Research Journal of Applied Sciences, Engineering and Technology 6(9): 1653-1657, 2013
ISSN: 2040-7459; e-ISSN: 2040-7467

Submitted: January 07, 2013 Accepted: January 31, 2013 Published: July 15, 2013

Fast Rendering of Realistic Virtual Character in Game Scene

[1, 2]Mengzhao Yang and [1]Kuanquan Wang
[1]School of Computer Science and Technology, Harbin Institute of Technology,
[2]School of Computer and Engineering, Heilongjiang University of Science and Technology, Harbin 150001, China

Abstract: Human skin is made up of multiple translucent layers and rendering of skin appearance usually acquire complex modeling and massive calculation. In some practical applications such as 3D game development, we not only approximate the realistic looking skin but also develop efficient method to implement easily for meeting needs of real-time rendering. In this study, we solve the problem of wrap lighting and introduce a surface details approximation method to give realistic rendering of virtual character. Our method considers that different thicknesses of geometry on the skin surface can result in different scattering degree of incident light and so pre-calculate the diffuse falloff into a look-up texture. Also, we notice that scattering is strongly color dependent and small bumps are common on the skin surface and so pre-soften the finer details on the skin surface according to the R/G/B channel. At last, we linearly interpolate the diffuse lighting with different scattering degree from the look-up texture sampled with the curvature and NdotL. Experiment results show that the proposed approach yields realistic virtual character and obtains high frames per second in real-time rendering.

Keywords: Real-time rendering, skin appearance, surface details approximation, wrap lighting

INTRODUCTION

Human skin is a highly translucent material and made up of multiple translucent layers, so it is a particularly complex material to model accurately (D'Eon and Luebke, 2007). The skin appearance is dominated by the Subsurface Scattering (SSS) of light diffusion, so we usually model the subsurface scattering to simulate the skin rendering. The subsurface scattering can be defined using the Bidirectional Surface Scattering Reflectance Distribution Function (BSSRDF) (Nicodemus *et al*., 1977), However the BSSRDF is a complex function that currently has to be massively calculated. A simple, but very effective dipole model (Jensen *et al*., 2001) for rendering skin was firstly presented via the light diffusion (Stam, 2001), which was quickly adopted in the practical application (Jensen and Buhler, 2002). Then multiple dipoles model (Donner and Jensen, 2005, 2006) extended the dipole model for light diffusion in human skin, which can capture the effects of discontinuities at the frontiers of multi-layered materials and give convincing rendering results. Recently, Won utilize the coloration and shading and design a 3D emotional method to make virtual character rendering (Won, 2010). However, these skin rendering methods have to take a long time to render character skin, so they are unsuitable for real-time rendering.

Several algorithms to simulate subsurface scattering and skin appearance in real-time have already been presented. Using Gaussian function with a customizable kernel to simulate the subsurface scattering in texture space (Borshukov and Lewis, 2003) is introduced and the technique introduced maps naturally onto GPU, but it still fails to capture the most complex subtleties of multiple scattering within skin material. On the other hand, the Modified Translucent Shadow Maps (MTSM) (Dachsbacher and Stamminger, 2003) was extended with irradiance and surface normal information, Using the modified depth map which was computed through the surface and connected shadowed regions to locations on the light-facing surface, they can obtain a good translucency. Later a two-pass Gaussian blurring in real-time was presented to simulate the skin (Gosselin, 2004), but the technique was not based on multi-layer translucent material and was only a very rough approximation of true scattering. Then, the work by D'Eon and Luebke (2007) and his colleagues simplified the multiple models (Donner and Jensen, 2005, 2006) by combining them with the idea of diffusion in texture space (Borshukov and Lewis, 2003; Dachsbacher and Stamminger, 2003). They approximated subsurface scattering in the three-layer skin with a weighted sum of Gaussian blurring for real-time rendering in texture space and obtained very realistic results. Compared with texture space

Corresponding Author: Mengzhao Yang, School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

rendering, screen space rendering (Jimenez *et al*., 2009) was provided a solution to simulate subsurface scattering at interactive frame rates and the skin rendering technology had been applied in 3D game development (Mittring and Dudash, 2011). These methods related above focused on the sense of reality and translucency for skin appearance. Besides the reality, in the 3D game development, however, the real-time performance should be given more consideration to make player switch different scenes rapidly.

In this study, we develop sufficiently simple method to implement realistic rendering so that it can be well integrated with existing GPU pipelines and give a high Frames Per Second (FPS) to meet the needs to switch 3D game scene quickly. To solve the problem of wrap lighting which is not based on real skin diffusion profile and doesn't account for surface curvature, we introduce a surface details approximation method to give a better realistic skin appearance and high FPS rendering.

FAST RENDERING METHOD

Wrap lighting is a simple method to approximate the diffuse lighting on the translucent materials and can simulate a good rendering appearance, however, it is not based on real skin diffusion profile and doesn't account for surface curvature, so the rendering result is not very realistic compared with our rendering. Our method is based on the observation that different thicknesses of skin surface give different scattering effect and scattering is strongly color dependent. We pre-blur diffuse BRDF with skin profile and parameterize BRDF with the curvature to give a better scattering effect and high speed of rendering.

Wrap lighting technique: Wrap lighting is the fast way to simulate scattering effect on skin appearance. Usually, diffuse lighting on the skin surface doesn't contributes any light when the surface normal is perpendicular to the light direction, so we can use wrap lighting to modify the diffuse situation. Through the modification, the lighting wraps around the object beyond the point where it would normally become dark. This can reduce the contrast of the diffuse lighting, which decreases the amount of ambient and fill lighting that is required. Though the wrap lighting is a simple technique to the approximation of diffuse lighting model, it can simulate a good rendering appearance of human skin while obtain a high rendering speed.

Figure 1 illustrates how to change the diffuse lighting function to include the wrap effect (Green, 2004). The value wrap is a floating-point number between 0 and 1 that controls how far the lighting will wrap around the object.

In the fragment shader, per-pixel lighting is computed and applied. Algorithm 1 gives part code of

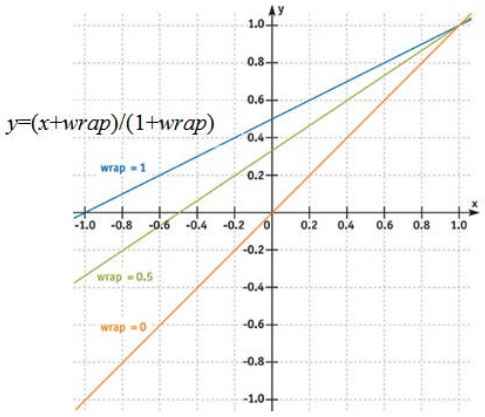

Fig. 1: Graph of the wrap lighting function

fragment shader to implement wrap lighting and compute color of per-pixel as follows:

Algorithm 1: Evaluation of Pixel Color through Wrap Lighting in the Fragment Shader.

Input: The 2D texture coordinate, *tex*; The value of wrap lighting function *NdotL_Wrap*; The diffuse light of per-pixel, *diffLight*; The color tint at transition from light to dark, *scatterT*; The specular light of per-pixel, *specLight.*

Output: Each pixel color of head mesh, *finalC:*

- *NdotL = tex.x*2-1*
- *NdotH = tex.y*2-1*
- *NdotL_Wrap = (NdotL+wrap)/(1+wrap)*
- *diffLight = max(NdotL_Wrap, 0.0)*
- *scatterT = smoothstep (0.0, scatter Width, NdotL_Wrap) *smoothstep(scatterWidth * 2.0, scatterWidth, NdotL_wrap)*
- *specLight = pow(NdotH, 40.0)*
- *if (NdotL_wrap< = 0) specLight = 0*
- *finalC.rgb = diffLight+scatterT*scatterColor*;
- *finalCr.a = specLight*
- return *finalC*

However the wrap lighting technique is not based on real skin diffusion profile, so it cannot give a better rendering result.

Surface details approximation method: The surface details approximation method aims to scale the physical details of skin surface and realizes the method in real-time quickly and easily.

Firstly, we have considered that there have different thicknesses of geometry of the whole skin surface. In some areas such as brow ridge, eyelid and nose, scattering is a lot more visible compared with on the forehead. This is because these areas have higher curvature and will result in a lot more diffuse falloff of

incident light, so the translucency is more obvious in these areas. We can pre-calculate the diffuse falloff of the whole skin surface into a look-up texture. The key to making the effect look good is that instead of having a 1-dimensional texture for *NdotL*, it is a 2D texture with the parameter 1/d in the y axis and the *NdotL* in the x axis and the parameter 1/d is approximated with a calculation of curvature based on similar triangles and derivatives. This will allow the falloff at the area of higher curvature to differ from that in the forehead or face which encompasses different falloffs for different thicknesses of geometry.

Secondly, we have noticed that scattering is strongly color dependent: red light scatters much farther than green and blue and the absorption properties of skin are very sensitive to changes in frequency. So the finer details on the skin surface are softened differently according to the R/G/B channel. We can obtain different soft map by the blending between a smoother normal map and the high detail normal map. Hence, we can basically start with our true surface normal which we would use for specular lighting, then pre-filter the new normal maps for R/G/B map using our skin profile again.

Thirdly, we have observed that some small bumps such as beard and pores are common on skin surface. In this case, light should scatter past several surface bumps, so the curvature breaks down at small sizes. Specular lighting shows the most dramatic, high-frequency changes such as bumps in the skin surface and these small bumps can be contained in the normal maps, so we can use normal map of specular lighting to represent these bumps. Through pre-filtering the normals by just simply blurring them and texture mapping, we can obtain the final rendering appearance which reflects small surface bumps.

Shader implementation of our method: Our method use just a pixel shader to account for scattering method related above. This base of this shader was a simple shader for diffuse, ambient and specular lighting and there are three texture maps used which are the original texture for character head, the normal and bump maps for original texture. If you have a texture you would like to use and do not have a normal or bump maps for it, there are tutorials online for creating normal maps and bump maps with Crazybump and other programs. Algorithm 2 gives part code of fragment shader to implement our method as follows:

Algorithm 2: Evaluation of Pixel Color through Our Method in the Fragment Shader.

Input: The texture map of original human face, *TexS*; The lerp value for calculating the amount of blur needed, *lerpAmount*; The diffuse light, *diffLight*; The curvature of the human face surface, *curvature*; The color of ambient, *ambientColor*; The diffuse degree of incident light, *diffLevel*; The specular color, *specColor*. The specular light, *specLight.*

Fig. 2: Original virtual character via texture mapping

Output: Each pixel color of head mesh, *finalCorlor*.

- *texColor = tex2D(TexS, tex)*
- *light.a = texColor.a*
- *light.r = Scatter(texColor.r)*
- *light.g = Scatter(texColor.g);light.b = Scatter(texColor.b)*
- *lerpAmount = saturate(Scatter(ambientColor.a))*
- *diffLight = lerp(light,texColor,lerpAmount)*
- *curvature = saturate(length (fwidth (normal))/ length(fwidth (float 4(worldNormal,1.0f))))*
- *rN=lerp(N_high,N_low,light.r); gN = lerp(N_high,N_low,light.g) bN=lerp(N_high,N_low, light.b)*
- *NdotL=float3(dot(rN,ambientColor.r),dot(gN,ambientCol r.g),dot(bN, ambientColor.b))*
- *diffLevel = DiffuseDegree(curvature, NdotL)*
- *H = normalize(V+L)*
- *Shininess = (40.0*diffLight.a)*128*
- *specularColor = float4(0.094f,0.114f,0.173f, 1.0f)*
- specLight = *pow(dot(H,N),shininess)*specColo*
- *diffLight = lerp(diffLevel,diffLight,lerpAmount)*
- *finalColor = (diffLight*texColor)+specLight*
- return *finalCorlor*

RESULTS

We have implemented the proposed method to give fast rendering of realistic appearance in real-time. With AMD Athlon II X4 Four Cores and NVIDIA GeForce GT230, we have realized the algorithm using HLSL shader and DirectX 9.0 programming in VS2010. We can achieve frame rates of approximately 464 frames per second, which is relatively high speed and is valuable practically in the real-time 3D games development.

We use the 3D head mesh and original face texture map from the IR-LTD digital scanned model (IR-LTD Model). Based on basic texture mapping and environment lighting, we can obtain the original rendering result for virtual character as shown in Fig. 2.

Then we use the wrap lighting technique to simulate scattering on the virtual character as shown in Fig. 3. Basically this technique creates a color shift as the lighting approaches zero. It certainly looks better

Fig. 3: Rendering of virtual character via wrap lighting

Fig. 4: Rendering of virtual character via our method

than the Fig. 2 but not great. In our running program, when zooming out our browser, we can see how this can be believable from a distance. For real-time 3D game, this technique could be easily applied to the stadium crowd to make them look more realistic. Another use might be when implementing a Level-of-Detail technique, changing to this method when the viewer gets far enough away can save processing time while maintaining a realistic looking model.

However the wrap lighting technique is not based on real skin diffusion profile and some details on the skin appearance such as whiskers and effects in areas with higher curvature cannot be clearly simulated. Using our method, we can obtain the rendering result as fast as wrap lighting but with much better results as shown in Fig. 4.

Our method can achieve the effects of subsurface scattering using only locally stored information and a custom shading model. This means that our shader becomes a simple pixel shader and no extra passes and no blurring is required as texture space rendering method or screen space rendering method, so it reduces the skin shading down to a single pass only and will scale linearly with the number of head mesh pixels that needs to be shaded. Being in O(N) time it makes this method acquire a high FPS.

Figure 2, 3 and 4 give a single and rigid impression for having no environment map as background. In the practical scene of 3D game, we display three virtual characters rendering obtained above at the same time in an office room as shown in Fig. 5 and show that three characters can be naturally integrated into the surroundings, which greatly enhance the sense of immersion when playing 3D game.

CONCLUSION

We have implemented a fast rendering method based on GPU for virtual character at high interactive speed. Considering that different thicknesses of skin surface give different scattering effect and scattering is strongly color dependent, we obtain the rendering result as fast as wrap lighting but with much better appearance. Experiments show that our proposed technique can not only generate realistic skin appearance, but also is easy to be implemented and well integrated with existing GPU pipelines. The vision of results in the practical office scene can enhance

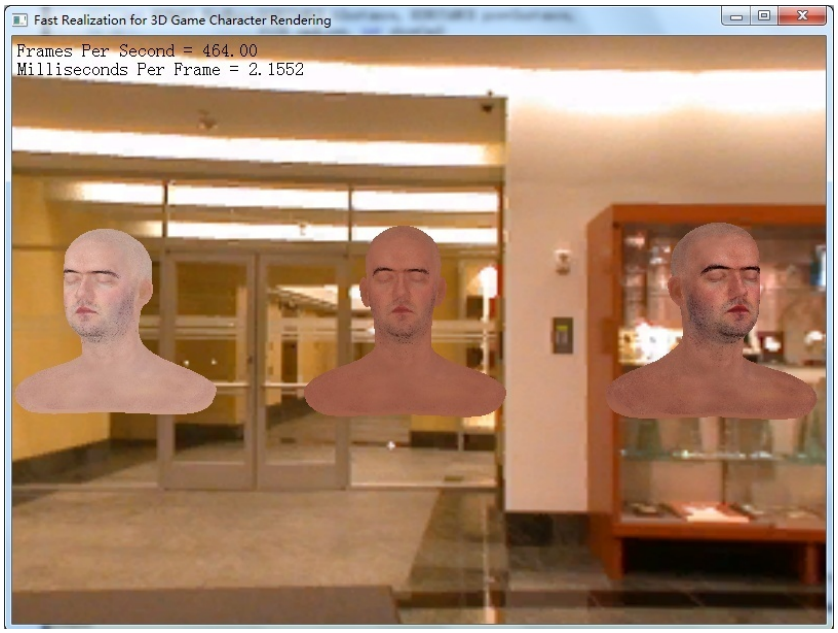

Fig. 5: Rendering of three virtual characters in a practical scene

immersion of digital character when playing game. Moreover, we design a fast rendering process and obtain a relatively high FPS, so it has an important application value for development in 3D game.

Even though we see our work as a step in allowing designers to quickly render and interact with realistic digital character, how the more realistic appearance is approximated is an interesting direction which we are currently working on.

ACKNOWLEDGMENT

This study was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61173086 and No. 61179009 and Science and Technology Foundation of Heilongjiang Province Education Department in China under Grant No. 11551435.

REFERENCES

Borshukov, G. and J. Lewis, 2003. Realistic Human Face Rendering for the Matrix Reloaded, SIGGRAPH Sketches and Applications. California, USA.

D'Eon, E. and D. Luebke, 2007. Advanced techniques for realistic real-time skin rendering. GPU Gems 3, 3: 293-347.

Dachsbacher, C. and M. Stamminger, 2003. Translucent shadow maps. Proceeding of EG Symposium on Rendering, pp: 197-201.

Donner, C. and H. Jensen, 2005. Light diffusion in multi-layered translucent materials. ACM Trans. Graph. (TOG), 24(3): 1032-1039.

Donner, C. and H. Jensen, 2006. A spectral BSSRDF for shading human skin. Proceeding of 17th Eurographics Workshop on Rendering, pp: 409-418.

Gosselin, D., 2004. Real time skin rendering. Proceeding of the Game Developer Conference, D3D Tutorial, Vol. 9.

Green, S., 2004. Real-Time Approximations to Subsurface Scattering. In: Fernando, R. (Ed.), GPU Gems. Addison-Wesley, Reading, pp: 263-278.

IR-LTD Model. 3D Scanning and Digital Character, Retrieved from: http:// www.ir-ltd.net/.

Jensen, H. and J. Buhler, 2002. A rapid hierarchical rendering technique for translucent materials. ACM Trans. Grap. (TOG), 21(3): 576-581.

Jensen, H., S. Marschner, M. Levoy and P. Hanrahan, 2001. A practical model for subsurface light transport. Proceeding of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ACM, pp: 511-518.

Jimenez, J., V. Sundstedt and D. Gutierrez, 2009. Screen-space perceptual rendering of human skin. ACM Trans. Appl. Percept. (TAP), ACM, 6(4): 3.

Mittring, M. and B. Dudash, 2011. The technology behind the diret X 11 unreal engine "Samaritan" demo. Proceeding of the Game Developer Conference (GDC).

Nicodemus, F.E., J.C. Richmond and J.J. Hsia, 1977. Geometrical Considerations and Nomenclature for Reflectance. National Bureau of Standards, Washington, D.C., pp: 52.

Stam, J., 2001. An illumination model for a skin layer bounded by rough surfaces. Proceeding of the 12th Eurographics Workshop on Rendering Techniques, Springer-Verlag, pp: 39-52.

Won, Y.T., 2010. 3D Emotional rendering method utilizing coloration and shading to make virtual character design in culture contents development. Int. J. Inform. Process. Manag., 1(1): 126-137.