다양한 텍스처 형식에 따른 실시간 렌더링의 FMQ 비교를 통한 효과적인 최적화(Optimization) 기법에 관한 연구

A Study on Effective Optimization by Comparison with FMQ of Real-time Rendering for Variable Surface Formats

채헌주, 유석호*, 경병표*
공주대학교 영상예술대학원 게임멀티미디어전공, 공주대학교 게임디자인학과*

Chae Heon-Joo, Ryu Seuc-Ho*, Kyung Byung-Pyo*
Dept. of Game Multimedia in Graduate School of Kong-ju National University*

요약

실시간 렌더링(Real-Time Rendering) 기술을 사용하는 게임 및 가상현실 환경에서 3차원 데이터를 처리할 때, 텍스처는 그래픽 카드에서 지원하는 텍스처의 형식에 따라 다른 결과를 가져온다. 이 텍스처 형식에 따라 발생되는 결과를 토대로, Frame Rate, Video Memory, Quality를 비교하여 텍스처 사용에 대한 최적의 방법을 찾는 데 필요한 자료를 제시하고자 한다.

Abstract

Textures used in the game and VR environments with real-time rendering technology have different results according to used texture format supported video card. We propose some ideas for the texture using method by comparison with Frame Rate, Video Memory, and Quality.

I. 서 론

3차원의 데이터를 실시간으로 처리하여 화면에 시각적으로 표시하는 실시간 렌더링에 있어서 데이터의 처리 능력을 향상시키는 것은 게임이나 가상현실과 같은 3차원 콘텐츠를 개발하는 개발자에게나 이를 사용하는 사용자에게 모두 중요한 문제이다[1]. 이렇게 처리 능력을 향상시키는 작업을 최적화(Optimization)라고 부르는데. 본 연구에서는 텍스처의 다양한 형식을 비교해 봄으로써 텍스처에 대한 효과적인 최적화 방법을 찾고 그 자료를 제시하고자 하였다.

이를 위하여 3차원 데이터의 환경을 적절히 구성하고 여러 가지 크기의 텍스처를 비교 대상으로 준비하였다. 그런 후 텍스처의 형식을 다양하게 변경하여 실시간 렌더링의 처리 능력을 3가지의 기본 항목들로 비교해 보고 그 결과에 대해 고찰해 보았다.

II. 본 론

1. 비교 방법

1.1 비교 항목

텍스처에 대한 실시간 렌더링의 성능을 평가하는 요소들은 매우 다양하다. 그 중에서 가장 기본적인 것이라고 할 수 있는 'Frame rate', 'Video Memory', 'Quality'의 3가지 항목을 선별하여 비교하고 이를 "FMQ 비교"라고 하기로 한다.

1) F(Frame rate) 항목

Frame rate는 단위 시간당 처리되는 장면의 수를 의미한다[2]. 이 수치에 대한 단위는 Frame Per Second(약자 FPS)를 사용하였다. 특히 Frame rate는 환경에 따라 수시로 변하기 때문에 1분 동안의 평균값을 구하여 비교하였다.

2) M(Video Memory) 항목

본 연구에서의 비디오 메모리는 텍스처가 그래픽 카드의 메모리 안에 할당되는 공간의 크기를 의미한다. 할당된 공간이 크다는 것은 그만큼 처리해야하는 데이터가 많다는 의미이므로 실시간 렌더링 시에 성능을 저하시키는 요인이 된다[6].

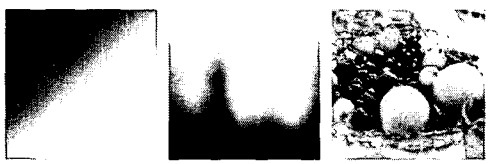
3) Q(Quality) 항목

시각적인 품질을 의미한다. 이 항목은 사실 주관적인 성격이 강하다는 것을 염두에 두고 '텍스처 형식'에 따라 시각적인 품질에 어떤 차이가 있는지를 알아보았다.

1.2 비교 환경

1) 텍스처 샘플

본 연구에서 주로 사용된 텍스처의 샘플은 [그림 1]과 같은 3종류의 이미지를 사용하였다. 이 이미지들은 투명도가 없는 24bit 색상 깊이를 가지고 있다.

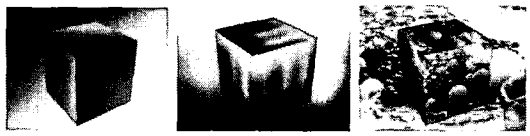


샘플(1) 샘플(2) 샘플(3)

▶▶ 그림 1. 텍스처 샘플

2) 3차원 장면

비교에 사용된 장면으로는 [그림 2]에서와 같이 배경에 비교 대상 텍스처를 배치하고 각 면에 동일한 텍스처를 입힌 정육면체를 1초에 90도 정도로 회전하도록 구성한 것이다. 또한 1개의 Point 형식의 조명을 사용하였다.



▶▶ 그림 2. 비교 장면

3) 그래픽 API

실시간 렌더링 분야에서는 주로 OpenGL과 DirectX가 사용되는데, 본 연구에서는 Microsoft 社의 DirectX 9.0c를 사용하였다.

4) 시스템 사양

모든 비교는 모두 하나의 하드웨어 시스템에서 이루어졌다. 덧붙여 다른 시스템에서 동일한 비교를 수행할 경우 결과는 달라질 수 있음을 밝혀 두는 바이다. 특히 그래픽 카드는 N사에서 제조하였고 128MB의 비디오 메모리가 탑재된 것을 사용하였다.

1.3 비교 조건

1) 텍스처 형식(Surface Format)

본 논문에서 언급한 "텍스처 형식"이라는 것은 Microsoft의 DirectX에서 규정한 "Surface Format"을 일컫는다[3]. 이 항목은 본 연구의 주된 비교 대상이다. 비교된 텍스처 형식은 총 9개로서 32 bits ARGB 8888, 32 bits RGB 888, 16 bits RGB 565, 16 bits RGB 555, 16 bits ARGB 1555, 16 bits ARGB 4444, Compressed DXT1, Compressed DXT3, Compressed DXT5 등이다.

2) 화면 크기(Screen Size)

화면 영역에 따라서도 성능이 크게 달라질 수 있다. 여기서는 640x480과 1024x768로 구분하여 비교하였다.

3) 텍스처 크기(Texture Dimension)

3가지의 텍스처 샘플의 크기를 각각 변경시켜 가면서도 비교해 보았다. 일반적으로 그래픽카드에서 텍스처의 처리 능력을 극대화 하려면 텍스처의 가로와 세로의 픽셀 단위의 크기가 2의 거듭제곱 형태로 되어야 한다[4]. 그러므로 본 연구에서도 텍스처의 크기를 128, 256, 512, 1024, 2048 등으로 가로와 세로를 동일하게 제작하여 사용하였다.

1.4 비교 도구

본 연구의 주요 비교 도구로는 Virtools 엔진을 사용하였지만, 초반에는 Virtools 엔진의 결과를 프로그래밍 방식의 결과와 비교하기 위하여 Microsoft의 Visual C++도 함께 사용하였다.

Virtools 엔진(이하 VT)을 비교 도구로 사용하기 전에 비교 도구로서의 타당성을 검증하기 위하여 Visual C++(이하 VC++)로 구현한 결과와 비교하여 보았다. 이 비교는 샘플(3)의 이미지와 500x500의 화면 크기만을 사용하였다. [그림 3]은 128x128 크기의 텍스처에 대하여 VC++와 VT의 Frame rate를 비교한 것이다. 여기에서 알 수 있듯이 두 도구 모두 1200~1250 FPS 사이에서 비슷한 결과를 나타내고 있다.

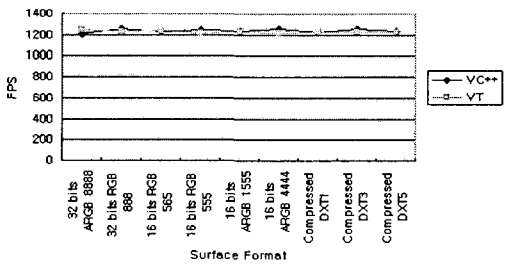
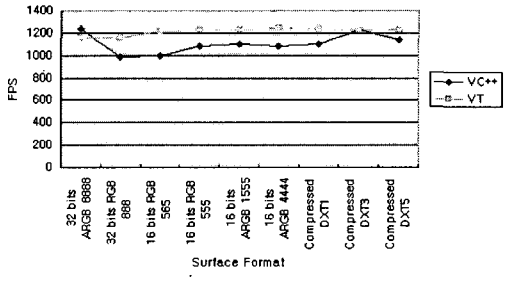


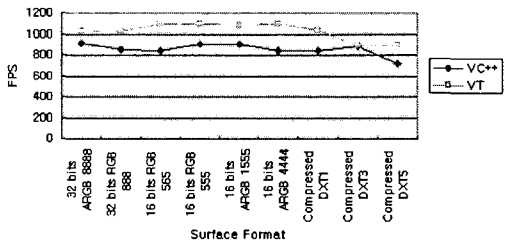
그림 3. 128x128 텍스처의 FPS 비교

[그림 4]는 512x512 크기의 텍스처에 대하여 VC++와 VT의 Frame rate를 비교한 것이다. VC++의 경우 대략 1000~1200 FPS 정도로 이지만 VT는 약 1200 FPS 로서 VC++를 상회하는 결과를 보이고 있다.



▶▶ 그림 4. 512x512 텍스처의 FPS 비교

[그림 5]는 512x512 크기의 텍스처에 대하여 VC++와 VT의 Frame rate를 비교한 것이다. 이전의 비교와 마찬가지로 수치는 더 낮아졌지만 VC++와 VT는 크게 차이가 나지 않는 범위 내에서 VT가 좀더 높은 Frame rate를 나타내었다. 이후의 비교에서는 모두 VT를 사용하였다.



▶▶ 그림 5. 2048x2048 텍스처의 FPS 비교

2. FMQ 비교 분석

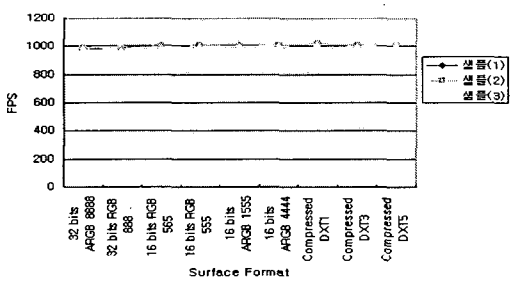
실시간 렌더링 시의 최적화를 극대화하기 위해선 빠른 처리 속도와 적게 차지하는 메모리 용량, 그리고 상대적으로 높은 품질을 구비하는 것이 핵심이라 판단하고 이들 각각을 쉽고 빠르게 파악할 수 있는 세 가지 요소 즉, FMQ(Frame rate, Video Memory, Quality)를 선택해 비교해 보았다.

2.1 F(Frame rate) 비교

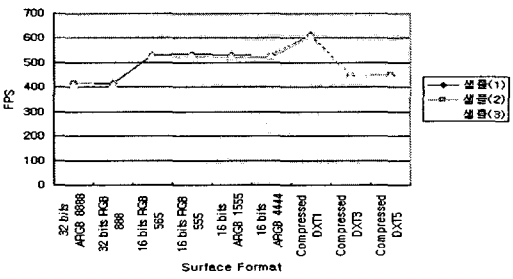
1) 640x480 화면 크기에서의 비교

[그림 6]과 [그림 7]은 640x480의 화면 크기에서

화면 크기보다 작은 128x128의 텍스처와 화면 크기보다 훨씬 큰 2048x2048 텍스처를 비교한 결과이다. [그림 6]을 보면 3종류의 샘플 모두 텍스처 형식에 따라서는 별 차이가 없는 것을 확인할 수 있다. 하지만 [그림 7]에서는 16bit 쪽이 32bit 보다는 수치가 높았고 압축된 형식 중에서 Compressed DXT1이 가장 높은 Frame rate를 보였다. 하지만, Compressed DXT3과 DXT5에서는 32bit 보다는 높았지만, 16bit 계열 보다는 낮은 결과를 보여주고 있다.



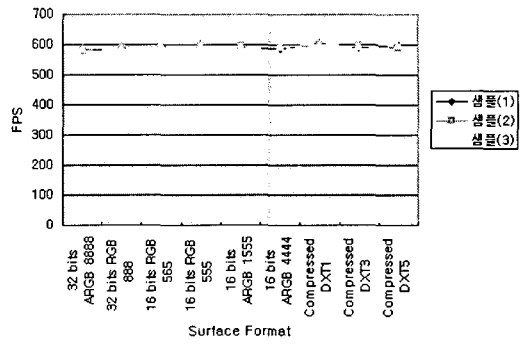
▶▶ 그림 6. 640x480에서의 128x128 텍스처



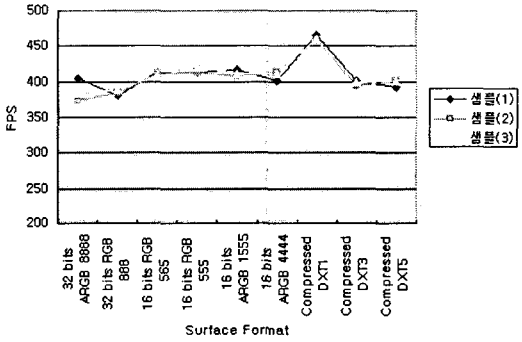
▶▶ 그림 7. 640x480에서의 2048x2048 텍스처

2) 1024x768 화면 크기에서의 비교

[그림 8]과 [그림 9]는 1024x768의 화면 크기에서 화면 크기보다 작은 128x128의 텍스처와 화면 크기보다 훨씬 큰 2048x2048 텍스처를 비교한 결과이다. 앞서 640x480 화면 크기의 경우와 마찬가지로 작은 크기의 텍스처의 경우에는 텍스처 형식 간에 거의 차이가 없지만 큰 크기의 텍스처의 경우에는 확연한 차이를 볼 수 있다. 역시 Compressed DX1 형식이 두드러지게 높은 수치를 나타내었고 나머지는 모두 비슷한 결과를 보이고 있다.



▶▶ 그림 8. 1024x768에서의 128x128 텍스처



▶▶ 그림 9. 1024x768에서의 2048x2048 텍스처

3) 화면 크기에 따른 Frame rate

앞서 640x480 화면 크기의 경우와 1024x768의 경우를 비교하여 보면 화면 크기가 커짐으로 인해서 전체적인 수치가 낮아졌음을 알 수 있다. 특히 주목할 점은 128x128 텍스처의 경우에는 400 FPS 정도의 차이를 보이고 있지만 2048x2048의 텍스처인 경우에는 형식에 따라 적게는 약 25 FPS에서 많게는 약 100 FPS의 차이를 보인다는 것이다. 즉, 크기가 큰 텍스처보다 작은 텍스처가 화면 크기에 따라 큰 폭의 Frame rate 차이를 보이는 것을 확인할 수 있다.

2.2 M(Video Memory) 비교

두 번째로 그래픽 카드에 점유하고 있는 메모리 공

간을 살펴보았다. 일부 형식에서는 예상대로 표현 형식의 bit 수에 따라 결정된다는 것을 [표 1]에서 보여주고 있다. 128x128 텍스처의 경우 32bit 로 표현되려면 한 텍셀 당 32bit의 메모리 저장 공간이 필요하므로 총 필요한 공간은 128x128x32bit가 된다. 이를 kilobyte로 환산하면 64KB가 되는 것이다.

반면에 압축 형식인 Compressed DXT1, DXT3, DXT5는 현저히 낮은 메모리 공간을 차지하고 있다. 원본에 비해 각각 12.5%, 25%, 25%의 메모리 공간만 있으면 된다. 시각적 품질이 많이 차이나지 않는다면 오히려 bit수를 낮추어 16bit로 처리할 때 보다 압축 형식을 사용하는 것이 메모리의 저장 공간을 절약할 수 있는 좋은 방법이 될 것이다.

[표 1] Video Memory 비교

Surface Format	128x128	2048x2048	상대비율
32 bits RGB 888	64KB	16MB	100%
16 bits RGB 565	32KB	8MB	50%
16 bits RGB 555	32KB	8MB	50%
16 bits RGB 4444	32KB	8MB	50%
Compressed DXT1	8KB	2MB	12.50%
Compressed DXT3	16KB	4MB	25%
Compressed DXT5	16KB	4MB	25%

2.3 Q(Quality) 비교

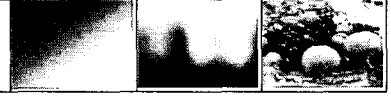
마지막으로 각 텍스처 형식에 따른 시각적 품질을 비교하였다. 시각적 품질의 비교에서는 텍스처의 크기와는 무관하므로 이 항목은 제외하고 3종류의 이미지를 비교하여 [표 2]와 같이 정리하였다.

먼저 32 bits RGB 888 형식에 대해서는 3종류의 샘플 모두 원본과 동일한 품질을 표현하고 있다. 또한 16 bits RGB 444 형식에서 샘플(1)과 샘플(2)에서 두드러지게 낮은 품질을 나타내었고, 압축 형식인 Compressed DXT1, DXT3, DXT5에서는 원본에 비해 약간의 색상 손실을 보였다. 특히 압축 형식 중에서는 Compressed DXT1이 나머지 보다 색상의 손실 정도가 약간 더 높았다.

원본 이미지들의 색상 깊이가 24bit이므로 32bit에서 원본과 동일하게 보이는 것과 16bit에서 약간의 이미지 손실이 있는 것은 당연하다. 또한 압축 형식에서도 원본이 압축이 될 때 데이터가 손실되므로 원본보다는 품질이 떨어지게 된다.

특이한 점은 샘플(1)과 샘플(2)에서는 이러한 사실을 어느 정도 육안으로 확인할 수 있지만, 샘플(3)에서는 자세히 살펴보지 않으면 잘 분간하기 어렵다는 것이다.

[표 2] Quality 비교

Surface Format	샘플(1)	샘플(2)	샘플(3)
32 bits RGB 888			
16 bits RGB 565			
16 bits RGB 555			
16 bits RGB 444			
Compressed DXT1			
Compressed DXT3			
Compressed DXT5			

III. 결 론

본 연구의 FMQ 비교 결과를 토대로 텍스처에 어떤 형식을 사용하느냐는 어떤 목적을 가지고 최적화를 수행하느냐에 따라 달려있다고 말할 수 있다. 높은 Frame rate를 위해서는 텍스처의 형식에 관계없이 작은 크기의 텍스처를 사용하고 화면 크기를 되도록이면 작게 하는 것이다. 큰 텍스처를 사용할 경우에는 Compressed DXT1 형식을 사용하는 것이 좋

을 것이다. 비디오의 메모리 공간을 절약하기 위해서는 역시 적은 크기의 텍스처를 사용하고 Compressed DXT1 형식을 사용하는 것이 좋다. 그리고 높은 시각적 품질을 원한다면 원본이 24bit 이상의 색상 깊이를 가지고 있을 때 32bit RGB 888 형식이나 32bit ARGB 8888 형식을 사용하는 것이 좋다. 하지만 다른 형식을 사용해도 품질에 많이 손상이 가지 않는다면 Frame rate와 Video Memory를 고려하여 선택하면 더 좋은 결과를 얻을 수 있을 것이다.

향후에는 좀더 다양한 시스템에서 비교 연구를 수행하여 종합적이고 일반적인 결과를 얻어내고 비교 항목과 환경을 좀더 확장하고자 한다. 또한 비디오 메모리에 저장되는 텍스처 형식이 아니라 JPG, BMP, DDS, PNG 등의 파일 형식에 따른 비교도 수행할 예정이다.

▌참 고 문 헌▐

[1] 유석호, TLC Rendering을 활용한 효과적인 3D합성에 관한 연구, 한국정보과학회, 2004.
[2] Tom Meigs, Ultimate Game Design: Building Game Worlds, p.6, McGraw-Hill, 2003.
[3] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_m/directx/direct3d/gettingstarted/direct3dsurfaces/surfaceformats.asp
[4] Kelly Dempski, Real-Time Rendering Tricks and Techniques in DirectX, pp.190, Course Technology PTR, 2002.
[5] Tomas Moller, Real-Time Rendering 2nd Edition, pp.117-179, AK Peters, LTD., 2002.
[6] 최광일 역, DirectX 실시간 렌더링 실전 테크닉, p.243, 정보문화사, 2003.
[7] Hell of Hope 역, C++와 DirectX9를 이용한 실시간 3D 지형엔진, p.225, 정보문화사, 2004.
[8] Virtools Dev User Guide, 2004.
[9] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/directx/graphics/ProgrammingGuide/GettingStarted/Direct3DTextures/basic/textureaddressmodes/addressmodes.asp
[10] http://msnd.microsoft.com/direcx