Journal of Korea Multimedia Society Vol. 13, No. 3, March 2010(pp. 359-368)

알고리즘 시각화를 위한 SVG 코드 생성 시스템

이향숙[†], 이수현[††]

요 약

알고리즘 시각화는 프로그램의 수행과정에 대한 이해를 용이하게 하여 프로그램의 오류 수정이나 개선에도 효과적으로 이용될 수 있을 뿐 아니라 컴퓨터 교육 분야에서도 폭넓게 활용될 수 있다. 알고리즘과 자료구조의 이해에 있어 알고리즘 자체의 설명보다는 동작과정을 직접 보여주는 시각화 방법이 더욱 바람직하다. 기존의 알고리즘 시각화 방법들은 특정 분야에 한정되어 사용할 수 있거나, 특정 환경이나 애플리케이션을 필요로 하여 폭넓은 분야에서 손쉽게 사용하는데 문제가 있었다. 본 논문에서 제안하는 시스템은 C 언어로 작성된 프로그램에 사용자가 간단한 시각화 명령을 추가하면 이를 자동 변환하여 SVG 애니메이션 코드를 생성한다. 생성된 애니메이션 코드는 웹 브라우저 상에서 실행될 수 있으며 MathML 등과 같은 다른 XML 애플리케이션이나 HTML, 스크립트 등과 결합하여 사용될 수 있다.

An SVG Code Generator for Algorithm Visualization

Hyang-Sug Lee[†], Su-Hyun Lee[††]

ABSTRACT

Algorithm visualization is useful for program testing, debugging and evaluating, as well as visual aids in education. When teaching algorithms and data structures, showing exact behaviors by graphics or animation is more suitable than just explaining them. Current systems for algorithm animation are limited to a couple of specific applications and need a special environment. In the proposed system, programmer writes source program in C and animator embeds visualization scripts in the appropriate location of the program. Then user can get an animation code in form of SVG and see a graphical representation on the web browser. Generated SVG animation code is platform independent and can also interact with other XML applications or HTML.

Key words: Algorithm animation(알고리즘 애니메이션), Visualization(가시화), XML, SVG

1. 서 론

인간의 두뇌는 문자의 형태보다 시각적 이미지와 멀티미디어 자료를 처리하고 이해하는데 더 적합하다. 기호, 이미지, 도표, 애니메이션 등을 사용한다면, 프로그램에 의한 형식적이거나 자연 언어로 설명하는 것보다 더 정확하고 더 효과적으로 정보를 전달할 수 있다. 그래픽을 이용하여 프로그램의 표현을 향상시키는데 초점을 두고 있는 것이 시각화이다[1,2].

소프트웨어 시각화는 문서 형태의 프로그램을 보는 것보다 훨씬 더 많은 양의 정보와 효과적인 이해 체계를 제공해 주며, 프로그램 수행을 연구하는데 있어서 강력한 접근법을 제공한다. 시각화 시스템은 컴퓨터 관련 교육이나, 알고리즘을 디자인하고 분석하며, 효율성을 평가하고, 프로그램을 문서화하는데 사용된다. 또한 프로그래머에 있어서는 프로그램의 이

※ 교신저자(Corresponding Author) : 이수현, 주소 : 경남 창원시 사림동(641-773), 전화 : 055)213-3816, FAX : 055) 286-7429, E-mail : sleepl@changwon.ac.kr
접수일 : 2009년 10월 19일, 수정일 : 2009년 11월 17일
완료일 : 2009년 11월 17일

[†] 창원대학교 컴퓨터공학과 졸
(E-mail : minomina@hanmail.net)
[††] 정회원, 창원대학교 컴퓨터공학과 교수
※ 이 논문은 2006년도 창원대학교 연구비에 의하여 연구되었음

해, 개발, 디버깅 등의 작업을 하는데 도움을 준다.

알고리즘의 교육이나 시각화를 위하여 BALSA, ZEUS, XTANGO, POLKA, SAMBA, JAWAA 등 일련의 시각화 시스템들은 개발 환경에 따라 각각 특색 있는 결과들을 만들어 내고 있으며, 국내에서도 다양한 시스템들이 구현되고 있다. 그러나 이런 시스템들은 대부분 개발자에 의해 시각화 프로그램의 내용이 고정되어 있어서 사용자가 원하는 내용으로 변경할 수 없고, 시각화면을 확인하기 위하여 제한된 시스템 환경이나 애플리케이션을 요구하며, 특정 알고리즘을 이해하기 위해 입력 데이터가 고정된 스크립트만을 이용하여 시각화되어지는 문제가 있다[3].

기존 시스템의 문제점을 개선하기 위해서 애니메이터로 하여금 보다 쉽고 자유롭게 시각화면을 구성할 수 있게 하고, 다양한 알고리즘에 대한 시각화가 가능하도록 하는 시스템이 필요하다. 또한 결과로 생성되는 시각화 코드의 애니메이션 효과를 확인하기 위해 요구되는 시스템 환경이나 애플리케이션에 대한 제약을 최소화하여야 한다. 본 논문에서는 이미 작성되어 있는 소스 프로그램에 간단한 시각화 스크립트를 삽입하고 프리프로세서 과정을 거쳐 컴파일하고 실행하면, 프로그램의 실행 결과와 함께 알고리즘을 시각화하기 위한 SVG 코드를 자동 생성하는 시스템을 제안하였다.

본 논문의 2절에서는 소프트웨어 시각화 분야에 대한 소개와 현재 진행 중인 연구 상태에 대한 고찰 및 XML에 기반을 둔 새로운 그래픽 포맷인 SVG에 대해 설명하고, 3절에서는 제안하는 알고리즘 시각화 시스템에 대한 전반적인 설계 내용을 기술한다. 4절에서는 구현된 시스템에 사례를 적용하여 보고, 5절에서 본 시스템과 다른 시스템과의 비교를 통하여 본 시스템의 우수성을 살펴본다. 6절에서 결론을 맺도록 한다.

2. 관련 연구

2.1 알고리즘 시각화

현재 소프트웨어 시각화에 대한 주요한 연구 주제는 알고리즘 시각화 프레임워크 및 시스템의 개발과, 프로그램 디버깅과 실행 추적을 위한 응용프로그램들이다. 또한 최근에는 코드의 구조적 시각화에 대한 연구 및 병렬 프로그램 시각화와 성능 측정, 소프트웨어 공학 계량, 멀티미디어 교육[4], 기초 프로그래밍 교육[5,6] 등을 비롯한 다양한 분야[7-9]에서 활용되고 있다.

이 중 특히 알고리즘 시각화 연구는 알고리즘의 동작을 이해하기 위한 시각화와 애니메이션을 생성하는데 집중되어 있는데, 일반적으로 이런 시각화는 자동으로 생성되지 않으며 각 알고리즘에 대한 적절한 표현을 설계하기 위한 애니메이터를 필요로 한다.

대표적인 알고리즘 시각화 시스템인 BALSA[2]는 개인 워크스테이션 기반의 X Windows 상에서 실행되는 응용프로그램으로 알고리즘 디자인과 분석 및 수업 보조 도구로써 이용되었다. 이를 발전시킨 ZEUS[10]는 객체지향적 설계와 그래픽적인 설명 및 동기화된 다중 뷰를 제공하며, XTANGO[11]는 명령어 결과를 이용해 애니메이션을 빠르게 생성할 수 있다. POLKA[12]는 다른 시스템보다 더 쉽고 빠르게 애니메이션을 구현할 수 있고, 특히 병렬 프로그램이나 계산을 애니메이션 하는데 사용될 수 있다. POLKA 기반의 SAMBA[13]는 어떠한 언어로 작성된 프로그램이라도 ASCII 명령어를 생성하여 프로그램을 시각화할 수 있다는 장점을 가지고 있으며, 이의 Java 버전인 JSAMBA[14]도 개발되었다. JAWAA[15]는 웹에서 쉽게 애니메이션을 생성하기 위한 스크립트 언어이다.

이와 같은 많은 시스템들이 알고리즘 애니메이션을 위해 개발되어 왔으나 이런 시스템들은 대부분 사용자가 디스플레이를 최적화하기 어려우며, 디스플레이가 기반 알고리즘에 제한적이고, 특정 환경이나 애플리케이션을 필요로 한다.

2.2 시각화 방법

애니메이션을 생성하기 위한 접근법은 첫째, 전형적인 프레젠테이션 툴의 사용, 둘째, 드래그 앤드 드롭이나 옵션 선택으로 시각 요소를 편집, 셋째, 소스 코드로부터 직접 애니메이션을 생성, 넷째, 함수 라이브러리로 구현된 함수 호출을 사용, 다섯째, 스크립트 언어에 의한 애니메이션 생성 등의 방법이 있다[16].

프레젠테이션 툴의 사용이나, 드래그 앤드 드롭이나 옵션 선택으로 시각 요소를 편집하는 방법은 쉽게 애니메이션을 생성할 수 있고, 별도의 프로그래밍 경험을 필요로 하지 않는다는 장점이 있으나, 애니메이션 제작에 시간이 많이 소요되며 실제 알고리즘의 동작과는 다른 애니메이션이 생성될 수 있다는 문제

점이 있다.

소스 코드로부터 애니메이션을 생성하는 방법은 시스템에 의해 자동으로 생성되므로 쉬운 애니메이션 생성을 제공할 수 있으나, 애니메이션 툴에 의해 지원되는 프로그래밍 언어가 제한적이며, 애니메이션 효과가 소스 코드 인터프리터에 고정코드로 삽입되므로 코드가 표현되는 방법을 사용자가 결정할 수 없다.

API 함수 호출에 의해 애니메이션을 생성하는 방법은 사용자가 주어진 주제에 적절한 애니메이션을 선택할 수 있으며, 부적절한 알고리즘 부분은 건너뛰고, 프레젠테이션의 주요 부분에 초점을 맞출 수 있게 해준다. 그러나 API 호출은 소스 코드에 직접적으로 삽입되거나 특정 애니메이션 생성 클래스로 정리되어 있으므로, 언어에 제약이 있고 기반 프로그래밍 언어에 대한 지식이 부족한 사용자는 활용하기 어렵다.

마지막으로 스크립트 언어에 의해 애니메이션을 생성하는 방법은, 직관적인 명령어를 사용하여 애니메이션의 정의가 명확하고 프로그래밍 지식 없이도 애니메이션 파일을 쉽게 읽을 수 있고 어떤 프로그래밍 언어에도 사용될 수 있다. 이 방법에서 애니메이션 코드 각각에 대한 이해는 용이한 반면, 원하는 효과를 생성하거나 이를 알고리즘 코드에 삽입하는 과정은 어려울 수 있다.

2.3 SVG

SVG[17]는 XML을 기반으로 2차원 그래픽을 표현하기 위해 만들어진 언어로서 W3C(World Wide Web Consortium)에 의해 제안된 XML 그래픽 표준이다. SVG로 제작한 그래픽 객체들은 벡터 방식으로 처리되므로 확대 또는 축소를 해도 그래픽의 품질을 유지할 수 있으며, 프레임 수가 많아지더라도 데이터의 크기가 비교적 작다는 장점이 있다. 또한 운영 체제와 관계없이 다양한 플랫폼에서 고품질의 그래픽을 구현할 수 있으며, 휴대용 장치 및 인쇄용으로 사용될 수 있다.

SVG는 XML 응용이므로 SMIL, GML, MathML 등 다른 XML 언어들과 결합하여 다양한 응용 애플리케이션의 개발이 가능하고, Javascript, Java, ASP, JSP, Visual Basic 등 기존의 웹 기술을 그대로 활용하여 사용자와 상호 작용하는 동적 그래픽 또는 애니메이션을 제작할 수 있다. 또한 SVG는 웹 표준인 CSS와 XSL을 지원하므로, 스타일 시트를 이용하여 문서의 레이아웃과 내용을 분리하고 그래픽 요소 및 속성을 효과적으로 제어함으로써 유지 관리 비용을 줄이고 손쉽게 업데이트 할 수 있다.

또한 텍스트로 기술되기 때문에 클라이언트의 입장에서 내용을 이해하기 쉽고, 별도의 저작 툴이 없어도 컨텐츠를 만들어 낼 수 있으며, 그래픽에 대한 검색 및 편집이 편리하고, 애플리케이션들이 SVG 문서를 쉽게 사용할 수 있다. 그리고 XML과 SVG의 DOM 인터페이스를 통하여 모든 그래픽 요소에 쉽게 접근할 수 있으므로 데이터베이스와 연동하여 웹 그래픽 문서를 동적으로 생성할 수 있어, 지리정보 시스템과 같이 그래픽이 많이 사용되면서 이에 대한 동적이고 상호작용적인 인터페이스가 강조되는 분야를 중심으로 기술개발이 활발하게 진행되고 있다.

본 제안 시스템에서는 많은 장점과 발전가능성이 내재되어 있는 SVG로 애니메이션 코드를 생성하면서도 애니메이터가 SVG 형식에 맞춰 작성해야 하는 불편함을 최소화할 수 있도록 시각화 명령들을 정의하였다. 즉 컴파일 전에 프리프로세서를 거치도록 함으로써 애니메이터는 좀더 쉽고 간단하게 시각화 작업을 할 수 있으며, 관찰자는 결과를 웹 브라우저에서 쉽게 확인할 수 있을 뿐 아니라, 다른 애플리케이션과 연동하여 다양하게 활용할 수 있다.

3. SVG 기반의 스크립트 언어

3.1 제안 시스템 개요

일반적인 시각화 모델에서 프로그래머는 처음 소스 프로그램을 작성한 사람을 말하며, 애니메이터는 프로그래머가 작성한 프로그램을 시각적 형태로 나타내 주는 사람, 관찰자는 최종적으로 시각화된 표현을 보면서 알고리즘을 학습하거나 이해하려는 사람을 말한다. 대체로 프로그래머와 애니메이터는 동일인일 수 있으며, 특히 알고리즘 교육을 위한 목적 또는 향후 소프트웨어의 유지보수를 위한 목적으로 시각화하는 경우에는 알고리즘을 가장 잘 이해하고 있는 프로그래머가 시각화 작업을 병행하는 것이 가장 효과적일 수 있다. 또는 디버깅의 용도로 시각화를 하는 경우에는 프로그래머와 애니메이터, 관찰자가 모두 동일인일 수도 있다.

국내에서의 알고리즘 시각화 관련 연구들은 대부분이 알고리즘 또는 프로그램 교육을 위한 목적으로

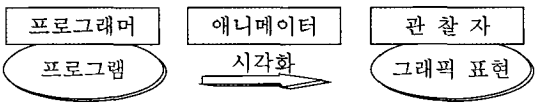


그림 1. 프로그램 시각화 모델

이루어지고 있다. 이런 시스템들은 관찰자의 입장을 중심으로 설계되었기 때문에 시각화 효과는 뛰어날 수 있지만, 미리 제공되어 있는 예제만을 시각화하거나 특정 알고리즘이나 프로그래밍 언어에 제한적인 경향이 많아 다양한 자료 구조를 표현하거나 알고리즘을 시각화하기에는 부적절하다.

본 연구에서는 다양한 알고리즘을 시각화하기 위한 방법으로 시각화 스크립트들을 정의하고, 이를 프로그램의 적절한 위치에 삽입하고, 프리프로세서를 통하여 시각화 코드를 생성하도록 하는 방법을 제안하고 있다. 그림 2는 제안 시스템의 전체적인 처리과정을 보여준다.

일반적으로 프로그래머가 C 소스 프로그램을 작성하여 컴파일하고 실행하게 되면 프로그램에 작성된 대로 결과를 얻게 된다. 이 프로그램을 시각화하기 위하여 애니메이터가 별도의 시각화 프로그램을 작성하는 것이 아니라, 스크립트들을 C 프로그램내의 적절한 위치에 삽입하고 프리프로세서를 거쳐 컴파일 및 실행하게 되면 원래 프로그램의 결과 뿐 아니라 애니메이션 코드를 함께 얻도록 하는 것이다. 애니메이션 생성의 각 단계는 다음과 같다.

Step 1. 애니메이터는 C 프로그램 내의 시각화할 부분에 "/** 시각화 명령 **/"의 형태로 스크립트를 삽입한다. 이와 같은 표현은 프리프로세서를 거치지 않고 컴파일하는 경우에는 주석으로 처리되어 원래 프로그램에 영향을 미치지 않도록 하기 위한 것이다.

Step 2. 스크립트가 삽입된 C 프로그램은 애니메이션 코드 생성을 위하여 작성된 라이브러리를 포함하는 프리프로세서를 거치게 되면 애니메이션 코드 생성을 위해 C 문법에 적합하도록 변환된 문장들을 포함하는 프로그램으로 재구성되게 된다. 이 과정을 거치게 됨으로써 애니메이션 코드를 직접 작성하지 않고 시각화 스크립트들을 이용하여 보다 쉽고 간단하게 시각화 작업을 할 수 있도록 애니메이터의 부담을 경감시킬 수 있다.

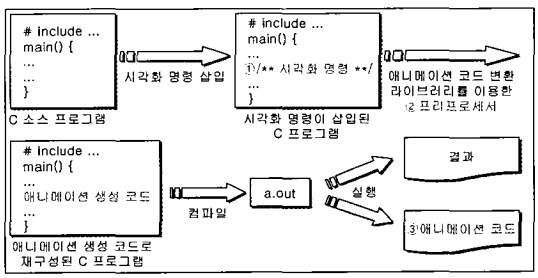


그림 2. 제안 시스템의 처리 과정

Step 3. 프리프로세서 이후에 애니메이션 생성 코드를 포함하도록 재구성된 C 프로그램을 컴파일 및 실행하게 되면 프로그램의 결과와 아울러 애니메이션 코드가 자동으로 생성된다. 이 애니메이션 코드는 SVG 형식으로 생성되므로, 관찰자는 작업 환경에 구애받지 않고 별도의 소프트웨어를 필요로 하지도 않으면서 웹상에서 시각화 표현을 확인할 수 있으며, HTML 및 다른 XML 언어들과 결합하여 다양하게 응용할 수 있다. 예를 들어, HTML 문서 내에 코드 파일을 삽입하고자 한다면 간단히 "<embed src="i-norder.svg" ... type="image/svg+ xml"/>"와 같은 형식으로 사용할 수 있다.

3.2 시각화 스크립트 구성 요소

시각화 스크립트는 크게 객체를 생성하는 명령과 객체에 대한 액션을 설정하는 명령으로 구분되며, 변수, 상수, 수식, 주석문 등을 포함한다. 각 명령을 나타내는 문장은 C 프로그램의 속성과 동일하게 항상 세미콜론(;)으로 끝낸다.

3.2.1 기본 요소

스크립트 내에 데이터 저장 영역을 확보하기 위하여 변수를 선언할 수 있다. 변수의 선언규칙은 C 언어의 변수 정의 규칙과 동일하게 적용된다. 즉, 변수의 이름은 문자, 숫자, 밑줄을 포함할 수 있으며, 첫 번째 문자는 영문자가 되어야 한다. 또한 대소문자를 구분하며, C 및 스크립트에서 정의된 예약어는 사용할 수 없다. 스크립트에서는 Var, Line, Text, Rect, Circle, Ellipse, Polygon, Path, display, move, motion. changeColor, delete, anim, set를 예약어로 정의하고 있다.

변수를 선언하는 명령은 *Var*(*변수명*)이며, 생성되는 변수는 초기값이 0인 정수형 변수로 간주된다.

또한 별도의 변수를 선언하지 않고, 소스 프로그램에서 선언 및 사용된 변수의 값을 시각화 명령 내에서 사용하고자 할 경우 "$변수명"의 형태로 표기

하고 사용할 수 있다. 이 경우 정수형으로 자동 형변환된 값이 사용되며, 시각화 명령 내에서 이 변수의 값을 변경할 수는 없다.

호출된 변수는 연산에 이용되는 경우와 문자열과 결합되어 이용되는 경우 두 가지로 구분되어 사용될 수 있다. "450 - ($i * 50)"과 같이 사용되는 경우는 변수의 값을 연산에 이용하여 값을 산출하게 되므로, 프로그램 내에서 변수 i의 값이 1이라고 가정하면 위 수식의 결과는 400이 된다. 그러나 "r$j"와 같이 다른 문자열과 결합된 형태인 경우에는 변수의 값으로 치환한 새로운 문자열을 생성하게 되어, 변수 j의 값이 1이라면 위 문자열은 "r1"으로 변환된다.

상수는 스크립트 내에 직접 포함되는 실제 상수만을 허용한다. 정수형 및 실수형(소수점 포함) 상수는 그대로 표기하고, 색상값과 문자열 상수는 " "으로 묶어서 표기한다. 색상값은 white, red, black 등과 같이 문자열로 표시하거나, #ffffff, #ff0000, #000000 등과 같이 "#RGB"로 표시할 수 있다.

스크립트 내에서는 할당 연산자 '='와 이항 산술 연산자 '+', '-', '*', '/' 만을 허용하며, 연산의 우선순위는 *, / ⇒ +, - ⇒ = 의 순서로 감소한다.

두 개의 슬래시(//)는 해당 문장의 이후 부분이 주석문이라는 것을 표시한다. 애니메이션 코드 내에 설명문을 생성하도록 하는 명령으로, 주석문의 내용은 무시되므로 실제 시각적 표현에는 아무런 영향이 없다.

3.2.2 객체 생성 요소

기본 객체로 선, 텍스트, 원, 타원, 사각형, 다각형, 경로를 생성하며, 여러 객체를 그룹으로 묶을 수 있다. 식별자(id)는 각 객체 및 그룹을 구별하며 액션 설정시 이용된다. 각 객체는 표시될 좌표 및 크기를 선언함으로써 생성되며, 객체는 선색(stroke)과 면색(fill)을 함께 지정할 수 있으나 객체 선언시 색상을 지정하지 않을 시에는 기본값으로 선색을 검정(black), 면색을 밝은 회색(lightgray)으로 생성한다. 단, 경로는 화면에 표시하기 위한 객체는 아니며, 다른 객체들이 움직일 경로를 표시하기 위한 목적으로 생성되므로 색상을 없음(none)으로 설정한다.

· **선 생성** : 좌표($x1$, $y1$)부터 ($x2$, $y2$)까지 이어지는 직선을 그린다.

```
[형식] Line(id, x1, y1, x2, y2 [, stroke])
[ 예 ] /** Line("l1", 30, 30, 70, 70, "red"); **/
```

· **텍스트 생성** : 좌표(x, y)에 문자열 *string*을 표시한다.

```
[형식] Text(id, x, y [, font-size, stroke-width,
        stroke, fill,] string)
[ 예 ] /** Text("t1", 50, 30, 15, 0.5, "red", "yellow",
        "Simple Text"); **/
```

· **사각형 생성** : 좌표(x, y)에 너비 *width*, 높이 *height*인 사각형을 그린다.

```
[형식] Rect(id, x, y, width, height [, stroke, fill])
[ 예 ] /** Rect("r1", 20, 30, 70, 50, "red", "yellow"); **/
```

· **원 생성** : 좌표(cx, cy)에 반지름 r인 원을 그린다.

```
[형식] Circle(id, cx, cy [, r, stroke, fill])
[ 예 ] /** Circle("c1", 50, 50, 30, "red"); **/
```

· **타원 생성** : 좌표(cx, cy)에 x축 반지름 rx, y축 반지름 ry인 타원을 그린다.

```
[형식] Ellipse(id, cx, cy, rx, ry [, stroke, fill])
[ 예 ] /** Ellipse("e1", 50, 50, 40, 20); **/
```

· **다각형 생성** : 정점의 개수(n)만큼 좌표($x1$, $y1$), ($x2$, $y2$) [, ($x3$, $y3$), ...]를 잇는 다각형을 그린다.

```
[형식] Polygon(id, n, x1, y1, x2, y2 [, x3, y3, ...,
        stroke, fill])
[ 예 ] /** Polygon("p1", 5, 48, 16, 16, 96, 96, 48,
        0, 48, 80, 96); **/
```

· **경로 생성** : 좌표($x1$, $y1$), ($x2$, $y2$) [, ($x3$, $y3$), ...]를 잇는 경로를 설정한다. 객체의 이동경로를 지정하기 위해서만 사용되므로 화면에 표시하지 않는 것을 기본으로 한다. 따라서 경로는 선색과 면색을 모두 없음(none)으로 지정한다.

```
[형식] path(id, x1, y1, x2, y2 [, x3, y3, ...,])
[ 예 ] /** Path("pt1", 10, 10, 50, 50, 80, 20); **/
```

· **그룹 생성** : 여러 객체를 그룹으로 지정하고자 하는데 사용된다.

```
[형식] { id
        객체 생성 명령1;
        객체 생성 명령2;
          ⋮
        }
[ 예 ] /** { "base"
        Line("hor_l", 10, 90, 90, 90);
        Line("ver_l", 50, 10, 50, 90);
        } **/
```

3.2.3 액션 설정 요소

생성된 각 객체들의 속성을 변경하거나 행위를 지정하기 위해서는 액션 설정 명령어를 사용한다. 액션 설정 명령을 이용하여 객체를 화면에서 제거하거나 다시 표시되게 할 수 있으며, 객체의 색상이나 위치, 크기 등의 속성을 변경할 수 있다. 변경되는 속성을 애니메이션으로 설정하고자 하는 경우 또는 해당 액션에 대한 시간을 조정하고자 하는 경우에는 객체 생성 명령이나 액션 설정 명령 삽입 시 & 기호를 함께 표기한 후, anim 또는 set 명령을 사용한다.

· **객체 표시** : 생성된 객체를 그래픽 화면에 표시하고자 할 때 display 명령을 사용한다. 이 때 생성시의 좌표에서 x축으로 *x*, y축으로 *y* 만큼 이동하여 표시하며, 표시될 좌표를 생략하는 경우에는 객체 생성시의 좌표에 그대로 표시한다.

```
[형식] display(id [, x, y])
[예1] /** display("base"); **/
      ⇒ 객체 base를 원래 선언된 위치에 표시
[예2] /** display("base", 200, 0); **/
      ⇒ 객체 base를 선언된 위치에서 x축으로 200
      만큼 이동하여 표시
```

· **객체 이동** : 그래픽 화면상에 표시된 객체를 좌표에 따라 이동시키고자 할 경우 move 명령을 사용하며, 형식은 다음과 같다.

```
[형식] move(id, x, y)
[ 예 ] /** Ellipse("e1", 40, 50, 30, 20); **/
       /** display("e1"); **/
       /** move("e1", 30, -30); **/
       ⇒ 객체 e1을 표시된 위치에서 x축 오른쪽으
       로 30, y축 위로 30만큼 이동
```

· **경로에 따른 이동** : 객체를 경로에 따라 이동하고자 할 경우 motion 명령을 사용할 수 있으며, 이동 경로는 path로 생성되어 있어야 한다.

```
[형식] motion(id1, id2)
[ 예 ] /** Circle("c1", 0, 0, 3, "red"); **/
       /** Path("pt1", 10, 10, 50, 50, 80, 20); **/
       /** motion("c1", "pt1"); **/
       ⇒ 객체 c1을 경로 pt1에 따라 이동
```

· **객체 제거** : 화면에 표시된 객체를 화면에서 제거하는 명령은 delete이며, 제거된 객체는 재생성하지 않아도 display 명령을 이용하여 다시 화면에 표시할 수 있다.

```
[형식] delete(id)
[ 예 ] /** delete("e1"); **/
```

· **색상 변경** : 표시된 객체의 면색을 변경하고자 하는 경우 changeColor 명령을 사용한다.

```
[형식] changeColor(id, color)
[ 예 ] /** Ellipse("e1", 20, 20, 40, 20); **/
       /** display("e1", 30, 30); **/
       /** changeColor("e1", "blue"); **/
       ⇒ 객체 e1의 색상을 파란색으로 변경한다.
```

· **액션 시간 설정** : & 표시를 갖는 액션 설정 명령과 결합하여 해당 객체에 대해 액션이 적용되는 시간을 set 명령으로 조정할 수 있다.

```
[형식] set(begin [,dur])
[ 예 ] /** Rect("r1", 10, 10, 70, 50, "green"); **/
       /** &display("r1"); **/
       /** set(3); **/
       ⇒ 애니메이션 시작 3초 후에 객체 r1을 display
       한다.
```

· **속성 변경** : anim 명령은 & 표시를 갖는 객체 생성 명령과 결합하여 해당 객체의 속성을 점진적으로 변경하며, 애니메이션 후에는 변경된 속성으로 유지된다.

```
[형식] anim(attribute, from, to, begin [, dur, re-
       peat])
[ 예 ] /** &Rect("r1", 10, 10, 70, 50, "green"); **/
       /** anim("width", 70, 20, 1, 4, 2); **/
       ⇒ 객체 r1의 너비(width)를 70에서 20만큼
       변경하되, 1초에 시작하여 4초간 애니메이션
       을 2회 반복한다. 즉 이 경우 애니메이션이 1
       회 동작하는 시간은 2초가 된다.
```

4. 적용 사례

이진 트리에서 각 노드를 방문하는 것을 순회하고 하며, 한 노드에서 취할 수 있는 행동은 왼쪽 서브 트리로의 이동(L), 현재 노드 방문(D), 오른쪽 서브 트리로의 이동(R)의 3개이다. 오른쪽 서브 트리보다 왼쪽 서브 트리를 먼저 방문해야 한다는 전제하에 순회의 방법은 DLR, LDR, LRD의 세 가지가 있으며, 이를 각각 전위(preorder), 중위(inorder), 후위(postorder) 순회라 한다.

중위 순회 알고리즘은 다음의 각 단계를 따르며, 각 서브 트리에서도 동일 알고리즘을 순환 적용함으

로써 전체 노드를 방문할 수 있게 된다.

Step 1. 왼쪽 서브 트리를 중위 순회한다.

Step 2. 루트 노드를 방문한다.

Step 3. 오른쪽 서브 트리를 중위 순회한다.

다음은 이진트리 순회 알고리즘에 따라 C 프로그램을 작성하고, 프로그램 내에서 시각화할 내용과 위치를 결정하여 삽입한 "/** 시각화 명령 **/"과, 프리프로세서와 컴파일 및 실행의 단계를 거쳐 생성된 애니메이션 코드 파일의 일부이다.

〔C 프로그램에 삽입된 시각화 명령〕

```
  for(i = 1; i < MAX; i++) {
    n = rand() % 100;
    new = (LINK)malloc(sizeof(NODE));
    new->left = NULL;
    new->right = NULL;
    new->data = n;
    if (n <= current2->data)
       current2->left = new;
    else
       current2->right = new;
    current1 = head;
```

```
/** Line(l$i, $x2, 15+$y2, $x1, $y1); **/
/** &display(l$i); **/
/** set($s_time); **/
/** { node$i$l$r
   Circle(c$i, $x1, $y1, 15);
   Text(t$i, $x1, 7+$y1, 20, "black", "black",
$n);
   } **/
/** &display(node$i$l$r); **/
/** set($s_time); **/
```

①

```
  s_time++;
  }

void visit(LINK t)
{
  int i, value;
  count++;
  value = t->data;
  for(i = 0; i < MAX; i++)
     if (value == loc[i]) break;
```

```
/** &changeColor(node$i$l$r, "lightpink");
**/
/** set($s_time); **/
/** Text(v$i, 50 * $count, 100, 20, "green",
$value); **/
/** &display(v$i); **/
/** set($s_time); **/
```

②

```
  s_time++;
}
```

〔생성된 애니메이션 코드〕

① 새로운 노드를 트리에 추가

```
<defs><g>
<line id="l1" x1="300" y1="165" x2="450" y2="200"
style="stroke-width:2;"/>
</g></defs>
<use xlink:href="#l1" class="lightgraish" style=
"visibility:hidden;">
<set attributeName="visibility"
attributeType="CSS" to="visible" begin="2"
dur="1s" fill="freeze"/>
</use>
<defs><g id="node112">
<circle id="c1" cx="450" cy="200" r="15"/>
<text id="t1" x="450" y="207"
style="font-size:20; stroke:black; fill:black;
text-anchor:middle;">97</text>
</g></defs>
<use xlink:href="#node112" class="lightgraish"
style="visibility:hidden;">
<set attributeName="visibility"
attributeType="CSS" to="visible" begin="2"
dur="1s" fill="freeze"/>
</use>
        ⋮
```

② 순회 알고리즘에 따라 방문된 노드의 색상을 변경하고 그 값을 트리 상단에 표기

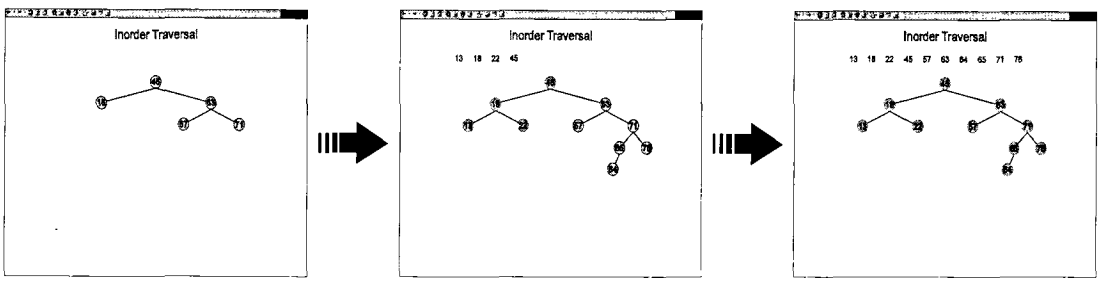


그림 3. 이진 트리의 순회 애니메이션

```
<use xlink:href="#node221" x="0" y="0"
style="fill:lightpink; visibility:hidden;">
<set attributeName="visibility"
attributeType="CSS" to="visible" begin="11"
dur="1s" fill="freeze"/>
</use>
<defs><g>
<text id="v1" x="50" y="100" style="font-size:20;
stroke:green; text-anchor:middle;">7</text>
</g></defs>
<use xlink:href="#v1" class="lightgraish" style=
"visibility:hidden;">
<set attributeName="visibility"
attributeType="CSS" to="visible" begin="11"
dur="1s" fill="freeze"/>
</use>
          ⋮
```

순회의 방법을 정의한 변수의 값만 변경하면 전위, 중위, 후위 순회의 결과를 다양하게 확인할 수 있다. 본 예제에서는 데이터의 값을 랜덤하게 발생시켜 이진 트리를 구성하게 되므로 때때로 편향된 이진 트리가 발생할 수 있다.

5. 타 시스템과의 비교

소프트웨어 시각화 중 특히 알고리즘 시각화 연구는 알고리즘의 동작을 이해하기 위한 시각화와 애니메이션을 생성하는데 집중되어 있다. 일반적으로 이런 시각화는 자동으로 생성되지 않으며, 각 알고리즘에 대한 적절한 표현을 설계하기 위한 애니메이터를 필요로 한다. 표 1에서 시각화 생성 방법을 2.2절의 기준으로 정리하였다.

표 1의 실행환경 측면에서 POLKA, ZEUS와 같은 구형 시스템들은 특별한 실행환경을 요구하였으나, 최근의 시스템들은 웹 환경을 채용하였음을 알 수 있다. 그러나 지금까지의 모든 시스템들은 단순히 디스플레이 목적으로 웹을 사용하였을 뿐이며, 시각화 이외의 객체들과 연동하는 기능은 제공되지 않았다. 본 제안시스템은 XML 기반의 기술인 SVG를 사용함으로 인해서 모든 XML 응용과 밀접하게 결합될 수 있어서 확장성면에서 장점이 있다.

표 1. 기존 시스템과의 비교

시스템	생성 방법	실행환경	확장성
POLKA	라이브러리	제한	×
JSAMBA	스크립트	Web	×
ZEUS	Zeus 코드	제한	×
JAWAA	스크립트	Web	×
제안 시스템	스크립트 자동생성	Web	○

6. 결 론

프로그램의 각 단계가 수행되는 과정을 동적으로 시각화한다는 것은 개발자의 입장에서 볼 때 많은 시간이 요구되는 작업이다. 기존 시스템들은 대부분 특정 알고리즘이나 프로그램에 대해 제한적인 경향을 가지거나, 특정 환경이나 소프트웨어에 대한 제약이 있어 적합한 시각화 효과를 생성하는 데 불편함을 초래하였다. 따라서 본 연구는 이러한 시각화 과정을 보다 쉽고 간단하게 하고, 다양한 알고리즘을 시각화 할 수 있도록 프로그램 내에 삽입되어 애니메이션 코드를 생성할 수 있는 시각화 스크립트를 설계하고 구현하였다.

제안한 시스템은 프로그래머에 의해 작성된 C 프로그램 내에 애니메이터가 시각화를 위한 스크립트들을 삽입하고, 프리프로세서를 거친 후 컴파일 및 실행함으로써 C 프로그램의 결과와 함께 SVG 포맷의 애니메이션 코드를 생성하여 관찰자로 하여금 시각화 표현을 학습할 수 있도록 하는 것으로, 본 시스템의 특징은 다음과 같다.

첫째, 스크립트가 "/** 시각화 명령 **/"의 형태로 삽입되므로 시각화 효과를 원하지 않는 경우 일반적인 방법으로 컴파일하게 되면 주석으로 인식되어 소스 프로그램에 대해 영향을 미치지 않는다.

둘째, 시각화 명령의 삽입 방법이 결정적이지 않으므로 사용자의 경험이나 프로그램 이해도에 따라 시각화 효과가 다소 차이가 날 수 있으나, 다양한 방법으로 시각화 할 수 있으므로 융통성과 확장성을 가진다.

셋째, 애니메이터가 직접 SVG 포맷의 애니메이션 코드를 생성하지 않고 정의된 명령어를 이용하여 애니메이션 코드를 자동으로 생성할 수 있으므로 작업의 부담을 경감시킬 수 있다.

넷째, 결과로 생성된 애니메이션 코드는 SVG 포맷이므로 특정 시스템 환경이나 소프트웨어를 필요로 하지 않으며, 웹 브라우저에서 확인할 수 있으며, 조작방법에 대한 교육 등을 필요로 하지 않는다.

다섯째, 소스 프로그램 내의 변수를 특정 형태로 불러 사용할 수 있도록 함으로써 실제 프로그램 실행 시 발생하는 정보를 이용할 수 있다. 즉, 입력 데이터의 변경이나 이에 따른 자료구조의 변경도 즉시 반영할 수 있다.

여섯째, 생성된 SVG 코드는 다른 XML 기반언어 및 HTML, 스크립트 등과 연동하여 다양한 방법으로 활용이 가능하다.

본 논문에서 제안하는 시각화의 방법이 정의된 스크립트 언어를 C 프로그램 내에 삽입하여 프리프로세서를 거치도록 설계되었으므로, 애니메이터는 작성된 C 프로그램을 분석하고 시각화 명령 및 이를 삽입할 적절한 위치 등을 판단해야 하는데, 이를 위해서는 애니메이터에게도 프로그래밍 경험이 요구된다고 할 수 있다. 따라서 제안한 시스템은 프로그래머가 직접 애니메이션 작업을 하는 것이 가장 효과적인 방법으로 사용자가 알고리즘을 효율적으로 이해하도록 지원하는 교육 분야에 적합할 것으로 판단된다.

본 논문에서 정의된 시각화 스크립트 명령은, SVG 포맷을 생성하기에 용이한 형태로 정의되어 있으므로 SVG에 대한 경험이 없는 사용자들도 본 시각화 명령을 보다 효과적으로 적용하기 위해서는 생성되는 애니메이션 코드에 무관하며, 자연어에 좀 더 가까운 언어로 발전시켜 나가야 할 것이다. 또한 SVG 자체 특성상 3차원 그래픽이나 멀티미디어에 대한 기능이 부족하므로, 향후 이의 업데이트에 따른 제안 시스템의 보완도 병행되어야 할 것이다.

참 고 문 헌

[1] C. A. Shaffer, M. Cooper, and S. H. Edwards, "Algorithm visualization: a report on the state of the field," Proceedings of the 38th SIGCSE technical symposium on Computer science education SIGCSE '07, 150-154, 2007.

[2] S. Asija, "Visualization of Object-Oriented Design Models," A Depaul University Graduation Thesis for the Degree Masters, Chicago IL, 1999.

[3] 이신주, 프로그램 시각화 시스템을 위한 애니메이션 코드 자동 생성기의 설계 및 구현, 창원대학교 석사논문, 2000.

[4] C. Evans and N. J. Gibbons, "The interactivity effect in multimedia learning," *Computers & Education*, Vol.49, No.4, 1147-1160, 2007.

[5] E. Lahtinen and T. Ahoniemi, "Kick-Start Activation to Novice Programming - A Visualization-Based Approach," *Electronic Notes in Theoretical Computer Science* (ENTCS), Vol.224, 125-132, 2009.

[6] C. D. Hundhausen and J. L. Brown, "Designing, visualizing, and discussing algorithms within a CS 1 studio experience: An empirical study," *Computers & Education*, Vol.50, No.1, 301-326, 2008.

[7] O. Seppala and V. Karavirta, "Work in Progress: Automatic Generation of Algorithm Animations for Lecture Slides," *Electronic Notes in Theoretical Computer Science* (ENTCS), Vol.224, 97-103, 2009.

[8] Y. Watanobe, N. Mirenkov, and R. Yoshioka, "Algorithm library based on algorithmic cyberFilms," *Knowledge-Based Systems*, Vol.22, No.3, 195-208, 2009.

[9] J. H. Cross, T. D. Hendrix, D. A. Umphress, L. A. Barowski, J. Jain, and L. N. Montgomery, "Robust Generation of Dynamic Data Structure Visualizations with Multiple Interaction Approaches," *ACM Transactions on Computing Education*(TOCE), Vol.9, No.2, 1-32, 2009.

[10] M. Brown, "Zeus: A System for Algorithm Animation and Multi-View Editing," *Computer Graphics*, Vol.18, No.3, 177-186, 1992.

[11] XTANGO, http://www.cc.gatech.edu/gvu/softviz/algoanim/xtango.html/

[12] POLKA, http://www.cc.gatech.edu/gvu/softviz/algoanim/polka.html/

[13] SAMBA, http://www.cc.gatech.edu/gvu/softviz/algoanim/samba.html/

[14] JSAMBA, http://www.cc.gatech.edu/gvu/softviz/algoanim/jsamba/

[15] S. Rodger, "Using Hands-On Visualizations to Teach Computer Science from Beginning

Courses to Advanced Courses," Second Program Visualization Workshop, Hornstrup Centert, Denmark, 2002.

[16] G. Rößling and B. Freisleben, "Approaches for Generating Animations for Lectures," AACE 11th International Society for Information Technology and Teacher Education Conference, 809-814, 2000.

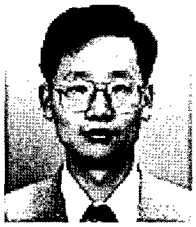
[17] A. Watt, C. Lilley, D. Ayers, R. George, C. Wenz, T. Hauser, K. Lindsey, and N. Gustavsson, *SVG Unleashed*, Sams Publishing, 2003.

이 향 숙

1990년 3월~1994년 2월 창원대학교 전자계산학과 이학사

2003년 3월~2005년 2월 창원대학교 컴퓨터공학과 공학석사

관심분야 : 알고리즘, 컴퓨터활용교육



이 수 현

1983년 3월~1987년 2월 광운대학교 전자계산학과 이학사

1987년 3월~1989년 2월 한국과학기술원 전산학과 공학석사

1989년 3월~1994년 8월 한국과학기술원 전산학과 공학박사

1994년 9월~1996년 2월 한국전자통신연구소 Post Doc. 및 선임연구원

1996년 3월~현재 창원대학교 컴퓨터공학과 교수

관심분야 : 프로그래밍언어, 컴파일러, 알고리즘