제22회 한국정보처리학회 추계학술발표대회 논문집 제11권 제2호 (2004. 11)

알고리즘 시각화를 위한 SVG 코드 생성기의 설계 및 구현

이향숙, 이수현
창원대학교 컴퓨터공학과
e-mail:minomina@kornet.net

A Design and Implementation of SVG Code Generator for Algorithm Visualization

Hyang-Sug Lee, Su-Hyun Lee
Dept. of Computer Engineering, ChangWon National University

요 약

일반적으로 프로그램의 수행과정은 직관적으로 파악하기 어려워, 이에 대한 이해를 돕기 위하여 시각화 분야에 대한 다양한 연구가 필요하다. 시각화는 프로그램의 디버깅이나 개선에도 효과적으로 이용될 수 있을 뿐 아니라 컴퓨터 교육 분야에서도 폭넓게 활용될 수 있다. 본 논문에서 제안하는 시스템은 C 언어로 작성된 프로그램에 간단한 시각화 명령을 추가하고 이를 자동 변환하여 SVG 애니메이션 코드를 생성한다. 생성된 애니메이션 코드는 웹 브라우저 상에서 실행될 수 있으며 알고리즘 시각화를 위한 별도의 프로그램 작성 없이 기존 알고리즘에 간단한 명령어를 추가하여 시각화할 수 있다는 장점이 있다.

1. 서론

프로그래머들은 기능, 구조, 처리 등을 묘사하고, 알고리즘을 고안하고, 표현하고, 발표하기 위해 그림이나 도표들을 사용해왔다. 그러나 준비된 기호, 이미지, 도표, 애니메이션 등을 사용한다면, 프로그래머에 의한 형식적이고 자연적인 언어들보다 더 정확하고 더 효과적으로 정보를 전달할 수 있다. 이렇게 프로그램의 표현을 향상시키는데 초점을 두고 있는 것이 소프트웨어 시각화이다. 소프트웨어 시각화 도구는 프로그래머의 노력을 감소시키고, 소프트웨어 개발시 성능향상과 유지보수에 기여한다[1-3].

일련의 애니메이션 시스템은 각자의 개발 환경에 따라 각각 특색있는 결과들을 만들어 내고 있으며, 현재까지 대부분의 연구들은 각 대학을 중심으로 이루어지고 있다. 그 중 대표적인 예인 BALSA는 개인 워크스테이션 기반의 XWindows 상에서 실행되는 응용프로그램으로 수년간 알고리즘 디자인과 분석 및 수업 보조 도구로써 이용되었으나, 사용자가 새로운 화면을 디자인하거나 뷰 윈도우의 내용을 변경하기 어렵다. 이를 발전시킨 ZEUS는 객체지향적 설계와 그래픽적인 설명 및 동기화된 다중 뷰를 제공하나 실행에 따른 장소의 제한이 있다. XTango는 알고리즘의 동작 명령어 결과를 가지고 애니메이션을 빠르게 생성할 수 있으나, 자료 구조를 표현하기에는 부족하다[2]. 이 밖에도 Polka, SAMBA 등의 시스템이 좀더 발전된 형태로 개발되어 왔으며, 국내에서도 ZEUS, Xtango, Polka 형식을 기반으로 시각화에 대한 시스템들이 구현되고 있으나, 이런 시스템들은 대부분 개발자에 의해 시각화 프로그램의 내용이 고정되어 있어서 사용자가 원하는 내용으로 변경할 수 없으며, 특정 알고리즘을 이해하기 위하여 입력 데이터가 고정된 스크립트만을 이용하여 시각화되어지는 단점이 있다.

제안 시스템은 이미 작성되어 있는 소스 프로그램에 간단한 시각화 명령을 추가하고 이를 프리프로세서를 거쳐 컴파일하고 실행하게 되면, 프로그램의

실행 결과와 함께 각 단계의 실행 과정과 변화되는 자료 구조 등을 보다 쉽고 다양하게 시각화하기 위한 SVG 애니메이션 코드를 자동 생성한다. 이는 특정 알고리즘 애니메이터에 결합되지 않고 사용자의 환경에도 구애받지 않고 사용할 수 있으므로, 사용자가 보다 다양하게 시각화면을 구성할 수 있도록 구현되었으며, 결과로 생성되는 시각화 코드는 웹 브라우저에서 자유롭게 활용될 수 있다.

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 결론과 향후 연구 과제를 제시한다.

2. 알고리즘 시각화

2.1 시각화 개요

소프트웨어 시각화는 소스 코드 리스트에서 추상화에 이르기까지 소프트웨어 시스템을 시각적으로 표현하는 모든 것을 포함한다. 소프트웨어 시각화는 전형적으로 알고리즘 시각화와 프로그램 시각화의 영역에서부터 데이터 시각화와 코드 시각화에까지 이른다[1].

현재 소프트웨어 시각화에 대한 연구는 다양하게 진행되어 오고 있다. 그 주요 주제 중 하나는 알고리즘 시각화 프레임워크와 시스템 개발이며, 이런 시스템들은 알고리즘 설계 및 분석 교육에 활발히 사용되어 왔다. 또다른 주제는 프로그램 디버깅과 실행 추적에 대한 응용프로그램들이다. 또한 최근에는 코드의 구조적 시각화에 대한 연구도 점차 증가하고 있는데, 이는 코드에 대한 거시적인 시각에 집중하는 것으로, 전형적인 예는 함수 호출도, 제어 흐름도, 파일 의존 및 상속 계층도 등이다. 추가로, 소프트웨어 시각화는 그 실행 활동에 집중하는 병렬 프로그램 시각화와 성능 측정, 소프트웨어 공학 계량에도 활용되고 있다[1,2].

그러나 본 논문에서는 알고리즘 시각화에 관한 사항만 논의하기로 한다.

알고리즘 시각화 연구는 알고리즘 운영의 이해에 도움을 주기위한 시각화와 애니메이션을 생성하는데 집중되어 있다. 일반적으로 이런 시각화는 자동으로 생성되지 않으며, 각 알고리즘에 대한 적절한 표현을 설계하기 위한 애니메이터를 필요로 한다. BALSA, XTango, Polka, ANIM, JAWAA 등과 같은 많은 시스템들이 알고리즘 애니메이션을 위해 개발되어 왔다. 그러나 이런 시스템들은 대부분 사용자가 디스플레이를 최적화하기 어려우며, 디스플레이가 기반 알고리즘에 제한적이고, 특정 환경이나 애플리케이션을 필요로 한다[1,2].

2.2 SVG

1998년 Adobe, Sun, Netscapes가 함께 벡터 그래픽을 위한 웹 문서 표준으로 포스트 스크립트에 기반한 PGML을 제안하고, Macromedia와 Microsoft가 VML을 제안하였다. W3C는 이들을 기반으로 새로운 플랫폼을 제정하기로 결정하고, 1998년 8월 Adobe, Macromedia, IBM, 코랠, 애플, HP, Microsoft, Autodesk, Netscape, CSIRO 등 많은 기업들이 참여하는 SVG 작업그룹을 창설하였다. 1999. 2월 처음 작업초안이 발표된 이래, 2001. 9월 SVG 1.0 spec, 2003. 1월 SVG 1.1 spec와 Mobile SVG spec, 2003. 7월 SVG 1.2 spec(Working Draft)이 발표되는 등, 지속적인 연구와 논의가 진행되고 있다[5,6].

XML 그래픽 표준인 SVG는 XML의 개방성(Openness), 상호운용성(Interoperability)등의 장점을 수용한 벡터 그래픽이므로, SMIL, GML, MathML 등 다른 XML 언어들과 결합하여 다양한 응용의 개발이 가능하고, HTML과 스크립트 등 기존의 웹 개발 툴과 상호운용이 가능하다. 따라서 지도, 웹 기반 GIS, 전자책, 문서출판, 애니메이션, 광고, 전자상거래, 교육 등 그래픽이 많이 사용되는 분야에 적용될 수 있다[5-7].

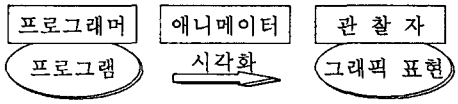
또한 SVG는 텍스트로 기술되기 때문에 클라이언트의 입장에서 내용을 이해하기 쉽고, 그래픽에 대한 검색 및 편집이 편리하며, 애플리케이션들이 SVG 문서를 쉽게 사용할 수 있다. 그리고 XML과 SVG의 DOM 인터페이스를 통하여 모든 그래픽 요소에 쉽게 접근할 수 있으므로 데이터베이스와 연동하여 웹 그래픽 문서를 동적으로 생성할 수 있어, 지리정보 시스템과 같이 그래픽이 많이 사용되면서 이에 대한 동적이고 상호작용적인 인터페이스가 강조되는 분야를 중심으로 기술개발이 활발하게 진행되고 있다[5,6].

본 제안 시스템에서는 애니메이션 코드를 SVG 포맷으로 자동 생성하도록 하여, 결과를 웹 브라우저에서 쉽게 확인할 수 있을 뿐 아니라, 나아가 다른 애플리케이션과도 연동이 가능하다.

3. 제안 시스템 설계 및 구현

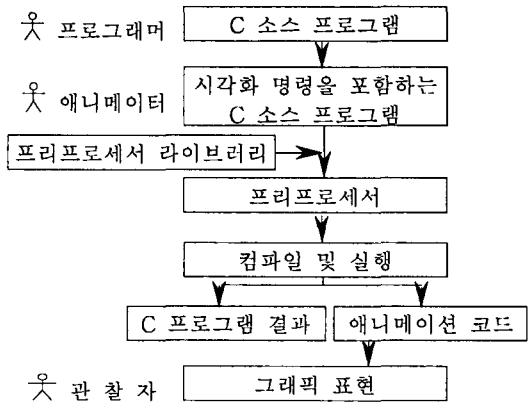
3.1 시스템 개요

일반적인 시각화 모델에서 프로그래머는 처음 소스 프로그램을 작성한 사람을 말하며, 애니메이터는 프로그래머가 작성한 프로그램을 시각적 형태로 나타내 주는 사람, 관찰자는 최종적으로 시각화된 프로그램을 학습하는 사람을 말한다. 대체로 프로그래머와 애니메이터는 동일인일 수 있으며, 디버깅의 용도로 시각화를 한 경우에는 프로그래머와 애니메이터, 관찰자가 모두 동일인일 수도 있다[2,4].



<그림1> 프로그램 시각화 모델

제안 시스템에서는 프로그래머에 의해 작성된 일반적인 C 소스 프로그램의 내부에 시각화에 필요한 명령을 포함시키고, 이를 프리프로세서를 거쳐 컴파일하고 실행하게 되면 일반적인 프로그램 실행 결과와 시각화에 필요한 애니메이션 코드가 생성되게 된다. 이 애니메이션 코드는 SVG 포맷으로 작성되어지며, 웹 브라우저 상에서 시각화되므로, 다른 소프트웨어를 필요로 하지 않고 환경에 구애받지 않고 사용할 수 있다. 다음 그림은 제안 시스템의 전체 구성도이다.



<그림2> 제안 시스템 전체 구성도

3.2 시각화 명령

시각화 명령은 크게 객체를 생성할 수 있는 명령과 객체에 대한 액션을 설정할 수 있는 명령으로 구분한다. 시각화 명령은 프로그래머에 의해 작성된 소스 프로그램에 영향을 주지 않도록 주석으로 인식되게 하기 위해서 */** 시각화 명령 **/*의 형태로 프로그램 내에 삽입되었다. 또한 소스 프로그램에서 사용되는 변수를 시각화 명령에서 공용으로 사용하고자 할 경우 *$변수명*의 형태로 표시하였다.

기본 객체로 선, 텍스트, 원, 타원, 사각형, 다각형을 생성한다. 각 객체는 식별자(*id*)에 의해 구별되며, 객체가 표시될 좌표 및 크기를 함께 선언함으로써 생성된다. 객체는 선색(*color1*)과 면색(*color2*)을 함께 지정할 수 있으며, 객체 선언시 색상을 지정하지 않을 시에는 기본값으로 선색을 검정(black), 면색을 밝은 회색(lightgray)으로 생성한다.

생성된 각 객체들의 속성을 변경하거나 행위를 지정하기 위해서는 액션 설정 명령어를 사용한다. 객체의 색상이나 위치, 크기 등을 변경할 수 있으며, 변경되는 속성을 애니메이션으로 표시되도록 지정할 수도 있다. 애니메이션을 설정하고자 하는 경우에는 애니메이션의 종료시간이나 반복횟수를 함께 지정할 수 있다.

다음의 표는 객체 생성 및 객체에 대한 액션을 설정하는 시각화 명령을 보여주고 있다.

시각화 명령		그래프 표현
객체생성	*Line(id, x1, y1, x2, y2 [,color])*	좌표(*x1, y1*)부터 (*x2, y2*)까지 이어지는 직선을 그린다.
	Text(id, x, y, string [,color])	좌표(*x, y*)에 문자열 *string*을 표시한다
	Circle(id, x, y, r [,color1, color2])	좌표(*x, y*)에 반지름 *r*인 원을 그린다.
	Ellipse(id, x, y, rx, ry [,color1, color2])	좌표(*x, y*)에 *x*축 반지름 *rx*, *y*축 반지름 *ry*인 타원을 그린다.
	Rect(id, x, y, width, height [,color1, color2])	좌표(*x, y*)에 너비 *width*, 높이 *height*인 사각형을 그린다.
	Polygon(id, n, x1, y1, x2, y2 [,x3, y3, ..., color1, color2])	정점(*n*)의 개수만큼 좌표(*x1, y1*), (*x2, y2*), [(*x3, y3*), ...]를 잇는 다각형을 그린다.
액션설정	*display(id [, x, y])*	객체 *id*를 (*x, y*)만큼 이동하여 표시한다. 이 때 표시되는 객체의 속성을 변경하고자 할 경우 아래의 anim 명령을 함께 사용한다.
	anim(attribute, from, to, begin [, end, repeat])	속성 *attribute*를 초기값 *from*에서 종료값 *to*만큼 변경하되, 애니메이션 시작시간을 *begin*, 종료시간을 *end*, 반복횟수를 *repeat*로 설정한다.

<표1> 객체 생성 및 액션 설정 시각화 명령

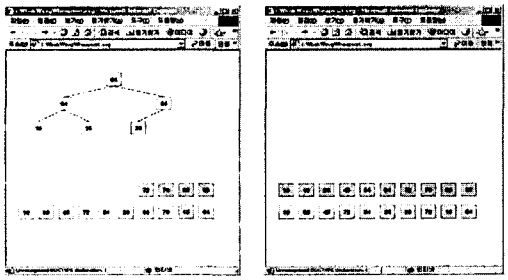
4. 적용 사례

본 절에서는 구현된 시스템을 이용하여 힙 정렬 알고리즘을 시각화하였다. 힙 정렬은 퀵 정렬과 같

이 가장 효율이 좋은 알고리즘 중의 하나이다.

먼저 입력된 데이터를 힙 상태의 이진트리로 구축하고, 루트노드를 힙 트리의 마지막 노드와 교환한 후 힙 트리의 영역을 n-1로 축소시킨다. 즉 힙 트리의 마지막 노드를 힙 구성에서 제외시키고, n-1개 항목으로 힙 트리를 재구성한다. 그리고 힙 트리가 한 개 노드로 구성될 때까지 두 번째와 세 번째 과정을 반복한다.

힙 정렬의 시각화 과정은 우선 입력되는 데이터를 각각 사각형 객체로 생성하여 화면 아래쪽에 일렬로 배치하고 다시 트리의 형태로 화면 중앙에 구성한다. 정렬이 완료된 노드를 차례로 트리에서 제거하면서 입력 데이터의 상단에 일렬로 배치하는 과정을 반복한다. 정렬이 완료되면 트리는 없어지고, 입력 데이터와 정렬된 데이터만 남게 된다. 다음 그림은 정렬의 과정과 완료 후 결과를 각각 보여준다.



<그림3> 힙 정렬 과정 및 결과

아래는 소스 프로그램 중 트리가 힙의 속성을 유지하도록 각 노드를 비교, 교환하는 부분에서 교환이 일어나는 경우, 해당 노드를 다시 화면상에 표시하도록 삽입한 시각화 명령과 생성된 시각화 코드의 일부이다.

```
/* 소스 프로그램에 삽입된 시각화 명령 */
- 중략 -
if( nLastNode != i ) {
/** { **/
/** display(item$nLastNode, $root_x, $root_y); **/
/** anim("visibility", " ", "visible", $start_time, $end_time);
**/
/** } **/
start_time += 0.5;
/** { **/
/** display(item$i, $sub_x, $sub_y); **/
/** display(line$nLastNode); **/
/** anim("visibility", " ", "visible", $start_time, $end_time);
**/
/** } **/
start_time += 0.5;
nTemp = A[i];
A[i] = A[nLastNode];
A[nLastNode] = nTemp;
heapify(A, nLastNode);
}
- 중략 -
```

```
/* 생성된 시각화 코드 */
- 중략 -
<g style="stroke:black; fill:lightgray; visibility:hidden;">
<use xlink:href="#item2" x="200" y="50"/>
<set attributeName="visibility" attributeType="CSS"
to="visible" begin="35.5s" end="39s"/>
</g>
<g style="stroke:black; fill:lightgray; visibility:hidden;">
<use xlink:href="#item0" x="100" y="105"/>
<use xlink:href="#line2" x="0" y="0"/>
<set attributeName="visibility" attributeType="CSS"
to="visible" begin="36s" end="39s"/>
</g>
- 중략 -
```

5. 결론

본 논문에서는 알고리즘 시각화 시스템을 위한 애니메이션 코드 자동 생성기를 설계, 구현하였다. 제안한 시스템은 실제 프로그램 자체의 실행에는 아무런 영향을 주지 않는 간단한 시각화 명령어를 소스 프로그램에 삽입하고 프리프로세서를 거친 후 컴파일 및 실행하면, SVG 포맷의 애니메이션 코드를 얻을 수 있다. 이 코드는 시스템이나 애플리케이션에 구애받지 않고 웹 브라우저에서 자유롭게 시각화 결과를 활용할 수 있다.

그러나 소스 프로그램에 시각화 명령을 삽입하는 부분에서 약간의 프로그래밍 경험과 SVG 포맷에 대한 기본적인 이해를 필요로 하므로, 프로그램에 대한 경험이 없는 학습자도 쉽게 입력할 수 있고 출력 코드 파일의 포맷에 구애받지 않는 시각화 명령을 개발하는 것이 향후 연구과제로 남아 있다.

참고문헌

[1] Sunita Asija, "Visualization of Object-Oriented Design Models", Depaul 대학교 석사논문, 1999.

[2] 이신주, "프로그램 시각화 시스템을 위한 애니메이션 코드 자동 생성기의 설계 및 구현", 창원대학교 석사논문, 1999. 12.

[3] 김주혁 외, "시각화를 이용한 알고리즘 교육 시스템의 설계 및 구현", 한국정보과학회논문지(C) 제4권 제3호, pp. 391~398. 1998. 6.

[4] 오진영 외, "트리구조를 위한 알고리즘 애니메이션 시스템의 설계 및 구현", 한국정보과학회 논문지(C), 제4권, 제4호, pp. 623~631, 1998. 8.

[5] Ola Andersson 외, "Scalable Vector Graphics (SVG) 1.1 Specification", W3C Recommendation, 2003. 1.

[6] 권형진 외, "웹 기반기술 표준화 종합보고서", 한국전산원 연구보고서, 2002. 12.

[7] J. David Eisenberg, "SVG Essentials", O'Reilly & Associates, 2002. 2.