과제번호 : 97-NI-01-03-A-04

국가 지리정보시스템 기술개발 사업

Development of National GIS Technology

GIS용 User Interface 기술개발

Development of User Interface Modules for GIS

서울대학교 사회과학연구원

과 학 기 술 부

제 출 문

과학기술부 장관 귀하

본 보고서를 "국가 지리정보 시스템 기술개발 사업" 과제 (세부과제 "GIS용 User Interface 기술개발")의 보고서로 제출합니다.

2000. 1. 31.

주관연구기관명 : 서울대학교 사회과학연구원

주관연구책임자 : 박 기 호
연 구 원 : 정 재 곤
유 은 혜

여 백

요 약 문

Ⅰ. 제 목

국가 지리정보 시스템 기술개발 사업 (GIS용 User Interface 기술 개발)

Ⅱ. 연구개발의 목적 및 필요성

본 과제는 고딕을 기반으로 한 소프트웨어 개발 중과제가 특정 벤더의 제품에 의존함으로써 향후 시장성이 불투명해짐에 따라 독립적인 차세대 개방형 GIS 시스템 아키텍쳐의 설계와 이를 통한 시스템 개발 틀(framework)을 제시하는 것이 필요하다는 판단 아래 차세대 GIS 시스템 개발을 통한 기반기술 확보 전략을 채택하였다.
차세대 GIS 시스템 아키텍쳐는 인터넷/인트라넷으로 대표되는 현재의 컴퓨팅 환경에 가장 잘 적응할 수 있어야 하며, 관련 표준의 준수를 통해 상호운용성을 확보할 수 있어야 한다.
본 과제는 GIS 도메인에서의 상호연동성과 관련된 기술기반 확보를 위해 그 핵심이 되는 OpenGIS 명세를 구현하고, 이를 통해 실제 운용 가능한 개방형 GIS 시스템을 설계하는 데 그 목적이 있다.

Ⅲ. 연구개발의 내용 및 범위

본 과제는 실제 운용 가능한 차세대 개방형 GIS 시스템 아키텍쳐를 설계하기 위해 분산 환경 하에서의 어떤 노드에서도 실행 가능한 매핑 커널을 설계, 구현하는 데 중점을 두었다.
코바/자바 기반의 오브젝트 웹을 목표로 매핑 커널이 수용해야 하는 기본적 시스템 요구사항들을 추출하였으며, 잘 알려진 디자인 패턴과 UML을 설계 과정에 도입하였다. 매핑에 필요한 여덟 가지 핵심 모델을 기반으로 설계된 매핑 커널, OpenViews는 100% 순수 자바로 구현되어 확장성과 유연성이 뛰어나며, 사용자 정의 심볼과 같이 프로그래머가 각 응용 프로그램에 매핑 커널을 최적화시킬 수 있는 틀을 제공하고 있다.

OpenViews를 내장하여 운용될 수 있도록 설계된 GIS 애플리케이션 서버, PASS는 본 과제를 통해 설계된 차세대 개방형 GIS 시스템, OpenWorld의 핵심이다. PASS는 SFCORBA 명세를 구현함으로써 상호운용성을 위한 공간 데이터 표준을 준수하도록 하였으며, 심볼과 데이터 제공자를 분산시킴으로써 다양한 네트워크 구성에 유연하게 대처할 수 있도록 하였다. 또한, PASS는 웹 기반의 전자지도 출판을 쉽고 빠르게 처리할 수 있도록 도와주는 역할을 할 수 있다.
매핑 커널과 GIS 애플리케이션 서버의 단위 테스트 및 실제 운용 테스트를 위해 다양한 프로토타입 시스템들을 설계, 구현하였다.

Ⅳ. 연구개발 결과

본 과제에서는 코바/자바 기반의 오브젝트 웹 GIS의 관점에서 새로운 개방형 시스템 아키텍쳐의 필요성을 인식하고 최신 정보 기술을 고려한 개방형 GIS 시스템 아키텍쳐를 제시하였다. 새로운 아키텍쳐의 핵심이 되는 매핑 커널은 100% 순수 자바를 기반으로 하여 ORB를 통해 통신하므로 이식성과 이동성이 뛰어나다. OpenViews 매핑 커널을 내장하도록 한 GIS 애플리케이션 서버 또한 코바/자바 기반의 컴포넌트들로 구성되었으며, 웹 기반의 전자지도 출판이 가능하도록 설계되었다. 특히 OpenViews가 갖는 멀티 뷰 기능을 이용하여 시스템 성능 향상 기법을 고안하였다. 실제 운용 가능성을 알아보기 위한 다양한 프로토타입 시스템이 설계, 구현되었으나, 앞으로도 충분한 벤치마킹 및 시스템 운영 테스트를 통하여 안정 커널을 개발하고 사용자 중심의 편집 툴을 보강하는 작업이 필요하다.

Ⅴ. 연구개발결과의 활용계획

OpenWorld 시스템 컴포넌트들은 앞으로의 확장, 보완을 통하여 코바/자바/OpenGIS 기술을 기반으로 한 분산 환경에서의 전자지도 출판과 퍼스널 GIS를 위한 기본 시스템으로 운영 가능하다. 특히, 새로 부각되고 있는 리눅스 기반의 GIS 애플리케이션 서버 시스템을 안정화, 실용화함으로써 저비용의 공간 데이터 통합 및 서비스가 가능하다는 장점을 가지고 있다.
이와 같은 장점을 최대한 부각시킴으로써 본 과제의 연구 개발 결과가 인터넷/인트라넷 기반의 전자 지도 출판 분야에서 충분히 활용될 수 있도록 전자 지도 서비스가 필요한 분야로의 GIS 시장 진출을 계획하고 있다.

SUMMARY

Ⅰ. Title

Development of National GIS Technology(Development of User Interface Modules for GIS)

Ⅱ. Needs and Purpose

To meet the requirements associated with interoperability for geospatial data access and geoprocessing, much effort to design prototypical systems confirming to OpenGIS™ specification has been made. In the perspective of Object Web GIS, however, current web-mapping technology mainly focuses on either developing a mapping library or a client application regardless of the future needs for interoperability such as an integration of OpenGIS™ standard for CORBA.
Staffs for this research project determined to develop next-generation GIS system architecture to provide a framework for interoperable open GIS system. The architecture should be interoperable through the implementation of associated standards and be operated in the current computing environment.
The purpose of this project is to implement OpenGIS specifications as a core component for interoperability in GIS domain and to design open GIS system architecture which can be operated in the current computing environment.

Ⅲ. Description

This project focuses on the deveopment and implementation of mapping kernel which is able to be run on any node in distributed environment to design next-generation open GIS system architecture.
Basic system requirements which were scheduled to be considered in the design process were extracted and well-known design pattern and Unified Modeling Language were used in the development process. The mapping kernel, OpenViews, designed based on eight essential models needed for

mapping tasks is highly extensible and flexible because it is a 100% pure Java package. It also provides a framework which can be helpful customizing applications(i.e. user-defined symbols).
The GIS application server, PASS, embedding OpenViews is the main component comprising the next-generation open GIS system architecture named OpenWorld. PASS implements spatial data standards for interoperability such as SFCORBA and are very flexbile to meet the requirements from various network topology through the distribution of symbol and data provider objects. PASS can also help publish digital map on the web easily and rapidly.
Many prototype application systems were designed and implemented to test the functionality and operational capability of OpenViews and PASS.

Ⅳ. Result

This research project presents open GIS system architecture using state-of-the-art information technology in consideration of the needs of new open system architecture in the perspective of CORBA/Java based object web GIS
The newly developed mapping kernel which is the core component of the architecture is based on 100% pure Java. It is also highly portable and mobile in case of being communicated via ORB.
The newly developed GIS application server embedding OpenViews mapping kernel consists of many components based on CORBA/Java and is able to publish digital map on the web. It also improves the performance using multi-view functions of OpenViews.
Many prototype application systems were developed and implemented for the unit test and performance test but the system architecture should be more stable as to be used in other projects and be more robust through the implementation of user-oriented editing tools.

Ⅴ. Marketing Strategy

The OpenWorld system can be maintained with low cost if the components are run on the Linux system. With this advantage, we plan to develop a digital map publishing package which can be used on the Linux system as an outstanding GIS application server and a Personal GIS package.

CONTENTS

목 차

제 1 장 서론

제 1 절 연구개발의 필요성

- GIS 소프트웨어 개발(Software Engineering) 분야에 있어서의 기반기술 확보 방안으로는 크게 (1) 기존 시장을 점유하고 있는 시스템의 원천 기술을 확보하는 방법과 (2) 기존 시스템들이 보유하지 못한 새로운 특징들을 기반으로 한 차세대 시스템 기술을 확보하는 방법이 있을 수 있다. 첫번째 방법은 기존 소프트웨어 개발 기술의 확보가 가능하지만 기존 시스템의 한계를 넘어서기 어려운 점이 있으며, 두 번째 방법은 아직 성숙하지 않은 신기술을 대상으로 현실화하는 과정에서 생기는 불확실성이 존재한다. 특히, 새로운 틈새 시장(Niche Market)의 개척과 이를 기반으로 한 GIS 소프트웨어 시장 침투 전략은 새로이 부각되고 있는 신기술들을 개발, 응용해야 한다는 점에서 정보기술의 발전방향과 흐름을 정확히 짚는 것이 우선적으로 요구된다.
- 본 과제는 고딕(Gothic)을 기반으로 한 소프트웨어 개발 중과제가 특정 벤더의 제품에 의존함으로써 향후 시장성이 불투명해짐에 따라 독립적인 차세대 개방형 GIS 시스템 아키텍처의 설계와 이를 통한 시스템 개발 틀(framework)을 제시하는 것이 필요하다는 판단 아래 **차세대 GIS 시스템 개발을 통한 기반기술 확보 전략을 채택**하였다.
- 새로운 정보기술을 이용한 GIS 소프트웨어의 개발을 위해서는 공간 데이터 처리와 관련된 다방면의 핵심 기반 기술들이 필요하다. 실제 소프트웨어 개발 과정(Software Development Process)에서는 공간 데이터베이스(Spatial Database), 공간 데이터 모델링(Spatial Data Modeling), 매핑 기술(Mapping Technology), 미들웨어 기술(Middleware Technology) 등을 통합하는 과정이 필요하며, 이를 위해서는 다양한 서브 시스템(subsystem)들의 개발과 이들의 통합을 가능하게 하는 차세대 GIS 소프트웨어 아키텍처(Next generation GIS software architecture)가 필요하다.
- 차세대 GIS 소프트웨어 아키텍처는 인터넷/인트라넷으로 대표되는 현재의 컴퓨팅 환경에 가장 잘 적응할 수 있어야 하며, 이를 위해 실제 시스템 구현 단계에 적용되는 코바/자바 등의 구현 환경(Environment for implementation)을 포함한 다양한 요소들이 고려되어야 한다. 왜냐하면, 현재 정보기술은 하루가 다르게 발전하고 있으며 코바/DCOM, 자바/ActiveX로 대변되는 새로운 개발환경(Development environment)의 대두는 컴포넌트 프레임워크(Component framework)와 함께 소프트웨어 아키텍처의 혁신을 예고하고 있기 때문이다. 실제로 ESRI, Intergraph 등 대형 GIS 벤더들은 인터넷/인트라넷 환경에 잘 적응하고

사용자들의 다양한 요구 사항을 만족시키기 위해 기존 시스템의 틀을 바꾸거나 새로운 시스템을 개발하고 있다.

- 본 과제에서는 GIS 소프트웨어 개발 분야에 있어서의 기반기술 확보를 위해 다음과 같은 연구과제의 필요성을 제시하였다.

O 차세대 GIS 시스템 아키텍쳐 제시의 필요성

틈새 시장을 공략하는 가장 좋은 방법은 새로운 정보기술의 확보라는 전제아래 웹/인터넷/인트라넷으로 대변되는 현재의 분산 컴퓨팅 환경에 효과적으로 적용가능한 차세대 GIS 시스템 아키텍쳐를 설계하는 것이다. 이는 소프트웨어 운영 환경이 변화하는 경우 그 변화에 가장 잘 적응할 수 있는 견고한 시스템 아키텍쳐만이 살아남을 수 있다는 인식에 기인한다.

O 실제 운영 가능한 프로토타입(prototype) 시스템 개발의 필요성

공간 데이터와 지리공간 정보 처리 워크플로우(geo-spatial information processing workflow)는 타 정보기술 분야에서 성숙한 핵심 기술들이 효과적으로 응용되는 것이 바람직하므로 그러한 핵심 기술들을 충분히 응용한 실제 운영 시스템을 설계, 구현하는 것이 필요하다. 이를 통해, 새롭고 안정적인 정보기술이 응용된 공간 데이터 처리 관련 기반 기술을 확보할 수 있다.

- 설계 및 개발된 시스템은 분산 컴퓨팅 및 공간 데이터 처리와 관련된 표준을 채택함으로써 타 시스템과의 호환성을 높일 수 있어야 한다. 즉, CORBA 및 OpenGIS 사양을 준수하는 명세(Specification)를 구현함으로써 표준 기술에 기반한 시스템을 설계할 필요가 있다.

제 2 절 연구개발의 목적

- 본 과제는 GIS 도메인에서의 상호연동성과 관련된 기술기반 확보를 위해 그 핵심이 되는 OpenGIS 명세를 구현하고, 이를 통해 실제 운용 가능한 개방형 GIS 시스템을 설계하는 데 그 목적이 있다.
- 차세대 GIS 시스템 아키텍쳐[1]는 미들웨어 인터페이스 표준인 OpenGIS에 기반하고 있으며, 이를 통해 상호운용성을 보장하는 전략을 취하고 있다. 시스템 아키텍쳐 설계는 기본적인 단위 테스트를 거치므로 제시된 아키텍쳐에 기반하여 구현된 실제 운용 시스템 프로토타입은 시스템의 성능과 시장 진출 가능성을 평가하는 데 사용될 수 있다.

---

[1] 아키텍쳐의 이름은 OpenWorld이다.

- 인터넷/인트라넷에서 공간 정보를 다루는 모든 사람들이 편리하게 사용할 수 있는 핵심 라이브러리들을 설계, 구현함으로써 전자지도 편집, 출판과 관련된 소프트웨어 개발 및 서비스에 드는 비용을 절감할 수 있다. 따라서, 개발 우선 순위가 높은 필수 라이브러리들을 중심으로 설계되고 지속적으로 확장될 수 있는 시스템을 현실화함으로써 시스템의 가용성을 높이는 전략을 취한다.

제 3 절 연구개발의 범위

1. OpenGIS/CORBA 기반 아키텍쳐 설계

- OpenGIS 명세는 IT/GIS 분야의 기술 변화에 적응하기 위하여 꾸준히 업데이트되고 있으며, 따라서 그 명세의 구현은 일정 시점에서의 초기 구현과 함께 지속적인 유지, 관리가 필요하다.
- 본 과제에서는 현재 활발히 진행되고 있는 OpenGIS 표준 명세의 구현[2]과 일치성 검증(Conformance Test)을 통해 GIS 툴의 국제적 산업표준에 맞는 차세대 GIS 개발에 필요한 기반요소를 확보한다. 본 과제에서는 특히 다음 사항들에 중점을 둔다.

  O Feature, Geometry 명세 구현

  OpenGIS 명세에 준하는 Feature, Geometry 관련 클래스의 구현을 통해 실제 운용 가능한 시스템을 만들기 위한 여러 가지 요구사항 및 조건들을 검토한다.

  O 필요한 클래스의 확장 구현

  검토된 보완사항에 근거하여 클래스를 확장 구현함으로써 운용을 위한 요소들을 점검한다.

  O 단위 테스트 및 구현 클래스에 기초한 시스템 설계

  설계/구현 클래스들에 기초하여 OpenGIS 기반 개방형 GIS 시스템[3]을 설계한다.

2. 설계 아키텍쳐에 기반한 GIS 컴포넌트 개발

---

[2] 본 과제의 경우 OpenGIS 기반 아키텍쳐 설계를 위한 기본 검토가 1998년경에 이루어져 1998년 현재 사용가능한 명세를 기준으로 초기 구현을 시작하였다.

[3] 본 과제에서 언급하는 개방형 GIS 시스템의 가장 큰 특징은 코바/자바에 기반함으로써 차세대 오브젝트 웹 환경에 빠르게 적응할 수 있다는 점이다.

- 소프트웨어 기술의 가장 영향력있는 패러다임(paradigm)인 분산 컴퓨팅(Distributed computing)/객체지향(Object orientation) 등에 기반하여 본 과제에서 설계된 시스템상에서 확장된 컴포넌트들을 추가로 구현한다. 시스템의 기본 기능을 확장하는 라이브러리들은 재사용이 용이한 컴포넌트 아키텍처(Component architecture)를 준수하기를 권장한다. 본 과제에서는 특히 다음 사항들에 중점을 둔다.

  ○ **필요한 컴포넌트의 확장 설계**
  WKSGeometry 레이어를 기반으로 하는 애플리케이션 아키텍처에서 운용할 수 있는 컴포넌트 확장 및 개발에 중점을 둔다.

  ○ **협력지원 지리정보 시스템을 위한 컴포넌트 설계**
  코바 이벤트 모델(Event Model)의 활용을 통해 지리정보의 브로드캐스팅(Broadcasting)을 특징으로 하는 전자지도 상황판(Wallboard), CSCW 기술 등을 보완 설계한다.

3. 웹과 연동하는 GIS 툴박스 개발

- 범용 웹브라우저를 통해 각 기관에서 공공성이 높은 지리정보를 서비스할 수 있도록 함으로써 실제 업무 효율성을 높일 수 있도록 한다.
- 센서스 자료나 세균성 이질의 발생현황 등 국민적 관심이 높은 지리정보를 간단하게 웹 상에서 제공할 수 있는 어플리케이션 툴박스(Toolbox)의 개발을 통해 다양한 기관에서 실시간 지리정보 서비스를 가능케 한다. 이를 위해 본 과제에서는 애플리케이션과 애플릿(applet)을 동시에 구현할 수 있는 방안을 모색하며, 설계된 아키텍처 또한 이러한 요구사항을 수용할 수 있도록 한다.

4. 자바기반 공간통계 분석기 개발

- 공간통계 분석기는 공간정보과학(Spatial Information Science)으로서 GIS의 영역확대를 위해 필요한 기능이다.
- 현재 GIS 데이터 및 시스템 구축이 어느 정도 이루어진 분야에서는 공간 데이터를 이용한 분석 기능을 요구하고 있다. 특히, 교통, 산업, 환경 분야에서 GIS 시스템 및 데이터를 효율적으로 사용하기 위해서는 공간통계 기능을 이용한 공간 분석(Spatial analysis)이 필수적이다.

- 본 과제에서는 설계된 시스템에 응용할 수 있는 기본적인 공간통계[4] 기능을 설계 및 구현함으로써 차후 수요가 증대될 수 있는 분석 분야에 유용하게 사용할 수 있도록 한다.

[4] 벡터 데이터 모델 기반의 공간통계 기능을 의미한다.

제 2 장 국내외 기술개발 현황

제 1 절 개방형 GIS 시스템 개발

- 지금까지 개방형 GIS 시스템을 개발하려는 노력은 OpenGIS 컨소시엄에 참가하고 있는 업계 및 공공기관을 중심으로 지속적으로 있어 왔다. OpenGIS 컨소시엄은 컨소시엄 표준을 제정함으로써 각각의 벤더들이 만든 시스템들간의 의사소통이 가능하도록 만드는 역할을 한다. 그러나, 각 벤더들이 추진중인 상호운용 가능한 소프트웨어 개발은 주로 기존의 제품을 OpenGIS 인터페이스에 맞추는 전략을 취하고 있어, 그 개발이 늦어지는 문제가 있어 왔다.
- 현재 공간 데이터베이스의 SQL 구현을 위한 표준 명세는 SDE와 Oracle Spatial이 OpenGIS의 일치성 검사(Conformance Test)를 통과하였으며, OLE/COM 구현을 위한 표준 명세는 여러 업체에서 다양한 제품을 생산해 내고 있다. 그러나, 코바 구현을 위한 표준 명세는 현재 제대로 구현된 경우가 거의 없다. OLE/COM 구현을 위한 표준 명세를 구현하는 작업은 기존 제품의 아키텍처를 표준적인 것으로 변환할 수만 있다면 쉽게 진행될 수 있다. 그러나, 코바 기반의 시스템은 지금까지 없었던 새로운 분산 환경을 요구하므로 많은 수작업을 요한다. 특히, 자바 기반의 시스템은 거의 모든 시스템 컴포넌트를 새로 만들어야 하는 번거로움이 있어 작업이 더디게 진척되고 있는 실정이다.
- C/S 환경에서 분산 환경으로 확장되면서 기존의 RPC로부터 RMI, DCOM, CORBA 등의 기술이 활용되고 있으며, COM, EJB 등의 컴포넌트 프레임워크가 제시되고 있다. GIS 소프트웨어 아키텍처의 설계에 있어서는 객체지향 모델링과 애플리케이션 개발의 접목을 시도하고 있으며, 이를 통해 다양한 사용자 요구사항을 만족시키려는 노력이 전개되고 있다.
- 컴포넌트 기술을 이용한 GIS 소프트웨어 개발 사례로는 ESRI 사의 MapObjects가 대표적으로 현재 버전 2.0에서는 Map, Geometry, Symbol, Address Matching, Spatial Reference 관련 오브젝트들이 ActiveX 오브젝트로 구현되어 있다. 국내에서는 MapObjects [5] 와 유사한 인터페이스를 가지는 MapBase 컴포넌트를 개발하는 프로젝트가 진행중이다.
- 인터넷/웹 기반의 전자 지도 서비스는 현재 그 수요가 크게 늘고 있는 상황으로 크게 서버 중심의 소프트웨어와 클라이언트 중심의 소프트웨어로 구분할 수 있다. 다수의 클라이언트가 접속하는 것을 가정하는 서버 중심의 소프트웨어로는 ESRI 사의 ArcIMS나 MapInfo의 MapExtreme[6] 등이 대표적이며, 자바 애플릿 중심의

[5] http://www.esri.com
[6] http://www.mapinfo.com

클라이언트 소프트웨어로는 OpenMap[7]을 들 수 있다. 국내에서는 Nextel의 NextMap이 개발되어 있으며 여러 기관 및 업체에서 서버 중심의 소프트웨어를 개발 중에 있다.

- 현재 OpenGIS, CORBA, Java, ActiveX 등을 기반으로 하는 GIS 소프트웨어 개발은 대안 모색의 단계를 지나 상품화 단계로 진입하였다고 판단된다.

제 2 절 웹-GIS 서비스 현황

- 개방형 GIS 시스템의 전단(front-end)으로 사용되는 클라이언트는 기존의 애플리케이션 형태 뿐만 아니라 웹에서 바로 접근할 수 있는 애플릿의 형태 또는 HTML 등의 형태를 지원할 수 있어야 한다. 웹 GIS는 이러한 요구사항을 만족시키기 위한 기반 기술을 축적하는 분야라고 할 수 있다. 지금까지 인터넷/웹 분야의 GIS 데이터 서비스는 다양한 방식으로 발전되어 왔는데, 현재 웹GIS의 형태로 지도를 서비스하는 사이트는 그 수가 크게 늘었으며, 따라서 그 현황을 알아보는 것이 필요하다.
- 이 단락에서는 오늘날 Web GIS의 가장 대표적인 응용인 인터넷 지도서비스를 중심으로 하여, 현재 운영되고 있는 Web GIS에 대하여 간략히 알아보고자 한다. 이를 위해서 외국에서 운영되는 지도 서비스 사이트와 우리나라에서의 사이트에 대해 예를 들어 살펴보고 그 구조와 상황, 그리고 제공되는 기능들에 대해서 사용자 관점에서 설명한다.

1. 해외 현황

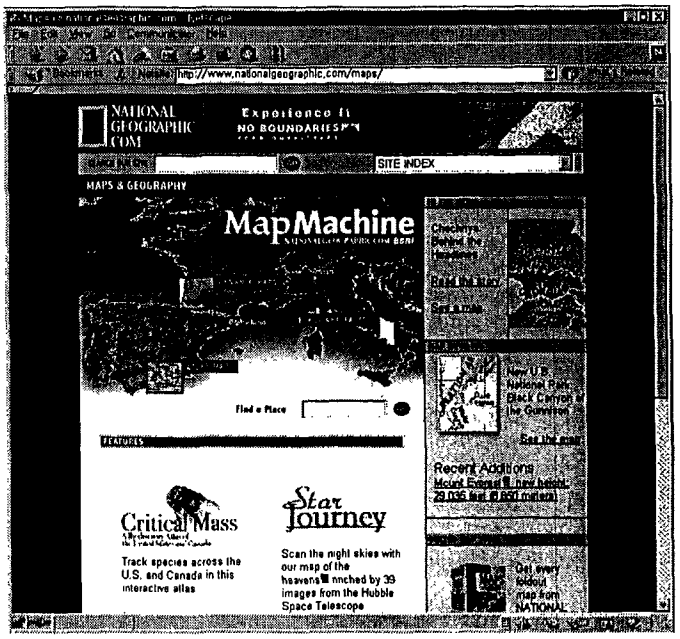
O National Geographic (http://www.nationalgeographic.com/maps/)

◆ 구조 및 상황

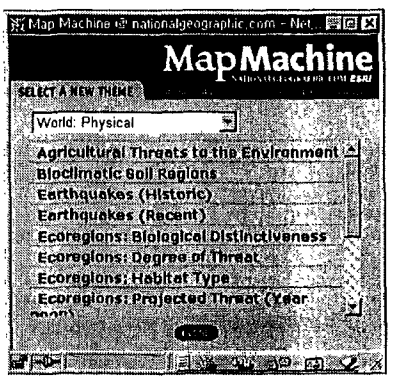
- [그림 1]은 현재 National Geographic 에서 서비스하는 지도 화면이다. 그림에서 알 수 있듯 이 서비스는 ESRI사의 기술로 구현되어 있으며, 동적 지도 서비스(Dynamic Maps), 단순 지도 서비스(Atlas Maps), 대륙별, 국가별 소개(Flags and Facts), GIS 데이터와 트레이닝 등이 서비스되고 있다.
- ESRI 제품군의 특징인 Theme에 따른 레이어 관리 대화 상자를 볼 수 있으며, 이러한 Theme은 이미 설정되어 있는 여러 종류의 것들을 [그림 2]처럼 그대로

[7] http://openmap.bbn.com

읽어 들이는 방식을 취하고 있다.



[그림 1] National Geographic 사의 지도 제공 서비스



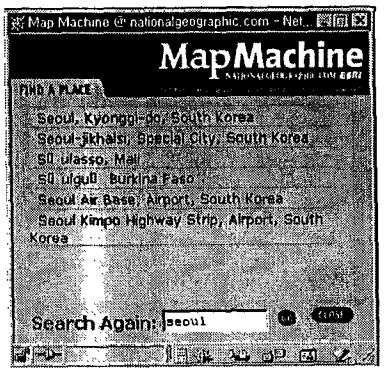
[그림 2] 레이어 관리

◆ 기능 설명

- 앞서 이야기한 것처럼 동적 지도 서비스, 단순 지도 서비스, 대륙별, 국가별 소개, GIS 데이터와 트레이닝 등의 서비스에서 특히 동적 지도 서비스를 중심으로 구현되는 기능을 설명하면 다음과 같다.

<주제어를 통한 지도 검색(Search) 기능>

- 이 기능은 서비스 도중 "Find a Place" 난에 원하는 지역의 이름을 넣으면 구현되는 기능이다. [그림 3]은 Seoul을 예로 든 것인데, 해당하는 이름과 같은 여러 대안들을 찾아내고 그것들 중에서 사용자가 원하는 것을 고를 수 있도록 되어 있다.



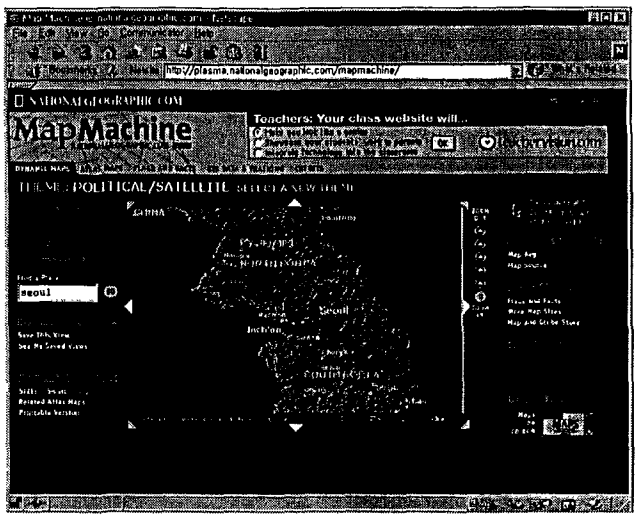
[그림 3] 주제어를 통한 검색

<지도의 확대, 축소(Zoom In, Zoom Out) 기능>

- 기본적인 뷰 제어 기능이다.

<지도 이동(Panning) 기능>

- [그림 4]는 실제로 Seoul을 검색어로 하여 찾아낸 화면이다. 기본적인 화면의 오른쪽에 확대, 축소 기능을 지원하는 툴바가 있으며, 화면상에서 마우스를 움직여서 지도 이동을 할 수 있게 되어 있다.



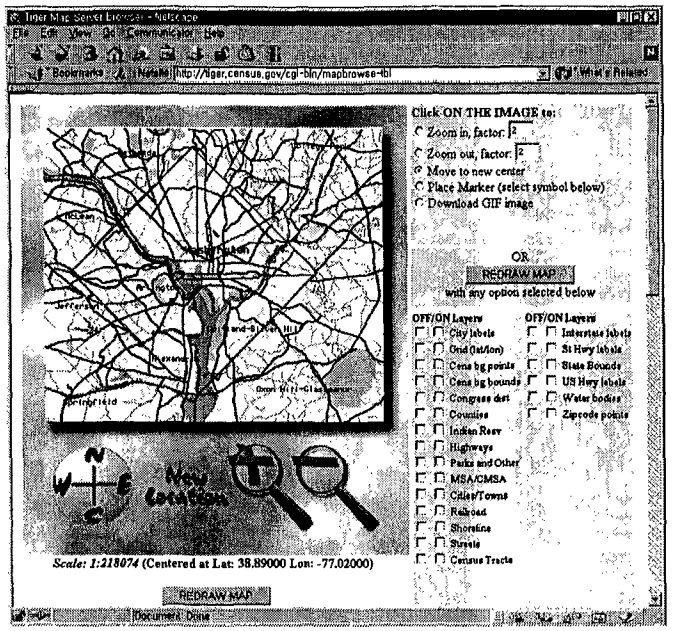
[그림 4] 검색된 서울 지역

- 전체적으로 National Geographic에서 제공하는 Web GIS 서비스는 학술적인 목적이나 데이터를 얻기 위한 목적에서 사용하는 것보다는 세계의 국가들과 대륙들이 어떻게 생겼고 전반적으로 어떤 특징을 가지는지를 알아볼 때 유용한 서비스라고 할 수 있다.

O U.S Bureau of Census (**http://tiger.census.gov/cgi-bin/mapbrowse-tbl**)

◆ 구조 및 상황

- 현재까지 가장 널리 알려진 Web GIS 사이트인 Tiger Map Server는 미국의 센서스기반 지도를 제공해 주는 곳이다. 미국 내의 지도만을 사용할 수 있기는 하지만, 제공하는 지도에 사용자가 원하는 마크를 표기할 수 있고, 도시 이름을 이용한 검색이나 우편번호를 이용한 검색을 지원하며, 각종 센서스 통계를 이용한 지도를 작성할 수 있고, 원하지 않는 도로나 그 밖의 것들을 각각 따로따로 켜고 꺼서, 사용자가 원하는 지도를 만들 수 있는 서비스를 제공한다.
- 또한 전체를 이미지 파일로 저장하여 사용자의 컴퓨터에 불러들이고 인쇄할 수 있도록 하는 서비스를 제공하고 있다. [그림 5]는 이 서비스의 초기화면이다. 첫 화면에서는 워싱턴을 중심으로 하는 지도를 보여주게 되며, 지도의 축척이 지도 하단에 표시되고 지도에 표시될 수 있는 범례들이 오른쪽에 체크박스 형태로 나타난다. 그리고 지도 자체의 범례는 웹페이지의 하단에 나타난다.



[그림 5] 센서스 지도 제공 서비스

◆ 기능 설명

- Tiger Web Server는 동적 지도 서비스를 제공하고 있으며, 다음과 같은 기능을 제공한다.

<지도의 확대, 축소(Zoom In, Zoom Out)>

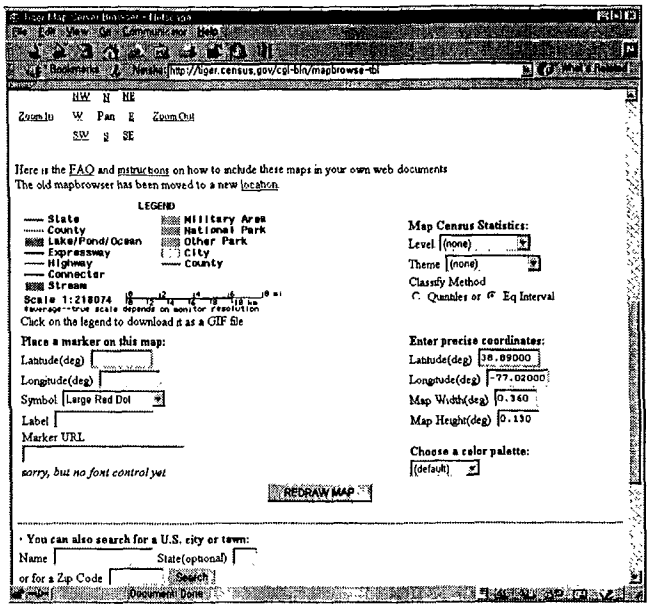
- 그림에서의 + 확대경과 - 확대경이 이 기능을 담당하며, 변화된 크기는 지도 아래의 Scale에 자동 갱신되며, 사용자가 원할 경우 더 정확한 축척을 제공한다.

<지도 이동(Panning) 기능>

- 방위표를 겸한 왼쪽 아래의 그림으로도 이동할 수 있고, 단순히 지도 위에 마우스를 클릭하면 클릭한 곳이 가운데로 오도록 지도가 이동한다. 사용자가 원할 경우는 정확한 위도와 경도를 입력하여 그 지점의 지도를 찾을 수도 있다.

<지도 다운로드(Download) 기능>

- 사용자가 원하는 지도를 gif 포맷의 이미지 파일로 다운로드 받을 수 있다.



[그림 6] 제공 기능

<지도 표시(Marking) 기능>

- [그림 6]에서 "Place a market on this map" 의 여러 방법들을 이용하여 사용자가 원하는 색상의 그래픽을 이용하여 지도상에 자신의 표시를 할 수 있다.

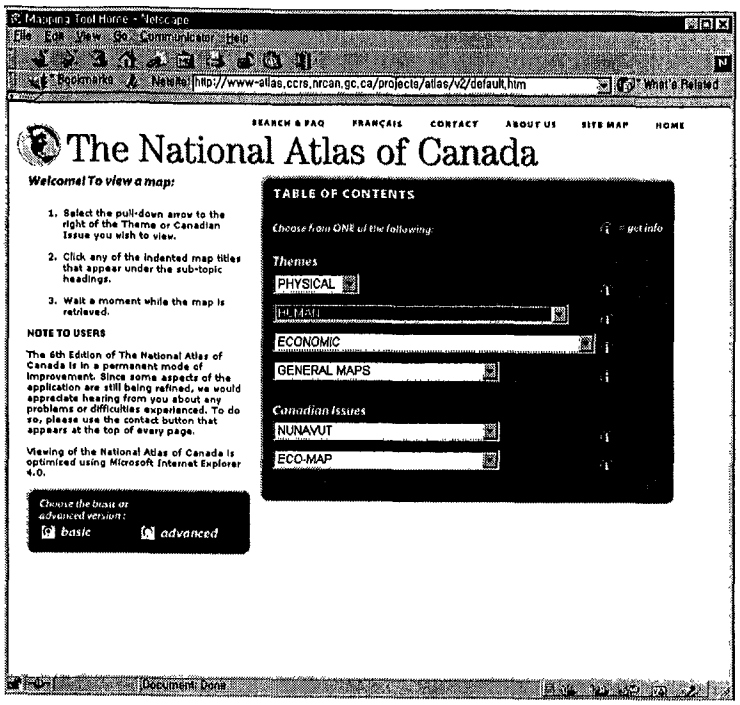
<센서스 통계 표시(Map Census Statistics) 기능>

- 센서스 통계가 표시되는 단위(Level)와 해당하는 주제(Theme)를 선택하고 분류법(Classify Method)을 선택하면 사용자가 원하는 결과물을 얻을 수 있다.

<주소와 우편번호(ZIP code)를 이용한 지도 검색(Search) 기능>

- 정확한 위도와 경도를 이용한 검색이 어려울 경우 도시의 이름이나 우편번호를 이용한 검색 역시 가능하다.

O Natural Resources Canada(http://atlas.gc.ca/)



[그림 7] 캐나다 지도 제공 서비스

◆ 구조 및 상황

- [그림 7]은 캐나다의 Web GIS 서비스의 메인 화면이다. 이 곳은 캐나다의 자료를 중심으로 인터넷 상에서 캐나다 지도를 보여주는 서비스를 제공하는 곳으로 현재 제공되는 것은 위에서 보는 대로 인문적인 것, 경제적인 것, 그리고 일반도와 주제도, 그 외의 부가적인 이슈들을 이용하여 지도를 생성할 수 있게 되어 있다. 아직까지는 기본적인 형태(basic과 advanced 중에서 advanced는 아직 지원되지 않음)만 지원된다. 원하는 하부주제를 리스트 상자에서 선택하면 그것에 맞는 지도가 자동으로 생성된다. [그림 8]은 그 예이다.

◆ 기능 설명

<지도 확대, 축소(Zoom In/ Out) 기능>

- 화면의 (+) 확대 버튼과 (-) 축소 버튼을 사용하여 이 기능이 지원되며, 아래의 축척이 자동으로 갱신된다.

<지도 이동(Panning) 기능>

- 화면의 손바닥 모양의 버튼을 사용하여 이 기능이 지원된다.

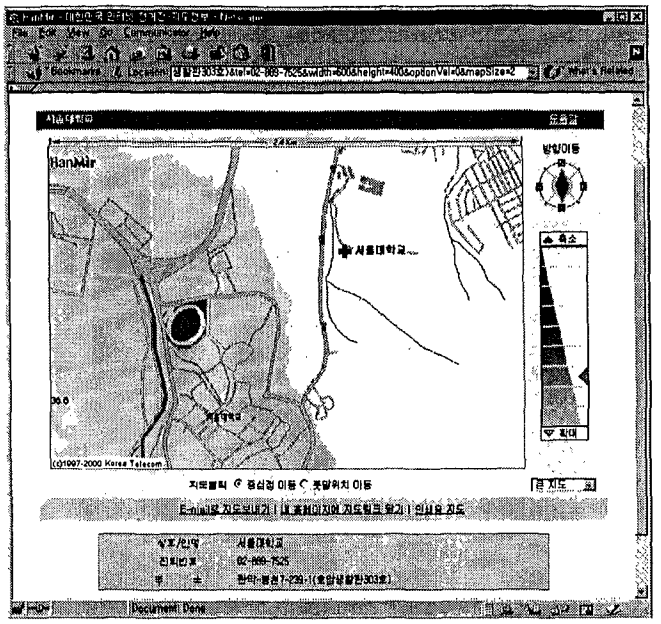
[그림 8] 지도 생성

<지도 설명>

- 기본적인 설명은 이텔릭체 I 버튼을 사용하여 지도에 클릭하면, 클릭된 부분의 간단한 속성이 나타난다.
- 이외에 지도 출력 기능과 Tiger Map Server에서처럼 필요한 지리사상들을 표현하고 감추는 것들이 체크상자 형태로 왼쪽에 구현되어 있다.

2. 국내 현황

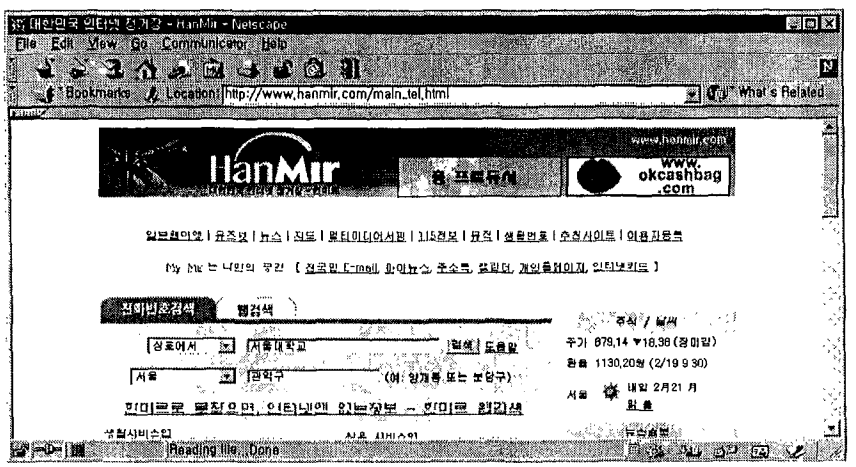
O Hanmir.com의 HanMir 지도정보



[그림 9] 지도 정보 제공

◆ 구조 및 상황

- 이전의 전화번호 검색 사이트에서(http://www.kt114.co.kr) 지도를 지원하는 형식으로 시작된 사이트로 현재는 포탈 사이트와 병합되었다. 한국의 대표적인 Web GIS 사이트이며 현재 전국의 데이터를 다루고 있다. 그러나 센서스 데이터 등의 학술적인 데이터는 사용되고 있지 못하고 정확한 축척 표시도 찾아보기 힘들다. 단순히 지도를 검색하고자 하는 사용자를 대상으로 한 서비스이다.



[그림 10] 검색 방법

◆ 기능 설명

<지도 검색(Search) 기능>

- 상호, 인명, 업종, 전화번호를 이용하여 지도를 검색할 수 있다. 원래 지도 검색을 위한 기능이 아니라 전화번호나 상호를 이용하여 그 해당지역이 어디인지를 파악하기 위한 것이 주 목적이었기 때문에 이러한 인덱스를 이용한 검색 기능이 구현되고 있다.

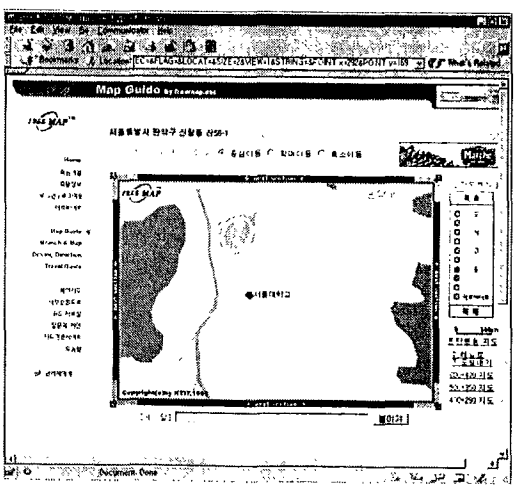
<지도 확대, 축소(Zoom In/ Out) 기능>

- 지도의 오른쪽에 지도를 확대 축소할 수 있는 바를 이용하여 지원된다.

<지도 이동(Panning) 기능>

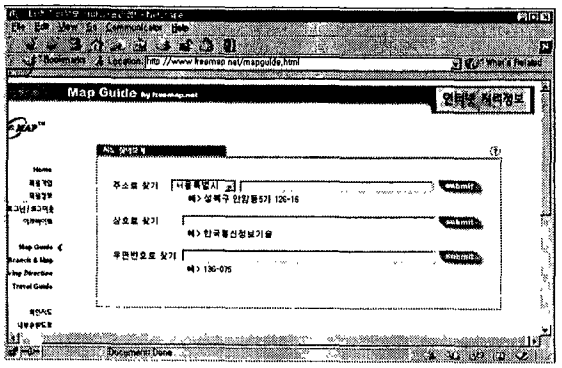
- 화면에 마우스를 클릭하면 이동 기능이 구현되는데, 중심점 이동, 풋말위치 이동 등의 선택사항이 있다.
- 이외에 E-mail로 지도보내기, 내 홈페이지에 지도 달기, 인쇄물 지도 등의 서비스를 지원한다. 전반적으로 학술적인 목적이나 데이터의 공유 등의 목적이 아닌 단순 검색을 위해서만 사용되고 있다.

○ 한국통신정보기술(주)의 Free Map(http://www.freemap.net/)



[그림 11] 지도 정보 제공

◆ 구조 및 상황



[그림 12] 검색 방식

- 대표적인 포탈 사이트인 Yahoo!와 연결되어 사용되는 Web GIS 사이트이며 현재 광역시까지를 다루고 있다. 표지말 등의 기능을 이용하여 사용자가 원하는 지도를 만들 수 있게 되어 있기는 하지만 축척별 지도 검색이 아닌 화면 크기에 맞춘 해상도별 지도를 볼 수 있게 되어있다는 점에서 학술적인 목적보다는 단순 지도 검색기능과 출력기능에 치우치고 있다. 그러나 축척 자체는 화면의 확대 축소에 따라 자동으로 갱신되고 있다.

◆ 기능 설명

<지도 검색(Search) 기능>

- 주소를 이용한 검색, 상호를 이용한 검색, 그리고 우편번호를 이용한 검색을 지원한다. 정확한 위도와 경도를 이용한 검색 등은 지원하지 않는다.

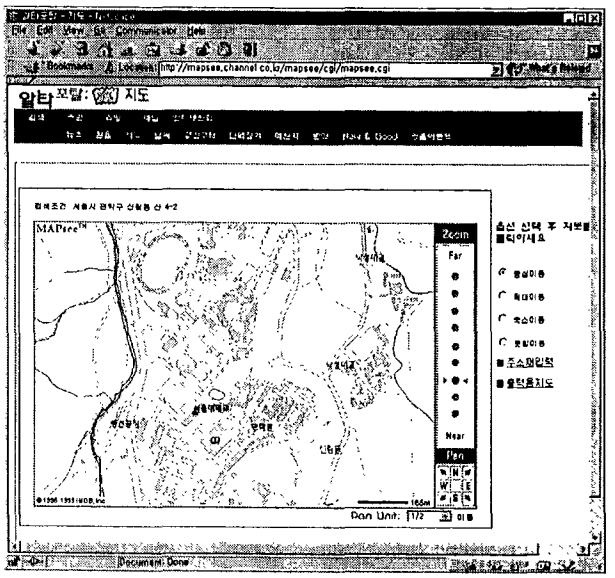
<지도 확대, 축소(Zoom In/ Out) 기능>

- 지도의 오른쪽에 지도를 확대 축소할 수 있는 바를 이용하여 지원된다.

<지도 이동(Panning) 기능>

- 화면에 마우스를 클릭하면 중심이 이동한다. 이외에 E-mail로 지도보내기, 인쇄물 지도 등의 서비스를 지원한다.

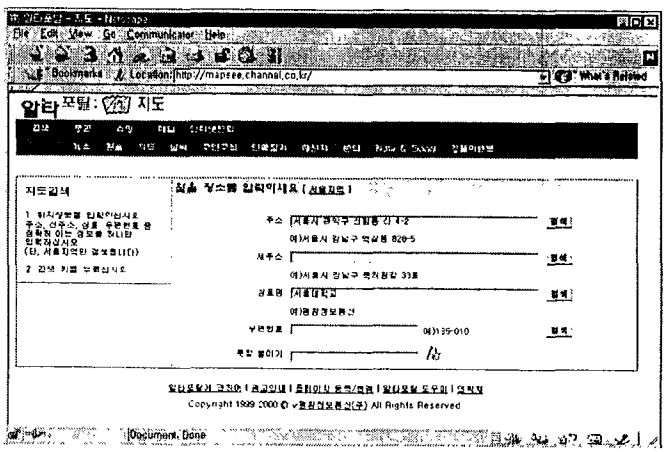
O Altavista와 평창정보통신㈜의 알타포탈 지도(http://mapsee.channel.co.kr)



[그림 13] 지도 제공 서비스

◆ 구조 및 상황

- 역시 대표적인 포탈사이트인 알타비스타에서 제공되는 Web GIS 사이트이며 현재는 서울특별시의 데이터만을 다루고 있다.



[그림 14] 알타포탈의 검색 방식

◆ 기능 설명

<지도 검색(Search) 기능>

- 주소를 이용한 검색과 상호명을 이용한 검색, 그리고 우편번호를 이용한 검색을 지원한다. 지원하는 데이터가 서울특별시에만 한정되므로, 서울에서 적용되고 있는 새주소로도 검색이 가능하다. 이외에 사용자가 마커를 붙일 수 있는 기능을 지원하고 있다.

<지도 확대, 축소(Zoom In/ Out) 기능>

- 지도의 오른쪽에 지도를 확대/축소할 수 있는 바를 이용하여 지원된다.

<지도 이동(Panning) 기능>

- 화면에 마우스를 클릭하면 이동하며, 중심이 이동하면서 축소, 확대 등의 기능도 지원하고 있다. 그리고 확대, 축소 바 아래에 방향을 나타내는 부분을 클릭해도 지도 이동 기능이 구현된다. 이외에 출력물 지도 서비스 기능을 지원한다.

제 3 절 애플리케이션 서버

- 효율적인 대용량 지리정보 서비스를 위해서는 애플리케이션 서버를 개발하는 방식이 많이 이용된다. 이번 단락에서는 애플리케이션 서버의 개발 현황을 파악하기 위해 그 유형과 현재 서비스되는 방식에 대해 간략히 기술한다.
- 2-티어 체계를 근간으로 하는 기존의 클라이언트/서버 체계는 분산 환경에서 요구되는 다양한 정보 처리 로직을 모두 서버 오브젝트가 담당해야 한다는 한계점을 지니고 있다. 따라서, 기존의 시스템은 구현 자체는 그리 어렵지 않지만 비즈니스가 발전함에 따라 변화에 대응에 적응할 수 있는 어플리케이션 개발에는 한계로 작용한다.
- 단일 어플리케이션 내부에서 MVC(Model, View, Controller) 모델을 이용해 3Tier 구조를 가진 어플리케이션을 개발하면, 변화에 대한 적응성을 높일 수 있지만 여전히 분산 시스템 구조에서 기인하는 고 비용을 유발하는 무거운(fat) 클라이언트가 문제가 된다. 이러한 기업 컴퓨팅 환경의 제약을 넘어서고자 하는 시도는 클라이언트/서버 환경을 2계층에서 3계층으로 변화시키고자 하는 시도를 낳

았다. 특히 인터넷, 인트라넷, 분산 객체들의 등장은 이러한 변화의 흐름을 가속화 시키고 있으며, 엔터프라이즈 시장 진입을 노리는 마이크로스프트사에서 윈도우 2000을 3계층으로 개발하면서 이에 대한 관심과 열기가 높아지고 있는 추세다.

- 현재의 엔터프라이즈 환경은 전지구적으로 산재해 있는 각각의 사용자 요구에 대한 해결을 제시해야 한다는 문제에 직면해 있다. 정보시스템 개발에 있어 클라이언트 시스템의 배포 문제가 심각하게 부상하고 있다. 빠른 변화에 적응, 대응할 수 있는 클라이언트 소프트웨어의 개발과 이를 다시 국지적으로 분배, 설치하는 것은 엄청난 비용을 필요로 한다. 다시 말해 이러한 기존의 클라이언트/서버 구조로는 새로운 변화에 대한 적응에 무리가 있었기에 새로운 3Tier 아키텍처가 등장하게 되었다.
- 현재의 엔터프라이즈 환경에서 요구하는 해결책을 제시한 것이 3Tier 아키텍처이다. 3tier 아키텍처는 변화하는 비즈니스 요구에 대응하여 서비스나 컴포넌트들을 적절히 추가하거나 변화시킬 수 있는 유연성을 가지고 있다.
- 3tier 아키텍처는 다음과 같은 세 부분으로 구성된다.

  O User Interface

  클라이언트 어플리케이션으로 구현되며 여기에 비즈니스적인 요소는 없다.

  O Business Logic

  이 부분은 상호 협동하는 객체들로 구성되어 있다. (ex. Network Component) 대개의 경우 비즈니스 로직은 다수의 레벨을 이루고 있기에 N-Tier 또는 multi-tier 시스템 아키텍처라 불리기도 한다.

  O Data Access

  데이터 서버를 통해 저장된 정보에 대한 접근을 가능케 한다.

- 어플리케이션 내부의 특정 비즈니스를 위한 어플리케이션 로직은 서버로 하여금 다양한 종류의 클라이언트에 대해 서비스를 제공할 수 있게 해 준다. 새로운 클라이언트 프로그램은 기존의 어플리케이션 서버를 이용할 수도 있다. 또한 어플리케이션 서버 자체도 하나의 분산 객체로 행동한다고 볼 수 있다.
- 기존의 어플리케이션 서버도 래핑(wrapping)하여 또다른 어플리케이션 서버로 보이게 한다든가 시스템 내부에서 하나의 컴포넌트로 구성 가능하다. 이러한 기술을 이용하면 현재 진행되고 있는 개발에 소요되는 비용 절감도 가능하다.
- 3Tier 아키텍처는 기존 시스템에 대해 확장과 적응을 가능케 함으로써 유연성을 제공한다. 객체 포장기술(object wrappers)을 이용함으로써 기존의 시스템을 분산 객체 환경에 적합하게끔 변화시킬 수 있다. OMG에서 제안한 IDL등은 3Tier 로 구성된 분산 환경을 개발, 디자인, 배포하는 문제에 대한 기반을 제공한다.

- 80년도 초반에 등장한 tier개념은 초기에는 터미널(1계층), 미니컴퓨터(2계층), 메인프레임(3계층)을 이르는 어플리케이션의 물리적 분할을 위해 사용되었다. 하지만 현재에는 '계층'이라는 용어는 클라이언트와 서버에 걸친 어플리케이션의 논리적 분할을 의미한다. 프로세싱 로드 분할은 클라이언트/서버에서 중심 되는 개념으로 작업부하를 어디에 둘 것인가 라는 설계 문제에 대한 해답을 제시한다. 또한 클라이언트와 서버는 적어도 사용자 인터페이스계층과 공유 데이터 계층이라는 2개 이상을 계층 구조를 가진다.
- 2계층은 프로세싱 로드를 두 개로 분할한다. 어플리케이션의 많은 부분이 클라이언트 측에서 실행되는 fat 클라이언트에서 이러한 구조를 볼 수 있다. 3계층은 프로세싱 로드를 인터페이스 로직을 실행하는 클라이언트, 비즈니스 로직을 실행하는 어플리케이션 서버, 데이터베이스/기존 어플리케이션으로 구분한다. 3계층에서는 어플리케이션 로직을 서버로 이동시키기 때문에 fat 클라이언트에 대한 상대적 개념으로 thin 클라이언트라 부른다.
- 2계층에서는 어플리케이션 로직이 클라이언트와 서버 양쪽에 존재한다. 이러한 구조는 개발이 용이하고 구조상의 단순 명료함을 제공하여 기존의 비주얼 개발툴을 사용해 2계층 구조의 어플리케이션 개발에 뛰어난 성능을 보였다. 하지만 기존 시스템의 확장이라는 문제에 부딪혀서는 마땅한 해결책을 제시하지 못했으며 대규모 업무에 있어서도 제대로 동작하지 않았다. 또한 최근에는 기업 전반에 걸쳐 클라이언트/서버 어플리케이션과 인터넷 기반의 전자 상거래가 전개되고 있으며, 많은 어플리케이션이 컴포넌트로 나뉘어지고 다중 프로세스로 분산되는 복잡한 상황이다.

	2 계층	3 계층
시스템 관리	복잡	덜 복잡
데이터	캡슐화 낮음(데이터테이블 노출)	캡슐화 높음(클라이언트는 서비스 또는 메소드 호출)
성능	나쁨(많은 SQL문들은 네트워크로 보내진다. 선택된 데이터는 분석을 위해 클라이언트로 다운되어야만 한다)	우수(서비스 요청들과 응답만이 클라이언트와 서버간에 보내어진다.)
어플리케이션 재사용	나쁨(클라이언트상의 monolithic application)	우수(서비스와 객체들의 재사용)
인터넷 지원	나쁨	우수(thin client들은 애플릿이나 빈처럼 다운로드가 쉽다.
하드웨어 아키텍처 유연성	제한적(하나의 클라이언트와 서비 구조)	우수(모든 3계층은 다른 컴퓨터들 상에 상주하고 2.3계층은 동일 컴퓨터상

		에 상주할 수 있다.)

[표 1] 2계층과 3계층 클라이언트/서버 비교

- multi-tier 개념이 등장하는 배경에는 인터넷이 있다. 과거의 소기업의 부서 수준을 넘어서는 전사적 내지는 전 지구적인 서비스의 요구에 응답하기 위해서는 기존의 2계층 클라이언트/서버 구조로는 한계가 있었다. 현재의 엔터프라이즈 환경에서는 수천개의 클라이언트들이 서비스하는 클라이언트/서버 어플리케이션을 업무에 적용하고 있다. 이같은 어플리케이션은 많은 서버상에서 수행되고 그만큼 많은 소프트웨어 컴포넌트로 구성된다. 엔터프라이즈 환경 내의 핵심 기능을 수행하고 있는 이같은 클라이언트/서버 어플리케이션은 전사적으로 확대되고 있으며 이에 따라 클라이언트와 서버가 더 이상 동일 지역에 위치할 필요가 사라지게 되었다.
- 다시 말해 컴퓨팅 환경이 변화하고 있어 어플리케이션들은 소규모의 지역적 단위를 넘어서서 전 지구적으로 상호 연결된 세계로 확장되고 있다. 3계층은 바로 이러한 상황 변화에 대처할 수 있게끔 설계된 아키텍처다. 3계층은 GUI를 제공하는 클라이언트에서 서비스나 메소드 호출을 통해 서버와 상호작용하며 어플리케이션 계층은 그 가운데에 위치한다.
- 3계층은 대규모 인터넷과 인트라넷 클라이언트/서버 어플리케이션의 요구에 부응하는 새로운 기술로서 대부분의 코드가 서버상에서 실행되며 특히 자바 애플릿과 같은 기술을 이용하면 네트웍상에서 관리하고 배포하기가 용이하다. 클라이언트는 2계층처럼 데이터베이스와 직접 상호작용하는 대신 서버상의 비즈니스 로직을 호출한다. 비즈니스 로직은 서버상의 데이터베이스에 접근하여 데이터베이스의 스키마를 클라이언트에게 노출하지 않아도 되게 함으로써 더욱 우수한 보안을 제공하기도 한다. 많은 서버들의 SQL 조회와 갱신에 대한 요구 대체는 성능면에서도 3계층이 2계층보다 월등한 결과를 제공한다.
- 2계층이 3계층에 비해 규모도 작으며 개발하기도 훨씬 용이하다. 하지만 그 범위와 규모가 확대됨에 따라 2계층 어플리케이션 개발은 시간과 비용 측면에서 이득을 가져다 주지 못한다. 3계층은 확장이 용이하므로 처음에는 범위와 기능면에서 작게 시작하고 나중에 전사적 혹은 인터넷을 배경으로 한 어플리케이션의 개발 방향이 근간의 추세이다.

파 오 손 면

때문에 보낸 메시지와 받은 메시지를 보관할 장소가 필요한데 이를 큐(Queue)라고 부른다. 이러한 특성은 즉각적인 응답을 기대하는 처리에는 부적합하지만 여러 가지 일들을 종합 처리한 후에 결과가 나오는 업무 보고서를 위한 통계자료 작성 등에는 상당한 장점을 가진다.

- ORB 기반 미들웨어는 90년대 들어 새로운 패러다임으로 등장한 '객체지향' 기술을 미들웨어 분야에 접목시킨 기술이다. 객체지향 미들웨어 분야에는 현재 큰 두가지 흐름이 있다. 마이크로소프트에서 추진하고 있는 DCOM(Distributed Component Object Model)과 OMG(Object Management Group)에서 추진하고 있는 CORBA(Common Object Request Broker Architecture)가 있다. 객체지향 미들웨어는 개발자들에게는 보다 쉽고 빠른 개발을 사용자들에게는 쉬운 사용을 관리자에게는 보다 용이한 관리 기법을 제공할 차세대 미들웨어로 주목을 받고 있으며 현재 타 분야에서의 객체 기술과 결합하여 많은 개발이 이루어지고 있다. 이렇듯 객체지향 미들웨어의 전망은 매우 희망적이다. 하지만 현재 판매되고 있는 객체 지향 미들웨어의 성능은 즉각적인 반응을 원하는 대량의 데이터 처리를 수용하기에는 아직 부족하다. 또한 양대 진영의 대립으로 아직 표준이 확실하지 않다는 점도 앞날의 변수로 작용한다.

제 3 장 연구개발 수행 내용 및 결과

제 1 절 개방형 GIS 시스템

1. 통합 지리정보 서비스

- 공간 문제는 본질적으로 복잡한 특성을 가지고 있으며, 공간 문제와 관련된 개인이나 단체들 역시 다양하고 복잡한 이해 관계를 가지고 있다. 따라서, 공간 데이터의 획득에서부터 분석과 공간 의사결정에 이르기까지 다학문적인 접근과 함께 다양한 개인 혹은 집단들 간의 상호 협력을 통한 문제 해결 방식이 요구된다 [1]. 예를 들어, 도시 정보 시스템의 운용을 위해서는 도시 설비, 건축 설계, 도시 계획, 재정, 법률, 환경, 사회, 경제와 관련된 다양한 분야의 전문가와 함께 이를 지원할 수 있는 공간 데이터 지원 정보 시스템을 구축하는 것이 필요하다.
- 최근에 폭발적으로 수요가 증가하고 있는 인터넷/웹은 이러한 문제에 있어 새로운 시스템의 구축과 함께 기존 시스템들과의 연계가 그 해결 방안임을 암시하고 있다. 실제로, 우리나라의 경우 각 지방자치단체별로 도시기반 정보, 지적 정보, 건축 설계 정보 등을 구축 또는 구축하려는 계획을 가지고 있으므로 막대한 예산이 소요되는 작업을 기관별로 다시 진행한다는 것은 국가적으로 큰 낭비가 아닐 수 없다. 따라서, 분산되어 있는 데이터베이스 간의 연계를 기초로 한 시스템 구축을 통해 통합 지리정보를 서비스하는 기능이 필요하다.
- 네트워크를 통해 공간적으로 분산되어 있는 GIS 사용자들 간의 협력이 가능하도록 하기 위해서는 기존의 시스템들을 연계(link)할 수 있는 방안이 필요하며, RPC(Remote Procedure Call), DCE, CORBA, DCOM 등의 기술들이 지금까지 제시되어 왔다. 이러한 기술들은 네트워크 상에 분산되어 있는 서로 다른 이질적(heterogeneous) 시스템들간의 접근 및 제어를 가능하게 한다.
- 현 단계에서 이질적인 시스템들을 통합하기 위해서는 기존의 시스템(legacy system)에 접근할 수 있는 통로를 열어주거나, 아키텍처 자체에 접근할 수 있는 개방형 시스템(open system)을 구축하는 것이 바람직하다. 개방형 시스템은 기존의 여러 업체들이 만들어 놓은 제품뿐만 아니라 새로이 개발되는 차세대 시스템들간의 상호운용성과 이식성(portability)을 보장할 수 있으므로 앞으로의 시스템 구축에 있어 새로운 방향을 제시하고 있다고 할 수 있다.
- 공간 정보를 다루는 GIS 시스템 간의 상호운용성을 확보하기 위해서는 공간 데이터 모델, 공간 데이터 포맷의 차이로 인해 발생하는 기술적인 문제, 기관들간에 사용하는 용어(ontology)의 불일치로 발생하는 의미무결성(semantics)의 문제들, 공간 데이터를 보유하고 있는 기관들 간의 보안이나 제도적인 문제 등을 해결해

야 한다.

- 개방형 시스템의 구현을 위해서는 표준(standards), 참조 기술(reference technology), 공통 인터페이스(common interface) 등을 정의, 적용해야 한다[2].

  ○ 컨소시엄 표준

  적용가능한 IT 기술 표준은 두 가지 형태로 구분될 수 있다. 공식 표준(formal standard)은 국제 표준화 기구(International Standard Organization)와 같은 공식적인 기관에서 정한 표준이며, 컨소시엄 표준(public consortium standard)은 유사한 기술을 제공하는 공급자들과 기술의 수요자들로 구성된 컨소시엄에서 합의를 통해 만들어진 표준이다.
  현재 분산 기술과 관련된 컨소시엄 표준으로는 OMG(Object Management Group)의 CORBA(Common Object Request Broker Architecture), OpenGIS 컨소시엄의 OGIS(Open Geodata Interoperability Specification) 등이 있다. 이 두 표준은 분산 기술의 표준과 분산 기술 중에서도 GIS 시스템 구축에 사용될 수 있는 대표적 표준이라는 점에서 의미가 크다.

  ○ 참조 기술

  참조 기술은 어떤 개발 주체가 자신이 개발한 새로운 기술을 널리 사용할 수 있도록 관련 소스코드나 개발 도구를 만들어 배포하는 기술을 뜻한다. 대표적인 예로 선 마이크로 시스템즈의 자바 개발 도구(Java Development Kit)를 들 수 있다. 참조 기술은 하나의 공통 기술이 주도적으로 사용될 경우, 개발자들이 그 기술에 대해 일관적인 시각을 가지고 개발할 수 있다는 점에서 중요하다. 즉, 일관된 실질 표준(de facto standard)은 상호운용성을 확보하는 데 있어서 큰 도움이 된다.

  ○ 공통 인터페이스

  공통 인터페이스는 소프트웨어 모듈들이 외부 시스템과 상호 작용하는 방법을 정의한 것이다. 개발자들은 이러한 공통 인터페이스를 구현함으로써 상호운용성과 확장성을 확보할 수 있다. OMG나 OpenGIS 컨소시엄의 여러 사양들은 이 공통 인터페이스를 정의하고 있다.

- 통합 지리정보 서비스를 염두에 둔 시스템은 현재의 분산 기술과 공간 데이터 표준을 준수함으로써 상호운용성을 확보해야 한다. 이러한 상호운용성의 확보를 위한 대안은 여러가지가 있을 수 있으나, **컨소시엄 표준이 갖는 중요성과 실질적인 상호운용성의 확보** 등을 감안하면 CORBA와 같은 표준 인터페이스와 함께 이에 기반한 OpenGIS 명세를 구현하는 것이 바람직하다. 특히, OpenGIS for

CORBA 기반의 OpenGIS 명세를 구현하고 그에 기반한 개방형 시스템을 설계하는 경우 수요가 증가하고 있는 인터넷/웹 상에서의 지리정보 서비스에 응용 가능하다.

2. 분산 데이터베이스

가. 분산 시스템의 정의

- 분산 시스템은 기본적으로 분산 처리기, 분산 데이터베이스, 통신 네트워크 등을 기본 요소로 가진다.
- 분산 처리기(distributed processor)는 지리적으로 분산되어 있는 컴퓨터 시스템을 말한다. 이것은 한 머신이 데이터를 처리할 수 있는 자체 능력을 가지고 있다. 분산 데이터베이스(distributed database)는 지리적으로 분산되어 있는 데이터베이스로 분산 처리기와 함께 한 머신에서의 정보 처리를 지원한다. 통신 네트워크는 분산되어 있는 시스템들을 서로 연결시켜 자원을 공유하게 함으로써 논리적으로 하나의 시스템과 같은 기능을 할 수 있도록 하는 망(network)을 의미한다.

나. 분산 데이터베이스 시스템의 특징

- 분산형 데이터베이스 시스템의 구축 목표 중 하나는 위치 투명성(location transparency)을 확보하는 것이다. 위치 투명성을 확보한다는 것은 사용자 또는 응용 프로그램이 접근하는 데이터가 어디에 있는지를 신경쓰지 않고도 획득이 가능하다는 것을 의미한다. 데이터의 위치 정보는 시스템이 제공하는 데이터 카탈로그(catalog)에서 관리할 수 있다. 위치 투명성을 보장하는 데이터베이스 시스템은 응용프로그램의 로직을 간단하게 할 뿐만 아니라 하나의 데이터베이스 시스템에서 다른 데이터베이스 시스템으로 데이터 저장소를 변화시키더라도 응용프로그램에 별다른 영향을 끼치지 않는다.
- 여러 개의 데이터베이스에 데이터를 나누어 저장하는 경우 데이터를 중복하여 저장하거나 원할 경우 중복하지 않게 저장할 수 있는 기능은 복사 투명성(replication transparency)으로 얘기될 수 있다. 데이터의 중복이란 하나의 논리적 데이터가 상이한 노드에 저장되는 것을 의미한다. 중복을 지원하는 시스템에서는 데이터에 대한 접근이 아주 복잡해질 수 있다. 예를 들어, 일반적으로 데이터에 대한 검색을 위해서는 가장 가까운 데이터 저장소에 접근하면 되지만 데이터 업데이트를 위해서는 중복된 데이터가 있는 모든 노드에 접근해야 한다. 복사 투

명성을 확보함으로써 중복된 데이터의 위치 식별이나 관리를 시스템에서 담당하도록 할 수 있다. 이것은 사용자나 응용프로그램이 데이터의 중복에 주의를 기울이지 않아도 된다는 것을 의미한다.

- 위치 투명성과 복사 투명성을 제공함으로써 사용자가 분산되어 있는 데이터베이스 시스템을 중앙집중식 데이터베이스 시스템과 별다른 차이없이 사용할 수 있도록 할 수 있다. 이것은 분산되어 있는 공간 데이터베이스 시스템에도 동일하게 적용된다.
- 분산 환경에서 구축되는 개방형 GIS 시스템은 공간 데이터베이스의 분산을 고려하지 않을 수 없다. 따라서, **공간 데이터베이스의 위치 투명성과 복사 투명성**의 확보가 가능한 방향을 모색하는 것이 중요하다.

3. 애플리케이션 중심의 시스템

- 기 구축된 공간 데이터는 서로 다른 개발 시점 및 개발 환경으로 인해 이질적인 공간 데이터베이스, 이질적인 매핑 라이브러리(mapping library)에 기반하고 있다. 예를 들어, 환경정보시스템을 구성하는 각각의 서브시스템들은 Windows NT, SUN Solaris, HP-UX 등과 같은 다양한 운영체제와 SDE[8], SDO[9] 와 같은 다양한 공간 데이터베이스, 그리고 MapObjects[10]와 같은 다양한 매핑 라이브러리들을 사용한다. 이렇게 다양한 라이브러리와 플랫폼을 사용하는 경우, 각 업무별로 소프트웨어 성능을 최적화할 수 있다는 장점을 가질 수 있으나, 서로 다른 부서에서 구축한 데이터를 다른 곳에서도 공유할 수 있도록 하는 통합 지리정보 서비스의 경우에는 이것이 가장 큰 걸림돌로 작용한다.
- 이러한 문제를 해결하기 위하여 지금까지는 데이터 교환 표준(Data transfer standard)의 제정이나 호환 포맷(common file format)을 통한 데이터 교환 방식을 고안하여 이용하였다. 그러나, 현재의 데이터 교환 방식은 자동화가 어렵고 애플리케이션이 자신의 고유 포맷으로 데이터를 변환하는 데 사용되는 리소스의 낭비, 그리고 실시간 적용이 힘들다는 이유 등으로 인하여 상호운용성과 같은 새로운 기법의 도입이 요구되고 있다.
- 상호운용성은 여러가지 측면에서 논의될 수 있으나, 크게 데이터 호환(Data interoperability)과 서비스 호환(Service Interoperability)으로 나누어 볼 수 있다. 데이터 호환은 데이터 포맷의 변환없이 다양한 시스템이 동일한 정보를 공유할 수 있도록 하는 것을 의미하며, 서비스 호환은 한 시스템의 기능을 다른 시스템이

[8] Spatial Database Engine from ESRI Inc.
[9] Spatial Data Cartridge from Oracle Inc.
[10] ActiveX based mapping library from ESRI Inc.

그대로 사용할 수 있다는 것을 의미한다. 다양한 시스템들을 통합함으로써 정보의 공유를 꾀하는 경우 현재는 주로 데이터의 공유에 관심을 두고 사업을 추진하는 것이 일반적이다.

- 그러나, 실질적인 애플리케이션 중심 시스템(application oriented system)은 다양한 데이터 제공자(data provider)와 필요한 경우, **서비스 제공자(service provider)를 통합**할 수 있어야 한다[3]. 공간 데이터 처리의 관점에서 보았을 때, 이러한 요구사항은 지리공간 데이터와 지리정보 처리에 관한 상당한 수준의 상호운용성을 요구한다. 즉, 단순히 데이터의 교환이나 통합의 수준에 그치는 것이 아니라 시스템 자체의 기능이 투명하게 드러남으로써 공간 분석과 같은 기능의 서비스가 가능하도록 요구한다.
- 현재 OpenGIS 콘소시엄(OGC[4])은 표준화(standardization)가 이러한 요구사항들을 만족시킬 수 있는 가장 좋은 방식 중의 하나라는 믿음아래 상호운용성을 보장할 수 있도록 명세서를 개발하고 있다. 그러나, 이러한 접근 방법은 항상 벤더에 종속적인 시스템과 이로 인한 기술적 진보의 제약을 포함한다.
- 실제로 상호운용성을 보장하는 시스템의 설계와 구현 노력이 오랫동안 이루어졌음에도 불구하고 많은 사람들이 효율적으로 사용할 수 있는 개방된 라이브러리나 시스템이 존재하지 않았다는 사실은 단순히 표준화 추진과 업계 주도의 시스템 개발에만 의존하는 현재의 한계를 보여준다고 할 수 있다.
- 따라서, 애플리케이션 중심의 개방형 시스템을 효율적으로 설계, 구현하기 위해서는 모든 사람들이 편리하게 사용할 수 있는 **필수적 개방형 라이브러리들을 설계, 구현하고 이를 공개하는 것**이 바람직하다고 할 수 있다.

4. 오브젝트 웹 GIS

- 많은 지리공간 정보 전문가들이 GIS의 미래가 상호운용 가능한 것이 될 것이라는 데 동의하는 것으로 보인다. 이러한 관점에서 GIS 시스템을 운용하는 데 필요한 몇 가지 요구사항이 정의될 수 있다[5]. 그 중에서도 핵심적인 세 가지 사항들은 다음과 같다.

  ○ 분산(distributed)

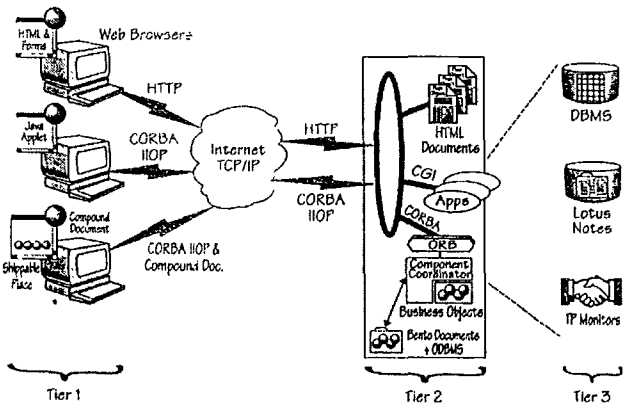
  네트워크 상의 여러 노드에 분포되어 있을 수 있는 데이터 저장소(data storage), 데이터 처리, 사용자 인터액션(user interaction) 등을 포함한다. 예를 들어, 노드 A에 있는 사용자는 노드 B에 있는 데이터를 노드 C로 보내어 처리하고 그 결과를 자신이 있는 노드 A에서 디스플레이 할 수 있다.

○ 분할 또는 분리(disaggregated or decoupled)

특정 벤더에 의해 만들어진 단일 시스템(monolithic system)이 여러 벤더들에 의해 만들어진 플러그-앤-플레이(plug-and-play) 컴포넌트로 대체된다. 이 컴포넌트들은 산업표준에 준하는 것으로 서로 상호운용될 수 있도록 설계된다.

○ 상호운용(interoperable)

위의 두 사항들을 만족시키기 위한 전제조건(pre-condition)이다.



[그림 16] CORBA/Java 오브젝트 웹[6]

- [그림 16]은 코바/자바 기반의 오브젝트 웹을 표현한 것이다. 위의 3-tier 구상은 곧 모든 tier가 분산된 상태로 확장 가능하므로 이상적인 배치에서는 멀티-티어(multi-tier) 에서의 유연성을 보장하는 측면이 강조된다. GIS 매핑의 관점에서 보면, 이 그림을 통해 어떤 tier에서도 운용될 수 있는 **매핑 커널(mapping kernel)**이 필요하다는 것을 알 수 있으며, 또한 코바/자바 기반의 오브젝트가 갖는 장점을 GIS 시스템에서 최대한 활용할 필요가 있다는 것도 확인할 수 있다.
- [표 2]는 분산된 지리정보 처리 환경에서의 매핑에 있어, 위와 같은 구상을 만족시키는 가능한 대안들(alternatives) 몇 가지를 보여주고 있다. 이 표는 빠르게 변화하는 소프트웨어 기술을 포함하고 있다.

특징	가능한 대안	관련 기술
분산 (Distributed)	다중 표현 (Multiple representation) 원격지 매핑 (Mapping at remote site)	멀티 뷰 기술 (Multiple view technology) 미들웨어 기술 (Middleware technology)
분할	매핑 컴포넌트	그래픽 프로그래밍

(Disaggregated)	(Mapping component) 매핑 커널 설계 (Design of Mapping Kernel)	(Graphic Programming) 컴포넌트 아키텍처 (Component Architecture) 객체지향 방법론 (Object-Oriented Programming)
분리 (Decoupled)	매핑 커널의 탐색 (Search for a Mapping Kernel) 매핑 컴포넌트로의 이벤트 전달 (Event dispatching to a Mapping Component)	위치 서비스 (Location Service) 이벤트 서비스 (Event Service)

[표 2] 분산된 지리정보 처리 환경에서의 매핑이 갖는 특징을 위한 가능한 대안들

- 이중에서도 단일한 지리정보 데이터를 가지고 다양하게 볼 수 있는 멀티 뷰 기술은 분산 환경에서 공간 뷰(geographic views)를 효과적으로 관리하기 위해 필수적이다. 예를 들어, 네트워크 관리 시스템(NMS:Network Management System)을 웹으로 서비스하는 회사의 경우, 지도 서비스를 하는 객체가 대용량 지리정보 데이터를 다수의 클라이언트 객체의 요구에 맞게 처리하여 결과를 보내주어야 하므로 데이터의 중복 사용이 전자지도 출판의 걸림돌이 될 수 있다. 이러한 경우에 애플리케이션 서버(Application Server)에서 멀티 뷰를 사용하면, 동일한 데이터를 재사용함으로써 부하를 줄일 수 있다.
- 분산이라는 측면은 기존의 단일 GIS 시스템의 각 기능을 분할함으로써 좀더 견고하고 재사용 가능한 컴포넌트들을 만들도록 요구한다. "매핑 커널" 이라는 용어는 따라서 분산된 지리정보 처리 환경에 있어서 어느 tier, 어느 플랫폼에서도 운용될 수 있는 이동 가능한 매핑 컴포넌트(Portable Mapping Component)를 의미한다고 할 수 있다.
- 오브젝트들을 분산된 노드들에 분리시킨다는 것은 각 오브젝트들을 탐색할 수 있는 방법을 요구한다. 특히, 이벤트 서비스나 메시지 서비스와 같이 분리된 오브젝트들을 하나로 볼 수 있도록 만드는 메커니즘이 필요하다.

5. 협력 지원 GIS 시스템

- 통합 공간 정보 서비스의 가장 간단한 예로 협력지원 시스템을 들 수 있다. 전자지도와 관련된 협력지원 시스템(Computer Supported Cooperative Work : CSCW)에서 가장 중요한 기능 중의 하나는 인터넷 또는 인트라넷 상에 분산되어 있는 여러 사람들이 동시에 공간 객체(spatial objects)들을 보고 조작할 수 있다

는 것이다. 이러한 의사소통 방식의 핵심은 공유 화이트보드(shared whiteboard)의 개념을 GIS에서 확장시킨 월보드(wallboard)이다.

- 월보드는 사용자의 접근 권한에 따라 서로 다른 작업 공간에서 움직일 수 있도록 한다. 그 중에 대표적인 두 가지 작업 공간의 예는 아래와 같다.

  O Within arm's length

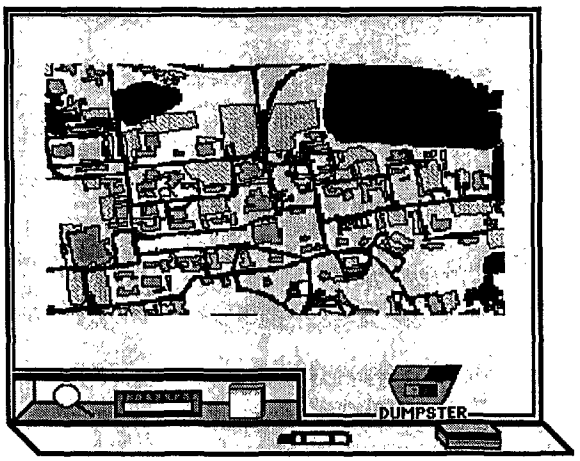
  손이 월보드에 직접 닿을 수 있는 위치에 있다는 의미로, 현재 보고 있는 공간 객체(Spatial Objects)를 직접 조작할 수 있는 권한을 사용자에게 준다 예를 들어, 객체를 선택해서 보고 그 객체를 이동시킬 수도 있으며, 특정 지역을 확대, 축소, 이동 혹은 회전시킬 수도 있다.

  O Within sight

  월보드를 주로 보는 사용자를 위한 작업공간으로, 뷰 기능만을 지원하며 때로는 다른 지원 시스템(예를 들어, 채팅 시스템)을 통한 의사소통이 있을 수 있다.

- 공간계획가를 포함한 의사결정자들은 일반적으로 지도, 보고서, 항공사진, 위성사진, 교통정보나 다른 관련 문건들을 모두 회의에 동원한다. 따라서, 많은 경우에 지도를 펼쳐두고 지도상의 어떤 지점이나 사건들에 관하여 설명하게 된다. 이 과정에서 통합 지리정보 서비스의 필요성이 부각된다.
- GIS개념을 원용하고 있는 월보드는 기존의 컴퓨팅 환경과 동떨어진 것이 아니며 다른 것들을 포괄하는 시스템으로서 네트워크 컴퓨팅 디바이스 등을 사용하여 구현한다.
- 월보드의 기본 구성요소 중에서 가장 중요한 것이 지도 디스플레이를 위한 영역이다. 많은 공간 계획 작업들이 전자 지도에 근거하고 있기 때문에 이러한 전자지도를 이용한 의사소통이 가능해야 한다. [그림 17]은 월보드의 기본 개념을 그림으로 표현한 것이다.
- 칠판 위에 그려져 있는 것이 대상 지역의 기본도이다. 공간 계획가들이 모두 이 지역을 대상으로 업무를 처리한다. 이 그림에서 보이는 마커펜이나 지우개 같은 도구는 마우스 조작으로 움직일 수 있다. 도구를 놓을 수 있는 부분이 보이고, 이 도구들을 이용하여 지도를 다룰 수 있다. 즉, 마커펜을 이용하여 중요한 지역을 표시할 수 있다.
- 가상 도구(virtual tools)들은 돋보기, 줄자, 메모지 등을 포함한다. 이 가상도구들은 필요한 경우 더 만들 수 있으며, 돋보기를 통하여 특정지역을 확대해 보거나 줄자를 통하여 거리를 재고, 메모지를 통하여 특정 지역이나 사실에 대해 월보드에 메모해둘 수도 있다. 쓰레기통이 있어서 조작 과정에서 나오는 여러가지 쓰레기들을 담아두었다가 복원할 수도 있다. 월보드의 기본 기능으로는 Selection,

Zoom, Pann, Rotation, Navigation 등을 들 수 있다.



[그림 17] 월보드의 기본 구상

- 통합 지리정보 서비스에 있어서 월보드가 부각되는 것은 구현 과정에 그 기반 기술로 분산 객체 기술(Distributed Object Technology)을 많이 사용하기 때문이다. 이 기술은 현재 전세계적으로 그 시장 규모가 크게 확대되고 있으며, 국내에서도 대부분의 소프트웨어 엔지니어링 업체가 채용하고 있거나 채용하려는 기술이다. 특히, 각각의 노드를 관리하거나 그들간의 인터액션(interaction)이 자유롭게 이루어지기 위해서는 다양한 기술적 검토가 있어야 하는데 분산 객체 기술은 이러한 작업에 있어 튼튼한 이론적, 기술적 기반을 제공해 준다.
- 월보드에서 사용하는 전자지도는 그 안에 들어있는 각각의 그래픽이 단순한 이미지가 아닌 객체로서 움직여야 사용자가 자신이 원하는 대로 조작할 수 있게 된다. 이 경우에 수많은 공간 객체가 일관된 프로세스에 의해 다중 사용자에게 전달되어야 한다는 문제가 생기게 된다. 멀티 지도 뷰는 이러한 문제를 해결할 수 있는 핵심 기술 중의 하나이다. 분산된 노드들에 존재하는 클라이언트들은 각각의 뷰를 가지고 있지만 그 뷰는 하나의 동일한 데이터 모델에 기반하고 있다. 이것은 일반적으로 MVC(Model-View-Controller)라고 불리는 소프트웨어 기술로 하나의 뷰가 보내는 정보를 자신만이 볼 수도 있고 여러 사람들에게 전송할 수도 있게 한다.
- 결국, 통합 지리정보 서비스를 염두에 둔 개방형 시스템은 협력 지원 지리정보 시스템에서 요구하는 기능들을 수렴할 수 있어야 한다. 협력 지원 지리정보 시스템의 대표적인 예로 부각되고 있는 월보드는 분산 객체 기술과 **멀티 지도 뷰의 설계 및 구현**을 통해 현실화할 수 있으므로 개방형 시스템의 기본적인 매핑 커널에는 이러한 기능이 필수적으로 요구된다고 하겠다.

제 2 절 전체 시스템 요구사항

1. 기본적 고려사항

- 분산 객체 기술을 사용하는 개방형 시스템을 설계하는 경우 가장 기본적인 고려사항으로는 다음과 같은 세 가지를 제시할 수 있다.

  ○ **분산 시스템**
  시스템 사용자들은 네트워크 상에 분산되어 있으므로 분산 시스템을 기본 개념으로 하여 설계하여야 한다.

  ○ **객체 지향 시스템**
  시스템의 유연한 확장과 재사용을 보장하기 위해 객체 지향 시스템으로 개발하여야 한다. 객체 지향 시스템은 기존의 시스템을 확장하기 위해 필요한 메커니즘을 제공한다.

  ○ **컴포넌트 기반 시스템**
  모듈 단위의 재사용성을 확보하고 유지, 관리를 자유롭게 하기 위해서는 컴포넌트 단위로 기능을 구분할 필요가 있다. 컴포넌트 기반의 시스템을 만들기 위해서는 가능한 컴포넌트 프레임워크를 만들고 각 컴포넌트를 정의하는 것이 필요하다.

- 앞 절에서 언급된 내용을 기반으로 개방형 시스템을 설계 및 구현하기 위한 기본적 고려사항을 다음과 같이 제시하였다.

  ○ **데이터와 서비스가 모두 통합 가능해야 한다.**
  기 구축된 시스템은 그 아키텍처가 개방됨으로써 데이터와 서비스 모두 공유 또는 통합될 수 있다. 이것은 단순한 데이터 통합의 차원을 넘어 기 구현된 기능을 사용할 수 있는 서비스 통합(Service Integration)의 단계를 의미한다.

  ○ **애플리케이션 중심의 시스템이 되어야 한다.**
  GIS 애플리케이션은 다양한 요구사항(user requirement)을 반영하고 있다. 상하수도 관리 시스템, 환경정보 시스템, 네트워크 관리 시스템 등 다양한 GIS 기술을 원용하는 도메인에서 필요로 하는 기능들은 매우 다양하므로 이렇게 다양한 요구사항들을 즉시 해결할 수 있는 시스템이 되어야 한다. 이를 위해 시스템의 확장성(extensibility), 이식성(portability), 이동성(mobility) 등이 고려되어야 한다.

○ 오브젝트 웹 환경을 고려해야 한다.

차세대 컴퓨팅 환경이라고 할 수 있는 오브젝트 웹은 강력한 인터넷 기반 시스템을 중심으로 웹 브라우저에서 상태(state)를 저장, 교환할 수 있는 분산 시스템으로의 열망을 표현하고 있다. GIS 시스템에서도 이와 같이 네트워크 상의 이동성과 클라이언트/서버간 상태 교환 및 저장 기능을 요구할 경우, 코바/자바 기반의 오브젝트 웹이 가지는 장점을 최대한 활용하는 전략을 필요로 한다.

○ 핵심 라이브러리 중심의 개발 전략을 세워야 한다.

다양한 애플리케이션의 요구사항과 다양한 tier, 플랫폼에서의 GIS 기능 구현 등 빠르게 변화하면서도 다양한 현재의 운영 환경에 적응하기 위해서는 필수적이고 핵심적인 라이브러리의 설계 및 구현을 기본 전략으로 채택하는 것이 바람직하다. 차세대 오브젝트 웹 환경에 적응할 수 있는 유연(flexible)하면서도 확장 가능(extensible)한 라이브러리를 먼저 구현하는 것이 아직 이전의 시스템 운영 방식에 머물러 있는 기존 벤더들의 기술 수준을 추월하고, 새로 부상하는 GIS 틈새 시장을 공략할 수 있는 방법이다.

- 본 과제에서는 이와 같은 기본적 고려사항들을 점검하여 개방형 GIS 시스템 개발 전략의 일환으로 소프트웨어 운영 플랫폼을 코바/자바로 정하였다. 코바 기반의 시스템은 분산 환경에서 이질적인 시스템으로의 접근을 가능하게 하며, 자바 기반의 시스템은 컴포넌트 아키텍처의 수용 등을 포함함으로써 확장성, 이식성을 높여준다. 따라서, 본 과제의 기본 전략은 **코바/자바를 기반으로 필수적인 GIS 라이브러리를 우선적으로 설계, 구현하는 것으로 한다.**
- [표 3]은 본 과제에서의 고려사항과 이를 해결하기 위해 선택한 전략을 개략적으로 보여주고 있다.

고려사항	선택 전략
분산 컴퓨팅 표준 API 데이터 및 서비스 통합	코바
개방형 시스템 표준 API	OpenGIS 명세
이식성 이동성 객체지향	자바
웹 서비스	웹 서버, JSP
확장가능 라이브러리 개발	컴포넌트 웨어

[표 3] 기본적 고려사항과 선택 전략

2. 핵심 라이브러리

- 여러 GIS 응용 프로그램에서 사용 가능한 핵심 기능(essential function)들에는 여러 가지가 있다. 그러나, 그 중에서도 가장 기본적인 기능은 데이터 관리(data management)와 매핑(mapping) 기능이다.

  ○ 데이터 관리

  공간 데이터를 처리하기 위한 가장 기본적인 단계는 데이터를 구축하고 관리하는 일이다. 공간 데이터를 효율적으로 처리하기 위해서는 공간 데이터베이스를 이용하는 것이 바람직하며, 단순 레이어 기반의 시스템으로도 충분한 경우에는 파일베이스(filebase) 시스템으로도 구동이 가능하다.
  라이브러리 형태의 시스템을 설계하는 경우에는 먼저 파일 기반의 시스템을 구현함으로써 로컬 시스템에서의 데이터 로딩/저장, URL을 이용한 네트워크 상의 로딩/저장을 가능하게 할 수 있다. 본 과제에서는 호환 가능한 공간 데이터 파일로 shapefile 포맷을 사용한다. 즉, 하나의 공간 데이터를 파일로 저장하는 경우에는 *.shp, *.shx, *.dbf 와 같은 세 종류의 파일이 있으면 충분하다.

  ○ 매핑 기능

  매핑은 GIS를 특징짓는 "시각화(visualization)"의 기본 요소라 할 수 있다 따라서, 많은 경우에 공간 데이터를 얼마나 효과적으로 디스플레이할 수 있는가가 그 시스템의 효율성과 우수성을 증명하는 방법으로 이용된다. 이 매핑 기능은 클라이언트 tier, 서버 tier, 애플리케이션 서버 tier 등 모든 tier에서 사용될 수 있으므로 가장 기본적이면서도 핵심적인 기능이라고 할 수 있다.

- 공간 연산자(spatial operator) 의 설계와 구현은 매핑과 더불어 가장 중요한 GIS 기능이라 할 수 있다. 그러나, 공간 연산자는 많은 경우에 공간 데이터베이스의 기본 기능으로 제공되며, 매핑 기능의 구현에 비해 우선순위가 낮으므로 본 과제의 기본 구현은 매핑 라이브러리로 한다. 물론 매핑 라이브러리의 기본 기능으로 공간 데이터 처리 모듈(geometry module)과 공간 객체 관리 모듈(feature module)이 필요하고 실제 설계상에 포함되지만 개념상의 단순화를 위해 매핑 라이브러리의 설계 및 구현을 목표로 세운다.

3. 기존 시스템의 한계점과 대안

- 현재 많은 GIS 시스템 컴포넌트들이 OLE/COM 모델에 기반하고 있다. 이러한 접근 방식은 ArcInfo[11]와 같은 기존 시스템의 핵심 기능들을 그대로 이식할 수 있게 함으로써 빠른 시스템 설계 및 구현을 보장한다. 그러나, 윈도우(Windows™)가 아닌 이질적 시스템(heterogeneous system)에서도 문제없이 운용될 수 있어야 하는 오브젝트 웹 GIS를 위한 이동가능한 매핑 시스템(mobile mapping system)을 설계하는 데에는 부적합하다.
- GIS 시스템을 구성하는 기본적인 컴포넌트들이 코바와 자바를 기반으로 하는 경우에는 자바 가상 기계가 작동하는 모든 시스템에서 오브젝트들이 운용 가능하며, IIOP 채널을 통해 의사소통할 수 있으므로 이기종 머신간의 공간 데이터 및 서비스 통합이 가능하다.
- 클라이언트의 측면에서 지리공간 데이터(geo-spatial data)를 OpenGIS Geometry 구조체나 Feature 인터페이스를 통해 네트워크 상에서 가져오는 일은 서버쪽 오브젝트(server-side object)를 구현하는 것보다 단순하다. 그러나, 대부분의 클라이언트 애플리케이션이 표준 포맷의 데이터를 읽어들일 때 유사한 방식을 사용한다는 것은 이 과정을 캡슐화할 필요가 있다는 것을 의미한다. 기존의 시스템들은 사용자가 서로 다른 데이터 포맷이나 시스템 운영에 필요한 요구사항을 직접 이해하고 조작하기를 기대한다. 그러나, 사용자 중심의 애플리케이션들은 시스템 중심적인 사고를 권장하지 않는다. 사용자 중심의 개방형 시스템은 데이터 제공자의 기능 중에서 기본적이고 공통적(예를 들어, 공간 검색이나 CORBA 스켈레톤 부분)인 것은 캡슐화함으로써 어떠한 데이터 제공자나 서비스 제공자라도 이용할 수 있도록 하는 기본 컴포넌트를 설계, 제공할 수 있도록 해야 한다.
- 웹 상에서 지리공간 정보를 서비스해야 하는 경우가 많아짐에 따라 GIS 기술을 사용하는 애플리케이션의 요구사항이 변화하면서 동시에 다양해지고 있다. 그러나 시스템이 자기자신의 포맷(proprietary format)과 시스템 아키텍처를 중심으로 운용되는 경우에는 타 시스템과의 통합이 어렵게 된다. 특히, 다른 업체에서 개발한 시스템과의 통합은 표준 인터페이스를 따르지 않는다면 데이터와 서비스 통합 자체가 불가능하다. 결국, 다양한 시스템 개발 요구사항들을 만족시키기 위해서는 시스템이 개방되도록 설계, 구현되어야 하며, 이식성(portability)과 확장성(scalability)의 문제가 해결할 수 있도록 핵심이 되는 매핑 커널과 기타 다른 컴포넌트들과의 분리가 가능하도록 해야 한다.

4. 설계 가이드라인 도출

[11] product from ESRI

- 앞에서 언급한 여러 측면들을 고려하여 다음과 같은 설계 가이드라인이 도출되었다. 이는 시스템 설계에 있어서의 방향키 역할을 하는 것으로 주로 외부 요인(external factor)들이나 상위 레벨에서의 기능적 요구사항을 의미한다.

  ○ OpenGIS 명세 준수(OpenGIS compliant)

  - 데이터 전송 및 서비스 제공을 위한 표준으로서 OpenGIS 명세를 채택하는 경우 그 명세를 준수하여야 한다.

  - ORB를 통한 지리공간 데이터의 획득 과정(data fetching process)을 추상화할 필요가 있으며, 데이터 제공자의 데이터베이스 유형(FileBase, RDBMS, OODBMS)에 관계없이 동일한 방식으로 데이터 접근이 가능해야 한다.

  - 데이터의 의미 무결성(semantics)을 준수해야 한다. 예를 들어, 여러 데이터 제공자에 채널을 열어두고 데이터를 획득하는 경우 객체의 아이덴티티(Object Identity)를 유지할 수 있는 방안 등을 고려해야 한다.

  ○ 확장가능성(Extensible)

  - 객체지향 설계 방법론(OOAD) 및 잘 알려진 디자인 패턴(well-known design pattern) 또는 UML등을 이용한 투명한 시스템 설계 과정을 통해 설계된 시스템의 유지, 보수가 수월해야 하며, 객체 또는 컴포넌트의 확장이 용이해야 한다.

  - 디자인 패턴을 사용하는 경우 모델링이 수월해지며, 객체지향 시스템 아키텍처를 표방하는 경우 커스터마이징(customizing)이 쉽다는 장점이 있다.

  ○ 유연성(Flexible)

  - 다양한 애플리케이션이 요구하는 기능에 부합할 수 있는 오브젝트의 분리가 가능해야 하며 각 컴포넌트의 네트워크 상 배치가 유연해야 한다. 이동가능하고 내장가능한(embeddable) 오브젝트로 설계함으로써 네트워크 상의 어떤 노드에서도 동일하게 운용될 수 있도록 한다.

  - 애플리케이션마다 매핑 오브젝트에 대해 요구하는 다양한 측면들을 수용할 수 있어야 한다. 예를 들어, 협력지원시스템(CSCW)에서 요구하는 브로드케스팅(broadcasting) 이나 GIS 애플리케이션 서버에서 문제가 되는 성능 개선을 위한 커스터마이징이 가능해야 한다.

  - 어떠한 유형의 배치도 수용할 수 있어야 한다. 멀티 티어 방식의 배치에서는 매핑 오브젝트들이 어느 tier에 존재해야 하는가가 미리 정해져 있는 경우가 거의 없기 때문이다.

  ○ 소프트웨어 유지/관리

  - 본 과제의 연구개발 진행 방식이 핵심 라이브러리와 같이 우선순위가

높은 것부터 순차적으로 설계 및 구현하는 방식을 취하므로 소프트웨어의 업데이트와 전체적인 통합 등에 있어서 효율적인 유지/관리가 필수적이다

- 다양한 개발 과정 및 시스템의 여러 측면을 표현하고 이해할 수 있는 통합 프로세스가 필요하다. 따라서, UML(Unified Modeling Language)과 RUP(Rational Unified Process)에 기반한 소프트웨어 개발 방법론을 수용하기를 권장한다.

○ 100% 순수 자바

- 차세대 오브젝트 웹 환경에 효과적으로 적응할 수 있도록 100% 순수 자바를 권장한다. 자바가 갖는 장점을 최대한 살릴 수 있도록 한다.

제 3 절 소프트웨어 개발 방법론

1. OMT

- OMT는 J. Rumbaugh가 고안한 것으로 90년대에 소개된 이후 전세계에서 널리 사용되어 온 방법이다. 이 방법은 하나의 시스템을 다양한 모델을 이용하여 표현하는데, 크게 객체 모델(Object Model), 동적 모델(Dynamic Model), 기능 모델(Functional Model)로 나누어진다.
- 객체 모델은 클래스 다이어그램과 서브시스템 다이어그램, 그리고 객체 다이어그램으로 표현된다. 동적 모델은 상태 다이어그램, 유스케이스 다이어그램, 동시성 객체의 메시지 다이어그램으로 표현된다. 기능 모델은 객체 메시지 다이어그램(또는 인터액션 다이어그램), 객체 지향 데이터 흐름 다이어그램으로 표현된다.
- OMT 방법은 분석(analysis), 시스템 설계(system design), 객체 설계(object design), 구현(implementation)의 4단계로 구분되며, 이 단계들이 반복적으로 수행된다. 분석 단계에서는 객체 모델, 동적 모델, 기능 모델이 도출된다. 시스템 설계 단계에서는 시스템의 기본 아키텍처가 도출된다. 객체 설계 단계에서는 더 세부적인 객체 모델, 동적 모델, 기능 모델이 정의된다. 구현 단계에서는 개발 작업을 수행하게 된다.

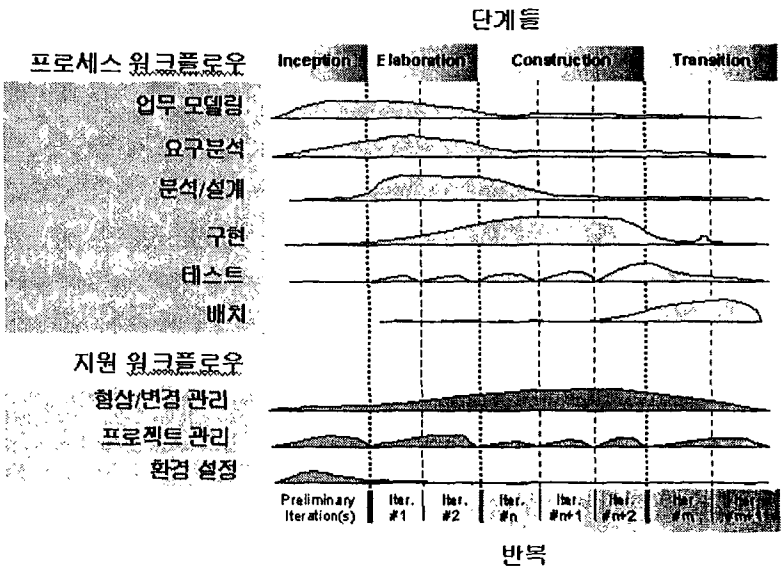
2. UML 모델링

- 본 과제는 시스템 개발 전반을 관리하기 위해 UML(Unified Modeling Language)을 사용하였다. UML은 객체지향 분석, 설계에 관한 기존의 다양한 객체지향 방법론들 즉, Rambaugh의 OMT(Object Modeling Technique), Booch의 방법론 등을 통합한 표준 모델링 언어이다.
- UML의 주요 구성요소로는 모델요소(model component), 관계(relationship), 확장 메커니즘(extension mechanism), 다이어그램(diagram) 등 네 가지가 있다. 모델 요소 중에서 구조적인 요소들은 모델을 구성하는 정적인 부분들이며, 개념적이거나 물리적인 요소들을 표현한다. 행위적 요소들은 모델에서 동적인 부분들이며, 시간과 공간에 따른 이벤트의 발생 또는 행위를 표현한다. 그룹화 요소들은 모델 구성 요소들을 조직화하는 부분이다. 관계는 모델 요소들을 의미적으로 연결하는 부분이다.

3. RUP

- RUP(Rational Unified Process)는 프로젝트 관리 및 시스템 엔지니어링과 관련된 경험적인 방법론을 제공한다. 특히 작업 과정에서 적절하게 사용할 수 있는 템플릿을 제공한다. 그 특징으로는 비주얼 모델링, 반복적 개발(iteration), 아키텍쳐 중심, 다양한 프로세스의 적용 등을 들 수 있다.
- 비주얼 모델링은 필요한 경우 구체적인 부분을 숨기거나 드러냄으로써 모델의 관리를 용이하게 하며, 또한 산출물 간의 일관성 유지가 가능하도록 한다. 유스케이스는 시스템에 요구되는 행위를 찾아내고 프로젝트에 관련된 사람들끼리 의사소통하며 이를 검증하는 중요한 요소이다. 반복적인 개발은 B. Boehm의 나선형 모델을 기반으로 프로젝트가 가지는 위험 요소를 조기에 발견함으로써 시기적절하고 효과적인 대처가 가능하도록 한다. 이것은 지속적으로 요구사항들을 발견, 분석함으로써 매 반복시마다 개발팀들이 예측 가능하고 효과적인 방법으로 프로젝트 산출물을 만들어 내도록 도와준다.
- 시스템 아키텍쳐는 시스템을 바라보는 다양한 관점들을 관리하는 데 사용되는 가장 중요한 산출물 중의 하나로 시스템 개발의 전 과정에 걸쳐 반복적이고 점진적인 개발을 제어하는 역할을 한다.
- 구현 과정에서는 서브시스템들을 구현하기 위한 관점에서 실제 코드들을 조직화해 나간다. 컴포넌트의 관점에서 클래스들을 소스 파일, 바이너리 코드, 실행 가능한 파일 등으로 구현한다. 개발된 컴포넌트 단위로 테스트를 수행하며, 개별적인 개발자 혹은 팀들에 의해 개발된 결과물들을 실행 가능한 시스템으로 통합한다.
- 테스트 과정에서는 제품의 품질을 평가하고 이를 문서화한다. 소프트웨어의 모든 컴포넌트들은 이 과정에서 제대로 통합되었는지 확인된다. 이렇게 테스트를 마친 컴포넌트들은 적절한 과정을 거쳐 배포된다.
- RUP에서 제공하는 워크플로우는 형상 관리, 환경 설정, 프로젝트 관리 등이 있다. 형상 관리는 프로젝트 자산에 대한 무결성을 유지하는 것이며, 환경 설정은 소프트웨어 개발 조직에게 필요한 적절한 프로세스나 개발 도구와 같은 개발 환경을 제공하는 것, 그리고 프로젝트 관리는 프로젝트를 계획하고 인력을 배치하며, 진행하고 모니터링을 하기 위한 실질적인 지침서를 제공하는 것을 목적으로 한다.
- Rational Unified Process 는 프로젝트를 네 단계(개념화, 초기 구축, 시스템 완성, 전이 단계)로 나누며, 각 단계는 각각 일회 이상 반복(Iteration)되는 개발 과정으로 구성된다. 한편, 이와 같은 반복적 접근법에서는 이정표(Milestones)를 통해 프로젝트가 매 반복 시마다 보다 나은 결과물을 향해 좀 더 발전된 방향으로 진행되고 있다는 것을 평가하고, 다음 단계로 이행할 시점과 기준을 제시한다.
- 프로제트의 시작 국면이라고 할 수 있는 '개념화 단계(Inception)'에서는 프로젝트의 최종 결과물에 대한 비젼과 비즈니스 사례를 명시하고, 프로젝트의 범위를

정의한다. 또한, '초기 구축단계(Elaboration)'에서는 프로젝트를 계획하고, 프로젝트의 특징을 명시할 뿐 아니라 기본 아키텍쳐 설계 작업을 수행한다. 그러나, 프로젝트의 실질적 수행은 '시스템 완성 단계(construction)'에서 이루어 진다. 한편, 프로젝트 수행의 최종 과정이라고 할 수 있는 '전이 단계(Transition)'에서는 완성된 시스템을 최종 사용자 집단에 배포하고, 사용자가 만족할 때까지 이에 대한 훈련, 지원, 유지보수를 수행한다.



[그림 18] RUP의 단계와 반복

- 또한, [그림 18]의 각 단계와 반복은 각각의 워크플로우로 구성되어 있는데, 반복적인 개발 접근을 통해 각 워크플로우의 중점 내용은 개발 주기 내의 단계와 반복에 의존하여 다양하게 구성된다. RUP의 워크플로우는 크게 핵심 프로세스 워크플로우와 지원 워크플로우로 구분할 수 있다.
- RUP의 핵심 프로세스 워크플로우에는 다음과 같은 것들이 있다.

  ㅇ 업무 모델링

  조직의 구조와 작업을 이해한다. 고객과 최종사용자, 개발자들이 조직에 대한 동일한 이해를 가지게 한다. 조직을 지원하는데 요구되는 시스템 요구사항을 도출한다.

  ㅇ 요구분석

  시스템이 어떤 기능을 수행해야 하는지에 대한 고객과 사용자간의 합의를 도출한다. 시스템 개발자에게 시스템에 대한 요구사항의 더 나은 이해를 제

공한다. 시스템의 기능을 정의한다. 반복(iteration)의 기술적 내용에 대한 계획의 기초를 제공한다. 시스템 개발을 위해 필요한 비용과 시간을 추산하는데 기초를 제공한다. 시스템에 대한 사용자 인터페이스를 정의한다.

- **분석/설계**

  요구사항을 어떻게 시스템을 구현할 것인지에 대한 스펙(specification)으로 변환하는 것이다.

- **분석**

  분석의 목적은 시스템 요구사항을 소프트웨어 설계자 관점의 형태로 변화시켜 가는 것으로서, 클래스와 서브시스템, 시스템의 기능적 요구사항에 초점을 두고 있다. 따라서, 대부분의 비기능적 요구사항과 구현환경에 대한 고려는 이루어지지 않는다.

- **설계**

  설계의 목적은 앞서 수행한 분석의 결과를 비기능적 요구사항, 구현환경 등에 적용시키는 것으로서, 분석을 더욱 정련(refinement)해나가는 과정을 포함한다.

- **구현**

  서브시스템을 구현하기 하기 위한 관점에서 실제 코드들을 조직화해 나간다 컴포넌트 관점에서 클래스들을 소스 파일, 바이너리 코드, 실행 가능한 파일 등으로 구현한다. 또한, 개발된 컴포넌트를 단위로 테스트를 수행한다. 개별적인 개발자 혹은 팀들에 의해 개발된 결과물들을 실행가능한 시스템으로 통합한다.

- **테스트**

  제품의 품질을 평가하고, 이를 문서화하는데 있다. 객체들과 컴포넌트들 간의 상호작용을 검증한다. 소프트웨어의 모든 컴포넌트들이 적절하게 통합되었는지를 검사한다. 모든 요구사항들이 적절하게 구현되었는지를 검사한다. 소프트웨어를 배치하기 전에 발견된 모든 오류들을 검정 했는지 알아본다.

- **배치**

  소프트웨어의 릴리스를 성공적으로 수행하고, 최종 사용자들에게 소프트웨어를 이전한다.

- 지원 워크플로우에는 다음과 같은 것들이 있다.

  - **형상/변경 관리**

    형상관리와 변경관리의 목적은 프로젝트의 자산들에 대한 무결성을 유지하는 것이다.

  - **형상관리**

산출물의 버전, 소프트웨어의 개발, 프로젝트의 적절한 운영, 이질적인 컴퓨팅 환경의 지원, 개발 위치의 투명성, 병렬적인 개발과정이 포함된다.

O **변경관리**

개발 중인 시스템의 결함 수정, 고객의 추가적인 기능 개선 요구를 시스템에 반영하고, 프로젝트의 현재 상황에 대한 모니터링 작업을 수행한다.

O **환경설정**

소프트웨어 개발 조직에게 필요한 적절한 프로세스와 개발 도구와 같은 개발 환경을 제공한다.

O **프로젝트 관리**

프로젝트를 계획하고, 인력을 배치·진행시키며, 모니터링을 하기 위한 실질적인 지침서를 제공한다. 또한, 위험 관리를 위한 틀을 제공하기도 한다.

4. 소프트웨어 개발 환경

가. 코바

- 현재 분산객체 환경을 실현하기 위한 기반 기술로 OMG가 제시하고 있는 CORBA가 산업 표준으로 자리잡아 가고 있다. OMG는 분산 객체 표준을 제정하기 위해 1989년 설립된 전산업계들 간의 컨소시엄으로 현재 800여 개 업체가 회원으로 등록되어 있다. OMG에서 제시하고 있는 CORBA 미들웨어는 분산된 환경 하에서 응용 프로그램들을 통합하고 상호운용할 수 있는 OMA(Object Management Architecture)라는 표준 기술에 근거하고 있다. OMA는 분산 애플리케이션의 개발과 객체의 생성, 소멸, 저장, 트랜젝션(transaction) 기능 등 분산 객체 환경에서 필요한 모든 서비스를 총칭한다. OMA는 다음과 같은 구성요소를 가지고 있다.

O **ORB** (Object Request Broker)

ORB는 분산 환경에서 객체들이 구현된 위치를 알지 못하더라도 서로 요청하고 응답할 수 있도록 지원한다. ORB는 분산, 이질적인 환경에서 애플리케이션 간의 상호운용을 보장하기 위한 기본 구성요소이다.

O **객체 서비스** (Object Service)

객체 서비스는 객체들의 사용과 구현을 위한 기본적인 기능들을 제공하는 서비스 집합이다. 서비스는 어떠한 분산 애플리케이션을 위해서도 사용될 수 있기 때문에 특정한 애플리케이션과는 독립적이다.

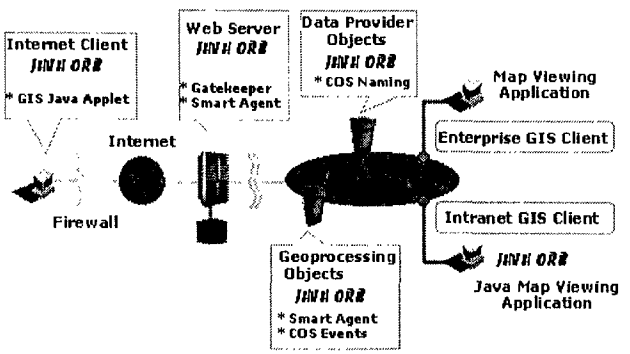
O 공통 서비스 (Common Facilities)

공통 서비스는 애플리케이션들이 공통적으로 사용할 수 있지만, 객체처럼 기본적인 것이 아닌 서비스들의 집합을 의미한다. 이 서비스에는 크게 두 가지 종류가 있는데, 하나는 시스템 관리, 그래픽 사용자 인터페이스(Graphic User Interface)와 같이 대부분의 애플리케이션에서 사용하는 수평적 공통 서비스 (Horizontal Facility Service)와 특정 응용에 적합한 수직적 공통 서비스 (Vertical Facility Service) 가 있다.

O 애플리케이션 객체 (Application Object)

애플리케이션 객체는 일반 개발자들이 개발하는 것으로 기존의 애플리케이션에서 요구하는 기능들과 다를 것이 없다.

- OMG는 분산 애플리케이션들 간의 상호운용성을 위하여 프로그래밍 언어가 아닌 인터페이스 정의 언어(Interface Definition Language : IDL)를 제공한다. OMG IDL을 사용하여 인터페이스와 데이터 구조를 정의할 수 있으며, 실제 구현에 있어서는 C, C++, Java, Smalltalk 등과 같은 프로그래밍 언어를 사용할 수 있다. 개발자들이 이 공통 인터페이스에 맞추어 시스템을 만드는 경우 IDL과 언어간의 매핑 과정을 통해 완전한 상호운용성이 보장된다.
- [그림 19]는 ORB 제품 중의 하나인 Visibroker™의 전략을 원용한 것으로 CORBA 기술을 사용하는 경우, 효율적인 오브젝트 배치가 가능하다는 점을 강조한다. 이와 같은 전략을 이용하기 위해서는 기존 시스템의 확장도 필요하지만 새로운 환경에 적합한 새로운 GIS 컴포넌트들의 개발이 더욱 필요하다.
- 코바/자바 기반의 시스템으로 구현하는 경우 C나 C++와 연동하는 시스템보다 시스템의 성능이 떨어지는 문제가 있을 수 있으나, 차후의 객체 이동성 등을 고려할 경우 100% 순수 자바 전략을 채택하는 것이 바람직하다.



[그림 19] CORBA 기반 전자지도 출판 및 관리 전략

(adapted from Visibroker strategy, see http://www.inprise.com)

나. OpenGIS 표준 명세

- OpenGIS 컨소시엄(OpenGIS Consortium : OGC)은 1994년 공간 정보 처리의 상호운용성을 위해 조직된 비영리 단체로 세계 각국의 산업계, 정부, 학계가 다수 참여하고 있다. OGC에서는 공간 정보의 상호운용성을 위해 OpenGIS 명세를 개발하고 관리한다. OpenGIS 명세는 소프트웨어 개발자들에게 사용자가 다양한 데이터 소스에 접근하고 이를 처리할 수 있도록 하는 표준을 제공한다.
- OpenGIS 프로젝트의 목적은 애플리케이션 개발자가 분산 네트워크 환경에서 공간 데이터와 공간 데이터 처리 기능을 사용할 수 있도록 하는 기술을 규정하는 것이다. OpenGIS 명세는 사상(feature), 커버리지(coverage) 등을 포함하는 개방형 지리데이터 모델과 지리정보를 접근, 교환, 관리, 조작, 표현할 수 있도록 하는 서비스 모델, 그리고 의미적인 상호운용성(semantic interoperability)을 실현하기 위한 방법으로 정보공동체 모델(information community model)을 정의한다.
- 이 명세는 여러 층의 추상 레이어들을 가지고 있으므로 특정한 운영체제, 프로그래밍 언어, 운영 환경, 분산 컴퓨팅 플랫폼(distributed computing platform)에 독립적이다. 현재 개발 중인 분산 컴퓨팅 플랫폼을 기반으로 상호운용이 가능한 GIS 시스템을 개발하기 위해 OpenGIS 명세에서는 세 가지 개념적 층을 제시한다.

  ○ 핵심 모델 (Essential Model)
  실세계 모델을 의미한다.
  ○ 사양 모델 (Specification Model)
  소프트웨어 구현과 독립적인 일반 모델을 의미한다.
  ○ 구현 모델 (Implementation Model)
  특정 환경에서 동작하는 소프트웨어 객체 모델을 의미한다.

- OpenGIS는 현재 구현 명세로 OLE/COM, CORBA, SQL 관련 명세를 정의, 배포하고 있으며, 지속적으로 갱신되고 있다. OpenGIS 명세를 정의하는 데 있어서는 객체지향 접근방법을 기본적으로 사용한다. 객체지향 모델링은 객체라는 블록들을 이용하므로 프로그램 안에서 객체를 하나의 단위로 보고 처리할 수 있도록 한다.
- OpenGIS에서 주장하는 "사양에 의한 상호운용성(Interoperability by means of specification)"은 소프트웨어 개발자가 합의된 사항을 준수하여 개발함으로써

상호운용성이 보장된다는 것을 의미한다.

제 4 절 매핑 커널 설계

1. 요구사항 분석

가. 유스케이스 모델

- 시스템이 실제 사용자에 의해 사용되거나 시스템 간의 상호작용을 알아보기 위한 기본적인 업무 모델링으로 유스케이스 모델링(UseCase Modeling)을 사용한다. 유스케이스 모델링은 시스템이 어떠한 일을 수행할 것인가를 분명하게 서술하는 모델링 기법으로 시스템의 외부에서 시스템과 상호작용하는 사람 혹은 시스템(actor), 그리고 이들이 수행하는 어떤 기능(UseCase) 와 이들간의 관계에 대해 정의할 수 있다.
- 일반적으로 개발자와 사용자 간에 시스템의 요구사항에 대한 합의 과정 동안 시스템의 기본적인 기능들에 대해 결정하게 되며, 이러한 내용들을 유스케이스 모델로 서술하게 된다.

나. 일반 기능

- GIS 시스템에서 사용되는 일반적인 매핑 라이브러리는 공간 데이터의 디스플레이와 속성 데이터로의 링크(link)를 지원하는 기능을 공통적으로 가지고 있다. 그러나, 여러 벤더들이 개발한 각각의 매핑 라이브러리는 그 기능이 다양하므로 일반적 기능과 확장 기능으로 구분하고 그 중에서 핵심적인 기능들을 정의할 필요가 있다.
- 일반적인 매핑 라이브러리의 기능으로는 다음과 같은 것들이 있을 수 있다.

  ○ 뷰 관리(View Management)

  뷰는 시스템에서 관리되고 있는 공간 데이터를 디스플레이하기 위한 공간을 정의한다. 이는 실제 지구상의 일정지역(연구지역)을 표현하는 것으로 스케일(Scale Management), 투영법(Projection), 범례 관리(Legend Control) 등을 포함한다.

  ○ 레이어 관리(Layer Management)

  유사한 특성을 갖는 객체들의 집합으로 규정될 수 있는 레이어는 도로, 토

지이용, 빌딩 등과 같이 관심있는 객체들의 효율적인 관리 및 처리를 위해 필요하다.

레이어 기반의 매핑 라이브러리는 기존의 공간 데이터 관리 개념과 일치한다는 점과 데이터를 효율적으로 관리할 수 있다는 두 가지 장점을 가지고 있다. 또한 편리하게 유사한 객체들끼리 그룹화할 수 있다.

객체지향 아키텍쳐를 가진 시스템은 레이어 개념을 도입하지 않는다. 이 시스템에서는 모든 지구상의 객체가 그 자체로서 의미를 가지므로 객체 단위로 처리된다.

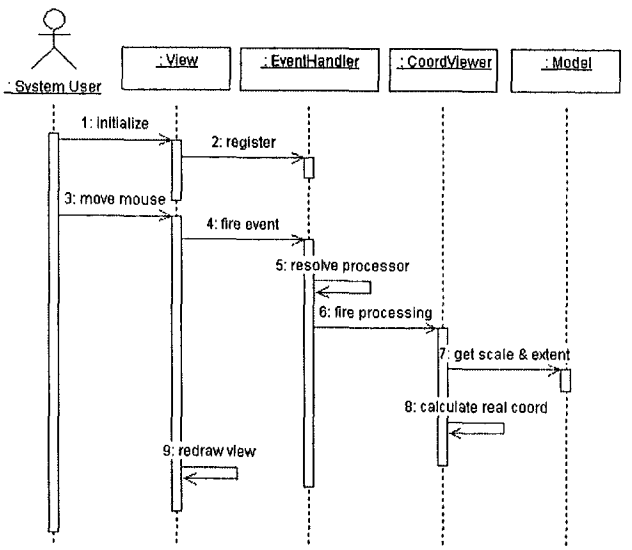
○ 객체 관리(Object Management)

공간 데이터를 객체로 인식하고 처리하는 것은 실제 시스템 설계 및 개발 시스템의 운영을 용이하게 한다. 또한 점, 선, 면과 같은 기본 벡터 데이터 타입과는 별도로 빌딩, 도로, 필지 등과 같이 확장된 클래스들을 정의함으로써 공간 데이터 모델링과 시스템 개발의 불일치를 줄일 수 있다.

○ 이벤트 처리(Event Handling)

윈도우 기반의 GIS 시스템은 사용자 이벤트를 효율적으로 받아들여 처리할 수 있어야 한다. 객체지향 기반 시스템은 이벤트 처리를 담당하는 독립적인 클래스를 두고 뷰에 등록하는 방식을 취하고 있다. 이벤트 처리 오브젝트는 매핑 라이브러리 또는 공간 데이터 처리 오브젝트의 메소드를 호출함으로써 사용자 이벤트와 시스템의 기능을 연결시킨다. 가장 대표적인 예로는 마우스의 움직임에 따라 현재의 좌표값을 디스플레이하는 경우를 들 수 있다.

[그림 20]은 이러한 이벤트 처리의 예(좌표값 디스플레이)를 시퀀스 다이어그램으로 보여주고 있다.



[그림 20] 좌표값 디스플레이를 위한 시퀀스 다이어그램

○ 모델 또는 공간 데이터베이스(Model or Spatial Database)

공간 데이터와 속성 데이터의 효율적인 관리를 위해 객체들을 통합 관리하는 모델 클래스의 정의가 필요하다. 메모리 상에서의 객체 처리는 시스템 리소스(system resource)를 많이 차지하므로 대용량 데이터의 처리를 위해서는 데이터베이스와 통합된 모델 클래스를 정의하기도 한다.

○ 속성 데이터 링크(Link to Attributes)

일반 데이터와 공간 데이터를 통합하여 처리하는 방식으로 사용되는 링크(link)는 공간 데이터를 통한 속성 보기와 속성 데이터 분류를 통한 공간 데이터 디스플레이로 크게 분류될 수 있다.

이러한 링크를 구현하는 것은 실제로 모델 또는 데이터베이스에 포함된 객체와 그 객체에 연결된 객체 정보를 통합하여 관리하는 것을 의미한다.

다. 확장 기능

• 매핑 라이브러리는 앞에서 정의한 기본 기능 이외에 시스템 개발자가 해야 하는 작업을 미리 확장 정의할 수 있다. 다음은 그 중 가장 대표적인 예이다.

○ 지도 레이블링(Map Labelling)

뷰에서 관리되고 디스플레이되는 객체들 중에는 공간 객체 이외의 객체들이 있을 수 있다. 가장 대표적인 것이 텍스트 레이블(text label)이며, 이는 속성 데이터 중에서 선별되어 공간 객체를 설명하는 객체로 사용될 수 있다. 이때, 레이블이 디스플레이되는 위치를 결정하는 방식이 지도 레이블링이라 할 수 있다.

지도 레이블링은 단순히 객체 중심에 레이블을 표시하는 경우, 레이블의 중첩, 중복이 발생할 수 있다는 문제점으로부터 시작된다. 이를 해결하기 위해 레이블링을 자동화하는 객체들을 구현할 수 있다.

○ 주소 찾기(Address Matching)

주소를 이용한 지도상의 위치 검색은 많은 애플리케이션이 사용자의 주소 정보를 입력으로 받는다는 점에서 매우 유용하다. 그러나, 우리나라의 경우 주소 찾기를 위한 기본적 체계(예를 들어, 도로 중심의 새주소 체계)가 아직 갖추어져 있지 않으므로 이 기능은 우선순위가 낮다고 보여진다.

라. 분산 환경에서의 문제점

- 앞에서 기술된 기능들을 분산 환경에서 구현하는 경우, 여러가지 문제점들이 발생할 수 있다. 다음은 그 문제점들 중의 대표적인 것을 간략히 기술한 것이다.

  ○ 데이터 타입(Data Type)

  일반적으로 공간 데이터의 관리 및 조작을 위해 사용되는 데이터 구조와 그래픽 디스플레이를 위한 데이터 구조는 어느 정도의 차이점을 보인다. 특히, 데이터 타입의 경우 OpenGIS Geometry를 포함한 대부분의 공간 데이터 포맷이 double 형을 지원하는 데 반하여 그래픽 라이브러리들은 float 형을 지원하는 것이 보통이다.

  고유 데이터 구조를 가지고 있는 여러 유형의 오브젝트를 디스플레이할 수 있는 매핑 오브젝트란 다양한 유형의 데이터들을 문제없이 처리할 수 있어야 한다는 것을 의미한다. 이러한 문제는 표준 API를 통해 네트워크 상에서 데이터가 획득(fetching)될 경우 서버측에서 타입을 통일하므로 문제가 단순해질 수 있다. 단, double 형과 float 형이 가지는 처리 속도에 차이가 있을 수 있는데, 처리 속도는 자바 가상 기계를 기반으로 처리하는 시스템에서는 중요한 문제가 되므로 새로 설계되는 매핑 라이브러리가 float 형을 지원하는 것도 하나의 대안이라고 보여진다.

  ○ 심볼 서비스(Symbol Service)

  일반적인 매핑 라이브러리는 심볼에 대한 고유 포맷을 가지고 있거나 그 정보를 속성으로 저장한다. 따라서, 사용자가 심볼을 새로 정의하기 위해서는 내부 클래스들에 접근할 수 있는 메커니즘이 있어야 하며 심볼 자체는 외부에 노출되지 않는다.

  이러한 경우에는 다른 시스템에서 정의한 심볼을 공유할 수 없다는 문제가 있으며, 원격지의 심볼 정의를 가져다 쓸 수 없으므로 공간 데이터 자체는 동일하게 사용하더라도 그것을 디스플레이하는 방식은 다르게 된다. 그래픽 디스플레이의 측면에서 보면 분산 컴퓨팅 환경하에서의 심볼 정보 공유는 가장 기본적인 매핑 기능 중의 하나가 된다.

2. 설계 가이드라인

- 매핑 커널을 설계하기 위해 다음과 같은 가이드라인을 제시하였다.

  ○ **UML** 다이어그램 작성

  시스템의 유지 관리 뿐만 아니라 시스템 구조를 명료하게 보여주기 위

하여 UML 다이어그램은 필수적이다. UML 다이어그램 중에서 각각의 클래스 또는 기능에 필요한 부분만을 선택하는 것을 원칙으로 한다. 기본적으로 UML 다이어그램은 RUP 프로세스의 일부로 간주한다.

○ 디자인 패턴의 사용

잘 알려진 디자인 패턴을 사용하는 경우 시스템 아키텍쳐가 좀더 단순하고 간결해지는 장점이 있다[7]. 또한 코드 분석 및 시스템 리엔지니어링(system reengineering)의 관점에서도 유용하다. 본 시스템 설계에 있어서는 시스템 아키텍쳐, 성능 개선 등의 측면에서 사용 가능한 디자인 패턴을 고려한다.

○ **100%** 순수 자바 고려

100% 순수 자바를 사용하는 것은 이미 본 과제 전반에 걸친 기본 요구사항으로 고려되고 있다.

○ 코바 디자인 패턴의 사용

분산 컴퓨팅 환경에서 적용할 수 있는 디자인 패턴은 일반적인 디자인 패턴에 비해 동적인 측면을 강조한다. 특히 코바 기반의 시스템은 클라이언트와 서버간 인터액션(interaction)이 중심이므로 시스템 성능이 이 부분의 디자인 패턴에 의해 크게 영향을 받는 경우가 많다.

3. 구성 모델

가. 클래스 다이어그램

- 앞 단락에서 정의된 필요한 기능들을 설계하는 과정에서는 디자인 패턴과 UML을 이용한 모델링이 필수적이다. 특히 클래스 다이어그램(Class Diagram)을 중심으로 하는 모델은 전체 소프트웨어의 아키텍쳐(architecture)를 결정한다는 점에서 더욱 중요하다.

나. 기본 모델

- 본 과제에서는 기본적으로 잘 알려진 디자인 패턴을 적극 사용하였으며, 다음과 같은 기본 모델(basic model)들을 설계 과정에서 정의하였다.

○ 핵심 모델 (Essential Model)

핵심 모델은 기본적인 데이터 관리를 포함하는 주요 아키텍처에 관하여 서술한다. 핵심 모델에서는 MVC(Model-View-Controller) 디자인 패턴과 Composite 디자인 패턴을 주로 사용하였다.

○ 공간 데이터 모델 (Geometry Model)

공간 데이터 모델은 공간 데이터의 처리와 관련된 구조를 정의한다. 공간 데이터를 처리하는 GIS 시스템은 기본 공간 데이터 모델에 큰 영향을 받는다. 따라서, 시스템의 전체 아키텍처는 공간 데이터 모델을 효과적으로 수용할 수 있어야 한다.

○ 속성 데이터 모델 (Attribute Model)

속성 데이터 모델은 일반 MIS 데이터베이스에서 획득할 수 있는 속성 데이터 처리에 관한 구조를 정의한다. 속성 데이터는 공간 데이터와 연계하여 처리될 수 있어야 하므로 공간 객체의 정의 과정에 속성 데이터 모델링 과정이 포함될 수 있다.

○ 지리공간 데이터셀 모델 (GeoDataSet Model)

지리공간 데이터셀 모델은 매핑 라이브러리가 외부 데이터를 획득하는 과정에서 필요한 구조를 정의한다. 개방형 매핑 라이브러리는 기본적으로 OpenGIS 명세를 준수하는 공간 데이터를 포함한 일반 외부 데이터들을 동일한 방식으로 획득할 필요가 있으므로 GeoDataSet 구조를 통해 외부 데이터셀을 추상화한다.

○ 심볼 모델 (Symbol Model)

심볼 모델은 공간 데이터의 디스플레이에 사용되는 심볼의 구조를 정의한다. 동일한 공간 데이터는 그 속성 또는 사용자 정의 심볼 정보에 따라 다른 형태로 디스플레이될 수 있어야 하므로 디스플레이에 있어 심볼은 중요한 위치를 차지한다.

○ 변환 모델 (Transformation Model)

변환 모델은 공간 데이터의 디스플레이 또는 좌표변환(Coordinate Transformation)에 있어 필요한 구조를 정의한다. 공간 데이터를 디스플레이할 때 사용자의 요구에 따라 특정 지역을 확대/축소하는 기능에서부터 변환 모델이 적용될 수 있다. 좌표 변환의 경우에는 실제 데이터를 원하는 투영법에 맞추어 변환할 수 있다.

○ 편집 모델 (Editing Model)

편집 모델은 공간 객체의 편집과 관련된 구조를 정의한다. 공간 객체를 편집하고 그 과정을 디스플레이하기 위해서는 또다른 유형의 그래픽 객체가 정의되어야 한다. 편집을 위한 그래픽 객체는 원래 객체를 레퍼런스(reference)할 수 있으며, 팩토리(Factory) 객체에 의해 만들어질 수 있다.

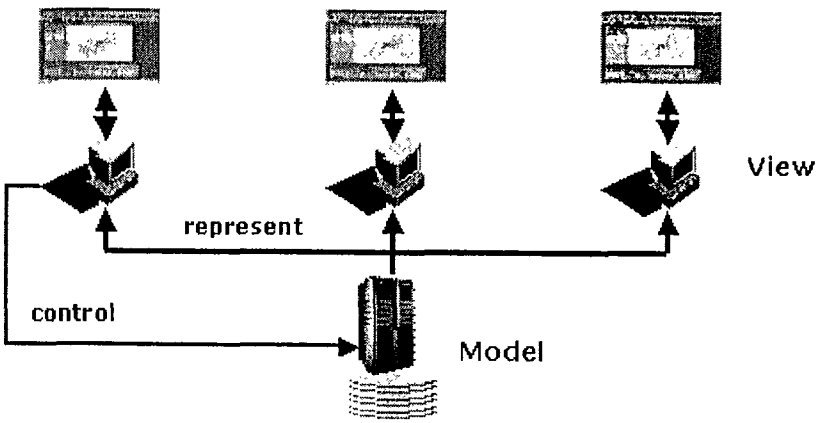
○ 분류 모델 (Classification Model)

분류 모델은 연속 데이터를 일정 구간으로 분류할 수 있도록 하는 구조를 정의하고 있다. 공간 객체들이 가지고 있는 속성을 기반으로 심볼을 정의하는 경우 연속 데이터는 일정 구간을 기준으로 분류하는 기능이 있어야 심볼 설정이 가능하다.

4. 핵심 모델

가. 모델-뷰 아키텍쳐

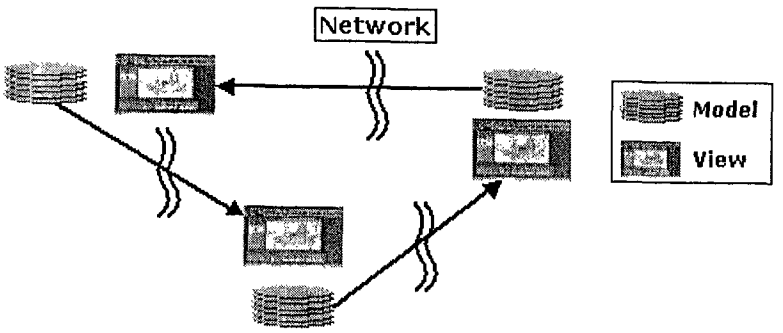
- 공간 및 비공간 데이터의 효율적인 관리를 위해서는 핵심 모델(Essencial Model)의 설계가 중요하다. 본 과제에서는 핵심 모델의 설계를 위하여 모델-뷰-콘트롤러 아키텍쳐(Model-View Architecture)를 사용하였다.
- MVC 아키텍쳐에서는 모델이 데이터 관리 및 조작의 역할을 하며, 뷰는 모델에서 제공하는 정보를 사용자에게 보여주는 역할을 한다. 이러한 아키텍쳐를 사용하는 경우에는 하나의 모델이 여러 개의 뷰를 가질 수 있으며, 하나의 뷰가 서로 다른 모델을 표현할 수도 있다. 콘트롤러는 모델과 뷰 사이에서 역할하거나 뷰로 입력되는 사용자 이벤트를 제어한다.
- [그림 21]은 MVC 아키텍쳐를 구성하고 있는 요소들간의 상호작용을 보여주고 있다. 공간 데이터를 관리하는 모델은 각각의 사용자 요구에 따라 여러 개의 뷰로 표현될 수 있다. 사용자의 이벤트에 따른 제어는 콘트롤러를 통해 가능하다.



[그림 21] MVC 구성요소간 인터액션

- 이와 같은 아키텍쳐는 분산 환경으로 확장됨에 따라 시스템간의 상호작용을 유

연하게 지원할 수 있는 기본 기능으로 부각될 수 있다. 이기종간을 연결하는 시스템은 다른 시스템상에 적재된 모델에 대한 뷰가 될 수 있으며, 동시에 다른 시스템들의 요구를 받아들이는 모델로 역할할 수도 있다. [그림 22]는 분산 환경으로 확장된 경우의 모델-뷰 시스템 아키텍처를 간략하게 보여주고 있다.



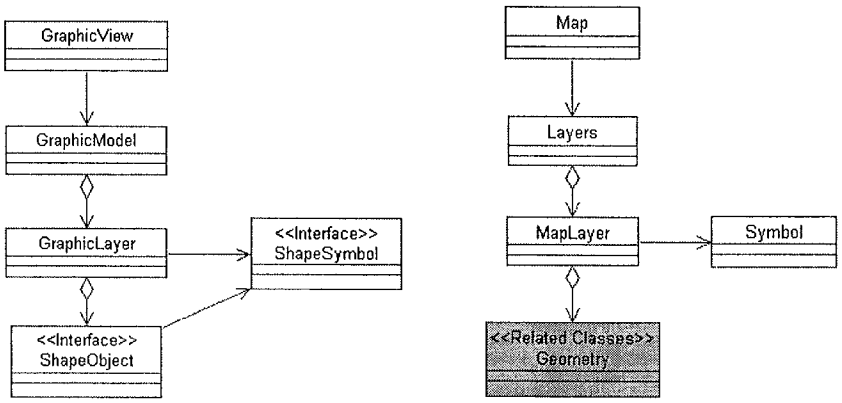
[그림 22] 분산 환경에서의 모델-뷰 아키텍처

나.기본 아키텍쳐

- OpenViews[12]의 핵심 모델 부분에서는 ESRI 사의 MapObjects API를 일정부분 수용하였다. 그 가장 큰 이유는 매핑을 담당하던 기존의 프로그래머들이 MapObjects의 API에 익숙해 있다는 사실에 있으며, 안정적으로 운영되는 기존의 시스템과 기능적 측면에서 벤치마킹(benchmark)이 필요하다는 이유도 있다.
- [그림 23]은 OpenViews의 핵심 클래스들과 MapObjects의 API를 비교한 것이다. OpenViews에서 수용한 MVC 아키텍쳐는 패턴을 쉽게 인지할 수 있도록 클래스 명명을 되도록이면 그 패턴에 맞추도록 하였다. GraphicView는 GraphicModel에서 관리되는 공간 데이터를 디스플레이하는 부분으로 실제 매핑이 이루어지는 클래스이고, 각각의 공간 데이터 오브젝트들은 ShapeObject 클래스의 인스턴스로 표현된다.
- ShapeSymbol은 각각의 ShapeObject 객체에 적합하도록 정의되었으며, ShapeObject와 ShapeSymbol 모두 사용자에 의해 확장 가능하도록 설계되었다. 예를 들어, 서울의 행정구역 경계 레이어를 표현하는 객체들은 PolygonShape로부터 상속을 받아 만들어진 클래스로부터 생성될 수 있으며, PolygonSymbol 또는 그 클래스로부터 상속된 클래스의 인스턴스로 심볼을 등록할 수 있다.
- MapObjects의 경우에 Map 클래스가 여러 개의 레이어를 가질 수 있도록 설계되어 있으며 이는 OpenViews에 똑같이 적용된다. 그러나, 레이어의 집합을 관리

[12] 본 과제에서 설계한 매핑 커널의 이름

하도록 설계된 Layers 클래스는 OpenViews에서 LayerCollection과 LayerIterator를 통해 관리된다. OpenViews에서 레이어 관리를 위해 사용하는 디자인 패턴은 Iterator 디자인 패턴이다.

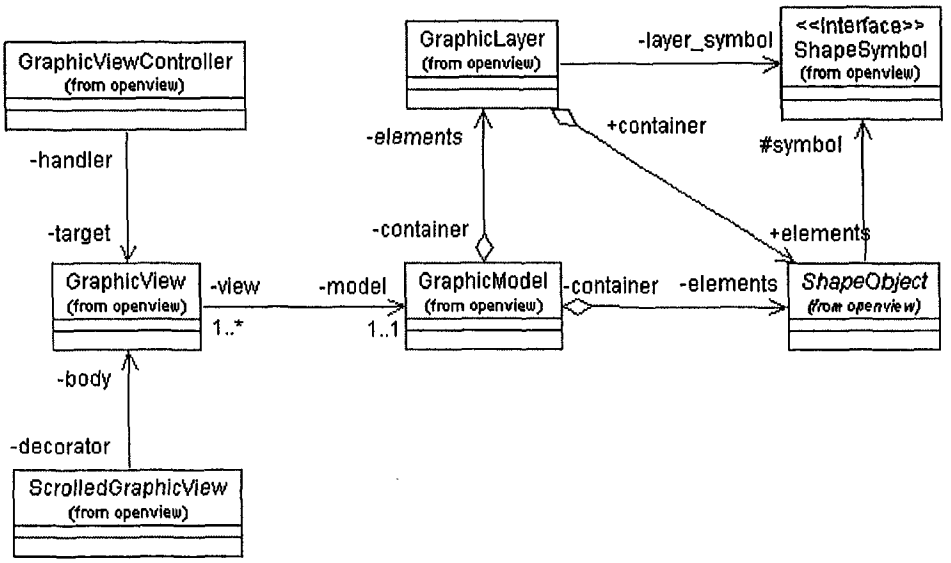


[그림 23] 핵심 모델의 비교 (OpenViews 클래스(왼쪽), MapObjects API(오른쪽))

다. 핵심 모델 다이어그램

- [그림 24]는 GraphicModel과 GraphicView를 중심으로 하는 OpenViews의 핵심 모델 다이어그램이다. 멀티 뷰를 구현하는 그래픽 모델과 그래픽 뷰의 관계는 一對多이며, 하나의 그래픽 모델은 하나 이상의 그래픽 레이어로 구성된다. 그래픽 오브젝트는 ShapeObject 인터페이스로 정의되며, 이 오브젝트는 그래픽 모델이나 그래픽 레이어를 구성하는 기본 요소로 추상 클래스(abstract class)이다.
- 그래픽 뷰는 데코레이터 디자인 패턴(Decorator Pattern)을 사용하여 스크롤이 가능한 ScrollGraphicView로 확장되며, GraphicViewController에 의해 사용자 이벤트가 처리된다. 그래픽 뷰 콘트롤러는 그래픽 뷰와 그래픽 모델을 모두 접근할 수 있으며, 따라서 데이터 관리/조작 및 뷰 제어가 가능하다.
- ShapeObject 클래스는 ShapeSymbol 클래스와 1:1의 관계를 가지며, 그래픽 오브젝트의 랜더링(rendering)을 심볼 인스턴스가 담당한다. 앞에서도 언급된 바와 같이 심볼은 그래픽 레이어와 그래픽 오브젝트 모두에 설정 가능하다. 이 방식은 AutoCAD와 같은 도면편집 프로그램에서 다루는 방식과 유사한 것으로 각각의 그래픽 오브젝트 뿐만 아니라 그 오브젝트의 집합인 레이어에도 속성을 부여한다.
- 그래픽 레이어에 설정된 심볼은 그 레이어에 포함된 모든 그래픽 오브젝트의 심볼을 제어하며, 그래픽 오브젝트 각각에 설정된 심볼은 그 오브젝트만을 담당한다

그래픽 오브젝트에 설정된 심볼은 그래픽 레이어에 설정된 심볼에 우선하여 선택된다. 즉, 그래픽 레이어와 그래픽 오브젝트 모두에 심볼이 설정된 경우, 그래픽 오브젝트의 심볼을 이용하여 랜더링한다.



[그림 24] OpenViews의 핵심 모델 다이어그램

라. 핵심 모델 명세

- 모델-뷰 다이어그램을 구성하는 주요 클래스들의 명세는 다음과 같다.

(1) GraphicViewController

- GraphicViewController는 그래픽 뷰와 연관되어 있는 사용자 이벤트를 처리한다. 자바에서의 일반적인 사용자 이벤트 처리 방식은 delegation으로 UI 오브젝트마다 이벤트 처리 객체를 등록하는 방식을 사용한다. OpenViews도 마찬가지로 View에 대한 이벤트 처리 객체, Object에 대한 이벤트 처리 객체를 등록하여 사용할 수 있도록 하고 있다.

Class name:
GraphicViewController

Category: openview

External Documents:

Export Control: Public

Cardinality: n

Hierarchy:

Superclasses: none

Associations:

target : GraphicView

mouse_listener : java::awt::event::MouseListener

key_listener : java::awt::event::KeyListener

Operations

Name	Signature
GraphicViewController	GraphicViewController ()
addFocusListener	void addFocusListener (FocusListener arg0)
addKeyListener	void addKeyListener (KeyListener arg0)
addMouseListener	void addMouseListener (MouseListener arg0)
addMouseMotionListener	void addMouseMotionListener (MouseMotionListener arg0)
getGraphicModel	GraphicModel getGraphicModel ()
getGraphicView	GraphicView getGraphicView ()
getTransformer	Transform2D getTransformer ()
removeFocusListener	void removeFocusListener (FocusListener arg0)
removeKeyListener	void removeKeyListener (KeyListener arg0)
removeMouseListener	void removeMouseListener (MouseListener arg0)
removeMouseMotionListener	void removeMouseMotionListener (MouseMotionListener arg0)

State machine: No

Concurrency: Sequential

Persistence: Transient

(2) GraphicModel

- GraphicModel은 각종 그래픽 오브젝트를 관리/조작하기 위한 객체이다. 그래픽 모델은 여러 개의 그래픽 레이어를 가지며 이 레이어들은 포함되어 있는 그래픽 객체들을 구분하는 역할을 한다. 결국 GraphicModel은 그래픽 오브젝트를 담고 있는 그릇으로 이들을 조작하기 위한 기본 객체라고 할 수 있다.

Class name:
GraphicModel

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Associations:

elements : ShapeObject
elements : GraphicLayer
view : GraphicView

Operations

Name	Signature
GraphicModel	GraphicModel (int arg0, int arg1, int arg2)
addGeoDataSet	int addGeoDataSet (GeoDataSet arg0)
addLayer	void addLayer (GraphicLayer arg0)
addObject	void addObject (ShapeObject arg0, boolean arg1)
addRecordSet	void addRecordSet (RecordSet arg0)
addSelectionController	void addSelectionController (SelectionListener arg0)

calculateMBR	SimpleRectangle calculateMBR (Transform2D arg0)
copyAllSelection	void copyAllSelection (int arg0, int arg1)
copySelection	void copySelection ()
countLayer	int countLayer ()
countObject	int countObject ()
countObject	int countObject (int arg0)
countRecordSet	int countRecordSet ()
countSelection	int countSelection ()
getAllObject	ShapeIterator getAllObject ()
getAllObject	ShapeIterator getAllObject (int arg0)
getAllRecordSet	ArrayList getAllRecordSet ()
getAllSelection	ShapeIterator getAllSelection ()
getAllView	Enumeration getAllView ()
getCurrentLayer	int getCurrentLayer ()
getLayer	GraphicLayer getLayer (int arg0)
getLayer	GraphicLayer getLayer (String arg0)
getLayerColor	Color getLayerColor (int arg0)
getLayerForShape	GraphicLayer getLayerForShape (ShapeObject arg0)
getLayerIndex	int getLayerIndex (String arg0)
getLayerIndexForShape	int getLayerIndexForShape (ShapeObject arg0)
getLayerName	String getLayerName (int arg0)
getLayerSymbol	ShapeSymbol getLayerSymbol (int arg0)
getLayerType	int getLayerType (int arg0)
getName	String getName ()
getObject	ShapeObject getObject (SimplePoint arg0, int arg1, GraphicView arg2)
getObject	ShapeObject getObject (SimplePoint arg0, GraphicView arg1)

getObject	ShapeObject getObject (String arg0)
getObjectController	ShapeObjectController getObjectController (ShapeObject arg0)
getObjectName	String getObjectName (ShapeObject arg0)
getObjectSymbol	ShapeSymbol getObjectSymbol (ShapeObject arg0)
getObjectSymbolFlag	boolean getObjectSymbolFlag (ShapeObject arg0)
getObjectWithGID	ShapeObject getObjectWithGID (int arg0, int arg1)
getOpenStoreGeometry	Int getOpenStoreGeometry (int arg0, Database arg1, String arg2)
getOpenStoreGeometry	int getOpenStoreGeometry (Database arg0, String arg1)
getRecordSet	RecordSet getRecordSet (int arg0)
getRecordSetForLayer	RecordSet getRecordSetForLayer (int arg0)
getSelectedObjects	ShapeIterator getSelectedObjects ()
getSelectingFlag	boolean getSelectingFlag ()
getSelection	ShapeSelection getSelection (ShapeObject arg0)
getSelection	ShapeSelection getSelection (SimplePoint arg0, GraphicView arg1)
getSelectionFactory	ShapeSelectionFactory getSelectionFactory ()
getSelectionFlag	boolean getSelectionFlag (int arg0)
getSelectionFlag	boolean getSelectionFlag (ShapeObject arg0)
getStreamFactory	GraphicModelStreamFactory getStreamFactory ()
isSelectable	boolean isSelectable (ShapeObject arg0)
isVisible	boolean isVisible (int arg0)
isVisible	boolean isVisible (GraphicView arg0, int arg1)
isVisible	boolean isVisible (ShapeObject arg0)
isVisible	boolean isVisible (ShapeObject arg0, GraphicView arg1)
readShape	int readShape (String arg0, String arg1)
readShape	int readShape (URL arg0)

readShapeDataSet	int readShapeDataSet (int arg0, String arg1, String arg2)
readShapeDataSet	int readShapeDataSet (String arg0, String arg1)
readShapeGeometry	int readShapeGeometry (int arg0, URL arg1)
readShapeGeometry	int readShapeGeometry (DataInputStream arg0)
readShapeGeometry	int readShapeGeometry (URL arg0)
redraw	void redraw ()
redrawAllView	void redrawAllView ()
redrawObject	void redrawObject (ShapeObject arg0)
redrawRegion	void redrawRegion (BoundedRegion arg0)
releaseAll	void releaseAll (int arg0, boolean arg1)
releaseAll	void releaseAll (boolean arg0)
removeAll	void removeAll (int arg0, boolean arg1)
removeAll	void removeAll (boolean arg0)
removeAllSelection	void removeAllSelection (boolean arg0)
removeLayer	void removeLayer (int arg0, boolean arg1)
removeObject	void removeObject (ShapeObject arg0, boolean arg1)
removeRecordSet	void removeRecordSet (int arg0)
removeSelectionListener	void removeSelectionListener (SelectionListener arg0)
selectAll	void selectAll (GraphicView arg0, boolean arg1)
selectAll	Void selectAll (boolean arg0)
setCurrentLayer	Void setCurrentLayer (int arg0)
setEditable	Void setEditable (ShapeObject arg0, boolean arg1)
setGeoDataSet	int setGeoDataSet (int arg0, GeoDataSet arg1)
setLayer	Void setLayer (int arg0)
setLayer	void setLayer (int arg0, GraphicLayer arg1)
setLayer	void setLayer (ShapeObject arg0, int arg1, boolean arg2)
setLayerColor	Void setLayerColor (int arg0, Color arg1)

setLayerColor	Void setLayerColor (int arg0, Color arg1, Color arg2)
setLayerName	void setLayerName (int arg0, String arg1)
setLayerSymbol	void setLayerSymbol (int arg0, ShapeSymbol arg1, boolean arg2)
setLayerType	Void setLayerType (int arg0, int arg1)
setName	void setName (String arg0)
setObjectController	void setObjectController (ShapeObject arg0, ShapeObjectController arg1)
setObjectName	boolean setObjectName (ShapeObject arg0, String arg1)
setObjectSymbol	void setObjectSymbol (ShapeObject arg0, boolean arg1, ShapeSymbol arg2, boolean arg3)
setRecordSetForLayer	void setRecordSetForLayer (int arg0, RecordSet arg1)
setSelectable	void setSelectable (int arg0, boolean arg1)
setSelectable	void setSelectable (ShapeObject arg0, boolean arg1)
setSelected	void setSelected (ShapeObject arg0, boolean arg1, boolean arg2)
setSelectingFlag	void setSelectingFlag (boolean arg0)
setSelectionFactory	void setSelectionFactory (ShapeSelectionFactory arg0)
setVisible	void setVisible (int arg0, boolean arg1, boolean arg2)
setVisible	void setVisible (GraphicView arg0, int arg1, boolean arg2, boolean arg3)
setVisible	void setVisible (ShapeObject arg0, boolean arg1, boolean arg2)
startRedraw	void startRedraw ()
stopRedraw	Void stopRedraw ()
swapLayer	void swapLayer (int arg0, int arg1, boolean arg2)

State machine: No

Concurrency: Sequential

Persistence: Transient

(3) GraphicView

- GraphicView는 그래픽 모델이 관리하는 그래픽 객체들을 사용자에게 보여주기 위해 랜더링하는 역할을 하는 객체이다. 물론 드로잉 관련 기능을 가지고 있는 오브젝트들은 따로 있으므로 뷰는 그 전체 영역이나 헤더 정보를 관리하는 기능을 가지고 있다.

Class name:
GraphicView

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Associations:

handler : GraphicViewController
model : GraphicModel
layer_symbol_container : LayerSymbolContainer
object_symbol_container : ObjectSymbolContainer
decorator : ScrolledGraphicView

Operations

Name	Signature
GraphicView	GraphicView ()
GraphicView	GraphicView (GraphicModel arg0)
GraphicView	GraphicView (GraphicModel arg0, Transform2D arg1)
addNotify	void addNotify ()

addTransformer	void addTransformer (Transform2D arg0)
calculateMBR	SimpleRectangle calculateMBR ()
calculateMBR	SimpleRectangle calculateMBR (Transform2D arg0)
getAntialiasing	boolean getAntialiasing ()
getController	GraphicViewController getController ()
getCopyAreaFlag	boolean getCopyAreaFlag ()
getDoubleBuffering	boolean getDoubleBuffering ()
getGraphicModel	GraphicModel getGraphicModel ()
getGridSpace	GridSpace getGridSpace ()
getLayerSymbol	ShapeSymbol getLayerSymbol (int arg0)
getMaximumSize	Dimension getMaximumSize ()
getMinimumSize	Dimension getMinimumSize ()
getObjectSymbol	ShapeSymbol getObjectSymbol (ShapeObject arg0)
getPreferredSize	Dimension getPreferredSize ()
getRatioFlag	boolean getRatioFlag ()
getRenderedImage	Image getRenderedImage ()
getSymbolContainer	SymbolContainer getSymbolContainer (int arg0)
getTransformer	Transform2D getTransformer ()
getTransparentFlag	boolean getTransparentFlag ()
getVisibleArea	Rectangle getVisibleArea ()
invalidateRect	void invalidateRect (SimpleRectangle arg0)
invalidateView	Void invalidateView ()
isVisible	boolean isVisible (int arg0)
makeVisible	Void makeVisible (SimplePoint arg0)
makeVisible	void makeVisible (SimpleRectangle arg0)
paint	void paint (Graphics arg0)
redrawAllView	void redrawAllView ()

removeAllController	void removeAllController ()
removeController	GraphicViewController removeController ()
removeControllerListener	Void removeControllerListener (ControllerListener arg0)
removeNotify	Void removeNotify ()
removeTransformerListener	Void removeTransformerListener (TransformerListener arg0)
setAntialiasing	Void setAntialiasing (boolean arg0)
setBounds	Void setBounds (int arg0, int arg1, int arg2, int arg3)
setController	Void setController (GraphicViewController arg0)
setController	Void setController (GraphicViewController arg0, AWTEvent arg1)
setCursor	void setCursor (Cursor arg0)
setDoubleBuffering	void setDoubleBuffering (boolean arg0)
setGraphicModel	void setGraphicModel (GraphicModel arg0)
setGridSpace	Void setGridSpace (GridSpace arg0)
setLayerSymbol	Void setLayerSymbol (int arg0, ShapeSymbol arg1, boolean arg2, boolean arg3)
setMaximumSize	Void setMaximumSize (Dimension arg0)
setMinimumSize	Void setMinimumSize (Dimension arg0)
setObjectSymbol	Void setObjectSymbol (ShapeObject arg0, ShapeSymbol arg1, boolean arg2)
setPreferredSize	Void setPreferredSize (Dimension arg0)
setRedrawMode	Void setRedrawMode (int arg0)
setRenderedMode	Void setRenderedMode ()
setTransformer	void setTransformer (Transform2D arg0)
setTransparentFlag	void setTransparentFlag (boolean arg0)
setViewToMBR	void setViewToMBR (SimpleRectangle arg0)
setViewToSize	void setViewToSize ()
setViewToSize	void setViewToSize (Insets arg0)

setVisible	void setVisible (int arg0, boolean arg1)

State machine: No
Concurrency: Sequential
Persistence: Transient

(4) GraphicLayer

- GraphicLayer는 유사한 종류의 그래픽 오브젝트들을 묶어주는 역할을 하는 객체이다. 그래픽 모델과 밀접하게 연관되어 있으며, 해당하는 그래픽 오브젝트들의 심볼 정보를 가지고 있기도 하다. 도로망, 빌딩 등 각종 지리 정보는 하나의 레이어로 생성되어 모델에 등록된 후 뷰에서 표현될 수 있다.

Class name:
GraphicLayer

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: IndexedShapeSet
Associations:

container : GraphicModel
elements : ShapeObject
layer_symbol : ShapeSymbol
obj_array[] : java::lang::Object
att_list : AttributeList
layer_name : java::lang::String
record_set : RecordSet
<no rolename> : RecordSet

Operations

Name	Signature
addAttributeName	void addAttributeName (String arg0)
calculateMBR	SimpleRectangle calculateMBR (Transform2D arg0)
countAttributeName	int countAttributeName ()
getAttribute	Object getAttribute (String arg0)
getAttributeName	String getAttributeName (int arg0)
getGraphicModel	GraphicModel getGraphicModel ()
getIndex	int getIndex ()
getName	String getName ()
getRecordSet	RecordSet getRecordSet ()
getSelectionFlag	boolean getSelectionFlag ()
getSymbol	ShapeSymbol getSymbol ()
getType	int getType ()
hasAttributeName	boolean hasAttributeName (String arg0)
isVisible	boolean isVisible ()
isVisible	boolean isVisible (GraphicView arg0)
removeAll	void removeAll ()
removeAttributeName	void removeAttributeName (int arg0)
removeObject	void removeObject (ShapeObject arg0)
setAttribute	void setAttribute (String arg0, Object arg1)
setAttributeName	void setAttributeName (int arg0, String arg1)
setName	void setName (String arg0)
setRecordSet	void setRecordSet (RecordSet arg0)
setSymbol	void setSymbol (ShapeSymbol arg0)
setSymbol	Void setSymbol (ShapeSymbol arg0, boolean arg1)
setType	void setType (int arg0)
addObject	void addObject (ShapeObject arg0)

addObjectIndex	void addObjectIndex (ShapeObject arg0)
getMBR	SimpleRectangle getMBR (Transform2D arg0)
getObjectCount	int getObjectCount ()
getIterator	ShapeIterator getIterator ()
getObject	ShapeObject getObject (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
getObjectWithGID	ShapeObject getObjectWithGID (int arg0)
removeAll	void removeAll ()
removeObjectIndex	void removeObjectIndex (ShapeObject arg0)
removeObject	void removeObject (ShapeObject arg0)

Attributes:

POINT_TYPE : int = 1
LINESTRING_TYPE : int = 2
POLYGON_TYPE : int = 3
COMPLEX_TYPE : int = 4
LABEL_TYPE : int = 5

State machine: No
Concurrency: Sequential
Persistence: Transient

(5) ShapeObject

- ShapeObject는 모든 그래픽 객체의 최상위 클래스로 그래픽 객체가 가질 수 있는 기본적인 속성을 정의한다. 이 클래스로부터 상속받는 하위 클래스들은 draw() 메소드를 재정의(override)해야 하며, 이 과정에서 다른 기능을 추가할 수 있다. ShapeObject의 정보를 이용하여 랜더링하는 객체는 ShapeSymbol 이다.

Class name:
ShapeObject

Category: openview

External Documents:

Export Control: Public

Cardinality: n

Hierarchy:

Superclasses: none

Associations:

container : GraphicModel

container : GraphicLayer

<no rolename> : GraphicHandler

graphic_bag : ShapeObjectBag

shape_name : java::lang::String

symbol : ShapeSymbol

<no rolename> : ShapeObjectContainer

<no rolename> : ShapeSelection

Operations

Name	Signature
ShapeObject	ShapeObject (ShapeObject arg0)
addActionListener	void addActionListener (ActionListener arg0)
addAttribute	void addAttribute (String arg0, Object arg1)
contains	boolean contains (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
copy	ShapeObject copy ()
createSelection	ShapeSelection createSelection ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getAttribute	Object getAttribute (String arg0)
getMBR	SimpleRectangle getMBR (Transform2D arg0)

getMBRCenter	SimplePoint getMBRCenter (Transform2D arg0)
getName	String getName ()
getShapeObjectContainer	ShapeObjectContainer getShapeObjectContainer ()
getSymbol	ShapeSymbol getSymbol ()
getSymbolFlag	boolean getSymbolFlag ()
isVisible	boolean isVisible ()
processActionEvent	void processActionEvent (ActionEvent arg0)
redraw	void redraw ()
removeActionListener	void removeActionListener (ActionListener arg0)
removeAttribute	boolean removeAttribute (String arg0)
replaceAttribute	boolean replaceAttribute (String arg0, Object arg1)
setAttribute	void setAttribute (String arg0, Object arg1)
setName	void setName (String arg0)
setShapeObjectContainer	void setShapeObjectContainer (ShapeObjectBag arg0)
setSize	void setSize (float arg0, float arg1)
setSymbol	void setSymbol (boolean arg0, ShapeSymbol arg1)
setVisible	void setVisible (boolean arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(6) ShapeSymbol

- ShapeSymbol은 ShapeObject의 심볼을 정의하는 객체로 ShapeObject의 공간 데이터를 이용하여 뷰에 랜더링하는 역할을 한다. 사용자가 ShapeObject를 랜더링하기 위해서는 기본적으로 ShapeSymbol을 오브젝트에 등록해야 한다. ShapeSymbol은 SymbolModel로 구성된다.

Class name:

ShapeSymbol

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Associations:

<no rolename> : GraphicLayer
<no rolename> : ShapeObject

Operations

Name	Signature
addSymbolModel	void addSymbolModel (SymbolModel arg0)
drawSymbol	void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getBackground	Color getBackground ()
getForeground	Color getForeground ()
getLineTextureFlag	boolean getLineTextureFlag ()
getRotate	Double getRotate ()
getStrokeModel	StrokeModel getStrokeModel ()
getTextureModel	TextureModel getTextureModel ()
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)
setRotate	void setRotate (double arg0)

State machine: No

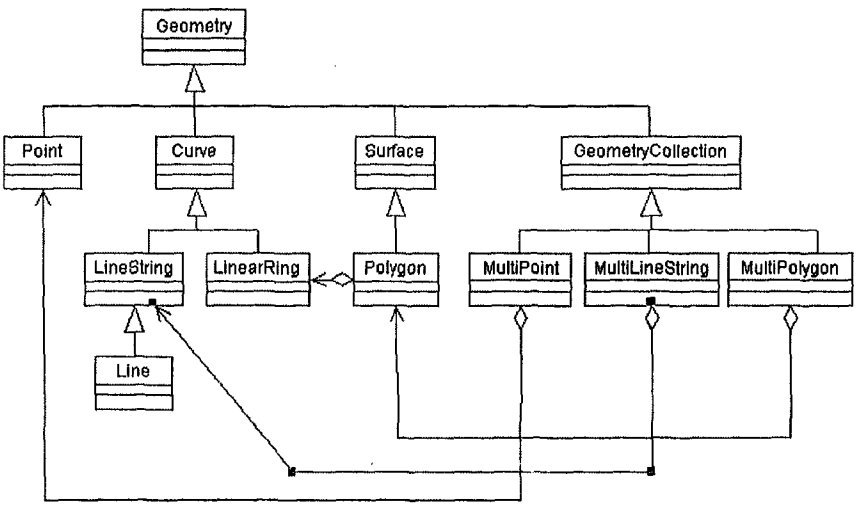
Concurrency: Sequential

Persistence: Transient

5. 공간 데이터 모델

가. 표준 아키텍쳐

- CORBA 기반의 OpenGIS 인터페이스를 통해 획득되는 공간 데이터를 효과적으로 처리하기 위한 공간 데이터 모델은 OpenGIS 콘소시엄에서 채택하고 있는 Geometry모델과 호환되는 것이 중요하다. 표준 공간 데이터 모델을 채택하는 경우, 차후 타 시스템과의 데이터 교환이나 시스템 운용이 편리해질 수 있으며 따라서 설계된 시스템과 다른 시스템과의 통합이 쉬워진다. [그림 25]는 OpenGIS 명세에서 정의한 Geometry모델이다.



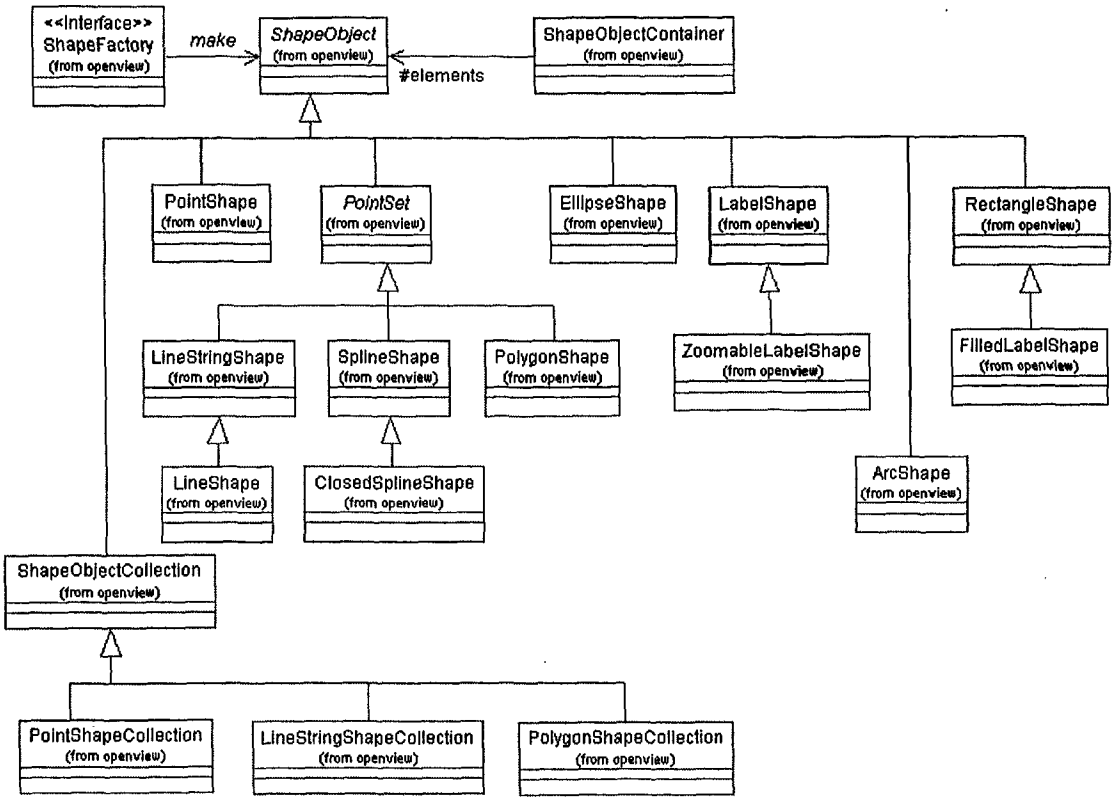
[그림 25] OpenGIS Geometry 모델

- 공간 데이터 모델은 기본적으로 벡터 데이터(점, 선, 면)를 표현할 수 있어야 하며, 이외의 복잡한 기하학적 그래픽으로의 확장이 가능해야 한다. 객체지향 프로그래밍 방식을 사용하는 경우에는 그래픽 오브젝트가 상속 가능하므로 범용 Geometry 클래스를 시스템에서 제공하는 경우에는 각종 애플리케이션을 위한 공간 데이터 오브젝트를 편리하게 만들 수 있다. 예를 들어, 네트워크 분석을 위해 도로망과 빌딩 데이터를 구축하는 경우, 도로는 LineString 시스템 클래스로부터

상속받은 Road 클래스로, 빌딩은 Point 시스템 클래스로부터 상속받은 Building 클래스로 표현될 수 있다. 결과적으로 매핑 라이브러리는 애플리케이션에 종속적이지 않은 범용 공간 데이터 타입과 메소드를 정의할 수 있어야 하며, 이것이 실제 애플리케이션 오브젝트로 쉽게 응용될 수 있는 구조를 가지고 있어야 한다.

나. 공간 데이터 모델 다이어그램

- 공간 데이터 모델을 표현하는 클래스 다이어그램은 [그림 26]과 같다. 공간 데이터는 기본적으로 ShapeObject 인스턴스로 관리/처리되며, 모든 공간 데이터 및 그래픽 오브젝트의 클래스를 정의하려면 이 클래스로부터 상속받아야 한다. 여러 개의 ShapeObject 집합을 표현하는 경우에는 ShapeObjectCollection 클래스를 이용한다.
- ShapeObject의 집합을 표현하려면 Collection을 사용하는데, Collection 관련 클래스로는 ShapeObjectGroup, ShapeObjectSet, ShapeObjectContainer가 있다. ShapeObjectGroup은 오브젝트를 그룹화하는 데 사용되는 오브젝트이며, ShapeObjectSet은 중복이 허용되는 집합, 그리고 ShapeObjectContainer는 중복이 허용되는 오브젝트의 저장소를 의미한다.
- ShapeObject는 모든 그래픽 오브젝트의 최상위 클래스를 정의하며, 대표적인 오브젝트로는 PointShape, LineStringShape, PolygonShape가 있다. SplineShape, ArcShape, EllipseShape, RectangleShape는 그 외의 그래픽 오브젝트들을 표현하기 위한 것이며, 레이블은 특별한 유형의 ShapeObject로 정의하고 처리한다. 오브젝트들의 모음으로 ShapeObjectCollection의 인스턴스를 만들 수 있으며, Collection으로는 Point, LineString, Polygon 유형의 Collection이 존재한다.
- ShapeObject들을 만드는 생성자(constructor)는 다양한 데이터 소스(data source)를 지원한다. 이 데이터 소스는 WKSGeometry, OpenGIS 구현 클래스(ex. PointImpl), 배열(array) 등을 포함한다.



[그림 26] 공간 데이터 모델의 클래스 다이어그램

다. 공간 데이터 모델 명세

(1) PointSet

- PointSet 클래스는 x, y 좌표를 가진 점들의 집합이다. PointShapeCollection 관련 클래스가 PointShape의 집합인데 비해 이 클래스는 단순 점 좌표의 집합을 표현하는 것으로 SimplePoint 형태로 추출될 수 있다.

Class name:
PointSet

Category: openview
External Documents:
Export Control: Public
Cardinality: n

Hierarchy:

Superclasses: ShapeObject

Associations:

points[] : SimplePoint

mbr : SimpleRectangle

mbr_coord[] : SimplePoint

Operations

Name	Signature
PointSet	PointSet (LineString arg0)
PointSet	PointSet (LinearRing arg0)
PointSet	PointSet (LineStringImpl arg0)
PointSet	PointSet (LinearRingImpl arg0)
PointSet	PointSet (PointSet arg0)
PointSet	PointSet (Point[] arg0)
PointSet	PointSet (Point[] arg0, boolean arg1)
PointSet	PointSet (WKSPoint[] arg0)
PointSet	PointSet (WKSPoint[] arg0, boolean arg1)
PointSet	PointSet (PointImpl[] arg0)
PointSet	PointSet (PointImpl[] arg0, boolean arg1)
PointSet	PointSet (SimplePoint[] arg0)
PointSet	PointSet (SimplePoint[] arg0, boolean arg1)
countPoint	int countPoint ()
createSelection	ShapeSelection createSelection ()
getMBR	SimpleRectangle getMBR (Transform2D arg0)
getMBRCenter	SimplePoint getMBRCenter (Transform2D arg0)
getPointAt	SimplePoint getPointAt (int arg0, Transform2D arg1)

insertPoint	void insertPoint (int arg0, float arg1, float arg2, Transform2D arg3)
removePoint	void removePoint (int arg0, Transform2D arg1)

State machine: No
Concurrency: Sequential
Persistence: Transient

(2) PointShape

- PointShape 클래스는 벡터의 점을 표현하는 그래픽 객체이다. 벡터의 기본 요소로 구성되는 SimplePoint와 달리 PointShape는 점 객체를 처리하기 위한 메소드들을 가지고 있다. 다시 말하면, SimplePoint는 primitive 단계에 해당하고 PointShape는 오브젝트 단계에 해당한다.

Class name:
PointShape

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObject
Associations:

point : SimplePoint
fcolor : java::awt::Color

Operations

Name	Signature
PointShape	PointShape (Point arg0, int arg1)
PointShape	PointShape (Point arg0, int arg1, int arg2)

PointShape	PointShape (WKSPoint arg0, int arg1)
PointShape	PointShape (WKSPoint arg0, int arg1, int arg2)
PointShape	PointShape (PointImpl arg0, int arg1)
PointShape	PointShape (PointImpl arg0, int arg1, int arg2)
PointShape	PointShape (PointShape arg0)
PointShape	PointShape (SimplePoint arg0, int arg1)
PointShape	PointShape (SimplePoint arg0, int arg1, int arg2)
copy	ShapeObject copy ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getMBR	SimpleRectangle getMBR (Transform2D arg0)
getMBRCenter	SimplePoint getMBRCenter (Transform2D arg0)
getPoint	SimplePoint getPoint ()
getType	int getType ()
setPoint	void setPoint (WKSPoint arg0)
setPoint	void setPoint (PointImpl arg0)
setPoint	void setPoint (SimplePoint arg0)
setSize	void setSize (int arg0)
setSymbol	void setSymbol (boolean arg0, ShapeSymbol arg1)
setType	void setType (int arg0)

State machine: No

Concurrency: Sequential

Persistence: Transient

(3) LineShape

- LineShape는 두 개의 점으로 구성된 기본 벡터를 표현하는 그래픽 객체이다. LineStringShape 객체와는 달리 두 개의 점으로만 구성되어 있으며, 벡터 처리 관련 메소드들을 가지고 있다.

Class name:

LineShape

Category: openview

External Documents:

Export Control: Public

Cardinality: n

Hierarchy:

Superclasses: LineStringShape

Associations:

start_point : SimplePoint

end_point : SimplePoint

Operations

Name	Signature
LineShape	LineShape (float arg0, float arg1, float arg2, float arg3)
LineShape	LineShape (Point arg0, Point arg1)
LineShape	LineShape (WKSPoint arg0, WKSPoint arg1)
LineShape	LineShape (LineImpl arg0)
LineShape	LineShape (PointImpl arg0, PointImpl arg1)
LineShape	LineShape (LineShape arg0)
LineShape	LineShape (SimplePoint arg0, SimplePoint arg1)
checkAllVisible	boolean checkAllVisible ()
contains	boolean contains (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
copy	ShapeObject copy ()
countPoint	int countPoint ()

createSelection	ShapeSelection createSelection ()
draw	Void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getEndPoint	SimplePoint getEndPoint ()
getStartPoint	SimplePoint getStartPoint ()
getMBR	SimpleRectangle getMBR (Transform2D arg0)
getMBRCenter	SimplePoint getMBRCenter (Transform2D arg0)
getPointAddFlag	boolean getPointAddFlag ()
getPointAt	SimplePoint getPointAt (int arg0, Transform2D arg1)
insertPoint	void insertPoint (int arg0, float arg1, float arg2, Transform2D arg3)
removePoint	void removePoint (int arg0, Transform2D arg1)
setEndPoint	void setEndPoint (SimplePoint arg0)
setStartPoint	void setStartPoint (SimplePoint arg0)
setSymbol	void setSymbol (boolean arg0, ShapeSymbol arg1)

State machine: No
Concurrency: Sequential
Persistence: Transient

(4) LineStringShape

- LineStringShape는 여러 개의 점으로 구성된 선형 객체이다. 내부적으로는 점을 표현하는 좌표의 배열로 표현되며, 각 세그먼트를 구성하는 벡터를 통해 각종 공간 연산자를 구현한다. 선 관련 그래픽 처리에 필요한 메소드들은 이 객체에서 담당한다. PointSet 클래스로부터 상속받는다.

Class name:
LineStringShape

Category: openview
External Documents:

Export Control: Public

Cardinality: n

Hierarchy:

Superclasses: PointSet

Operations

Name	Signature
PolylineShape	PolylineShape (LineString arg0)
PolylineShape	PolylineShape (LineStringImpl arg0)
PolylineShape	PolylineShape (PolylineShape arg0)
PolylineShape	PolylineShape (Point[] arg0)
PolylineShape	PolylineShape (Point[] arg0, boolean arg1)
PolylineShape	PolylineShape (WKSPoint[] arg0)
PolylineShape	PolylineShape (WKSPoint[] arg0, boolean arg1)
PolylineShape	PolylineShape (PointImpl[] arg0)
PolylineShape	PolylineShape (PointImpl[] arg0, boolean arg1)
PolylineShape	PolylineShape (SimplePoint[] arg0)
PolylineShape	PolylineShape (SimplePoint[] arg0, boolean arg1)
contains	boolean contains (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
copy	ShapeObject copy ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getMBR	SimpleRectangle getMBR (Transform2D arg0)
setSymbol	void setSymbol (boolean arg0, ShapeSymbol arg1)

State machine: No

Concurrency: Sequential

Persistence: Transient

(5) LabelShape

- LabelShape는 텍스트나 그림과 같은 레이블을 표현하는 그래픽 객체이다. 일반적인 개념의 레이블은 텍스트와 그것을 표현하기 위한 폰트 등을 위한 것이지만 이 그래픽 객체는 이미지와 같은 확장된 유형의 레이블을 지원한다.

Class name:
LabelShape

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObject
Associations:

fcolor : java::awt::Color
label_string : java::lang::String
label_font : java::awt::Font
center_point : SimplePoint

Operations

Name	Signature
LabelShape	LabelShape (LabelShape arg0)
LabelShape	LabelShape (SimplePoint arg0, String arg1)
copy	ShapeObject copy ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getCenter	SimplePoint getCenter ()
getFont	Font getFont ()

getLabel	String getLabel ()
getLabelMBR	SimpleRectangle getLabelMBR (Transform2D arg0)
getMBR	SimpleRectangle getMBR (Transform2D arg0)
getMBRCenter	SimplePoint getMBRCenter (Transform2D arg0)
getMultilineFlag	boolean getMultilineFlag ()
setCenter	void setCenter (SimplePoint arg0)
setFont	void setFont (Font arg0)
setLabel	Void setLabel (String arg0)
setSymbol	void setSymbol (boolean arg0, ShapeSymbol arg1)

State machine: No
Concurrency: Sequential
Persistence: Transient

(6) ArcShape

- ArcShape는 원의 일부로 표현되는 아크 그래픽 객체이다. 중심점, 원의 반지름 시작 각도, 끝 각도로 구성되는 데이터는 각종 메소드에 의해 처리된다.

Class name:
ArcShape

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObject
Associations:

mbr : SimpleRectangle
fcolor : java::awt::Color

Operations

Name	Signature
ArcShape	ArcShape (SimpleRectangle arg0, float arg1, float arg2)
contains	boolean contains (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
copy	ShapeObject copy ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getDeltaAngle	float getDeltaAngle ()
getMBR	SimpleRectangle getMBR (Transform2D arg0)
getMBRCenter	SimplePoint getMBRCenter (Transform2D arg0)
getStartAngle	float getStartAngle ()
setDeltaAngle	void setDeltaAngle (float arg0)
setStartAngle	void setStartAngle (float arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(7) EllipseShape

- EllipseShape는 타원을 표현하는 그래픽 객체이다. 타원은 세로축과 가로축을 통해 표현되며, 중심점과 관련된 데이터를 중심으로 그려질 수 있다. 원형 그래픽 객체(Circle)은 이 객체를 이용하여 표현되며, 그 관계는 LineStringShape와 LineShape간의 관계와 같다.

Class name:
EllipseShape

Category: openview
External Documents:

Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObject
Associations:

fcolor : java::awt::Color
drawrect : SimpleRectangle

Operations

Name	Signature
EllipseShape	EllipseShape (EllipseShape arg0)
EllipseShape	EllipseShape (SimplePoint arg0, int arg1)
EllipseShape	EllipseShape (SimpleRectangle arg0)
contains	boolean contains (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
copy	ShapeObject copy ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getMBR	SimpleRectangle getMBR (Transform2D arg0)
getMBRCenter	SimplePoint getMBRCenter (Transform2D arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(8) PolygonShape

- PolygonShape는 여러 개의 점으로 구성된 폴리곤 객체를 표현한다. 실제 여러 개의 폴리곤이 서로 복잡하게 얽혀있는 것은 여러 개의 링으로 구성해야 하지만 이 그래픽 객체는 기본적인 그래픽 객체의 기능만을 가지고 있다. 즉, 폴리곤 처리 관련 메소드만을 가지고 있다.

Category: openview

External Documents:

Export Control: Public

Cardinality: n

Hierarchy:

Superclasses: PointSet

Operations

Name	Signature
PolygonShape	PolygonShape (LinearRing arg0)
PolygonShape	PolygonShape (LinearRingImpl arg0)
PolygonShape	PolygonShape (PolygonShape arg0)
PolygonShape	PolygonShape (Point[] arg0)
PolygonShape	PolygonShape (Point[] arg0, boolean arg1)
PolygonShape	PolygonShape (WKSPoint[] arg0)
PolygonShape	PolygonShape (WKSPoint[] arg0, boolean arg1)
PolygonShape	PolygonShape (PointImpl[] arg0)
PolygonShape	PolygonShape (PointImpl[] arg0, boolean arg1)
PolygonShape	PolygonShape (SimplePoint[] arg0)
PolygonShape	PolygonShape (SimplePoint[] arg0, boolean arg1)
contains	boolean contains (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
copy	ShapeObject copy ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
setSymbol	setSymbol ()

State machine: No

Concurrency: Sequential

Persistence: Transient

(9) RectangleShape

- RectangleShape는 직사각형 형태의 그래픽 객체로 기본적인 그래픽 처리가 최소점과 최대점으로 구성된 직사각형을 중심으로 이루어지는 경우에 사용된다.

Class name:
RectangleShape

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObject
Associations:

drawrect : SimpleRectangle

Operations

Name	Signature
RectangleShape	RectangleShape (RectangleShape arg0)
RectangleShape	RectangleShape (SimpleRectangle arg0)
contains	boolean contains (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
copy	ShapeObject copy ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getMBR	SimpleRectangle getMBR (Transform2D arg0)
getMBRCenter	SimplePoint getMBRCenter (Transform2D arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(10) ShapeObjectContainer

- ShapeObjectContainer는 여러 개의 ShapeObject를 모아두는 그릇 역할을 한다. 담겨진 ShapeObject들은 중복이 허용되며 하나의 Container안에 들어 있는 오브젝트들은 일련의 기능들을 위한 target이 되는 경우가 많다.

Class name:
ShapeObjectContainer

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : ShapeObjectSet
<no rolename> : ShapeIteratorImpl
elements : ShapeObject

Operations

Name	Signature
ShapeObjectContainer	ShapeObjectContainer ()
ShapeObjectContainer	ShapeObjectContainer (int arg0)
ShapeObjectContainer	ShapeObjectContainer (int arg0, int arg1)
addElement	void addElement (ShapeObject arg0)
capacity	Int capacity ()
clone	Object clone ()

contains	boolean contains (ShapeObject arg0)
copyInto	void copyInto (ShapeObject[] arg0)
count	int count ()
elementAt	ShapeObject elementAt (int arg0)
elements	ShapeIterator elements ()
ensureCapacity	void ensureCapacity (int arg0)
firstElement	ShapeObject firstElement ()
indexOf	int indexOf (ShapeObject arg0)
indexOf	int indexOf (ShapeObject arg0, int arg1)
insertElementAt	void insertElementAt (ShapeObject arg0, int arg1)
isEmpty	boolean isEmpty ()
iterator	ShapeIterator iterator ()
lastElement	ShapeObject lastElement ()
lastIndexOf	int lastIndexOf (ShapeObject arg0)
lastIndexOf	int lastIndexOf (ShapeObject arg0, int arg1)
removeAll	void removeAll ()
removeAllElements	void removeAllElements ()
removeElement	boolean removeElement (ShapeObject arg0)
removeElementAt	void removeElementAt (int arg0)
setElementAt	Void setElementAt (ShapeObject arg0, int arg1)
setSize	void setSize (int arg0)
size	int size ()
toString	String toString ()
trimToSize	void trimToSize ()

State machine: No

Concurrency: Sequential

Persistence: Transient

(11) SplineShape

- SplineShape는 Spline 곡선을 표현하는 그래픽 객체이다.

Class name:
SplineShape

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: PointSet
Associations:

fcolor : java::awt::Color

Operations

Name	Signature
SplineShape	SplineShape (SimplePoint[] arg0, boolean arg1)
contains	boolean contains (SimplePoint arg0, SimplePoint arg1, Transform2D arg2)
copy	ShapeObject copy ()
createSelection	ShapeSelection createSelection ()
draw	void draw (Graphics arg0, Transform2D arg1, SymbolHandler arg2)
getEndDecoration	int getEndDecoration ()
getMBR	SimpleRectangle getMBR (Transform2D arg0)
setEndDecoration	void setEndDecoration (int arg0)

State machine: No
Concurrency: Sequential

Persistence: Transient

(12) ShapeFactory

- ShapeObject를 생성하는 과정에서는 Factory 디자인 패턴을 사용한다. ShapeFactory는 인터페이스로 이 인터페이스를 구현하는 각각의 Factory들은 사용자의 요구에 따라 적당한 ShapeObject를 생성하는 역할을 한다. 이 과정에서 Factory 객체가 객체의 생성과 관련된 여러 가지 전처리(pre-processing), 후처리(post-processing)를 할 수 있다.

Class name:
ShapeFactory

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Associations:

<no rolename> : ShapeObject in association make

State machine: No
Concurrency: Sequential
Persistence: Transient

(13) ShapeObjectCollection

- ShapeObjectCollection은 ShapeObject의 집합을 표현한다. 이 클래스는 OpenGIS 명세에서 말하는 GeometryCollection의 기능과 동일하다.

Class name:
ShapeObjectCollection

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObject
State machine: No
Concurrency: Sequential
Persistence: Transient

(14) PointShapeCollection

- PointShapeCollection은 PointShape를 담고 있는 그룻으로 ShapeObjectCollection으로부터 상속받는다.

Class name:
PointShapeCollection

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObjectCollection
State machine: No
Concurrency: Sequential
Persistence: Transient

(15) LineStringShapeCollection

- LineStringShapeCollection은 LineStringShape를 담고 있는 그룻으로 ShapeObjectCollection으로부터 상속받는다.

Class name:
LineStringShapeCollection

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObjectCollection
State machine: No
Concurrency: Sequential
Persistence: Transient

(16) PolygonShapeCollection

- PolygonShapeCollection은 PolygonShape를 담고 있는 그룻으로 ShapeObjectCollection으로부터 상속받는다.

Class name:
PolygonShapeCollection

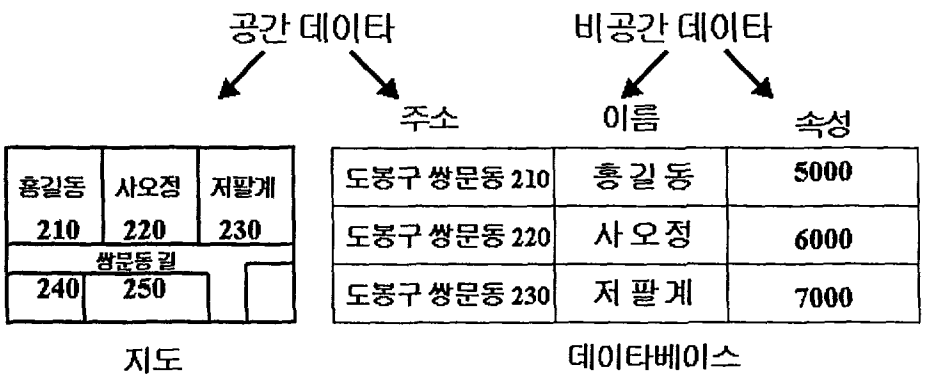
Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObjectCollection
State machine: No
Concurrency: Sequential
Persistence: Transient

6. 속성 데이터 모델

가. 관계형 모델

- 일반 MIS 데이터베이스에서 획득하는 데이터나 GIS 시스템 내부에서 처리되는 속성 데이터는 공간 데이터와 유기적인 관련을 맺고 있다. 예를 들어, 빌딩을 표

현하는 공간 객체가 가지고 있는 속성들은 그 빌딩의 공간 객체 내에 표현되거나 외부 링크를 위한 고유 키를 사용함으로써 공간 객체들을 통해 검색될 수 있다. [그림 27]은 속성 데이터의 관계형 표현과 함께 공간 데이터와 속성 데이터의 연계를 표현한 것이다.

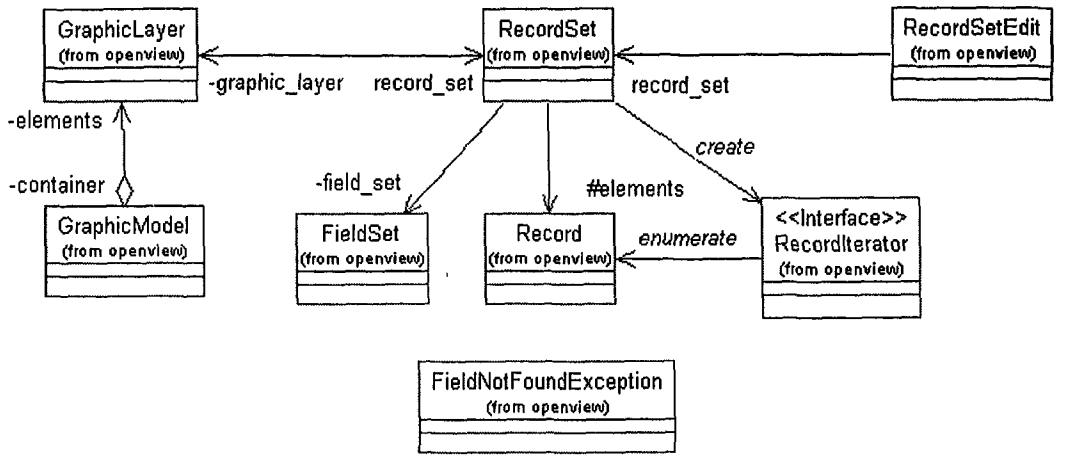


[그림 27] 공간 데이터와 속성 데이터의 연계

- 결국, 속성 데이터는 공간 데이터와 연계되어 처리될 수 있어야 하므로 속성 데이터를 효율적으로 관리하는 메모리 상의 데이터 모델과 함께 공간 데이터와의 연계 모델을 정의해야 한다. 본 과제에서는 관계형 데이터 모델을 응용하여 속성 데이터를 처리할 수 있는 모델을 작성하였다.

나. 속성 데이터 모델 다이어그램

- 속성 데이터의 관리 및 처리를 위한 모델 다이어그램은 [그림 28]과 같다.
- 속성 데이터는 RecordSet 이라는 객체에 의해 일관되게 관리된다. 이것은 ShapeFile 데이터를 읽었을 경우 *.dbf 파일에 포함된 내용과 같으며, 데이터베이스를 읽었을 경우에는 하나의 테이블로 비교될 수 있다. 하나의 RecordSet 객체는 기본적으로 하나의 GraphicLayer에 등록되며, 따라서 객체의 속성은 오브젝트별로 관리되거나, RecordSet에 의해 레이어별로 관리될 수 있다.



[그림 28] 속성 데이터 모델 다이어그램

- RecordSet 객체는 헤더 정보(필드 정의)를 가지고 있는 FieldSet 객체와 각각의 투플 정보를 가지고 있는 Record로 구성된다. Iterator 패턴을 사용하는 RecordSet은 각각의 Record 아이템을 접근하기 위해 RecordIterator를 사용한다. RecordSet을 편집하거나 이에 편리하게 접근하는 위해서는 RecordSetEdit 객체를 사용하며 실제 ShapeObject와 이 데이터 모델에 들어있는 속성과의 링크는 GraphicView 안의 메소드를 사용하는 방식과 RecordSetEdit를 통한 방식 두 가지가 있다.

다. 속성 데이터 모델 명세

- 속성 데이터의 관리 및 처리를 위한 모델 명세는 다음과 같다.

(1) FieldNotFoundException

- 이 클래스는 사용자가 요청한 필드가 발견되지 않았을 때 발생하는 예외 클래스이다.

Class name:
FieldNotFoundException

Category: openview

External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Operations:

Name	Signature
FieldNotFoundException	FieldNotFoundException (String arg0, int arg1)
getMessage	String getMessage ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(2) RecordSetEdit

- 이 클래스는 RecordSet에 포함된 데이터들에 접근하기 위한 클래스이다. Delegation 디자인 패턴을 사용하였으며, RecordSetEdit를 통해 RecordSet 안에 있는 데이터를 조작할 수 있다. RecordSet은 그래픽 모델에 등록되어 일반 MIS 테이블로 기능할 수도 있고, GraphicLayer에 등록되어 공간 데이터와의 링크를 가질 수도 있으므로 이 클래스에는 RecordSet 객체가 어떤 종류인가를 결정하는 메소드와 함께 레이어 관련 속성 테이블일 경우 등록된 레이어를 알아볼 수 있도록 지원한다.

Class name:
RecordSetEdit

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Associations:

record_set : RecordSet
<no rolename> : RecordSetTableModel

Operations:

Name	Signature
RecordSetEdit	RecordSetEdit ()
RecordSetEdit	RecordSetEdit (RecordSet arg0)
checkType	boolean checkType (Object arg0, Column arg1)
getColumnIndex	int getColumnIndex (String arg0)
getItem	Object getItem (int arg0, int arg1)
getItem	Object getItem (int arg0, String arg1)
getRecordSet	RecordSet getRecordSet ()
hasRecordSet	boolean hasRecordSet ()
isLayer	boolean isLayer ()
isMIS	boolean isMIS ()
searchRecord	Record searchRecord (int arg0, Object arg1)
searchRecord	Record searchRecord (String arg0, Object arg1)
searchRecordIndex	int searchRecordIndex (int arg0, Object arg1)
searchRecordIndex	int searchRecordIndex (String arg0, Object arg1)
setItem	void setItem (int arg0, int arg1, Object arg2)
setItem	void setItem (int arg0, String arg1, Object arg2)

State machine: No
Concurrency: Sequential
Persistence: Transient

(3) Record

- 이 클래스는 하나의 투플을 표현한다. 투플을 구성하는 아이템들은 FieldSet에서 정의된 방식대로 채워져야 하며, 사용자의 요구에 따라 관련 객체들을 리턴한다. 각각의 Record 오브젝트에 접근하기 위하여 RecordIterator가 존재한다. Record 객체 안에 들어있는 각각의 아이템들은 자바 객체로 구성되어 있으므로 자바의 List 데이터 구조를 사용할 수 있다.

Class name:
Record

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: java::util::LinkedList
Associations:

<no rolename> : RecordIterator
<no rolename> : RecordSet

Operations:

Name	Signature
Record	Record ()
getItemCount	int getItemCount ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(4) RecordIterator

- 이 클래스는 RecordSet 안에 포함된 각각의 레코드를 접근하기 위한 방법을

제공한다. Iterator 디자인 패턴이므로 인덱스를 이용해야 할 필요가 있는 경우에는 RecordSetEdit 클래스를 사용한다.

Class name:
RecordIterator

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Associations:

<no rolename> : RecordSet in association create
<no rolename> : Record in association enumerate

Operations:

Name	Signature
hasNext	Boolean hasNext ()
next	Record next ()
remove	void remove ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(5) FieldSet

- 이 클래스는 RecordSet에 들어가는 아이템들에 관한 정의 방식을 제공한다. 하나의 Field는 type과 name으로 구성되는데 일반 dbf 파일에서 요구하는 type, name, length 정보 등을 담을 수 있다.

Class name:
FieldSet

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: java::util::LinkedList
Associations:

<no rolename> : RecordSet

Operations:

Name	Signature
FieldSet	FieldSet ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(6) RecordSet

- 이 클래스는 레코드의 집합을 표현하며, 일반 데이터베이스에서 사용되는 테이블의 개념과 비교될 수 있다.

Class name:
RecordSet

Category: openview
External Documents:
Export Control: Public
Cardinality: n

Hierarchy:

Superclasses: none

Associations:

<no rolename> : RecordIterator in association create
<no rolename> : Statistics in association summarize
<no rolename> : Classification in association classify
<no rolename> : GraphicLayer
elements : Record
field_set : FieldSet
graphic_layer : GraphicLayer
<no rolename> : RecordSetEdit
<no rolename> : ShapeReader

Operations:

Name	Signature
RecordSet	RecordSet ()
RecordSet	RecordSet (String arg0)
RecordSet	RecordSet (String arg0, GraphicLayer arg1)
add	void add (int arg0, Record arg1)
add	boolean add (Record arg0)
addField	void addField (Column arg0)
addField	void addField (String arg0, int arg1)
clear	void clear ()
contains	boolean contains (Record arg0)
countField	int countField ()
countRecord	int countRecord ()
create_iterator	RecordIterator create_iterator ()
ensureCapacity	void ensureCapacity (int arg0)
get	Record get (int arg0)

getField	Column getField (int arg0)
getField	Column getField (String arg0)
getFieldIterator	ListIterator getFieldIterator ()
getFieldSet	FieldSet getFieldSet ()
getGraphicLayer	GraphicLayer getGraphicLayer ()
indexOf	int indexOf (Record arg0)
isEmpty	boolean isEmpty ()
isLayer	boolean isLayer ()
isMIS	boolean isMIS ()
isPersistent	boolean isPersistent ()
lastIndexOf	int lastIndexOf (Record arg0)
remove	Record remove (int arg0)
set	Record set (int arg0, Record arg1)
setFieldSet	void setFieldSet (FieldSet arg0)
setGraphicLayer	void setGraphicLayer (GraphicLayer arg0)
setPersistent	void setPersistent (boolean arg0)
size	int size ()
toArray	Record[] toArray ()
trimToSize	void trimToSize ()

State machine: No
Concurrency: Sequential
Persistence: Transient

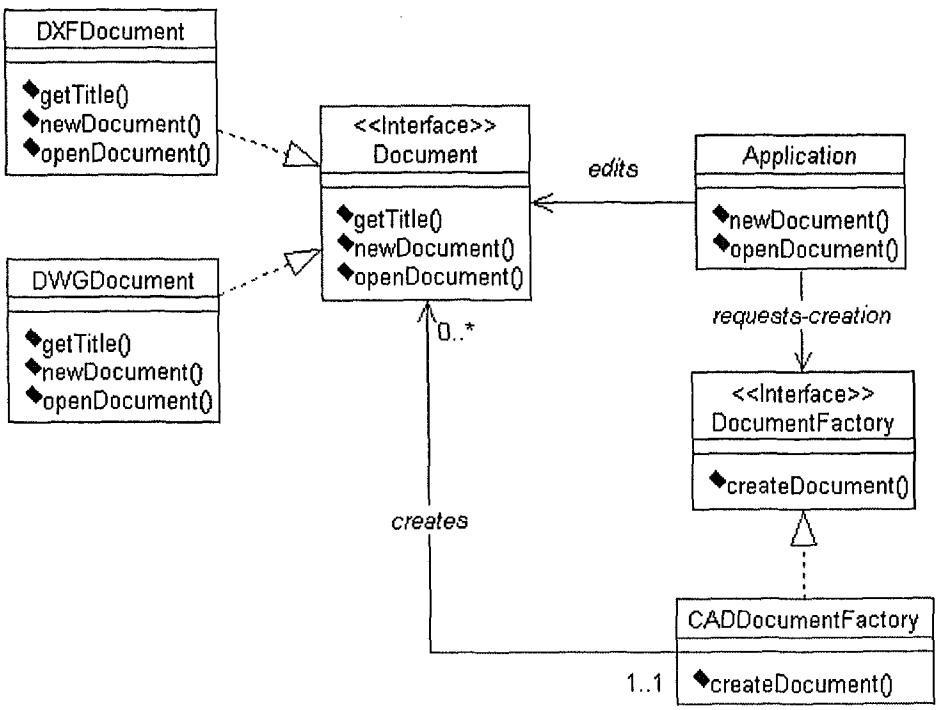
7. 지리공간 데이터셋 모델

가. 데이터의 위치 투명성

- 매핑 라이브러리가 외부 데이터를 획득하는 과정을 추상화하는 지리공간 데이터

셀 모델은 기본적으로 데이터의 위치 투명성(Location Transparency)을 보장할 수 있어야 한다.

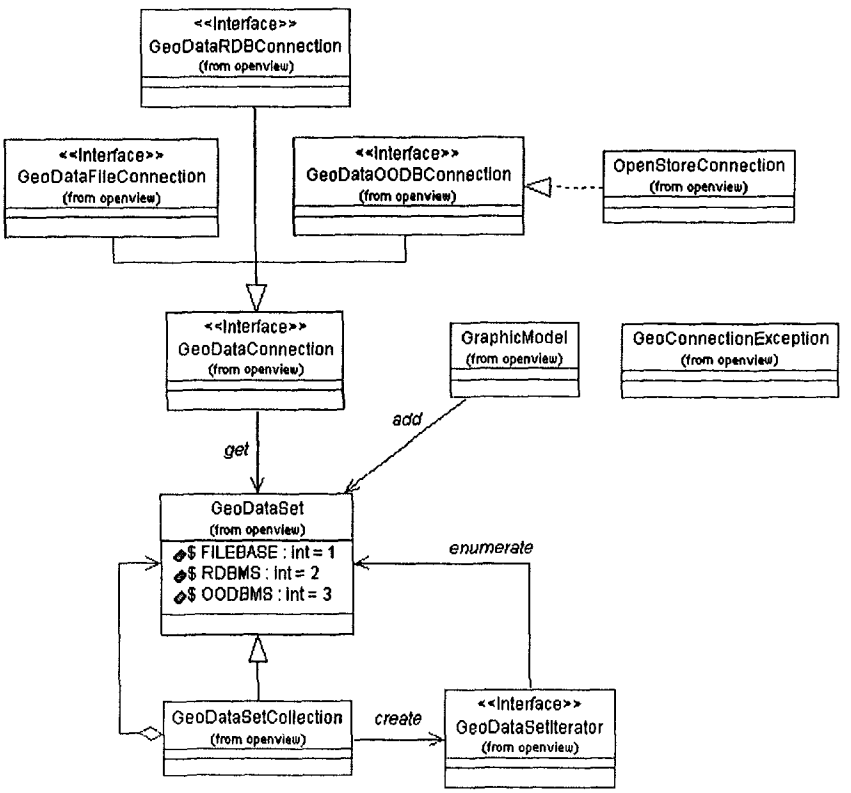
- 데이터의 위치 투명성은 상호운용성에 있어서 가장 기본적인 사항의 하나이다. 따라서, 데이터의 위치 투명성을 보장하기 위해서는 CORBA 인터페이스를 이용한 원격지 데이터 접근이 기본적으로 지원되어야 한다. 이러한 기능을 지원하기 위한 매핑 라이브러리의 구조는 확장 가능해야 하며 매핑 라이브러리는 데이터 로딩 모듈과 독립적이어야 한다.
- 애플리케이션과 데이터 로딩 모듈과의 독립성을 보장하기 위해 Factory, Builder 와 같은 잘 알려진 객체 생성 패턴(creational pattern)을 사용할 수 있다. [그림 29]는 DXF 파일을 새로 만들거나 열기 위해 사용한 Factory 디자인 패턴을 표현한 것이다.



[그림 29] DXF IO를 위한 Factory 디자인 패턴

나. 지리공간 데이터셀 모델 다이어그램

- 지리공간 데이터셋 모델 다이어그램은 [그림 30]과 같다. 지리공간 데이터셋 모델의 핵심은 GeoDataSet이다. 이 클래스는 일반적인 GeoDataSet의 공통 변수들을 수렴한 데이터 모델로 GeoDataConnection 클래스에 의해 로컬, 리모트 사이트에 있는 공간 데이터셋에 일관되게 접근한다.
- 기본적으로 GeoDataSet은 여러 개가 있을 수 있으므로 Collection과 Iterator 디자인 패턴으로 표현될 수 있다. GeoDataSet은 FileBase, RDBMS, OODBMS에 상관없이 일관된 방식을 취해야 하지만 각 벤더들이 만들어 낸 공간 데이터베이스는 나름의 특성을 가지고 있을 수 있으므로 확장 가능한 데이터 모델이 필요하다. 예를 들어, SDE 와 Oracle Spatial은 OpenGIS에서 요구하는 API를 만족하지만 두 공간 데이터베이스를 일관된 방식으로 접근한다고 해서 각 공간 데이터베이스에 최적화된 상태라고 할 수는 없다. 즉, 일관된 접근 방식과 함께 특정 공간 데이터베이스 또는 파일에 최적화시키는 방법을 만들어 놓아야 한다.



[그림 30] 지리공간 데이터셋 모델 다이어그램

다. 지리공간 데이터셋 모델 명세

• 지리공간 데이터셋의 모델 명세는 다음과 같다.

(1) GeoConnectionException

• 이 클래스는 GeoDataConnection으로 외부 데이터셋을 연결하거나 연결된 상태에서 조작을 할 때 발생할 수 있는 예외를 표현한다.

Class name:
GeoConnectionException

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: java::lang::Exception

Operations:

Name	Signature
GeoConnectionException	GeoConnectionException ()
GeoConnectionException	GeoConnectionException (String arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(2) GeoDataFileConnection

• 이 클래스는 FileBase 기반의 데이터셋을 로딩하기 위한 메커니즘을 제공한다. 본 과제에서 설계한 매핑 라이브러리는 기본 데이터 파일로 ShapeFile을 사용하며, 기본적인 데이터 타입은 *.shp 와 *.dbf를 사용한다.

Class name:

GeoDataFileConnection

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: GeoDataConnection

Operations:

Name	Signature
getBase	String getBase ()
getURL	URL getURL ()
setBase	Void setBase (String arg0)
setURL	void setURL (URL arg0)
connect	void connect ()
disconnect	void disconnect ()
getGeoDataSet	GeoDataSet getGeoDataSet (String arg0)
getGeoDataSetCollection	GeoDataSetCollection getGeoDataSetCollection ()
isConnected	boolean isConnected ()
list	String[] list ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(3) GeoDataOODBConnection

• 이 클래스는 OODBMS에 들어가 있는 데이터셀에 접근하기 위한 메커니즘을 제공한다.

Class name:
GeoDataOODBConnection

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: GeoDataConnection

Operations:

Name	Signature
getURL	String getURL ()
setPassword	void setPassword (String arg0)
setServer	void setServer (String arg0)
setURL	Void setURL (String arg0)
setUser	void setUser (String arg0)
connect	void connect ()
disconnect	void disconnect ()
getGeoDataSet	GeoDataSet getGeoDataSet (String arg0)
getGeoDataSetCollection	GeoDataSetCollection getGeoDataSetCollection ()
isConnected	Boolean isConnected ()
list	String[] list ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(4) GeoDataRDBConnection

- 이 클래스는 RDBMS에 저장된 데이터셀에 접근하기 위한 메커니즘을 제공한다

Class name:
GeoDataRDBConnection

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: GeoDataConnection

Operations:

Name	Signature
getURL	String getURL ()
setPassword	void setPassword (String arg0)
setServer	void setServer (String arg0)
setURL	void setURL (String arg0)
setUser	void setUser (String arg0)
connect	void connect ()
disconnect	void disconnect ()
getGeoDataSet	GeoDataSet getGeoDataSet (String arg0)
getGeoDataSetCollection	GeoDataSetCollection getGeoDataSetCollection ()
isConnected	boolean isConnected ()
list	String[] list ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(5) GeoDataSetCollection

• 이 클래스는 GeoDataSet의 집합을 표현한다. 각각의 GeoDataSet은 GeoDataSetIterator를 통해 획득될 수 있으며, list() 메소드에 의해 각 GeoDataSet의 이름을 알아낼 수 있다. GeoDataSet과 GeoDataSetCollection은 composite 디자인 패턴을 사용한다.

Class name:
GeoDataSetCollection

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: GeoDataSet
Associations:

<no rolename> : GeoDataSet
<no rolename> : GeoDataSetIterator
<no rolename> : LocalIterator
elements : GeoDataSet

Operations:

Name	Signature
GeoDataSetCollection	GeoDataSetCollection ()
GeoDataSetCollection	GeoDataSetCollection (String arg0)
add	void add (int arg0, GeoDataSet arg1)
add	boolean add (GeoDataSet arg0)
clear	void clear ()
contains	boolean contains (GeoDataSet arg0)

create_iterator	GeoDataSetIterator create_iterator ()
ensureCapacity	void ensureCapacity (int arg0)
get	GeoDataSet get (int arg0)
indexOf	int indexOf (GeoDataSet arg0)
isEmpty	boolean isEmpty ()
lastIndexOf	int lastIndexOf (GeoDataSet arg0)
remove	GeoDataSet remove (int arg0)
set	GeoDataSet set (int arg0, GeoDataSet arg1)
size	int size ()
toArray	GeoDataSet[] toArray ()
trimToSize	void trimToSize ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(6) GeoDataSetIterator

• 이 클래스는 GeoDataSetCollection 객체에서 각 GeoDataSet 객체를 획득하기 위한 메커니즘을 제공한다.

Class name:
GeoDataSetIterator

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : GeoDataSetCollection
<no rolename> : GeoDataSet in association enumerate

Operations:

Name	Signature
hasNext	boolean hasNext ()
next	GeoDataSet next ()
remove	void remove ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(7) GeoDataConnection

• 이 클래스는 GeoDataSet에 접근하기 위한 API를 제공한다. 이 클래스는 원격지의 데이터셋에 접근하는 경우 그 머신의 상태와 데이터셋의 정보를 가져올 수 있도록 한다.

Class name:
GeoDataConnection

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : GeoDataSet in association get

Operations:

Name	Signature
connect	void connect ()
disconnect	void disconnect ()
getGeoDataSet	GeoDataSet getGeoDataSet (String arg0)
getGeoDataSetCollection	GeoDataSetCollection getGeoDataSetCollection ()
isConnected	boolean isConnected ()
list	String[] list ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(8) GeoDataSet

- 이 클래스는 지리공간 데이터셀을 표현하는 기본 객체를 정의한다.

Class name:
GeoDataSet

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : GeoDataSetCollection
<no rolename> : GeoDataSetIterator
<no rolename> : GeoDataConnection in association get
<no rolename> : GraphicModel in association add
name : java::lang::String

<no rolename> : GeoDataSetCollection

Attributes:

FILEBASE : int = 1
RDBMS : int = 2
OODBMS : int = 3

Operations:

Name	Signature
GeoDataSet	GeoDataSet ()
GeoDataSet	GeoDataSet (String arg0)
getName	String getName ()
getType	int getType ()
isPersistent	boolean isPersistent ()
setPersistent	void setPersistent (boolean arg0)
setType	void setType (int arg0)

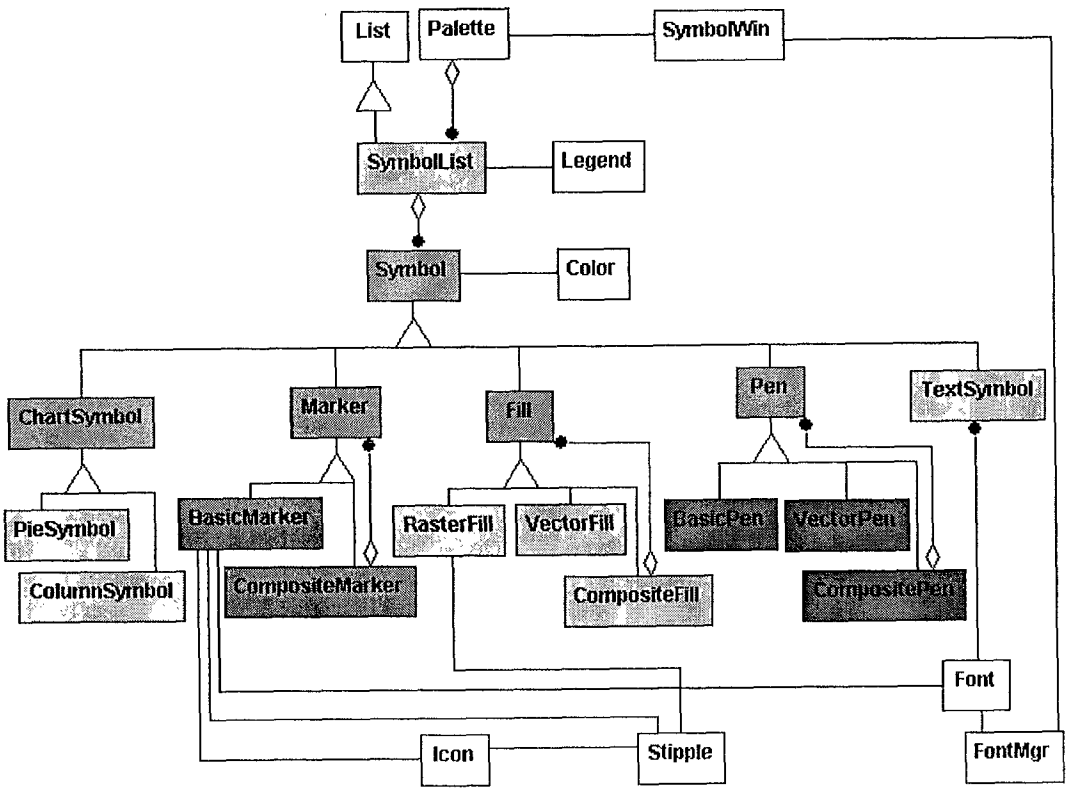
8. 심볼 모델

가. 데이터와 심볼

- 심볼은 동일한 공간 데이터를 뷰마다 다른 형식으로 표현하는 역할을 담당한다. GIS 시스템의 기본 데이터를 시각화하는 것이 타 MIS 시스템과의 주요 차이점이라고 한다면, 심볼은 GIS 시스템의 구현에 있어 가장 중요한 부분 중의 하나가 된다. 매핑 라이브러리의 경우에도 심볼을 어떻게 정의하는가에 따라 시각화의 제공 기능이 달라질 수 있다.
- 일반적인 참조(reference)와 주제도에 공통으로 사용될 수 있는 세 가지 기본 유형의 심볼로 점(point), 선(line), 면(area)을 들 수 있다. 데이터의 특성에 따라 서로 다른 심볼을 사용하여 표현할 수 있으며, 같은 유형의 공간 참조 데이터라고 할지라도 다른 심볼을 사용하여 다양한 유형으로 표현할 수 있다. 구역과 같이 기본 공간객체를 면으로 하여 집계(aggregation)된 자료는 양적(quantitative)인 것

이므로 면 심볼 중에서 양적인 자료를 표현할 수 있는 것이어야 한다.

- [그림 31]은 ArcView의 심볼 아키텍처를 보여주고 있다.



[그림 31] ArcView 3.2의 심볼 다이어그램

- 아크뷰 시스템에서는 심볼 기능을 구현하기 위한 최상위 클래스로 Symbol 클래스를 정의하였으며, 이 클래스는 색상과 관련된 정보를 담고 있다. 심볼의 하위 클래스로는 Marker, Pen, Fill이 있으며, 각각 점, 선, 면 데이터를 표현하기 위한 기본 정보들을 담고 있다.
- Marker 클래스는 학교, 우체국, 여관, 목욕탕 등과 같은 점 정보를 디스플레이 하는 경우에 일반적인 학교 기호, 목욕탕 기호 등을 점 좌표를 가지고 있는 오브젝트와 매치시키는 역할을 한다. BasicMarker는 가장 기본적인 마커의 역할을 담당하는데 이 기능은 CompositeMarker로 확장될 수 있다.
- Pen 클래스는 선의 굵기와 연속성을 설정하는 역할을 한다. 등고선의 굵기 조정이나 일점쇄선, 이점쇄선이 여기에 해당한다. Pen의 경우에도 CompositePen으로 확장될 수 있다.
- Fill 클래스는 폴리곤의 내부를 칠하는 기능을 주로 담당한다. 폴리곤의 내부에

는 Hatch와 같은 패턴이 있을 수 있으며, 이 기능은 Fill 클래스로부터 확장되거나 이 클래스의 정보를 이용하여 구현될 수 있다.

- 아크뷰에서 제공하는 Fill 클래스 관련 enumeration에는 다음과 같은 것들이 있다.

  #VECTORFILL_STYLE_CROSSHATCH
  Cross hatching

  #VECTORFILL_STYLE_DOT
  Dotted line

  #VECTORFILL_STYLE_DOTDENSITY
  Random dots (for dot density mapping)

  #VECTORFILL_STYLE_HATCH
  Single hatching

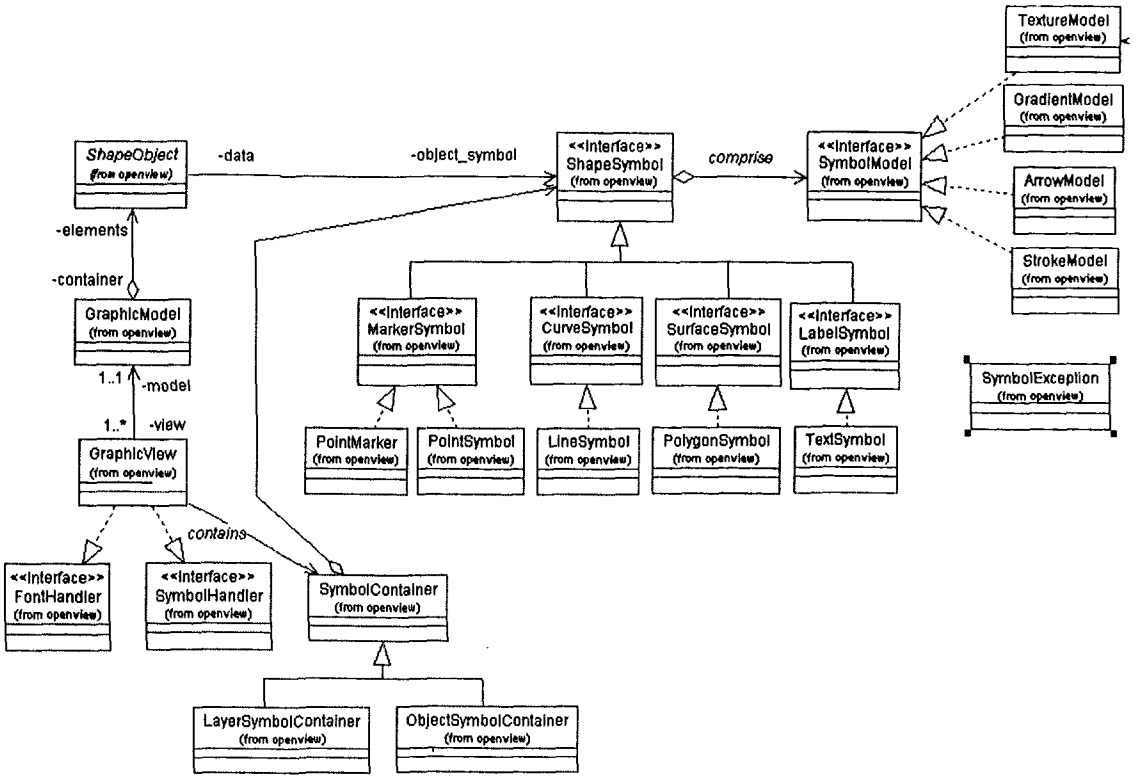
  #VECTORFILL_STYLE_RANDDOT
  Dotted line with random displacement

  #VECTORFILL_STYLE_RECTANGLE
  Filled rectangles

- 아크뷰와 같이 시스템에서 심볼 관련 상수를 정해주는 경우 사용자는 그 정보를 이용하여 심볼 설정을 하는 것이 일반적이다. 그러나, 그 심볼을 확장하는 기능에 있어서는 제한이 많다. 예를 들어, 새로운 도로 심볼을 설정하고자 하는 경우 객체지향 개념에 맞도록 클래스를 정의하기 위해서는 PolygonSymbol로부터 상속받은 RoadSymbol이어야 할 것이다.
- OpenViews에서는 이러한 문제점을 해결하기 위해 자바의 객체지향 아키텍처를 그대로 사용하였으며, 사용자의 개념과 일치하는 심볼 설계와 확장이 가능하도록 하였다.

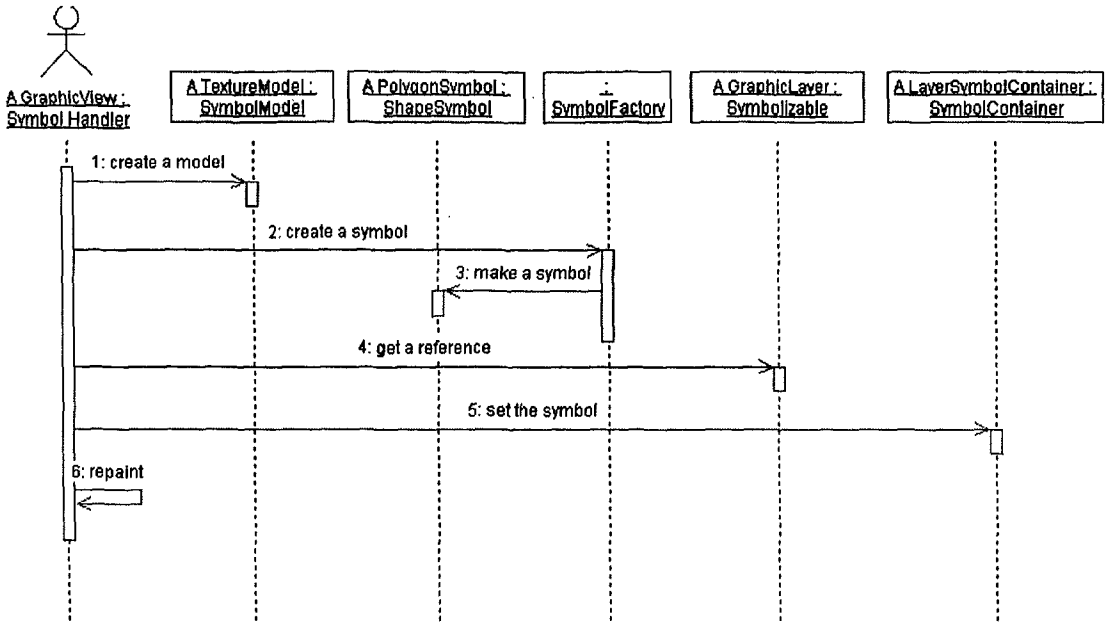
나. 심볼 모델 다이어그램

- 심볼 모델 다이어그램은 [그림 32]과 같다.



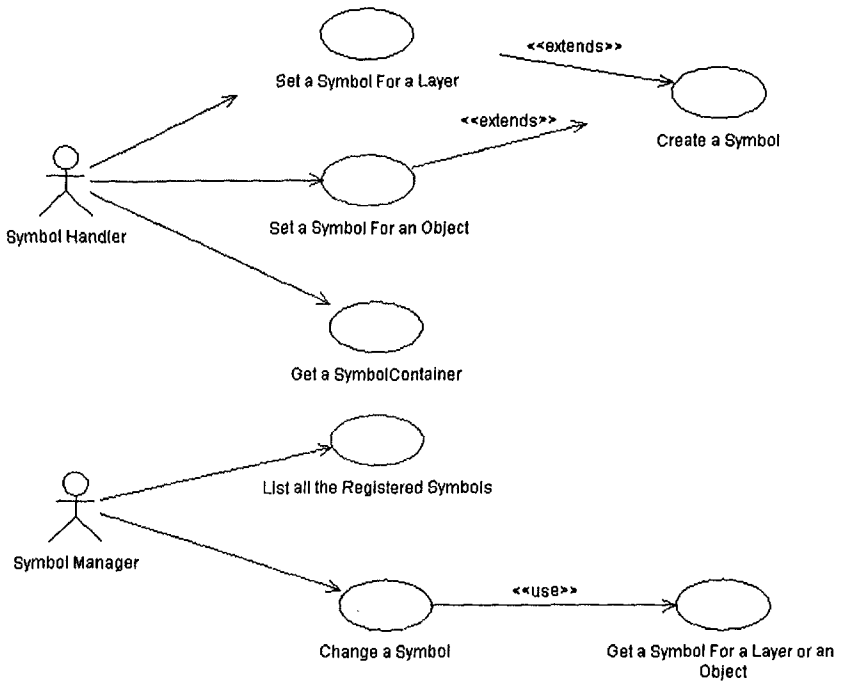
[그림 32] 심볼 모델 다이어그램

- OpenViews에서 심볼을 정의하는 최상위 클래스는 ShapeSymbol로 ShapeObjecct에 일대일로 매치된다. ShapeObject 또는 GraphicLayer에 등록될 수 있는 ShapeSymbol은 실제 오브젝트의 랜더링 역할을 담당하며, 그래픽 뷰는 자신이 공간 데이터를 직접 랜더링하지 않음으로써 랜더링과의 독립성을 유지한다.
- Marker, Pen, Fill로 구성되는 ESRI 사의 제품군과는 달리 OpenViews에서는 자바의 2D 기능을 그대로 사용할 수 있도록 SymbolModel을 정의하여 사용한다 SymbolModel에는 선의 끝부분을 화살표 모양으로 설정해 주는 ArrowModel, 선의 형태를 결정해 주는 StrokeModel, 면의 내부에 관한 TextureModel, 그리고 점차 변하는 유형의 GradientModel로 구성된다.
- ShapeObject에 심볼을 설정하는 과정은 필요한 ShapeSymbol을 만들고 원하는 SymbolModel을 생성하고 추가하면 된다. [그림 33]은 심볼 설정을 위한 시퀀스 다이어그램을 보여준다.

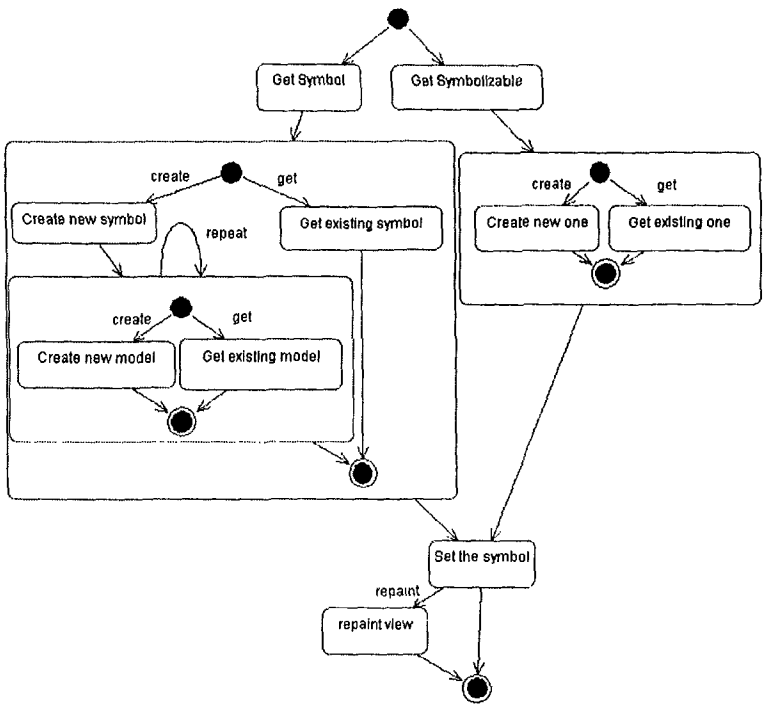


[그림 33] 레이어 심볼을 설정하는 과정

- 심볼을 설정하기 위한 첫번째 과정은 심볼과 심볼 모델을 생성하는 것이다. 심볼 객체를 생성하기 위해서는 SymbolFactory 객체를 이용하며, GraphicLayer 객체에 등록한다. 시스템 내부적인 처리에 있어서는 레이어 심볼을 관리하는 기능을 LayerSymbolContainer가 담당한다.
- [그림 34]는 레이어 심볼 관리를 포함해서 OpenViews내에서 심볼을 설정하는 업무(UseCase) 모델을 표현한 것이다. 심볼에는 레이어 심볼과 오브젝트 심볼이 있으므로 두 경우를 생각할 수 있으며, 심볼 설정 과정에서 심볼 생성 작업이 부가적으로 필요하다.
- 심볼 정보를 변경하거나 등록된 심볼 리스트를 보는 등의 일은 심볼 매니저(SymbolManager) 클래스가 담당한다. 심볼 매니저는 일반 GIS 시스템에서 심볼을 관리하는 모듈의 기능과 다를 것이 없으나, 기본적인 심볼 정보가 SymbolModel 오브젝트에 의해 관리된다는 점에서 차이가 있다.



[그림 34] 심볼 관리 유스케이스 다이어그램



[그림 35] 심볼 설정 액티비티 다이어그램

- 심볼 설정의 과정을 전체적으로 보려면 액티비티 다이어그램이 유용하다. [그림 35]는 이러한 과정을 보여주고 있다. 시스템에 등록된 심볼이 있는 경우에는 그 심볼을 가져다 사용하지만 없는 경우에는 새로운 심볼을 생성한다.

다. 심볼 모델 명세

- 다음은 심볼 모델과 관련된 명세이다.

(1) ArrowModel

- 이 클래스는 선의 양 끝을 화살표 모양으로 그릴 것인가에 대한 정보를 갖고 있는 심볼 모델이다. 이 클래스에는 화살표 모양 심볼과 관련된 상수들이 정의되어 있으며, 그 상수를 통해 사용자가 설정할 수 있도록 한다. 화살표 모양은 점의 배열 순서와 관련되어 있으며, LineString의 경우에는 선의 배열 순서와 관련된다.

Class name:
ArrowModel

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Attributes:
DIRECT_NONE : int = 0
DIRECT_START : int = 1
DIRECT_END : int = 2
DIRECT_BOTH : int = 3
DIRECT_SEQ : int = 4
TYPE_SOFT : int = 0
TYPE_SHARP : int = 1

Operations:

Name	Signature
ArrowModel	ArrowModel (int arg0)
draw	void draw (Graphics arg0, SimplePoint arg1, SimplePoint arg2)
getArrowType	int getArrowType ()
setArrowType	void setArrowType (int arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(2) GradientModel

- 이 클래스는 면 내부의 패턴을 결정하며 특히 연속적인 색상의 변화를 표현한다 예를 들어, 왼쪽에서 오른쪽으로 갈수록 빨간색에서 파란색으로 변화하는 면이 있다고 할 때 이 심볼을 이용할 수 있다. 이러한 패턴은 래스터 시스템에서 특히 유용하지만 벡터 객체에 적용될 경우에도 표현이 좀더 다양해진다는 장점이 있다.

Class name:
GradientModel

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

anchor1 : SimplePoint
anchor2 : SimplePoint
paint : java::awt::Paint

Operations:

Name	Signature
GradientModel	GradientModel ()
GradientModel	GradientModel (int arg0, int arg1, int arg2, SimplePoint arg3, SimplePoint arg4, boolean arg5)
getAnchor1	SimplePoint getAnchor1 ()
getAnchor2	SimplePoint getAnchor2 ()
getColor1	Color getColor1 ()
getColor2	Color getColor2 ()
getTexture	Paint getTexture ()
getTransparency	int getTransparency ()
setAnchor1	void setAnchor1 (SimplePoint arg0)
setAnchor2	void setAnchor2 (SimplePoint arg0)
setColor1	void setColor1 (Color arg0)
setColor2	void setColor2 (Color arg0)
setTransparency	void setTransparency (int arg0)
update	void update ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(3) CurveSymbol

- 이 클래스는 선형 객체와 관련된 심볼의 메소드를 정의한다. 선형 객체는 일반 직선을 포함해서 다양하게 정의될 수 있다. 사용자 정의 선형 객체의 경우에도 문제없이 심볼을 정의할 수 있도록 Curve 객체에는 CurveSymbol 객체를 할당한다.

Class name:

CurveSymbol

Category: openview

Stereotype: Interface

External Documents:

Export Control: Public

Cardinality: n

Hierarchy:

Superclasses: ShapeSymbol

Operations:

Name	Signature
getArrowDirect	int getArrowDirect ()
getArrowModel	ArrowModel getArrowModel ()
getLineTextureModel	LineTextureModel getLineTextureModel ()
setArrowDirect	void setArrowDirect (int arg0)
addSymbolModel	void addSymbolModel (SymbolModel arg0)
drawSymbol	void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getBackground	Color getBackground ()
getForeground	Color getForeground ()
getLineTextureFlag	boolean getLineTextureFlag ()
getRotate	double getRotate ()
getStrokeModel	StrokeModel getStrokeModel ()
getTextureModel	TextureModel getTextureModel ()
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)

setRotate	void setRotate (double arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(4) FontHandler

- 이 클래스는 레이블에 사용되는 일반 텍스트의 폰트를 설정하는 역할을 한다 인터페이스 타입을 가지고 있으며, 주로 GraphicView가 이 역할을 담당한다.

Class name:
FontHandler

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

State machine: No
Concurrency: Sequential
Persistence: Transient

(5) LayerSymbolContainer

- 이 클래스는 레이어 심볼을 관리한다.

Class name:
LayerSymbolContainer

Category: openview
External Documents:
Export Control: Public

Cardinality: n
Hierarchy:
Superclasses: SymbolContainer
Associations:

<no rolename> : GraphicView

Operations:

Name	Signature
LayerSymbolContainer	LayerSymbolContainer ()
SymbolContainer	SymbolContainer ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(6) LineSymbol

- 이 클래스는 LineShape 오브젝트의 심볼이다. 관련된 심볼 모델은 ArrowModel, StrokeModel, TextureModel이다. 일반적으로 선형 객체는 Texture를 가질 수 없지만 OpenViews에서는 선형 객체도 Texture를 가질 수 있도록 하였다.

Class name:
LineSymbol

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

model_list : java::util::LinkedList

Operations:

Name	*Signature*
LineSymbol	LineSymbol ()
LineSymbol	LineSymbol (int arg0, boolean arg1, double arg2, int arg3)
addSymbolModel	void addSymbolModel (SymbolModel arg0)
drawSymbol	void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getArrowDirect	int getArrowDirect ()
getArrowModel	ArrowModel getArrowModel ()
getBackground	Color getBackground ()
getForeground	Color getForeground ()
getLineTextureFlag	boolean getLineTextureFlag ()
getLineTextureModel	LineTextureModel getLineTextureModel ()
getRotate	double getRotate ()
getStrokeModel	StrokeModel getStrokeModel ()
getTextureModel	TextureModel getTextureModel ()
setArrowDirect	void setArrowDirect (int arg0)
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)
setRotate	void setRotate (double arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(7) LineTextureModel

- 이 클래스는 선형 객체에 Texture를 주기 위해 특별히 확장되었다.

Class name:
LineTextureModel

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: TextureModel

Operations:

Name	Signature
LineTextureModel	LineTextureModel (int arg0)
LineTextureModel	LineTextureModel (int arg0, int arg1, int arg2, boolean arg3, int arg4, String arg5)
TextureModel	TextureModel (int arg0)
TextureModel	TextureModel (int arg0, int arg1, int arg2, boolean arg3, int arg4, String arg5)
getBackground	Color getBackground ()
getForeground	Color getForeground ()
getImagePath	String getImagePath ()
getModel	int getModel ()
getSize	int getSize ()
getTexture	Paint getTexture ()
isTransparent	boolean isTransparent ()
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)

setImagePath	void setImagePath (String arg0)
setModel	void setModel (int arg0)
setSize	void setSize (int arg0)
setTexture	void setTexture (TexturePaint arg0)
setTransparent	void setTransparent (boolean arg0)
update	void update ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(8) MarkerSymbol

- 이 클래스는 점 객체에 대한 심볼을 정의한다. 점 객체에 대한 심볼은 크게 벡터 형식의 심볼과 이미지 형식의 심볼로 나누어질 수 있다. 벡터 형식의 심볼은 여러 벡터 객체가 그룹화되어 하나의 심볼을 구성하는 것으로 CAD와 같은 시스템에서 설정하는 Shape와 유사하다. 이미지 형식의 심볼은 점 객체에서 얻어진 좌표값을 이용하여 그 위치에 특정 이미지를 뿌리는 방식이다.

Class name:
MarkerSymbol

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeSymbol

Operations:

Name	Signature

getSize	Float getSize ()
setSize	void setSize (float arg0)
addSymbolModel	void addSymbolModel (SymbolModel arg0)
drawSymbol	void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getBackground	Color getBackground ()
getForeground	Color getForeground ()
getLineTextureFlag	boolean getLineTextureFlag ()
getRotate	double getRotate ()
getStrokeModel	StrokeModel getStrokeModel ()
getTextureModel	TextureModel getTextureModel ()
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)
setRotate	void setRotate (double arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(9) ObjectSymbolContainer

- 이 클래스는 오브젝트 심볼을 관리하는 역할을 담당한다.

Class name:
ObjectSymbolContainer

Category: openview
External Documents:
Export Control: Public
Cardinality: n

Hierarchy:
Superclasses: SymbolContainer
Associations:

<no rolename> : GraphicView

Operations:

Name	Signature
ObjectSymbolContainer	ObjectSymbolContainer ()
SymbolContainer	SymbolContainer ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(10) PointMarker

- 이 클래스는 PointShape의 좌표값을 이용하여 이미지를 설정하는 역할을 한다 앞에서 언급한 이미지 형식의 심볼을 설정한다.

Class name:
PointMarker

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

img : java::awt::Image
name : java::lang::String

Operations:

Name	Signature
PointMarker	PointMarker ()
PointMarker	PointMarker (Image arg0, float arg1)
addSymbolModel	void addSymbolModel (SymbolModel arg0)
drawSymbol	void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getBackground	Color getBackground ()
getForeground	Color getForeground ()
getImage	Image getImage ()
getLineTextureFlag	boolean getLineTextureFlag ()
getName	String getName ()
getRotate	double getRotate ()
getSize	float getSize ()
getStrokeModel	StrokeModel getStrokeModel ()
getTextureModel	TextureModel getTextureModel ()
imageUpdate	boolean imageUpdate (Image arg0, int arg1, int arg2, int arg3, int arg4, int arg5)
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)
setImage	void setImage (Image arg0)
setImage	void setImage (String arg0)
setImage	void setImage (URL arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)
setRotate	void setRotate (double arg0)
setSize	void setSize (float arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(11) PointSymbol

- 이 클래스는 PointShape에 벡터 형식의 심볼을 설정하는 역할을 담당한다. 기본적인 시스템 제공 심볼은 상수로 정의되어 있다.

Class name:
PointSymbol

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

fcolor : java::awt::Color
bcolor : java::awt::Color

Attributes:

TYPE_SQUARE : int = 1
TYPE_DIAMOND : int = 2
TYPE_CIRCLE : int = 4
TYPE_CROSS : int = 8
TYPE_PLUS : int = 16
TYPE_FILLED_SQUARE : int = 32
TYPE_FILLED_CIRCLE : int = 64
TYPE_FILLED_DIAMOND : int = 128
TYPE_TRIANGLE : int = 256
TYPE_FILLED_TRIANGLE : int = 512

Operations:

Name	Signature
PointSymbol	PointSymbol ()
PointSymbol	PointSymbol (int arg0, float arg1, Color arg2, Color arg3)
addSymbolModel	void addSymbolModel (SymbolModel arg0)
drawSymbol	void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getBackground	Color getBackground ()
getForeground	Color getForeground ()
getLineTextureFlag	boolean getLineTextureFlag ()
getRotate	double getRotate ()
getSize	float getSize ()
getStrokeModel	StrokeModel getStrokeModel ()
getTextureModel	TextureModel getTextureModel ()
getType	int getType ()
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)
setRotate	void setRotate (double arg0)
setSize	void setSize (float arg0)
setType	void setType (int arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(12) PolygonSymbol

- 이 클래스는 면 객체의 심볼을 담당한다. ArrowModel, StrokeModel, TextureModel, GradientModel 등 정의된 모든 모델이 사용될 수 있으며, 면의

내부 뿐만 아니라 면을 표현하는 선형 객체에도 Texture가 설정될 수 있다. 면 내부에 그려지는 Hatch 패턴은 상수로 정의되어 있다.

Class name:
PolygonSymbol

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

model_list : java::util::LinkedList

Operations:

Name	Signature
PolygonSymbol	PolygonSymbol ()
PolygonSymbol	PolygonSymbol (int arg0, int arg1, boolean arg2, double arg3, boolean arg4, boolean arg5)
addSymbolModel	Void addSymbolModel (SymbolModel arg0)
drawSymbol	Void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getFillFlag	boolean getFillFlag ()
getForeground	Color getForeground ()
getLineTextureFlag	boolean getLineTextureFlag ()
getRotate	double getRotate ()
getSimpleFlag	boolean getSimpleFlag ()
getStrokeModel	StrokeModel getStrokeModel ()

getTextureModel	TextureModel getTextureModel ()
setBackground	void setBackground (Color arg0)
setFillFlag	void setFillFlag (boolean arg0)
setForeground	void setForeground (Color arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)
setRotate	void setRotate (double arg0)
setSimpleFlag	void setSimpleFlag (boolean arg0)
getBackground	getBackground ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(13) StrokeModel

- 이 클래스는 선의 형태를 정의한다. 일점 쇄선이나 이점 쇄선이 제공되며, 사용자가 새로 확장, 정의할 수 있다. 시스템 정의 심볼은 상수로 정의된다.

Class name:
StrokeModel

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

stroke : java::awt::BasicStroke
TYPE_DEFAULT : java::awt::BasicStroke
TYPE_DASH : java::awt::BasicStroke
TYPE_DASH_DOT : java::awt::BasicStroke

TYPE_DASH_DOT_DOT : java::awt::BasicStroke
TYPE_DOT : java::awt::BasicStroke
TYPE_DOT_DOT : java::awt::BasicStroke

Operations:

Name	Signature
StrokeModel	StrokeModel ()
StrokeModel	StrokeModel (float arg0, int arg1, int arg2, float arg3, float[] arg4, float arg5)
StrokeModel	StrokeModel (BasicStroke arg0)
StrokeModel	StrokeModel (BasicStroke arg0, float arg1)
equalsDashArray	boolean equalsDashArray (float[] arg0)
getDashArray	float[] getDashArray ()
getDashPhase	float getDashPhase ()
getEndCap	int getEndCap ()
getLineJoin	int getLineJoin ()
getLineWidth	float getLineWidth ()
getMiterLimit	float getMiterLimit ()
getStroke	BasicStroke getStroke ()
setDashArray	void setDashArray (float[] arg0)
setDashPhase	void setDashPhase (float arg0)
setEndCap	void setEndCap (int arg0)
setLineJoin	void setLineJoin (int arg0)
setLineWidth	void setLineWidth (float arg0)
setMiterLimit	void setMiterLimit (float arg0)
update	void update ()
zoomDash	void zoomDash (float arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(14) SurfaceSymbol

- 이 클래스는 일정한 면적을 가지는 심볼에 대해 정의한다. OpenGIS 명세에서 정의하는 Surface 객체에 대한 심볼 정의를 위해 사용한다.

Class name:
SurfaceSymbol

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeSymbol

Operations:

Name	Signature
getFillFlag	boolean getFillFlag ()
setFillFlag	void setFillFlag (boolean arg0)
addSymbolModel	void addSymbolModel (SymbolModel arg0)
drawSymbol	void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getBackground	Color getBackground ()
getForeground	Color getForeground ()
getLineTextureFlag	boolean getLineTextureFlag ()
getRotate	double getRotate ()
getStrokeModel	StrokeModel getStrokeModel ()

getTextureModel	TextureModel getTextureModel ()
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)
setRotate	void setRotate (double arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(15) SymbolException

- 이 클래스는 심볼 설정과 생성에 관련된 예외 상황을 알려준다.

Class name:
SymbolException

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: java::lang::Exception

Operations:

Name	Signature
SymbolException	SymbolException ()
SymbolException	SymbolException (String arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(16) SymbolHandler

- 심볼 설정을 하는 주요 메소드들을 정의한다. OpenViews 매핑 라이브러리는 심볼을 설정할 수 있는 권한을 여러 객체에게 주고 있으며, 각각의 객체가 갖는 위치에 따라 심볼 설정이 달라진다.
- GraphicView가 SymbolHandler로 역할하는 경우에는 그 뷰에만 심볼이 보여진다. GraphicModel이 SymbolHandler로 역할하는 경우에는 그 모델에 등록된 모든 뷰에서 그 심볼을 사용한다. 특히 ShapeObject에 설정된 심볼은 어떤 뷰든 상관없이 볼 수 있다. 단, 심볼 설정의 우선순위는 View > Model > Layer > Object 순이다.

Class name:
SymbolHandler

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Attributes:

LAYER_SYMBOL : int = 1
OBJECT_SYMBOL : int = 2

Operations:

Name	Signature
getLayerSymbol	ShapeSymbol getLayerSymbol (int arg0)
getObjectSymbol	ShapeSymbol getObjectSymbol (ShapeObject arg0)
getSymbolContainer	SymbolContainer getSymbolContainer (int arg0)
setLayerSymbol	void setLayerSymbol

	(int arg0, ShapeSymbol arg1, boolean arg2, boolean arg3)
setObjectSymbol	void setObjectSymbol (ShapeObject arg0, ShapeSymbol arg1, boolean arg2)

State machine: No
Concurrency: Sequential
Persistence: Transient

(17) TextSymbol

- 이 클래스는 레이블 중에서 텍스트와 관련된 오브젝트의 랜더링 방식을 정의한다.

Class name:
TextSymbol

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

outline_color : java::awt::Color
fcolor : java::awt::Color
font : java::awt::Font

Operations:

Name	**Signature**
TextSymbol	TextSymbol ()
TextSymbol	TextSymbol (Font arg0, Color arg1, Color arg2, double arg3, boolean arg4, boolean arg5)

addSymbolModel	void addSymbolModel (SymbolModel arg0)
drawSymbol	void drawSymbol (Graphics arg0, Transform2D arg1, ShapeObject arg2)
getBackground	Color getBackground ()
getFont	Font getFont ()
getForeground	Color getForeground ()
getLineTextureFlag	boolean getLineTextureFlag ()
getOutlineColor	Color getOutlineColor ()
getOutlineFlag	boolean getOutlineFlag ()
getRotate	double getRotate ()
getStrokeModel	StrokeModel getStrokeModel ()
getTextureModel	TextureModel getTextureModel ()
getUnderlineFlag	boolean getUnderlineFlag ()
setBackground	void setBackground (Color arg0)
setFont	void setFont (Font arg0)
setForeground	void setForeground (Color arg0)
setLineTextureFlag	void setLineTextureFlag (boolean arg0)
setOutlineColor	void setOutlineColor (Color arg0)
setOutlineFlag	void setOutlineFlag (boolean arg0)
setRotate	void setRotate (double arg0)
setUnderlineFlag	void setUnderlineFlag (boolean arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(18) TextureModel

- 이 클래스는 면의 내부를 칠하거나 선형 객체의 모양을 결정하는 경우에 사용될

수 있다. 일반적인 Texture를 결정한다.

Class name:
TextureModel

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

imagePath : java::lang::String
paint : java::awt::Paint

Attributes:

TYPE_DEFAULT : int = -1
TYPE_EMPTY : int = 0
TYPE_IMAGE : int = 1
TYPE_FDIAGONAL : int = 2
TYPE_BDIAGONAL : int = 3
TYPE_HORIZONTAL : int = 4
TYPE_VERTICAL : int = 5
TYPE_CROSS : int = 6
TYPE_DIAGONAL_CROSS : int = 7

Operations:

Name	Signature
TextureModel	TextureModel (int arg0)
TextureModel	TextureModel (int arg0, int arg1, int arg2, boolean arg3, int arg4, String arg5)
getBackground	Color getBackground ()

getForeground	Color getForeground ()
getImagePath	String getImagePath ()
getModel	Int getModel ()
getSize	Int getSize ()
getTexture	Paint getTexture ()
isTransparent	boolean isTransparent ()
setBackground	void setBackground (Color arg0)
setForeground	void setForeground (Color arg0)
setImagePath	void setImagePath (String arg0)
setModel	void setModel (int arg0)
setSize	void setSize (int arg0)
setTexture	void setTexture (TexturePaint arg0)
setTransparent	void setTransparent (boolean arg0)
update	void update ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(19) LabelSymbol

- 이 클래스는 텍스트 정보를 포함한 레이블 객체에 관한 랜더링 정보를 설정한다

Class name:
LabelSymbol

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n

Hierarchy:
Superclasses: ShapeSymbol
State machine: No
Concurrency: Sequential
Persistence: Transient

(20) SymbolModel

- 이 클래스는 심볼 정보를 담고 있는 데이터 구조를 정의한다. SymbolModel에는 ArrowModel, StrokeModel, TextureModel, GradientModel이 있으며, 사용자에 의해 이러한 시스템 클래스들이 확장될 수 있다.

Class name:
SymbolModel

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : ShapeSymbol in association comprise

State machine: No
Concurrency: Sequential
Persistence: Transient

(21) ShapeSymbol

- ShapeObject에 설정할 수 있는 심볼에 관한 최상위 클래스이다. 이 심볼은 각각의 오브젝트 또는 한 레이어나 모델 또는 뷰에 들어있는 오브젝트 전체에 설정될 수 있다.

Class name:
ShapeSymbol

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : SymbolModel in association comprise
<no rolename> : SymbolContainer in association contains
<no rolename> : GraphicLayer
data : ShapeObject

State machine: No
Concurrency: Sequential
Persistence: Transient

(22) SymbolContainer

- 심볼을 담고 있는 그룻으로 심볼을 관리하는 최상위 클래스이다.

Class name:
SymbolContainer

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: java::util::HashMap
Associations:

<no rolename> : GraphicView
<no rolename> : ShapeSymbol in association contains

Operations:

Name	Signature
SymbolContainer	SymbolContainer ()

State machine: No
Concurrency: Sequential
Persistence: Transient

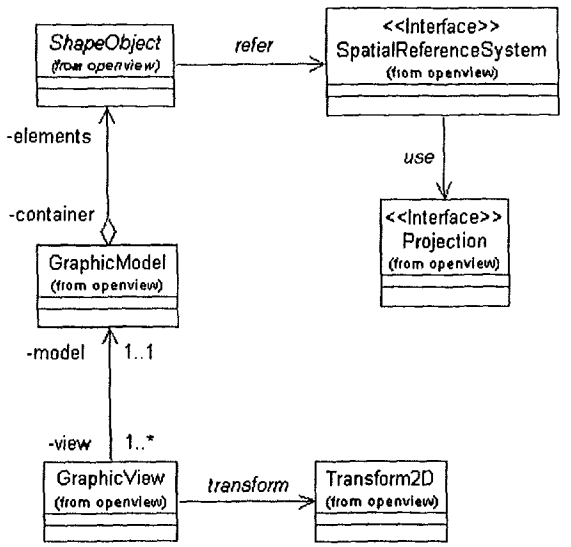
9. 변환 모델

가. 좌표 변환

- 지구좌표를 지도좌표로 변환하기 위해서는 다양한 좌표변환 기법이 동원된다. 마찬가지로 하나의 뷰 안에서 일정 부분을 확대/축소하는 경우 이러한 좌표 변환이 필요하다. 확대/축소를 위한 기법은 크게 두 가지가 있다. 하나는 전체 뷰의 범위와 데이터의 범위에 관련된 비율을 이용하여 확대/축소를 하는 방법이며, 두 번째는 행렬에 기반한 변환 기법을 사용하는 것이다. 가장 잘 알려진 AffineTransformation 과 같은 방법은 가장 간단하게 데이터를 변환하는 방법 중의 하나이다.

나. 변환 모델 다이어그램

- 변환 모델 다이어그램은 [그림 36]과 같다.



[그림 36] 변환 모델 다이어그램

- Transform2D 클래스는 뷰 안에서 오브젝트가 가지는 스크린 좌표의 변환을 담당한다. 스크린 좌표의 변환 이외에 지구좌표나 지상좌표의 변환 또는 정해진 좌표계 정보는 SpatialReferenceSystem 클래스에서 담당한다. Projection 클래스는 하위 클래스들을 이용하여 투영법을 지원한다.

다. 변환 모델 명세

- 변환모델 명세는 다음과 같다.

(1) Transform2D

- 이 클래스는 스크린 좌표 변환을 담당한다.

Class name:
Transform2D

Category: openview
External Documents:
Export Control: Public

Cardinality: n

Hierarchy:

Superclasses: none

Associations:

<no rolename> : GraphicView in association transform

Operations:

Name	Signature
Transform2D	Transform2D (Transform2D arg0)
apply	Void apply (PointImpl arg0)
apply	Void apply (SimplePoint arg0)
apply	Void apply (SimpleRectangle arg0)
equals	*boolean equals (Object arg0)*
getDeterminant	double getDeterminant ()
getx0	double getx0 ()
getx11	double getx11 ()
getx12	double getx12 ()
getx21	double getx21 ()
getx22	double getx22 ()
gety0	double gety0 ()
rotate	Void rotate (double arg0, double arg1, double arg2)
scale	void scale (double arg0, double arg1, double arg2, double arg3)
translate	Void translate (double arg0, double arg1)

State machine: No

Concurrency: Sequential

Persistence: Transient

(2) Projection

- 이 클래스는 객체가 담겨있는 뷰의 투영법을 결정한다.

Class name:
Projection

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : SpatialReferenceSystem in association use

State machine: No
Concurrency: Sequential
Persistence: Transient

(3) SpatialReferenceSystem

- 이 클래스는 객체가 담겨있는 뷰의 좌표체계에 관한 정보를 담고 있다.

Class name:
SpatialReferenceSystem

Category: openview
Stereotype: Interface
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none

Associations:

<no rolename> : ShapeObject in association refer
<no rolename> : Projection in association use

State machine: No
Concurrency: Sequential
Persistence: Transient

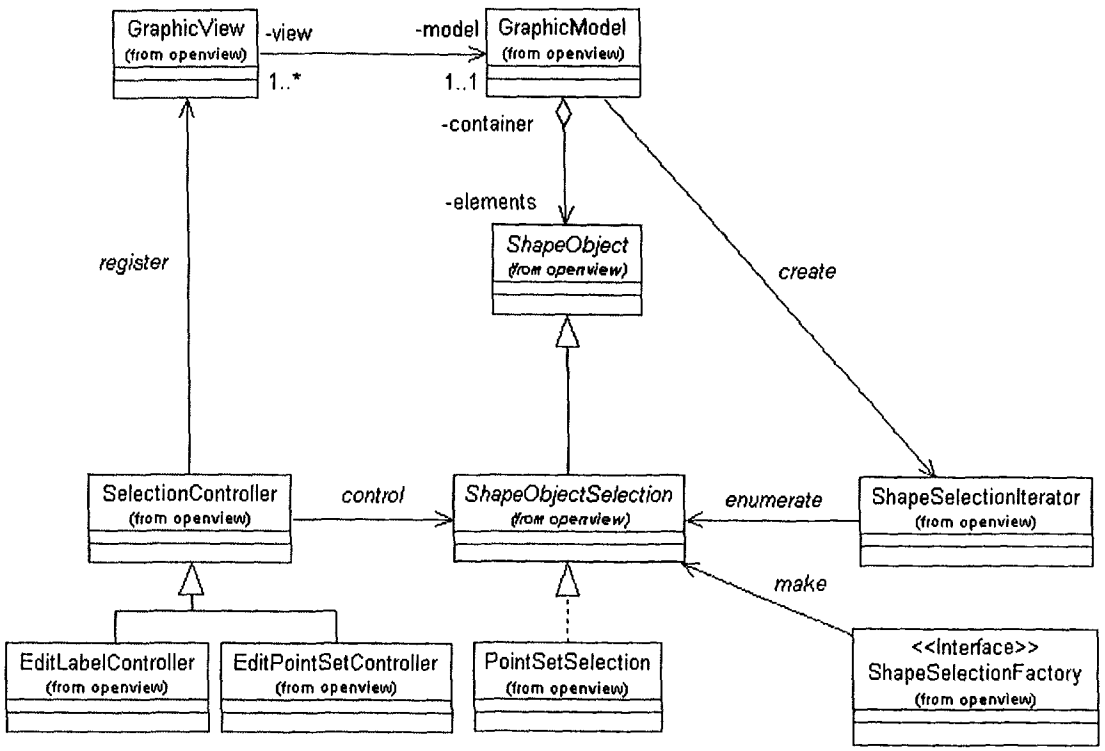
10. 편집 모델

가. 공간 데이터 편집

- 원하는 유형의 속성 데이터를 미리 가지고 있는 경우에는 매핑 커널에서 부가적인 작업을 할 필요가 없으나 사용자 또는 프로그래머는 수시로 필요한 데이터를 생성하는 것이 일반적이다. 편집 모델은 공간 데이터 및 속성 데이터에 관한 접근과 함께 좌표값 등을 바꾸어 보여주거나 바꾸어 저장하는 메커니즘을 정의한다.
- 뷰어 애플리케이션에서 요구하는 것보다 편집 애플리케이션에서 요구하는 정보는 훨씬 더 다양하다. 실제로 공간 데이터를 편집하기 위해서는 snapping, clipping, intersection 로직 등이 필요하다.
- OpenViews는 공간 데이터 편집을 위한 클래스들을 제공하고 있으며, 실제 공간 데이터의 선택과 편집, 이동, 공간 연산자 등에 관련된 기본 연산이 가능하다.

나. 편집 모델 다이어그램

- 편집 모델 다이어그램은 [그림 37]과 같다.



[그림 37] 편집 모델 다이어그램

- ShapeObjectSelection 오브젝트는 ShapeObject를 선택하였을 때 애플리케이션에 의해 생성되며, 주로 점 데이터의 집합으로 이루어지는 PointSet 오브젝트를 편집하는 데 사용된다.
- ShapeObjectSelection 객체는 ShapeObjectSelectionFactory 오브젝트에 의해 만들어지는 데 SelectionController에 의해 제어된다.

다. 편집 모델 명세

- 편집 모델 명세는 다음과 같다.

(1) EditLabelController

- 이 클래스는 레이블에 들어있는 텍스트 정보를 갱신하기 위해 사용되는 콘트롤러이다.

Class name:
EditLabelController

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: SelectionController
State machine: No
Concurrency: Sequential
Persistence: Transient

(2) EditPointSetController

- 이 클래스는 일련의 점 좌표를 가지고 있는 PointSet을 편집하기 위한 콘트롤러로 점의 추가, 삭제 등이 가능하도록 도와준다.

Class name:
EditPointSetController

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObjectController, SelectionController

Operations:

Name	Signature
EditPointSetController	EditPointSetController ()
getCursor	Cursor getCursor ()
getIndex	int getIndex ()

processEvent	boolean processEvent (ShapeObject arg0, AWTEvent arg1, ShapeObjectControllerContext arg2)
setCursor	void setCursor (Cursor arg0)
getObject	ShapeObjectController getObject (String arg0)
processEvent	boolean processEvent (ShapeObject arg0, AWTEvent arg1, ShapeObjectControllerContext arg2)

State machine: No
Concurrency: Sequential
Persistence: Transient

(3) PointSetSelection

- 이 클래스는 PointSet 클래스 안에 정의된 점들을 선택하는 데 필요한 기능들을 정의하고 있다. PointSet 객체를 알고 있는 클래스이다.

Class name:
PointSetSelection

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: HandlerSelection

State machine: No
Concurrency: Sequential
Persistence: Transient

(4) ShapeObjectSelectionIterator

- 선택된 ShapeObject를 표현하는 ShapeObjectSelection 객체들에 대한 접근 객체이다. Enumeration을 하는 동안 필요한 작업들을 수행할 수 있다.

Class name:
ShapeSelectionIterator

Category: openview
External Documents:
Export Control: Implementation
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : GraphicModel in association create
<no rolename> : ShapeObjectSelection
State machine: No
Concurrency: Sequential
Persistence: Transient

(5) ShapeObjectSelection

- 이 클래스는 ShapeObject에 대한 Selection 정보를 저장하고 있다. 사용자가 마우스 이벤트를 통하여 ShapeObject를 선택하는 경우 선택된 정보를 저장하고 이후의 처리 작업을 기다리는 과정을 거친다. 또한 필요한 경우 뷰에 선택 정보에 대한 랜더링을 담당한다. ShapeObject로부터 상속을 받았으므로 관련된 모든 기능을 수행할 수 있다.

Class name:
ShapeObjectSelection

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: ShapeObject
Associations:

<no rolename> : ShapeSelectionIterator
in association enumerate
<no rolename> : ShapeSelectionFactory
in association make
<no rolename> : SelectionController
in association control

Operations:

Name	Signature
ShapeObjectSelection	ShapeObjectSelection (ShapeObject arg0)
setController	void setController (String arg0)
copy	ShapeObject copy ()
getController	String getController ()
getObject	ShapeObject getObject ()
isVisible	boolean isVisible ()

State machine: No
Concurrency: Sequential
Persistence: Transient

(6) SelectionController

- 이 클래스는 ShapeObjectSelection과 ShapeObject 와의 선택 및 등록 관계를 담당하는 콘트롤러이다.

Class name:
SelectionController

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:

Superclasses: none

Associations:

<no rolename> : GraphicView in association register

<no rolename> : ShapeObjectSelection
in association control

Operations:

Name	Signature
SelectionController	SelectionController ()
selectObject	void selectObject (ShapeObject arg0)
getEdit	boolean getEditFlag ()
getDrag	boolean getDrag ()
getMove	boolean getMove ()
getMultiple	boolean getMultiple ()
getVisibleFlag	boolean getVisibleFlag ()
releaseAll	void releaseAll ()
selectObject	void selectObject (ShapeObject arg0)
selectionChanged	void selectionChanged (SelectionChangedEvent arg0)
setDrag	void setDrag (boolean arg0)
setEdit	void setEdit (boolean arg0)
setMove	void setMove (boolean arg0)
setMultiple	void setMultiple (boolean arg0)
setVisibleFlag	void setVisibleFlag (boolean arg0)

State machine: No

Concurrency: Sequential

Persistence: Transient

11. 분류 모델

가. 데이터 분류와 표현

- 분류 모델은 연속 데이터를 일정 구간으로 분류할 수 있도록 하는 구조를 정의하고 있다. 분류 모델을 정의하는 가장 큰 이유는 공간 객체들이 가지고 있는 속성을 기반으로 심볼을 정의하는 경우 연속 데이터는 일정 구간을 기준으로 분류하는 기능이 있어야 심볼 설정이 가능하기 때문이다.
- 분류의 과정에서는 클래스(class) 수나 영역(category)을 어떻게 나눌 것인가가 주요 관심사가 된다. 일반적으로는 평균, 표준편차 및 사용자가 임의로 지정한 클래스 수를 이용하여 그 분포를 표현하며, 구 또는 동과 같은 행정단위를 기준으로 하여 수집된 자료를 이용한다.
- 클래스간의 간격을 설정하는 다양한 방법 및 특성은 [표 4]에 나와있는 바와 같다. OpenViews 시스템에서는 이 중에서 동일간격 분류(equal interval), 표준편차를 이용한 분류(standard deviation), 빈도수를 이용한 분류(equal frequency)를 구현하였다

방법	특징
Equal Steps	일련의 데이터가 만들어내는 히스토그램이 정사각형일 때 특히 유용하게 사용될 수 있다.
Standard Deviation	일련의 데이터가 정규 분포를 따르는 경우에만 사용되어야 한다. 평균으로부터 얼마나 분산되어 있는가를 알고자 하는 경우에 유용하다.
Arithmetic Progression	일련의 데이터가 등차수열의 형태를 띠는 경우에 유용하다.
Geometric Progression	일련의 데이터가 등비수열의 형태를 띠는 경우에 유용하다.
Quantiles	각 구간에 포함된 관찰치의 수가 동일하도록 만드는 방법으로 기본 공간단위의 크기가 크게 차이를 보이는 경우에는 잘못 해석될 수 있다.
Natural Breaks	빈도 분포에 있어 급격한 변화가 보이는 지점을 구간의 경계로 하는 방법으로 데이터의 특성을 잘 나타낼 수 있다.
Optimal	natural break 방법의 확장으로 계산된 index값을 만들어낸다.

[표 4] 분류 방법

- 구현된 세 가지 분류 방식의 분류 단계는 다음과 같다.

O 동일간격 분류

- 1단계

  데이터(R)의 범위를 계산한다.
  R = H - L (단, H : 가장 큰 값, L : 가장 작은 값)

- 2단계

  공통 간격(Common Difference)을 구한다.
  CD = R / n (단, n : 클래스의 수)

- 3단계

  클래스 경계를 결정한다.
  L + 1 * CD = 첫 번째 경계
  L + 2 * CD = 두 번째 경계
  ...
  L + (n-1) * CD = 마지막 경계 (단, n은 클래스 경계 수)

O 표준편차를 이용한 분류

- 1단계

  정규분포를 따른다고 가정한다.

- 2단계

  평균과 표준편차를 구한다.

- 3단계

  평균에서 표준편차를 빼거나 더함으로써 클래스 경계를 결정한다.

O 구간수를 이용한 분류

- 1 단계

  일련의 데이터를 오름차순으로 정렬한다.

- 2 단계

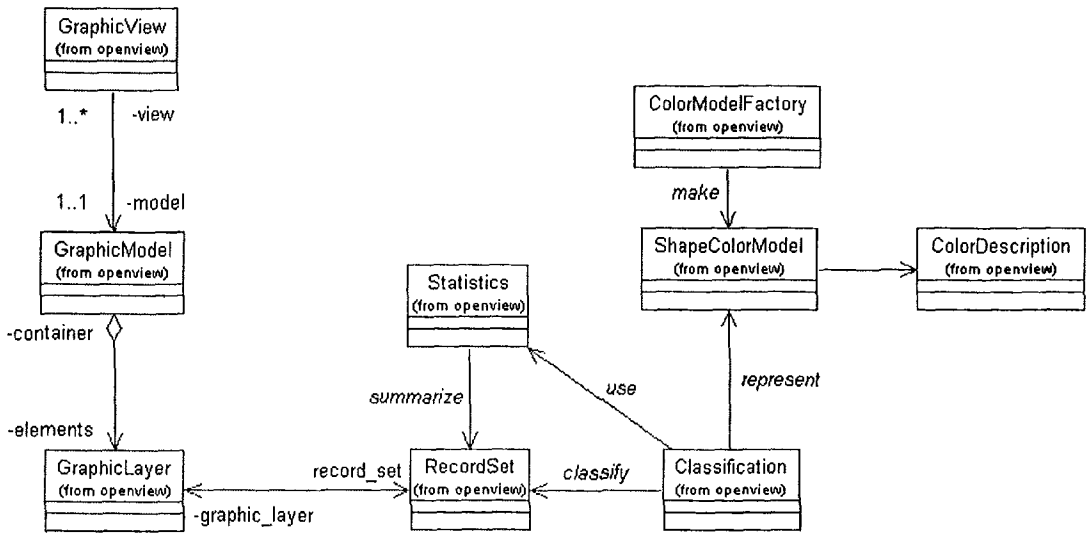
  각 클래스에 들어갈 데이터의 수를 정한다.
  K = 데이터의 수 / 클래스의 수

- 3 단계

작은 수부터 시작해서 각 클래스마다 K 개의 데이터를 할당한다.

나. 분류 모델 다이어그램

- 분류 모델 다이어그램은 [그림 38]과 같다.



[그림 38] 분류 모델 다이어그램

- Classification 클래스는 구현된 세 가지 분류 방식을 통해 입력된 연속 데이터를 구간별로 나누어준다. 사용자는 클래스 수와 간격을 임의로 조정할 수 있다. 기본적인 분류 대상 데이터는 속성 데이터이므로 RecordSet 안의 속성 데이터가 Classification 객체를 위해 사용된다. RecordSet은 가장 기본적인 통계 데이터(합계, 평균, 표준편차 등...)를 추출하기 위하여 Statistics 객체를 이용할 수 있다.
- ShapeColorModel은 분류된 연속 데이터의 심볼을 자동으로 설정하기 위하여 사용된다. ColorDescription 객체는 기본적인 색상 정보를 표현한다.

다. 분류 모델 명세

- 분류 모델 명세는 다음과 같다.

(1) Statistics

- 이 클래스는 기본적인 통계정보를 제공한다. 가장 기본적인 통계정보 이외의 복잡한 공간분석 및 공간 데이터를 포함한 통계 분석은 공간통계로의 확장과 이를 통한 분석 모듈의 추가를 통해 구현된다.

Class name:
Statistics

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : RecordSet in association summarize
<no rolename> : Classification in association use

Operations:

Name	Signature
Statistics	Statistics ()
average	double average (double[] arg0)
inverseFCdf	double inverseFCdf (double arg0, double arg1, double arg2)
inverseNormalCdf	double inverseNormalCdf (double arg0)
inverseTCdf	double inverseTCdf (double arg0, double arg1)
kurtosis	double kurtosis (double[] arg0)
linearFit	Double[] linearFit (double[] arg0, double[] arg1)
maximum	double maximum (double[] arg0)
median	double median (double[] arg0)

minimum	double minimum (double[] arg0)
normalCdf	double normalCdf (double arg0)
range	double range (double[] arg0)
skew	double skew (double[] arg0)
slope	double slope (double[] arg0, double[] arg1)
standardDeviation	double standardDeviation (double[] arg0)
tCdf	double tCdf (double arg0, double arg1)
variance	double variance (double[] arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(2) ColorModelFactory

- 이 클래스는 연속 데이터의 표현에 필요한 ColorModel을 생성하는 메소드를 제공한다. 기본적으로 제공되는 ColorModel에는 Single, Elevation, BlackWhite 가 있다.

Class name:
ColorModelFactory

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : ShapeColorModel in association make

Operations:

Name	Signature
ColorModelFactory	ColorModelFactory ()
makeBlackWhite	ShapeColorModel makeBlackWhite ()
makeElevation	ShapeColorModel makeElevation (int arg0)
makeSingle	ShapeColorModel makeSingle (byte arg0, byte arg1, byte arg2)

State machine: No
Concurrency: Sequential
Persistence: Transient

(3) Classification

- 이 클래스는 분류 방법을 제공한다. 동일 구간, 동일 간격, 표준 편차를 이용한 분류 방식은 기본적으로 OpenViews 라이브러리가 제공한다.

Class name:
Classification

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : RecordSet in association classify
<no rolename> : Statistics in association use
<no rolename> : ShapeColorModel in association represent

Attributes:

STDEV : int = 1
EQINT : int = 2

EQNUM : int = 3

Operations:

Name	Signature
Classification	Classification (boolean arg0)
getClassLimit	double getClassLimit (int arg0)
getClassNumber	int getClassNumber (double arg0)
getClassValue	int getClassValue (double arg0)
getInterval	int getInterval ()
setAllValues	void setAllValues (double[] arg0)
setInterval	void setInterval (int arg0)
setMethod	void setMethod (int arg0)

State machine: No
Concurrency: Sequential
Persistence: Transient

(4) ColorDescription

- 이 클래스는 기본적인 색상 정보를 표현한다.

Class name:
ColorDescription

Category: openview
External Documents:
Export Control: Implementation
Cardinality: n
Hierarchy:
Superclasses: none
Associations:

<no rolename> : ShapeColorModel
name : java::lang::String

Attributes:

r : int
g : int
b : int

Operations:

Name	Signature
ColorDescription	ColorDescription (String arg0, int arg1, int arg2, int arg3)

State machine: No
Concurrency: Sequential
Persistence: Transient

(5) ShapeColorModel

- 이 클래스는 ShapeObject의 연속 데이터 필드에 적용될 수 있는 색상 모델을 정의한다. ShapeColorModel을 일련의 ShapeObject에 적용하는 경우 자동으로 각 오브젝트의 심볼을 설정할 수 있다.

Class name:
ShapeColorModel

Category: openview
External Documents:
Export Control: Public
Cardinality: n
Hierarchy:
Superclasses: java::awt::image::IndexColorModel
Associations:

<no rolename> : ColorModelFactory in association make

<no rolename> : Classification in association represent
<no rolename> : ColorDescription

Attributes:

SINGLE : int = 1
BLACKWHITE : int = 2
ELEVATION : int = 3

Operations:

Name	Signature
ShapeColorModel	ShapeColorModel (int arg0, int arg1, byte[] arg2, int arg3, boolean arg4)
ShapeColorModel	ShapeColorModel (int arg0, int arg1, byte[] arg2, int arg3, boolean arg4, int arg5)
ShapeColorModel	ShapeColorModel (int arg0, int arg1, byte[] arg2, byte[] arg3, byte[] arg4)
ShapeColorModel	ShapeColorModel (int arg0, int arg1, byte[] arg2, byte[] arg3, byte[] arg4, int arg5)
ShapeColorModel	ShapeColorModel (int arg0, int arg1, byte[] arg2, byte[] arg3, byte[] arg4, byte[] arg5)
ShapeColorModel	ShapeColorModel (int arg0, int arg1, int[] arg2, int arg3, boolean arg4, int arg5, int arg6)

State machine: No
Concurrency: Sequential
Persistence: Transient

제 6 절 매핑 라이브러리 구현

가. 프로그래밍 가이드라인

- 프로그래밍 가이드라인은 실제 시스템의 구현에 있어서 여러 사람의 공동 작업을 도와주거나 코드 분석을 쉽게 할 수 있는 지침을 제공하는 데 그 목적이 있다. OpenViews 라이브러리는 코바/자바 기반의 매핑 라이브러리이므로 코바와 자바 코딩에 있어서의 지침과 시스템 구현시 IDL 설정의 공동 작업을 위한 지침을 기록한다.

(1) CORBA IDL 표기 지침

(가) 사용자 정의 항목의 이름

사용자 정의 항목(Token)은 명사구를 사용하되 단어의 첫 글자는 항상 대문자로 표기한다. 단, 예외 항목의 경우에는 반드시 "Exception"으로 끝나도록 한다. "_" 문자는 사용하지 않는다.

○ 사용자 정의 항목
interface, struct, union, enum, exception, typedef, const, …

○ 사용예
LoginManager, GeometryCollection, CommandNotFoundException, …

(나) 오퍼레이션 이름

오퍼레이션(Operation)의 이름은 동사구를 사용하되 첫 단어는 동사로 시작하도록 한다. 첫 단어의 첫 글자는 소문자로 하고, 두 번째 이하 단어들은 첫 글자를 대문자로 한다. "_" 문자는 사용하지 않는다.

○ 사용예
getUserInfo, makeGeometry, …

(다) 애트리뷰트 이름

애트리뷰트(Attribute) 이름은 명사구를 사용하되 단어의 첫 글자는 항상 소문자로 표기한다. 두 번째 이하 단어들은 첫 글자를 대문자로 한다. "_" 문자는 사용하지 않는다.

○ 사용예

userName, requestTime, …

(라) 모듈 이름

모듈(Module) 이름은 모두 소문자로 표기한다. "_" 문자는 사용하지 않는다.

○ 사용예

geometry, feature, …

(마) 주석

주석 표기는 자바 지침을 사용한다.

(2) Java 표기 지침

(가) 모듈 이름

모듈(Module) 이름은 모두 소문자로 표기한다. "_" 문자는 사용하지 않는다.

(나) 패키지 이름

패키지(Package) 이름은 모두 소문자로 표기한다. "_" 문자는 사용하지 않는다.

○ 사용예

package com.openworld.openview

(다) 클래스 이름

클래스(Class) 이름은 명사구를 사용한다. 첫 글자는 항상 대문자로 표기한다. 단, 예외(Exception) 클래스의 경우에는 반드시 이름이 "Exception" 으로 끝나도록 한다. 시스템 내부 커널에 해당하는 클래스를 제외하고는 "_" 문자를 사용하지 않는다.

○ 사용예

Geometry, Point, …

(라) 인터페이스 이름

인터페이스 이름은 형용사구를 사용한다. 첫 글자는 항상 대문자로 한다.

○ 사용예

Observable, Clonable, …

(마) 메소드 이름

메소드 이름은 동사구를 사용한다. 이름의 첫 단어는 동사로 시작하며, 모두 소문자로 표기한다. 두 번째 이하 단어들의 첫 글자는 대문자로 시작한다. "_" 문자는 사용하지 않는다.

○ 사용예

getUserInfo, makePoint, …

(바) 변수 이름

클래스 변수(Class variable)는 대문자로 시작하며, 인스턴스 변수(Instance variable)는 소문자로 시작한다. 두 번째 이하 단어들의 첫 글자는 대문자로 시작한다. "_" 문자는 사용할 수 있으나 권장하지 않는다.

○ 사용예

userId, userName, …

(사) 매개변수 이름

내부 변수를 위한 매개변수의 경우에는 동일한 단어를 피하기 위해 "_"

로 시작할 수 있다. 내부 변수와 충돌하는 경우가 아니면 변수 이름을 위한 표기지침을 이용한다.

○ 사용예

public void setName(_name)

(아) 상수 이름

상수(Constant) 이름은 명사구를 사용한다. 모든 글자는 항상 대문자를 사용한다. 단어간의 연결은 "_" 를 사용한다.

○ 사용예

COMMAND_OK, LINE_OFF

(자) 패키지 명명 순서

패키지 이름을 명명하는 순서는 역순으로 한다.

○ 사용예

com.openworld.openview

(차) 주석

javadoc 유틸리티를 위한 표기 지침을 준수한다.
다음은 그 표기 지침이다.

```
package com.openworld.openview; // 패키지 명명

import java.util.* // 사용 패키지 지정

/** 모든 주석의 시작은 "/**" 로 한다.
 * 클래스에 대한 전반적인 사항을 기록한다.
 * 필요에 따라 클래스 사용을 위한 간략한 샘플 소스를 포함할 수 있다.
 * 필수 기재 사항은 다음과 같다.
 * @author 작성자 이름
 * @version 버전 번호, 날짜
```

```
 *
 * 기타 기재 사항은 다음과 같다.
 * @see 관련 클래스 또는 메소드 기록
 * @since 최초 버전 번호
 */

public final ClassName extends ParentClassName implements InterfaceName
{

        /** 변수
         * 간략한 변수내용 정리
         */
        public double x;

        /** 메소드
         * 메소드의 내용 정리와 함께 용도에 대한 상세한 내용 정리
         * 필요에 따라 간략한 샘플 코드 기록 가능
         */
        public Point getPoint(Point arg) {

        }
}
```

나. OpenViews 라이브러리 기능 구현

- OpenViews 라이브러리를 이용한 애플리케이션 개발은 Jbuilder나 Visual Café와 같은 RAD 툴을 이용하여 UI를 개발하는 동시에 지도 처리 관련된 기능을 OpenViews에 연결시킬 경우 그리 많은 시간을 요하지 아니다.
- 이 단락에서는 단계별 애플리케이션 개발과 필요한 코드를 간략하게 살펴봄으로써 실제 구현된 라이브러리의 구조를 확인해 보고자 한다.

(1) 애플리케이션 개발 단계

- 다음은 가장 간단한 Shapefile 디스플레이부터 시작해서 복잡한 분류 기법에 이르기까지 다양한 OpenViews의 개발 과정을 설명한다.

O 1단계 : Shapefile 디스플레이

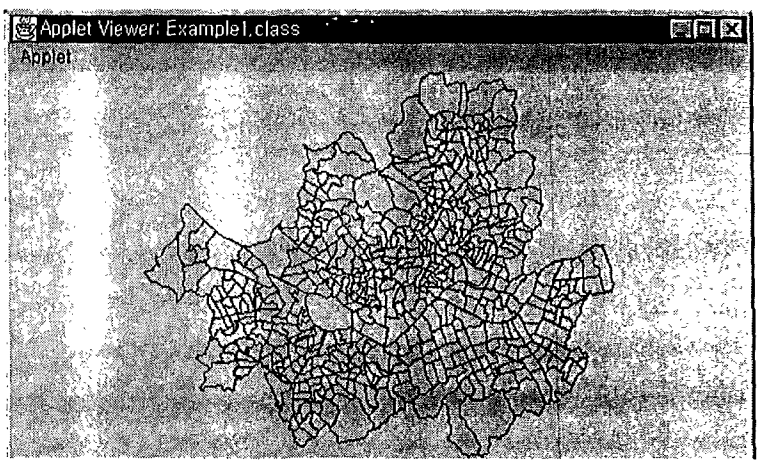
- 어떤 공간 데이터셋을 로딩하기 위해서는 먼저 그래픽 모델과 뷰를 초기화하는 작업이 필요하다. 그래픽 뷰는 그래픽 모델 객체를 파라미터로 받아 초기화할 수도 있고 뷰를 초기화한 후에 모델을 등록시킬 수도 있다

```
GraphicModel model = new GraphicModel();
GraphicView view = new GraphicView();
view.setGraphicModel(model);
```

- Shapefile을 디스플레이하기 위해서는 두 가지 방법을 사용할 수 있다. 우선 Shapefile이 OpenViews에서 사용하는 기본 공간 데이터 파일이므로 모델 클래스에 정의된 readShape() 메소드를 사용할 수 있다. *.shp 파일 안에 들어있는 공간 데이터만을 요구하는 경우에는 readShapeGeometry() 메소드를 사용할 수 있다.

```
String data_base = ".";
String data_name = "seoul_dong.shp";
URL data_url = new URL(data_base, data_name);
model.readShapeGeometry(data_url);
```

- [그림 39]은 서울의 동 데이터를 로딩한 결과를 보여주고 있다. 실행 상태는 애플릿이다.



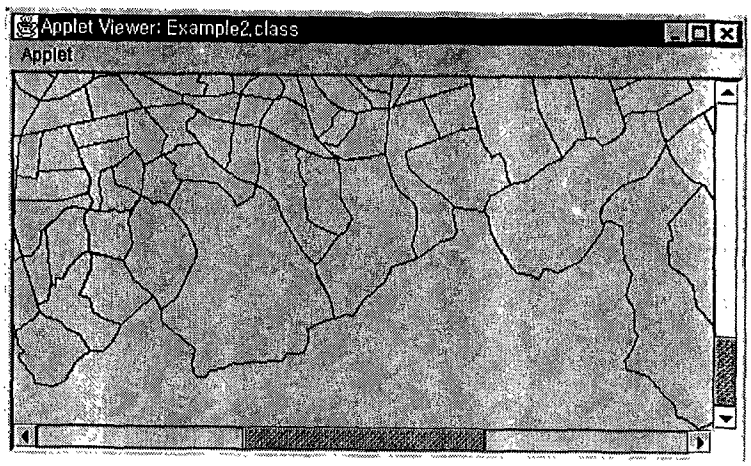
[그림 39] 서울 동 데이터 로딩

O 2단계 : 뷰 콘트롤

- 확대/축소와 같이 사용자 이벤트에 따라 뷰를 제어하는 기능은 콘트롤러가 가지고 있다. 콘트롤러는 크게 뷰를 제어하는 것과 오브젝트를 제어하는 것이 있는데 기본적인 기능은 뷰 콘트롤러만으로 충분하다. 오브젝트 단위의 제어는 오브젝트 콘트롤러를 사용한다.

```
GraphicViewController zoomin = new ZoomInController();
view.addController(zoomin);
```

- 위 코드는 확대 콘트롤러를 생성하고 이를 뷰에 등록하는 과정을 보여준다. 뷰와 콘트롤러의 관계는 1:다로 하나의 뷰가 여러 개의 콘트롤러를 가질 수 있으며, Chain of Responsibility 디자인 패턴을 사용한다. [그림 40]는 확대 콘트롤러를 등록한 후 관악구 지역을 중심으로 확대한 상태를 보여주고 있다.

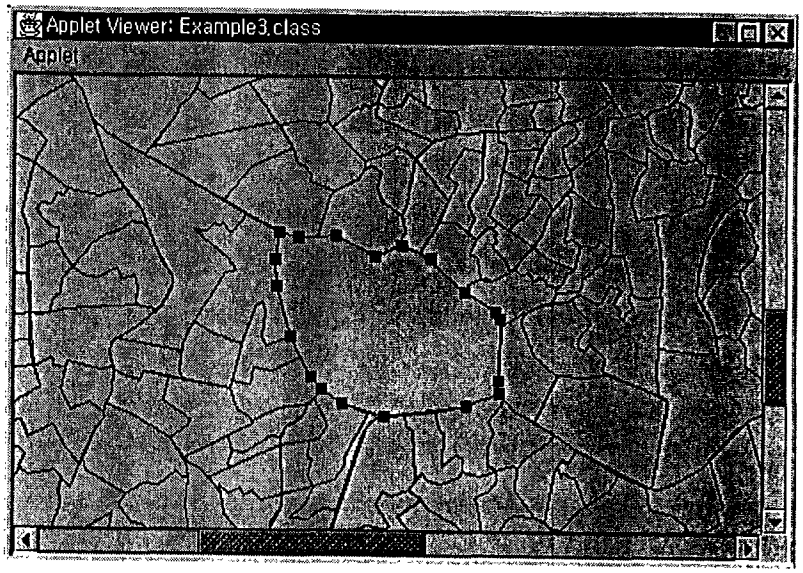


[그림 40] 확대 콘트롤러를 이용한 특정 지역의 확대

O 3단계 : 오브젝트 선택

- 공간 데이터와 함께 속성 데이터가 로딩되는 경우 특정 오브젝트가 가지고 있는 속성을 확인할 수 있는 메커니즘이 제공된다. 앞에서도 언급한 것처럼 이 기능은 GIS의 가장 기본적인 기능 중의 하나이다. [그림 41]은 특정 오브젝트를 선택했을 때 SelectionObject를 생성하고 그 오브젝

트의 상태를 디스플레이하는 것을 보여주고 있다. 가장 간단한 단계에서는 이 작업을 SelectionController가 대신해 준다.

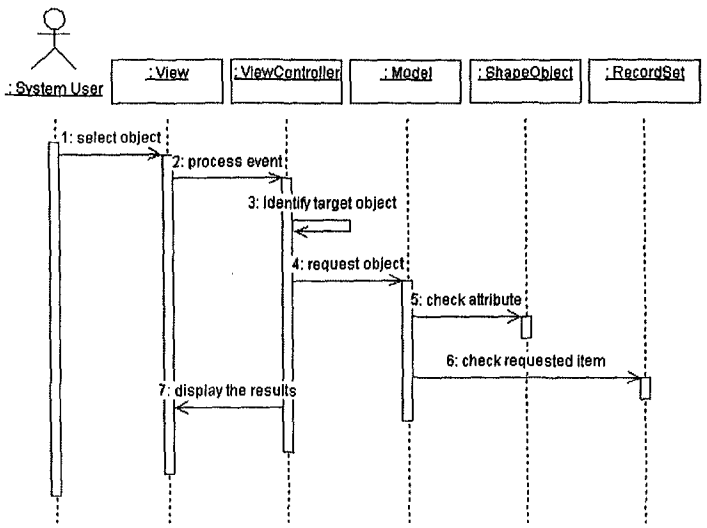


[그림 41] 여의도 오브젝트의 선택

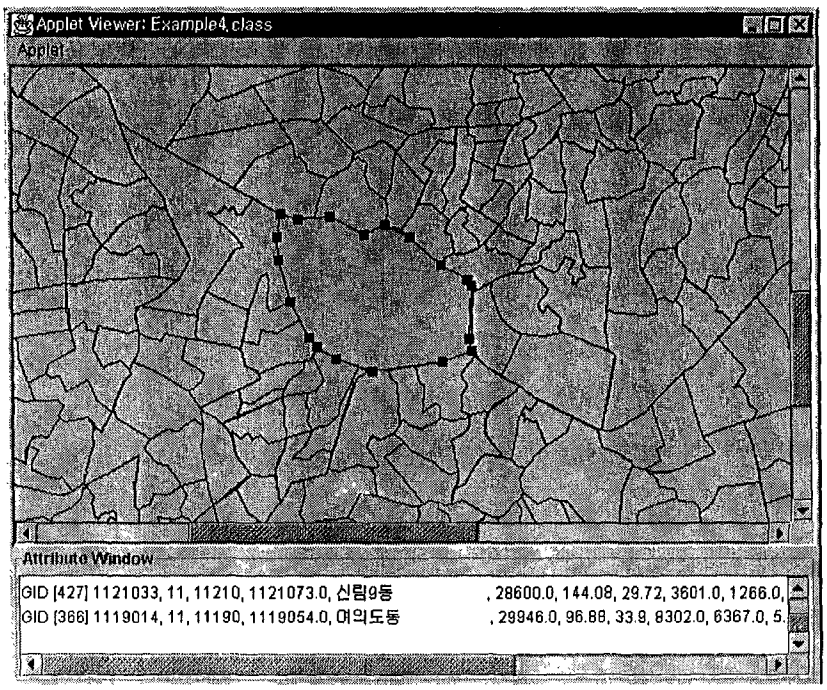
- 공간 데이터를 선택함으로써 속성 데이터를 확인할 수 있는 수단이 제공된다.

○ 4단계 : 오브젝트 선택

- 속성 데이터의 확인 과정은 [그림 42]에서 보는 것처럼 사용자의 이벤트를 전달받은 콘트롤러가 그 정보를 모델에 전달함으로써 가능하다.
- 먼저 시스템 사용자는 마우스 등의 디바이스를 이용하여 특정 오브젝트를 선택하고 그 중의 특정 필드값을 요구하는 이벤트를 발생시킨다. 이 이벤트는 뷰의 정보를 이용하며, 그 제어는 뷰 콘트롤러가 담당한다. 뷰 콘트롤러는 들어온 이벤트의 좌표값 등을 이용하여 특정 오브젝트의 정체를 확인하고 그 오브젝트를 모델로부터 요구한다. 획득한 오브젝트가 가지고 있는 속성들은 그 오브젝트의 ID 혹은 GID를 통해 RecordSet으로부터 확인될 수 있다. 그 값은 텍스트를 위한 패널이나 뷰에 뿌려질 수 있다.



[그림 42] 속성 확인 절차를 보여주는 시퀀스 다이어그램

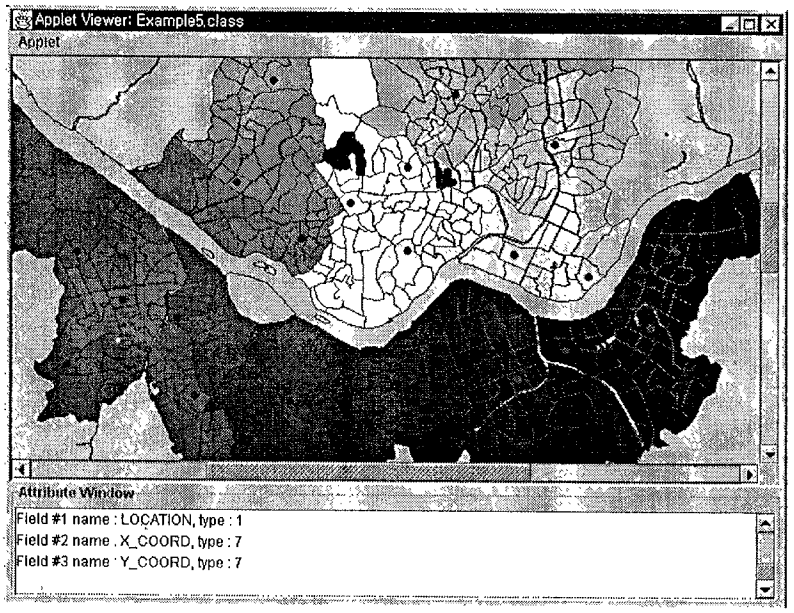


[그림 43] 오브젝트 속성 확인

- [그림 43]은 여의도 오브젝트를 선택하였을 때, 그 오브젝트가 가지고 있는 정보를 애트리뷰트 윈도우에 디버깅한 상태를 보여주고 있다.

O 5단계 : 멀티 레이어와 컬러 모델

- 하나의 뷰는 여러 개의 레이어를 표현할 수 있다. [그림 44]는 하나의 뷰에 서울의 동 경계, 수계망, 대기 오염 측정망 위치 자료 등 세 개의 레이어를 로딩한 결과를 보여주고 있다. 각각은 면, 면, 점 벡터를 이용하고 있으며, 따라서 서로다른 심볼을 사용할 수 있다. 현재 서울의 동을 표현하는 면은 서로 다른 색상을 가지고 있는데, 서울의 동이 가지고 있는 인구와 같이 연속 데이터를 기본으로 하는 아이템의 경우에는 컬러 모델을 이용할 수 있다. 현재 사용된 컬러 모델은 Black & White Color Model이다.



[그림 44] 컬러 모델의 사용

- 시스템에서 제공하는 컬러 모델을 사용하는 경우에는 다음과 같은 코드가 사용될 수 있다.

```
ColorModelFactory factory = new ColorModelFactory();
ColorModel color_model;
if (type == ShapeColorModel.SINGLE) {
        color_model = factory.makeSingle((byte)198, (byte)128, (byte)78);
} else if (type == ShapeColorModel.BLACKWHITE) {
        color_model = factory.makeBlackWhite();
```

```
} else if (type == ShapeColorModel.ELEVATION) {
        color_model = factory.makeElevation(128);
}
```

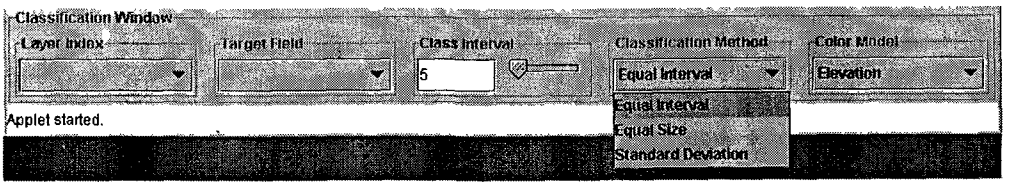
- 사용자는 원하는 컬러 모델을 확장, 정의하고 이를 실제 레이어에 적용할 수 있다.

O 6단계 : 분류

- 연속 데이터를 분류하는 기능은 심볼 설정에 있어 가장 중요한 기능 중의 하나이다. 그러나, 분류를 하는 과정에서 여러 가지 기법이 사용될 수 있으며, OpenViews 라이브러리는 크게 세 가지 분류 기법을 제공하고 있다. 사용자는 분류 기법, 클래스 간격 등을 설정할 수 있다.
- [그림 45]는 서울시 대기오염 자료를 이용하여 각 동의 대기오염 수치를 추정하고(추정 방법은 interpolation), 그 추정된 연속 데이터를 분류하여 각 동의 심볼을 설정한 결과를 보여주고 있다. 사용된 컬러 모델은 녹색과 파란색을 양끝값으로 하는 Elevation이다.

[그림 45] 서울시 대기오염 데이터의 추정과 분류

- 자료의 분류를 위한 인터페이스는 다음과 같이 작성될 수 있다.



[그림 46] 자료의 분류를 위한 인터페이스 작성예

(2) 심볼 설정

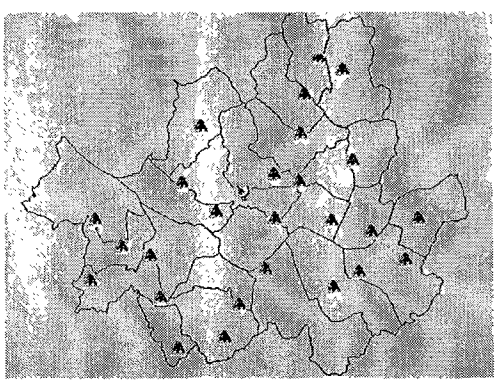
- 주어진 공간 데이터와 속성값을 이용하여 디스플레이하는 과정에 사용되는 심볼은 OpenViews에서 다양하게 정의, 사용될 수 있다. OpenViews에서는 자바 2D 라이브러리에서 제공하는 기능들을 그대로 사용할 수 있도록 심볼 모델을 정의하였으며, 이를 이용하여 사용자가 무한대로 확장 가능하다.

O 점 심볼

- PointShape 오브젝트에 설정되는 시스템 정의 점 심볼은 크게 PointMarker와 PointSymbol이 있다. 아래 코드는 이미지를 이용하는 PointMarker의 사용예를 보여주고 있다.

```
Image img = Toolkit.getDefaultToolkit().getImage(img_name);
MarkerSymbol symbol = new PointMarker(img, 15);
view.setLayerSymbol(layer_index, symbol, true, true);
```

- [그림 47]은 서울시 대기오염 측정망 위치에 필요한 이미지를 심볼로 정의하여 설정한 결과를 보여주고 있다.

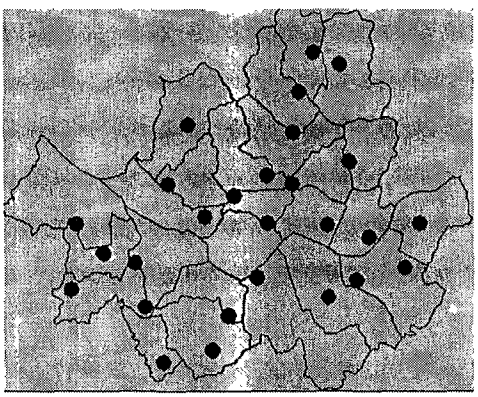


[그림 47] 서울시 대기오염 측정망 위치를 표현하는 심볼

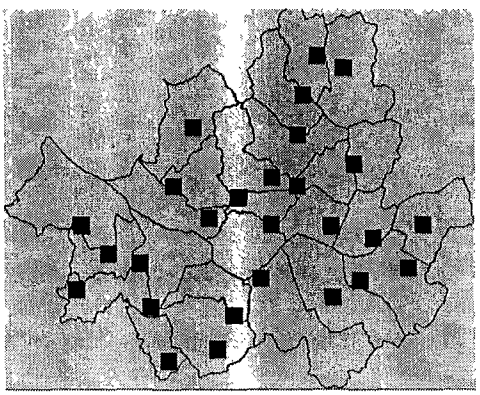
- PointSymbol 중에서 기본적으로 사용되는 모양들은 시스템에서 제공할 수 있다. 다음 코드는 원형의 심볼을 설정하는 방법을 보여준다.

```
symbol = new PointSymbol(PointSymbol.TYPE_CIRCLE, 8, fcolor, bcolor);
```

- [그림 48, 49]는 PointSymbol이 디스플레이된 상태를 보여준다. 이 심볼의 크기나 색상은 사용자가 조절할 수 있다.



[그림 48] 원형 심볼의 설정예



[그림 49] 사각형 심볼의 설정예

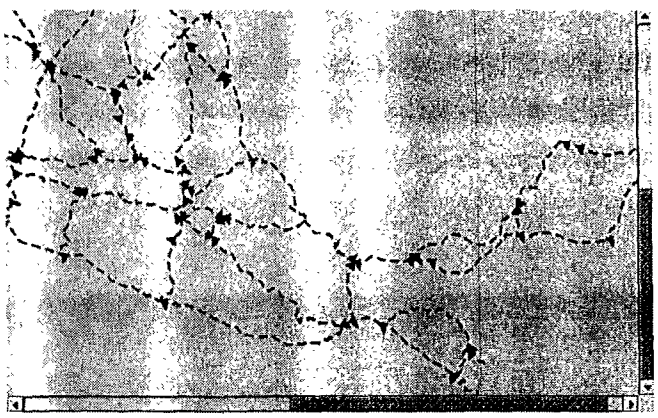
O 선 심볼

- 선형 객체는 크게 두 개의 점으로 표현되는 단순 순과 여러 개의 선형 세그먼트로 구성되는 LineString 이 있다. 이러한 선형 객체는 선 심볼을 이용하여 디스플레이 방식을 결정할 수 있으며, 사용자는 자신의 입맛에 맞는 심볼을 설정하여 제공할 수 있다. 예를 들어, 2차선 도로를 표현하는

심볼이 사용되는 기관마다 다를 경우 각 기관의 선호에 맞는 심볼을 정의하여 설정할 수 있다.

- [그림 50]은 LineShape에 설정된 심볼을 보여주고 있다. 이 심볼은 다음 코드에서 보이는 바와 같이 특정한 방향성을 화살표로 보여주는 ArrowModel을 내부적으로 가지고 있다.

```
TextureModel texture;
StrokeModel stroke;
ArrowModel arrow;
stroke = new StrokeModel(StrokeModel.TYPE_DEFAULT, 2.0F);
symbol.addSymbolModel(stroke);
texture = new LineTextureModel(TextureModel.TYPE_CROSS);
symbol.addSymbolModel(ltexture);
arrow = new ArrowModel(ArrowModel.TYPE_SOFT);
symbol.addSymbolModel(arrow_model);
```



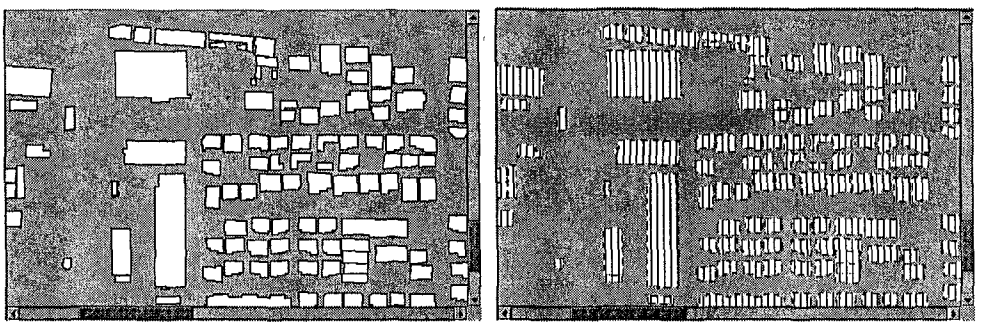
[그림 50] 선형 심볼과 **ArrowModel**

O 면 심볼

- 면 심볼은 앞에서 언급한 각종 심볼 모델들을 전부 사용하는 가장 복잡한 벡터 객체라고 할 수 있다. 다음 코드는 폴리곤에 적용되는 심볼을 초기화하는 과정을 보여준다.

```
symbol = new PolygonSymbol(fcolor, bcolor, false, 0.0D, true, false);
```

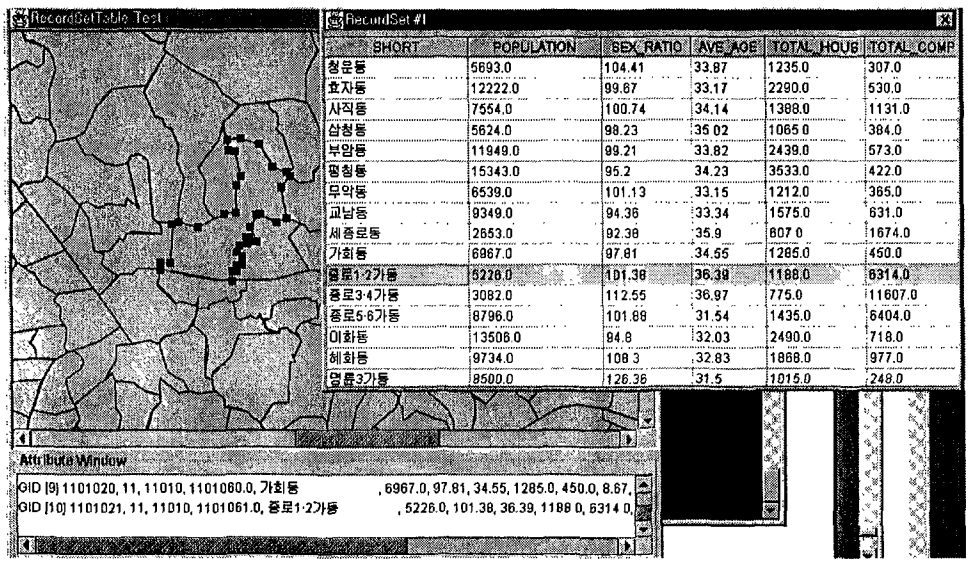
- [그림 51]은 폴리곤 경계만을 표현하는 경우와 내부 hatch를 사용하는 경우를 보여주고 있다. 내부 hatch의 경우에는 사용자가 확장, 정의할 수 있는 메커니즘과 함께 hatch 선간의 간격, 폭 등을 지정할 수 있도록 되어 있다.



[그림 51] 폴리곤 심볼의 설정예

(3) 속성 디스플레이

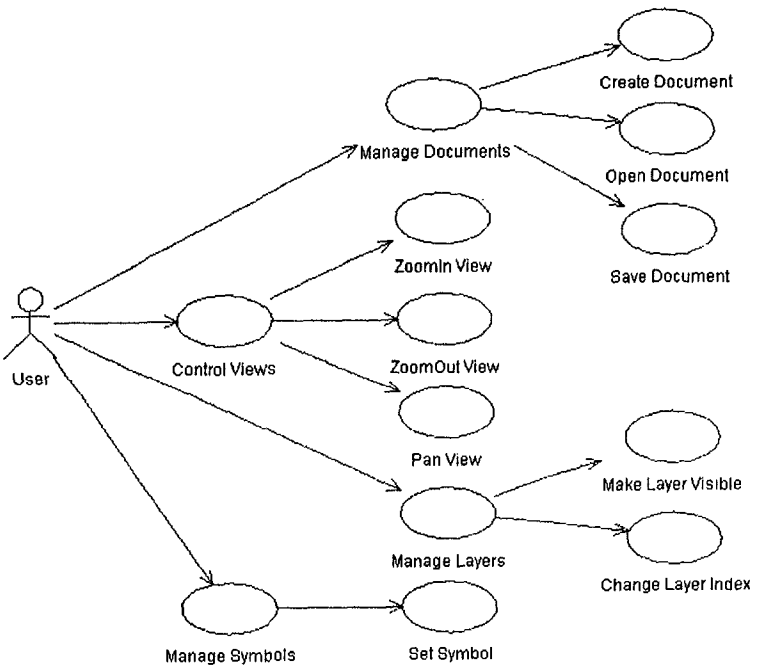
- 공간 데이터와 해당 속성의 링크 및 디스플레이를 위해 사용자는 테이블 위젯을 사용할 수 있다. [그림 52]는 로딩된 레이어와 그 레이어에 등록된 RecordSet을 이용하여 테이블 형태로 디스플레이하고 실제 공간 데이터 오브젝트와 속성을 링크하는 방법을 보여주고 있다.



[그림 52] 공간 데이터 오브젝트와 속성의 링크

(4) 애플리케이션 개발

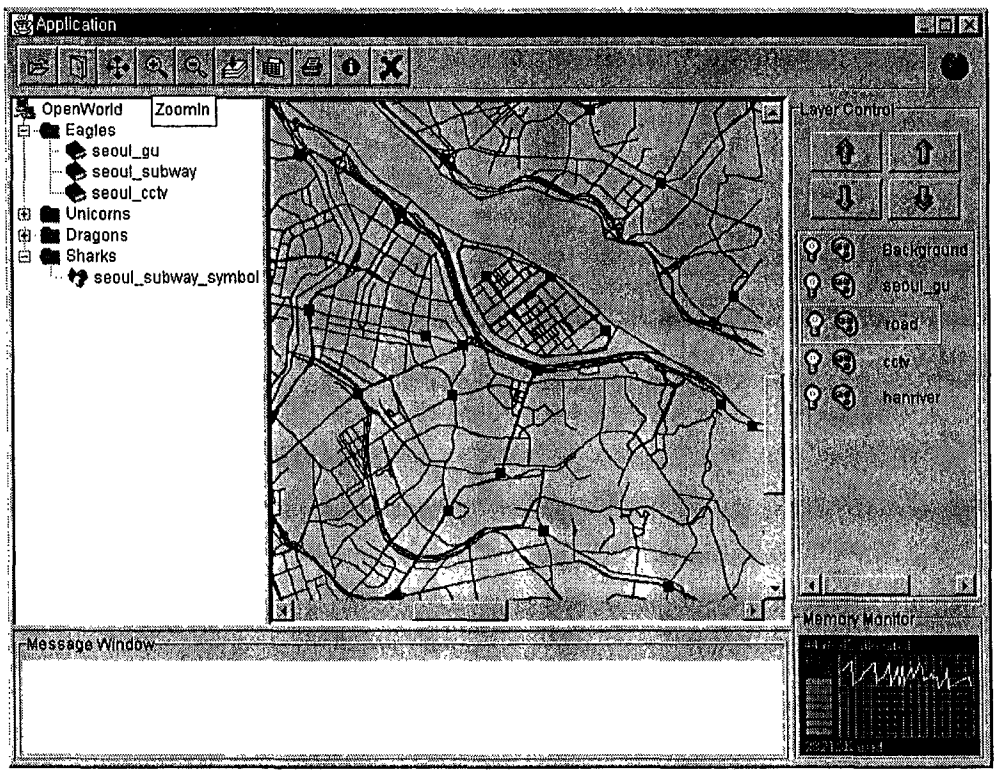
- OpenViews를 이용하면 지도 뷰어 애플리케이션을 간단하게 만들 수 있다. 애플리케이션은 크게 문서 생성, 열기, 쓰기와 같은 문서 관리 기능, 확대/축소/이동과 같은 뷰 콘트롤 기능, 레이어 보이기/감추기나 레이어 인덱스 관리와 같은 레이어 관리, 심볼 설정을 위한 심볼 관리 등의 기본 기능을 가지고 있어야 한다. [그림 53]은 애플리케이션 개발에 있어 필요한 기본 기능들을 정의한 유스케이스이다.



[그림 53] 기본적인 애플리케이션 유스케이스

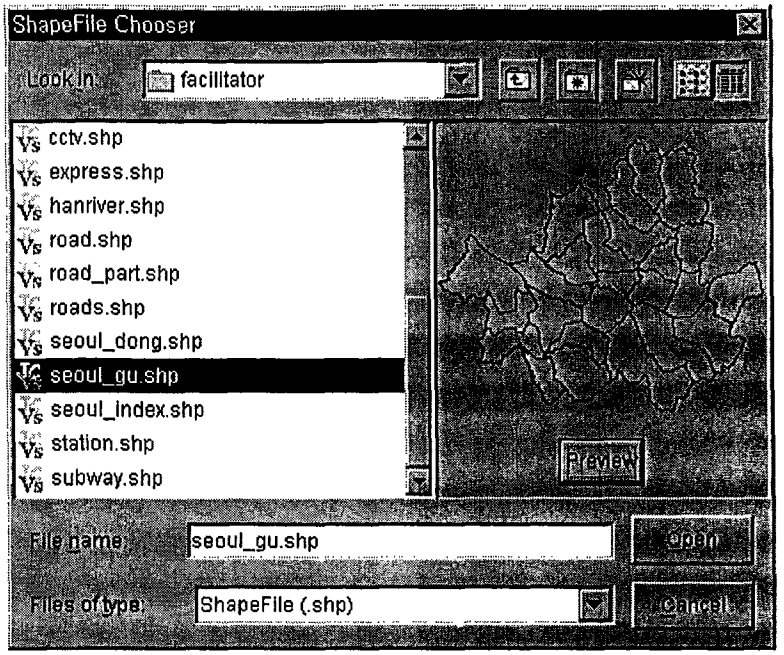
- [그림 54]는 구현된 기본 애플리케이션의 사용자 인터페이스를 보여주고 있다. 이 인터페이스는 사용자 기호에 따라 바뀔 수 있으나, 기본적인 유스케이스를 만족한다.
- 뷰에는 네 개의 레이어가 로딩되어 있으며, 모두 간단한 심볼로 설정되어 있다. 로딩된 레이어는 서울 동 경계 레이어, 서울 도로망 레이어, 서울 CCTV 위치 레이어, 한강 레이어이며, 모두 전경색(foreground color)을 노란색으로, 선의 색을 검은색으로 하였다. 상위에 있는 패널은 각종 기능 버튼이 들어있는 콘트롤 패널이며, 메인 패널의 왼쪽에는 트리뷰, 가운데에는 그래픽 뷰, 오른쪽에는 레이어 콘트롤 뷰로 구성되어 있다.

- 트리뷰(treeview)는 로딩할 레이어의 현황을 보여주는 기능을 하는 것으로, 로컬 또는 원격지의 데이터셋을 모두 로딩할 수 있는 브라우저 역할을 담당한다. 그래픽 뷰는 특정 대상 지역의 지도 디스플레이를 담당하며, 오른쪽의 레이어 콘트롤 패널(layer control panel)은 레이어 인덱스 관리, 심볼 관리 등의 역할을 담당한다. 메시지 윈도우(message window)에는 애플리케이션 운영에 있어 발생되는 예외, 로그 등이 출력된다.



[그림 54] 기본 애플리케이션 사용자 인터페이스

- [그림 55]는 로컬 데이터셋을 로딩하는 경우 필요한 파일 브라우저를 보여준다
- 파일 브라우저에서는 Shapefile과 같은 공간 데이터셋을 로딩할 수 있으며, 미리보기(preview) 기능이 있어 데이터를 로딩하기 전에 미리 볼 수 있도록 한다.



[그림 55] Shape file을 로딩하기 위한 파일 브라우저

- [그림 56]은 도로망과 관련된 속성 데이터 테이블을 보여준다.

RecordSet #4

ROAD_NAM	ROAD_ID	PAVE_COD	FEB_COD
도봉로	3	1.0	2.0
	0	1.0	2.0
도봉로	3	1.0	2.0
	0	1.0	2.0
	0	1.0	2.0
	0	1.0	2.0
	0	1.0	2.0
동부간선도로	0	1.0	2.0
동부간선도로	0	1.0	2.0
동1로	0	1.0	2.0
도봉로	3	1.0	2.0

[그림 56] 도로망 속성 테이블

제 7 절 OpenWorld 시스템 설계

1. SFCORBA 명세 구현

- OpenWorld 는 오브젝트 웹을 목표로 하는 코바/자바 기반의 개방형 GIS 시스템으로 설계된다. 따라서, 기본적으로 분산 컴퓨팅 환경을 고려해야 하며, 공간 데이터의 상호운용성과 관련된 표준으로 대두되고 있는 OpenGIS 명세에 준하는 시스템이어야 한다.
- OpenGIS의 구현 사양 중에서 코바 기반 사양은 1999년말 현재 버전 1.0이 배포되고 있으며, 이 문서를 통해 관련 IDL과 함께 구현에 필요한 기본 개념들을 공유하고 있다. 따라서, OpenGIS 표준에 준하는 방식으로 데이터셋을 보내고 받기 위해서는 SFCORBA(OpenGIS Implementation Specification for CORBA)를 구현하는 것이 요구된다.
- OpenGIS에서 제공하는 사양은 Feature와 Geometry에 관련된 것으로 커버리지 또는 레이어 단위의 데이터 획득은 OpenGIS 사양에 맞추어 처리하기 어렵다 따라서, 현 상태에서는 기본적인 공간 객체는 OpenGIS의 사양을 따르고 커버리지와 관련된 정보는 새로 정의된 GeoDataSet 모델을 따르는 것으로 하는 것이 바람직하다.
- CORBA로 분산 애플리케이션을 개발할 때, 가장 먼저 해야하는 작업은 CORBA IDL을 정의하는 것이다. OpenGIS 명세에서 제공하는 IDL을 사용하면 공간 객체들의 유형, 그 객체들의 속성과 메소드, 시스템 운영시 발생할 수 있는 예외상황 등을 알 수 있다. IDL정의 이후에 진행되는 작업을 간략히 정리하면 다음과 같다.

  - O 전처리 **(precompile)**

    이 과정에서는 C/C++이나 자바와 같은 특정 언어로 작성된 스텁(stub)들이 자동으로 생성된다. 서버 객체를 위한 스켈레톤(server skeleton)은 구현 클래스를 작성하기 위한 기본 토대가 되며, 클라이언트 객체의 요청을 받아 실제 서버에 전달하는 역할을 한다. 클라이언트 객체를 위한 스텁(client stub)은 스켈레톤과 역할이 상반되는 것으로 클라이언트가 서버에게 서비스를 요청하기 위해 호출한다. 클라이언트가 요청한 메소드의 이름과 그 매개변수들을 네트워크 스트림의 형태로 보낼 수 있도록 변환하는 과정을 거친다.
  - O 서버 객체 구현 **(server object implementation)**

    서버 객체는 자동으로 생성된 스켈레톤 클래스를 기반으로 하며, 주요 서

버 기능을 담당한다.

- Feature 모듈은 공간 객체와 그것들의 집합체를 생성, 접근, 질의할 수 있는 기능을 제공한다. 이 모듈에서 가장 중요한 인터페이스인 Feature는 공간적, 비공간적 속성값을 가지고 있는 실세계의 객체를 표현한다. 예를 들어, 도로 객체는 공간적 속성으로 선형 데이터를, 비공간적 속성으로 이름, 차선 수, 길이 등을 가질 수 있다. Feature 인터페이스는 클라이언트가 Feature 객체에 접근할 수 있는 메커니즘을 제공한다. 예를 들어, 클라이언트는 Feature 객체의 공간적 속성값을 검색하기 위해 get_geometry() 메소드를 호출한다.
- Feature 구현 클래스가 관계형 데이터베이스에 들어있는 공간 데이터와 매핑되는 경우에는 하나의 Feature가 하나의 레코드를 나타낸다. FeatureType 인터페이스는 클라이언트가 객체들의 유형에 대한 세부사항들에 접근할 수 있는 메커니즘을 제공한다. 예를 들어, 도로 사상은 "ID", "NAME", "LENGTH"와 같은 속성의 이름과 정수, 문자열, 실수와 같은 유형을 가진다.
- 모든 Feature 객체들은 기본적으로 하나의 FeatureType 객체를 가지고 있으며, FeatureCollection은 Feature의 집합체를 위한 인터페이스이다. FeatureCollection은 Feature로부터 상속을 받는 composite 디자인 패턴을 사용하므로 어떤 Feature 객체는 Feature 객체 하나일 수도 있고, 여러 Feature 객체들이 모인 집합체일 수도 있다.
- 일반적으로 Feature들의 집합체는 다음과 같은 두 가지 경우에 생성될 수 있다.

  ○ **Persistency** 유지

  Feature들의 집합체로 정의하는 경우에는 데이터베이스에 저장된 객체와 같이 지속성을 유지시킬 수 있다. 즉, Feature를 삭제하면 그 Feature는 영구적으로 없어질 수 있다. 이러한 기능은 FeatureCollection으로부터 상속받은 ContainerFeatureCollection이 제공한다. 따라서, 이 객체는 개념적으로 공간 데이터베이스와 일치한다.

  ○ 클라이언트의 질의 결과

  Feature들의 집합체는 클라이언트의 질의 결과로 임시 생성될 수 있다. 이 경우에 집합체는 Feature들을 소유하는 것이 아니라 단순히 참조하고 있다. 클라이언트들은 언제나 FeatureCollection 객체를 통해 Feature 객체에 접근하기 때문에 분산된 시스템들이 정보를 주고받는 기본 단위가 된다.

- OpenGIS의 Feature Geometry 추상 사양에서는 Geometry를 좌표값을 갖는 기하 정보와 이것을 실세계와 연결짓는 공간 참조 체계(Spatial Reference System)의 결합이라고 정의한다. 모든 공간 객체들은 공간적 범위, 공간 참조 체계와 같은 많은 공통적인 특성을 가진다.
- Geometry 인터페이스는 Geometry 모듈의 최상위 인터페이스이며, 나머지 인터페이스는 차원(dimension)에 의해 구분된다. Geometry 객체는 단 하나의 기하정보로 이루어질 수도 있고 동일한 유형의 기하정보를 가진 집합체로 구성될 수도 있다.
- 본 과제에서는 공간 데이터를 네트워크 상에서 상호 교환하기 위한 기본 단위로 OpenGIS에서 제시한 WKSGeometry를 사용한다. WKSGeometry는 공간 좌표 체계와는 독립적으로 실제 연산에 사용되는 공간 데이터의 유형들을 정의한 것이다. Geometry 모듈에는 Envelope, WKSPoint, WKSRing, WKSLineString 등이 있다.
- 다음은 기본적인 WKSGeometry 관련 IDL을 보여주고 있다.

```
struct WKSPoint {
    double x;
    double y;
};

typedef sequence<WKSPoint> WKSPointSeq;
typedef sequence<WKSPoint> WKSLineString;
typedef sequence<WKSLineString> WKSLineStringSeq;
typedef sequence<WKSPoint> WKSLinearRing;
typedef sequence<WKSLinearRing> WKSLinearRingSeq;

struct WKSLinearPolygon {
    WKSLinearRing externalBoundary;
    WKSLinearRingSeq internalBoundaries;
};

typedef sequence <WKSLinearPolygon> WKSLinearPolygonSeq;

enum WKSType {
    WKSPointType,WKSMultiPointType, WKSLineStringType,WKSMultiLineStringType,
    WKSLinearRingType, WKSLinearPolygonType, WKSMultiLinearPolygonType,
```

```
    WKSCollectionType
};

union WKSGeometry
    switch (WKSType) {
        case WKSPointType:
            WKSPoint point;
        case WKSMultiPointType:
            WKSPointSeq multi_point;
        case WKSLineStringType:
            WKSLineString line_string;
        case WKSMultiLineStringType:
            WKSLineStringSeq multi_line_string;
        case WKSLinearRingType:
            WKSLinearRing linear_ring;
        case WKSLinearPolygonType:
            WKSLinearPolygon linear_polygon;
        case WKSMultiLinearPolygonType:
            WKSLinearPolygonSeq multi_linear_polygon;
        case WKSCollectionType:
            sequence<WKSGeometry> collection;
};

struct Envelope {
    WKSPoint minm;
    WKSPoint maxm;
};
```

- OpenGIS 사양에서 제공하지 못하는 커버리지 기능을 제공하기 위하여 OpenGeoDataSet 인터페이스를 정의할 수 있다. 아래 코드는 본 과제에서 분산 애플리케이션 개발을 위하여 정의한 구조체이다.

```
struct OpenGeoDataSet {
    wstring name;
    OGIS::WKSType wks_type;
    OGIS::Envelope envelope;
```

```
        OpenPropertyTypeSeq property_type;
        OpenFeatureDataSeq feature_data;
    };
```

- 이 구조체를 통해 하나의 레이어에 포함된 객체들을 가져올 수 있다.

2. GIS 애플리케이션 서버 설계

가. GIS 애플리케이션 서버의 개념

- 현재 C/S 환경에서 분산 환경으로 시스템을 적응시키기 위해 CORBA/COM 기술이 많이 사용되고 있으나, 단순한 2-레이어 시스템은 사용가능한 서비스(functional server)의 수가 증가하는 데 대해 유연하게 대처하기 어렵다. 이러한 측면은 글의 앞부분에서 제시된 요구사항을 수렴하기 어렵다는 것을 의미하므로, 중계자(mediator)를 이용한 새로운 대안이 필요하다[7]. GIS 애플리케이션 서버는 네트워크 상에 존재하는 데이터 제공자와 서비스 제공자에 대한 정보를 제공하는 것은 물론, 매핑을 통해 클라이언트가 원하는 유형의 지도를 출판하는 기능을 하는 중계자라고 정의될 수 있다.
- 현재 애플리케이션 서버(application server)라는 용어는 다양한 의미로 사용되고 있다. 서버의 확장 모듈(server extension), 미들웨어 서버(middleware server), 오브젝트 캐쉬(object cache), 객체 서버(object server) 등의 의미로 사용되는 애플리케이션 서버는 크게 웹 정보 서버(Web Information Server), 수동적 컴포넌트 서버(Passive Component Server), 능동적 애플리케이션 서버(Active Application Server)로 구분될 수 있다[8]. 웹 정보 서버는 주로 브라우저의 데이터베이스에 대한 접근과 웹 출판(web publishing)에 관심을 두며, 수동적 컴포넌트 서버는 데이터 접근과 컴포넌트에 대한 런타임 프레임워크를 제공하는 데 초점을 둔다. 이에 반해서 능동적 애플리케이션 서버는 백-오피스(back-office) 애플리케이션과 새로운 상위 레벨의 로직을 위한 프레임워크에 관심을 둔다.
- GIS 애플리케이션 서버가 오브젝트 웹 환경에서 효율적으로 운용될 수 있도록 하려면, 실제로 stateless한 현재의 웹 서비스를 stateful하게 만들고 오브젝트간 협력을 원활히 만드는 메커니즘의 정의가 필요하므로 GIS 애플리케이션 서버 아키텍쳐는 이러한 특징들을 수용할 수 있어야 한다. 따라서, GIS 애플리케이션 서버는 다음과 같은 특징을 포함한다.

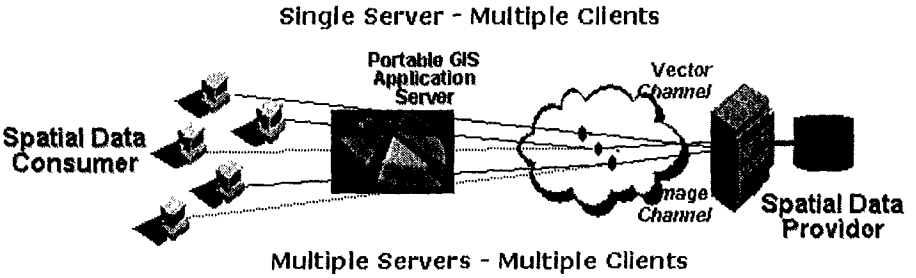
O 능동성(Active GIS Application Server)

공간 데이터 처리와 관련된 복잡한 로직들을 모두 수용할 수 있어야 한다. 예를 들어, 공간 통계를 이용한 질병 데이터의 분석 및 디스플레이가 필요할 경우 GIS 애플리케이션 서버는 공간 통계와 관련된 기능을 매핑 작업 이전에 수행할 수 있어야 한다. 현재 대부분의 설계 시스템들은 이와 같은 기능을 고려하지 않고 있다.

O 이식성(Portable GIS Application Server)

multi-tier 환경에서 운용되는 GIS 애플리케이션 서버는 네트워크 상의 어떤 노드에서도 운용 가능해야 한다. 따라서, 클라이언트 오브젝트, 미들웨어 오브젝트, 서버 오브젝트가 모두 수용할 수 있어야 한다.

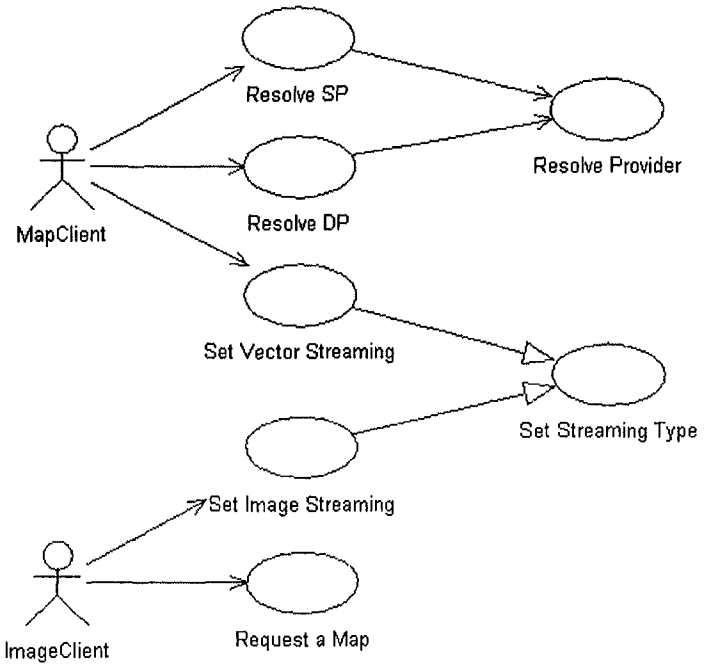
- [그림 57]은 GIS 애플리케이션 서버의 이식성을 이용한 시스템 배치의 유연성을 표현한 것이다. GIS 애플리케이션 서버는 클라이언트부터 서버 오브젝트에 이르기까지 컴포넌트에 관한 정보를 알고 있는 오브젝트에 의해 아무런 수정 없이 사용될 수 있다.



[그림 57] 포터블 GIS 애플리케이션 서버

나. 동적 지도 생성

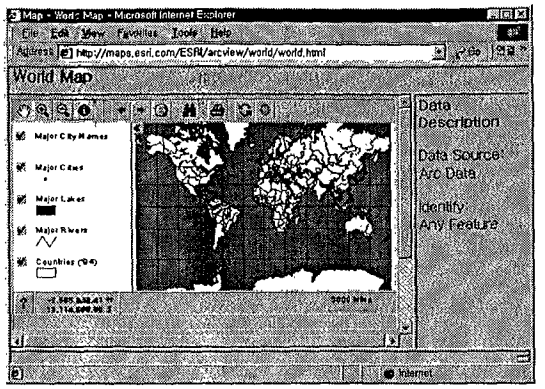
- [그림 58]은 전자 지도 출판과 관련된 GIS 애플리케이션 서버의 유스케이스 다이어그램이다. 분산 환경에서 운영되는 애플리케이션 서버는 다양한 요구사항을 고려해야 하지만 분산된 오브젝트간의 인터액션과 관련된 가장 기본적인 업무는 그림과 같이 정리될 수 있다.

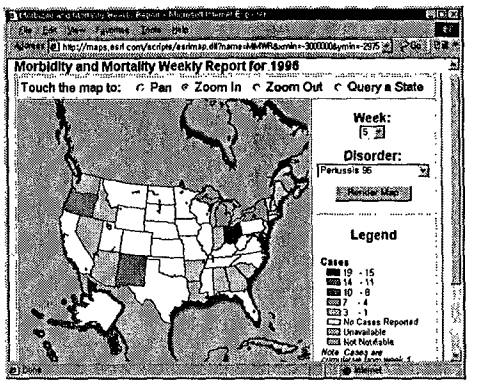


[그림 58] GIS 애플리케이션 서버의 유스케이스

- MapClient는 벡터 객체 기반의 클라이언트를, ImageClient는 이미지 기반의 클라이언트를 의미한다. 클라이언트의 종류에 따라서 애플리케이션 서버는 서로다른 기능을 갖추고 있어야 한다. 예를 들어, 벡터 객체 기반 클라이언트의 요구는 공간 객체들을 직접 요구하는 것이므로 도로망 데이터 객체를 그대로 클라이언트까지 전달할 필요가 생긴다. 분산된 오브젝트간의 인터액션을 원활하게 하기 위해서는 데이터 제공자와 심볼 제공자[13]를 위치에 관계없이 찾아낼 수 있는 방법을 제공해야 한다.
- GIS 애플리케이션 서버의 설계에 있어 가장 크게 고려해야 할 사항은 다수의 클라이언트가 요구하는 사항들을 효율적으로 처리할 수 있어야 한다는 점이다. 이는 동적으로 로딩되는 지리공간 데이터의 양이 일반 데이터보다 훨씬 크다는 점에서 더욱 중요하다. 웹 상에서 지리정보를 서비스하는 기존의 시스템들은 주로 미리 정의된 데이터와 심볼들을 이용하므로 데이터의 동적인 로딩이 그리 필요하지 않다. 그러나, 능동적이고 효율적인 GIS 애플리케이션 서버를 위해서는 동적인 지도 생성과 서비스가 필요하므로 성능 개선이 중요해진다.

[13] 본 과제에서는 심볼 객체를 분산시키는 방법을 사용하고 있다.

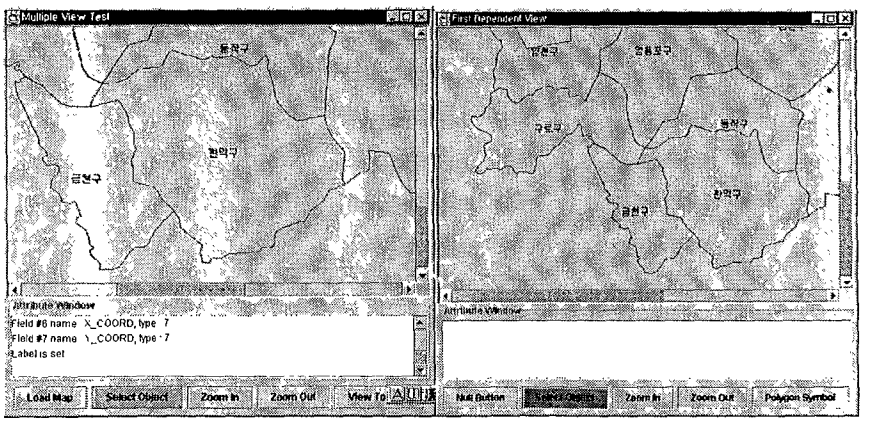




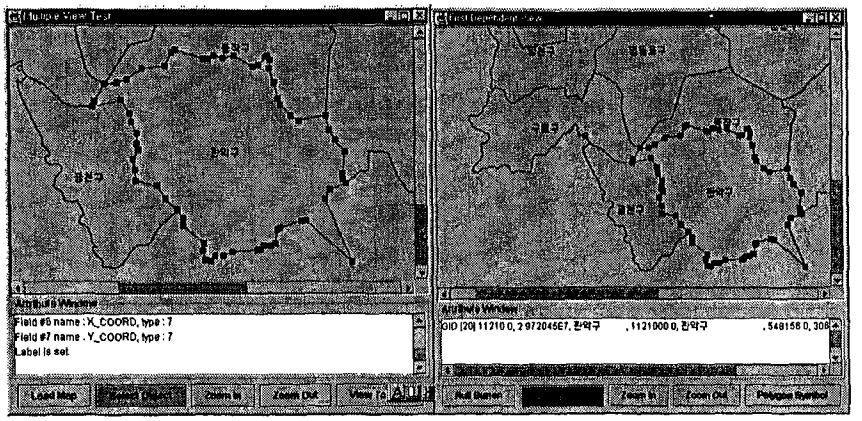
[그림 59] ESRI사의 지도 서비스 (미리정의된 데이터와 심볼을 이용)

다. 멀티 뷰 응용

- 앞에서도 언급한 바와 같이 멀티 뷰는 하나의 공유된 공간 데이터를 이용하여 서로 다른 사용자에게 동일 혹은 상이한 뷰를 보여주는 기술로 데이터 공유를 위한 기반 기술이다[9]. [그림 60]과 [그림 61]은 서울 구 경계를 담고 있는 공간 데이터를 로딩한 후에 두 개의 서로 다른 뷰로 디스플레이한 것을 보여주고 있다.
- 이 두개의 뷰는 하나의 데이터를 공유하고 있으므로 실제 담고 있는 객체들은 동일하다. 그러나, 확대/축소를 하는 과정이나 모양의 변화에 있어서는 다른 방식으로 보여줄 수 있다. 이 시스템에서는 확대/축소 과정이 뷰마다 독립적으로 운영되며, 객체를 선택하고 편집하는 경우에는 두 개의 뷰가 동일한 상황을 지켜볼 수도 있다.



[그림 60] 두 개의 뷰가 서로 다른 방식으로 확대/축소하는 경우

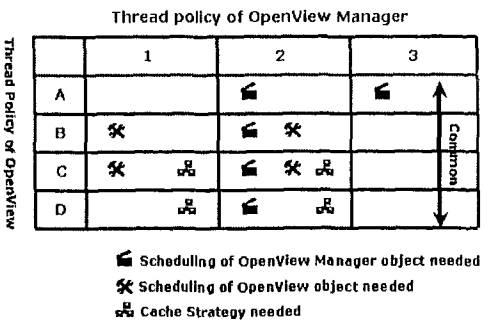


[그림 61] 두 개의 뷰가 똑같이 편집되는 경우

- 클라이언트 요구를 효과적으로 처리할 수 있는 방법으로 사용되는 오브젝트 캐쉬(Object Cache)는 보통 타일 단위로 레이어의 데이터를 가져와 처리하면서 데이터의 재사용성을 높이는 데 반해 멀티 뷰는 이미 가져온 데이터를 여러 뷰가 공유하면서 데이터의 재사용성을 높인다는 점에서 두 가지 기법이 동시에 사용되는 것이 성능 향상의 측면에서 바람직하다고 할 수 있다.

라. 오브젝트 배치

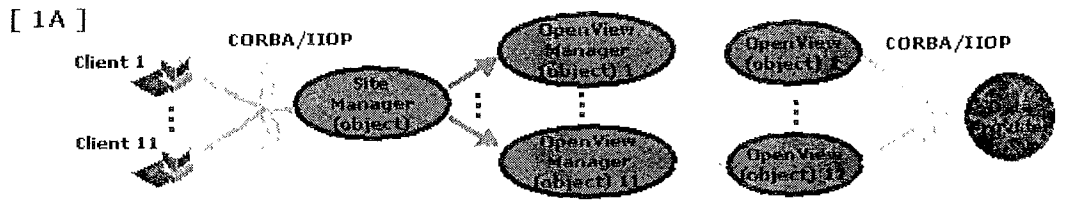
- 분산 환경에서의 오브젝트 배치는 애플리케이션 서버의 설계 과정뿐만 아니라 설계 및 구현된 시스템의 운영과정에서의 성능 개선을 위해서도 중요하다. 오브젝트의 효율적인 배치 상태를 고안하기 위해서는 구성 오브젝트의 확인과 함께 그것들 간의 인터액션에 관한 정보를 알고 있어야 한다.
- [그림 62]는 오브젝트 배치에 있어 고려해야 할 사항에 관한 매트릭스를 보여준다.



[그림 62] 오브젝트 배치 고려사항에 관한 매트릭스

- 애플리케이션 서버가 이미지 클라이언트와 함께 연동하여 운영되는 경우에는 OpenViews 매핑 커널이 애플리케이션 서버의 구성 컴포넌트로 동작한다. 매트릭스의 가로축은 OpenViews 매니저가 취할 수 있는 쓰레드 정책을 나열하고 있으며, 세로축은 OpenViews 자체의 쓰레드 정책을 나열하고 있다. OpenViews 매니저의 쓰레드 정책은 1. Thread per Request, 2. Thread Pool, 3. Shared Object 로 크게 분류할 수 있으며, OpenViews의 쓰레드 정책은 OpenViews 자체를 하나의 오브젝트로 취급하는 방법과 그 외에 D. 멀티 뷰를 이용하는 방법이 있을 수 있다. 하나의 오브젝트로 취급하는 경우에는 A. Thread per Request, B. Thread Pool, C. Shared Object 가 있을 수 있다.
- 매트릭스 내에 존재하는 각각의 정책들은 나름대로의 장단점을 가질 수 있으며, 이 과정에서 운영을 위한 캐쉬 고안 등 여러 가지 기법이 동원될 수 있다. 각 정책을 선택하였을 경우에 사용될 수 있는 기법들은 심볼로 표현되었다. 첫번째 심볼은 OpenView 매니저 오브젝트에 대한 스케쥴링이 필요하다는 의미이며, 두번째 심볼은 OpenView 오브젝트에 대한 스케쥴링, 그리고 세번째 심볼은 캐쉬 정책이 필요하다는 것을 말해준다.
- 각 정책을 선택하였을 경우에 가능한 배치를 요약하면 다음과 같다.

O 1A

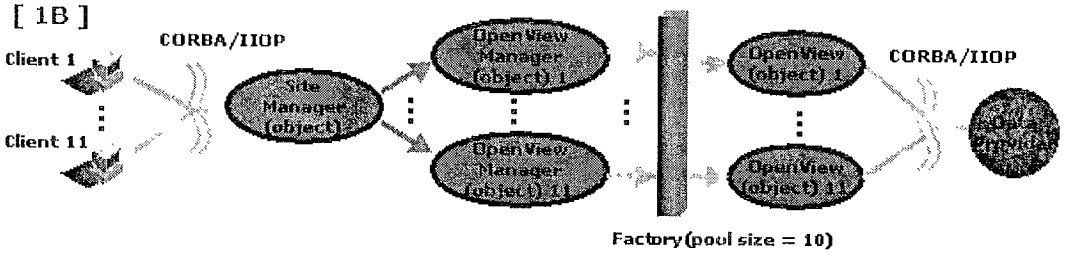


[그림 63] **1A**의 경우

1A는 오픈뷰 매니저가 Thread per Request, 오픈뷰가 Thread per Request로 운영되는 경우이다. 그림에 나오는 클라이언트는 1에서 11까지의 임의 개수를 표현한다. 클라이언트와 애플리케이션 서버, 그리고 애플리케이션 서버와 데이터 제공자는 CORBA/IIOP 채널을 이용하고 있다. Site Manager는 커뮤니케이션 스트림의 유형이 이미지인지, 벡터 객체인지를 구별하고 그에 따라 오픈뷰 매니저를 호출하는 역할을 한다.
오픈뷰 매니저는 클라이언트 요구 개수만큼 인스턴스를 생성하며, 각각의 인스턴스는 하나의 오픈뷰 객체를 생성하고 사용한다. 가장 단순한 유형

의 아키텍쳐로 오픈뷰 매니저가 인스턴스화되고 사용되는 방식은 CGI(Common Gateway Interface)를 이용한 방식과 유사하다. 데이터 제공자로의 연결은 오픈뷰 매니저가 담당한다.

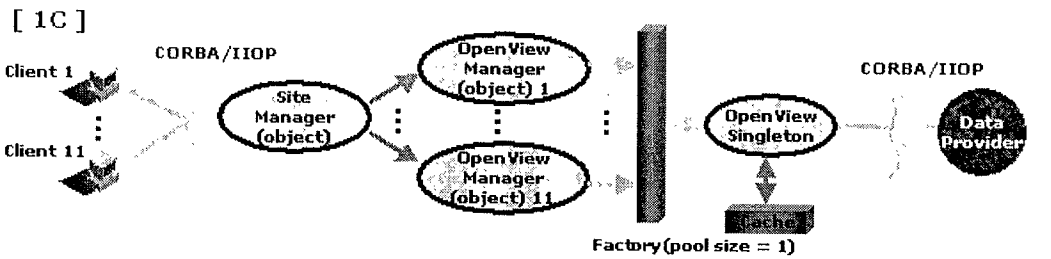
O 1B



[그림 64] **1B**의 경우

1B는 오픈뷰 매니저가 Thread per Request, 오픈뷰가 Thread Pool로 운영되는 경우이다. 오픈뷰 매니저는 클라이언트 요구가 들어올 때마다 생성되지만 각각의 오픈뷰 매니저 인스턴스는 오픈뷰 팩토리를 통해 오픈뷰를 획득한다. 팩토리의 최대 인스턴스 개수는 Pool의 크기에 의해 정해지며 이 개수를 넘어서면 그 이후의 오픈뷰 매니저 인스턴스는 오픈뷰 인스턴스 생성때까지 기다려야 한다. 공간 데이터 처리 과정에서 오픈뷰 객체가 차지하는 리소스가 매핑 과정에 연관되어 많다는 점에서 리소스를 줄이는 일차적인 방법일 수 있으나, 제한된 수의 클라이언트 요구를 받는다는 문제가 있다.

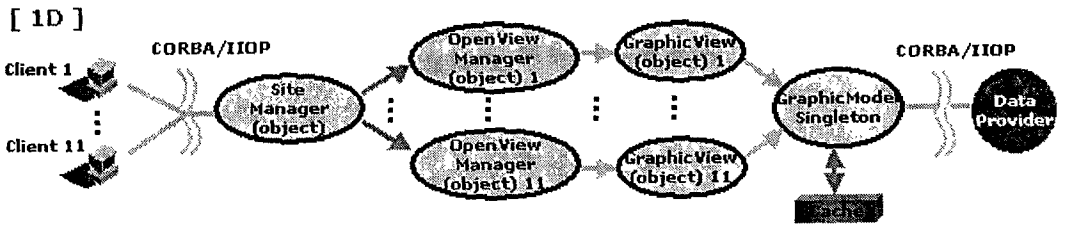
O 1C



[그림 65] **1C**의 경우

1C는 오픈뷰 매니저가 Thread per Request, 오픈뷰가 Shared Object 로 운영되는 경우이다. Singleton 패턴을 사용하는 경우, 오픈뷰는 클라이언트 요구를 순서대로 처리해 주는 하나의 인스턴스만을 생성한다. 이 방식은 1B의 서브셋이라고 할 수 있으며, 공간 데이터의 양이 크고 메모리 리소스가 많은 경우 데이터를 메모리 상에 한꺼번에 올려놓고 처리할 수 있는 방법이다.

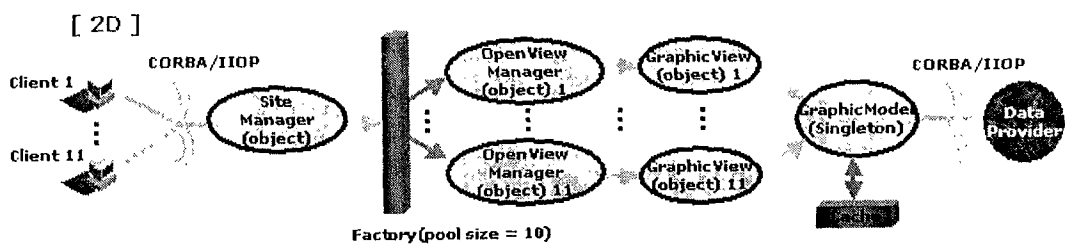
○ 1D



[그림 66] **1D**의 경우

1D는 오픈뷰 매니저가 Thread per Request, 오픈뷰가 멀티 뷰를 사용하여 운영되는 경우이다. 각각의 오픈뷰 매니저 인스턴스는 하나의 그래픽 뷰를 인스턴스화하는데 그 그래픽 뷰는 해당 그래픽 모델을 접근하여 데이터를 처리한다. 이 과정에서 그래픽 모델은 싱글톤으로 인스턴스를 생성하며, 캐쉬가 사용된다. 1C의 방식과 유사하지만 클라이언트의 요구를 병렬적으로 처리한다는 점에서 다수의 클라이언트 요구를 무리없이 처리할 수 있는 방법이다. 역시 많은 공간 데이터가 충분한 메모리 상에 로드될 때 유용하다.

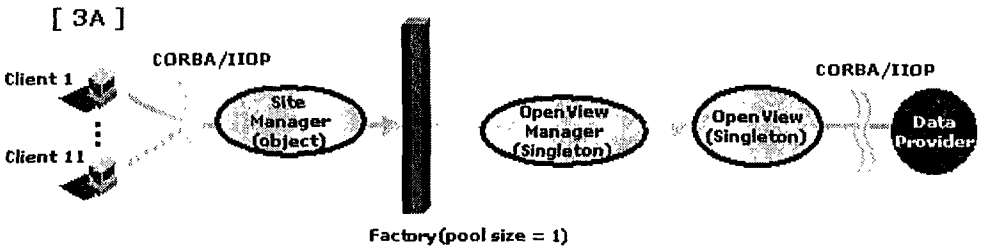
○ 2D



[그림 67] **2D**의 경우

2D는 오픈뷰 매니저가 Thread Pool, 오픈뷰가 멀티 뷰로 운영되는 경우이다. 오픈뷰가 멀티 뷰로 운영되는 것이 각각의 객체로 처리하는 방식보다 성능이 좋으므로 2A, 2B, 2C의 경우는 생략하였다.
각각의 클라이언트 요구마다 생성되는 오픈뷰 매니저 인스턴스는 팩토리에 의해 그 최대 개수가 제어된다. 역시 최대 개수가 10개이므로 마지막 열한번째 클라이언트 요구는 기다리게 된다. 오픈뷰 매니저는 각각 하나의 그래픽 뷰 인스턴스를 만들어 내며 공유되는 그래픽 모델에 의해 데이터가 처리된다.
이 배치는 클라이언트의 요구를 Pool의 크기를 변경함으로써 제어할 수 있다는 점과 멀티 뷰의 기능을 그대로 이용할 수 있다는 점에서 가장 좋은 배치들 중의 하나라고 할 수 있다.

O 3A



[그림 68] **3A**의 경우

3A는 오픈뷰 매니저가 Shared Object, 오픈뷰가 하나의 오브젝트로 운영되는 경우이다. 나머지 경우는 오픈뷰 매니저에 의해 기능이 제한되는 경우이므로 생략하였다.
이 배치에서는 오픈뷰 매니저가 싱글톤으로 생성되며, 이 인스턴스는 하나의 오픈뷰 인스턴스를 가진다. 클라이언트 요구는 순서대로 스케쥴링되면서 기다리므로 클라이언트의 요구가 많을 때는 응답 시간이 늦어지는 단점이 있다.

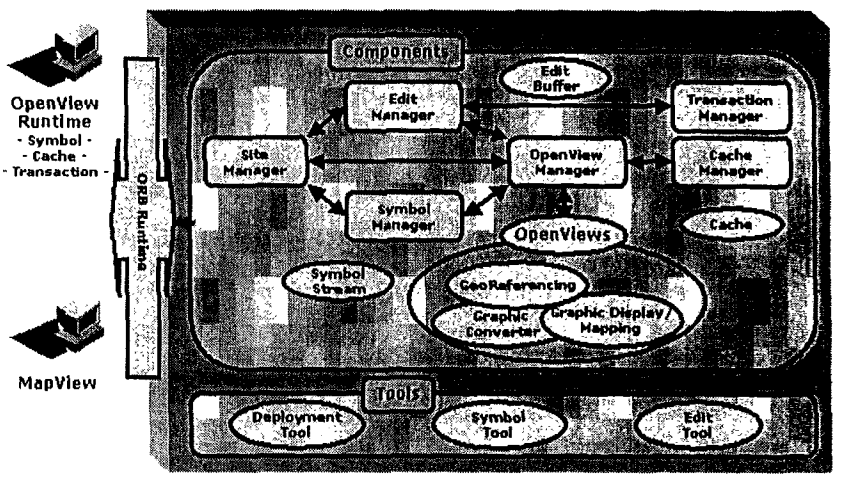
- 인터넷/인트라넷에서 사용되는 GIS 애플리케이션 서버는 기본적으로 대용량의 공간 데이터와 함께 다수의 클라이언트 요구를 가정한다. 따라서 공간 데이터의 크기와 클라이언트의 요구, 그리고 사용 가능한 리소스(hardware resource)의 양에 따라 적절히 배치하는 것이 필요하다. 결국 애플리케이션 서버는 사용자가

배치를 위한 오브젝트별 정책을 얼마나 유연하게 할 수 있는가가 여러 가지 클라이언트의 요구사항을 만족시킬 수 있는가를 결정짓는다고 할 수 있다.

- PASS 아키텍쳐는 이러한 다양한 요구사항을 만족시킬 수 있도록 오브젝트별 정책을 설정할 수 있도록 한다.

마. PASS의 구성 컴포넌트

- [그림 69]는 PASS의 주요 구성 컴포넌트들을 다이어그램으로 표현한 것이다. 사이트 매니저(Site Manager)는 클라이언트의 요구사항을 이해하고 네트워크 상의 오브젝트 정보를 관리한다. 오픈뷰 매니저(Openview Manager)는 내장된 오픈뷰 라이브러리를 제어하는 역할을 하며, 심볼 매니저(Symbol Manager)는 네트워크 상의 심볼 서버들로부터 심볼 정보를 획득하고 데이터에 맞는 심볼을 설정하는 역할을 한다. 캐쉬 매니저(Cache Manager)는 데이터 제공자로부터 넘어온 데이터를 캐쉬하고 편집 매니저(Edit Manager)는 데이터 갱신을 위한 작업을 진행한다.



[그림 69] PASS의 주요 구성 컴포넌트

(1) 사이트 매니저

- 사이트 매니저는 웹 출판(web publishing)을 위한 컴포넌트로 CGI 구성 모듈과 유사한 역할을 한다. 단, CGI 방식은 상태가 없는 데 반해서 사이트 매니저는 클라이언트와 서버 객체간 연결을 보장하고 있으므로 상태를 지속적으로 유지한다. 클라이언트는 이미지 스트림과 벡터 스트림을 모두 받을 수 있어야 하므로 사

이트 매니저는 이 유형을 파악하고 그에 맞는 결과를 리턴해 준다. 예를 들어, 이미지 스트림 방식을 사용하는 클라이언트의 경우에는 오픈뷰 매니저가 만들어 낸 결과물을 이미지로 변환하여 서비스하여야 하며, 벡터 스트림 방식을 사용하는 클라이언트는 매핑 커널을 가지고 있으므로 데이터 제공자로부터 획득한 객체를 전처리를 거쳐 클라이언트에게 넘겨주면 충분하다.

- 사이트 매니저의 또다른 역할을 delegation으로 객체 편집이나 트랜젝션 관리와 같은 역할 이외에 오픈뷰 매니저에게 클라이언트 요구를 전달하거나, 심볼 매니저에게 요구된 심볼 정보를 넘겨줌으로써 원하는 심볼을 심볼 제공자 서버 객체로부터 받아올 수 있도록 하는 역할을 담당한다.

(2) 오픈뷰 매니저

- 오픈뷰 매니저는 애플리케이션 서버 안에 내장된 오픈뷰를 직접적으로 제어하는 역할을 담당한다. 공간 데이터는 데이터 제공자로부터 획득하며 캐쉬 매니저를 통해 획득한 데이터를 캐쉬하거나 오픈뷰로 보내어 원하는 지도 이미지를 생성한다. 지도 이미지 생성에는 심볼 매니저로부터 전달받은 심볼을 이용하며, 디스플레이된 이미지를 로컬에서 이미지 스트림으로 변환한 후에 클라이언트로 넘겨주는 역할을 하기도 한다.
- 이미지 스트림으로 클라이언트에게 전달하는 경우에는 디스플레이된 지도 이미지를 원하는 스케일로 변환하는 과정이 주요 작업이 되며, 이 과정에서 지도의 범위(extent)를 확인한다. 오픈뷰 내에 존재하는 공간 객체 및 속성 데이터는 객체 구별자를 통해 구별된다.

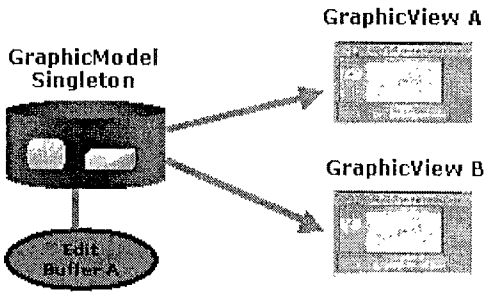
(3) 캐쉬 매니저

- 데이터 제공자로부터 획득된 공간 데이터는 그 양이 많을 뿐 아니라 얼마나 넓은 범위의 데이터를 클라이언트가 요구할 지 예측할 수 없는 경우가 대부분이다. 더욱이 개방형 시스템에서는 통합 지도정보 서비스를 위해 여러 개의 공간 데이터베이스를 연결하여 이용하므로 최악의 경우에는 남한 전체의 10,000분의 1 도면 데이터를 모두 로딩해야 하는 경우가 생길 수도 있다. 또한 동일 지역의 데이터를 여러 클라이언트가 동시에 요구하는 경우 다시 데이터 제공자로부터 데이터를 가져오는 것은 자원의 낭비라 할 수 있다.
- 캐쉬 매니저는 다양한 캐쉬 기법을 이용하여 애플리케이션 서버의 이러한 문제를 보완할 수 있다. 일반적으로 사용되는 캐쉬 기법으로는 타일링 방식이 있으며, 폴리곤 조합(composition)/분리(decomposition) 방식을 이용할 수도 있다.
- 오픈뷰 매니저는 캐쉬 매니저를 이용하여 여러 데이터 제공자에 대해 공통되게

접근할 수 있는 인터페이스를 제공받게 된다. PASS에서는 데이터의 유형과 성격에 따라 타일링 방식과 폴리곤 조합/분리 방식을 사용할 수 있도록 함으로써 유연성을 확보하는 방안을 제시하고 있다.

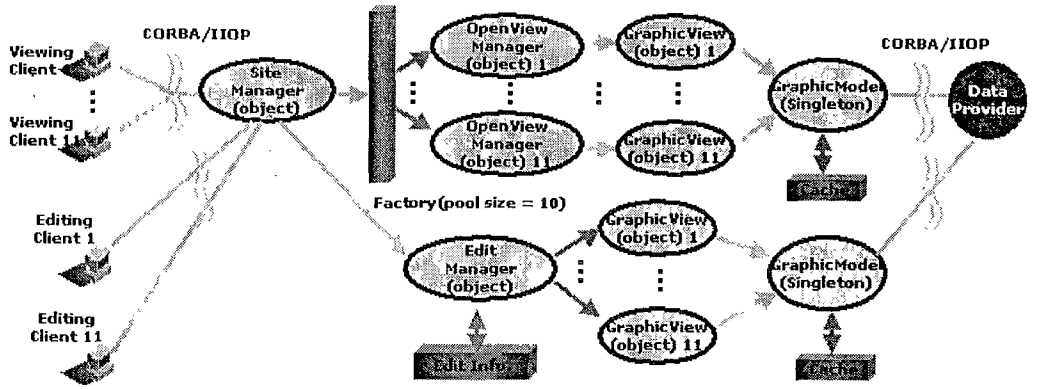
(4) 편집 매니저

- 클라이언트에서 벡터 객체를 내려받아서 편집한 후에 다시 공간 데이터베이스에 저장하거나, 애플리케이션 서버 수준에서 편집한 후에 다시 공간 데이터베이스에 저장하는 경우, 공간 데이터의 편집 기능은 데이터 갱신에 있어 중요하다.
- PASS에서는 공간 데이터의 편집을 담당하는 편집 매니저를 두고 있다. 오픈뷰는 멀티 뷰를 지원하므로 하나의 그래픽 모델이 편집 클라이언트와 뷰 클라이언트에 의해 공용으로 레퍼런스되는 경우에는 현재 편집되고 있는 상황을 실시간으로 공유할 수 있는 기능을 제공할 수 있다.
- 또한 버퍼 안에 편집 정보를 저장해 놓음으로써 지속적인 편집 정보의 관리가 가능하다. [그림 70]은 편집 매니저가 그래픽 모델을 두 개의 뷰가 공유하면서 편집하는 경우를 보여주고 있다.



[그림 70] 편집 매니저가 그래픽 모델을 공유하는 경우

- 애플리케이션 내의 오픈뷰가 편집 매니저와 함께 공간 데이터를 편집하는 경우에는 데이터 제공자가 관리하는 원 데이터와의 데이터 일관성을 유지하는 방안을 모색하는 것이 중요하다. [그림 71]은 공간 데이터 편집을 위한 다이어그램을 보여주고 있다.



[그림 71] 공간 데이터 편집을 위한 다이어그램

- 편집에 관련된 정보는 편집 매니저가 관리하며, 그래픽 뷰는 각각의 클라이언트 요구에 맞게 생성된다. 이 과정에서도 마찬가지로 오브젝트 배치나 쓰레드에 관한 여러가지 전략이 마련될 수 있다.

(5) 심볼 매니저

- 심볼 매니저는 원격지에 심볼 관련 스트림을 저장하고 이를 원하는 클라이언트에게 전달하는 심볼 서비스 제공자를 이용한다. 심볼 매니저는 클라이언트 요구에 맞는 이미지를 생성하기 위해 심볼을 찾아내는 작업을 담당하며, 로컬 머신에 있는 경우에는 그 심볼을 사용하고 원격지 심볼을 요구하는 경우에는 원격지 심볼을 이용하는 방식을 취한다.

바.심볼 서비스

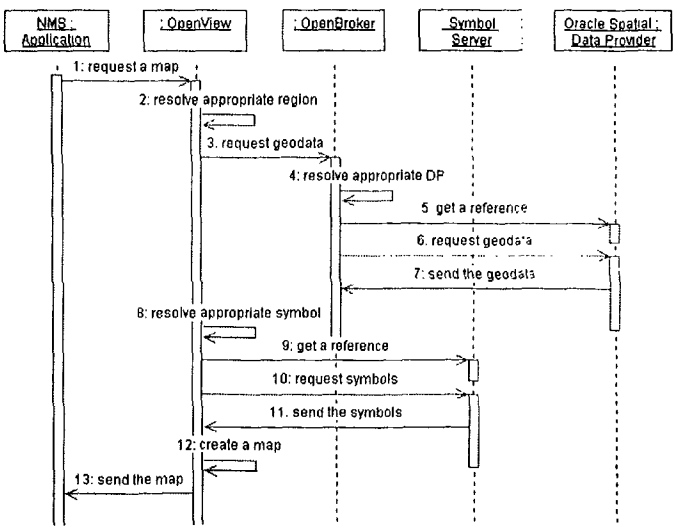
- 기존의 GIS 시스템들은 나름대로의 심볼 체계를 이용하였으므로 표준화에 있어서도 데이터를 공유하기 위한 노력은 있었으나 심볼을 호환되게 하기 위한 노력은 미미했다. 예를 들어, 국가 수치지도(National Digital Map) 를 위해 정의된 코드체계에 맞게 심볼을 주어야 하는 경우, 각각의 시스템이 나름대로의 심볼 처리를 위한 메커니즘을 정의해야만 했다.
- 표준을 통해서 공간 데이터를 공유하도록 만들듯이 심볼을 분산된 환경에서 공유시키는 경우 많은 장점이 생길 수 있다. 예를 들어, 특정 코드값에 대한 심볼이 바뀌는 경우 규정을 정하는 기관은 그 심볼을 새로이 정의하고 심볼 서비스 오브젝트로 등록하면 네트워크 상의 어느 곳에서든지 그 심볼을 받아볼 수 있을 것이

다.

- 그러나, 앞에서도 언급된 바와 같이 심볼 인터페이스는 아직 표준으로 정의되지 않았으므로 심볼 서비스를 위해서는 나름대로의 인터페이스가 필요하다. OpenViews는 기본적으로 자바 2D API를 확장하여 심볼을 정의하므로 자바 2D API를 이용할 수 있는 심볼 인터페이스가 정의되어야 한다. 정의된 심볼 인터페이스는 다음과 같다.

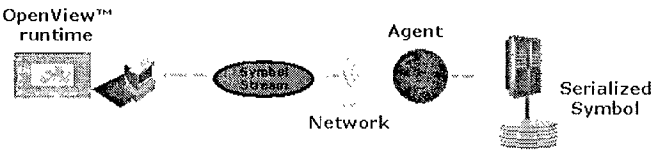
```
interface SymbolServiceProvider {
        exception ServiceProviderException {string why;};
        // the name of the provider
        void getName(out wstring name);
        // list symbols
        void listSymbol(out StringSeq names);
        // get symbol size
        void countSymbol(in wstring name, out long count);
        // get OpenSymbol
        void getSymbol(in wstring name, out OpenSymbolSeq symbol)
                raises (ServiceProviderException);
};
```

- [그림 72]는 NMS(Network Management System) 시스템이 특정 지역의 장비 분포를 알아보기 위해 공간 데이터와 심볼 서비스를 이용하는 시나리오(scenario)를 시퀀스 다이어그램(sequence diagram)으로 표기한 것이다.



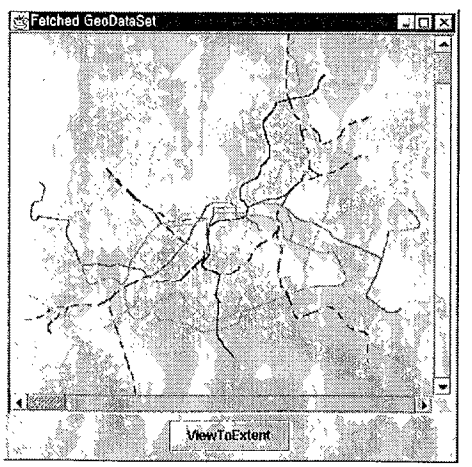
[그림 72] OpenViews를 이용한 일반적인 지도 생성 메커니즘

- NMS 애플리케이션이 OpenViews 인스턴스를 통해 지도 생성을 요구하면 실제 지도 매핑 작업은 OpenViews 인스턴스가 담당하게 된다. 생성된 인스턴스는 먼저 ORB를 통해 데이터 제공자에게 원하는 스케일과 지역 정보를 전송하여 공간 데이터를 획득하고 원하는 심볼을 심볼 제공자에게 요청하면 그에 해당하는 심볼이 넘어오므로, 이 정보들을 이용하여 매핑을 하고 그 결과를 클라이언트에게 넘겨주는 과정을 거친다.
- [그림 73]은 심볼을 분산시켜 놓은 후 가져오기 위한 시스템 배치도로서 디스크 상에 저장된 심볼은 심볼 서비스 오브젝트를 통해 획득될 수 있다. 심볼은 앞에서 기술된 인터페이스를 통해 접근가능하며, 위의 데이터 제공자로부터 가져온 오브젝트에 등록할 수 있다.



[그림 73] 사용자 정의 심볼의 분산과 획득을 위한 기본 시스템 배치도

- [그림 74]는 [그림 73]의 구조를 이용하여 분산되어 있는 심볼 서버 오브젝트로부터 획득한 심볼을 디스플레이한 것이다. 서울 지하철 노선도가 이용되었으며, 심볼은 서울 지하철 노선도 심볼과 유사하도록 정의되었다. 사용자 정의 심볼을 가져오는 방식은 마찬가지로 앞에서 정의된 심볼 인터페이스를 이용하였다.



[그림 74] 서울 지하철 노선도 심볼의 획득과 디스플레이

제 8 절 프로토타입 시스템 설계 및 구현

1. 애플리케이션 프로토타입 개발

- 본 과제에서는 오픈뷰 매핑 커널이 기능을 확인하고 실제 업무에서 활용될 수 있는 기능을 보완하기 위해 로컬 머신에서 사용할 수 있는 Map Viewing 애플리케이션 프로토타입을 개발하였다. 자바 기반의 매핑 커널을 사용하는 애플리케이션 프로토타입은 마찬가지로 100% 순수 자바를 기반으로 하였다. 현재 다수의 Map Viewing 상용 프로그램이 있지만 OS 시스템에 종속적인 것이 일반적이므로 OS에 맞는 여러 가지 버전을 준비해야 하는 번거로움이 있다. 본 절에서는 OS에 독립적이라는 자바의 잇점을 이용하여 여러 시스템에서 사용가능한 GIS 애플리케이션 프로그램을 개발하는 과정을 기술한다.

2. 개발 환경

- 공간 데이터 저장 및 타 시스템과의 호환을 위해 데이터 포맷은 GIS 업계에서 가장 일반적으로 사용되는 공개 포맷 중 하나인 Shapefile을 사용한다. 모델링 도구로는 Rational Rose™를 사용하여 설계 관련 다이어그램들을 작성한다. 개발 당시 오픈뷰는 버전 0.22, 자바 개발툴은 SDK 1.2.2 를 사용하였다. 애플리케이션 프로토타입의 운영환경은 로컬 머신으로 제한하였다.
- 사용한 공간 데이터는 국가 토지정보체계 사업에서 구축한 일부 데이터이다.

3. 설계

가. LMIS View의 정의

- 토지정보 관련 데이터를 관리하는 뷰로서 사용자가 로컬 머신에서 Shapefile을 로딩하고, 줌인/줌아웃, 패닝, 데이터베이스 보기, 사상 찾기, 심볼 변경, 사상 확인, 라벨 처리 등의 기능을 수행할 수 있게 한다.
- 예상되는 프로그램의 문제점으로는 자바 VM으로 인한 속도 저하와 많은 용량의 메모리 소모 등이 있다.

나. Actor

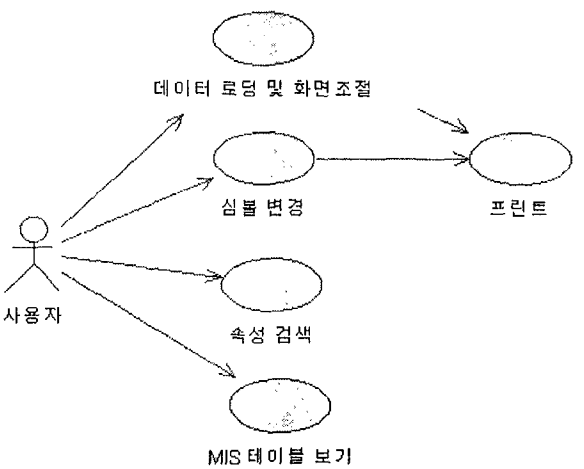
- 사용자: Shapefile을 보고 프린트하고 저장하려 한다.

다. 유스케이스

- 다음과 같은 요구사항이 LMIS View에서 해결되어야 한다.

  - 사용자가 GraphicModel에 로컬 머신의 Shapefile을 로딩한다.
  - 로딩이 끝나면 GraphicView를 줌인/줌아웃/패닝을 실행한다.
  - 사용자가 GraphicView에 로딩된 레이어의 심볼을 변경하고 저장한다
  - 사용자가 화면 상에 보이는 GraphicView를 프린트한다.
  - 사용자가 GraphicLayer에 해당되는 RecordSet을 확인한다.
  - 사용자가 속성정보를 이용하여 해당 ShapeObject를 검색한다.

- 위의 요구 사항에 기반한 유스케이스는 다음과 같다.
  - 데이터 로딩 및 화면 조절
  - 심볼 변경
  - 프린트
  - 속성 검색
  - MIS 테이블 보기



[그림 75] LMIS View의 유스케이스 다이어그램

라. 패키지와 클래스 구성

(1) ViewControll 패키지

- 이 패키지에는 GraphicModel에 데이터를 로딩하고 GraphicView를 확대/축소 및 이동하는 기능을 제공하는 클래스들이 들어있다.

클래스명	기능
ContentPane	GUI의 ContentTable과 ViewControll의 MapController를 연결시켜주는 기능을 제공한다.
CoordListener	GraphicView 상의 커서 위치를 Status에 출력한다.
MapController	GraphicModel에 Shapefile을 로딩하고 GraphicView 화면을 조절한다. GraphicViewController를 GrahpicView에 추가/제거하며, 그래픽 모델에 로딩된 GraphicLayer의 레이어를 관리한다.
MeasureController	이벤트로 입력받은 화면상의 두 지점간의 거리를 측정하는 기능을 제공한다.
PanController	GraphicView의 시점을 원하는 지점으로 이동하게 하는 기능을 제공한다.
ShapeFilter	파일을 로딩할 때 Shapefile만을 디스플레이할 수 있게 한다.

(2) GUI 패키지

- 사용자가 프로그램과 의사전달을 할 수 있는 인터페이스를 생성한다. 예) 메뉴, 버튼, 툴.

클래스명	기능
Content	GraphicModel에 로딩된 파일에 해당하는 GraphicLayer의 이름을 저장하며 레이어의 turn on/off, 활성화/비활성화에 대한 정보를 저장한다.
ContentTable	Content 들의 배열을 VerticalLayout의 형태로 보여주며 이 Content 들의 배열 순서를 변경한다.
LmisView	실제 사용자가 사용하는 Graphic User Interface이다. 상단에 OpenMenu, OpenTool, 좌측에 ContentTable, 중앙에 GraphicView, 하단에 Status로 이루어져 있다.

OpenMenu	사용자가 프로그램의 기능을 수행할 수 있게 하는 메뉴를 제공한다.
OpenTool	사용자가 프로그램의 기능을 수행할 수 있게 하는 버튼, 툴을 제공한다.
OverView	현재 GraphicView의 범위를 전체 범위와 비교하여 보여주며, 화면을 조절할 수 있는 기능을 제공한다.
Status	커서의 좌표를 보여주며 거리의 계산 결과와 진행상태를 보여준다.
VerticalFlowLayout	수직으로 Content를 배열하게 하는 기능을 제공한다.

(3) SymbolControll 패키지

- 심볼을 변경할 수 있는 그래픽 인터페이스와 심볼변경을 하는 기능을 제공하는 클래스들로 이루어져 있다.

클래스명	기능
DefaultSymbol	디폴트 심볼 정보를 갖고 있다.
GraduatedSymbolChooser	Graduated Color 스키마로 심볼 종류를 변경하는 인터페이스를 제공한다.
LinePreview	LineSymbolChooser에서 변경한 폴리라인 심볼 유형을 반영한다.
LineSymbolChooser	폴리라인의 심볼을 변경하는 인터페이스를 제공한다.
PointPreview	PointSymbolChooser에서 변경한 포인트 심볼 유형을 반영한다.
PointSymbolChooser	포인트의 심볼을 변경하는 인터페이스를 제공한다.
PolygonPreview	PolygonSymbolChooser에서 변경한 포인트 심볼 유형을 반영한다.
PolygonSymbolChooser	폴리곤의 심볼을 변경하는 인터페이스를 제공한다.
SingleSymbolChooser	GraphicLayer의 심볼 타입을 Single로 변경할 수 있는 인터페이스를 제공한다.
SymbolArray	GrahpicLayer(Polygon 타입)에 대한 심볼정보(색상, 패턴, 크기 등)를 저장한다.
SymbolController	SymbolChooser에서 선택한 심볼에서 얻어지는 SymbolArray를 이용하여 실제 GraphicLayer의 심볼을 변경하는 기능을 제공한다. 그리고 현재 GraphicLayer가 가지고 있는 SymbolArray를 리턴하는 기능을 제공한다.
SymbolEditor	실제 사용자가 심볼을 변경할 때 사용하는 Graphic User

	Interface이다.
UniqueSymbolChooser	GraphicLayer의 심볼 타입을 특정 필드값에 따른 Unique로 변경할 수 있는 인터페이스를 제공한다.
UniqueSymbolTableModel	PreviewTable에 적용되는 TableModel이다.

(4) PrintControll 패키지

- 프린트와 관련된 기능을 제어한다.

클래스명	기능
PrintContoller	사용자가 현재 GrahicView를 프린트할 수 있게 하는 기능을 제공한다.

(5) TableControll 패키지

- 테이블과 관련된 기능을 제어한다.

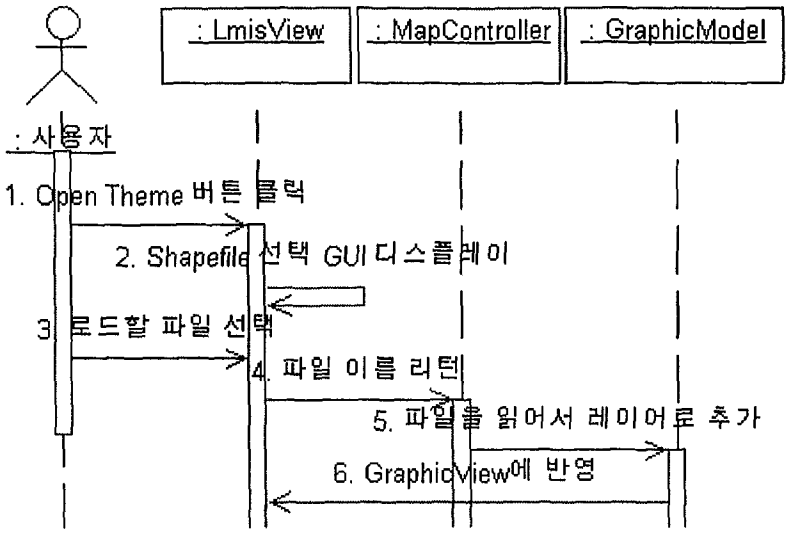
클래스명	기능
TableController	MIS 테이블과 GraphicView를 상호 연결하는 기능을 제공한다.
Identifier	이벤트로 입력된 포인트를 포함하는 ShapeObject에 대한 MIS 테이블 레코드(속성)를 보여준다.
QueryBuilder	속성 질의를 할 수 있는 인터페이스를 제공한다.

마. 시퀀스 다이어그램

(1) 데이터 로딩 및 화면 조절

- 데이터 로딩 시퀀스는 다음과 같다.

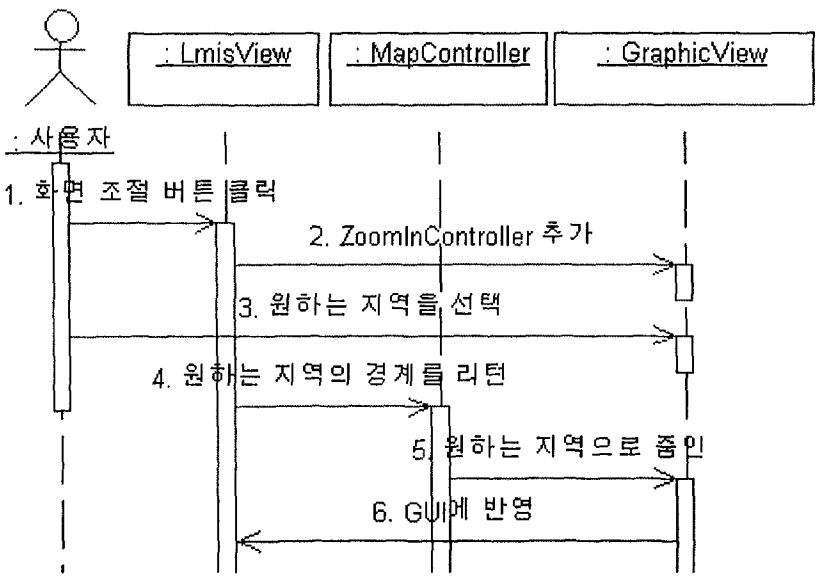
① 사용자가 LMIS View의 툴바나 메뉴에서 Open Theme을 선택한다.
② 파일을 선택할 수 있는 ShapeFileChooser GUI가 나타나고 사용자가 로드할 파이을 선택한다.
③ 파일을 GraphicModel에 GraphicLayer로 로드한다.
④ GraphicView를 다시 그린다.



[그림 76] 데이터 로딩 시퀀스

- 화면조절(줌인/줌아웃) 시퀀스는 다음과 같다.

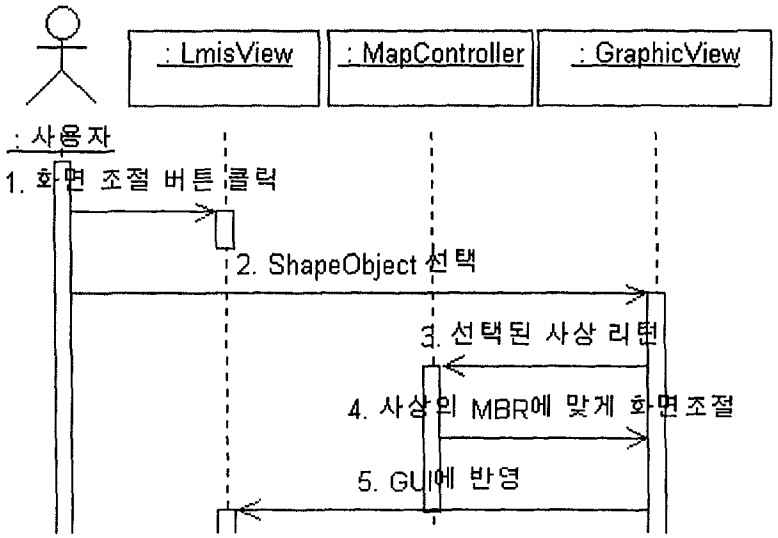
① 사용자가 LMIS View에서 화면조절(예) 줌인) 버튼을 클릭한다.
② GraphicView에 ZoomInController를 설정한다.
③ 사용자가 확대/축소할 지역을 선택한다.
④ 선택한 영역에 맞게 GraphicView를 확대/축소한다.
⑤ GraphicView를 다시 그린다.



[그림 77] 줌인 시퀀스

- 선택한 사상으로 확대하는 시퀀스는 다음과 같다.

① 사용자가 선택한 사상으로 확대하는 버튼을 클릭한다.
② GrahpicView에 SelectionController를 설정한다.
③ 사용자가 GraphicView상의 사상을 선택한다.
④ 선택된 사상의 MBR에 맞게 GraphicView를 확대/축소한다.
⑤ GraphicView를 다시 그린다.

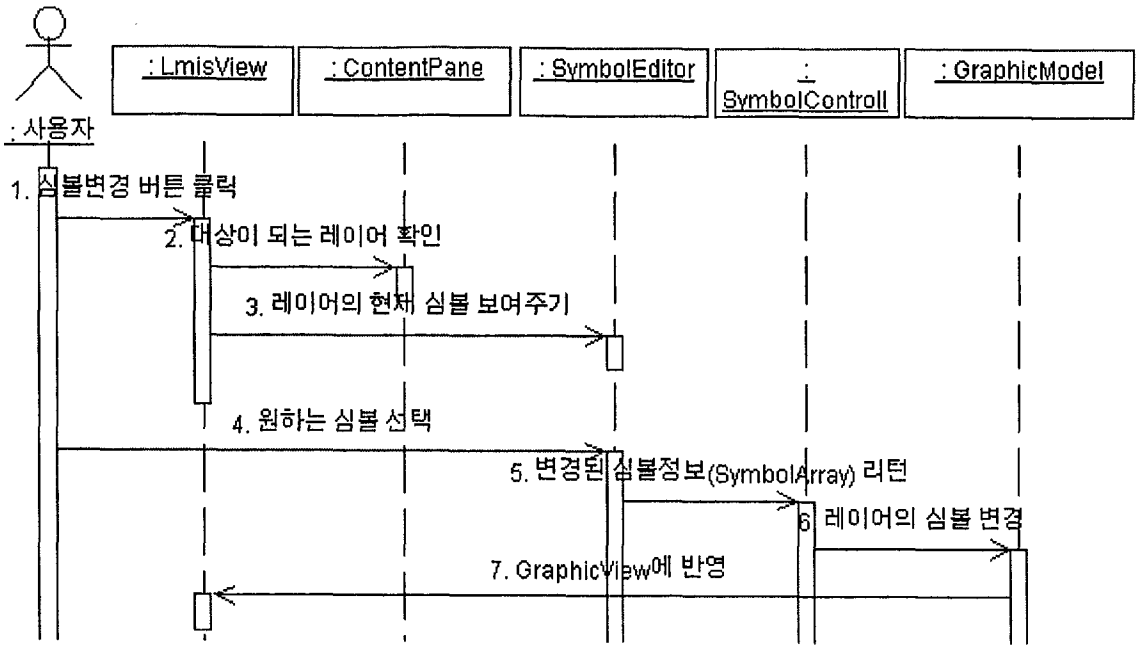


[그림 78] 선택한 객체로의 줌인 시퀀스

(2)심볼 변경

- 심볼 변경 시퀀스는 다음과 같다.

① GraphicView에 하나의 GraphicLayer가 활성화되어 있는 상태에서, 사용자가 심볼편집기 버튼을 클릭한다.
② 심볼편집기가 디스플레이되고 심볼 타입을 선택하면, 현재 레이어의 심볼정보가 나타난다.
③ 원하는 심볼타입과 분류기준이 되는 필드명을 선택한다.
④ 변경된 심볼정보를 GraphicLayer에 적용한다.
⑤ GraphicView를 다시 그린다.

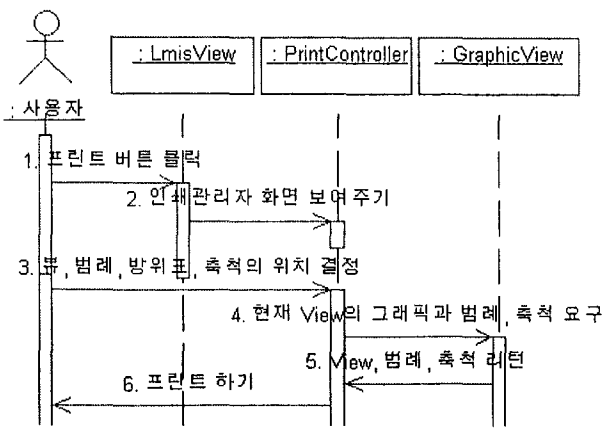


[그림 79] 폴리곤 레이어의 심볼 변경 시퀀스

(3) 프린트

- 프린트 시퀀스는 다음과 같다.

① 사용자가 LMIS View에서 프린트 버튼을 클릭한다.
② 프린트 미리보기 화면이 나타난다.
③ 사용자가 뷰, 범례, 축척, 방위표의 위치와 크기를 설정한다.
④ 프린트를 시작한다.

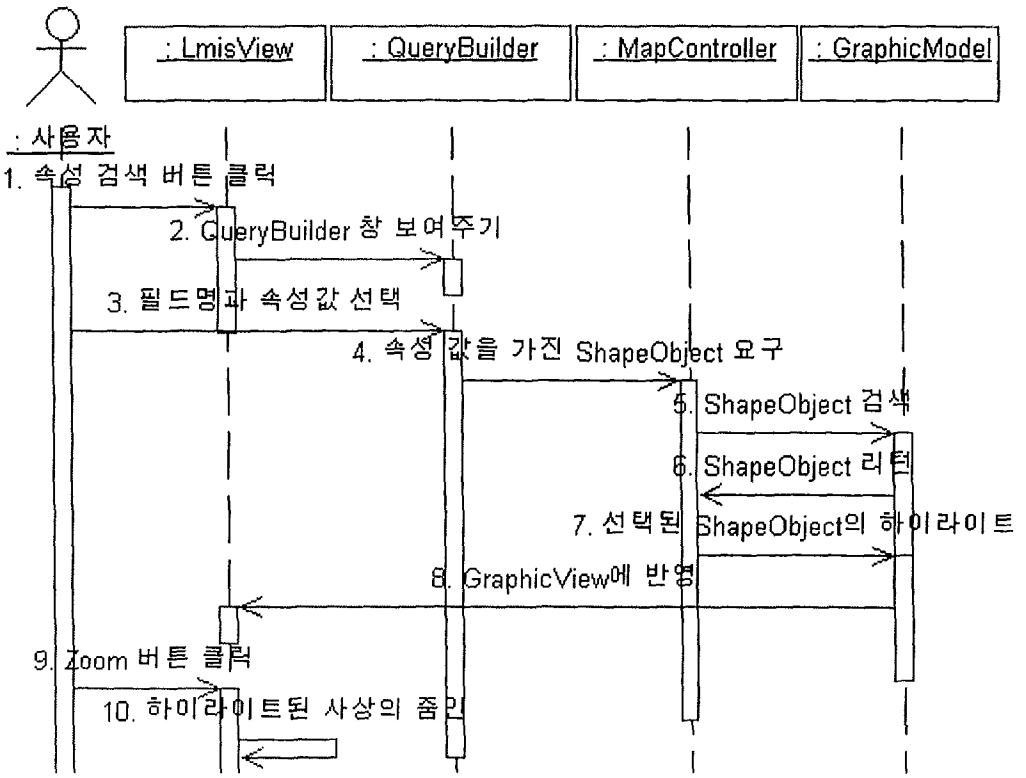


[그림 80] 프린트 시퀀스

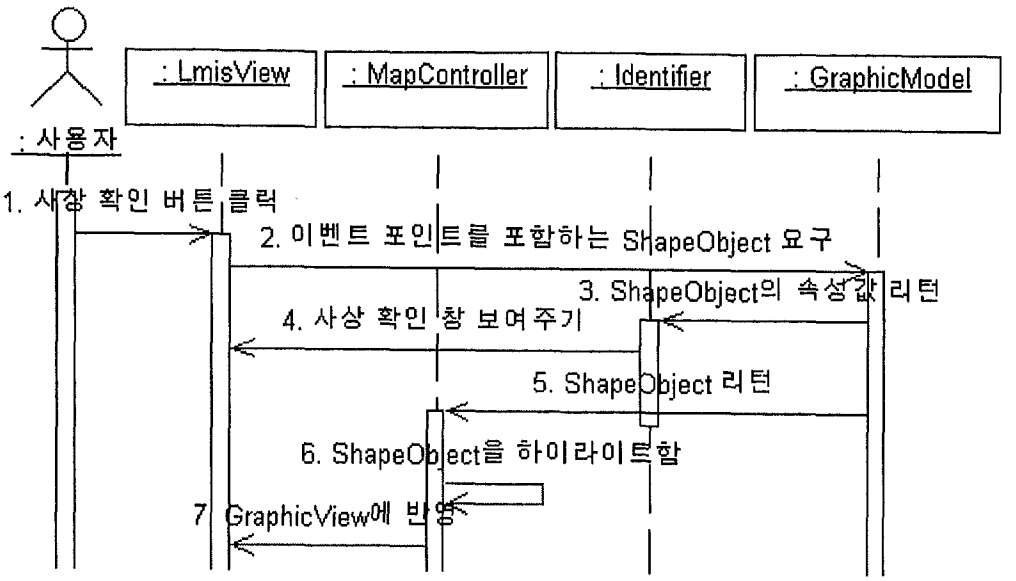
(4) 객체 검색

- 객체 검색(테이블/그래픽 뷰) 시퀀스는 다음과 같다.

① 사용자가 LMIS View에서 사상 검색 버튼을 클릭한다.
② 사상검색 대화상자가 디스플레이된다. 하나의 GraphicLayer가 선택되어 있을경우 레이어의 이름이 대화상자의 타이틀로 설정된다.
③ 사용자가 필드명과 필드값을 선택한다.
④ 선택한 속성값을 가진 사상을 검색하고, 심볼을 변경한다. 대화상자의 줌 버튼을 클릭할 경우 검색한 사상의 MBR에 맞게 GraphicView를 확대/축소한다. 선택해지 버튼을 누르면 선택이 취소되고 원래의 심볼로 설정한다.
⑤ GraphicView를 다시 그린다.



[그림 81] 객체 검색 시퀀스 (테이블)

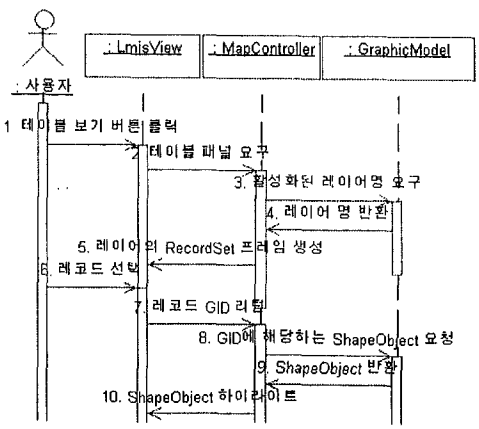


[그림 82] 객체 검색 시퀀스 (그래픽 뷰)

(5) MIS 테이블 보기

- 테이블 보기 시퀀스는 다음과 같다.

① 사용자가 LMIS View에서 테이블 보기 버튼을 클릭한다.
② 활성화된 레이어가 있을 경우 각각의 레이어에 대한 레코드셋을 보여주는 테이블이 디스플레이된다.
③ 테이블의 레코드를 선택하면, 테이블의 Row ID가 반환되며 이에 해당하는 사상을 검색하여 심볼을 변경한다.
④ GraphicView를 다시 그린다.



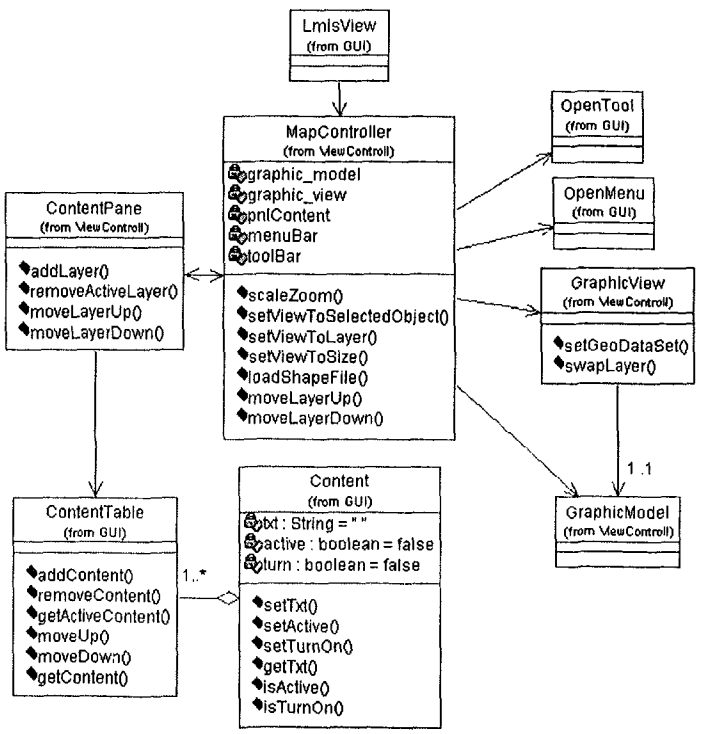
[그림 83] MIS 테이블 보기 시퀀스

(6) 클래스의 속성과 메소드 추출

- 사용자 요구사항을 분석하여 만든 유스케이스와 시퀀스 다이어그램을 통해 클래스의 속성과 메소드를 생성할 수 있다.

(가) 데이터 로딩 및 화면 조절

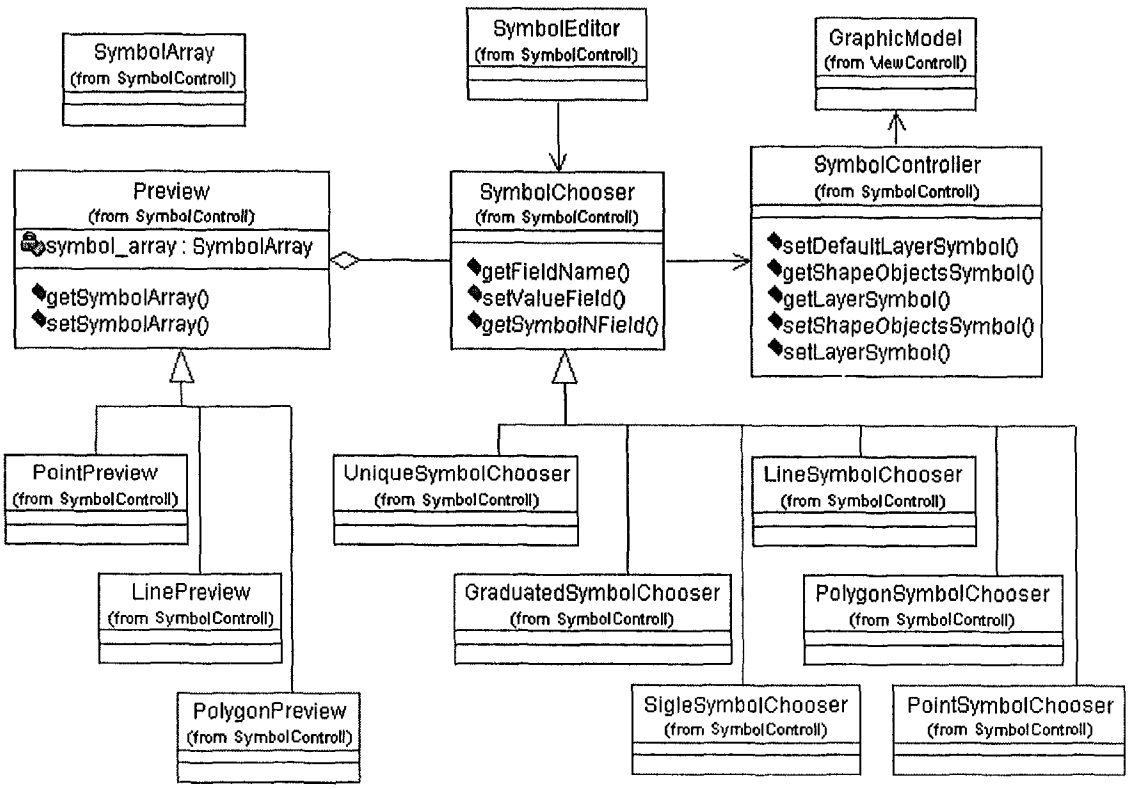
- 레이어관리를 시각적으로 나타내는 ContentTable과 실제 데이터를 로드하고 화면조절을 실행하는 MapController, 그리고 전자와 후자를 연결시키는 기능을 담당하는 ContentPane이 핵심적인 클래스들이다.
- ContentTable은 레이어의 이름과 활성화/비활성화, 켜기/끄기 등의 정보를 저장하고 있는 Content들로 이루어져 있으며 이 Content들의 순서를 결정하는 메소드들을 포함해야 한다.
- 반면에 실제 GraphicModel에서 레이어의 디스플레이 순서를 변경하는 메소드는 MapController가 가지고 있다. 줌인/줌아웃 등의 화면 조절을 담당하는 메소드도 MapController에 포함된다.
- ContentPane은 이들 두 클래스 간의 메소드를 연결하며 동시성을 관리한다.



[그림 84] 데이터 로딩 및 화면 조절 클래스 다이어그램

(나) 심볼 변경

- 선택된 심볼을 시각적으로 디스플레이해주는 Preview 클래스와 실제 선택된 심볼을 GraphicLayer에 적용하는 클래스로 구분된다.
- SymbolChooser에서는 사용자가 현재 레이어의 심볼 정보를 보거나 선택할 수 있는 GUI를 제공하며, 사용자가 새로운 심볼을 선택하면 Preview에 이 심볼정보가 SymbolArray 객체로 전달되어 미리보기 기능을 구현한다. 사용자가 적용버튼을 클릭한 경우 반화된 SymbolArray를 입력으로 하여 SymbolController가 GraphicLayer의 전체 레이어 또는 일부 사상의 심볼을 변경한다.



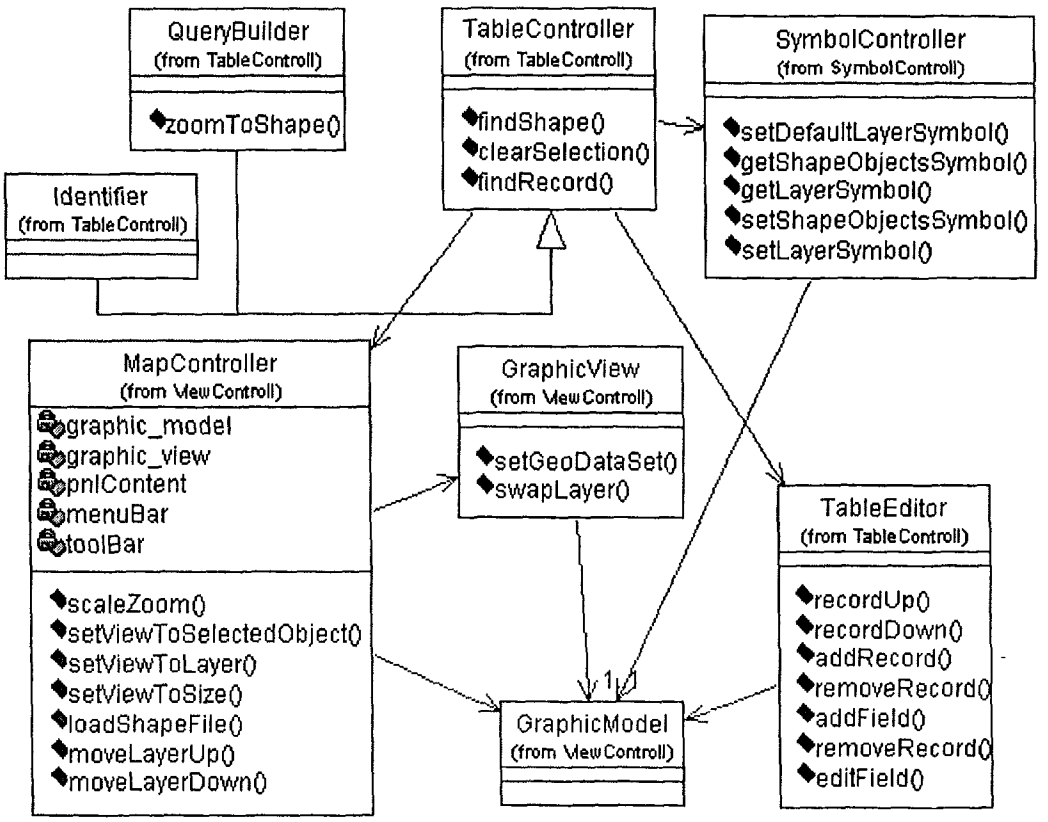
[그림 85] 심볼변경 클래스 다이어그램

(다) 사상 검색

- QueryBuilder에서는 선택된 레이어의 모든 필드명을 보여주고 필드명이 선택되면 모든 필드값을 보여준다. 사용자가 필드값을 선택할 경우 TableController가 이를 반환받아 사상을 검색하는 메소드가 필요하며, 검색한 사상은

SymbolController에 반환하여 심볼을 변경하는 메소드가 필요하다.

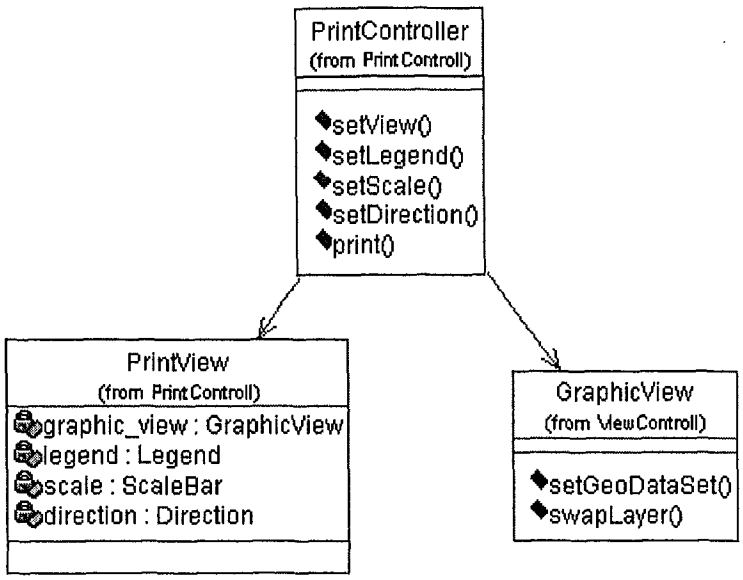
- 선택한 사상을 확대하는 경우 MapController에서 선택한 사상에 맞게 GrahpicView를 확대하는 메소드를 가지고 있어야 한다. TableEditor에서는 레코드셀의 순서나 컴럼의 순서를 변경하는 메소드를 제공한다.



[그림 86] 객체 검색 클래스 다이어그램

(라) 프린트

- PrintView는 사용자가 설정한 뷰, 스케일, 방위표, 범례의 위치와 크기를 미리 보여주어야 한다. 그리고 PrintContoller는 사용자가 선택한 위치에 입력된 크기로 이들을 설정하는 메소드가 필요하다.



[그림 87] 프린트 클래스 다이어그램

4. 구현

O Content 클래스

- GraphicModel 에 로딩된 파일에 해당하는 GraphicLayer 의 이름을 저장하며 레이어의 turn on/off, 활성화/비활성화에 대한 정보를 저장한다.

public class Content
extends javax.swing.JPanel

생성자 요약
Content() 기본 생성자
Content(java.lang.String txt) Content 이름을 설정하는 생성자

메쏘드 요약	
void	addClickCount() 클릭한 수를 하나씩 더한다. 클릭한 수가 홀수이면

	활성화되고 짝수이면 비활성화된다.
int	getClickCount() 클릭한 수를 반환한다
java.lang.String	getText() Content 이름을 반환한다
boolean	isActive() 활성화되어있으면 true 를 반환한다
boolean	isTurnOn() 레이어가 켜져 있으면 true 를 반환한다
void	setActive(boolean bln) 활성화/비활성화를 설정한다. 심볼변경이나 테이블 보기, 사상 검색, 레이어 변경/삭제 등 여러가지 작업이 활성화된 레이어에 대해서만 적용된다.
void	setMapController(MapController conMap) MapController를 설정한다
void	setText(java.lang.String txt) Content 이름을 설정한다
void	turnOff() 레이어를 끈다. 레이어가 Off되어 있으면 GraphicView에 시각적으로 보이지 않으며 MBR 정보도 갖지 못한다.
void	turnOn() 레이어를 켠다

O ContentPane 클래스

- GUI 의 ContentTable 과 ViewControll 의 MapController 를 연결시켜주는 기능을 제공한다.

public class ContentPane
extends javax.swing.JPanel
implements java.awt.event.MouseListener

생성자 요약
ContentPane(MapController conMap)

MapController를 설정하는 생성자

메쏘드 요약	
void	addLayer() 로드한 파일을 선택할 수 있는 GUI를 제공하고, 파일이 선택되면 GraphicModel에 GraphicLayer를 추가하고 ContentTable에 Content를 추가한다. Content의 추가는 ContentTable의 메소드를 사용하여 GraphicLayer의 추가는 MapController의 메소드를 사용한다. 사용자가 파일을 선택하지 않거나, 레이어가 이미 GraphicModel에 존재하는 경우 실행이 되지 않는다.
int	getActiveLayerIndex() 활성화된 레이어 하나의 인덱스를 반환한다
java.lang.String	getActiveLayerName() 활성화된 레이어 하나의 이름을 반환한다. ContentTable의 인덱스 구조와 GraphicModel의 인덱스 구조가 완전히 반대이므로 두 클래스간의 연결이 필요한 경우 레이어의 이름을 참조하는 것이 좋다.
int[]	getActiveLayersIndex() 활성화된 모든 레이어의 인덱스를 반환한다
java.lang.String[]	getActiveLayersName() 활성화된 모든 레이어의 이름을 반환한다
ContentTable	getContentTable() ContentTable을 반환한다
void	mouseClicked(java.awt.event.MouseEvent evt) 레이어관리 버튼의 액션을 설정한다. 레이어 관리 버튼은 Up, Down, Top, Bottom이 있다.
void	moveLayerBottom() GraphicModel에서 GraphicLayer의 인덱스를 마지막으로 설정하고 ContentTable에서 Content를 마지막으로 내린다
void	moveLayerDown() GraphicModel에서 GraphicLayer의 인덱스를 한단계 내리게 하고 ContentTable에서 Content를 한단계 내린다
void	moveLayerTop() GraphicModel에서 GraphicLayer의 인덱스를 1로

	설정하고 ContentTable에서 Content를 첫번째로 올린다
void	moveLayerUp() GraphicModel에서 GraphicLayer의 인덱스를 한단계 빠르게 하고 ContentTable에서 Content를 한단계 올린다
void	removeActiveLayers() 활성화된 레이어들을 제거한다
void	removeAllLayers() 모든 레이어를 제거한다
void	removeLayer(int idx) 인덱스를 이용하여 레이어를 제거한다
void	saveShapeFile() Shapefile을 저장한다(아직 구현되지 않음)

O CoordListener 클래스

- GraphicView 상의 커서 위치를 Status 에 출력한다.

public class CoordListener
extends java.lang.Object
implements java.awt.event.MouseMotionListener

생성자 요약
CoordListener(MapController conMap) MapController를 설정하는 생성자

메쏘드 요약	
Void	mouseMoved(java.awt.event.MouseEvent me) 마우스의 현재 좌표를 Status에 나타낸다

O DefaultSymbol 클래스

- DefaultSymbol 클래스는 디폴트 심볼 정보를 갖고 있다

public final class DefaultSymbol

extends java.lang.Object

필드 요약	
static int	BDIAGONAL 폴리곤 텍스쳐 심볼 모양: Backward Diagonal
static int	Circle 포인트 모양: Circle
static int	Cross 포인트 모양: Cross
static int	CROSS 폴리곤의 텍스쳐 심볼 모양: Cross
static int	DCROSS 폴리곤의 텍스쳐 심볼 모양: Diagonal Cross
static int	DEFAULT_IMAGE_SIZE 폴리곤의 디폴트 텍스쳐 심볼 크기: 4
static int	DEFAULT_OUTLINE_WIDTH 폴리곤의 디폴트 외곽선 두께: 1
static double	DEFAULT_POINT_ANGLE 포인트의 디폴트 회전각도: 0.0d
static java.awt.Color	DEFAULT_POINT_BCOLOR 포인트의 디폴트 배경색: Black
static java.awt.Color	DEFAULT_POINT_FCOLOR 포인트의 디폴트 내부색: Green
static int	DEFAULT_POINT_SHAPE 포인트의 디폴트 모양: Circle
static int	DEFAULT_POINT_SIZE 포인트의 디톨트 크기: 3
static java.awt.Color	DEFAULT_POLYGON_BCOLOR 폴리곤의 디폴트 배경색: White
static java.awt.Color	DEFAULT_POLYGON_FCOLOR 폴리곤의 디폴트 내부색: Green
static java.awt.Color	DEFAULT_POLYGON_OCOLOR 폴리곤의 디폴트 외곽선 색: Black
static int	DEFAULT_TEXTURE 폴리곤의 디폴트 텍스쳐: NO_TEXTURE

static int	Diamond 포인트의 모양: Diamond
static int	DOT 폴리곤 텍스쳐 모양: Dot
static int	FDIAGONAL 폴리곤 텍스쳐 모양: Forward Diagonal
static int	FilledCircle 포인트의 모양: Filled Circle
static int	FilledDiamond 포인트의 모양: Filled Diamond
static int	FilledSquare 포인트의 모양: Filled Square
static int	FilledTriangle 포인트의 모양: Filled Triangle
static int	HORIZONTAL 폴리곤 텍스쳐의 모양: Horizontal
static int	NO_TEXTURE 폴리곤 텍스쳐의 모양: 텍스쳐 없음
static int	Plus 포인트의 모양: Plus
static int	Square 포인트의 모양: Square
static int	Triangle 포인트의 모양: Triangle
static int	VERTICAL 폴리곤 텍스쳐의 모양: Vertical

생성자 요약
DefaultSymbol() 기본 생성자

O GeoView 클래스

- 실제 사용자가 사용하는 Graphic User Interface 이다. 상단에 OpenMenu, OpenTool, 좌측에 ContentTable, 중앙에 GraphicView, 하단에 Status 로 이루어져 있다.

public class GeoView
extends javax.swing.Jframe

생성자 요약
GeoView(java.lang.String name) 타이틀을 설정하는 생성자

메쏘드 요약	
static void	main(java.lang.String[] args) main 함수

O Identifier 클래스

- 이벤트로 입력된 포인트를 포함하는 ShapeObject 에 대한 MIS 테이블 레코드(속성)를 보여준다.

public class Identifier
extends com.openworld.openview.GraphicViewController
implements java.awt.event.MouseListener

생성자 요약
Identifier(MapController conMap) MapController를 설정하는 생성자

메쏘드 요약	
void	mouseClicked(java.awt.event.MouseEvent evt) 이벤트로 입력되는 포인트를 포함하는 ShapeObject를 찾고 ShapeObject의 속성값을 보여주는 프레임을 생성하며, 선택된 ShapeObject의 심볼을 빨간 심볼로 설정한다

O MapController 클래스

- ESRI shape 파일을 그래픽뷰에 로딩하고 조작하는 클래스. (class for handling GraphicView, loading Geodata and making Image Stream...etc)

public class MapController
extends javax.swing.JPanel
implements com.openworld.openview.ControllerListener

생성자 요약
MapController(java.awt.Frame frm) 기본 생성자

메쏘드 요약	
void	controllerChanged(com.openworld.openview.ControllerChangedEvent evt) : 특별한 구현 내용 없음.
int	countGraphicLayer() GraphicModel에 포함된 GraphicLayer의 수를 반환한다
void	createOverview() Overview를 생성한다
void	createTable() ContentPane에서 활성화된 레이어들의 테이블을 각각 생성한다
void	createTable(int layer_index) 레이어 인덱스에 대한 GraphicLayer의 RecordSet 테이블을 생성한다
void	createTable(java.lang.String layer_name) 레이어명을 가진 GraphicLayer의 RecordSet 테이블을 생성한다
SimplePoint	getCentroid(com.openworld.openview.ShapeObject shpObj) ShapeObject의 중심점을 반환한다. 각 ShapeObject가 가지는 포인트들의 산술 평균이 되는 포인트를 반환하며 라벨을 디스플레이할 때 이 포인트를 기준으로 한다.

ContentPane	getContentPanel() ContentPanel을 반환한다
void	getCoord(int x, int y) 사용자가 마우스를 GraphicView 상에서 움직일 때마다 Status에 변경된 좌표값을 나타낸다
Java.lang.Object[]	getFieldName(java.lang.String layer_name) 레이어의 필드명을 배열로 반환한다
Java.lang.Object[]	getFieldValue(java.lang.String layer_name, java.lang.String fieldname) 입력된 레이어와 필드명에 해당하는 컬럼값을 배열로 반환한다
GraphicModel	getGraphicModel() GraphicModel을 반환한다
GraphicView	getGraphicView() GraphicView를 반환한다
OpenMenu	getOpenMenu() OpenMenu를 반환한다
OpenTool	getOpenTool() OpenTool을 반환한다
float	getScale() 현재 GraphicView의 축척을 반환한다
ShapeObject	getShapeObject(int gid) 입력된 GID를 가진 ShapeObject를 반환한다
ShapeObject	getShapeObject(java.lang.String input_value, java.lang.String field_name, int index) 입력된 속성값을 가진 ShapeObject를 반환한다
java.util.Vector	getShapeObjects(java.lang.String input_value, java.lang.String field_name, java.lang.String layer_name) 입력된 속성값을 가진 ShapeObject를 반환한다
Status	getStatus() Status를 반환한다
SymbolController	getSymbolController() SymbolController를 반환한다
boolean	hasGraphicLayer(java.lang.String file_name) 로드할 파일이 GrahpicModel에 이미 있으면 true를 반환한다. 새로운 파일을 로드할 경우 이 메소드를 호출하

	며 true가 반환되면 로드하지 않는다.
int	loadShapeFile(java.lang.String _path, java.lang.String _name, boolean bln) Shapefile을 GraphicModel에 로드하고 레이어 인덱스를 반환한다
void	makeLabel() 활성화된 레이어에 대한 라벨을 생성한다
void	makeLabelLayer(java.lang.String field_name, int index) 입력된 레이어와 필드명을 이용하여 라벨 레이어를 생성한다. 라벨의 기준점은 getCentroid()의 반환값을 사용한다.
void	makeLabelLayer(java.lang.String field_name, java.lang.String layer_name) 입력된 레이어와 필드명을 이용하여 라벨 레이어를 생성한다
boolean	moveLayerBottom(java.lang.String strCont) 레이어의 인덱스를 마지막으로 설정한다. ContentTable 상에서 Content의 위치가 가장 아래쪽으로 바뀌고, GraphicModel에서는 인덱스가 제일 작은 값이 된다. 메소드가 완료되면 true를 반환한다.
boolean	moveLayerDown(java.lang.String strCont, boolean bln) 레이어의 인덱스를 1만큼 증가시킨다. ContentTable에서는 인덱스가 하나 증가하고 GraphicModel에서는 인덱스가 하나 감소한다. 메소드가 완료되면 true를 반환한다.
boolean	moveLayerTop(java.lang.String strCont) 레이어의 인덱스를 1로 설정한다
boolean	moveLayerUp(java.lang.String strCont, boolean bln) 레이어의 인덱스를 1만큼 감소시킨다
void	msrDistance(java.awt.Point p1, java.awt.Point p2) 두 포인트 간의 거리를 계산하고 Status에 나타낸다
void	printGraphicView() 현재의 GraphicView를 프린트한다
void	removeAllCtrl() 모든 GrahpicViewController를 GraphicView에서

	제거한다
void	removeGraphicLayer(int idx) 입력된 인덱스를 가진 GraphicLayer를 제거한다
void	removeGraphicLayer(java.lang.String layer_name) GraphicLayer를 제거한다
void	removeLabel() 라벨 레이어를 제거한다
void	removeOverview() Overview를 제거한다
void	removeSelCtrl() ZoomoutController를 GraphicView에서 제거한다
void	ScaleZoom(float fltScale) 입력 축척에 맞게 GraphicView의 축척을 바꾼다. 스케일을 일벽하는 텍스트필드에서 이벤트가 발생할 경우 실행되는 메소드이다. 현재의 축척을 입력된 값과 비교하여 GraphicView의 축척을 변경한다.
void	setBxZminCtrl() ZoominController를 GraphicView에 설정한다
void	setBxZmoutCtrl() ZoomoutController를 GraphicView에 설정한다
void	setIdCtrl() Identifier를 GraphicView에 GraphicViewController로 설정한다
void	setMeasureCtrl() MeasureController를 GraphicView에 설정한다
void	setPanCtrl() PanController를 GraphicView에 설정한다
void	setScale(float fltNewScale) 입력된 축척값에 맞게 GraphicView의 축척을 설정한다
void	setSelCtrl() SelectionController를 GraphicView에 설정한다
void	setViewToLayer(java.lang.String layerName) 활성환된 하나의 레이어의 MBR에 맞게 GraphicView의 축척을 설정한다
void	setViewToLayer(java.lang.String[] layerName)

	활성화된 여러 레이어 배열의 MBR에 맞게 GraphicView의 축척을 설정한다
void	setViewToSelectedObject() 선택된 ShapeObject의 MBR에 맞게 GraphicView의 축척을 설정한다
void	setViewToShapeObject(com.openworld.openview.ShapeObject[] shape) ShapeObject 배열의 MBR에 맞게 GraphicView의 축척을 설정한
void	showGraphicLayer(int idx, boolean bln) GraphicLayer를 GraphicView 상에 보이게/보이지 않게 설정한다
void	showGraphicLayer(java.lang.String strCont, boolean bln) GraphicLayer를 GraphicView 상에 보이게/보이지 않게 설정한다
boolean	swapShapeLayer(int up, int dn, boolean bln) 두 GraphicLayer의 인덱스를 바꾼다. 레이어명과 심볼 정보 등 레이어에 관련된 여러 정보도 같이 변경된다.
boolean	swapShapeLayer(java.lang.String up_name, java.lang.String dn_name, boolean bln) 두 GraphicLayer의 인덱스를 바꾼다

O MeasureController 클래스

- 이벤트로 입력받은 화면상의 두 지점간의 거리를 측정하는 기능을 제공한다.

public class MeasureController
extends com.openworld.openview.GraphicViewController
implements java.awt.event.MouseListener, java.awt.event.MouseMotionListener, java.lang.Runnable

생성자 요약
MeasureController(MapController conMap) MapController를 설정하는 생성자

메쏘드 요약	
void	mouseDragged(java.awt.event.MouseEvent me) 드래그 시작점으로부터 현재 커서까지의 거리를 계산하고 Status에 나타낸다
void	mouseEntered(java.awt.event.MouseEvent e) 커서를 CROSSHAIR 커서로 설정한다
void	mouseExited(java.awt.event.MouseEvent e) 커서를 디폴트 커서로 설정한다
void	mousePressed(java.awt.event.MouseEvent e) 마우스 드래그 시작점을 설정한다
void	run() 드래그 선을 그린다

O OpenMenu 클래스

- 사용자가 프로그램의 기능을 수행할 수 있게 하는 메뉴를 제공한다.

public class OpenMenu
extends javax.swing.JMenuBar
implements java.awt.event.ActionListener

생성자 요약
OpenMenu(MapController conMap) MapController를 설정하는 생성자

메쏘드 요약	
void	actionPerformed(java.awt.event.ActionEvent evt) 메뉴아이템의 액션을 설정한다. 메뉴는 File, Edit, View, Table, Help이 있으며 메뉴아이템은 File에 Close, Close All, Save, Print, Exit, Edit에 Remove Theme, Remove All Theme, View에 Add Theme, Add Event Theme, New Theme, Themes On, Themes Off, Full Extent, ZoomIn, Zoom Out, Zoom To Themes, Zoom To Selected, Find, Table에 Convert to Shapefile, Auto-Label, Remove Labels, Remove Overlapping Labels, Convert OverLapping Labels, Table, Query, Select By Theme, Create Buffer, Clear Seleted Feature 등이

	있다.

O OpenTool 클래스

- 사용자가 프로그램의 기능을 수행할 수 있게 하는 버튼, 툴을 제공한다.

public class OpenTool
extends javax.swing.JToolBar
implements java.awt.event.MouseListener

생성자 요약
OpenTool(MapController conMap) MapController를 설정하는 생성자

메쏘드 요약	
Void	mouseClicked(java.awt.event.MouseEvent evt) 버튼의 액션을 설정한다. OpenMenu에 해당되는 버튼과 툴이 있다.

O Overview 클래스

- 현재 GraphicView 의 범위를 전체 범위와 비교하여 보여주며, 화면을 조절할 수 있는 기능을 제공한다.

public class Overview
extends com.openworld.openview.GraphicView

생성자 요약
Overview(com.openworld.openview.GraphicView target) GraphicView를 설정하는 생성자

메쏘드 요약	
void	setBounds(int x, int y, int width, int height) GraphicView의 메소드를 오버로드. 경계상자에 맞게 GraphicView의 축척을 조절한다. Overview에 보이는 반투명 상자의 크기

	를 변경하거나 위치를 변경하면 실제 LMIS View 상의 GraphicView에 그대로 반영된다.

O PanController 클래스

- GraphicView 의 시점을 원하는 지점으로 이동하게 하는 기능을 제공한다.

public class PanController
extends com.openworld.openview.GraphicViewController
implements java.awt.event.MouseListener, java.awt.event.MouseMotionListener

생성자 요약
PanController(MapController conMap) MapController을 설정하는 생성자

메쏘드 요약	
void	mouseEntered(java.awt.event.MouseEvent e) 커서를 Hand 커서로 바꾼다
void	mouseExited(java.awt.event.MouseEvent e) 커서를 디폴트 커서로 바꾼다
void	mousePressed(java.awt.event.MouseEvent e) 시작점을 설정한다
void	mouseReleased(java.awt.event.MouseEvent e) 릴리스 포인트와 시작점간의 거리를 계산하여 GraphicView를 이동시킨다

O PointPreview

- PointSymbolChooser 에서 변경한 포인트 심볼 유형을 반영한다.

public class PointPreview
extends javax.swing.JPanel

생성자 요약
PointPreview()

기본 생성자
PointPreview(java.awt.Color fcolor, java.awt.Color bcolor, int point_shape, int point_size, int point_angle) foreground, background, outline 색상과 fill 패턴, 패턴 크기, outline 두께를 설정하는 생성자

메쏘드 요약	
java.awt.Color	getBackgroundColor() background 색상을 반환한다
java.awt.Color	getForegroundColor() foreground 색상을 반환한다
double	getPointAngle() point의 각도를 반환한다
int	getPointSize() point의 크기를 반환한다
int	getPointType() point의 타입을 반환한다
void	paint(java.awt.Graphics g) JPanel의 메소드를 오버로드
void	setBackgroundColor(java.awt.Color clr) background 색상을 설정한다
void	setForegroundColor(java.awt.Color clr) foreground 색상을 설정한다
void	setPointAngle(double ang) point의 각도를 설정한다
void	setPointSize(int size) point의 크기를 설정한다
void	setPointType(int shp) point의 타입을 설정한다

O PolygonPreview

- PolygonSymbolChooser 에서 변경한 포인트 심볼 유형을 반영한다.

public class PolygonPreview

extends javax.swing.JPanel

생성자 요약
PolygonPreview() 기본 생성자
PolygonPreview(java.awt.Color fcolor, java.awt.Color bcolor, java.awt.Color ocolor, int pattern, int bsize, int width) foreground, background, outline 색상과 fill 패턴, 패턴 크기, outline 두께를 설정하는 생성자
PolygonPreview(SymbolArray arrayS) SymbolArray를 입력받아 심볼을 설정하는 생성자

메쏘드 요약	
Java.awt.Color	getBackgroundColor() background 색상을 반환한다
int	getFillPattern() fill 패턴을 반환한다
Java.awt.Color	getForegroundColor() foreground 색상을 반환한다
int	getImageSize() fill 패턴의 크기를 반환한다
Java.awt.Color	getOutlineColor() outline 색상을 반환한다
int	getOutlineWidth() outline의 두께를 반환한다
SymbolArray	getSymbolArray() PolygonPreview가 가지고 있는 심볼을 SymbolArray 형태로 반환한다
void	paint(java.awt.Graphics g) JPanel의 메소드를 오버로드
void	setBackgroundColor(java.awt.Color clr) background 색상을 설정한다
void	setFillPattern(int _pattern) fill 패턴을 설정한다
void	setForegroundColor(java.awt.Color clr) foreground 색상을 설정한다

void	setImageSize(int bsize) fill 패턴의 크기를 설정한다
void	setOutlineColor(java.awt.Color clr) outline 색상을 설정한다
void	setOutlineWidth(int width) outline의 두께를 설정한다
void	setSymbolArray(SymbolArray arrayS) 입력받는 SymbolArray로 PolygonPreview의 심볼을 설정한다

O PolygonSymbolChooser

- 폴리곤의 심볼을 변경하는 인터페이스를 제공한다.

public class PolygonSymbolChooser
extends javax.swing.JPanel
implements java.awt.event.ActionListener

생성자 요약
PolygonSymbolChooser() 기본 생성자
PolygonSymbolChooser(PolygonPreview prvPolygon) 타입을 Single로 설정하는 생성자
PolygonSymbolChooser(UniqueSymbolChooser chooserUS) 타입을 Unique로 설정하는 생성자

메쏘드 요약	
Void	actionPerformed(java.awt.event.ActionEvent evt) 버튼의 액션을 설정한다

O QueryBuilder 클래스

- 속성 질의를 할 수 있는 인터페이스를 제공한다.

public class QueryBuilder
extends javax.swing.JFrame
implements java.awt.event.MouseListener

생성자 요약
QueryBuilder() 기본 생성자
QueryBuilder(MapController conMap) MapController를 설정하는 생성자

메쏘드 요약	
Void	mouseClicked(java.awt.event.MouseEvent evt) 버튼의 액션을 설정한다
Void	setLayerName(java.lang.String str) 레이어의 이름을 설정하고 타이틀을 설정한다

O ShapeFilter 클래스

- 파일을 로딩할 때 Shapefile 만을 디스플레이할 수 있게 한다.

public class ShapeFilter
extends javax.swing.filechooser.FileFilter

생성자 요약
ShapeFilter() 기본생성자

메쏘드 요약	
boolean	accept(java.io.File f) 디렉토리와 shapefile만을 받아들인다
java.lang.String	getDescription() 이 파일필터의 설명을 반환한다
Static java.lang.String	getExtension(java.io.File f)

	입력된 파일의 확장자를 반환한다
Static java.lang.String	getName(java.io.File f) 입력된 파일을 이름만을 반환한다

O SingleSymbolChooser 클래스

- GraphicLayer 의 심볼 타입을 Single 로 변경할 수 있는 인터페이스를 제공한다.

public class SingleSymbolChooser
extends javax.swing.JPanel

생성자 요약
SingleSymbolChooser() 기본 생성자
SingleSymbolChooser(SymbolArray symbol_array) SymbolArray를 설정하는 생성자

메쏘드 요약	
SymbolArray	getSymbolArray() PolygonPreview에서 설정된 SymbolArray를 반환한다
void	setSymbolArray(SymbolArray arrayS) 입력된 SymbolArray를 PolygonPreview에서 설정한다

O Status 클래스

- 커서의 좌표를 보여주며 거리의 계산 결과와 진행상태를 보여준다.

public class Status
extends javax.swing.JPanel

생성자 요약

Status() 생성자
Status(java.lang.String strXY) 커서의 위치에 대한 라벨의 텍스트 설정하는 생성자

메쏘드 요약	
java.lang.String	getLblXY() 커서의 위치를 반환한다
void	setLblDist(java.lang.String strDist) 거리 계산의 결과에 대한 라벨의 텍스트를 설정한다
void	setLblXY(java.lang.String strXY) 커서의 위치에 대한 라벨의 텍스트를 설정한다
void	setPrgsBar() 진행상태바를 설정한다

O SymbolArray 클래스

- GrahpicLayer(Polygon 타입)에 대한 심볼정보(색상, 패턴, 크기 등)를 저장한다.

public class SymbolArray
extends java.lang.Object

필드 요약	
static int	ETC_TYPE 레이어의 데이터 타입: 기타
static int	LABEL_TYPE 레이어의 데이터 타입: 라벨
static int	LINE_TYPE 레이어의 데이터 타입: 폴리라인
static int	POINT_TYPE 레이어의 데이터 타입: 포인트
static int	POLYGON_TYPE 레이어의 데이터 타입: 폴리곤

생성자 요약
SymbolArray() 생성자
SymbolArray(java.awt.Color fcolor, java.awt.Color bcolor, java.awt.Color ocolor, int fpattern, int imgsize, int width) foreground, background, outline의 색상과 fill 패턴, 패턴 크기, outline 굵기를 각각 설정하는 생성자
SymbolArray(SymbolArray array_sym) SymbolArray를 입력으로 받는 생성자

메쏘드 요약	
java.awt.Color	getBackgroundColor() background 색상을 반환한다
int	getDataType() 레이어의 데이터 타입을 반환한다
int	getFillPattern() fill 패턴을 반환한다
java.awt.Color	getForegroundColor() foreground 색상을 반환한다
int	getImageSize() 패턴의 크기를 반환한다
java.awt.Color	getOutlineColor() outline 색상을 반환한다
int	getOutlineWidth() 외곽선의 두께를 반환한다
void	setBackgroundColor(java.awt.Color clr) background 색상을 설정한다
void	setDataType(int type) 레이어의 데이터 타입을 설정한다
void	setFillPattern(int _pattern) fill 패턴을 설정한다
void	setForegroundColor(java.awt.Color clr) foreground 색상을 설정한다
void	setImageSize(int imgsize) 패턴의 크기를 설정한다

void	setOutlineColor(java.awt.Color clr) outline 색상을 설정한다
void	setOutlineWidth(int width) 외곽선의 두께를 설정한다

O SymbolController 클래스

- SymbolChooser 에서 선택한 심볼에서 얻어지는 SymbolArray 를 이용하여 실제 GraphicLayer 의 심볼을 변경하는 기능을 제공한다. 그리고 현재 GraphicLayer 가 가지고 있는 SymbolArray 를 리턴하는 기능을 제공한다.

public class SymbolController
extends java.lang.Object

생성자 요약
SymbolController(MapController conMap) MapController를 설정하는 생성자

메쏘드 요약	
SymbolArray	getLayerSymbol(java.lang.String layer_name) Single 타입을 심볼일 경우 레이어 전체의 심볼을 반환한다
SymbolArray[]	getShapeObjectsSymbol(java.lang.String layer_name) 레이어의 모든 ShapeObject에 대한 각각의 심볼을 배열로 반환한다
SymbolArray	getShapeObjectSymbol(com.openworld.openview.ShapeObject shape, boolean random_flg) ShapeObject에 대한 심볼을 반환한다
void	setDefaultLayerSymbol(int layer_index) 레이어에 디폴트 심볼을 설정한다 Single 심볼인 경우 foreground 색상을 랜덤하게 설정된다
void	setLayerSymbol(SymbolArray arraySymbol, java.lang.String layer_name) 입력된 레어어 전체에 입력된 심볼을 설정한다

void	setShapeObjectsSymbol(java.lang.Object[][] objSymbol, java.lang.String field_name, java.lang.String layer_name) SymbolEditor로부터 입력받은 Preview, 필드값 배열을 이용하여 레이어의 모든 ShapeObject의 심볼을 설정한다
void	setShapeObjectSymbol(SymbolArray arrayS, com.openworld.openview.ShapeObject shape) 입력된 ShapeObject에 입력된 심볼을 설정한다

O SymbolEditor 클래스

- 실제 사용자가 심볼을 변경할 때 사용하는 Graphic User Interface 이다.

public class SymbolEditor
extends javax.swing.JFrame
implements java.awt.event.ActionListener, java.awt.event.ItemListener

생성자 요약
SymbolEditor() 기본 생성자
SymbolEditor(MapController conMap) MapController를 설정하는 생성자

메쏘드 요약	
void	actionPerformed(java.awt.event.ActionEvent evt) Apply, Cancel 버튼의 액션을 설정한다
SymbolArray	getSingleSymbolArray() 심볼 타입이 Single일 때 SymbolArray를 반환한다
void	itemStateChanged(java.awt.event.ItemEvent evt) 심볼 타입 콤보박스의 액션을 설정한다
void	setSingleLayerSymbol() 심볼 타입이 Single일 때 현재의 SymbolArray를 가져와서 레이어의 심볼을 설정한다
void	setUniqueLayerSymbol() 심볼 타입이 Unique일 때 현재의 SymbolArray배열과

	분류기준이 되는 필드명을 가져와서 레이어의 심볼을 설정한다

O UniqueSymbolChooser 클래스

- GraphicLayer 의 심볼 타입을, 특정 필드값에 따른 Unique 로 변경할 수 있는 인터페이스를 제공한다.

public class UniqueSymbolChooser
extends javax.swing.JPanel

생성자 요약
UniqueSymbolChooser() 기본 생성자
UniqueSymbolChooser(MapController conMap, java.lang.String layer_name) MapController와 레이어명을 설정하는 생성자

메쏘드 요약	
java.lang.String	getFieldName() 현재 레이어에서 심볼설정의 기준이 되는 필드명을 반환한다
PolygonPreview	getSelectedPolygonPreview() PreviewTable에서 선택된 레코드의 PolygonPreview를 반환한다
Java.lang.Object[][]	getSymbolNField() 설정된 심볼과 필드값의 배열을 반환한다

O UniqueSymbolTableModel 클래스

- PreviewTable 에 적용되는 TableModel 이다.

public class UniqueSymbolTableModel
extends javax.swing.table.AbstractTableModel

생성자 요약
UniqueSymbolTableModel(PolygonPreview[] prvPolygon, java.lang.Object[] valueField) PolygonPreview 배열과 Object 배열을 입력받는 생성자

메쏘드 요약	
java.lang.Class	getColumnClass(int columnIndex) 컬럼의 클래스를 반환한다
int	getColumnCount() 컬럼 수를 반환한다
java.lang.String	getColumnName(int columnIndex) 컬럼 이름을 반환한다
int	getRowCount() 레코드 수를 반환한다
java.lang.Object	getValueAt(int rowIndex, int columnIndex) 입력 값으로부터 셀값을 설정한다
boolean	isCellEditable(int rowIndex, int columnIndex) 셀이 편집가능하면 true를 반환한다
void	setValueAt(java.lang.Object aValue, int rowIndex, int columnIndex) 셀값을 편집한다

O VerticalFlowLayout 클래스

- 수직으로 Content를 배열하게 하는 기능을 제공한다.

public class VerticalFlowLayout
extends java.awt.FlowLayout
implements java.io.Serializable

필드 요약	
static int	BOTTOM 내부 패널의 정렬 모양: 아래로 정렬

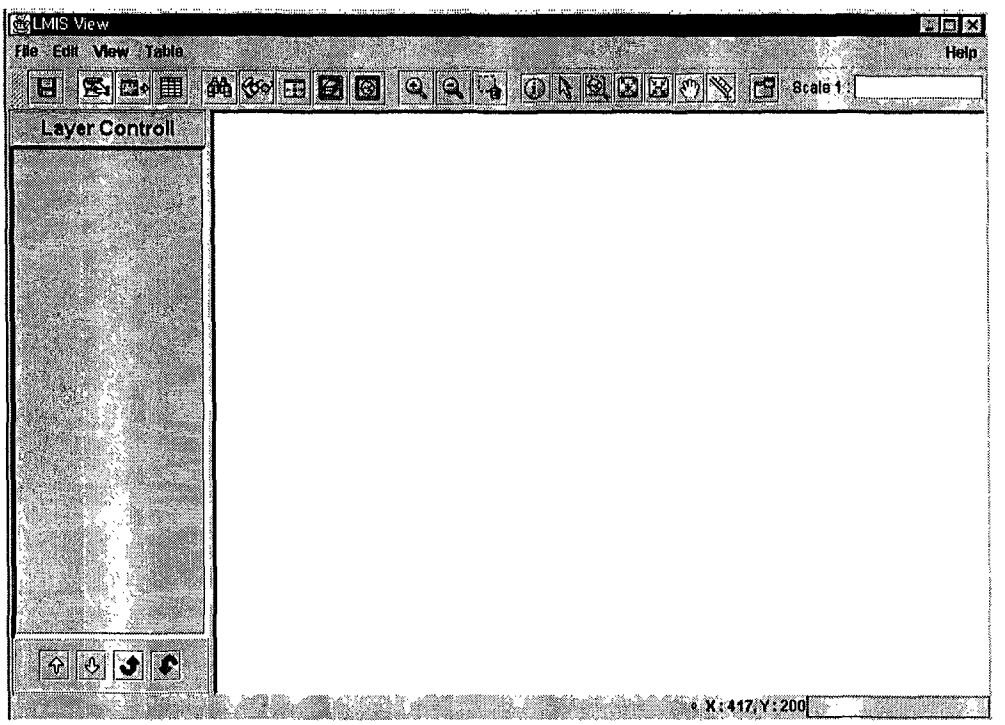
static int	MIDDLE 내부 패널의 정렬 모양: 가운데 정렬
static int	TOP 내부 패널의 정렬 모양: 위로 정렬

생성자 요약
VerticalFlowLayout() 기본생성자

5. 프로토타입 시스템 구현

가. LMIS View의 초기 화면

- 상단에 메뉴바와 툴바, 왼쪽에 레이어관리 테이블, 가운데에 뷰, 그리고 하단에 상태표시바로 이루어져 있다.



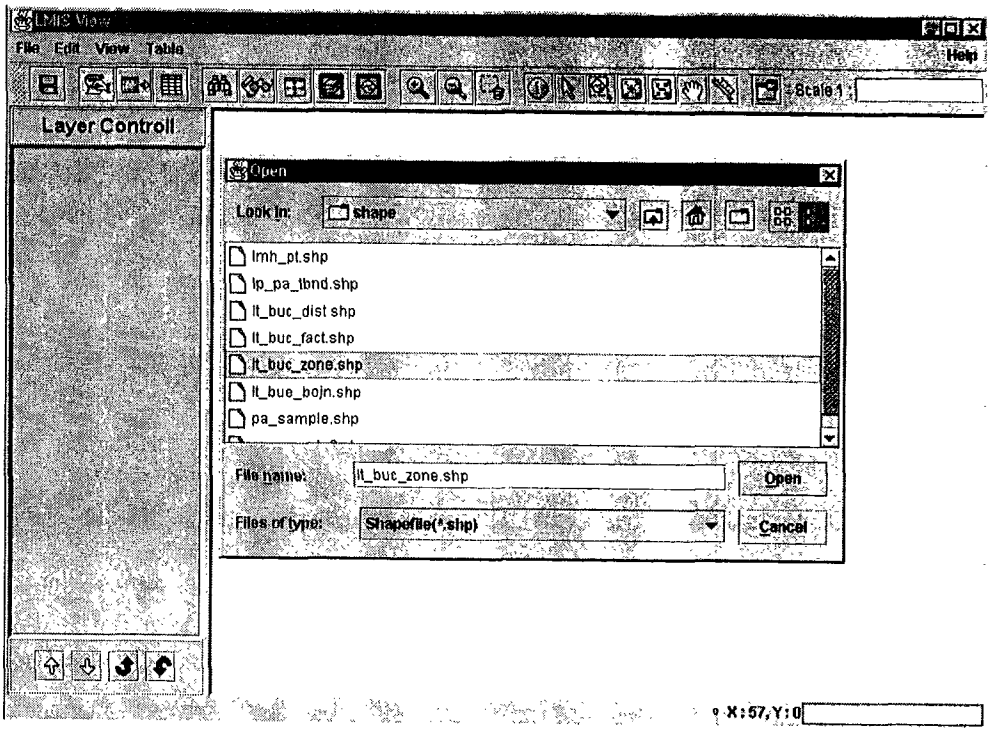
[그림 88] LMIS View 초기 화면

[그림 89] 버튼

[그림 90] 툴

나. 데이터 로딩

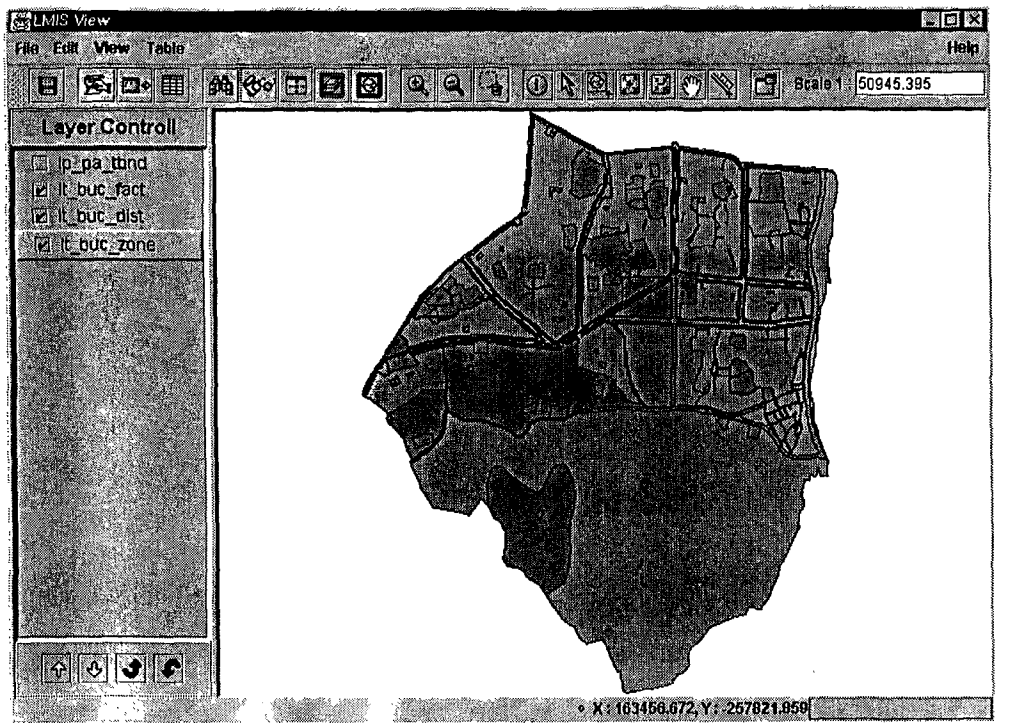
- 데이터 로드 버튼을 누르면 파일 선택 대화상자가 나타나며 ShapeFilter에 의해 디렉토리와 shapefile만 나타난다.



[그림 91] 파일 선택 대화상자

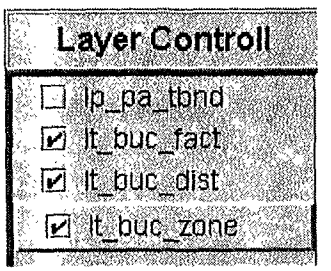
- 파일을 선택하면 GraphicView에 레이어가 추가되며 동시에 ContentTable에

Content가 생성된다.

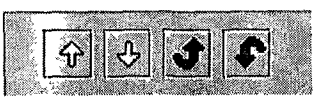


[그림 92] 파일이 로드된 화면

- ContentTable 내부의 Content들의 순서는 레이어 관리 버튼에 의해 변경할 수 있다.



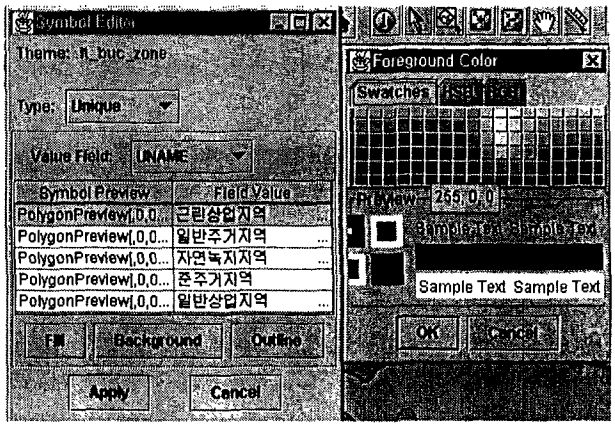
[그림 93] ContentTable



[그림 94] 레이어 관리 버튼

다. 심볼 변경

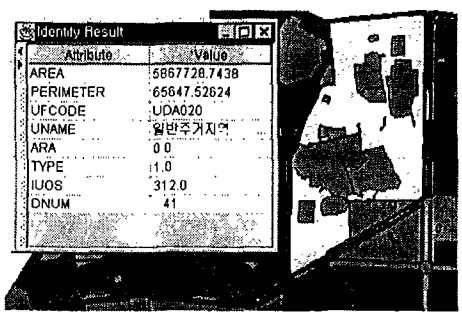
- 레이어를 활성화한 상태에서 심볼 편집기를 선택하면 자동으로 현재 레이어의 심볼이 반영된다. 내부색, 바탕색, 외곽선 색, Fill 패턴, 패턴 크기, 외곽선 굵기 등을 선택할 수 있다.



[그림 95] 심볼 편집기(내부색을 변경하는 작업중)

라. 속성 검색(뷰)

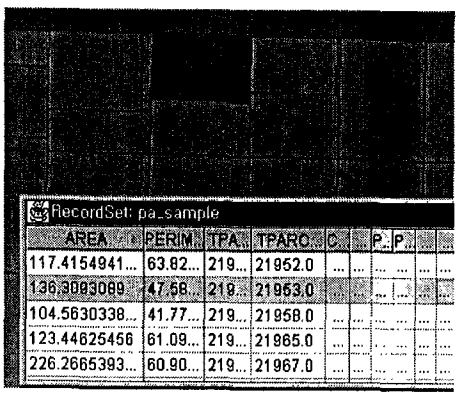
- 메뉴 바에서 Identify(사상 확인) 버튼을 클릭한 다음 커서를 뷰 창으로 가져가 확인하고자 하는 사상을 선택한다. 사상이 하이라이트 되고 해당 속성을 보여주는 창이 나타난다.



[그림 96] 속성 검색(뷰)

마. 속성 검색(테이블)

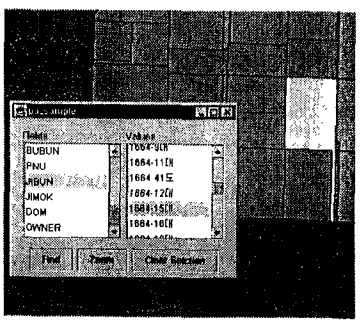
- 레이어를 활성화하고 테이블 열기 버튼을 클릭하면 레이어의 모든 속성이 테이블 창에 나타난다. 테이블의 레코드를 선택하면 해당 레코드를 속성으로 갖는 사상이 뷰에서 선택된다.



[그림 97] 사상 검색(테이블)

바. 사상 검색(질의)

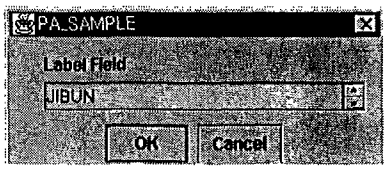
- 검색 버튼을 클릭하여 QueryBuilder 창을 띄우고 나서 속성을 선택하면, 입력된 속성을 갖는 사상이 뷰 상에서 선택된다. 예) 지번 검색



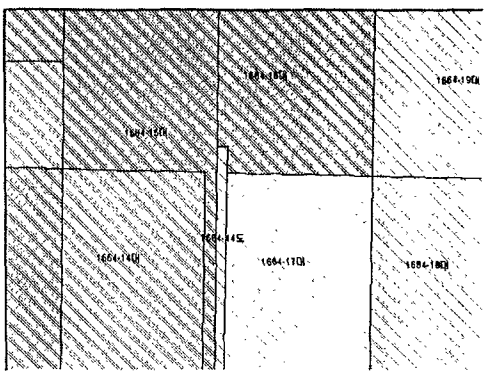
[그림 98] 사상 검색(질의)

사. 레이블 보기

- 테이블 메뉴에서 자동 라벨링을 선택하면 라벨링 대상 필드를 선택할 수 있는 대화상자가 나오고, 선택된 필드명에 해당하는 속성값을 사상의 기하 중심점에 디스플레이한다.



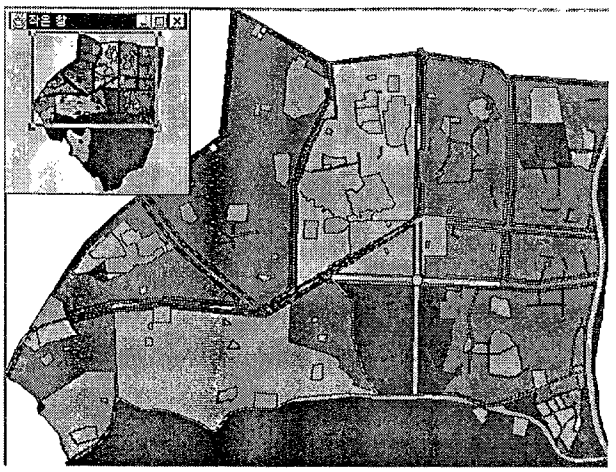
[그림 99] 라벨 필드 선택상자



[그림 100] 라벨이 추가된 그래픽뷰

아. Overview 창

- OverView는 뷰가 확대되었을 경우 전체 범위에서 현재 뷰가 차지하는 위치와 크기를 보여준다. 그리고 OverView 상에서의 조작을 통해서도 뷰의 축척과 위치를 변경할 수 있다.



[그림 101] OverView 창과 그래픽 뷰

제 9 절 공간 통계 분석기

1. 공간 분석 기법

- 공간통계기법들과 GIS 시스템의 통합 및 연계(link)에 있어서 Openshaw는 기존의 GIS 시스템이 갖는 공간 분석 기능과 통계적 기능을 연계하는 것에 대해 반대하고 풍부한 GIS 데이터의 요구에 부응하는 새롭고 일반적인 공간 분석방법 개발을 주장하였다. 그러나, 현실적으로는 기존의 GIS의 공간 분석 기법들을 사용하는 것과 더불어 일반적이고 보편적으로 적용될 수 있는 공간 통계 기법들을 추가하는 방법이 좀더 유용하다. 따라서 GIS와 연계하기에 적합한 공간 통계 분석 기법들을 선택할 필요가 있다. 일반적이고 유용한 공간 통계 기법을 선택하는 기준으로는 다음과 같은 항목들을 들 수 있다.

  - GIS 시스템에서 제공하는 공간 정보를 보다 현실적으로 표현하는데 충분히 유용한가?
  - 많은 분야에서 폭넓게 이용 가능한가?
  - 계산이 용이한가?
  - 공간적 이질성에 대한 탐사 및 비정규적인 그래프화에 도움이 되는가?
  - 개념적으로 관련되거나 함께 고려되어야 할 비교적 적은 수의 자료에도 적용이 가능한가?

- 위의 항목을 기준으로 기존의 공간 통계 분석 기법들을 분류하면 다음 표와 같다.

Data Structure	Dimensionality	
	Univariate	Multivariate
Locational Data	Nearest Neighbour Methods K-functions	Bivariate K Functions Space Time Interaction
	Kernel Density Estimation	Adaptive Kernel Density Estimation

Attribute Data	Kernel Regression Bayesian smoothing - ICM	Kernel Regression Bayesian smoothing - ICM
	Spatial Autocorrelation Spatial Correlograms Variograms	Multivariate Spatial Correlation
	Trend Surface Analysis Kriging	Spatial Regression Co-Kriging Spatio-temporal models
		Spatial General Linear Modelling
		Cluster Analysis Canonical Correlation Multidimensional scaling
Interaction Data	Spatial Interaction Methods	Augmented Spatial Interaction models

[표 5] 공간 통계 분석 기법의 분류

- 위의 표에서 위치 데이터는 사상들의 단순위치들로 구성된 데이터이고 이에는 event data, object data, point process가 있으며 그 예로는 특정지역내에 질병 발생 위치들, 서로 다른 형태의 사상들이 함께 발생하는 경우의 multi-data, 시계열적 경향들의 위치들이 차례로 표시되는 데이터 등을 들 수 있다. Attribute data는 위치와 관련된 값이나 속성 데이터이고 그 예로는 어떤 위치에서의 토양속성에 관한 데이터를 들 수 있다. Interaction data는 위치들의 pair이나 link와 관련된 각각의 양적인 측정치 데이터를 말한다.
- 또한 표에서는 단일변량과 다변량으로 나누고 있는데 단일변량 분석은 하나의 패턴을 분석하는 것으로 각 위치에서의 경향이나 공간적 자기상관 등을 파악하는 데 도움을 주는 것이다. 그 예로는 거주지와 shopping outlet간의 개별적인 흐름에 관한 데이터를 들 수 있다. 다변량분석은 하나 이상의 패턴들 간의 관계 또는 하나의 패턴과 다른 여러개의 패턴들과의 관계를 분석하는 것으로 각각 그 위치들에 대해서 공간적인 상관관계를 파악하는 것이다. 그 예로는 수용자(demand)로 성격지워지는 연결 원점들로부터 요구치(attractiveness)로 성격지워지는 목적지들까지의 방향량(vector of mesurements)에 관한 데이터를 들수 있을 것이다.
- 위의 표에서 제시한 공간 분석 기법들에 대한 개념과 이 분석 기법을 사용함으로써 얻을 수 있는 이득에 대해서 살펴보면 다음과 같다.

○ Nearest neighbor method and K-function

GIS에서 이러한 기법이 주는 이점은 사용자가 interactive하게 지도로부터 event를 직접 선택하여 data set으로부터 event를 삭제하거나 삽입하는 기능, 직접적으로든, 공간검색을 통해서든 study area를 dynamic하게 정의하는 기능, 연구지역을 동질한 하위지역으로 구분하고, 이들 지역 각각에서의 event를 알아내기 위해 point-in-polygon function을 사용하는 기능 등을 이용할 수 있게 한다는 것이다.

이것은 또한 면적과 경계를 계산하고, 지역의 topology로부터 직접 추출한 더 현실적인 event상호간 연결성을 측정하는 데 사용하며, 다양한 connectivity 측정에 기초를 둔 다른 order의 neighbor를 정의하는데 사용될 수 있다.

GIS기능에 활용할 다른 이점은 연구지역의 edge에서 발생하는 왜곡을 수정하는 것과 관련된 이러한 종류의 분석이 지닌 일반적인 문제와 관련이 있다. 보정기술은 경계모양의 상세한 지식을 요구한다. GIS기능은 이점에 관해서 특별히 유용하다. 그리고, 다른 edge correction이 정량적으로 다른 경계에 적용되어야 하는지를 연구하는 것과 같은 많은 새로운 가능성을 소개한다.

○ Kernel and Bayesian smoothing method

GIS에 이러한 기술을 접목시킬 때 생기는 이점은 주로 등고선이나 3-D projection의 결과물로 나온 surface를 visualize할 수 있는 분야에서 생긴다. 그들은 또한 local covariance structure 의 정의에 사용될 수 있는데, 예를 들어 variogram을 공간적으로 smooth된 지도를 생각해 볼 수 있을 것이다. 또한 smoothing을 physical query와 결합시킬 수도 있다.

edge effect를 수정하고, 공간적인 연결성, 인접성을 추출하고, 특히 불규칙한 속성자료의 smoothing을 하는 것 등의 GIS 기능을 개발하는 것이 이점이 있지만, 이러한 기술들은 상대적으로 덜 개발된 영역이다.

이 방법은 data set 에서 핵심적인 local feature는 남겨두고 data set 에서의 변이성을 제거하기 위한 비모수통계기법의 한 영역이다. 공간적 맥락에서 이것은 hot spot이나, 균일한 지역을 알아내고, 가능한 모델을 정의하고, 관찰된 데이터에 model이 얼마나 적합한지를 분석하는 중요한 기법이다.

○ Spatial autocorrelation and covariance structure

이 분야에 관련된 기술들은 모두 속성자료의 공간적 공분산구조를 찾는 것과 관련이 있다. 즉, 같이 변화하는 주변이나 인접 값들이 있는지, 있다면 어떠한 방식으로 이루어지는가를 알아내는 것이다.

이러한 기법들을 이용하는 데서 생기는 이점은 대개 W matrices로서 알려진 위치들 사이의 proximity matrices의 구성을 도와주는 영역에서 생긴다. W matrices는 많은 자기상관기법에서 필요한 입력이다. 이것들은 때로는 Euclidean distance, adjacency, 공통경계의 존재, 공유하는 경계선의 길이로 구성된다. 그러나, 그것의 잠재성은 물리적인 장벽을 설명하거나, 네트워크 구조에 관련되거나, 부가적인 interaction(flow)에 기초해있는 공간단위(areal unit)들 간의 더 세련된 관련성 측정을 이끌어내는 데에 있다.

지역의 공간선택(spatial selection of region)은 또한 특별히 전체 picture와는 두드러지게 다른 전체지역 중 일부 지역의 correlation구조에 대한 쉬운 연구를 가능하게 하는 application을 가진다.

variogram의 계산을 위해 공간상에서 방향을 interactive하게 정의하는 기능은 변이가 고정되어 있는지를, 만약 그렇다면 그것이 isotropic(등방향의) 한지, 반대로 anisotropic한지를 평가하는데 가치가 있다.

O Geostatistical and spatial econometric modelling

GIS에 이런 기법들을 적용하면 보다 현실적인 공간 가중 행렬들의 유도와 공간 공분산 구조의 설명에 관련된 것에서 유용하다. 또한 어떤 한계 외부의 점들을 주어진 지도나 표면의 형태속에 결과를 같이 표현할 수 있다. 특히 kriging의 경우는 내삽된 표면과 그것에 관련된 예측가능한 오차를 동시에 지리적으로 표현할수 있다. 사실 인위적으로 표면을 smoothing하는데 지금까지 주로 사용되어 온 표준 결정론적 tessellation 기법만큼이나 최근의 논문에서 kriging을 사용하고 있다. 그렇지만 일반적인 경우 공간 회귀 모델에 대한 수치적인 검증은 매우 크고 비대칭적인 공간 가중치의 행렬을 필요로 하기 때문에 GIS에 통합되기 보다는 전문적인 소프트웨어로 다루는 것이 효율적이다. 이것은 또한 bootstrap이나 jackknife resampling에 기초한 robust 검증 방법에도 적용된다.

O Spatial general linear modelling

Spatial general linear model들은 본래 모델링된 속성들이 단순히 범주적이거나 수 혹은 부분을 묘사하거나, 특별한 고려가 요구되는 경우에 앞에서 언급한 spatial regression model들을 확장한 것이다. 이것들은

분할표들에 대한 log-linear modelling과 포아송 분포 및 이항 분포 변수들의 모델링에 대한 개념들에 대한 공간적 일반화들로 구성된다. 이러한 모델들의 공간적 형태는 비교적 개발이 미약하고 이론적인 문제를 내포하고 있는데 이런 모델들을 다루고 있는 통계적 소프트웨어들은 공간적인 문제에 포함되는 불규칙적 가중 행렬들을 처리하지 못하고 있다. 그렇지만 이것들은 공간적 사회-경제문제의 자료들에서 일반적으로 발생하는 양적, 수, 할당을 나타내는 속성들을 처리하는 특별한 방법들의 필요성을 강조하기에 충분히 필요하다.

O Multivariate techniques

다변량 데이터의 모델링 방법은 관심지역에 대한 하나의 대응변수와 그것의 공간적 변동을 설명할수 있는 다른 변수들과의 관계를 모델링한것과 관련이 있다. 종종 몇 개의 가능한 대응 변수들이 동시에 처리되어야 할 경우가 있고 이러한 경우들로 인해 전통적인 다변량 통계 기법들에 관한 보다 넓은 확장이 고려되어야 한다. 이런 기법들의 대부분은 공간적으로 의존적인 데이터를 위해서 시작된 것은 아니지만 데이터를 압축하는 도구로서 유용하며, 공간 내용을 검사하기 위한 변수들의 결합들을 구분하기 위해서도 유용하다. cluster analysis는 지리적인 패턴을 검사할 때 데이터의 공간상에서 관측치들의 자연적인 군집들을 구분하기에 유용하다. 이런 classification 방법들에 공간적인 강력함을 내재하기 위하여 많은 방법들이 제안되어 왔다. Canonical correlation analysis(정준 상관 분석)는 공간적으로 최대한 분리되어 있는 대응 변수들의 결합을 위한 검색을 가증하게 하고, Multidimensional scaling이 지리적인 배열과 관련된 데이터 공간에서의 관측치들에 대한 기하학적인 배열을 위한 검색을 가능하게 한다.

O Spatial interaction models

공간 상호작용 연구에서의 일반적인 문제는 원지점에서의 수요와 목적지에서의 견인력 그리고 원지점과 목적지 사이의 일반화된 이동 거리 및 비용이라는 용어들에서 원지점으로부터 목적지 사이의 관측된 흐름의 패턴을 모델링하는 것이다. 종래의 사용된 모델들은 수직 지형에서 시작되었지만 최소 이동거리, 엔트로피의 극대화 같은 다양한 최적화 문제에 이론적으로 적용된 일반적인 중력 모형들이다. 그런 모델들은 원점들이나 목적지들에 대한 총 관측된 흐름을 줄이기 위하여 매우 유용할 것이다. 통계적인 관점에서 그런 모델들은 최대 우도법이나 반복 가중 최소 자승법에의해 검증된 모수들을 가진 일반 선형 모델들의 에

로서 생각될 수 있을 것이다. 이것들은 상용 GIS에서 사용되고 있는 것으로 흐름이 항상 최근린 목적지로 향한다는 가정을 내재하고 있는 단순 결정론적인 location/allocation 모델들로부터 구별되어야 한다.

- 기본적인 공간통계의 기능을 자바로 설계/구현하는 과정은 공간 데이터와 속성 데이터간의 연계를 기본 전제로 한다. 본 과제에서는 앞에서 언급된 공간 통계의 기능 중에서 우선 순위가 높은 것들을 중심으로 공간 분석기를 설계하였다.

2. 공간 통계 분석기 설계

- 공간통계 분석기는 공간패턴 분석기(Spatial Pattern Analyzer)와 공간현상의 상관 분석기 (Map Comparison and Areal Association)로 구성되며 구체적 구현대상이 되는 통계량들은 아래와 같다.

  - Map Comparison and Correlation Measures
  - Cramer's V on Area Cross-Tabulation
  - Jaccard's Similarity Coefficient
  - Coefficient of Areal Association
  - Odds Ratio
  - Contrast
  - Yule's Alpha
  - Lorenz Curve
  - Index of Dissimilarity
  - Gini Coefficient
  - Concentraion and Localization Indices
  - Segregation Index
  - Location Quotient
  - Weight of Evidence

- 설계된 공간통계량은 다음과 같다.

O Areal Association

둘 이상의 변수간의 공통적인 경향을 파악하는 것으로 가시화된 지도와 수치 지도를 통해서 연관성의 강도를 알 수 있다.

O Binary Map Comparison

두장의 이진 지도를 연관성을 파악하는 것으로서 상관도를 측정하며, 도출된 상관도에 기초하여 가중치를 부여한다. 이진 지도를 상관도를 측정하는 방법으로 Area Cross-Tabulation이 있다.

O Lorenz Curve

공간적 분포의 유사성을 나타내는 지수이다.

Index of Dissimilarity는 이상적인 곡선과 도출된 곡선간의 최대 수직 거리를 계산한 값이다. Gini Coefficient는 이상적인 곡선과 도출된 곡선간의 면적을 계산한 값이다.

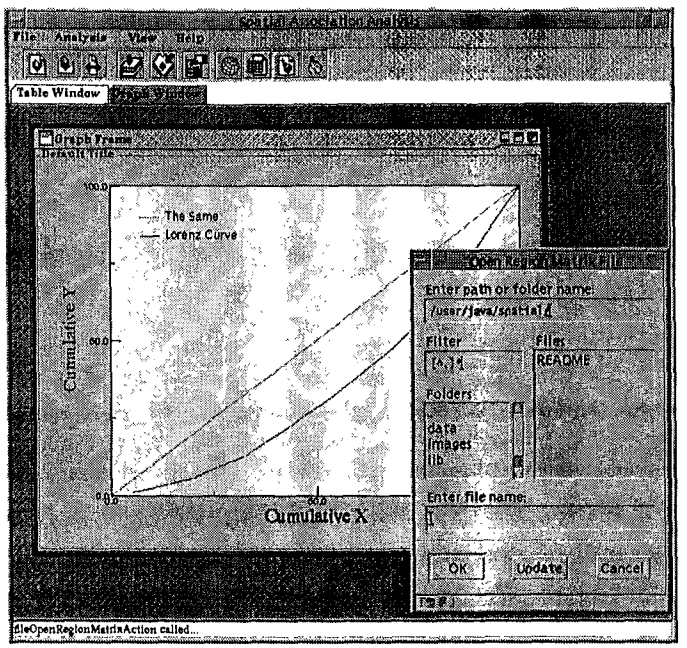
O Segregation Indices

비유사성을 나타내는 계수로서 1990년대에 들어서 공간적 상호작용 요소를 포함하게 되었다.

O Location Quotient

여러 지역들이 규범이 될만한 값(예를 들면 전국적 평균)에 비교해서 다른 정도를 측정하는 값이다. 이 값은 집중도를 나타내며, 1일 때 평균과 같고 값이 클수록 집중도가 높음을 말한다.

- [그림 102]는 이러한 항목을 설계, 구현함으로써 만들어진 공간 분석기의 사용자 인터페이스를 보여준다.



[그림 102] 공간 분석기의 사용자 인터페이스

- 공간패턴 분석기는 OpenViews 매핑 커널과의 인터페이스를 통하여 각 공간 단위들이 가지고 있는 통계량을 읽어들인 후에 앞에서 언급된 바와 같이 여러 가지 의미있고 유용한 통계량들을 새로 계산해 낸다.
- 계산된 통계량은 3차원 그래픽 또는 위의 그림에서 보이는 바와 같은 그래프로 표현될 수 있다.

제 4 장 연구개발 목표의 달성도 및 대외기여도

제 1 절 연구개발 목표의 달성도

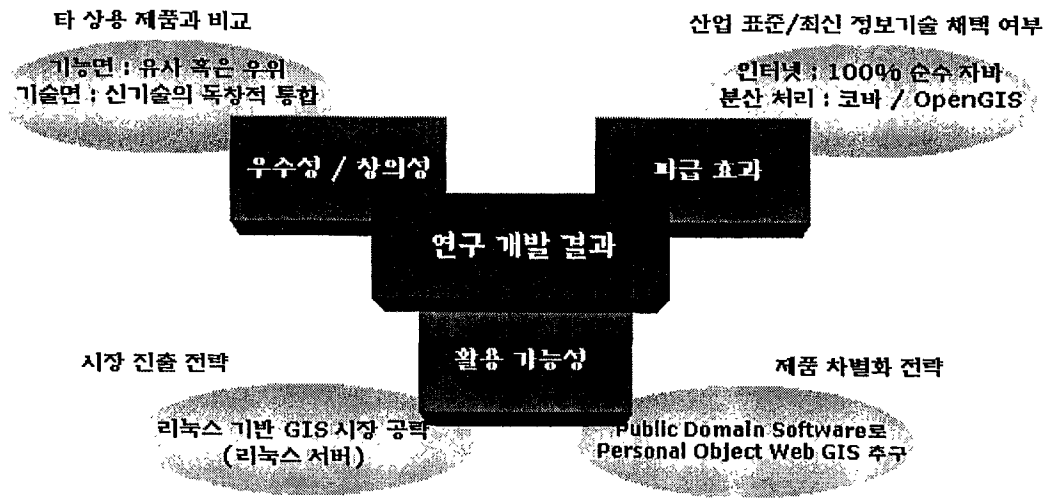
- 본 과제의 주요 연구 범위와 그에 따른 목표 달성도는 [표 6]에서 보는 바와 같다.

연구 범위	목표 달성도
OpenGIS / CORBA 기반 아키텍처 설계	100 %
설계 아키텍처에 기반한 GIS 컴포넌트 개발	100 %
웹과 연동하는 GIS ToolBox 개발	100 %
자바기반 공간통계 분석기 개발	100 %

[표 6] 연구 범위와 목표 달성도

제 2 절 연구개발 결과의 파급효과

- 본 과제의 연구 개발 결과는 전자지도 편집, 출판 분야에 있어 저렴한 비용과 효율적인 처리를 장점으로 하는 인터넷 기반 GIS 애플리케이션 등을 개발할 수 있는 핵심 기술을 확보, 보급함으로써 GIS 기술의 국산화와 관련 분야 응용에 획기적인 전기를 마련할 수 있을 것으로 예상된다.
- 특히, 본 과제에서 제시한 개방형 GIS 시스템 아키텍처는 그간 심도있는 검토 및 운용 방안에 대한 검증 없이 추진되어 온 다른 여타 개방형 시스템 개발 프로젝트의 위험성을 줄일 수 있는 기본 연구로 기여할 수 있을 것으로 판단된다.
- 본 연구 결과의 전체적인 평가 다이어그램은 [그림 103]에서 보는 바와 같다.



[그림 103] 연구 결과에 대한 전체적인 자체 평가 다이어그램

- 본 과제의 연구 결과는 기능면에서, 현재 상업용으로 배포되고 있는 유사 제품과의 비교시 동일한 기능 혹은 그보다 우수한 기능을 가진 것으로 평가되며, 기술면에서, 아직 유사 제품군에서 적절하게 채용하지 못하고 있는 최신 기술을 독창적으로 통합한 것으로 평가된다.
- 파급 효과로는 본 과제의 연구 결과가 100% 순수 자바 기반이므로 분산컴퓨팅 환경에서 생명주기를 늘릴 수 있어 인터넷 기반의 Personal GIS 제품으로서 확산 가능할 것으로 판단된다. 또한 코바/OpenGIS 기반 아키텍쳐를 가지고 있으므로, 국제 산업 표준을 준수하는 분산, 인터넷에 적용 가능한 제품으로 활용 가능할 것이다.
- 구체적인 연구개발 결과의 활용 가능성으로는, Public Domain 소프트웨어로서 Linux 시장을 적극 공략할 수 있으며, 웹 편집, 출판 기능을 강화하여 인터넷 기반의 대규모 프로젝트에 활용 가능한 제품군으로 보완 개발 가능할 것으로 판단된다.

제 5 장 연구개발결과의 활용계획

- 본 과제의 연구개발 결과는 기본적으로 코바/자바/OpenGIS 에 기반한 개방형 아키텍쳐를 가지고 있으므로 다양한 프로젝트에 활용될 수 있을 것으로 판단된다 OpenViews 는 현재 국가에서 추진중인 토지정보체계의 미들웨어 컴포넌트로 개발되고 있으며, 그 외에도 GIS 애플리케이션 서버 구축을 통한 전자 지도 편집 및 출판 분야에 적극 활용될 수 있을 것으로 판단된다.
- 본 과제의 연구개발 결과는 크게 OpenWorld 아키텍쳐와 OpenViews 매핑 커널로 이루어져 있으며, 차후 GIS 틈새 시장을 적극 공략할 수 있는 소프트웨어 패키지로 확장 개발할 예정이다.
- 특히, 리눅스 시장을 적극 공략함으로써 현재 제대로 활용할 수 있는 GIS 소프트웨어가 부족한 리눅스 기반 시스템에서 본 과제의 결과물을 활용할 수 있도록 하고, 저비용의 리눅스 기반 GIS 애플리케이션 서버를 구축함으로써 국가에서 추진중인 다양한 공공분야 지도정보 서비스에 큰 도움이 되도록 할 방침이다.

제 6 장 참고문헌

[1] Landgraf, G., 1999, *Evolution of EO/GIS Interoperability towards an Integrated Application Infrastructure*, In : Schek, H., et al, Proceedings of Second International Conference, INTEROP '99, Springer-Verlag

[2] The Open GIS Consortium : http://www.opengis.org/

[3] Orafali, R., Harkey, D., Edwards, J., 1997, *Instant CORBA*, John Wiley & Sons, Inc.

[4] Jeff Nelson, 1999, *Programming Mobile Objects with Java*, John Wiley & Sons.

[5] William R. Cockayne, Michael Zyda, 1997, *Mobile Agents*, Prentice Hall.

[6] NCGIA Research Initiatives : http://www.ncgia.ucsb.edu

[7] Gio Wiederhold, 1999, *Mediation to Deal with Heterogeneous Data Sources*, in Interoperating Geographic Information Systems, Second International Conference, INTEROP '99 Proceedings, Springer-verlag

[8] OpenStream consulting Inc., 1998, Analysis of the Application Server Market.

[9] A. Frank and S. Timpf, 1994, *Multiple Representations for Cartographic Objects in a Multi-scale Tree--An Intelligent Graphical Zoom. Computers and Graphics* 18(6): 823-829

[10] Reifer, Donald J., 1997, *Practical Software Reuse*, John Wiley & Sons Inc., New York.

[11] Szyperski, C., 1998, *Component Software*, ACM Press, New York.

[12] Grand M., 1998, *Patterns in Java*, John Wiley & Sons, New York.

[13] Goodchild, M., et al., 1999, *Interoperating Geographic Information Systems*, Kluwer Academic Publishers, London.

[14] Vckovski, A., 1998, *Interoperable and Distributed Geoprocessing in GIS*, Taylor & Francis, London.

[15] Buehler K., Mckee L., 1998, *The OpenGIS Guide : Introduction to Interoperable Geoprocessing and the OpenGIS Specification*, 3rd ed., Open GIS Consortium, Inc.

[16] Genesereth, Micheal R., 1995, *Interoperability : An Agent-Based Framework*, AI Expert, Vol. 10(3)

[17] Powersoft, 1996, *Business Application Development for the Internet*, Powersoft White Paper, http:://www.sybase.com/powersoft/products