그래프 구조 기반의 회의 정보 가시화[1)]

Graph-based Information Visualization for Meeting Information

김리라*, 양상욱**, 김영일***, 전차수****, 최영*****, 김래현******, 박세형*******,

*경상대학교 산업시스템 공학부(lily327@paran.com) **중앙대학교 기계공학부(araato@paran.com)
***경상대학교 산업시스템 공학부(zero1.kim@citek.co.kr) ****경상대학교 산업시스템공학부(csjun@gsnu.ac.kr)
*****중앙대학교 기계공학부(yychoi@cau.ac.kr) ******한국과학기술연구원(laehyunk@kist.re.kr)
*******한국과학기술연구원 (sehyung@kist.re.kr)

Abstract

정보가시화는 정보를 기하학적으로 표현하는 연구 분야로 정량적 정보를 테이블, 도표 등의 형태로 표현하는 정량 정보 가시화와 그래프나 네트워크와 같은 구조적 자료를 기하학적으로 표현하는 graph visualization이 있다.

본 연구에서는 그래프 기반의 정보 가시화를 이용한 회의 정보 가시화 프로그램을 소개하고자 한다. 이는 연구개발이나, 프로젝트 관리, 브레인스토밍 등의 회의에 있어서 태스크, 자원, 일정, 문서 등으로 구성되는 회의정보를 대상으로 한다. 최초 정보 생성, 정보 수정, 정보 가시화 기능을 갖고 있으며 그래프 자동 배치 모듈, 가시화 모듈, 사용자 인터페이스 모듈 등으로 구성되어 있다. 그래프 자동 배치는 오픈 소스로 제공되는 GraphViz를 사용하였고, 가시화는 OpenGL을 이용하였다.

회의 정보들 사이의 복잡한 관계를 그래프 구조로 표현하여 업무와 자원의 분배, 관련된 문서 검색을 쉽게 하여 회의 정보를 직관적이고, 빠르고 쉽게 조작하고, 이해하는데 유용하다.

1. 서론

최근 생활공간 내에서 실생활의 질 향상을 위한 지능적 정보 제공 서비스 및 자연스러운 동작을 통해 컴퓨터 시스템을 제어할 수 있는 내추럴 인터페이스 기술에 의한 지능형 반응 공간(Intelligent Responsive Space, IRS) 기술이 제안되고 있다. 현재 지능형 반응 공간 기술 수준은 서비스를 기반으로 개별 에이전트 기술 및 비구조적인 에이전트 기술을 위주로 개발되고 있으며, 더불어, 내추럴 인터페이스 기술 수준은 포인팅 추적에 의한 동작 감지 기술 및 그래픽 표현 중심의 디스플레이 기술 위주로 개발되고 있다.[1]

본 연구는 "지능형 실감반응 회의 공간[1]"의 일부 모듈로서 실감아이콘의 내추럴 인터페이스를 지원할 목적으로 물리기반 그래프 구조의 회의 정보를 가시화 모델을 만들고 사용자 정보조작을 위한 인터페이스 기술과 디스플레이 기술을 구현하였다.

1) 본 논문은 2005년 한국과학기술연구원의 지원으로 이루어졌습니다.

정보 가시화(information visualization)는 주어진 데이터가 사용자에게 보다 유용한 정보가 될 수 있도록 "정보의 기하학적 표현"을 연구하는 분야로 많은 연구 그룹, 관련 학회, 저널이 있다[2, 3]. 정보 가시화는 정보의 다양성에 따라 정량적 정보를 테이블, 도표 등의 형태로 표현하는 정량 정보 가시화와 그래프나 네트워크와 같은 구조적 자료를 기하학적으로 표현하는 graph visualization으로 나누어진다.

본 연구는 graph visualization에 속하며 회의정보에 적합한 정보 모델링과 정보 가시화, 정보 조작을 구현하는데 초점을 맞추고자 한다.

그래프 구조 기반의 정보 모델링 기술을 개발 및 구현함으로써, 연구개발이나 프로젝트 관리, 브레인스토밍 등의 회의를 진행함에 있어서 추상적 정보를 자연스럽게 가시화하여 참여자가 실감있게 공유하고, 사용자의 생각과 의견을 직관적으로 반영하는 것을 목적으로 하고 있다.

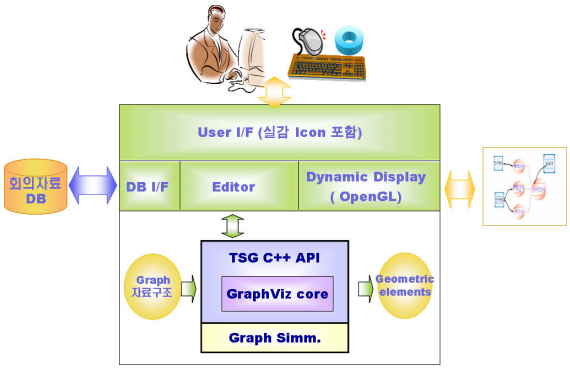


그림 1. 정보가시화 프로그램 구조

2. 기존 연구

정보 가시화는 추상적 데이터로부터 사람의 인식을 돕기 위해서 컴퓨터를 이용한 사용자 인터페이스를 제공하여 인터랙티브한 가시화를 진행하는 과정이라 할 수 있다. [4]

정보 가시화 연구에 있어서 중요한 포인트는, 데이터의 분석 및 이해를 통하여 정보를 표현하는 새

로운 가시적 표현을 만들어 내는 것이다. Stephen의 연구[2]에서는 이러한 가시적 표현을 정보에 대한 함축 또는 은유(metaphor)라는 단어로 설명한다.

지금까지 연구되어온 다양한 가시화 기법들의 분류를 그림 2를 참고로 살펴보면, 가시화 되어야 하는 데이터의 종류, 가시화 기술 자체, 인터랙션 및 변형 방법의 세 가지 기준에 의해서 분류 될 수 있다.[3]

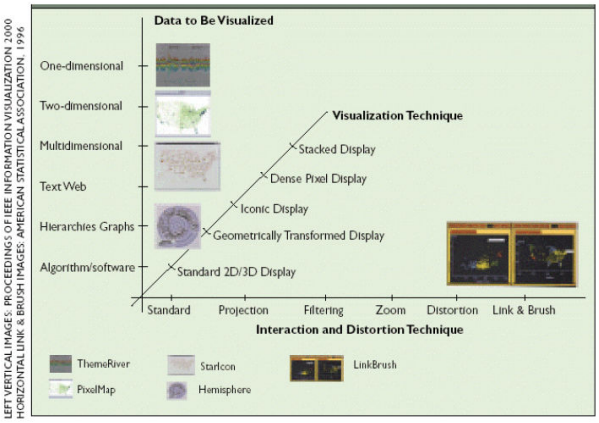


그림 2. 가시화 기법의 분류

정량 정보 가시화 분야에는 SmartDiscovery[5], InfoZoom[6], Spotfire[7] 등의 상용화된 시스템이 있으며 이들은 나름대로 여러 가지 특징을 갖고 있다[8, 9]. 구조적 자료(structural information)를 기하학적으로 표현하는 graph visualization 분야는 주로 소프트웨어공학, 데이터베이스, 웹 디자인, 네트워킹 등에서의 자동 그래프 드로잉에 적용되고 있으며 이에 대한 연구 또한 활발하다[10, 11].

Graph Drawing은 정보 간 관계를 가지고 있는 토폴로지 데이터 *G*로부터 2D 혹은 3D 상의 형상 데이터를 만들어 내는 것이라 할 수 있다[11]. Graph drawing은 기원전 4세기 아리스토텔레스의 noli turbare circulos meos!(Don't disturb my circles!) 문제로부터 18세기 오일러의 쾨니히스베르크(칼리닌그라드)시의 프레겔 강(江)의 7개의 다리를 건너는 문제(Königsberg bridge problem)등이 그 기원이라고 할 수 있으며, 최근의 Reingold-Tilford for trees 알고리즘[12] 까지 많은 연구가 이루어져 있지만 현재도 해결해야할 문제들이 남아 있다[13].

상용 diagramming tool인 MS Visio는 복잡한 개념, 프로세스, 시스템을 문서화하고 구성하는 비즈니스 및 기술 다이어그램을 만들 수 있다[14]. 마인드 맵핑 (mind mapping) 소프트웨어인 SimTech System사의 ThinkWise[15]는 각종 회의에서 산발적으로 나오는 의견과 정보를 쉽게 저장하고 분류하는 기능이 있으며, 각 항목별 일정 관리 및 타 문서로의 변환이 가능하다. 일정 관리에 초점을 맞춘 가시화 시스템인 MS Project는 리소스의 최적화, 작업 우선순위 결정, 전체 비즈니스 목표에 따른 포트폴리오 형태로 프로젝트를 배열하는 기능 등을 제공한다[16]. 한편 KAIST 최병규 교수팀은 BOP (bill of process) 모델을 이용한 체계적 공정관리 기능, 로딩 시뮬레이션을 통한 신속한 부하연동 일정계획수립, 부하 상황과 납기를 고려한 외주계획 수립, 간트 차트를 이용한 일정조회/편집 등의 기능을 가진 LSE (loading schedule editor)를 개발하여 보급하고 있다[17]. 또한 건설현장에서의 공정관리를 가시화하기 위한 건설정보 가시화 시스템에 관한 연구도 있다[18].

본 연구에서는 그래프의 자동 배치를 위해서는 공개된 유틸리티인 GraphViz를 이용하였다. GraphViz는 그래프 가시화 소프트웨어며, 텍스트로 된 스크립트로부터 여러 가지 레이아웃의 그래프 이미지나 웹 페이지를 위한 SVG 혹은 Postscript, PDF 등의 도큐먼트 요소로 출력하는 기능을 가지고 있다. [19]

3. graph의 자동 배치

본 연구에서는 그래프 가시화에 대해, 초기 배치를 위하여 GraphViz의 소스 코드를 Win32 환경에 포팅하여 API 레벨에서 활용하고, 물리 기반 가시화 및 사용자 인터페이스로부터 그래프 요소들을 생성, 삽입, 수정, 삭제, 검색 등의 오퍼레이션을 할 수 있도록 C++ 클래스 라이브러리를 구현하여 TSGLib 라고 명명 하였다.

3.1 GraphViz layout- dot & neato

GraphViz는 노드와 에지로 표현되는 데이터 구조로부터 시각적 그래프를 배치하고, 그 결과를 이미지 파일이나 postscript 등으로 출력해 주는 도구이다. 그래프 배치를 위한 레이아웃 방법 중 가장 대표적인 것이 dot 와 neato 레이아웃이다.[20, 21]

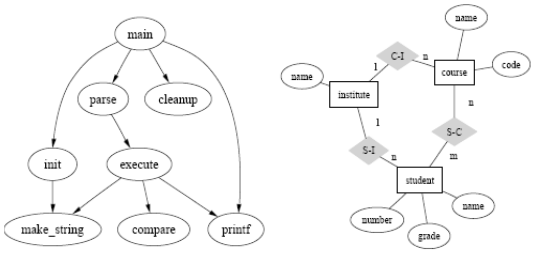


그림 3. GraphVis의 레이아웃 예

dot 레이아웃은 계층을 가진 방향성 그래프를 그리는데 적합하다. 그림 3의 좌측은 dot 레이아웃의 결과로 출력된 이미지이다. 데이터의 내용상으로 main으로부터 수직적으로 내려가는 구조를 가지고 있음을 볼 수 있다.

그림 3 우측의 neato 레이아웃은 계층이나 방향성을 가진 관계보다는 동등한 여러 요소들 간의 네트워크 구조를 표현하는 데 더 적합하다.

3.2 GraphViz API 분석 및 포팅

GraphViz 어플리케이션은 스크립트들을 파일로 받아들여서 결과를 이미지로 출력을 하지만, 본 연구에서 그래프 레이아웃 모듈이 다른 프로그램 요소들과 함께 실시간으로, 객체 단위의 입/출력을 하는 것이 필요하다. 따라서 GraphViz의 소스 프로그램과 그 구성요소를 분석하여 API레벨에서 활용 가능하도록 Win32 환경으로 포팅하는 작업이 필요하였다. 다음 그림 4는 GraphViz의 소스 모듈들의 구성을 나타내고 있다.

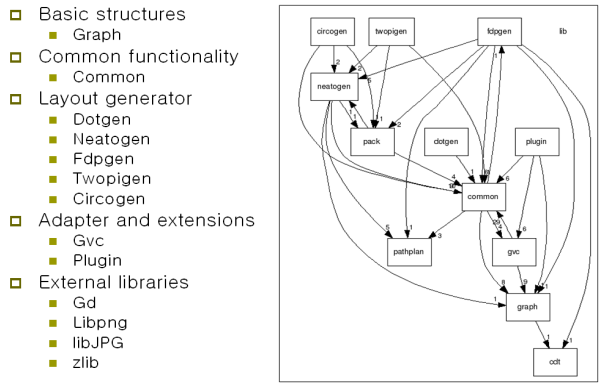


그림 4. GraphViz의 모듈들

이들 모듈들을 Win32 환경으로 포팅 하여 MFC (Microsoft Foundation Classes)들과 함께 응용 프로그램을 만들 수 있게 하기 위해서 VisualC++ 7.0 환경에서 컴파일하였다.

GraphViz로 하여금 node와 edge를 생성하고 연결시키도록 한 후, 레이아웃을 수행하도록 하였다. 그 결과 레이아웃 된 각 노드의 위치와, 베지어 커브로 노드들을 연결하는 에지를 얻을 수 있었다.

3.3 TSGLib C++ class library

GraphViz를 Win32 환경에서 사용할 수 있도록 포팅한 결과, 이미지의 출력과 관련된 외부 라이브러리를 제거하고도 16개의 라이브러리 모듈로 남았다. 모듈 간 인터페이스의 복잡성을 없애고, 기능의 확장을 보다 용이하게 하기 위해서 C++ 래퍼(wrapper)를 도입하였다.

GraphViz 모듈을 둘러싸도록 C++의 class library API를 구현 하고 이 클래스 라이브러리를 TSGLib라 이름 지어, 이를 중심으로 본 연구에 사용되는 모듈 간 통합이 이루어 질 수 있도록 하였다.

TSGLib는 GraphViz 코어 라이브러리를 정적 링크를 이용하여 포함하며, DLL로 작성된다. TSGLib는 그래프 자체와, 노드, 에지에 대한 추상화를 제공한다. 사용자는 TSGGraph 객체를 생성하여, 노드 및 에지의 생성, 삭제, 검색을 위한 그래프를 구성하고, 레이아웃을 수행하도록 명령을 줄 수 있다. 이후에 레이아웃 된 형상 정보를 얻어 올 수 있다.

4. 물리기반 정보가시화 모델

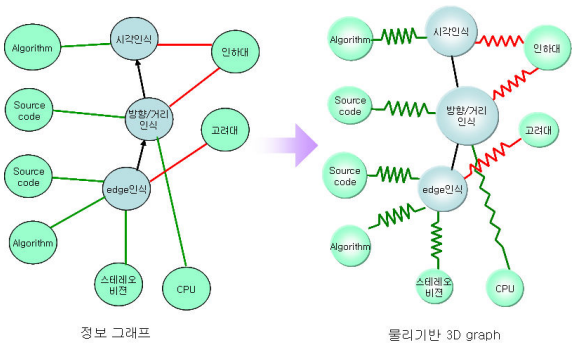


그림 5. 물리 기반 그래프 개념

GraphViz를 이용한 자동배치는 무난하지만 사용자를 최상으로 만족시키는 데는 한계가 있다. 자동으로 배치된 그래프를 사용자가 직접 수정하고 싶은 경우, 관련성이 높은 노드들을 정해진 규칙에 따라 함께 움직이면 실감성이 더욱 높아질 수 있다. 이를 위해 본 연구에서는 물리기반의 정보가시화모델을 제안한다.

그림 5의 좌측에는 일반적인 정보 그래프의 예가 나타나있고, 우측에는 에지를 스프링으로 간주한 물리 기반 그래프가 나타나 있다. 사용자가 특정 노드를 드래깅 할 때, 관계된 노드들이 움직이는 모습을 구현하기 위해 스프링-넷을 이용하였다.

4.1 물리 기반 가시화를 위한 전 처리 과정

물리 기반 가시화를 위해서 스프링-넷을 구성하여 노드를 움직일 때, 경계조건에 의해서 잡아주는 노드가 없다면 전체의 그래프가 노드와 함께 움직이는 강체 운동(rigid body translation)을 하게 될 것이다. 또한, 연결 관계가 많지 않은 노드의 경우, 스프링 넷의 방정식을 풀기에 충분한 조건이 만들어질 수 없는 경우가 생길 수 있다. 따라서 데이터 간의 관계에 의한 에지에 의해서만 스프링을 구성하는 것 외의 더 많은 스프링이 필요할 수도 있고, 적절한 경계 조건을 부여함으로써, 움직이고자 하는 노드는 움직이게 하지만 전체 그래프의 모양을 유지할 수 있도록 하는 것이 필요하다.

그래프 내의 정보 관계에 의한 스프링 외 부가적인 스프링의 삽입은 원래의 그래프의 형상에 따르도록 하였다. 움직이지 않게 고정하는 경계조건은 그래프의 배치 정보를 살펴서 최 외곽에 있는 노드들에게 부여하도록 하면 이러한 문제를 해결 할 수 있다.

본 연구에서는, 레이아웃이 배치된 형상으로부터 노드간의 인접 정보를 찾고 최 외곽 노드를 판별하기 위해서 배치된 노드에 대한 보로노이 다이어그램(Voronoi diagram)을 생성하였다.

그림 6는 보로노이 다이어그램의 인접 정보를 참조하여 부가적인 스프링을 삽입한 예를 보여주고 있다. 부가적인 스프링은 원래의 정보관계에 의한 스프링보다 낮은 스프링 상수를 가지게 하고, 이 비율은 조절 될 수 있다.

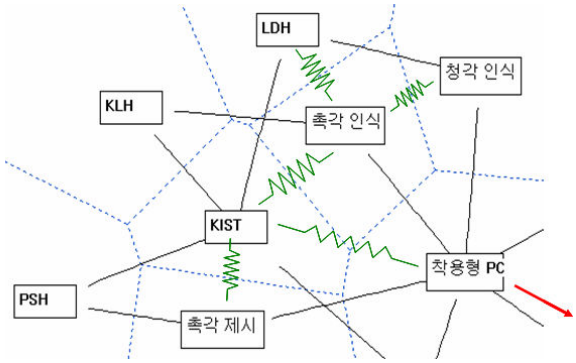


그림 6. 인접 정보에 의한 부가 스프링 삽입

그림 7은 보로노이 다이어그램으로부터, 최 외곽 노드를 판별하는 방법을 보여주고 있다. 보로노이 다이어그램 외곽의 볼록 다각형(convex hull) 특성으로부터 닫혀 있지 않은 보로노이 셀은 전체 점집합의 최 외곽에 있다는 사실을 유도할 수 있다.

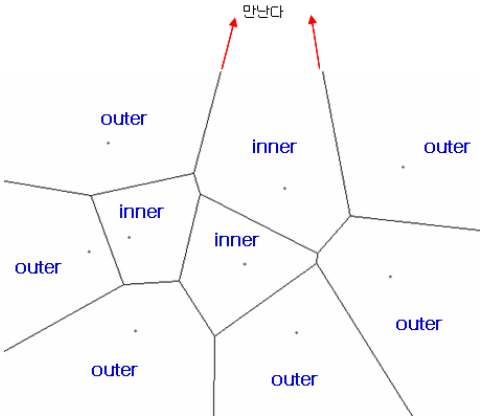


그림 7. 최 외곽 노드 판별

4.2 Voronoi Diagram 구현

물리 기반 가시화를 위한 보로노이 다이어그램 계산은 노드의 삽입 및 변경 시점마다 일어나야 하므로 빠르게 계산되어야한다. 사용자가 특정 노드를 드래깅하고 있다면 드래깅에 의해 노드가 움직이는 매 순간마다 계산되어야 한다. 또한 애니메이션을 위해서라면 초당 30프레임을 유지할 수 있도록 충분히 빨라야 한다. 따라서 빠르고 가볍게 동작하는 보로노이 다이어그램 모듈이 필요하다. 이미 구현되어 공개된 여러 라이브러리를 사용할 경우, 보로노이 다이어그램을 위한 모듈 뿐 아니라 그 기반이 되는 모듈들을 함께 이용하는 경우가 많아 시스템 자체가 무겁게 될 우려가 있어 직접 구현하였다.

보로노이 다이어그램은 식 3.1과 같이 정의된다.

$$P=\{p_1,p_2,\cdots,p_n\}\ \text{set of 2-dim Euclidian plane}$$

$$p_n:\text{site}$$

$$\text{Vornoi Region } V(p_i);\ p_i \text{ site에가장 가까운 point 들의집합}$$

$$V(p_i)=\{x,|p_i-x|\le|p_j-x|\ \ \forall j\neq i\} \qquad \text{(식 3.1)}$$

점들에서 가장 가까운 영역을 해당 점의 보로노이 영역(혹은 보로노이 셀)이라고 하는데, 이러한 영역을 구하고 데이터 구조화 하는 것이 보로노이 다이어그램을 구현하는 것이다. 알려진 알고리즘으로는 다음과 같은 것들이 있다. [20]

- Intersection of Halfplane : $O(n^2\log n)$
- Incremental Construction : $O(n^2)$
- Divide and Conquer : $O(nlogn)$
- Fourtune's Algoritm : worst-case $O(nlogn)$

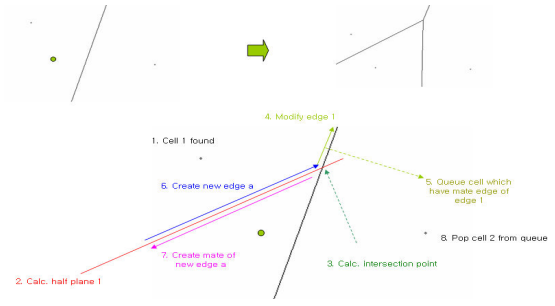
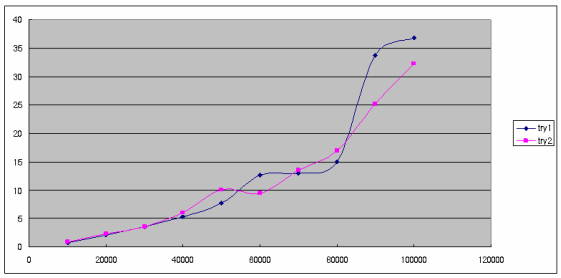


그림 8. $O(nlogn)$ Incremental construction 알고리즘

본 연구에서 구현한 알고리즘은 incremental construction 이면서 $O(nlogn)$의 시간복잡도(time complexity)를 갖는다. 그림 8의 상단 좌측과 같이 두 개의 기존 사이트에 의해서 셀 분할이 이루어져 있는 상태에서, 하나의 점이 삽입되어서 우측 상단과 같은 결과가 나오는 과정의 예를 들어서 본 연구에서 구현한 알고리즘을 설명하면 다음과 같다.

1) 새로운 점이 추가될 기존의 보로노이 셀을 찾는다. 기존의 보로노이 셀 중 한군데에 추가되기 때문에, 기존 보로노이 셀 중에서 새 점을 포함하는 셀을 찾는 것이다. 기존 사이트는 k-d 트리 안에 있기 때문에, 새 점과 가장 가까운 기존 사이트를 찾으면, 그 사이트를 중심으로 하는 셀이 찾고자 하는 셀이 되며, 찾는 과정에서 $O(\log n)$의 시간 복잡도가 소요된다. 알고리즘의 이후 단계에서는 상수의 시간 복잡도가 소요된다.
2) 새 점과 기존 사이트에 의한 하프 플레인을 계산한다.
3) 계산된 하프 플레인이 기존 보로노이 셀과 만나는 점들을 계산한다. 이 과정에서는 기존 보로노이 셀의 하프 에지 수만큼 시간 복잡도가 소요되는데, 보로노이 셀의 최대 하프 에지의 수는 6이기 때문에[20] 상수의 시간 복잡도가 소요된다.
4) 계산된 하프 플레인과 교차점에 의해서 기존 보로노이 셀의 하프 에지들을 수정한다.
5) 이때 수정에 영향을 받는 하프 에지가 있다면, 하프 에지가 마주보는 셀이 다음에 수정될 수 있도록 큐에 넣는다.
6) 3)의 과정에서 계산된 교점에 의해 생성된 하프에지를 기존 셀에 넣는다.
7) 전 단계에서 생성된 하프에지의 메이트 에지를 생성해 둔다. 다음 인접 셀에 대한 변형 과정에서 사용하기 위해서이다.
8) 큐에 들어 있는 셀들에 대해서 4) 부터의 과정을 반복한다.

위의 알고리즘에 따르면, n개의 점이 있을 때, 하나의 새 점이 삽입되는 시간 복잡도는 $O(\log n)$이다. 따라서 n개의 점이 삽입되는 전체 단계에서는 $O(nlogn)$이 된다.



npts	10000	20000	30000	40000	50000	60000	70000	80000	90000	100000
try1	0.797	2.032	3.53	5.328	7.672	12.688	13.031	14.953	33.75	36.75
try2	0.844	2.343	3.547	5.985	10.063	9.5	13.531	16.969	25.156	32.266

단위 : 초(seconds)

그림 9. 보로노이 다이어그램 알고리즘 성능 테스트 결과

구현된 보로노이 알고리즘의 성능을 테스트하기 위해서 랜덤하게 포인트들을 생성하면서 시간을 측정 해 보았다. 측정은 Pentium4 3.4GHz의 PC에서 이루어졌다. 100 포인트에 대한 다이어그램 생성

시간은 10번 수행 했을 때 모두 0.000000 초로 10^{-6} 초미만의 시간이 걸리는 것을 확인하였고, 1,000 포인트는 10번 수행 평균 0.047 초가 걸림을 확인하였다. 보다 많은 포인트에 대한 수행 시간은 그림 9에 나타나있다.

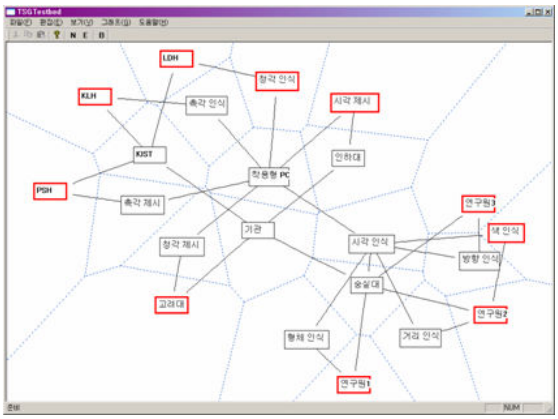


그림 10. Voronoi diagram을 이용한 가시화 그래프의 인접정보 검색

그림 10에는 구현된 보로노이 다이어그램을 가시화 그래프에 적용하여 노드의 형상간 인접 정보와 최 외곽 노드를 구한 화면이다. 최 외곽 노드는 붉은색으로 굵게 표시되어 있다.

4.3 Spring-net에 의한 물리 모델 시뮬레이션

물리 모델 시뮬레이션은 노드의 위치 변경이 있을 때, 연관 관계에 의한 다른 노드들의 변경을 시각화하여 보여주기 위해서 적용한다. 본 연구에서는 정보 가시화 그래프에 있어서 정보와 그 관계를 노드와 에지로 표현하고 이를 스프링-넷으로 모델링 하여 물리 시뮬레이션을 구현하였다.

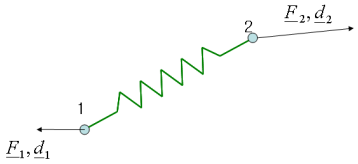


그림 11. 스프링 요소

그림 9에 보이는 스프링-넷 상의 하나의 스프링(에지)의 예로부터, 힘과 변형의 관계를 식으로 나타내면 식 3.2와 같다. 물리 시뮬레이션은 이 식을 풀어냄으로써 이루어진다.

$$\begin{bmatrix} \underline{K}_{11} & \underline{K}_{12} \\ \underline{K}_{21} & \underline{K}_{22} \end{bmatrix} \begin{Bmatrix} \underline{d}_1 \\ \underline{d}_2 \end{Bmatrix} = \begin{Bmatrix} \underline{F}_1 \\ \underline{F}_2 \end{Bmatrix} \quad \text{(식 3.2)}$$

하나의 노드가 움직이게 되면, 각 요소에 대하여 3.2의 식을 적용하여 요소 강성행렬을 만들고, 각 요소 강성행렬을 조합하여 전체 시스템 강성 행렬을 구성한다.

n개의 노드가 dim 차원의 공간에 있다면 전체 강성 행렬은 $(n\times\dim)rows\times(n\times\dim)columns$의 요소를 가지는 대칭행렬이 된다. 경계 조건에 의해서 해당 변위 벡터에 0의 값이 들어가게 되고, 그 외의 변위 벡터 항목들은 미지수로 남는다. 또한 현재 움직이는 노드의 힘 벡터 항목에 움직인 노드의 변위로 힘이 계산되어 대입되고, 나머지 힘은 미지수로 남게 된다.

전체 강성행렬과, 경계 조건 및 외력 조건이 주어질 때 나머지 미지수를 구하는데, 이 때 경계 조건이 불충분하여 전체 그래프의 강체 운동이 일어나게 되면 미지수를 구할 수 없다. 이를 방지하기 위해서 4.1에서 설명한 전 처리 과정을 통해 충분한 경계 조건 및 연결 조건을 부여하도록 하였다.

구현된 결과로, 다음의 그림 12과 그림 13을 살펴보면, 그림 11과 같은 가시화 그래프에서 사용자가 006 노드를 좌측 상단으로 움직였을 때, 그림 11에서는 006 뿐만 아니라 인접한 다른 노드들의 위치도 살짝 움직였음을 확인할 수 있다. 그림 상으로 시작과 끝만 보이고 있지만 실제의 실행에서는 사용자가 006 노드를 드래깅 하는 동안 전체의 그래프의 모양이 실시간으로 변하고 있음을 확인 할 수 있었다.

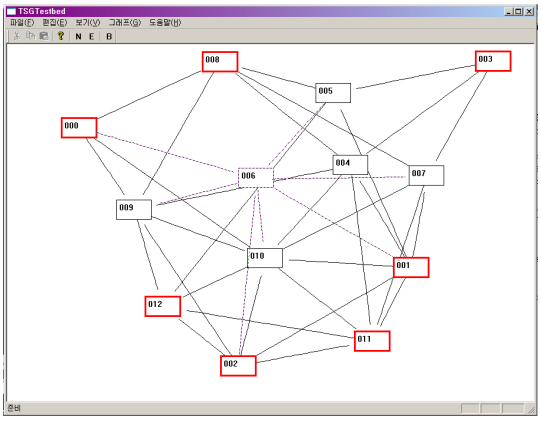


그림 12. 물리 기반 시뮬레이션 직전

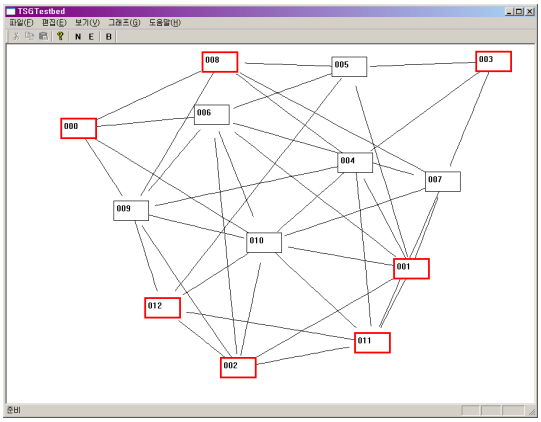


그림 13. 물리 기반 시뮬레이션 직후

5. 정보 가시화 program

정보 가시화 프로그램은 그림 1과 같이 기존의 회의 자료로부터 회의 내용을 입력받고 사용자에 의해 정보를 생성 할 수도 있다. 본 연구에서는 태스크, 팀, 자원, 문서, 일정으로 5가지를 가시화 하였다. 그 중 태스크

와 팀 2가지만 그래프의 노드로 표현하여 레이아웃 하였다. 태스크와 팀을 각각 다른 타입의 노드로 처리해 multi- layer기능을 구현했다. TSGlib를 이용하여 그래프를 생성하고 자동 배열을 한다. OpenGL을 이용하여 가시화 하며, 사용자의 조작에 의해 기본적인 viewer 기능과 필요한 노드만 보여주는 필터링 기능을 구현하였다.

5.1 그래프 생성/배열 모듈

그래프 생성/배열 모듈은 TSGlib를 이용한다. 그래프를 입력하는 방법은 두 가지가 있다. 첫 번째 방법은 이미 작성된 회의 자료를 읽는 것이다. GraphViz lib에서 읽을 수 있는 dot 파일을 읽어 그래프를 생성한다. dot 파일을 읽어올 때는 태스크의 일정을 기술한 스케줄 파일과 자원/문서의 list를 기술한 list 파일을 읽어온다. 두 번째 방법은 사용자가 최초 정보를 생성하는 경우이며, 원하는 위치에 노드와 에지를 생성하여 그래프 구조를 직접 입력하는 방법이다.

그래프 구조를 입력 받아 TSGlib를 이용하여 그래프의 레이아웃을 자동으로 결정한다. 레이아웃 방법은 4.1에서 설명한 dot과 neato가 있다. dot파일로 입력받은 경우 레이아웃은 neato 레이아웃으로 초기화 되어 있고, 직접 입력한 경우는 노드를 생성할 때의 초기위치를 저장한다. 한번 레이아웃 되었든 아니던 상관없이 다시 레이아웃을 하게 되면 그림 14과 같이 dot와 neato 로 레이아웃 사이의 전환이 자유롭게 이루어진다. 그래프의 배치 결과는 내부 데이터 구조인 node 배열과 edge 배열에 저장한다.

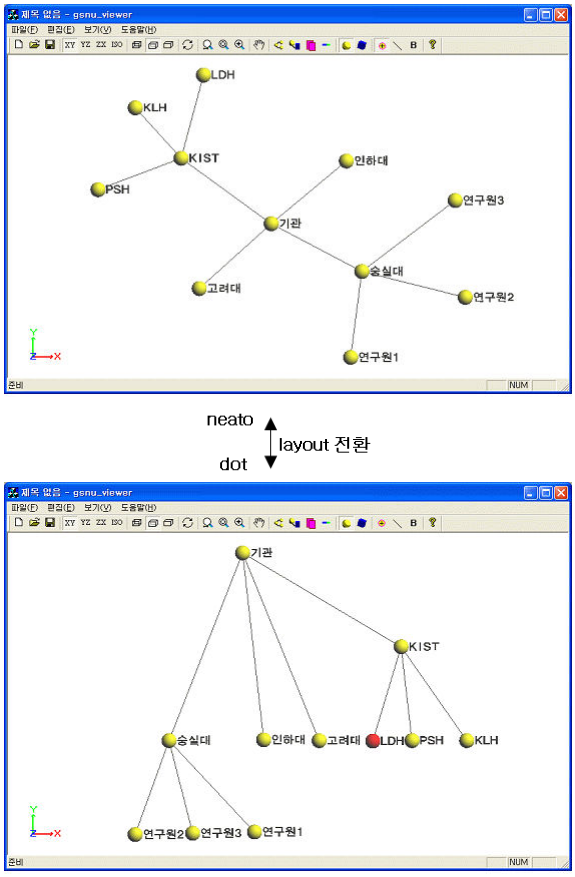


그림 14 그래프 레이아웃의 전환의 예

본 연구에서는 태스크와 팀을 노드 타입으로 구분하기 위해서 GraphViz의 노드 모양을 달리 하였다. 그래프 구조 이외의 자료인 일정, 문서, 자원은 그래프 구조에 연결되어 내부 데이터 구조 에 저장된다.

5.2 가시화 모듈

그래프 생성/배열 모듈에서 저장한 데이터를 바탕으로 그래프의 내용을 가시화 한다. 입체적으로 표현하기 위해 OpenGL을 이용하여 렌더링 하였고, OpenGL에서 렌더링 하기 힘든 한글 처리를 위해 Free Type, FTGL library 이용하였다.

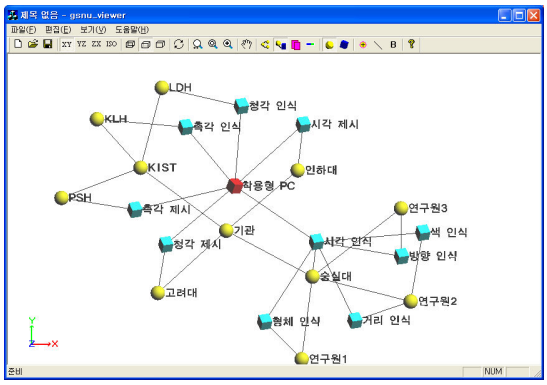


그림 15 전체 그래프 구조 가시화 예

저장된 그래프에는 그림 15와 같이 팀 노드와 태스크 노드가 섞여서 레이아웃 되어 있으므로 모든 노드를 한꺼번에 모두 디스플레이하면 복잡하여 사용자가 원하는 정보에 집중할 수 없다. 정보 가시화에서는 사용자에 의해 주목되는 부분의 노드와 에지만을 구분하여 표현하는 것이 중요하다. 사용자 조작에 의해 여러 가지 방법으로 디스플레이 할 노드를 필터링 하도록 프로그램을 구현하였다.

multi-layer 기능, 전체/ 선택 전환 기능, 다른 타입 노드 표시 기능, 자원/ 문서 리스트 표시 기능, 일정 표시 기능이 있다.

그림 16은 multi-layer 기능의 예이다. 기본적으로 둘 중 하나의 타입을 선택하여 볼 수 있게 하였다.

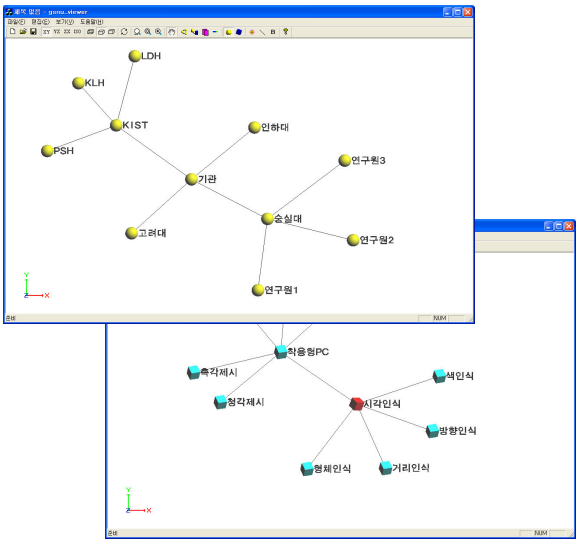


그림 16 multi-layer 기능

layer로 필터링 된 그래프라도 그래프구조가 복잡하면 전체 구조를 보기보다 선택된 노드를 중심으로 그 상위 노드와 그 하부 구조만 보는 것도 의미있는 일이다. 전체/선택 전환 기능으로 그림 17과 같이 선택된 노드의 상위노드와 일정 level 만큼만 하위 노드만 디스플레이 한다.

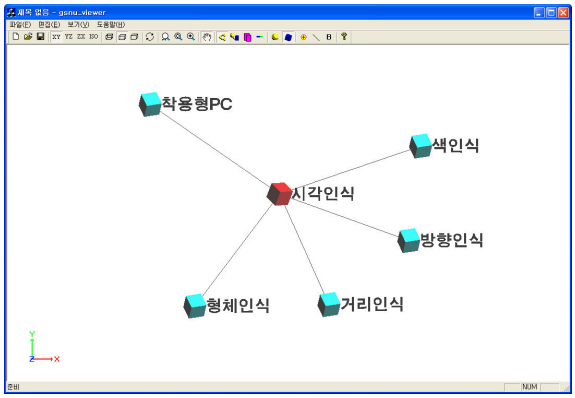


그림 17 선택된 노드 중심으로 하위 상위 노드와 하위 노드 한 level display

사용자가 일반적으로는 하나의 노드 타입에 주목하지만 태스크를 팀에 할당하는 문제 같은 경우 두 노드 타입 간 관계에 관심을 가지게 된다. 그림 18과 같이 태스크 노드 중심으로 가시화 하지만 해당 태스크를 담당하는 팀을 보고자하는 경우나 그 반대의 경우 다른 노드 타입 표시 기능을 사용한다.

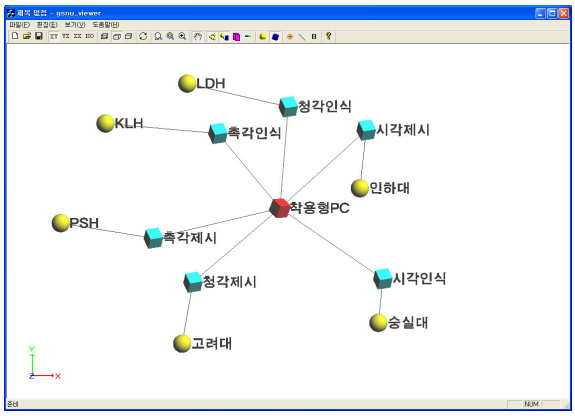


그림 18 다른 타입 노드 표시 기능의 예

그래프 외에 일정과 문서/자원 리스트가 있다. 하나의 노드로 처리할 경우 자원과 문서도 자동 배치가 되어 원하는 위치에 표현하기 힘들 것으로 예상하여 연결된 노드 아래에 위치시켰다. 한 노드에 여러 문서나 자원들이 있을 것을 예상해 리스트로 나열했다. 그림 19과 같이 태스크 노드에는 태스크에서 사용하는 자원들이 표시되고 팀 노드에는 개인이 작성한 문서들이 표시 된다.

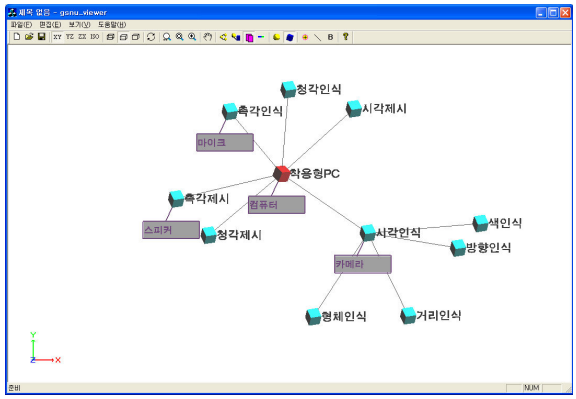


그림 19 태스크 노드 자원 표시/팀 노드 문서 표시

일정도 노드로 처리 하지 않고 연결된 태스크 노드의 위에 위치시켰다. 현재 날짜를 기준으로 진행수준을 표시하기 위해 지나간 일정과 남은 일정을 색으로 구분하여 표시 하였다. 전체적인 일정을 따로 표시하기 위해 그림 20과 같이 창을 추가로 만들어 표시할 수도 있다. 이는 간트 차트를 이용했다.

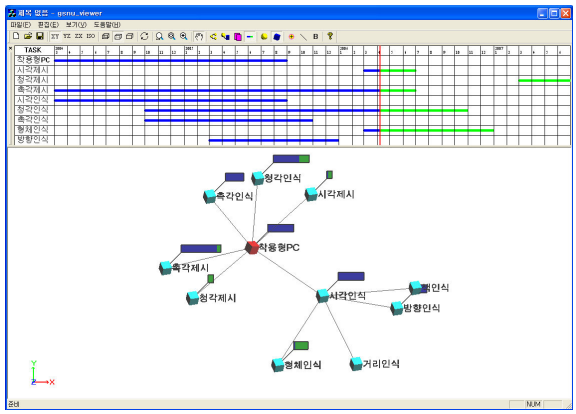


그림 20 일정표현을 한 그래프

5.3 사용자 인터페이스 모듈

다양한 display를 제공하기 위한 사용자 인터페이스가 필요하다. 사용자 인터페이스 모듈은 툴바와 메뉴를 제공하고, 마우스 선택과 드래깅으로 노드 선택, 이동 등 사용자의 입력을 받는다. 5.1의 그래프 생성, 수정, 삭제, 배열기능의 툴바의 버튼과 다이얼로그 박스가 있고, 5.2의 다양한 display를 위한 툴바의 버튼들이 있다. 그 외 기능으로는 기본적인 viewer의 기능이 있다. viewer의 기능으로 rotation, zoom-in/out, zoom-window, zoom-fit, panning 이 있다.

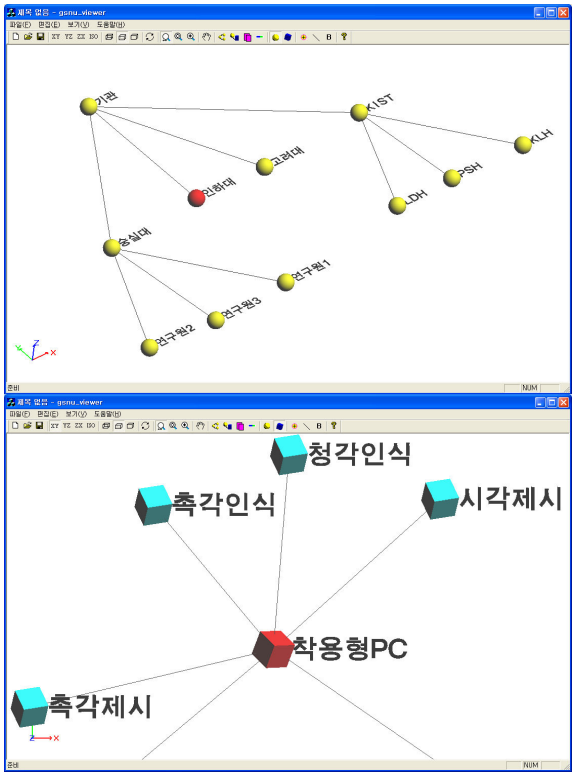


그림 21 viewer 기능을 적용한 예

6. 결론

본 연구는 그래프 구조 기반의 회의 정보 가시화 기술을 개발하였다. 연구개발이나 프로젝트 관리, 브레인스토밍 등의 회의를 진행함에 있어서 추상적인 정보를 가시화함으로써 참여자가 쉽게 파악하고, 사용자의 생각과 의견을 직관적으로 반영하는 것을 목표로 하고 있다.

태스크, 팀, 리소스, 일정 등 회의를 구성하는 요소들의 관계를 효율적으로 표현하기 위한 방법을 찾고자 정보 가시화 동향과 그래프 구조에 대해 문헌조사를 진행하고 이러한 과정을 통해서 본 연구에 적합하다고 판단되는 GraphViz와 같은 공개 유틸리티를 찾았다. 이에 대한 분석과 확장을 통하여 그래프의 생성 및 자동 배치, 수정 등의 기본적인 기능을 구현하였다. 또한 자동배치된 결과를 사용자가 직접 수정하는 경우 관련성이 높은 인접한 노드들이 일정한 규칙으로 함께 움직일수 있도록 하기위해 스프링-넷을 이용한 물리적 가시화 모델를 제안하였다.

본 연구를 통하여 개발된 가시화 응용 프로그램은 차후 회의 정보 뿐 아니라 그래프 구조로 되어 있는 정보들에 적용되어 복잡하게 연결되어 있는 정보들을 조작하고, 파악하는데 이용될 수 있다.

그래프 기반의 정보 모델링 기술 자체의 측면에서 본다면, 개발된 단층의 그래프에서 더 확장되어서 다층 (multi-layer)의 그래프 구조를 지원 한다든지, 현재 스프링-넷으로 구성된 물리 모델을 발전시켜서 질량-스프링-댐퍼 구조의 현실적이고 동적인 물리 모델을 구현하거나 햅틱 장치 등을 도입함으로써 사람이 느끼기에 더욱 자연스러운 요소 간 관계의 애니메이션이 가능하도록 할 것이다.

참고문헌

[1] 염기원, 이중호, 이승수, 엄주일, 박준구, 김래현, 조현철, 김건희, 권미수, 유호연, 손영태, 표정국, 김태수, 박면웅, 박세형, 하성도, 박지형, “지능형 반응 공간 기술 개발을 위한 시스템 아키텍처”, HCI학회, 2006

[2] Stephen G. Eick, "Visualizing Online Activity", Communications of the ACM, Aug. 2001, Vol. 44, No. 8, pp. 45-50

[3] Daniel A. Keim, "Visual Exploration of Large Data Sets", Communications of the ACM, Aug. 2001, Vol. 44, No. 8, pp. 38-44

[4] Xerox PARC User Interface Research Group, http://www.parc.xerox.com/istl/projects/uir/default.html

[5] Inxight Software, http://www.inxight.com, 2005.

[6] InfoZoom, http://www.infozoom.com, 2005.

[7] Spotfire, http://www.spotfire.com, 2005.

[8] R. Amar and J. Stasko, "Knowledge Precepts for Design and Evaluation of Information Visualizations", IEEE Transactions on Visualization and Computer Graphics, Vol. 11, No. 4, pp.432-442, 2005

[9] A. Kobas, "An Empirical Comparison of Three Commercial Information Visualization Systems", Proc, InfoVis 2001, pp.123-130, 2001

[10] Ulrika Wiss and David Carr,"A Congnitive Classification Framework for 3-Dimensional Information Visualization", Technical Report, 1998, Luleå Univ. of Tech.

[11] Course by Prof. Dr. Franz Brandenburg (U of Passau), http://www.csse.monash.edu.au/~gfarr/research/GraphDrawing02-Mel.ppt

[12] Edward M. Reingold and John S. Tilford, "Tidier drawings of trees", IEEE Transactions on Software Engineering, Vol 7. No. 2, pp. 223-228, 1981

[13] Open Problems in Graph Drawing, http://problems.graphdrawing.org/index.php/Main_Page

[14] Microsoft, http://www.microsoft.com/korea/office/visio/prodinfo/overview.asp,2005

[15] SimTech System, http://www.thinkwise.co.kr, 2005.

[16] Microsoft,http://www.microsoft.com/korea/office/project/prodinfo/overview.asp,2005

[17] KAIST VMS Lab,http://vms.kaist.ac.kr/research_area/mes.aspx,2005

[18] 임형철, 송영석, 김창덕, 송성진, “공업화부재의 생산/운반/양중관리를 위한 정보가시화 시스템 개발에 관한 연구”, 대한건축학회 학술발표대회논문집, 제23권 제1호, pp.815-818, 2003

[19] Graphviz - Graph Visualization Software http://www.graphviz.org/

[20] Joseph O'Rouke, "Computational Geometry in C second edition", Cambridge Univ. Press 1994, 1998

[21] Drawing graphs with dot, http://www.graphviz.org/Documentation/dotguide.pdf

[22] Drawing graphs with NEATO http://www.graphviz.org/Documentation/neatoguide.pdf