C++를 위한 대화식 다중 뷰 시각 프로그래밍 환경

류 천 열† 정 근 호†† 유 재 우††† 송 후 봉†††

요 약

본 논문은 다중 뷰를 이용한 대화식 시각 프로그래밍 환경에 관한 연구로서, C++ 언어 프로그래밍을 위한 클래스의 시각화와 호출되는 멤버 함수의 흐름을 시각화하는 뷰들을 제공한다. 본 연구는 클래스에 대한 새로운 시각 기호를 정의하고, 시각 기호를 이용한 다양한 뷰의 대화식 시각 프로그래밍 환경을 구성 하였다. 대화식 다중 뷰 시각 프로그래밍 환경은 객체지향 언어에서 클래스의 표현과 객체간의 실행 관계를 시각적으로 표현하므로써 객체지향 프로그램의 전체 구조에 대한 파악이 용이하여 프로그램의 개발이 손쉬워지고, 초보자를 위한 교육과 훈련에도 유용하게 사용될 수 있다.

An Interactive Multi-View Visual Programming Environment for C++

Chun Yeol Ryou† Geunho Jeong†† Chae Woo Yoo††† Hoo Bong Song†††

ABSTRACT

This paper describes the intractive visual programming environment using multi-view which shows the tools of visualization for class and the visualizations for called member-function flow in C++ language. This research defines new visual symbols for class and constructs interactive visual programming environment of various views by using visual symbols. Our proposed interactive multi-view visual programming environment can represent visualization for representation of class and execution relationships between objects in the object-oriented language, which is easy to understand the structure of object-oriented program, therefore our proposed interactive visual programming environment enables easy program development, and can use of education and trainning for beginner in useful.

1. 서 론

반세기 전에 컴퓨터가 처음으로 만들어진 후, 컴퓨터 기반 기술의 발전으로 컴퓨터의 성능이 급속히 향상되고, 사회 모든 분야에 컴퓨터의 사용이 다양해짐에 따라 프로그래밍 영역이 점점 확대되고 복잡해져서 프로그램 개발자의 프로그래밍 작업을 효율적으로 지원할 수 있는 개발 환경과 사용자 인터페이스(interface) 까지도 고려하게 되었다. 이로 인하여 프로그래밍 개발 환경이 문자 기반 환경에서 그래픽 기반 환경으로 옮겨 가게 되었다[1, 2].

이러한 프로그래밍 환경들로는 Cornell 대학의 CPS(Cornell Program Synthesizer), Carnegie-mellon 대학의 IPE(Incremental Programming Environment), Brown 대학의 PECAN 그리고 Xerox PARC의 Interlisp 등이 있다[18].

그러나, 이로 이해 사용자들은 더욱 더 많은 프로그래밍 작업을 편리한 개발 환경에서 수행할 수 있게 되었지만, 이 프로그래밍 환경을 지원해야 하는 프로그램 개발자들은 커져만 가는 컴퓨터와 실제 세계와의 의미적 단절(semantic gap)을 줄이는 작업을 계속해 왔다. 이로 인한 부담

† 정 회 원 : 유한전문대학 전자계산학과 부교수
†† 정 회 원 : 숭실대학교 대학원
††† 정 회 원 : 숭실대학교 컴퓨터학부 교수
논문접수 : 1995년 4월 13일, 심사완료 : 1995년 10월 20일

은 기존의 방법으로는 감당하기 힘들게 되어, 이를 해결하기 위한 새로운 방법이 강구 되었는데 이것이 객체지향 방법론과 프로그래밍 개발 환경의 개선이다.

객체지향 방법론은 문제와 그 해결 방법을 인간의 실제 생활 환경에 접근한 방법론으로 처리하는 데이타와 그 데이타 처리에 이용하는 메소드(method)들을 하나의 모듈인 객체에 통합시킨 개념이다. 이는 추상화(abstraction), 정보 은폐(information hiding), 모듈화(modulation) 등의 소프트웨어 공학적인 개념을 지원하며, 소프트웨어의 재사용성을 높여준다[4].

프로그래밍 개발 환경의 개선은 대화식 시각 프로그래밍 환경을 생각할 수 있다. 기존의 프로그래밍 개발 환경은 프로그램의 작성, 컴파일, 링크, 실행, 디버깅 작업을 반복적으로 처리하는 일괄 처리 방식으로 프로그래밍 하였으나, 이는 문법적으로 완전한 프로그램에만 적용할 수 있는 방식으로, 문법적으로 불완전한 프로그램의 일부분이나 프로그램의 수정 부분 그 자체는 처리할 수 없어서 재사용할 수 없었으나, 대화식 프로그래밍 환경은 완전하지 않은 프로그램의 일부분도 그 즉시 구문 분석이나 의미 분석을 처리할 수 있어 프로그래밍이 쉬워지고 재사용할 수 있다.

또한 사용자와 컴퓨터의 대화를 손쉽게 하기 위해서 그래픽 사용자 인터페이스(GUI : Graphic User Interface) 개념을 도입하여 입력된 원시 코드를 사용자가 보기 좋고 이해하기 쉬운 형태로 출력하는 프로그램 시각화(visualization of program)를 이용할 뿐만 아니라 시각 프로그래밍(visual programming)은 이해하기 쉬운 시각 기호를 사용하여 프로그램을 작성하고, 이를 원시 코드로 출력하는 개념으로 시각 기호를 이용한 직관적이고 능률적인 프로그래밍이 가능해진다[5]. 객체지향 방법론의 채용은 재사용성과 확장성 등의 장점을 제공하고, 시각 프로그램은 개발 대상에 대한 높은 인지도와 이해도를 제공하고, 특히 객체지향 방법론에서 중심이 되는 클래스와 객체는 기존의 구조적 프로그램에서의 프로그램 구성 요소에 비해 독립성이 강한 특성을 갖기 때문에 시각 프로그래밍에 적합한 대상이다. 객체지향 방법론과 시각 프로그래밍의 장점을 동시에 얻기 위해 이들의 결합이 필요하다.

논문의 구성은 2장에서 객체지향 시각 프로그래밍 환경의 기본 개념들에 대하여 자세히 언급하고, 3장에서는 설계된 C++ 프로그래밍 개발 환경에 대하여 기술한다. 그리고 4장은 시스템 구현후의 실제 예제에 대한 실행 결과를 제시한다.

2. 객체지향 시각 프로그래밍

2.1 프로그램의 시각화와 시각 프로그래밍

프로그램의 시각화는 원시 코드의 시각화를 제일 먼저 생각해 볼 수 있다. 원시 코드의 시각화로는 1986년에 Baecker와 Marcus에의해 C 언어를 위한 SEE 시각 컴파일러가 개발 되었다[6].

이 시스템에서는 다양한 글꼴과 기법을 사용하여 프로그램의 원시 코드에 대한 가독성을 높여, 단순 문자열의 원시 코드보다는 적절한 시각화로 프로그램의 이해도가 크게 향상 되었다.

또한, 오래 전부터 프로그램을 설계할 때에 프로그램의 정적인 흐름을 표현하는데 순서도(flow chart)와 NSD(Nassi-Schneiderman Diagram)을 이용하여 프로그램의 시각화를 시도해 왔다. 이와 같은 노력으로 프로그래밍 개발 환경이 프로그램 작성자의 손에 의해 종이 위로 옮겨졌으며, 그래픽 출력 장치의 보급으로 그림을 이용한 시각화가 가능해졌다. 그림을 이용한 시각화 시스템으로는 1986년 Levien에 의해 개발된 Lisp을 위한 시각 문법 편집기(visual syntax editor)와 Clark과 Robinson에 의해 NSD를 기반으로 프로그램의 제어 흐름 구조를 시각화한 파스칼을 위한 그래픽 상호작용 프로그램 모니터가 있다. 그림을 이용한 시각화 시스템은 기존의 프로그래밍 언어로 작성된 프로그램을 잘 이해할 수 있는 장점을 가지고 있어서 프로그래밍 시간과 공간 그리고 비용을 효율적으로 사용할 수 있었으나, 프로그램의 어느 한부분만을 강조한 것이고 그 외 부분들은 약화되어 보일 수 있었다.

그래서 프로그램의 여러 부분을 다 같이 시각화하는 다중 뷰를 고려하게 되었다. PECAN은 Brown 대학에서 대수적인 프로그래밍 언어를 위해 설계 개발된 대표적인 다중 뷰 시스템으로,

프로그램의 정보를 내부적으로 추상 구문 트리로 표현하고, 이 추상 구문 트리를 공유하여 외부적으로 프로그램이나 의미에 일치하는 수식 트리, 동적으로 변화하는 프로그램의 흐름을 나타내는 순서도 등의 다중 뷰를 제공하여 단일 뷰에서의 단점을 극복하게 되었다.

2.2 객체지향과 GUI

객체지향에서 가장 기본적인 개념들은 클래스(class), 메세지(message), 상속성(inheritance) 등이다. 이 개념들은 객체(object)를 기반으로 하고 있다.

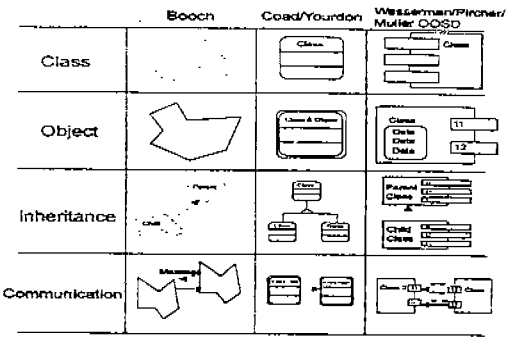
객체지향 개념은 실세계의 기능을 보다 이해하기 쉽고 간단하게 표현하는 것이 가능하도록 해준다. 그리고 이러한 객체지향 개념을 가진 프로그래밍 환경에서 사용자 인터페이스, 특히 GUI 개념을 강조함으로써 프로그래머들은 단순하면서도 훨씬 효율적인 프로그래밍 환경을 제공받게 된다. GUI를 제공하는 방법은 시스템에 따라 다를수 있는데, 보통은 아이콘, 버튼, 대화상자 등과 같은 그래픽 요소들을 사용한다[5].

프로그래머는 객체지향 프로그래밍의 모듈성, 재사용성, 확장성과 같은 특성을 모두 이용할 수 있다. 그러므로 객체지향적인 환경에서의 모든 GUI의 개발 영역은 객체 중심적이고 객체의 기본 개념 특성을 거의 유사하게 또는 실질적으로 제공한다.

따라서 객체지향의 사용자 인터페이스 개발 도구에는 무엇보다 객체의 시각적인 표현이 중요하다. 객체는 캡슐화, 추상화, 다형성, 상속성의 특성이 있으므로 객체에 대한 시각적인 표현에서는 객체의 특성이 잘 묘사되어야 개발자의 이해를 도울 수 있다.

2.3 객체지향 요소의 시각 기호

C++ 언어에서는 클래스에 대한 정의가 가장 기본적이고 중요한 요소이다. 클래스는 데이타 멤버 필드와 메소드의 멤버 함수로 구성 된다. 개발자 관점에서 클래스의 모든 멤버요소와 상속에 대한 상세한 정보가 제공되어야 한다. 즉, 멤버 필드와 멤버 함수의 내용과 접근 제어(access control)에 대한 구분, 클래스들간의 상속관계와



(그림 1) 일반적인 객체지향 시각 기호
(Fig. 1) General Object-Oriented Visual Symbol

그에 따른 클래스 처리영역에 대한 정보, 그리고 클래스간의 메세지를 이용한 통신과 같은 정보의 표현을 위한 시각적인 기호 정의가 필요하다[7, 10, 12, 13, 16, 17].

객체지향 요소를 정의한 기호 표기법은 여러가지 시각적인 방식으로 정의되어 왔으나 표준적인 정의는 아직 확실히 정의되지 않았다. 본 절에서는 Booch, Coad/Yourdon, Wasserman/Pircher/Muller에 의해 정의된 기호를 (그림 1)과 같이 살펴본다[4, 12, 16, 17].

Booch의 기호는 클래스의 추상성을 강조하여 비정형화된 구름 형태를 하고 있는 점과 클래스 내의 데이타를 객체가 아닌 클래스 간의 연관관계를 표기하는 점이 특징이다. Coad/Yourdon의 기호는 둥근 모서리의 사각형으로 클래스를 표시하며 이를 다시 데이타와 함수부분으로 구별하여 캡슐화를 나타낸다. OOSD(Object-Oriented Structured Design)는 구조적 설계의 구조도와 Booch의 기호, 클래스의 상속성을 기반으로 작성된 기호이다. 여기서 Booch의 기호는 복잡한 형태를 가지고 있으며 Coad/Yourdon의 기호가 간단한 형태이다.

3. 대화식 다중 뷰 시각 프로그래밍 환경의 설계

본 절에서는 객체지향 시각 기호를 정의하고, 이를 이용하여 C++ 언어를 대상으로 한 프로그램 개발 환경을 설계하고 구현한 사항을 기술한다. 이 시스템은 C++ 언어의 특성을 시각적

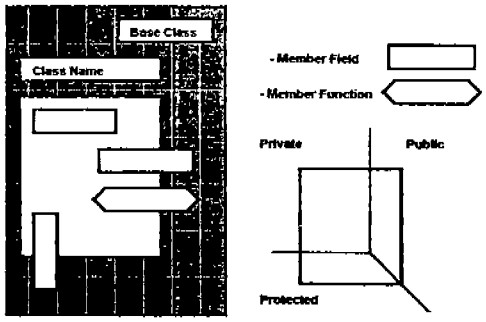
으로 표현하는 부라우저(browser)를 중심으로 구현 되었으며 확장성을 고려하여 설계하고 구현 하였다.

3.1. 객체지향 시각 기호

3.1.1 클래스의 시각 기호

C++ 소프트웨어 개발 환경에서 시각 기호 설계시 클래스에 대한 충분한 양의 상세하고 명료한 시각 정보의 표현이 가능해야 한다. 클래스에 조합되는 정보 즉, 멤버 함수와 멤버 필드의 정보 및 private, public, protected의 접근 제어와 클래스간의 상속 관계를 함축적인 방법으로 표현하여야 한다[12].

본 시스템에서는 Wasserman/Pircher/Muller의 OOSD의 기호와 유사한 Raimund K. Ege [12]의 클래스 시각 기호를 선택 하였으며, 이는 (그림 2)와 같은 기호로 정의된다.



(그림 2) Raimund K. Ege의 클래스 다이어그램
(Fig. 2) Class Diagram of the Raimund K. Ege

Raimund 시각 기호에서 클래스는 큰 시각형으로 표현하고, 클래스의 구성 요소로서 멤버 필드는 직사각형, 멤버 함수는 장방형의 육각형으로 표현하여 클래스의 큰 시각형 안에서 표현된다.

이로서 클래스 내부에 있는 멤버 필드와 멤버 함수는 캡슐화되어 표현 되므로서 클래스 외부로부터 은폐된다. 클래스의 멤버들에 대한 접근 제어는 세 부분으로 분류 하는데, private 멤버는 클래스 사각형 내부의 좌측 상단에 가로로 열거하고, public 멤버들은 우측에 클래스 사각형의 내부와 외부의 중간에 가로로 표현된다. protected 멤버는 클래스 사각형의 하단에 사각형의 내부와 외부의 중간에 세로로 표현한다. 또한 클래스간의 상속 관계를 표현하기 위해 큰 사각형의 클래스 기호의 우측 상단에 베이스 클래스와 상속 관계를 나타낸다.

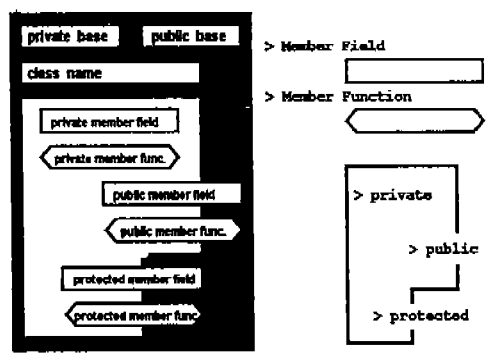
그러나, Raimund의 클래스 시각 기호를 이용하여 클래스간의 상속성을 그래프로 세로 방향으로 표현할 때[1] protected 멤버의 이름이 세로 방향에 나타나고 상속관계 표현시 모호성이 발생될 수 있어 (그림 3)과 같이 수정, 설계하여 제시한다.

수정, 설계된 클래스 다어어그램은 클래스 이름을 중심으로 상단에는 베이스 클래스가, 하단에는 클래스의 구조가 나타난다. Raimund의 클래스 다이어그램에서는 기본적으로 좌측 상단을 내부로, 우측과 하단을 외부로 보고 클래스의 접근 제어를 표시 하였으나, 시각적인 인지도와 구현시의 문제때문에 이를 수정하여, 베이스 클래스의 경우는 좌측을 내부로, 우측을 외부로 간주하고 좌측에 오는 베이스 클래스는 private 처리로, 우측의 경우는 public을 나타내는 것으로 정의한다.

또한 클래스 몸체 내부에 나타나는 멤버는 private으로, 외부와 접하는 멤버는 public으로 하고, 클래스 몸체 하단에 요철을 가함으로서 제한적으로 외부와 접하는 멤버를 protected로 정의한다.

3.1.2 메세지 전달에 따른 시각 기호

객체지향에서의 연산은 객체간의 메세지 전달



(그림 3) 수정, 설계된 클래스 다이어그램
(Fig. 3) Modified and Designed Class Diagram

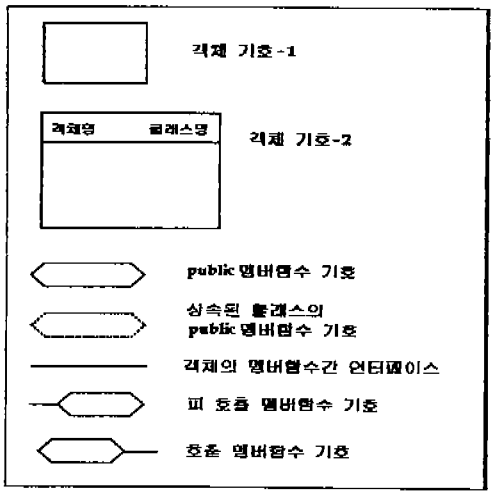
에 의해 이루어진다. 일반적으로 객체지향 프로그램에서 수동 객체는 어떤 요구에 의해 처리되고 그 처리 결과를 되돌려 준다.

C++에서 클래스는 멤버 필드와 멤버 함수로 구성되고, 멤버 함수의 실행으로 클래스의 멤버 필드에 값이 변경된다. 또한 다른 클래스의 멤버 필드의 처리가 필요하다면 해당 클래스에 메세지를 전달하여 필요한 처리를 수행한다. C++에서의 객체 실행은 먼저 클래스 문법에 의해 객체 지향적인 기본 처리 단위를 정의하고, main 모듈의 시작으로 실제적인 실행을 시작한다. 정의된 클래스 데이타 타입은 새로운 객체 이름으로 재 선언되고 객체의 생성이나 메세지 호출로 인해 멤버 함수를 실행한다.

호출 함수에 대한 처리에 대하여 절차적 프로그래밍 언어는 정적 바인딩(static binding)으로 컴파일시 호출 함수에 따른 실행 코드가 생성되는 반면에 객체지향적 언어는 실행시 코드 생성되는 동적 바인딩(dynamic binding) 처리를 한다. 그러나 C++언어는 강력한 타입 기반의 프로그래밍 언어이므로 클래스 타입에 함수 처리시 정적 바인딩, 동적 바인딩의 차이가 없다.

객체간의 실행에 발생하는 메세지의 구성은 다음과 같다[9].

Instance1.MethodA(ParameterX)



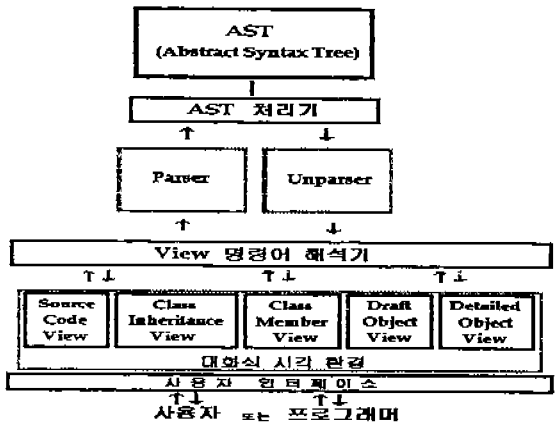
(그림 4) 메세지 전달 시각 기호
(Fig. 4) Visual Symbol for Message Passing

Instance1은 메세지를 전달받는 객체명으로 클래스의 재 선언형이고, Method.A는 호출하기 위한 메소드명으로 이것은 클래스의 public 멤버 함수명이 된다. 그리고 ParameterX는 Method.A에 전달되는 변수들이다. 이러한 메세지의 전달은 결국에는 객체간의 public 멤버 함수가 호출하는 것이다. 메세지의 전달로 객체간의 실행 제어를 시각 기호로 표현할 때 클래스의 시각적인 표현과의 일관성을 유지해야 하고, 객체간의 상속성 관계에 대한 추적도 이루어져야 한다. 시각적인 다이어그램은 실행하려는 객체를 하나의 정보 단위로 하여 호출 관계에 나타내는 메세지의 구성 요소인 객체명, 멤버 함수명 그리고 호출 관계를 시각적인 기호로 표현하여 (그림 4)와 같이 정의한다.

3.2. 대화식 다중 뷰 시각 프로그래밍 환경 시스템의 구성

본 논문의 시스템은 (그림 5)와 같이 정의하였다. 다중 뷰를 이용한 시각적이고 대화식 C++ 개발 환경에서는 프로그램에 대한 내부 자료 구조를 추상 구문 트리로 구성하였다.

시스템 구성은 사용자 인터페이스와의 접속으로 여러가지 그래픽 뷰들이 지원되는 대화식 시각 환경(interactive visual environment), 뷰에서의 명령어를 해석, 처리하여 파싱할 수 있게 하는 뷰 명령어 해석기(view command inter-



(그림 5) 대화식 다중 뷰 시각 프로그래밍 환경
(Fig. 5) Interactive Multi-View Visual Programming Environment

preter), 입력되는 프로그램의 파싱을 위한 점진적 파서와 역파서(unparser), 그리고 파싱 결과를 트리로 관리하는 추상 구문 트리 처리기로 구성된다.

3.2.1 대화식 시각 환경(Interactive Visual Enivironment)

대화식 시각 환경은 여러가지 뷰들로 구성되는데 본 시스템은 원시 코드 뷰, 클래스 뷰, 메세지 전달 뷰로 분류한다. 이 시각 환경은 기본적으로 시스템과 사용자 인터페이스가 접속되는 개발자의 직접적인 부분이다. 클래스 뷰에는 클래스의 이름과 상속성을 그래프로 시각화하고 개발자에 의해 직접 수정 가능한 클래스 상속뷰, 클래스의 상속성과 클래스내의 멤버 구성을 상세히 시각화하여 시각 프로그래밍할 수 있는 클래스 멤버 뷰로 구성한다.

메세지 전달 뷰는 정의된 객체에 의한 메세지 전달 상황을 표현하는 것으로 public 멤버 함수에 의한 객체간의 호출을 전체적으로 시각화하여 표현한 개요 객체 뷰, 객체의 멤버 함수 구성과 상속성을 고려하여 자세하게 표현하여 메세지 전달에 의한 객체간의 통신을 시각적으로 프로그래밍 가능한 상세 객체 뷰로 이루어져 있다.

이 환경에서는 시각 프로그래밍 작업을 위해 마우스 처리 명령어를 이용 한다.

3.2.2 뷰 명령어 해석기(View Command Interpreter)

뷰에서 사용되는 명령어를 해석하는 부분으로 텍스트 편집 명령, 클래스 상속 처리 명령 그리고 그에 따른 클래스 멤버 편집 명령, 객체간의 메세지 전달 처리 명령, 화면 처리 명령과 화일 처리 명령으로 구분할 수 있으며 명령어의 종류는 다음과 같다.

텍스트 편집 명령
  Left, Right, Up, Down, Insert, Delete, Backspace

클래스 상속 처리 명령
  Class Create, Class Remove, Base Class Create, Base Class Remove

클래스 멤버 편집 명령
  Member Field Create, Member Field Remove, Member Function Create & Remove

객체간의 메세지 전달 처리 명령
  Add Call, Delete Call

화면 처리 명령
  Page-up. Page-down, Beginning of File, End of File, Top, Bottom

화일 처리 명령
  New, Read, Write, Open, Append

3.2.3 파서(Parser)

시각 프로그래밍 환경에서는 클래스 정보나 메세지 처리와 같은 시각 다이어그램을 사용하게 되므로 비단말 노드로부터 프로그램의 입력을 파싱할 수 있도록 multiple entry parser를 구성하여 처리했다[1].

multiple entry parser는 placeholder를 유도하는 모든 비단말 노드에 생성규칙 S → Ti ni 을 문법에 추가하여 구성한다. 예를들면 S를 시작심볼, statement는 비단말 심볼이라고 할때 문법에는 다음과 같이 생성규칙이 첨가된다.

S → StmtNull statement

여기서 StmtNull은 새로이 추가된 태그 단말 심볼로서 시각 프로그래밍되어 입력되는 다이어그램을 입력받아 이 태그 심볼에 자동 삽입하므로서 파싱하고 추상화 구문트리를 생성한다.

3.2.4 역파서(Unparser)

AST를 입력으로 하여 들여쓰기(indentation), 줄바꿈(line break), 쪽바꿈(page break) 등에 맞추어 AST에 해당하는 원래의 텍스트나 그래픽 심볼을 화면이나 인쇄 장치에 보기좋은 형식으로 출력한다. 또한 프로그램의 의미를 직관적으로 표현하는 시각 프로그래밍을 지원하기 위하여 다양한 형태의 아이콘과 그래프를 이용하며, 프로그램의 구문을 모두 표현하는 것이 아니라 의도하는 목적에 따라 다양한 형태로 추상화하여 표현한다. 또한 이들 다양한 표현형태 간의 일치성의 확보를 위한 작업이 수행된다.

3.2.5 추상 구문 트리 처리기(Abstract Synt-ax Tree Handler)

추상 구문 트리와 심볼 테이블과 같은 시스템 정보를 관리한다. 대화식 시각 환경에서의 시각 프로그래밍 작업으로 생성 또는 삭제되는 심볼 노드에 의한 트리의 처리와 unparsing scheme으로 화면에 출력되는 프로그램은 내부적으로 각 노드와 일치되는 심볼, 특히 클래스의 멤버 함수와 멤버 필드 그리고 메세지의 정보를 이용하여 마우스에 의해 포인트되는 노드를 찾을 수 있도록 한다. 노드의 정보가 변화될 때 뷰에서의 클래스나 메세지의 시각적인 표현이나 텍스트 그리고 추상화 구문 트리 노드가 일치하도록 하고, 마우스 포인터에 의해 노드를 추적할 수 있도록 항상 최근의 상태를 유지하여야 한다.

3.2.6 추상 구문 트리(AST)

C++ 객체지향 언어의 문법에 따라 추상 구문 트리를 정의하고 프로그램은 내부적으로 이 추상 구문 트리에 따라 유도되는 트리로 표현한다. 추상 구문 트리는 다음과 같은 형식으로 정의된다.

X0 : attr(X1, X2, ···, Xk)

attr는 attribute로 노드의 이름을 나타내고 Xi는 문법의 비단말 심볼들이다.

특히 본 시스템은 확장성을 고려하여 설계하였기 때문에 개발 환경에서 프로그램에 대한 내부 표현 방법이 중요하다. 왜냐하면 대화식 개발 환경 도구에 나타나는 프로그램 정보에는 일관성이 있어야 하기 때문이다. 프로그래밍 환경의 표준적인 데이타 표현 방법인 AST를 이용하는 경우는 프로그램 심볼의 가장 핵심적인 단일구조로서 이해가 용이하고, 새로운 도구 추가시 AST에 별도의 정보 추가나 변경없이 필요에 따라 간단하게 작업할 수 있다.

4. C++ 개발 환경의 구현

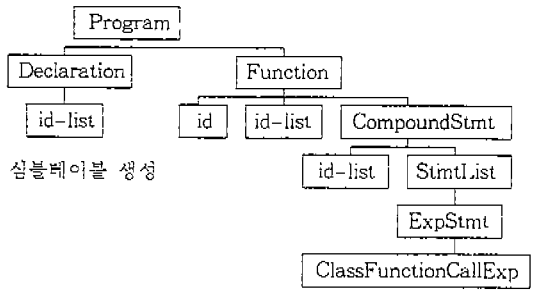
4.1. 구현 환경

C++ 프로그램의 클래스 정의와 메세지에 대한 처리를 대화식이고 시각적으로 표현하는 개발 환경의 구현은 UNIX 환경의 SUN SPARC 워크스테이션을 사용하였으며, 사용자 인터페이스로는 X 윈도우 시스템의 Xlib, OSF/Motif 그리고 C 언어를 사용하였다.

개발환경에서 C++ 프로그램의 내부 표현은 구문 분석에 의한 AST를 이용으로 (그림 6)의 구조와 같이 선언부에서의 클래스 정의와 함수 처리부의 메세지 처리를 중심하여 기본적인 프로그램의 심볼 테이블을 작성 하였다. 시각적으로 프로그래밍되는 C++의 파싱을 위한 multiple entry parser와 어휘분석기는 설계의 문제점을 줄이기 위해 Lex와 Yacc를 이용하여 구축하였다.

4.2. 대화식 시각 프로그래밍

본 C++ 개발환경 시스템의 초기 화면은 원시 코드 뷰트 프로그램 코드를 입력하면 이를 파싱하여 AST와 심볼 테이블을 구축한다. 대화식 시각 프로그래밍 환경의 클래스 뷰와 메세지 전달 뷰의 선택으로 파싱된 결과를 용도에 따라 그래픽 다이어그램을 이용하여 시각화한다. 또한 시각화된 뷰에서는 시각 프로그래밍이 가능하므로 마우스 클릭으로 마우스 위치와 일치하는 추상 구문 트리 노드를 찾아서 그 노드에 대한 정보를 조작하고 다른 뷰들의 시각적인 일관성을 유지하기 위해 자동 수정된다.

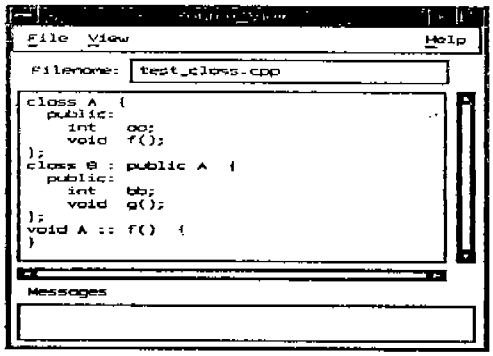


(그림 6) C++ 프로그램의 기본 AST 구조
(Fig. 6) Fundamental AST Structure of C++ Program

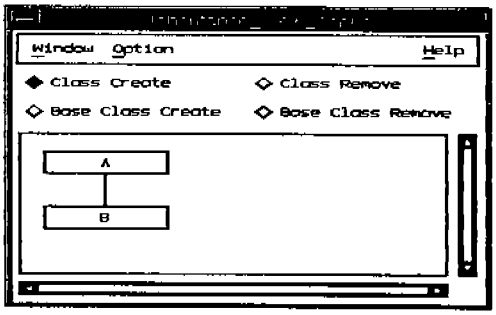
4.2.1 원시 코드 뷰

원시 코드 뷰에서 아래와 같은 원시 코드를 입력하면 파서는 이를 파싱하여 추상 구문 트리를 구성하고 다시 역파싱하여 (그림 7)과 같이 원시 코드 뷰에 이를 표시한다.

```
class A {
public:int aa;
        void f();
};
class B : public A {
public:int bb;
        void g();
};
void A :: f() {
}
void B :: g() {
}
```



(그림 7) 원시 코드 뷰
(Fig. 7) Source Code View

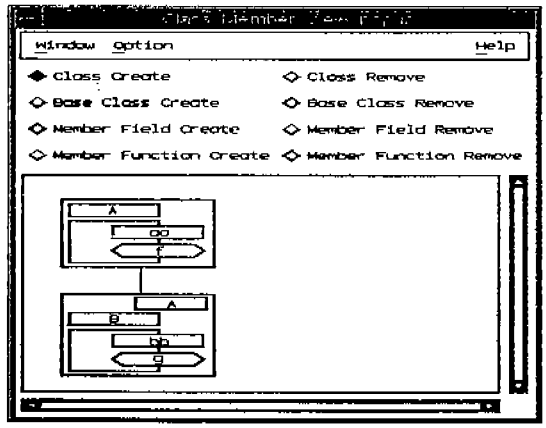


(그림 8) 클래스 상속뷰
(Fig. 8) Class Inheritant View

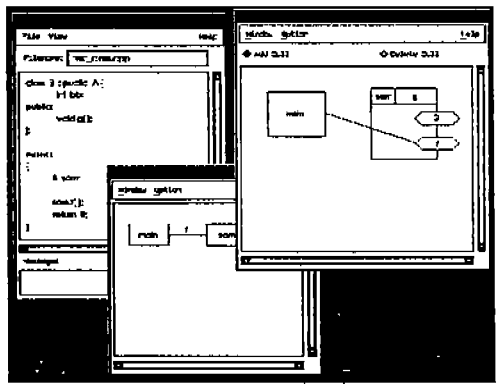
4.2.2 클래스 뷰

클래스 뷰는 클래스 상속 뷰(Class_Inheritance_View)와 클래스 멤버 뷰(Class_Member View)로 구성되어 있으며, 클래스 상속 뷰는 직사각형으로 표시되는 class A와 class B의 상속 관계만을 (그림 8)과 같이 표시한다.

클래스 멤버 뷰는 앞에서 정의한 클래스 다이어그램을 이용하여 각각의 클래스 내부의 데이타와 멤버 함수 등을 (그림 9)와 같이 표시한다. class A 의 aa는 직사각형으로, f는 육각형으로 표시되며, class B 에서는 bb가 직사각형, g가 육각형으로 표시되어 각각 멤버 필드와 멤버 함수임을 나타낸다.



(그림 9) 클래스 멤버 뷰
(Fig. 9) Class Member View



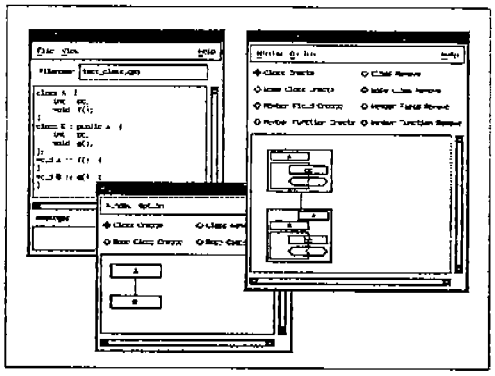
(그림 10) 메세지 전달 뷰들
(Fig. 10) Message Passing Views

4.2.3 메세지 전달 뷰

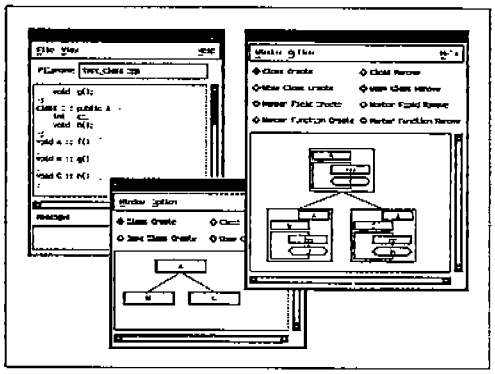
메세지 전달 뷰는 (그림 10)과 같이 개요 객체 뷰(Draft_Object_View)와 상세 객체 뷰(Draft_Object_View)로 구성되어 있으며, 개요 객체 뷰는 메세지 전달로 인한 객체간의 접속을 객체명 중심으로 시각화하여 프로그램 실행에 대한 전체적인 흐름 파악에 유용하며, 상세 객체 뷰는 메세지 전달이 처리되는 객체와 객체내의 멤버 함수들의 정보들을 상세하게 표시하며, 상세 객체 뷰를 이용하여 원시 프로그램의 입력도 가능하다.

4.3. 실행 결과와 분석

대화식 시각 프로그래밍 환경에서 작성된 예제 프로그램의 클래스 뷰를 같이 전개해 놓은 것이 (그림 11)과 같다.



(그림 11) 클래스 뷰들
(Fig. 11) Class Views



(그림 12) 변형된 클래스 뷰
(Fig. 12) Changed Class Views

Class A 에서 상속되는 새로운 class C를 추가하기 위해서는 우선 상속 관계를 정의하기 위하여 (그림 10)의 클래스 상속 뷰에서 Class Create 버튼을 선택한 다음 새로운 class C를 class A로부터 마우스를 끌어서 그리고, 원시 코드 뷰에서 클래스의 내용을 직접 입력하거나 클래스 멤버 뷰에서 Member Field Create 버튼을 선택하여 변수명 cc를 입력하고, Member Function Create를 선택한 후 함수명 h를 입력하면 (그림 12)와 같은 화면이 구성된다.

이러한 방식으로 원시 코드 뷰와 클래스 뷰는 프로그램 작성 도중 및 작성된 프로그램에서의 클래스 구조의 이해와 수정이 용이하므로, 객체지향 언어의 시각 프로그래밍 환경에서 효율적으로 이용될 수 있다.

대화식 시각 프로그래밍 환경을 기존의 문자기반 편집 환경과의 비교하면 다음과 같다.

첫째, 프로그램의 작성이 쉽고 빠르다. 문자기반 환경에 비해 상대적으로 적은 수의 입력으로 동일한 프로그램의 작성이 가능하다. (그림 11)에 나타난 바와 같이 class C를 추가하는 경우 8회의 마우스 조작으로 상속 관계와 멤버의 정의가 가능하다.

둘째, 프로그램의 요소를 시각 기호에 의해 표시함으로 분석이 용이하다. 또한 특정 뷰에서 발생한 변화가 다른 뷰에도 즉시 반영되므로 다양한 뷰에 의해 프로그램의 구성을 점검할 수 있다.

〈표 1〉 객체지향 개발환경들의 비교
〈Table 1〉 Comparision of Object-Oriented Development Environments

	Sparc Works	Visual C++	본 논문에서 제안한 시스템
클래스의 상속성 시각화	가능	가능	가능
클래스 내의 member와 함수에 관한 시각화	불가	불가	가능
시각화된 기호를 이용한 시각 프로그래밍	불가	불가	가능

셋째, 언어의 문법에 대한 고려를 덜어준다. 시각 기호의 편집과 수정에 따라 원시 코드 뷰에서 자동으로 코드가 생성되므로 문법적 오류가 발생되지 않는다. 또한 원시 코드 뷰에서 직접 프로그램을 입력하는 경우에도 시각 기호로 표현하기 위한 파싱 과정에서 문법의 오류에 대한 검사를 수행하므로 쉽게 오류를 판별할 수 있다.

넷째, 프로그램에서 원하는 부분에 집중할 수 있다. 개요 객체 뷰와 상세 객체 뷰를 제공하여 전체적인 흐름과 상세 흐름을 필요에 따라 선택적으로 표시하여 볼 수 있다.

또한 객체지향 개발환경과 시각화를 제공하는 다른 시스템들과의 비교는 표 1)과 같다.

5. 결 론

C++를 위한 대화식 다중 뷰 시각 프로그래밍 환경은 클래스를 중심 개념으로 클래스의 멤버 정보와 특성을 충분히 표현하는 시각 다이어그램이 제공되는 시각적인 환경으로 대화식 처리가 되는 여러가지 뷰들을 지원한다. 이것은 C++ 프로그램에서의 클래스와 객체간의 메세지 처리를 시각화하고 시각 프로그래밍하는 개발 환경의 설계로 클래스 구조에 대해 쉽게 이해할 수 있고 이를 수정, 확장시켜 새로운 코드의 골격을 쉽게 구축할 수 있고, 클래스의 public 멤버 함수를 이용한 메세지 전달로 인한 객체간의 실행 관계를 시각적으로 표현하므로서 객체 프로그램의 전체 구조에 대한 파악이 용이하다. 그러므로 점진적인 개발로 인한 소프트웨어의 재사용 및 변경시의 신속한 이해와 작업이 용이하여, 강력하고 효율적인 객체지향적인 개발 환경으로서의 기능이 충분하고, 또한 C++ 초보자를 위한 클래스와 객체 실행에 관한 교육과 훈련에도 유용하게 사용될 수 있다.

보다 더 완전한 시각적이고 대화적인 객체지향 C++ 프로그램 개발 환경을 지원하기 위하여 프로그램 성능 평가 기능, 디버깅 기능이 지원하는 개발 도구가 요구되고 객체의 생성, 소멸에 따라 효율적인 메모리 관리가 필요하다. 또한 C++ 의 완전한 문법적 처리와 의미적 해석 그리고 코드 생성과 같은 컴파일러에 대한 연구도 필요하다.

참 고 문 헌

[1] J.R.Horgan, D.J.Moore, "Techniques for Improving Language-Based Editors", ACM, 1984.

[2] Lawrence A. Rowe, Michael Davis, Eli Messinger, Carl Meyer, Charles Spirakis, and Allen Tuan, "A Browser for Directed Graphs", Software-Practice and Experience, Jan. 1987.

[3] T.W Reps, and T. Teitelbaum, "The Synthesizer Generator", Springer-Verlag, 1989.

[4] Anthony I. Wasserman, Peter A. Picher, and Robert J. Muller, "The Object-Oriented Structured Design Notation for Software Design Representation", IEEE Computer, Vol. 23 No.3, Mar. 1990.

[5] Shi-Kuo Chang, "Principles of Visual Programming System", Prentice-Hall, 1990.

[6] Ronald M. Baecker and Aaron Marcus, "Human Factors and Typography for more Readable Program", Addison-Wesley, 1990.

[7] Paul Harmon, Brian Sawyer, "ObjectCraft, A Graphical Programming Tool for Object-Oriented Application", Addison-Wesley Publishing Company, 1991.

[8] Bjarne Stroustrup, "The C++ Programming Language", 2nd ed, Addison-Wesley, 1991.

[9] Adrian Nye, "Xlib Refernece Manual", O'Reilly & Associates, 1992.

[10] Raimund K. Ege, "Programming in an Object-Oriented Environment", Academic Press, 1992.

[11] Moises Lejter, Scott Meyers and Steven P. Reiss, "Support for Maintainnig Object-Oriented Programs", IEEE Transactions on Software Enginnering. vol 18.

No12, 1992.
[12] Edward Yourdon, "Object-Oriented System design: an Integrated approach", Prentice hall international editors, 1992.
[13] James Martin, James J. Odell, "OBIECT-ORIENTED ANALYSIS & DESIGN", Prentice-Hall, Inc., 1992.
[14] Tim O'Reilly, "Motif Reference Manual", O'Reilly & Associates, 1993.
[15] 유재우, 이승희, 조민구, "X-윈도우 기반 그래픽 프로그램 편집기", HCI'93 학술대회 발표논문집, Feb. 1993.
[16] David E. Brumbaugh, "Object-Oriented Development", Wiley, 1994.
[17] Grady Booch, "Object-Oriented Analysis and Design with Applications", 2nd ed. The Benjamin/Cummings, 1994.
[18] Nan C. Shu, "Visual Programming", Van Nostrand Reinhold, 1988

류 천 열

1979년 숭실대학교 전자계산학과(학사)
1985년 숭실대학교 산업대학원 전자계산학과(석사)
1995년 숭실대학교 대학원 박사과정중
1986년~현재 유한전문대학 전자계산학과 부교수
관심분야 : 프로그래밍언어, 시스템소프트웨어, 데이터 통신, 컴퓨터 그래픽스

정 근 호

1993년 숭실대학교 전자계산학과(학사)
1995년 숭실대학교 대학원 전자계산학과(석사)
1995년 숭실대학교 대학원 박사과정중
관심분야 : 프로그래밍언어, 한글폰트, HCI

유 재 우

1976년 숭실대학교 전자계산학과(학사)
1978~85년 한국과학기술원 전산학과(석사, 박사)
1986~87년 Cornell Univ. Visiting Scientist
1983~현재 숭실대학교 컴퓨터학부 교수
관심분야 : 컴파일러, 프로그래밍 환경, HCI

송 후 봉

1956년 육군사관학교(학사)
1986년 중앙대학교 전자계산학과(석사)
1989년 조선대학교 전기공학과(박사)
1971년~현재 숭실대학교 컴퓨터학부 교수
관심분야 : 운영체제, 프로그래밍 언어, 컴퓨터 보조학습