이미지 손실없는 확대/축소가 가능한 아바타 생성 시스템

김원중* · 장미화*

The Avata Construction System for Image Lossless Scaling

Won-jung Kim* · Mi-hwa Jang

이 논문은 정보통신부에서 지원한 2001년도 대학 기초연구지원 사업의 연구비를 지원받았음

요 약

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 손상없이 수정이 가능하여 각 부분 요소의 재사용성을 크게 높인 아바타(웹 캐릭터) 생성 시스템을 설계 및 구현하였다. SVG는 텍스트로 기술되기 때문에 그래픽에 대한 검색이 편리하고, 어플리케이션들이 SVG문서를 쉽게 사용할 수 있으며, 선, 폴리곤, 텍스트, 이미지 등의 모든 그래픽 요소에 쉽게 접근할 수 있으므로 데이터베이스와 연동하여 웹 그래픽 문서를 동적으로 생성할 수 있다. 또한 연구 결과를 이용하여 웹 상에서 발생할 수 있는 어떠한 컨텐츠에도 사용할 수 있는 기술을 구현할 수 있을 것이다.

ABSTRACT

In this paper, we designed and implemented Avata construction system using XML(eXtensible Markup Language) and SVG(Scalable Vector Graphic). The Web character created with Avata(or Web character) construction system are displayed in same form without damage of image, regardless terminal type and user can modify and change image easily in form that want. Compare with existing Web character system, the Reusability of web character part element is increased greatly with Avata construction system of this paper.

Because SVG is described by text, graphic retrieval is convenient, and applications can use easily SVG document. Also, SVG can create web graphic document dynamically with database because can access easily in all graphic primitives of line, Polygon, text, image etc. As well as web character using study finding, we may develop usable technology to some contents on World Wide Web.

키워드

Avata, XML(eXtensible Markup Language), SVG(Scalable Vector Graphic), Web character, Vector, Bitmap

*순천대학교 컴퓨터과학과

접수일자 : 2002. 4. 13

Ⅰ. 서 론

정보의 바다라고 불리는 인터넷은 무한한 정보를 기반으로 이제는 우리들의 일상생활에서 일부분이 되었다. 웹 캐릭터는 아바타(Abata)라는 이름을 가지고 다양한 분야에 활용되고 있는데, 아바타는 분신(分身)·화신(化身)을 뜻하는 말로 사이버공간에서 사용자의 역할을 대신하는 애니메이션 캐릭터를 말한다. 아바타는 그래픽 위주의 가상사회에서 자신을 대표하는 가상육체로 현실세계와 가상공간을 이어주며, 익명과 실명의 중간 정도에 존재한다. 과거 네티즌들은 사이버공간의 익명성에 매료되었지만 이제는 자신을 표현하려는 욕구가 발생하게 되어 이 두 가지를 모두 충족시켜주는 아바타가 생기게 되었다.

현재 이용되고 있는 대부분의 게임이나 채팅서비스에는 주로 몇 가지의 캐릭터를 조합하거나 이미 완성된 아바타를 제공하고 있다. 하지만 그래픽기술이 향상되면서 서비스 제공자가 이미 만들어놓은 기성품(Ready-made)을 이용하는 것이 아니라, 문자ID처럼 사용자가 자신만의 개성있는 아바타를 직접 만들 수 있는 나만의 아바타가 등장하게 되었다.

이처럼 웹 상에는 많은 웹 캐릭터를 이용한 다양한 서비스가 이루어지고 있는데, 이런 웹 캐릭터를 만드는 웹 캐릭터 생성시스템은 크게 두 가지 특징을 가지고 있다.

먼저 조합형 웹 캐릭터 빌더(Builder)로 웹 상의 표준 그래픽 포맷인 JPEG, GIF 등의 비트맵 이미지를 이용하고 있다. 비트맵 방식은 픽셀에 대한 압축기법으로 한 번 이미지의 크기나 형태를 결정하면 변형이 어렵다는 단점이 있다. 특히 웹 상에서 특정한 변형을 일으킨다거나 사용자가 요구하는 이미지로의 변환이 불가능하기 때문에 사람의 얼굴을 생성하기 위한 웹 캐릭터 빌더를 구성하기 위해서는 얼굴형에서부터 눈썹, 코 등과 같이 얼굴을 구성하고 있는 모든 부분 요소를 많은 경우의 수를 고려하여 맵(Map)화 하여 가지고 있어야 한다[2,5,6]. 이렇게 처리되어 있는 부분요소들을 조합하는 빌더는 미리 만들어진 이미지를 사용자에게 선택하게 함으로써 사용자는 자신만의 특성과 개성을 가진 아바타를 생성하기가 매우 어렵다.

본 논문에서는 비트맵(Bitmap) 형식의 이미지를 사용할 때의 단점을 극복하기 위한 방안으로 벡터(Vector) 형식의 이미지 중 차세대 마크업 언어로 부각되고 있는 XML의 그래픽 표준인 SVG로 캐릭터 생성 시스템을 구현하였다. 또한 각 사용자의 요구에 맞는 변형이 쉽게 이루어 질 수 있도록 사용자에게 친숙한 인터페이스를 제공하였으며, 모든 데이터를 데이터베이스에 저장함으로써 언제든지 사용자가 쉽게 자신의 캐릭터를 사용하거나 수정할 수 있도록 하였다. 기존 시스템의 경우에는 클라이언트(웹브라우저)에서 확대, 축소 등의 작업 요청이 발생하면 웹 서버에서 다시 프로세싱을 하여야 하지만, 본 연구에서는 SVG문서를 이용하여 웹 브라우저에서 곧 바로 확대, 축소 등의 작업을 수행하므로 데이터의 전송량을 크게 줄일 수 있고, 처리 속도도 매우 빠른 장점이 있다.

본 논문의 구성은 다음과 같다. 2장은 관련연구로 XML, SVG에 대해 소개하고, 3장에서는 웹 캐릭터 생성 시스템의 구조와 각 모듈의 기능들을 설명하였다. 4장에서는 사용자 인터페이스를 중심으로 시스템의 구현에 대해 기술하고, 5장에서 본 논문의 결론과 향후 연구 과제를 제시하였다.

Ⅱ. 관련 연구

2.1 XML/SVG

SVG(Scalable Vector Graphic)는 XML(eXtensible Markup Language)의 그래픽 표준으로 텍스트로 기술되기 때문에 그래픽에 대한 검색이 편리하고, 어플리케이션에 SVG 문서를 쉽게 사용할 수 있다. 또한 XML과 SVG의 인터페이스를 통하여 선, 폴리곤(Polygon), 텍스트, 이미지 등의 모든 그래픽 요소에 쉽게 접근할 수 있으므로 데이터베이스와 연동하여 웹 그래픽 문서를 동적으로 생성할 수 있다.

2.1.1 XML

XML(eXtensible Markup Language)은 HTML과 같은 고정된 형식이 아닌 확장이 가능한 언어이다. HTML은 태그가 한정되어 있지만 XML은 문서의 내용에 관련된 태그를 사용자가 직접 정의할 수 있으며, 그 태그를 다른 사람들이 사용하도록 할 수도 있다. XML은 본질적으로 다른 언어를 기술하기 위한 언어, 즉 메타언어이다.

XML은 SGML의 간략화된 버전으로 SGML의 실용적인 기능만을 모은 부분집합이라 할 수 있다. XML은 SGML의 장점과 일반성을 최대한 수용하고, SGML의 특정 부분을 발췌/요약하여 전문가와 일반인 모두 쉽게 배워 사용할 수 있는 언어이다. XML이 SGML의 부분 집합이므로 SGML과 XML간의 변환이 용이하고 XML 문서를 SGML Application에서 그대로 사용할 수 있다. 일반적으로 XML은 Web Application에서 SGML의 복잡한 기능과 HTML의 단순성의 문제를 해결해 주기 위해 만들어진 일종의 SGML 라이트(Light)로 생각할 수 있다[2,6].

2.1.2 SVG

XML의 그래픽 표준인 SVG(Scalable Vector Graphic)는 XML의 개방성, 상호운용성 등의 장점을 벡터 그래픽에 모두 수용하였고, SMIL, GML, MathML 등 다른 XML 언어들과 결합하여 다양한 응용의 개발이 가능하므로 광고, 전자상거래, 프로세스 컨트롤, 지리정보, 교육 등 그래픽이 많이 사용되는 분야에 크게 적용될 수 있다. SVG는 XML문서이므로 XML문서의 구조를 그대로 따른다. SVG문서는 논리적 구조(Logical Structure)와 물리적 구조(Physical Structure)를 갖고 있다. 물리적으로 SVG 문서는 개체(Entity)라는 요소들로 이루어지며 개체는 다른 개체들을 참조해 그것들을 문서 안에 포함시킬 수 있다. 논리적으로 SVG문서는 선언(Declaration), 요소(Element), 주석(Comment), 문자참조(Character Reference) 그리고 처리 명령어(Processing Instruction)로 구성되며, 명시된 마크업에 의해 문서 안에 표시된다.

즉, 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].

Table. 1 Function comparison of Graphic Standard

	SVG	Flash	AI	JPEG	GIF
이미지형식	Vector	Vector	Vector	Bitmap	Bitmap
애니메이션 유무	O	O	×	×	O
스크립트 유무	O	O	×	×	×
크기변화 유무	O	O	O	×	×
HTML 표준	O	O	×	O	O
XML Based	O	×	×	×	×

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

Ⅲ. 아바타 생성 시스템

아바타 생성 시스템(Avata Construction System)의 데이터베이스로는 XML 포맷을 쉽게 저장할 수 있는 MS SQL Server 2000을 사용하였다. 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 웹 캐릭터 생성 시스템은 그림 1과 같이 User_Request_Analyzer, User_Info_ Store, Class_Info_Store, Character_Part_Store, Web_Character_Store, Web_Character_Engine, Web_Character_Viewer로 구성되어 있으며, 각 모듈의 구성 및 기능은 다음과 같다.

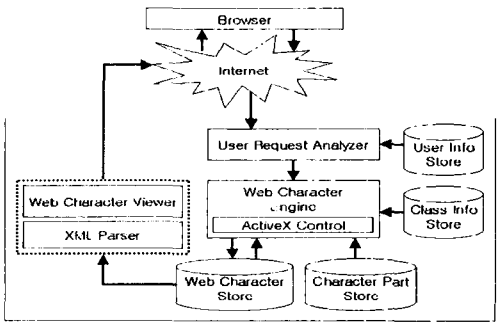

Fig. 1 The Structure of Avata Construction System

3.2 User_Request Analyzer

Login한 사용자의 User_ID는 사용자가 종료하기 전까지 Session에 저장되어 Web Character를 구현하고자 할 때, 그림 2와 같이 데이터베이스를 선택할 수

있도록 한다. 그림 2의 User_ Request_Analyzer는 사용자가 선택한 Character 데이터가 캐릭터 생성에 관한 것인지 아니면 수정에 관한 것인지 판단하여, Character_Part_ Store에서 기본 데이터를 가져올 것인지 Web_ Character_Store의 미리 정의한 데이터를 추출할 것인지 결정하고, 사용자의 각 부분 요소에 대한 변형된 수치 값을 분석하여 Web_Character_ Engine에서 처리하도록 데이터를 가공한다.

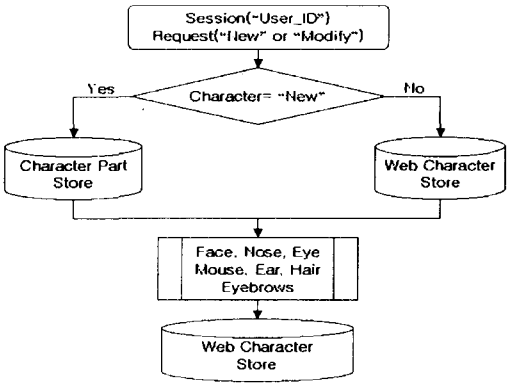

Fig. 2 Flow Chart of User_Request_Analyzer

3.3 User_Info_Store

웹 캐릭터를 만들고자 하는 사용자의 기본 데이터가 저장되는 곳이다. 사용자의 이름과 웹 상에서 사용할 ID, Password가 저장되어 있는 것은 다른 웹사이트와 동일하나, 사용자가 이미 작성한 웹 캐릭터를 가지고 있는지 여부를 나타내는 Character 변수 데이터를 가지고 있는 점이 다르다.

Table. 2 Structure of User_Info_Store Table

```
CREATE TABLE [dbo].[User_Info] (
 [User_ID]          [varchar] (12)  NOT NULL ,
 [Passwd]           [varchar] (8)   NOT NULL ,
 [Name]             [varchar] (12)  NOT NULL ,
 [Identity_No]      [varchar] (13)  NOT NULL ,
 [ZipCode]          [varchar] (6)   NOT NULL ,
 [Address_Top]      [varchar] (50)  NOT NULL ,
 [Address_Bottom]   [varchar] (50)  NULL ,
 [Phone]            [varchar] (11)  NULL ,
 [Character_Flag]   [varchar] (1)   NOT NULL
) ON [PRIMARY]
```

표 2는 User_Info_Store Table의 구조로서 데이터베이스의 최적화를 위해서 길이에 제한을 두었고, 주소의 뒷부분과 전화번호를 제외하고는 NOT NULL로 처리하여 반드시 데이터를 입력받도록 하였다.

3.4 Class_Info_Store

사용자에 의해 선택된 캐릭터의 색상이 저장되어 있다. 선택된 색상은 저장되어 있는 클래스 데이터에 적용되어 새로운 스타일 시트를 생성한다.

3.5 Character_Part_Store

캐릭터의 각 부분 요소(Basic Element)에 대한 일련의 데이터를 제공한다. 즉, 부분 요소의 기본 캐릭터들에 대한 SVG 데이터가 분석되어져 데이터베이스화 되어 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, 변환없이 데이터만 제공하는 곳이다. 파서(Parser)를 통해 가변 포인트 값을 수정한 뒤, 마지막으로 Web_Character_Store에 저장될 때 기본적인 형태는 이 곳에서 제공된다.

본 연구에서 구현한 시스템에서는 사람의 얼굴에 대한 총 7개의 기본 캐릭터별로 포인트 번호를 의미하는 ID값과 좌표 값인 D1, D2, D3, 색상 선택을 위한 Class 필드로 구성되어 있다.

표 3은 각 부분 요소의 데이터를 ID 순으로 저장하고 있다. 데이터를 제공만 하는 곳이기 때문에 한 번 구축된 데이터를 프로그램 상에서 수정하지는 못한다.

Table. 3 Structure of Character_Part_Store Table

```
CREATE TABLE [dbo].[Part_FACE] (
        [Id]     [varchar]  (3)   NOT NULL ,
        [D1]     [varchar]  (20)  NOT NULL ,
        [D2]     [varchar]  (20)  NOT NULL ,
        [D3]     [varchar]  (20)  NOT NULL ,
        [Class]  [varchar]  (20)  NOT NULL ,
) ON [PRIMARY]
```

3.6 Web_Character_Store

사용자의 요청에 의해 수정된 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 사용

자의 정보를 XML Parser를 통해서 조합하면 된다.

표 4는 사용자가 각 부분요소의 데이터를 수정 한 뒤 저장하는 Web_Character_Store Table의 구조로서 XML/SVG 구현상 사용자의 데이터를 Part와 ID 순으로 저장하고 있다.

Table. 4 Structure of Web_Character_Store Table

```
CREATE TABLE [dbo].[Path] (
    [User_ID]    [varchar]  (12)   NOT NULL ,
    [Part]       [varchar]  (10)   NOT NULL ,
    [Id]         [varchar]  (3)    NOT NULL ,
    [D1]         [varchar]  (20)   NOT NULL ,
    [D2]         [varchar]  (20)   NOT NULL ,
    [D3]         [varchar]  (20)   NOT NULL ,
    [Class]      [varchar]  (20)   NOT NULL ,
    [Comment]    [varchar]  (255)  NULL
) ON [PRIMARY]
```

3.7 Web_Character_Engine

User_Info_Store에서 가져온 사용자 정보와 Character_Part_Store에서 추출된 캐릭터 구성 요소의 기본형 데이터에 각 변형된 포인터 값을 변경 및 재조합한다. 수정된 각 부분 데이터를 새로운 포인터 값으로 생성한 뒤, SVG 형식의 데이터를 생성하고 Web_Character_Store에 저장한다. 또한 사용자가 기존에 생성하였던 캐릭터를 수정하는 경우에도 데이터를 Web_Character_Store에서 가져온다는 것을 제외하고는 비슷한 절차에 따라 수행된다.

그림 3은 Web_Character_Engine에서의 데이터 처리 흐름을 나타내고 있다. 캐릭터의 색상 처리를 위해 Class_Info_Store에서 Class에 따른 색상 값을 저장하고 있다.

그림 3에서 Part_Character_Before 파일은 해당 Part명을 생성하고, User_Info_Store의 User_ID, Character_Part_Store에서의 ID, Class, D를 추출한 뒤 ActiveX Control로 구현한 UI(User Interface)를 이용하여 데이터를 수정, 변환한다. 또한 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.

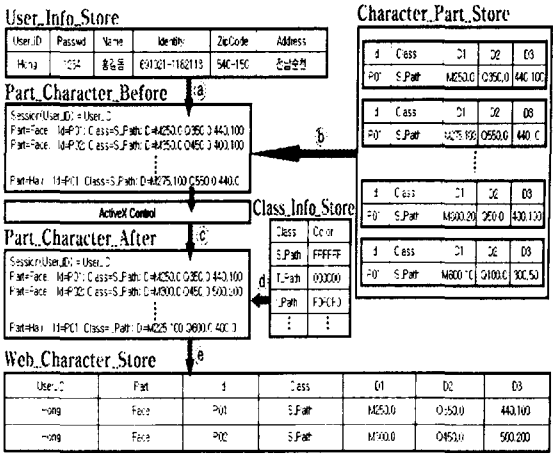

Fig. 3 Data Flow of Web_Character_Engine

그림 3의 데이터 변환 흐름에서 각 부분의 기능은 다음과 같다.

ⓐ, ⓑ 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 보낸다.

ⓒ, ⓔ 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 수정된 데이터를 Web_ Character_Store에 저장한다.

ⓓ 색상 데이터 선택

사용자가 선택한 색상으로 Class_Info_Store에서 Class명을 추출한다.

3.8 Web_Character_Viewer

Web_Character_Store에서 생성한 데이터를 XML/SVG 형식의 웹 문서로 변환하는 역할을 한다. 이렇게 변환되어진 웹 문서는 사용자가 사용하는 브라우저에서 XML/SVG 형식의 웹 캐릭터를 디스플레이 한다.

Web_Character_Store에 저장되어 있는 데이터를

XML/SVG 형식으로 변환하는 시스템을 구현하는데 MS SQL 2000에서 제공하는 Template 방식을 사용하였다. MS SQL에서는 템플릿 파일을 XML형식으로 변환하기 위하여 URL Query나 XPath Query, Template Query 등을 지원하고 있으나, 본 연구에서 Template 방식을 채택한 이유는 다음과 같은 장점이 있기 때문이다.

① 템플릿 실행결과를 유효한 XML 문서로 만드는 것이 가능하다.
② SQL문이나 XPath Query에 전달할 수 있는 매개변수를 정의할 수 있다.
③ Namespace를 선언할 수 있다.
④ XSL 스타일 시트를 결과 문서에 적용하도록 지정할 수 있다.
⑤ 보안을 향상시킨다.

XML의 구성을 정의하는 DTD(Document Type Definition)는 내부 DTD로 구현되었고, Web Character의 XML Source의 유효성 여부를 판단한다. Web Character를 구성하는 DTD는 표 5와 같다.

Table. 5 DTD of Web Character

```
<!ELEMENT CHARACTER (FACE, NOSE, EYE, MOUSE,
EAR, HAIR, EYEBROWS)> - ①

 <!ELEMENT FACE        (path)> --- ②
 <!ELEMENT NOSE        (path)>
 <!ELEMENT EYE         (path)>
 <!ELEMENT MOUSE       (path)>
 <!ELEMENT EAR         (path)>
 <!ELEMENT HAIR        (path)>
 <!ELEMENT EYEBROWS    (path)>

 <!ELEMENT path      EMPTY > --- ③
  <!ATTLIST  path
             id    ID       #REQUIRED
             class CDATA  #IMPLIED
             d     CDATA  #REQUIRED >
```

Ⅳ. 구 현

본 논문에서는 사람 캐릭터의 얼굴 부분을 대상으로 파일럿 시스템(Pilot System) 개발하였고, 얼굴의 부분요소로는 얼굴, 코, 눈, 머리, 입, 귀, 눈썹의 7가지로 나누어 캐릭터를 생성하였다.

표 6은 본 논문에서 구현한 MS SQL을 이용한 Template Source의 Header 부분으로 XML로 구현될 때 어느 파라미터를 사용할 것인지를 정의하며, 각 부분의 기능은 다음과 같다.

① 유효한 XML문서를 만들기 위해 ROOT 엘리먼트를 생성하는 부분이다. 속성으로는 두 가지를 사용하고 있는데 첫 번째는 xmlns: sql="urn: schemas-microsoft-com:xml-sql"로 NameSpace를 정의하고 있는 부분이다. 웹 상에서 같은 XML 태그가 사용될 때 구분하기 위해서 사용된다. 그리고 두 번째 속성은 XSL을 지정하는 것이다. 즉, 결과문서를 일정한 포맷으로 보고 싶다면 이 곳에 XSL을 지정한다.

Table. 6 Header of Character Template

```
<ROOT xmlns:sql-"urn:schemas-microsoft-com:xml
-sql" sql:xsl-'Character_XSL.xsl' >---------- ①

 <sql:header>------------------------------ ②
 <sql:param name-'USER_ID'></sql:param>-- ③
<!---------------------------------- ④
 <sql:param name-'PART_1'>FACE</sql:param>
 <sql:param name-'PART_2'>NOSE</sql:param>
 <sql:param name-'PART_3'>EYE</sql:param>
 <sql:param name-'PART_4'>HAIR</sql:param>
 <sql:param name-'PART_5'>MOUSE</sql:param>
 <sql:param name-'PART_6'>EAR</sql:param>
 <sql:param name-'PART_7'>EYEBROWS
    </sql:param>----------> </sql:header>
```

② 파라미터를 생성할 수 있는 템플릿 내의 정의된 부분이다. 반드시 header라고 정의되어 있는 부분에서 파라미터를 생성할 수 있다. 이렇게 생성된 파라미터는 저장프로시저나 템플릿, XPath Query 어디에서나 사용이 가능하다. Web Character Template에서는 USER_ID와 부분 요소를 나타내는 7개의 Part명을 파라미터로 처리하였다.
③ 어느 사용자의 웹 캐릭터인지 확인하고 적용하는 매개변수 USER_ID를 정의한다.
④ 거의 동일한 패턴의 템플릿이지만 각 부분 요소에 맞는 정의를 하기 위해 각각의 Part명을 매개변수로 사용한다. 각각의 Part명이 기본 값이 되어 만일 템플릿 상에서 정의되지 않아도 기본 값을 그대로 사용하게 된다. 본 연구에서는 표에서 볼 수 있는 것처럼 얼굴 캐릭터의 부분요소로 얼굴, 코, 눈, 머리, 입, 귀, 눈썹의 7가지를 고

려하였다.

표 7은 표 6에서 정의한 파라미터를 이용하여 MS SQL 문장으로 얼굴 캐릭터의 부분요소 데이터를 추출하는 부분으로 각각의 기능은 다음과 같다.

Table. 7 Query for Character Template

```
<!--
<FACE>------------------------------ ⑤
 <sql:query>------------------------- ⑥
  SELECT   D, CLASS ---------- ⑦
  FROM     PATH ---------------- ⑧
  WHERE    USER_ID = @USER_ID  ⑨
    AND      PART    = @PART_1
    FOR      XML     AUTO
 </sql:query>
 </FACE>------------------------------>
<!--
 <NOSE>------------------------------ ⑩
 <sql:query>
          SELECT   D, CLASS
          FROM     PATH
          WHERE    USER_ID = @USER_ID
          AND      PART    = @PART_2
      FOR      XML     AUTO
 </sql:query>
 </NOSE>------------------------------>
<!--
중략---------------------------------- ⑪
</ROOT>
```

⑤ FACE Part 부분을 생성하는 템플릿이다.

⑥ Namespace가 sql로 선언이 되어 있듯이 사용되는 자식 엘리먼트의 명칭이 "sql:"로 시작한다. sql: query 라고 선언되어 있는 이 부분에 SQL 문장을 삽입할 수 있다.

⑦, ⑧ Path라는 엘리먼트를 생성하고 좌표값과 색상값인 D와 CLASS라는 속성을 생성하는 구문이다. 쿼리의 FROM절에 있는 테이블 명이 엘리먼트가 되고 추출하는 필드 명이 속성이 된다. 모든 부분에 거의 동일한 형식으로 제시된다.

⑨ Path 테이블에서 사용자가 Login한 ID값과 일치하는지 확인한다.

⑩ NOSE Part 부분을 생성하는 템플릿이다.

⑪ EYE Part, HAIR Part, MOUSE Part, EAR Part, EYEBROWS Part 부분을 생성하는 템플릿이다.

이렇게 정의한 템플릿은 Web_Character_ Store에서 완성된 캐릭터를 호출한다.

4.2 사용자 인터페이스

구현된 아바타 생성 시스템은 로그인과 7가지의 얼굴 부분 요소들을 정의할 수 있는 사용자 인터페이스로 구성되어 있다.

그림 4는 Web Character 생성을 위한 첫 로그인 화면이다.

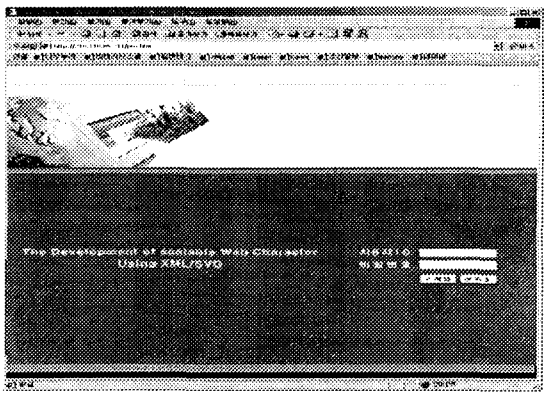

Fig. 4 Login of Avata Construction System

그림 4의 로그인 화면을 이용해 로그인이 되면 웹 캐릭터를 생성할 수 있는 그림 5의 화면으로 이동한다. 그림 5는 로그인했을 때, 처음으로 보여지는 캐릭터 생성 화면이다.

그림 5의 화면은 3개의 프레임으로 구성되어 있다. Character_Part_Store 또는 Web_ Character_Store에서 가져온 데이터들을 결합하여 표현하는 좌측 상단 화면과 캐릭터의 7개 Part인 FACE, NOSE, MOUSE, EYE, EAR, HAIR, EYEBROWS를 선택할 수 있는 좌측 하단 메뉴 화면, Control이 적용되어 가변 포인트 값을 수정할 수 있는 메인 화면으로 구분된다. 좌측 상단에 완성된 화면은 Character_Part_ Store에 있는 부분요소 데이터나 Web_ Character_Store에 저장된 데이터를 템플릿을 통해 가져온 후 XML 형식으로 된 캐릭터를 디스플레이 한다.

그림 5에서 사용자가 부분 요소 중 얼굴과 관련된 데이터를 수정할 수 있는 화면을 볼 수 있다. 사용자에게 제공되는 얼굴 모형에는 8개의 가변 포인트가 있다. 이 포인트는 사용자가 얼굴 모형을 변형하는 데 있어서 수정할 수 있는 가장 최적화된 위치의 포인트이다.

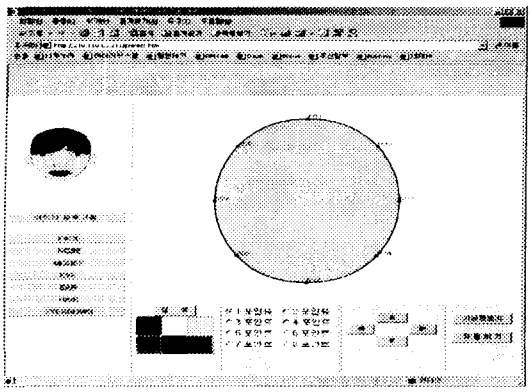

Fig. 5 User Interface

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.

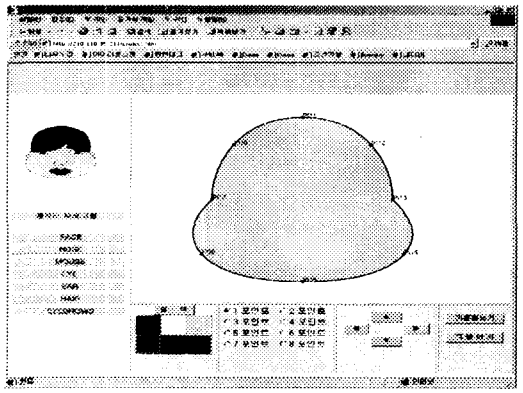

Fig. 6 Result after changing FACE control point

그림 5의 기본으로 제공되는 화면에서 FACE의 4번 포인트와 6번 포인트를 변경한 화면이 그림 6이다. 포인트 값을 변경한 후 좌측 상단의 완성 캐릭터의 "캐릭터 새로고침" 버튼을 클릭하면 변경된 캐릭터가 디스플레이 되고, 하단 우측의 "적용하기" 버튼을 클릭하면 데이터베이스에 저장된다. "적용하기" 버튼 바로 위의 "기본형보기" 버튼을 선택하면 각 부분 요소들에 대한 기본적인 캐릭터 중에서 본인이 원하는 것을 선택할 수 있다.

본 논문에서 구현한 웹 캐릭터 생성 파일럿(Pilot) 시스템은 위와 같은 방법으로 사람 얼굴의 7개 부분 요소에 대해 고정적인 가변 포인트를 가지고 있는 기본 캐릭터를 제공한다. 사용자는 각 기본 캐릭터를 선택하여 수정할 수 있는 포인트의 좌표 값을 수정함으로써 크기 및 모양을 결정할 수 있다. 캐릭터 자체가 SVG 형식으로 되어 있으므로 수정된 좌표 값이 바로 적용될 수 있다. 비트맵 형식의 캐릭터 부분 요소에는 많은 데이터가 필요하지만, 본 논문에서는 벡터 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 생성시키면 Web_ Character_Engine은 최종적으로 수정된 7개 부분 요소의 SVG 데이터와 사용자의 정보를 Web_ Character_Store에 저장하여 언제나 그 사용자의 독자적인 캐릭터를 이용할 수 있게 한다.

V. 결 론

현재까지 인터넷에서 제공되는 그래픽들은 향상된 그래픽 생성 툴들에 의해 시각적으로는 향상되어 있으나 데이터베이스와 연계한 동적인 그래픽을 제공하지 못하였다. SVG(Scalable Vector Graphic)는 그래픽의 시각적인 질을 보장하면서도 동적이고 상호작용적인 인터페이스가 강조되는 분야에도 사용이 가능한 XML 기반의 차세대그래픽 언어이다.

이에 본 논문에서는 XML과 SVG를 이용하여 다양한 해상도의 단말기에서도 이미지의 손상없이 디스플레이가 가능하고, 부분 요소들의 재사용성을 크게 높인 아바타(또는 웹 캐릭터) 생성 시스템을 설계 및 구현하였다. 구현 시스템은 그래픽 표현을 위해 벡터 형식을 사용하므로 화소 단위로 표현되는 비트맵 이미지와는 달리 확대, 축소 시에도 이미지에 변형이 일어나지 않으며 웹 상에서 만들어지는 어떠한 컨텐츠에도 사용할 수 있다.

향후 연구과제로 본 논문에서 구현한 웹 캐릭터는 기존 비트맵 형식의 캐릭터가 제공하는 색상에 비해

색상 선택이 원활하지 않아서 색상에 대한 보완이 있어야 한다. 또한 사용자가 마우스를 사용하여 원하는 부분을 직접 선택하여 수정하지 못하고 정해져 있는 포인트 값을 방향키를 이용하여 간접적으로 수정해야 하는 점이 개선되어야 할 것이다.

참 고 문 헌

[1] Accessibility Features of SVG W3C Note 7 August 2000
http://www.w3.org/TR/2000/NOTESVG-access-20000807/

[2] Adobe User to User Forum
http://www.adobe.com/support/forums/ main.html

[3] David Hunter외 5인 공저, "Beginning XML", pp.1-16, 정보문화사, 2000

[4] Doing it with SVG
http://wdvl.internet.com/Authoring/Languages/XML/SVG/Doinglt/index.html

[5] 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

[6] Frank Boumphrey외 11인 저, "Professional XML Application", p.979, 정보문화사, 1999

[7] Integration by Parts: XSLT, XLink & SVG
http://www.xml.com/pub/a/2000/03/22/style/index.htm

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

[9] SVG : putting XML in the picture
http://www.gca.org/papers/xmleurope2000/papers/s34-03.html#toc

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

저 자 소 개

김원중(Won-Jung Kim)
1987년 전남대학교 계산통계학과(이학사)
1989년 전남대학교 대학원 전산통계학과(이학석사)
1991년 전남대학교 대학원 전산통계학과(이학박사)
1999년~2000년 Iowa State University 교환교수
1992년~현재 순천대학교 정보통신공학부 부교수
※관심분야 : 소프트웨어공학, 시스템 모델링, 객체지향 시스템, 인터넷 서비스

장미화(Mi-Hwa Jang)
1999년 순천대학교 전자계산학과(이학사)
2001년 순천대학교대학원 컴퓨터과학과(이학석사)
※관심분야 : 시스템 모델링, XML응용, 인터넷 서비스