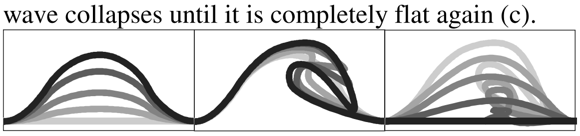
Because the breaking time for every wave is always known from its function $t_{break}(x)$, it is possible to compute the amount of foam for every point at the ocean surface at every time. For estimating the foam amount at a given point (x,y) at the current time $t_{current}$, all waves recently passed y are considered. For every wave, the exact time t_{wave} when it passed y is estimated by reorganizing the two functions above to $t_{current}$:

$$t_{wave} = t_{start} + (t_{end}-t_{start})\sqrt[r]{\frac{y-y_{start}}{y_{end}-y_{start}}}$$

If the wave produced foam at that moment (this can simply be tested by using the function for breaking) the foam amount is faded over time by using a function that uses as input the time difference between t_{wave} and $t_{current}$ (for the simplest case, this is a linear function). Finally, the highest foam value from all considered waves is taken as the actual foam value for that point.

4. PROCEDURAL WAVE SHAPES

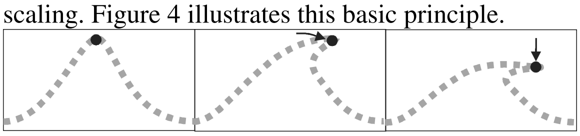
For procedural modelling the following basic "life cycle" of a breaking wave is considered (refer to figure 3). When a wave is born, it comes up from ocean level and has a round shape (a). When it breaks, the front part dents inside and the top part falls down thus forming a tube (b). Afterwards, the wave collapses until it is completely flat again (c).



(a) (b) (c)

Figure 3: Phases of the life cycle of a breaking wave.

The basic idea for procedural modelling the wave shapes is to use a combination of four functions: cosines function, exponential function, rotation and scaling. Figure 4 illustrates this basic principle.



(a) (b) (c)

Figure 4: Example for a procedural wave shape. (a): combination of cosine, exponential and scaling function; (b): rotation; (c): scaling.

For wave animation, the function parametrization is blended over time. The formulas and parametrization presented here are obtained by experiment using only visual control, whereas physically-based wave shape modelling would of course also be possible here. The functions map an input space parameter s ($0 \leq s < 1$, running from the back to the front part of the wave) onto the respective (y,z) position at the ocean surface in dependence of the current time.

	round		**breaking**		**collapsing**	
	front	**back**	**front**	**back**	**front**	**back**
k_1	1	1	2	$\frac{7}{10}$	2	$\frac{1}{2}$
k_2	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1-t^{0.85}}{4}$	$\frac{1}{4}$	0	$\frac{1}{4}$
k_3	0	0	$\frac{t^{0.85}}{4}$	0	$\frac{1-t}{4}$	$\frac{t^{0.7}}{2}$
k_4	0	0	4	0	4	40
k_5	0	0	$t^{\frac{3}{2}}$	$t^{\frac{3}{2}}$	1	1
k_6	0	0	16	4	16	4
k_7	$\frac{4t}{5}$	$\frac{4t}{5}$	$\frac{4+t}{5}$	$\frac{4+t}{5}$	1-t	1-t

Table 1:Constants for procedural wave shape description

Because the front and the back part of the wave have a different behavior, s is splitted for the front and back part (the wave lip is always defined by s=0.5). For every part the functions are parameterized separately. Therefore s_1 and s_2 are used where s_1 runs from 0 to 0.5 (or respectively 0.5 to 1) and s_2 always runs from 0 to 1. For the first case the two values are:

$$s_1 = \frac{(2s)^{k1}}{2} \quad ; \quad s_2 = (2s)^{k1}$$

For the second case (s is between 0.5 and 1) the values are:

$$s_1 = \frac{1+(2s-1)^{k1}}{2} \quad ; \quad s_2 = 1-(2s-1)^{k1}$$

The constant k_1 is introduced for a possibly uniform parametrization of the wave, which is desireable for the rendering process. The values for k_1 can be extracted from tabular 1 for the different wave phases. With the values s_1 and s_2, the following calculation is used to obtain the coordinates (y,z) for the wave shape:

$$z_1 = \frac{k_2(1+\cos((s_2-1)\pi))}{2} + k_3 {s_2}^{k4}$$

$$\phi = \frac{\pi k_5 {s_2}^{k6}}{2}$$

$$y = \left(\frac{1}{2}-s_1\right)\cos(\phi) - z_1\sin(\phi) + \frac{1}{2}$$

$$z = \left[\left(\frac{1}{2}-s_1\right)\sin(\phi) + z_1\cos(\phi)\right]k_7$$

For a smooth animation the constants k_i are linearly blended over the three different phases of the life cycle in dependence of a time parameter t that runs from 0 to 1 during every phase. Table 1 shows all constants k_i for the front and back sides for every phase.

5. RENDERING ISSUES

The use of todays graphics hardware is an obvious choice for achieving interactive frame rates. For doing so the ocean scene is sampled and triangulated as was presented by Hinsinger [Hin02]: sampling points are uniformly distributed on the screen (i.e. every n-th pixel is a sampling point) and a ray from the center of projection is shot through every sampling point on the image plane. The intersection point from the ray with the ocean level plane is computed and for the resulting point the ocean surface position is estimated. By using this technique, perfect uniform sampling is obviously not guaranteed and aliasing also happens but both effects are normally not visible due to the moving nature of the scene. The resulting polygon mesh is a full polygonal description of the ocean surface, including closed wave tubes. It is stripped to further accelerate the rendering process.

As was presented by Hinsinger [Hin02], the normal vectors per vertex needed for polygon shading can be computed analytically by spatial derives of the formulas describing the wave. Because this is quite complex for breaking waves, the normal vectors are calculated here by accumulating the normals from all adjacent faces (obtained by vectorproduct) and normalizing the result.

Finally, the actual foam density value is calculated per vertex as was described in section 3.1. Foam is visualized as a tiled transparent texture applied to the polygons forming the ocean surface as was also presented by Jensen [Jen01]. The texture opacity is set for every vertex with respect to its current foam amount value. The use of an animated texture for foam may further enhance realism.

6. CONCLUSIONS

The callenge of this work is a procedural method for breaking wave modelling to be used for simple interactive ocean animating. The appearance of the ocean scene can be computed everytime without the need for transfering information over time. Furthermore, in combination with a rendering method that uses adaptive sampling, the output frame rates are decoupled from the size of the ocean scene. The proposed method was implemented by using C++ and OpenGL on a Pentium4 running at 1800 MHz and NVIDIA Geforce III graphics hardware. 10 breaking waves were animated. A screen resolution of 1024x768 pixels and a sampling point distance of 4 pixels (being a good tradeoff between high output frame rates and good image quality) results to approximately 50000 sampling points and a frame rate of 35 frames per second.

On the other hand, in a procedural model everything has to be modelled by hand so that much work has to be done to obtain a more realistic model. This includes the support of different wave forms (for instance spilled breaking waves) and allowing different wave sizes and a more complex wave behaviour. Furthermore, assuming a given ocean coast scene, a mapping operation that automatically adapts the parametrization for the breaking waves (for instance for wave refraction) would be highly desirable.

Many more components have to be included to enhance the realism. The presented model can easily be combined with models for deep-water waves (including sinusoids as in figure 1, trochoids or FFT-based approaches ([Tes01])) and small wave ripples modelled as animated bump maps ([Jen01]). Aside from the waves, environment reflection maps can be used to model sky reflections on the ocean surface by using graphics hardware ([Sch01]). Finally, spray produced by the breaking waves should be modelled. Because a particle simulation would be to costly for large scenes, time dependend functions that define the particle movements should be used for that.

7. REFERENCES

[Fou86] Fournier, A., and Reeves, W.T. A Simple Model of Ocean Waves. In ACM SIGGRAPH Proceedings, Vol.20, No.4, pp.75-84, 1986.

[Hin02] Hinsinger, D., Neyret, F., and Cani, M.P. Interactive Animation of Ocean Waves. In Symposium on Computer Animation, 2002.

[Jen01] Jensen, L. S., and Golias, R. Deep-Water Animation and Rendering. In Gamasutra September 2001.

[Max81] Max, N. L. Vectorized Procedural Models for natural terrain: Waves and Islands in the Sunset. In Computer Graphics, Vol.15, pp.317-324, 1981.

[Pea86] Peachey, D.R. Modelling Waves and Surf. In ACM SIGGRAPH Proceedings, Vol.20, No.4, pp.65-74, 1986.

[Sch01] Schneider, J., and Westermann, R. Towards Real-Time Visual Simulation of Water Surfaces. In Proceedings of the Vision Modelling and Visualization Conference 2001, pp.211-218, 2001.

[Smi02] Smith, B.W. Realistic Simulation of Curling and Breaking waves. Masters Thesis, www.csee.umbc.edu/~bsmith15/799/thesis.pdf, 2002.

[Tes01] Tessendorf, J. SIGGRAPH 2001 Course Notes, Course 47: Simulating Nature: Realistic and Interactive Techniques, ACM SIGGRAPH, 2001.

[Tso87] Tso, P. Y., and Barsky, B.A. Modelling and Rendering Waves: Wave-Tracing Using Beta-Splines and Reflective and Refractive Texture Mapping, In ACM Transactions on Graphics, Vol.6, No.3, 1987, pp.191-214, 1987