Patriot Tracking Device를 이용한 가상현실 엔진 구현

김은주*, 이용욱*, 송창근*
*한림대학교 컴퓨터공학과
e-mail : {ejkim628, tturtle, cgsong}@hallym.ac.kr

Implementation of Virtual Reality Engine Using Patriot Tracking Device

Eun-Ju Kim*, Yong-Woog Lee*, Chang-Geun Song*
*Dept. of Computer Engineering, Hallym University

요 약

본 연구는 개인용 PC 에 장착할 수 있는 저가의 가상현실게임 엔진을 설계하고 구현한다. 가상현실 엔진구현에서는 주요한 입출력 장치인 Tracker 와 HMD(Head Mounted Display) 그리고 조이스틱과 마우스의 장착이 필수적이다. 가상현실 엔진을 연동하기 위한 입출력 클래스를 설계하고 입력장치로 마우스와 조이스틱, 출력장치로 HMD 를 장착하였으며 Tracker 의 구현은 상업용 제품인 Polhemus 의 Patriot Tracker 를 이용하였다.

1. 서론

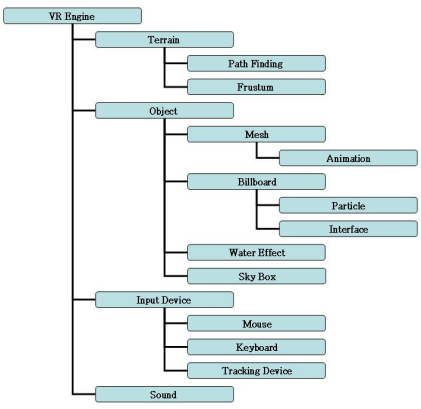
최근 들어서 가상현실 엔진과 게임 엔진들이 늘어나고 있다. 두 가지 엔진 모두 최신의 기술들을 이용해 높은 Quality 그래픽을 제공하여 사용자에게 가상의 3D 공간에서 현실과 같은 수준의 체험을 할 수 있게 하는 것에 중점을 두고 있다는 공통점이 있다. 하지만 서로 다른 사용자 인터페이스의 차이를 보이고 있다. 즉, 게임 엔진은 보통 마우스, 키보드, 조이스틱의 사용자 인터페이스를 제공하지만 가상현실 엔진은 HMD(Head Mounted Display), Tracker, Glove, 3D-Mouse 등의 다양한 입출력 장치들을 이용한 사용자 인터페이스를 제공한다는 것이다. 본 논문은 가상현실을 적용한 엔진을 개발하면서 가상현실 엔진에 필수적 요소인 사용자가 가상환경에 쉽고 편리하게 접근 할 수 있는 인터페이스를 제공 할 수 있도록 HMD 와 Tracker 장비를 이용하여 가상현실 엔진을 구현하였다

가상현실 엔진 분석과 예제프로그램, Tracking Device 의 구성과 사용방법, HMD 와 Tracker 의 연동, 마지막으로 결과와 향후 연구과제에 대해서 기술한다.

본 연구는 2005 년 한국산업기술재단(KOTEF)의 연구비 지원으로 수행되었습니다.

2. 가상현실 엔진 설계

가상현실 엔진은 DirectX 를 기반으로 제작 되었고 가상환경을 구성하기 위한 지형, 객체, 사운드 등의 기본적인 입출력 기능들을 제공하고 있다. 가상현실 엔진의 대략적인 구성은 (그림 1)과 같다.



(그림 1) 가상현실 엔진 구성

<표 1> 엔진 기능

구분	구성요소	내 용
Terrain	Terrain	HeightMap 지형을 생성, 관리
	PathFinder	A* 알고리즘을 이용한 길 찾기
	Frustum	QuadTree 알고리즘을 이용해 렌더링 최적화
Object	Mesh	X 파일을 통해 3D - Mesh 를 로딩, 관리
	Animation	Mesh 의 애니메이션 기능을 관리
	Billboard	빌보드
	Particle	빌보드를 이용한 파티클 효과
	Interface	빌보드를 이용한 2D 인터페이스
	Water	Terrain 에 수면 효과를 적용
	SkyBox	스카이 박스
Input Device	Mouse	마우스를 통한 입력
	Keyboard	키보드를 통한 입력
	Tracker	Tracking Device 로 부터의 입력
Etc	Camera	Direct3D 카메라
	Sound	사운드 파일을 로딩하여 출력

2.1 엔진 분석

<표 1>은 엔진 기능을 구분하여 정리한 내용이다.

2.1.1 Terrain

Terrain 은 HeightMap 을 이용한 지형 엔진이다. LightMap 이라는 이미지 파일을 불러들여 각 픽셀들의 명암을 추출하고 명암의 강도에 따라 지형의 높낮이를 결정해 지형을 생성하는 기능을 한다.

Frustum 은 카메라의 View, Projection Matrix 를 이용해 지형이 현재 사용자의 시야에 들어오면 출력하고, 들어오지 않으면 출력하지 않도록 하는데 이때 시야에 들어오는지를 판별하기 위해 QuadTree 알고리즘을 사용하여 렌더링시에 불필요한 부분을 제외하여 최적화를 하게 된다.

2.1.2 Object

오브젝트들은 지형 이외에 가상환경을 꾸밀 수 있는 도구들이다. 크게는 Mesh 와 Billboard 가 있는데 Mesh 는 3D-MAX 나 마야 등의 3D 그래픽 저작도구를 이용해 제작된 객체들을 불러들일 수가 있으며 Mesh Animation 을 통해 그 객체들이 가지고 있는 애니메이션 정보를 추출하여 실제 프로그램상에서 제작할 때와 같은 애니메이션을 볼 수 있도록 해준다.

2.1.3 InputDevice

입력장치는 프로그램과 사용자의 상호작용에 있어 가장 중요한 요소이다. 기본적으로 컴퓨터가 가지고 있는 입력장치는 마우스와 키보드인데 DirectInput 과 Window Message Procedure 를 이용해 입력을 받을 수가 있다. 그 외에 가상현실 입력장치인 Tracking Device 를 지원하기 위해 Tracker 라는 클래스가 추가 되었다.

2.1.4 Etc

입력장치와 객체들 이외에 사운드를 출력하기 위한 Sound 클래스나 사용자가 가상환경 상에서 바라보는 시점을 뜻하는 Camera 클래스 등이 속한다.

Sound 는 DirectSound 를 이용해 각종 WAV, MIDI, WMA 등의 사운드 파일을 프로그램 실행시에 연주 할 수 있는 기능이며 Camera 는 Direct-X 의 View, Projection 행렬을 이용해 사용자의 위치, 보는 방향, 회전 등을 표현한다. Camera 는 사용자의 시점을 표현하는 이외에도 Terrain 의 Frustum 이나 Billboard 의 방향을 결정하기 위해서도 쓰이는 중요한 도구이다.

3. Tracking Device

Tracking Device(이하 트랙커)는 센서에서 취합되는 도구의 상대적 위치, 회전 값을 반환해 주는 장비로 가상현실에서 사용자의 행동을 알아내는데 사용된다. 본 논문에서는 Polhemus 사의 Patriot 트랙커를 이용하여 가상환경에 적용하였다.

3.1 Tracker Class

3.1.1 Tracker Class 기능

Patriot 의 SDK 기능들을 하나의 클래스로 캡슐화 하였다. 다른 회사 제품의 트랙커로의 확장을 위해 대표적인 기능들을 위주로 멤버, 함수들을 구성하였다. <표 2>는 클래스 멤버 함수의 기능을 정리한 것이다.

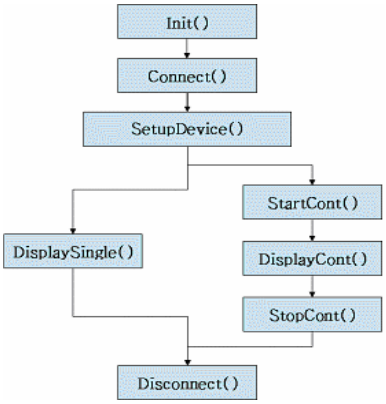
<표 2> Tracker Class 멤버함수

자료형	명 칭	설 명
Bool	Init()	Device 를 사용하기 위한 장치 디바이스, 데이터 클래스 등을 초기화
Bool	Connect()	Device 에 연결
Void	Disconnect()	Device 와의 연결을 해제
Bool	SetupDevice()	Device 에 대한 설정
Void	DisplaySingle()	Device 에서 순간 값을 읽어 들인다.
Bool	StartCont()	Device 에 실시간으로 값을 읽기 시작하겠다는 명령을 내린다.
Bool	StopCont()	실시간으로 값을 읽는 것을 중단하겠다는 명령을 내린다.
Void	DisplayCont()	실시간으로 측정되는 값을 읽어 들인다.
Void	ConvertDataFrame()	값을 변환한다.

ConvertDataFrame()은 트랙커가 넘겨준 정보를 통일된 방식으로 변환하여 주는 함수이다. 트랙커들마다 정보를 구조체, 바이너리, 텍스트 등 다양한 방식으로 반환하기 때문에 클래스 외부에서 통일된 방식으로 접근하기 위해 필요하다. 트랙커 장비가 바뀌게 된다면 본 클래스의 각 멤버함수에 해당하는 내용과 ConvertDataFrame() 함수의 내용을 일부 또는 전체를 수정하여 준다면 손 쉽게 재사용할 수가 있다.

3.1.2 Tracker flow

트랙커에서 사용상의 흐름은 (그림 2)과 같다.



(그림 2) Tracker flow

트랙커를 이용하기 위해선 Init()과 Connect(), SetupDevice() 함수들을 이용해 Tracking Device 에 연결하여 값을 받아들일 준비 과정을 마치고 프로그램의 수행시간 동안 실시간으로 DisplaySingle()함수나 DisplayCont()함수를 이용하여 Tracking 정보를 읽어 들인다.

DisplaySingle()함수는 별도의 준비 과정 없이 정보를 읽어 올 수 있지만, DisplayCont()함수는 StartCont()함수로 정보를 읽기 시작하겠다는 신호를 Device 에게 알린 후에 사용 할 수 있다.

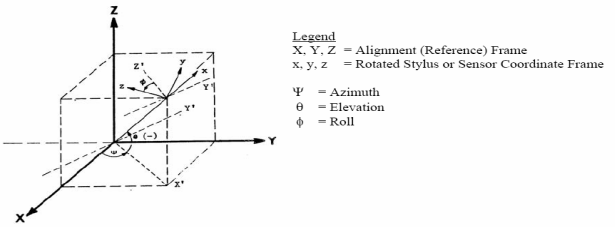
DisplayCont()함수의 사용을 중지하기 위해서는 StopCont()함수를 이용해 Device 에게 정보 읽기의 끝을 알려주어야 한다. 사용상에는 별도로 준비 과정을 거치는 DisplayCont()함수보다 정보가 필요한 즉시 사용 할 수 있는 DisplaySingle()함수가 편리하지만 지속적으로 정보를 필요로 하는 경우에는 DisplaySingle()함수보다 DisplayCont()함수가 더욱 월등한 프레임율을 유지하기 때문에 DisplayCont()함수를 사용하는 것이 실시간으로 실행되는 프로그램에 부담을 줄여준다.

정보를 읽어 들인 후에는 ConvertDataFrame() 함수를 통해 버퍼에 저장된 정보를 X, Y, Z, Azimuth, Elevation, Roll 의 의미 있는 정보로 변환하여 클래스 외부에서 손 쉽게 접근 할 수 있도록 별도의 멤버 수에 저장하여 주는 작업을 거쳐야 한다.

Device 의 사용이 끝나면 Disconnect()함수를 호출하여 장치와의 연결을 해제하여야 한다.

3.1.3 트랙커가 반환하는 정보

트랙커를 가상현실 엔진상에서 사용하기 위해서는 각 값에 대한 인지가 필요하다. 트랙커가 반환하는 값은 X, Y, Z, Azimuth, Elevation, Roll 이 있고, 각 값에 대한 내용은 다음과 같다.



(그림 3) 트랙커 반환 값

(그림 3)에서 X, Y, Z 좌표는 Source 의 위치에 따른 Sensor 의 3D 위치 값이고, Azimuth 는 Z 축을 중심으로 한 X,Y 축의 변환 각, 그리고 Elevation 은 변환된 Y' 축을 중심으로 X' ,Z 축의 변환 각이며 Roll 은 변환된 X' 축을 중심으로 Y' ,Z' 축의 변환 각이다.

3.1.4 엔진의 Camera 클래스에 트랙커 값을 적용

트랙커가 반환하는 값을 알아냈으면 그것을 가상현실 엔진에 적용시켜야 한다. 간단한 예로 가상환경에서의 사용자의 뷰 즉, 카메라에 적용 시키는 방법을 보도록 한다. 가상현실 엔진은 Direct3D 로 제작되었다. Direct3D 에서 카메라는 다음과 같은 정보를 필요로 한다.

Eye 은 카메라의 위치를 나타내는 위치 벡터이고, Lookat 은 카메라가 바라보고 있는 목적지의 위치 벡터 그리고 Up 은 카메라의 위쪽이 어느 방향인지에 대한 방향 벡터이다. 기본적으로 장비를 통해 얻은 X, Y, Z 값을 사용하지만 Y 의 경우 지형의 높낮이에 따라 변하는 기본적인 시점의 높이 값을 더하고 사용자의 눈이 지형보다 높은 곳에 있다는 것을 감안하여 추가적으로 대략적인 눈의 높이를 한 번 더 더하였다. Z 에 -1 을 곱한 것은 트랙커는 오른손 좌표계이고, Direct3D 는 왼손 좌표계라는 차이 때문이다.

```
m_vEye.x=pdi->m_X;

m_vEye.y=m_TerrainHeight+m_SkyHeight+30.f;

m_vEye.z=pdi->m_Z * -1;
```

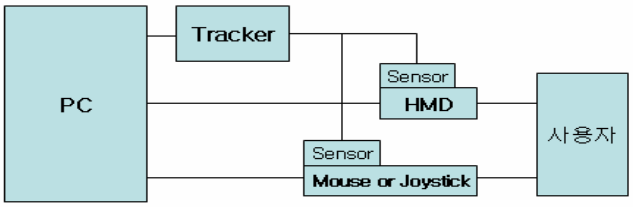
(그림 4) Eye(m_vEye)를 구하는 예

(그림 4)에서 구현한 m_vEye 에서 장비의 Azimuth, Elevation 값을 이용해 카메라의 방향 변환 매트릭스를 구한 다음 그것을 이용해 방향 벡터를 구하고, 그 값에 m_vEye 벡터를 더하여 카메라의 위치, 방향 즉 Lookat 값을 구할 수 있다. Azimuth 와 Elevation 는 각도를 뜻하기 때문에 값을 사용할 때는 라디언 값으로 변환하여 주어야 한다. 회전 행렬을 곱할 때는 순서가 중요하다. 장치 값의 변환 순서상 Azimuth 의 변환 후 Elevation 이 되기 때문에 회전 행렬은 Elevation 에 Azimuth 의 회전값을 곱해야만 제대로 된 결과를 얻어 낼 수가 있다. 그런 과정을 위해 matTmp 를 이용 하여 matRot 에 곱해주는 방식을 이용 하였다. 그리고 그 회전행렬과 기본으로 정의한 노멀 벡터를 곱하여 현재 장치에서 얻어 온 사용자의 시점 방향 벡터를 얻어내고 그 값에 앞에서 구한 사용자의 위치 벡터(Eye)를 더하여 사용자의 위치와 보는 방향을 계산하였다.(그림 5)은 Lookat 을 구현한 것이다.

```
D3DXMatrixRotationX( &matTmp,
        (pdi->m_Elevation) * DEG_TO_RAD);
matRot=matTmp;
D3DXMatrixRotationY( &matTmp,
        (pdi->m_Azimuth) * DEG_TO_RAD);
matRot*=matTmp;
…
D3DXVec3TransformCoord(&m_vTarget,
                        &vLook,&matRot);
m_vTarget+=m_vEye;
```

(그림 5) Lookat(m_vTarget)을 구하는 소스 예

4. HMD 와 Tracker 의 연동

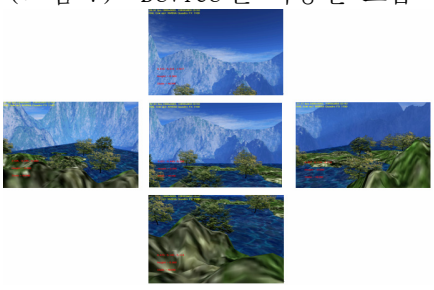


(그림 6) HMD 와 Tracker 의 연동 시스템 구성도

(그림 6)은 가상현실 시스템의 구성도이다. 사용자는 PC 에 연결된 HMD 를 통해 가상환경을 볼 수 있고, 마우스나 조이스틱으로 제어를 할 수 있다. HMD 는 Tracker 의 센서와 연결되어 사용자의 머리가 움직임에 따라 자동으로 화면 전환이 이루어지고, Mouse 역시 Tracker 의 센서가 연결되어 가상공간에 적용된다.

(그림 7)과 (그림 8)는 가상현실 엔진을 이용해 간단한 가상 환경을 만들어 사용자가 Tracker 가 부착된 HMD 를 착용하여 가상환경을 체험하는 모습니다.

(그림 7) Device 를 착용한 모습



(그림 8) 가상환경에서의 상,하,좌,우 시점변환

5. 결과 및 향후 연구 과제

현재의 환경은 간단한 HeightMap 의 지형에 Billboard 로 세워진 나무와 풀, SkyBox 등으로 구성되어 있고, 센서가 부착된 HMD 를 착용하여 고개를 움직이면 Tracker 는 사용자의 고개의 방향과 위치를 추적하여 Direct3D 의 카메라를 제어하게 된다. 그리고 사용자는 마치 자신이 가상의 실제 환경상에 있는 것처럼 자유로이 가상환경을 탐색 할 수가 있다.

본 연구를 통하여 4 가지의 향후 연구과제를 제시한다. 첫째로 다양한 Tracking Device 를 장착할 수 있도록 개선해야 한다. 본 연구는 Patriot 를 대상으로 Direct3D 의 Camera 를 구현하여 보았지만 Tracking Device 는 Patriot 외에도 다양한 종류가 있다. 현재 또 다른 Tracking 장비인 Polhemus 사의 Fastrak 까지 구현이 된 상태이지만 이외에 다른 장비들도 장착 할 수 있도록 해야 한다. 둘째로 입력방식의 개선에 대한 연구가 필요하다. Tracking Device 의 Sensor 를 이용한 입력 방식에는 많은 제약이 있다. 그렇다고 고가의 기타 장비들을 구입하기보다는 Sensor 를 최대한 활용하여 다양한 행동을 할 수 있는 User Interface 를 연구해야 한다. 셋째로 보다 현실적인 가상환경을 제작해야 한다. 가상현실 엔진을 이용해 단시간에 가상환경을 제작하였지만 아직 단조로워 보인다. 가상현실 엔진의 기능을 최대한 활용하여 더욱 사실적이고 몰입감 있는 가상환경을 제작하여야 한다. 넷째로 여러 응용분야의 연구가 확대 되어야 한다. 지금까지 연구한 것은 가상환경을 설정하는 것이었지만 앞으로는 가상환경과 관련된 응용분야를 연구해야 한다.

참고문헌

[1] DirectX C++ SDK Manual ,Microsoft
[2] Patriot User Manual ,Polhemus
[3] Polhemus Tracker SDK v2.1.3 ,Polhemus
[4] IT EXPERT 3D 게임 프로그래밍 ,한빛미디어 , 김용준
[5] DirectX 9 를 이용한 3D 게임 프로그래밍 입문 , 정보문화사, Frank D,Luna 최현호 역,