# 뷰 모핑을 사용하여 관측 방향에 따라 변화하는 빌보드 기법

김승완[o†] 송주환[††] 권오봉[†]
† 전북대학교 전자정보공학부
†† 전주대학교 교양학부
kswsamson@mail.chonbuk.ac.kr[o], jwsong@jeonju.ac.kr, obgwun@moak.chonbuk.ac.kr

# A Billboarding Techniques dependent on View Direction Using View Morphing

Seungwan Kim[o†] Juwhan Song[††] Oubong Gwun[†]
† School of Electronics & Information Engineering Chonbuk National University
†† School of Liberal Arts Jeonju University

요 약

이 논문은 그래픽스 API인 OpenGL에서 제공하는 기존의 빌보드 기법을 좀 더 사실적으로 표현하기 위한 방법에 대해서 고찰한다. 빌보드 기법은 다각형에 텍스처 이미지를 매핑하여 이 다각형이 항상 시점을 바라보도록 표시하여 어느 정도의 사실성을 유지하는 3차원 영상을 빠르게 생성한다. 그러나, 기존의 빌보드 기법은 물체의 위치에 관계없이 항상 동일한 이미지를 사용하여 사실성이 떨어지는 단점이 있다. 이 논문에서는 기존의 빌보드 기법과 관측 방향에 따라 다른 이미지를 생성하는 뷰 모핑 기법을 조합하여 관측 방향에 따라 물체가 변화하여 보다 사실적인 3차원 영상을 생성하는 빌보드 기법을 제안한다.

## 1. 서 론

빌보드 기법은 원래 폭탄이 폭발하는 모양이나 나무 등과 같이 어느 방향에서 보나 모양이 비슷한 물체를 빠른 시간 안에 렌더링하기 위해 사용되는 기법이다. Doom이나, Daggerfall 같은 게임들은 적들을 2차원 스프라이트(sprite)로 그리는데, 이러한 경우에도 빌보드 기법을 이용해서 적들을 기하 모델링하는 폴리곤들이 항상 관측자를 바라보게 만듦으로써 더욱 효과적인 입체감을 지닐 수 있다[1]. 이러한 물체들은 여러 개의 다각형으로 모델링 하는 것보다 이미지를 매핑한 한 개의 다각형으로 물체를 생성하는 것이 이미지의 생성 속도와 사실성을 증가시킬 수 있다[2]. 빌보드는 복잡한 물체의 텍스처를 폴리곤에 매핑하고 매핑된 폴리곤이 항상 시점을 바라보도록 변환하는 기법이다. 나무와 같이 축을 중심으로 좌우 대칭인 물체의 경우는 축의 회전 변환만으로 어느 정도 사실적인 이미지를 생성할 수 있다[3].

기존의 빌보드 기법에서는 한 장의 텍스처 이미지만을 사용하여 장면을 구성하였다. 예를 들어 숲을 만들 경우 관측자가 보는 방향에 관계없이 한 종류의 나무만을 사용한다. 즉, 한 장의 텍스처 이미지만을 시점과 관측 방향에 관계없이 항상 시점만을 바라보도록 하여 텍스처 이미지가 매핑된 폴리곤이 관측방향과 수직이 되도록 하였다[4]. 이러한 이유로 기존의 빌보드 기법으로 생성한 장면은 사실성이 떨어지는 단점이 있다.

이 논문에서는 뷰 모핑 기법을 사용하여 시점에 따라 다른 텍스처 이미지를 사용하는 빌보드 기법을 제안한다. 텍스처 이미지를 빌보드 사각형에 매핑할 때, 관측 방향의 변화에 따라 뷰 모핑을 이용하여 적당한 이미지를 생성하여 사용함으로써 기존의 빌보드에서 물체의 모양이 모두 동일한 단점을 보완할 수 있다.

이 논문은 2장에서 기존의 빌보드 원리에 대해 설명하고, 3장에서는 뷰 모핑 기법, 4장에서는 뷰 모핑 기법을 사용한 빌보드 기법에 대해 논한다. 그리고, 5장에서는 구현 결과와 이 논문에서 제안한 방법과 기존의 빌보드 기법을 비교하고, 6장에서 결론을 내린다.

## 2. 기존의 빌보드 원리

여기에서는 나무와 같이 y축에 대칭이고 이 축을 중심으로 회전하는 간단한 경우에 대하여 알아본다. 이미지가 매핑되는 폴리곤을 빌보드라고 하는데, 이 폴리곤의 법선 벡터가 시점을 바라보도록 회전시키면 된다. 회전각은 <식 1>과 같이 표시된다.

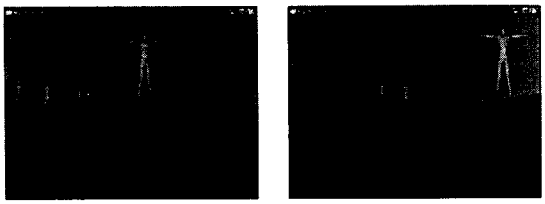
$$\theta = \pi - \cos^{-1}(L_{os} B_n) \qquad \text{<식 1>}$$

여기서 $B_n$은 빌보드의 법선 벡터이고 $L_{os}$는 시점에서 빌보드가 위치한 곳을 보는 관측 방향 벡터이다. 일단 회전각이 계산되면 <식 1>을 이용하여 y축을 중심으로 하는 회전행렬 $R$을 생성한다. 그리고, 해당 장면의 모델뷰 행렬 $M$과 결합시켜 조합행렬 $MR$을 생성한다. 해당 물체를 조합행렬 $MR$을 이용하여 빌보드를 기하변환한다.

그래픽스 API인 OpenGL을 이용하면 간단히 회전행렬을 구할 수 있다. 빌보드는 시점을 보아야 하기 때문에 기본적으로 빌보드의 법선 벡터를 관측자가 빌보드를 보는 방향과 역으로 보게 하면 된다. 따라서 관측행렬로 이루어지는 회전을 반대로 적용하면 된다. 적용 방법은 다음과 같다.

① 현재의 모델뷰행렬을 임의의 배열에 저장한다.
② 저장된 모델뷰행렬의 역행렬을 구한다.
③ 역행렬을 회전행렬로 사용한다.

일반적으로 역행렬을 구하는 것은 매우 복잡하나 행렬이 직교이면 역행렬이 원래 행렬의 전치행렬이 된다. 여기에서는 행렬이 직교이기 때문에 쉽게 역행렬을 구할 수 있다. 이러한 빌보드 기법으로 생성한 이미지를 [그림 1]에 표시한다.



[그림 1] 기존의 빌보드 기법

[그림 1]에서 보이는 각 오브젝트들은 기존의 빌보드 기법을 이용하여 생성한 것이다. 이 장면에서는 그래픽스 API인 OpenGL에서 제공하는 알파 블렌딩 함수와 안개 효과를 주어 멀리 보이는 빌보드 오브젝트(사람)들을 좀더 사실감 있게 표현하였다. 그러나 빌보드 오브젝트는 시점이 어디에 위치하든 모두 같은 모양인 단점이 있다.

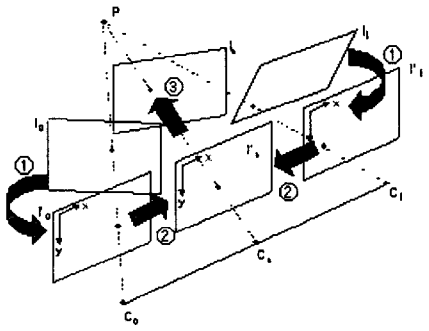
### 3. 뷰 모핑 기법

뷰 모핑 기법[5]은 일반적인 이미지 모핑 기법에서 물체의 위치나 시점의 변화를 반영하지 않아 포함하게 되는 부자연스러운 화상 왜곡(distortion)을 고치기 위한 기법이다.

뷰 모핑의 개요는 이미지 모핑을 하기 전에 먼저 두 이미지를 가상 관측면과 일치시키는 전변환(pre-warp) 처리를 거친 뒤, 이 변환된 이미지들을 이미지 모핑하여 중간 이미지를 생성하고, 이 중간 이미지를 원래 이미지의 관측면에 일치시키는 후변환(post-warp)처리로 구성된다. 모핑 처리에서는 여러 가지 이미지 모핑 기법을 사용할 수 있다. [그림 2]는 다음과 같이 3 단계로 처리되는 뷰모핑 기법을 보인다.

① Pre-warping
가상 뷰에 평행하게 원본 이미지들을 투영변환한다.
② Image Morphing
pre-warping 된 이미지들의 대응점들 사이를 선형보간한다.
③ post-warping
다시 역투영변환하여 결과 이미지의 생성한다.

동일 평면상에서의 이미지 변환에는 일반적인 어파인 변환이 그대로 적용되기 때문에 물체의 형태가 변하지 않는 성질을 이용하여 단계 ①에서 두 이미지들을 먼저 중간의 동일 평면으로 떨어뜨린 다음 두 이미지를 선형 이미지 모핑한다.



[그림 2] 뷰 모핑 기법

[그림 3]은 위에서 설명한 것같이 두 번의 회전을 거친 이미지들과 두 이미지 사이를 뷰 모핑한 전체 처리 과정의 도식이다. 먼저 이미지 상에 8개의 대응점을 이용하여 두 이미지 사이의 기본행렬 $F$를 구한다. 기본행렬 $F$와 두 이미지의 좌표점간의 관계식은 [그림 3]의 위쪽에 표시되어있다. 이미지를 중간 평면으로 떨어뜨리고자 하는 회전행렬은 평행한 두 뷰 사이의 기본행렬 $F_0$(<식 2>), 회전행렬의 모양, 에파이폴(epipole) 위치 변화의 성질을 이용하여 구한다.

$$F_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$$

<식 2> 두 뷰 사이의 기본 행렬 $F_0$

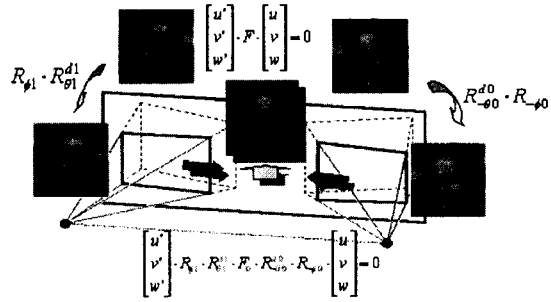
<식 3>과 <식 4>는 이미지의 평행 축에 대한 회전각 $\theta_i$와 이미지의 스캔라인의 평행을 맞추기 위한 회전각 $\pi_i$를 각각 보인다.

$$\theta_i = -\frac{\pi}{2} - \tan^{-1}\left(\frac{d_i^y e_i^x - d_i^x e_i^y}{e_i^z}\right)$$

<식 3> 평행축에 대한 회전각 $\theta_i$

$$\pi_i = -\tan^{-1}\frac{e_i^{-y}}{e_i^{-x}}$$

<식 4> 스캔라인의 평행을 맞추기 위한 회전각 $\pi_i$



[그림 3] 뷰 모핑의 전체 처리과정 개요

뷰모핑은 관측방향에 따라서 다른 이미지를 생성한다. 여기에 착안하여 기존의 빌보드 기법과 뷰 모핑을 접목시키면 빌보드 사각형이 지형 위에 랜덤하게 위치한 경우 항상 모든 텍스쳐 사각형이 동일 이미지가 아니라 관측자의 시선에 맞추어 텍스쳐 이미지를 뷰 모핑시켜 사용함으로서 훨씬 더 입체감 있는 장면을 생성할 수 있다. 다음장에서는 기존의 빌보드 기법과 뷰 모핑을 결합시키는 방법에 대해서 설명한다.

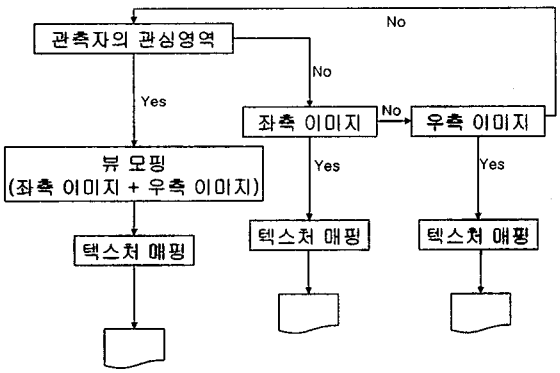
### 4. 뷰 모핑을 사용하는 빌보드 기법

이 기법은 빌보드 기법에 뷰모핑을 접목한 것으로 기존의 빌보드 기법에서 보아왔던 것과는 달리, 장면에 랜덤하게 위치해 있는 빌보드 위의 물체들이 항상 관측자에게 동일하게 보이는 것이 아니고, 장면에서 관측자의 관측영역 안에 위치한 빌보드를 관측영역 밖의 좌측과 우측에 위치한 빌보드 사각형에 매핑

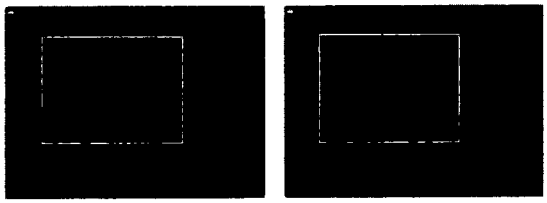
되어 있는 이미지를 뷰모핑으로 조작한 후 매핑하여 관측 방향에 따라 다르게 보이게 하는 기법이다. 장면에서 관측자의 관측 방향에 수직으로 위치한 빌보드 사각형의 텍스쳐 매핑 방법은 기존의 빌보드 기법을 그대로 사용하되, 관측자의 관심 영역 밖에 있는 빌보드 사각형에 매핑할 두 개의 이미지를 뷰 모핑하여 사용한다. 뷰 모핑을 사용하는 빌보드 기법은 다음과 같이 3단계로 처리된다.

① 관측자의 관심영역 밖의 빌보드사각형을 선택한다.
관측자의 관심영역 밖의 좌측과 우측의 빌보드사각형을 선택하여 적당한 텍스쳐 이미지를 매핑한다.
② 뷰 모핑으로 관심 영역안의 이미지를 구한다.
관측자의 관심영역 안에 있는 빌보드 이미지를 좌측과 우측의 빌보드 이미지를 사용하여 3 장에서 설명한 뷰 모핑 기법을 이용하여 구한다.
② 기존의 빌보드 기법 사용하여 텍스쳐 매핑한다.
기존의 빌보드 기법을 사용하여 ②에서 구한 이미지를 해당 빌보드 사각형에 텍스쳐 매핑한다.

[그림 4]는 뷰 모핑을 사용하는 빌보드 기법의 플로차트를 보인다. [그림 5]b 는 [그림 4]의 플로차트를 기초로 하여 기존의 빌보드 기법에 뷰 모핑을 접목하여 관측자의 관심영역 밖의 좌측과 우측에 매핑되어 있는 빌보드 사각형의 텍스쳐 이미지를 이용하여 관측자의 관심영역 안의 빌보드 사각형에 뷰 모핑 으로 구한 이미지를 텍스쳐 매핑하여 구현한 결과이다. 이 결과는 뷰 모핑을 이용한 빌보드 기법이 기존의 빌보드 기법보다 훨씬 더 입체감 있는 이미지를 생성함을 보인다.




[그림 4] 플로차트



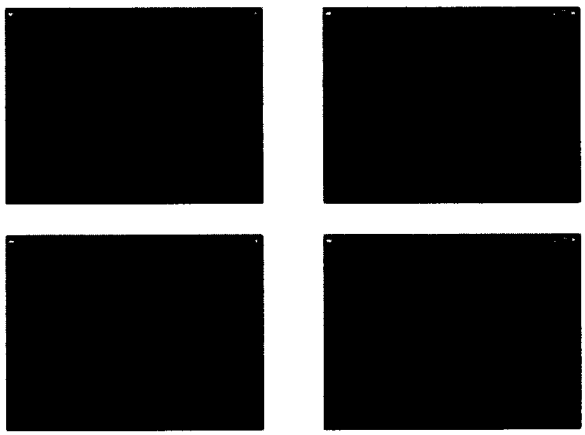
a. 기존 b. 제안한 기법

[그림 5] 기존의 빌보드 기법과 제안한 빌보드 기법

[그림 5]의 a와 b는 기존의 빌보드 기법과 제안한 기법으로 생성한 영상의 비교를 보인다. [그림 5]a 에서는 관측자의 관심영역 안 밖의 빌보드의 모양이 모두 동일하게 보이는 단점이 있다. 그러나 [그림 5]의 b는 관측자의 관심영역이 아닌 곳에 위치해 있는 서로 다른 이미지가 매핑되어 있는 빌보드사각형을 관측자의 관심영역 안에 있는 빌보드사각형에 뷰 모핑을 사용하여 매핑하여 기존의 빌보드 기법보다 다양한 장면을 생성할 수 있다.

## 5. 결론 및 구현 결과

이 논문에서는 뷰 모핑과 기존의 빌보드 기법을 결합시킨 빌보드 기법을 제안하였다. 이 기법은 기존의 빌보드 기법에서 보이는 빌보드사각형이 관측자의 관심영역 밖으로 벗어나 있어도 항상 관측자의 정면만을 바라보고 빌보드에 동일한 이미지를 사용하기 때문에 단조로운 영상을 생성한 것과는 다르게 관측자의 관심영역 밖에 있는 빌보드사각형을 뷰 모핑을 이용하여 관측 방향에 따라 다른 빌보드 이미지를 생성하여 빌보드에 사용하여 보다 사실적인 장면을 생성할 수 있게 하였다. 그리고, 제안하는 방법은 여러 장의 텍스쳐 이미지를 사용하지 않고 단 두 장의 텍스쳐 이미지만을 가지고 뷰 모핑을 사용하기 때문에 기존의 빌보드 기법과 같이 오버헤드와 메모리가 많이 소요되지 않는 장점이 있다. [그림 6]은 제안한 방법으로 구현한 결과 화면이다. 좌측 상단의 화면은 이 기법을 사용한 초기 화면이다. 우측 상단의 화면은 이 기법에서 관측자의 관심영역이 우측으로 이동한 화면이고, 우측 하단의 화면은 관측자의 관심영역이 좌측으로 이동한 화면이다.



[그림 6] 구현 결과

<참고문헌>

[1] Kevin Hwakins, and Dave Astle "OPENGL GAME PROGRAMMING", pp478, 2001

[2] http://www.lighthouse3d.com

[3] http://www.opengl.org

[4] Alan Watt, "3th Edition 3D Computer Graphics", ADDISON-WESLEY, pp235, 2000.

[5] S. M. Seitz and C. R. Dyer, View Morphing, SIGGRAPH 96, 21-30, 1996.