구조화된 그래픽 표현을 위한 XML 기반의 SVG 저작 시스템

김택천* · 김진수* · 정회경*

SVG Editing System based on XML for Structured Graphic Representation

Tak-chen Kim* · Jin-soo Kim** · Hoe-kyung Jung*

요 약

인터넷의 급속한 발전은 기존의 정적인 웹을 탈피하여 사용자들로 하여금 더욱더 동적이고 다양한 컨텐츠를 요구하는 형태로 바뀌어 가는 실정이다. 이에 따라, SVG(Scalable Vector Graphics)는 기존의 인터넷에서 사용되던 비트맵 기반의 디스플레이 보다 훨씬 정교한 그래픽 표현을 제공하기 때문에 기능이나 장치 호환성의 문제없이 벡터 그래픽을 표현한다. 또한, 그래픽에 대한 논리적인 구조를 기술함으로써 인덱싱, 검색, 저장 또는 공유가 가능하도록 정의하고 있다. 그러나 복잡한 SVG 구문을 자세히 모르고도 편리하고 SVG 그래픽을 구현할 수 있는 SVG 저작 시스템이 요구되고 있다.
이에 본 논문에서는 SVG에 관한 기초기술 연구 및 구조화된 SVG 문서를 사용자 중심의 편집 인터페이스를 통해 일반 사용자들이 손쉽게 그래픽 객체를 직접 저작함에 따라 복잡한 SVG 구문을 자동으로 생성하는 SVG 문서 저작 시스템을 설계 및 구현한다.

ABSTRACT

A rapid development of Internet is changing users' desire from existing static contents to dynamic and diverse ones. Thus the SVG provides more affluent and sophisticated graphic expressions than an existing method based on bitmap, it can faithfully display vector graphics without scarifying any functions or the problem of device compatibility. In addition, it allows indexing, searching, storing, and sharing by the description of the logical structure of graphics. Since there are, however, very few people who know the complex SVG syntax and make use it, an editing system, which enables users to utilize the SVG graphics easily, has been in need.

Therefore, in this thesis, we do research on basic technology on the SVG, design and make an editing system for SVG documents. The system, therefore, provides uses with a user-friendly editing interface and enables them to write graphic objects easily, and generates complex SVG documents automatically.

키워드

XML, SVG, 그래픽 편집기, Vector graphics

Ⅰ. 서 론

기존 인터넷에서는 그래픽에 대한 표현의 고급화와 보편성을 제공하는 방식을 주축으로 이루어져 왔다, 그러나, 이러한 시스템들로 작성된 그래픽 문서는 표현의 고급화에 중점을 둔 나머지 정보

*배재대학교 컴퓨터공학과

접수일자 : 2004. 6. 12

의 교환과 공유라는 인터넷 기반에서의 그래픽 처리에 관한 동기를 고려하지 않고 각기 독자적인 구조로 인해 상호 호환성이나 재사용에 따른 비용 낭비 차원의 여러 문제점을 내포하고 있는 실정이다.

이에 따라, W3C에서는 인터넷상에서 벡터 그래픽 표현의 효율적인 처리와 저장 및 공유를 가능하게 하는 XML(eXtensible Markup Language)의 한 어플리케이션인 SVG(Scalable Vector Graphics)를 제정하였다[1,2,3]. SVG는 기존의 인터넷에서 사용되던 비트맵 기반의 방법보다 풍부하고 상호작용이 가능한 동적인 벡터 그래픽 표현이 가능하며 웹상에서 상호 호환성을 갖는다, 또한, 그래픽에 대한 논리적인 구조를 기술함으로써 인덱싱, 검색, 교환 또는 공유가 가능하도록 정의하고 있다.

그러나 전문적인 지식을 가지고 있지 않은 일반 사용자가 SVG 문서를 기술하는데는 어려움이 많기 때문에 일반 사용자들이 SVG 문서를 쉽고 편리하게 생성할 수 있는 시스템의 필요성이 점차 증대되고 있다.

이에 본 논문에서는 SVG에 대한 기초기술 연구 및 인터넷 상에서 사용되는 벡터 그래픽 처리를 위해 일반 사용자들이 손쉽게 그래픽 객체를 생성하고 편집할 수 있도록 그래픽 저작 시스템에 관한 구조를 정의하였다. 또한, 저작된 각 그래픽 객체들을 통해 SVG 문서를 생성하는 코드 생성기를 정의하여 벡터 그래픽을 XML 기반의 논리구조로 변환하는 SVG 문서 저작 시스템을 설계 및 구현한다.

본 논문의 구성은 다음과 같다. Ⅱ장에서는 SVG의 개념, Ⅲ장에서는 SVG DOM 처리기와 각 노드별 그래픽 객체의 렌더링 엔진 설계에 대해 기술하고, Ⅳ장에서는 구현 및 고찰에 대해 기술한다. 마지막으로 Ⅴ장에서는 결론을 기술한다.

Ⅱ. SVG 기본 개념

2.1 SVG 개요

SVG는 XML에 기반한 인터넷 상에서 그래픽을 기술하기 위한 마크업 언어로 W3C에서 제정하였다. 이의 주된 목적은 동적이고 자유롭게 변환 가능하며, 사용자와 상호 작용할 수 있는 그래픽을 플랫폼 독립적으로 표현할 수 있도록 하는데 있다[5,6]. 또한 모바일 기기에서의 그래픽 표현을 위한 SVG 모바일 프로파일이 권고 되고 있다[7,8].

SVG[9]는 해상도에 상관없이 확대와 축소가 가능한 확장성을 가지며, 벡터 그래픽이고, 네임스페이스(Namespace)를 지원하여 SMIL(Synchronized Multimedia Integration Language)[10]과의 조합이 가능하며, 다양한 형태로 표현이 가능한 특성을 갖는다.

2.2 SVG 구성

SVG는 XML 문서 이므로 XML 문서 구조를 그대로 따른다. SVG 문서는 논리적 구조와 물리적 구조를 가지고 있다. 물리적으로 SVG 문서는 개체(entity)라는 요소들로 이루어지며 개체는 다른 개체들을 참조해 그것들을 문서 안에 포함시킬 수 있다. 논리적으로 SVG 문서는 선언(Declarration), 엘리먼트(Element), 주석, 문자참조, 처리 명령으로 이루어져 있다.

2.2.1 SVG 구성요소

SVG 구성은 크게 네 가지 엘리먼트로 구분되며, 이는 다음과 같다.

(1) SVG 엘리먼트

SVG 문서의 구성은 'svg' 엘리먼트 내부에 포함된 다수의 SVG 엘리먼트들로 구성된다. 이 엘리먼트는 문서의 최상위 엘리먼트로 "<svg>"로 마크업하며, 항상 존재해야 하는 엘리먼트이다.

(2) 그래픽 엘리먼트

이는 사용자의 뷰포트(Viewport) 영역에 사각형, 원, 선 등과 같은 그래픽 객체들을 표현하기 위한 엘리먼트들로 구성된다. 그래픽 엘리먼트는 각각의 특성에 맞는 속성 값에 따라 해당 그래픽 객체를 생성한다. 그래픽 엘리먼트는 rect, circle, ellipse, line, polyline, polygon, text, image 등이 있다.

(3) 컨테이너 엘리먼트

이는 여러 그래픽 엘리먼트들을 그룹화해서 다른 엘리먼트를 포함한다. 'g' 엘리먼트의 경우는 'desc'나 'title' 엘리먼트를 포함해서 정보를 나타낸다. 또한 SVG는 URI 참조를 사용하여 다른 객체를 확장하여 만든다. 'defs' 엘리먼트의 경우는 참조된 엘리먼트들을 위한 엘리먼트에 해당한다.

(4) 기타 엘리먼트

SVG 문서는 그 밖에 참조, 부연 설명이나 상태처리 등에 사용되는 엘리먼트들로 구성된다. 이는 표 1에 나타내었다.

표 1. 기타 엘리먼트
Table 1. The others Element

엘리먼트명	설 명
desc, title	렌더링 되지 않는 텍스트 부가 정보를 정의
symbol	'use' 엘리먼트에 의해 사용되어지는 템플리트 객체를 정의
use	템플리트 객체의 인스턴스를 만들어 사용하는 엘리먼트
switch	하위의 엘리먼트 중 조건에 맞는 엘리먼트를 선택하는 엘리먼트
tspan	폰트, 색상, 위치 변경 등을 위해 'text' 엘리먼트 내에 포함시킴
tref	특정 id 속성 값을 가진 'text' 엘리먼트를 참조하여 내용을 표현

III. SVG 문서 저작 시스템 설계

본 시스템은 SVG 문서를 입력으로 받아들여 문서의 논리 구조를 처리하기 위한 SVG DOM 처리기와 각각의 그래픽 객체를 사용자 뷰포트에 렌더링 시키기 위한 렌더링 엔진으로 나뉘어진다. 그림 1은 전체 시스템의 구성도이다.

입력문서는 DOM 형태의 트리로 메모리에 생성하고 변환 결과물은 그래픽 객체의 렌더링을 위해 각 노드에 해당하는 그래픽 객체별로 분류된다. 각 객체는 템플리트로 저장 후 관리하고, 결과 객체와 속성 및 프로퍼티(Properties)들은 렌더링 객체로 변환되어 메모리에 저장한다. 각 그래픽 객체는 렌더링 엔진으로 전달되고, 렌더링 모듈에서 각각의 정보들은 렌더링 모듈 객체로 저장 관리되어 사용자의 뷰포트에 디스플레이 되는 과정을 거친다.

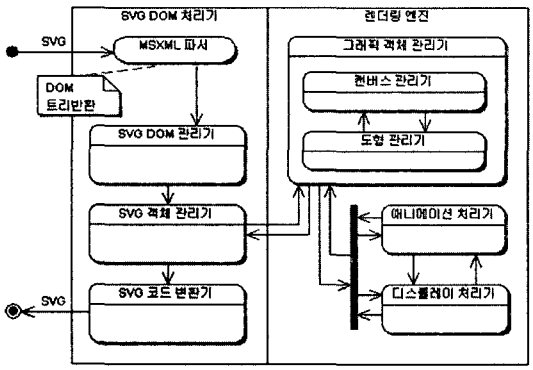


그림 1. 전체 시스템 구성도
Fig 1. Block diagram of general system

사용자 편집에 따른 그래픽 객체 정보의 변경은 템플리트에 저장된 해당 객체의 포인터를 이용해 SVG DOM 처리기의 데이터 구조를 변경시키고 변경된 정보에 따라 그에 해당하는 SVG 논리구조를 수정하게 되어 사용자가 저작한 그래픽에 해당하는 SVG 문서를 결과물로 생성하게 되는 과정을 거친다.

SVG DOM 처리기는 타 시스템에서도 사용할 수 있도록 DLL(Dynamic Linking Library) 형태로 설계하였다. 이를 이용해 트리 형태의 연결 구조나 선형 연결 구조, 또는 두 형태의 혼합된 연결 구조 등의 데이터 구조를 만들어 SVG의 논리 구조를 이용하는 다른 시스템으로의 확장이 용이 하도록 설계하였다.

3.1 SVG DOM 처리기 설계

입력된 SVG 문서는 파서 API를 통해 파싱한 결과를 메모리에 DOM 트리 형태로 저장한다. 저장된 DOM 객체는 SVG 객체 관리기를 통해 각 SVG 논리 구조에 해당하는 기본 정보를 수집하고 관리하게 된다. 객체 관리기에 수집된 정보들은 렌더링 엔진의 데이터로 입력되는 자료가 된다.

3.1.1 SVG DOM 관리기

SVG의 논리적인 정보를 처리하기 위해서는 기존의 DOM 만으로는 처리하기가 어려우므로, 본 시스템에서는 SVG 문서의 논리적인 구조상에서 필요한 정보를 추출 및 저장, 공유하기 위한 기반이 되도록 SVG DOM 관리기를 설계하였다. SVG DOM 관리기의 데이터 처리 과정 순차 흐름도를 그림 2에 보인다.

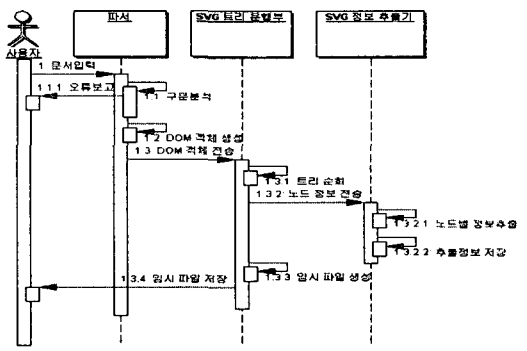


그림 2. SVG DOM 관리기의 데이터 처리 과정
Fig. 2.Data Processing Flow of SVG DOM Manager

SVG 문서가 입력되면 XML 파서를 통해 일련의 구문분석 과정을 거친 후 오류 검증이 끝난 완전한 문서의 경우 SVG 트리 운행부로 DOM 객체를 전송하고 SVG 트리 운행부에서는 전달받은 DOM 객체를 재귀순환 하면서 SVG 트리의 각 노드 정보들을 SVG 정보 추출기로 전송한다. 또한, 사용자 운영체제의 임시폴더 경로를 찾아 "~파일명.xml"과 "~파일명.svg" 두 개의 파일을 작성하게 되고, 문서의 물리적인 구조를 변경하기 위해서는 XML의 구조를 기존 DOM 인터페이스를 이용해 변경하고 구조가 변경될 때마다 임시 메모리에 저장된 SVG 파일에 변경된 내용을 실시간으로 적용하도록 관리한다.

SVG DOM 객체 트리 운행은 입력된 DOM 트리에 대해 깊이 우선 탐색(Depth First Search : DFS)으로 처리하며 각 SVG 객체의 특징에 따라 분류 후 SVG 객체 관리기를 호출한다. SVG 트리 운행부로부터 전달받은 노드 정보들은 그래픽 객체로의 변환을 위해 SVG 정보 추출기에서 각 그래픽 객체 생성을 위한 논리 정보를 추출한다. 추출된 정보들은 렌더링 엔진과의 상호작용을 위해 SVG 객체 관리기에 의해 직렬화 시킨 후 연결 리스트에 저장된다.

3.1.2 SVG 객체 관리기

SVG DOM 관리기로부터 전달받은 각각의 DOM 노드 정보들은 렌더링 엔진과의 상호작용을 위해 그래픽 객체를 위해 각 그래픽 객체의 프로퍼티에 해당하는 정보인 논리 정보로 변환되는 과정을 거쳐야 한다.

SVG 객체 관리기의 기본 클래스는 CObject를 상속받는 SVGObject 클래스에 정의되며, 이는 그림 3과 같다.

```
#ifdef _IMPORT_THIS
#define TFC_EXT_API AFX_API_IMPORT
#define TFC_EXT_CLASS AFX_CLASS_EXPORT
#else
#define TFC_EXT_API AFX_API_EXPORT
#define TFC_EXT_CLASS AFX_CLASS_IMPORT
#endif

class TFC_EXT_API SVGObject : public CObject
{
public:
        NODEDATA        NodeData;
        // 각 SVG DOM 노드들을 관리
public:
        CSVGRect* m_pRect;
        CSVGEllipse*    m_pEllipse;
        CSVGCircle*     m_pCircle;
        CSVGLine* m_pLine;
        CSVGPoly* m_pPoly;
}

TFC_EXT_API typedef CTypedPtrList <CPtrList,
SVGObject*> CSVGObject;
```

그림 3. 객체 관리기의 기본 클래스 설계
Fig. 3. Basic Class Design of Object Manager

TFC_EXT_API 정의는 DLL의 특성상 DLL Import와 DLL Export 두 가지를 동시에 사용하기 위하여 조건부 정의를 사용하였다. 이 SVGObject 클래스를 상속받는 각 그래픽 엘리먼트에 해당하는 관련 클래스들을 SVGObject와 분리시켜 선언함으로써 해당 멤버 참조만으로 데이터 구조를 관리할 수 있도록 캡슐화 시켜 설계하였고, 각 SVG 엘리먼트들을 분리하여 관리할 수 있도록 CTypedPtrList 템플리트 리스트를 전역으로 선언하여 사용하도록 하였다. 이는 모든 SVG 엘리먼트들을 기본 클래스의 멤버로 구성하는 경우 메모리 적재량의 크기가 늘어나는 점에 착안하여 설계하였다. 또한, 각 SVG 노드들의 기본 정보 및 프로퍼티 정보를 통합하여 관리하기 위해 NODEDATA 타입의 데이터 구조를 멤버로 갖는다. 그림 4는 SVG 노드 정보 저장을 위한 기본 데이터 구조의 설계를 보인다.

```
TFC_EXT_API typedef struct tagNODEDATA
{
  bool              bError;
  CString           szNodeName; // 노드의 이름
  CString           szText;     // 노드의 텍스트값
  DOMNodeType       pNodeType;  // 노드의 종류
  IXMLDOMNode*      pNode;      // 기준 노드. (삽입 시)
  UINT              nInsertType; // 삽입위치 지정.(삽입 시)
  BOOL              bExistAtt;  // 노드의 속성 존재여부.
  // 각 엘리먼트 프로퍼티 관리
  RECTELEMENT       RectElement
  CIRCLEELEMENT     CircleElement
  ELLIPSEELEMENT EllipseElement
  LINEELEMENT       LineElement
  TEXTELEMENT       TextElement
  POLYELEMENT       PolygonElement
  POLYLINEELEMENT PolylineElement
  ANIMATEELEMENT AnimateElement
}NODEDATA;
```

그림 4. SVG 노드 정보 관리를 위한 기본 데이터 구조
Fig. 4. Basic Data Suructure for Node Information Manage

이 데이터 구조는 SVG 코드 변환기와 데이터를 교환함으로써 SVG 구문을 자동으로 생성해 내는 추상적인 골격을 제시한다.

3.1.3 SVG 코드 변환기

SVG 코드 변환기는 해당 객체의 논리 정보를 이용하여 SVG 구문을 생성해 내기 위한 역할을 수행한다. SVG 코드 변환기의 데이터 처리 흐름은

입력된 객체의 논리 정보를 토대로 각각의 그래픽 엘리먼트에 해당하는 구문을 작성하기 위해 Set SVGElementNode() 함수를 호출한다. 이 함수는 파라미터로 주어진 DOMNodeType에 의해 해당 노드를 생성(createNode)하고 해당 노드에 속성(Attributes)들이 존재하거나 프로퍼티 값이 존재할 경우 SetSVGAttribute() 함수를 호출한다. 이렇게 생성된 노드는 기존 DOM 트리의 해당 위치에 삽입되기 위해 InsertSVGNode() 함수를 호출하여 각 위치별로 insertBefore 또는 appendChild 시킨 후 변경된 구조의 새로운 SVG DOM 트리를 생성한다. 변경된 SVG 구조는 임시 메모리에 저장된 SVG 문서에 실시간으로 변경된 내용이 적용되도록 설계하였다.

3.2 렌더링 엔진 설계

렌더링 엔진은 그래픽 객체들을 효율적으로 관리하기 위한 그래픽 객체 관리기와 관리된 그래픽 객체를 사용자의 뷰에 렌더링 시키기 위한 디스플레이 처리기 그리고, SVG 애니메이션을 위한 애니메이션 처리기로 분류한다. 그래픽 객체 관리기는 사용자 뷰의 물리적 크기를 논리적 크기로 변환하는 캔버스 관리기와 각 SVG 그래픽 객체들을 생성, 저장 및 편집하기 위한 도형 관리기로 분류한다. 이들 각각의 모듈이 서로 독립적으로 동작하게 함으로써 모듈의 독립성에 중점을 두어 설계하였다.

3.2.1 그래픽 객체 관리기

입력된 SVG 문서에 대해 SVG DOM 처리기에서 필요한 정보의 수집이 완료되면 렌더링 엔진의 그래픽 객체 관리기에 수집된 정보를 저장한다. 이들 정보를 토대로 그래픽 객체 관리기는 캔버스 관리기와 도형 관리기를 호출함으로써 사용자의 뷰에 렌더링되기 위한 사전 작업을 완료하게 된다. 그림 5는 그래픽 객체 관리기의 순차 흐름도를 보인다.

입력된 SVG 문서의 논리 구조 정보 수집이 완료된 SVG DOM 처리기는 캔버스 관리기와 도형 관리기로 구조 정보를 전송한다. 캔버스 관리기는 구조 정보에서 'svg' 엘리먼트의 'width', 'height' 프로퍼티 정보를 추출하여 실제 캔버스의 물리적인 크기를 논리적인 사용자 뷰포트 크기로 매핑한 후 그래픽 저작의 배경 화면이 되는 캔버스를 생성하고 매핑된 논리적인 크기를 SVG DOM 처리기에 전송하여 SVG 문서 구조 정보를 변경한다.

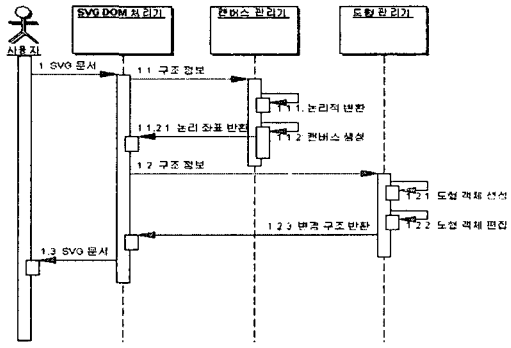


그림 5. 그래픽 객체 관리기의 순차 흐름도
Fig. 5. Sequential Flow of Graphic Object Manager

도형 관리기는 전달 받은 구조 정보를 실제 사용자 뷰포트에 렌더링 하기 위한 그래픽 객체로 생성한 후 사용자 편집에 따라 변경된 논리 정보를 SVG DOM 처리기에 전송하여 SVG 문서 구조 정보를 변경한다. 해당 SVG 그래픽 객체를 관리 및 편집하기 위해 CDrawObj 클래스를 설계하였고 이를 토대로 모든 SVG 그래픽 객체들을 제어 및 관리하기 위해 CDrawTool 클래스를 설계하였다.

3.2.2 애니메이션 관리기

애니메이션 처리기는 그래픽 객체 관리기에 의해 관리된 정적인 도형 객체를 동적인 객체로 렌더링 시키기 위한 모듈로 SVG의 애니메이션은 SMIL 표준을 따라 데이터를 처리한다[14]. 이는 또한 타임 테이블(Time Table) 형태의 사용자 인터페이스를 가지며 타임 라인(Time Line)을 이용해 애니메이션 시간을 지정하여 시간 중심으로 처리된다. 애니메이션 처리기를 사용하기 위해 SVG DOM 처리기의 애니메이션 플래그인 m_bSvgAnimate의 값을 "TRUE"로 설정하여 타임 라인에 객체를 추가하는 방식으로 설계한다.

애니메이션 처리기는 CScrollView를 상속 받은 CTimelineView를 기본 클래스로 설계하였으며 애니메이션의 실행을 위해 CTimelineView의 멤버인 m_nPlayTime을 사용한다.

사용자로부터 선택된 그래픽 객체는 SVG 애니메이션을 위해 타임 라인에 객체의 애니메이션 타임을 설정하고 타임 라인 컨트롤을 변경시킨다. 변경된 정보는 CTimeRuler 클래스에서 애니메이션 규칙을 설정하고 변경된 내용을 토대로 SVG

DOM 처리기의 해당 노드 정보를 변경한다. 사용자 인터페이스를 통해 애니메이션을 실행시키면 미리 설정된 애니메이션 정보를 그래픽 객체 관리기에 적용하여 m_nPlayTime에 설정된 시간만큼 사용자 뷰포트에서 애니메이션을 실행하도록 설계하였다.

3.2.3 디스플레이 처리기

그래픽 객체 관리기와 애니메이션 처리기에서 설정 및 변경된 정보를 토대로 SVG 그래픽을 렌더링 시키기 위해 디스플레이 처리기는 메모리 리스트에 저장된 모든 정보를 사용자 뷰포트에 연속적으로 렌더링 시킨다.

IV. 시스템 구현 및 고찰

4.1 시스템 구현

본 시스템은 IBM-PC 호환 컴퓨터(Pentium IV -2.4G)에서 개발하였으며, Windows 2000 운운영체제에서 Visual C++ 6.0을 사용하여 구현하였고, SVG 문서 파싱을 위한 파서는 MSXML 4.0을 사용하였다.

4.1.1 SVG DOM 처리기 구현

입력된 SVG 문서를 메모리에 트리 구조 형태로 생성하기 위해 MSXML 파서의 API(Application Programming Interface)를 이용하였고, 생성된 DOM 객체는 깊이 우선 방식(DFS)의 트리 탐색을 통해 템플리트 리스트에 저장하도록 구현하였다. 리스트의 해당 인덱스에 저장된 포인터를 이용하여 객체의 프로퍼티가 저장된 구조체를 연결함으로써 각 노드의 속성값들을 저장하도록 하였다. 표 2는 SVG DOM 처리기에 사용되는 주요 API를 나열한 것이다.

1) SVG 객체 관리기

SVG DOM 관리기에서 추출된 각각의 DOM 노드 정보들은 렌더링 엔진과의 상호작용을 통해 그래픽 객체 생성을 위한 논리 정보로 변환하는 환경을 제공한다. 이 모듈은 DLL과 렌더링 엔진 사이의 상호작용에 많은 호환성을 제공하기 위해 별도의 헤더파일로 구현한 후, 사용하는 위치에 따라 DLL Import 또는, DLL Export의 두 가지 기능을 동시에 사용할 수 있도록 조건부 정의를 사용하여 구현하였다.

표 2. SVG DOM 처리기에 사용되는 주요 API 함수
Table 2. SVG DOM Manager Used to Main API Function

함수 명 (API 함수)	기 능
WalkTree(IXMLDOMNode *pNode)	재귀적 호출을 통한 SVG 구조 트리 생성 모듈
GetNodeData_EachType(IXMLDOMNode* pNode)	각 노드별 데이터 추출 모듈
GetNode_Element(IXMLDOMNode *pNode)	기준 노드의 엘리먼트 정보 추출 모듈
GetAttribute_byNode(NODEDATA NodeData, IXMLDOMNode *pNode)	기준 노드의 속성 정보 추출 모듈
CreateDOMDocument(CString szFullPath)	파일 경로를 이용한 DOM 객체 생성 모듈
SVGOpenDocument(LPCTSTR lpszPathName)	파일 경로를 이용해 문서를 오픈
GetIXMLDOMDocument(void)	현재 작업중인 SVG DOM 객체를 반환

2) SVG 코드 변환기

객체 관리기에 의해 저장된 SVG 논리 정보를 이용하여 SVG 구문을 생성하기 위한 SVG 코드 변환기는 렌더링 엔진 내에 구현된 디스플레이 처리기의 사용자 이벤트에 따라 실행된다. 사용자의 마우스 이벤트가 발생하면, 렌더링 엔진의 도형 관리기에 사용자가 저작한 그래픽 객체를 생성하고 생성된 객체의 논리 정보를 토대로 SVG 구문을 작성하도록 구현하였다. SVG 구문을 작성하는데 사용되는 주요 API 함수는 표 3과 같다.

표 3. SVG 코드 변환기에 사용되는 주요 API 함수
Table 3. SVG CODE Translator Used to Main API Function

함수 명 (API)	기 능
SetSVGElementNode(IXMLDOMDocument *pXmlDoc, IXMLDOMElement* pBaseNode, CString szNodeName)	기준 노드를 중심으로 SVG 엘리먼트 설정 모듈
SetSVGAttributesForElement(IXMLDOMDocument* pXmlDoc, IXMLDOMNode* pBaseNode, NODEDATA pNodeData)	기준 노드를 중심으로 SVG 속성 설정 모듈
SetTempFileTo_XML(IXMLDOMDocument* pDoc)	임시 메모리에 현재 DOM 객체를 파일로 저장

4.1.2 렌더링 엔진 구현

렌더링 엔진은 사용자들에게 그래픽 저작에 필요한 인터페이스를 제공하도록 구현하였으며 특히, 구현된 각 모듈들은 모듈간의 종속관계를 최소화시켜 모듈의 독립성에 중점을 두어 구현하였다. 렌더링 엔진은 그래픽 객체를 효율적으로 관리하기 위한 그래픽 객체 관리기 모듈과 SVG 애니메이션을 실행하기 위한 애니메이션 처리기 모듈, 그래픽 객체를 사용자 뷰포트에 렌더링 시키기 위한 디스플레이 처리기 모듈로 일반화 시켰으며, 기타 그래픽 저작에 필요한 사용자 인터페이스 모듈들을 사용하여 구현하였다.

4.1.3 SVG 문서 저작 시스템 구현 결과

그림 6은 시스템의 화면 구성을 보여준다. 전체 화면은 일반적인 그래픽 툴의 형태로 이루어지며, 자식 윈도우의 중앙에 SVG 캔버스를 위치시킴으로써 저작되는 그래픽 객체들을 한눈에 볼 수 있도록 인터페이스를 사용자 중심적으로 하였다. 특정 객체에 애니메이션이 추가될 경우 화면의 하단에 위치한 애니메이션 컨트롤에 등록되어 사용자가 쉽게 SVG 그래픽 객체에 애니메이션을 추가하도록 구현하였다.

SVG 미리 보기 창과 원문 보기 창은 기존의 다른 시스템과는 달리 지저분한 윈도우들을 줄이기 위해 팝업 다이얼로그로 구현하여 사용자가 확인하고자 할 때만 보여질 수 있도록 하였다.

그림 6. SVG 저작 시스템의 화면 구성
Fig. 6. UI of SVG Editing System

4.2 고찰

본 논문에서 구현한 SVG 문서 표현 시스템은 문서의 검증 및 정보 추출을 위해 MSXML 4.0 파서를 사용하여 렌더링 객체를 생성하였으며, 저작된 그래픽 객체를 SVG 구문으로 변환하기 위한 코드 변환기를 구현하였다.

렌더링 객체 생성은 표준화된 DOM 인터페이스가 문서의 구조만을 처리함으로써 노드들의 논리 정보를 올바로 이해할 수 없는 문제점이 존재한다. 따라서, SVG 그래픽 객체의 논리 정보를 올바로 이해하고 처리할 수 있는 SVG DOM 처리기를 별도로 구현하였다. 이에 따른 장점은 첫째, 기존 XML 파서의 의존도를 최소화함으로써 SVG 문서 처리에 유연성을 제공하며, 둘째, 다른 시스템에 적용할 수 있도록 DLL 형태로 구성함에 따라 그래픽 처리에 따른 메모리 사용률을 최소화 시켰다는 점이다.

본 논문에서 제안한 전체 시스템의 장점은 첫째, W3C에서 제안하는 SVG 1.1의 렌더링 처리에 대한 구성을 기반으로 구현함으로써 표준화에 따른 처리 시스템으로써 변화에 능동적으로 대처 가능하다는 점이고, 둘째, 시스템을 모듈화하여 부분적인 수정 및 대체와 이식이 가능하다는 것과 WYSIWYG 방식의 인터페이스를 제공하여 사용자가 쉽게 SVG 문서를 저작할 수 있다는 점이다.

현재의 SVG 저작 시스템들은 SVG 1.0 명세를 기준으로 설계되었으며, SVG 문서 구조를 처리하는데 있어 기존의 XML 파서에 의존적이어서 렌더링 처리와 구조 변환에 있어서 SVG의 모든 면을 수용하기에는 한계가 있으며 이들 처리를 위해 다른 엔진을 설계함으로써 시스템 실행 속도와 메모리 사용에 막대한 영향을 미치는 점을 감안할 때 본 시스템의 구현은 의미가 있다.

향후 네임스페이스(Namespace)와 스크립트(ECMA Script) 정의 등의 부가 기술 처리에 대해 보완해야 한다. 또한 모바일 환경에서 SVG 문서 처리에 대하여도 지속적인 관심과 연구가 요구된다.

V. 결 론

기존의 표현 중심적인 그래픽 처리는 비구조적인 처리 방법에 따른 많은 문제점을 내포하고 있었다. 이러한 문제점이 대두되기 시작하면서 W3C에서는 구조적으로 벡터 그래픽 처리를 하기 위한 SVG를 제정하였고, 이에 따른 여러 관련 어플리케이션들이 요구되었다. 기존 시스템들이 처리하고 있는 범위는 SVG의 일부분에 국한되어 있고, SVG가 추구하는 확장성과 다양한 이미지 포맷의 구조화가 부족한 실정이다.

이에 본 논문에서는 SVG 1.1 문서의 입력에 따

라 문서의 논리 구조를 처리하기 위한 SVG DOM 처리기와 생성된 SVG 그래픽 객체를 올바로 렌더링 시키고, 저작된 그래픽 객체들을 SVG 구문으로 변환해 주는 SVG 문서 저작 시스템을 설계 및 구현하였다.

본 시스템의 장점으로는 W3C에서 제안하는 SVG 1.1의 권고안을 따르고 있어 표준화에 따른 처리 시스템이라는 점이고, 결과적으로 표준화의 변화에 빠른 대처가 가능하다. 또한 시스템의 대부분을 모듈화하여 부분적인 수정 및 대체와 이식이 가능하다는 장점이 있다.

본 논문 결과는 SVG 관련 시스템 개발에 많은 영향을 줄 수 있으며, 텍스트기반의 SVG 포맷으로 저장된 이미지를 검색하여 사용자에게 제공할수 있는 기반이 되고, 인터넷 상에서 광고나 전자상거래, 지리정보, 교육 등 그래픽이 많이 사용되는 여러 분야에서 SVG 문서를 표현하기 위해 본 시스템이 유용하게 사용되리라 사료된다.

향후 스크립트 및 네임스페이스 처리에 대한 연구와 다른 어플리케이션과의 결합을 위해 추가적인 렌더링 연구가 진행되어야 할 것이다. 또한 모바일 환경에서 SVG 문서 처리를 위한 연구를 추가해야 할 것이다.

참고문헌

[1] 정희경, "WWW 문서 작성을 위한 차세대 언어 XML 가이드", 그린

[2] W3C, eXtensible Markup Language (XML) Version 1.0 (Second Edition), http://www.w3.org/TR/REC-xml, Oct. 6, 2000

[3] W3C, Scalable Vector Graphics(SVG) Version 1.1, http://www.w3.org/TR/SVG11, Jan. 14, 2003

[4] W3C, Document Object Model, http://www.w3.org/DOM

[5] W3C, Cascading Style Sheets, level 1, http://www.w3.org/TR/REC-CSS1.html, Jan. 11, 1999

[6] W3C, Cascading Style Sheets, level 2, http://www.w3.org/TR/REC-CSS2/, May 12, 1998

[7] W3C, Mobile SVG Profiles: SVG Tiny and SVG Basic, http://www.w3.org/TR/SVGMobile/, Jan. 14, 2003

[8] W3C, Mathematical Markup Language(MathML) Version 2.0(Second Edition), http://www.w3.org/TR/MathML2/, Oct. 21, 2003

[9] 나방현, 심규찬, 이종연 공저, "XML 그래픽 입문", 21세기사

[10] W3C, Synchronized Multimedia Integration Language(SMIL) Version 2.0, http://www.w3.org/TR/smil20, Aug. 07, 2001

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

[12] Kurt Cagle, "SVG Programming : The Graphical Web", July 2002, Apress

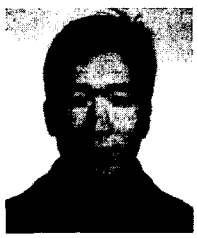
[13] XML.com Graphics Section, http://www.xml.com/graphics/

[14] W3C, SMIL Animation, http://www.w3.org/TR/smil-animation/, Sep. 04, 2001

[15] Microsoft Platform SDK, http://www.microsoftcom/msdownload/platformsdk/sdkupdate/

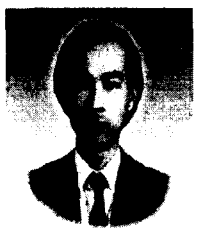
[16] MSDN Online (XML), http://msdn.microsoft.com/xml

저자소개



김택천(Tak-Chen Kim)

2001년 배재대학교 컴퓨터공학과 졸업(학사)
2003년 배재대학교 컴퓨터공학과 졸업(석사)
2003년~현재 배재대학교 컴퓨터 공학과(박사과정)
※관심분야 : XML, 데이터 마이닝, Spatial Database, Knowledge Representation



김진수(Jin-Soo Kim)

1975년 숭실대학교 전자계산학과(학사)
1985년 홍익대학교 전자계산학과(석사)
2001년 충북대학교 전자계산학과(박사)
1988년~현재 배재대학교 컴퓨터공학과 교수
※관심분야 : 데이터 마이닝, 지능형데이터베이스, Knowledge Representation, Web-based Decision Support

정희경(Hoe-Kyung Jung)

1985년 광운대학교 컴퓨터공학과(공학사)
1987년 광운대학교 컴퓨터공학과(공학석사)
1993년 광운대학교 컴퓨터공학과(공학박사)
1994년~현재 배재대학교 IT공학부 컴퓨터공학과 부교수
※관심분야 : 멀티미디어 문서정보처리, XML, SVG, ebXML, MPEG-21, Web Service