論文

3차원 지형정보를 활용한 조종석 디스플레이

김인중*, 황호연**, 박성수**

Cockpit Display using 3D Geographic Information

In-Joong Kim*, Ho-Yon Hwang**, Sungsu Park**

ABSTRACT

In this research, a cockpit display system using 3 dimensional geographic information was developed. A coordinate transformation method from WGS84 to TM was first studied. Octree data structure was used for efficient 2D and 3D graphic display. Also, a 3D graphic engine was developed for fast display with large amount of geographic data which can be practically used in aircraft onboard computer having low performance. This 3D engine contains additional function such as geographic and object information loading, many kinds of camera mode, aircraft position and rotation control function, character strip display. In the future research, actual GPS signal should be used and additional information that can help pilot's decision should be display.

Key Words : Cockpit Display(조종석 표시판), Navigation System(항법시스템), GPS(위성측위시스템), Geographic Information(지리정보), Graphic Engine(그래픽 엔진)

1. 서 론

항공기에 장착된 여러 가지의 전자화된 계기들은 조종사들에게 예전의 다이얼이나 게이지 및 인디케이터 등을 이용한 계기들에 비해 보다 조종하기에 편한 환경을 제공하며, 나쁜 기상조건 하에서도 안전한 비행을 할 수 있도록 많은 도움을 준다. 하지만 대부분의 조종석 디스플레이 시스템(Cockpit Display System)은 근본적으로 2차원적인 정보밖에 제공해주지 못하기 때문에 조종사에게 안전과 관련된 더 많은 정보를 제공하기 위해서는 근본적으로 현재 2차원 그래픽을 기반으로 하는 디스플레이 시스템을 3차원 그래픽 기반의 디스플레이 시스템으로 수정하는 작업이 이루어 져야 한다.

위성 측위 시스템인 GPS(Global Positioning System)를 항법 시스템에 이용하고자 하는 시도가 여러 연구 집단에서 이뤄지고 있다. 특히, 미 연방 항공청에서는 항공기의 착륙 유도 서비스를 위해서 근거리 오차 보정 시스템(LAAS: Local Area Augmentation System)을 정의하고, 이에 대해서 산업체와의 협력을 통한 관련 기술 개발 및 상용 시스템 개발에 박차를 가하고 있다[1-4].

본 연구는 3차원 지형 데이터에 항공사진을 맵핑(Mapping)하여 조종석 디스플레이 시스템을 개발하는 것이다. 고도와 좌표가 있는 지형 데이터를 사용하여 3차원 격자구조의 지형을 만들고 이를 효율적인 알고리즘으로 관리하며 또한 항공사진을 지형에 정확하게 Mapping하여 비행기의 캐노피에서 내려다보는 모습과 디스플레이 시스템에서의 모습을 똑같도록 구현하여 악천후나 야간과 같은 시계가 불량한 상황에서도 안정적으로 비행할 수 있는 화면을 만들어 내는 것이다. 또한 하늘의 터널(Tunnel in the Sky)를 표시하여 이착륙 시 안전한 이착륙을 유도할 수 있는 기능도 구현하였다. 항공기에 탑재되는 컴퓨터는 일반적으로 사용하는 데스크 탑 PC 보다도 성능이 떨어지기 때문에 보다 효과적인 알고리즘을 사용하여 빠른 화면 플래쉬 속도(Reflash Rate)를 유지할 수 있는 방법을 찾아야 한다.

따라서 본 연구에서는 빠른 플래쉬 속도를 유지하기

2011년 7월10일 접수 ~ 2011년9월20일 심사완료
* 세종대학교 대학원
** 세종대학교 항공우주공학과
연락저자, E-mail: hyhwang@sejong.ac.kr
서울시 광진구 군자동 98

위하여 LOD(Level of Detail), OCTree, Frustum Culling 등의 다양한 알고리즘과 효과적인 시스템 구축, 클래스(Class) 설계 등을 이용하여 효과적인 3차원 조종석 디스플레이 시스템을 개발하는 연구를 수행하였다.

2. 본 론

2.1 3D 조종석 디스플레이 시스템

3D 조종석 디스플레이 시스템은 기본 3차원으로 이루어진다. 이전에는 3차원이라고 하면 고성능이면서 비싸고 거대한 컴퓨터에서의 전유물 같이 생각되었으나 근래에 들어 저가의 고성능 컴퓨터와 함께 효율적인 3D 가상공간 구축을 위한 OpenGL 라이브러리(Library)가 등장하고, 센서로는 GPS를 이용하게 되면서 기능적으로 뿐만 아니라 경제적으로도 구현 가능하게 되었다. 따라서 본 연구에서는 이를 구현하기 위하여 좌표계, 데이터의 변환과 이를 표현하기 위한 3D 엔진(Engine)을 연구, 개발하였다.

2.1.1 좌표계

2차원 기반의 디스플레이 시스템에서 3차원 기반의 디스플레이 시스템으로의 전환에 추가되는 부분 중 가장 핵심적인 부분이 지형 정보의 추가이며, 이를 위해 선행되어야 할 것이 디스플레이 시스템의 좌표계를 설정하는 것이다. 본 연구에서 사용하게 되는 국토지리정보원(구 국립지리원)의 3차원 지형 정보 데이터의 기반 좌표계가 TM(Transverse Mercator) 좌표계이기 때문에 디스플레이 시스템의 기반 좌표계를 TM 좌표계로 설정하였다. 그러나 대부분의 GPS를 이용하여 구해지는 비행기의 위치 해는 WGS84(Worldwide Geological System) 좌표계를 기반으로 한다. 따라서 실시간으로 입력되는 비행기의 WGS84좌표계 기반의 좌표를 TM 좌표로 전환할 필요성이 있다.

2.1.1.1 평면 좌표 변환

WGS84 좌표를 TM 좌표로 변환하는 과정은 다음 네 가지로 나뉜다.

1) 위, 경도로 표시된 WGS84 좌표를 ECEF(Earth Centered Earth Fixed)좌표로 변환한다.
2) 변환된 ECEF 좌표를 베셀(Bessel) 타원체 기준의 ECEF 좌표로 변환한다. 이때 Bursa-Wolf 모델(7 Parameter)을 이용한다.
3) 베셀 타원체 기준의 ECEF 좌표를 위, 경도 좌표계로 변환한다.
4) 주어진 기준점을 기준으로 TM 좌표계로 투영시켜 최종적인 항공기의 TM좌표를 구한다.

2.1.1.1.1 WGS84의 Bessel ECEF로의 변환

WGS84를 베셀 타원체 기준의 ECEF로의 변환은 한 타원체에서 다른 타원체 상의 좌표로 변환하는 것을 의미한다. 변환 요소로는 일반적으로 7가지의 요소를 사용한다. 이를 7 변수(Parameter)라고하며 이를 사용하는 모델을 Bursa-Wolf 모델이라 한다. 7개의 변수는 Table 1과 같다.

Table 1 Bursa-Wolf model 7 parameters

$Tx,\ Ty,\ Tz$	타원체 중심간의 X, Y, Z방향의 평행 이동량
ω, ϕ, κ	X, Y, Z축을 중심으로 하는 회전 이동량
S	축적계수

Table 1의 변수들을 이용한 타원체간의 변환식은 다음과 같다.

$$\begin{aligned} |X_b| &= |T_x||1\kappa - \phi||X_W| \\ |Y_b| &= |T_y| + (1-S)| - \kappa 1\omega||Y_W| \\ |Z_b| &= |T_z||\phi - \omega 1||Z_W| \end{aligned} \quad (1)$$

위의 식에서 X_b, Y_b, Z_b는 베셀 타원체상의 X, Y, Z좌표이고 X_W, Y_W, Z_W는 WGS84 타원체상의 X, Y, Z좌표이다. 회전의 이동량은 라디안(radian)단위이다. 그러나 국토지리정보원의 작업규정에는 S, ω, ϕ, κ값을 모두 0으로 정하였기 때문에 다음과 같이 간단하게 표현된다.

$$\begin{aligned} |X_b| &= |T_x||X_W| \\ |Y_b| &= |T_y| + |Y_W| \\ |Z_b| &= |T_z||Z_W| \end{aligned} \quad (2)$$

여기서 $T_x = -146.44m$, $T_y = 507.89m$, $T_z = 608.46m$이다. 위의 공식은 베셀 타원체에서 WGS84 좌표계로 변환하는 공식이다. 반대로 WGS84에서 베셀 타원체로의 수식으로 바꾸는 방법은 X_b, Y_b, Z_b와 X_W, Y_W, Z_W로 바꾸고 '+'를 '-'로 바꾸어 준다. 이를 수행한 최종 공식은 다음과 같다.

$$\begin{aligned} |Xw| &= |Tx||Xb| \\ |Yw| &= |Ty| - |Yb| \\ |Zw| &= |Tz||Zb| \end{aligned} \quad (3)$$

여기서 제시한 좌표 값은 모두 3차원 직교 좌표 값이므로 다시 경위도 좌표 값으로 바꾸어야 한다.

2.1.1.1.2 ECEF의 TM 좌표로의 변환

베셀타원체 기준의 ECEF의 값인 3차원 직교 좌표값을 최종 TM좌표로 변환하기 위하여 가우스 상사 이중투영법(Gauss Conformal Double Projection)을 이용하였다. 가우스 상사 이중투영법은 투영면상에 있어서의 방향각 및 변을 일정법칙에 따라 이를 지구상으로 투영하고, 다시 구면상에서 평면상으로 투영하는 방법이며, 또한 그 역계산도 할 수 있다. 이의 변환은 (주)지오매니아 개발 라이브러리인 GDK(GeoMania Development Kit)를 이용하여 수행하였다. 수행 후 최종적으로 얻어지는 값은 TM 좌표계에서의 경위도 좌표가 된다.

2.1.1.2 고도 좌표 변환

평면좌표 이외에 항공기의 디스플레이 좌표계에서의 고도 데이터를 구해야 한다. 지도의 기준 고도 좌표계가 해발고도를 기준으로 하므로, 활주로 착지점의 해발고도와 WGS84의 고도 차이를 구한 후, 이후에 입력되는 항공기의 WGS84 고도에서 그 차이만큼 빼주면, 원하는 디스플레이 좌표계에서의 비행기의 고도가 구해지게 된다.

2.1.2 데이터

데이터는 크게 3D 지형 데이터, 3D 지물 데이터, 텍스쳐(Texture) 데이터, 2D 지형 데이터 그리고 기타 정보 데이터로 구분된다. 3D 지형 데이터는 각 위치의 고도데이터가 추가되어 격자형으로 구성된 3차원 데이터이며, 3D 지물 데이터는 공항과 같은 건물이 된다. 텍스쳐 데이터는 지형에 맵핑(Mapping)할 항공사진과 건물 및 비행체 등의 모든 질감 데이터이다. 2D 지형 데이터는 지형의 평면지도 정보가 주가 된다. 기타 정보 데이터는 각각의 3D화면 또는 2D화면에 표시할 사용자에게 필요한 정보 데이터가 된다.

2.1.2.1 3D 지형 데이터

3D 지형 데이터에는 3차원 지형을 생성하기 위한 일정간격의 고도데이터가 저장된 *.VOX의 파일을 가지고 있다. 이 파일은 국토지리정보원 등을 통해 얻은 지형자료를, (주)지오매니아의 지도 제작용 소프트웨어인 GeoMania V.2.5를 통해, 지형의 고도정보를 가지고 있는 GeoMania 자체 포맷인 GDF 등으로 1차 변환하였다. 이렇게 변환된 데이터는 본 연구를 위해 개발된 3D 지형 로더에서 직접 읽을 수 없기 때문에, GeoMania 개발 라이브러리인 GDK(GeoMania Development Kit)를 사용하여, 3D 지형 로더에서 읽어 들일 수 있는 포맷으로 변경하였다. 이 파일에는 고도데이터 만이 들어있다. 3D 지형 로더에서는 이 파일을 읽어 들여 실제 3차원 지형을 구성하게 된다(Fig. 1).

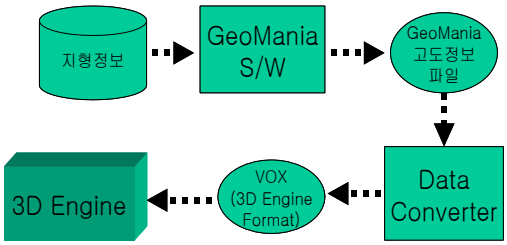

Fig. 1 Data conversion process

2.1.2.2 공간 자료구조 : OCTree

공간자료 구조는 보통 계층적으로 구성된다. 즉 최상위 레벨의 내용이 그 아래 내용을 포함하고, 그 레벨은 다시 그 아래 내용을 포함하고 있다는 것을 의미한다. 이러한 계층적 구조를 사용하는 주된 이유는 여러 가지 유형의 검사들이 매우 신속하게 이루어질 수 있다는 것이다. 통상 $O(n)$의 비용이 $O(\log n)$으로 개선된다. 대부분의 공간 자료구조는 만드는 비용이 많이 들고, 보통 전처리 과정에서 만들어진다.

이러한 공간 자료구조들의 종류로는 바운딩 볼륨 계층구조 (Bounding Volume Hierarchies:BVHs), BSP 트리의 변종, 8진 트리(OCTree)같은 것이 있다. BSP 트리와 8진 트리(OCTree)는 공간 분할(Space Subdivision)에 기반한 자료구조이다. 모든 리프 노드를 합한 공간은 장면의 전체 공간과 동일하다. 8진 트리는 정규적이며 이것은 공간을 균일하게 분할한다는 것을 의미한다. 다소 제한적이기는 하지만, 이러한 균일성은 효율성의 원천이 된다. 초기 생성의 비용이 많이 들지만 본 연구에서는 구축된 자료구조의 실시간적인 변화가 없기 때문에 초기 비용만 많이 들뿐 추가적인 비용은 없다. 따라서 본 연구에서는 공간자료구조로 8진 트리(OCTree)를 사용하였다(Fig. 2).

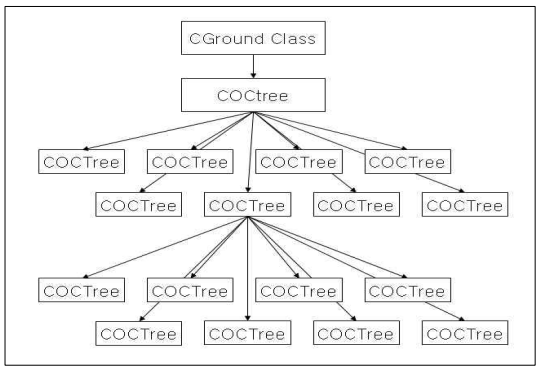

Fig. 2 COCTree class

3D 지형 로더에서 읽어들인 데이터는 OCTree 자료구조를 이용하여 관리하게 된다. OCTree란 지형을 3차원 격자로 구성하여 디스플레이 성능을 올리기 위한 방법이다. 이것을 개념적으로 설명하면 이 OCTree는 각 노드당 8개의 자식들을 가질 수 있으며 각 자식은 다시 노드당 8개의 자식을 가질 수 있는 구조체이다. 이 OCTree는 입방체(정육면체)로 감싸인 3차원 공간을 표현하는데 이상적인 구조로 간주되고 있으며 이는 원하는 크기(Depth Level)까지 분할이 이루어진다(Fig. 3). 이 분할되어진 구조체는 시야 각뿔대(View Frustum)에 의해 결정된 보이는 영역, 즉 그려져야 할 영역에 대해 실제 보이는지를 검사하게 된다. 이를 컬링(Culling)이라 하며 이는 3D 그래픽에서 중요한 부분으로 다루어지고 있다. 시야 각뿔대에서 검사에 통과한 지형만이 그려지게 되며 이는 불필요한 노드(Node)를 제거한다. 또한 트리(Tree)의 특성상 상위에서의 검사가 실패했다면 하위(자식 노드)에 대해서는 검사를 할 필요가 없어지기 때문에 검사의 수를 줄일 수 있는 이점이 있다. 본 연구에서 사용하는 것과 같이 상대적으로 넓은 지형적 영역을 가지고 있는 구조에서는 아주 효율적인 방법이다.

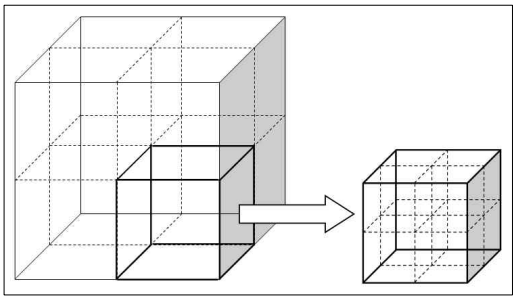

Fig. 3 Concept of OCTree structure

본 프로그램에서 사용되도록 개발된 CGround 클래스(Class)는 COCTree 클래스를 루트(Root)로서 하나를 가지고 있다. 이 COCTree 클래스는 다시 8개의 하위 OCTree를 가지게 되며, 하나의 OCTree는 다시 반복적으로 8개의 하위 OCTree를 가지게 된다. 이 순환은 지정된 Divide 레벨(Level)까지 깊이(Depth)를 가지도록 생성된다. Divide 레벨은 적당한 깊이가 되도록 설정을 하여야 한다. Divide 레벨 설정을 작게 설정하면 컬링(Culling)의 기능이 감소하여 많은 지역을 그려야 하며, 레벨을 크게 설정하면 꼭 필요한 부분만을 그릴 수는 있으나 영역을 검사하는데 Overhead를 줄 수가 있다. 따라서 적정수준의 Divide 레벨을 찾아야 한다. Divide 레벨은 초기화 시점에 설정을 한다.

OCTree는 최하위 단계에서만이 실제 지형데이터를 가지고 있다. 중간 단계의 OCTree는 지형데이터는 가지고 있지 않으며 하위 레벨의 OCTree를 가지고 있는 연결자 역할만을 수행한다. 중간 단계의 OCTree가 가지고 있는 데이터는 자신의 공간영역에 대한 중심점과 넓이만을 가지고 있으며, 이 데이터를 Culling에 이용하게 된다.

Fig. 4는 Test지형을 OCTree로 분할 후 각 Tree Node의 공간영역을 8면체의 박스로 그린 모습이다. 지형이 존재하는 부분은 조밀하게 분할되어 있다는 것을 알 수 있으며 지형이 없는 부분은 분할이 되지 않는다는 것을 알 수 있다. 분할이 되지 않았다는 것은 다른 연산들(Culling, Line Detect, 충돌 체크)에서도 이 부분은 제외된다는 의미이다.

2.1.2.3 지물 데이터

지형이외의 지물을 표현하기 위해서는, 지물을 모델링한 데이터가 따로 필요하며 그것을 지형에 통합하여 넣어야 한다. 지물들은 Object라 부르며, Fig. 5와 같은 데이터 구조를 사용한다. 지물은 지형과 다르게 설계자가 직접 제작하여야 한다. 제작에 사용한 도구는 3D Studio Max를 사용하였으며 데이터는 ASCII 포맷인 *.ASE 파일로 변환 출력하여 사용하였다. 이 지물 또한 위에서 설명한 시야 각뿔대(View Frustum)에 의해 실제 보이는 것만이 그려지게 된다. 검사는 하나의 Object단위로 이루어진다.

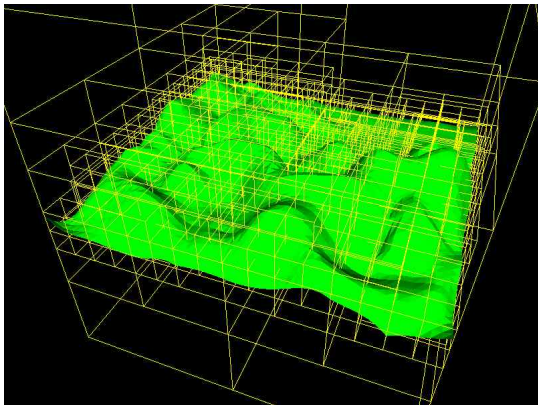

Fig. 4 OCTree division example

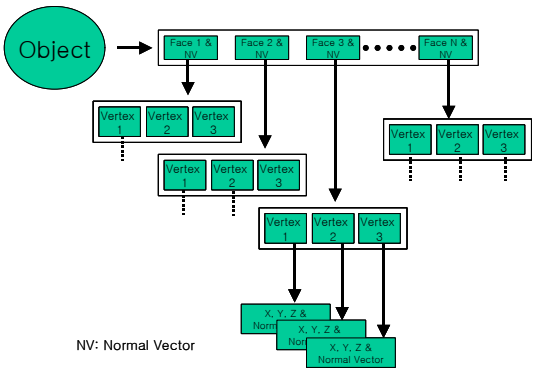

Fig. 5 3D object data structure

2.1.2.4 텍스쳐 데이터

지형/지물 데이터에는 또한 텍스쳐 데이터가 들어가게 된다. 지형 텍스쳐 데이터는 실제 서울시의 항공사진을 사용하고 있다. 이 데이터는 Detail이 높기 때문에 확대를 해도 많이 깨지지 않는 장점이 있으나, 데이터의 크기가 매우 큰 단점이 있다. 텍스쳐 데이터는 한번에 모두 Memory에 로드 후 사용하는 것이 가장 좋으나 이미지 데이터의 크기가 큰 관계로 모두 로드를 할 수는 없다. 따라서 부분적으로 필요한 부분만 로드 할 수 있어야 한다. 이를 효율적으로 관리하기위한 방법으로 전체 지형데이터를 격자로 관리하는 것을 사용하였다.

위의 지형데이터 관리에서 사용한 OCTree 하나의 크기로 격자를 구성한 후 실제 텍스쳐 Image를 격자의 크기로 분할한다. 분할되어진 이미지는 각 OCTree에 텍스쳐 Source로 들어가게 되고 그려지게 될 때 텍스쳐 Source가 로드 되어있지 않으면 새로 로드 하는 방식을 사용하였다. 로드 되어진 후 일정시간 동안 사용되지 않으면 Memory의 효율을 위하여 Delete를 수행하여 메모리의 사용량을 줄이게 된다.

2.1.2.5 2D 지형 데이터

2D 지형 데이터는 2차원 평면 지도를 의미한다. 이 지도는 사용자가 필요시에 3차원에서 2차원 지형으로 간단한 버튼의 조작으로 바로 변경하여 볼 수 있으며 지도에 항공기의 위치와 방향이 표시된다. 따라서 사용자는 항공기 운항 중 자신의 위치를 2차원 지도상에서 자세히 파악하고 싶을 때 바로 파악을 할 수가 있다. 또한 이 자료는 주변에 어떤 건물이 있는지를 파악할 수도 있다.

2.1.2.6 기타 정보 데이터

기타 정보 데이터에는 부가적으로 표시할 정보들이 저장되어있다. 저장되어있는 정보로는 다음과 같다.

1) 비행장의 위치
2) 비행장의 Tunnel-in-the-sky 정보
3) 작전 정보
4) 기타 필요 정보
5) 비행경로 정보

회랑(Corridor, Tunnel-in-the-sky)이라는 것은 이름 그대로 하늘에 가상적인 터널을 생성시키는 기법을 말한다[5].

회랑(Corridor, Tunnel-in-the-sky)은 항공기가 따라가야 할 정확한 경로가 결정되어 있다면, 그 경로 위의 임의의 한 점이 터널의 중심점이 되고, 수평방향과 수직방향의 크기를 결정하면, 평면 사각형 형태의 터널이 한 개 생성되며, 이 터널의 아랫변의 중심부에서 지면으로의 Anchor Line을 그리게 되면, 하나의 터널 객체가 생성되게 된다. 이후 필요로 하는 위치마다 원하는 크기의 터널을 생성하여 입체적인 터널을 생성하게 된다. 회랑은 지형이 복잡한 공간에서 항공기가 가야할 정확한 방향과 공간영역을 알려주는 기능에 사용할 수 있다. 이는 항공기가 저고도로 비행 중에 지형과의 충돌을 막을 수 있는 방법으로서 안전한 비행경로를 미리 입력하여 회랑을 생성하여 정해진 비행경로를 따라가면 안전하다는 것을 알려줄 수 있다.

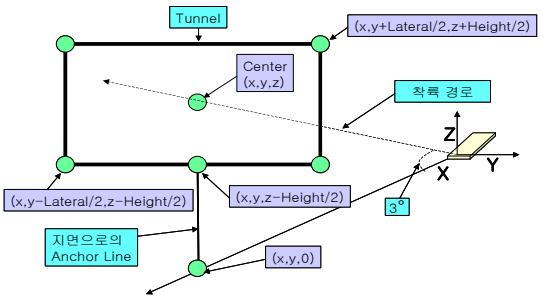

Fig. 6 Creation of tunnel-in-the-sky indicating landing path

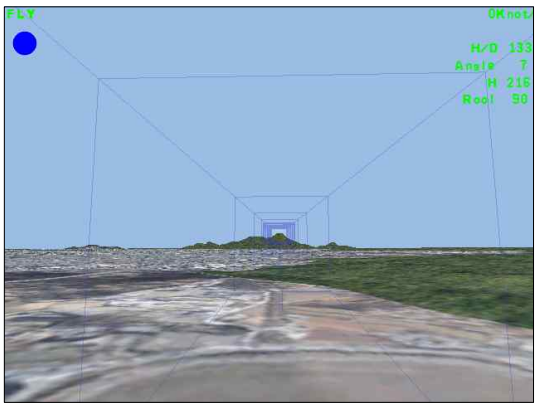

Fig. 7 Tunnel in the Sky 1
- View from inside tunnel

2.2 3차원 디스플레이

3D 조종석 디스플레이 시스템을 구현하기 위해서는 방대한 양의 데이터가 필요하며 이를 PC에서 효율적으로 관리하고 디스플레이 하는 방법이 매우 중요하다. 또한 위의 3D 정보를 화면에 표시하기 위하여 실시간으로 출력 가능한 3차원 그래픽 엔진의 개발이 필요하였다.

2.2.1 3D Engine

3D Engine을 구현하기 위해, 기본적인 Renderer로는 OpenGL을 사용하였다. OpenGL은 SGI사가 개발하고

공개한 산업계 표준 3D 렌더링 라이브러리이다. 예전 OpenGL은 고가장비의 그래픽카드에만 가속기능이 구현되어 있었으나, 현재는 대부분의 PC용 그래픽 카드에서 OpenGL의 스펙을 90% 이상 지원하고 있어, OpenGL로 개발된 응용 프로그램들도 현재 대부분의 PC에서 원활히 동작한다. 그러나 본 연구에서 실제로 3D Engine이 사용될 플랫폼은 항공기에 탑재되어야 하는 산업용 PC이므로 일반 Desktop PC보다 성능이 떨어질 수 있다는 점을 감안하여 8진 트리 공간자료 구조를 이용하여 내부처리 방법을 효율적으로 관리하도록 하였다. 또한 다양한 정보를 입력받고 이를 화면에 출력함과 동시에 다양한 Utility Function을 개발하여 추가 입력정보 없이 프로그램 내에서 다양한 정보를 도출할 수 있도록 하였다. 이는 부족한 정보를 자체 프로그램에서 생성하므로 정보입력을 위한 새로운 하드웨어 장치를 요구하지 않게 된다.

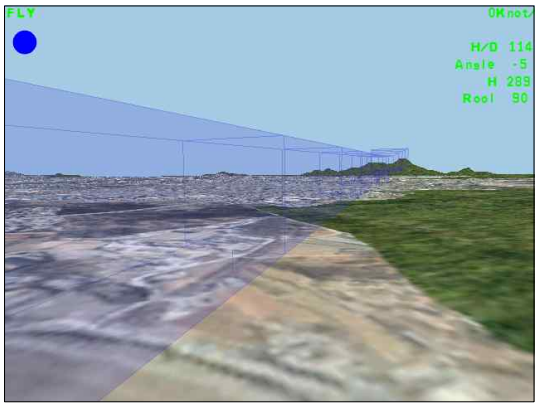

Fig. 8 Tunnel in the Sky 2
- View from outside tunnel

2.2.2 주요기능

3D Engine은 기본적으로 갖추어야할 기능 이외에도 본 연구에 필요한 다음과 같은 기능을 추가로 가지고 있다.

1) 지형정보(*.VOX), 지물 정보(*.ASE) 로딩기능
2) 여러 가지의 카메라 모드
3) 항공기 위치 및 회전 제어 기능
4) 화면 문자열 표시기능

2.2.2.1 지형정보(*.VOX), 지물정보(*.ASE) 로딩기능

지형정보는 *.VOX의 확장자를 가지는 파일에 저장되어 있다. 이 파일을 CGround 클래스에서 읽어 들여 3차원 지형을 구축하게 된다. *.VOX 파일은 다음의 정보들을 가지고 있다.

1) Width : 지형의 가로 길이(TM 좌표계 상에서)
2) Height : 지형의 세로 길이(TM 좌표계 상에서)
3) WCount : 지형의 가로 방향의 포인트 수
4) HCount : 지형의 세로 방향의 포인트 수
5) RealX : 지형의 실제(TM 좌표계 상에서) 위치 X
6) RealY : 지형의 실제(TM 좌표계 상에서) 위치 Y
7) Voxels : 지형의 실제 고도 데이터들

위의 정보들을 이용하여 실제 3차원 격자 지형을 만들고 이를 OCTree 자료구조를 이용하여 분할 구성하게 된다.

지물정보는 *.ASE의 확장자를 가진다. 이는 3D Studio MAX에서 바로 Export할 수 있는 파일로서 공개 버전의 3D 데이터 파일 포맷이다. 이 파일은 CASELoader 클래스에서 읽어 들여 사용되어진다.

2.2.2.2 카메라 모드

본 3D Engine은 다양한 카메라 모드를 지원한다. 사용자는 원하는 카메라 모드를 선택하여 화면을 볼 수 있으며 모드는 다음과 같다.

1) FLY Mode(Default View): 기본 모드로서 항공기 안에서 진행방향으로 바라보는 화면이다. 카메라의 위치는 항공기의 위치이며 바라보는 방향은 항공기의 진행 방향이다(Fig. 9).
2) ATTACH Mode(Back View) 항공기의 모습을 포함한 디스플레이 모드: 항공기를 출력하는 모드이기 때문에 카메라는 항공기의 약간 뒤쪽 상단에 위치하게 된다. 기본적으로 자동으로 시점을 잡으며 수동으로 카메라의 위치를 바꿀 수 있다(Fig. 10).
3) TOP Mode(Top View) 항공기에서 아래쪽을 바라보는 디스플레이 모드: 항공기는 화면 중앙에 위치하며 그곳에서 수직으로 지형을 내려다본 모습이다(Fig. 11).
4) CAM Mode(Eye View) : 항공기의 진행 방향과 상관없이 사용자의 임의대로 Camera의 방향을 바꿀 수 있다. Camera의 위치는 항공기의 위치와 일치한다. 시점은 좌우 360도, 상하 ± 90도 회전이 가능하다.
5) FREE Mode : 사용자가 화면상의 항공기를 조종하게 된다. 실제 비행과는 상관이 없으며 임의적인 조작모드이다.

2.2.2.3 추가기능

카메라 모드 이외에 다음의 추가 기능을 가지고 있다.

1) 화면의 확대/축소 기능: 모든 카메라 모드에서 X0.5 ~ X2.0까지의 모드를 지원한다.
2) 지상과의 고도 표시: 지상과의 고도를 표시한다. 고도는 2가지를 표시한다. 높이

0을 기준으로 계산한 고도와 실제 3D 지형과의 고도차를 표시한다.

3) Sky Box 표시: 착륙을 위한 각 공항의 Sky Box를 표시한다.

4) 2D 지도 표시: 2D Raster 지도를 표시한다(Fig. 12).

Fig. 9 FLY View Mode

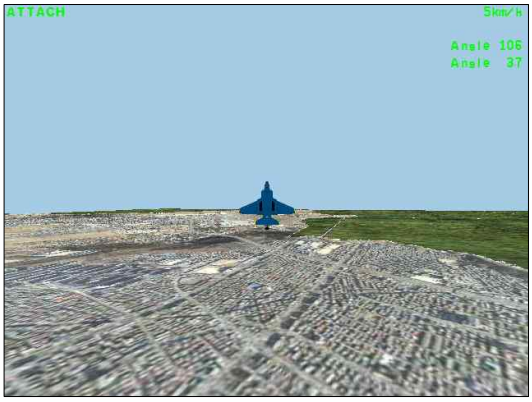

Fig. 10 ATTACH View Mode

Fig. 11 TOP View Mode

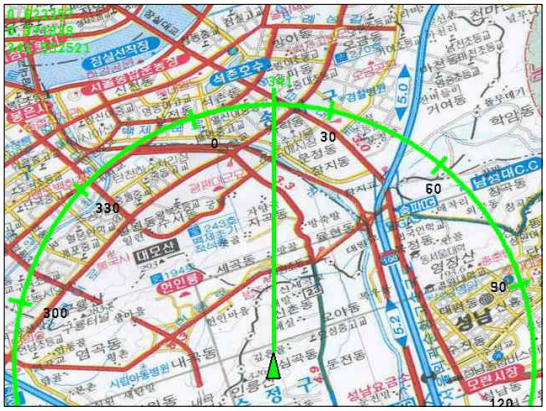

Fig. 12 2D raster map

2.3 사용 클래스

본 프로그램을 위해 개발한 클래스 명과 기능은 Table 2와 같다.

Table 2. Class List

번호	이름	기능
1	C2DMap	2D 지도를 관리한다.
2	CAirport	항공기의 정보를 관리한다. 위치, 방향, 속도, Zoom, Camera Mode
3	CFrustum	View Frustum을 생성 관리한다. Culling을 위한 기본 클래스이다.
4	CGLMain	3D 화면을 구성하는 Main 클래스이다. OpenGL 화면의 생성과 소멸, 항공기 객체, 지형객체, 공항, Tunnel - in -the-sky,텍스쳐등의 다양한 객체와 기타 모든 정보를 관리하고 화면에 출력한다.
5	CGround	지형을 관리한다. 하위에 COCTree 클래스를 포함하고 있으며, 지형의 넓이, 실제 좌표, 지형의 분할 정도, 지형 데이터의 로딩, 지형의 생성, 지형의 소멸 등을 관리한다.
6	CModel	항공기의 3D Model 정보를 관리한다. 항공기의 3차원 정보를 파일로부터 읽어 들여 3D Model을 생성하고 화면에 출력하는 역할을 수행한다.
7	CMyFont	화면에 사용자 Text를

		출력한다. Text를 화면에 출력하여 사용자에게 정보를 제공하여 준다.
8	CMyTexture	텍스쳐를 읽어들이고 관리하는 클래스이다.
9	COCTree	지형 정보를 OCTree형태로 분할, 관리한다.
10	CTGA	Image File중의 하나인 TGA File을 로딩한다.
11	COneTile	2D 이미지중 하나의 Tile Image를 관리한다.
12	CNineTile	COneTile 클래스 9개를 관리한다.
13	CTileImage	CNineTile 클래스를 관리한다.
14	CTIS	Tunnel in the sky를 관리한다. 파일로부터 정보를 읽어 들여 Tunnel in the sky를 생성하고 관리한다.

위 표의 14개의 클래스 중에서 맨 마지막의 CTIS 클래스에 대해서만 자세히 설명하고 나머지 클래스에 대해서는 분량이 방대하므로 본 논문에서는 설명을 생략하고자 한다.

2.3.1 CTIS 클래스

본 클래스는 Tunnel-in-the-sky를 관리하고 Rendering을 수행한다. 설정된 파일에서 Tunnel-in-the-sky의 정보를 읽어 들여 Sky Box를 생성하고 화면에 출력하는 역할을 수행한다. 항공기의 고도에 따라 Tunnel-in-the-sky의 색을 변화 시킨다. 본 프로그램에서는 해수면과의 고도차를 이용하여 현재 항공기의 고도가 2000M보다 높으면 Tunnel-in-the-sky를 그리지 않으며, 1000M보다 높으면 주황색으로 표시하고 그 이하이면 푸른색으로 표시하여 화면상에서 자신의 고도가 얼마인지 대략적으로 파악하도록 하였다.

OneBox Struct는 Tunnel-in-the-sky에서 하나의 박스를 구성한다. 이 박스가 모여 하나의 완성된 Tunnel을 구성하며, OneBox는 중심점과 네 개의 꼭짓점 좌표를 가지고 설정된다. Rendering을 수행할 때는 박스와 박스의 꼭짓점을 연결하여 하나의 완성된 Tunnel-in-the-sky를 그리게 된다.

Rendering 설정은 Fig. 13과 같다.

```
glPolygonMode(GL_FRONT_AND_BACK, GL_FILL);
glEnable(GL_BLEND);
glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
```

Fig. 13 TIS Renderng basic setup

Tunnel-in-the-sky는 투명 객체이므로 면의 앞과 뒤를 모두 그리기 위하여 glPolygonMode를 GL_FRONT_AND_BACK으로 설정하였다. 이렇게 하면 평면의 앞과 뒤에 상관없이 모두 그리게 된다. 기본설정은 앞면만을 그리는 GL_FRONT이다. GL_BLEND를 Enable시켜 투명객체를 그리는 것이며 그리는 방법은 GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA로 설정하였다.

입력 데이터는 ASCII 형태로 파일에 저장되어 있다. 파일의 구조는 Fig. 14와 같다. Header는 Tunnel-in-the-sky의 3차원 위치 좌표, Tunnel의 개수이며 이후 각 Tunnel의 정보가 저장된다. Tunnel의 정보는 Tunnel의 3차원 위치 좌표, 폭, 높이, Yaw, Pitch 값의 순으로 저장된다.

```
7481.0 100.0 16691.0
15
0.0 100.0 0.0 100.0 100.0 -45.0 90.0
200.0 110.0 200.0 100.0 100.0 -45.0 90.0
400.0 120.0 400.0 100.0 100.0 -45.0 90.0
:
800.0 140.0 800.0 100.0 100.0 -45.0 90.0
```

Fig. 14 Tunnel-in-the-sky Data File

입력을 받은 데이터는 CTIS 클래스의 AddBox Function을 호출하여 생성된다. 생성 순서는 다음과 같다.

```
void AddBox(int i, float X, float Y, float Z,
            float Width, float Height,
            float Yaw, float Pitch);
```

1) struct OneBox객체를 생성하고 기본 데이터를 삽입한다. Center에는 X, Y, Z의 중심 값을 Corner[]에 네 꼭지점의 좌표를 넣는다.
2) Corner[]의 꼭지점을 Yaw와 Pitch 만큼 회전시킨다.
3) OneBox의 중심점을 원하는 곳으로 이동시킨다.

회전 Function은 Fig. 16과 같고 이 Function을 이용하여 모든 OneBox의 데이터가 각각의 Yaw, Pitch만큼 회전하게 된다.

3. 결 론

본 연구에서는 3차원 조종석 디스플레이 시스템을 연구 개발하였다. 본 시스템에서 지형 데이터를 로딩하여 3차원 지형데이터를 생성하고 이를 OCTree자료구조를 이용하여 관리하며, Frustum을 이용하여 Culling을 하였다. 지형은

LOD에 의해 먼 곳은 단순하게 표현하고, 가까운 곳은 실제와 최대한 동일하도록 하였으며 LOD에 의해 발생하는 가까운 지형과 멀리 있는 지형을 효과적으로 이어주는 방법을 연구하였다. 또한 다양한 정보를 화면에 출력하여 사용자의 편의를 증대시킬 수 있는 방법을 연구하였다.

```
struct OneBox{
    VECTOR3 Center;
    VECTOR3 Corner[4];
};

class CTIS
{
    private:
        FILE *m_fp;
        int m_Count;
        OneBox *m_SkyBox;
        float m_StartX, m_StartY, m_StartZ;

        void AddBox(int i, float X, float Y, float Z,
                    float Width, float Height,
                    float Yaw, float Pitch);

        void rotate_global_axis_Y(VECTOR3 *vtx, float ang);
        void rotate_global_axis_X(VECTOR3 *vtx, float ang);
    public:
        CTIS();
        ~CTIS();

        void LoadFromFile(char *);
        void Draw(float Height);
};
```

Fig. 15 CTIS Class Header

```
void CTIS::rotate_global_axis_Y(VECTOR3 *vtx, float ang)
{
        if(ang != 0)
        {
                double cs=cos(ang*__TORAD);
                double sn=sin(ang*__TORAD);
                double tmp = vtx->x;
                vtx->x = cs*vtx->x - sn*vtx->z;
                vtx->z = sn*tmp + cs*vtx->z;
        }
}

void CTIS::rotate_global_axis_X(VECTOR3 *vtx, float ang)
{
        if(ang != 0)
        {
                double cs=cos(ang*__TORAD);
                double sn=sin(ang*__TORAD);
                double tmp = vtx->z;
                vtx->z = cs*vtx->z - sn*vtx->y;
                vtx->y = sn*tmp + cs*vtx->y;
        }
}
```

Fig. 16 Tunnel-in-the-sky rotation function

본 프로그램을 개발하기 위하여 다양한 자료구조와 알고리즘을 연구하였으며, 이를 최종 구현하기 위하여 3D Engine을 개발하였다. 3D Engine은 필요한 다양한 기능을 가지고 있으며 최대한 쉽게 다른 기능을 추가할 수 있도록 개발하였다. 본 시스템은 항공기의 다기능 시현기(MFD: Multi Function Display)에 유용하게 사용될 것으로 기대된다.

추후, GPS와의 실제 연동시험을 통해 정확도를 상승시키는 일과 실제 운용할 사람들로부터 요구되는 기능적 추가사항을 보완하여 보다 다양한 기능을 추가하여야 한다.

후 기

이 논문은 2009년도 세종대학교 교내연구비 지원에 의한 논문임

참고문헌

[1] http://www.flightgear.org/, Flightgear Homepage

[2]http://www.cc.gatech.edu/gvu/people/peter.lindstrom

[3]http://www.sgi.com/software/opengl/advance97/notes

[4]http://satrec.kaist.ac.kr/rswww/r2_homepage

[5] 김종대, 기창돈, 김진홍, 황호연, “비행기 착륙 궤적의 자동 생성 및 3차원 Tunual-in-the-Sky 디스플레이”, 2003년도 한국항공우주학회 춘계학술발표회 논문집, pp. 478-482.

[6] 황호연, “DGPS 기준국 신호분석 소프트웨어”, 한국항공운항학회지 제15권 제2호, 2007, pp. 1-8

[7] 이재영, 황호연, 김정엽, “RC 헬리콥터 최적화 설계를 위한 변수설계 기법”, 한국항공운항학회지 제16권 제4호, 2008, pp. 1-11