Procedural Models on Image Synthesis for Ocean Animation

CESAR TADEU POZZER[1], SÉRGIO ROBERTO MATIELLO PELLEGRINO[2]
[1,2]ITA – Instituto Tecnológico de Aeronáutica, Pç. Mal Eduardo Gomes, 50, São José dos Campos, SP, Brasil
[1]pozzer@inf.puc-rio.br, [2]pell@comp.ita.br

Abstract: This paper presents image-synthesis techniques to define, animate and visualize scenes simulating the behavior of the open-sea surface. Wave modeling is obtained by means of the bump mapping technique. Wave texture, defined by noise functions, is applied by means of a solid-space animation technique to generate surface animation. An implementation based on the Phong model is used to determine diffuse and specular reflections. As a result of the synthesis process, a set of frames is obtained, to be used to compose the animation.

1 Introduction

Modeling natural phenomena is, beyond doubt, one of the most complex tasks in the Image Synthesis field. This can be justified by observing the physical rules and properties that dictate the behavioral aspect of such phenomena. Since most of them are asymmetric and non-rigid elements, they add great complexity to the model. Both their definition and their behavior in time result from external factors such as wind, temperature, etc.

The correct evaluation, or emulation, of such external factors is the key to obtain an image or animation with a high degree of realism, which in this case refers to visual aspect and level of detail. In an accurate representation of a natural phenomenon, the use of complex equations is made necessary – in this case, relative to Fluid Theories.

Commonly adopted solutions employ approximate equations, such as those presented in Max [1]. His wave model consists of several superimposed linear sinusoidal waves simulating low-amplitude ocean waves. However, the high periodicity of the obtained surface is a downside. TS'O [2] uses wave refraction to determine wave behavior and direction. The surface is represented by beta-splines and is rendered using texture mapping [13], which simulates light reflection and refraction.

Attempting to simulate fog and foam, Peachey [3] and Fournier et al. [4] employ particle systems [5] combined with simple hydrodynamic equations. In Fournier et al. [4], the sea surface is considered as a set of particles performing circular or elliptical stationary orbits, which can simulate wave breaks.

In order to abstract the complexity of the equations, Perlin [6] simulated surface perturbation by using noise functions combined with Blin's [7] bump mapping technique. One can notice that this method works well when the observer is far enough from the surface for the sea to seem flat. Other recent works refer to the use of procedural methods, such as the fractal method [8], combined with the bump mapping technique.

In the present paper, sea-surface definition is achieved by means of procedural methods similar to Perlin's [6]. Solid-space animation techniques [8] are employed in the animation process, and illumination techniques based on the Phong model [9] are applied to the scene-visualization process.

This paper is organized as follows: Section 2 presents the techniques developed for defining the surface. In Section 3, solid-space definition and animation is discussed. Section 4 deals with illumination techniques. The results are presented in Section 5, and final conclusions are drawn in Section 6. All image results are shown in the last page.

2 Surface Modeling

The ocean surface is defined by a flat plane. To achieve the irregular 3D appearance of the waves, the bump mapping technique [7] and a noise function [6] are used, producing values employed in the perturbation of the surface's normal vector. In this paper, such values are considered as texture values and are used in the definition, disposition and motion of the waves.

2.1 Noise-Function Definition

A feature of the noise function is that it generates values with no correlation with any previously generated one. In the scope of this work, the generation of random values prevents controlling the transition among animation frames, as well as the shape, size and disposition of the waves. A two-dimensional matrix of pseudo-random numbers (PRN) is then used. It is defined solely by integer coordinates, reflecting the characteristics of the ocean surface, and its size influences pattern repetitions on the surface. This matrix is a height field and its values are used as control points during the scene rendering process.

With the PRN matrix, the noise function is then defined as an indexing function of PRN relative to the coordinates associated to the real scene, which are defined by means of a sampling process using ray casting.

A crucial point is defining the matrix values, as the matrix is to determine the ocean shape during the animation. Depending on the ocean's behavior (wave perturbation, speed and shape), different texture configurations must be obtained. For non-regular surfaces, a good solution is to use the following algorithm:

```
for each texture pair (x,y)
    texture[x][y] = random( )*perturb
```

Algorithm 1: Simple noise function

where the perturb variable is used as a perturbation limiter. An example of a result of this algorithm is shown in Figure 1(a), obtained by means of the biparametric interpolation.

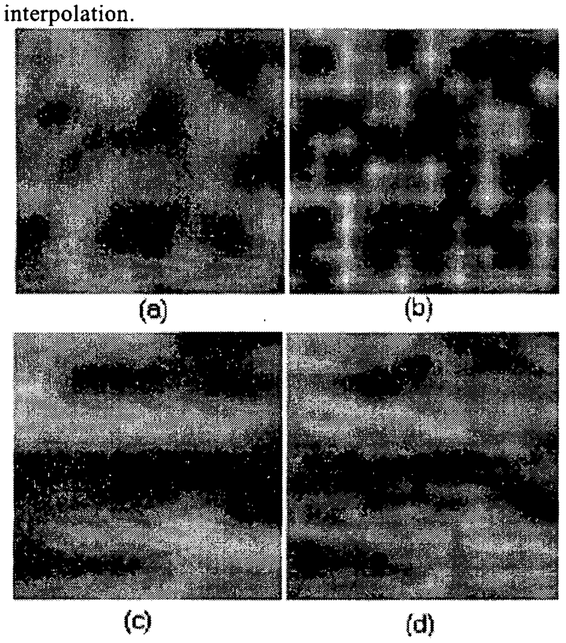

Figure 1: Texture examples: (a) random values and B-spline interpolation; (b) exponential interpolation; (c) multiplicity = 6; (d) two texture layers

For generating waves with a more controllable shape, an adaptation of this method was developed in which data blocks with the same value are defined, thus generating the desired aspect for the local surface. Such data can even be easily modified by manipulating the PERTURB and MULTIPLICITY parameters, the latter being responsible for block size definition, as shown in Algorithm 2.

```
value  = random()*perturb
lenght = random()*multiplicity
for count = 0 to lenght
    texture[x+count][y] = value
```

Algorithm 2: Regular wave generation

Figure 1(c) illustrates the effect value multiplicity has over wave definition, in contrast to the surface generated by Algorithm 1. The light and dark longitudinal regions represent the wave's crest and trough, respectively. One can notice that the wavelength, given by the distance between two crests, is not homogeneous in every region of the surface, thus producing more random results, especially during the animation.

2.2 Texture Evaluation

Since texture is defined only by integer coordinates, some interpolation techniques are necessary to evaluate the texture at any position. The kind of interpolation used affects the visual aspect obtained, and can generate surfaces with smooth or rough transitions.

Figure 1(a) and 1(b) shows 3-degree polynomial interpolation and exponential interpolation examples, respectively, for the same control points. The exponential interpolation is local to the control points and uses only four neighboring control points, forming parallel lines with rough variations (lighter regions). Discontinuities can be noticed, which on the one hand can simulate the agitated sea but on the other hand make some regions prone to the appearance of alias.

Using B-spline interpolation (Figure 1(a,c,d)), smoother surfaces are obtained, representing little-perturbed water without highlighting the presence of control points. Its disadvantage is the great processing time needed for evaluating it. This kind of interpolation is prone to generating alias only when the samples are taken from points near the horizon (far from the observer).

When the surface has been created, the perturbation value is computed taking the direction of the normal vector in each point in the texture. Assuming T determines a texture patch, the normalized perturbation vector P_T is given by

$$\vec{P}_T = \frac{\left(\frac{\partial T}{\partial x} \times \frac{\partial T}{\partial y}\right)}{\left\|\left(\frac{\partial T}{\partial x} \times \frac{\partial T}{\partial y}\right)\right\|}$$

To reduce the problem of an over-smooth perturbation, two texture layers, T_1 and T_2, are evaluated in different scales. Usually a 1x4 scale relation was adopted. The larger the scale, the smaller the wave. As a result, textures such as the one illustrated in Figure 1(d) – which derives from Figure 1(c) – are obtained.

At each texture layer, the multiplying factors M_1 and M_2 are also associated. They allow controlling the perturbation degree without the need for generating new control points. The resulting perturbation vector P_r is given by the equation below and is used to change the direction of the normal vector in directions x and y.

$$\vec{P}_r = \vec{P}_{T_1} * M_1 + \vec{P}_{T_2} * M_2$$

The new normal direction, to be used in the diffuse and specular illumination computation, is given by

$$\vec{N}_r = \vec{N} + \vec{P}_r$$

3 Surface Animation

Due to the fact that surface animation is associated to texture, the latter must undergo changes through time to better represent wave motion on the ocean. These changes must affect not only wave disposition, but also their shape.

Wave disposition is based on texture displacement along axes x and y. With this strategy, only wave dislocation can be simulated, in a specific direction. Shape variation is a more complex task that requires smooth transition between every animation frame. Simply generating a new texture at each frame would result in rough transitions, which would not represent a realistic behavior.

The solution developed consists in generating a solid (three-dimensional) texture [10] composed by a set of two-dimensional texture layers (Figure 2), independently generated by Algorithm 2.

In the present work, the use of solid texture has a slightly different focus than the original proposition, for two reasons:

1. The rendered object (water) is not considered three-dimensional, but rather a flat plane on which bump mapping is applied;
2. The solid texture's values are used as a perturbation function (Blin's [7] bump function).

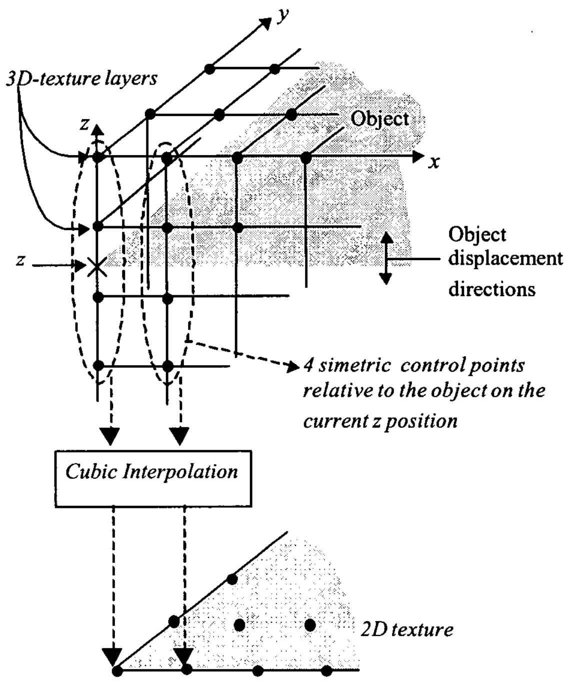

Figure 2: 2D texture generation from a solid texture

There are two approaches for animating a solid texture: changing its value in time or moving the object being rendered along its space. In this work, the object is represented by the shaded plane illustrated in Figure 2, which moves along the z axis inside the solid texture. As the plane moves through the solid texture, a two-dimensional texture is extracted, to be used in frame generation. This texture is actually de PRN described in Section 2.1.

The extraction follows an interpolation process among these layers and is closely related to the animation, which is based on the solid-space animation concept. Each element in this plane is computed by interpolating the four neighboring control points (2 above and 2 below) located on the solid texture along the z axis, whose x and y coordinates are equal to the point on the plane being computed. As this is a reticulate plane, only integer coordinates can be defined. The interpolation technique employed is 3-degree Catmul-Rom [8]. The number of points to be defined depends on the texture dimension, which is the same as the solid texture's.

With this animation technique, 3 parameters are used in the animation: x and y for wave displacement (speed and direction) and a z parameter (shape change) that determines which solid region in the 3D texture will be used to define the control points for the 2D texture.

4 Rendering Issues

Since the three-dimensional characteristic of the ocean's surface is not determined by a geometric model, illumination techniques are needed to give the waves a 3D aspect.

Specular and diffuse reflection factors are associated to each light source, and not to the surface. Therefore, different light sources, which are considered as a point source, may have different behaviors over the same surface.

Scene rendering follows a sampling process in which each point is computed independently from its neighbors, and plane-ray intersection operations, adaptive sampling and other point-driven methods [11] can be directly applied to the surface. The present work employs ray casting with uniform sampling or regular supersampling.

4.1 Illumination Geometry

The influence of the light sources on the generated image is given by angles α, β and γ (Figure 3). Angles α and β represent the incidence and the reflection angles of the light ray, respectively, and determine the elevation of the source and the observer in relation to the synthesis plane. The value of γ, which relates source alignment to the observer, is given by the angle between N_1 and N_2. These three angles are used to compute the diffuse reflection (DR) and specular reflection (SR) parameters, whose purpose is to attenuate the reflected light, whose maximum value occurs when $\alpha = \beta$ and $\gamma = 0$.

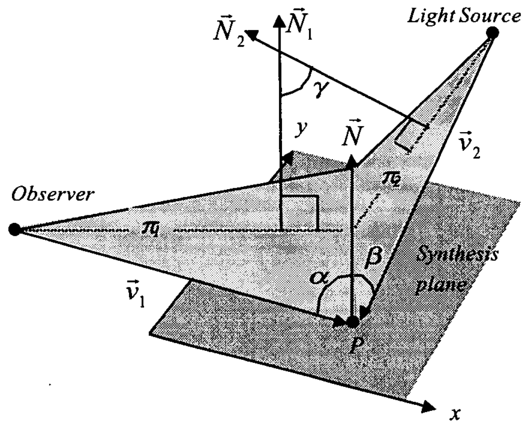

Figure 3: Illumination geometry

4.2 Specular Reflection

Since the Phong model presents smooth color transitions, we developed a similar specular-reflection model which attempts to break this smoothness by varying the light-incidence angle. This model is divided in two stages:

1. Computing the δ factor, which represents the attenuation of the reflected light, according to α, β and γ;
2. Using this factor as an index in a lookup table representing the specular intensity distribution.

For computing δ, the specular-reflection coefficients $Spc\gamma$ and $Esp\alpha\beta$ associated to each light source are used. They attenuate reflection, taking into account angles (α-β) and γ, respectively. δ varies between 0 and 90, and is defined by equation

$$\delta = \cos(\mathrm{abs}(\alpha - \beta))^{Esp\alpha\beta} * \cos(\gamma)^{Esp\gamma}$$

With this equation, reflected light can be concentrated on a section relative to the longitudinal and transversal axes that will represent the region with maximum reflection, as illustrated in Figure 4. In the results, the effect these parameters have on the image are shown.

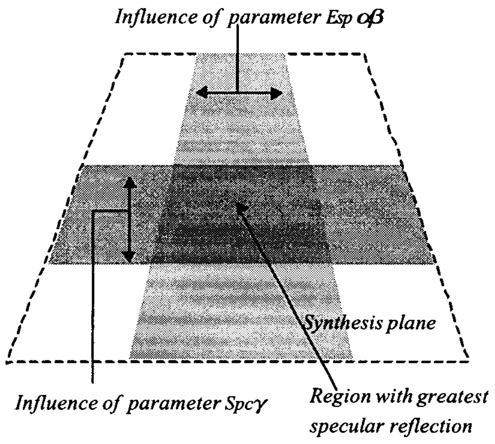
Influence of parameter Esp $\alpha\beta$
Synthesis plane
Influence of parameter Spcγ
Region with greatest specular reflection

Figure 4: Areas of specular reflection concentration

The δ factor having been defined, it is used as an index in the lookup table presented on Figure 5. This table is divided into three sections (represented by numbers 1, 2 and 3), parallel to axis y. Sections 2 and section 3 represent the conventional Phong model each. Both the curve and the distances are given by parameters lim_1, lim_2, n_1, n_2. The two latter represent specular-reflection coefficients, similar to the Phong model, and influence the shape of the curve for each associated section.

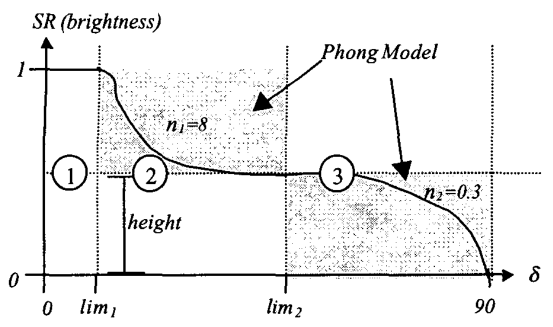

Figure 5: Lookup table of specular intensity

Depending on δ different SR values are obtained. If $\delta < lim_1$, SR will have maximum reflection, represented by the white color. Other values of δ are mapped between $[lim_1, lim_2]$ or $[lim_2, 90]$ and rescaled between [0, 90], so that the curve will have a correct interpretation – which, as in the Phong model, varies between [0, 90]. The SR coefficient is used for computing the color specularly reflected by the surface.

Some further observations can be drawn about the model:

1. It can be converted to the conventional Phong model if $lim_1 = 0$, $lim_2 = 90$ and $height = 0$;
2. The minimum specular-reflection value can be *height* if $lim_2 = 90$;
3. The maximum specular-reflection value can be *height* if $lim_1 = lim_2 = 0$;
4. It will not have maximum reflection if $lim_1 = 0$.

Based also on the SR value, the diffuse-attenuation coefficient (*KDiffuse*), to be used in the diffuse-reflection model, is computed by equation

$$KDiffuse = (1 - SR)$$

4.3 Diffuse Reflection

The contribution of diffuse reflection (DR) is similar to that of specular reflection, being given by the following equation, which does not consider the observer's position:

$$DR = \cos(\beta)^{Dif\beta} * \cos(\gamma)^{Dif\gamma} + Ia * kdDiffuse * Kd$$

where $Dif\beta$ and $Dif\gamma$ are diffuse-reflection coefficients, with the same effect as specular coefficients. *Ia* represents ambient illumination, and *Kd* represents the diffuse-reflection coefficient of the surface. Diffuse-reflection distribution is also similar to that of specular reflection (Figure 5). Therefore, using coefficients $Dif\beta$ and $Dif\gamma$ one can concentrate diffuse light around the specular reflection.

Just as in specular reflection, DR is used to compute the texture's color contribution to the final image. By summing specular and diffuse reflections, the final intensity of the image is obtained:

```
Finalcolor = DR*WaterColor +
                SR*SpecularLightSourceColor
```

4.4 Results

The images presented in this section show how the variation of perturbation and illumination parameters affects the generated image. All these images, in color, can be found at
http://www.inf.puc-rio.br/~pozzer/ocean/index.html
http://www.comp.ita.cta.br/~pell/ocean/index.html

Image 1 uses one texture layer with high perturbation, and though there are no hidden waves the bump mapping method generates a good 3D effect. Maximum specular reflection can be seen on the brighter regions, which tend to appear as light blue when specular reflection decreases. Specular and diffuse reflections are shown in Images 2 and 3. The yellowish regions represent specular reflections, while the others represent diffuse reflections. The remaining images illustrate scenes with horizon. Image 5 shows horizon color transitions and sunlight reflection on the ocean. Specular reflection is concentrated near the direction of the sun. Diffuse reflection is blue. On Images 2, 4, 5 and 6 two texture layers are used. Images 4 and 6 display night scenes. Image 6 is a fictitious scene with two moons, and it shows the different specular and diffuse reflections associated to each light source.

All images were generated on an IBM PC Pentium III 600 Mhz with 128Mb RAM, using a JDK 1.2.1 compiler. Image 2 took 21 seconds to be generated and image 5 took 14 seconds, both at a 290x260 pixel resolution.

5 Implementation Details

An application was developed in Java 1.2 that implements visual resources such as synthesis-plane orientation, light-source specification (position, intensity, color) and an automatic animation method. Specific detail on this implementation, as well as illumination and animation techniques and results, can be found in Pozzer [12].

The source code of the application, configuration files used for generating the images presented in this work and the images and animations themselves are available at
http://www.inf.puc-rio.br/~pozzer/ocean/index.html
http://www.comp.ita.cta.br/~pell/ocean/index.html

6 Conclusions

Illumination has clearly the most important role in defining the details that provide realism to the images. With a correct parameter configuration, the illumination techniques developed were shown to be very generic in the definition of different kinds of scenes. Specular reflection, which is the most complex type of reflection to simulate, presented color variations in the provided images that generated good results, especially when the horizon is defined.

The bump mapping technique has fulfilled our expectations. Though it is a simplified technique, it was able to simulate the surface with some degree of realism, even without hidden surface detection.

Concerning texture definition, using the original and the modified noise function we were able to define several behaviors and shapes. With relatively small matrices a non-repetitive aspect was obtained which is similar to a real surface. The animation technique for such surface also proved to be very efficient, since it was able to perform smooth and continued transitions between animation frames.

On performance evaluation, both the techniques used and the Java language were efficient, since the processing time for the images did not take longer than 30 seconds.

Acknowledgments

We would like to thank Mr. Andrew Glassner for the discussions on illumination. This research was supported by CAPES.

References

[1] MAX, N. L. Vectorized Procedural Models for Natural Terrains: Waves and Islands in the Sunset. *Computer Graphics*, v.15, n.3, Aug. 1981, pp. 317-324.

[2] TS'O, P. Y., BARSKY, B. A. Modeling and Rendering Waves: Wave-Tracing Using Beta-Spline and Reflective and Refractive Texture Mapping. *ACM Transaction on Graphics*, v.6, n.3, Jul. 1987, pp. 191-214.

[3] PEACHEY, D. R. Modeling Waves and Surfaces. *Computer Graphics*, v.20, n.4, Aug. 1986, pp. 65-74.

[4] FOURNIER, A., REEVES, W. T. A Simple Model of Ocean Waves. *Computer Graphics*, v.20, n.4, 1986, pp. 75-84.

[5] REEVES, W. T. Particle Systems - A Technique for Modeling a Class of Fuzzy Objects. *Computer Graphics*, v.17, n.3, Jul. 1983, pp. 359-376.

[6] PERLIN, K. An Image Synthesizer. *Computer Graphics*, v.19, n.3, Jul. 1985, pp. 287-296.

[7] BLIN, J. F. Simulation of Wrinkled Surfaces. *Computer Graphics*, v.12, n.3, Aug. 1978, pp. 286-292.

[8] EBERT, D. et al. *Texturing & Modeling*. 2.th. San Diego: AP Professional, 1998, 415 pp.

[9] PHONG, B. Illumination for Computer-Generated pictures. *Communications of the ACM*, v.18, n.3, Jun. 1975, pp. 311-317.

[10] PEACHEY, D. R. Solid Texturing of Complex Surfaces. *Computer Graphics*, v.19, n.3, Jul. 1985, pp. 279-286.

[11] COOK, R. L. Stochastic Sampling in Computer Graphics. *ACM Transactions on Graphics*, v.5, n.1, Jan. 1986, pp. 51-71.

[12] POZZER, C. T. *Uso de técnicas de síntese de imagens aplicadas a um ambiente de representação de superfícies líquidas estáticas e dinâmicas.* MSc Thesis, Computer Science Department, ITA, Feb. 2000.

[13] BLIN, J. F. Texture and Reflection in Computer-Generated Images. *Communications of the ACM*, v.19, n.10, Oct. 1976, pp. 542-547.

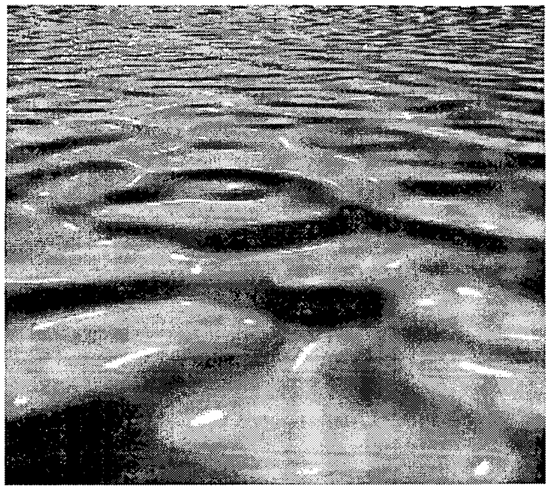
Image 1

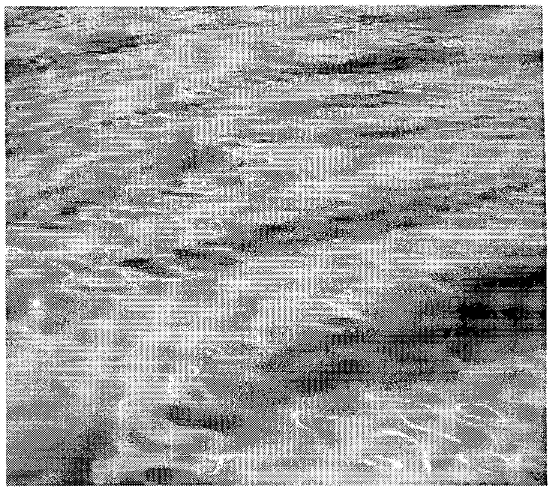
Image 2

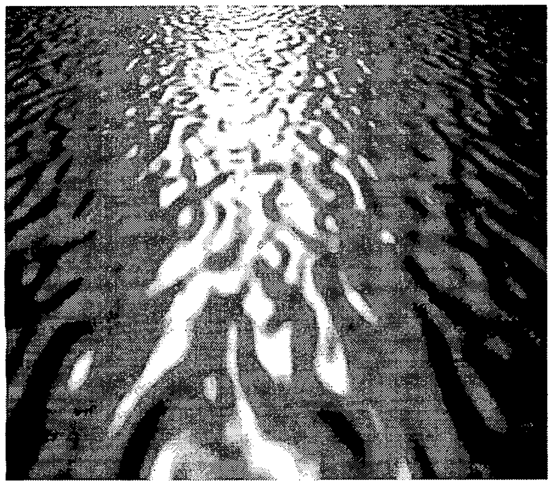
Image 3

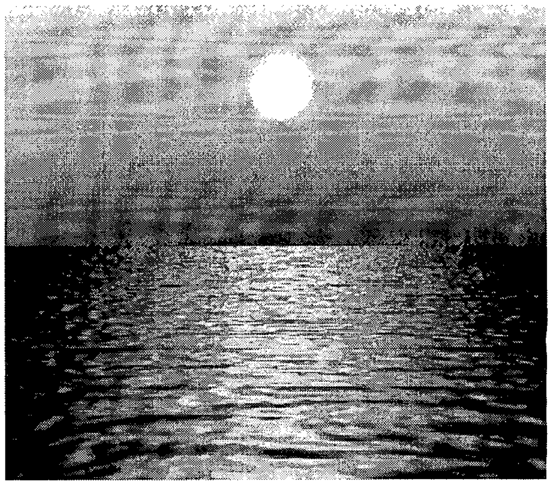
Image 4

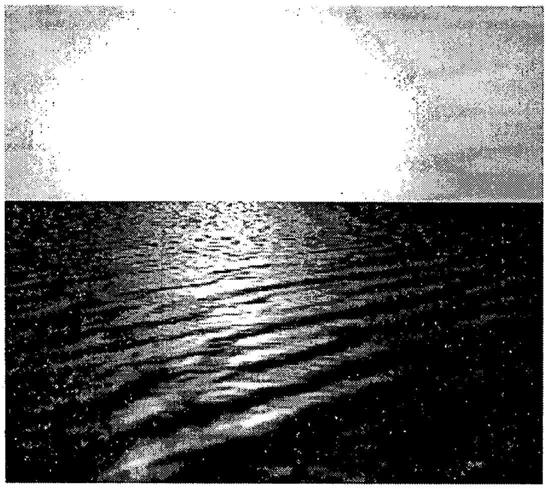
Image 5

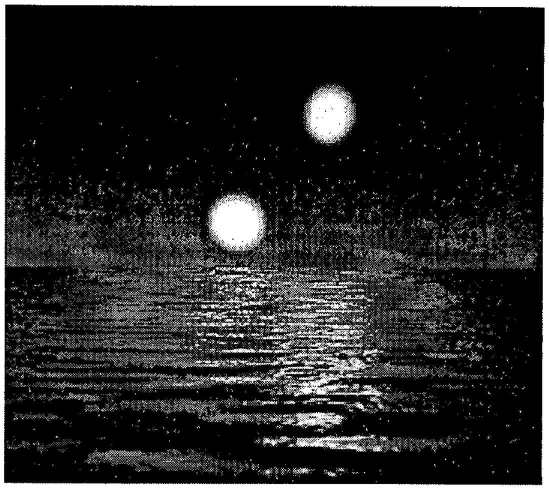
Image 6