2001년 한국정보처리학회 추계 학술발표논문집 제8권 제2호

XML/SVG를 이용한 Scalable한 Web Character 구축

장미화° 조이기 김재갑 정강용 김원중
순천대학교 컴퓨터과학과
{roseston, dbcho}@sunchon.ac.kr, jaegab@scjc.ac.kr,
jung740@hanmail.net, kwj@sunchon.ac.kr

The Development of scalable Web Character Using XML/SVG

Mi-Hwa Jang° Lee-Gi Cho Jae-Gab Kim Gang-Yong Jung Won-Jung Kim
Dept. of Computer Science, Sunchon National Univ.

요 약

PC, Notebook, PDA, Cellular telephone, Smart phone 등 많은 형식의 인터넷 단말기가 사용되고 있지만, 인터넷 상에 존재하는 많은 그래픽 요소들은 이러한 단말기의 형식에 많은 제약을 받고 있다. 또한 사용자가 원하는 형태로 직접 변형하는 것이 어렵고, 그래픽 부분 요소의 재사용이 불가능하다. 본 논문에서는 차세대 마크업 언어로 각광받고 있는 XML(Extensible Markup Language)의 그래픽 표준인 SVG(Scalable Vector Graphic)를 이용하여 어떤 단말기에서도 같은 형식으로 디스플레이 되고 사용자가 원하는 형태로 이미지의 수정 및 부분 요소의 재사용성을 높인 웹 캐릭터 빌더(Builder)를 설계 및 구현하였다. 연구 결과를 이용하면 에이전트 상의 웹 캐릭터 혹은 웹 상에서 일어날 수 있는 어떠한 컨텐츠에도 사용할 수 있는 기술을 구현할 수 있을 것이다.

1. 서론

인터넷상의 도우미라고 할 수 있는 에이전트는 관리자의 개입이 없어도 정해진 스케줄에 따라 인터넷상에서 정보를 수집하거나 몇몇 다른 서비스를 수행하는 프로그램을 말한다.

현재 서비스되고 있는 전형적인 에이전트 프로그램은 미리 제공된 메개 변수들을 이용하여 전체 또는 일부의 인터넷을 검색하고 그 중 관심이 있는 정보를 모아서 매일 또는 일정시간대 별로 제공하는 것 등이 포함된다. 이와 같은 에이전트는 개인화 서비스 시대에 적합한 서비스이며, 에이전트 서비스를 사용자에게 제공할 때, 가장 많은 인터페이스 형식으로 제공되고 있는 것이 웹 캐릭터이다.

현재 만들어지고 있는 여러 가지 웹 캐릭터들은 래스터(Rester) 방식을 채택하고 있는데, 이러한 래스터 방식은 픽셀에 대한 압축기법이기 때문에 한 번 이미지의 크기나 형태를 결정하면 변형이 어렵다는 단점이 있다. 특히 웹 상에서 특정한 변형을 일으킨다거나 사용자가 요구하는 이미지로의 변환이 불가능하기 때문에 웹 캐릭터 빌더를 구성하기 위해 얼굴형에서부터 눈썹, 코 등과 같이 얼굴을 구성하고 있는 모든 부분 요소를 많은 경우의 수를 고려하여 Map화하여 가지고 있어야 한다[5]. 이는 개인화에 맞는 서비스를 제공하기 위한 웹 캐릭터 구성에 크나큰 제약 사항이 아닐 수 없다.

본 논문에서는 래스터형식의 이미지에서 보이는 단점을 해결한 벡터 형식의 이미지 중 차세대 마크업 언어로 부각되고 있는 XML의 그래픽 표준인 SVG로 캐릭터를 구현하였다. 또한 각 사용자의 요구에 맞는 변형이 이루어질 수 있도록 쉬운 인터페이스를 제공하였고, 모든 데이터를 데이터베이스에 저장함으로써 언제든지 사용자가 쉽게 자신의 캐릭터를 사용할 수 있도록 하였다.

본 논문의 구성은 다음과 같다. 2장에서 XML과 SVG에 대해 소개하고, 3장에서는 Web Character Builder의 설계와 구현에 대해 설명하였다. 마지막으로 4장에서 본 논문의 결론과 향후 연구 과제에 대해 기술하였다.

2. XML과 SVG

XML(Extensible Markup Language)은 기존의 SGML / HTML이 가지는 표현의 복잡성 또는 제약성을 제거한 데이터의 저장과 교환을 위한 개방성과 상호 운용성을 지원하는 텍스트 형식의 언어이다. SVG(Scalable Vector Graphic)는 2차원 벡터 그래픽 표현을 위한 XML 응용으로서, XML의 유용성이 웹을 통한 그래픽의 표현에도 적용될 수 있도록 고안되었다. XML 그래픽 표준으로 자리잡은 SVG는 벡터 그래픽을 사용함으로써 인터넷 정보단말기의 해상도에 관계없는 통일된 그래픽 관점과 그래픽 요소를 통한 상호작용을 지원함으로써 탁월한 사용자 인터페이스를 제공할 수 있게 한다. 또한, SVG는 동적인 그래픽 생성 기능에 의해 XML의 컨텐츠와 스타일을 분리하여 컨텐츠의 재사용성을 높이고자 하는 목적을 충족시킬 수 있다[1,5].

XML은 사용자가 필요에 의해 의미론적인 태그를 만들어 사용할 수 있는 메타 마크업 언어이기 때문에 HTML보다 홈페이지 구축 기능, 검색 기능 등이 향상되었고 클라이언트 시스템의 복잡한 데이터 처리를 쉽게 할 수 있다. 또한 인터넷 사용자가 웹에 추가할 내용을 작성, 관리하기가 매우 쉽다. 본 논문에서 웹 캐릭터를

구성하고, 웹 캐릭터 부분요소를 정의하는데 적용한 SVG는 XML 기반의 그래픽 표준으로 구조화된 데이터베이스 상에서 관리할 수 있는 장점이 있다[3,5].

본 논문에서 제안하고 있는 웹 캐릭터를 구성하는 SVG를 다른 그래픽 표준과 비교하면 [표 1]과 같다.

[표 1] SVG와 다른 그래픽 표준 비교 분석표

	SVG	Flash	AI	JPEG	GIF
이미지형식	Vector	Vector	Vector	Raster	Raster
애니메이션 유 무	O	O			O
스크립트 유 무	O	O			
크기변화 유 무	O	O	O		
HTML 표 준	O	O		O	O
DB 사용	O				
XML Based	O				

레스터 형식의 그래픽 표준은 한 번 이미지의 크기나 형태를 결정하면 변형이 불가능하기 때문에 웹 캐릭터를 생성하기 위해서는 많은 부분 요소의 집합을 가지고 있어야 한다. 반면 [표 1]에서 알 수 있듯이 벡터(Vector) 형식의 그래픽 표준은 크기나 모양 자체를 수정하기가 매우 용이하다.

특히 본 논문에서 제안하는 웹 캐릭터를 구성하는 SVG형식의 부분 요소의 특징을 살펴보면 다음과 같다.

1) Text
SVG는 XML 기반의 그래픽 표준으로 모든 그래픽 형식을 Text화하여 구성하고 있다. 그러므로 웹 상에서 구현하고 있는 캐릭터를 Text의 간단한 수정만으로도 원하는 다른 모양의 캐릭터로 변형시킬 수 있다. 다른 Vector 형식의 그래픽 표준은 웹 상에 구현하기 이전에 모든 변형에 대한 작업이 이루어져야 하기 때문에 직접적인 변형은 어렵다. 이 점이 본 논문에서 제시하는 SVG를 사용함으로써 누릴 수 있는 가장 효과적인 면이라고 할 수 있다.

2) XML Base
SVG는 구성 형식이 XML을 지향하고 있다. 특히 구조화 된 데이터베이스를 구성할 수 있어서 체계화된 부분요소 데이터베이스와 웹 캐릭터 데이터베이스를 구축할 수 있기 때문에 웹 캐릭터 Builder를 만들 수 있다.

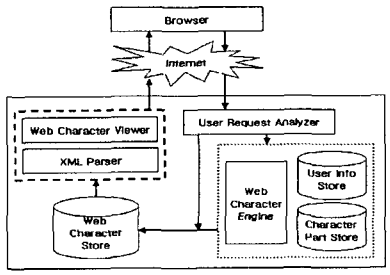
3) User Interface
SVG Source는 사용자가 요구하는 포인터 값을 이해하기 어려운 수치 값으로 표현한다. 이러한 값을 웹 캐릭터 엔진을 통해 사용자가 쉽게 접근할 수 있는 값으로 변환하고 쉬운 인터페이스 형식으로 제공하기 때문에 사용자가 SVG source 자체를 이해할 필요가 없다.

3. 설계 및 구현

웹 캐릭터 시스템을 구성하는 데 있어 데이터베이스는 XML 포맷을 쉽게 저장할 수 있는 MS SQL Server 2000을 선택하였다. 그에 따라 MS SQL Server 2000에 가장 접근이 용이한 ASP(Active Server Page 3.0)와 Windows 2000 Server를 이용하여 구현하였다. 또한 XML 데이터를 처리하는데 있어서 MS의 MSXML 3.0 Parser를 사용하였고, 사용자가 수정하는 컨트롤러는 ActiveX를 사용하였다. 브라우저는 IE 5.0 이상의 버전에 SVG Viewer Plug-In이 설치되어 있어야 한다.

3.1 시스템 구조

본 연구에서 구현한 Web Character Builder시스템은 [그림 1]과 같이 User Request Analyzer, Character Part Store, User Info Store, Web Character Store, Web Character Viewer로 구성되어 있다.



[그림 1] 시스템 구조도

사용자는 브라우저를 통해 새로운 웹 캐릭터를 만들 것인지 아니면 미리 만들어진 웹 캐릭터를 수정할 것인지를 선택한다. 선택한 데이터를 User Request Analyzer는 웹 캐릭터를 구성하는 부분 요소의 포인트 값을 Web Character Engine에서 추출할 것인지 Web Character Store에서 추출할 것인지를 먼저 결정한다.

먼저 새롭게 캐릭터를 생성하는 웹 캐릭터 엔진에서는 User Info Store에서 사용자 정보를 가져오고 Character Part Store에서 각 부분 요소에 대한 해당 기본 데이터를 추출하여 온다. 추출한 데이터를 사용자가 수정하게 되고 그 수정한 데이터를 User Request Analyzer는 웹 캐릭터 엔진에 추출한 부분 요소 기본 데이터를 보내고, 생성된 데이터를 SVG Source 형식으로 변환 처리한 뒤 Web Character Store에 각 사용자에 맞는 캐릭터 데이터를 저장한다. 두 번째로 사용자가 캐릭터를 수정하는 경우는 Web Character Store에서 미리 저장된 데이터를 추출하여 수정하고 이 데이터는 첫 번째 방법에서 제시하는 처리를 통해 다시 Web Character Store에 저장된다.

구현 시스템의 중요 모듈의 기능들은 다음과 같다.

1) User Request Analyzer
사용자가 선택한 데이터가 캐릭터 생성에 관한 것인지 아니면 수정에 관한 것인지 판단하여 Character Part Store에서 추출할 것인지 Web character Store의 생성된 정보를 추출할 것인지 결정하고, 사용자의 각 부분 요소에 대한 변형된 수치 값을 분석하여 Web Character Engine에서 처리하도록 수정 데이터를 처리한다.

2) User Info Store
웹 캐릭터를 만들고자 하는 사용자의 기본 데이터를 제공한다.

3) Character Part Store
각 부분 요소에 대한 일련의 데이터를 제공한다. 부분 요소의 기본 캐릭터들에 대한 SVG Source가 분석되어져 데이터베이스화되어 있다. 적용되는 포인트 값은 XML형식의 데이터 값이므로 모든 데이터 타입은 문자열을 취한다. 웹 상에서 사용자에게 제시되는 기본 캐릭터는 바로 Character Part Store에 저장된 데이터이고 이 곳은 수정, 변환 없이 데이터만 제공하는 곳이다. 물

론 Parser를 통해 가변 포인트 값을 수정한 뒤 마지막 Web Character Store에 저장될 때 기본적인 형태는 이곳에서 제공된다.

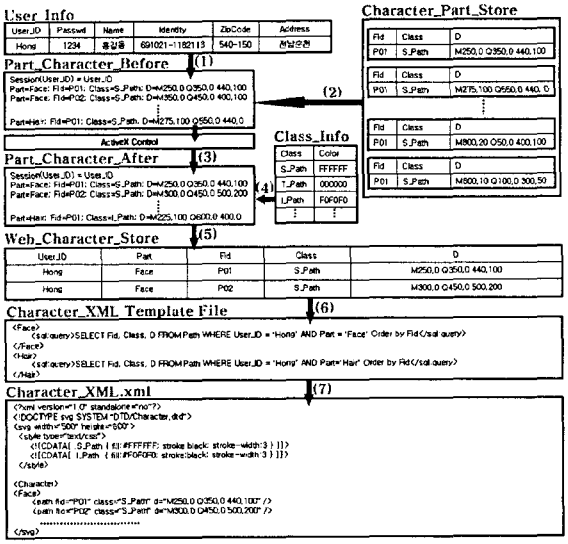
4) Web Character Engine
User Info Store에서 가져온 사용자 정보와 Character Part Store에서 추출된 캐릭터 구성 요소의 기본형 데이터에 각 변형된 포인터 값을 변경 및 재 조합하여 새로운 포인터 값으로 생성한 뒤, 다시 SVG 형식의 데이터를 생성하고 Web Character Store에 저장한다.

5) Web Character Store
사용자의 요청에 의해 변형된 포인터 값이 저장된 저장소이다. 모든 부분이 사용자와 연결되어 있고 디스플레이 되는 구성 요소의 모든 값은 SVG 형식으로 저장이 되어 있기 때문에 웹 상에서 구현 될 때에는 사용자의 정보를 XML Parser를 통해서 조합하기만 하면 된다.

6) Web Character Viewer
Web Character Store에서 생성한 데이터를 XML/SVG 형식의 HTML문서로 변환하는 역할을 한다. 이렇게 변환되어진 HTML문서는 사용자가 사용하는 브라우저에서 XML/SVG 형식의 Web Character를 디스플레이 한다.

3.2 구현

[그림 2]는 구현 시스템에서의 데이터의 변환 흐름을 도식화한 것이다.



[그림 2] 데이터 처리도

[그림 2]의 Part_Character_ Before파일은 해당 Part명을 생성하고 User_Info의 userid, Character_Part_Store의 fid, class, d를 추출한 뒤 ActiveX Control로 구현한 UI(User Interface)를 이용하여 데이터를 수정한다. 사용자에 의해 선택된 색상은 Class_Info Store에 저장되어 있는 클래스 데이터에 적용되어 새로운 스타일 시트를 생성한다. 변환된 데이터는 Part_Character_After를 통해 Web_Character_Store에 저장된다. Character XML 템플릿 파일은 Web_Character_ Store의 데이터를 각 부분 요소에 맞는 형식으로 가져오기 위해 질의문으로 구성되어진 MS SQL 2000의 XML 지원 형식을 이용하여 생성한 파일이다. 이 파일을 이용하여 Character_XML.xml파일에서 각 부분 요소별로 XML/SVG Source화된 웹 캐릭터를 구현한다.

데이터 변환 흐름에서 각 부분의 기능은 다음과 같다.

(1), (2) 부분 요소 데이터 변환
ASP 파일에서 요구하는 사용자 정보와 부분 요소 정보를 데이터베이스에서 추출하여, 사용자 정보는 Session 정보로 저장하고 부분 요소 정보는 특정 ASP 파일에서 요구하는 형태로 보내어 진다.

(3), (5) ASP 파일에서의 데이터 변환
ASP 파일에서는 이렇게 입력받은 데이터를 사용자의 요구에 의해 특정한 데이터로 바꾸는데 이 때 컨트롤러는 ActiveX로 구현하였다. Session에 저장되어 있는 사용자 정보와 특정 파일에서 생성한 부분 요소명, 사용자에 의해 수정 된 데이터를 Web_Character_Store에 저장한다.

(4) 색상 데이터 선택
사용자가 선택한 color로 Class_Info Store에서 Class명을 추출한다.

(6) Character_XML 템플릿 파일
MS SQL에서 제공하는 템플릿 파일을 이용해 각 부분 요소에 맞는 질의문을 파일화하여 저장한다.

(7) Character_XML.xml
이렇게 저장한 파일이 사용자의 요청에 의해 실행되면 데이터베이스에서 원하는 정보를 추출하여 각 부분 요소별로 XML/SVG Source화하여 완성된 웹 캐릭터를 생산한다.

[표 2]는 웹 캐릭터를 구성하는 요소들을 규정해 놓은 DTD(Document Type Definition)이다.

[표 2] Character 구성 DTD

```
<!ELEMENT CHARACTER (FACE, NOSE, EYE, MOUSE,
                     EAR, HAIR, EYEBROWS)>
    <!ELEMENT FACE      (path)>
    <!ELEMENT NOSE      (path)>
    <!ELEMENT EYE       (path)>
    <!ELEMENT MOUSE     (path)>
    <!ELEMENT EAR       (path)>
    <!ELEMENT HAIR      (path)>
    <!ELEMENT EYEBROWS  (path)>
    <!ELEMENT path      EMPTY >
    <!ATTLIST path
              fid   ID #REQUIRED
              class CDATA #IMPLIED
              d     CDATA #REQUIRED >
```

[표 2]의 DTD는 얼굴, 코, 눈, 입, 귀, 머리, 눈썹의 부분 요소를 선언하였고 각 부분 요소는 형태를 표현할 path를 포함한다. path에는 각 포인트의 번호가 되는 fid와 곡선의 좌표 값인 d를 필수 속성으로 정의하였고 스타일을 지정해 주는 class는 선택 속성으로 정의하였다.

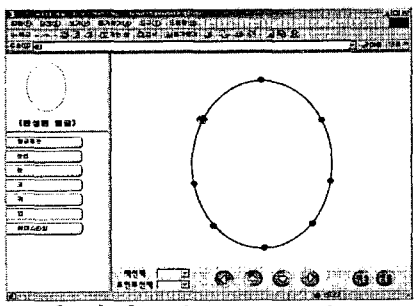
[표 3]은 FACE를 생성하기 위한 XML/SVG 코드이다. FACE를 만들기 위해 SVG에서 곡선을 그리는 형식 중 Path를 이용한 Quadratic Bezier Curve명령어를 사용하였다. Quadratic Bezier 곡선은 시작점과 끝 점 그리고 한 개의 컨트롤 포인트에 의해 정의된다.

[표 3] Quadratic Bezier Curve를 이용한 FACE 생성

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg SYSTEM "DTD/Character.dtd'>
<svg width="500" height="600" viewBox="-50 -50 1200 600">
  <!-- svg contents here -->
  <style type="text/css">
   <![CDATA[ .S_Path { fill:#FFFFFF; stroke:black;
                       stroke-width:3 } ]]>
  </style>
<CHARACTER>
 <FACE>
  <path fid="P01" class="S_Path" d="M250,  0 Q370,  0 440,100"/>
  <path fid="P02" class="S_Path" d="M440,100 Q500,190 500,300"/>
  <path fid="P03" class="S_Path" d="M500,300 Q500,410 440,500"/>
  <path fid="P04" class="S_Path" d="M440,500 Q370,600 250,600"/>
  <path fid="P05" class="S_Path" d="M250,600 Q130,600  60,500"/>
  <path fid="P06" class="S_Path" d="M 60,500 Q  0,410   0,300"/>
  <path fid="P07" class="S_Path" d="M  0,300 Q  0,190  60,100"/>
  <path fid="P08" class="S_Path" d="M 60, 10 Q130,  0 250,  0"/>
 </FACE>
 <NOSE>
    <!-- ..... -->
</svg>
```

[그림 2]의 Character_Part_Store에는 Path의 번호를 의미하는 fid와 Path의 좌표 값을 의미하는 d가 존재한다. 두 값은 템플릿을 사용하여 화면에 가져오고 사용자의 색상 선택으로 class 값을 추출해 온다. [표 3]에서 이 세 값으로 하나의 Path가 생성이 되고 변환된 Path 값을 얼굴, 코 등으로 그룹화 한다. 이렇게 그룹화 된 각각의 부분 요소가 조합되어 웹 캐릭터가 생성된다.

구현된 Web Character Builder는 [그림 3]과 같이 생성하고자 하는 웹 캐릭터를 얼굴, 코, 눈, 입, 귀, 머리, 눈썹의 7개의 부분 요소로 나누고 각 부분 요소별로 각각의 기본 캐릭터를 제공하도록 구성되어 있다. 이렇게 구성된 Builder를 사용하여 사용자는 자신만의 독특한 캐릭터를 생성할 수 있는 것이다.



[그림 3] FACE User Interface

[그림 3]의 화면은 사용자가 부분 요소 중 얼굴과 관련된 데이터를 수정하는 화면을 나타내고 있다.

사용자에게 제공되는 얼굴 모형에는 8개의 가변 포인트가 있다. 이 포인트는 사용자가 얼굴 모형을 변형하는데 있어서 수정할 수 있는 가장 최적화된 위치의 포인트이다. 얼굴의 가변 포인트는 상, 하, 좌, 우의 네 개와 그 사이의 포인트 값 네 개로 이루어져 총 8개의 가변 포인트가 있다. 사용자는 이렇게 제공되는 8개의 가변 포인트를 방향키를 이용하여 원하는 형태의 얼굴윤곽과 크기를 결정할 수 있다. 그리고 색상 선택을 통하여 얼굴의 색상을 결정하게 된다.

위와 같은 방식으로 웹 캐릭터 빌더에 7개의 부분 요소의 기본 캐릭터는 각 부분 요소별로 고정적인 가변 포인트를 가지고 있다. 사용자는 각 기본 캐릭터를 선택하여 수정할 수 있는 포인트의 좌표 값을 수정함으로써 크기 및 모양을 결정할 수 있다. 캐릭터 자체가 SVG형식으로 되어 있으므로 수정된 좌표 값이 바로 적용될 수 있는 것이다. 레스터 형식의 캐릭터 부분 요소에는 많은 데이터가 필요하지만, 본 논문에서는 Vector 형식이면서 웹 상에서 데이터가 바로 수정이 가능하므로 캐릭터의 부분 요소는 기본 캐릭터 하나가 되는 것이다. 기본 캐릭터를 구성하는 포인트 사이의 선은 SVG의 2차원 베지어 커브(Quadratic Bezier Curve)명령을 사용하여 선의 곡선 정도를 결정하였다.

이렇게 사용자가 7개의 부분 요소의 값을 수정, 변경하여 캐릭터를 완료시키면 Web Character Builder는 마지막으로 수정된 7개 부분 요소의 SVG Source와 사용자의 정보를 Web Character Store에 저장하여 언제나 그 사용자의 독자적인 캐릭터를 이용할 수 있게 한다.

4. 결론

현재 웹 상에 존재하는 많은 웹 캐릭터 빌더는 래스터(Rester) 형식이기 때문에 많은 이미지 맵을 가지고 있어야 하고 원하는 얼굴형을 정확히 구현하는 것은 거의 불가능하다. 하지만 본 논문에서 구현한 캐릭터 빌더는 여러 맵을 보면서 선택해야 하는 번거로운 인터페이스를 하나의 부분 요소를 직접 수정하게 함으로서 간략화 하였고 원하는 형태의 얼굴형을 만들 수 있다. 향후 연구 과제로 본 논문에서 구현한 웹 캐릭터는 기존 래스터 형식의 캐릭터가 제공하는 색상에 비해 색상 선택이 원활하지 않아서 색상에 대한 보완이 있어야 한다. 또한 사용자가 원하는 부분을 직접 선택하여 수정하지 못하고 정해져 있는 포인트 값을 방향키를 이용하여 간접적으로 수정해야 하는 점이 개선되어야 할 것이다.

5. 참고 문헌

[1] F. Yoshikawa, K. Wada, K. Toraichi, K. Mori, H. Kiduka, K. Katagishi and A. Okamoto, "A Note on a High Resolutional Communication System Using Fluency Theory", 1997 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp.916-919, August 1997

[2] John D. Hobby, "Space-Efficient Outlines from Image Data via Vertex Minimization and Grid Constrains", Graphical Models and Image Processing, Vol.59, No.2, pp.73-88, 1997

[3] Kazuo Toraichi, "On a Method of Automatically Compressing Fonts with High Resolution", Pattern Recognition, Vol.26, No.2, pp.227-235, 1993

[4] Lejun Shao, Hao Zhou, "curve Fitting with Bezier Cubics", Graphical Models and Image Processing, Vol.58, No.3, pp.223-232, 1996

[5] W3C Scalable Vector Graphics (SVG), http://www.w3c.org/Graphics/SVG/