3D 영상 게임에서 이미지 스페이스 글로우 효과

최학현, 이명학, 김정희
게임솔루션 부설연구소

Image Space Glow Effects for 3D Visual Game

Hak-Hyun Choi, Myung-Hak Lee, Jung-Hee Kim
Dept. of Game Lab, Game Solution
woono7@yahoo.co.kr

요 약

3D 영상엔 화려하고 멋진 그래픽 기술들이 존재한다. 본 논문에서는 영상 게임 연출자들이 표현하고 싶어하는 백열광 효과인 Image Space Glow 효과를 구현하는 방법과 프로그램 디자인과 프로그램 구현을 3D 라이브러리를 이용하는 데에 초점을 두었다.

1. 서론

Glow란, 백열광을 발하다라는 의미로 3D Visual Game에서의 Image Space Glow 효과는 어둠 속에서 밝은 빛이 갑자기 들어올 때 눈부신 영상효과를 의미하고 이것을 Texture Image Data로 Rendering을 하여 얻은 특수효과 이미지이다. 본 논문의 연구 배경은 게임에서 표현하고자 하는 눈부신 효과의 표현과 구현의 제약에서 벗어나서 3D 영상 게임의 제대로 된 백열광 이미지를 얻어 Image Space Glow 효과를 구현함에 있다. 기존의 백열광 효과의 구현 연구는 주로 게임 기획제작자들이 일반 색상 값을 응용한 기술의 기본 원리를 이용하여 구현 한 것에 불과하다. 본 논문에서는 현재 사용되는 백열광 효과의 구현의 단점과 문제점을 파악하여 그에 대한 해결책을 제시하고 그것을 Image Space Glow 효과라는 결과물을 통해서 프로그램을 제작하고 구현하여 3D 영상 게임에서의 향상된 백열광 효과를 얻을 수 있는지 구현 여부를 점검하는데 초점을 두고 있다.

2. 문제점 및 해결 방안

2.1 기존 백열광 이미지 효과의 단점과 문제점 파악

2.1.1 지속적인 눈부신 효과 표현의 부적합

일반적으로 눈이 부신 효과를 구현하는 방법은 몇 가지가 있다. 그 여러 가지 중 색상 값을 실시간으로 조절하여 명도나 채도 값을 변경하는 방법과 감마값을 실시간으로 조절하는 방법이 있다[2].

2.1.2 이미지요소의 외곡 현상

단순히 RGB값을 상승시키다보니 색상의 외곡현상이 일어난다. 색상이 왜곡되다 보니 이미지의 요소들이 형체를 알아보기 힘들어 지게 되어 버린다. 결국 화면을 밝아지게 되고 이미 밝은 이미지 요소들은 사라져서 흰 물체가 되어 버리고 만다. 즉 관찰자인 게임 유저는 연출하고자 하는 화면을 제대로 이해하지 못한 체로 화면 안에는 아무것도 없게 되어 버리는 흰 화면을 보게 되고 마는 것이다.

2.1.3 Fade In 표현 기법과 혼란

영화에서 어두운 화면에서 점점 밝아지는 화면을 F.I(fade in)이라고 한다[3]. 이런 기법은 장면의 전환을 위해 사용되는 영화 촬영기법이다. 즉 눈부신 효과

를 얻기 위한 장면 연출법은 아니다. 위의 RGB값을 상승 시키는 것은 영화의 F.I과 같은 기법이라서 눈부신 효과와 별차이점이 없게 된다. 그럼 감마를 상승시키면 되는데 이 감마를 상승시키면 전체적으로 색상의 밸런스가 무너져 버리고 오히려 이미지의 탁탁한 느낌을 주게 된다.

2.2 연구 목표 및 해결 방안

2.2.1 기존 백열광 효과 대체 방안 연구

기존의 백열광 효과는 단순히 RGB값을 상승시키거나 감마값을 상승시키는 방법으로 특별한 알고리즘이라고 볼 수 없다. 이를 실제 이미지 데이터를 가공하여 처리하는 방법으로 대처할 수 있는 다른 구현 방법을 모색하고자 한다.

2.2.2 백열광 효과의 지속적인 표현 처리 연구

기존의 백열광 효과는 지속시간이 짧고 결국엔 흰 화면이 되어 버린다는 단점이 있다. 그래서 어떻게 하면 지속적으로 표현과 결과적으로 흰 화면이 되지 않게 유지 할 수 있는지 방법을 제시하는데 비중을 두고 있다.

3. 이미지 스페이스 글로우 효과

눈이 부신 효과를 구현함에 있어 텍스쳐 데이터 이미지를 이용하여 이미지 스페이스 글로우 효과를 적용하면 단순한 백열광효과에서 벗어 날수 있고 지속적이면서 영화의 F.I 기법과 확실히 다른 이미지 데이터를 얻어 우리가 원하는 최종 3D 영상 게임을 얻게 되어 위의 문제점들이 한 번에 해결 된다는 점이고, 이 기능을 잘 활용하여 백열광 효과 데이터를 3D 영상 게임에서 구현이 가능하게 되면 연구의 목적을 이룰 수 있다.

3.1 기존 백열광 효과의 대체 방안

기존의 백열광 효과를 단순히 RGB값으로 처리하는 방법 대신에 렌더링된 이미지 데이터를 가공하여 알파채널로 활용하는 방법과 안티알리싱 효과를 동시에 사용하여 좀더 나은 백열광 효과를 표현할 수 있는 방법으로 대처하고자 한다.

- 이미지 데이터의 알파 값으로 변환 처리 방식
- 안티알리싱을 이용한 이미지 번짐 처리 방식

3.2 이미지 스페이스 글로우 효과 구현 방법

이미지 스페이스 글로우 효과를 표현하기 위한 이미지 데이터를 얻는 과정을 아래 [표 1]과 같은 알고리즘 방법으로 구현한다.

입력	랜더링이 끝난 이미지 데이터 소스 입력
구현 과정	원본 이미지 데이터 소스를 그레이스케일 이미지 데이터로 변환. 그레이스케일 이미지를 알파채널로 사용. 알파채널의 이미지의 가로세로 비율을 1/2로 낮춤. 작아진 알파채널 이미지를 다시 원래 사이즈로 확대. 원본 이미지데이터의 알파채널 값을 새로 구현 알파채널 값으로 교체.
출력	최종 이미지 스페이스 글로어 효과 처리가 된 이미지를 모니터에 출력

[표 1] 알고리즘 구현 방법

4. 이미지 스페이스 글로우 효과 설계

4.1 구성

백열광 효과를 얻기 위한 이미지 스페이스 글로우 효과 테스터기는 메인 윈도우, 메뉴, 로더, 렌더, 허드, 이미지 스페이스 글로우 효과처리로 구성되어 있고 각 구성의 기능은 아래에서 자세히 설명하도록 하겠다.

4.1.1 메인 윈도우

전체화면 윈도우 혹은 창모드 윈도우등으로 구성되며 이 소프트웨어를 시작하기 위한 처음 생성 부분이다. 메인 윈도우의 초기 크기에 따라서 모든 비주얼씬은 그에 맞게 조정이 되게 제작되어 있다.

4.1.2 메뉴

메뉴에선 테스트 하고 싶은 형태를 고르거나 현재 자신의 컴퓨터 사양에 대한 정보, 혹은 실험 환경 시스템 환경 설정, 각종 기타 설정 및 데이터의 저장을 담당하고, 프로그램 종료와 테스트 진행을 할 수 있도록 구성 되어 있다.

4.1.3 로더

테스트 환경이 설정 되었다면 그것에 맞게 환경을 준비하는 부분이다. 테스터기의 특징상 3D Visual이기 때문에 월드를 구성하기 위한 데이터, 캐릭터 데이터, 각종 텍스쳐 데이터와 음향 데이터등을 로딩 한다.

4.1.4 3D 영상 엔진 및 렌더

모든 데이터가 준비되었다면 각 부분을 연출하고자 하는 의도대로 세팅을 하고 최종 이미지로 볼 수 있게 렌더링을 해 주는 부분이다. 모든 장면은 렌더링 과정을 거쳐서 최종 이미지로 볼 수 있게 해주는 역할을 담당하고 있으며 이 부분은 미리 제작해 둔 3D 영상 엔진을 이용하여 표현하였다.

4.1.5 이미지 스페이스 글로우 효과처리

일반적으로 백열광 효과를 사용하지 않는 비주얼씬 부분과 본 논문에서 제시한 이미지 스페이스 글로우 효과를 구현하여 본 비주얼씬을 실제로 비교해 볼 수 있도록 선택적으로 처리해주는 부분이다. 이 부분에서 두 가지 비주얼씬를 실시간으로 전환하여 처리할 수 있도록 구성 되어 있다.

4.1.6 HUD

현재 진행되고 있는 실험의 테스트 상황의 각종 정보를 수치로 볼 수 있게 해주는 부분이다. 허드에선 렌더링 처리 속도와 이미지 데이터의 폴리곤 수, 텍스쳐 데이터의 양 혹은 기타 정보들에 대해서 정확하고 자세히 실시간으로 처리하여 보여 준다.

4.2 설계

이미지 스페이스를 이용한 글로우처리하기 위한 소프트웨어 설계와 같다.

5. 실험 및 결과 분석

5.1 구현

윈도우와 3D게임 엔진을 이용한 이미지스 페이스 글로어효과 테스터기는 클래스로 구현한다.

5.2 실험

5.2.1 실험 환경

이미지스 페이스 글로우효과 테스터기는 다음과 같은 컴퓨터 시스템 사양에서 실험하였다.

- 운영체제 : Windows XP Professional
- BIOS : Award Medallion BIOS v6.0
- CPU : Intel(R) Pentium(R) 4 CPU 1.80GHz
- 메모리 : 1024MB RAM
- Page File : 404MB used, 2061MB available
- DirectX Version : DirectX 9.0 (4.09.0000.0900)
- OpenGL Version : OpenGL 1.3
- 3D 그래픽 카드 : RADEON 9700/9500 SERIES
- 제조업체 : ATI Technologies Inc.
- Chip type : Radeon 9700AGP (ND)
- DAC type : Internal DAC(400MHz)
- 비디오 메모리 : 128.0 MB
- 테스트 해상도 : 1024 x 768 (32 bit) (85Hz)
- 모니터 : 삼성 모니터 Max Res: 1600,1200
- 비디오 가속 : ModeMPEG2_D ModeMPEG2_C
- 3D 그래픽 엔진 : NitroFamily Game Engine

5.2.2 실험 방법

위와 같은 동일한 실험 환경에서 이미지 스페이스 글로우 효과처리 테스터기를 실행한다. 그리고 백열광 효과를 이용할 3D 캐릭터와 주변 환경을 아래와 같이 설정하여 테스트를 한다.

- 테스트를 위해 필요한 지형 공간을 생성한다.
- 마련된 공간에 소품 및 캐릭터를 배치한다.
- 일반적인 3D 게임 영상을 본다.

- 이미지 스페이스 글로우효과를 처리했을 경우 3D 영상 게임의 변화여부를 측정한다.
- 구현이 되었는지 최종 확인 정검을 한다.

5.2.3 실험 결과

일반 3D 영상 게임의 그래픽이 눈이 부신 이미지 데이터를 얻기위한 백열광 효과인 이미지 스페이스 글로우 효과를 적용 하였을 경우 어떻게 변하였는지의 결과는 아래 그림과 같다.

[그림 1] 이미지 스페이스 글로어효과 구현 결과

5.3 결과 분석

위에서 실험한 결과인 [그림 1]에서 보는 봐와 같이 지속적인 백열광 효과가 가능한 이미지 스페이스 글로우 효과를 구현하였다는 것을 볼 수 있다.

6. 결론

본 논문에서는 단순한 RGB값을 이용한 백열광 효과가 아닌 이미지 데이터를 가공하여 Image Space Glow를 이용한 백열광 효과를 구현하였음을 증명하였고, 이것을 구현하기 위한 방법과 설계 그리고 소프트웨어를 제작하였다. 게임에서 일반적 게임장면의 연출이 가능하다. 그러나 게임에서 완전한 백열광 효과를 얻기 위한 문제점을 해결하고자 하였다. 그리하여 본 논문에서는 가장 효과적이고 최신 기법인 백열광 효과 구현 방법을 알아보았고, 그의 문제점을 파악하여 이미지 스페이스 글로우 효과를 구현하기 위한 방법들을 제안하였다. 이로 인하여 확실히 눈부신 이미지 효과를 얻었을 뿐만 아니라 지속적이고, 눈이 부신 영상 속에서도 물체의 외곡이 없이 구분이 가능한 발전된 3D 게임 영상 처리 기술을 구현하였다. 본 논문의 가장 근본적인 목적인 눈이 부신 3D 영상 게임 연출을 할 수 있게 구현하는 것이었으며, 이로 인하여 3D 영상 게임에서 보다 더 화려한 연출을 할 수 있는 그래픽 표현 영역을 증가 시켰다. 또한 3D 게임 하드웨어의 영상 처리능력을 최대한 활용 할 수 있었다. 마지막으로 이러한 3D 영상 게임의 그래픽처리 기술은 발전은 보다 많은 3D 영상 처리 기법들이 게임에서 표현이 가능해 지고, 실시간에서, 보다 실사와 가깝게 영상을 출력해 주게 될 것이며, 그리하여 앞으론 좀더 발전된 기술들을 이용한 실시간 랜더링 기법으로 영화와 똑같은 3D 영상 게임을 즐길 수 있는 날이 곧 올 것이라 기대해 본다.

[참고문헌]

[1] Roger T.Stevens, Computer Graphics Dictipnary CHARLES RIVER MEDIA, 2001

[2] Richard S. Wright, Jr. Michael Sweet/ 남기혁 역, OpenGL Super Bible Second Edition, 인포북, 2001

[3] Mason Woo, Jackie Neider, Tom Davis, Dave Shreiner, OpenGL Programming Guide Third Edition ADDISON-WESLEY, 2001

[4] Adrain Perez with Dan Royer/ 이주리 역, Advanced 3D Game Programmin using DirectX 민프레스, 2001

[5] Crooks, Clayton E, 3D Game Programming With DirectX 8.0, Charles River Media

[6] 메이슨 우/ 남기혁 역, OpenGL 프로그래밍 가이드(제3판), 인포북, 2003

[7] 김용준, 3D 게임 프로그래밍 IT EXPERT, 한빛미디어, 2003