Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 846964, 8 pages
http://dx.doi.org/10.1155/2014/846964

Research Article

Subsurface Scattering-Based Object Rendering Techniques for Real-Time Smartphone Games

Won-Sun Lee, Seung-Do Kim, and Seongah Chin

Division of Multimedia Engineering, Xicom Lab, Sungkyul University, Anyang 430 742, Republic of Korea

Correspondence should be addressed to Seongah Chin; solideochin@gmail.com

Received 4 July 2014; Accepted 22 July 2014; Published 17 August 2014

Academic Editor: Jong-Hyuk Park

Subsurface scattering that simulates the path of a light through the material in a scene is one of the advanced rendering techniques in the field of computer graphics society. Since it takes a number of long operations, it cannot be easily implemented in real-time smartphone games. In this paper, we propose a subsurface scattering-based object rendering technique that is optimized for smartphone games. We employ our subsurface scattering method that is utilized for a real-time smartphone game. And an example game is designed to validate how the proposed method can be operated seamlessly in real time. Finally, we show the comparison results between bidirectional reflectance distribution function, bidirectional scattering distribution function, and our proposed subsurface scattering method on a smartphone game.

1. Introduction

Computer graphics on PC and console games that employ BRDF (bidirectional reflectance distribution function) rendering and advanced rendering techniques have made a considerable progress [1–3]. However, BSSRDF (bidirectional surface scattering reflectance distribution function) [4, 5] taking into account subsurface scattering does not seem to operate in smart phone games because it requires a considerable amount of computing. A technique for reducing power dissipation also was proposed when the smartphone is in a static state [6]. Recently, a new subsurface scattering model has been proposed called beam diffusion. The new model is a hybrid method-photon beam diffusion that combines Monte Carlo integration with the diffusion approximation. The rendering results are almost identical to traditional diffusion methods while it seems somewhat faster and stable. Even though this model seems accurate, the methods require minutes on a multicore PC as well [7]. Some image space subsurface scattering techniques in PC environments were studied, in which they required preprocessing to reduce running time. Also, natural scenes that seem to be highly complex could be rendered using a hardware accelerated fractal based method [8]. In the past, because of the limitations of the smartphone hardware, a few efforts were made to overcome the limitation of the graphics quality. Nowadays, the computational power of smartphones has continually advanced, and now, we can expect to solve the previous shortcomings of smartphones. However, subsurface rendering techniques that are optimized for smartphones have not been offered yet. Thus far, we have not found an approach to develop subsurface rendering techniques that can be realized in a real-time smartphone game as shown in Table 1.

In this paper, we present a real-time subsurface rendering technique without any preprocessing that is optimized for smartphone games that can be operated seamlessly. This contributes to enhancement of rendering in smartphones in which high quality of the graphics tends to draw users' interests. We focus on synthesizing a game character that has a skin texture associated with optical parameters for the proposed subsurface scattering for a smartphone. For this, Unity (http://unity3d.com/) shader is used for executing a subsurface scattering algorithm, and vertex and fragment shaders are used for optimizing the proposed method. The preliminary version of research was reported in UCES 2013 workshop. We have extended our research by developing our own subsurface color model that is crucial for real-time subsurface effect in games. Performance evaluation and user study have been carried out as well.

Table 1: Related studies.

Mobile rendering	Real-time Rendering	Subsurface rendering
Clockcycle [9]	Jensen et al. [4], Yang and Wang [10]	
	Yan et al. [11]	Doner and Wann Jensen [5]
Zhang and Shigeo [12]		
None		

The rest of this paper is organized as follows. Section 2 describes BRDF and BSSRDF rendering technologies related to this research. The critical flow of the proposed algorithm is described in Section 3. In Section 4, we explain the game design and development. Section 5 presents the conclusion.

2. Background and Literature Review

2.1. BRDF. The modeling is achieved through the light reflected from the surface by ambience processing, and the rendering equation is used for expressing the surface properties of the materials. The BRDF model, which is slightly more complex than the existing lighting equation, shows an improved result and is becoming the base for the current real-time rendering techniques [3]. There have been previous studies that enhance skin rendering using BRDF [13, 14]. In general, it is a function that shows how the light reflects on a surface, and the reflectance of light changes according to the position of the light and the point that is proportion to the direction of the surface and the tangent value. To be accurate, it shows the ratio between the outgoing radiance and the differential incoming irradiance.

2.2. BSSRDF. BSSRDF added to the BRDF, according to which the incidence and the reflection point express the same surface reflection, is an eight-dimensional function that describes the transport phenomena of the light between two points due to the scattering of the light at the subsurface [15]. Since the aforementioned BRDF is created on the basis of the assumption that the same light becomes the input and the output at the same point, it becomes the approximate value of BSSRDF. When BSSRDF is applied, not only does it express translucent objects, but it also enables a more realistic expression of the human skin [15, 16]. The human skin is a good object to apply the BSSRDF, which can even calculate the scattering phenomena, because it is a multiple-layer translucent object that undergoes a combination of transmission and reflection.

The BSSRDF utilizes the absorption coefficient, the reduced scattering coefficient, and the Fresnel index to express the effect caused by scattering and absorption within the material; the BSSRDF equation can be organized using the absorption coefficient, the reduced scattering coefficient, and the Fresnel index, which are parameters measured according to the materials. On the basis of this theory, many translucent objects have various reflectance values, and hence, their properties have to be expressed cautiously through the ratio of the diffusion approximation and the single scattering.

3. Rendering for Smartphones

3.1. BRDF Shader. Unity is widely known as a multiplatform engine, and a high-completion game can be developed by dealing with both the CPU and the GPU. GPU has to use ShaderLab, a script language, since it does not have a structure, such as the virtual machine of the CPU. In addition, forward pipeline, which calculates the lighting per pixel, is used for producing a game.

Among the Unity shader programs, the vertex fragment shader program was used for implementing the BRDF model. The important point here is that multiple shader languages are supported by ShaderLab, and since it supports multiple platforms, computer graphics have to be used for consistent processing, even on a mobile device.

We produced a special custom ramp that supports a customized light illumination model rather than default light models for the resolution of light treatment, and the shader was realized by using the Ward BRDF model [3] equation for the lighting ramp. Equation (1) is expressed as the tangent space; α_x and α_y denote the roughness of the surface; degrees $\theta_i, \theta_o, \phi_i$, and ϕ_o define the input and output of the light of the BRDF model:

$$f_{r(i,o)} = \frac{\rho_s}{4\pi\alpha_x\alpha_y\sqrt{\cos\theta_i\cos\theta_o}} \times e^{-\tan^2\theta_h((\cos^2\phi_i/\alpha_x^2)+(\sin^2\phi_o/\alpha_y^2))}, \quad (1)$$

where i is an incident direction of the light and o indicates outgoing direction. ρ_s controls the magnitude of the lobe, and θ_h is an angle between a normal and a half vector.

The formula is transformed by giving the scale value through ShaderLab and Ward BRDF model that can control the diffuse, ambient, and the specular light, and the scale value at the slider interface using the property phrase at the ShaderLab was produced. The ultimate data type of $f_r(i,o)$ was given as a fixed type for the optimization at the mobile device.

3.2. Scaled BSSRDF Shader. Because BSSRDF [4] takes minutes on a multicore PC, we modify it by scaling that can be running in real-time. The rendering equation has to be accurately implemented at the ShaderLab for the optical parameters to accurately express the peculiar properties of the objects. If the surface shader program is used, it is possible to produce a shader effectively and easily; however, BSSRDF was implemented using the vertex and fragment shader program and it is a rendering model that used Jensen's bipolar light source method. The diffusion term is achieved through formula (2):

$$S_d(x_i, \vec{w}_i; x_o, \vec{w}_o) = \frac{1}{\pi} F_t(\eta, \vec{w}_i) R_d(\|x_i - x_o\|) F_t(\eta, \vec{w}_o). \quad (2)$$

Formula (3) is the formula of $R_d(r)$ term of Jensen's BSSRDF model [4]:

$$R_d(r) = \frac{\alpha'}{4\pi}\left[(\sigma_{tr}d_r + 1)\frac{e^{-\sigma_{tr}d_r}}{\sigma_t' d_r^3} + z_v(\sigma_{tr}d_v + 1)\frac{e^{-\sigma_{tr}d_v}}{\sigma_t' d_v^3}\right], \tag{3}$$

where $\alpha' = \sigma_s'/\sigma_t'$, $\sigma_t' = \sigma_s' + \sigma_a$, $\sigma_{tr} = \sqrt{3\sigma_a\sigma_t'}$,

$$d_r(r) = \sqrt{z_r^2 + r^2}, \quad z_r = \frac{1}{\sigma_t'},$$

$$d_v(r) = \sqrt{z_v^2 + r^2}, \quad z_v = z_r + 4AD, \quad A = \frac{1 + F_{dr}}{1 - F_{dr}}, \tag{4}$$

$$F_{dr} = \frac{-1.44}{\eta^2} + \frac{0.71}{\eta} + 0.668 + 0.0636\eta, \quad D = \frac{1}{3\sigma_t'}.$$

The parameters used in formula (3) are all defined by σ_a and σ_s', and the $R_d(r)$ term's value is calculated using the light and camera's property provided by the update function of the Unity script. Original BSSRDF model takes running times about seven minutes on a multicore PC. Hence, instead of using the distance between the points of the object, the distance calculated on the basis of the location of each point on the object and the light source of the approximated model was used as parameter r. Furthermore, the distance was controlled by multiplying the ratio of the scale value in order to consider the distribution of the light's transmission and the size of the object in Unity.

3.3. Our Subsurface Scattering. The scaled BSSRDF mentioned in Section 3.2 seemed to be overly smooth that could be only suitable for rendering like a candle. It does not seem ideal for rendering human skin. Some advanced BSSRDF techniques are also proposed such as the hierarchical BSSRDF [17] that provides a rapid hierarchical rendering technique for translucent materials. However, we do not recognize that this BSSRDF can be suitable for mobile applications.

Authors propose a novel method that takes into consideration the drawbacks such as being overly smooth and taking longer running times. The proposed method needs to recover those limitations by employing our subsurface scattering color model [18] that takes optical parameters and return a set of final subsurface color values. We have a subsurface scattering color that can be computed in (5) that takes optical parameters

$$f_d = (1 - F_{dr,t})\left(\sigma_s' P(g)\left\{d - \sigma_a d^2 + \frac{1}{3}\sigma_a^2 d^3\right\} + F_{dr,b}\left(1 - \sigma_a d - \sigma_s' P(g)\, d\right)^2\right), \tag{5}$$

where σ_a and σ_s' are the absorption and reduced scattering coefficients, respectively. $P(g)$ is the integration of phase function for backward. d is the depth. $F_{dr,b}$ is the diffuse Fresnel reflectance at the bottom boundary. $F_{dr,t}$ represents the diffuse Fresnel reflectance at the top boundary.

Our final rendering color is defined as (6)

$$f_{r(i,o)} = f_d\left(\rho_d \cos\theta + \frac{20\rho_{ts}}{d^{2.5}}\right) + f_s L \cos\theta \times \left[\frac{\rho_s}{4\pi\alpha_x\alpha_y\sqrt{\cos\theta_i \cos\theta_o}} e^{-\tan^2\theta_h(\cos^2\phi_i/\alpha_x^2 + \sin^2\phi_o/\alpha_y^2)}\right], \tag{6}$$

where f_d is the subsurface surface color. f_s is ρ_d (diffuse coefficient), and ρ_s (specular coefficient) and ρ_{ts} (translucent coefficient) can be optimized using an experiment. L is the light intensity. d is the distance from the light source. (θ_i, ϕ_i) is the incident light vector, and (θ_o, ϕ_o) is the reflected light vector. f_d is obtained using optical parameters that is defined as formula (5). ρ_d (diffuse coefficient), ρ_s (specular coefficient), and ρ_{ts} (translucent coefficient) can be derived from the experiments meaning that the optimal parameters are determined as viewing rendering results. For the skin, we determine $\rho_d = 1$, $\rho_s = 0.2$, and $\rho_{ts} = 0.01$. The first term means diffusive reflectance and second term is for subsurface diffuse light that is used for subsurface effect for translucent materials. The last term determines the specular light.

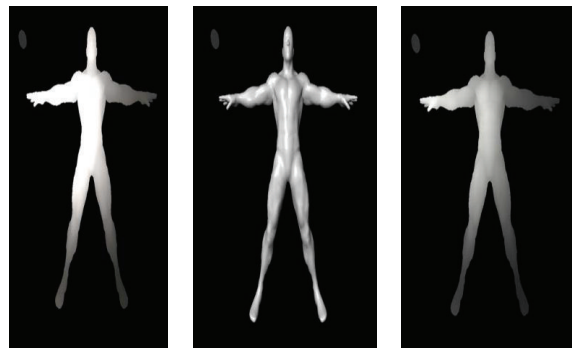
4. Experiments

Unique optical features of the material that we wish to render have to be considered when rendering a material. We implemented three rendering methods described in Section 3, using vertex and fragment program under Unity 4.3 ShaderLab. To reduce running times in GPU for mobile environments, we let some computations that are not necessary to run in GPU operate in CPU.

4.1. Rendering Results. Prior to comparing the rendering results, we carried out rendering of translucency that took into account optical parameters of a white coated material because the results of translucency can be easily observed. In Figure 1, we can observe the translucent effects of BSSRDF (a) and our method (c) rendering results. The rendering result for BRDF (b) does not show translucency but glitters because BRDF cannot consider subsurface scattering. In the first case, to express the absorption of the light and the scattering, the parameter of a white coating was used to express the feeling of a translucent material; the color of the skin was compared by using texture. A considerably softer and more translucent feeling at the fingertips and both feet can be seen in the case of BSSRDF and our model. However, the rendering result of BSSRDF seems rather translucent than our subsurface scattering (OSS).

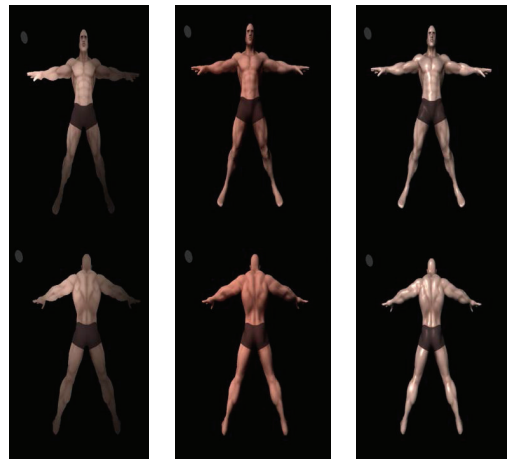
Levenberg-Marquardt method [19] was used to acquire optical parameters of the white coated material. In this simulation for Figure 1, we used $\sigma_s' = 1.94$ for red, $\sigma_s' = 1.94$ for green, $\sigma_s' = 1.874$ for blue, $\sigma_a = 0.000905$ for red, $\sigma_a = 0.001083$ for green, $\sigma_a = 0.002038$ for blue, and $\eta = 1.0936$ for Fresnel.

Then, after using skin 1's $\sigma_a = (0.032, 0.17, 0.48)$ and $\sigma_s' = (0.74, 0.88, 1.01)$ values to find $\alpha = \sigma_s'/(\sigma_s' + \sigma_a)$, the diffuse value was obtained by substituting the α value, acquired



(a) (b) (c)

Figure 1: The rendering results of a white translucent material. BSSRDF (a), BRDF (b), and OSS (c) are shown.



(a) (b) (c)

Figure 2: The rendering results of BSSRDF (a), our subsurface scattering (OSS) method (b), and BRDF (c).

using the parameters of skin instead of α, into the BRDF approximation. These optical parameters were measured by using our customized optical imaging equipment [20].

The comparison subject BRDF's texture was obtained by multiplying the subsurface scattering color value by the texture RGB value. Since the value of $R_d(r)$ determines the object's color in the BSSRDF model, the gray level's texture was used for comparing the three models under the same conditions.

Figure 2 shows three rendering results. The light source is located at upper (a). The screen shots in (c) represent the rendering result of BRDF that cannot convey scattering effects at all. However, the human skin should have been scattered. This seems like a plastic that mostly glitters. The BSSRDF rendering results shown in (a) seem more translucent than they should be.

The areas of around chest and head are much blurrier by subsurface scattering effect than they are. Comparing our method (shown in Figure 2(b)) with other two methods, our subsurface scattering (OSS) method seems reasonable because our rendering results show more relevant translucency than BSSRDF (a). In short, we can observe that the proposed rendering results seem more natural than others. For instance, more uniform shading distributions in the upper body and legs are observed in our rendering results as well.

Moreover, Figure 3 shows the comparison result of the BRDF, the BSSRDF, and OSS, which controlled the coefficients to be used in a game.

The rendering results from BSSRDF and OSS are smoother than one from BRDF. However, the BSSRDF is quite blurred and produces subsurface scattering that is not ideal to render human skin. OSS seems to produce somewhat medium effect between BSSRDF and BRDF since the algorithm is defined by conveying subsurface scattering parameters of the skins and BRDF effect as well. The proposed rendering results show acceptable rendering results superior to other methods in the screen shots of the smartphone game that we designed.

Also, we carried out a user study in order to evaluate reputations from users about three rendering results. The 50 subjects whose age ranges from 20 to 30 participated in our questionnaire. They are currently working for IT (information technology) related occupations or having experiences

Figure 3: BRDF (a), BSSRDF (b), and OSS (c) in each picture applied to the smartphone game.

with playing mobile games. The scores range 0–5 Likert scale points representing the higher scores indicate more reliable judgment than the lower scores. They were asked to answer two survey questions while viewing rendering results that were implemented in the smartphone game. For the first question, we asked how similar do you feel that it is like real skin.

The result scores in Figure 4 show BRDF (3.06), modified BSSRDF (2.91), and OSS (3.38). Also we asked them if they recognized that skin is translucent. Then, we again examined the question only if their responses were "yes."

More interesting analysis on the first question was found when we asked only a participant who recognized that skin is translucent. The score BRDF decreases by 2.69 from 3.06 whereas modified BSSRDF score becomes 3.21 from 2.91 and OSS score increases by 3.61 from 3.38 as shown in Figure 5.

This fact seems to support a prior study concerning comparison between experienced and inexperienced game player [21]. In short, our proposed method shows acceptable rendering results.

4.2. Game Development. In order to realize our proposed method into a real-time game, we have developed a game named "Xicom runner" using Unity. The game is composed of a main screen and a play screen, as shown in Figure 6. In the main screen, there is an option button to select a rendering method. Further, there is a start button to start the game. The score increases in the running process of the character, and the score is obtained when the character continues to run. If the character collides with obstacles, then the game is over. When the replay button that appears afterwards is clicked, the game returns to the play screen. The game considered in this study is easy and simple, since it was produced to test and display real-time rendering. Further, the position and the view of the camera were not fixed but were set to move flexibly in order to follow the flow of the game. The map was composed with sets of various objects to test the performance and the limits of the rendering.

4.3. Optimization Technique. For seamless real-time rendering of the shader, not only the optimization of the shader code but also the optimization of the game code is required.

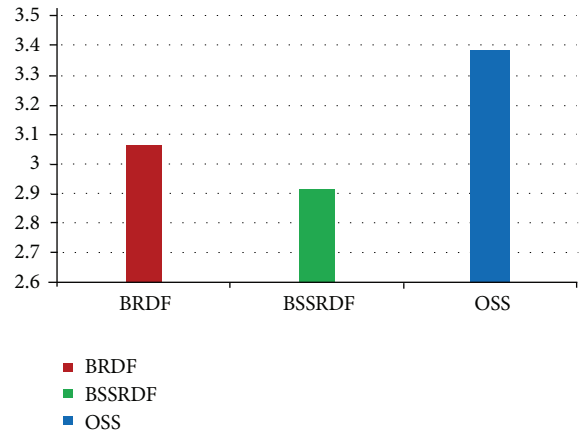


Figure 4: Rendering result scores on BRDF (left), BSSRDF (middle), and OSS (right).

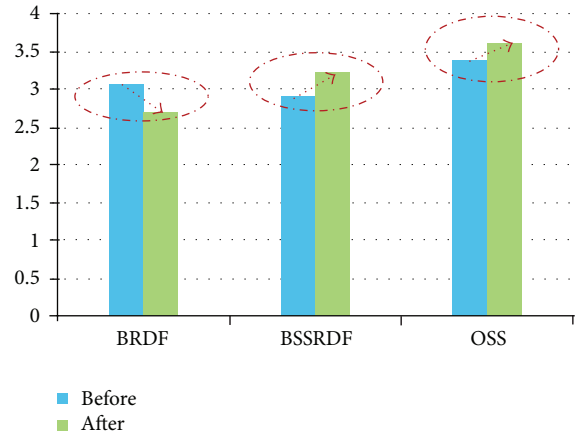


Figure 5: Score changes on BRDF (left), BSSRDF (middle), and OSS (right).

We developed the relevant code by considering the development engine Unity's properties and the following details for the optimization of the game code.

(a) (b)

Figure 6: Main screen (a) and play screen (b).

In order to avoid a bias that could happen to real-time game play among three rendering methods, we apply the following optimization techniques to all three rendering methods.

First, minimize the number of unnecessary variables and object creation. In other words, where variables and objects with the same attributes and functions are repeatedly created, they should be referred to instead of being constructed dynamically by declaring them as a member or global beforehand. By doing this, the process of initializing and deleting these attributes and functions every time is eliminated, and the burden of operations is reduced because it only has to refer to them.

Second, preprocess all the resources, such as texture, that are used in the game. Setting a resource in the memory requires a considerable number of operations and is time consuming. Therefore, if dynamically, each frame loads the resource, it will burden the system. To prevent this, the resource should be set in the memory beforehand and referred to when needed by preprocessing the resource loading.

Third, the GUI skin that is produced by Unity should not be used and should be self-produced and used. Because the GUI skin produced by Unity takes up more memory than expected, to minimize this, we self-produced and used the texture and minimized unnecessary memory use and operations.

Lastly, minimize the operations within the OnGUI() function. In Unity, to output text or textures on the screen, the functions related to the GUI in the OnGUI() function have to be used. However, caution must be taken when using the OnGUI() function, because each frame is called twice. Consequently, when control statements or expressions are inserted into this function, it becomes a problem as each calculation is run twice. Therefore, it is advantageous to exclude as many expressions and control statements as possible in the OnGUI() function.

4.4. Performance Test. In general, even in PC or console games, an advanced shader is applied only to the object that accounts for the largest proportion in the screen. This is because the application of an advanced shader to objects that have low importance and are not noticeable is a waste of memory and increases the number of unnecessary operations.

Table 2: GPU running times.

	GPU USAGE (ms)		
Sampled frame (30f)	BSSRDF	BRDF	Our method
1	0.179	0.146	0.111
2	0.153	0.148	0.109
3	0.174	0.133	0.113
4	0.181	0.137	0.114
5	0.175	0.144	0.112
6	0.165	0.155	0.109
7	0.174	0.14	0.115
8	0.182	0.159	0.109
9	0.184	0.115	0.133
10	0.217	0.158	0.122
Average	0.1784	0.1435	0.1147

We applied three shading codes to only the character model in the game we made that accounts for the biggest proportion in the screen. Other objects in the game such as tables, chairs, walls, and floors used the default illumination model in Unity to reduce GPU running times.

To verify the proposed method, we realized rendering algorithms under Unity 4.3 ShaderLab. We analyzed algorithm performance with respect to GPU running times and memory usage. To capture sample frames, we made the character that was implemented with shaders including BRDF, BSSRDF, and our method be animated. The 10 peak frames every 30 frames were captured.

For GPU performance test, we analyzed running times from 10 sampled frames using Unity profiler with GPU (*ms*). The average running times of GPU appear with 0.1784 for BSSRDF, 0.1435 for BRDF, and 0.1147 for our method, respectively.

In short, we found that modified BSSRDF and BRDF took longer operations than our proposed method as shown in Table 2. Even if three methods can quite work well in real time, the rendering results of our method are more outstanding than BSSRDF and BRDF. Moreover, our proposed method is superior to the other two methods with respect to GPU running time.

Table 3: Memory usage.

	Memory (kb)		
Memory	BSSRDF	BRDF	Our method
Shader memory	28.3	23.4	24.9
Skinned mesh renderer	0.6	0.6	0.6
Shader object	0.177	0.177	0.177
Scene memory	116.2	116	116.2

In addition, we performed memory usage tested and shown in Table 3 because the memory usage can affect somewhat game performance. If the more memory is occupied, then more access time is required. In particular, mobile games need to take into account small amount of memory if possible.

For memory test, we extracted some criteria such as shader memory, skinned mesh renderer, shader object, and scene memory. Shader memory represents the memory that is used for performing of BSSRDF, BRDF, and our method. Scene memory is composed of lots of components. Among them, we select three criteria. In shader memory, BRDF used the smallest portion of memory 23.4 (kb). Our proposed method used 24.9 (kb) and BSSRDF took 28.3 (kb). Other criteria show not a big difference. Our proposed method is acceptable in memory usage as well.

5. Conclusion

In this paper, we introduced the BRDF and modified BSSRDF and our proposed method for effective rendering taking into consideration the properties of object materials and optimized them for a smartphone environment. We applied these shaders to a smartphone game. As a result, we found that our proposed subsurface scattering method seemed slightly better than modified BSSRDF and BRDF with respect to rendering appearance and GPU running times. Also our method is superior to BSSRDF in the shader memory usage. Further, it was verified that the game could be played in real-time when the proposed shaders were used.

It was experimentally verified that our proposed method was more appropriate for rendering a character's translucent skin. The value of this research lies in the application of subsurface scattering, which could not be realized real-time on a smartphone thus far, by reproducing it in real time. In the proposed approach, we rather focus on rendering a character model. As future work, more materials need to be simulated simultaneously to achieve more enhanced quality of a scene for smart phone games that definitely comes with tradeoff between running time and visual appearance. We might estimate a fact that running time of a game is contingent on input factors such as the number of object, the number of polygon, the number of light, and the number of shader component. Precise analysis would be interesting. However, we have to determine optimal properties of the input factors for a real-time game.

Also, skin is used in many applications. A thresholding technique is critical when extracting human skin region that can be affected by various lights [22].

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the Korea Foundation for the Advancement of Science and Creativity and funded by the Korean Government. And this research was also partially funded by Korea Evaluation Institute of Industrial Technology, KEIT (CiMR no. 10043453).

References

[1] S. McAuley, S. Hill, N. Hoffman et al., "Practical physically-based shading in film and game production," in *Proceeding of the SIGGRAPH 2012, ACM SIGGRAPH Courses Article*, 2012.

[2] M. Kurt and D. Edwards, "A survey of BRDF models for computer graphics," *ACM SIGGRAPH Computer Graphics—Building Bridges—Science, the Arts & Technology*, vol. 43, no. 2, article 4, 2009.

[3] B. Walter, "Notes on the ward BRDF," Tech. Rep. PCG-05-06, 2005.

[4] H. W. Jensen, S. R. Marschner, M. Levoy, and P. Hanrahan, "A practical model for subsurface light transport," in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*, pp. 511–518, 2001.

[5] C. Doner and H. Wann Jensen, "A spectral BSSRDF for shading human skin," in *Proceedings of the 17th Eurographics Conference on Rendering Techniques*, pp. 409–417, 2006.

[6] H. Cho and M. Choi, "Personal mobile album/diary application development," *Journal of Convergence*, vol. 5, no. 1, pp. 32–37, 2014.

[7] R. Habel, P. H. Christensen, and W. Jarosz, "Photon beam diffusion: a hybrid Monte Carlo method for subsurface scattering," *Computer Graphics Forum*, vol. 32, no. 4, pp. 27–37, 2013.

[8] J. Divya Udayan, H. S. Kim, J. Lee, and J.-I. Kim, "Fractal based method on hardware acceleration for natural environments," *Journal of Convergence*, vol. 4, no. 3, pp. 6–12, 2013.

[9] Real-time BRDF on IOS, 2013, http://clockcycle.net/real-time-brdf-ios.

[10] M. Yang and K. Wang, "Real-time rendering for multi-layered translucent materials such as human skin under dynamic environment lighting," *Applied Mechanics and Materials*, vol. 274, pp. 423–426, 2013.

[11] C. Yan, T. Yue, J. Liu, and C. Zhao, "A novel method for dynamic surface modeling and real-time rendering," in *Proceedings of the 21st International Conference on Geoinformatics (GEOINFORMATICS '13)*, pp. 1–5, IEEE, Kaifeng, China, June 2013.

[12] Z. Zhang and M. Shigeo, "Real-time hair simulation on mobile device," in *Proceedings of the Motion on Games*, ACM, 2013.

[13] R. Stephen, H. Marschner Stephen, P. F. Eric, E. Kenneth, and P. Donald, "Image-based brdf measurement including human skin," in *Proceedings of 10th Eurographics Workshop on Rendering*, pp. 139–152, 1999.

[14] S. R. Marschner, S. H. Westin, E. P. F. Lafortune, K. E. Torrance, and D. P. Greenberg, "Reflectance measurement of human skin," Tech. Rep., Cornell University, 1999.

[15] H. Nakai, Y. Manabe, and S. Inokuchi, "Simulation and analysis of spectral distributions of human skin," in *Proceedings of 14th International Conference on Pattern Recognition*, vol. 2, pp. 1065–1067, 1998.

[16] N. Tsumura, N. Ojima, K. Sato et al., "Image-based skin color and texture analysis/synthesis by extracting hemoglobin and melanin information in the skin," in *Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Technicques (SIGGRAPH '03)*, pp. 770–779, July 2003.

[17] H. W. Jensen and J. Buhler, "A rapid hierarchical rendering technique for translucent materials," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '02)*, pp. 576–581, San Antonio, Tex, USA, July 2002.

[18] T. Choi and S. Chin, "Wound recovery synthesis on a 3d face using subsurface scattering with multi-layered skin features," in *Proceedings of the 26th International Conference on Computer Animation and Social Agents (CASA '13)*, Istanbul, Turkey, May 2013.

[19] D. M. Bates and D. G. Watts, *Nonlinear regression analysis and Its applications*, John Wiley & Sons, New York, 1988.

[20] T. Choi, S. Lee, and S. Chin, "A method of combining the spectrophotometer and optical imaging equipment to extract optical parameters for material rendering," *Journal of Sensors*. In press.

[21] G. Christou, "A comparison between experienced and inexperienced video game players," *Human-Centric Computing and Information Sciences*, vol. 3, article 15, 2013.

[22] Z. H. Al-Tairi, R. W. Rahmat, M. I. Saripan, and P. S. Sulaiman, "Skin segmentation using YUV and RGB color spaces," *Journal of Information Processing Systems*, vol. 10, no. 2, pp. 283–299, 2014.