가상현실 장비를 이용한 캐릭터의 위치에 따른 카메라 추적 시뮬레이션

김용범○, 김은주
한림대학교 컴퓨터공학과
{stylemove○,ejkim628}@hallym.ac.kr

Camera Chase Simulation by Position of Character using Virtual Reality Equipment

Young-Bum Kim○, Eun Ju Kim
Dept. of Computer Engineering, Hallym Univ.

요 약

본 연구는 가상현실 장비인 트랙커와 HMD을 이용하여 가상 체험형 게임 제작시 유연한 비행 시뮬레이션이 가능하도록 카메라 이동을 조작하여 몰입감을 증대하고자 한다. 카메라의 조작을 위하여 Eular 공식을 적용하였으며, 회전의 효과는 사용자가 하늘을 나는 듯한 느낌과 비행할 때 느낄 수 있는 매스꺼움 현상을 경험할 수 있게 한다.

1. 서론

영화 분야를 중심으로 발전해 온 가상현실 기술은 입체 영상 장비를 사용하지 않는 단순 3차원 입체 영화의 시작으로 1833년 찰스 휘스턴이 고안한 입체 거울을 근간으로 현재 HMD(Head Mounted Display)[1]의 원조격인 디스플레이 장치를 개발하면서 본격화 되었다[2]. 이를 시점으로 다양한 가상 장비들이 발달 되었으며 기존의 MMORPG(Massive Multiplayer Online Role Playing Game), FPS(First Person Shooting) 게임에서 느낄 수 있는 몰입감 있는 게임의 개발을 목적으로 가상현실 장비를 이용하고 있다.

본 논문에서는 여러 가상현실 장비 중 인간이 시각을 통해 외부 세계로부터 정보를 입수 할 비중이 다른 감각들(청각, 후각, 미각, 촉각)에 비해 크다는 특성[3]을 이용, HMD와 트랙커를 이용하여 유연한 카메라 이동을 구현하고, MultiTexture, ColorIndexBuffer, UVAnimainon 등 다양한 그래픽 기술을 모듈화를 하여 사실적인 가상환경을 구현하는데 있다. 또한 트랙커를 이용한 가상 환경에서 사용자가 하늘을 나는 듯한 느낌과 비행시 느끼는 매스꺼움을 카메라 이동에 의하여 경험 하고자 한다.

2. 관련연구

게임 산업이 발달됨으로써 사실적인 게임에 초점을 맞추어 가상현실 게임에 대한 연구가 진행되고 있다[4][5]. Tracker를 이용한 가상현실 게임의 경우 위치 정보를 토대로 물리 엔진을 캐릭터에 적용하여 가상 환경에서 사용자의 움직임을 사실적으로 표현하고자 한다[6]. 하지만 카메라 이동에 의한 구현이 유연하지 않기 때문에 사실적인 묘사가 힘들다. 따라서 본 논문은 가상현실 장비인 트랙커를 이용하여 카메라 이동에 의한 가상 환경의 사실적인 묘사를 위하여, 트랙커를 이용하여 추출할 수 있는 각 축에 대한 회전각을 얻어 유연한 비행 시뮬레이션이 되도록 한다.

3. 본론

3.1 가상 환경 구축

그림 1은 가상환경에서의 시뮬레이션과 게임을 위한 기본 환경 시스템이다.

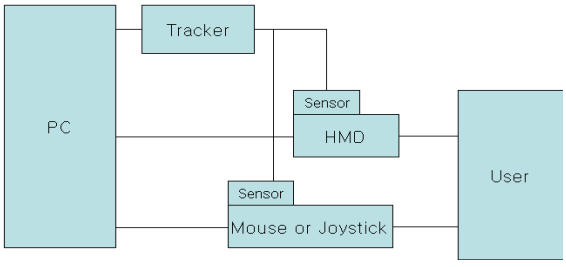

그림 1. Composition of virtual reality

마우스 및 키보드 입력을 통하여 가상의 캐릭터를 제어하는 것이 아니라 트랙커를 이용하여 사용자의 움직임을 추적하여 게임상의 사상된 캐릭터를 제어함으로써 가상환경에서 마치 자신이 이동하는 착각을 느끼게 된다. 그림 1은 트랙커의 기준이 되는 축 Transmiter를 중심으로 HMD와 Joystick에 각각 Receiver를 부착하여 사용자에 대한 움직임을 추적한다. HMD에 부착된 Receiver는 사용자의 시선에 대한 회전값을 결정하고, Joystick에 부착된 Receiver는 사용자의 이동을 결정하는데 사용한다. 이때 HMD에 부착된 Receiver를 이용하여 이동과 회전을 결정할 때 따르는 사용자의 이동에 대한 공간의 제약을 Joystick에 Receiver를 하나 더 부착하여 해결한다.

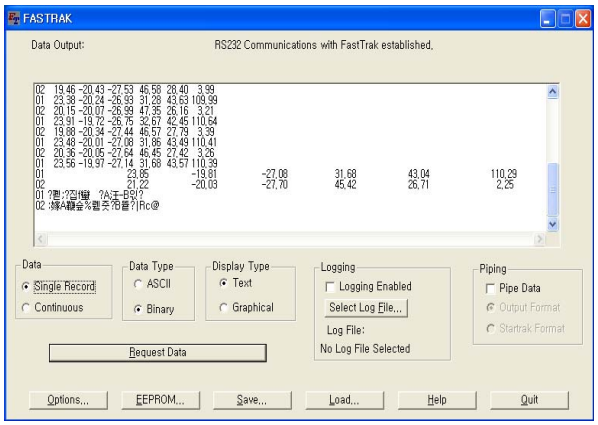

그림 2. Tracker`s output data

트랙커에서 출력하는 데이터는 앞 열을 기준으로 Receiver번호, x, y, z position , Azimuth, Elevation, Roll 값을 나타낸다. 그림 2는 이와 같은 입력을 2개의 Receiver를 통하여 받는 과정을 보여 준다

3.2 카메라 이동

OpenGL 시스템에서 유연한 카메라를 구현하기 위해 gluLookat() 함수와 glGetMatrixf() 함수를 이용한다. 현재 매트릭스의 3x3 행렬을 이용하여 4원수(Quaternion)로 변환한 후, x, y, z, w의 값을 구한다. 다음 Q=cos(2/θ)+(x, y,z) sin(2/θ)에서 θ(theta)를 이용하여 glRotate(x, y, z,θ)를 계산한다. 이동 함수인 glTranselatef(tx, ty, tz)를 호출한 후 회전 하여 렌더링을 한다. 이와 같은 복잡한 구조를 보안하기 위하여 Eular변환[7]을 사용하여 트랙커에서 얻어진 데이터를 처리하여 카메라 클래스를 완성하였다.

Eular 변환은 수학자 Leonard Euler(1707-1783)의 이름에서 유래되어 자신(카메라)이 임의의 다른 객체의 방향을 정하는 행렬을 만들어내는 방법을 제안했다.

$$E(h,p,r) = Rz(r)*Rx(p)*Ry(h)$$

그림 3. Conversion formula of Euler

그림 3의 수식은 그림 4를 통하여 그 값을 확인할 수 있다. 그림 4에서 머리는 y축 방향을 가리키고, 시선은 음의 z축 방향, 머리를 좌우로 돌리는 head(도리각) h, 끄덕이는 pitch(끄덕각) p , 갸웃거리는 roll(갸웃각) r으로 구성된다. 이들 회전 행렬들을 결합하면 그림 5의 수식이 완성된다.

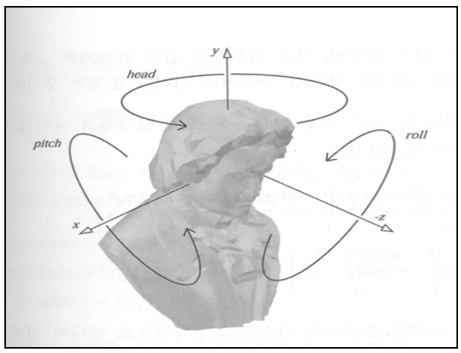

그림 4. Eular Angle

$$F = \begin{pmatrix} \cos r\cos h - \sin r\sin p\sin h & -\sin r\cos p & \cos r\sin h + \sin r\sin p\cos h \\ \sin r\cos h + \cos r\sin p\sin h & \cos r\cos p & \sin r\sin h - \cos r\sin p\cos h \\ -\cos p\sin h & \sin p & \cos p\cos h \end{pmatrix}$$

그림 5. Euler's formula

```
void update(float timeDelta) {
        tVector3 c;
        if(fttest.IsStartContinuous()) {
                unsigned int size=fttest.ReadContinuous(ft_data);
                c.x = ft_data[0].x - ft_tmp.x;c.y = ft_data[0].y - ft_tmp.y;
                .....
                ft_tmp2.z = ft_data[0].azimuth / 3.14;
                objv = objv+c*10;
        }
        float angSpeed = 0.05;
        float newAngSpeed = angSpeed * timeDelta;
        Matrix rotmat(objangle.x,objangle.y,objangle.z);
        viewmat = rotmat*viewmat;
        objangle = objangle*expf(-2.0f*timeDelta);
        tVector3        sideVec(viewmat.p[0], viewmat.p[1], viewmat.p[2]);
        tVector3        upVec(viewmat.p[3], viewmat.p[4], viewmat.p[5]);
        tVector3        dirVec(viewmat.p[6], viewmat.p[7], viewmat.p[8]);
        objlook = dirVec;    objright = sideVec;  objup = upVec;
        campos = (objpos   -objlook*3 );
        float moveSpeed = 5;
        objpos = objpos+objv*timeDelta;
       objv = objv*expf(-2.0f*timeDelta);
float m[16]={viewmat.p[0],viewmat.p[3],-viewmat.p[6],0,
                viewmat.p[1],viewmat.p[4],-viewmat.p[7],0,
                 viewmat.p[2],viewmat.p[5],-viewmat.p[8],0,
-(viewmat.p[0]*campos.x+viewmat.p[1]*campos.y+
viewmat.p[2]*campos.z),-(viewmat.p[3]*campos.x + viewmat.p[4]*campos.y
+ viewmat.p[5]*campos.z),
 (viewmat.p[6]*campos.x+viewmat.p[7]*campos.y+
viewmat.p[8]*campos.z),1
};
        glMultMatrixf(m);
}
```

그림 6. Camera Update Source

그림 5의 식을 이용하여 매 프레임마다 카메라의 뷰 행렬과 캐릭터의 위치를 결정한다. 입력 값은 ReadContinuous() 함수를 이용하여 트랙커의 위치값(x, y, z)과 회전각(pitch, roll, head)를 받아온다. 이 때 트랙커에서 얻어진 회전각을 이용하여 회전 행렬 rot-mat() 값을 구한다. 회전 행렬과 이전의 뷰 행렬을 곱해서 뷰 변환을 하고, 뷰 변환 이후 그림 4의 x, y, -z의 벡터를 siedeVec(), upVex(), dirVex()함수를 통하여 값을 얻는다. 캐릭터의 위치와 회전이 결정되면 위 카메라의 위치를 캐릭터가 보는 lookvector 방향의 뒤로 빼서 카메라가 캐릭터를 추적할 수 있게 한다. 최종적으로 그림 5의 식을 완성하여 카메라의 회전과 이동을 결정한다. 그림 6은 위에서 설명한 내용을 구현한 것이다.

3.3 카메라 추적을 위한 시스템 구성도

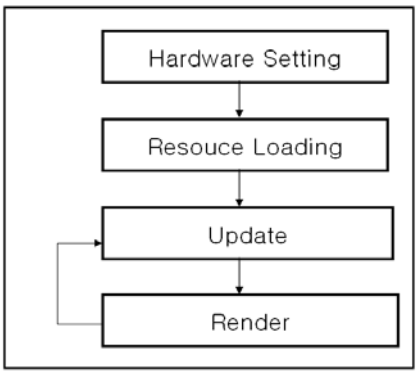

그림 7. Block diagram of presented system

(1) HardWare Setting

가상현실 어플리케이션은 현실과 가까운 사실적인 묘사가 필요하기 때문에 이를 뒷받침 해 줄만한 그래픽 카드가 필요하다. 사용자의 그래픽 카드 사양에 초점을 맞추어 프로그램을 개발하게 되면 컴퓨터의 사양에 종속 되어진다. 멀티텍스처 등과 같은 그래픽 렌더링 기술이 지원이 되는지 여부, 가상현실 장비가 연동되어 있는지 여부, 풀 스크린의 여부 등을 이 과정에서 처리 한다.

```
void checkHardware(){
        if (!isExtensionSupported("GL_ARB_multitexture"))
            chkFlag = NOT_SOPPORT_MTEXTURE;
        glActiveTextureARB = (PFNGLCLIENTACTIVETEXTUREARBPROC)
wglGetProcAddress("glActiveTextureARB");
        glMultiTexCoord2fARB = (PFNGLMULTITEXCOORD2FARBPROC)
wglGetProcAddress("glMultiTexCoord2fARB");
        glClientActiveTextureARB = (PFNGLACTIVETEXTUREARBPROC)
wglGetProcAddress("glClientActiveTextureARB");
        if (!isExtensionSupported("GL_EXT_compiled_vertex_array"))
              chkFlag = NOT_SOPPORT_V_ARRAY;
        ......
}
```

그림 8. Source Code of Check Graphic Card

그림 8에서 isExtensionSupported()함수를 이용하여 GL_ARB_multitexture 상수를 통해 다중 텍스처링을 지원하는지 판단한다. 이 때 다중 텍스처링을 지원하게 되면 다중 텍스처링을 위한 텍스처 좌표를 지정하고, glActiveTextureARB 텍스처 단위를 설정한다. 그리고 glClientActiveTexture 정점 배열을 위한 텍스처 단위를 활성화 한다. GL_EXT_compiled_vertex_array는 버텍스 배열이 지원되는지의 여부를 확인한다.

(2) Resource Loading

그래픽 리소스와 사운드 리소스, 트랙커, 조이스틱과 같은 객체들을 로딩하여 초기화를 한다. 지형, 물, 스카이박스를 하나의 클래스로 통합하여 관리함으로써 원하는 지형을 손쉽게 생성하도록 하였다. 이들 모두 멀티텍스처를 이용한 디테일 맵과 UV Animation을 이용하여 사실적인 묘사가 되도록 하였다.

(3) Update

캐릭터의 시점 변환과 오브젝트들(지형, 적, 건물)과의 충돌 체크, 빌보드 계산, 시간의 변경에 따른 텍스처의 UV좌표 변화, 반사 효과, 빛에 따른 조명 효과 등을 처리한다.

(4) Render

Update 단계까지 처리된 그래픽 요소들을 렌더링하고, 사운드를 출력한다.

5. 구현

그림 9. Screen of when move on the ground

그림 10. Screen of when fly sky

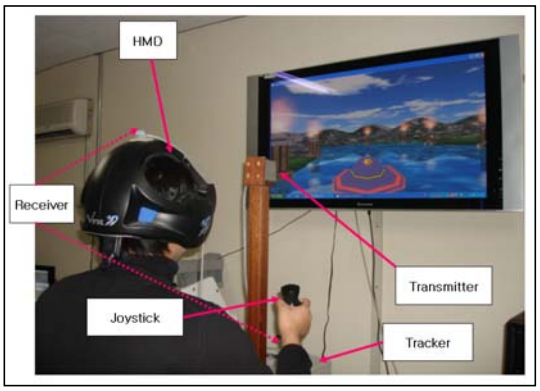

그림 11. Operation view of tracker

그림 11에서 보이는 것처럼 가상의 환경 시스템을 이용하여 캐릭터가 지상을 이동할 때의 모습(그림 9)과 비행 할 때의 모습(그림 10)을 시뮬레이션 하였다. 현재 자신의 움직임을 외부에서 확인하기 위해 3인칭 캐릭터를 이용하였고, 캐릭터의 움직임에 따라 카메라 추적을 적용하여 실세계와 비슷한 매스꺼움의 현상을 확인하였다.

6. 결론

본 연구는 가상현실 장비인 트랙커와 HMD을 이용하여 가상 체험형 게임 제작시 유연한 비행 시뮬레이션이 가능하도록 카메라 이동을 이용하여 확인하고자 하였다. 카메라의 유연한 이동을 통하여 사용자는 하늘을 나는 듯한 느낌과 비행할 때 느낄 수 있는 매스꺼움 현상을 경험할 수 있었다. 이와 같은 효과는 사람이 시각에 민감한 특성을 이용하여 몰입감을 증대 시킬 수 있지만 아직 현실과 가까운 경험은 많이 부족하다. 따라서 카메라의 이동에 위한 시각적인 효과와 함께 사물에 입력을 가하였을 때 그 사물이 실세계와 비슷한 물리적인 요소를 고려하여 반응을 하고, 더 나아가 Haptic 같은 사물의 질감을 느낄 수 있는 장비들을 추가로 연동한다면 좀 더 현실과 가까운 경험을 하게 될 것이다.

ACKNOWLEDGEMENT
본 연구는 2006년 지방대학 혁신역량 강화사업(NURI) 문화 콘텐츠(CT) 인력양성 3차년도 사업의 연구비 지원으로 수행되었습니다.

[참고문헌]

[1] ITS Corp., http://www.razor3d.co.kr/
[2] Vr-Panorama Corp., http://vr-panorama.co.kr/
[3] 오문석, 이재익,"가상현실 게임 디자인을 위한 입체영상 제작기업에 관한 연구," 인터넷 온라인 게임의 품질평가에 관한 연구, 디지털디자인학 연구 , *KCI Vol11* 권 제 2호, 2005년 6월
[4] 전계범, 김은주, 송창근, "체감형 스노우보드 시뮬레이션 게임을 위한 지형 엔진 연구,"한국멀티미디어학회 춘계학술발표대회 논문집, pp. 161-164, 2006
[5] 김은주, 안상혁, 송창근, 김선정, "골프 스윙 센서를 이용한 체감형 골프 게임," 한국정보처리학회 추계학술발표대회 논문집, 제13권 제2호, pp. 175-178, 2006
[6] 김원철, 김은주, 송창근, "3D 스노우보드 게임을 위한 캐릭터 이동에 관한 연구,"한국멀티미디어학회 춘계학술발표대회 논문집, pp. 513-516, 2006
[7]Shoemake, Ken,"Euler angle Conversion," in Paul S.Heckbert, ed., Graphics Gems IV, Academic Press , pp. 207-221, 1994.