International Journal of Multimedia and Ubiquitous Engineering
Vol.11, No.11 (2016), pp.343-352
http://dx.doi.org/10.14257/ijmue.2016.11.11.31

Comparison of Transparency Techniques in PSNRs Based on Deferred Rendering Using Multiple Render Targets

Youngsik Kim

Department of Game and Multimedia Engineering, Korea Polytechnic University
kys@kpu.ac.kr

Abstract

In three dimensional (3D) computer games, the deferred rendering is an effective way to process realistic visual effects such as dynamic lights, shadow, depth of field (DOF), and high dynamic range (HDR) using multiple geometric buffers(G-buffers) regardless of the scene complexity. However, the G-buffer only stores information about a single pixel in each texel, so transparency with alpha blending on deferred rendering is difficult compared to forward rendering. The conventional way to process transparency on deferred rendering is to separate opaque and transparent renderings. But the object sorting required in it causes the speed degradation. Nevertheless the transparency techniques without object sorting such as screen door, interfaced, and stochastic can reduce the rendering time, but those cause the image quality degradation. This paper compares transparency techniques on deferred rendering. The Game Institute 3D game engine is modified to measure the rendering speeds of transparency techniques. The deferred rendering can speed up the rendering time of about 1.48 times than the forward rendering. The transparency techniques without object sorting can also speed up the rendering time of about 1.1 times than the conventional technique. Plus, six 4K ultra high definition (UHD) game images are used to measure the extent of quality degradation of the transparency techniques without object sorting in terms of peak noise signal ratio (PSNR). Consequently, the interlaced technique can get better PSNRs of about 124.8% in the same than both the screen door and the stochastic techniques.

Keywords*: deferred rendering, transparent rendering, peak noise signal ratio (PSNR)*

1. Introduction

In 3D computer games, various special effects such as dynamic lights, shadow, depth of field (DOF), high dynamic range (HDR), and so on are becoming widely used. However, it is almost impossible for the forward rendering to perform those effects since it requires multi-pass rendering for the scene. The deferred rendering is an effective way to process special effects, because it overcomes the burden of performing multiple passes on the scene by reducing it to only rendering a full screen quad for each pass using multiple geometric buffers (G-buffers) regardless of the scene complexity [1].

However, deferred rendering has several drawbacks. First, it needs considerable memory to store multiple G-buffers, which impacts on the texture cache performance and the rendering speed. Second, it can suffer from anti-aliasing effects since the anti-aliasing has to be processed after the accumulation is done in the lighting and post-processing phase. Third, it can't handle transparency efficiently since the G-buffer only stores information about a single pixel in each texel, so blending on deferred rendering is not as simple as on forward rendering [2].

The conventional way to process transparency on deferred rendering is to separate opaque and transparent renderings. It is not doing deferred rendering on polygons that need to be blended. First, it performs opaque surfaces on deferred rendering and then

ISSN: 1975-0080 IJMUE
Copyright ⓒ 2016 SERSC

transparent surfaces with object sorting on forward rendering. But its object sorting causes the speed degradation.

Various studies [3, 4, 5, 6, 7] for order independent transparency (OIT) have been carried out. Also, some studies [8, 9, 10] for the 3D rendering effects have been carried out. The A-buffer [3] stores a list of transparent surfaces per pixel. However, the A-buffer requires the unpredictable amount of memory since it stores all transparent fragments at once. The depth peeling [4] breaks the blended complexity into layers and uses dual depth comparisons per sample to extract and composite each semi-transparent layer in a separate rendering pass. The depth peeling involves executing the complete deferred pipeline for each layer from filling the G-buffer to source shading, but requires an unbounded number of rendering passes. The screen door transparency [5] uses a stippling pattern to mask the transparent polygons without object sorting so that some pixels of the background can be seen through the mask. For example, if the alpha value is 50%, it skips all the even pixels in one row and all the odd pixels in the next. The screen door transparency is a simple way but it can cause an aliasing such as a moire pattern. The interlaced transparency [6] interlaces transparent objects with opaque ones in the geometry phase, then performs lighting, and then de-interlaces and blends them in the composition phase. In the geometry phase, all transparent pixels are rendered interlaced and only every odd horizontal line is rendered. In the composition phase, within each two rows, the minimum alpha value across the two vertically adjacent pixels is taken, and the two pixels are blended. The interlaced transparency overcomes the lighting inconsistency between opaque and transparent objects. But the vertical blur can be produced in the interlaced transparency. The stochastic transparency [7] extends the screen-door transparency with randomly chosen (sub-) pixel stipple patterns. The stochastic transparency creates transparency by simply omitting individual (sub-) pixels of transparent surfaces with the probability set by the alpha value. Since Stochastic Transparency is a Monte-Carlo-Algorithm, it produces much noise for a low number of samples.

Transparency or alpha blending is the process of blending a foreground transparent image with its background. It requires two colors, C_s (source pixel color) and C_f (fragment pixel color), and an alpha level, α . The final pixel color, C, is composited with the following formula [Eq.1].

$$C = C_s \times \alpha + C_f \times (1 - \alpha) \quad \text{[Eq. 1]}$$

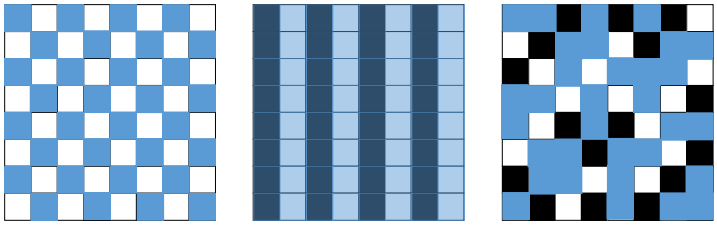
In Figure 1, the correct transparency image (c) between the background image (a) and the foreground image (b) is produced when the alpha value is 0.5. Figure 2 presents the blended images produced by transparency techniques without object sorting based on deferred rendering. The images of the screen door transparency (a), the interlaced transparency (b), and the stochastic transparency (c) are shown under the same condition with the background image, the foreground image, and the alpha = 0.5 in Figure 1.

(a) Background (b) Foreground (α=0.5) (c) Correct Transparency

Figure 1. Correct Transparency with Alpha = 0.5

 Copyright ⓒ 2016 SERSC

Nevertheless the transparency techniques without object sorting such as screen door, interfaced, and stochastic can reduce the rendering time, but those cause the image quality degradation. This paper compares transparency techniques on deferred rendering. The Game Institute 3D game engine [11] is modified to measure the rendering speeds of transparency techniques. Plus, six 4K ultra high definition (UHD) game images are used to measure the extent of quality degradation of the transparency techniques without object sorting in terms of peak noise signal ratio (PSNR).



(a) Screen Door Transparency (b) Interlaced Transparency (c) Stochastic Transparency

Figure 2. Transparency Techniques without Object Sorting based on Deferred Rendering

2. The Deferred Rendering Based on the Game Institute 3D Game Engine

In this Section, the experimental environment is constructed by modifying the source code of the Game Institute 3D game engine [11] in order to evaluate the transparency techniques based on deferred rendering. The algorithms of deferred rendering based on the Game Institute 3D game engine [11] are presented in Figure 3, Figure 4, and Figure 5.

In Figure 3, the function named FrameAdvance in the Game Institute 3D game engine, which is performed every frame, calls four sub-functions. (1) OpaqueRenderPass function performs opaque objects drawing. (2) OpaqueFogPass function applies fog to the opaque objects. (3) SkyboxPass function draws the skybox. (4) TransparentRenderPass function performs final transparent objects drawing. Among them, (1) OpaqueRenderPass function and (4) TransparentRenderPass function based on deferred rendering are explained in Figure 4 and Figure 5, respectively.

Pseudo Code : FrameAdvance( )

```
// Perform opaque object draw
1. OpaqueRenderPass( );
// Apply fog to the opaque objects
2. OpaqueFogPass( );
// Draw the skybox
3. SkyboxPass( );
// Perform final transparent object draw
4. TransparentRenderPass( );
```

Figure 3. Frame Advance() [11]

Copyright ⓒ 2016 SERSC

Pseudo Code : OpaqueRenderPass( )

Step1. Base Pass
1. Clear the output buffer to the correct color
2. Run the per-object base pass

Step2. Geometry Pass
3. Set the G-Buffer targets for writing
4. Initialize targets to their appropriate values
5. Run the per-object geometry pass
6. Finish writing to the G-Buffer targets

Step3. Lighting Pass
7. Begin reading from the G-Buffers
8. Run the deferred lighting pass
9. Finish reading from the G-Buffer targets
10. Execute for opaque geometry

Figure 4. OpaqueRenderPass() [11]

Pseudo Code : TransparentRenderPass( )

Step1. Geometry Pass
1. Set the G-Buffer targets for writing
2. Run the per-object geometry pass
3. Finish writing to the G-Buffer targets

Step2. Lighting Pass
4. Set the transparent writing target
5. Clear it to the required color
6. Begin reading from the G-Buffers
7. Run the deferred lighting pass
8. Finish reading from the G-Buffer targets
9. Finish writing to the lighting buffer

Step3. Composite Pass
10. Begin reading from the lighting buffer
11. Run the final composite pass
12. Finish reading from the lighting buffer
13. Execute for transparent geometry using back to front sorted rendering

Figure 5. Transparent Render Pass() [11]

In Figure 4, the OpaqueRenderPass function is composed of three major steps which perform multiple lights on the screen space with the G-buffer. In Step 1, Base Pass clears the output buffer. In Step 2, Geometry Pass sets the G-buffer targets for writing, initializes targets, performs the per-object geometry pass, and finishes writing to the G-buffer targets. Finally, in Step 3, Lighting Pass performs the deferred lighting pass by reading from the G-buffer targets and executes opaque objects rendering.

In Figure 5, the function of TransparentRenderPass is composed of three major steps which perform transparent object rendering. In Step 1, Geometry Pass prepares the G-buffers for targets, performs the per-object geometry pass, and writing to the G-buffers. In Step 2, Lighting Pass sets the transparent writing target, clear it, performs the deferred lighting pass by reading from the G-buffer, and finishes writing to the lighting buffer.

 Copyright ⓒ 2016 SERSC

Finally, in Step 3, Composite Pass performs the final composite pass by reading from the lighting buffer and executes for transparent geometry using back to front sorted rendering.

3. Performance Evaluation Using the Game Institute 3D Game Engine

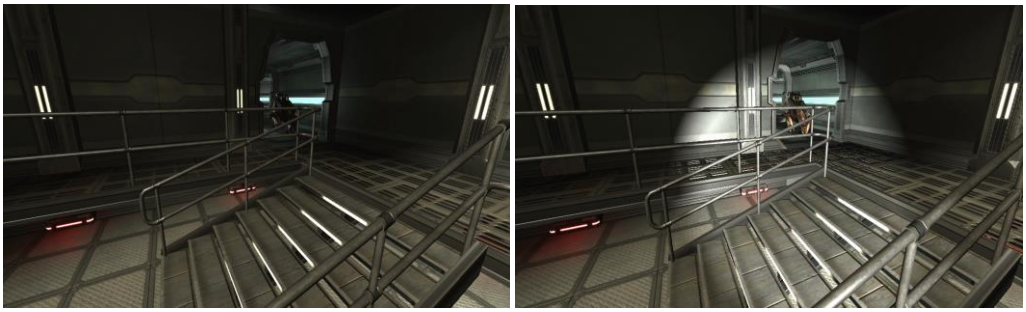
In this Section, the performance of transparency techniques both on deferred rendering and on forward rendering is analyzed. For the experiments, Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz 3.40GHz, 8GB RAM, and nVidia GeForce GTX650 Ti are used. Also, the Game Institute 3D game engine [11] is modified to measure the rendering speeds of transparency techniques.

The ten simulation models for the performance comparison of transparency techniques at two kinds of screen resolutions, 1280x720 and 2560x1600, are constructed in Table 1. Plus, the rendering speeds, frame per second (FPS), for the ten simulation models are measured. There are four criteria to configure the simulation models. First, ten models are divided into two major parts according to the rendering policy such as forward or deferred. Models of A, B, C, and D models are simulation models on forward rendering, but E, F, G, H, I, and J models are on deferred rendering. Second, each part for forward and deferred rendering models is divided into two parts whether include the transparent surface or not. A, B, E, F, G, and H models include the transparent surface, but C, D, I, and J models don't include one. The third criterion is Torch Light On or Off. A, C, E, F, and I models set Torch Light on. B, D, G, H, and J models set Torch Light off. The fourth criterion is whether or not object sorting among the models on deferred rendering with transparent surfaces. E and G models require object sorting. But F and H models don't require one, which perform transparency without object sorting. F and H models stand for the screen door transparency [5], the interlaced transparency [6], and the stochastic transparency [7] techniques.

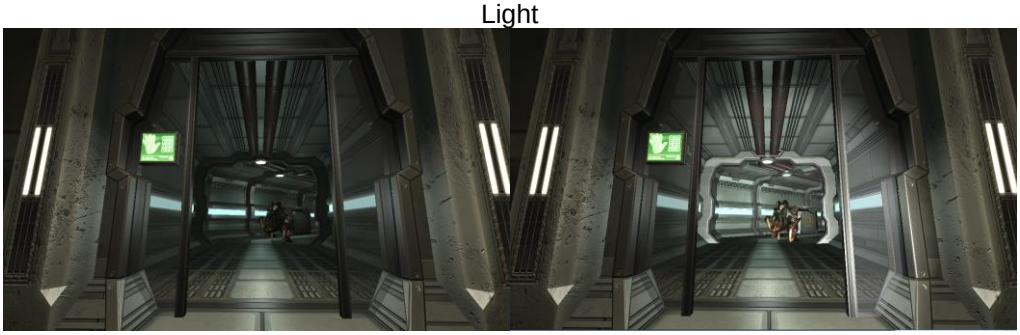
In Figure 6, various screen shots for the simulation models of Table 1 are shown. Figure 6 (a) stands for D and J models which don't include the transparent surface and set Torch Light off. Figure 6 (b) represents C and I models which don't include the transparent surface and set Torch Light on. Figure 6 (c) stands for B, G, and H models which include the transparent surface and set Torch Light off. Figure 6 (d) represents A, E, and F models which include the transparent surface and set Torch Light on.

The average rendering speed for forward rendering (A, B, C, and D) models is 233 or 87 at 1280x720 or 2560x1600 screen resolution, respectively. The average rendering speed for deferred rendering (E, F, G, H, I, and J) models is 343 or 87 on 1280x720 or 2560x1600 resolution, respectively. Consequently, there is no difference in terms of rendering speed between the deferred and forward rendering at the high screen resolution. But at the low resolution, the deferred rendering can speed up the rendering time of about 1.48 times than the forward rendering. Because, at the low screen resolution, the deferred rendering can process only a screen quad for each rendering pass using multiple G-buffers regardless of the scene complexity. However, at the high screen resolution, the scene complexity of the Game Institute 3D game engine can't play a big role on the

Copyright ⓒ 2016 SERSC



(a) Non-Transparent Scene without Torch Light (b) Non-Transparent Scene with Torch Light



(c) Transparent Scene without Torch Light (d) Transparent Scene with Torch Light

Figure 6. Screen Shots in Game Institute 3D Game Engine [11]

Simulation Model	A	B	C	D	E	F	G	H	I	J
Forward or Deferred Rendering	Forward Rendering				Deferred Rendering					
Transparent Scene (Yes/No)	Yes		No		Yes				No	
Torch Light (On/Off)	On	Off	On	Off	On		Off		On	Off
Sorting based on Deferred Rendering	NA				Yes	No	Yes	No	NA	
Rendering Speed (FPS) @ 1280x720 Screen	247	265	206	213	278	308	299	328	412	438
Rendering Speed (FPS) @ 2560x1600 Screen	71	76	96	106	71	78	77	84	102	110

Table 1. Simulation models and Rendering Speed in Game Institute Game Engine

Rendering speed. If the scene complexity would be very high, the deferred rendering could get an advantage in the rendering speed at the high screen resolution.

C, D, I, and J models which don't include transparent surfaces can also speed up the rendering time of about 1.1 times and 1.4 times than A, B, E, F, G, and H models which include transparent surfaces at the low and high screen resolution, respectively. The reason is that transparent surfaces require the additional rendering pass with object sorting.

Copyright ⓒ 2016 SERSC

Image 1 [12] Image 2 [13] Image 3 [14]

Image 4 [15] Image 5 [16] Image 6 [17]

Figure 7. Benchmark UHD (4096x2160 Resolution) Game Images

Also, B, D, G, H, and J models which set Torch Light off can speed up the rendering time of about 1.1 times in the same at the low and high screen resolution than A, C, E, F, and I models which set Torch Light on. Undoubtedly, Torch Light needs the additional rendering time.

Finally, based on deferred rendering, F and H models which are the transparency techniques without object sorting can also speed up the rendering time of about 1.1 times in the same at the low and high screen resolution than E and G model which are the conventional techniques with object sorting. F and H models stand for the screen door, the interlaced, and the stochastic transparency techniques. As the scene complexity gets higher, the impact of the object sorting on the rendering speed can get bigger. Although the rendering techniques, which don't need object sorting such as the screen door, the interlaced, and the stochastic transparency techniques, can speed up the rendering time, those ones make the image quality getting worse compared to the correct blended image due to some aliasing effects.

4. Performance Evaluation in PSNRs Using 4K UHD Game Images

In this Section, the extent of the image quality degradation of the transparency techniques which don't need object sorting on deferred rendering is evaluated. For the experiments, this paper prepares six 4K ultra high definition (UHD) game images in Figure 7. For the transparency processing (alpha blending), six background images are chosen from Figure 7. The foreground image is fixed as the 4096x2160 single color image whose (RGB) is (128,128,128). The extent of quality degradation of the screen door, the interlaced, and the stochastic transparency techniques are measured in terms of peak noise signal ratio (PSNR) [12] by comparing with that of the correct blended image.

PSNR [12] is an engineering term for the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. PSNR is most commonly used to measure the quality of reconstruction of lossy compression codecs (*e.g.*, for image compression). PSNR is most easily defined via the mean squared error (MSE) in [Eq. 2].

Copyright ⓒ 2016 SERSC

$$PSNR = 10 \times \log_{10}(\frac{MAX^2}{MSE}), \quad where$$

$$MAX = 255$$

$$MSE = \frac{\sum_{x=1}^{Width}\sum_{y=1}^{Height}[f(x,y)-g(x,y)]^2}{Width \times Height}$$

$$f(x,y) = correct\ value\ at\ (x,y)$$

$$g(x,y) = experimental\ value\ at\ (x,y)$$

$$Width = 4096$$

$$Height = 2160$$

[Eq. 2]

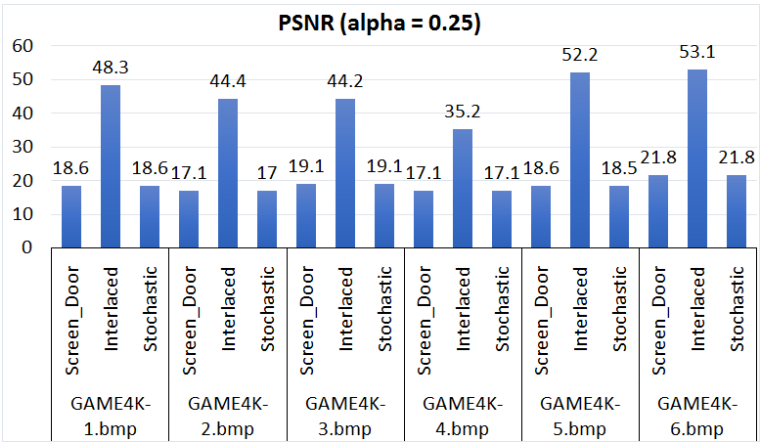
In this paper, MAX is the maximum value of the image pixel which can be computed by subtracting the minimum from the maximum. MAX of the each channel (R, G, or B) of the image is 255 (= 255-0) because of the 8bit channel. PSNR is usually expressed in terms of the logarithmic scale such db. The lower the image quality loss gets, the higher PSNR becomes. If there is no image loss, PSNR can't be defined because MSE is zero.

Figure 8 presents PSNRs for the techniques such as the screen door, the interlaced, and the stochastic transparency about six UHD games in three alpha values. The interlaced transparency can get the best PSNR of 41.2 in average for three alpha values. The PSNR value for the screen door is 18.3 which is the same as one of the stochastic transparency. Figure 8 shows that the periodic stippling pattern of the screen door makes the same impact on the image degradation as the probabilistic random pattern of the stochastic door. Consequently, the interlaced technique can get better PSNRs of about 124.8% in the same than both the screen door and the stochastic techniques.

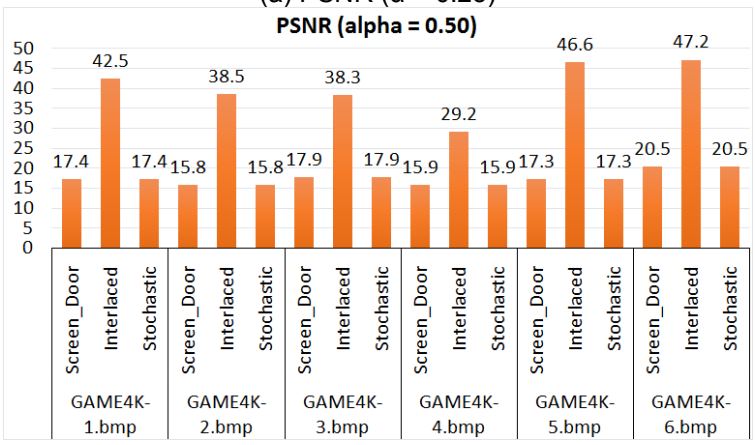
For the PSNRs of the interlaced technique for six UHD game image of Figure 7, one for the Image 6 is the best of 48.0 and one for the Image 4 is the worst of 30.0. So the difference between ones for the Image 6 and Image 4 becomes 59.9%. Because Image 6 is the most similar to the foreground image whose (RGB) is (128,128,128), but Image 4 is the most different from that.

According to the variation of alpha values, the interlaced technique can get better PSNRs of about 146.8%, 130.9%, and 96.9% than both the screen door and the stochastic techniques with 0.25, 0.50, and 0.75 alpha values, respectively. The higher the alpha value gets from 0.25 to 0.75, the smaller the difference of the image degradation between the transparency techniques. As the alpha value increases, the PSNRs of both the screen door and the stochastic techniques don't change by 18.7, 17.5, and 18.7, but the PSNRs of the interfaced technique decreases about 46.2, 40.4, and 36.9. As the alpha value increases in the interlaced technique, the ratio of the foreground image to be blended decreases.

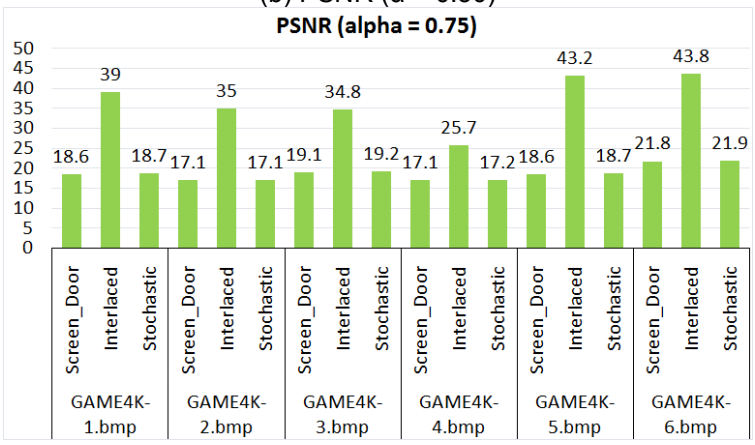
 Copyright ⓒ 2016 SERSC



(a) PSNR (α = 0.25)



(b) PSNR (α = 0.50)



(c) PSNR (α = 0.75)

Figure 8. Psnrs of Transparency Techniques for UHD (4096x2160 Resolution) Game Images with Various Alpha Values

Copyright ⓒ 2016 SERSC

5. Conclusion

This paper compares transparency techniques on deferred rendering. The Game Institute 3D game engine is modified to measure the rendering speeds of transparency techniques. The deferred rendering can speed up the rendering time of about 1.48 times than the forward rendering. The transparency techniques without object sorting can also speed up the rendering time of about 1.1 times than the conventional technique. Plus, six 4K UHD game images are used to measure the extent of quality degradation of the transparency techniques without object sorting in terms of PSNR. Consequently, the interlaced technique can get better PSNRs of about 124.8% in the same than both the screen door and the stochastic techniques.

References

[1] F. P. Placeres, "Overcoming Deferred Shading Drawbacks, ShaderX5 Advanced Rendering Techniques", Charles River Media, **(2007)**.
[2] S. Hargreaves and M. Harris, "Deferred Shading", GDC, **(2004)**.
[3] L. Carpenter, "The A-buffer, an antialiased hidden surface method", In Proceedings of SIGGRAPH, **(1984)**, pp.103-108.
[4] C. Everitt, "Interactive order-independent transparency", NVIDIA white paper, http://developer.nvidia.com, **(2001)**.
[5] H. Fuchs, "Fast spheres, shadows, textures, transparencies, and image enhancements in Pixel-Planes", In Proceedings of SIGGRAPH, (1985), pp. 111-120.
[6] E. Eric, "Stochastic transparency", IEEE Transactions on Visualization and Computer Graphics, vol. 17, no. 8, **(2011)**, pp. 1036-1047.
[7] D. Pangerl, "Deferred Rendering Transparency", ShaderX7 Advanced Rendering Techniques, Charles River Media, **(2009)**.
[8] Y. Lee, B. M. Tuan, K. Jeong, and J. Kim, "GPU-based Pen-and-Ink Rendering for Smoke Animation", Journal of Korean Society for Computer Game, vol. 24, no. 2, **(2011)**, pp. 101-107.
[9] J. H. Kim, "Study of 3D graphic-based stereoscopic camera control considering visual fatigue", Journal of Korean Society for Computer Game, vol. 24, no. 4, **(2011)**, pp. 89-98.
[10] K. S. Lee and Y. Kim, "Implementation of A Stereoscopic 3D Game To Reduce Visual Artifacts Using Oculus Rift", Journal of Korean Society for Computer Game, vol. 27, no. 4, **(2014)**, pp. 193-201.
[11] Game Institute Game Engine, http://www.gameinstitute.com
[12] PSNR, https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio
[13] Image 1, Asura's Wrath from http://www.giantbomb.com/
[14] Image 2, Tom Clancy's the Division from http://www.dlh.net/
[15] Image 3, Tom Clancy's the Division from http://www.dlh.net/
[16] Image 4, Devil May Cry 5 from http://-www.gamehdwall.com/
[17] Image 5, ForgeMaster from http://www.over3000.net/
[18] Image 6, China Lake Grenade Launcher Pop Gun from http://www.polycount.com/

Author

Youngsik Kim, received the B.S., M.S., and Ph.D degree in Dept. Computer Science from the Yonsei University, Korea, in 1993, 1995, and 1999 respectively. He had worked for System LSI, Samsung Electronics Co. Ltd from Aug. 1999 to Feb. 2005 as a senior engineer. Since March 2005 he has been working for Dept. of Game & Multimedia Engineering in Korea Polytechnic University. His research interests are in 3D Graphics and Multimedia Architectures, Game Programming, and SOC designs.

 Copyright ⓒ 2016 SERSC