Modeling and Rendering Waves: Wave-Tracing Using Beta-Splines and Reflective and Refractive Texture Mapping

PAULINE Y. TS'O and BRIAN A. BARSKY
University of California, Berkeley

The graphical simulation of a certain subset of hydrodynamics phenomena is examined. New algorithms for both *modeling* and *rendering* these complex phenomena are presented.

The modeling algorithms deal with wave refraction in an ocean. Waves refract in much the same way as light. In both cases, the equation that controls the change in direction is Snell's law. Ocean waves are continuous but can be discretely decomposed into wave rays or *wave orthogonals*. These wave orthogonals are *wave-traced* in a manner similar to the rendering algorithm of ray-tracing. The refracted wave orthogonals are later traversed and their height contributions to the final surface are calculated using a sinusoidal shape approximation and the principle of wave superposition. The surface is then represented by *Beta-splines*, using the tension (or $\beta 2$) shape parameter to easily add more complexity to the surface.

The rendering algorithms are based on the use of *texture maps* and Fresnel's law of reflection. In each algorithm, two texture maps are used to simulate *reflection* and *refraction*. Based on surface normal orientation and Fresnel's law, a weighting is calculated that determines what fractions of reflected color and refracted color are assigned to a point. These algorithms are more efficient, though less accurate, alternatives to standard ray-tracing techniques.

Categories and Subject Descriptors: I.3.0 [**Computer Graphics**]: General; I.3.3 [**Computer Graphics**]: Picture/Image Generation—*display algorithms*; *viewing algorithms*; I.3.4 [**Computer Graphics**]: Graphics Utilities—*software support*; I.3.5 [**Computer Graphics**]: Computational Geometry and Object Modeling—*curve, surface, solid, and object representations*; *geometric algorithms, languages, and systems*; *modeling packages*; I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism—*animation*; *color, shading, shadowing, and texture*

General Terms: Algorithms

Additional Key Words and Phrases: Beta-splines, Fresnel, hydrodynamics, texture mapping, wave refraction, waves, wave-tracing

1. INTRODUCTION

Recently, the modeling of natural phenomena has captured the attention of an increasing number of computer graphics researchers. Because natural objects are often quite asymmetric and nonrigid, they serve as excellent standards for modeling complexity. Much of the natural phenomena being examined in

This work was supported in part by the Defense Advanced Research Projects Agency under contracts N00039-82-C-0235 and N00039-84-C-0089, the National Science Foundation under grants ECS-8204381 and DCI-845199, and the State of California under a Microelectronics Innovation and Computer Research Opportunities grant at the University of California, Berkeley.
Authors' address: Berkeley Computer Graphics Laboratory, Computer Science Division, University of California, Berkeley, CA 94720

computer graphics have already been studied in fluid mechanics. The term *fluids* pertains not only to liquids, but also to gases. Fluid mechanics examines not only the flow of liquids, but also the flow of of air or wind, and so is relevant to the forms of clouds and sand storms, the movement of plants in the breeze, and the shaping of the land and seas.

Unfortunately, the theory of fluid mechanics is incomplete, primarily because the equations that describe fluid systems are nonlinear and are thus often difficult to solve. Nature and time can also complicate matters by hiding many of the initial conditions of a given situation. For example, the behavior of turbulent water seems random, but it must obey a set of equations known as the Navier–Stokes equations [23]. We are prevented from even trying to apply these nonlinear equations because all the initial conditions of the system are unknown. Often we must resort to less precise statistical methods or other approximations to predict properties of a turbulent system.

Under certain assumptions, however, fluid mechanics provides a set of simple equations that gives a first approximation to the behavior of water waves. These equations are the result of what is called *linear small-amplitude wave theory* [11, 23, 32], a theory that can be used effectively to construct a model of an ocean surface.

The rendering of natural phenomena usually comes hand-in-hand with the modeling; that is, once a model has been created, it is natural to want to visualize it also. Many types of rendering algorithms are currently being used in the computer graphics field. By taking advantage of special properties of the models, some of these algorithms can be used in new and more satisfactory ways.

A synopsis of previous work is presented in Section 2 of this paper, an algorithm for modeling ocean waves is discussed in Section 3, and Section 4 suggests several algorithms for rendering the resulting model.

2. PREVIOUS WORK

Over the past decade researchers have made various attempts to model ocean waves. Several researchers have used a normal perturbation approach similar to that described in [6], [7], and [18]. Perlin, for example, bases his perturbation on a scalar-valued function similar to a cycloid [29]. Another approach, which generates long-crested wave models with seven parameters to control the properties of textures, is described in [30]. Clamped analytical functions are used to antialias textured surfaces in [27]. The general approach works well when the viewer is far enough away from the water so that the ocean surface appears flat. When the viewer is near the surface, however, the three-dimensional nature of the surface becomes very important.

Others have taken this three-dimensionality into account. In 1981, Nelson Max presented a paper that used simple wave theory to generate height fields [25]. In work that has occurred since this present paper was submitted for publication, by using a combination of particle systems and simple hydrodynamics, both Peachey [28] and Fournier and Reeves [15] have constructed models that address waves that refract and break. In an even more recent article [24a], Masten et al. use a well-known power spectrum of waves under steady wind and infinite fetch to Fourier synthesize the surface. The result has a good appearance although it is valid only for deep water waves and difficult to animate correctly.

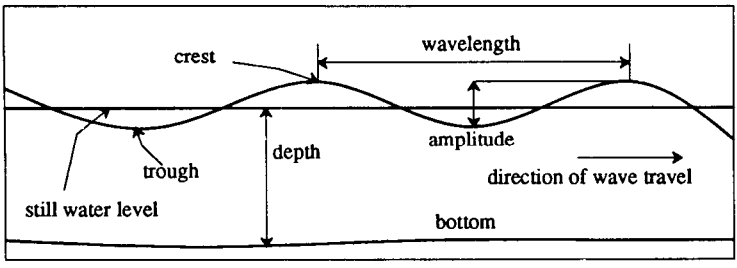

Fig. 1. Basic characteristics of a wave train.

3. MODELING OCEAN SURFACE WAVES

3.1 Background

3.1.1 *Basic Terminology.* Although waves disturb the entire volume of water through which they pass, we shall only examine the perturbation they cause at the surface of the water. Many of these surface waves are quite regular in form. A *solitary wave* is a wave that can be created by a single pulse in shallow water, whereas a *wave train* is a series of waves traveling in the same direction created by multiple pulses. The features of a wave-train profile are illustrated in Figure 1.

A locally maximum high point of a wave is called a *crest* and a locally minimum point a *trough*. The *wavelength* is the distance between two crests. Two properties of a wave propagating through a medium are *speed* and *direction*. The speed or *celerity* of the wave can be measured by the movement of its crests. The *depth* of water is usually measured by the distance between the water's bed and the average surface height of the water or by the height of the water when at rest. The *height* of a wave is the vertical distance between a trough and an adjacent crest. The *amplitude*, a, is half the height. The *period*, T, of a wave is the time interval necessary for two successive crests to pass through a fixed point in space. The *frequency* of a wave is the number of unit cycles per unit time and is related to the period by $m = 2\pi/T$. A *wave front* is the continuous planar curve created by the crest of a three-dimensional wave. *Wave orthogonals* are the normals to the wave front. Table I lists the symbols and the associated terms as they are used in this paper.

The qualitative depth of water is usually measured by the ratio of depth, d, to wavelength, λ. Water is considered shallow if $0 < d/\lambda < \frac{1}{20}$. It is of intermediate depth if $\frac{1}{20} < d/\lambda < \frac{1}{2}$. Water is deep if $d/\lambda > \frac{1}{2}$ [11]. These ratios are only rough guidelines and may differ slightly from reference to reference.

A flow is often considered to be composed of *fluid elements*—discrete, infinitesimally small volumes of fluid particles. Intuitively, a flow is *irrotational* if each fluid element does not spin about its center of gravity. Each element may follow a circular path, however, like a rock swinging in a slingshot, provided that it does not rotate around its local center of gravity. Mathematically, the condition of irrotational fluid flow is denoted by

$$\text{curl } V = \frac{\partial u}{\partial x} - \frac{\partial v}{\partial y} = 0.$$

Table I

Symbol	Definition
λ	Wavelength
η	Surface function; x, y, t, variables
ω	Angular frequency
ρ	Fluid density
α	Amplitude of a wave
C	Wave speed or celerity
d	Depth of the water
g	Acceleration due to gravity
k	Wave number
m	Wave frequency
p, q	Wave direction variables
t	Time
T	Wave period
u, v, w	Velocity components of V
V	Velocity vector
x, y, z	Spatial coordinates

A flow is considered *uniform* in this paper if the horizontal velocity of an ocean wave does not change with vertical depth; that is, a fluid element on the water's surface has the same velocity as any fluid element directly below it.

3.1.2 *Linear Small-Amplitude Wave Theory.* Nelson Max used linear small-amplitude wave theory to model an ocean surface [25]. The theory's basic assumptions are that fluid density is constant and fluid flow is uniform and irrotational. These assumptions, of course, restrict the valid domain of the theory. For example, in shallow water, nonlinear convective inertia terms are significant, meaning that flow is nonuniform. The limits of linear small-amplitude wave theory become apparent here since it is in shallow water where waves break and the theory fails to predict this phenomenon.

However, given the above assumptions, the surface of the water can then be approximated by the superposition of simple sinusoidal wave trains. A wave train passing through a one-dimensional surface can be expressed as

$$\eta(x, t) = a \sin(kx - \omega t).$$

The variable k is the wave number, that is, the number of cycles per unit distance. It is related to λ by $k = 2\pi/\lambda$. The variable ω is the angular frequency in radians per second and is defined as $\omega = kC$, where C is the wave celerity. The superposition of several wave trains is simply

$$\eta(x, t) = \sum_{i=1}^{i=n} a_i \sin(k_i x - \omega_i t),$$

where n is the number of wave trains.

The above formulation of a wave train can be generalized for a two-dimensional surface by specifying direction variables p and q, such that $p^2 + q^2 = k^2$, where p and q are real numbers. A single wave train can then be described by the equation

$$\eta(x, y, t) = a \sin(px + qy - \omega t).$$

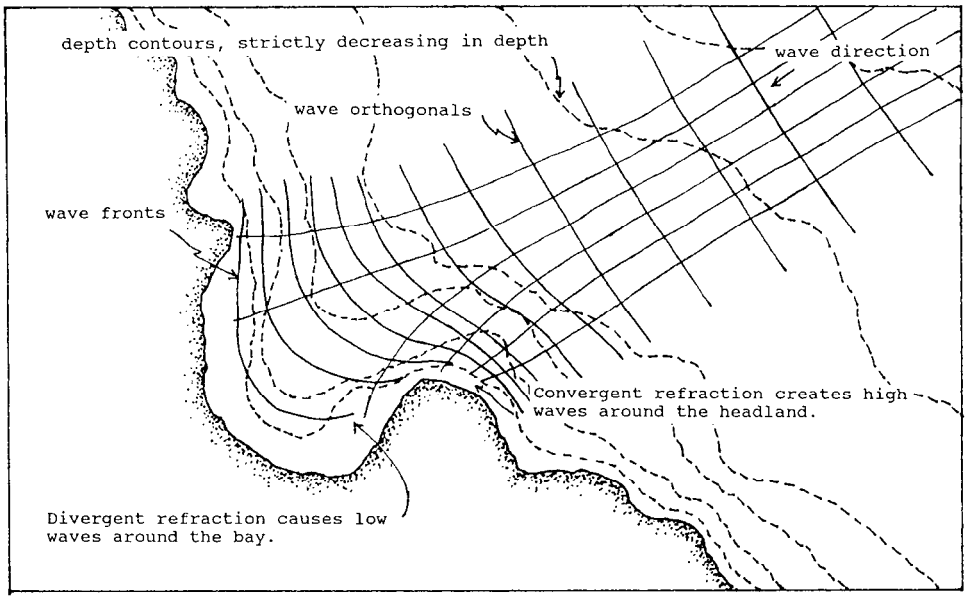

Fig. 2. Refraction diagram.

The surface produced by n wave trains can be described [25] by

$$\eta(x, y, t) = \sum_{i=1}^{i=n} a_i \sin(p_i x + q_i y - \omega_i t).$$

Linear small-amplitude wave theory is valid only when the aforementioned assumptions hold, but because the resulting equations are so simple, engineers have often used them to approximate behavior in other situations. In the following sections, linear small-amplitude wave theory is applied to shallow water phenomena.

3.1.3 *Wave Refraction.* Refraction is the deflection of a wave, such as a light or sound wave, as it propagates through different media. Water waves refract in much the same way as rays of light, and this deflection explains why incoming wave fronts are usually parallel or nearly so to the direction of the shoreline. Refraction of water waves occurs when a wave passes from one depth to another (Figure 2). When the wave orthogonal intersects obliquely with a contour line of the ocean floor, it bends toward or away from the normal of the contour line at the point of intersection according to Snell's law:

$$\frac{\sin i}{\sin r} = \frac{C_i}{C_r}.$$

Here, i and r are the incident and refractive angles, respectively, between the normal to the wave front and the normal of the contour line, that is, the angles before and after refraction (Figure 3). The variables, C_i and C_r, are likewise the celerities of the wave train before and after refraction.

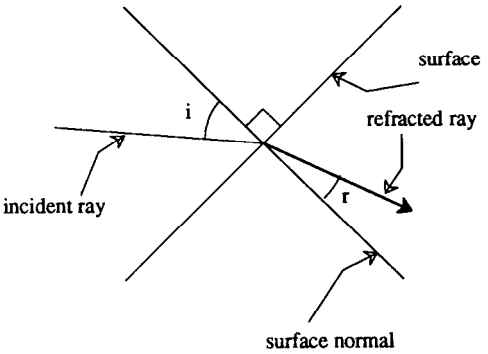

Fig. 3. Snell's law of refraction.

The celerity and wavelength of a linear small-amplitude wave each depend on both the depth and the period of the wave [11, 23]. In its general form, the wavelength is described by the following implicit equation [23]:

$$\lambda = \frac{gT^2}{2\pi} \tanh \frac{2\pi d}{\lambda}. \tag{3.1}$$

Finding the wavelength from the equation involves solving a transcendental function. The numerical method of Newton's iteration is suitable for this task [13]. Figure 4 illustrates the behavior of this function for a limited set of depths. Given the wavelength, the wave celerity is described by

$$C = \frac{\lambda}{T} = \frac{gT}{2\pi} \tanh \frac{2\pi d}{\lambda}. \tag{3.2}$$

When the wave propagates through shallow water, the function tanh can be approximated by a straight line of unit slope. Then,

$$\tanh \frac{2\pi d}{\lambda} \approx \frac{2\pi d}{\lambda}. \tag{3.3}$$

Substituting eq. (3.3) into eq. (3.1) yields the following for wavelength:

$$\lambda = \frac{gT^2}{2\pi} \cdot \frac{2\pi d}{\lambda},$$

$$\lambda^2 = gT^2 d,$$

$$\lambda = T\sqrt{gd}.$$

The celerity thus becomes

$$C = \frac{\lambda}{T} = \sqrt{gd}.$$

Thus, in shallow water, waves tend to slow down. In deep water,

$$\tanh \frac{2\pi d}{\lambda} \approx 1. \tag{3.4}$$

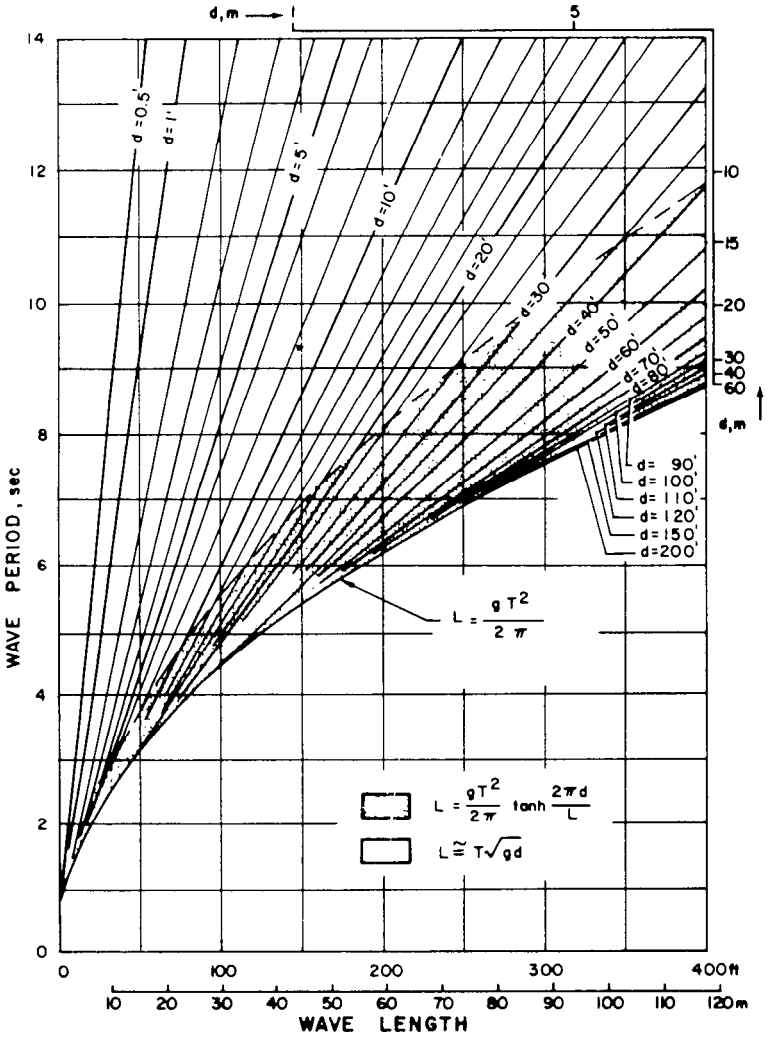

Fig. 4. Wavelength (here denoted as L) as a function of depth and period (from [23], © Springer-Verlag, 1976).

Substitution of eq. (3.4) into eq. (3.1) yields

$$\lambda = \frac{gT^2}{2\pi}$$

and into eq. (3.2) yields

$$C = \frac{gT}{2\pi}.$$

Wave refraction is mostly a shallow and intermediate depth phenomenon. When the water depth exceeds about $\lambda/3$, refraction effects are small and can be

ignored [23]. In some situations, refraction results are inaccurate. As the steepness of the floor slope increases, wave reflection effects also become evident, thus changing not only the amplitude of the wave as energy is lost, but also the shape of the entire surface as a new wave train is created. Although there are no hard boundaries, when the slope is greater than $\frac{1}{10}$, wave refraction theory should be considered incomplete. Also, if wave orthogonals cross one another owing to refraction, high-amplitude waves or *caustics* may be created according to the principle of superposition. With high-amplitude waves, nonlinear effects become significant. In other words, when wave orthogonals converge or diverge too drastically, complications arise. A measure of this convergence and divergence is the ratio of the distance between wave orthogonals in deep water (where refraction is negligible) to the distance between the wave orthogonals at the current point. This ratio is called the *refraction coefficient*. Le Méhauté provides the rule of thumb that when the refraction coefficient is much different from unity, say less than $\frac{1}{2}$ or greater than 2, then the physical model predicted by wave refraction alone should be doubted.

Although wave refraction is not a panacea for all problems of modeling the ocean surface, it can add a significant degree of realism to the current models being used. For example, in Max's short film, "Carla's Island," the ocean waves that move toward the two islands should wash up parallel to the islands' shores. It is obvious, though, that they do not; rather, they appear to cut right through the islands as if the islands were made of air.

3.2 Modeling Algorithm: Wave-Tracing

3.2.1 *Overview.* One solution to the calculation of wave refraction is very similar to the well-known graphics technique of *ray-tracing*. In basic ray-tracing algorithms, a view-ray is projected from the eyepoint through each pixel of the screen (typically, 512 × 512 pixels) [20, 31, 36]. Every ray is tested for intersections with objects in the scene. When a ray intersects an object, the ray may reflect and/or refract, possibly creating two rays where there was only one before. Each of these resulting rays is also recursively followed to test for intersections with objects. The effects of an initial ray are represented by a binary tree where each node represents a ray–object intersection and each branch represents a reflected or refracted ray. After the tree is created, lighting calculations can be performed by traversing the tree.

With *wave-tracing* the idea is the same. A ray representing a wave front's direction of propagation is followed and tested for the nearest intersection with the contour lines of the ocean floor. The angle of refraction is calculated by means of Snell's law, and the resulting ray is tested for intersection, and so on. Instead of a binary tree, a simple list suffices to keep track of the newly formed rays because reflections are ignored and thus only one ray is propagated at each intersection.

Once the wave lists have been created, they must be traversed for every time t. To store the final results, an $m \times n$ array or grid is needed. This grid quantizes the ocean surface into discrete x, y points. Grid size depends only on the resolution or amount of sampling that the user desires. As a ray in a list is followed, if it "comes close" to one of these discrete points, then its contribution

to the ocean's surface is calculated by

$$\eta_i(x, y, t) = a_i \sin(p_i x + q_i y - \omega_i t). \tag{3.5}$$

The function η_i is the contribution of only one wave front to the final ocean surface. Because of the principle of superposition, we can save η_i in the array and simply add later contributions to the running sum [33].

In addition to creating lists instead of binary trees, wave-tracing differs from ray-tracing in other ways. Because wave refraction is independent of the viewer's eyepoint, these wave lists need to be calculated only once, as long as the topography of the land surface and the direction of the incoming waves remain constant. For most applications, these assumptions are reasonable. Another simplifying difference is that wave refraction is essentially two dimensional. The intersection testing deals only with vectors lying on the $z = 0$ plane and thus is computationally less expensive than conventional ray-tracing.

3.2.2 *Constructing the Contour Lines.* There are two approaches to the construction of the shore information necessary for refraction and rendering. The first is to derive contour lines from an existing three-dimensional model of the land. The second is the converse: to derive the three-dimensional model from contour information.

Deriving the contour lines from three-dimensional polygonal data may be computationally expensive, but certainly not impossible. One could simply test each polygon for intersection with a set of planes defining each contour depth. Fortunately, again, this would only have to be done once for any given animation. If the existing model is represented by a mathematical surface, such as a spline, then constructing the contour lines may be even simpler. To give one example, this approach is desirable if the stochastic procedural model proposed by Fournier et al. in [14] is used to model the land. With their algorithm, the user has only rough control over the shape of things to come. In cases like this, contour lines would have to be derived after the fact.

The second approach has been implemented by us. The user inputs a straight-line approximation of the contour lines using a digitizing tablet. The user must also specify the depth represented by each contour line. A Beta-spline [1–3] is used to smooth out the noise of the free-hand data. This smoothing is important because it is otherwise difficult to prevent minor perturbations in the contour lines from significantly affecting the refraction behavior of a wave orthogonal. Figure 5 illustrates a sample set of contour lines used for later figures.

The three-dimensional surface can easily be constructed by sampling each spline at fixed parametric intervals or using adaptive subdivision with some flatness criterion [4, 5, 21, 22]. Then a set of x, y coordinates can be calculated from the spline equations and the z coordinate is simply the depth represented by the contour line. From here, the x, y, z points can be considered as the defining points for a bivariate spline surface or the defining vertices of polygons.

3.2.3 *Constructing the Initial Wave Fronts.* To start the process of wave-tracing, the user must specify one direction vector for each initial wave train. The number of initial wave trains is typically two or three. In addition to these vectors, a number must be given indicating the sampling resolution of the wave

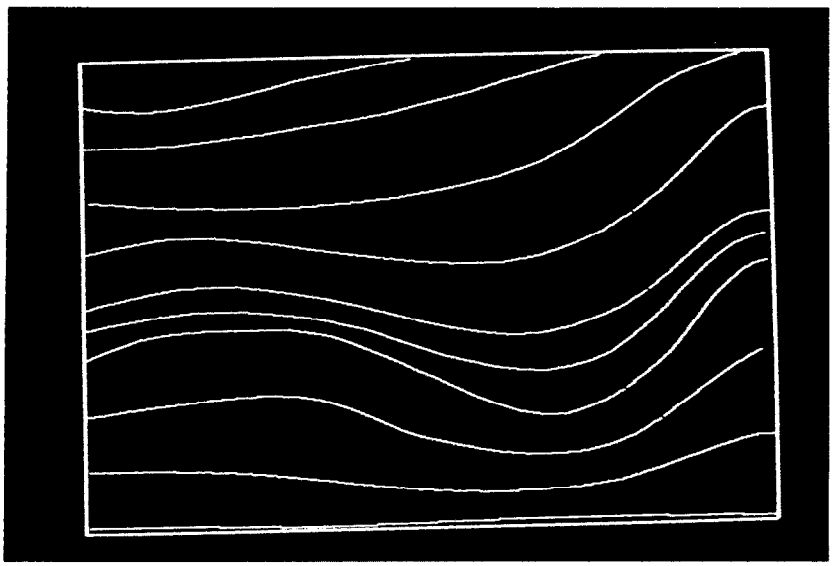

Fig. 5. The contour lines used as input for Figure 14. The depths in meters of each contour line from topmost to bottommost are 2.0, 1.25, 0.75, 0.5, 0.3, 0.2, 0.1, −0.1, −0.1, 0.0.

trains. That is, to perform wave refraction calculations, the continuous wave trains must be decomposed into a representation by discrete wave orthogonals. The resolution number determines how many wave orthogonals there will be for a given initial wave train.

It is not apparent how one should select such a resolution number. Currently, for each grid line, we generate at least one wave orthogonal per wave front. For example, if the grid is $m \times n$, then each wave front will be decomposed into $n + m$ wave orthogonals. Intuitively, the size of the grid should at least depend on the complexity of the contour lines and on the smallest wavelength of the waves. Questions of resolution and sampling arise again in the next section.

3.2.4 *Traversing the Wave Lists.* After the lists of wave orthogonal intersections have been created by wave-tracing the initial wave orthogonals, these lists must be traversed to find the orthogonals' contribution to the final surface height. Perhaps the simplest way to traverse a wave list is to treat the path between intersections of contour lines as straight lines. Next, using an algorithm identical to a simple line drawing algorithm (such as Bresenham's [8]), discover the nearest discrete points on the $m \times n$ quantizing grid. Then, for some units on either side of these discrete points, say in the x direction, use eq. (3.5) to obtain the contribution to $\eta(x, y, t)$. The number of units taken will depend on the resolution of the wave orthogonals (Figure 6). Currently, the number of units taken is zero. That is, only the discrete points found by the line drawing algorithm are affected by the traversal of a particular wave list.

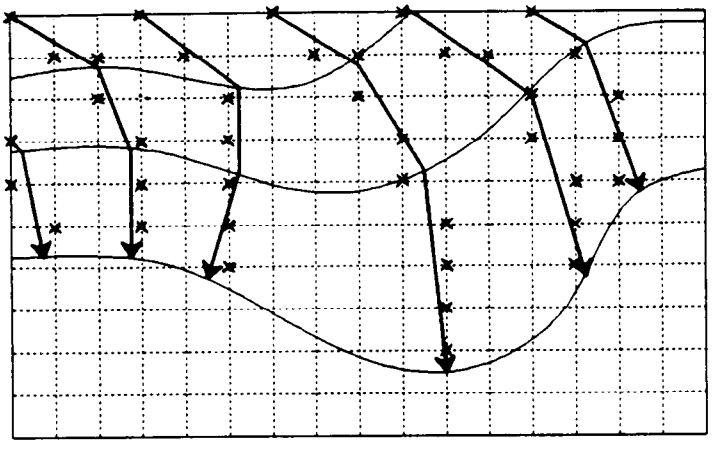

Fig. 6. The crosses represent a discretized wave orthogonal.

The variables p_i and q_i of eq. (3.5) must be reconstructed from the direction of the wave orthogonal. Recall from Section 3.1.2 that p_i and q_i are such that

$$p_i^2 + q_i^2 = k_i^2.$$

Then p_i and q_i can be found by

$$p_i = k_i \cos \theta_i, \qquad q_i = k_i \sin \theta_i,$$

where θ_i is the angle between the wave orthogonal and the x-axis.

Notice that eq. (3.5) depends on the angular frequency and amplitude and recall that these variables ultimately depend on the depth of the water and the period of the current wave train. Hence, as a wave orthogonal is traced over different contour lines, the frequency and amplitude must be recalculated. Thus the initial amplitude, the period, and the vector specifying the present wave direction are needed as input.

One could pass a spline through the contour line intersections, but it has yet to be determined whether this would make an appreciable difference. One could also smoothly weight the units on either side of the approximated wave orthogonal. What we have now is a step function that determines the influence of a wave orthogonal on nearby grid points. One could imagine using a Gaussian curve, a sine curve, etc. Again, it is not presently known how these different curves would affect the quality of the model.

As diverging wave orthogonals are traced, as in the upper second and third orthogonals in Figure 2, questions of resolution may arise again. At first it appears that there is an insufficient number of wave orthogonals to produce a realistic image; that is, a completely smooth patch on the ocean would appear. Indeed, such a smooth patch could easily occur if only one long-period wave is used to model the ocean. However, it is much more natural to model the ocean with more than one wave train. In addition to the long-period wave train, at least one wave train of much shorter period is also used to add small wind ripples and

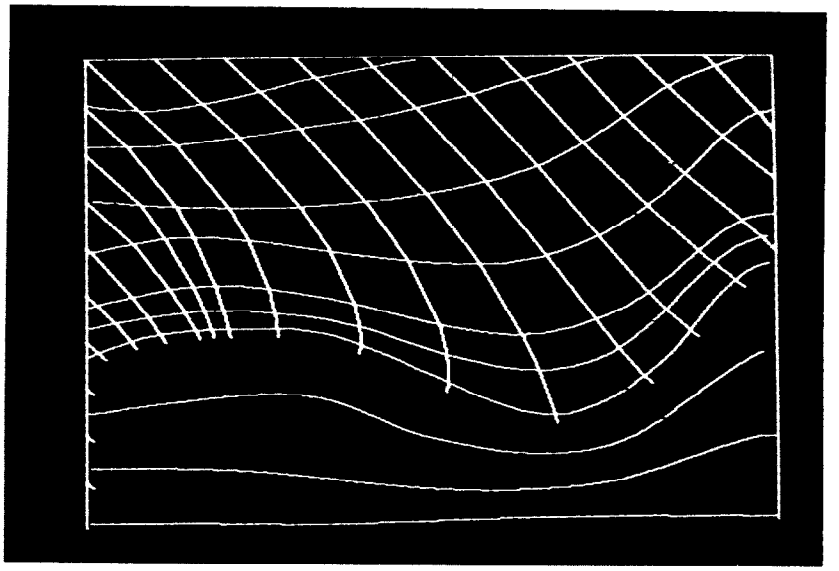

Fig. 7. The effects of refraction on the wave orthogonals of one wave train. The period of this wave is 6 seconds. Notice the divergence of the wave orthogonals in the center of the photograph. This divergence would appear as a perfectly flat spot on the ocean surface if there are not enough wave orthogonals or if the divergence is high enough. The initial direction vector is (0.5, −0.7).

textural complexity to the larger waves. Recall that refraction is noticeable only when the depth is approximately less than $\lambda/3$. If the period of these smaller waves is selected to be very small, then their corresponding wavelengths will cause refraction effects to be negligible everywhere, except right at the shoreline. This adds textural complexity to calm spots while retaining their basic property of only having low-amplitude waves. Figures 7 and 8 show the effects of refraction using the same contour line information but different periods and directions for the wave trains.

The above solution actually imitates nature fairly closely. The height of waves in bays is small precisely because of their refraction and that is what gives bays their protective quality. In addition, most of the finer details of an ocean surface are due to wind waves of very small amplitude, which, again, hardly refract [32].

3.3 Implementation Details

As previously mentioned, several researchers have used normal perturbation techniques to represent ocean waves. This approach is inadequate, in general, because properties such as shadowing and realistic silhouetting are lost. For this reason we have chosen to treat the ocean as a true three-dimensional surface. In our case we use three-dimensional Beta-splines to represent the water's surface [1–3]. The xy quantizing grid and $\eta(x, y, t)$ values are treated as the control vertices of a Beta-spline surface. This provides a compact representation of a complex surface, as well as smoothing the discontinuity of the rectangular step function discussed in the last section.

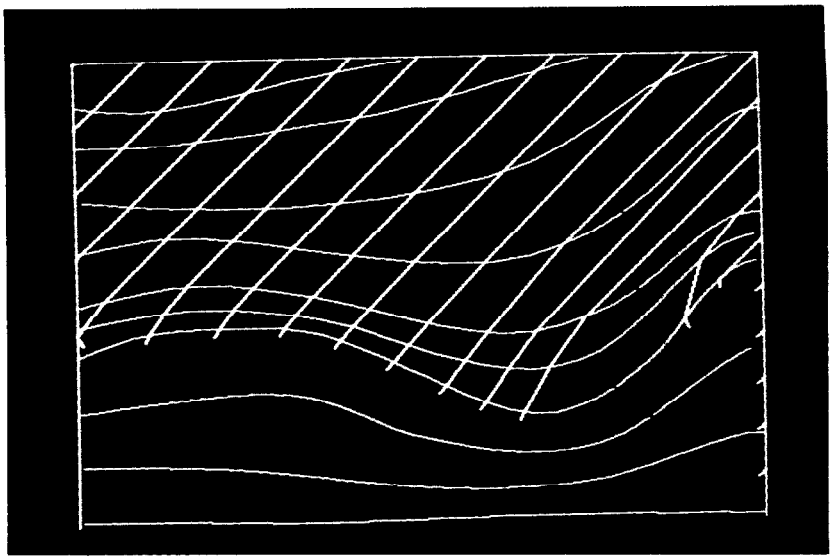

Fig. 8. The effects of refraction on another wave train. The period of this wave is 1 second. The initial direction vector is (−0.5, −0.7). Notice that refraction effects are not noticeable until the wave is very close to the shore. Notice also that in the area where the wave orthogonals diverged in Figure 7, the wave orthogonals of the smaller period wave do not appreciably diverge. So if both these wave fronts are used, then the dead calm spot implied by Figure 7 would not appear. The small ripples of this second wave front would add surface variation to that spot.

In addition, the Beta-spline has a useful characteristic shape parameter known as the *tension* (or $\beta 2$) shape parameter [3, 4]. Intuitively, higher values of the tension parameter correspond to sharper edges within the surface. This tension parameter can be varied locally; that is, it can take on different values at different control vertices. Here, the tension associated with a control vertex varies with the height of that control vertex. That is, the greater the z value, the greater the tension value. This relation creates smooth troughs and sharper crests, as illustrated in Figure 9. This shape parameter manipulation is a convenient, albeit ad hoc, method of approximating the true nonsinusoidal shape of most waves [23, 25].

Since the motion of the control vertices occurs in true three dimensions, a potential problem can occur at a wave's trough near the shore. At such a point the trough may drop so far below the still water level that the underlying shore becomes exposed before the expected shoreline. Theoretically, the problem occurs because, as mentioned earlier, shallow water behavior is actually dependent on nonlinear terms, which we are ignoring. If this occurrence is undesirable, it can be most easily handled by reassigning the contour line depths when rendering the shore. A less tractable approach would be to lower the amplitude of the waves.

Wave-tracing and refraction simplify two other implementation issues: *spatial periodicity* and *temporal periodicity*. The waves in Max's "Carla's Island" are very regular; the direction and shape of a wave train there remain constant over

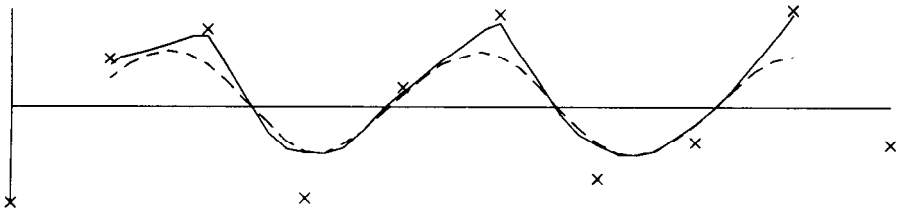

Fig. 9. The effects of the tension shape parameter on the shape of a wave profile. The control vertices of the wave profile drawn with a dashed line have tension values of zero. The control vertices of the profile drawn with a solid line have tension values that are a simple linear function of the positive height of the control vertex. When the height is less than zero, the tension is set to zero. The crosses mark the control vertices for the profiles. These control vertices are derived from regularly sampling a sinusoidal curve.

space. Refraction, by its very nature, disrupts such spatial regularity somewhat because the refracting group of wave orthogonals causes bends and changing amplitudes throughout the wave train.

Although spatial periodicity is not desirable, temporal periodicity is desirable because of the limited computer memory in which to store the animation frames and/or limited computer time in which to calculate the frames. Since the input to the wave-tracing algorithm consists of period, amplitude, and initial wave direction, specifying the temporal regularity of the animation is trivial. The animation will repeat itself every n seconds, where n is the smallest common multiple of all the wave periods.

Finally, although wave-tracing is considerably less expensive than ray-tracing, the running time is not negligible. For a sample short animation, 400 wave orthogonals were traced and traversed 42 times over the 341 line segments composing the contour lines. The total run time was 16 CPU minutes on a DEC VAX-11/750 with a floating-point accelerator running UNIX 4.3 BSD.[1] The medium resolution rendering time of just one of these frames, however, takes about ten times as long.

4. RENDERING SURFACE WAVES

4.1 Background

Once a water surface has been calculated, the next step is to visualize it. The spline representation can be subdivided into a collection of polygons in three dimensions. At this point, any of the standard transformations and rendering algorithms found in computer graphics [12] can be applied to these polygons.

Ray-tracing is a rendering algorithm which is elegant and flexible. The basics of ray-tracing were described at the beginning of Section 3.2 as an analogy to the modeling technique of wave-tracing. Although standard ray-tracing techniques would certainly produce satisfactory images, the amount of CPU time required to apply these techniques is usually prohibitive. Two texture mapping schemes without the use of ray-tracing are examined here as less costly rendering alternatives.

[1] VAX is a trademark of Digital Equipment Corporation. UNIX is a trademark of AT&T Bell Laboratories.

Texture mapping [6, 7, 9] is a method of adding color information to a surface. Some background information and references to the relevant literature are given in [18]. With this method, for example, a marble texture can be "wall-papered" onto a building composed of flat polygons. In its most common form, a texture mapping algorithm involves three elements: a modeled object that is to receive the texture; a *texture map*, that is, a rectangular array of color triples providing texture information; and a mapping from the object coordinates to the row/column indices of the texture map.

The mapping from object coordinates to texture map coordinates is fairly simple if the surface of the object is represented by a bivariate parametric spline. That is, the sampling of the spline surface is controlled by two parameters, say u and v. As the spline is sampled, if the current u and v values are scaled to fit within the range of the texture map indices, then a simple workable mapping has been created. This mapping is the one used in the implementation of the algorithms discussed below.

4.2 Rendering Algorithms

4.2.1 *Overview.* Each of the algorithms to be presented uses *two* texture maps to supply color information for a point on the water's surface. One texture map contains *reflected* color information, while the other contains *refracted* color information. The final color for a point is a linear combination of the two types of color information. The algorithms differ in the calculation of the weightings given to the two values. Each algorithm uses viewer orientation, surface normal orientation, and certain laws of optics in different ways to construct a weighting scheme.

4.2.2 *The Fresnel Texture Mapping Algorithm.* The optics principle of interest is Fresnel's law of reflection. Intuitively, it states that when a glancing ray of light strikes the surface of water at near 90 degrees to the surface normal, it is almost entirely reflected [17, 24]. At angles less than 90 degrees, the ray is also refracted. These two situations make simple texture mapping unsatisfactory, especially in an animation where surface normals change with every frame (see Figure 10). If, however, two texture maps are provided, one for reflective information and the other for refractive information, then a reasonable image may be created.

The percentage of light, I, reflected from the surface can be predicted from Fresnel's law for natural light as a function of the angle of incidence, i, and the angle of refraction, r, (see Figure 3) as follows [17]:

$$I = \frac{\sin^2(i - r)}{2 \sin^2(i + r)} + \frac{\tan^2(i - r)}{2 \tan^2(i + r)}.$$

In this algorithm a line is drawn between the eye and each vertex of each polygon to be rendered. The angles of incidence and refraction are calculated by Snell's law given this line, the surface normal, and a refractive index for water of 1.33 [33]. These angles are substituted into Fresnel's formula, which yields the percentage of light reflected from the surface. This percentage is used to weight the reflective texture map information, and its complement is used to weight the refractive texture map information. The final color information for

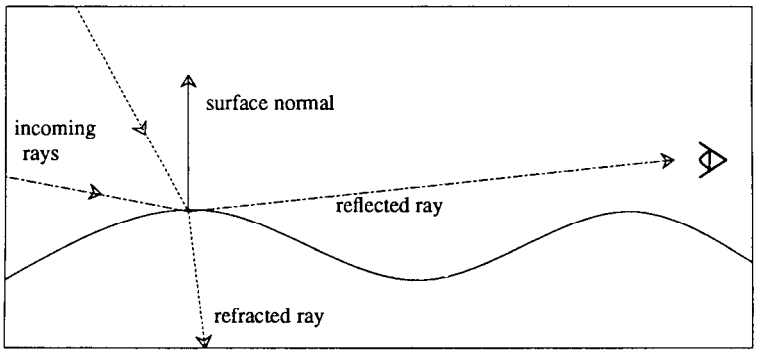

Fig. 10. Reflected and refracted rays.

the point can then be written as

$$(R_f, G_f, B_f) = I \cdot (R_{refl}, G_{refl}, B_{refl}) + (1 - I) \cdot (R_{refr}, G_{refr}, B_{refr}), \qquad (4.1)$$

where (R_f, G_f, B_f) is the final color triplet for the point, $(R_{refl}, G_{refl}, B_{refl})$ is the color triplet from the reflective texture map, and $(R_{refr}, G_{refr}, B_{refr})$ is the color triplet from the refractive texture map. As discussed in Section 4.1, note that the texture maps are indexed according to the current xy location on the surface.

The disadvantage of this algorithm is that it may be quite slow. In fact, it is similar to *ray-casting* in reverse.[2] That is, instead of rays being cast from the eye to the object, rays are cast from the object to the eye. Although no expensive intersection tests are required as in ray-casting, calculating the angles of incidence and refraction is still not cheap. Also, solving Fresnel's equation requires the costly evaluation of four trigonometric functions. If ray-casting is to be the rendering algorithm of choice, then using Fresnel's law precisely may add little extra overhead. Our environment, however, is based on a z-buffer rendering algorithm.

4.2.3 *The Approximate Fresnel Texture Mapping Algorithm.* In this algorithm, we approximate Fresnel's equation with a simple piecewise function that is dependent only on the angle of incidence. The same line as above, from the eyepoint to the surface point, is used to find the incident angle. The angle of refraction, however, is not explicitly calculated. Instead, given Snell's law and the index of refraction for water, the angle of refraction is written in terms of the incident angle. Fresnel's law is then evaluated for a small set of incident angles and an appropriate approximation is made of the resulting curve.

We approximate Fresnel's curve by a piecewise function consisting of a horizontal line and a parabola (see Figure 11). We choose a parabola because of its evaluation efficiency. It does not require the computation of square root,

[2] The ray-casting algorithm is a special case of ray-tracing that stops following a ray after its first intersection with an object. In other words, the ray-traversal tree is one level deep and no refraction effects are taken into account.

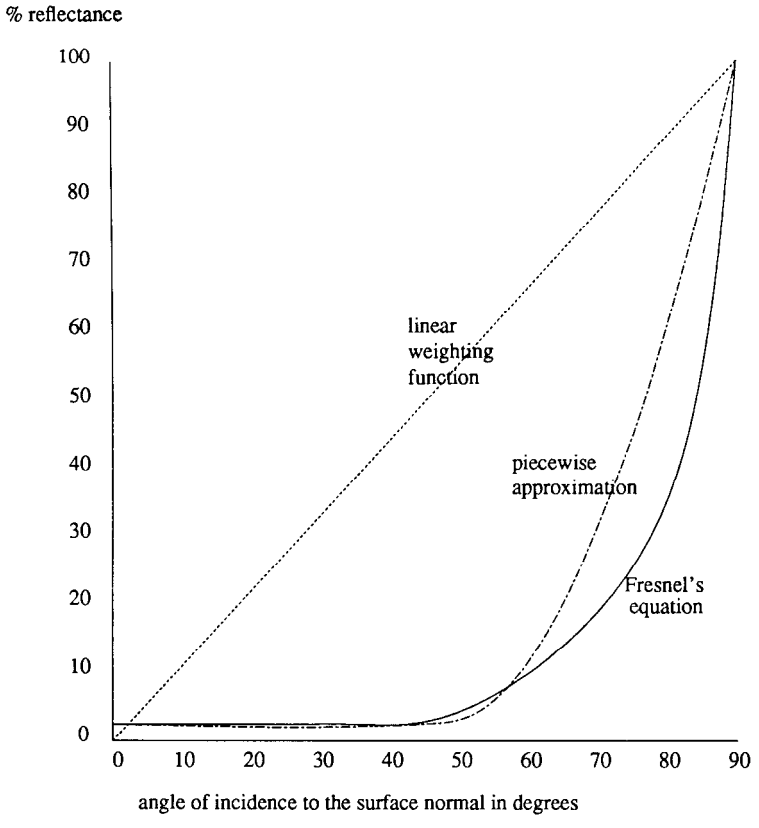

Fig. 11. A comparison of the different weighting functions used.

trigonometric, exponential functions, or the like. Of course other curve formulations offering a closer approximation are also possible.

The appropriate function is selected on the basis of several considerations. The horizontal line and the minimum of the parabola are calculated from Fresnel's law for natural light striking the surface at normal incidence. According to [17], when the angle of incidence with the surface normal approaches zero, implying minimum reflection, the limit of Fresnel's equation is

$$I = \left(\frac{n-1}{n+1}\right)^2.$$

Again using $n = 1.33$ for water, then $I \approx 2$ percent, which means that the horizontal line and the y minimum of the parabola should also equal 0.02 at minimum reflection. At an angle of about 45 degrees, the percentage of reflectance starts to increase noticeably. Thus, we place the x value of the parabola's minimum at 45 degrees. For a line at 90-degree incidence with the surface normal, corresponding to maximum reflectance, the y value of the parabola should indicate 100 percent reflectance. The parabola's axis of symmetry is constrained

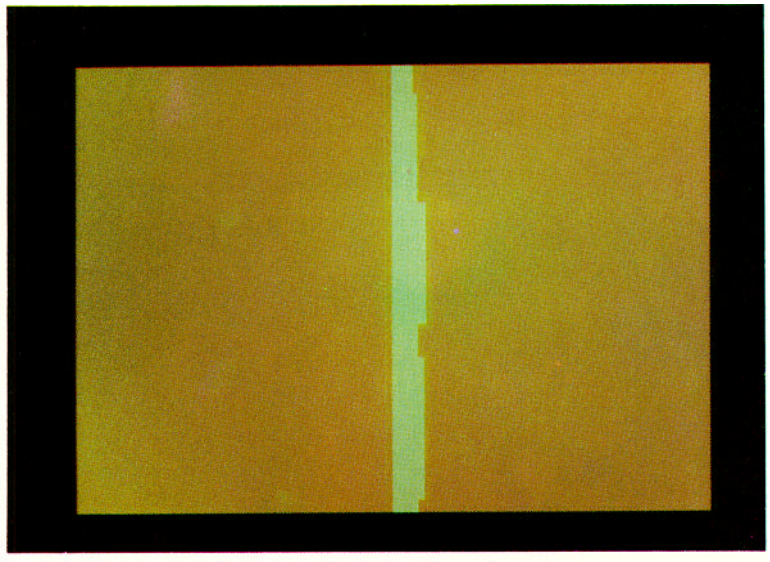

Fig. 12. The reflection texture map used for Figure 14. Notice that the hard edges between the color changes in the texture map do not appear in the final image owing to the dithering or perturbation of indices. The sun streak in the middle was arbitrarily painted by hand.

to be parallel to they axis so that the weighting function is strictly increasing in the range of interest. Thus, the weighting function, in radians, becomes

$$I = \begin{cases} 0.02, & i < \dfrac{\pi}{4} \\ \dfrac{392}{25\pi^2}\left(i - \dfrac{\pi}{4}\right)^2 + 0.02, & i \geq \dfrac{\pi}{4}. \end{cases}$$

Again the angle of incidence is found and then the final color for the point is calculated by **eq.** (4.1). The final images in this section were rendered with this approximated **Fresnel's** equation algorithm.

4.3 The Texture Maps

The contents of the texture maps themselves are somewhat involved. There are many factors that should affect the contents of the reflective texture map, for example, the position of the light source (here assumed to be the sun), the position of the viewer, and the roughness of the waves. A good summary of these factors is found in **[26].** The fact that there are such variables to be handled is a major disadvantage of these algorithms. However, it is probable that the creation of the texture maps could be automated to a large extent. Figure 12 shows a sample reflective texture map.

The refractive texture map could be the color of deep water at one end and gradually change to the texture of the underlying shore on the other end. The color of the shore texture could even be a little darker than the actual shore to

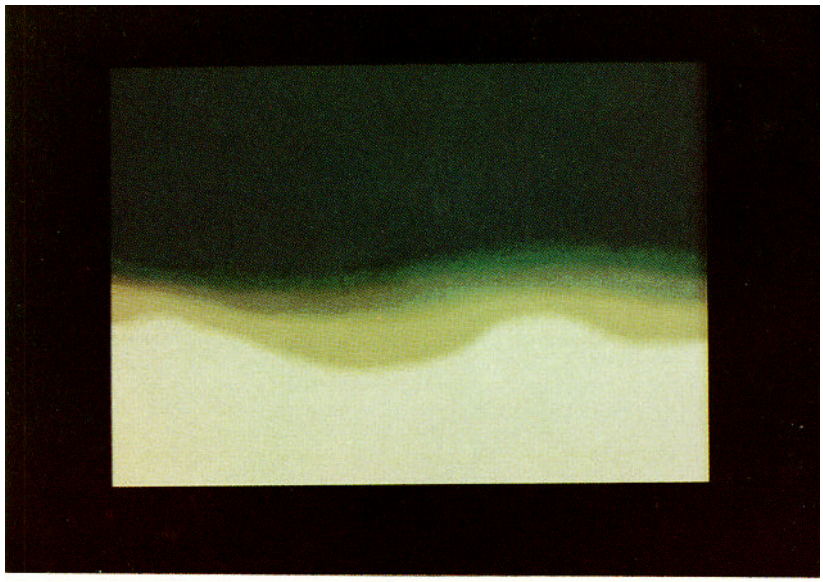

Fig. 13. The refraction texture map used for Figure 15.

imitate the way objects appear darker when wet. Figure 13 shows a sample refractive texture map.

4.4 Implementation Details

In the current implementation, the texture maps are created by hand, and the column of sunlight is also roughly approximated by hand. In addition, the indices into each texture map are randomly perturbed in order to soften any sharply defined color boundaries within the texture maps. This dithering also prevents the texture mapped reflection from appearing fixed when it should be shifting with the movement of the waves. This shifting occurs in reality because waves that are angled differently pick up reflected light from different parts of the environment. The perturbation of the indices roughly simulates this effect. This perturbation need not be based on a pseudorandom number generator. Any reasonable function will suffice, like a simple hashing function based on height such as that used in our implementation. If, however, the water is calm and an undisturbed reflection is desired, this method should be abandoned.

It should be emphasized that the algorithms presented in this section are strictly texture map related. Other aspects of rendering, such as hidden surface removal and lighting, are separate issues. For example, it makes little difference to the approximate Fresnel algorithm whether a z-buffer or a scan-line hidden surface algorithm is used. It also makes little difference whether Gouraud or Phong shading is used. In the course of implementation, however, two considerations were uncovered.

The first concerns the choice of algorithm for hidden surface removal. Because the wave-tracing algorithm coerces wave fronts to wash in parallel to most

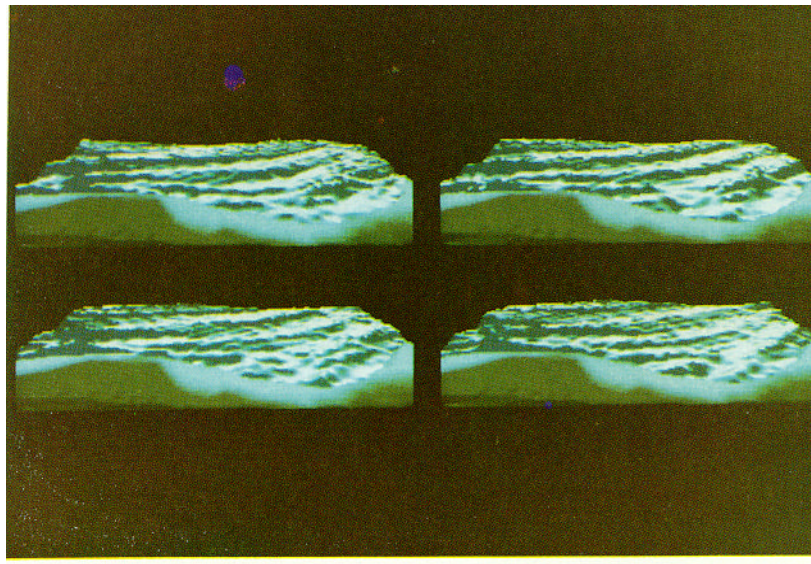

Fig. 14. Four frames from a short animation. The contour information and wave fronts used are shown in Figures 5, 7, and 8. The initial amplitude for the wave in Figure 7 is 0.2 meter and the initial amplitude for the wave in Figure 8 is 0.1 meter. The size of the Beta-spline grid was 50 × 50. There were two wave orthogonals generated for every grid line. This animation repeats itself every 6 seconds. We have found that this period seems a bit short. On the average, waves breaking on Venice Beach, California appear to have about a 10-second period.

shorelines, it can be trivial to approximate small waves lapping against the shore. If a z-buffer algorithm or some other hidden surface algorithm that can handle intersecting polygons is used, then merely cornpositing the water surface polygons with the land surface polygons will result in lapping waves over time.

The second consideration is actually more of a reminder. If a directional light source is used in shading, it should coincide with the colors on the reflective texture map. For example, if the column of sunlight is colored to represent a golden sunset, then the directional light source should also be golden.

Finally, we present some rough run-time figures resulting from using the approximate Fresnel texture mapping algorithm. Figure 14 shows four Beta-spline ocean surfaces, each defined by a 50 x 50 grid of control vertices and subdivided into $(8 \times 48)^2 = 147{,}456$ triangles. The reflective texture map used is shown in Figure 14. The refractive texture map was a solid blue. The renderer uses a z-buffer hidden-surface algorithm and Phong shading. The images are not antialiased. The total CPU time needed for subdividing and rendering one of these surfaces is about 158 minutes on a DEC VAX-11/750 with a floating-point accelerator running UNIX 4.3 BSD. Figure 15 is another image rendered with the approximate Fresnel texture mapping algorithm, with the refractive texture map shown in Figure 13. The reflective texture map was a solid light blue.

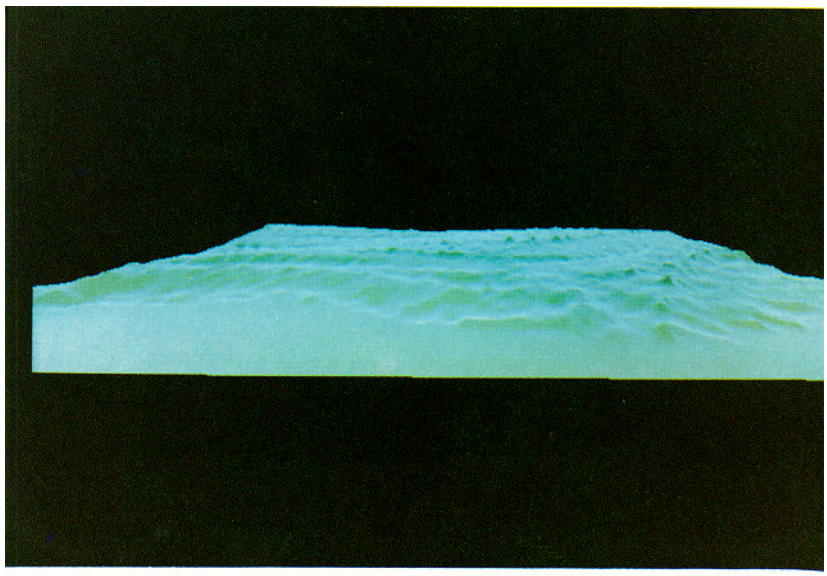

Fig. 15. A still image created using an 80 × 50 Beta-spline mesh. One wave orthogonal was generated for each grid line. The same contour lines were used as in Figure 14, but different depths were assigned to the contour lines. Again from topmost to bottommost, the depths in meters are 5.0, 4.5, 4.0, 2.5, 1.0, 0.5, 0.3, 0.1, −0.1, −0.1. The same two wave fronts were also used, but given different amplitudes. The amplitudes are 0.4 meter and 0.2 meter for the waves in Figure 7 and Figure 8, respectively.

5. FUTURE WORK

There is much work that could be done in these areas of modeling and rendering fluids. The next fundamental step should be to obtain a better understanding of the sampling and resolution problems discussed in Section 3. Wave-tracing is fundamentally a point-sampling technique and thus it inherits all the problems of such techniques. It is expected that solutions and issues previously presented for these techniques (e.g., for ray-tracing and antialiasing) are also appropriate here.

Next, wave reflection effects may be as simple to implement as wave refraction. Wave diffraction is computationally more difficult, but simple cases could be used. The combination of wave-tracing refracting, reflecting, and diffracting wave orthogonals could probably be effectively used to model convincing river flows.

With refraction in place, modeling breaking waves upon the shore should be easier than in the past. There are only four basic types, each with distinct characteristics: spilling, plunging, collapsing, and surging [16]. The surface contours of the underlying shore dictate which type of wave occurs. Since refraction forces wave fronts to become parallel to the shore, a wave front is now in a position to be manipulated into a breaking wave. Also, the same surface contour information used in the refracting wave model can be used in the breaking wave

model. However, the height fields used in wave-tracing are clearly inadequate for representing curling, breaking waves.

Along the same lines, the phenomenon of beach run-up can now be considered also. Beach run-up occurs after a wave has hit the shore. Momentum carries water farther up onto the beach, momentarily wetting the sand and depositing foam. Of course, proper lighting models for wet sand and foam should also be examined.

There are a couple of obvious improvements on the texture mapping algorithms that would increase their speed primarily by reducing the number of polygons that must be processed. One of these improvements would be to cull backfacing polygons, and the other to subdivide the spline surface adaptively according to some flatness criterion. Another speed improvement may be to replace the piecewise function used in the approximate Fresnel algorithm with a look-up table containing the results of evaluating Fresnel's law over a large range of incidence angles.

Currently, perhaps the slowest process of all is the creation of the texture maps. Finding a convincing combination of colors for water, sand, and sky is difficult. The easiest solution we found was to try to match the colors from photographs.

If additional environment information were provided, however, some aspects of texture map creation could be automated. For example, it was mentioned in Section 4 that the refractive texture map could change with depth. Currently, this is approximated by hand, but it should be relatively straightforward to use the contour depth information to automatically generate the change in the refractive color information and, in fact, even to replace the refractive texture map with this procedural method.

Another improvement would be to index spherical texture maps according to the angles of reflection and refraction. In particular, this method, known as *reflection mapping*, would alleviate the need to create the reflected patch of sunlight by hand and similar texture map construction problems.

Finally, finding computationally cheaper versions of the algorithms presented here and more physically accurate models are always areas of future work.

6. CONCLUSIONS

In this paper we have presented an algorithm for modeling ocean waves that addresses the phenomenon of water wave refraction. This algorithm of wave-tracing is quite similar in nature to the rendering algorithm of ray-tracing. Modeling wave refraction through wave-tracing produces two major features: the simple removal of undesirable spatial periodicity and the simple handling of desirable temporal periodicity.

Approximating the resulting ocean surface with a Beta-spline surface provides us with definite advantages over a polygonal representation. Most important, the tension (or $\beta 2$) shape parameter allows us to easily add more complexity to the surface. Surface representations can also be more compactly stored than polygonal representations.

Parametric surface representations, such as the Beta-spline, also permit simple mappings to be applied to texture maps. Because of this property, two rendering

algorithms based on texture maps have been presented that are not as physically accurate as ray-tracing, but are computationally quicker. These algorithms attempt to simulate the effects of both light reflection and refraction by using a weighted average of the color information from two texture maps. Although the general technique was found to be relatively fast, the satisfactory creation of the two texture maps was found to be a nontrivial problem.

ACKNOWLEDGMENTS

The authors would like to thank Professors Carlo Séquin and Ronald Yeung of the University of California, Berkeley, for their many helpful suggestions and two doctoral students at the Berkeley Computer Graphics Laboratory, Tony DeRose and Mark Dippé, who wrote the software for surface modeling and rendering, respectively.

REFERENCES

(Note: References [10], [19], [34], and [35] are not cited in text.)

1. BARSKY, B. A. The Beta-spline: A local representation based on shape parameters and fundamental geometric measures. Ph.D. dissertation, Univ. of Utah, Salt Lake City, Dec. 1981.
2. BARSKY, B. A. *Computer Graphics and Geometric Modelling Using Beta-Splines.* Springer-Verlag, Heidelberg, 1987.
3. BARSKY, B. A., AND BEATTY, J. C. Local control of bias and tension in Beta-splines. In *SIGGRAPH '83 Conference Proceedings* (Detroit, Mich., July 25–29). ACM, New York, 1983, pp. 193–218. Also *ACM Trans. Graph. 2*, 2 (Apr. 1983), 109–134.
4. BARSKY, B. A., AND DEROSE, T. D. The Beta2-spline: A special case of the Beta-spline curve and surface representation. *IEEE Comput. Graph. Appl. 5*, 9 (Sept. 1985), 46–58. Correction in "Letter to the Editor," *IEEE Comput. Graph. Appl. 7*, 3 (Mar. 1987), 15.
5. BARSKY, B. A., DEROSE, T. D., AND DIPPÉ, M. D. An adaptive subdivision method with crack prevention for rendering Beta-spline objects. Tech. Rep. UCB/CSD 87/348, Computer Science Division, Electrical Engineering and Computer Sciences Dept., Univ. of California, Berkeley, Mar. 1987.
6. BLINN, J. F. Simulation of wrinkled surfaces. In *SIGGRAPH '78 Conference Proceedings* (Atlanta, Ga., Aug.). ACM, New York, 1978, pp. 286–292.
7. BLINN, J. F., AND NEWELL, M. E. Texture and reflection in computer generated images. *Commun. ACM 19*, 10 (Oct. 1976), 542–547.
8. BRESENHAM, J. E. Algorithm for computer control of digital plotter. *IBM Syst. J. 4*, 1 (1965), 25–30.
9. CATMULL, E. E. A subdivision algorithm for computer display of curved surfaces. Ph.D. dissertation, Univ. of Utah, Salt Lake City, Dec. 1974. Also Tech. Rep. UTEC-CSc-74-133, Dept. of Computer Science, Univ. of Utah, Salt Lake City.
10. CURRIE, I. G. *Fundamental Mechanics of Fluids.* McGraw-Hill, New York, 1974.
11. EAGLESON, P. S., AND DEAN, R. G. Small amplitude wave theory. In *Estuary and Coastline Hydrodynamics*, A. T. IPPEN, Ed. McGraw-Hill, New York, 1966, pp. 1–92.
12. FOLEY, J. D., AND VAN DAM, A. *Fundamentals of Interactive Computer Graphics.* Addison-Wesley, Reading, Mass., 1982.
13. FORSYTHE, G. E., MALCOLM, M. A., AND MOLER, C. B. *Computer Methods for Mathematical Computations.* Prentice-Hall, Englewood Cliffs, N.J., 1977.
14. FOURNIER, A., FUSSELL, D., AND CARPENTER, L. Computer rendering of stochastic models. *Commun. ACM 25*, 6 (June 1982), 371–384.
15. FOURNIER, A., AND REEVES, W. T. A simple model of ocean waves. In *SIGGRAPH 86 Conference Proceedings* (Dallas, Tex., Aug. 18–22). ACM, New York, 1986, pp. 75–84.
16. GALVIN, C. J. JR., Wave breaking in shallow water. In *Waves on Beaches and Resulting Sediment Transport*, R. E. Meyer, Ed. Academic Press, Orlando, 1972, pp. 413–456.
17. HARDY, A. C., AND PERRIN, F. H. *Principles of Optics.* McGraw-Hill, New York, 1932.

18. HARUYAMA, S., AND BARSKY, B. A. Using stochastic modeling for texture generation. *IEEE Comput. Graph. Appl. 4*, 3 (Mar. 1984), 7–19. Errata *IEEE Comput. Graph. Appl. 5*, 2 (Feb. 1985), 87.
19. HUGHES, W. F., AND BRIGHTON, J. A. *Fluid Dynamics.* McGraw-Hill, New York, 1967.
20. KAY, D. S. Transparency, refraction, and ray tracing for computer synthesized images. Master's thesis, Cornell Univ., Ithaca, N.Y., Jan. 1979.
21. LANE, J. M., AND CARPENTER, L. C. A generalized scan line algorithm for the computer display of parametrically defined surfaces. *Comput. Graph. Image Process. 11*, 3 (Nov. 1979), 290–297.
22. LANE, J. M., CARPENTER, L. C., WHITTED, J. T., AND BLINN, J. F. Scan line methods for displaying parametrically defined surfaces. *Commun. ACM 23*, 1 (Jan. 1980), 23–34.
23. LE MÉHAUTÉ, B. *An Introduction to Hydrodynamics and Water Waves.* Springer-Verlag, New York, 1976.
24. MARTIN, L. C. *Geometrical Optics.* Philosophical Library, New York, 1956.
24a. MASTEN, G. A., WATTERBERG, P. A., AND MAREDA, I. F. Fourier synthesis of ocean scenes. *IEEE Comput. Graph. Appl. 7*, 3 (Mar. 1987), 16–23.
25. MAX, N. L. Vectorized procedural models for natural terrain: Waves and islands in the sunset. In *SIGGRAPH '81 Conference Proceedings* (Dallas, Tex., Aug. 3–7). ACM, New York, 1981, pp. 317–324.
26. MINNAERT, M. *The Nature of Light and Colour in the Open Air.* Dover, New York, 1954.
27. NORTON, A., ROCKWOOD, A. P., AND SKOLMOSKI, P. T. Clamping: A method of antialiasing textured surfaces by bandwidth limiting in object space. In *SIGGRAPH '82 Conference Proceedings* (Boston, Mass., July 26–30). ACM, New York, 1982, pp. 1–8.
28. PEACHEY, D. R. Modeling waves and surf. In *SIGGRAPH '86 Conference Proceedings* (Dallas, Tex., Aug. 18–22). ACM, New York, 1986, pp. 65–74.
29. PERLIN, K. An image synthesizer. In *SIGGRAPH '85 Conference Proceedings* (San Francisco, Calif., July 22–26). ACM, New York, 1985, pp. 287–296.
30. SCHACHTER, B. J. Long crested wave models. *Comput. Graph. Image Process. 12* (Feb. 1980), 187–201.
31. SPEER, L. R., DEROSE, T. D., AND BARSKY, B. A. A theoretical and empirical study of coherent ray-tracing. In *Proceedings of Graphics Interface '85* (May 27–31). Canadian Information Processing Society, Montreal, 1985, pp. 1–8. Revised version in *Computer-Generated Images—The State of the Art*, N. Magneuat-Thalmann and Daniel Thalmann, Eds. Springer-Verlag, New York, 1985, pp. 11–25.
32. STOKER, J. J. *Water Waves.* Interscience, New York, 1957.
33. TIPLER, P. A. *Physics.* Worth, New York, 1976.
34. TRICKER, R. A. R. *Bore, Breakers, Waves and Wakes: An Introduction to the Study of Waves on Water.* American Elsevier, New York, 1965.
35. TS'O, P. Y. J. Graphical simulation of hydrodynamics: Modeling and rendering waves. M.S. thesis, Univ. of California, Berkeley, Calif., Dec. 1985.
36. WHITTED, J. T. An improved illumination model for shaded display. *Commun. ACM 23*, 6 (June 1980), 343–349.
37. WIEGEL, R. L. *Waves, Tides, Currents and Beaches: Glossary of Terms and List of Standard Symbols.* Council on Wave Research, The Engineering Foundation, July 1953.

Received December 1985; revised March 1987; accepted April 1987