객체 지향 C++ 클래스 생성을 위한 시각 프로그래밍 도구

하 수 철†

요 약

본 논문은 전문가 또는 비전문 개발자에게 문제 영역의 물리적 세계를 쉽게 파악하는 능력을 제공하고, 아이콘 및 기호의 사용을 통하여 사용자 편리성을 갖고 조작할 수 있는 시각 프로그래밍 도구에 관한 연구이다. 특히 중점을 둔 것은 C++ 객체 지향 소프트웨어 생성을 위한 시각적인 접근이다. 이로써 초보자도 C++ 객체 지향적인 특성을 점진적으로 이해할 수 있으며, C++의 클래스를 용이하게 작성할 수 있게 한다. 이를 위해 시각적인 은유법을 도입하는데 테이블 형태로 객체와 클래스들을 표현하면서 그 자체가 아이콘으로 간주되도록 테이블 아이콘을 제안한다. 수퍼 테이블 아이콘, 중간 테이블 아이콘, 그리고 세부 테이블 아이콘의 3 단계 테이블 아이콘을 설정함으로써 객체 지향 패러다임에 적합한 개념 진화를 도모할 수 있다. 이들 테이블 아이콘은 단순한 상형문자의 아이콘이 아니라 활성화가 가능하여 테이블 형태로 확장되며, 그 내부에 원하는 엔티티들을 삽입하거나 불필요한 엔티티들을 삭제할 수 있는 능력을 가진다. 이 테이블 아이콘들은 C++ 프로그램 설계와 구현을 위해 고안된 C++gram[18]이라는 다이아그래밍 기법에 적용된다.

A Visual Programming Tool for Constructing Object-Oriented C++ Class

Soo Cheol Ha†

ABSTRACT

This paper describes a visual programming tool which provides novice programmer as well expert with the abilites to capture real physical world of problem domain and to manipulate it user-friendly using icons and symbols. Therefore, novice can understand object-oriented features of C++ incrementally and construct classes easily. For this, we introduce some visual metaphors which are displayed as tables. The tables can not only represent objects and classes, but also be considered themselves as icons. We have named these tables as table-icons. Three levels of table-icons (i.e., Super Table-Icons, Intermediate Table-Icons and Detailed Table-Icons) are proposed to follow up appropriate evolution of object-oriented concepts. Table-icons are not simple pictographs but are activated and expanded to table forms. And then, developer can insert necessary entities into table body and delete useless entities from it. These table-icons are applied to a diagramming technique, C++gram [18], which is suggested for designing and implementing C++ programs.

1. 서 론

일반적으로 프로그래밍은 전문가에게 조차도 복잡하며 시간 소비성 작업으로 간주된다. 이것은 신뢰성 있는 프로그램의 효율적인 생성을 위해 프로그래밍 언어의 향상과 여러 가치 있는 프로그래밍 언어 특성들이 증강되어야 함을 의미한다. 이들 특성들 중에는 자료 추상화, 모듈화, 병행 프로그래밍, 사용자 정의 자료형, 형 대조 및

· 이 연구는 부분적으로 한국학술진흥재단의 지방대 육성학술연구 조성비의 지원('91년)과 한국과학재단 해외연수지원('92년)에 의한 결과임.

† 정 회 원 : 대전대학교 컴퓨터공학과 부교수

논문접수 : 1994년 8월 30일, 심사완료 : 1994년 12월 12일

예외처리 등이 포함된다. 고급 프로그래밍 언어에 대한 최근의 연구들은 함수 프로그래밍, 논리 프로그래밍, 그리고 객체 지향 프로그래밍과 같은 새로운 프로그래밍 패러다임에 기반을 두고 언어 개발에 기여하는 방향으로 추진되고 있다[1].

전통적인 컴퓨터 시스템은 내부 상태를 사용자에게 거의 보여주지 않고 있다. 이 때문에 사용자는 컴퓨터에서 내보내지는 약간의 상태표시만으로 대규모의 복잡한 시스템을 조작하고 추측한다. 선형, 기호적 컴퓨터 언어들은 과거 30여년동안 연구되어 광범위한 정제를 거쳐왔지만 이제 컴퓨터 언어 설계자들에게 직면한 것은 편리하고 자연스러운 시각 프로그래밍 언어(visual programming language)의 제공이라는 도전이다[2]. 소프트웨어 개발 환경의 한 부분을 차지하고 있는 프로그래밍 환경은 단일 프로그래머의 프로그래밍 과정에 직접적인 연관이 있다. 지난 수십년 동안 프로그래머의 생산성 증대를 위한 프로그래밍 언어의 개발 연구가 계속되었으나 그 잠재력은 거의 소모되었다. 소프트웨어 생성을 위해 더 새롭고 생산적인 접근 방법은 컴퓨터를 향하여 인간의 요구사항을 표현할 수 있고 인간의 능력을 극적으로 향상시켜 주어야 하는 방향으로의 전환점을 맞고 있다. 자연어에 근접한 언어를 창조하는 것을 제외한 가장 유력한 접근법이 그래픽적 언어의 창조이다[19]. 즉, 각 분야의 전문가적인 입장에서 컴퓨터를 사용하는 사람들이 현대적인 프로그래밍 분야의 전문 지식을 추가로 습득하지 않아도 될 뿐만 아니라, 초보 사용자도 복잡한 환경에 관한 세부 지식을 알지 않고서도 그래픽 워크스테이션에서 프로그램이 가능한 새로운 언어를 개발하는 것이다. 이렇게 하여 사용하기 쉬운 컴퓨터 환경을 획득하게 된다면 생산성면에서 뿐만 아니라 컴퓨터를 통해 얻어지는 인간 심리적 복지 향상 측면에서 중요한 요인이 될 것이다[17].

본 논문은 전문가 또는 비전문 프로그래머에게 실제 물리적 세계를 쉽게 파악하는 능력과 아이콘(icon) 및 기호의 사용을 통하여 자연스럽게 대상 컴퓨터 시스템을 조작할 수 있는 능력을 제공하는 소프트웨어 개발 환경에서의 시각프로그래밍에 관한 연구이다. 특히 중점을 둔 것은 다이아그래밍 기법에 기반하여 C++ 객체 지향 소프트웨어 생성을 위한 시각적인 접근으로서 초보자가 C++객체 지향적인 특성을 점진적으로 이해하면서 C++의 클래스(class)를 용이하게 작성하도록 하는 프로그래밍 도구의 구축이다. 본 논문의 제2장에서는 시각 프로그래밍 환경들을 분석하고 본 논문이 목표로하는 시스템의 시각화 과정 및 연구 배경에 대해 기술한다. 제3장에서는 목표로 하는 시각 프로그래밍 환경의 구성을 위한 시각 원소의 선정 정책을 정의하고 시각적 은유 모델을 설정하고 그에 따른 방법론을 논한다. 또한 제4장에서는 프로그래밍 도구의 내부 구조를 정의한다. 끝으로 제5장에 본 논문의 평가 및 결론을 기술한다.

2. 시각 프로그래밍

시각 프로그래밍은 "그림의 도움에 의한 프로그래밍", "컴퓨터에게 일련의 연산을 반복하도록 가르치는 프로그래밍" 으로 새로운 연구 분야의 이론적인 특성들을 다루는 원리들의 결합체이다[4]. 또한 사용자가 이차원 이상의 방식으로 프로그램을 기술할 수 있는 모든 시스템을 말한다[3].

시각 프로그래밍 연구는 프로그래밍의 재래식 문장 체계와는 달리 시각적 그래픽 체계에서 프로그래밍 하는 방안을 탐구한다. 그래픽스, 이미지, 비디오, 마이크로 전자 기술의 발전으로 저가의 이미지 정보시스템을 생성, 전달, 표시, 조작하도록 설계하는 것이 가능하게 되어, 사용자가 시스템과 상호 작용을 통하여 프로그램을 만들 수 있게 되었다. 이러한 상호 작용을 광의로 시각 언어(visual language)라고 부르기도 하지만, 시각 언어라는 용어는 여러 연구자들간의 해석이 일치되지는 않는다. 광의로 해석하면 1)언어에

의해 조작되는 객체가 시각적이라는 의미와 2)언어 자체가 시각적이다라는 의미로 구분할수 있다. 시각 언어에서 시각화되는 정보는 1)화일, 자료, 프로그램등의 객체, 2) 프로그램의 알고리즘, 3)자료 구조등이다. 분류1)에 해당되는 예로는 Macintosh 시스템이 있으며 아이콘으로 객체를 시각화 한다. 분류2)에 해당되는 예는 참고 문헌[16]이 그 예이다. 폼(form) 또는 그래픽스로써 자료구조를 시각화하는 예는 참고문헌[8, 15]이 그예로써 분류3)에 해당한다. 본 논문의 경우는 분류1)중에서 프로그램 생성을 시각화하고 있다.

시각 프로그래밍 환경으로는 Xerox Lisp 기계상의 Smalltalk-80으로 구현된 Programming-by-Rehearsal을 예로 들 수 있다. 이 Rehearsal World는 완전한 시각 프로그래밍을 지원하는 초기 환경 중의 하나로서 비프로그래머용의 시각 프로그래밍 환경이다[9, 12].

Rehearsal의 세계는 다수의 미리 정의된 기본 실행자를 갖고 있으며, 각각은 대규모의 미리 정의된 단서의 집합을 이해한다. 실행자라고 부르는 기본 컴퍼넌트는 단서를 보냄으로써 스크린상에서 각각 다른 것과 상호작용을 한다. Rehearsal 세계에 의해 생성되는 실제 코드는 Smalltalk이지만 프로그래밍 과정은 그래픽적이거나 시각적으로 지향된 조작을 통해 발생한다. 따라서 사용자는 프로그램하기 위해 Smalltalk를 알 필요가 없다. ThinkPad[13] 상에서의 프로그래밍은 프로그램이 해야 할 사항을 컴퓨터에 시연(demonstration)하면서 동시에 스크린 상에서 그래픽적으로 자료를 조작함으로써 수행한다. 프로그래머에게는 프로세스를 실행할 방법을 문장(text)형태로 기술하는 것보다 직접 프로세스를 실행하는 것이 더 쉽게 여겨지도록 한다. 아이콘 언어(iconic language)는 아이콘을 광범위하게 사용하거나 독립적으로 사용하는 시각 언어로서, 상형문자(pictograph)는 관련된 아이콘의 구조화된 집합으로 정의되며, 아이콘 문장은 특별한 언어학상의 규칙에 따르는 정형화된 아이콘 구조로 정의된다. 또한 구술적인 방법으로 사상 또는 행위를 전달하는 이미지 사용에 관점을 둔다. 아이콘 언어로는 PLAY, VICON, HI-VISUAL을 예로 들 수 있다. PLAY[5]는 아이들을 위한 아이콘 프로그래밍 시스템으로 희극적인 문자들의 설계와 이야기 및 실행되는 애니매이션의 뷰(view)를 구성하는 것이 가능하다. VICON[5]은 아이콘 구조의 여러 기본 연산을 제공하는데, 그것은 아이콘 관리, 아이콘 편집, 아이콘 해석 등이다. 실행되는 계산과정과 동일하게 응용 도메인의 객체들을 표현하기 위하여 아이콘을 사용한 것이 HI-VISUAL[5, 6, 14] 언어이다.

시각언어의 분류를 요약하면 (그림 1)과 같다[5, 6].

시각언어의 분류	조작 대상	언어의 가시성
시각작용 지원언어	시각표현법을 가진 논리적 객체	선행으로 표현된 구문
시각프로그래밍 언어	시각표현법을 가진 논리적 객체	시각적으로 표현된 구문
시각 정보처리 언어	부과된 논리적 표현법을 가진 시각적 객체	선행으로 표현된 구문
아이콘시각정보처리 언어	부과된 논리적 표현법을 가진 시각적 객체	시각적으로 표현된 구문

(그림 1) 시각 언어의 분류
(Fig. 1) Classification of Visual Languages

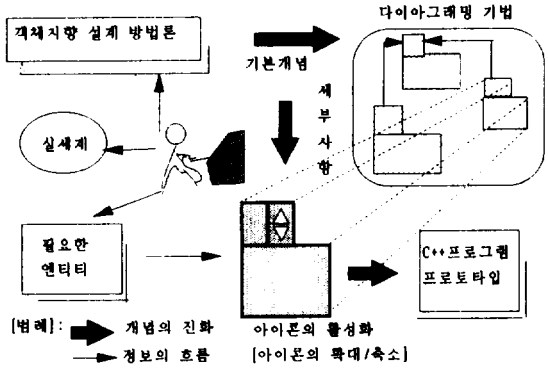
(그림 1)의 기준에 따르면, 소프트웨어 설계 및 구현에서의 그래픽(iconic)표현법인 DFD (Data Flow Diagram)[22], Boochgram[23], Object Diagram[24]등은 시각 작용을 지원하는 언어로 간주된다. 한 예로서 자료 흐름 패러다임에 기반을 둔 언어는 먼저 DFD를 프로그램 설계 과정의 일부로 간주하고 그후 텍스트 구문으로 번역한다. 이 개념이 본 논문의 주제로 도입된다.

C++를 생성하기 위한 시각 프로그래밍 환경에 대한 연구로 ObjectCraft[9], 문헌[10, 11] 등의 연구가 있는데 이들 연구는 시각 화면으로부터 C++ 코드를 생성하고 있지만 시각 화면 자체가 프로그램 설계 단계에서 사용된 다이아그램을 그대로 구현 단계로 유지하는 것이 아니라는 점이 본 논문의 관점과 다르다.

3. 시각 프로그래밍 도구의 구성

3.1 시각화 과정

본 논문의 시각 프로그래밍 도구는 객체 지향 소프트웨어 생성을 위해 설계에 사용된 다이아그램(diagram)으로부터 시각적 기법의 도움을 얻어 프로그램 구문으로의 변환을 목표로 삼는다. (그림 2)는 목표 시스템의 시각화과정을 보인다 [21].



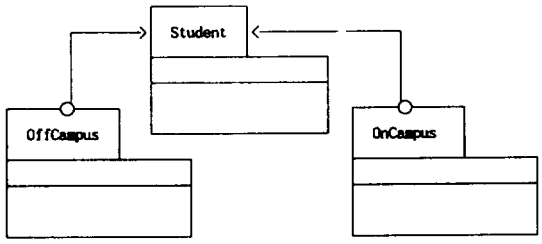
(그림 2) 시각화 과정
(Fig. 2) Process of Visualization

객체 지향 설계의 그래픽적 표현은 매우 유용하고 객체에 대한 다이아그램은 객체 지향 패러다임을 이용하여 소프트웨어를 생성하기 위한 사람들에게 큰 도움을 제공한다. 즉,이것은 객체 지향적 계산의 관점에서 본다면 객체들 간에 수행되는 대화를 명확하게 표현할 수 있다는 의미이다[18]. (그림 2)는 다이아그램 자체를 하나의 큰 아이콘으로 간주하여 그 아이콘이 개념 확대에 의해 진화되거나 변신할 수 있음을 나타내고 있다. 확대된 아이콘은 그 내부에 프로그램 단위(여기에서는 클래스를 의미)가 될 필요한 애트리뷰트(attribute)들을 기재할 수 있다. 여기에서 사용하는 다이아그램은 C++gram으로 다음 사항을 만족하기 위해 제안된 C++를 위한 다이아그래밍 기법이다[18].

① 상위 추상화 레벨로부터 구현 세부 사항까지를 표현할 수 있을 것.
② 클래스 중심의 철학이 부각될 것.
③ 전위(front-end) CASE 도구 구현시를 고려할 것.

이 조건들에 덧붙여 클래스 생성을 위한 시각적 은유법을 사용한 시각 프로그래밍 구성하고 있다.

(그림 3)은 C++gram의 예이다.



(그림 3) 클래스의 관계성 표기 예
(Fig. 3) Example of Class Relationship

이 C++gram의 의미론은 요약하면 다음과 같이 정의된다[18].

3.1.1 일반화

유사한 객체들의 가계(families)나 'a kind of', 'is a', 'category' 또는 'specialization' 관계를 일반화(generalization)라 하는데 이를 C++gram으로 표기할 수 있다. CT1……,CTn(n≥2)을 C++의 클래스 형이라 하면, 수퍼 클래스 CT1과 부속 클래스 CT2…, CTn은 부분 집합 관계를 나타내는 일반화가 된다. 이 일반화의 개념은 1) 특수화(specialization), 2) 제한화(restriction)로 간주할 수도 있다. 즉, 수퍼 클래스 보다 더 특징적인 서술을 필요로 할 때 수퍼 클래스로부터 부속 클래스들을 얻을 수 있다. 이러한 개념을 C++gram은 표기할 수 있다.

3.1.2 집단화

클래스 형을 CT1,…,CTn(n≥2)이라 하자. 집단형(aggregate-type) CT1과 성분형(component-type) CT2,...,CTn을 가진 컴포넌트 관계를 집단화(aggregation)로 정의하면, 객체들의 그룹이나, 'a part of', 'has a part' 관계가 표현되며 이 개념을 C++gram으로 표기할 수 있다.

3.1.3 관련화

CT1, CT2를 클래스의 형이라 하고 집합형(set-

type) CT1과 멤버형(member-type) CT2가 멱집합(powerset) 관계를 나타낼 때를 관련화(association)라 하자. 즉, 클래스 형 CT1 의 각 객체는 클래스 형 CT2의 객체들의 집합이다. 이 개념을 C++gram으로 표기할 수 있다.

3.2 시각 모델의 설정

시각 프로그래밍은 사용자가 만든 그림과 동작의 조작으로부터 어떤 컴퓨터의 언어 또는 프로그램으로 직접 번역하여 실행이 가능한 코드를 생산하는데 사용 될 수 있다. 이때에는 분명한 구문과 의미를 요구하는데 언어의 적용 필드가 광범위한 분야에 적용되는 컴퓨터 프로그래밍과는 달리 적은 부분으로 제한되지 않으면 안된다는 제약을 갖게 된다[4].

3.2.1 시각 원소와 선정 정책

프로그램 생성과정을 시각적으로 표현하기 위한 시각적 인터페이스를 도입하려면 시각적 커뮤니케이션의 기본을 구성하는 세가지 원소인 의미론, 구문론, 실용론에 입각한 정의가 필요하다[7]. 이에따라 설정된 원소의 특성은 다음과 같다[19].

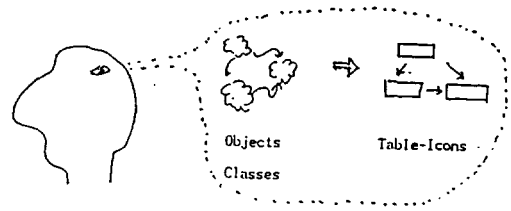
① 의미론 : 시각 이미지의 의미는 쉽게 이해되어야 한다.
- 시각 이미지는 기억하기 쉬워야 한다.
- 설정된 이미지 범주는 일관성을 가져야 한다.
- 이미지는 서로 관련성이 높게 설계하여야 한다.

② 구문론 : 각 이미지는 다른 것들과 일관된 연관성을 가져야 한다.
- 시각 원소는 각 응용부문 또는 특정 의미를 위한 그룹화가 가능해야 한다.
- 시각 원소는 모양, 크기, 색깔, 농도, 묘사된 품질, 조합, 행동, 전반적 상영등을 포함해야 한다.
- 가장 중요한 이미지는 현저하게 나타내야 한다.
- 시각 이미지 흐름은 작업 흐름에 적합해야 한다.

③ 실용론 : 각 이미지는 사용자와 그의 습관에 적합해야 한다.
- 시각 이미지는 사용 범위내의 어느 곳으로부터도 가시적이어야 한다.
- 시각 이미지는 확대나 축소시 형체를 망가뜨리지 않아야 한다.
- 진행시 작업을 중단시키지 않아야 한다.
- 보기에 싫증이 나지 않아야 한다.

이들 원소에 입각하여 본 논문에서 고안할 시각적 은유를 위해 선정하는 각 시각 원소의 선정 정책은 다음과 같다.

① 모양(아이콘)
- 툴(tool)은 설계를 지원하기 때문에 설계기법과 컴퓨터에 관련된 친숙한 이미지가 활용된다.



② 크기
- 표시되는 윈도우 크기는 한번에 판독될 수 있도록 한다.
- 반전되는 메뉴, 아이콘, 윈도우 및 다른 항목은 굵은 문자로 제공 되고 눈에 잘 띄도록 강조된다.

③ 조합
- 디스플레이시 두드러지는 항목은 일관성과 식별성을 유지하기 위해 10개 이내로 제한한다.

④ 밀도
- 인간공학적 관점으로부터 쉽게 포착할 수 있는 정보의 최적량(테이블 밀도, 최대문자 등)은 디스플레이, 윈도우 등의 한정된 영역에서 설정된다.
- 중요도를 강조하는 부분은 더 높은 밀도

로 보여준다.

⑤ 행동

- 작업은 팝업(pop-up), 깜박거림(blinking) 등의 시각적 트릭을 사용 한다.
- 사용자의 관심이 집중되어 있는 곳(보통 커서위치)과 다음 작업이 발생하는 곳이 떨어져 있다면 시각적 트릭을 사용하여 강조한다.

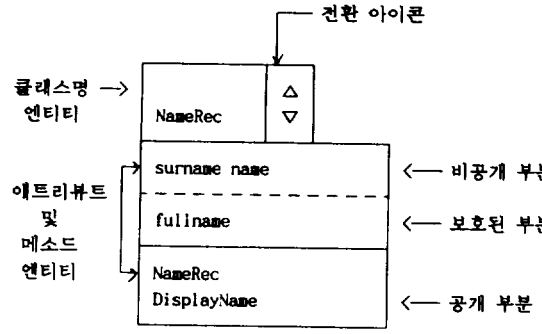
3.2.2 시각적 은유 모델 설정

(그림 2)에서 볼 수 있듯이 객체 지향 설계 방법론을 사용하여 다이아그램들로부터 각각의 클래스로 완성하기 위한 세부적인 애트리뷰트들과 메소드(method)들을 입력할 때 적합한 시각적 은유법(metaphor)이 필요하다. 본 연구의 시각적 프로그래밍 환경의 대상을 객체 지향 프로그래밍 언어 C++ 클래스 생성 환경으로 삼고 있다. 따라서 C++를 위한 다이아그램(C++gram)의 시각적인 조작과 사용자와의 친밀한 인터페이스 제공에 적합한 시각적 은유법이 필요하다. 다이아그래밍 툴의 화면상에는 클래스간의 관계를 나타내는 다이아그램들이 표시되므로 이 다이아그램을 테이블(table) 또는 레이블 붙은 테이블로 간주한다.

테이블 자체가 특정 상형문자(pictograph)로의 의미에 꼭 부합되지는 않지만 일종의 아이콘으로 간주할 수 있다. 하나의 프로그램을 완성하기 위해 여러 클래스들이 결합될 수 있으므로 여러 테이블의 등장도 가능하기 때문이다. 이 테이블 이미지는 3.2.1절의 의미론, 구문론, 실용론에 적합하다. 이를 테이블 아이콘(Table-Icon)이라 명한다. 테이블 아이콘의 사용자에게는 아이콘 간의 계층적 관계유지, 세부 묘사를 다단계로 정의하는 것, 다이아그래밍 기호의 상호 연결 계층을 정의하는 능력을 부여하기 위하여 세가지 종류의 테이블 아이콘으로 나누어 이를 해결한다. 이 테이블 아이콘은 수퍼 테이블 아이콘(Super Table-Icon), 중간 테이블 아이콘(Intermediate Table-Icon)과 세부 테이블 아이콘(Detailed Table-Icon)으로 구성되어 계층구조를 형성한다[20, 21].

(1) 수퍼 테이블 아이콘

수퍼 테이블 아이콘은 사용자가 클래스를 추상적으로 구성 할 때 사용되며 클래스명의 엔티티 부분, 애트리뷰트 부분, 확장 기능의 아이콘으로 나누어진다. 또한 애트리뷰트 부분은 사용자 의도에 의해 외부로부터의 접근을 금지하는 비공개 부분(private part)과 현재의 자식관계(children relationship)에 해당하는 유도된 부속클래스(derived subclass) 내부를 통하여는 접근이 허용되는 보호된 부분(protected part), 그리고 외부로부터의 접근을 개방하는 공개부분(public part)의 3가지로 구성한다. (그림 4)는 수퍼 테이블 아이콘의 구성도를 보인다.



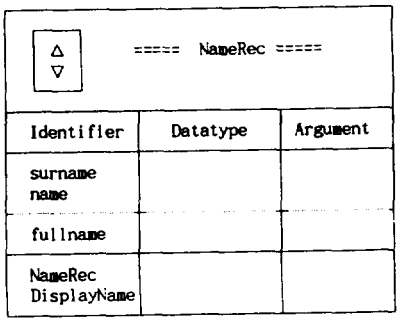
(그림 4) 수퍼 테이블 아이콘
(Fig. 4) Super Table-Icon

이 수퍼 테이블은 사용자의 요청에 의해 중간 테이블로의 변경이 가능하며 테이블 내의 애트리뷰트의 각 원소들은 마우스나 화살표 키에 의해 원하는 내용을 지정할 수 있다. 수퍼 테이블로부터 중간 테이블로의 전환은 전환 아이콘 위치의 마우스 클릭으로 이루어 진다. 두가지 형태의 아이콘(수퍼 테이블 아이콘 내의 전환 아이콘)은 선택할 대상의 방향 및 행동의 변경자(modifier)의 역할을 한다. 어느 한가지의 선택을 하면 새로운 테이블 아이콘이 기존의 내용을 유지한 채 새로 생성되어 개방된다. 즉 방향에 따라 상향식(bottom-up)행동을 취할 것인지 하향식(top-down) 행동을 취할 것인지가 결정되도록 해석된다.

(2) 중간 테이블 아이콘

중간 테이블 아이콘은 수퍼 테이블 아이콘에서 하향으로의 방향 지정에 의해 자동적으로 생성된다. 그 내용은 수퍼 테이블의 내용에 의존한다. 한편 중간 테이블 아이콘의 엔티티를 추가하거나 삭제하고 상향으로의 방향 지정을 하면 수퍼 테이블 아이콘의 엔티티들은 변경이 일어난다.

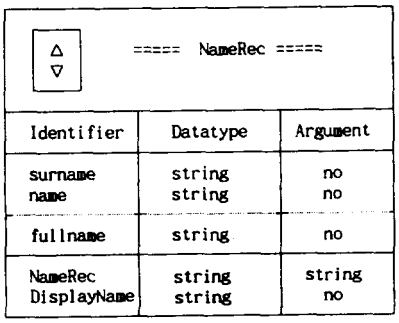
(그림 5)는 (그림 4)로부터 자동 변환된 중간 테이블 아이콘과 생성된 엔티티들을 보여준다.



(그림 5) 중간 테이블 아이콘
(Fig. 5) Intermediate Table-Icon

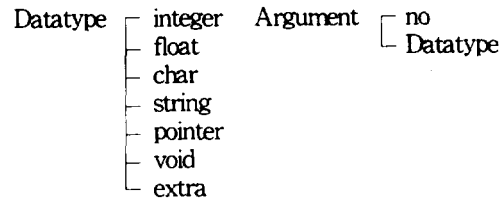
수퍼 테이블 아이콘으로부터 자동 생성되는 것은 중간 테이블 아이콘의 식별자(identifier)란의 엔티티로서 자료형(datatype)란과 인수(argument)란의 엔티티들은 커서나 마우스를 이동하면서 해당 애트뷰트들을 삽입하게 된다.

(그림 6)은 애트리뷰트들이 삽입된 중간 테이블 아이콘의 예이다.



(그림 6) 중간 테이블 아이콘의 애트리뷰트 삽입
(Fig. 6) Insert of attributes into Intermediate Table-Icon

커서나 마우스를 자료형란이나 인수란에 위치하면 자료형란이나 인수에 해당되는 선택메뉴가 자동적으로 나타난다. 그 메뉴는 다음과 같다.

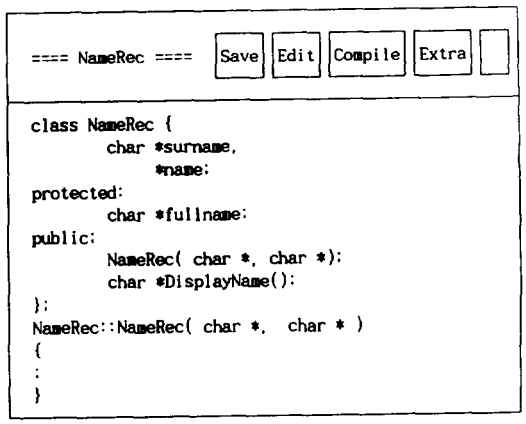


인수가 여러 개 존재한다고 판단되어 엔티티를 삽입하면 해당 갯수만큼 중간 테이블 아이콘의 튜플(tuple)단위의 확장이 발생한다.

(3) 세부 테이블 아이콘

중간 테이블 엔티티가 완성되고 난 후 하향 전환 아이콘을 누르면 세부 테이블 아이콘으로 변환되는데 이 세부 테이블 아이콘의 엔티티는 중간 테이블 아이콘의 엔티티 내용에 의한 C++프로그램의 프로토타입(prototype)을 생성한다.

(그림 7)은 (그림 6)에 의해 세부 테이블 아이콘 내에 생성된 C++ 프로그램의 프로토타입을 보여준다. 이 세부 테이블 아이콘은 그 자체가 통합 편집 환경을 유지하므로 추가 편집, 컴파일, 저장등의 조작을 할 수 있다.



(그림 7) 세부 테이블 아이콘
(Fig. 7) Detailed Table-Icon

이 테이블 아이콘들에 도입된 시각적 원소 선정 정책은

- 테이블 아이콘의 표제는 디스플레이 되는 클래스의 종류를 의미한다.
- 선택된 테이블 아이콘은 색깔 또는 색조를 바꾸어 표시된다.
- 테이블 아이콘 내의 튜플은 윈도우(테이블 아이콘)와 비례하여 확대 또는 축소가 가능하도록 한다.
- 세부 테이블의 메뉴 중 사용빈도가 낮거나 세부작업용 메뉴는 별개로 설정한다.

이다.

3.3. 시각 프로그래밍 방법론

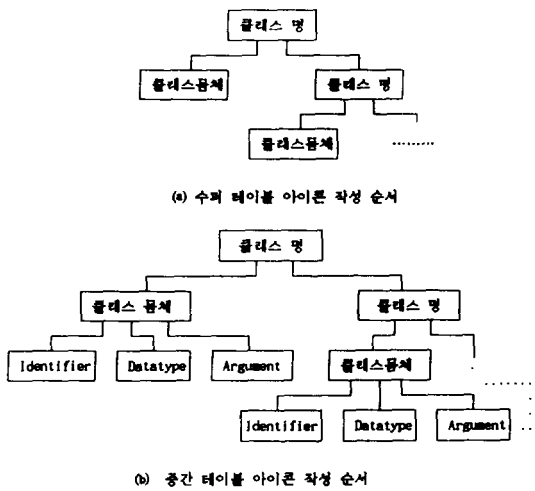
클래스를 기반으로 하는 C++ 객체 지향 프로그램을 시각적으로 작성을 위하여 클래스를 선정하고 그 컴퍼넌트를 효과적으로 완성시키는 단계는 다음과 같이 정의한다[20].

1) 소프트웨어 시스템의 요구 사항(또는 사용자의 요구사항)을 인간이 영유하는 조직의 개념으로 파악하여 문제의 공간을 이해한다.
2) 유사한 객체(object)들의 가계(family)나 일반적인 성질을 보유한 구조, 위치, 사건(event), 다른 소프트웨어 시스템, 별개의 역할, 또는 조직의 단위 들을 찾아 낸다.
3) 객체의 인스탄스(instance) 또는 분류구조를 기술하는데 사용되는 자료 원소인 애트리뷰트를 식별한다.
4) 이들 단계를 3개의 테이블 아이콘 작성에 적용하여 클래스를 완성한다.

그러나, 객체 지향 패러다임에 이해가 깊지 않은 사람들은 이 개발 절차를 따르는 것이 쉽지는 않을 것이다. 테이블 아이콘의 운행(traversal)은 전통적인 개발 방법인 하향식(topdown), 상향식(bottomup) 모두에 적용이 가능하므로 객체 지향 패러다임에 익숙하지 않은 사람에게도 효과적인 도구가 될 수 있다.

(그림 8)은 테이블 아이콘을 이용한 개발 순서를 보여준다.

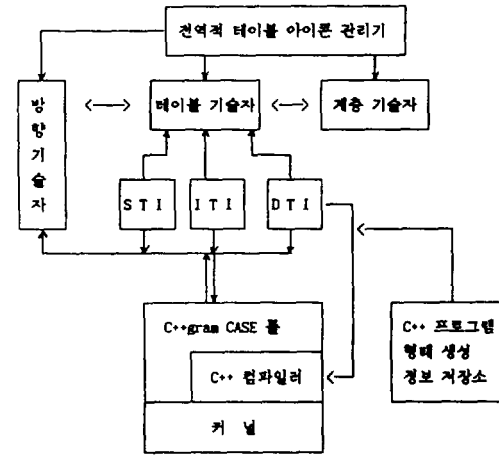
하향식 개발은 (그림 8)의 트리(tree) 구조에서 전위순 운행법(preorder traversal)으로 진행될 수 있고, 상향식 개발은 후위순 운행법(postorder traversal)으로 추진될 수 있다.



(그림 8) 테이블 아이콘을 이용한 개발 절차
(Fig. 8) Development Procedure Using Table-Icons

4. 시각 프로그래밍 도구의 내부 구조

본 연구에서 목표한 시스템은 크게 나누어 일련의 테이블 아이콘들을 관리하기 위한 디스플레이 공간 관리기인 전역적 테이블 아이콘 관리기(Global Icon Manager)와 수퍼 테이블 아이콘

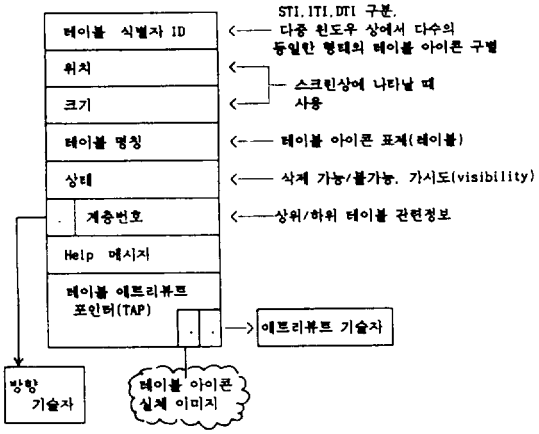


(그림 9) 시스템 구성도
(Fig. 9) System Configuration

(STI),중간 테이블 아이콘(ITI),세부 테이블 아이콘(DTI) 내부에 생성되는 엔티티 및 애트리뷰트들을 관리하는 기술자 (descriptor)로 구성된다.

4.1 테이블 기술자

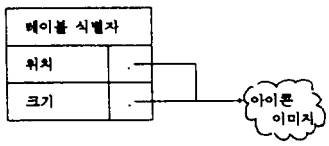
테이블 기술자(Table Descriptor)는 아이콘을 테이블 형태의 프레임으로 보여주기 위한 정보를 관리하는 것으로 STI,ITI,DTI를 구분해주며, 다중 윈도우상에서 다수의 동일한 형태의 테이블을 구별한다. 이 TD의 구조는 다음 그림과 같다.



(그림 10) 테이블 기술자
(Fig. 10) Table Descriptor

4.2 방향 기술자

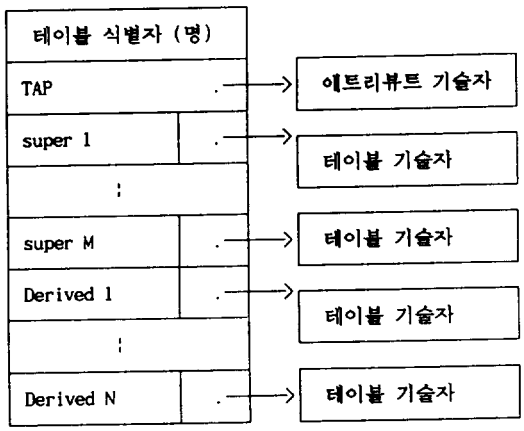
방향 기술자(Direction Descriptor)는 STI, ITI, DTI 간의 하향 또는 상향 상태 변환 정보를 관리한다. 이 DD 의 구조는 (그림 11)과 같다.



(그림 11) 방향 기술자
(Fig. 11) Direction Descriptor

4.3 계층 기술자

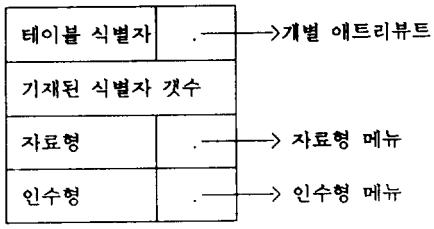
계층 기술자(Hierarchy Descriptor)는 테이블 아이콘간의 계층 관계를 관리한다. HD 구조는 (그림 12)와 같다.



(그림 12) 계층 기술자
(Fig. 12) Hierarchy Descriptor

4.4 애트리뷰트 기술자

설정된 클래스와 관련된 메소드, 애트리뷰트의 정보를 관리하는 애트리뷰트 기술자(Attribute Descriptor) 구조는 (그림 13)과 같다.



(그림 13) 애트리뷰트 기술자
(Fig. 13) Attribute Descriptor

5. 결 론

컴퓨터 응용시 특정 분야의 전문 사용자라 할지라도 현대적인 그래픽스 기반의 컴퓨터에 친숙해 있지 못하거나 전통적인 프로그래밍 언어가 텍스트 기반 구문을 갖고 있기 때문에 프로그램을 작성하기가 어려운 경우에 직면하게 된다. 전통적인 방법으로는 컴퓨터 사용 욕구를 충족시키는데 한계가 있는데, 그 이유는 인간의 생각을 컴퓨터로 표현하기 위한 직접적인 방법이 부족하기 때문이다. 인간은 그래픽적 시각 표현에 쉽게

친숙해지며 친숙한 만큼 그 표현을 자주 사용하게 된다.

시각 프로그래밍과 프로그램 시각화라는 용어는 부분적으로 프로그래머가 소프트웨어 및 그들이 구축한 프로그램과 상호 작용을 하는데 새로운 방법의 정신을 획득하게 해준다. 즉 텍스트만의 표현 방법이 아닌 시각 및 그래픽 수단을 통하여 소프트웨어를 다루는 능력을 얻게 된다.

소프트웨어 설계 명세가 다이아그램 기법으로 표현되었다고 해도, 표현된 다이아그램들 자체를 직접 이용하여 프로그램으로 구현하지 못하는 일반적인 객체지향 다이아그래밍 기법들의 적용 방식과는 달리 본 논문의 방식은 직접 다이아그램 내부를 사용자가 시각적으로 조작하는 하나의 동일한 작업대(workbench)에서 실행시킬 수 있다. 즉 DFD[22], Boochgram[23], Object Diagram [24] 등이 CAD 툴을 이용하여 운용되는 방법이 아닌 다이아그램 자체가 활성화 되는 방법을 사용한다는 점이 다른 기존의 방법들과 다르다.

본 논문에서 제시하는 테이블 아이콘은 단순한 상형문자의 아이콘이 아니라 활성화 되는 테이블 형태로 확장되며, 그 내부에 원하는 엔티티들을 삽입하거나 불필요한 엔티티들을 삭제할 수 있는 능력을 가진다. 물론 상향/하향의 의미를 가지는 일반적인 방향 아이콘을 테이블 아이콘의 내부 아이콘으로 포함하고 있다.

수퍼 테이블 아이콘의 모체는 C++를 위한 객체지향 다이아그래밍 기법으로 제안된 C++-gram[18]로부터 시작되었다. 수퍼 테이블 아이콘으로부터 개념의 진화를 위해 중간 테이블 아이콘이 도입되었으며, 중간 테이블 아이콘으로부터 C++클래스의 프로토타입이 자동 생성되고, 엔티티의 추가를 가능케 하는 세부 테이블 아이콘이 편집 환경처럼 조작될 수 있다. 이 3단계의 개념 진화는 세부 사항을 나중으로 미루는 객체 지향 패러다임에 적합하여 비전문 사용자가 객체지향 패러다임의 개념을 습득하는데 용이함을 제공받게 된다. 또한 전문 사용자는 개념 진화는 물론 엔티티,애트리뷰의 추가, 삭제에 따른 피드백(하향식 개발의 반대개념)을 시각적으로 할 수 있어 단순한 다이아그램으로부터 프로그램을 생성할 수 있고 이를 평가한 후 역으로 다이아그램(기본 설계 골격)의 수정에 도움을 얻게 된다.

앞으로의 과제는 본 논문이 제안하는 환경의 추가적인 구현이며 클래스의 작성뿐만 아니라 C++ 프로그램의 논리를 시각적으로 표현하는 방법에 관한 연구가 계속 되어야 하겠다.

참 고 문 헌

[1] T. Ichikawa and M. Hirakawa, "Visual Programming-Toward Realization of User-Friendly Programming Environments", Preceedings 2nd Fall Joint Computer Conference, pp. 129-137, 1987.

[2] A. L. Ambler and M. M. Burnett, "Influence of Visual Technology on the Evolution of Language Environments, IEEE Computer, pp. 9-22, Oct. 1989.

[3] B. A. Myers, "The State of the Art in Visual Programming and Visualization : Graphics Tools for Software Engineers, Eds. A. Kilgour and R. Earnshaw, Cambridge Univ. Press, 1989.

[4] S. Dahl and K. Lindgvist, "Visual Programming as an Interface Between Program and User?", 1989 IEEE Workshop on Visual Languages, pp. 18-22, 1989.

[5] S.-K. Chang, Principles of Pictorial Information Systems Design, Prentice-Hall, pp. 262-301, 1989.

[6] S.-K.Chang, "Visual Language : A Tutorial and Survey", IEEE Software, Vol. 4, No. 1, pp. 29-39, Jan. 1987.

[7] Tadao Ichikawa, et al, Visual Languages and Applications, Plenum Press, NY, pp. 99-119, 1990.

[8] N. C. Shu, "FORMAL : a Forms-oriented

Visual Directed Application Development System," IEEE Computer, Vol. 18, No. 8, pp. 38-49, 1985.

[9] P. Harmon and B.Sawyer, ObjectCraft: A Graphical Programming Tool for Object-Oriented Applications, Addison-Wesley, 1991.

[10] 김상욱 외3인, "객체 지향 시각 프로그래밍을 위한 객체 관리", 한국정보 과학회 논문지 제21권 1호, pp. 196-206, 1994. 1.

[11] 김상욱 외6인, "객체 지향 프로그래밍을 위한 시각화 시스템", 한국정보과 학회 논문지 제20권 2호, pp. 1773-1792, 1993, 12.

[12] W.Finzer and L.Gould, "Programming by Rehearsal", Byte, pp. 187-210, Vol. 9, No. 6, June 1984.

[13] R.V. Rubin, E.J. Golin, and S.P.Reiss, "ThinkPad: A Graphical System for Programming by Demonstration," IEEE Software, Vol. 2, No. 2, pp. 73-79, Mar. 1985.

[14] M.Hirokawa and M. Tanaka, "An Iconic Programming System, HI- VISUAL", IEEE Trans. SE., Vol. 16, No. 10, pp. 1178-1184, Oct. 1990.

[15] N.M.Zloop, "QBE/OBE : A Language for Office and Business Automation", IEEE Computer, Vol. 14, No. 5, pp. 13-22, 1981.

[16] R.J.K. Jacob, "A State Transition Diagram Language for Visual Programming", IEEE Computer, Vol. 8, No. 8, pp. 55-59, 1985.

[17] 하수철, "프로그램 작성을 위한 시각 프로그래밍", 대한 전자공학회 학술 발표 논문집, Vol. 14, No. 2, pp. 189-192, 1991. 11.

[18] 하수철,원유헌, "C++에서의 객체 지향 모델링을 위한 다이아그래밍 툴", 대한 전자공학회논문지, Vol. 2-B, No. 2, pp. 121-129, 1992. 2.

[19] 하수철, "시각 프로그래밍 환경 구축을 위한 원소", 정보 과학회 춘계 학술 발표 논문집, Vol. 20, No. 1, pp. 297-300, 1993.

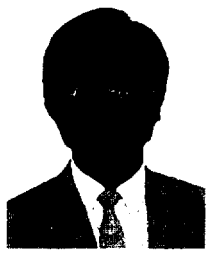
[20] Soo Cheol Ha, "A Visual Tool for C++ Program Development", IEEE TENCON 93/ Beijing China, pp. 280-283, Oct. 1993.

[21] Soo Cheol Ha, "Visual Diagramming Tool by C++gram", Proceedings of JTC-CSCC '94, pp. 721-725, 1994.

[22] T.DeMarco, Structured Analysis and System Specification, Yourdon Press, NY, 1978.

[23] G.Booch, Object-Oriented Design with Applications, Benjamin/Cummings Pub., 1991.

[24] J.Rumbaugh, and et al, Object-Oriented Modeling and Design, Pentice-Hall, NJ, 1991.



하 수 철

1981년 홍익대학교 전자계산학과 졸업(학사)
1986년 홍익대 대학원 전자계산학과 졸업(석사)
1990년 홍익대 대학원 전자계산학과 (이학박사)
1981년~84년 Army Logistics Command, System Analyst
1992년 Florida State Univ. 객원교수, Univ. of Texas 객원교수
1987년~현재 대전대학교 컴퓨터공학과 부교수
관심분야 : 소프트웨어 공학, 객체지향화(OOA/OOD/OOP), 프로그래밍 언어, 시각 프로그래밍.