《主 題》

시각 프로그래밍 기술

김 상 욱
(경북대학교 자연과학대학 전자계산학과)

■차 례■

Ⅰ. 서 론

시각 프로그래밍이란 프로그래밍 과정에서 이차 이상의 시각적 표현을 사용함으로써 컴퓨터로 처리하는 일에 편의를 제공한다[2,3,5,8,11,14,17]. 이 글은 시각 프로그래밍에 대한 정의와 그것을 이용한 프로그래밍 기술과 발달 과정을 알아보고, 시각 프로그래밍에 대한 공통의 이해를 구축하고, 여러 시각 프로그래밍을 집중 분석하려 한다[11]. 그 후, 시각 프로그래밍 언어, 즉, 시각적 표현으로 프로그램을 작성할 수 있는 언어에 대하여 여러 측면, 다이어그램, 아이콘, 테이블에 기반을 둔 접근 방식을 분석하고, 지식 표현 시각 언어인 ScreenPlay를 제시하고 컴파일 기술을 설명한다[7,11,13,37]. 또한 사용자 명령을 편리하고 쉽게 사용하도록 사용자 명령에 대한 시각 명령 기술에 대하여 살펴본다[18,19,20,21]. 끝으로 공동작업을 지원하기 위한 시각 프로그래밍 기술은 무엇이며, 어떻게 설계되어야 하며, 어떠한 방향으로 연구되어야 하는지를 제시한다[22-35].

Ⅱ. 시각 프로그래밍

1. 시각 프로그래밍의 발달 배경

기존 텍스트 중심 프로그래밍 방법은 사용자가 프로그래밍 언어를 이용하여 설계한 후, 설계된 프로그램이 컴퓨터에 의해 효율적으로 번역되어 실행되도록 한다. 이 때는 사용자는 프로그램을 설계할 때 까지는 상당한 노력과 시간을 투자하여야 한다. 그러나, 최근 몇년 동안 개인용 컴퓨터와 사용자가 급증하였고, 컴퓨터의 적용 분야도 늘어나게 되었다. 이러한 현상은 여러 분야의 많은 사용자의 증가를 가져왔고, 사용자 관점에게 사용하기 편리한 컴퓨터를 요구하게 되었다. 특히 프로그래밍에서는 많은 시간을 소요하므로 컴퓨터를 편리하게 사용하고, 프로그램을 배우는 데 최소한의 시간과 노력만을 요구하게 되었다. 이러한 문제를 쉽게 풀기 위한 방법 중의 한 방법이 그림에 의한 "시각 프로그래밍 기술"이다. 시각 환경을 이용한 시각 프로그래밍 기술은 컴퓨터 발달 과정에서 혁신적인 방법이다[1-17]. 시각 프로그래밍 기술은 그림을 사용하므로 다음과 같은 잇점을 지닌다.

- 흥미성 : 시각 환경과 도구의 발전
- 이해성 : 시각 언어의 발전
- 표현성 : 소프트웨어의 설계 방법의 발전
- 의미성 : 지식 표현의 발전
- 정확성 : 공동 작업의 발전

이 글은 시각 프로그래밍에 대한 공통적인 이해를

추구하고, 시각 언어에 대해서 여러 측면에서 관찰, 분석하여 시각 언어 기술의 발전을 위한 철학을 제시하게 된다.

2. 시각 프로그래밍이란?

시각 프로그래밍에 대한 정의는 사용하는 사람이나 연구지에 따라 각기 다르게 사용된다. 단순히 시각적인 모든 환경을 의미하기도 하고, 반드시 프로그래밍이란 개념이 있어야 한다고 하기도 한다. Mayer는 "사용자가 프로그램을 이차원 이상의 표현 방법으로 작성하는 것이다"라고 하고[11], Shu는 "의미있는 그래픽의 도형등의 표현으로 프로그래밍할 수 있는것"이라고 정의한다[11]. 또한 S.K. Chang은 "지식의 의미가 시각적 도형으로 표현되는 것이다"라고 한다[12-15]. 결국, 시각 프로그래밍이란 프로그래밍에 필요한 모든 제반 여건과 프로그래밍의 모든 단계를 일차원이 아닌 이차원 이상으로 표현하고 지식을 표현하여 컴퓨터로 하여금 사용자의 의미를 이해하도록하는 방법이다.

그러므로 시각 프로그래밍의 발달 방향은 크게 세 부류로 분석할 수 있다[11]. 첫째, 시각 환경이다. 이러한 시각 환경에서는 그래픽 기술과 그래픽 장치들이 프로그램의 구축과 이해, 디버깅, 정보의 재생과 표현, 소프트웨어 설계를 지원할 수 있도록 하여야 한다. 둘째, 시각 언어이다. 시각 언어는 시각적인 정보를 처리할 수 있고, 시각적인 대화 기능을 제공하고, 시각적인 표현으로 프로그래밍할 수 있도록 설계된다. 이때는 컴파일러와 같은 시스템 프로그램이 필요하게 된다. 세째, 공동작업 지원 시각 인터페이스이다. 단순 프로그래밍 기능외에 제스처, 손놀림, 목소리, 글등의 멀티미디어를 사용한 지식을 표현, 전달하고, 동시에 공동 작업과 공동 프로그래밍할 수 있어야 한다[22-23].

3. 시각 도구와 시각 언어에 의한 프로그래밍의 차이

시각 프로그래밍을 제공하는 두 경우인 시각 도구와 시각 언어는 각각의 특징과 함께 공통점과 차이점을 가지고 있다.

시각 도구는 자료 정보, 프로그램과 실행 상황, 소프트웨어와 설계 방법과 같이 주로, 서로 다른 각 객체 클래스들의 구조, 컴퓨터 내부의 표현과 실행, 소프트웨어 컴포넌트등의 "시각화"에 중점을 두고 있다. 물론 각 경우에 있어서의 문제 해결 방법은 서로 다르다. 반면, 시각 언어는 시각 정보를 처리하고, 시각적 대화를 지원하며, 시각적 표현으로 프로그래밍 등의 "명령이나 지시"를 이용하여 컴퓨터가 하여야 할 일의 지시를 시각적으로 표현한다.

시각 도구는 모두 컴퓨터와 대화할 수 있는 시각적인 도구를 제공한다는 것이다. 즉, 보여주는 것(showing)이 기본적인 기술이다. 시각 도구는 어떤 것도 프로그래밍 기술의 측면에서 볼 때 언어적인 새로운 요소를 제공하지 못하고 있으며, 이들의 강조점은 시각적 도구에서 언어가 아니라 사용자와의 대화에 있는 것이다. 시각 도구가 대화의 수단으로 "보여주는 것"을 이용하는 반면, 시각 언어는 컴퓨터에게 "말하는 것"을 이용하여 할 일을 시각적으로 나타낸다. 이러한 특징이 시각 도구와 시각 언언에 대한 뚜렷한 차이를 보여 준다[3].

Ⅲ. 시각 도구에 의한 프로그래밍 기술

1. 자료 정보의 시각화

자료 정보의 시각화는 시각 도구에서 자료 자체의 시각화와 자료에 대한 정보 즉, 자료 구조의 시각화이다. 이것은 데이타베이스 관리 시스템에서 내부적으로는 기존의 데이타베이스로 저장되어졌으나, 스크린 상에서는 그래픽 형식으로 표현되어진 자료를 보여줌으로써 사용자가 키보드를 사용할 필요없이 포인팅 디바이스를 이용하여 더 정확하게 자료를 얻을 수 있도록 하기 위해서이다. 이러한 방식은 질의어를 학습하여 습득할 필요없이 질문에 답할 수 있도록 하고 있다. 또한 프로그래밍에서 중요한 역할을 가지는 자료 구조와 데이타베이스 스키마를 그래픽으로 표현하여 복잡한 프로그래밍을 처리해야만 하는 사용자에게 도움을 주기 위해서 이다[1, 21].

이러한 시스템들은 재생되어진 정보의 시각화를 위하여 이용하고 마음대로 읽고 쓸 수 있는 정보의 의미로서 "직접 처리"를 지원하고 있다. 이런 분류의 시스템은 SDMS, VGQF, KAESTLE, ISIS, AMETHYST, IBS, the Metaphor system, "그래픽 객체-관계 데이타베이스 스키마"등이 있다.

2. 프로그램과 실행의 시각화

프로그램과 실행의 시각화는 시각 도구에서 프로그램의 시각화, 프로그램의 실행 상태의 시각화, 실행 결과의 시각화를 위한 그래픽 시각 도구의 제공이다. 프로그램과 실행 상태의 시각화는 그 프로그램이 무

엇을 하는지, 어떻게 실행되는지, 왜 실행되는지, 실행과 그 결과는 무엇인지를 명확하게 알 수 있다[21].

소스 코드의 Pretty-Printing, 다이어그램으로 기존의 프로그램의 표현, 프로그램과 실행상태를 구문 트리, 심볼 테이블, 변수, 제어의 흐름, 실행 시간 스택 등으로 보여주는 것, 애니메이션 형태로 프로그램의 실행 과정을 보여주는 것 등으로 확대되었다. SEE 컴파일러, LISP를 위한 시각 구문 에디터, LISP의 삼차원 실행 화면, Prolog를 위한 아이콘 기반 설계 방법, CEDAR, PECAN, GARDEN, PROVIDE, "VIPS : 시각 디버거", "시각 파스칼", "스몰토크를 이용한 애니메이션 프로그램", BALSA, BALSA-II 이 분야를 나타낸다[11].

3. 실행 결과의 시각화

실행 결과를 시각적으로 표현하는 방법에는 그래픽 터미날, 흑백이나 칼라, 프린터, 스크린 프로젝터, 플로터등 매우 다양하며 각 출력 도구에 따라 시각화 기술이 달라지게 된다[21].

4. 소프트웨어 개발의 시각화

오늘날의 프로그램은 대용량의 자료와 라이브러리나 소프트웨어 컴포넌트의 재사용을 이용한 대규모 프로그램이 특징이다[2, 5, 7, 9, 15]. 소프트웨어 개발의 시각화는 이러한 "대규모프로그램"을 설계할 때 프로그램의 이해와 설계 개발을 지원하는 것을 목적으로 한다. Lampson에 의하면 컴퓨터 시스템에 의한 설계는 알고리즘의 설계와는 많은 차이가 있는데, 요구사항에 의한 외적 인터페이스는 덜 정확하게 정의되더라도, 내부 인터페이스는 더욱 복잡하고, 변화에 민감한 여러 인터페이스를 요구하게 된다고 한다.

대규모 시스템은 복잡하기 때문에 소프트웨어의 개발을 시각화하기 위하여는 효율적인 환경과 수학적 계산의 정확성을 지원하는 두 방법으로 지원한다. PV 시스템, 그림을 이용한 소프트웨어, CASE 툴, ObjectCraft 등은 효율적인 환경을 중시하여 사용자에게 그래픽 툴을 제공하는 좋은 예이며, PagaSys 시스템은 수학적인 정확한 계산을 중요시 하는 시스템이다. 즉, 형식 다이어그램과 프로그램 사이의 일관성이 그 시스템에 의해 증명되거나 반증될 수 있도록 하기 위해 그래픽 표현을 도입한다[15].

Ⅳ. 시각 언어에 의한 프로그래밍 기술

1. 시각 정보의 처리 언어

1970년대 초에 그림이나 이미지등으로 표현된 자료의 처리를 위한 영상 처리 시스템과 문자와 숫자를 포함하는 자료의 생성, 관리, 조작, 재생, 저장을 위한 데이타베이스 관리 시스템이 개발되었다. 영상 처리 시스템은 대개 지리학, 의학, 과학, 공학에 응용되었는데, 각 시스템은 특정 환경에서 특정한 목적을 위해 만들어졌다. 이때, 그림이나 이미지로 표현된 자료의 공유란 어려운 일이었다. 또한 데이타베이스 관리 시스템은 문자와 숫자를 포함하는 데이타를 공유하고 처리하는데는 효율적이지만 그림이나 이미지로 나타낸 자료의 처리는 기존의 질의어를 사용하여 처리하거나 표현하기가 매우 어려웠고, 공간적인 관계를 처리하기나 표현하는 질의어를 만드는 것 역시 어려웠다[11, 13].

1970년대 후반기에 데이타베이스 관리 시스템과 그림으로 표현된 자료를 동시에 저장 및 처리할 수 있게 되었다. 그러므로 생성되고 분석되어진 이미지 자료의 양적 증가와 그 이미지 자료를 처리하고, 공유할 필요성과 능력이 오늘날의 시스템을 개발하게 된 동기이다.

이때의 간단한 방법은 기존의 데이타베이스 질의어에 이미지를 처리하는 능력을 포함시키는 것이다. 따라서, 이미지로 나타난 자료를 위한 여러 질의어는 확대된 데이타베이스 질의어로서 구현되어지도록 하였다. 예를 들어, GRAIN(the Graphics-oriented Relational Algebraic INterpreter)은 RAIN의 확장이고, GEO-QUEL은 QUEL의 지질학적인 확장이다. 그리고, PSQL(Pictorial Structured Query Language), ISQL(Image SQL), IDEMS는 SQL(Structured Query Language)의 확장이다[3, 11, 15].

그러나 주의하여야 할 것은 대부분의 경우에 시각 질의어가 처리하는 것이 이미지 자료의 객체이지만, 질의어 자체가 그림이거나 이미지가 아니다. 오늘날에는 시각 자료를 처리하는 질의어 뿐만 아니라, C, C++, Pascal 등 시각 자료를 처리할 수 있는 언어는 다른 면에서 볼 때 모두 시각 정보의 처리 언어에 해당된다.

2. 시각적 대화의 지원 언어

시각 언어 시스템에서 그래픽 화면과 화면의 포인

팅 디바이스가 컴퓨터와의 대화에 중요한 역할을 한다. 이러한 언어에서 그래픽 화면을 만드는 전형적인 방식은 매개 변수의 입력을 허용하고, 데이타베이스를 처리할 수 있는 프로그램을 작성한 후에 요구된 화면을 만들어 낼 수 있는 그래픽 처리를 위한 서브루틴을 호출하는 것이다[3, 8].

그러므로, 시각적 대화 방식이 증가됨에 따라 시각적 대화를 지원할 수 있는 시각 언어가 개발되었다. SDMS의 ICDL(the Icon-Class Description Language), 프로그래밍에서 규칙 베이스를 이용하여 시각적 대화를 지원하는 언어인 HI-VISUAL, Squeak, Coral (Constraint-based Object-oriented Relations And Languages), "그래픽 에디터를 위한 인터페이스 서술" 등이 이러한 언어의 예이며, 다양한 형태의 시각적 대화를 지원한다[11, 15].

3. 시각 프로그래밍 언어

시각 프로그래밍 언어는 사용자가 그래픽 표현을 사용하여 직접 프로그래밍 할 수 있도록 한 언어이다. 시각 프로그래밍 언어는 전형적인 일차원 프로그래밍 언어로 쓰여진 단어나 숫자를 어떤 시각적 표현을 사용하는 언어로 정의하는 언어이다[15].

이 언어에서는 자료나 정보가 어떠한 타입도 가지지 않는다. 즉, 문자, 숫자, 그림, 소리 또는 이런 여러 형태의 자료 조합에 의한 타입이 없는 자료이다. 시각 프로그래밍 언어에서 중요한 것은 "언어 그 자체는 어떤 의미있는 시각적 표현에 의하여 프로그램 되어야 한다"는 것이다. 그러므로 시각 프로그래밍 언어를 이용한 프로그램은 보편한 의미에서의 언어 구성 요소로 구성되어져야 한다. 이 언어를 이루는 기본 요소인 아이콘, 선, 박스, 화살표, 구성된 형태등은 잘 정의된 "구문과 의미"를 갖추고 있으며, 이들 요소로 표현된 "문장"인 아이콘들을 연결한 제어나 자료 흐름의 경로, 특정 구조를 갖춘 도표, 정형화된 형태등은 "구문 분석(Syntax Analysis)"되고 "의미 해석(Semantic Analysis)"되어 번역되어질 수 있다[13, 14, 15].

시각 프로그래밍 언어의 설계 원리로 보면,

(1) 플로우차트와 다이어그램 이용 설계
(2) 아이콘이나 그래픽 심볼 이용 설계
(3) 테이블 이용 설계

등으로 나눌 수 있다.

플로우차트와 다이어그램을 이용하여 설계된 시각 프로그래밍 언어는 기존의 프로그래밍 언어의 확장 또는 기존의 프로그래밍 언어와 결합하기 위해 컴퓨터가 해석 가능한 단위로 만들어졌다. 아이콘이나 그래픽 심볼이 이용된 시각 프로그래밍 언어는 그림으로 나타낸 표현으로써 프로그래밍 개념을 수행하고 프로그래밍을 사용자에게 학습하게하기도 한다. 테이블을 이용하여 설계된 시각 프로그래밍 언어는 다른 시각 프로그래밍 언어처럼 정형화된 구조의 그래픽 표현들이 언어의 중요한 부분으로 설계되지만, 아이콘 시스템에서의 아이콘과는 다르게 설계되었다.

가. 다이어그램을 이용한 프로그래밍 언어

칠판이나 종이에 연필로 그림을 그리듯이 도표, 그래프, 다이어그램등이 프로그램의 문서화나 설명을 위해서 시각적으로 도움을 주었다. 그러나 이러한 그래픽 지원이 그 자체로 실행 가능한 프로그램을 의미하지는 않는다[11, 18].

본래 프로그래밍은 여러 단계를 필요로 한다. 즉, 문제의 분석, 프로그램의 추상화를 위하여 다이어그램을 이용한 도표화, 코딩, 컴파일러나 인터프리터에 의한 번역, 테스트이다. 얼마전까지 이러한 방식의 문제점은 프로그램을 나타내는 코드와 시각화된 다이어그램 둘 다 실행이 끝날 때까지 유지되어야 한다는 것이다.

도표를 실행 가능하게 하는 것은 프로그램의 추상화와 코딩의 두가지 다른 처리를 하나로 만드는 것이므로, 프로그램을 더욱 쉽게 이해할 수 있게 하고, 더 쉽게 문서화 할 수 있게 하며, 유지할 수 있게 한다.

또한 프로그래밍의 어떤 면은 다이어그램 방식으로 가장 잘 나타낼 수 있다. 예를 들어 상태 전이 다이어그램은 사용자 인터페이스 설명을 위하여 적합하다. 각 상태에 전이 규칙을 줌으로써, 사용자가 각 상태에서 무엇을 할 수 있는지, 그 결과는 무엇일지를 외적으로 나타낼 수 있다. 다른 예로, 동시에 작용하는 처리의 상호 관계를 문자보다는 다이어그램으로 더 잘 나타낼 수 있다. 결국, 다이어그램 형태가 추구하는 프로그래밍의 관점을 위하여 개발된 기본 개념은 널리 사용되는 규정을 따르는데, 예를 들어, FGL, GPL에서의 자료 흐름 다이어그램, USE, Jacob의 상태 전이 다이어그램에서의 상태 전이 다이어그램, VERDI, GSDL, PFG의 Petri Nets, FPL, Pascal/HSD, GAL, "그래픽에 기반을 둔 프로그래밍을 지원하는 시스템"인, Pigsty/I-PIGS의 여러 형태의 플로우차트 형식이 있다.

다이어그램 언어를 이해하고, 발달된 추세를 보기 위해, "그래픽에 기반을 둔 프로그래밍을 지원하는 시스템"인 PIGS와 Pigsty/I-PIGS를 좀 더 자세히 알아본다.

"그래픽에 기반을 둔 프로그래밍을 지원하는 시스템"은 기존의 프로그래밍 언어를 그래픽으로 확장한 것으로써, 도표(chart)를 이용한 최초의 시스템이다. 도표를 실행 가능하게 하기 위해 Nassi와 Shneiderman에 의해 제시된 구조화된 다이어그램을 "헤더(headers)"에 포함하도록 확장하였는데, 이를 확장된 NSD라 한다.

NSD는 두 부분으로 구성된다. 선언부인 헤더 부분은 다이어그램의 이름, 그 기능에 대한 해설(comment), 지역 변수와 인자에 대한 정의를 포함하고, 명령부인 몸체 부분은 NSD인 순차 구문(sequential), IF, CASE, DO-LOOP으로 되어 있는데, 이는 PL/I 언어의 부분 집합이다. NSD 구문은 제어 흐름과 수행되는 연산을 명세한다. 도표를 그리고 수정하도록 하기 위하여 그래픽 에디터를 지원한다. 에디팅은 화면상의 한 위치를 지적하거나 문자를 타이프 하는 것이다. 각 문자 S, I, C, L을 타이프하는 것은 시스템에게 지적된 위치에 SIMPLE, IF, CASE, DO-LOOP를 구성하도록 한다. T는 지적된 NSD 구문으로 들어가게 하는 문자이다.

NSD 프로그램을 실행하기 위해 도표를 PL/I 소스 프로그램으로 번역하는 전처리기를 거친 후 PL/I 컴파일러에 의해 컴파일되고 실행된다.

편리한 환경을 제공하기 위해 "대화적인 그래픽 지원을 위한 프로그래밍"을 나타내는 PIGS가 있는데, 이 PIGS 프로그램을 실행 가능한 도표의 형태로 Programming Support System에서 개발된 NSD를 구축하지만, 기본 언어로는 PL/I 대신 Pascal이 사용된다.

PIGS는 테스트, 디버거, 실행을 대화적으로 지원하는데 중점을 두고 있다. 컴파일러 보다는 인터프리터가 프로그램과 대화할 수 있고, 실행동안 변경할 수 있도록 하기 위해 사용된다. 실행이 진행되는 동안 사용자는 NSD 프로그램의 논리의 흐름을 따라갈 수도 있고, 볼 수도 있다. 구문의 윤곽과 포함된 문자가 NSD 구문이 실행되는 동안 그래픽 터미날에서 밝게 나타난다.

최근에는 PIGS의 개념이 Pigsy/I-PIGS에서 동시 수행가능한 프로그래밍을 지원하는 데까지 확장되었는데, 이 Pigsty는 Pascal 언어와 CSP(Communicating Sequential Process)에 기반을 두고 있다. I-PIGS는 Pigsty를 지원하는 프로그래밍 환경이다.

PIGS처럼 Pigsty도 문자와 다이어그램을 프로그램을 나타내기 위해 혼합하여 사용한다. Pigsty의 순차 구문은 Pascal 언어로, 제어 구문은 도표 형태로 나타내어진다. Dijkstra의 선택문 IF-FI와 반복문 DO-OD는 ALT-/ALT와 *ALT-/*ALT가 된다. Pigsty 프로그램은 각각이 박스로 표현된 하나 이상의 순차적인 처리를 구성하게 된다. 처리는 한방향 링크를 통해 서로 통신하게되고, 통신과 동기 방법은 CSP와 같다. I-PIGS는 시뮬레이션된 동시 수행 가능한 실행 방법을 통해 도표 프로그램을 실행시키고, 실행되는 동안 데드 록을 감지하게 된다.

위의 세가지 예들은 Nassi와 Shneiderman의 구조화된 다이어그램 방식의 발전 과정을 나타내고 있는데, 다이어그램의 기본적인 특징은 비슷하면서 기존의 프로그래밍 언어와 결합되거나 확장되었다.

나. 아이콘 프로그래밍 언어

아이콘 시스템은 아이콘을 객체와 행위를 나타내기 위해 사용한다. Star 시스템은 아이콘 시스템으로서 컴퓨터와 대화하기 위해 아이콘과 포인팅 디바이스를 이용한다. 처음에 사용자는 Star의 초기 화면인 "desktop"을 보게된다. 문서, 서류철, 화일 서랍, 바구니 등이 desktop상에 아이콘으로 디스플레이된다. 사용자는 그것을 선택함으로써 아이콘을 열 수 있고, 키보드로 OPEN 키를 치면 윈도우라 불리는 더 큰 형태로 확장된다. 아이콘은 내용이 윈도우에 나타나고, 사용자는 문서를 읽고, 서류철이나 화일 서랍의 내용물을 조사하여 편지를 보내거나 받는다. Star는 스크린상에서 문서를 만들어내는 강력한 기능을 갖춘 에디터를 가지고 있다[11, 15, 17, 18].

Star 시스템은 문서를 새로 작성하고, 재생하고, 분류하는 사무직 근로자를 위해 설계된 것으로 인간과 컴퓨터의 인터페이스를 단순하게 함으로써 쉽게 친해질 수 있도록 했다. 이 시스템의 아이콘을 이용한 "desktop metaphor"의 후반기 제품들에 많은 영향을 끼쳤다. 이러한 아이콘과 윈도우를 사용한 사용자 인터페이스는 여러 시스템에서 보여지고 있다.

처음에는 아이콘 프로그래밍 언어의 기능이 단순히 아이콘을 이용한 사용자 인터페이스의 기능을 구비하였지만 후에는 "명령어"를 아이콘화하는 추세로 바뀌게 되었다.

최근 몇년동안 상당히 많은 아이콘 프로그래밍 언어가 알려졌다. Lisp에 기반을 둔 실행 가능한 그래픽으로서 VennLISP, ICONLISP, 그리고 Tinkertoy가 있고, 논리 프로그래밍에 기반을 둔 아이콘 프로그래밍

언어로서 Dialog.I가 있으며, Milner의 CCS(Calculus of Communication Systems)에 기반을 둔 형식적 의미를 지원하는 IDEOSY와 Clara가 있다. 또한 그래픽에서 Pascal과 C처럼, Pict와 BLOX는 알고리즘 언어를 표현한다. "Show and Tell"은 수수께끼와 같은 문제를 다룬다. PROGRAPH는 확장된 HI-VISUAL이고, G 언어는 반복문, CASE, IF, SWITCH와 다른 제어 구문을 가진 자료 흐름의 개념을 강화하였다. PROGRAPH2는 자료 흐름과 객체 위주의 개념을 혼합했고, InterCONS는 직접적인 처리 기술을 구비한 자료 흐름의 개념을 구현했다. BridgeTalk는 초보자가 프로그램을 배울 수 있는데 기반을 두었다.

위 언어의 대부분은 초보자나 최종 사용자가 프로그래밍 언어를 즐겁게 배울 수 있는 컴퓨터의 세계로 인도할 수 있는데 중점을 두었다. 이제 Pict와 Labview의 G 언어에 대해서 예를들어 알아본다.

Pict는 운영 체제나 명령어(command language) 단계에서 사용자 인터페이스를 넘어서는 목적으로 아이콘을 이용한 초기 버전이며, 프로그램을 구현할 수 있다. 기존 언어의 프로그램 또는 부프로그램의 이름, 인자 전달 모드, 자료 구조, 변수, 프로그램의 실행등이 다양한 종류의 아이콘으로 표현되고, Pascal의 REPEAT-UNTIL, WHILE문은 색깔과 방향성을 가진 경로로 표현된다.

프로그램의 작성은 계산 수행을 위해 필요한 여러 연산을 표시하는 아이콘을 미리 정의된 집합에서 선택하여 스크린 상의 프로그래밍 영역에 놓는다. 그러면, 이 아이콘들은 제어의 흐름을 나타내는 경로로 연결된다.

Pict 시스템은 프로그래밍 언어에서 BASIC이나 간단한 Pascal과 같으며, 순환적이고 임의로 서브 루틴을 호출할 수 있지만, 프로그램 모듈의 크기에 제한이 있고, 모듈 사이에 전달되는 인자의 수가 적기 때문에 확장되어질 필요가 있다.

반대로 LabVIEW의 G 언어는 어떤 실세계의 문제들도 해결할 수 있다. LabVIEW(Laboratory Virtual Instrument Engineering Workbench)는 기계 제어, 자료 획득, 분석, 계산, 디스플레이를 포함하는 과학적 응용에 대한 소프트웨어 구성 시스템이다. 사용자는 기술자, 과학자들로서 프로그램의 경험이 전혀 없는 사람들이다. LabVIEW 프로그램은 front panel과 실행 가능한 블럭 다이어그램으로 구성된 가상의 기계이다. Front panel의 역할은 그래픽 표현, 즉, 아이콘이 스위치, 다이얼, 손잡이, 디지탈이나 아날로그 계량기, strip-도표의 입출력을 제어한다. 블럭 다이어그램은 기계를 실행하는 프로그램이다. 사용자는 메뉴로부터 원하는 아이콘을 얻고, front panel이나 블럭 다이어그램에 놓는다. 아이콘을 연결하는 경로는 한 아이콘으로부터 다음 아이콘으로의 자료 흐름의 경로이다.

블럭 다이어그램을 형성하는데 사용되는 언어가 G이다. 이것은 순수한 자료 흐름 다이어그램에서 반복 연산과 조건문을 어려움없이 수행하는 자료 흐름 모드에 기반을 둔 그래픽 언어로서 다음과 같은 4가지의 제어 흐름의 구조가 제공된다.

(1) 사용자에게 실행 단계에서 정확한 순서를 정의하도록 한다.
(2) 반복 순환문(for)이 제공된다.
(3) case 선택 구조가 제공된다.
(4) 무한 루프(while)가 제공된다.

또한, 시프트 레지스터가 반복문과 무한 루프에서 다음 반복문의 입력이 되는 실행의 결과를 가지게 하여 한계를 정할 수 있게 하고, 순환적 호출을 가능하게 한다.

G 언어는 4개의 타입 즉, 실수/실수 배열, 부울/부울 배열, 스트링/스트링 배열, 그리고 구조문을 지원하며, 내장 함수로서 배열 산술식, 행렬과 벡터 대수, 통계식 함수, 신호 처리 루틴등을 지원한다. 내장 함수와 가상 기계는 아이콘으로 나타내고, 서브 루틴처럼 실행된다. 또한 더 큰 기계의 다이어그램으로 연결될 수 있다.

다. 테이블을 적용시킨 언어

사용자의 상당수가 테이블과 친숙하기 때문에 테이블은 사람과 컴퓨터사이의 자연스러운 인터페이스이다. Spreadsheet 프로그램은 테이블을 사용하는 목적으로 설계되었는데, 사용자에게 테이블에 관련된 계산을 성공적으로 지원하지만 그 기능은 제한적이다. 이 spreadsheet은 특정 종류의 프로그램의 작성은 쉬우나, 다른 종류의 프로그램의 작성은 무척 어렵기 때문에 정확한 정의 영역이 있는 인터페이스이다[11, 15, 18].

테이블을 응용한 분야는 자료의 입력, 디스플레이, 데이타베이스 질의, 유지 보수이다. 테이블을 적용시킨(form-oriented) 언어는 QBE/OBE, QBE, QBE/PC, QPE, FORMANAGER, IDEAL, "Fill-in-the-form programming," FILLIN, Forms, PICQUERY, FORMAL

이 있다. 이 언어의 대부분은 관계형 데이타베이스를 지원하나, FORMAL은 계층 구조 자료 구조로서, 테이블 형식보다 더 복잡한 자료를 처리할 수 있다. 이 FIRMAL은 사용자에게 훨씬 광범위한 편리를 제공하고, 자료의 적용 범위도 매우 광범위하다. 테이블을 이용한 언어인 FORMAL에 대하여 좀 더 알아본다.

FORMAL(Forms ORiented MAnipulation Language)은 최종 사용자를 위해 개발되었지만, 프로그래머에게 전형적인 프로그래밍의 개념을 가르치려고 설계되지도 않았고, 기존의 프로그래밍 언어의 확장으로 연결하려는 목적도 아니었다. 단지, 프로그래밍의 복잡성을 배우지 않고서도 복잡하고 많은 자료를 컴퓨터로 처리하기 쉽도록 설계되었다.

FORMAL에 의해 지원되는 자료 처리의 기능은 입력과 출력 테이블 표제로부터 자료의 재구성 기능을 가지고, 반복하여 자료 구조를 나열할 수 있고, 수학적인 연산과 스트링 연산을 할 수 있다. 경우에 따른 다른 할당을 할 수 있으며, 실행시에 사용자에 의한 필드 값을 공급 받을 수 있다. 테이블 혹은 상위 인스턴스로부터의 순서화를 제공 받고, 계층 경로로 진행되는 집합적 연산(count, SUM, AVG, MAX, MIN)을 지원 받는다. 이런 능력의 구비로, 매우 복잡한 자료 처리와 응용이 광범위하게 지원 가능하다.

그러나, FORMAL은 비절차적이고, 어떤 규정도 정해져 있지 않으며, 제어 구조도 없다. 사용자는 어떤 결과를 얻기 위해 컴퓨터에게 방법을 지시하거나, 알고리즘을 설계하거나, 코드화 하지 않는다. 결과는 컴파일러에 의해 생성된 코드에 의해 자동적으로 얻어진다. 이것은 FORMAL 컴파일러가 사용자로부터, 자료를 재구성하고 연산을 처리하기 때문이다.

컴파일러는 두 단계로 수행된다. 첫째, 입출력 자료의 구조 차이를 인식하여 다양한 변형 규칙의 기능을 수행한 후, 그 결과는 정의된 입력과 출력으로 대응시켜 얻을 수 있다. 두번째 단계에서는 구성이 시작되는데, 이때는 목적 언어에 저장된 지식과 실행 시간의 효율성이 이용된다. 그 결과는 즉시 실행 가능한 프로그램이 된다.

4. 지식 표현 언어

지식 표현 언어에서는 특별한 자료, 값, 객체들을 컴퓨터 화면에 처리되어질 지식의 단위인 개념으로 표현한다. 사용자는 이 개념들을 이용하여 지식을 표현하게 된다[6, 7, 36, 37, 38].

가. 지식과 개념 그래프

지식 표현 언어는 지식을 표현하기 위하여 개념 그래프를 사용하는데, 지식 특성을 다음과 같이 개념 그래프로 표현한다[6, 36, 37, 38].

[KNOWLEDGE-OBJECT]-
(abstract data type)→[KNOWLEDGE-CLASS]
(receive message)→[UNIQUE OWN BEHAVIOR]

[KNOWLEDGE-CLASS]-
(consist of)→[MEMBER:*x]
(degree of visualization)→[DATA ENCAPSULIZATION]
(reusability)→[INHERITANCE]

나. 지식 표현 환경

지식 표현 환경이란 사용자가 프로그래밍할 때, 지식 처리 개념에만 전념할 수 있게 하는 환경을 의미하는데, 이 환경에서 지식을 생성하여 프로그래밍하는 단위는 지식 객체의 추상화이다. 이 언어에서는 지식을 생성하는 것을 개념 그래프로 다음과 같이 표현한다[36].

[KNOWLEDGE PROCESSING PROGRAMMING ENVIRONMENT]-
(them)→[USER]→(devote oneself to)→[KNOWLEDGE CONCEPT]
(characterize)→[KNOWLEDGE PROCESSING PROGRAMMING CHARACTERS]
(generate)→[KNOWLEDGE]→(abstract data type)→[CLASS]

지식 표현 프로그래밍 환경은 사용자가 프로그래밍 작업을 할 때, 지식을 생성한 후에 컴퓨터 화면에 그 생성되어진 지식과 개념을 표현하여야 하므로, 화면 위에 제공되는 아이콘을 이용하여 프로그래밍할 수 있게 한다. 각 아이콘은 이미 정의되어 있으며, 이미지로 표현되는데, 각 이미지는 하나의 논리적인 의미를 가진다. 그러므로 지식을 시각적으로 표현하기 위한 개념 그래프는 다음과 같다.

[VISUAL PROGRAMMING ENVIRONMENT]-
→(them)→[USER]→(programming with)
→[ICON:*i]

[ICON:*i]-
(physical view)→[IMAGE]→(one to one mapping)
→[UNIQUE BEHAVIOR]

(logical view) → [UNIQUE BEHAVIOR]

여기에서 ICON:*i의 i는 프로그래밍 작업을 돕기 위하여 화면에 제공되는 모든 아이콘이다. 지식 표현 프로그래밍 환경은 지식의 추상화, 상속성을 잘 표현할 수 있도록 하는 시각 프로그래밍 시스템의 환경인데, 이 지식 표현 언어로 ScreenPlay[36, 37, 38]가 있다.

다. ScreenPlay

ScreenPlay는 지식 표현 언어로서 사용자와 컴퓨터 시스템을 사용하기 위하여 지식 처리를 시각적으로 프로그래밍할 수 있는 자동 프로그래밍과 인터페이스를 제공한다[36, 37, 38].

ScreenPlay 프로그래밍 환경은 모두 객체들로 구성되어 있으며, 사용자가 프로그래밍할 때 화면에 나타나는 아이콘으로서 아이콘 각각은 ScreenPlay 프로그래밍 환경을 구성하는 객체와 일대일 대응 관계에 있으며, 사용자는 이러한 아이콘을 이용하여 시각적으로 프로그래밍 한다. 또한 아이콘 각각은 전달되는 메세지에 따라 정확하게 고유의 행위를 설정하여 메세지를 처리한다. 사용자가 ScreenPlay 환경에서 프로그래밍할 때, 생성되는 것은 지식의 추상화인 클래스이다.

ScreenPlay의 각 모듈과 기능은 논문[37, 38]에 자세히 설명되어 있다.

5. 컴파일러 기술

시각 언어의 컴파일러 기술의 응용은 시각적인 모든 응용 분야의 사용자 인터페이스에 대한 설계에 기초가 되는데, 시각 언어의 컴파일 기술은 Lakin[4], Chang[12, 13, 14, 15] 등에 의하여 많이 연구되었다.

기존의 시각 정보 처리 언어, 시각 프로그래밍 언어, 아이콘 시각 정보 처리 언어에 사용되는 컴파일러는 주로 이차원의 공간 파싱 방법에 의한 구조 분석에 중점을 두고 있다.

Lakin[4]은 시각 문장, 즉, 시각 통신 객체에 대한 파싱 방법과 번역 기술을 조사, 분석하였다. Lakin[4]에 의하면 객체는 시각 표현을 위한 논리 객체와 관계있는데, 컴퓨터 화면 위에 아이콘을 기본적인 구문에 따라 표현하고 있으며, 구문에 따라 정렬된 시각 문장은 그 자체가 문법의 구문에따른 아이콘의 파싱 절차이다. 이러한 절차를 실행 그래프라 한다. 이 연구는 시각 프로그래밍 언어의 파싱에 매우 효과적이나 사용자는 문법의 구문을 이미 알고 있어야만 한다.

Lakin[4]에 의하여 개발되어진 시각 언어인 VennLisp[4]에서는 시각 객체가 계산에만 사용되는 것이 아니라, 그 계산 결과를 표현하는데 사용되기도 한다. VennLisp의 공간 파서는 먼저 공간적으로 시각 객체로 둘러 쌓인 부분의 관계를 찾아(위-왼쪽부터), 대응되어지는 파싱 트리를 구성한다.

Chang[12, 13, 14, 15]에 의하면 공간 파싱은 기존의 시각 언어와 자신이 제시하는 아이콘 시각 정보 처리 언어 둘 다 적용이 가능한데, Chang은 논문[12]에서 그림 문법과 그림 우선 순위 문법을 사용하여 객체의 공간 정렬을 파싱하는 기술을 제시하였다. 이 방법은 각 아이콘에 우선 순위를 주고, 아이콘의 우선 순위에 따라 파싱한다. Chang은 시각 언어 컴파일러의 실질적인 응용을 설명하기 위하여 아이콘에 관한 편집기와 시각 데이타 베이스 인터페이스를 포함한 여러 예를 들고 있다[15].

M.Tomita는 이차원 공간에서 표현되어진 그림 즉, 이차원 언어를 텍스트 언어의 파싱 방법인 Earley의 파싱 방법을 확장하여 제시하였다. 이 방법은 Context-Free 스트링 문법의 일반화로서 이차원의 문법을 설명하고 있으며, Earley의 방법을 확장하여 위치 LR 파싱 방법을 사용한다. 이 알고리즘의 주된 아이디어는 이차원 공간에서의 시각 언어를 파싱하기 위하여 그 다음 기호를 선택하는 방법이다[12].

지식 표현 언어의 컴파일러는 지식을 객체로 표현한 시각 문장을 컴퓨터 화면으로부터 받아들인 후, 그 객체를 구성하는 시각 문장의 구조를 분석하여, 지식 아이콘 문법을 나타내는 형식 문법에 따라 구문 분석을 수행하고 파싱 트리를 생성한다. 이 컴파일러의 특징은 지식 단위로 프로그래밍하는 도중에 각 아이콘이 그려질 때 직접적으로 파싱하며, 점진적 파싱을 하고 있다. 이 파싱 트리를 사용하여 의미 분석을 하고 시각 문장의 의미를 생성한다. 지식 표현 언어의 컴파일러는 지식의 시각 표현에 의한 개념 그래프로서의 의미 분석에 중점을 두고 있다.

V. 사용자 명령을 위한 시각 프로그래밍 기술

1. 시각 명령 언어의 요건

사용자 명령을 위한 시각 프로그래밍 기술을 지원하기 위한 언어의 설계를 위한 요건으로 다음과 같은 내용이 고려되어져야 한다[39]. 첫째, 명령 언어는 명령을 위한 것이므로 간단한 신택스를 가져야 한다. 복잡한 신택스는 인터프리터의 능력을 향상시키지만 프로그래밍하기에는 불리하다. 둘째, 명령 언어로 프로그래밍이 가능하여야 한다. 사용자 명령을 위한 시

각 프로그래밍 언어는 기존의 유닉스 쉘과 마찬가지로 명령 언어를 프로그래밍하여 명령 언어의 능력을 증대시키려는데 의도를 두고 있으며, 시각 언어에 기초하여 직접 조작으로 해결될 수 없는 것까지 융통성 있게 처리해줄 수 있다. 세째, 명령 언어는 빠르고 효율적인 인터프리터를 요구하게 된다. 이를 위해 시각 명령 언어는 점진적 파싱 방법을 사용한다. 따라서 프로그래밍이 끝나면 파싱이 완료되어 명령어에 대한 결과가 수행되어 파싱에 의한 시간 손실을 줄여 준다. 네째, 시각적 요소의 설계에 있어 기존에 존재하는 아이콘과 호환성이 있도록 설계되어 기존 사용자의 사용을 원활히 하여야 한다[18, 19, 20, 21].

2. 시각 명령 언어의 프로그래밍 기술

사용자 명령을 위한 시각 프로그래밍 언어는 아이콘의 배열에 기초한 언어이며 신택스 분석에 의해 각 아이콘은 하나의 객체로 구분된다. 구분된 각 객체들은 명령 언어에서 필드로 인식되어 실행된다. 따라서 시각 명령 언어에서는 명령어와 인자가 똑같이 하나의 객체로 처리된다. 시각 명령 언어는 직접 조작과 시각 언어를 모두 사용한다. 단순한 마우스의 조작으로 가능한 것은 직접 조작에 의해 수행가능하며, 변수 관리, 히스토리 관리, 앨리어스 관리는 직접 조작으로 가능하다. 예로, 변수 테이블에서 직접 변수를 할당할 수 있으며 히스토리 테이블에서 원하는 곳의 명령을 마우스 클릭에 의해 수행할 수 있다. if, for, while 등의 제어문을 사용한 명령문이나 새로운 명령어의 정의는 시각 언어에 기초하여 설계된다. 따라서 시각 명령 언어는 기존 명령어의 간편성을 유지하면서 유용성은 높인다. 시각 명령 언어의 명령어는 시각 명령 언어에서 제공되는 내부 명령어와 시각 명령 언어로 사용자에 의해 정의된 외부 명열어로 나누어진다. 외부 명령어를 사용함으로서 사용자는 명령어를 필요에 따라 확장할 수 있기 때문에 명령어의 능력을 향상시킬 수 있다[39].

Ⅵ. 공동작업을 지원하는 시각 프로그래밍 기술

1. 배 경

1980년대의 개인용 컴퓨터의 보급에 따라 개인 작업이 필요에 의하여 처리되었으며, 컴퓨터 환경은 단일 사용자용 프로그램 중심이었다. 이러한 프로그램은 제약적이고, 윈도우나 터미날을 통하여 각자 사용할 수 있었다. 그러므로 다른 응용 프로그램은 중앙집중식 화일 시스템을 통하여 사용하였다. 그러므로 이러한 환경은 컴퓨터와 인간 자신과의 작업 중심이었으며, 공동작업은 고려되지 않았었다. 그후 워크스테이션 발달에 따른 컴퓨터의 처리 능력 향상, LAN, ISDN 고속 네트워크의 발전으로 인하여 공동작업을 지원할 수 있는 환경이 되었으며, 동시에 공동작업을 지원하는 소프트웨어의 필요성이 인식되었다. 그러므로 여러 분야에서 공동작업을 할 수 있는 의사 결정 시스템의 지원, 조정 시스템, 절차 자동화, 인터페이스 분야의 학자에 의한 연구의 필요성이 강조되었고, 사회학, 심리학, 인류학, 언어학 분야와의 연관 관계에 의한 공동작업을 지원할 수 있는 연구가 중요하게 되었다. 동시에 시각 프로그래밍 기술도 공동작업을 지원할 수 있는 시각 프로그래밍 기술로 확장, 발전 연구되어왔다[22-35].

MERMAID는 멀티미디어 분산 회의 시스템 MERMAID(Multimedia Environment for Remote Multiple-Attendee INterative Decision-Making)는 일본 전기(주) C&C 시스템 연구소에서 개발한 원격지 다수간에 멀티미디어 정보를 사용하여 실시간으로 통보, 교환, 공유, 처리할 수 있는 데스크 탑 회의 시스템이다. MONET는 WEST VIRGINIA 대학에서 개발한 실시간 멀티미디어 데스크탑 회의 시스템으로 분산 환경에서 X-window 응용 프로그램을 공유할 수 있도록 하였다. Quilt는 Bellcore에서 개발한 것으로 문서 공유 문서에 주석 달기, 집필자 간의 전자 메일로 메시지 교환등의 기능을 제공하는 공동 문서 작성 시스템이다. Object Lens는 MIT에서 개발한 Information Lens는 전자 메일 뉴스 시스템에 있어 정보의 범람을 극복하고, 그룹에서의 정보의 공유를 촉진 시키기 위한 정보 자동 필터링 시스템이다. gIBIS는 MCC의 gIBIS는 그룹에 의하여 소프트웨어의 설계 공정을 지원하는 하이퍼 텍스트 시스템으로, 의사 결정의 내부 구조와 종속 관계를 이해하고, 대량의 비공식 정보를 획득하며, 그 정보의 인터 링크와 정보 검색의 효율적 방법을 제공한다[35].

2. 기존 시각 인터페이스

UNIX에서는 MIT의 X-Window, MS-DOS에서는 PM(Presentation Manager)가 주류를 이루고 있다. 현재의 UNIX 환경에서는 X-Window를 기본 윈도우 시스템으로 사용하는 OSF 사의 MOTIF와 UI(UNIX Internation)의 Openlook이 가장 대표적이다. MOTIF는 조작성과 사용자의 Look & Feel이 PM과 비슷하

게 설계되어 있어서 개인용 컴퓨터와의 상호 호완성을 가지고 있다. 위의 연구 방법을 진행하기 위하여 X-Window에서 C++로 구축한다.

CSCW에서는 다수 사용자가 동시에 작업할 수 있는 멀티 유저 인터페이스의 기능이 요구되므로 이를 위하여 X-Window 시스템에는 공유 윈도우 구현을 위한 기본 기능이 내장되어 있어, 기존의 단독 사용자용의 X-Window 용 응용 프로그램을 변경없이 다수 사용자용의 X-Winodw 응용 프로그램으로 사용할 수 있도록 지원한다. 공유 윈도우에서의 데이타 일관성 유지를 위한 독립서이나 발언권 통제, 제어 기능에 관한 연구가 병행된다.

3. 공동작업을 지원하는 객체 지향 시각 인터패이스

이러한 연구 과정에서 진정한 객체 지향 사용자 인터패이스를 통한 시각 프로그래밍 기술도 발전되고 있다[40]. 객체 지향 사용자 인터패이스에서의 시각 프로그래밍 기술은 외부적인 관점을 중요하게 다루게 되는데, 이러한 시스템의 사용자는 인터패이스에 대한 사용자의 개념적인 모델과 객체 지향 프로그래밍 언어에 의하여 지원받는 구조 사이에서의 동질 구조를 가지도록 할 수 있도록 하는 것이다. 이러한 연구 기술에서는 객체 지향 프로그래밍 언어와 객체 지향 인터패이스 사이에서의 유사성과 매핑을 깊이있게 연구되어 진다. 이러한 유사성은 "외부적"관점에서의 사용자 인터패이스의 설계를 명확하고 자연스럽게 객체지향 응용 프레임웍에 대응되도록 하여야 한다[40].

Ⅶ. 문제점 및 전망

1. 문제점

시각 프로그래밍 기술에 대하여 여러 사람들이 특별한 시각 시스템의 개발과 시각 언어의 설계, 공동작업 지원 인터패이스 쪽으로 연구 방향을 잡고 있지만, 시각 프로그래밍 기술에서의 가장 중요한 문제는 시각 환경을 어떻게 효율적으로 제공하여 주는가에 있다. 즉, 이러한 효율적인 환경을 제공하기 위한 인터페이스에 대한 일은 아이콘에 의한 프로그래밍 기술에서 효과를 보아 온 것이 사실이지만, 지금까지의 아이콘에 의한 시각 프로그래밍 기술 환경으로 충분한지 살펴 볼 필요가 있다. 시각 프로그래밍 기술에서의 문제점은 다음과 같다.

- 애매한 아이콘 모양
- 아이콘의 다양한 의미
- 컴퓨터 화면에 의한 프로그램의 제한
- 다양한 입, 출력 도구

지식 처리를 효과적으로 이루기 위한 방법은 아이콘을 인간의 지식이나 개념에 가장 가깝게 표현할 수 있어야 하며, 이 지식을 효과적으로 표현할 수 있도록 아이콘이 설계되어야 한다. 즉, 인간 사고에 의한 지식과 컴퓨터 화면에 표현되는 지식 사이의 거리를 좁혀야 한다. 이와 같이 하려면 다음과 같은 문제점을 고려하여야 한다.

- 개념이나 지식의 활성화
- 지식 표현과 처리의 지능화
- 시각 매체의 현실성

그러므로 시각 프로그래밍 기술은 위의 사실에 근거하여 다음에 충실하여야 한다.

- 정확한 시각적 의미
- 정확한 처리 절차
- 다양하고 정확한 지식 표현
- 컴퓨터 화면에 충분한 프로그래밍
- 용이한 번역
- 삼차원 포인팅
- 공동작업 지원
- 인간과 인간의 상호 작용
- 시각 Reasoning 지원
- 시각적인 정보 검색

이러한 문제점과 요구 조건을 해결하기 위하여 시각 프로그래밍 기술에 대한 이론적이며 기초적인 연구 또한 요구되어져야 한다.

2. 전 망

시각 프로그래밍 기술은 인간과 컴퓨터의 상호 작용의 새로운 차원을 이루는 기술이 됨으로써, 멀티미디어나 인간과 컴퓨터의 상호 작용 시스템에 효율적으로 사용되어질 수 있는 차세대 언어이다. 또한, 게임 세대(Game Generation)의 사용자가 주로 사용하게 될 차세대의 지능형 컴퓨터 시스템에서는 필수적으로 사용되어질 기술이다. 그러므로 많은 연구 분야

와 사용을 위하여 연구되어질 이유가 확실히 있으며, 전망이 매우 밝은 연구 분야이다.

현재의 소프트웨어는 주로 아이콘을 사용하여 처리하거나 개발하는 시스템으로 변화하고 있다. 아이콘 역시 복잡하고, 시간에 따라 어떤 특성을 지니는 아이콘의 형식으로 바뀌고 있으며, 비디오의 특성을 이용한 시간 중심의 새로운 아이콘도 개발 제시되어야 한다. 그외에도 과연 시각화가 효과적이며 사용자에게 좋은 시스템으로 적용되어질 수 있는가, 또는 시각화의 효과나 한계점은 무엇인가를 연구하여야 하는데, 다음과 같은 점은 특히 강조되어야 할 연구 이슈이다.

- 리커젼 개념의 시각화
- 멀티미디어의 추상화 자료형
- 비디오등 동적인 객체의 시각화
- 동적 객체의 자료형
- 제한된 화면의 크기
- 병렬 실행
- 공동 작업의 지원
- 인간과 인간의 상호 작용
- 문화와 시각화

이외에도 많은 분야가 있으며, 계속 연구되어져야 한다. 결국 시각 언어의 연구와 사용은 연구소, 학교, 일반 사용자에게까지 그 편리성이 강조되기 때문에 계속 확산되어질 것이다. 그러므로 이러한 시각 언어는 형식 언어, 그래픽, 교육 방법, 인지 심리학, 시각 디자인, 의학, 공학, 언어학등 각 분야에서 신중하게 연구되어지고 있으며, 그 전망이 매우 밝을 수밖에 없다.

사용자와 친숙한 시각 언어를 사용하는 시각 프로그래밍 환경을 인식시키기 위하여는 인간과 컴퓨터 사이의 통신 능력의 간격을 좁히는 것이다. 즉, 지금과 같이 사용자는 컴퓨터 중심의 개념과 프로그래밍 방법을 배우는데 힘을 기울일 필요가 없이, 진정으로 사용자와 친숙한 시각 프로그래밍 환경의 개발을 위한 방법으로 컴퓨터와의 대화를 즐겁게 할 수 있도록 하여야 한다. 이와같이 하여 차세대 컴퓨터의 기본 언어로서의 효과적인 시각 언어와 컴파일러 기술을 연구하여야 한다.

참 고 문 헌

1. Bove and Rhodes, Que's Machintosh Multimedia Handbook, p. 432, Que, 1990.
2. Claudia Crimi, Angela Guercio, Giuliano Pacini, Genoveffa Tortora, maurizio Tucci, "Automating Visual Language Generation", IEEE Transactions on Software Engineering, Vol. 16, No. 10, 1990, pp. 1122-1135.
3. Ephraim P. Glinert and Steven I. Tanimoto, "Pict: An Interactive Graphical Programming Environment", IEEE Computer, Vol. 17, No. 11, 1984, pp. 7-25.
4. Fred Lakin, "Spartial Parsing for Visual Languages", Visual Languages, edited by S. K. Chang et al., Plenum Pub. Co., 1986.
5. Genoveffa Tortora, Paolo Leonicini, "A Model for the Specification and Interpretation of Visual Languages", IEEE Proccedings Workshop on Visual Language, 1988, pp. 52-60.
6. J. F. Sowa, Conceptual Structures, p.481, Addison-Wesley, 1984.
7. Sangwook Kim, "Primitive Data Type and Operation of High-Level Language for Intelligent Computer", Proceeding of International Workshop on Intelligent Computer 1990, InCom '90, Daejeon, Korea, November, 1990, pp. 28-29.
8. Kenneth N. Lodding, "Iconic Interfacing", IEEE Computer Graphics and Application, Digital Equipment Corporation, Vol. 3, No. 4, 1983, pp. 11-20.
9. Khoros Group, Khoros Manual(Release 1.0), p. 264, University of New Mexico, 1991.
10. Masahito Hirakawa, Miniru Tanaka, Tadao Ichikawa, "An Inconic Programming System HI-VISUAL", IEEE Transactions on Software Engineering, Vol. 16, No. 10, 1990. pp. 1178-1184.
11. N. C. Shu, "Visual Programming : Perspective and Approache", IBM System Journal, Vol. 28, No. 4, 1989, pp. 525-547.
12. Shi-Kuo Chang, Michael J. Tauber, Bing Yu, and Jing-sheng Yu, "A Visual Language Compiler", IEEE Transaction on software Engineering. Vol. 15, No. 5, 1989, pp. 506-525.

13. Shi-Kuo Chang, "Visual Language : A Tutorial and Survey", IEEE Software, Vol. 4, No. 1, 1987, pp. 29-39.
14. Shi-Kuo Chang, The Principal of Visual Language, p. 372, Prentice-Hall International Editions, 1990.
15. Shi-Kuo Chang, "A Visual Language Compiler for Information Retrieval by Visual Reasonning", IEEE Transaction on Software Engineer, Vol. 16, No. 10, 1990, pp. 1136-1149.
16. Sueann Ambron and Kristina Hopper, Learning with Interactive Multimedia, p. 383, Microsoft Press, 1990.
17. Tadao Ichikawa and Masahito Hirakawa, "Iconic Programming : Where to Go?", IEEE Software, November 1990, pp. 63-68.
18. Ben Shneiderman, Designing the User Inteface : Strategies for Eeffective Hunman-Computer Interaction, Second Edition, Addision-Wesley Pub. Co, 1992.
19. Mark G. Sobell, A Practical Guide to Unix System V, Second Edition, The Benjamin/Cummings Pub. Co., 1991.
20. Ousterhout, J, "Tcl : An Embeddable Command Language", Proc. USENIX Winter Conference, January 1990, pp. 133-146.
21. Brad A. Myers, "Taxomomies of Visual Progamming and Program Visualization", Journal of Visual Languages and Computing, Vol. 1, 1990, pp. 97-123.
22. Haruo Takemura and Fumio Kishino, "Cooperative Work Environment Using Virtual Workspace", ACM 1992 Conference on CSCW, pp. 1992, 226-232.
23. Paul Resnick, "HyperVoice : A Phone-Based CSCW Platform", ACM 1992 Conference on CSCW, 1992, pp. 218-225.
24. Catherine G. Wolf and James R. Rhyne, "Communication and Information Retrieval with a Pen-based Meeting Support Tool", ACM 1992 Conference on CSCW, 1992, pp. 322-329.
25. Hiroshi Ishii and Naomi Miyake, "Toward An Open Shared Workspace : Computer and Video Fusion Approach of TeamWorkStation", Communications of the ACM, Vol. 4, No. 12, December 1991.
26. R. E. Newman-Wolfe, M. L. Webb, and M. Montes, "Implicit Locking in the Ensemble Concurrent Object-Oriented Graphics Editor", ACM 1992 Conference on CSCW, 1992, pp. 265-272.
27. Mark SteFik, Gregg Foster, Daniel G. Bobrow, Kenneth Kahn, Stan Lanning, and Lucy Suchman, "Beyond The Chalkboark Computer Suputer For Collaboration And Problem Solving In Meetings", Communications of the ACM, Vol. 30, No. 1, Jan, 1987, pp. 32-47.
28. Atul Prakash and Michael J. Knister, "Undoing Actions in Collaborative Work", ACM 1992 Conference on CSCW, 1992, pp. 273-280.
29. Brent Reeves and Frank Shipman, "Supporting Communication between Designers wigh Artifact-Centerd Evolving Information Spaces", ACM Conference on CSCW, 1992, pp. 394-401.
30. Ellis, C.A., Gibbs, S. J., Rein, G. L., "Groupware : Some Issures and Experiences", Communications of the ACM, Vol. 34, No. 1, January, 1991, pp. 680-689.
31. Paul Calder, "The Object-Oriented Implementation of a Document Editor", OOPSLA '92 Conference Preceedings, 1992, pp. 154-165.
32. Joel Richardson, Peter Schwarz, and Luis-Felipe, "CACL : Efficient Fine-Grained Protection for Objects", OOPSLA '92 Conference Preceedings, 1992, pp. 263-275.
33. John Bowers and Tom Rodden, "Exploding the Interface : Experiences of a CSCW Network", INTERCHI '93 Conference Preceedings of the Human Factors in Computing Systems, 1993, pp. 255-262.
34. William W. Gaver, Allan MacLean, Kathleen A. Carter, and William Buxton, "Realizing a Video Environment : EuroPARC's RAVE System", CHI '92 Conference Preceedings of Human Factors In Computing Systems, 1992, pp. 27-36.
35. 박용진, "CSCW(Computer Supported Cooperative Work) 연구 동향", 정보과학회지, 제9권, 제5호, 1991, 10.
36. 김상욱, 이춘희, "무엇이 객체 중심 프로그래밍 환경인가?," '92 한국인지과학회 춘계 학술대회논문집, 1992, 5, pp. 171-183.
37. 김상욱, 박지은, 김만수, 구경민, 서정민, 서호연, 이춘희, "객체 지향 프로그래밍을 위한 시각화 시스템", 정보과학회논문지, 한국정보과학회, 제20권, 제12호, 1993, 12, pp. 1773-1792.
38. 김상욱, 서정민, 서호연, 조창식, "객체 중심 시각 프로그래밍을 위한 객체 관리", 정보과학회논문

지, 한국정보과학회, 제21권, 제1호, 1994, pp. 169-206.
39. 김상욱, 조창식, "시각 명령 언어의 개발", 한국정보과학회 춘계학술발표회 논문집, 제21권, 제1호, 1994, 4, pp. 355-358.
40. 김상욱, 김정수, 안춘근, 진윤숙, "그룹 편집기에서의 객체에 대한 다중 접근의 제어", 한국정보과학회 춘계학술발표회 논문집, 제21권 제1호, 1994, 4, pp. 875-878.

김 상 욱

• 1979년 : 경북대학교 전산과학 학사학위 취득
• 1981년~1989년 : 서울대학교 전산과학 석사학위 박사학위 취득
• 1988년 : 경북대학교 자연과학대학 전자계산학과 부교수 재직
※ 주관심분야 : 컴퓨터 언어, 시각 언어 컴퓨팅, 객체 지향 컴퓨팅, 언어 정보와 지식 처리, 멀티미디어 컴퓨팅