韓國地形空間情報學會誌 第16卷 第2號 2008年 6月 pp. 57-65

연구논문

레이저레이더 시뮬레이션을 위한 3차원 객체 모델링
3D Object Modeling for Laser Radar Simulation

김근한* · 전철민**
Kim, Geun Han · Jun, Chul Min

要 旨

레이저레이더 시뮬레이션의 성능을 향상시키기 위해서는 시뮬레이터의 레이저신호로 대응되는 공간의 범위와 해당 사물의 위치 및 속성정보를 정확하고 빠르게 획득해야 한다. 또한 시뮬레이션에 사용되는 데이터는 지형, 건물 및 차량과 같은 복잡한 3차원 객체들이며 광범위한 지역을 대상으로 하므로 가시화를 위한 일반적인 3차원 모델링 툴로는 빠르게 데이터를 추출하고 연산을 수행하기 어렵다. 본 연구에서는 이와 같은 복합적인 형태의 3차원 객체를 데이터베이스에 저장하고 필요한 질의를 수행하며, 가시화 부분과 연동할 수 있는 기법을 제시하였다. 이를 위해 3차원 다면체를 토폴로지 기반으로 데이터모델링을 수행하는 과정과 이러한 객체를 공간 DBMS를 이용하여 구현하는 과정을 예시하였다. 또한 DB에 저장된 데이터를 접근하여 가시화하는 과정을 VRML을 이용하여 구현하고, 시뮬레이션 레이저신호와의 연산 테스트를 실시하였다. 향후 데이터모델에 대한 연구와 가시화 부분에서의 LOD적용 등의 문제를 해결한다면 시뮬레이션뿐 아니라 보다 다양한 상황에 적용할 수 있을 것이다.

핵심용어 : 3차원 모델, 데이터모델링, PostGIS, VRML, 레이저레이더

Abstract

The improvement of the performance in laser radar simulation requires fast retrievals of the spatial locations and attributes of objects in response to the laser signals of the simulators. Since the data used in simulation are complex 3D objects such as terrain, buildings and vehicles, and are of large sizes, commonly used 3D modeling tools are not suitable for this use. We proposed a method to store such 3D objects in a database, perform required queries and integrate with visualization tools. We showed the processes for the data modeling based on 3D topological concepts and then building a spatial DBMS. Also, we illustrated the process for accessing and visualizing the stored data using VRML and performed test computations using some laser signal data. With further enhancement on data modeling and LOD problems in visualization, the proposed method will be practically applied in different situations including laser simulation.

Keywords : 3D model, data modeling, PostGIS, VRML, laser radar

1. 서 론

3차원 정보를 제공해 주는 LADAR(Laser Radar) 탐색기를 차세대 정밀타격 유도무기의 센서로 이용하는 관련 개발, 연구가 활발하게 진행되고 있다. 이러한 LADAR 탐색기를 장착한 유도무기를 개발하고 실험하기 위하여 실제로 LADAR 탐색기가 장착된 유도무기를 사용한다면 천문학적인 비용을 감수해야 하고 다양한 표적에 대하여 시험하기는 매우 어렵다. 따라서 LADAR 탐색기를 사용하여 표적을 촬영한 것처럼 시뮬레이션 할 수 있는 컴퓨팅 환경을 구축해야 한다. 이렇게 구축된 LADAR 시뮬레이터는 다양한 환경, 표적들에 대하여 시험이 가능하며, 유도무기, LADAR 탐색기, 표적인식 알고리즘 등의 개발과 시험을 용이하게 할 수 있다.

본 연구에서는 LADAR 시뮬레이션을 개발하기 위하여 필요한 기능인 표적인식을 위한 3D 지형 및 표적 모델링의 기능을 공간 DBMS와 VRML을 이용하여 구현하였다. 이를 위해 3차원 지형지물의 위치정보와 속성정보를 다

2008년 5월 21일 접수, 2008년 6월 15일 채택
* 서울시립대학교 대학원 공간정보공학과 석사과정 (nani0809@uos.ac.kr)
** 교신저자·정회원·서울시립대학교 공간정보공학과 교수 (cmjun@uos.ac.kr)

면체와 다면체를 구성하는 면으로 이루어지도록 데이터 모델링을 실시하였다. 또한 표적인식에 필요한 공간 객체의 정보를 DB에 저장하고, 이 저장된 정보를 공간 DBMS의 공간 연산을 이용하여 시뮬레이터에 제공하고, 공간 객체의 가시화 및 시뮬레이션 결과를 가시화하는 과정을 제시하였다.

2. 연구방법의 개요

레이저레이더 시뮬레이션은 가상의 비행물체에서 레이저신호를 지상의 객체에 송수신하여 3차원 객체의 속성 및 위치정보를 획득한다. 하지만 시뮬레이션에서 이용되는 실험 범위는 광범위하고 실험 객체들도 복잡한 형태를 지닌다. 일반적으로 사용하는 3차원 모델링 툴들은 파일기반구조를 가지며 객체들을 한꺼번에 메모리에 올려놓고 가시화를 하게 되어 객체수가 많거나 광범위한 지역을 다룰 때에는 메모리 overflow 문제가 발생하거나 시뮬레이션 속도가 느려지는 문제가 발생할 수 있다. 또한 시뮬레이션 과정 중 실제 비행물체의 속도에 맞춰서 레이저신호와 교차하는 건물이나 객체만을 빠르게 추출해야 하는데 일반적인 3차원 모델링 툴들은 이러한 연산자나 질의 기능을 제공하지 않는다. 따라서 본 연구에서는 시뮬레이션에서 사용될 모든 3차원 지형과 사물의 정보를 공간 DB에 저장하고 비행물체의 이동 범위에 해당하는 일부의 3차원 객체만을 검색하여 해당 정보들을 실시간으로 가시화해주는 방법을 채택하였고, 레이저신호와 교차하는 객체의 속성정보를 획득하기 위하여 공간 DB에서 제공하는 공간 연산을 이용하는 방법을 채택하였다.

본 연구는 그림 1과 같은 과정으로 진행하였다. 레이저레이더 시뮬레이션에서는 정확한 3차원 표적 모델과 환경 모델을 효율적으로 관리하고 이들을 조합, 선택하여 다양한 3차원 표적 모형 공간을 구성해야 한다. 본 연구에서는 3차원 객체로서 지형, 건물, 탱크를 시범적으로 채택하여 진행하였다. 이러한 3차원 객체들을 공간 DB에 저장, 활용할 수 있도록 데이터 모델링 방안에 대하여 연구하였고, 3차원 정보를 자동으로 획득하고, 공간 DB에 저장하는 방안에 대한 연구를 진행하였다. 시뮬레이션에서 사용할 레이저신호를 위한 공간 연산 쿼리에 대한 방법과 쿼리 결과로 획득한 정보를 VRML을 이용하여 가시화하는 방법에 대하여 제시하였다.

본 연구에서 연구의 범위는 영등포구의 일부지역을 연구범위로 지정하였고, 수치지도와 도화원도를 이용하여 지형과 건물을 구성하는 점의 좌표를 자동으로 획득하고 탱크는 기존에 제작된 VRML 탱크 모델의 점의 좌표 값

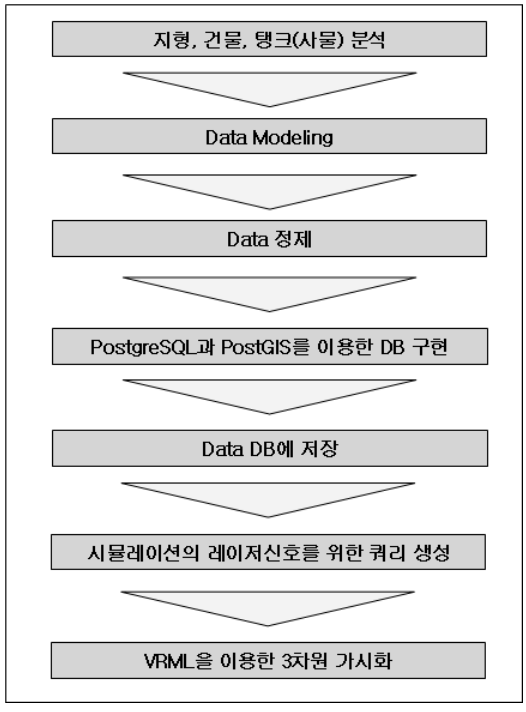


그림 1. 연구 프로세스

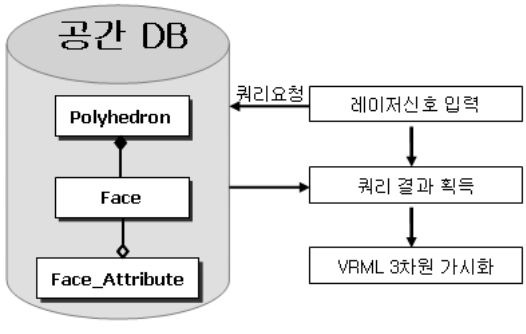


그림 2. 테스트 시스템 구조도

을 사용하였다. 이렇게 획득한 데이터를 공간DBMS에 적용하여 그림 2와 같이 실험 테스트 해보았다.

3. Data Modeling

레이저레이더 시뮬레이션은 3차원의 표적에 레이저 신호를 발사하여 표적의 반사 성질을 획득하고, 사물의 반사도를 기반으로 표적의 정보를 획득하게 된다. 이를 위해서는 3차원 지형, 지물을 면(face)단위로 데이터를 모델링하여 표적의 면과 레이저신호가 교차할 수 있도록

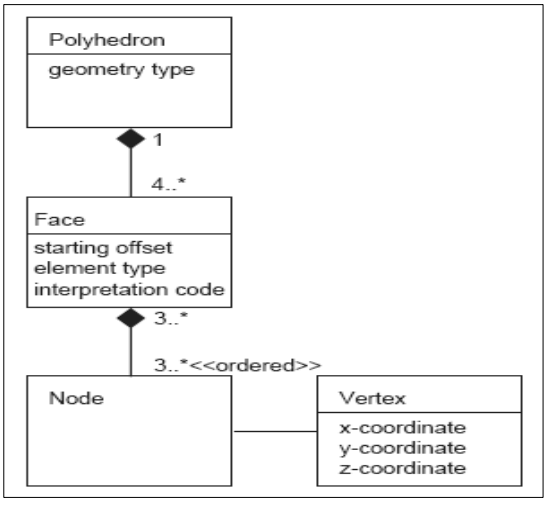


그림 3. 다면체 저장을 위한 UML 클래스 다이어그램 (Zlatanova, 2000)

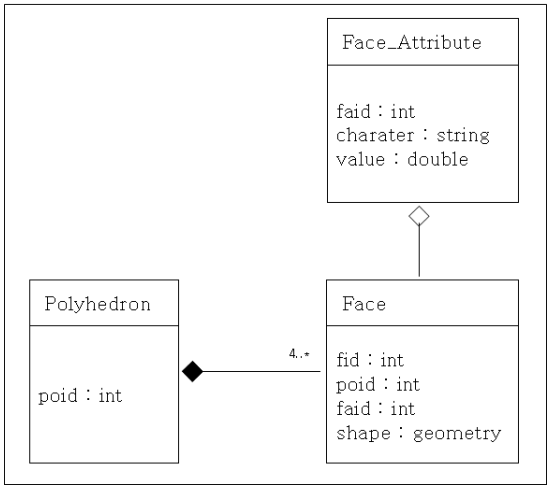


그림 4. 3차원 지형지물의 UML 클래스 다이어그램

하여야만 한다.

3차원 데이터 모델링과 관련된 최근 연구를 살펴보면 Zlatanova(2000)가 제시한 데이터 모델링 기법을 들 수 있다. Zlatanova는 3D 토폴로지를 구현하기 위해서 그림 3과 같이 다면체, 다면체를 이루고 있는 면, 면을 구성하고 있는 점의 위상적 관계를 정의하였다. 이러한 데이터 모델링 기법은 3차원 공간 객체를 2D 공간 DBMS로의 통합 연구(Stoter, 2002), DBMS에서 3차원 오브젝트들의 가시화(Stoter, 2003)의 연구와 공간 DBMS에서의 3차원 공간 오브젝트의 모델링에 관한 연구(Arens, 2005) 등의 연구에서도 유사하게 적용하고 있다.

하지만 이와 같이 객체들의 관계를 정의하면 실제 질의를 수행 할 때, face의 점 정보를 얻기 위해 node, vertex로의 조인을 두 단계 더 하게 됨으로서 쿼리 결과의 획득이 늦어져 전체 시뮬레이션 성능의 저하를 가져올 수 있다. 또한 레이저 시뮬레이션에서는 Zlatanova의 연구에서와 같은 건물 뿐 아니라 탱크와 같은 복잡한 형태(약 1000개 이상의 면으로 구성)도 포함되며 광범위한 지역에 걸쳐 많은 객체를 대상으로 한다.

따라서 본 연구에서는 그림 3의 완전한 토폴로지 구조를 일부 역정규화를 하여 그림 4와 같이 하나의 다면체와 다면체를 이루고 있는 면들의 관계를 정의하여 부분적인 토폴로지 형태를 갖도록 하였다. 이러한 데이터모델링을 활용하여 실제 질의를 수행할 때, face-node-point간의 조인질의를 생략하고 polyhedron과 face의 조인질의만 이루어질 수 있도록 함으로써 데이터의 접근 성능을 향상시켜 시뮬레이션의 수행 속도를 높이고자 하였다.

건물과 탱크는 하나의 다면체와 다면체를 이루고 있는 면으로 구성하였고, 지형 정보는 TIN을 기반으로 지형 표면이 면들로 구성되도록 하였다. 그리고 각각의 면은 반사도와 같은 속성 값을 갖도록 모델링하였다. 그림 4의 'Face_Attribute'는 객체면에 저장되는 반사도와 같은 성질을 의미한다. 표 2는 건물이 3D Cube 일 때의 face 테이블의 값을 나타낸다. 그림 5는 Polyhedron과 Face 테이

표 1. 건물이 3D cube 일 때 face 테이블

FACE table			
FID	BID	AID	SHAPE
1 (lower face)	1	aid	polygon((x4,y4,z4, x3,y3,z3, x2,y2,z2, x1,y1,z1, x4,y4,z4))
2 (side 1)	1	aid	polygon((x3,y3,z3, x4,y4,z4, x8,y8,z8, x7,y7,z7, x3,y3,z3))
3 (side 2)	1	aid	polygon((x4,y4,z4, x1,y1,z1, x5,y5,z5, x8,y8,z8, x4,y4,z4))
4 (side 3)	1	aid	polygon((x1,y1,z1, x2,y2,z2, x6,y6,z6, z5,y5,z5, x1,y1,z1))
5 (side 4)	1	aid	polygon((x3,y3,z3, x2,y2,z2, x6,y6,z6, z7,y7,z7, x3,y3,z3))
6 (upper face)	1	aid	polygon((x5,y5,z5, ,x6,y6,z6, x7,y7,z7, z8,y8,z8, x5,y5,z5))

```
Create table Polyhedron(
        poid int PRIMARY KEY);

Create table Face (
        fid int PRIMARY KEY,
        poid int,
        aid int,
        shape geometry);
```

그림 5. Polyhedron과 Face 테이블 생성 SQL 정의

```
505
18
184.015625  444.437500  23.110000
183.218750  444.343750  23.110000
(중략)
181.734375  456.406250  16.280380
19
184.015625  444.437500  22.700000
(중략)
182.515630  264.093750  20.000000
182.515630  265.187500  20.000000
```

그림 6. 도화원도에서 추출한 건물 데이터의 텍스트 파일 형식

블을 생성하는 테이블 정의문이다.

4. 지형지물의 데이터 획득

본 연구에서는 지형, 건물, 탱크 이렇게 3가지의 데이터를 사용하여 실험하였다. 대상 지역으로는 영등포구의 일부 지역을 선택하였고, 해당 범위의 지형과 건물데이터를 사용하였다. 지형 정보는 수치지도의 등고선과 표고점 레이어를 선택하여 지면의 고도 값을 추출한 후, TIN을 생성하였으며, 건물의 좌표 정보는 도화원도에 포함된 지면 및 건물의 지붕면 모델 및 벽면 모델의 좌표를 자동으로 텍스트 파일로 추출 생성하였다(김성준, 2007). 탱크의 정보는 기존 탱크 VRML모델의 코드에서 사용된 좌표 값을 사용하였다. 추출된 지형과 지물을 구성하는 점의 좌표 정보들은 텍스트 파일에 따로 저장하였고, DB에 지형지물의 정보를 삽입할 때 사용하였다.

지형의 좌표(TIN)는 그림 7과 같이 TIN을 구성하는 점의 좌표 값(X, Y, Z)과 TIN을 구성하는 3개의 인덱스 값으로 이루어진 2개의 텍스트 파일을 생성하였다. 인덱스 값은 지형 포인트 좌표 값을 나열한 텍스트 파일에서 나오는 좌표 값의 순서대로 번호를 부여하였다.

건물의 좌표 추출은 바닥면은 지형과 만나기 때문에 추출에서 제외하였고, 건물의 지붕면과 옆면만을 추출하였다. 건물 좌표는 그림 6과 같이 첫줄에 실험에 사용될 전체 건물의 개수가 주어지고, 그 다음 줄에 한 건물의 지붕면의 vertex의 수가 주어진다. 그 밑에 건물의 지붕을 이루고 있는 포인트들의 좌표 값과 건물 옆면을 이루고 있는 포인트 좌표 값들이 나열되고, 한 건물의 지붕면과 옆면의 포인트 좌표 값이 다 나타내어지면 그 다음 건물의 vertex의 수가 주어지고 지붕면과 옆면을 이루고 있는 포인트의 좌표 값을 제공하는 방식으로 마지막 건물까지 반복하여 텍스트 파일에 나타나게 된다.

탱크와 같은 경우에는 기존에 제작된 VRML 파일을 이용하여 VRML 코드에서 사용한 좌표 값을 추출하였다. 탱크의 VRML 모델 코드 형식을 살펴보면 우선 탱크 모델을 생성하기 위해서 사용된 모든 포인트의 좌표 값을 나열하고, 폴리곤을 이루는 포인트들의 인덱스 값을 나열한다. 인덱스 값은 텍스트 파일에서 나열되는 포인트 좌표 값의 순서대로 0번을 시작으로 번호를 부여하게 된다. 탱크의 VRML 모델 코드의 coordIndex 명령어는 면을 그리기 위해 세 개 이상의 인덱스 값으로 구성하고, 마지막 값은 항상 '-1'로서 면을 그리는 명령어이다(박경

포인트 좌표 값			TIN을 구성하는 포인트 번호		
X	Y	Z	Point_1	Point_2	Point_3
182.484375	496.375000	14.990018	5252	5905	6087
177.250000	497.906250	14.990018	5252	6509	6087
177.343750	498.250000	14.990018	6158	5522	10573
177.109375	498.312500	14.990018	692	5252	5748
173.562500	499.500000	14.990018	692	5252	5905
195.265625	495.718750	14.727057	10048	5252	6509
183.578125	498.937500	14.727057	7354	6158	5522
458.078125	497.843750	14.900414	5711	5710	5707
(이하 생략)			(이하 생략)		

그림 7. 지형의 좌표 값 및 TIN을 이루는 좌표 번호

배, 2006). 탱크 VRML 모델의 코드에서 사용된 포인트 좌표 값들과 coordIndex 명령어에 포함된 인덱스 값을 이용하여 폴리곤을 구성하는 포인트의 좌표 값을 획득할 수 있다.

5. 공간객체의 DB 저장

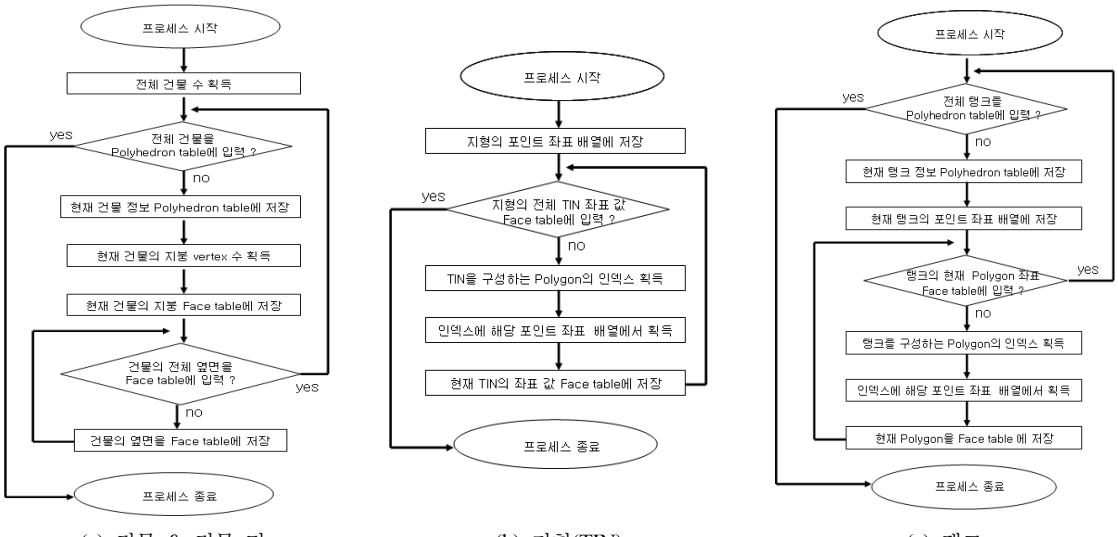
본 연구에서 사용된 3차원 객체들은 객체의 수도 많고 각각이 수많은 면들로 구성되어진다. 이러한 면들의 위치와 속성정보들을 SQL 명령어를 이용하여 수작업으로 DB에 입력하기에는 불가능하다. 따라서 본 연구에서는 지형과 지물을 구성하는 면들의 좌표 정보가 있는 텍스트파일과 VRML을 이용하여 위치정보를 읽고, DB에 저장하는 과정을 자동화하였다.

하나의 면은 반사도와 같은 속성 값을 갖도록 DB가 설계되었지만 본 연구에서는 다면체의 쿼리와 가시화에 중점을 두었기 때문에 실제 속성 값의 입력은 테스트에서 제외하였다. 향후 연구에서는 건물 각각의 면에 반사도와 같은 실제 속성 값을 적용하여 실험할 예정이다.

건물 및 건물의 면의 정보를 DB 테이블에 저장하는 프로세스는 그림 8의 (a)와 같다. 시뮬레이션에 사용 할 건물 전체의 수를 획득한다. 그리고 건물의 지붕면의 vertex 수를 획득하고 지붕의 포인트 좌표 값을 이용하여 폴리곤의 형태로 Face 테이블에 저장한다. Vertex의 개수만큼 옆면의 포인트 좌표 값을 이용하여 폴리곤의 형태로 Face 테이블에 저장한다. 위 과정을 건물 전체의 수만큼 반복하여 전체 건물과 건물 면의 정보를 DB 테이블에 저장한다.

TIN의 면 정보를 DB 테이블에 저장하는 프로세스는 그림 8의 (b)와 같다. 지형을 나타내는 TIN의 정보는 TIN을 구성하는 포인트들의 좌표 값들을 나타내는 텍스트 파일과 TIN을 구성하는 포인트들의 인덱스 값을 나타내는 텍스트 파일로 나누어져 있다. 따라서 TIN의 면 정보를 Face 테이블에 저장하는 과정은 포인트들의 좌표 값들을 나타내는 텍스트 파일에서 좌표 값들을 전부 추출하여 임시로 만든 배열에 저장한다. 다음 과정으로 TIN을 구성하는 포인트들의 인덱스 값을 나타내는 텍스트 파일에서 인덱스 값들을 읽고 인덱스에 해당하는 포인트의 좌표 값을 추출하여 Face 테이블에 폴리곤의 형태로 저장한다. 위 과정을 모든 TIN의 폴리곤이 Face 테이블에 저장 될 때까지 반복 수행한다.

그림 8의 (c)는 탱크의 정보와 탱크의 면 정보를 DB 테이블에 저장하는 과정을 나타낸다. 탱크의 자료는 VRML 파일의 코드에 존재하는 포인트들의 좌표와 인덱스를 이용하여 탱크의 정보를 DB 테이블에 저장한다. VRML 파일 코드에서 포인트들의 좌표를 전부 추출하여 임시로 만든 배열에 저장한다. 그리고 VRML 파일 코드에서 탱크의 면을 구성하는 포인트들의 인덱스 값들을 하나의 면 단위로 읽고, 각각의 인덱스에 해당하는 포인트의 좌표 값을 추출하여 Face 테이블에 폴리곤의 형태로 저장한다. 위 과정을 탱크의 모든 폴리곤이 Face 테이블에 저장 될 때까지 반복 수행한다.



(a) 건물 & 건물 면 (b) 지형(TIN) (c) 탱크

그림 8. 3차원 지형지물 정보 DB 저장 프로세스

6. 쿼리를 이용한 공간분석

DBMS의 주류를 이루고 있는 Oracle, IBM DB2, Informix, Ingres와 비 상업화 DBMS인 PostgreSQL과 MySQL은 공간 데이터 타입과 공간 함수들을 OGC의 SQL을 위한 Simple Features Specification을 기반으로 더 확장하거나, 축소하여 구현하고 있다. 그리고 포인트, 라인, 폴리곤과 같은 공간형상의 저장, 복원, 질의, 업데이트를 제공한다 (Stoter, J.E, 2003). 현재 오픈소스로 제공되는 공간 DBMS 중 PostgreSQL와 PostGIS는 비교적 안정적인 성능 및 기능을 지원하는 것으로 평가되고 있다. 따라서 본 연구에서는 사물의 3차원 위치 정보 및 속성정보를 저장, 쿼리하는데 PostgreSQL와 PostGIS를 사용하였다.

PostgreSQL은 오픈소스로 제공되고 있는 객체-관계형 데이터베이스 관리시스템(ORDBMS)의 일종이다. PostgresSQL은 기본적으로 소스수준까지의 정보도 공개를 하며, 사용, 복사, 수정, 배포에 있어서 자유로우며 무료로 구할 수 있고, 객체-관계형 데이터 모델과 ADT에 기반한 확장성의 특징을 가지고 있다.(PostgreSQL, 2008) 또한 PostGIS는 PostgreSQL에서 지리적 객체들을 저장하고 연산하기 위한 오픈 소스 GIS 소프트웨어 프로그램이다. PostGIS는 OGC의 SQL을 위한 Simple Features Specification을 기반으로 구현되어 있다(PostGIS, 2008).

기본적으로 레이저레이더 시뮬레이션에서 사용되는 레이저 신호와 같이 시작점과 끝점의 좌표가 주어진 직선이 있으면 해당 직선과 교차하는 건물의 정보와 해당 건물 면의 정보를 시뮬레이터에 전달하게 된다. PostGIS에서 제공하는 함수 중 ST_Intersects 함수를 사용하면 시작점과 끝점의 좌표 값이 주어진 직선과 임의의 폴리곤들 중에서 서로 만나는 폴리곤의 정보를 획득할 수 있다 (Paul Ramsey, 2005).

하지만 PostGIS에서는 객체의 좌표 값은 3차원으로 저장을 할 수 있지만, 실제로 3차원 좌표 값을 함수에 대입하고 쿼리문을 실행하여도 결과물은 2차원의 좌표 값만을 적용한 함수의 결과를 얻게 된다. 따라서 ST_Intersects 함수를 사용하여 획득한 건물 면의 정보는 Z값이 무시된 객체들의 2차원 좌표 값을 함수에 이용하여 획득한 결과값을 나타낸다(Stoter, J.E, 2003).

그림 9는 레이저신호에 해당하는 직선과 건물을 나타내는 3D cube의 모습을 나타낸다. Cube를 구성하는 면의 좌표 값을 DB에 저장하고 그림 10과 같은 쿼리문을 이용하여 그림 11의 결과를 획득했다. 결과에서 확인할 수 있듯이 실제로 레이저신호에 해당하는 직선과 만나는 점은 건물의 옆면과 바닥면이지만 그림 11의 결과와 같이 쿼리 결과에서는 건물의 윗면까지 결과로 나타난다.

표적인식 시뮬레이션의 연산 속도와 같은 기능을 향상시키기 위해서는 레이저 신호와 교차하는 건물의 벽면을 결과 값으로 시뮬레이션에 전달해 줘야 하지만 위에서 살펴보았듯이 PostGIS에서 기본적으로 제공하는 함수의 제약으로 인해 레이저신호와 교차하는 정확한 면 단위의 결과 획득이 어렵다. 즉, Z값만을 무시하고 X, Y의 좌표 값만을 이용하여 연산을 이용하기 때문이다. 하지만 X, Y의 좌표 값에 해당하는 정보는 정확히 검색된다. 즉 높이가 주어진 면 단위의 정확한 검색은 불가능하지만 어떤 건물에 포함된 벽면인지는 확인이 가능하다. 따라서

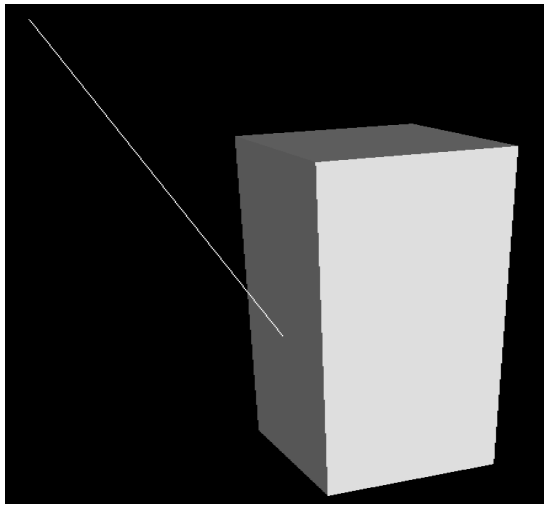


그림 9. 레이저신호와 건물의 예

함수명	쿼리문
ST_Intersects	SELECT ST_AsEWKT(Polygon.shape) FROM polyhedron INNER JOIN Polygon ON polyhedron.phid = Polygon.phid WHERE ST_Intersects(GeomFromEWKT(Polygon.Shape), GeomFromEWKT('linestring(50 50 0,50 200 400)'))=true;

그림 10. 쿼리문에서의 ST_Intersects 함수의 사용 테스트 예

출력창

데이터의 출력 | 해석 | 메세지 | 히스토리

	st_asewkt text
1	POLYGON((0 0 0,100 0 0,100 100 0,0 100 0,0 0 0))
2	POLYGON((100 100 0,0 100 0,100 100 300,0 100 300,100 100 0))
3	POLYGON((0 0 300,100 0 300,100 100 300,0 100 300,0 0 300))

그림 11. 그림 10의 쿼리 실행 결과

본 연구에서는 우선 레이저신호와 교차하는 정확한 건물의 면 단위의 정보의 제공이 아니라 레이저 신호와 교차하는 건물 단위의 정보와 실제 교차하는 건물 면이 포함된 연산결과로 획득한 건물 면들의 정보를 시뮬레이터에 전달한다는 가정 하에 연구를 수행하였다.

7. 3차원 가시화

VRML은 Virtual Reality Modeling Language의 약자로 인터넷 환경에서 상호 작용하는 3차원 환경을 개발하기 위해 제안된 스크립트 언어로 국제 표준(ISO/IEC 14772-1) 3차원 모델링 언어이다. VRML에는 기본적인 입체 도형들이 정의되어 있으며 애니메이션, 사운드 등을 삽입 할 수 있을 뿐만 아니라 상호작용이 가능하여 3차원 가상현실을 구현 할 수 있다(정경배, 2006).

OpenGL이나 DirectX와 같은 저수준 그래픽 라이브러리 일수록 세부적인 표현이 가능하고 성능도 뛰어나지만 개발하기가 어려운 점이 있다. 즉, 간단한 3D 지형이나 사물을 표현하기 위해서는 하드 코딩을 해야만 하고, 그 방법 또한 복잡하고 어렵다. 반면에 VRML은 프로그래밍 경험이 없는 개발자라도 단시간에 프로그래밍 문법을 이해할 수 있으므로 쉽게 접근할 수 있는 언어 중에 하나이다(황철희, 2006). 따라서 본 연구에서는 저수준 그래픽 라이브러리보다 사용하기가 편리하고 초보자도 비교적 쉽게 3차원 장면을 생성해낼 수 있는 VRML을 우선적으로 연구에 적용하여 실험을 하였다.

본 연구에서는 DB에 저장된 객체들의 정보들을 자동으로 VRML 파일로 변환하는 방법을 개발하여 사용하였다. DB에 저장된 지형, 건물, 탱크의 폴리곤을 구성하는 포인트 좌표 값과, 이 포인트들의 인덱스 값들을 VRML의 coordIndex 명령어를 사용하여 VRML 파일로 변환하였으며, 이렇게 변환한 VRML 파일을 이용하여 가시화 테스트를 진행하였다. 그림12는 DB에 저장된 건물의 정보를 이용하여 나타낸 것이고, 그림 13은 지형과 건물을 동시에 가시화한 결과물이다. 그림 14는 DB에 저장된 탱크의 정보를 VRML을 이용하여 표현한 결과물이다.

지형과 건물은 같은 좌표체계를 공유하므로 하나의 VRML 파일에서 가시화 할 수 있지만, 탱크와 같은 경우에는 기존의 VRML 모델의 좌표 값을 그대로 가지고 있기 때문에 지형과 좌표체계가 일치하지 않는다. 따라서 본 연구에서는 DB에 저장된 탱크의 정보만을 이용하여 가시화 테스트 해보았고 추후 연구를 통하여 탱크의 좌표체계를 지형의 좌표체계로 변환하여 지형과 함께 가시화하는 연구를 진행할 예정이다.

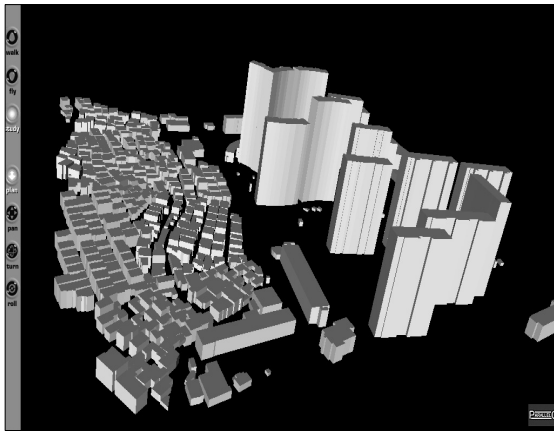


그림 12. VRML을 이용한 건물의 가시화

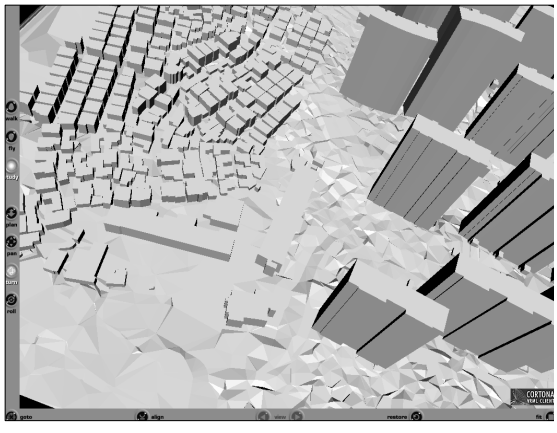


그림 13. VRML을 이용한 지형과 건물의 가시화

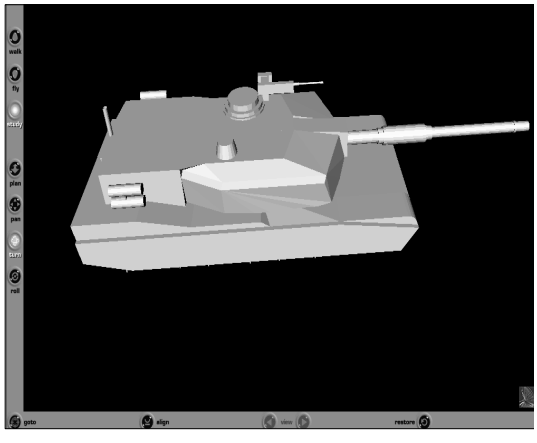


그림 14. VRML을 이용한 탱크의 가시화

8. 실제 데이터를 적용한 실험 테스트

레이저레이더 시뮬레이션에서 레이저신호에 해당하는 임의의 직선의 시작점과 끝점이 주어졌다고 가정하고 본 연구에서 구축한 DB를 이용하여 공간검색을 수행해 보았으며 검색결과로 획득한 정보를 이용하여 가시화 테스트를 하였다.

그림 15와 같이 레이저 신호에 해당하는 직선이 임의의 건물과 교차할 때 그림 16은 실제 구축한 DB와 가상의 레이저신호인 직선의 양 끝점의 좌표를 ST_Intersects 연산에 대입하여 질의한 쿼리문이다. 그림 17은 그림 16의 쿼리 실행 결과를 나타낸다.

또한 그림 16의 쿼리 결과로 획득한 그림17의 건물 면의 정보를 기존 건물의 VRML 파일에 포함해서 가시화 테스트 해보았다. 그림 18은 쿼리를 이용하여 획득한 건물의 면을 다른 색으로 표시하여 기존 건물의 VRML 파일에 포함하여 표현한 것이다.

본 테스트를 통하여 레이저신호에 해당하는 직선의 좌표 값과 구축된 DB에서 ST_Intersects 함수를 이용하여 레이저신호의 직선과 교차하는 건물의 면의 정보를 획득하고자 하였다. ST_Intersects의 연산을 통하여 레이저신호와 만나는 정확한 면의 정보는 획득할 수는 없지만, 어떤 건물과 레이저신호가 교차하는지는 정확히 확인할 수 있고, 건물을 구성하고 있는 벽면 중에서 레이저신호와 만나는 몇 개의 벽면을 후보군으로 추출할 수 있었다. 그리고 그림 17과 같이 레이저 신호가 건물의 지붕면을 통과하면 본 연구에서는 건물의 바닥면은 고려하지 않았기 때문에 레이저신호와 교차하는 정확한 지붕면만을 검색할 수 있었다. 이렇게 추출한 벽면의 후보군만을 시뮬레이터의 입력데이터로 적용하여도 연산에 큰 도움이 된다. 또한 이렇게 추출한 벽면을 기존의 건물의 VRML 파일에 추가하여 가시화함으로서 레이저레이더 시뮬레이터의 가시화 기능까지도 제공할 수 있다.

9. 결 론

본 연구에서는 3차원 표적을 공간 DB에서 활용할 수 있도록 하나의 다면체와 다면체를 이루고 있는 면들의 관계를 정의하여 부분적인 토폴로지 형태를 갖도록 하였다. 이러한 데이터모델링을 활용하여 실제 질의를 수행할 때, polyhedron과 face의 조인질의만 이루어질 수 있도록 함으로써 데이터의 접근 성능을 향상시켜 시뮬레이션의 수행 속도를 높이고자 하였다. 그리고 3차원 지형지물의 정보를 수치지도, 도화원도, 탱크의 VRML 파일에서 자동으로 추출하고, 추출한 지형지물의 정보를 공간 DB

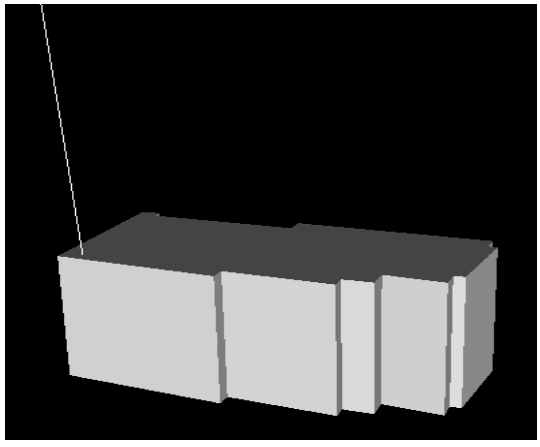


그림 15. 레이저 신호와 교차하는 건물 모델

함수명	쿼리문
ST_Intersects	SELECT fc.fid, ST_AsEWKT(shape) FROM face fc INNER JOIN polyhedron ph on fc.poid = ph.poid WHERE ST_Intersects(GeomFromEWKT(fc.Shape), GeomFromEWKT('linestring(184 443 23, 185 445 150)'))=true;

그림 16. 구축한 DB에서의 ST_Intersects 함수의 사용 예

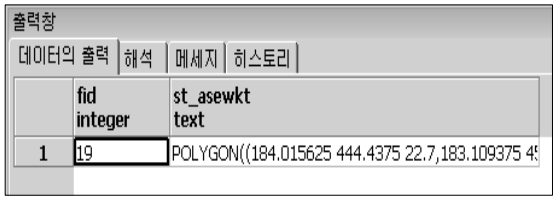


그림 17. 그림 16의 쿼리 실행 결과

그림 18. 쿼리 결과를 가시화한 결과

에 저장하였다. 또한 PostGIS에서 제공하는 공간 연산을 이용하여 레이저 신호와 교차하는 건물의 면들을 추출하는 테스트를 하였으며 공간 DB에 저장된 정보를 질의를 통하여 획득하고, 이를 VRML을 이용하여 3차원으로 가시화해 보았다. 테스트 결과, 몇 가지의 문제점을 보완한다면 본 연구에서 제시한 방법을 레이저레이더 시뮬레이션에서의 표적인식과 공간 및 표적의 가시화 부분에 활용할 수 있는 것으로 판단된다.

PostGIS에서 기본적으로 제공하는 함수가 2차원의 좌표만을 지원함으로서 실제 획득 가능한 객체의 정보는 제한적이었다. 따라서 향후 연구에서는 3차원 좌표 값을 연산에 반영할 수 있는 함수의 제작이나, 기존의 공간 DBMS에서 3차원 연산을 제공하는 DBMS를 이용하여 연구할 예정이다. 또한 VRML을 이용하기에는 가시화 수행 시간이 오래 걸렸고, 연구범위를 소규모로 테스트 하였지만 메모리 overflow 문제가 발생하였다. 따라서 시뮬레이션의 수행속도와 실시간 가시화의 속도를 맞추고 메모리 overflow문제를 해결하기 위하여 저수준 그래픽 라이브러리인 OpenGL을 이용하여 연구를 수행할 예정이다. 추후 연구에서 이러한 문제점들을 보완한다면 레이저레이더 시뮬레이션뿐 아니라 3차원모델링 분야에 적용할 수 있을 것이라 판단된다.

감사의 글

본 연구는 한국과학기술원 영상정보특화연구센터의 II-21 "레이저 영상 신호특성모델링 기법 연구" 과제를 통한 방위사업청과 국방과학연구소의 연구비 지원으로 수행되었습니다.(계약번호 UD070007AD)

참고문헌

1. 김성준, 이임평, 민성홍, 이동천, 박진호, 2007, "도화원도를 이용한 3차원 건물모델의 자동생성", *한국지형공간정보학회*, 제 15권, pp. 3-14
2. 박경배, 2006, *가상현실을 위한 VRML & X3D*, 21세기사, p.70
3. 황철희, 2006, "비트맵 이미지를 이용한 VRML 제작에 관한 연구", *한국사진학회지 AURA*, 제 15권, pp. 14-22
4. Arens, C., Stoter, J.E., and van Oosterom, P.J.M., 2005, "Modelling 3D spatial objects in a geo-DBMS using a 3D primitive", Computers & Geosciences, Vol. 31, No. 2, pp. 165-177
5. Paul Ramsey, 2005, "PostGIS Manual", version 1.3.2, pp. 40
6. Stoter, J.E., and van Oosterom, P.J.M., 2002, "INCORPORATING 3D GEO-OBJECTS INTO A 2D GEO-DBMS", ACSM-ASPRS 2002 ANNUAL CONFERENCE PROCEEDINGS
7. Stoter, J.E., Zlatanova, S., 2003. "Visualising and editing of 3D objects organised in a DBMS". Proceedings EUROSDR Workshop : Rendering and Visualisation, pp. 14-29
8. Zlatanova, S., 2000, "3D GIS for urban development", PhD thesis, TUGraz , Austria, p.222
9. PostGIS, 2008, URL:http://postgis.refractions.net/documentation/
10. PostgreSQL, 2008, PostgreSQL Global Development Group, URL:http://www.postgresql.org/about/