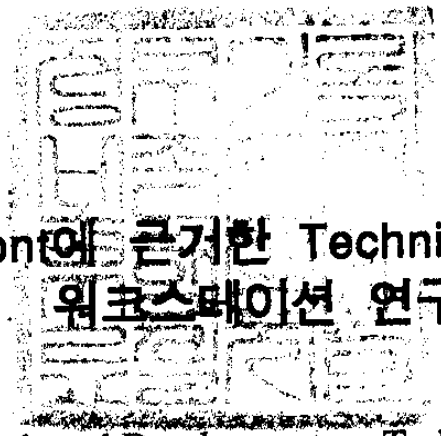


Metafont에 근거한 Technical Publishing 워크스테이션 연구개발

A Study and Development on Technical Publishing Workstation by using Metafont

주관연구기관

한국과학기술원

수행연구기관

광운대학교

과 학 기 술 처

제 출 문

과학기술처장관 귀하

본 보고서를 "워크스테이션 및 주변기기 개발" 사업의 세부과제 "Metafont"에 근거한 Technical Publishing Workstation 연구개발" 사업의 최종보고서로 제출합니다.

1990. 5

주관연구기관명: 한 국 과 학 기 술 원
수행연구기관명: 광 운 대 학 교
총 괄 책 임 자: 전 길 남(한국과학기술원 책임연구원)
연 구 책 임 자: 이 수 연(광 운 대 학 교 교 수)
연 구 원: 최 기 호(" ")
홍 은 선(광 운 대 학 교 박사과정)
김 차 종(" ")
정 회 경(" ")
임 광 택(" ")

요 약 문

I. 제 목

Metafont에 근거한 Technical Publishing 워크스테이션 개발

II. 연구개발의 목적 및 중요성

본 연구는 메타폰트에 근거한 Technical Publishing 워크스테이션을 연구개발함으로서 사무행정 분야 및 출판분야에 있어서 고품위의 한글처리 환경을 조성하고 워크스테이션 개발 기술을 국내기업에 정착시켜 수입대체 효과 및 해외시장으로의 확대진출을 도모한다.

III. 연구개발의 내용 및 범위

1. Compound Document Tool의 개발
   - Flexible Spread Sheet 연구개발
   - Graphics정보, Raster image정보, Texure정보의 integration

2. Publishing System의 X 윈도우 시스템으로 이주
   - X 윈도우 시스템 및 MOTIF의 분석
   - Batch Mode Publishing시스템의 이식

3. Interactive Mode Publishing System의 기초연구
   - Icon Based Man Machine Interface 연구
   - WYSIWYG alike한 레이아웃 처리의 연구

IV. 연구개발 결과 및 활용에 대한 건의

1. 개발결과

- Compound Document Graphic Edtor
- WYSIWYG 처리 위한 환경구축
- X윈도우 및 Motif환경으로 이주
  - . Font Generation System
  - . Structured Editor
  - . Preview Editor

2. 활용방안

- Full WYSIWYG방식과 Markup방식이 동시에 가능한 시스템의 연구개발을 계속 추진
- DDE(Dynamic Document Editing)환경 구축
- 국내기업에 기술 전파

목 차

제 1 장 서 론 1

제 2 장 Compound Document Graphics Editor 3

2.1 그래픽 에디터의 구성 3

2.2 그래픽 에디터의 설계 4

제 3 장 Publishing시스템과 X 윈도우 시스템의 결합 21

3.1 X 윈도우 시스템의 특징 및 구조 21

3.2 그래픽 사용자 인터페이스(Motif) 26

3.3 X윈도우 시스템으로 이주를 위한 전략 28

3.4 Document Editor 29

3.5 Preview부 38

제 4 장 WYSIWYG alike한 레이아웃 처리 41

4.1 Batch Mode레이아웃 처리 41

4.2 WYSIWYG alike한 레이아웃 처리 46

4.3 Preview Editor 54

제 5 장 실험 및 결과 56

제 6 장 결론 68

참고문헌 69

제 1 장 서 론

최근들어 컴퓨팅 환경은 새로운 메디아(media)의 출현과 더불어 사용자 중심 인터페이스(interface)기술의 통합을 가능하게 하고 있으며, 추상적인 텍스트(text)중심의 처리보다는 객체지향(object-oriented)적 처리방식과 이기종간의 서로 다른 환경에서 자원공유(sharing resouce)및 분산환경(distributed environment)을 가능하게 하고있다. 이러한 컴퓨터 기술 발전에 힘입어 문서처리 분야에서도 종래의 비대화적이고 전문가 위주의 방식에서 사용자 중심의 처리방식으로 급변하고 있다. 전자적 문서 처리의 노력으로 워드프로세서(Word Processor) 등과 같은 여러가지 사무기기들이 사실상 현실화 되어가고 있으며, 다양한 문서를 얻기위해 최근들어 전자출판시스템(EPS:Electronic Publishing System, DTP:DeskTop Publishing)[1]과 같은 시스템들이 보편화 되어 가고 있는 추세이며 관심이 급증하고 있다.[11,12]

문서는 텍스트 및 도형 그래픽(geometric graphic), 라스터 그래픽(raster graphic)등을 포함한다. 이러한 문서 정보를 갖는 문서처리 시스템으로는 Ventura, PageMaker 등이 있다[13,14]. 이러한 문서처리 시스템들은 선진 외국을 중심으로 활발히 진행되어 왔다.

한편 최근들어 고성능 워크스테이션 및 네트워크 기술이 발전함에 따라 이를 기반으로 하는 그래픽 사용자 인터페이스(Graphical User Interface)의 표준화의 움직임이 국제적으로 활발히 진행되고 있다. UNIX운영체제가 대형에서 부터 PC수준까지 실행이 가능해짐에 따라 UNIX를 기반으로 하는 그래픽 사용자 인터 페이스가 더더욱 확산되고 있는 추세에 있으며 이러한 워크스테이션의 표준 윈도우 시스템으로 X 윈도우 시스템이 자리를 굳혀가고 있는 실정에 있다.

또한 이러한 X 윈도우 시스템을 기반으로 사용자에게 더 편리한 환경을 제공하기 위해서 OSF(Open Software Foundation)와 UI(Unix Internalization)에서는 Motif[2]와 OpenLook[3]을 GUI표준으로 제시하고 있다. 이들 그룹들은 나름대로의 지지기반을 가지고 그래픽 사용자 인터페이스로서 국제표준으로 만들려고 하고 있으며 이에 따라 한국, 일본,중국등 영어 문화권이 아닌 언어도 support하려 하고 있다. 이에 국내에서도 한글 그래픽 사용자 인터페이스를 연구개발 중에 있다[6].

본 과제에서는 이와같은 세계적인 GUI의 추세에 부응하여, 전자출판시스템의 개발환경을 UNIX운영체제상의 X윈도우 시스템을 기반으로 한 GUI로 하고 그 위에서 전자출판시스템과 Compound Document Tool, Metafont Generation Tool을 통합한 Technical Publishing 워크스테이션의 개발을 최종 목표로 한다.

본 보고서는 2차년도에 연구개발한 Batch Mode Publishing시스템을 UNIX상의 X윈도우 시스템에 이주시키기 위해서 그래픽 사용자 인터페이스 MOTIF의 분석과 이에 따른 기술적인 문제에 대해서 기술하고, Geometric/Raster Graphic, Text를 통합하고 Business Chart를 포함하는 Compound문서를 작성하기 위해 개발한 Compound Document 작성 tool에 대해서 설명한다.

본 보고서는 다음과 같이 구성되어 있다.

2장에서는 Compound Document 그래픽 에디터의 구성 및 구현에 대해서 살펴보고, Spread Sheet부와 Business Chart부에 대해서 기술한다.

3장에서는 2차년도에 개발한 Batch Mode시스템과 X윈도우 시스템과의 결합을 위해 X윈도우 시스템의 특징 및 구조에 대해서 기술하고 그래픽 사용자 인터페이스 Motif에 대해서 설명하며, 실제 이주 시 발생되는 기술적인 문제에 대해서 논한다.

4장 에서는 WYSIWYG alike한 레이아웃 처리에 대해서 상세히 설명하며 5장에서는 실험 및 결과를 보이고 6장에서는 결론을 맺는다.

제 2장 Compound Document 그래픽 에디터 부

본 장에서는 문서내에 삽입될 그래픽 정보 를 생성하고 수정하는 그래픽 작성용 에디터의 구성 및 설계에 대해서 기술하고 에디터내 각부의 세부적인 처리과정 및 방법에 관해서도 논하기로 한다.

2.1 Graphic Editor의 구성

본 절에는 그래픽 정보를 다루는 그래픽 에디터의 구성에 대해 서술한다. 여기서 그래픽 정보란 Chart, Graphic primitive (geometric / raster), Graphical Text 등을 의미한다. 그래픽 에디터에서 만들어진 그래픽 정보의 최종 결과는 문서내의 그림 정보로써 이용되므로 그래픽 에디터는 system 전체로 보아서 입력부에 해당한다. 에디터의 간략한 구성도는 그림 2-1 와 같으며 기능상 분류하여 spread sheet 부, 그래픽 발생 및 조정부, 출력부로 나누어 진다.

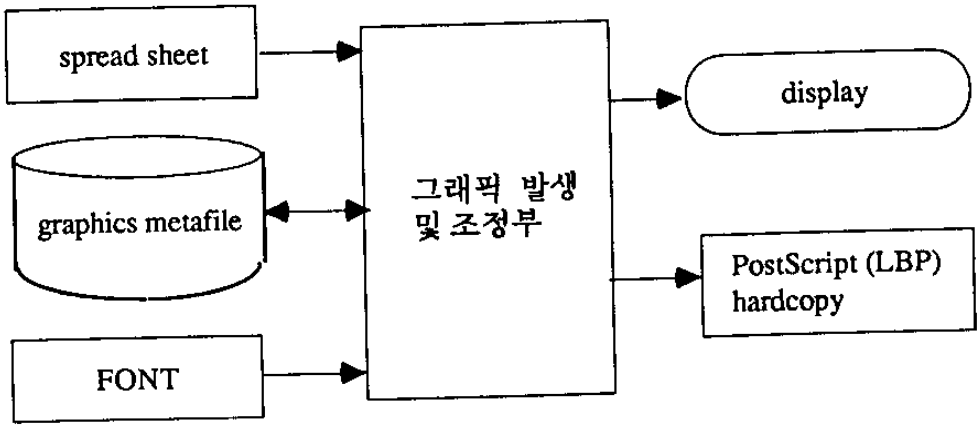


그림 2-1 그래픽 에디터의 간략한 구성도

Spread Sheet 부는 그래픽 발생 및 조정부에서 생성할 chart 를 구성하기 위해 필요한 데이타 (Label, Regend, Constant) 를 작성하고 이들 데이타에 대한 화일 입출력을 담당하는 부분이며 그래픽 발생 및 조정부는 Spread sheet에서 작성된 데이타를 이용하여 chart를 발생하며 그 밖에 Graphic primitive (geometric / Raster , Text) 등의 생성 및 조정를 담당한다. 그리고 출력부는 그래픽 발생 및 조정부에서 처리한 결

과를 에디터 전용의 metafile (혹은 CGM) 로 출력을 하며 또한 LBP 로의 HardCopy 를 위해 PostScript file 출력을 한다.

2.2 Graphic Editor의 설계

본 절에서는 그래픽 에디터의 각부의 설계에 대해 논하고 각각의 기능들에 대한 서술과 아울러 처리 알고리즘에 대해서도 논하기로 한다.

2.2.1 Spread Sheet부의 설계

Spread Sheet부는 그래픽 에디터의 내부에서 기능적으로 보면 입력부에 속한다. 그래픽 에디터에서 Chart를 생성하기 위해서는 정해진 데이타와 형식이 필요하다. 즉 spread sheet는 Chart data 작성용 에디터로써 이에 필요한 특징과 출력 형태는 LOTUS 123 과 Exclaim (X window상에서 구현된 LOTUS 123) 의 구조를 참조하였다.

2.2.1.1 데이타 모델

데이타 구조는 data value, data type, data mask, label value로 되어 있으며 이러한 데이타는 격자 형의 테이블 형태를 가진다. 즉 그림 2-2와 같이 2개의 independent variable과 하나의 dependent variable 로 나누어 지는데 한쌍의 independent variable이 하나의 dependent variable을 구성한다. 일반적으로 independent variable은 그 data type이 문자열이며 dependent variable은 실수 값을 가진다.

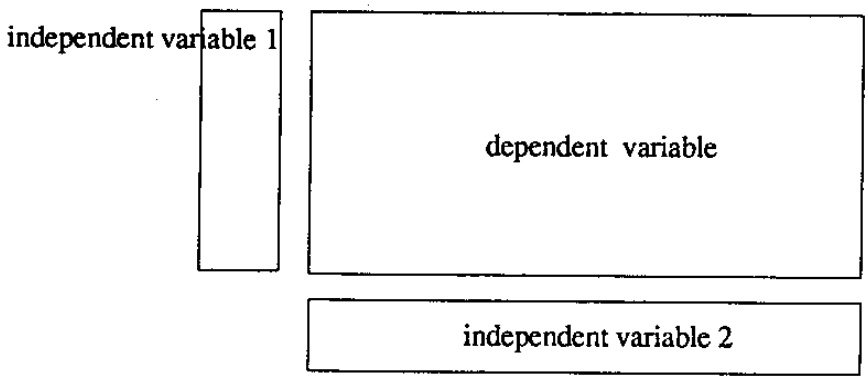


그림 2-2 Spread Sheet 데이타 모델

2.2.1.2 구성

Spread Sheet는 다음 그림 2-3 과 같은 module로 구성된다.

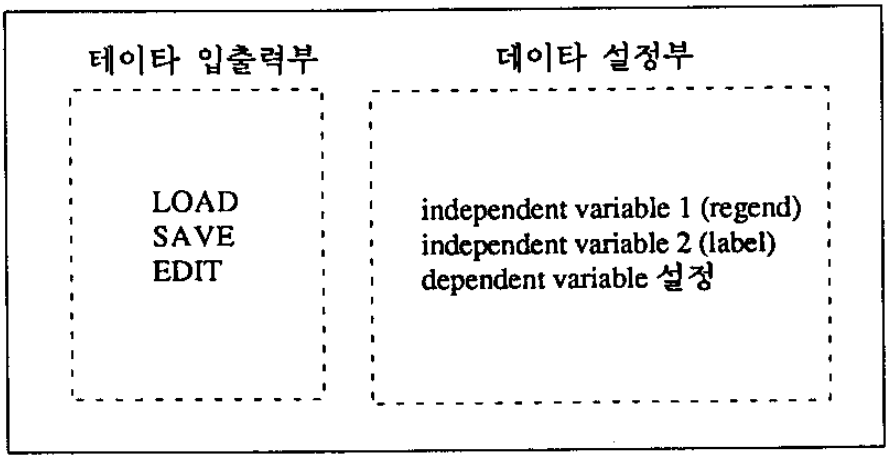


그림 2-3 spread sheet 의 구성도

2.2.1.2.1 Data 입출력 부

Chart용 데이타는 실제로 spread sheet 환경에서 얻어진다. 이때 먼저 spread sheet의 입력부에서 여러가지 데이타 타입과 데이타 값를 작성해야 한다. 이러한 데이타에 대한 생성과 수정된 결과가 정해진 포오맷을 가지고 spread sheet 전용의 데이타 화일로 저장 또는 피드백 (feedback) 되어 재 편집이 가능하도록 하였다.

1) Cell 좌표계 표현

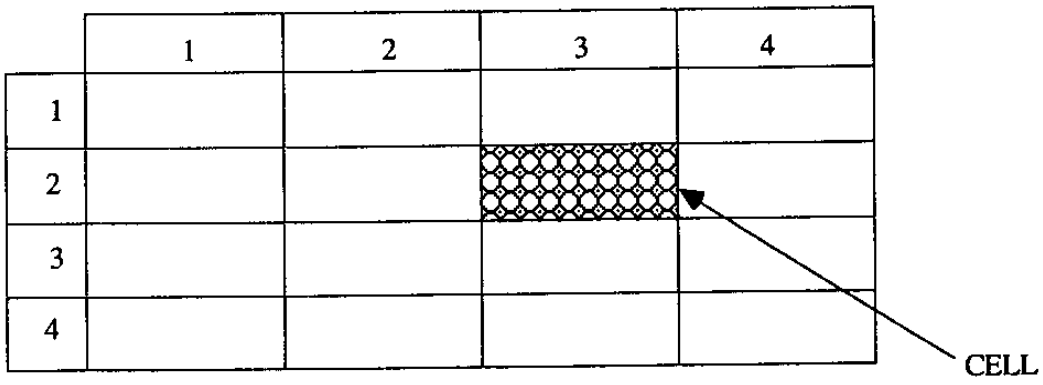


그림 2-4 셀 좌표

각각의 cell들은 하나의 데이타 타입과 데이타 값를 가지고 있으며 cell 데이타를 입력하기 위해 multiline Text Editor를 두어 일반 텍스트 에디터같은 문자 삽입, 제거등의 기능을 제공하였다. 또한 입력된 데이타의 길이에 따라 같은 가로, 세로 축의 sibling cells 의 위치와 크기를 조절하여 전체 적인 균형이 맞도록 설계하였다. 다음은 그 예를 보인다.

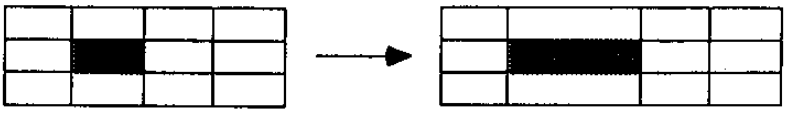


그림 2-5 셀 크기 조절 예

2) 데이타 화일 저장 구조

spread sheet 데이타를 화일로 임시 저장할때 다음과 같은 구조를 가지고 처리한다.

X	Y	LEN	DATA	ATTR

그림 2-6 Spread Sheet 화일 포오맷

X, Y : cell 좌표계에서 현 데이타의 cell 위치를 지정

LEN : cell내에 기록된 데이타의 길이를 지정

DATA: 실재 데이타 값 (일반적으로 문자열로 취급)

ATTR : cell내의 데이타에 대한 속성들을 지정 (ie, 폰트, 크기,서체 등)

2.2.1.2.2 데이타 설정부

여기서는 chart를 발생하는데 필요한 요소들 (regend, axis label, actual data) 을 지정하는 기능을 제공한다. 그림 2-7 은 그 설정 예를 보여주고 있다.

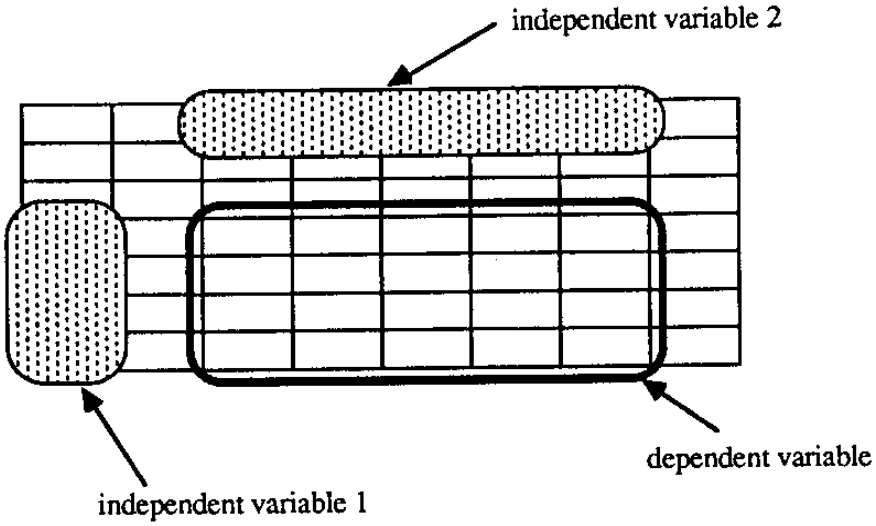


그림 2-7 데이타 설정 예

independent variable 1 (regend) : chart의 regend 문자열를 지정

independent variable 2 (axis label) : chart의 axis에 나타낼 label문자열를 지정

dependent variable : 실수값으로 된 actual data를 지정

2.2.2 기타 입력 데이타들

Editor전용의 geometric / raster graphics metafile을 입력으로 하고 있으며 이는 출력과 같은 형태를 가지므로 후에 출력부에서 다시 논하기로 한다. 그래픽 발생 및 조정부에서 텍스트를 발생하기 위해서는 폰트 resource 가 필요한데 X window 에서 제공하는 bitmap fonts를입력으로 하고 있다. 여기에서 쓰인 폰트들은 PostScript interpreter를 내장한 laser printer (LBP) 의 폰트 리스트와 일대일 대응하므로 텍스트의 hardcopy처리가 쉽게 된다.

2.2.3 그래픽 발생 및 조정부

그래픽 발생 및 조정부는 spread sheet에서 작성된 데이타를 이용하여 chart를 구성하는 부분이며 또한 선 도형등의 그래픽 프리미티브의 발생등를 행하게 된다. 여기서는 그래픽을 작성하는데 있어 base가 되는 canvas를 정의하고 이 canvas에 chart나 geometric / raster graphics, text등을 발생시키는 과정과 방법에 대해 논하기로 한다.

2.2.3.1 canvas 정의

1) canvas는 연관된 그래픽 정보를담는 하나의 프레임으로 볼 수 있으며 직사각형 영역으로 표현된 window 이다.
2) canvas는 최상위에 있는 root canvas와 그 아래 여러개의 child canvas로 구성된 그림 2-8 같은 depth 2 인 계층 구조를 가진다
3) root canvas는 이동과 크기조정이 불가능한 최상위의 canvas를 의미한다.
4) child canvas들은 root canvas에 상대적인 좌표을 가지며 크기조절이 가능하며 canvas 들이 중첩될때 불투명한 특성을 가진다.
5) child canvas는 서로 link로 연결되어 삭제와 삽입이 가능하며 나름대로의 stacking order를 가진다.
6) child canvas들은 root canvas 의 영역을 벗어나 생성될 수 있으나 그 계층구조는 유지하고 있다.

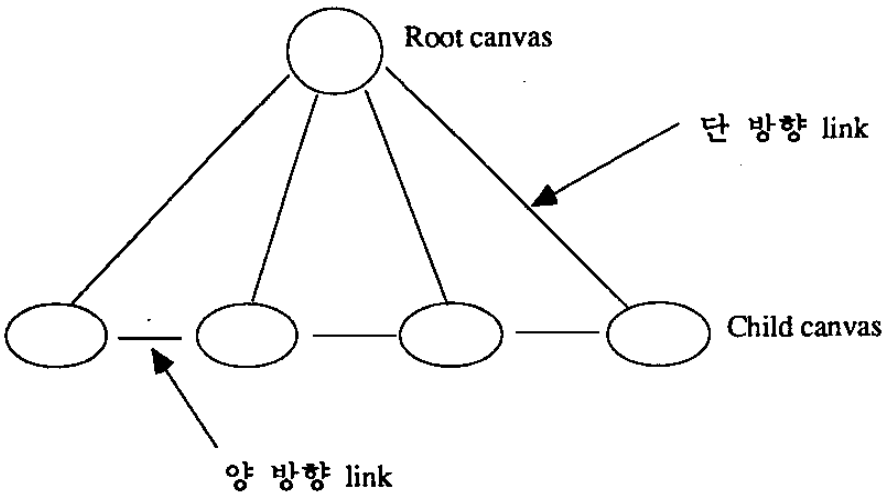


그림 2-8 Canvas 의 계층 구조

2.2.3.2 canvas 운영

1) 마우스 포인터 (mouse pointer) 를 이용하여 canvas의 꼭지점들을 지정하여 직사각형 영역을 만든다
2) 현재 만들어진 canvas가 스택의 최상위에 놓인다

3) desktop metaphor의 개념으로 선택된 canvas는 스택의 최상위로 놓이며

4) canvas의 크기와 위치조정은 canvas의 모퉁이에 있는 control pointer(handle) 을 이용한다.

5) canvas내에서 생성된 그래픽 정보들은 canvas의 사각형 영역을 벗어나지 못한다

6) canvas를 단위로 한 화일 입출력을 하며 또한 hardcopy의 한 단위가 된다.

2.2.3.3 Chart 정의

이 절에서는 chart 를 구성하는데 필요한 chart의 type, chart environment, chart parameter, chart data model에 대해 논하도록 한다.

2.2.3.3.1 chart 타입

chart 타입은 크게 2가지로 분류할 수 있는데 각각의 chart는 또한 subtype을 가진다. 즉 방향성, 모양 (shape), 강조(extrusion), 원근성 (입체) 이 그것이다.
주요 chart 타입은 다음과 같다.

- BAR (vertical & horizontal, line, scatter)
- PIE

2.2.3.3.2 chart environment

chart environment는 직사각형으로 정의된 영역이며 그림 2-9 와 같이 chart area , margin 그리고 plot area로 나뉜다.

1) chart area는 chart가 구성될때 생기는 영역을 나타내는 것으로 margin과 plot area를 포함한다.

2) margin은 plot area의 네 방면에 정의하며 chart area와 plot area간의 간격을 지정한다.

3) plot area 는 chart가 구성되어 그려지는 영역으로 chart area의 한 부분이다.

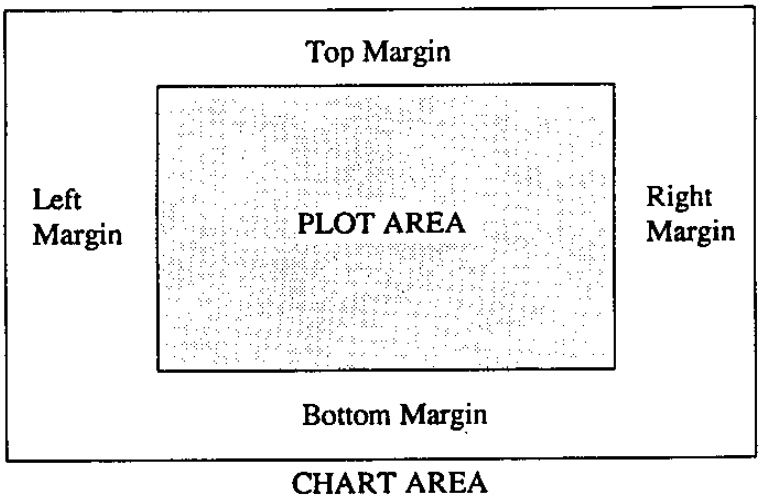


그림 2-9 Chart Environment

2.2.3.3.3 chart parameter

chart parameter는 chart를 생성함에 있어서 여러가지 style 을 정의하는 parameter 이다. chart parameter는 다음과 같다.

-Axis label

-Axis tick mark

-Chart title

-Chart regend

-Pie chart thickness

-pie chart explosion

-chart element spacing

-chart element fill pattern

2.2.3.3.4 data model

chart의 data structure는 spread sheet와 유사하다. 그림 2-10 은 그 구조를 나타내고 있다.

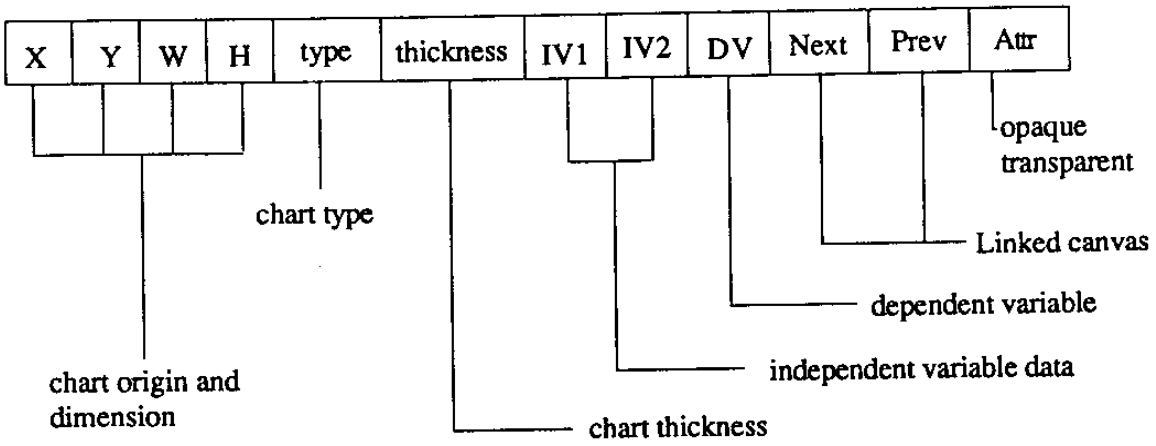


그림 2-10 챠트의 데이타 구조

2.2.3.4 chart 생성 및 처리 과정

chart 의 생성을 위해서는 먼저 spread sheet 에서 정의된 label, axis label, actual data를 필요로한다. 여기서 얻어진 데이타는 chart 구조에 적절히 맞도록 변화시킨다. chart 타입과 chart area 를 지정하여 chart environment 를 정하며 chart 에 style parameter 를 주어 세부적인 에디팅이 가능하도록 한다. 이때 chart 의 크기조정과 이동은 chart 를 생성할때 생기는 handle 을 이용한다.

다음 그림은 chart 의 생성 및 처리 과정에 대한 구성도이다.

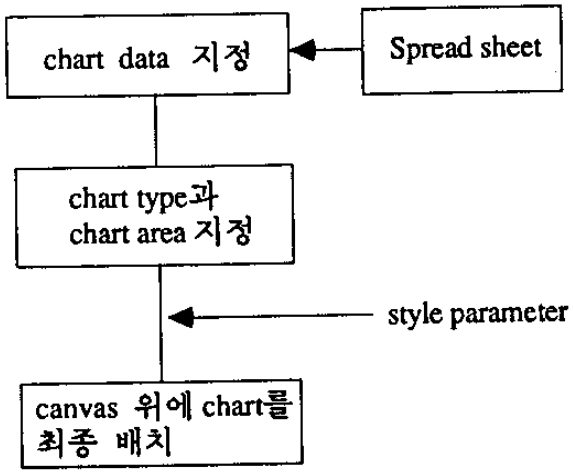


그림 2-11 챠트 처리 과정

2.2.3.5 그래픽 파레트

그래픽 파레트는 graphic object primitive를 생성하고 수정하기 위한 아이콘 (icon) 으로 만들어진 툴 박스이다.

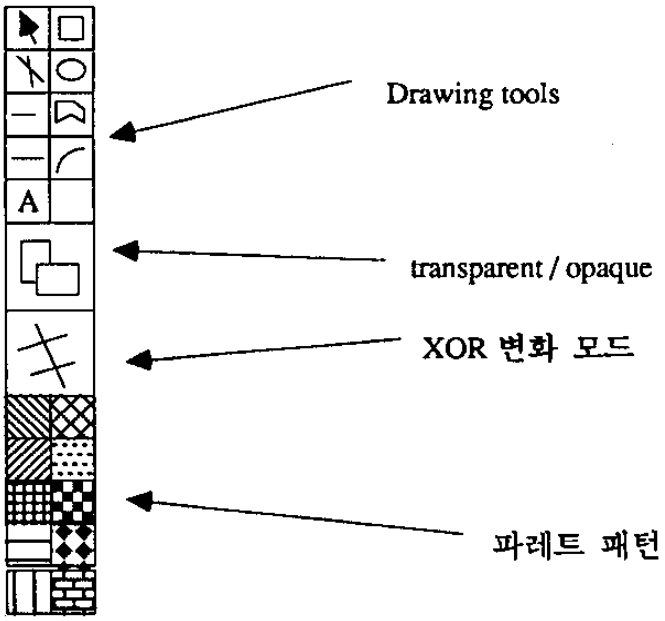


그림 2-12 그래픽 파렛트

2.2.3.6 Drawing Tools

canvas위에 chart를 구성하는 것이외에도 선도형, 텍스트등의 graphic primitive 를 생성할 필요성이 있으므로 geometric (예를 들어 선, 다각형, 사각형, 타원, arc) graphic primitive, raster graphics (image), text 을 처리하는 Drawing tool 에 대해 논하기로 한다.

2.2.3.6.1 선택 툴

그래픽 오브젝트를 선택하기 위해 이용되는 부분으로 선택된 오브젝트는 4 개의 control pointer (handle)이 생기며 이 handle 중의 하나를 선택하여 그래픽 오브젝트의 크기 조절을 하며 오브젝트의 다른 부분을 드래깅(dragging) 하여 오브젝트의 이동을 한다.

2.2.3.6.2 그 밖의 drawing tool들

그래픽 오브젝트의 생성에 관련된 부분으로 여기서 생성 가능한 오브젝트들은 선, 화살표, 양 방향 화살표, 텍스트, 사각형, 타원, 다각형, arc등이다.

2.2.3.6.3 텍스트 툴

텍스트 오브젝트는 앞서 언급한 바와 같이 X window에서 제공하는 5 가지의 폰트를 이용하는데 각각의 폰트에 대해 자유로운 크기를 제공하기 위해 기준이되는 폰트 크기 (14)를 정하여 bitmap 축소/확대 알고리즘을 이용, 자유로운 폰트 크기를 지원한다. 이를 처리하기 위해서 먼저 텍스트를 pixmap 에 그리는데 pixmap에서는 임의의 pixel에 대해서 pixel 값을 얻을 수 없기때문에 다시 이미지 (image) 로 변환을 하여야 한다. 위 과정을 그림 2-13에 도식화 하였다.

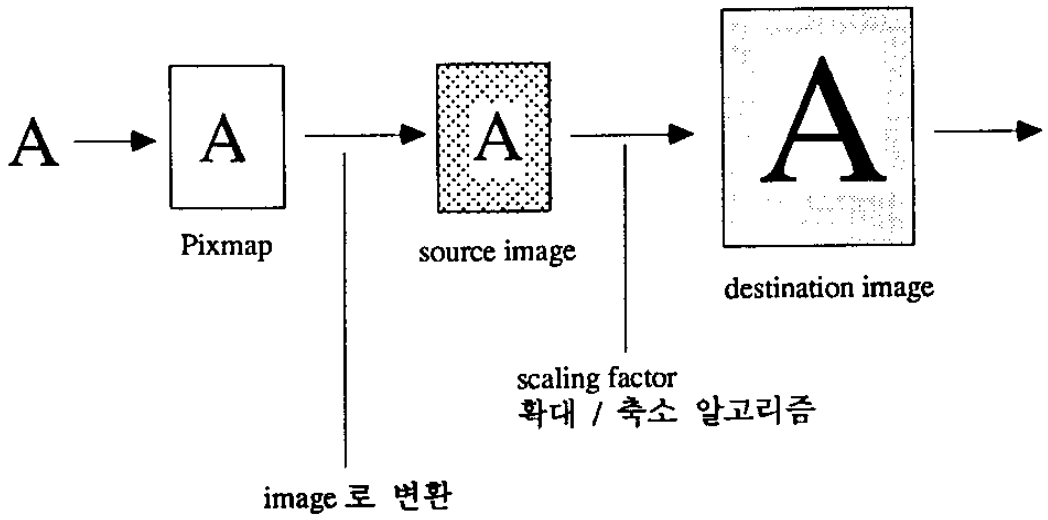


그림 2-13 이미지 변환과 확대

이때 확대./축소 알고리즘은 inverse distance method를 이용하였는데 처리시 다른 알고리즘에 비해 정보의 손실이 작고 속도가 빠른 장점을 가지고 있다. [16]

2.2.3.6.4 그래픽 오브젝트의 데이타 구조

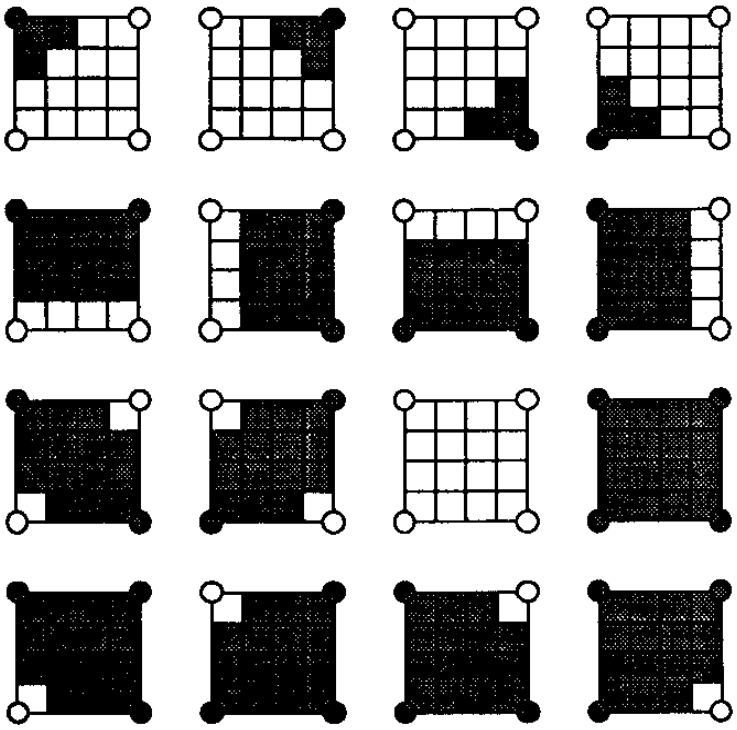


Table pattern example of 16 segmented area

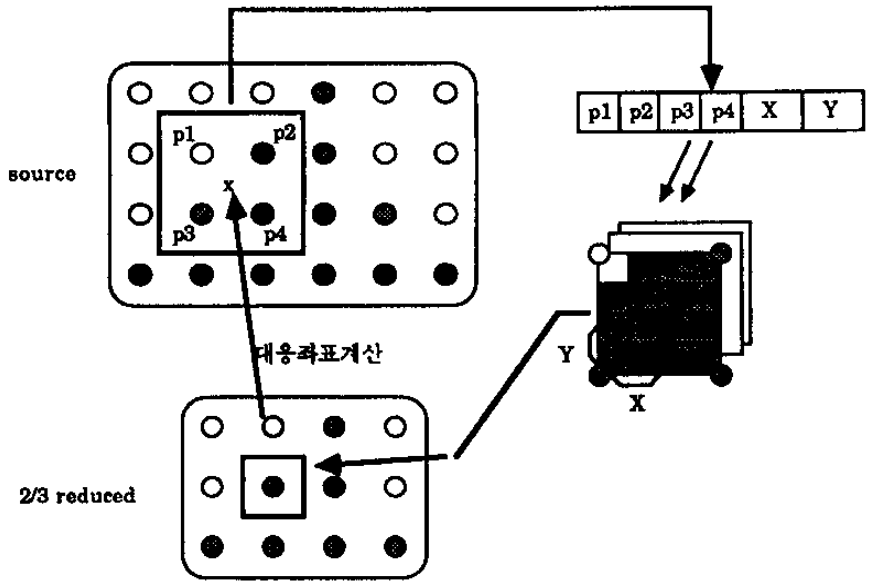


그림 2-14 High speed inverse diatance method

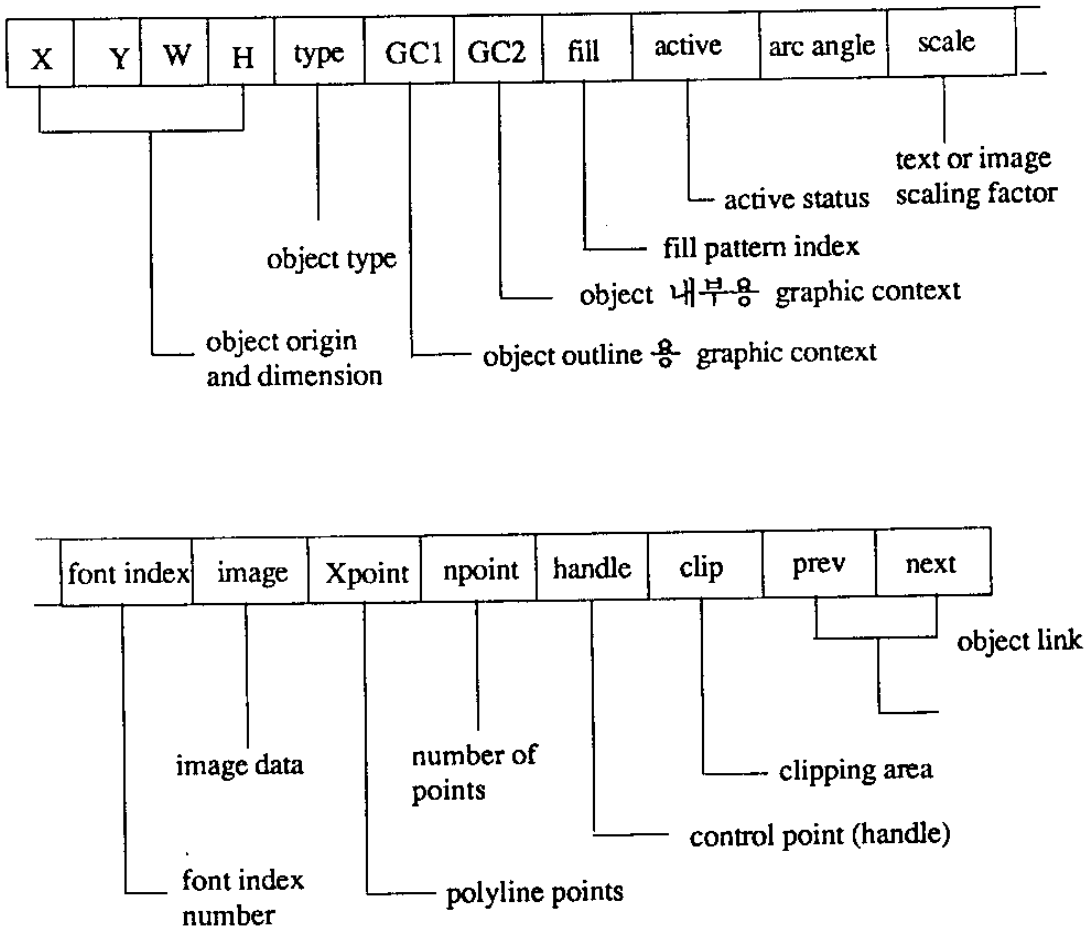


그림 2-15 그래픽 오브젝트의 데이타 구조

2.2.3.7 오브젝트 수정자

오브젝트수정자는 선택 툴로 선택되어진 오브젝트에 대한 속성을 조정하는 부분으로써 선의 굵기, XOR 변화 모드, 투명 / 불투명 속성을 줄 수 있다.

2.2.3.7.1 선의 굵기

텍스트이외의 모든 graphic primitive object를 대상으로 선의 두께를 조절할 수 있다

2.2.3.7.2 XOR 변화 모드

현재 canvas에 그리려고 하는 오브젝트를 source라하고 이미 canvas에 그려 져 있는 오브젝트를 destination이라 했을때 source object와 destination object가 서로 교차될때 교차되는 pixel 사이의 조합 관계를 exclusive-or로 한다. 그림 2-16 은 이러한 처리 방법의 예를 보이고 있다.

그림 2-16 XOR 변화 모드의 예

2.2.3.7.3 투명, 불투명성

채움 패턴으로 채워져 있는 오브젝트에 대해 이러한 속성을 줌으로써 pattern의 흰 pixel에 대해 opaque 또는 transparent한 특성을 갖게 할 수 있다.

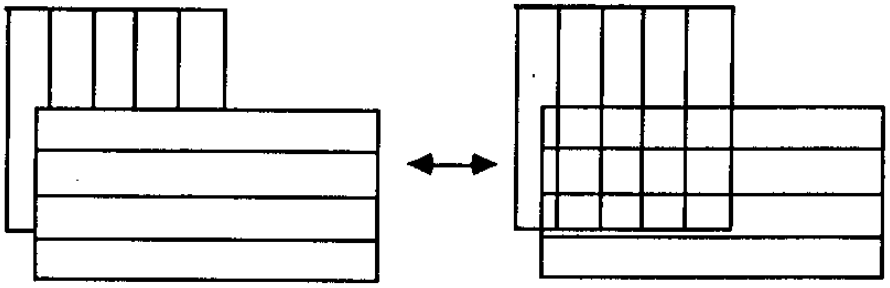


그림 2-17 투명, 불투명 성

2.2.3.7.4 채움 패턴

오브젝트 내부와 outline에 대한 채움 패턴을 두어 다양한 pattern을 구사한다.

2.2.3.8 그래픽 프리미티브 (Graphic primitive) 조정부

그래픽 프리미티브 조정부는 현재 선택된 오브젝트의 resource을 복사 매개체인 클립 보오드 (clipboard) 에 저장하여 오브젝트의 삭제, 복사, 첨부등의 기능을 수행하고 또한 오브젝트의 stackng order를 변화시키기 위한 부분에 관한 것이다.

2.2.3.8.1 resource 전달 관계

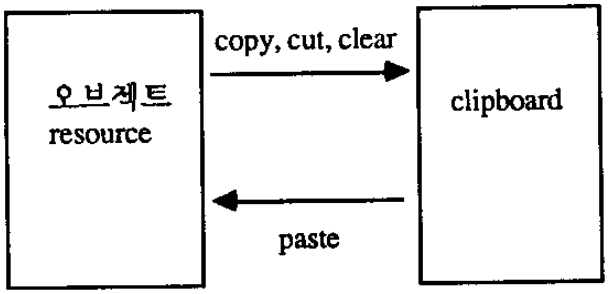


그림 2-18 Resource 전달 관계

2.2.3.8.2 stacking order

그래픽 오브젝트는 생성시에 stack의 최상위 (또는 link의 끝)에 놓인다. 오브젝트들이 겹쳐 있을때 그 들 중 선택된 오브젝트는 stack의 top 또는 stack bottom에 놓을 수 있다.

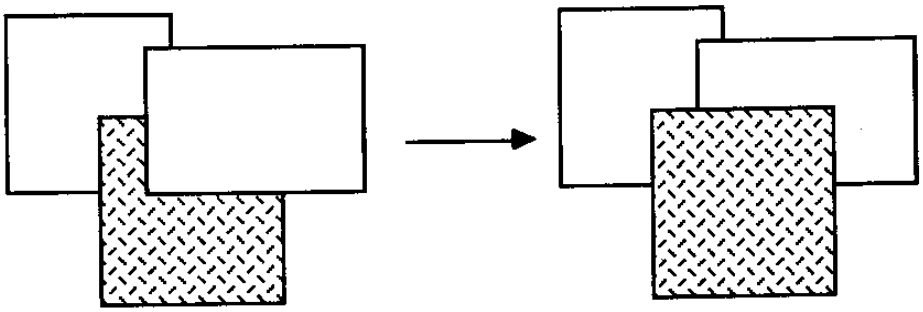


그림 2-19 스택 순서 조정

2.2.3.9 raster 처리

raster 정보는 이미 만들어진 비트 맵를 image로 변화시켜 canvas에 사상시키며 image에 대한 크기 조절을 자유롭게 할 수 있다. image 의 크기 조절에 필요한 알고리즘은 텍스트 툴부에서 사용했던 inverse distance method를 이용하였으며 축소된 image를 대상으로 또 다른 확대를 하면 정보 손실이 크므로 원 image와 사상할 image 2개를 두어 처리한다.

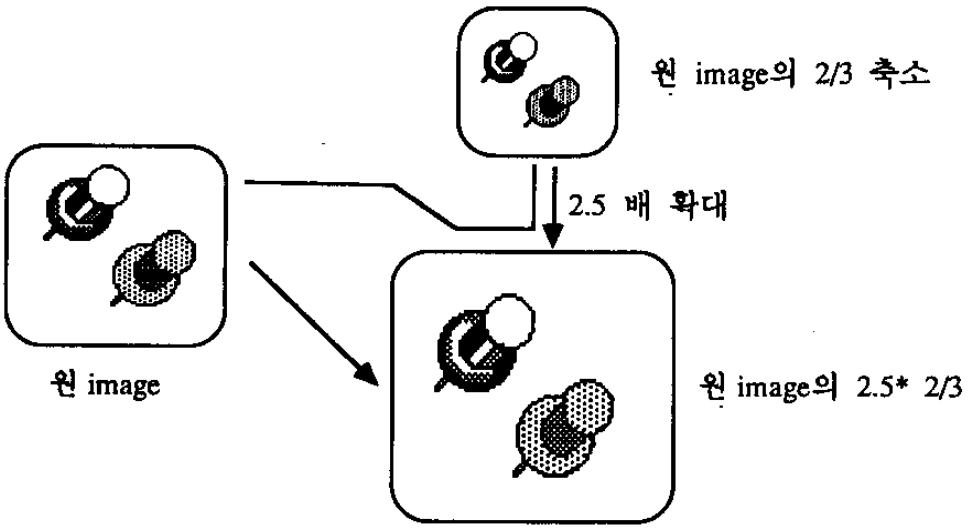


그림 2-20 이미지 처리 예

2.2.4 Graphic editor 출력부

그래픽 발생 및 조정부에서 처리된 정보를 정해진 형식을 갖는 화일로 변환하여 최종 결과를 내는 출력부에 대해 서술한다.

2.2.4.1 구성

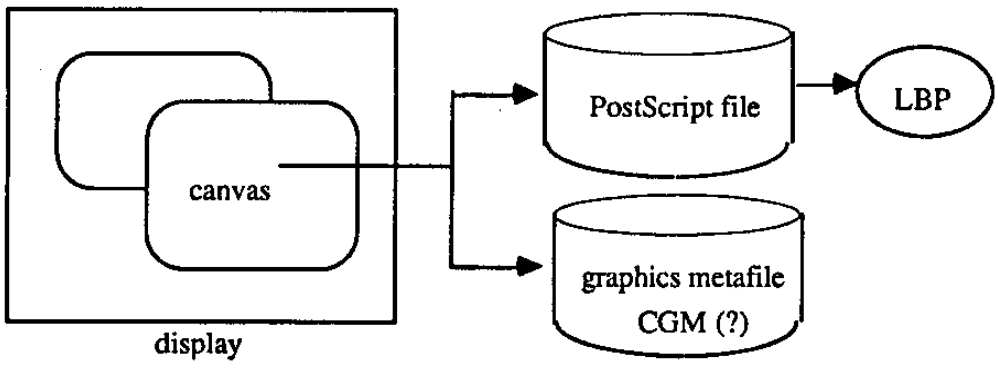


그림 2-21 출력부 구성도

출력부에서는 고품질의 output을 얻기위해 LBP용 page description language의 표준이 되어 있는 PostScript file로 그 출력을 내는데 이는 그래픽 오브젝트에 대한 자유로운 축소/확대가 가능하며 특히 텍스트의 hardcopy가 용이하다.

2.2.4.2 metafile

그래픽 발생 및 조정부에서 만들어진 그래픽정보는 다시 피드백 (feedback) 편집이 가능하도록 일정 형식을 가지는 geometric / raster metafile로 저장한다. 그 형식은 그래픽 오브젝트의 형식과 같다.

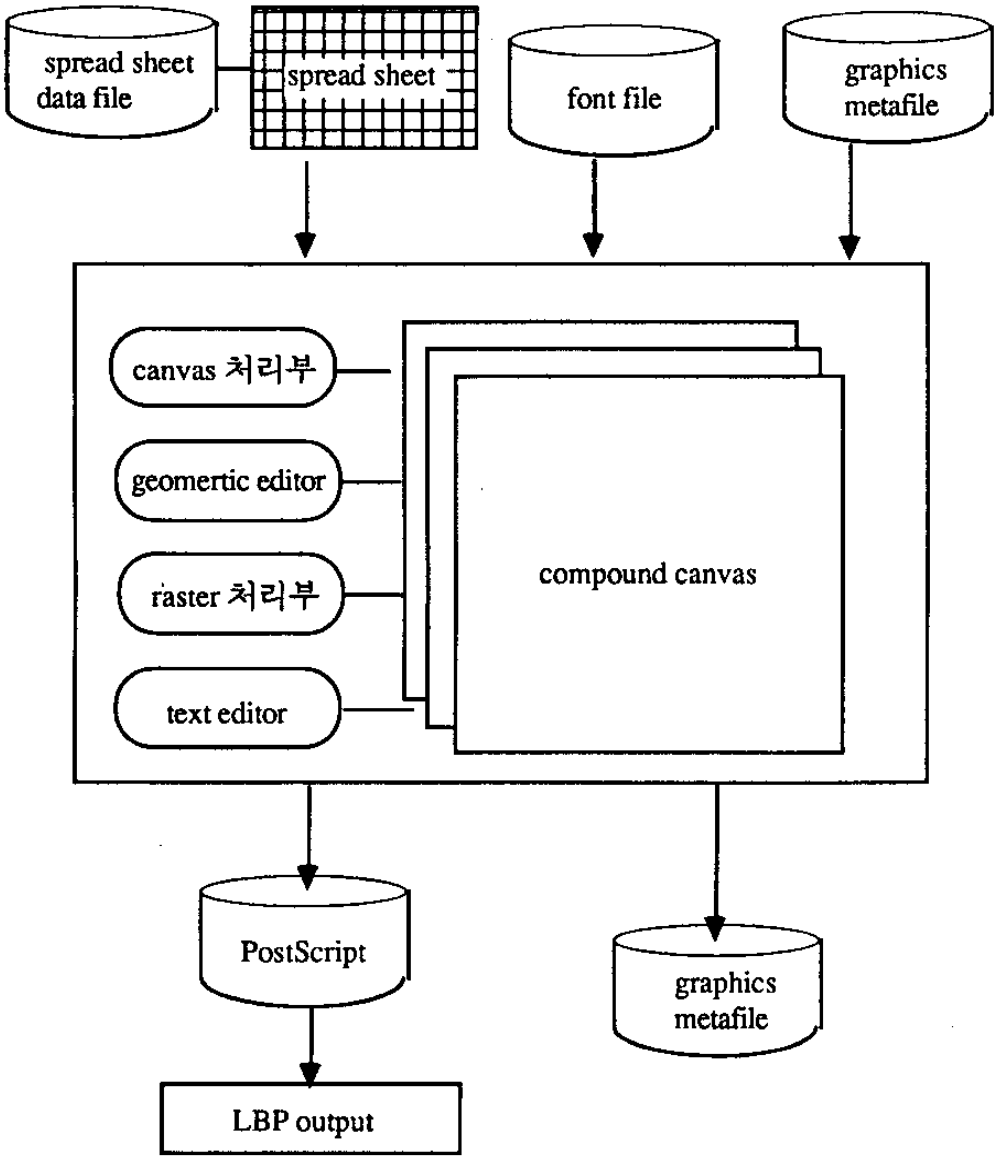


그림 2-22 그래픽 에디터의 세부 구성도

제 3 장 Publishing System과 X 윈도우 시스템의 결합

본 장은 본 과제의 2차 년도에 수행한 Batch Mode Publishing시스템과 현재 수행중인 그래픽 에디터와 통합하고 이와 동시에 이들 시스템들을 X 윈도우 시스템과 결합하기 위한 문제에 대해서 기술한다.

X 윈도우 시스템[1]은 사실상 워크스테이션 상의 표준 윈도우 시스템으로 되고 있으며 네트워크를 기반으로한 그래픽 사용자 인터페이스로 사용되고있다. 또한 UNIX와 X를 기반으로 한 워크스테이션에서의 그래픽 사용자 인터페이스(Graphical User Interface : GUI) 에 대한 표준화의 움직임이 국제적으로 활발히 진행되고 있는 추세속에서 Motif[2] 나 OpenLook[3]이 양대 표준으로 대두되고 있다. 본 장에서는 이러한 그래픽 사용자 인터페이스를 제공하는 X와 Motif 에 Publishing System의 이주를 위한 전략과 X의 특징 및 구조 를 설명하고 Motif를 이용한 Menu구성에 관해서 기술한다.

3. 1 X 윈도우 시스템의 특징 및 구조

3.1.1 X 윈도우 시스템의 특징

가. Design 전략

- 분산 환경에서 network transparent, device independent, multitasking 윈도우와 그래픽 시스템을 제공한다.

- 한 응용 프로그램이 여러 윈도우의 사용이 가능하고 또한 한 디스플레상에서 여러개의 응용 툴들이 윈도우로 표현 가능하고 각각 한 윈도우에서 서로 다른 일을 실행가능 하다.

- Overlapping or hidden window, soft font text, 2-d graphics drawing이 제공된다.

나. 장치 독립성

- X에서는 장치독립성을 위해 윈도우 드로잉 작업을 2부분으로 분류 하고 있다. : Client/ Server model

- Client : 응용프로그램으로 server에게 윈도우, 텍스트, 다른 오브젝트 드로잉을 요구한다.

- Server : 디스플레이상에 오브젝트를 드로잉 한다.

다. Client와 Server간의 통신

- Client는 X 프로토콜 형태의 명령을 패킷으로 서버에게 보낸다. 워크스테이션 서버는 하드웨어 의존적인 디바이스 드라이버를 가진다. X 서버는 스크린,키보드, 포인팅 디바이스를 제어하고, Client는 Xlib(graphics & 윈도우 함수들의 라이브러리)를 이용하여 워크스테이션과 연결된다.

라. Network Transparent

- 어떤 기계에 운영되고 있는 응용프로그램이 다른 기계에 있는 디스플레이 장치를 이용할 수 있으며, 이때 두개의 하드웨어는 똑같은 구조나 O.S를 가질 필요가 없다.

마. X server

X윈도우상에서 한 윈도우를 opening하는데에 통신의 체인은 7개의 링크를 가진다.

Application->Xlib->OS->X protocol->OS->X server->Screen

Server 는 워크스테이션상의 마우스, 키보드, 디스플레이스크린을 제어한다. single workstation은 여러개의 스크린을 가질 수 있고, single computer는 다른 그래픽 터미날을 갖는 하나 이상의 서버를 실행 시키수도 있다. 각 워크스테이션은 자신의 서버를 갖는다. single 서버는 많은 client응용프로그램으로부터의 요구를 받아들일 수 있기 때문에, 한 스크린상에 다른 프로그램에서 출력되는 결과를 갖는 여러 윈도우를 갖을 수 있다. client 프로그램은 서버 machine위에서 실행될 수도 있고 네트워크상의 다른 서버위에서도 실행가능하다.

- Server 의 주된 역활은 client application이 필요로하는 자원을 공유하게 하는것이다. 기본적인 두개의 자원으로는 text처리와 드로잉을 위한 processor

time과 screen space가 있다. 또한 server는 작업 스케쥴링과 메모리 관리, client의 통신 링크를 관리한다. 서버는 이러한 일을 OS하에서 실행한다. X윈도우는 분산 시스템으로 사용될 수가 있는데 , X 윈도우가 다른 워크스테이션상에 윈도우를 open할때 현재 디스플레이에 윈도우를 그리는것은 remote cpu가 한다. 구조적인 관점에서 서버를 보면, 서버는 X protocol format으로된 client요구 메세지를 받거나 번역하는 device-independent한 layer를 갖고, 특정한 오퍼레이팅 시스템과의 인터페이스 역활을 하는 operating-dependent한 layer를 갖으며 , 특정한 하드웨어를 support하기 위한 device-driver의 집합체인 device-dependent한 layer로 구성된다.

3.1.2 X윈도우 구조

X 윈도우 시스템은 공학용 워크스테이션을 위한 소트트웨어 환경이다.

X는 프로그래머와 응용 프로그램의 사용자에게 에게 풍부하고 복잡한 환경을 제공한다. X의 토대는 base 윈도우 시스템이다.

그림 3-1 은 base window를 토대로 하여 layer를 구성하고있는 전체적인 X 윈도우의 구조를 난타낸다.

이 구조에서 베이스 윈도우 시스템은 X 네트워크 프로토콜을 사용하여 외부 세계와 인터페이스 한다. 그 위의 네트워크 프로토콜은 단일 중앙처리장치 또는 여러 중앙처리 장치 들 사이의 내에서 작동하도록 되어 있는 부분으로, 장치 독립성과 X위에서 네트워크 트랜스패런트를 제공한다. 또한 베이스 윈도우와의 유일한 인터페이스를 하는 부분도 네트워크 프로토콜 부분이다. X 응용 프로그램은 직접적으로 네트워크 프로토콜을 사용 할수없고 프로그래밍 인터페이스를 통해서만 가능한데, 이러한 일을 하도록 되어있는 C 프로그래밍 서부루틴 패키지를 Xlib라 한다. Xlib는 응용 프로그램을 네트워크 프로토콜에 인터페이스 시키고 그런다음 base 윈도우 시스템과 인터페이스 시킨다. 대부분의 응용 프로그램은 네트워크 프로토콜의 복잡성을 프로그래머에게 감춘 X 툴키트를 이용 한다.

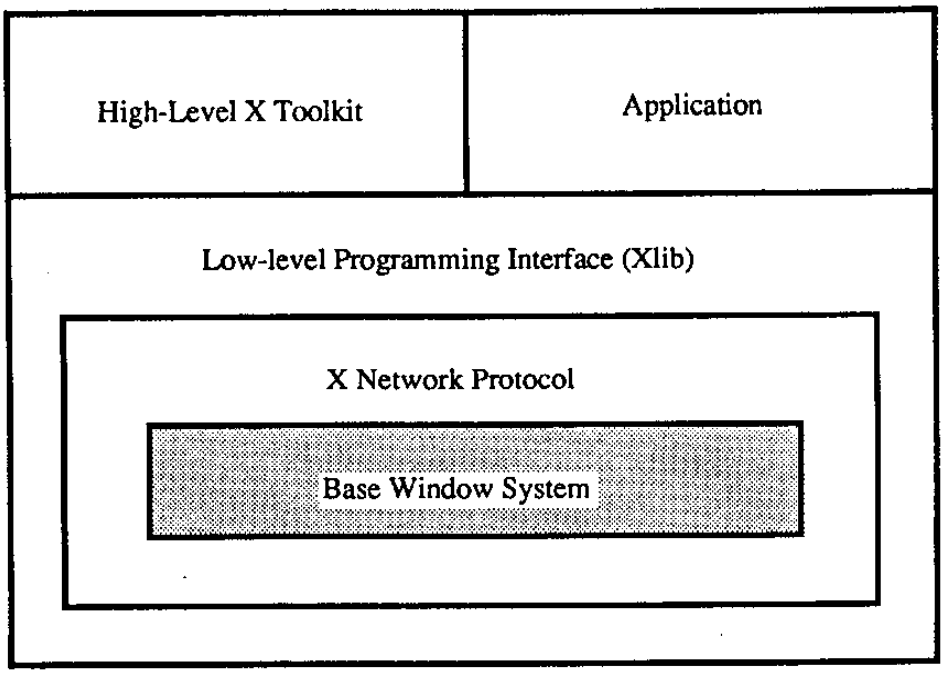


그림 3-1. X 윈도우 시스템의 구조

3.1.3 X 윈도우의 프로그래밍 환경

가. Xlib

X 윈도상에서 모든 윈도우 작업과 드로잉을 하기 위해서는 Xlib를 사용하여 프로그래밍해야한다. Xlib 는 C, Pascal, FORTRAN, Modula-2, ADA와 같은 프로그래밍 언어에서 불러쓸 수 있으며, 그래픽 라이브러리를 포함하여 200개 이상의 프로시쥬어로 구성되어 있고, bit map처리뿐만 아니라 clipping, stippling, tiling 등을 처리 할 수 있다.

또한 윈도우를 구성하고 생성하기 위한 프로시쥬어도 갖고 있는데 예를 들면 XCreateWindow, XResizeWindow, XDestroyWindow등이있고, event, query, 폰트 처리, 키보드, 포인터, 칼러의 제어를 할 수 있는 프로시쥬어도 포함하고 있다.

X 윈도우를 열기 위한 단계는 다음과 같이 8가지로 나타낼 수 있다.

1. XOpenDisplay를 사용하여 서버에게 display connection을 열어준다.

2. XCreateWindow로 최상위 윈도우를 생성한다.

3. 최상위 윈도우를 위한 표준 속성을 설정하고, 윈도우 매니저를 위해 힌트를 준다.

4. 그래픽 컨텍스트와 같은 윈도우 자원을 생성한다.

5. 필요한 다른 윈도우를 생성한다.

6. 이들 윈도우를 위해 필요한 event를 생성한다.

7. 윈도우를 맵시킨다.

8. event loop에 들어간다.

나. X 프로토콜

응용 프로그램과 Xlib루틴이 링크가 되면, X 윈도우 시스템은 이 응용 프로그램의 실행시 각 Xlib에 상응하는 X 프로토콜을 생성하여 서버에게 보낸다. X 프로토콜 request는 그림 3-2 에서 볼 수 있듯이 가변 길이를 가진 데이타 패킷으로, 처음 한 바이트는 각 request의 타입을 나타내는 8비트 Opcode로 시작하고, 다음에 길이를 지정하는 16비트 필드가 따르고, 그 다음은 하나 또는 그 이상의 부가적인 데이타 타입이 따른다.부가되는 데이타는 숫자 파라미터, 좌표, 텍스트 스트링일 수 있다.

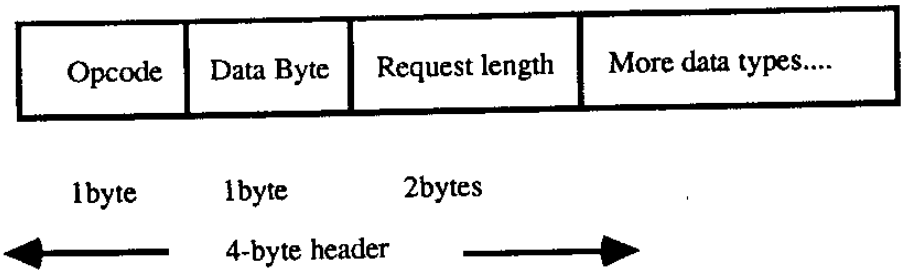


그림 3-2. Generic Request Packet

다. X toolkits

X toolkits는 Xlib보다 상위 레벨에 존재하는 Xlib 함수로 만들어진 조립식 루틴으로 프로그램의 개발을 빠르게 한다.

X 툴키트를 위해 이미 여러개의 prototype 및 완전한 X 툴키트가 개발되어 있고, 이를 사용하기 위한 잘정의된 widget이라 불리는 style이 support되고 있다. widget은 그래픽 오브젝트의 상태에 대한 정보를 가지는 데이타 구조로 완전한 inheritance 메카니즘을 가지는 클래스에 속한다.

3.2 그래픽 사용자 인터페이스(Motif)

3.2.1 그래픽 사용자 인터페이스[5]

사용자 인터페이스란 컴퓨터와 사용자간의 공유부분으로 키보드나 마우스를 이용해서 화면에 어떤 결과를 출력하기 위해 하게되는 입력을 말하거나 응용 툴 프로그래머가 새로운 툴을 개발 할때 사용하게 되는 부분을 말한다.

이 부분을 흔히 응용 프로그래머 인터페이스(Application Program Interface : API)이라 한다. GUI는 사용자 인터페이스중에서도 사용자가 화면을 보면서 입출력을 하게되는 부분으로 윈도우나 스크롤바, 메뉴의 모양이나 이를 만들어 내기위한 도구로서의 툴키트를 말한다.

3.2.2 Motif

Motif는 GUI의 일종으로 Xt intrinsic을 기반으로 하여 구성된 Widget의 집합이다. 이는 Xt intrinsic의 상위 레벨에 위치하면서 보다 낮은 레벨의 X 윈도우 시스템에 쉽고 빠르게 접근할 수 있게 해주며, 그래픽 사용자 인터페이스 툴키트, 윈도우 매니저, 스타일 가이드로 구성된다. 사용자에게 편리하고 일관된 사용자 인터페이스를 제공하기 위해 서로 다른 많은 widget을 가지는데 각 widget은 동적으로 생성되며 상태정보를 포함한다.

각 widget은 한 class에 속하고 각 class는 정적으로 할당되고 초기화되며, class를 위한 operation을 포함하는 구조를 가진다. 기본적인 widget class내에는 Core class가 존재하는데 이것은 모든 다른 Class에 의해서 상속받는 자원을 포함한다. Core class 아래에는 두개의 class가 존재하는데 , Composite class와 Primitive class이다. Primitive Class는 button(cascade button, Drawn button, push button, toggle button) 들을 위한 label을 정의하고 있으며, 이밖에도 arrow button, list, scrollbar, separator, text를 위한 widget들로 구성된다.
Composite class는 shell 및 manager widget으로 구성된다.

한편 Motif의 다른 widget으로는 dialog widget 및 gadget도 포함된다.

다. 사용자 인터페이스 참조 모델

그림 3-3는 NIST(National Institute of Science and Technology)에서 제시한 사용자 인터페이스 참조 모델를 나타낸다. 현재 OSF(Open Software Foundation)의 Motif 와 UI(Unix International)의 OpenLook은 이 모델을 기반으로한 GUI의 표준으로 내놓고 있다. 이 들중의 어느 하나가 표준으로 될 지는 모르지만, 현재 까지는 motif가 많이 보급되어 있기때문에 본 과제에서는 이를 이용하여 사용자 인터페이스 부분의 이주를 현재 진행중이다.

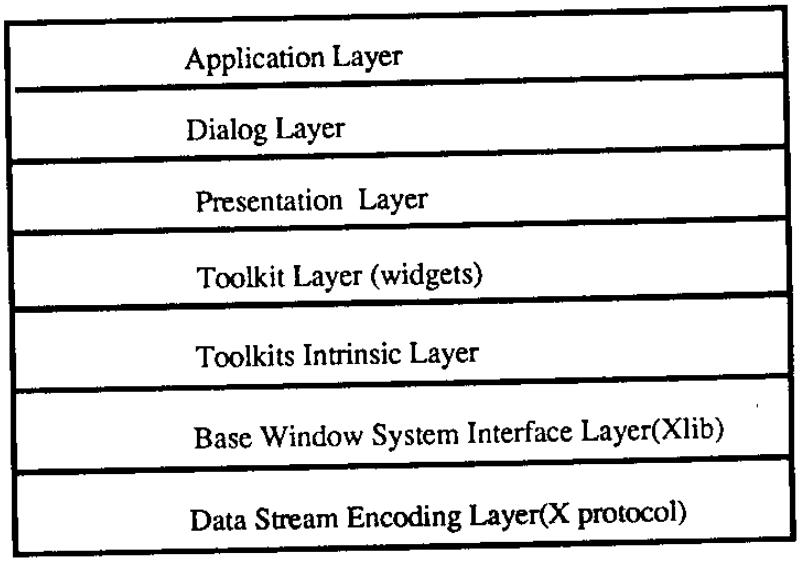


그림 3-3 사용자 접속 참조 모델

3.3 X 윈도우 시스템으로의 이주를 위한 전략

DOS운영체제에서 개발된 Publishing System을 UNIX 운영체제상의 X 윈도우 시스템으로 이주하기 위해서 다음과 같이 프로그램을 크게 두가지로 구분한다.

- 표준 C언어와 라이브러리로만 작성된 소스 프로그램
- 그래픽 및 X 윈도우 시스템과 관련되는 부분

C언어는 높은 이식성과 포터블한 성질을 가진 언어로서, 표준 C언어로 작성된 프로그램은 거의 수정없이 UNIX운영체제가 실행되고 있는 어느 기계위에서나 실행가능하다. 따라서 DOS운영체제에서 표준 C언어와 라이브러리로만 작성된 프로그램은 소스프로그램의 별다른 수정없이 곧바로 이주가능하다.

그러나 그래픽 및 X윈도우와 관련되는 프로그램의 이주는 절대적으로 X의 그래픽 특성 및 메카니즘에 따라야 하며, 현재 DOS상에서 개발된 Publishing 시스템의 전부를 변경해야한다.

3.4 Document Editor

DOS 운영체제에서 개발된 Publishing System을 Unix운영체제로 이식하는 절차에 대해서 기술한다. 특히 그래픽 사용자 인터페이스를 제공하는 X와 Motif를 중심으로본 시스템을 재구성하였기 때문에 이들과의 관계를 구체적으로 상술한다.

3.4.1. 시스템구성

X 윈도우 시스템은 각각의 윈도우를 객체 지향적으로 관리하기 때문에 시스템의 구성요소 즉 기능별 모듈들이 각기 독립적인 객체로서 존재할수있다. 따라서 각 모듈상의 Button, List , Menu등의 Window구성은 비교적 간단한 반면 Resource의 inheritance나 callback routine의 사용은 세심한 주의가 요구된다.

전체 시스템 구성을 Motif 상의 Widget 구조로서 나타내면 그림 3-4과 같다.

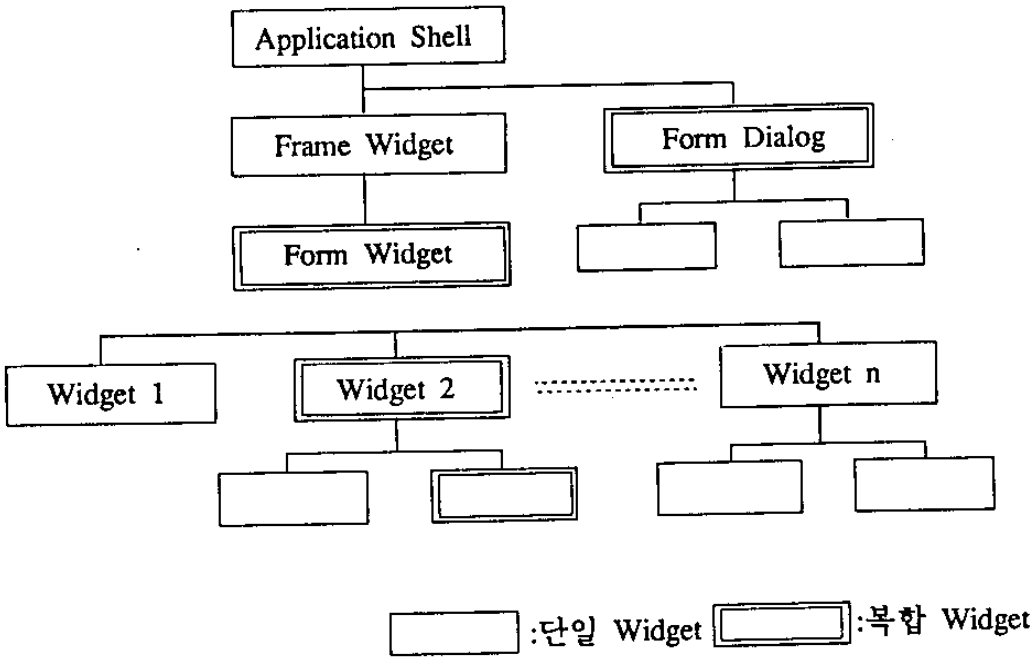


그림 3-4. Widget 구조로 본 시스템 구

본 시스템은 하나의 Application Shell 하에서 각 기능별 모듈을 단일 Widget 또는 복합 Widget으로 구성하였다. 여기서 단일 Widget은 Child Widget을 갖지 못하는 Widget을 뜻하며 복합 Widget은 하나 이상의 Child Widget을 가질수 있는 Widget을

말한다. Form dialog는 원칙적으로 Application Shell과는 다른 dialog shell을 갖는 Widget조이나 Application Shell에서 관리될수 있는 예외적인 Widge이다.

Frame Widget은 단지 Form Widget의 경계를 확인시켜주는 Widget으로 보조적 기능의 단일 Widget이다. Form Widget은 어떤 윈도우가 하나 이상의 Child를 갖고 이들이 서로 다르게 배치되고자 할때 사용할수 있는 layout 전용 Widget이다. 이 Widget을 사용함으로써 사샐상의 시스템구성이 가능해진다.

Widget 1은 단일 Widget으로 예를들면 pulldown menu를 갖지않는 push button widget(garget) 또는 cascade button widgetgarget)이다. Widget 2와 widget n은 복합 Widget으로서 그 구성은 그림 3-4과 같은 것이 대표적이다. 예를들면 label widget 과 list widget 또는 DrawingArea widget 등이 있다. 그림 3-5는 본 시스템을 시작시켰을 때의 시작 화면을 그림 3-4에 관련하여 나타낸 것이다.

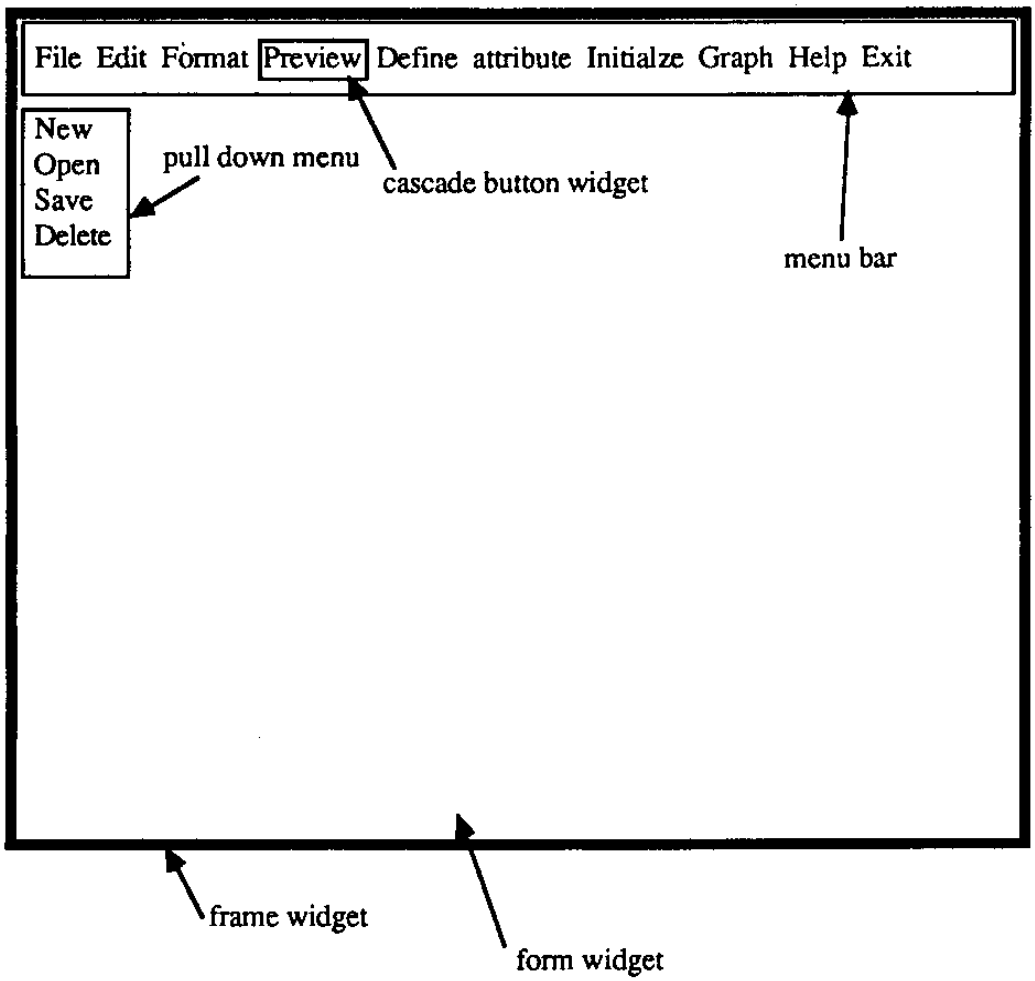


그림 2-2. 시스템 시작 화면 구성

본 시스템은 문서를 작성하는 문서 작성 및 편집부, 출력하는 Format부, 작성된 문서를 SoftCopy 해 보는 Preview부 , 그리고 graphic 요소를 새성하는 graphic editor부로 구성되고 , 이를 위해 그림 3-5와 같은 화면 구성을 갖는다. 시작 화면상의 menu는 문서작성 및 편집을 위한 menu를 주로 배치시켰고, Preview를 위한 menu와 graphic editor에 대한 menu 는 각각의 form dialog widget상에 배치하는 것으로 했다. 자세한 것은 3.5절 에서 상술하기로 한다.

전체 구성을 위해 Frame widget을 parent로한 Form widget을 생성하고, child로서 복합 widget구조인 MenuBar widget을 생성한다. Menubar 상의 menu들은 단일 또는 복합 Widget의 button widget들로 pulldown menu를 갖는 것과 갖지않는 cascade button widget로 구성된다. File, Define Attribute 는 pulldown menu 를 갖고 Preview, Graph 는 dialog Shell 상에서 자신의 menu를 가지며 그외의 것은 단일 Widget으로 작동된다.

MenuBar아래의 빈 공간은 현재로써는 아무런 widget 으로도 구성되지 않았으나 button 이 선택되었을때 해당 작업이 이루어 지는 작업영역으로 확보된 영역이다. 따라서 Drawing Area Widget 이나 Scroll Bar 와 같은 Widget 으로 구성될 수 있다. 이하 각각의 모듈을 설명한다.

3.4.2. 문서 작성 및 편집 (Document Editor)

그림 3-5 의 Menu 중 Preview 와 Graph를 제외한 menu들이 문서 작성 및 편집에 관련하고 있기 때문에 그들에 대한 설명도 함께 하기로 한다.

3.4.2.1 화일부

화일 Button은 New, Open, Save, Delete 하는 4개의 Push Button을 갖는 Cascade Button 으로 구현 하였다, 즉 pulldown menu를 갖는 복합 Widget 이다. 각각의 menu를 pulldown menu 형태로 묶기 위해서는 menu 용 pane을 생성해야 하고 (이것도 하나의 Widget 임) 이 Widget 을 parent 로 하여 각각의 pulldown menu item 들이 push button widget 이라는 단일 Widget 형태로 존재하게 된다.

여기서는 문서 화일의 생성, 로드, 저장, 삭제 등 화일 관리에 관련된 기능을 수행하는 모듈이다. 따라서 화일 리스트로 부터 로드, 삭제 할 화일을 지정한다든

지 새로운 화일명을 입력하는 것이 주요 내용이 된다. 이를 위해 FileSelectionBoxDialog Widget 과 PromptDialog Widget를 사용하였다.

이 Widget 들은 Dialog Shell 을 갖는 Widget 들이기 때문에 Form Widget 의 배치 규정을 따르지 않고 application shell로써 구현되는 window 로 부터 일정 간격을 유지하면서 자동 배치된다.

텍스트 편집을 위해 작업영역으로써 DrawingArea Widget 과 수평, 수직 ScrollBar Widget 을 생성한다. 텍스트폰트는 18x11 pixel 크기의 "CourB18" 폰트를 사용하였고 라인당 126 문자를 수용할 수 있는 논리 화면을 설정하였다. 이 Widget 은 화일 open, new buton 이 active 될때 manage 되어 편집 menu 와 함께 디스플레이 된다.

3.4.2.2 에디트 부

문서를 생성하거나 기존의 문서를 로드하여 논리 오브젝트와 레이아웃 오브젝트를 설정하고 문자 또는 라인당의 삭제 및 삽입과 같은 문서 편집 전반에 관한 기능을 포함하고 있다.

본 시스템은 문를 논리 구조와 레이아웃 조로 표현하고 있는 ISO 의 문서 구조에 관한 국제 표준에 준해서 문서를 작성하도록 하였다. 따라서 논리구조인 장, 절, 각주등을 논리 오브젝트로, 레이아웃 구조인 페이지, 프레임, 블록등을 레이아웃 오브젝트로하여 문서를 표현하게 하였다.

에디트 명령에는 다음과 같은 것이 있다.

. Set Logical Attributes : 논리 오브젝트를 설정한다. 논리 오브젝트는 다시 interparagraph 명령군과 Tag 명령군으로 구분하여 설정하는데 interparagraph 명령은 paragraph 으로 취급되는 문서 내용에만 적용될 수 있는 명령이고 Tag 명령은 한 이상의 paragraph 에 적용될 수 있는 논리 명칭으로써 보다 high level 적으로 문서를 표현 할 수 있게 한다,.

. Show Logical Attributes : Paragraph 의 경계, 설정된 interparagraph 및 tag 명령의 적용 범위를 표시해 줌으로써 대화적으로 문서의 논리 오브젝트를 설정할 수 있게 한다.

. Set Layout Attributes: 문서의 레이아웃을 설정한다. 여러가지 레이아웃 스타일

화일로 부터 레이 아웃을 확인해보고 원하는 레이아웃 화일을 지정함으로써 문서 내용의 레이아웃이 결정된다. 또 특정 프레임으로 처리 되어야 할 문서 부분을 지정할 수 있게 함으로써 특정 프레임 구조를 갖는 레이아웃 스타일과 문서를 보다 쉽게 적용시킬 수 있다.

. Show Layout Attributes : 현재 시스템에 등록된 레이아웃 스타일 화일을 확인하거나 이미 설정된 레이아웃 구조를 확인해 볼 수 있게 함으로써 레이아웃 설정 및 변경의 자유도를 높힌다. 또 특정 프레임의 내용을 확인해 볼 수 있어 이것의 수정 및 변경을 가능하게 한다.

. Tag Range Indicatior : Tag 명령의 설정 범위를 전체적으로 확인 할 수 있게 한다. 따라서 부분적인 확인을 위한 Show logical attribute와는 달리 디스플레이 되고 있는 문서 내용에 대한 tag 명령의 범위를 한 눈에 알아 볼 수 있게 함으로써 효율적으로 tag 명령을 설정 할 수 있다.

상기의 에디트 명령은 pulldown menu 상의 Push Button 으로 구현되고 하나의 button 이 선택되면 관련 submenu를 디스플레이 하기 위한 Widget 이 생성되어 manage 되는데 이들을 텍스트와 함께 디스플레이 됨으로써 대화적으로 텍스트를 편집할 수 있게 한다. submenu 를 갖는 Widget 은 다음과 같은 특성을 공통적으로 갖는다.

.텍스트를 디스플레이 하기 위한DrawingArea Widget 의 왼편에 배치

.각각의 submenu를 갖기 위해서는 복합 Widget 이 어야 한다.

(1) Set Logical Attributes

Logical Attributes 를 설정하기 위한 Widget은 하나의 DrawinArea Widget 내에 두개의 list widget 으로 구성 하였다. 즉 interparagraph list widget 과 ag list widget 으로 구성하였다.

어떤 interparagraph명령은 자신의 Sheet를 가지고 있는데 underline, font, figure,footnote,maxfont,Running footnote, chapter와 section numbering 명령이 여기에 속한다. 각 Sheet는 주로 Dawing Area widget로 구현되며 Toggle button 또는 pushbutton

widget으로 구성된다.

(2) Show Logical Attributes

Drawing Area Widdget 내에 3개의 push button widget과 하나의 Drawing Area Widget을 생성하여 manage시킨다. 3개의 push button은 각각 paragraph boundary, interparagraph command, tag name show를 위한 것이고 명령 list를 디스플레이하기 위해 Drawing Area widget를 설정하였다. 논리 속성 확인시에도 설정시와 마찬가지로 interparagraph 명령 확인 sheet를 가지게 되는 데 구현방법은 설정시의 sheet 와 동일하다.

(3) Set Layout Attributes

시스템에서 제공하고 있는 레이아웃 스타일 화일을 지정하기 위해 FileSelectionBox widget을 생성하고 이것을 통해 선택된 화일의 내용 즉 기본 프레임,특정 프레임,분할된 프레임을 보여주기 위한 각각의 widget도 생성하여 manage되어야 한다. 후자의경우는 Define-Attributes 모듈에서 상술하기로 한다.

(4) Show Layout Attributes

3개의 push button을 만들어 시스템에서 제공하는 레이아웃 스타일 화일, 현재 설정된 레이아웃 스타일, 특정 프레임의 구조를 확인해본다. 사용되는 Widget은 Define-attributes 모듈에서 생성된 Widget이므로 자세한 것은 후술한다.

(5) Tag Range Indication

텍스트 상에 설정된 tag name을 텍스트가 디스플레이된 Drawing Area Widget에 표시해주기만 하므로 새로운 Widget을 생성할 필요가 없다.

3.4.2.3 Format 부

작성된 문서를 LBP로 출력하기 위하여 포맷팅하는 부분이므로 특별한 widget 필요치 않다. 그러나 최종 출력 presentation medium을 선택해야 하므로 이를 위한 widget이 생성되어 출력 페이지의 크기가 정해져야 한다. 여기서 요구되는 widget 역시 Define-Attributes 모듈에서 설명하기로 한다.

3.4.2.4. Define-Attribute부

논리 오브젝트와 레이아웃 스타일을 정의하고 변경할수 있도록 한다.

따라서 사용자는 시스템에서 제공하는 논리 오브젝트와 레이아웃 오브젝트외에도 사용자 정의의 논리, 레이아웃 오브젝트를 사용할수 있다.

(1) Logical Attribute Define

시스템 논리 속성 정의는 모든 사용자에게 정의 자격을 부여할수 없기때문에 특정인 에게 속성 정의 자격을 부여하고 이를 점검해야 한다. 이를 위해 암호(password)입력을 위한 widget을 Drawing Area Widget으로 구현하였다. 논리 속성정의에 필요한 widget에는

i)논리 속성명을 디스플레이 하기 위한 widget,

ii) 논리 속성명을 입력하는widget,

iii) 논리 속성의 요소가 되는 기본 명령 세트를 위한 widget이 있는데 i)은 list widget으로 ii)는 Prompt Dialog widget 으로 iii)은 togle button과 push button widget을 포함하는 drawing area widget으로 구현하였다. 또 속성 정의의 계속 여부를 확인 하기 위해 question dialog widget도 요구 된다.

(2) Logical Attributes Modification

다양한 논리 속성을 가질수 있도록 하기위해 그리고 논리 속성에 대한 선택의 폭을 높히기 위해 시스템 정의의 논리 속성 뿐만 아니라 이미 정의된 사용자 정의의 논리 속성들도 새로운 내용으로 변경될수 있다. 이를 위해서는

i) 시스템 정의 존리 속성 리스트,

ii) 사용자 정의 논리 속성 리스트

iii) 변경을 위한 기본 명령 세트

iv) 새로운 속성명 정의 widget이 필요하다.

i),ii)는 수직 scrollbar를 갖는 list widget으로 iii)은 toggle button 과 push button widget을 포함하는 drawing area widget으로 iv) 입력 string을 받을 수 있는 drawing area widget으로 구현 하였다. 또 논리 속성 변경의 계속 여부를 확인하기 위한 guestion dialog widget도 요구된다.

(3) Logical Attributes Define & Modification

레이아웃 스타일 화일을 정의하고 변경하는 부분은 처리 기능이 다소 차이가 있을 뿐 widget구성은 동일하므로 같이 설명하고자 한다. 단, 레이아웃 속성 정의 때

는 레이아웃 대상이 되는 표현 매체을 선택 해야 한다는 것, 속성 변경때는 레이아웃 스타일 화일을 선택해야 한다는 것이 다르다. 전자는drawing area widget에 toggle button을 포함시켜 구현하였고 후자는 FileSelectionBox widget으로 부터 변경하고자 하는 화일을 선택할 수 있다.

레이아웃 정의 및 변경에 필요한 widget에는

i) 정의 또는 변경을 위한 명령들을 포함하는 widget

ii) 프레임 속성 설정 widget

iii) 프레임 선택 widget

iv) 새로운 레이아웃 화일 명 입력 widget

v) 프레임들을 drawing하는 작업영역 widget이 있다.

i)은 하나 이상의 push button과 하나의 drawing area widget을 포함하는 복합 widget으로 구현할 수 있는데 여기서drawing area widget은 진행중인 프레임 drawing에 관한 정보를 디스플레이 하기 위한 것이고 push button은 레이아웃 정의 또는 변경 명령에 해당된다.

ii), iv)는 push button을 통해 입력 스트링을 표시할 수 있는 Drawing Area Widget으로 구현하였고, iii)은 push button만을 갖는 widget으로 구현하였다.

v)는 mouse를 이용하여 테이아웃을 구성하는 프레임들을 그려보일 수 있도록 Drawing Area Widget을 사용해야 한다. 이 Widget 은 다른 것과 달리 graphic context가 Exclusive-OR특성을 가져야 한다. 이는 프레임 조정시 사용하는 Rubber band를 위해 필수적이기 때문이다.

5. Initialization부

페이지, 장, 절의 시작 번호를 설정하기 위한 것으로, 각각의 값을 입력하고 번호 체제를 선택할 수 있어야 한다. 이를 위해 입력값을 받을 수 있는 push button과 toggle button widget을 포함하는 Drawing Area Widget으로 구현하였다.

6. Help부

문서작성을 돕기 위해 각 모듈의 기능적 특성, 이용방법을 설명하기 위한

것이다. 이것은 미리 작성된 모듈별 설명 화일(text file)을 단지 디스플레이 하여 사용자로 하여금 도움을 얻을 수 있도록 하면 되기 때문에

i)menu bar,

ii)scroll bar

iii)working area(text draw area)를 위한 widget으로 구현하였다.

i)은 모듈의 이름을 선택할 수 있게 하고 ii)는 논리화면 크기보다 긴 화일에 대해 scrolling을 가능하게 하고 iii)은 선택된 모듈의 도움말을 디스플레이 하기 위한 것이다. 이러한 구성은 각각을 독립적으로 구성시킬 수도 있으나 Motif 에서 제공하는 convenience function을 통해 i), ii), iii)을 하나의 widget(즉, text widget)으로 구성시켰다.

3.5 Preview 부

3.5.1 X윈도우 시스템과의 관계

Preview부는 formatter에서 처리된 문서의 결과를 하드카피 하기전에 디스플레이상으로 확인하는 부분으로 Normal View와 Reduced View로 구성되는데 Normal View는 실제 문서의 모습과 거의 비슷하게 디스플레이상에 보이고, Reduced View는 전체 모습을 보기 위해 Greeking box로 표현한다.

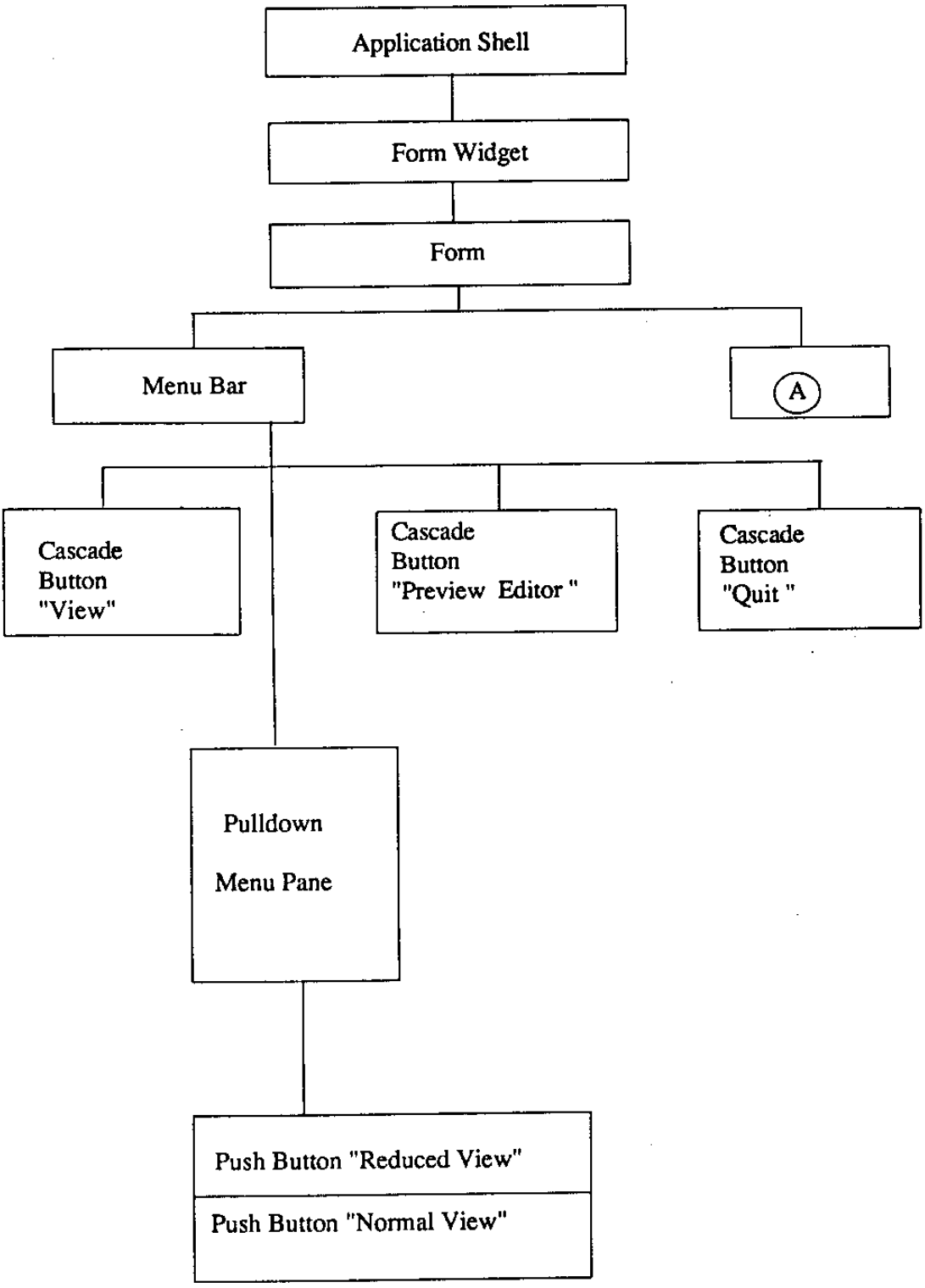
Preview를 X윈도우로 이주 하기위해서는 다음과 같은 PC의 디스플레이의 하드웨어에 의존적인 부분의 변경이 있어야한다.

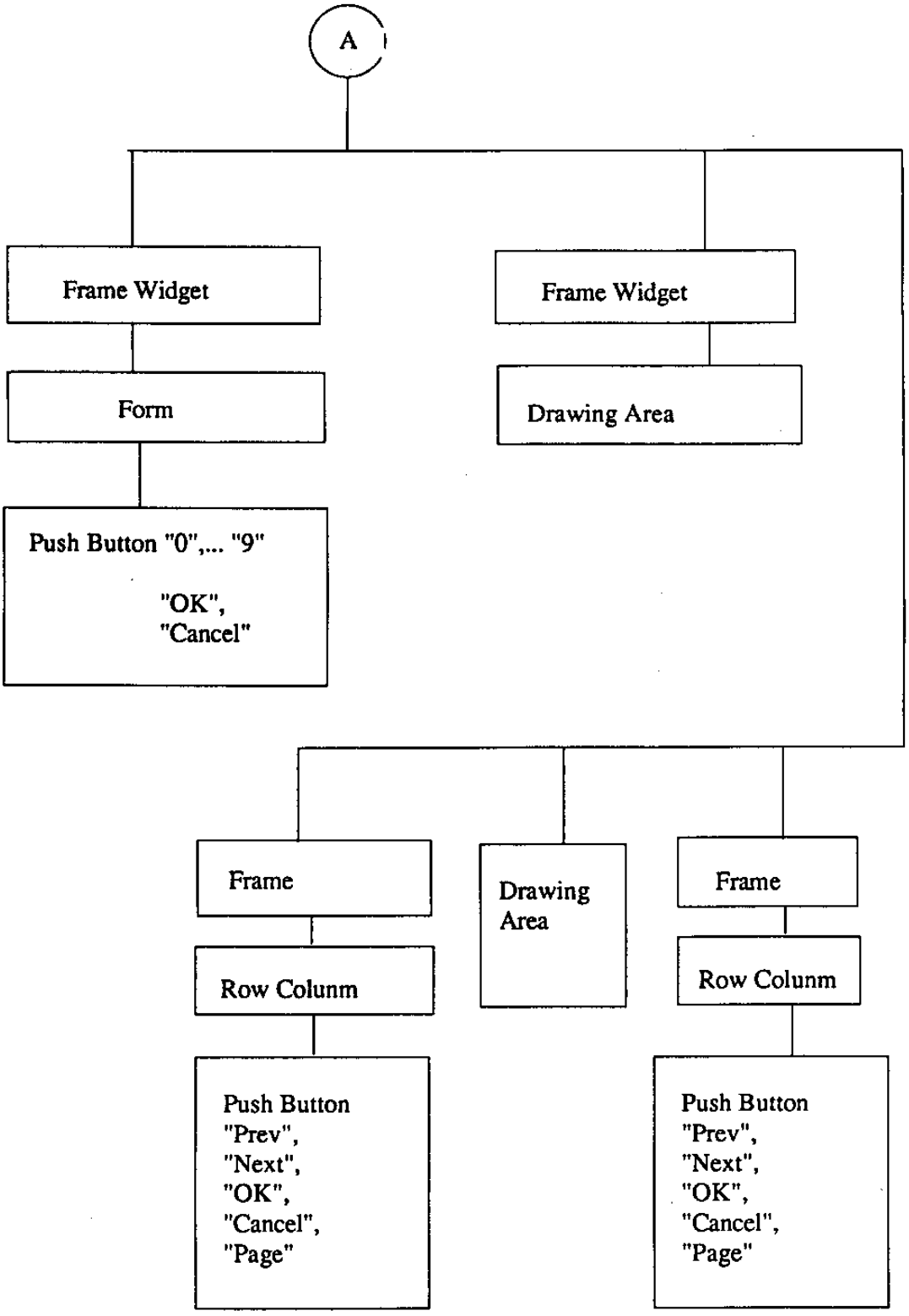
DOS환경에서 개발된 Publishing System은 그래픽 처리를 위해 PC의 그래픽보드에 맞는 device-driver를 설계하였고, 이러한 디스플레이의 하드웨어에 대해 device-dependent한 부분과 device-independent한 부분 사이의 인터페이스 역활을 하는 상위레벨의 그래픽 루틴을 독자적으로 구현하였다. 한편 X윈도우는 client-server모델로, client는 디스플레이 하드웨어에 완전한 독립성을가지고 있어 응용프로그램은 XLIB를 통해 그래픽 처리를 할 수 있으며, server는 이러한 응용 프로그램의 요구에 따라 디스플레이 하드웨어에 의존적인 부분을 처리하도록 되어 있다. 여기서 server는 DOS환경에서 device-driver와 같고 client는 그래픽 루틴과 같다. 따라서 그래픽 처리는 X윈도우 그래픽 라이브러리를 이용하여 처리할 수 있으므로 그래픽 루틴은 더이상 필요치 않다.

Normal View에서 각각의 한글을 발생시키기 위해서 자,모음의 그래픽 프리미티브를 동적합성 규칙에 의해 합성하여 가상적인 논리화면에 매핑하는 방식을 취하고 있다. 이때 발생되는 글자는 비트맵 형태로 저장되고, 한 페이지 전부를 디스플레이에 보이는데에는 디스플레이의 해상도에 의존한다. 해상도에 의존하는 문제를 문제를 해결하기 위해 디폴트 크기의 윈도우를 일단 만든 다음 사용자가 윈도우를 resize하거나 scroll bar를 통해 조정하는 방식을 취해야 한다.

Reduced View하나의 윈도우에 X윈도우의 기능을 이용 각각의 character를 box 모양으로 greeking한다.

3.5.2 Preview Editor의 구성도




A
Frame Widget
Form
Push Button "0",... "9"
"OK",
"Cancel"
Frame Widget
Drawing Area
Frame
Row Colunm
Push Button
"Prev",
"Next",
"OK",
"Cancel",
"Page"
Drawing
Area
Frame
Row Colunm
Push Button
"Prev",
"Next",
"OK",
"Cancel",
"Page"

제 4 장 WYSIWYG alike한 레이아웃 처리 및 Preview Editor

본 장은 Document Editor에서 작성한 구조화된 입력문서정보에 따라 레이아웃을 행하며 토큰 단위로 텍스트 및 문서 지정명령을 읽고 실행하여, 레이아웃 및 포맷팅을 행하는 레이아웃 처리부에 대해서 논한다.

아울러, 2차년도에 개발한 Batch Mode 처리방식에서 WYSIWYG alike한 방식으로 구조를 변경하고 처리방식을 개선한 데이타 구조 및 운영에 대해서 상세히 기술한다.

4.1 Batch Mode 레이아웃 처리

Document Editor에서 논리적 구조로 만들어진 문서의 내용에 모양과 크기, 그리고 general makeup을 행하여 레이아웃 처리 결과는 physical structure즉, formatted file로 만든다. 문서의 실제모습(physical structure) 은 다른 layout을 정의 함으로서 달라지지만, 내용은 똑같이 남아있고 presentation-independent한 정보이다.

논리적 구조로 만들어진 결과는 레이아웃 처리를 위해 넘겨질때 적절히 텍스트속에 embedded되고, formatter는 이 화일에서 하나씩 토큰 단위로 읽어들여 명령을 해석하고 레이아웃을 실행하며 문서내용에 위치와 모양, 크기를 정한다. 이때 레이아웃 처리된 결과로 생성된 출력화일 (formatted file)은 preview와 preview editor, reduced view, normal view, 문서의 hard copy를 위해 사용된다.

formatting과정은 pass1과 pass2의 두 단계를 거쳐 실행된다. Pass1단계는, 입력 텍스트와 레이아웃을 행하기 위한 각종 명령이 혼용된 입력 스트링으로 부터 caption및 특정프레임만을 미리 처리하는 레이아웃 처리의 초기 단계이다.

이 단계에서는 레이아웃을 실행하지는 않으면서 pass2에서 효율적으로 레이아웃을 처리할 수 있도록 미리 전처리를 하는것으로, 본 연구에서는 특정 프레임의 텍스트를 기본 프레임의 텍스트보다 우선적으로 입력하게 되어 있는바 이러한 이유때문이며, caption을 미리 처리하는 것은 pass2단계에서 그림영역을 자동할당 할

때 마다 이 그림과 관계되는 caption string을 처리하게 되면 현재 레이아웃 처리중인 다른 텍스트 정보와 같은 level에서 처리되므로 정보의 관리가 되어 caption만을 미리 처리한다.

pass2단계는, 레이아웃을 개시하여 각종 입력 정보에 따라 텍스트와 그림을 배치하는 실질적인 레이아웃 처리단계이다. 특히 그림중심으로 볼때 현재 그림의 위치가 본문의 내용과 밀접한 관계(context sensitive)를 가지므로 본문의 내용과 함께 그림명령을 사용하여 시스템이 그림영역을 자동적으로 할 당하는 단계이다.

위 단계를 거쳐 만들어진 문서는 본 연구 보고서에 서 기술하는 preview 기능을 통하여 출력 결과를 확인 할 수 있고, 만약 사용자의 의도대로 되지 않았을 경우에는 피드백하여 편집부에서 입력 스트림의 부분적인 수정 및 재편집을 한다.

또한 pass2단계를 거쳐 자동적으로 발생된 그림영역은 그림과 관계되는 본문의 내용과 떨어져서 레이아웃 될 수 있기 때문에, 이러한 그림영역은 원래의 그림에 관계되는 위치로 조정하여야 한다. 이렇게 재편집 및 그림조정을 거친 문서는 다시 레이아웃 처리 될 수 있다.

그림 4-1 은 위 과정의 전체적인 구성도이다.

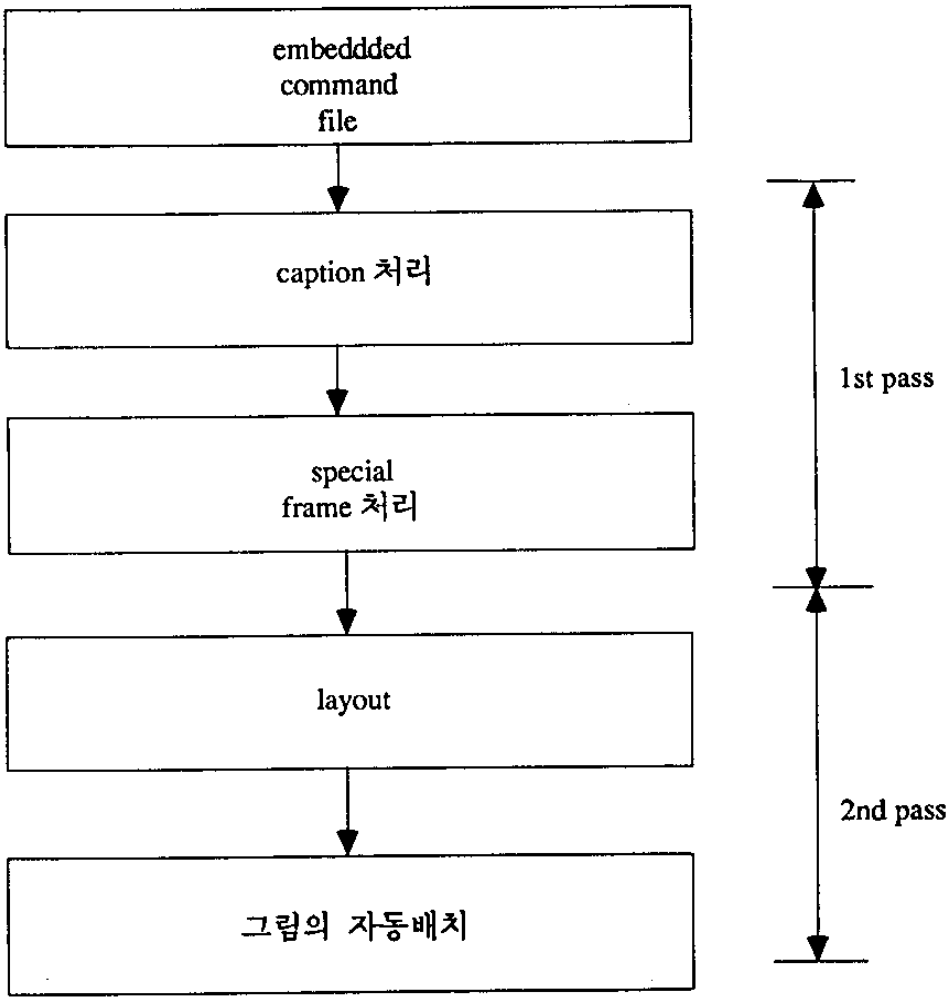


그림 4-1. 레이아웃 처리과정

레이아웃 처리시 문서지정 명령의 적용되는 범위를 효율적으로 관리하기 위해서 stack 데이타 구조를 이용한다. page전체에 적용되는 명령, frame, block, inter-paragraph에 적용되는 명령을 stack구조를 이용하면 각 문서내용에 지정되는 명령의 유효범위를 관리하는데 편리하다. 그림 4-2는 stack데이타 구조속의 각 element는 나타낸다.

```
typedef struct {
      int start_x;    /* x position of frame */
      int start_y;    /* y position of frame*/
      int width; /* width of frame */
```

```
        int height;     /* height of frame */
        int left_m;     /* left margin */
        int right_m;    /* right margin */
        int bottom_m;   /* bottom margin */
        int top_m; /* top margin */
        int line_sp;    /* line space */
        int para_sp;    /* paragraph space */
        int indent_x;   /* indent x wdith */
        int indent_y;   /* indent y width */
        int indent_z;   /* indent z width */
        int f_mode;     /*  */
        int f_type;     /* typeface */
        int f_xsize;    /* font's x size */
        int f_ysize;    /* font's y size */
        int style; /* justify,centering, left/right aligned */
        int w_mode;     /* garo, sero */
        int attribute;  /* text, raster, geometric, caption */
        int bcolor;     /* background color */
} STK;
```

그림 4-2 레이아웃 처리를 위한 스택 데이타 구조

Batch Mode 방식의 레이아웃 처리는 1page부터 끝 page까지 한번에 처리를 하기 때문에, preview editor에서 그림영역의 이동, 크기 조정을 할때 자주 레이아웃 처리가 되어야하고 각 페이지에 해당하는 출력화일 (Fomatted File)을 만들어야 하므로 비효율적일 수가 있다.

따라서 현재 대두되고 있는 WYSIWYG 방식에서와 같이 layout 처리된 모습을 user에게 빠르게 보여주는 것을 고려할때, 현재의 batch mode방식의 데이타 구조 변경과 내부처리방식을 수정하면 WYSIWYG alike한 구현이 가능하고 시스템의 성능을 높일 수 있으며, 기존의 Batch Mode특성을 가질 수 있다.

그림 4-3 은 Batch Mode 출판 시스템의 전체 구성도[7, 8]를 나타낸다.

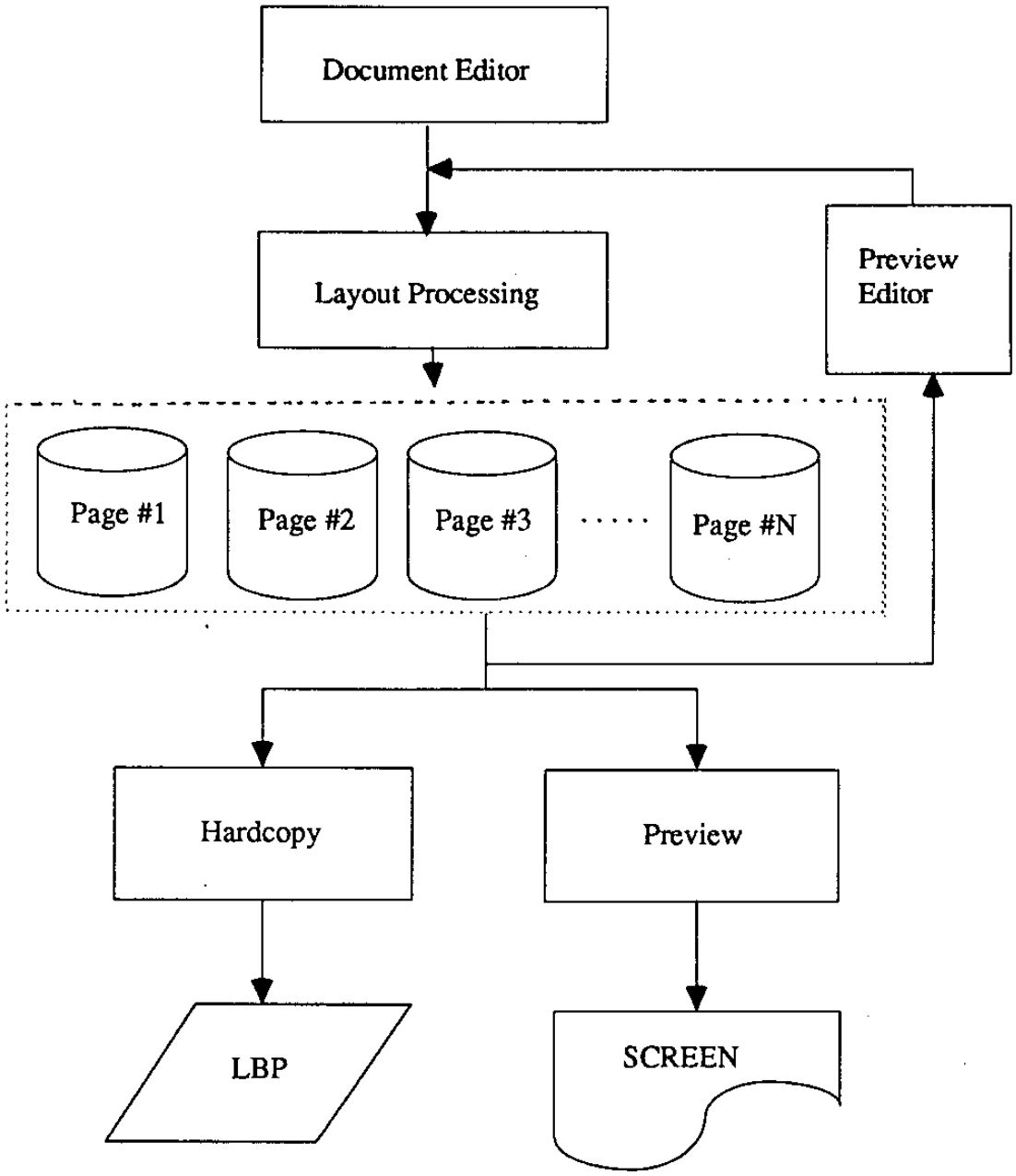


그림 4-3. Batch Mode 전자출판 시스템의 전체 구성도

4.2 WYSIWYG alike한 구조의 레이아웃 처리

Batch Mode 방식의 레이아웃 처리는 1page부터 문서의 끝 page까지 처리되고, 그 결과는 formatted file 형태의 출력 file로 매 페이지마다 생성된다.

앞 절에서도 언급한것과 같이 preview editor에서 그림영역의 조정없이 단지 문서의 출력만을 할때는 Batch Mode의 고유의 장점을 가지지만, 빈번한 그림의 조정과 조정에 따른 결과(layout처리결과)를 보여주기 위해서 매번 1페이지 부터 페이지 끝까지를 레이아웃 처리하고 각 페이지에 대응되는 처리결과를 만들어야 하는 비효율적인 단점을 가지고 있다.

따라서 본 연구에서는 Batch Mode의 처리방식과 WYSIWYG alike한 구조를 가지는 레이아웃 처리방식을 제안한다.

가. 데이타 구조

```
typedef struct {
     int  page; /* page number */
     long offset; /* embedded source file에서 해당 page의
                     file pointer */
     int  is_haveFF; /* formatted file의 유무 */
     STK  *stack_info; /* 해당 page의 stack 정보 */
} stack_DS;
```

여기서 offset은 embedded source file 을 가지고 레이아웃 처리를 하면서 구분되는 page의 시작 file pointer를 나타낸다. is haveFF는 해당 페이지의 fomatted file의 유무를 나타내는데, normal view 또는 reduced view, preview editor를 위해 해당 페이지만 생기고, is haveFF 상태비트의 역활은 연속적으로 같은 페이지가 지정되었을 때 연속적인 formatted file의 생성을 방지하기 위해 사용된다. STK는 앞절에서 기술한 stack구조를 나타내며, 어떤 page의 레이아웃처리가 끝났을때 다음 페이지에 영향을 주는 명령을 보관 하며, page,frame,block,interparagraph level의 명령을 모두 보관한다.

나. 데이타 구조의 운영

Document Editor에서 작성된 embedded commnad file은 실제 layout을 처리하기 전까지는 개념이 전혀 없다. 즉, 한번 레이아웃을 처리하기전까지는 page경계 및 문서의 내용에 적용되는 명령을 알수없다.

Batch Mode형식의 레이아웃 처리는 이러한 page경계 및 다른 페이지에 미치는 명령의 범위가 1페이지 부터 처리되면서 내부 스택을 이용하여 쉽게 관리할 수 있고, 매 페이지마다 처리된 결과를 fomatted file로 만든다.

그러나 preview editor서 그림영역의 조정시, 예를들면, 3페이지의 그림을 좀더 크게 하였고, 2 페이지의 그림을 1페이지로 옮겼을때, 3페이지 이상의 페이지는 처리될 필요가 없지만, Batch Mode레이아웃 처리 모델 자체가 1페이지 부터 끝까지 처리해야 하기 때문에 3 페이지 이상의 레이아웃 처리와 동시에 출력 formatted file을 매번 만들어야 하는 단점이 있다.

formatted file을 만드는 과정은 publishing system의 performance에 상당한 영향을 미치며, 만약 1000 페이지 이상의 문서가 있다면 위의 예를볼때 상당히 비효율적인 문제가 된다.

WYSIWYG alike한 layout처리는 사용자가 지정한 페이지만 또는 그림영역의 이동에 따른 영향 받는 페이지들을 처리하는 방식인데, 그림영역 이동에 따른 모든 페이지들을 끝까지 처리하는 방식이 아니라 사용자가 지정한 페이지 까지만 처리하는 방식이다. 차후 사용자가 그 뒤페이지를 지정하게 되면 처리된 마지막 페이지 부터 방금 지정한 페이지 까지 처리한다. 그림 4-4 는 WYSIWYG alike한 layout처리를 위한 데이타 구조의 운영예를 나타낸다.

레이아웃처리를 위해 임의 의 특정 페이지가 지정되었을때, 앞절에서 기술한것과 같이 레이아웃 처리의 처음이라면 pass1단계를 처리한 다음, 1페이지부터 해당 페이지까지 레이아웃 처리를 실행하게 된다.

처음 1 페이지를 처리하면서 논리정보 데이타 구조에서 1페이지의 처음을 나타내기 위한 embedded command file pointer로 0000L 로 offset에 지정되고, 만약 1 페이지를 reduced view,normal view page로 지정했다면 1 페이지에 해당되는 formatted

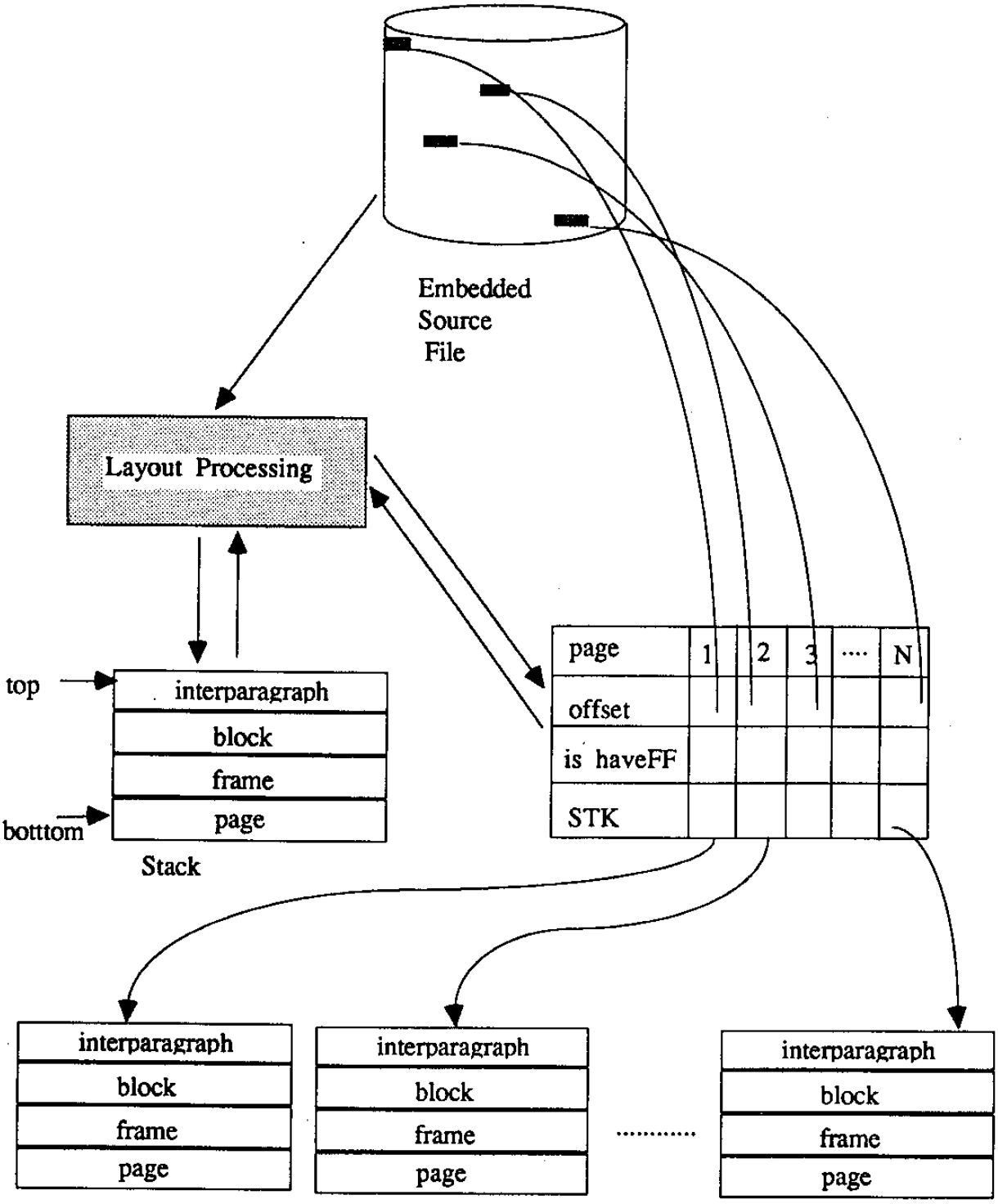


그림 4-4. WYSIWYG alike한 layout 처리를 위한
데이타구조의 운영 예

file이 생성되면서 is haveFF 상태 bit에 1이 설정된다.

문서에서 1페이지와 2페이지, 그 이상의 페이지 경계는 이미 설정된 레이아웃 내에 내용이 다 배치되는가의 여부로 결정되지만, 페이지 전체에 적용될 수 있는 명령이 있을 수 있고, 현재 frame,block,interparagraph에 적용되고 그 명령의 유효 범위가 끝나지 않았다면 계속 다음 페이지 에서도 적용되어야 하므로 1 페이지가 끝나면서 stack에 남아있는 정보를 보관하고 있다가 2 페이지 레이아웃 처리 시작 시 이 정보를 이용 계속 레이아웃을 행하게 된다.

물론 이 명령의 유효 범위가 끝나게 되면 stack에서 처리된 정보는 사라지고 새로운 정보가 설정되어 다음 레이아웃 처리를 하게 된다. 마찬가지로 다른 페이지도 이런 방식으로 운영되어 임의의 페이지가 처리될때마다 formatted file 보다는 이러한 논리적인 정보만을 가지게 된다.

만약에 이미 처리된 페이지가 reduced view를 위해 지정되었다면 논리정보 데이타 구조에서 file pointer를 가지고 embedded command file에서 그 위치에서 부터 레이아웃 처리를 시작하고 이와 동시에 STK에 보관되었던 STK정보를 레이아웃 처리를 위한 stack에 설정하게 된다. 단, page, frame, block, interparagraph의 statck element의 x,y,w,h정보를 해당 페이지의 레이아웃에 따라 바뀌게 된다.

4.3 Preview Editor

그림영역의 자동할당 알고리즘[8]은 그림영역 확보 및 위치를 레이아웃 처리시 자동적으로 결정하여 대체적으로 context sensitive한 문제를 해결하여 주지만 극단적인 경우에는 그림에 관계되는 문맥과 상관없는 다른 위치 및 수 페이지 뒤로 밀려 다른 페이지에 할당될 수 가 있다.

Preview Editor는 이러한 그림을 사용자의 의도대로 또는 문맥에 관련되는 위치로 조정할 수 있게 하는 editor이다.

WYSIWYG alike한 레이아웃 처리 방식을 갖는 Preview Editor의 그림 조정방식은 Batch Mode방식의 그림 조정방식과 동일하지만, 레이아웃처리 방식이 다르기 때문에 그림 조정에 따른 효율적인 page관리가 요구된다.

본절에서는 Preview Editor의 그림조정 방법보다는 그림조정에 따른 WYSIWYG alike한 방식에서 레이아웃 처리를 위한 페이지 관리기법에 대해서 기술한다.

4.3.1 Preview Editor에서 Layout처리를 위한 페이지 관리

Batch Mode 방식의 레이아웃 처리는 앞절에서도 언급한 바와같이 1 페이지 부터 문서의 끝 페이지까지 다 처리 한 다음, 각 페이지에 해당하는 문서의 처리 결과를 Formatted File형태로 만든다. 이 출력된 formatted file은 view(Normal/Reduced), Preview Editor에서 해당 페이지의 레이아웃 처리된 실제 모습을 보여주기 위해서 Preview Editor가 시작되기 전에 1 페이지부터 끝 페이지까지 처리된 결과이며, 이 Batch Mode방식은 그림 조정을 끝마치고 조정된결과에 따라 다시 1페이지 부터 reformatting을 하는방식으로 처리된다.

WYSYWYG alike한 방식은 처음부터 레이아웃을 처리하여 Formatted File을 만드는것이 아니라 사용자가 페이지를 지정 했을 경우, 가장 최근에 처리된 다음 페이지 부터 해당 페이지 까지 처리한다.

한편 문서의 내용은 그림 영역이 이동된 위치부터 뒤로 모든 내용이 영향을 받기 때문에 그림 영역조정중에 이 영향 받은 페이지들에 대한 레이아웃 처리를

할 수도 있지만, 본 연구에서는 Preview Editor의 editting 기간중에 그림이 조정될때마다 레이아웃을 행하지는 않으면서 editting을 벗어났을 경우에 레이아웃 처리될 시작 페이지 부터 지정 페이지 까지 처리 하는 방식을 연구하였다.

이러한 처리를 설명하기 위해서 다음과 같이 용어를 정의한다.

SP : 레이아웃 처리될 시작 페이지

SRC : 그림을 조정하고자 하는 source page

DEST : 그림을 조정하고자 하는 destination page

MAX(p1, p2) : p1과 p2를 비교 큰값을 return

MIN(p1,p2) : p1과 p2를 비교 작은값을 return

EM : Preview Editor에서 레이아웃 처리될 기준이 되는 페이지

초기에 SP와 EM은 SP = EM 이며 normal view, reduced view또는 Preview Editor에 의해서 새로운 값으로 변경될 수 있다. Preview Editor에서는 그림 조정을 위해서 두 페이지를 지정 하는데 (그림 조정을 하고자 하는 페이지 SRC와 지정된 그림의 위치를 다른 페이지에 옮기거나 그림의 영역을 조정하고자 하는 DEST 페이지) 지정된 두 페이지중 작은 페이지를 기준으로 그 다음 모든 페이지는 영향을 받게 된다.

따라서 다음과 같은 관계가 형성된다.

$$SP = MIN(SRC,DEST)$$

위의 식은 그림을 옮기는 한 시점에서의 관계이고 연속적으로 그림을 조정할 때 마찬가지로 위 식을 이용 새로운 SP'이 생기게 된다. 새로 생긴 SP' 와 바로 전에 생긴 SP는 다시 비교되어 새로운 SP가 생기게 된다.

$$SP = MIN(SP,SP')$$

그림 조정이 연속적으로 계속 될때 마다 위 두 식이 수행되고, 여기서 새로 설정된 SP는 Preview Editor을 벗어났을때 layout이 시작될 page를 가리킨다.

한편 SRC와 DEST, 두 페이지 가 지정되어질때 EM과 비교되어 레이아웃이

처리되는지 안되는지의 여부를 나타내게 되는데, 지정된 페이지가 EM보다 크게 되면 EM+1 페이지부터 지정된 페이지까지 레이아웃이 처리된다.

위의 관계는 다음과 같이 표현될 수 있다. SRC와 DEST 두 페이지 다 적용하기 위한 변수를 PG라 하고 새로운 EM을 EM'이라한다.

EM' = MAX(EM,PG)

if(EM < PG) {

EM' = PG

EM+1 페이지 부터 PG까지 레이아웃 처리

}

else if(EM >= PG) {

EM' = EM

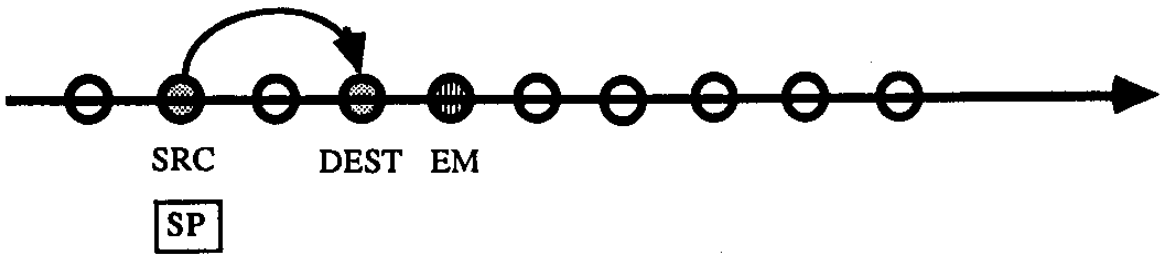
레이아웃 처리는 하지 않는다

}

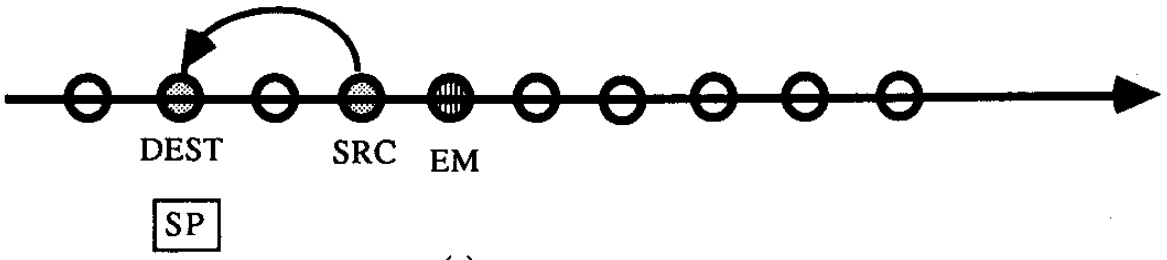
else

그림 4-5 는 위의 관계를 설명하기 위한 예다.

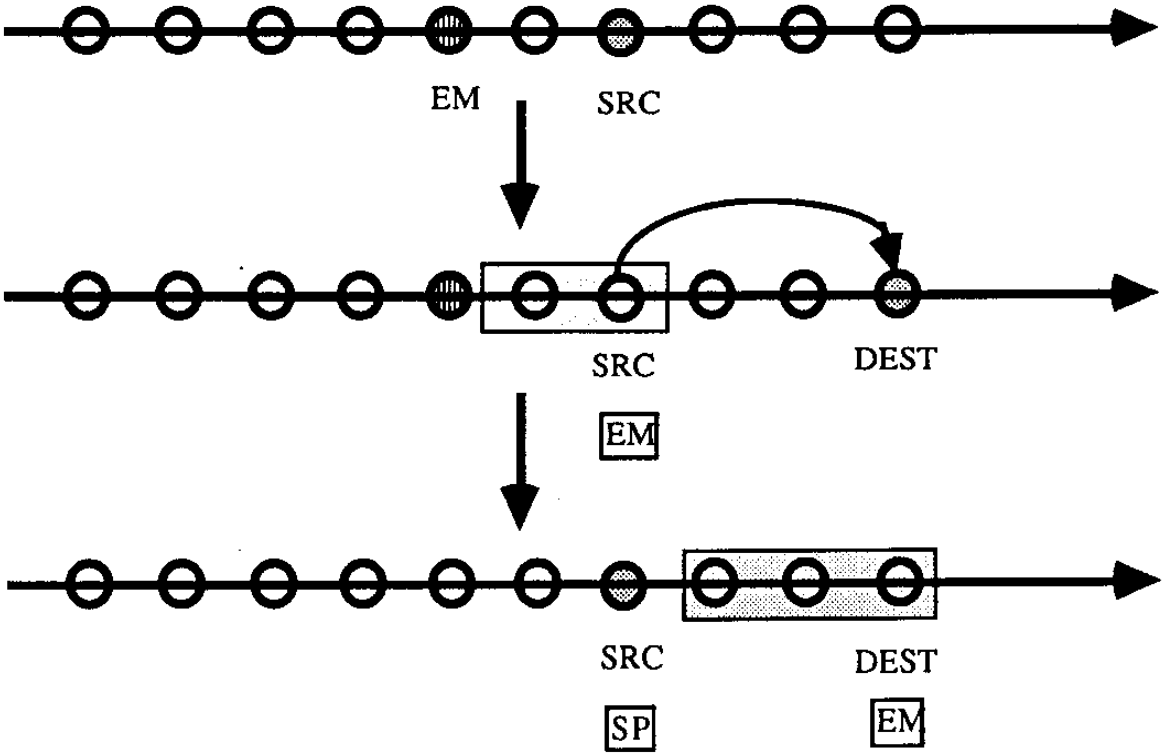
(a)



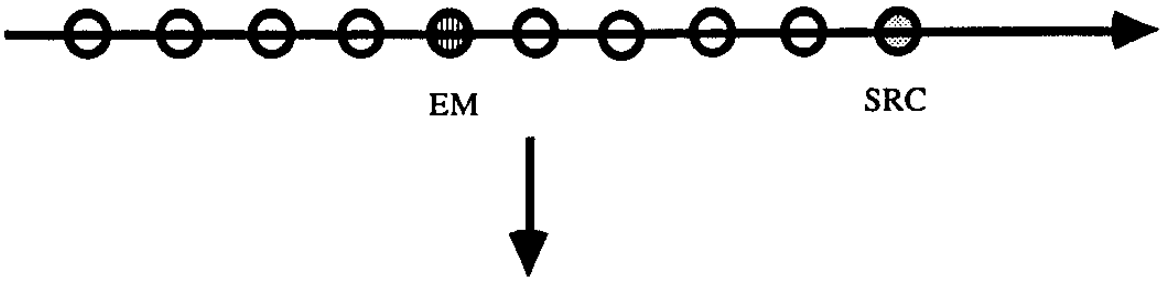
(b)

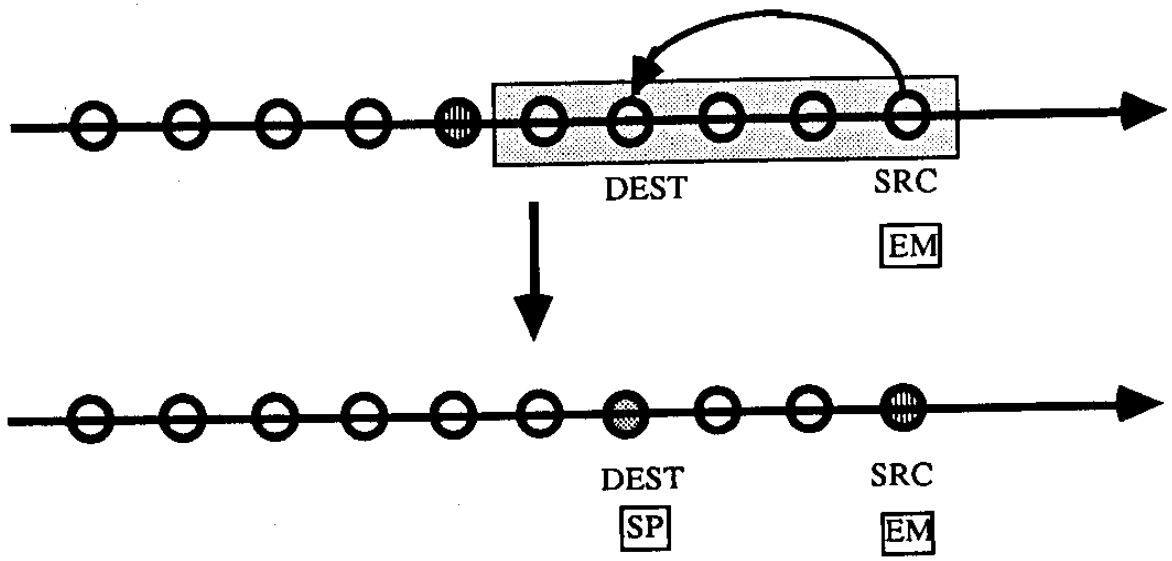


(c)

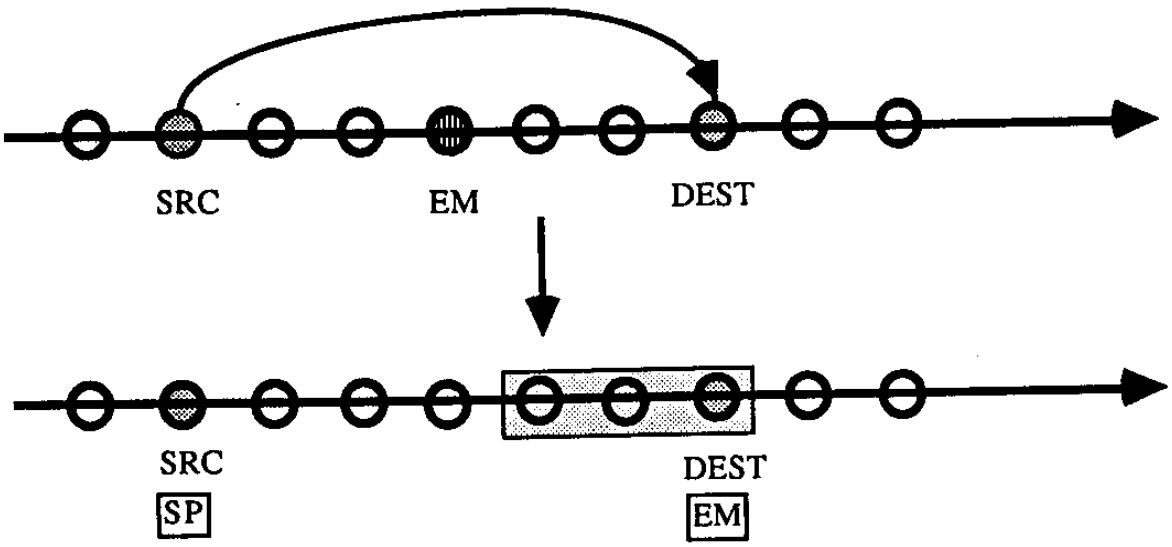


(d)

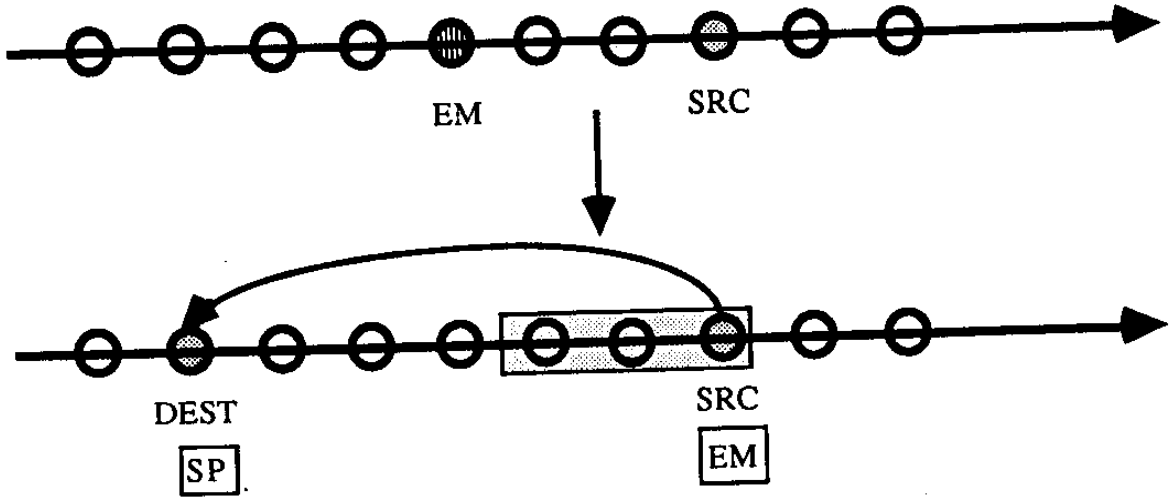




(e)



(f)



(g)

그림 4-5. preview editor에서의 그림 조정

제 5 장 실험 및 결과

5.1 실험

본 시스템의 전체구성도를 그림 5-1에 보인다. 본 실험을 위한 처리 기종은 Sun Micro system사의 SUN 3/50, 3/80를 host로 하는 시스템에 NCD16 X Terminal을 네트워크로 연결하여 사용하였다. OS는 UNIX BSD 4.3 version을 이용하였고 X 윈도우 시스템은 X11R3, Motif는 version 1.0.A를 이용하여 실험 하였다.
프로그램 언어는 C언어를 사용하였다.

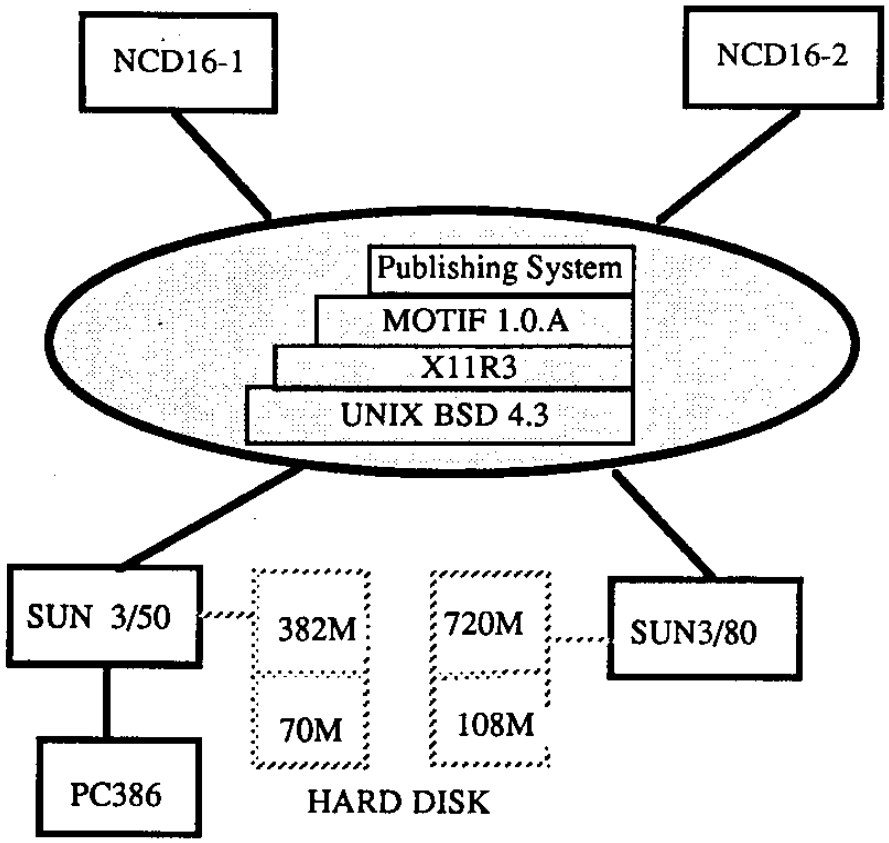


그림 5-1. 시스템 전체 구성도

5.2 실험결과

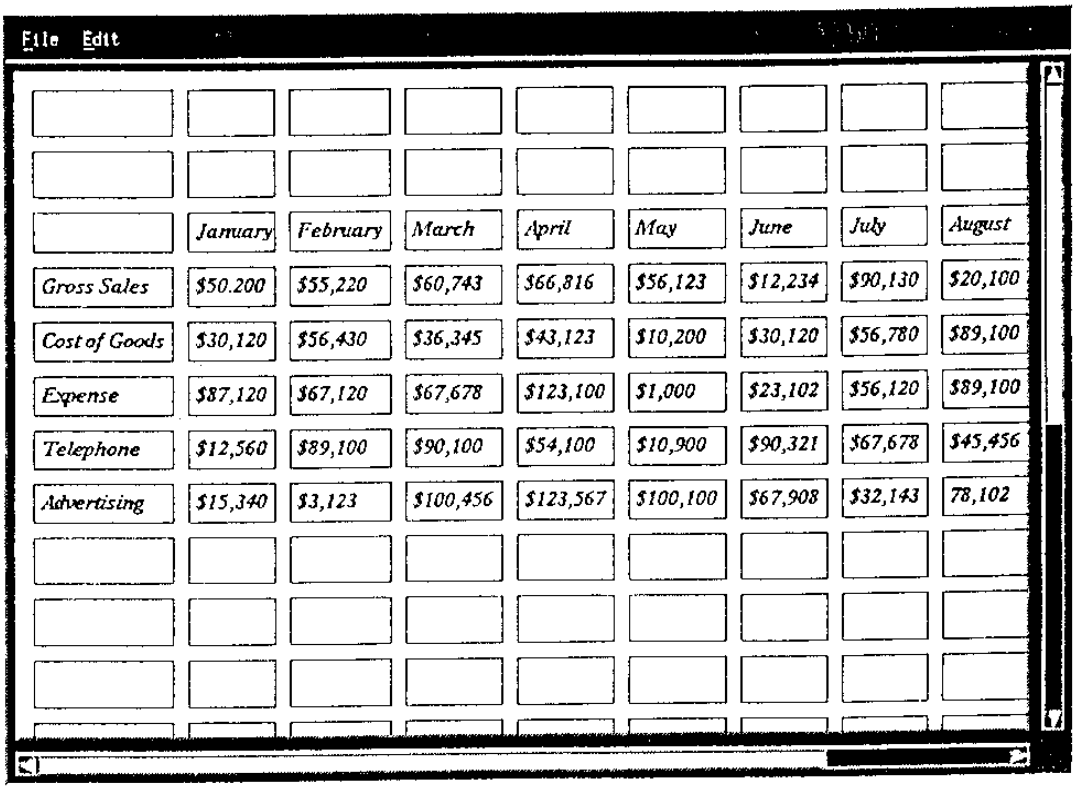


그림 6-1. Spread Sheet

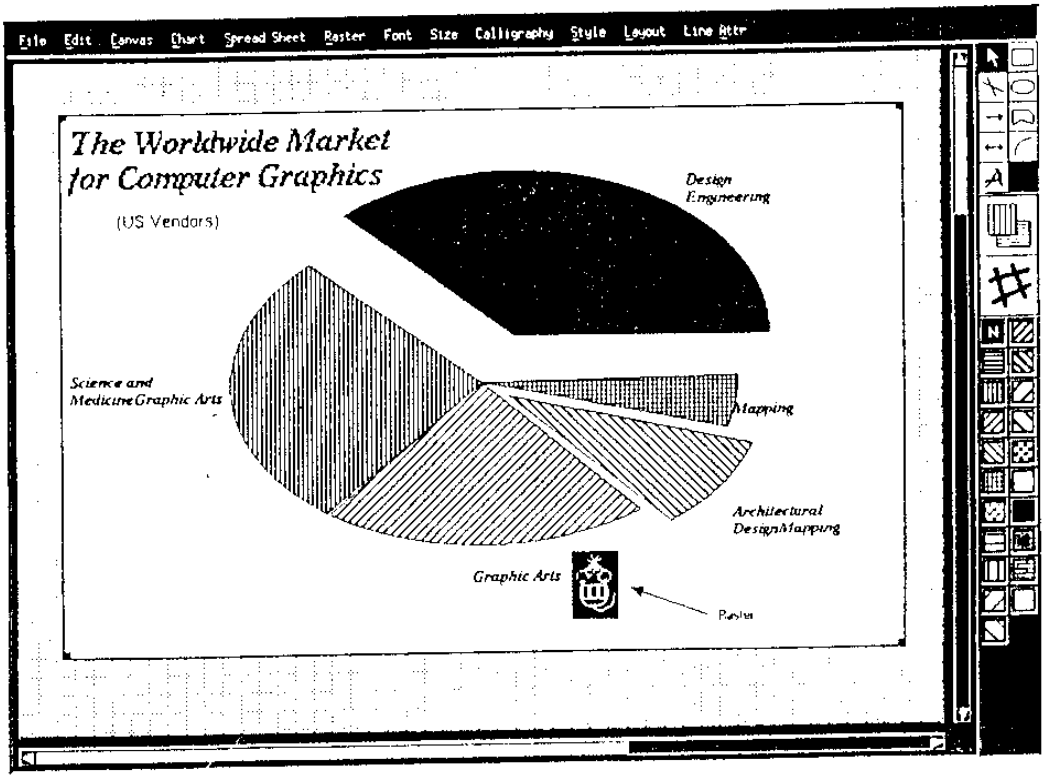


그림 6-2. 2-D Pie Chart의 예

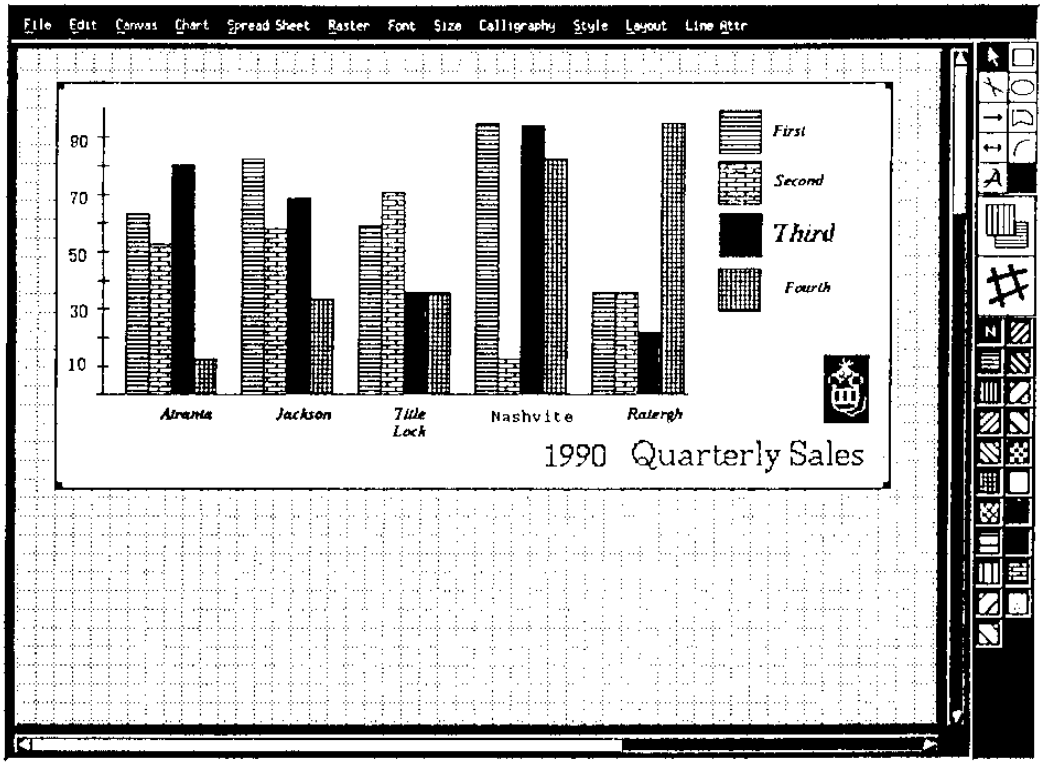


그림 6-3. 3-D vertical bar chart의 예

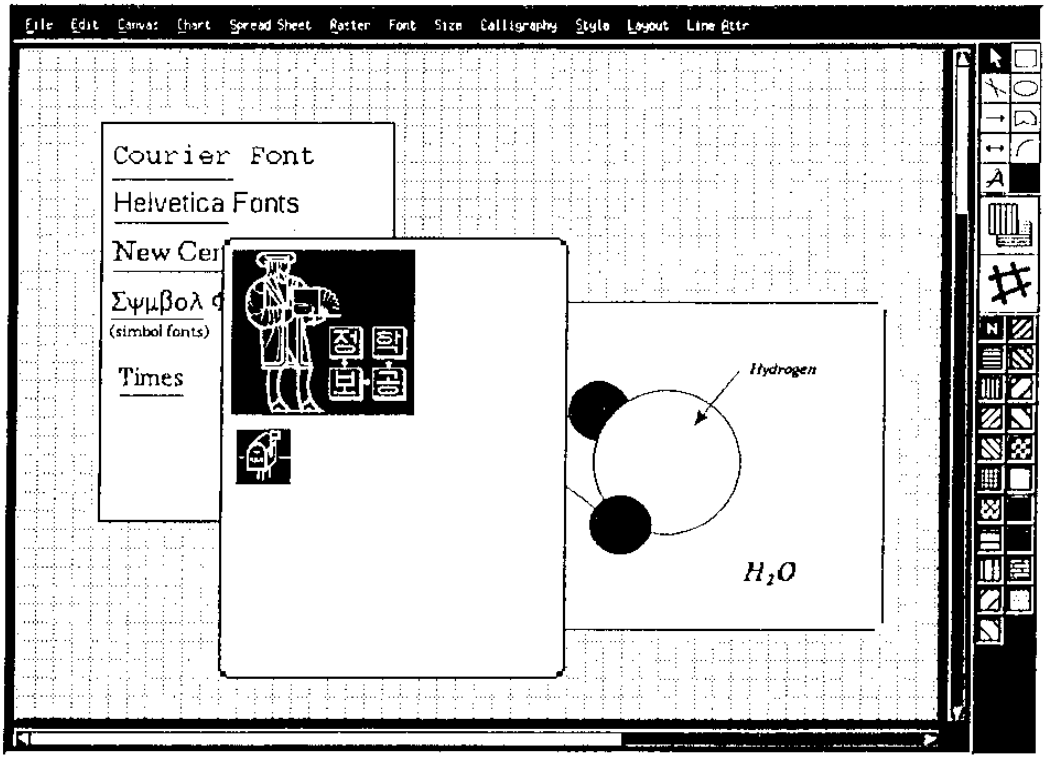


그림 6-4 . Multiple canvas 사용 예

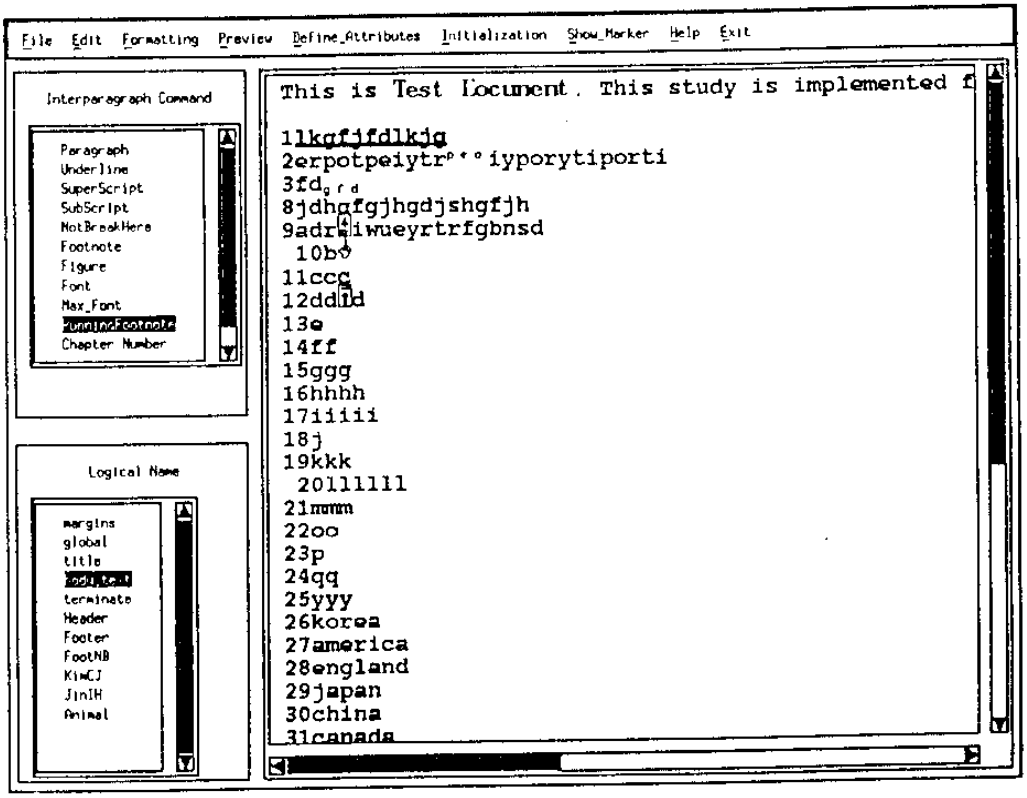


그림 6-5. 논리적 명칭 설정 예

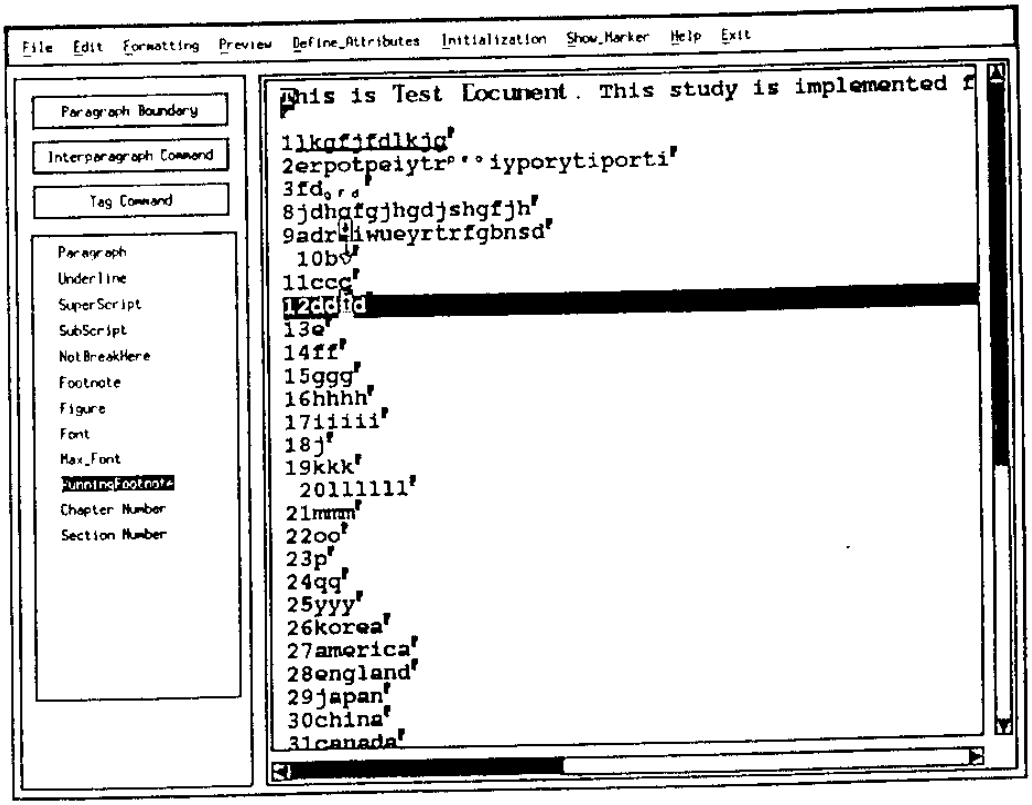


그림 6-6. 논리적 명칭 show 예

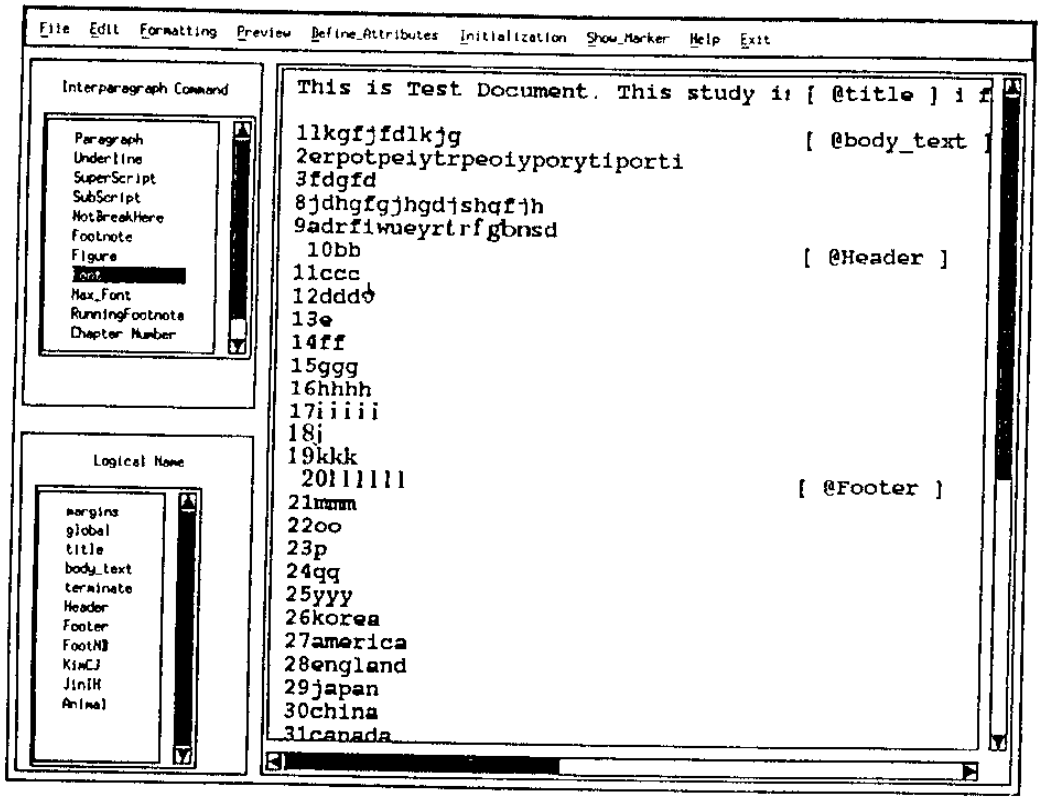


그림 6-7. tag 범위 표시

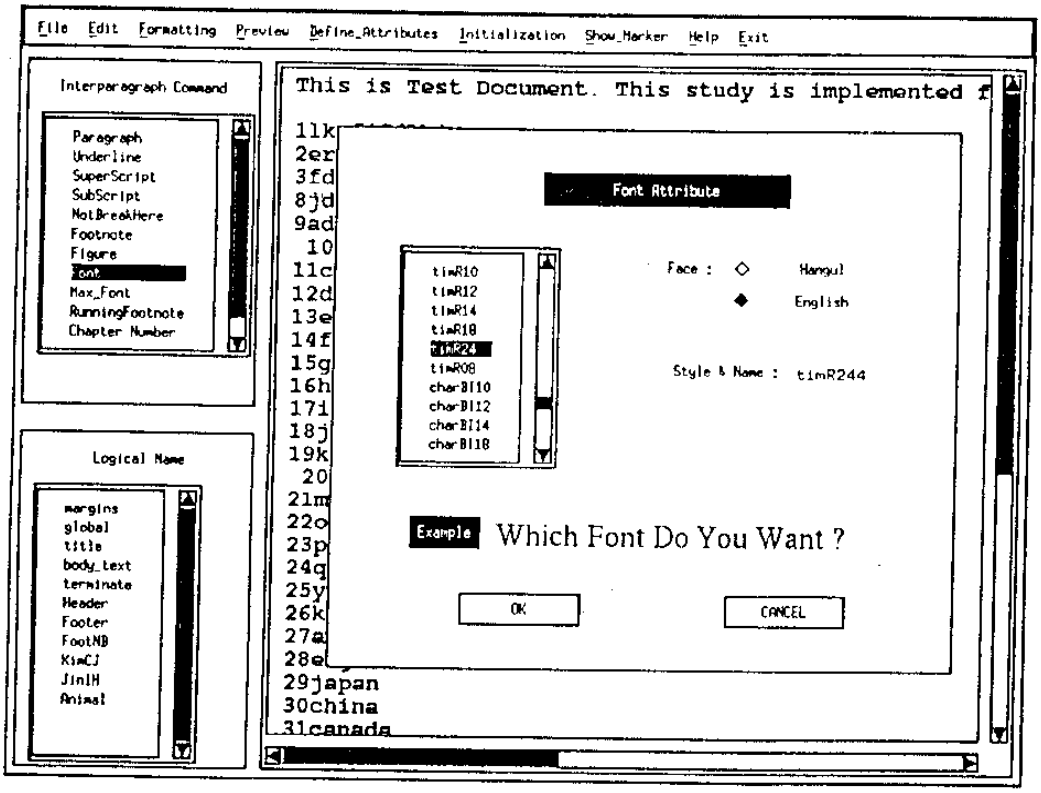


그림 6-8. 논리적 명칭 속성 정의 예

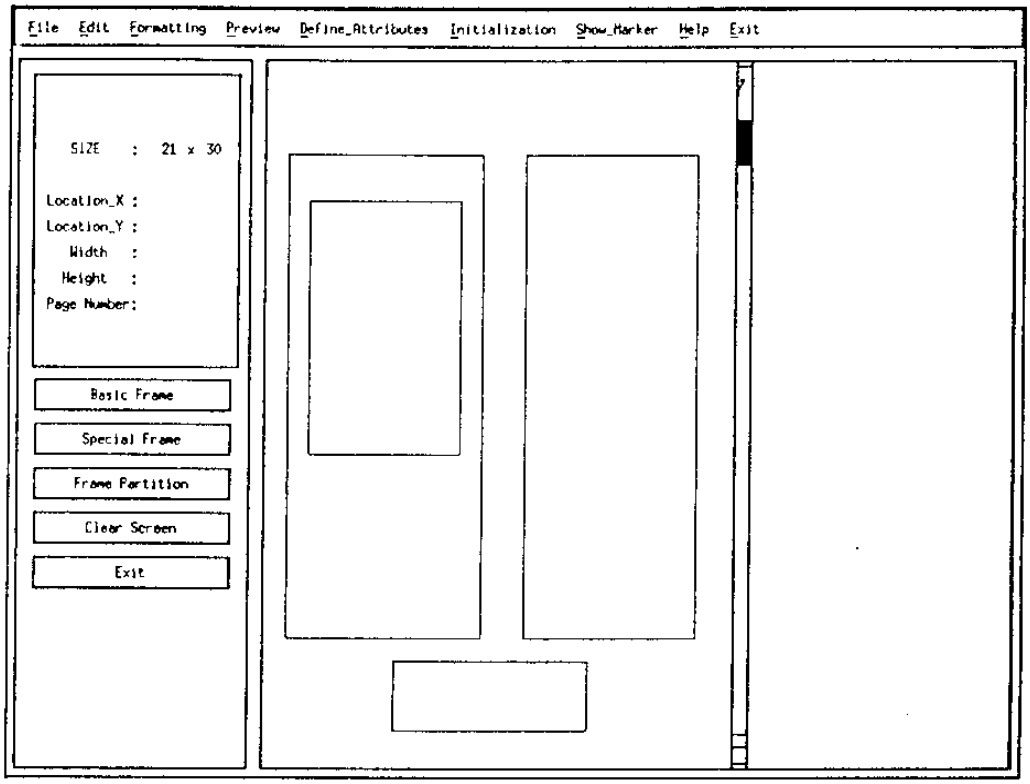


그림 6-9. 레이아웃 설정 예

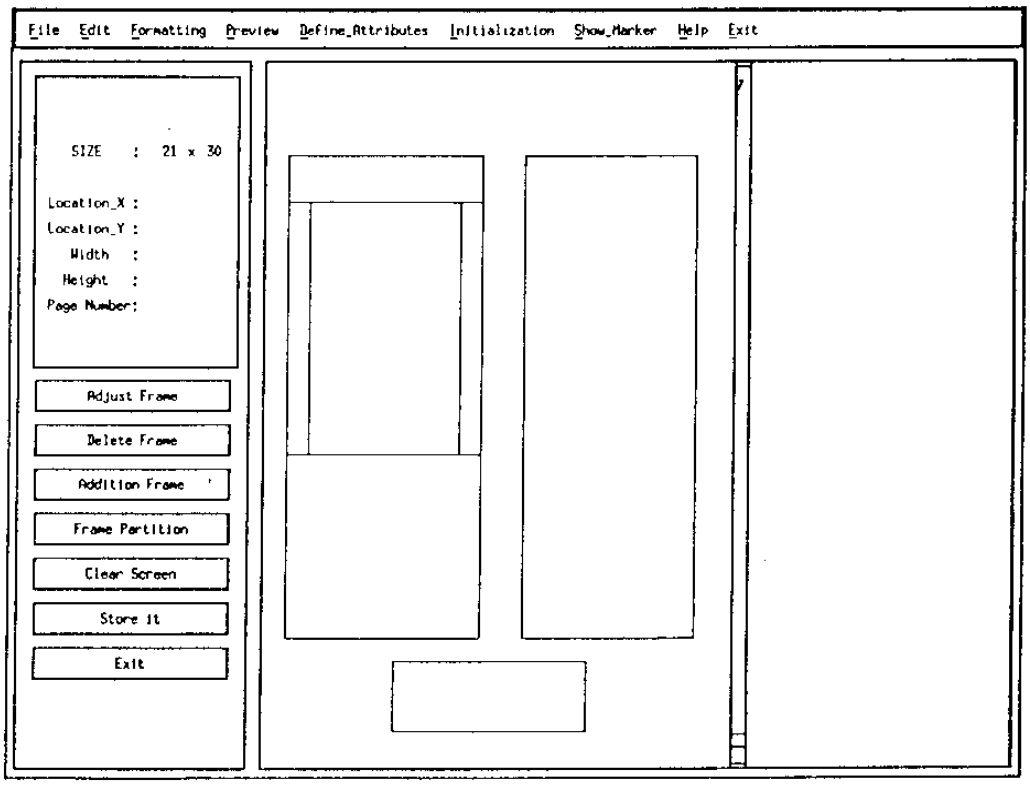


그림 6-10. 레이아웃 show 예

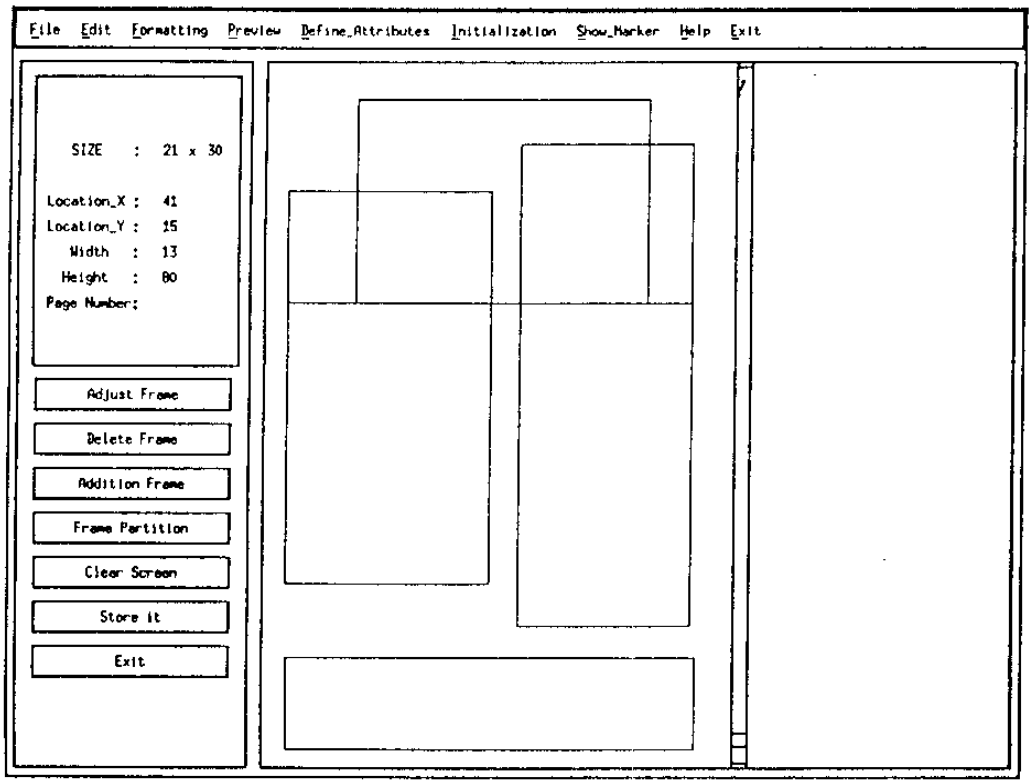


그림 6-11. 레이아웃 변경 예

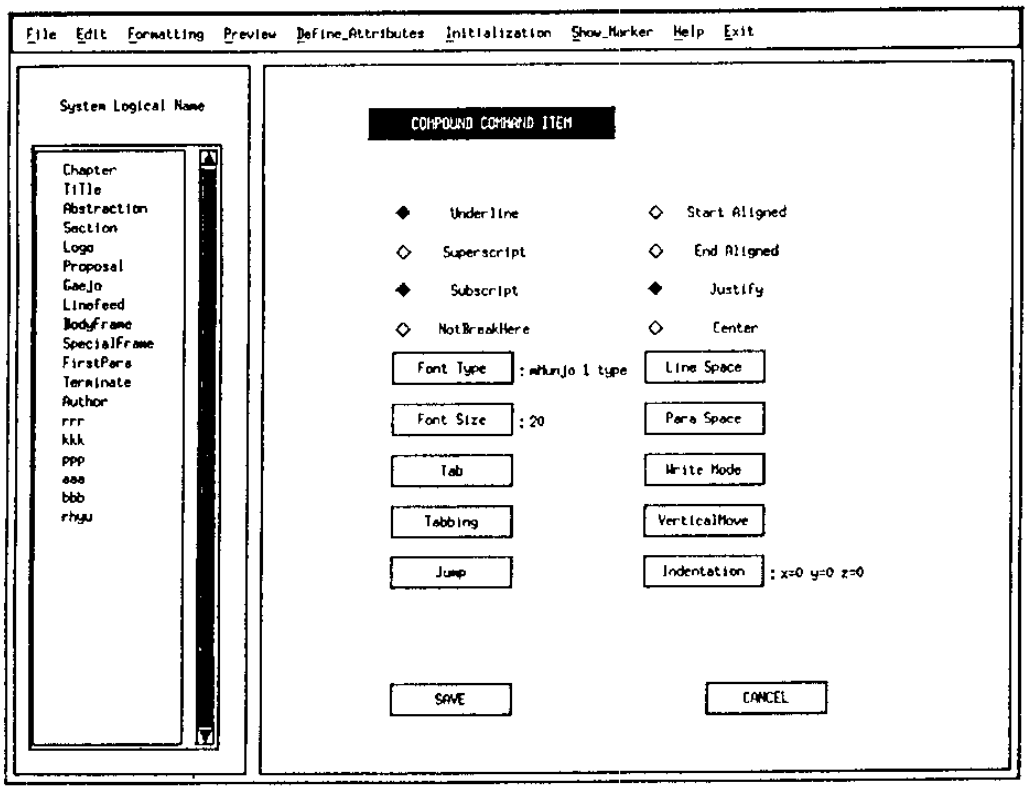


그림 6-12. 속성 설정 예

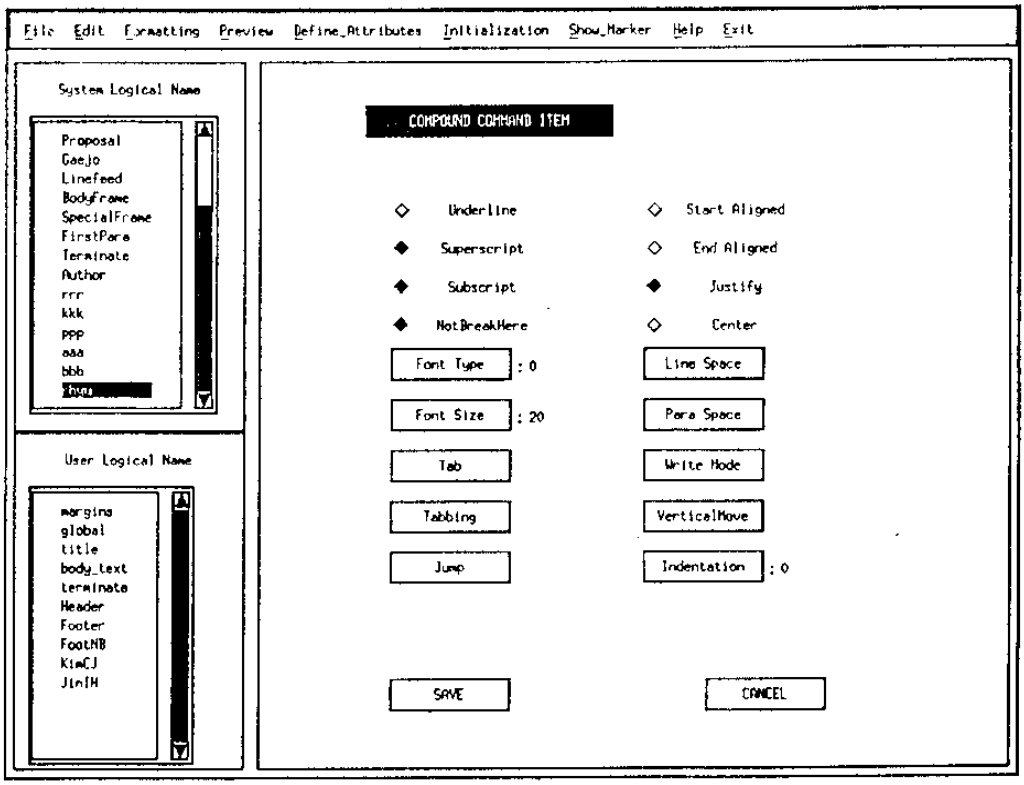

File Edit Formatting Preview Define_Attributes Initialization Show_Marker Help Exit
System Logical Name
Proposal
Gaejo
Linefeed
BodyFrame
SpecialFrame
FirstPara
Terminate
Author
rrr
kkk
PPP
aaa
bbb
User Logical Name
margins
global
title
body_text
terminate
Header
Footer
FootNB
KimCJ
JinIH
COMPOUND COMMAND ITEM
Underline
Superscript
Subscript
NotBreakHere
Start Aligned
End Aligned
Justify
Center
Font Type : 0
Font Size : 20
Tab
Tabbing
Jump
Line Space
Para Space
Write Mode
VerticalMove
Indentation : 0
SAVE
CANCEL

그림 6-13. 속성 변경 예

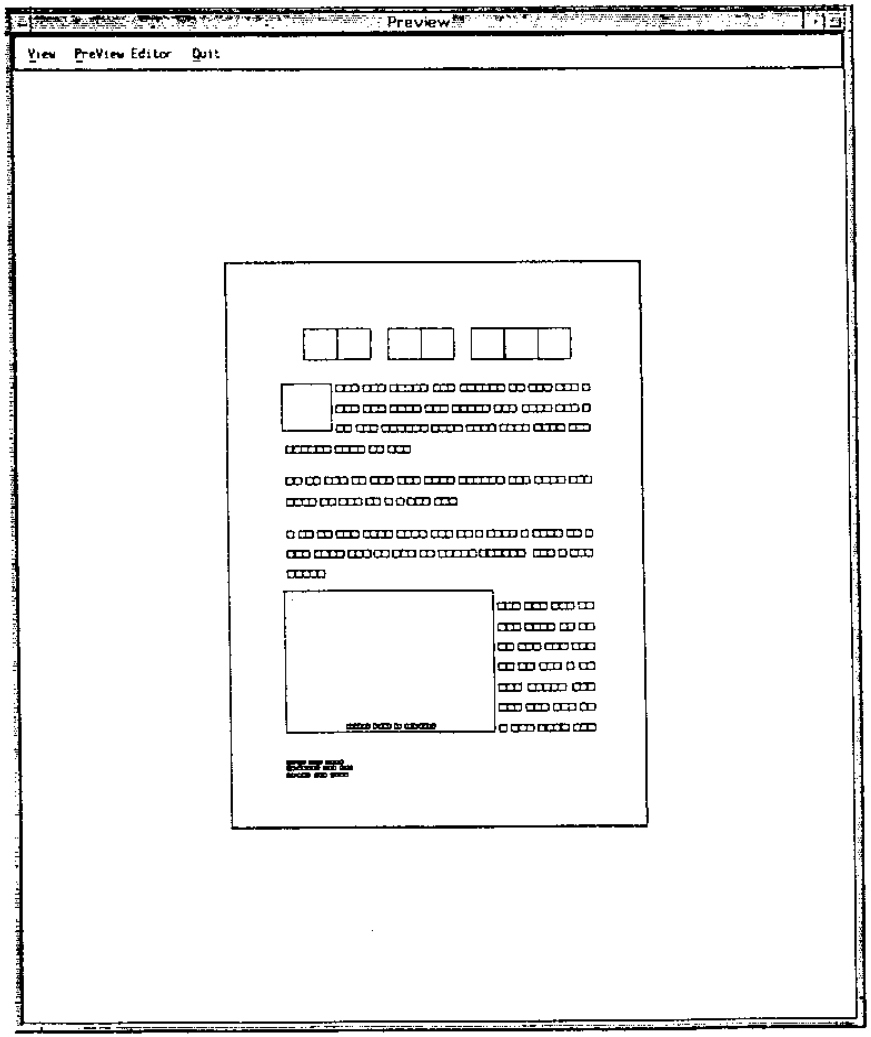


그림 6-14. Preview 예 1

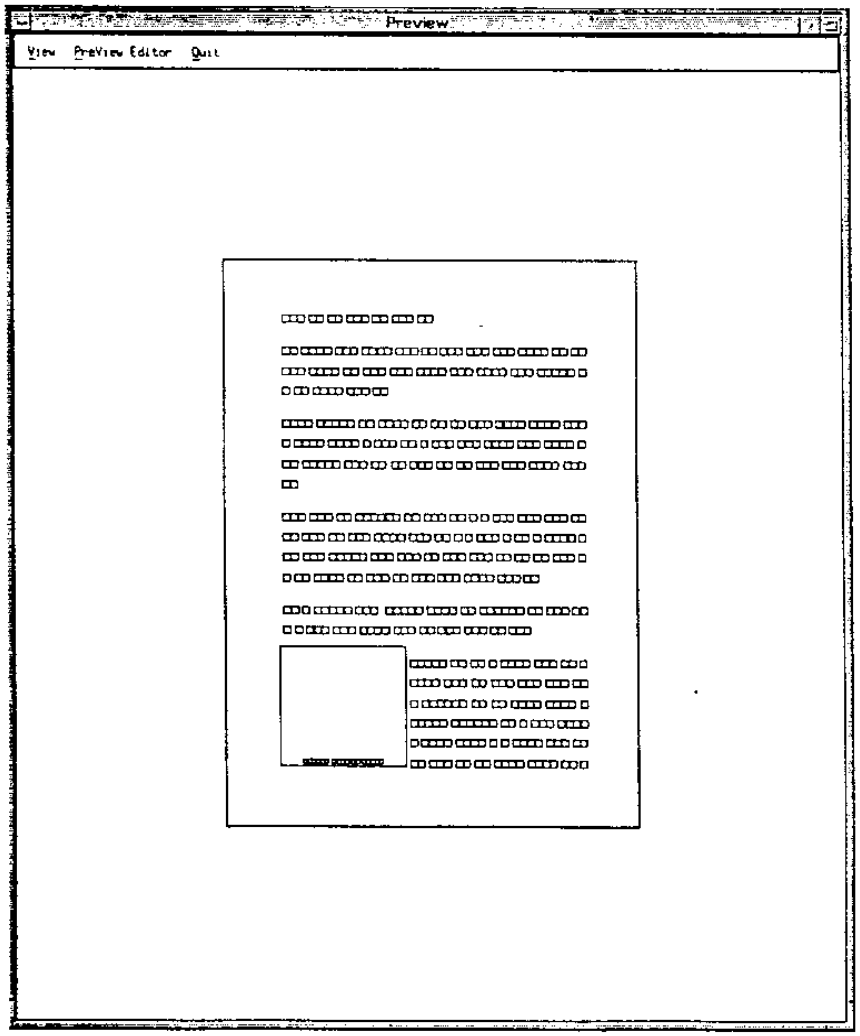


그림 6-15. Preview 예 2

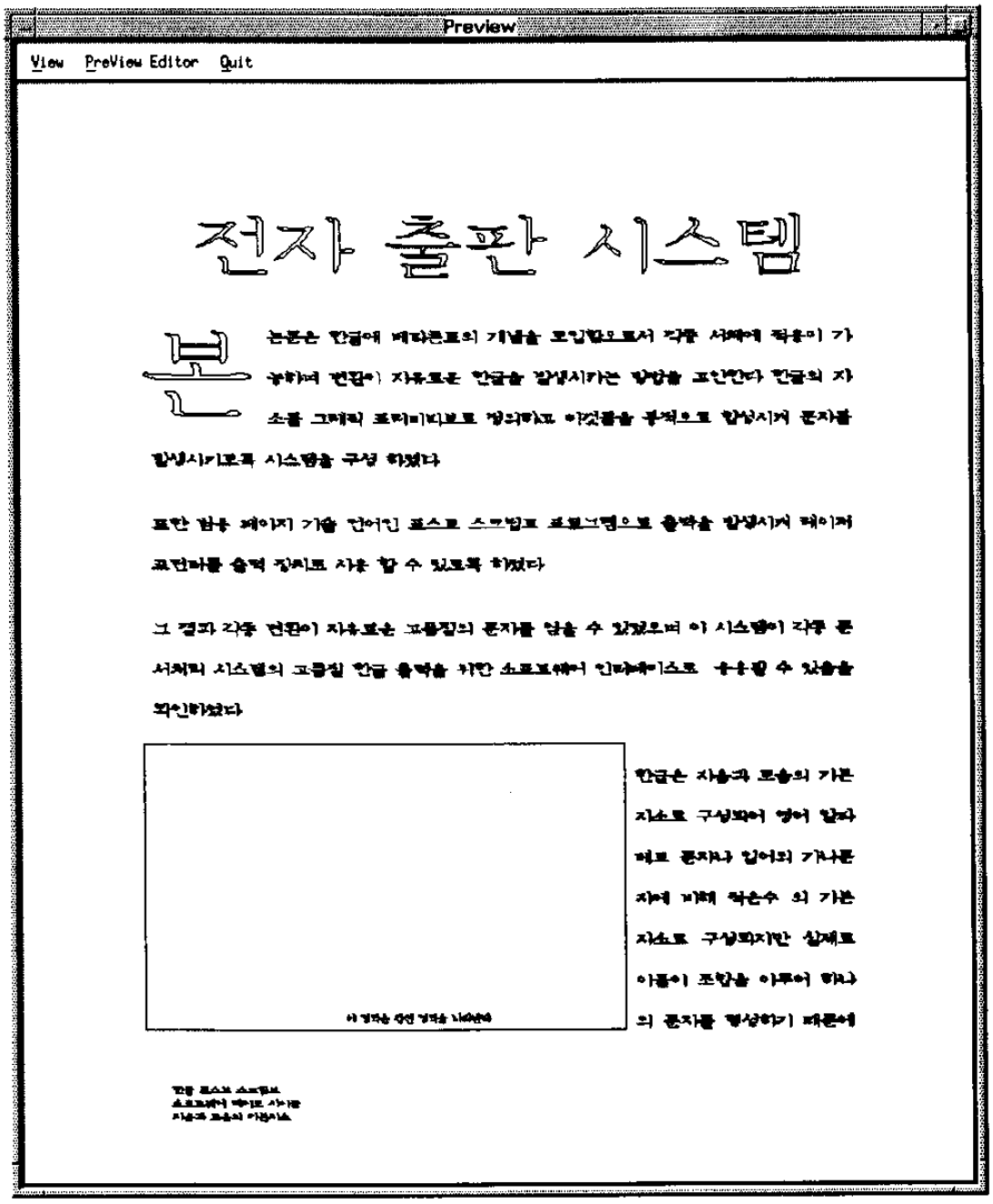


그림 6-16. Preview 예 3

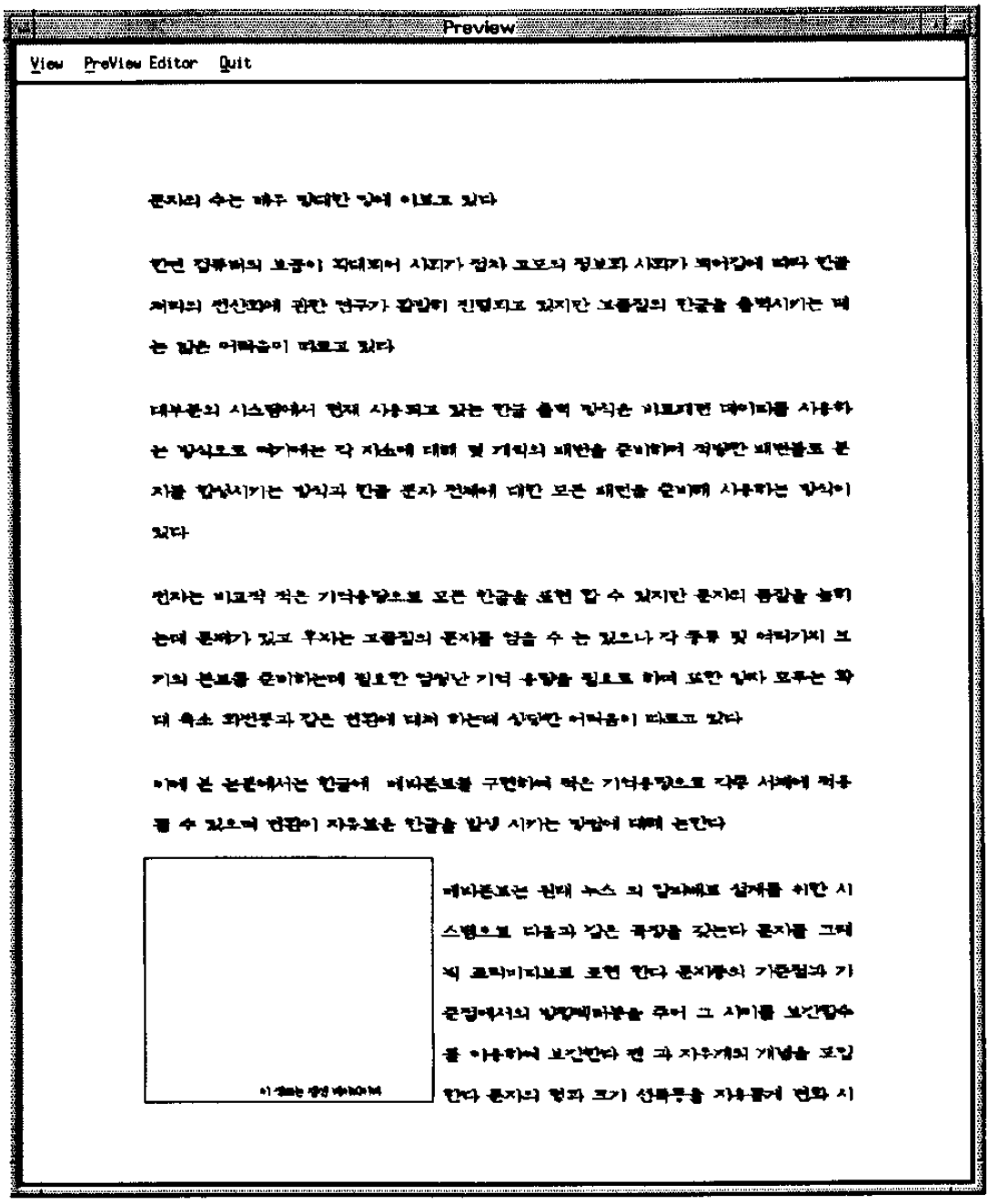


그림 6-17. Preview 예 4

제 6 장 결 론

UNIX 운영체제상에서 표준 윈도우 시스템으로 되고 있는 X 윈도우 시스템과 그래픽 사용자 인터페이스 Motif환경으로 Batch Mode Publishing 시스템의 이주를 완성하였고 복잡한 그래픽 및 Image가 포함되는 문서를 작성하기 위한 Compound Document graphic Editor를 연구개발 하였다.

또한 Batch Mode Publishing시스템의 내부 데이타 구조 및 레이아웃 처리를 개선하여 WYSIWYG alike한 레이아웃 처리 환경을 구축하였다.

이 개발된 시스템은 복잡한 그래픽 및 text, geometric / raster 그래픽의 혼용되는 business graphic작성 및 이들 mutimedia가 혼재하는 문서를 만들 수 있으며, WYSIWYG alike한 처리 방식을 갖고 UNIX운영체제 상에서 네트워크를 통한 Publishing System으로 사용될 수 있다.

: 차후 연구 방향은 Full WYSIWYG방식과 Markup방식이 동시에 가능한 시스템의 연구 개발과 DDE(Dynamic Document Editing)환경 구축을 위한 연구를 할 계획이다.

참 고 문 헌

1. Oliver Jones, "Introduction to the X window System", Prentice-Hall, 1989

2. "OSF/Motif Programmer's Guide", OSF, 1989

3. "OPEN LOOK graphical User Interface Functional Specification", Sun Microsystems, 1988

4. Adobe system, "PostScript language - Program Design", addition-wesley publishing Co., 1988

5. 조명진 외 3명, "한글 X 윈도우 시스템 및 그래픽 사용자 접속에 관한 연구", 한글날 기념 학술 발표 논문집, 한국 정보과학회지, 인지 과학회, 1989

6. 조명진,하종성, "한글 X 윈도우 시스템 Manual", Integration Project Report, KAIST, 1989

7. 임광택, 이수연, "Preview기능을 갖는 한글 전자출판 시스템", 한국인지/정보과학회 공동 주최, 한글날 학술 발표회, 1989

8. 임광택, 이수연, "Preview기능을 갖는 Batch Mode 전자출판 시스템에 관한 연구", 광운대학교 대학원 석사학위 논문, 1989. 2

9. 김형준, 이수연, "한글 메타 폰트의 개념을 도입한 전자 출판 시스템에 관한 연구", 광운대학교 대학원 석사학위 논문, 1988.2

10. 강한종외 3인, "다목적 한글 Metafont Generation에 관한 연구", 과학기술처 '87 특정연구 결과 발표회 논문집, 1988

11. Reid B.K, "SCRIBE Introductory User's Manual", CarnegieMellon Univ., 1978

12. Donald E. Knuth, "Tex and METAFONT - New Directions in typesetting", Digital Press and American Mathmetical Society, 1979

13. Xerox Corp., "Ventura Publisher Edition Reference Guide", Ventura software Inc., 1987

14. Aldus Corp., "Pagemaker User manual and Reference Manual", Aldus Corp., 1987

15. ISO/IEC JTC1/SC18/WG5 N747, User Requirement for Busineess Charting

16. Hiroshi Shojima., "이치화상의 각종 확대/축소방식의 성능 평가를 위한 처리 속도 개량 방식", 정보처리 학술 논문지, Vol.26, No.5, 1985

17. 김차종,이수연, "구조적 문서 작성 에디터", '한국정보과학회 봄 학술 발표 논문집 , Vol.16, No.1, pp. 449-452, 1989.4

18. 임광택,이수연, "Preview 기능을 갖는 Batch Mode 전자출판시스템", '한국정보과학회 봄 학술 발표 논문집 , Vol.16, No.1, pp. 453-456, 1989.4

19. 김차종,강한종,이수연, "다목적 폰트 제작 시스템", 정보과학회 논문지, Vol. 16, No. 5, Sep 1989

주 의

1. 이보고서는 과학기술처에서 시행한 특정연구개발사업의 연구보고서 이다.

2. 이 연구개발내용을 대외적으로 발표할 때 에는 반드시 과학기술처 에서 시행한 특정연구개발사업의 연구결과임을 밝혀야한다.