Real-time rendering of shallow water

Vladimir Belyaev
Applied Math. Department,
St. Petersburg State Polytechnical University, St. Petersburg, Russia
vladimir@d-inter.ru

Abstract

The correct color of water and bottom refraction is one of the most important parts of realistic visual water simulation. The paper is devoted to the real time rendering of scenes with shallow water when the bottom is visible. It covers the model of light transport at the water surface and through the water medium that allows us to calculate spectral intensity of light at the surface; the human vision imitating model that converts spectral intensity into RGB-color; and practical method of real time low cost integration of this method into existing methods of water surface modeling.

Keywords: real-time rendering, water surface simulation, water color, light transport equation, human vision.

1. INTRODUCTION

The paper is devoted to the modeling of scenes with shallow water when the bottom is visible. In this case very important role plays an effect of how water colorizes the bottom and the water color itself when bottom goes deep (see Fig. 4).

So our aim is to develop the model of “water colorizing” of the bottom and integrate it into early developed method of visualizing of such scenes [1]. We do not intent to develop completely physically correct model of this process because it is unnecessary: it is very hard to guarantee adequateness of the color on the display. Everybody knows the situation when the same picture looks different on different displays. Moreover display is unable to reproduce the range of intensities that human eye can perceive.

Our methods are real time oriented (to be more correct, the water modeling system is a part of virtual reality application), so it should take not more than 25% of overall processors time, while the refresh rate should be 30 or 60 times per sec.

This strong limitation leads us to the idea that all calculations for the model should be performed on the preconditioning stage or else in the GPU (graphics card processor) which works much faster due to its architecture and specialty.

We have to consider three parts of modeling:

- developing physical model of water color taking the bottom into account that depends on different conditions (e.g. time of day, characteristics of water and dredges in it, etc.);
- imitating human vision: conversion of the image in the energy-spectral representation into RGB-color one;
- integration into the visualization method, proposed by Jensen in [2] and modified by us [1].

2. RELATED WORK

There is not much attention paid in the literature to the problem of water surface color. Roughly, we can separate existing approaches onto three directions.

The first one imitates the reflection and refraction phenomena, but does not take water coloring into account (or water color is imitated by bottom coloring by-hand) [1-2]. This method is widely used in many modern computer games.

The classic work of second direction is [3]. It adequately imitates the water color, using enough simple model, and approximates its parameters. But this model assumes that the bottom is deep enough and can not be seen.

The third direction calculates the light field in the water medium, integrates the light transportation equation with different approximations. E.g. in work [4] the second-order scattering approximation is used and accelerated by means of modern graphics cards. This method gives the most physically correct result, but can not be used in our case due to performance restrictions.

The process of RGB-color calculation from its energy-spectral representation (light intensity as a function of wave length) might be divided on two parts: chromatic and luminance, as in human eye perception. There are not many methods that consider both phenomena (e.g. [5]). Most works are dedicated to one of the two parts. The papers [6-8] and small section of [9] examines chromatic adaptation only. Luminance adaptation or HDR (High Dynamic Range) problem is also very popular among researchers. One of the most important works here is [10].

But the conversion from spectral representation to RGB color is not possible in real time now, so we need to perform all calculations on the preprocessing stage and just use its results in real time.

3. OPTICAL PHENOMENA ON THE SURFACE AND IN THE WATER MEDIUM

Before we start, we have to specify the list of phenomena we want to model. Let us look at the Fig. 1. Water is lit by direct sun light (E_{sun}) and the sun light scattered by atmosphere and clouds. Light is refracted at water surface, then it is absorbed, scattered and reflected from the bottom. We can speak of the function $B(\vec{x},\vec{d})$ - it specifies the light field in the water medium. It gives the amount of light at the point $\vec{x}$ in the direction $\vec{d}$.

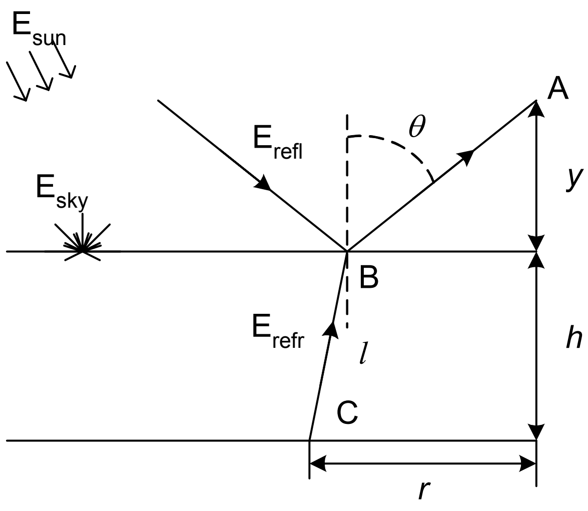

Figure 1. Anatomy of the water surface lighting

Light seen by a viewer can be divided on two components: photons that are not absorbed or scattered on the way from the bottom to the surface and others. The first component "transmits" the bottom image to the viewer. This is not completely correct because the photons scattered at very small angles also contribute to the bottom image, making it looking blurry. This is highly depend on the scattering indicatrix. But in our model we will neglect the influence of this effect, because it is hard to calculate and the correct real-time bottom image blurring too much time consuming procedure.

Formally, this process can be written as follows: the light intensity coming into observer's eye is given by the formula:

$$E = E_{refl}F(\theta) + E_{refr}\left(1 - F(\theta)\right) \qquad (1)$$

Here E_{refl} – light intensity, coming from the above-water environment along the reflection vector., E_{refr} –intensity of light coming from underwater scene, $F(\theta)$ – Fresnel term. E_{refr} – can be written as follows:

$$E_{refr} = \alpha_l E_{bottom} + \left(1 - \alpha_l\right) E_{scatter} \qquad (2)$$

E_{bottom} – intensity of light, reflected from the bottom in the CB (see Fig. 1) direction, $E_{scatter}$ – the value of light field (for the average color bottom) in the point B in the direction CB., α_l – the probability that photon will not be absorbed or scattered on the CB way. This probability is given by Buger's law [9]:

$$\alpha_l = e^{-\varepsilon l},\ \varepsilon = \chi + \sigma \qquad (3)$$

Here χ - absorption coefficient, σ - scattering coefficient.

To calculate E for the water surface we need to know $B(\vec{x}, \vec{d})$ function. The next section will be dedicated to the approximate calculation of this function.

4. THE MODEL OF LIGHT TRANSPORT IN THE WATER MEDIUM

There is no analytical solution of light transport equation in general case, so there are many different simplifications, assumptions and approximations of the general case (for example, [9,11]).

For our case we have chosen so called four-flow model [11] with main assumption of the dependence of light field on depth only. This, surely, leads us to inability to calculate light field for non-averaged color of bottom, and of course this method does not allow us to simulate caustics [2].

The problem is reduced to one dimension case, so instead of function $B(\vec{x}, \vec{d})$ we will use four functions (flows): $F_+(z)$, $F_-(z)$ – flows of diffuse light along and against the Z-axis and $G_+(z)$, $G_-(z)$ – flows of directional light. Of course, we are losing energy angular distribution, but it is not very important thing to lose: it would be hard to store, very hard to calculate and impossible to use by real graphics hardware.

The model gives us the system of differential equations [11]:

$$\begin{cases} \frac{dG_+}{d\tau} = -(k + S_1 + S_2)G_+ \\ \frac{dG_-}{d\tau} = (k + S_1 + S_2)G_- \\ \frac{dF_+}{d\tau} = -(k + S)F_+ + SF_- + S_1G_+ + S_2G_- \\ \frac{dF_-}{d\tau} = (k + S)F_- - SF_+ - S_1G_- - S_2G_+ \end{cases} \qquad (4)$$

Here $\tau = \varepsilon z$, $k = \chi/\varepsilon$, $S \approx \frac{\sigma}{\varepsilon}\left(1 - \frac{1}{2}X_1\right)$, $S_1 \approx \frac{\sigma}{2\varepsilon}\left(1 + X_1\right)$, $S_2 \approx \frac{\sigma}{2\varepsilon}\left(1 - X_1\right)$, X_1 – the first coefficient in the factorization of normalized scattering indicatrix by Legandre polynoms. The coefficients S_1 and S_2 can be interpreted as approximate forward and backward scattering coefficients of the directional light, while S is the backscattering coefficient for the diffuse light.

The analytical solution of (4) is given in [11]:

$$\begin{cases} G_+(\tau) = C_1 e^{-\lambda\tau} \\ G_-(\tau) = C_4 e^{\lambda\tau} \\ F_+(\tau) = C_1 A_1 e^{-\lambda\tau} + C_2 e^{-\lambda_0\tau} + C_3 e^{-\lambda_0\tau} + C_4 A_2 e^{\lambda\tau} \\ F_-(\tau) = C_1 A_2 e^{-\lambda\tau} + C_2 A_3 e^{-\lambda_0\tau} + C_3 A_4 e^{-\lambda_0\tau} + C_4 A_1 e^{\lambda\tau} \end{cases} \qquad (5)$$

where $\lambda = k + S_1 + S_2 = 1$, $\lambda_0 = \sqrt{k(k + 2S)}$,

$$A_1 = \frac{SS_2 + (\lambda + k + S)S_1}{\lambda_0^2 - \lambda^2},\ A_2 = \frac{SS_1 + (-\lambda + k + S)S_2}{\lambda_0^2 - \lambda^2},$$

$$A_3 = \frac{1}{A_4} = \frac{k + 2S - \lambda_0}{k + 2S + \lambda_0}.$$

Let us introduce the following coefficients: R_{ff}, T_{ff} - reflection and transmission coefficients for diffuse light for air-water transition;

R_{dd}, T_{dd} - reflection and transmission coefficients for directional light; R_{df}, T_{df} - reflection and transmission coefficients for directional light turning into diffuse. So, we can write such equations:

$$\begin{aligned} R_{ff} + T_{ff} &= 1 \\ R_{df} + R_{dd} + T_{df} + T_{dd} &= 1 \end{aligned} \qquad (6)$$

The similar coefficients for water-air transition we denote in the same way adding mark "~".

For the bottom we introduce reflection coefficients: $\hat{R}_{dd}, \hat{R}_{df}, \hat{R}_{ff}$ (with the same index meaning as for air-water reflection coefficients).

The boundary condition for (5) can be written in the form:

$$\begin{cases} G_+(0) = T_{dd}E_{sun} + \tilde{R}_{dd}G_-(0) \\ F_+(0) = T_{df}E_{sun} + T_{ff}E_{sky} + \tilde{R}_{df}G_-(0) + \tilde{R}_{ff}F_-(0) \\ G_-(\tau_0) = \hat{R}_{dd}G_+(\tau_0) \\ F_-(\tau_0) = \hat{R}_{df}G_+(\tau_0) + \hat{R}_{ff}F_+(\tau_0) \end{cases} \quad (7)$$

Here $\tau_0 = \varepsilon h$. Using equations (5) and (7) it is possible to calculate coefficients $C_1 - C_4$ of the solutions (6).

$$\begin{cases} C_1 = \dfrac{T_{dd}D^2}{D^2 - \hat{R}_{dd}\tilde{R}_{dd}}E_{sun} \\ C_4 = \dfrac{T_{dd}\hat{R}_{dd}}{D^2 - \hat{R}_{dd}\tilde{R}_{dd}}E_{sun} \end{cases}, here\, D = e^{-\lambda\tau_0} \quad (8)$$

Coefficients C_2, C_3 can be found from the system:

$$\begin{pmatrix} 1-\tilde{R}_{ff}A_2 & 1-\tilde{R}_{ff}A_3 \\ \dfrac{A_2 - \hat{R}_{ff}}{D_0} & D_0\left(A_3 - \hat{R}_{ff}\right) \end{pmatrix}\begin{pmatrix} C_2 \\ C_3 \end{pmatrix} = \\ = \begin{pmatrix} T_{df}E_{sun} + T_{ff}E_{sky} + C_1\left(\tilde{R}_{ff}A_4 - 1\right) + C_4\left(\tilde{R}_{ff}A_1 + \tilde{R}_{df} - 1\right) \\ \dfrac{C_1}{D}\left(\hat{R}_{df} - A_4 + \hat{R}_{ff}A_1\right) + C_4D\left(A_4\hat{R}_{ff} - A_1\right) \end{pmatrix} \quad (9)$$

So now we have kind of $B(\vec{x}, \vec{d})$ function.

For the case of water without waves (just a water mirror) the coefficients with the *dd* index are completely defined by Fresnel term. For the calculation of the coefficients with *df* and *ff* indices we need to average them over the hemisphere. When water surface is crisp this method will give us incorrect results. A model of rough surface should be used for the calculation of the coefficients (e.g. [12]). The simpler idea is to build water surface and calculate average coefficient values using it. For example, let us consider that we have a water surface patch *N* x *N* generated by method [1] or [2]. R_{ff} could be calculated by formula:

$$R_{ff} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\int_0^{2\pi}\int_0^{\pi/2} B(\theta,\varphi)F\left(\alpha(\theta,\varphi,\vec{n}_{ij})\right)\cos\theta d\theta d\varphi}{N^2\int_0^{2\pi}\int_0^{\pi/2} B(\theta,\varphi)\cos\theta d\theta d\varphi} \quad (10)$$

If we assume that $\tilde{R}_{dd} \approx 0$ (the reflection from the bottom is nearly diffuse) we can write E_{bottom} for (2):

$$E_{bottom}(\vec{x}) = \hat{R}_{df}(\vec{x})G_+(\tau_0) + \hat{R}_{ff}(\vec{x})F_+(\tau_0) = \\ \hat{R}(\vec{x})\left(G_+(\tau_0) + F_+(\tau_0)\right) \quad (11)$$

Here we simplified the formula, assuming that $\hat{R}(\vec{x})$ - bottom reflection coefficient, depending on the position at the bottom, is the same for directional and diffuse light.

Let us consider that $G_-(0) \approx 0$, then we get:

$$E_{scatter} = F_-(0) \quad (12)$$

Putting (2), (11) and (12) into (1) we get:

$$E = E_{refl}F(\theta) + \left(1 - F(\theta)\right)\times \\ \times\left[\alpha_l\hat{R}(\vec{x})\left(G_+(\tau_0) + F_+(\tau_0)\right) + \left(1-\alpha_l\right)F_-(0)\right] \quad (13)$$

Here $\vec{x}$ - is the point C actually (see Fig. 1).

5. IMITATING HUMAN VISION

As it was mentioned before we will need to convert some images from the energy-spectral representation to RGB color. Defining a problem, we have to note that we do not want to make RGB picture from some complex image in spectral representation, as it is made, for example in [5][7]. As it will be explained in the next section, our task is much simpler: we just need to make RGB tables of water color for different bottom depth and distance to viewer (variables *r* and *h*). To do this we, obviously, do not need to use methods that subdivide the image on the parts and then process every part separately. It is simpler and more effective to use method dealing with the whole image at once. As it was said earlier, this process consists of two stages: chromatic and luminance adaptation. For the chromatic adaptation we chose simple enough von Kries's method [6]. The von Kries transform is essentially thought of as proportionality law, where individual components of vision are independently scaled. At first we transform spectrum to CIE XYZ coordinate system using $x(\lambda)$, $y(\lambda)$, $z(\lambda)$ base functions:

$$\xi = \int F(\lambda)\xi(\lambda)d\lambda \quad (14)$$

Here ξ denotes the X, Y, Z coordinates. $F(\lambda)$ – is the energy-spectral representation of light intensity. Note that as interval of visible light we will use 380 nm to 780 nm.

Then we convert XYZ color to LMS (the coordinate system of human eye cones) by the transform [6]:

$$\begin{pmatrix} L \\ M \\ S \end{pmatrix} = \begin{pmatrix} 0.400 & 0.707 & -0.080 \\ -0.023 & 1.165 & 0.046 \\ 0.000 & 0.000 & 0.912 \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad (15)$$

The same operation is performed for so called white point value ($F(\lambda)$ of the color that appears to be white on the picture). So we will get L_w, M_w, S_w. To not involve luminance in the calculations, $F(\lambda)$ and $F_{white}(\lambda)$ are scaled to make luminance integral (16) equal 1.

$$I = \int v(\lambda)F(\lambda)d\lambda \quad (16)$$

Here $v(\lambda)$ – spectral effectiveness of human eye cones [13]. Adapted LMS values are calculated by (14):

$$L' = L/L_w;\; M' = M/M_w;\; S' = S/S_w \quad (17)$$

Then those values are transformed to RGB values (by inverting (15) and then applying XYZ to RGB transform [13]). Now we have RGB color and its luminance with high intensity range. To shrink this range we used tone-mapping operator described in [14]:

$$\begin{cases} L_d = m(L_{wa}) L_{da} \left(\dfrac{L_w}{L_{wa}} \right)^{\gamma_w / \gamma_d} \\ L_{df} = \dfrac{L_d \left(1 + L_d / L_{white}^2 \right)}{1 + L_d} \end{cases} \qquad (18)$$

Here L_w – uncompressed luminance, L_{df} – compressed luminance.

All the details about meaning and tweaking of the image-dependant constants $m(L_{wa})$, L_{da}, L_{wa}, γ_w, γ_d, L_{white} can be found in [14]. The only reason why this tone-mapping operator is placed here is to show its complexity and non-linearity. The consequences of this complexity and the information losing operation (14) leads us to obvious thought that (denote Φ as a energy-spectral to RGB representation operator):

$$\Phi\big(F(\lambda) \odot G(\lambda)\big) \neq \Phi\big(F(\lambda)\big) \odot \Phi\big(G(\lambda)\big) \qquad (19)$$

Here symbol $\odot$ denotes any arithmetic operation. Non-equality here may be interpreted as "not even close". That means that if we want to apply operator Φ separately to the different parts of some equation then we need special tricks to make the result look realistic.

To make the compression operator work correctly we need to tweak its parameters. The best way is to use the image from the same scene which appearance is known, e.g. sky image. The sky spectral image can be built using different more or less complex models (like [15][16]), but as we need it for tweaking process only, we can use simpler model described in [9]. The example of using this model can be found in [17]. The example of sky image is given on the Fig. 2.

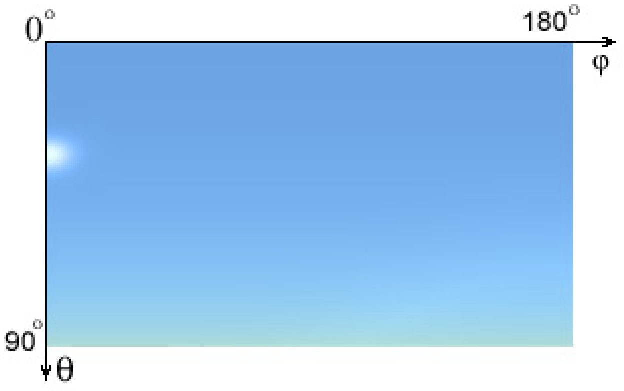

Figure 2. The example of sky picture in (φ,θ) coordinate system (sun position $\theta = 30°$).

The resulting spectral sky image can be used to obtain values E_{sun} and E_{sky} necessary for formula (7).

6. PUTTING THE MODEL AND RENDERING SYSTEM TOGETHER

Now using (13) we can calculate the light intensity coming into viewer's eye A. But nearly all the values in this formula depend on the wave length. So we can speak about implicit parameter λ. So, we can calculate spectrum of the light intensity in every pixel of the image with given sampling frequency. Then we can convert this image into RGB representation using, for example, method [5]. But, in spite of growing CPU performance, this method can not be even referred as interactive.

Traditionally, all calculations in the computer graphics are performed in the RGB color system. Modern graphics accelerators do it as well. To use them we need to calculate as much as possible on the preprocessing stage using spectral representation and convert the results into RGB, saving results as textures.

Let us shortly describe water rendering method we used in [1]. At first the water surface considered flat. We "reflect" the camera from the water surface, render above-water scene into it to get a "reflection texture". The "refraction texture" is obtained by rendering underwater scene into the main camera. The mapping is calculated by projecting water point into corresponding cameras and distorting texture coordinates according the water normals [1]. In pixel shader (the meaning of words "pixel shader" and other graphics hardware terms can be found in [18]) textures are linearly interpolated using Fresnel coefficient (as in (1)).

So we need to write into "refraction texture" the contents of square brackets of (13) instead of the bottom.

$$E_{refr} = \alpha_l \hat{R}(\vec{x})\big(G_+(\tau_0) + F_+(\tau_0)\big) + \big(1 - \alpha_l\big) F_-(0) \qquad (20)$$

Note that in our case $\hat{R}(\vec{x})$ is just RGB texture of bottom, and this means that we need to perform multiplication on it in the pixel shader (we can not do it on the preprocessing stage). On the other hand, calculation of the parts of this formula into textures and then using them in the pixel shader would give completely incorrect results due to non-linearity of operator Φ (see previous section and equation (19) especially).

This means that desired simple transformation of formula (20) into form (21) is impossible:

$$T_{refr} = T_0(r,h) T_{bottom} + T_1(r,h) \qquad (21)$$

Here T denotes RGB textures.

That is why we need to modify this formula to get adequate results. After a number of experiments we choose the empiric formula, that produces realistic looking result of mixing bottom part of formula (21) (left component) and water color (right component).

$$T_{refr} = a(r,h) T_0(r,h) T_{bottom} + (1 - a(r,h)) T_1(r,h) \qquad (22)$$

Here $a(r,h)$ is actually wavelength independent coefficient of interpolation between color of water for the specified bottom color and water color for some average bottom. This coefficient is chosen as:

$$a(r,h) = 1 - \left(1 - \frac{1}{400} \int_{380}^{780} e^{-\varepsilon(\lambda) l(r,h)} d\lambda \right)^{\gamma} \qquad (23)$$

Coefficient γ allows us to control the depth of transition from one component of color to another.

To reduce number of textures in the pixel shader the $a(r,h)$ value is put into alpha-channel of texture T_0. So for the calculation of (22) we need just version 1.0 pixel shader and one texture slot stays free for adding e.g. caustics effect.

Note, that alternative representation of (20) like (24)

$$T_{refl} = T_\alpha(r,h) T_{bottom} T_0(h) + (1 - T_\alpha(r,h)) T_1(h) \qquad (24)$$

(here $T_{\alpha}(r,h)$ contain RGB values of α_l) dramatically reduces quality of the model, because the conversion from energy-spectral representation to RGB would be done earlier and because of (19).

The examples of textures T_0 and T_1 for different water types are given on Fig. 3.

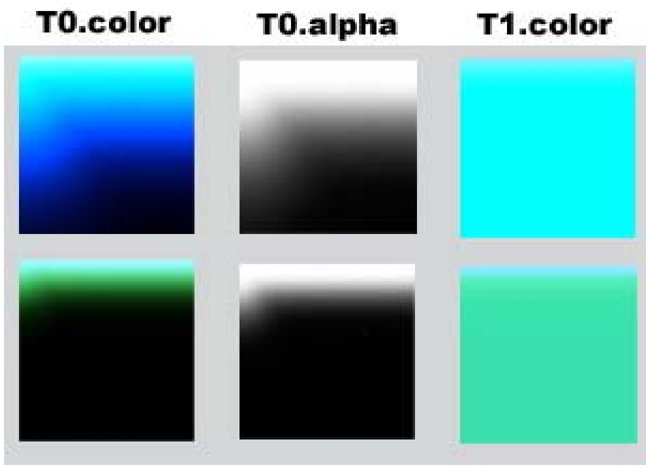

Figure 3. Examples of textures T_0 and T_1 (upper row – Red Sea water [9], lower raw – Arabian Sea [9]).

7. CONCLUSIONS AND FUTURE WORK

This paper has presented a full ready-to-use receipt of adding the refraction of the bottom and water color into the real-time water simulation application based on well-known reflection and refraction modeling principle. We constructed a new physically based model of light transport allowing to calculate water surface energy-spectral image, developed a method of integrating this model into real time applications. This method employs human vision imitation mixed from two well-known methods: (chromatic adaptation and tone-mapping) in order to render convincing images of water surface (compare Fig. 4 left and right pictures, see Fig. 5). The method shows about 130 FPS (P4 2.4GHz, ATI Radeon 9800 Pro) that makes us think that the water simulation with the water surface color imitation still can be a part of virtual reality application.

Water surface modeling states a number of challenging problems that are close to the topic of this paper. One of them is integrating of cheap real-time caustics into the model. The next one is to imitate different types and sizes of crisp on the water surface.

8. REFERENCES

[1] Vladimir Belyaev, Real-time simulation of water surface. GraphiCon-2003, Conference Proceedings, OOO “MAX Press”, 2003, pp. 131-138

[2] Lasse Jensen. “Deep-Water Animation and Rendering”. Gamasutra, September 26, 2001. http://www.gamasutra.com/gdce/jensen/jensen_01.htm

[3] Premoze. S, and Ashikhmin, M. “Rendering Natural Waters”, Computer Graphics Forum, v. 20, no. 4 (2001), pp. 189-200

[4] Kei Iwasaki, Yoshinori Dobashi and Tomoyuki Nishita, A Volume Rendering Approach for Sea Surfaces Taking into Account Second Order Scattering Using Scattering Maps, Volume Graphics (2003). Proc. Eurographics’2003, 2003, pp. 129-136.

[5] Sumanta N. Pattanaik, James A. Ferwerda, Mark D. Fairchild, Donald P. Greenberg, A Multiscale Model of Adaptation and Spatial Vision for Realistic Image Display. Proceedings of the 25th annual conference on Computer graphics and interactive techniques, 1998, pp. 287 – 298.

[6] G. M. Johnson, Computer Synthesis of Spectroradiometric Images for Color Imaging Systems Analysis. http://www.cis.rit.edu/research/mcsl/research/gmjthes.pdf

[7] Garrett M. Johnson and Mark D. Fairchild, Full-Spectral Color Calculations in Realistic Image Synthesis. IEEE Computer Graphics, July/August 1999 (Vol. 19, No. 4), pp. 47-53.

[8] Henrik Wann Jensen, Simon Premoze, Peter Shirley, William B. Thompson, James A. Ferwerda, Night rendering. www.cs.utah.edu/vissim/papers/night/nightTech.pdf

[9] Ocean optics. Vol. 1. Physical ocean optics. SA USSR. Shirshov University of oceanology, 1983.

[10] John Erwin Tumblin, Three Methods of Detail-Preserving Contrast Reduction for Displayed Images. PhD Dissertation, 1999. http://www.cs.northwestern.edu/~jet/dissert.html

[11] Akira Ishimaru, Wave Propagation and Scattering in Random Media, Volume 1, Academic Press, New York, San Francisco, London, 1978.

[12] Oren, Michael, and Shree K. Nayar, “Generalization of Lambert’s Reflectance Model”, Computer Graphics (*SIGGRAPH 94 Proceedings*), pp. 239-246, July, 1994.

[13] William K. Pratt, Digital Image Processing, John Wiley and sons, New York, 1978.

[14] Boris Barladian, Robust Parameter Estimation for Tone Mapping Operator. GraphiCon-2003, Conference Proceedings, OOO “MAX Press”, 2003, pp. 106-111.

[15] Jaroslav Sloup, A Survey of the Modelling and Rendering of the Earth's Atmosphere. Proceedings of the 18th spring conference on Computer graphics, pp. 141-150, 2002.

[16] Yoshinori Dobashi, Kazufumi Kaneda, Hideo Yamashita, Tomoyuki Nishita, Method for Calculation of Sky Light Luminance Aiming at an Interactive Architectural Design.

[17] Belyaev V.S. Water surface color modeling for real-time applications. Proceedings of science and practical conference. St. Peterburg, 2003, pp. 319-326.

[18] James Leiterman, Learn Vertex and Pixel Shader Programming With DirectX 9, Wordware Publishing, 2004.

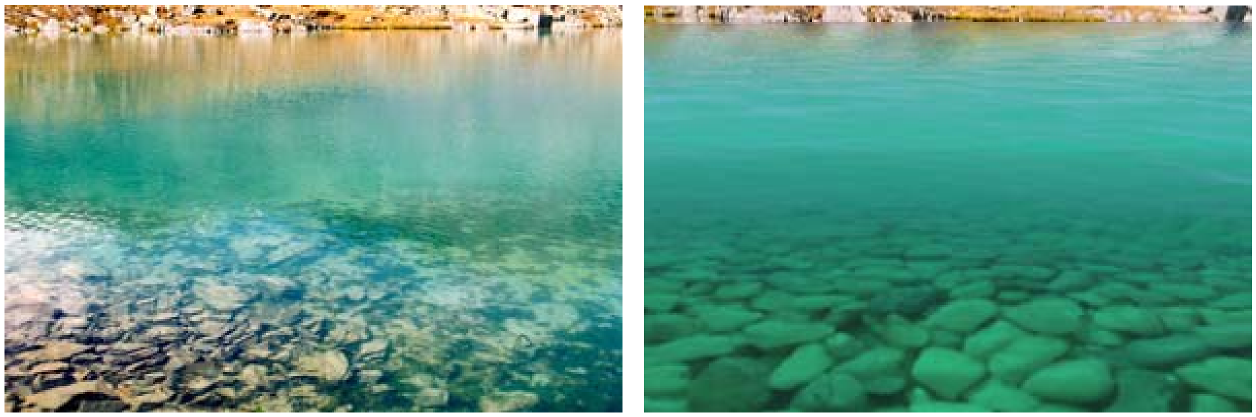

Figure 4. One of the lakes from “seven lakes” place near mountain Beluha, Altai, Russia. Photo on the left, generated picture on the right

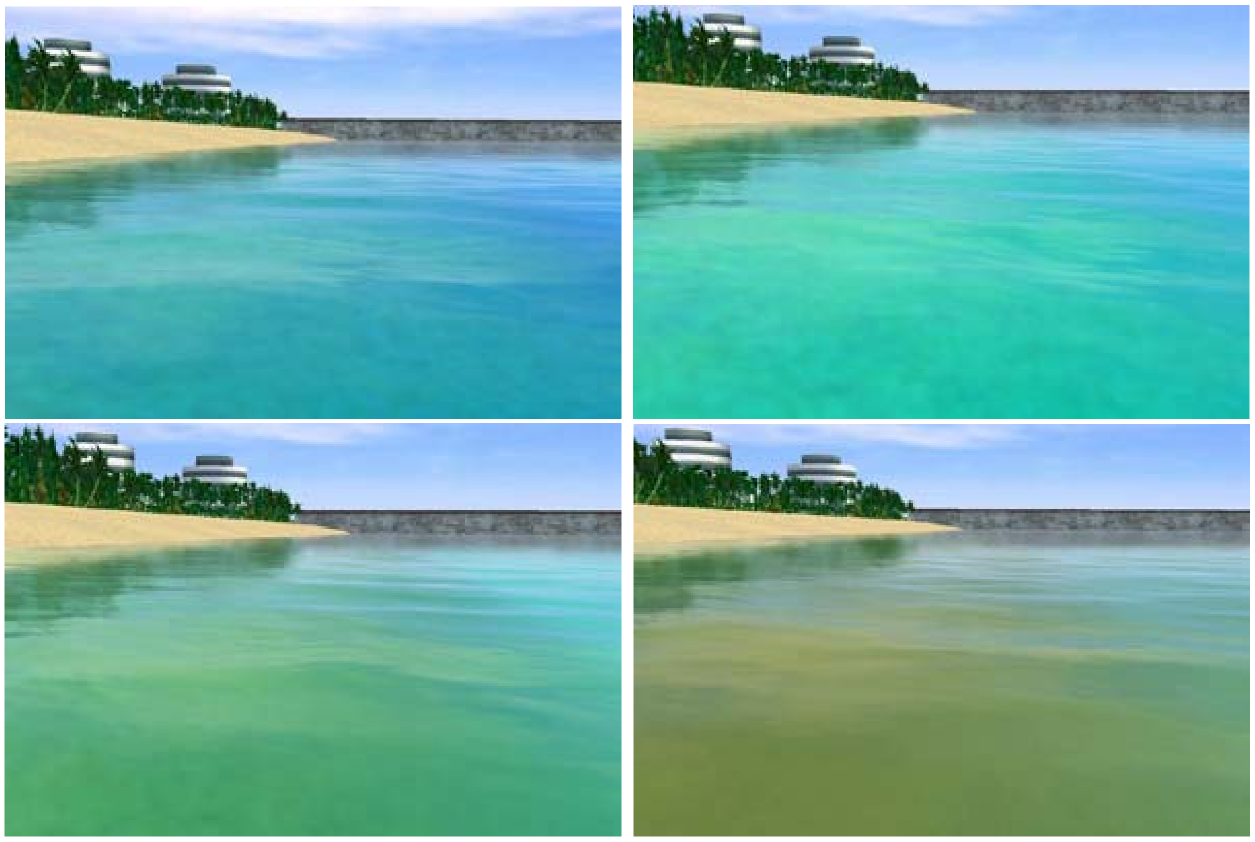

Figure 5. Different water types: clear tropical water (first row), muddy coastal water (second row).

About the author

Vladimir Belyaev is a Ph.D. student at Applied Math. Department,
St. Petersburg State Polytechnical University, St. Petersburg, Russia
His contact email is vladimir@d-inter.ru.