Modeling Waves and Surf

Darwyn R. Peachey

Department of Computational Science
University of Saskatchewan
Saskatoon, Canada

ABSTRACT

Although modeling natural phenomena is recognized as one of the greatest challenges of computer graphics, relatively little time has been spent on modeling ocean waves. The model presented in this paper is suitable for the rendering and animation of waves approaching and breaking on a sloping beach. Waveforms consist of a phase function which correctly produces wave refraction and other depth effects, and a wave profile which changes according to wave steepness and water depth. Particle systems are used to model the spray produced by wave breaking and collisions with obstacles. A scanline algorithm for displaying the wave surface is presented, along with a method of integrating separately rendered particle systems with other surfaces. Hidden surface removal for both waves and particles is done using a novel variation of the A-buffer technique. Methods of implementing the model are presented and compared with previous rendering techniques.

CR Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism.

Additional Key Words and Phrases: A-buffer, clamping, particle systems, stochastic modeling, surf, water, wave refraction, waves.

1. Introduction

Modeling natural phenomena has always been among the most challenging problems in computer graphics, because natural phenomena have an inherent complexity far beyond that of most man-made objects. Significant progress has been made in modeling a variety of phenomena, including terrain [5], clouds [7], fire [14, 17], trees [1, 18, 19], and grass [18]. Relatively little time has been spent in modeling the appearance and behavior of the oceans.

Turner Whitted, in his film "The Compleat Angler", was among the first to attempt to render waves in water. Using ray tracing [21] Whitted animated realistic reflections from ripples in a small pool.

The ripples were created by bump mapping the flat pool surface, perturbing the surface normal according to a single sinusoidal function [22]. Another early effort was the Pyramid Films leader produced by Information International in 1981, which used a similar technique with cycloidal waveforms. More recently, Ken Perlin [14] has used bump mapping with a richer texture map to convincingly simulate the appearance of the ocean surface as one might see it from an aircraft well out to sea. Perlin used a set of 20 cycloidal waveforms, each radiating in a circular fashion from a randomly placed center point.

Although bump mapping is inexpensive and has been effectively used to simulate waves in the cases cited above, bump mapping is not sufficient to simulate waves in general. Since the actual surface is flat, bump mapped waves do not exhibit realistic silhouette edges or intersections with other surfaces. Another limitation of bump mapped waves is that they cannot shadow one another or cast shadows on other surfaces.

To avoid these shortcomings, Nelson Max [10] used a "height field" algorithm to render explicitly modeled wave surfaces for his film "Carla's Island". His wave model consisted of several superimposed linear sinusoidal waves simulating ocean waves of low amplitude.

In this paper we present a model of ocean waves which is capable of simulating the appearance and behavior of waves as they approach a sloping beach, steepening, breaking, and producing a spray of water droplets from the crests of the waves. Wavefronts are correctly refracted by the transition to shallow water, so that they align themselves parallel to the beach. The model is also capable of simulating spray resulting from the collision of waves with partially submerged obstacles. To our knowledge, no computer graphics model of these phenomena has previously been presented, although Fournier and Reeves [6] are involved in similar research.

2. Wave Fundamentals

A rich body of theory and observation exists concerning the behavior of waves in general, and water waves in particular [9, 11, 20]. There are several classes of water waves observed in the ocean:

- the tides, which have very long wavelengths and periods.
- seismic waves (tsunamis)
- internal waves
- surface gravity waves
- capillary (surface tension) waves, which have very short wavelengths and periods

Surface gravity waves and capillary waves generally result from the action of wind on the surface of the water, so they are collectively called "wind waves". Since our goal is to produce realistic pictures of waves and surf as they might be seen from the beach, we will deal only with surface gravity waves, which are usually the most noticeable waves on the beach during any short period of observation. (Small capillary wave ripples may be bump mapped on the wave surface for additional detail.)

The simplest surface gravity wave in water is a sinusoidal function of the form:

$$(1)\quad f(x,t) = A\cos\left(\frac{2\pi(x - Ct)}{L}\right)$$

where x is the distance from an origin point, A is the amplitude of the wave, L is the wavelength, C is the propagation speed, and t is the time. The period, T, is the time between successive crests of the wave passing a particular point. The wavelength, period, and speed are related by the equation $C = L/T$. The frequency of the wave is $1/T$. The wave number of the wave, which is the spatial analog of the frequency, is $\kappa = 2\pi/L$. The magnitude of a water wave is often specified in terms of its height, H, rather than its amplitude, where $H = 2A$ for the simple sine wave. The steepness of a wave is the ratio $S = H/L$.

The motion of the wave must be distinguished from the motion of the water through which the wave is propagating. While the wave moves past a given point, the water at that point moves in a circular or elliptical orbit as shown in Figure 1. The water in the crest of the wave moves in the same direction as the wave, while the water in the trough of the wave moves in the opposite direction. The net motion of the water is zero in an ideal sinusoidal wave. The water must complete one orbit in the same time that it takes for a complete wavelength of the wave to pass a given point, namely the period of the wave, T. Since the diameter of the orbit is H, the average orbital speed of the water is:

$$(2)\quad Q_{avg} = \frac{\pi H}{T} = \frac{\pi HC}{L} = \pi SC$$

A wave breaks at the crest when the orbital speed Q of the water at the crest exceeds the speed C of the wave itself. This limits the steepness of a stable wave, since Q grows as S grows. Typical values for the steepness of ocean waves have been observed to be between 0.05 and 0.1.

Various hydrodynamic models of wave motion have been constructed by assuming that sea water is a perfectly nonviscous, incompressible fluid. The general model is nonlinear and has no convenient solutions [20]. Therefore, many simplified or idealized models are used for various situations. One of the more commonly used hydrodynamic models is the Airy model of sinusoidal waves of small (negligible) amplitude. The Airy model is linear, and predicts that the propagation speed and wavelength of a wave will depend on the depth of the water, d, as follows:

$$(3a)\quad C = \sqrt{\frac{g}{\kappa}\tanh(\kappa d)} = \sqrt{\frac{gL}{2\pi}\tanh\left(\frac{2\pi d}{L}\right)}$$

$$(3b)\quad L = CT$$

where g is the acceleration of gravity at sea level, 9.81 m/sec^2. In deep water, $\tanh(\kappa d)$ approaches 1, so C approaches $gL/2\pi$. In shallow water, $\tanh(\kappa d)$ approaches κd, so C approaches $\sqrt{gd}$. To achieve five percent accuracy in these approximations, it is sufficient for "deep" to mean $d \geq L/4$ and for "shallow" to mean $d \leq L/20$ [9].

As surface gravity waves are driven to large amplitudes by the wind, their shapes change, with the crests becoming more sharply peaked and the troughs becoming shallower and flatter. The Gerstner/Rankine wave model [9] gives an exact solution to the hydrodynamic equations for a wave of non-negligible amplitude in deep water. This model predicts a trochoidal or cycloidal waveform, approaching sinusoidal shape when the wave steepness is small. Another popular model is the Stokes wave model [9, 20] which is an infinite Fourier series which resembles the trochoidal wave up to the third order terms. The Stokes model predicts a slight dependence of wave speed on steepness, and a maximum wave steepness of 0.142. Nelson Max [10] used only the first-order term of the Stokes model for most of his waves, and used the first and second-order terms for the wave with the largest amplitude.

As waves approach the shore from deep water, the crests of the waves tend to become parallel to the shoreline regardless of their initial orientation. This is a result of *wave refraction*, a bending of the waves due to the dependence of propagation speed on the depth of the water. Since waves move more slowly in shallower water, the part of a wave which enters shallow water first will be retarded, and the remainder of the wave which is still in deeper water will move faster. This tends to turn the wave crest to be parallel to the line of transition to shallower water. Wave models which ignore refraction may produce "impossible" situations, such as the wave crests running perpendicular to the beach in [10].

Although the speed and wavelength of a wave are reduced as it enters shallower water, the period remains constant and the amplitude remains the same or increases slightly. The orbital speed of the water, which is directly related to the period and amplitude of the waves, stays the same even as the wave speed declines. This leads to a change in the shape of the waves, with the front of the crests becoming steeper and eventually breaking when the speed of the wave drops below the orbital speed of the water. When breaking occurs, droplets of water moving faster than the wave leave the wave surface in the form of spray. The existing hydrodynamic wave models do not adequately describe the breaking of waves.

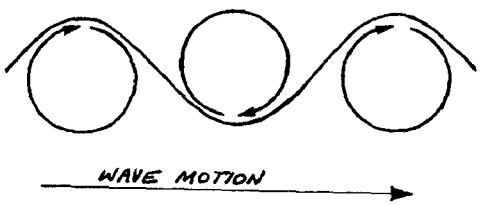

Figure 1: Orbital motion

3. A Model of Waves and Surf for Computer Graphics

The goal of our model of waves and surf is to allow us to synthesize convincing images of ocean waves as they might be seen on a beach. Moreover, the model must be suitable for animation, since it is the motion of waves and spray which give the strongest impression of realism to the viewer. Since no existing hydrodynamic model can claim to fully and realistically describe the behavior of any real ocean waves, we will not attempt to use such a model directly. However, we will depend on the predictions of the Airy model for the relationship between the depth of the water and the speed and wavelength of waves (even though many of our waves are neither sinusoidal nor of small amplitude).

3.1 Basic Model

We represent the ocean surface as a single-valued function of three variables:

$$y = f(x, z, t)$$

where (x, y, z) is the usual 3D Cartesian modeling space with the Y axis directed upwards, and t is the time, which is advanced for each successive frame of an animation. The use of a single-valued height function for the wave model means that we are prevented from producing waves whose crests actually curl forward. On the other hand, this representation of the surface permits us to easily combine numerous waves into one surface by superposition (simply adding together the heights of the individual wave components).

Our wave function f is a sum of several long-crested linear waveforms W_i with amplitudes A_i propagating in various directions from various origin points:

$$(4)\quad f(x, z, t) = \sum_{i=1}^{n} A_i W_i(x, z, t)$$

Instead of using W_i directly, we prefer to split W_i into a composition of two functions:

$$W_i(x, z, t) = w_i(\mathit{fraction}\left[\theta_i(x, z, t)\right])$$

The functions w_i are called *wave profiles*, and are single-valued periodic functions of one parameter with a value between 0 and 1. This parameter value is the fractional part of the *phase* function, $\theta_i(x, z, t)$. The separation of W_i into a wave profile and a phase function makes it easier to describe and change the wave profile to give different wave shapes. It also allows us to address phase-related problems without concern for the final shape of the waveform. We discuss the phase function in the next section, and delay the discussion of wave profiles until section 3.3.

Each wave component can be completely characterized by giving its period T_i, its amplitude A_i, its origin, and its direction. Since each component may have a different origin and direction, the phase functions θ_i have a simpler form if expressed in a per-component coordinate system in which the wave starts at the coordinate system origin and propagates in the +X direction. A simple 2D transformation may be used to convert the coordinates of the point (x, z) into the corresponding coordinates $(\bar{x}_i, \bar{z}_i)$ in the coordinate system of component i.

3.2 The Phase Function

The phase function has a very simple dependence on the time t. Just as each wave component has the same constant period T_i at all points in space, it is also true that the wave component has the same constant rate of phase change at all points in space, namely the frequency:

$$\frac{\partial \theta_i}{\partial t} = -\frac{1}{T_i}$$

(The negative sign of the derivative is necessary to make the waves propagate in the direction of increasing phase values.) If we know the phase at a particular time $\theta_i(x, z, t_0)$ for all points (x, z), we can compute the phase for any frame of an animation by using the rule:

$$(5)\quad \theta_i(x, z, t) = \theta_i(x, z, t_0) - \frac{t - t_0}{T_i}$$

We will ignore the time-dependence of θ_i in the following discussion, and describe the phase at a fixed time t_0, assuming that the phase is 0 at the component origin point at time t_0 (which we can arrange, if necessary, by moving the origin point).

Unfortunately, the dependence of the phase on (x, z) is not nearly so tractable. It is necessary for our graphics model to include the effects of depth on wavelength and speed, in order to produce realistic wave refraction effects, such as the alignment of crests parallel to the beach, and in order to accurately depict the motion of the waves in shallow water. The implication of the dependence of wavelength and speed on depth is that the phase function depends on the cumulative effects of the depth of the water between the wave origin and the point of interest. In general, there is no simple expression for the phase function in water of varying depth. If the depth of the water is constant, then according to the Airy model (equation 3) the wavelength and speed are also constant. In this case, the phase function (expressed in the per-component coordinate system) is simply:

$$(6)\quad \theta_i(\bar{x}_i, \bar{z}_i) = \frac{\bar{x}_i}{L_i}$$

Since the wavelength L_i is variable in water of varying depth, it appears that the phase function must be evaluated as an integral of a depth-dependent phase-change function over the distance from the origin to the point of interest. In one-dimensional terms:

$$(7)\quad \theta_i(\bar{x}_i) = \int_0^{\bar{x}_i} \theta_i'(u)\, du$$

In the constant-depth case, the derivative is:

$$(8)\quad \theta_i'(u) = \frac{1}{L_i} = \frac{1}{C_i T_i}$$

where C_i or L_i may be computed from the Airy model of equation 3. Note that this equation may be obtained by differentiating equation 6.

Without justification, we use the same derivative (phase-change function) in water of varying depth. We must integrate this function as suggested by equation 7 in order to obtain a phase function will which give us the wave refraction effects we want. We numerically integrate from the origin of each component along the direction of propagation in deep water to obtain a grid of phase values. The grid for each component is stored in a file and is loaded during rendering of the wave surface in order to look up phase values for the component. Bilinear interpolation among grid values is used to produce phase values between grid points. This interpolation procedure does not seem to produce objectionable artifacts in the final images, because the interpolated phase is not rendered directly, but rather is used as the parameter of the wave profile function. Of course, a time adjustment for the particular frame being rendered must be subtracted from the phase (equation 5). It should be emphasized that the expensive numerical integration of the phase function for each component is done only once, not once for each frame.

3.3 Wave Profiles

The wave profile functions introduced in section 3.1 are single-valued periodic functions of one parameter:

$$w_i(u),\ 0 \le u < 1$$

The values of w_i are interpreted as vertical displacements of the ocean surface from the rest position. In order for the wave amplitude A_i to have its desired effect, it is normally required that the values of w_i range over the interval $[-1, 1]$. The crest of the wave is conventionally located at $u = 0$, so $w_i(0) = 1$. In order that

the wave surface be continuous, it is required that the function be continuous on [0, 1) and that:

$$\lim_{u \to 1} w_i(u) = w_i(0) = 1$$

The simplest method of handling wave profiles is to use a fixed function for all w_i in all situations. For example, the function $w_i(u) = \cos(2\pi u)$ meets all of the requirements for a wave profile and will give a simple sinusoidal wave shape.

For greater realism, the wave profile function is changed according to the wave steepness, S, and the ratio, δ, between the depth of the water and the deep-water wavelength L_i^{deep}.

$$L_i^{deep} = \frac{gT_i^2}{2\pi}$$

$$\delta = \frac{d}{L_i^{deep}}$$

The steepness controls a linear blending between a sinusoidal function (when the steepness is small) and a sharp-crested quadratic function (when the steepness is large). The sharp-crested function is:

$$w_i(u) = 8\,|u - \tfrac{1}{2}|^2 - 1$$

This function superficially resembles a cycloid; we do not use an actual cycloid because it has no convenient formulation as an explicit function of the phase. The change of the wave profile as a function of steepness produces a realistic change in appearance from long smooth swell with rounded crests to short choppy waves with sharp crests.

The depth ratio, δ, controls the asymmetry of the wave profile. When δ is large, the wave profile is evaluated normally. When δ is small, the parameter u is exponentiated to shift its values toward the low end of the interval [0, 1). This has the effect of steepening the front of the wave crest and stretching out the back of the crest. Similar asymmetry in wave profiles is easily observed as waves enter the shallow water near a beach and approach the breaking point. Figure 2 illustrates the different wave profiles produced by this model.

Since waves break and dissipate much of their energy as they enter very shallow water, our wave model reduces the amplitude of each wave component so that the vertical displacement of the ocean surface never exceeds the depth of the water. This reduction only takes effect when the depth of the water is comparable to the sum of the various amplitudes A_i.

3.4 Spray

Spray from breaking waves and from the collision of waves with partially submerged obstacles is an important aspect of modeling waves and surf. Particle systems [17] are a natural mechanism to use to simulate the behavior of the population of water droplets which make up the spray.

As mentioned earlier, waves break when the speed of circular motion of the water in the crest of the wave exceeds the speed of the wave itself. Based on the assumption of uniform circular motion, breaking would occur when $Q_{avg} > C$. From equation 2, $Q_{avg} = \pi SC$, so the condition becomes $\pi S > 1$ or $S > 1/\pi$. Clearly the assumption of uniform circular motion is incorrect, since the maximum steepness actually observed is much smaller, in the neighborhood of 0.1. The water in a steep wave moves faster in the crest than would be predicted by the uniform motion assumption. A "corrected" particle speed Q is computed by multiplying Q_{avg} by the factor $1/\pi S_{max}$ where S_{max} is the desired maximum steepness at the breaking point.

Generation of the particle system for a given breaking wave crest is relatively straightforward. The initial position of each particle is at the crest of the wave. The initial velocity of the particle is in approximately the same direction as the wave motion, with a speed of Q. A stochastic perturbation with a Gaussian distribution is added to the particle velocity to avoid excessively uniform particle behavior. An increasing number of particles is generated as the speed differential between the wave speed and Q increases.

Notice that the criterion for the generation of spray due to breaking is entirely dependent on wave steepness and not on the depth of the water. Usually breaking will result from the reduction in wavelength and consequent steepening when a wave enters shallow water. In this case, the generation of spray will be accompanied by an asymmetric wave profile as described in the previous section. However, it is also possible for waves to break in deep water when the amplitude becomes sufficiently large (storm conditions). In this case, the wave profiles will be symmetrical and sharp-crested, approximating a cycloid.

A similar particle system model is used to simulate the spray resulting from waves striking obstacles (rocks, piers, etc.) along the beach. Particles are generated when the crest of a wave is near the seaward face of the obstacle. The crest is the relevant part of the wave for spray generation because the water in the crest is moving toward the beach with its maximum speed, and therefore generates the most spectacular spray.

The rate of particle generation increases from zero at a time slightly before the crest meets the obstacle, to a maximum value when the crest is at the obstacle, and then falls back toward zero as the crest passes the obstacle. The initial positions of the particles lie approximately on the curve along which the wave surface intersects the obstacle. The initial velocities of the particles are chosen stochastically, with a magnitude that is a constant fraction of the speed Q at which the water is striking the obstacle, and a direction which is the reflection direction for an ideal elastic collision, plus a Gaussian perturbation (Figure 3).

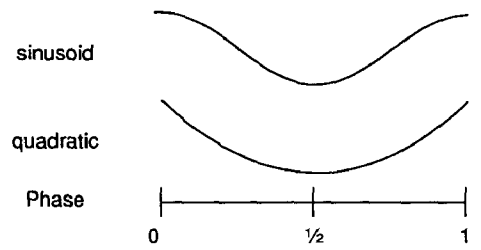

Figure 2: Wave profiles

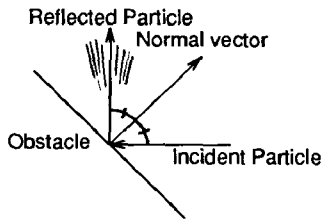

Figure 3: Obstacle impact

Once a particle has been generated as a result of wave breaking or obstacle impact, its behavior is simulated according to gravitational kinematics as described in section 4.3.

3.5 Designing the Beach

Since so much of the wave and spray model depends on the depth of the water at a given point, the final images are greatly affected by the shape of the terrain which makes up the beach, and especially by the submarine contours of the terrain. The images in this paper are based on a hypothetical beach called "Babbage Beach". Figure 4 is a topographic map of Babbage Beach with yellow indicating high cliffs, green indicating low beach, white and light blue indicating shallow water, and darker blue indicating deep water. The water at the top of the map averages 10 meters deep. Adjacent lines of the black grid are 100 meters apart. The red bands indicate the crests of a wave component with a period of six seconds. Wave refraction effects are clearly visible in the shape of these bands. Figure 5 is a perspective view of the terrain without water.

The Babbage Beach terrain was entered as a coarse grid of manually generated elevations. This data was smoothed and interpolated with some stochastic variation to produce a much finer grid of elevations. At present, the resulting terrain is rendered as a collection of triangles with the grid points as vertices. Bilinear interpolation is used to determine the elevation at an arbitrary point for use by the wave and spray model. Alternatively, an interpolating spline surface could be passed through the grid points for smoother interpolation and a more realistic appearance.

If a beach is to be designed without simply digitizing a contour map of a real beach, some study of coastal geomorphology [15] is useful in determining realistic landforms and submarine contours. The slope of the beach depends on the type of sand particles or rocks which make up the beach. Larger particles lead to steeper beaches; a slope between 1:10 and 1:30 is typical for sandy beaches, while beaches made of stones or slate-like "shingle" can be as steep as 1:2 or 1:3. The nature of the beach also depends on the steepness of the large waves which strike it. Steep winter storm waves tend to erode the above-water part of the beach while building up underwater sand bars. The lower waves of the summer season move the sand landward and build up an above-water hill called a "berm".

4. Rendering

The model described in section 3 has been implemented as part of two image synthesis systems, a ray tracing system called "Portray" [13] and an A-buffer-like system called "Pixie". The ray tracing implementation does not include the particle system model of spray.

4.1 Ray Tracing Waveforms

In the Portray ray tracing system, a modified *regula falsi* iterative root-finding technique was used to solve the ray-surface intersection equation directly. The advantage of ray tracing as a means of rendering the wave surface is that it easily and correctly handles the images of the sky and of objects reflected in the water. However, these reflections usually can be simulated by the use of reflection environment maps in non-ray tracing systems. The surface of the ocean is usually so rough that reflections cannot be seen very clearly, and an approximation to the correct reflection will suffice. The disadvantages of ray tracing are its immense cost (nearly 30 CPU hours on a Pyramid 90x for some wave surfaces), the difficulty of rendering complex procedural models such as particle systems, and the tendency for aliasing to arise from point sampling distant waves near the horizon. Stochastic sampling is a possible, but expensive, solution to the latter problem.

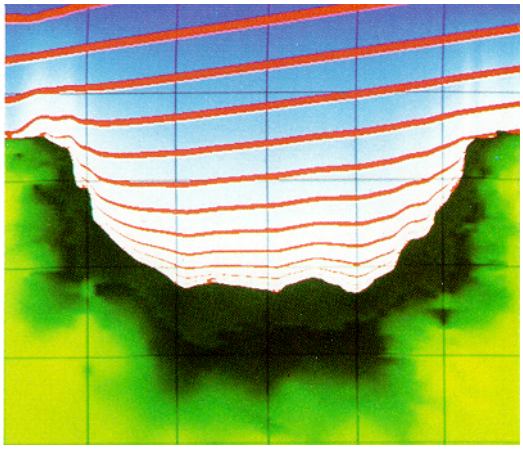

Figure 4: Topographic map of Babbage Beach

Figure 5: Beach terrain without water

4.2 Scanline Rendering of Waveforms

Because ray tracing is expensive and inconvenient for rendering complex ocean phenomena, we would like to develop a hidden surface technique more suited to the task. The Z-buffer hidden surface technique [3] seems to offer the greatest modeling freedom, since any element of a model may be rendered at any convenient time. Unfortunately, the Z-buffer technique requires a great deal of memory and is very prone to aliasing.

The A-buffer technique has been proposed [2] as an anti-aliased alternative to the Z-buffer. Unfortunately, the A-buffer does not really offer the same degree of rendering freedom as the Z-buffer. A pixel of the A-buffer is equivalent to a pixel of the Z-buffer when the pixel is entirely covered by an opaque surface. However, when a pixel is only partially covered or the covering surface is not opaque, the A-buffer stores the sub-pixel information as an arbitrarily long linked list of "fragment" structures, each of which is 28 bytes long in the original implementation. If the model being rendered produces a large number of such complex pixels, the memory consumption of the A-buffer may become prohibitive, and the paging of virtual memory may severely impact performance. To alleviate these problems, the A-buffer system pages the pixel array in software and renders surfaces in "approximately scanline order".

Since it is necessary to render in approximately scanline order to make practical use of the A-buffer, we decided to build an A-buffer-like system, Pixie, in which rendering takes place in *strictly* scanline order. Although the basic A-buffer techniques are still used at each pixel, our approach has a number of advantages. Instead of a buffer array with one entry per image pixel, we need only a single row buffer with one entry per image column. The vastly reduced storage requirements eliminate any need to page the A-buffer in software, and free us to store more information per pixel. In particular, there is no need to use the smaller 8-byte per pixel Z-buffer-like data structure for completely covered pixels. Even pixels which are covered by a single opaque surface are represented as a fragment list. This simplifies the algorithms and allows us to maintain 12 bits of color resolution and both minimum and maximum Z values for *all* pixels. An 8×8 bit pixel mask is used instead of the 4×8 mask originally proposed. The buffer itself is simply an array of fragment list pointers, with a NULL pointer indicating an empty pixel. When all rendering for the current scanline has been completed, an output routine is called to pack the fragment list (if any) at each pixel of the scanline, write the final color and coverage information out to the image files, and reclaim the fragment storage, setting the buffer fragment list pointers back to NULL.

Given the design of the Pixie rendering system, it was necessary to develop a scanline display algorithm for rendering wave surfaces. The algorithm is an adaptation of the general Lane/Carpenter parametric surface subdivision algorithm [8]. Our wave model may be viewed (in the appropriate coordinate system) as a function $y = f(x, z)$ at a particular time t. The form of this function is described in section 3. A suitable parametric representation of the surface in terms of parameters (u, v) is:

$$x(u, v) = u$$

$$y(u, v) = f(u, v)$$

$$z(u, v) = v$$

The algorithm maintains a set of polygons to be rendered and a set of wave surface patches to be rendered. Each set consists of a linked list for each scanline. The algorithm begins with the x and z coordinates of the initial wave surface in the modeling coordinate system. This initial wave patch is inserted in the linked list for the scanline on which the patch is first visible. As each scanline is reached, the algorithm checks each patch in the list for the scanline to see whether it can be accurately represented by a single polygon. If so, the patch is discarded and the appropriate polygon is inserted in the polygon display list to be rendered by an anti-aliased polygon scan conversion algorithm. If the patch is still too large to render as a polygon, the patch is subdivided into four subpatches, and each of these patches is inserted in the linked list for the scanline on which it first appears. Subdivision of the patch is done according to the perspective projection of the patch into screen space, in such a way that each of the four subpatches covers an approximately equal area on the screen. A patch is rendered as a polygon only when the patch covers approximately one pixel or less.

To implement the Lane/Carpenter scanline display algorithm requires that we are able to determine a fairly tight lower bound on the screen space Y coordinate of a given surface patch (assuming the Y coordinates increase going down the screen beginning with scanline 0 at the top). For any wave surface patch, a lower bound on the screen Y coordinate (scanline number) of the visible part of the patch may be found by transforming the modeling space coordinates (x_i, A, z_i) into screen space for each patch corner (x_i, z_i). This is true because the amplitude A gives an upper bound on the height of the wave function. Assuming the camera (eyepoint) is located above the plane $y = A$ in modeling space, the screen Y coordinate of the point $(x_i, f(x_i, z_i), z_i)$ cannot be less than the screen Y coordinate of (x_i, A, z_i).

Aliasing of distant waves is prevented by a form of "clamping" [12] which reduces the amplitude of waves which are very short relative to the pixel diameter. Waves shorter than two pixel diameters are completely ignored (their amplitude is zero).

It is interesting to compare this wave rendering algorithm to the "height field" algorithm used by Max [10] and originally proposed by Fishman and Schachter [4]. The height field algorithm renders a single-valued function of two variables by scanning columns of the image from bottom to top. Thus the algorithm would not be directly applicable to a scanline-oriented rendering system such as Pixie. A more serious problem is that the permissible viewing geometry is quite restricted with the height field algorithm. The camera must be above the top of the waves (maximum height field value), the viewing direction must lie in a horizontal ($y = k$) plane (the camera cannot be pointed somewhat upward or downward), and the camera must be upright (cannot be tilted from side to side). Our algorithm, as an adaptation of the general Lane/Carpenter technique, has no such viewing restrictions. However, a slightly more complex scanline bounding test must be used when the camera lies below the $y = A$ plane or is tilted very sharply to one side.

4.3 Rendering Spray

Our model of spray consists of the particle systems described in section 3.4. It is very difficult to directly render particle systems in scanline order. Instead, the particle systems are simulated by a separate program which contains the same wave model as the main Pixie rendering program. This program outputs a file of pixel information in scanline order which is used by Pixie to combine particles with the other elements of the scene and determine which surfaces are visible in each pixel.

The particle program advances through time in steps of ½-frame (1/48th second). At each step, new particles are generated according to the rules for wave breaking and obstacle impact. A data structure is allocated for each new particle to store its current position, its previous position, and its velocity. Each old particle is moved according to its average velocity during the step, and the velocity is updated so that the particle accelerates downward with the correct acceleration of gravity. Particles are deleted if they drop below the lower bound of the wave surface. No attempt is made to determine the actual wave level in the vicinity of the particle, since hidden surface removal will later be done by Pixie.

The particle population usually stabilizes after about ½ second, so that the number of new particles being generated is roughly equal to the number of particles being deleted. In order to ensure that all particles that could appear in the image have been generated, it is necessary to start the model far enough back in time that a particle generated in the initial step with the maximum upward velocity will have fallen below the lower bound of the wave surface by the time of the image.

When the particle population at the time of the desired frame has been generated, the particles are clipped to the viewing volume and transformed into screen space using exactly the same perspective projection used in Pixie to render the other elements of the scene. Each particle is drawn as an anti-aliased line segment joining its positions at the beginning and end of the ½-frame interval. This effectively provides temporal anti-aliasing (motion blur) for the particles, which are the fastest moving objects in each image. (The

waves move and change relatively slowly and smoothly so temporal aliasing is not a serious problem.)

The particles are drawn into a "sparse Z-buffer" in the particles program. The sparse Z-buffer consists of a linked list of pixel structures for each scanline of the image. When some particle is drawn in a given pixel, the appropriate scanline list is searched for the pixel, and a new pixel is added to the list if no other particle has previously been drawn into that pixel. In practice, the particles cover a relatively small portion of the entire image, so only about 10% of the total image pixels appear in the sparse Z-buffer. Because space consumption is quite small, it is possible to keep the color, coverage, and maximum and minimum screen Z value (perspective depth) for each pixel. Pixel data which overlap in Z value are stored in the same pixel structure; data with disjoint depths are stored in separate structures even if they appear in the same scanline and column of the image. Coverages of particles are determined from the distance to the particle and its velocity. The coverages of all particles in a pixel are added together subject to a maximum value of 1. The colors of particles are combined using a sum weighted by their coverages. Finally, the maximum and minimum Z values are updated according to each new particle in the pixel.

When all particles have been drawn into the sparse Z-buffer, the contents of the buffer are written into a temporary file. This file is read by Pixie during the rendering of the other elements of the scene. Since the particle file is in scanline order, Pixie can easily read the information (if any) which is relevant to the scanline currently being rendered. Each pixel in the particle file is passed to the A-buffer hidden surface routines, using the color, coverage (opacity), and maximum and minimum Z values which were determined by the particles program. Using this information, Pixie can determine whether the particles obscure or are obscured by other surfaces in the scene, such as the wave surface or the surface of an obstacle.

In the original work on particle systems [17], the particles were rendered by a separate program which produced color and coverage information for each image pixel in the form of an RGBα image [16]. This image was combined with other image elements using a digital compositing scheme. For images of waves and surf, it would be very difficult to determine clear depth relationships between particles and the surfaces in the scene, so that compositing of images could be used. The combination of particle information with other surface information in the A-buffer at the time of rendering solves this problem by automatically determining where surfaces or particles are visible.

Our current illumination model for spray particles is quite primitive. Particles are treated as small white spheres. Since it is prohibitively expensive to determine which particles are in the shadow of the waves or other particles, all of the particles are shaded the same color. To achieve a more realistic appearance, a random component is used to vary the shade a little. A slight darkening of the particles based on the magnitude of their downward velocity is used to simulate the shadowing of the particles by other particles and objects. This trick has worked fairly well, but clearly it would be easy to construct counter-examples where the effect would be quite unrealistic.

5. Examples

Figures 6 through 10 were produced using Pixie and the wave and surf model described in section 3. Figure 6 shows a group of moderately high waves approaching the beach. Figure 7 shows a similar group of waves with a three times larger amplitude. The increased amplitude makes these waves steep enough to break and

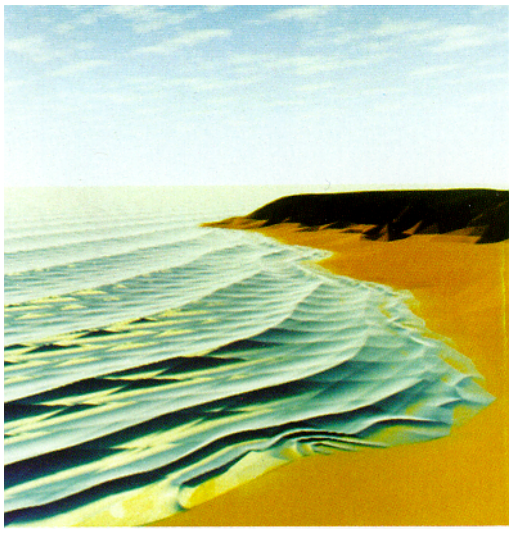

Figure 6: Waves on Babbage Beach

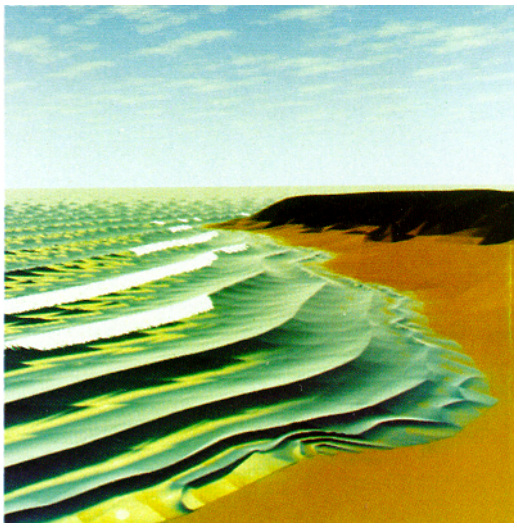

Figure 7: Breaking waves

generate spray when the wavelength is reduced in shallow water. Figure 8 is the same as Figure 7, with the addition of an pyramid-shaped rock as an obstacle. A small plume of spray is produced from the impact of a wave with the obstacle. Figures 9 and 10 show a sunward view of the beach in the late afternoon and at sunset.

These examples were produced using only three wave components, the main waves with a period of 6 seconds and two secondary components with periods of 2 and 2.5 seconds. Realism would be enhanced by the addition of a time-dependent bump-mapped texture pattern of small ripples.

Each of the examples was generated in approximately one hour of CPU time on a Pyramid 90x computer (comparable to a VAX-11/785 FPA). The implementation has not yet been carefully tuned or optimized. The simulation and rendering of the particle systems took approximately 10 CPU minutes with 2.5 megabytes of virtual

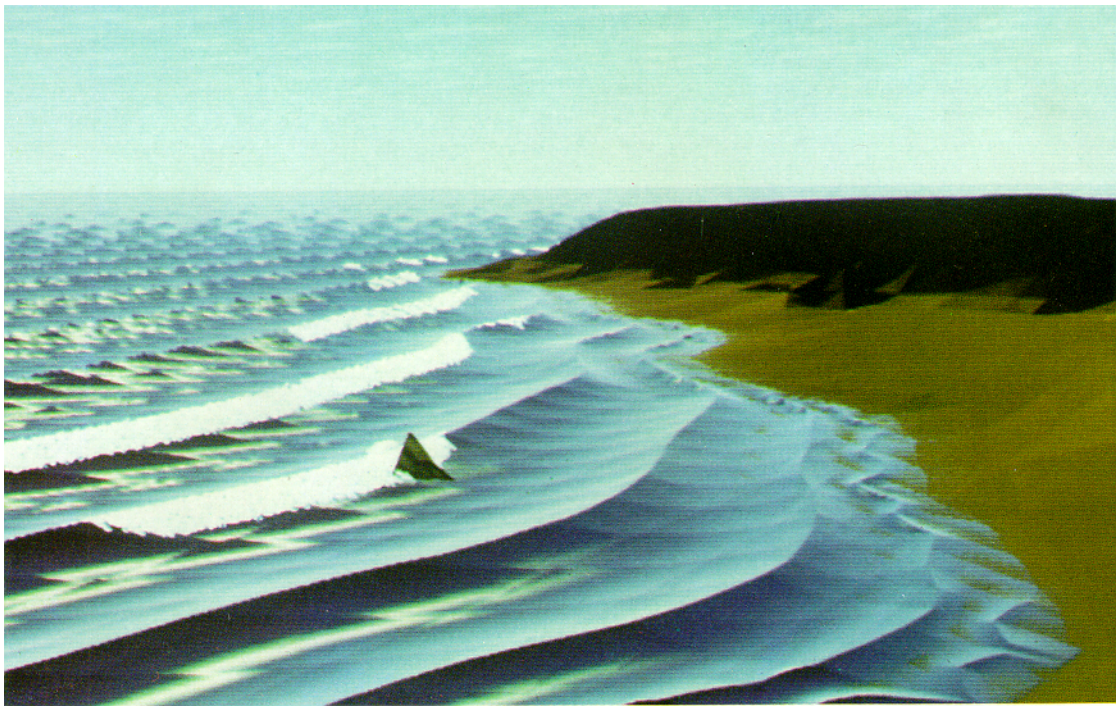

Figure 8: Breaking waves with obstacle

memory to simulate an active population of 56000 particles from a total of 95000 particles that were generated. Pixie took about 40 CPU minutes and 3 megabytes of virtual memory to render the wave surface and terrain, and to determine the visible surfaces at each pixel, including the particle coverage and depth information supplied by the particle program. The generation of the numerically integrated phase function table for each wave component consumed from 2 to 5 CPU minutes.

6. Conclusions and Future Research Directions

We have presented a graphical model of waves and surf which is capable of rendering and animating realistic images of these ocean phenomena. This is one of the first attempts to model breaking waves and surf in the field of image synthesis. In our model the shapes of waves change as they approach the shore, with the fronts of the waves steepening markedly as the depth of the water decreases. When the waves break, spray is simulated by a particle system in order to give the appearance of "whitecaps". Particle system spray is also used to model the impact of waves on obstacles. Wave refraction effects and the change of speed and wavelength in shallow water are simulated using a numerically integrated phase function which need only be computed once for a given wave component, even if many frames of animation are generated.

The rendering of the wave model is done by an adaptation of the Lane/Carpenter subdivision algorithm for parametric surfaces to the particular case of single-valued functions of two variables. This is a scanline algorithm, and does not suffer from the restrictions on viewing geometry inherent in Max's height field algorithm. The wave rendering algorithm is incorporated in a novel implementation of the A-buffer, with performance and simplicity advantages over the original A-buffer scheme. Particle systems are rendered by a separate program which produces a scanline sorted file of pixel information suitable for the A-buffer. This allows us to integrate particle systems with other surfaces at low cost and with greater flexibility than can be achieved using compositing techniques.

This work has concentrated on modeling the geometry and motion of waves, surf, and spray. Further work on illumination models is needed, especially to approximate shadowing of spray particles, refraction of light from spray particles, and transmission of light through steep wave crests. Texturing of wave surfaces might be used to simulate foam, but care must be taken that such texture maps behave sensibly from frame to frame of animation. The effectiveness of particle systems in modeling spray suggests that they might be used to model fountains, waterfalls, and perhaps even rapids.

In our model each wave component is long-crested and has a fixed amplitude. To more realistically model the ocean surface, it would be desirable to add the concept of "wave groups" to the model. A wave group would consist of a phase offset and an amplitude function which would produce a moving "bump" of short-crested waves which could be out of phase with other wave groups of the same wave component. The wave group would move at one-half the wave speed as predicted by theory, in the same general direction as the waves themselves. Several wave groups could share a single phase function.

Acknowledgements

This research was supported by the Natural Sciences and Engineering Research Council of Canada through infra-structure grant no. A2527. The work could not have been done without the support and facilities of the Department of Computational Science and the University of Saskatchewan. The encouragement and assistance of my wife, Judy, was vital to the completion of this paper.

Figure 9: Sunset

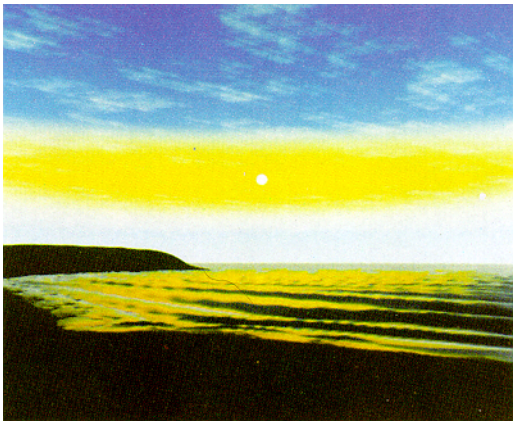

Figure 10: Late afternoon

References

[1] Bloomenthal, J. Modeling the mighty maple, *Computer Graphics 19*, 3 (July 1985), 305-311.

[2] Carpenter, L. The A-buffer, an antialiased hidden surface method, *Computer Graphics 18*, 3 (July 1984), 103-108.

[3] Catmull, E. *A Subdivision Algorithm for Computer Display of Curved Surfaces*, University of Utah, December 1974.

[4] Fishman, B. and Schachter, B. Computer display of height fields, *Computers and Graphics 5* (1980), 53-60.

[5] Fournier, A., Fussell, D., and Carpenter, L. Computer rendering of stochastic models, *Commun. ACM 25*, 6 (June 1982), 371-384.

[6] Fournier, A. and Reeves, W. A simple model of ocean waves, *Computer Graphics 20*, 3 (August 1986).

[7] Gardner, G. Visual simulation of clouds, *Computer Graphics 19*, 3 (July 1985), 297-303.

[8] Lane, J. and Carpenter, L. A generalized scan line algorithm for the computer display of parametrically defined surfaces, *Computer Graphics and Image Processing 11* (1979), 290-297.

[9] Kinsman, B. *Wind Waves: their generation and propagation on the ocean surface*, Prentice-Hall, Englewood Cliffs, N.J., 1965.

[10] Max, N. Vectorized procedural models for natural terrain: waves and islands in the sunset, *Computer Graphics 15*, 3 (August 1981), 317-324.

[11] Milne-Thomson, L. *Theoretical Hydrodynamics, 5th edn.*, Macmillan & Co., London, 1968.

[12] Norton, A., Rockwood, A., and Skolmoski, P. Clamping: a method of antialiasing textured surfaces by bandwidth limiting in object space, *Computer Graphics 16*, 3 (July 1982), 1-8.

[13] Peachey, D. PORTRAY–an image synthesis system, *Proc. Graphics Interface '86*, Vancouver, May 1986.

[14] Perlin, K. An image synthesizer, *Computer Graphics 19*, 3 (July 1985), 287-296.

[15] Pethick, J. *An Introduction to Coastal Geomorphology*, Edward Arnold Ltd, London, 1984.

[16] Porter, T. and Duff, T. Compositing digital images, *Computer Graphics 18*, 3 (July 1984), 253-259.

[17] Reeves, W. Particle systems – a technique for modelling a class of fuzzy objects, *Computer Graphics 17*, 3 (July 1983), 359-376.

[18] Reeves, W. and Blau, R. Approximate and probabilistic algorithms for shading and rendering structured particle systems, *Computer Graphics 19*, 3 (July 1985), 313-322.

[19] Smith, A. Plants, fractals, and formal languages, *Computer Graphics 18*, 3 (July 1984), 1-10.

[20] Stoker, J. *Water Waves: The Mathematical Theory with Applications*, Interscience Publishers, New York, 1957.

[21] Whitted, T. An improved illumination model for shaded display, *Commun. ACM 23*, 6 (June 1980), 343-349.

[22] Whitted, T. The hacker's guide to making pretty pictures, SIGGRAPH '85 Course Notes: Image Rendering Tricks, July 1985.