3ds max의 익스포트를 이용한 3차원 SVG 표현

김승완° 권오봉 박덕규 정혜진
{kswsamson°, obgwun, parkdg11}@chonbuk.ac.kr, hi-jin@daum.net

3D SVG Presentation Using Export of 3ds max 3D

Seungwan Kim° Oubong Gwun Deokgyu Park Hyejin jeong
Division of Electronics and Information Engineering,Chonbuk National University

요 약

SVG(Scalable Vector Graphic)는 웹 개발자 디자이너 및 사용자가 간단한 선언 방식의 프로그래밍 모델을 통해 HTML의 한계를 뛰어 넘어 견고한 비주얼 컨텐츠와 대화형 기능을 작성할 수 있는 W3C의 표준 XML 기반의 이미징 모델이다[1]. 웹에서의 SVG는 확장형 벡터 그래픽으로서 2차원 이미지를 이미지의 손상 없이 표현하게 된다. 이 논문에서는 2차원에 국한되어 있는 SVG를 확장하여 3차원 이미지를 표현하고자 한다. 2차원 이미지의 표현은 x축과 y축의 평면 축만 있는데 비해 3차원 이미지의 표현은 깊이 정보인 z축을 가지고 있어야 3차원 이미지를 표현하게 된다. 비트맵 그래픽과 달리 벡터 그래픽인 SVG를 이용하여 웹 브라우저에서 3차원 오브젝트를 표현하는 방법에 대해 고찰하고자 한다.

1. 서 론

컴퓨터에서 이루어지는 실세계의 표현은 3차원 모델링을 사용해야만 가능하다. SVG는 2차원 그래픽을 표현하기 위해 XML을 기반으로 만들어진 그래픽 표준 언어이다. SVG는 SMIL, GML, MathML 등과 같은 XML 언어들과 결합시켜 다양한 웹 애플리케이션으로의 응용이 가능하며[2]. 더 나가서 SVG는 3차원에서도 충분히 사용이 가능하다. 즉 SVG는 2차원 이미지를 보이기 위한 웹 애플리케이션이나. 여기에서는 SVG를 사용하여 2차원 이미지에만 국한하여 표현하지 않고 3차원 이미지 표현을 위해 SVG를 확장하기로 한다. 3차원 이미지를 표현하기 위해서는 xy축만을 정의하지 않고, 깊이를 나타내는 z축의 정보를 포함하고 있어야 한다. 이렇게 xyz축의 정보를 가지고 3차원 SVG를 구현하고자 한다.

2장에서는 SVG의 구현 원리에 대하여 설명하고, 3장에서는 3차원 그래픽 방식에 대하여, 4장에서는 3차원 오브젝트를 표현하기 위한 SVG의 구현 방법에 대하여 설명한다. 마지막으로 5장에서는 결과를 보인다.

2. SVG의 구현 원리

SVG란 XML 그래픽 표준으로 XML의 개방성과 상호운용성 등의 장점을 벡터 그래픽에 모두 수용한 것으로 응용 분야에는 여러 가지 광고, 전자상거래, 프로세스 컨트롤, 지리정보, 교육 등의 많은 그래픽을 필요로 하는 분야에 적용된다. SVG의 장점은 텍스트로 기술이 되기 때문에 기존의 그래픽보다 검색이 용이하고, 여러 애플리케이션들이 SVG 문서를 사용하는데 있어 용이하다. 또한 라인, 폴리곤[3], 텍스트, 이미지 등의 모든 그래픽 요소에 쉽게 접근이 가능하므로 데이터베이스와 연동하여 웹 그래픽 문서의 동적인 생성이 가능하다.

스케일러블(Scalable)이란 해상도에 관계없이 이미지의 손상이 되지 않는 확대, 축소가 가능한 웹 관련 기술이다. 또한 XML의 마크업 언어로 기술이 되기 때문에 무한한 확장성을 포함하고 있고, 다른 XML 문서에 참조될 수 있으며 다른 SVG 그래픽 내부에도 포함이 가능하다.SVG의 문서 구조는 XML 문서 구조와 비슷하며 이미지를 표현하기 위해서는 다음과 같은 구조를 사용한다. SVG에서는 2차원 오브젝트를 표현하기 위한 엔티티(Enity)는 사용되고 있지만, 3차원 오브젝트를 표현하기 위한 엔티티는 사용되고 있지 않다.

```
<?xml version = "1.0" standalone = "no" ?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 20000802//EN"
"http://www.w3c.org/TR/2000/CR-SVG-20000802/DTD/svg-20
000802.dtd">
<svg width = "300" height = "300">
  <rect x = "100" y = "100" width = "100" height = "70"
  style = "fill: red" />
</svg>
```

[그림 1] rectangle

[그림 1]과 같이 단순한 2차원 이미지나 애니메이션 등을 이용할 때는 2차원을 표현하는 SVG 파일만으로도 가능하지만, 3차원 오브젝트를 표현하기 위해서는 z축을

고려해야 한다.

3. 3차원 오브젝트의 표현

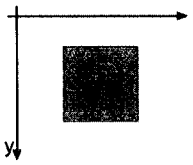
3차원 오브젝트를 표현하기 위해서 선행되어야 하는 것은 3차원 오브젝트의 생성이다. 2장에서 언급한 SVG를 이용하여 2차원 이미지를 표현하는 것은 어렵지 않다. 그러나 3차원을 표현하기 위해서는 z축을 고려하여야 한다. 3ds max를 사용하여 육면체[그림 2]를 생성하여 SVG 파일로 변환을 하고자 한다. 이 때, 3ds max에서 사용되는 익스포트(export) 파일은 ase 파일로 아스키(ASCII) 값으로 저장되는 파일이다. 여기에서는 육면체의 8개의 점에 대한 좌표만을 가지고 있다. 이 8개의 좌표를 모두 순차적으로 연결하여 육면체를 생성한다.

```
*MESH {
  *TIMEVALUE 0
  *MESH_NUMVERTEX 8
  *MESH_NUMFACES 12
  *MESH_VERTEX_LIST {
    *MESH_VERTEX    0    -5.0000  -5.0000  0.0000
    *MESH_VERTEX    1    5.0000   -5.0000  0.0000
    *MESH_VERTEX    2    -5.0000  5.0000   0.0000
    *MESH_VERTEX    3    5.0000   5.0000   0.0000
    *MESH_VERTEX    4    -5.0000  -5.0000  10.0000
    *MESH_VERTEX    5    5.0000   -5.0000  10.0000
    *MESH_VERTEX    6    -5.0000  5.0000   10.0000
    *MESH_VERTEX    7    5.0000   5.0000   10.0000
}
```

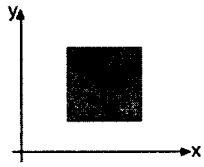
[그림 2] 3ds max에서의 box

SVG를 이용하여 [그림 2]와 같은 육면체를 생성하기 위해서는 각 점의 xy축의 정보만을 가지고 표현해야 한다. 여기에서 xyz축의 좌표를 회전하면 우측의 그림과 같이 표현된다. SVG의 이미지 표현에서는 래스터 그래픽[4]이 아닌 벡터 그래픽을 사용하기 때문에 각 축의 마이너스(-) 축이 없다. 2차원에서 벡터 그래픽 방식과 래스터 그래픽 방식의 축의 표현 방법은 [그림 3]과 [그림 4]에서 나타낸다. 래스터 그래픽의 경우는 실세계의 좌표계와 동일하지만, 비트맵 그래픽에서는 실세계의 좌표계와의 y축을 반대로 생각해야 한다. 3ds max에서는 래스터 그래픽 방식이며, SVG에서는 비트맵 방식이다. 이 점을 고려하여 3ds max에서 생성한 박스를 ase 파일로 익스포트하여 SVG로 구현하고자 한다.

4. 구현 및 결과



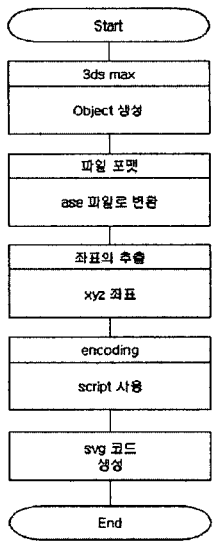
[그림 3] 벡터 그래픽



[그림 4] 비트맵 그래픽

4.1 제안 알고리즘

선행되어진 3차원 오브젝트의 표현에서 각 축을 45°씩 회전하였다. 회전되어진 방향에서 스크린에 오브젝트를 투영하기 위해서는 각각의 회전되어진 정점의 좌표를 구한다. 각 정점의 xy축의 좌표를 구하여 SVG 파일로 변환을 한다. 이 변환이 끝나면 웹에서 2차원 이미지만을 표현할 수 있는 2차원 SVG 파일을 3차원 이미지로 표현할 수 있다. 이를 표현하기 위하여 다음과 같은 알고리즘을 제안한다.



[그림 5] 제안알고리즘

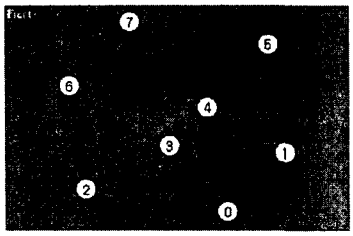
첫 번째 단계는 3ds max에서 간단한 오브젝트를 생성하는 것으로부터 시작한다. svg는 XML 기반의 텍스트로 작성된 언어이기 때문에 텍스트를 사용하는 아스키 파일로 저장한다. 3ds max에서의 아스키 파일은 ASE 파일 포맷으로 각 정점과 면을 텍스트로 표현할 수 있다. ASE 파일로 변환한 후 각 정점의 좌표를 추출해야 한다. 추출된 좌표로 svg에서 3차원으로 표현할 수 있는 코드를 생성할 수 있다. 각각의 좌표를 추출하는 데 있어서 오브젝트의 뒷면은 보이지 않기 때문에 계산할 필요가 없다. 다음은 각 정점의 xy 좌표를 추출한 ase 파일의 한 부분이다. 각 정점의 z축을 생각하지 않고, xy축

만을 순차적으로 연결하여 3차원 오브젝트의 표현이 가능하다.

```
*MESH {
  *TIMEVALUE 0
  *MESH_NUMVERTEX 8
  *MESH_NUMFACES 12
  *MESH_VERTEX_LIST {
    *MESH_VERTEX    0    12.5000   6.4645   27.5000
    *MESH_VERTEX    1    17.5000   13.5355  32.5000
    *MESH_VERTEX    2    3.9645    11.4645  28.9645
    *MESH_VERTEX    3    8.9645    18.5355  33.9645
    *MESH_VERTEX    4    11.0355   1.4645   36.0355
    *MESH_VERTEX    5    16.0355   8.5355   41.0355
    *MESH_VERTEX    6    2.5000    6.4645   37.5000
    *MESH_VERTEX    7    7.5000    13.5355  42.5000
  }
```

4.2 SVG로의 변환

웹에서 간단한 오브젝트인 3차원 이미지를 보이기 위한 준비가 되었다. 회전되어 있는 3차원 오브젝트의 경우에도 각 정점에서의 xy축의 값이 구해져 있으므로 SVG 파일에서는 쉽게 이 오브젝트를 구현할 수 있다. 3차원 오브젝트를 표현하는 방법은 3ds max에서 익스포트된 ASE 파일을 분석하여 여기에서 추출되는 메시버텍스(MESH_VERTEX)를 이용하여 간단한 폴리곤을 생성하는 SVG 파일의 구조에 xy축의 좌표값만을 입력하면 3차원 오브젝트의 SVG로의 표현이 가능하다. 이를 표현하기 위해서는 래스터 그래픽 방식인 3ds max의 정점의 순서[그림 6]를 인지해야 한다.



[그림 6] 3ds max의 정점의 순서

[그림 6]에 표현된 정점의 순서를 따라 표현이 되는 부분만을 SVG 코드에 끼워 넣어 3차원 SVG로의 표현이 가능하다. 각 면을 나타내기 위한 SVG 코드에서는 z축의 좌표 값이 아닌 xy축의 좌표 값만을 필요로 하기 때문에 z축의 계산은 제외한다. 또한 투명도가 없기 때문에 후면의 보이지 않는 면을 계산에서 제외한다. 이러한 방식으로 간단한 오브젝트의 xy축의 좌표 값만을 계산하여 줌으로써 간단하게 SVG 파일로 변환하여 표현할 수 있다. 각 정점의 순서는 정점 0을 기준으로 해서 좌표값을 추출한다. 각 좌표값의 추출은 다음과 같다.

FACE 1 : 0, 1, 5, 4
FACE 2 : 0, 2, 6, 4
FACE 3 : 0, 1, 3, 2

각각의 면에 대한 정점의 좌표를 이어 폴리곤을 생성한 후 SVG 코드에 삽입하면 모든 처리가 종료된다. 종료된 SVG 파일을 웹 브라우저에서 확인하면 3차원 육면체로 보이는 이미지가 생성이 된다. 각 면의 정점의 순서는 래스터 그래픽 방식과는 다르게 벡터 그래픽 방식으로 표현되므로 일반적인 그래픽 표현 방식과는 반대로 표현된다.

[그림 7] xy축 좌표 추출 후 SVG 변환

[그림 7]의 좌측과 우측은 육면체를 각각 3면으로 분리하여 SVG로 추출한 경우이다. 육면체의 모든 면을 계산하는 것보다 스크린에 출력되는 3면만을 계산하여 계산 속도의 양을 줄이게 된다. 출력되는 3면을 계산하여 SVG 파일로 변화하여 스크린에 출력한 결과는 [그림 8]과 같다.

[그림 8] SVG 변환 결과

5. 결론 및 향후 연구

지금까지 2차원만을 대상으로 한 XML 기반의 SVG에서 3ds max를 사용, ASCII 코드로 익스포트한 3차원 오브젝트를 웹에 표현하였다. 현재의 구현 결과는 회전된 육면체에 대한 xy축의 좌표만을 알고 있으면 되었다. 더욱 복잡한 오브젝트를 표현하기 위해서는 xy축의 좌표만이 아닌 z축 좌표의 정보를 포함하는 스크립트를 생성하여 변환하고자 한다.

[참고문헌]

[1] http://www.adobe.com
[2] http://multimedia.yonsei.ac.kr
[3] 나방현, 심규찬, 이종현 공저, XML 그래픽 입문, 21세기사, pp.36~44.
[4] Edward Angel, 구자영 역, OpenGL을 이용한 컴퓨터 그래픽스, pp.38~45.