# 안드로이드 플랫폼을 위한 3D 폰트 라이브러리

김경연*, 배하연, 권류혁, 김유성
*인하대학교 정보통신공학과
e-mail : yskim@inha.ac.kr

# 3D Font Library for Android Platform

KyoungYeon Kim*, HaYeun Bae, Ryu-Hyeok Gwon, Yoo-Sung Kim
*Dept. of Information and Communication Engineering, Inha University

요 약

본 논문에서는 안드로이드 플랫폼에서 3D 타이포그래피를 쉽게 표현할 수 있도록 지원하는 라이브러리를 설계 구현하였다. 현재까지 모바일 플랫폼에서 3D 타이포그래피를 지원하기 위한 라이브러리는 iOS에서 사용가능한 FTGL ES만이 공개되었고 이를 직접 안드로이드 플랫폼에서 사용할 수 없으며 제공되는 3D 표현 기능 또한 제한적이었다. 본 연구에서는 iOS용 FTGL ES를 안드로이드 플랫폼에서 사용할 수 있도록 변환하고 OpenGL ES기능을 활용하여 다양한 3D 폰트 표현을 지원할 수 있는 3D 폰트 라이브러리를 개발하였다. Freetype 라이브러리를 통해 폰트의 Glyph 정보들을 얻고 이를 이용해 테두리, 양각, 음각 등 다양한 3D 문자 표현 기능을 구현하였고, 공간상에서 문자들의 배치를 수학적으로 모델링하여 다양한 3D 문자 배치 기능을 구현하였다. 개발된 3D 폰트 라이브러리를 이용하여 다양하게 문자들을 3D 공간에 표현할 수 있으면서, 기존 3D 타이포그래피 표현 방법들에 비해 더 쉽고 빠르게 3D 타이포그래피를 표현할 수 있도록 지원한다.

## 1. 서론

국내에서는 2003년도에 발표된 엔씨소프트의 3D 온라인 게임 '리니지2'를 시작으로 애니메이션, 모바일 게임 등 계속해서 3D 콘텐츠가 강세를 보이고 있고, 최근엔 스마트폰이 널리 보급되면서 3D 콘텐츠들이 활발히 개발되고 있다[1]. 3D 콘텐츠의 활용이 확대되면서 3D 콘텐츠에서 사용하는 타이포그래피에 대한 관심 역시 증가 하였다.타이포그래피는 활판 인쇄술을 넘어 글자 및 활자에 관한 서체, 디자인, 조판, 가독성을 포함하고 그것에서 발생하는 조형적인 속성 모두를 말한다[2]. 따라서 3D 콘텐츠에 3D 타이포그래피가 사용된다면 평면적 공간에서는 불가능했던 문자 표현의 다양한 가능성들을 선사할 것이다[2]. 또한 기존의 평면에서의 타이포그래피에서는 불가능한 다양한 시점(view point)의 표현 및 변환, 3차원 공간에서의 배치 등은 문자열의 표현 가능성을 더욱 다양하게 제공하여 활용도를 높여줄 수 있을 것이다[2].

3D 타이포그래피는 아름다움과 읽기 쉬움이라는 두 가지 속성을 만족해야 한다[3]. 우선, 아름다움이란 디자인 측면에서 레이아웃을 의미한다. 효과적으로 사물을 배치하게 함으로써 아름다움을 증가시킨다. 둘째, 읽기 쉬움이란 언어 또는 내용 측면에서 글이 얼마나 읽기 쉽게 쓰여 있는가를 의미하는 'readability'가 아니라 디자인 측면에서 글자들을 배치한 모양이 얼마나 읽기 쉬[1]운가를 의미하는 'legibility'를 뜻한다. 읽기 쉬움은 글자의 배치와 효과와 시점에 따라 달라질 수 있다[3].

최근에는 안드로이드 모바일 플랫폼이 많이 이용되고 있기 때문에 안드로이드 플랫폼에서 3D 타이포그래피를 지원하는 3D 폰트 라이브러리의 필요성이 증대되고 있다[4]. 따라서 3D 폰트 라이브러리를 제작하기 위한 노력들이 꾸준히 존재해왔다. 처음에는 포토샵이나 일러스트를 이용해 문자를 하나씩 그려서 만들었는데 이는 많은 시간이 들어서 다양한 문자를 표현하는 데에는 한계가 있었다. 또한, 작성한 문자를 재활용하기 위해 폰트 파일을 3D처럼 보이도록 만들어서 타이포그래피에 사용하였지만, 이 또한 여러 시점으로의 표현이 불가능했고, 3D를 표현하기 위해 글꼴 렌더링 라이브러리인 FTGL과 FTGL ES가 개발되었지만, 공간상에서 문자들을 자유롭게 배치하는 등의 다양한 3D 표현이 부족하고, 안드로이드 플랫폼에서 사용이 불가능하다는 한계점을 갖고 있었다.

본 논문에서는 3D 타이포그래피의 속성인 아름다움과 읽기 쉬움을 만족하면서, 폰트 라이브러리가 가져야 하는 편리함과 자동화라는 측면을 더해 보다 쉽게 3D 타이포그래피를 안드로이드 플랫폼에서 사용할 수 있는 3D 폰트 라이브러리를 제안한다. 2D 폰트 라이브러리인 freetype과 FTGL ES를 안드로이드에 포팅한 후 3D 타이포그래피에서 필요로 하는 투명도 조절, 색상 변경, 위치 지정 등 다양한 문자 효과들과 트위스트, 반원, sin파 등의 수학적 모델링으로 구현된 문자열 효과들을 표현하여 기존 3D 타이포그래피가 만족해야하는 아름다움과 읽기 쉬움 뿐 아

이 논문은 (주)정글시스템과 인하대학교 산학협력 연구과제의 연구비 지원에 의하여 연구되었음

니라 편리함과 자동화를 더해 3D 타이포그래피를 쉽고 빠르게 지원하는 3D 폰트 라이브러리를 개발하였다.

본 논문의 구성은 다음과 같다. 2절에서는 관련연구를 설명하고, 3절에서는 제안하는 라이브러리에 대해 설명한다. 4절에서는 라이브러리의 실행결과를 예시하고, 5절에서는 결론을 내린다.

## 2. 기존 3D 타이포그래피 표현 방법

대부분의 3D 타이포그래피는 포토샵이나 일러스트를 이용하여 문장을 3D 이미지로 만들어서 사용하는 방식으로서 프로그램에서 코드나 문자를 불러와 사용하는 것이 아니기 때문에 미리 만들어 놓은 글자모양 혹은 문구만 사용이 가능하기 때문에 시간과 비용이 많이 들고, 재활용이 어렵다.

3D 폰트의 재활용성을 높이기 위해 3D로 표현된 폰트로 제작하는 방법이 등장하였다. 이 방법은 FontLab Studio나 FontForge와 같은 프로그램을 이용하여 스케치된 시안을 가지고 폰트를 생성한다. 이때 사용하는 스케치 자체를 평면 위에 3D처럼 보이는 폰트를 만든다[6]. 이는 한번 제작한 문자들을 계속 재활용 할 수 있지만, 제작한 폰트만 3D로 사용이 가능하며, 3D 이미지를 2차원 평면에 그린 것이기 때문에 폰트를 여러 시점에서 볼 수 없다.

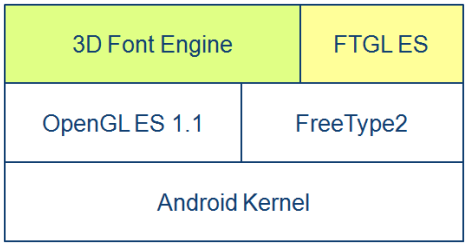
이후 여러 폰트를 3D로 표현하기 위해 FTGL을 사용하는 방법이 등장하였다[3]. FTGL은 글꼴 렌더링 라이브러리이다. 3D 그래픽스 라이브러리인 OpenGL과 Freetype 폰트 라이브러리를 사용하여 폰트 파일을 열고 디코딩 한 후 출력해 준다. 이 방법은 모든 폰트를 3D로 만들 수 있으며 Bitmaps, Anti-aliased pixmaps, Texture maps, Outlines, Polygon meshes, Extruded polygon meshes로 구분된 총 6가지의 모드를 지원하고 있다[6]. 이 중 Extruded만 3D 폰트 렌더링 모드이다. Extruded는 3D 공간에 3D 폰트를 그리는 것으로 다른 모드들과 달리 문자의 깊이(Depth)값을 이용해 3D 폰트를 표현한다[2]. FTGL은 컴퓨터에서 사용하는 라이브러리이고 모바일 플랫폼을 위한 FTGL ES가 존재하지만 iOS에서만 사용이 가능하고 아직 안드로이드는 지원이 되지 않고 있다. 또한 FTGL ES에서는 FTGL에서 지원했던 6가지 기능 중에서 Buffer, Outline, Polygon, Texture 4가지 모드만 지원을 해 3D로 표현되는 Extruded는 지원이 되지 않아 3D 폰트 렌더링이 불가능하다[6].

즉 현재는 안드로이드에서 3D 폰트들을 자유롭고 다채롭게 표현하는 라이브러리가 존재하지 않다. 본 논문에서는 안드로이드에서 여러 폰트들을 3D로 표현할 수 있고, 다양한 3D 효과를 줄 수 있는 3D 폰트 라이브러리를 제안한다.

## 3. 안드로이드용 3D 폰트 라이브러리

안드로이드 플랫폼에서 문자를 3D로 표현하기 위해선 폰트 라이브러리인 FreeType2와 3D 그래픽 라이브러리인 OpenGL ES가 필요하다. FreeType2를 이용해 폰트의 Glyph 정보를 얻어 오고, 기존 iOS에서 작동하는 FTGL ES를 안드로이드에서 사용할 수 있도록 JNI를 이용해 포팅하여 OpenGL ES를 이용해 글꼴의 3차원 렌더링을 지원한다. OpenGL ES를 이용해 렌더링 하였기 때문에 렌더링된 문자열들을 여러 시점에서 관측할 수 있고, OpenGL ES에서 제공하는 투명도 효과 등을 사용할 수 있다. 또 FTGL ES을 이용하였기 때문에 FTGL ES에서 제공하는 문자열 효과 기능과 레이아웃 효과 기능을 사용할 수 있다.

본 논문에서 제안하는 3D 폰트 라이브러리의 시스템의 구성은 (그림 1)과 같다.




(그림 1) 3D 폰트 라이브러리

안드로이드에 포팅된 FTGL ES에서는 <표 1>과 같은 기능들을 제공하고 있다. 폰트, 모드, 색상, 크기는 문자열에 대한 설정이고, 레이아웃 길이와 정렬은 레이아웃에 대한 설정이다. 텍스쳐 효과의 경우 iOS의 고유 기능을 사용하기 때문에 안드로이드에선 지원하지 않았다.

<표 1> 안드로이드 FTGL ES에서 사용 가능한 기능

| 기능 | 설명 |
|---|---|
| 폰트 | 적용하는 폰트 파일 |
| 모드 | 폰트 표현방법<br>종류 : (Bitmap, Outline, Polygon) |
| 색상 | 문자열의 색상, 범위 : RGBA 공간 |
| 크기 | 문자열의 크기 |
| 레이아웃 길이 | 문자열의 가로 최대 길이 |
| 정렬 | 문자열의 정렬 방법<br>종류 : (왼쪽\|오른쪽\|가운데\|양쪽) 정렬 |

FTGL ES에서 제공하지 않는 문자열 효과 기능들을 새롭게 추가하여 제안하는 3D 폰트 라이브러리를 구현하였다. 현재 개발된 라이브러리는 <표 2>와 같은 추가 기능들을 제공하고 있다. <표 2>의 기능들은 단순히 폰트에 따라 문자열을 그리는 것이 아니라, 3D 타이포그래피에서 필요한 아름다움과 읽기 쉬움에 추가로 편리함과 자동화를 갖춰서 보다 빠르고 손쉽게 3D 타이포그래피를 만들 수 있도록 도와준다.

우선 3D로 표현하기 위해 기존 모드에 Extruded 모드를 추가 구현하였다. 앞면만을 렌더링 하는 Polygon 모드와 다르게 새롭게 깊이(depth) 변수를 추가하여 앞면을 렌더링한 후, 깊이만큼 뒤에서 뒷면을 렌더링하고 앞면과 뒷면의 Glyph 점들을 연결하여 옆면을 렌더링 하였다. 또한 각 면의 렌더링 여부를 변수로 두어 양각, 음각을 표현할

수 있도록 하였다.

<표 2> 제안하는 라이브러리에서 사용 가능한 기능

| 기능 | 설명 |
|---|---|
| 양각 | 문자열이 앞으로 튀어나온 효과 |
| 음각 | 문자열이 뒤로 들어간 효과 |
| 면 단위 설정 | 면마다 색상, 텍스쳐, 투명도, 렌더링 여부를 설정할 수 있음 |
| 낱글자 설정 | 낱글자마다 색상, 텍스쳐, 투명도, 면 단위 설정, 위치를 설정 함 |
| 테두리 | 문자열의 테두리 두께 설정 |
| 텍스쳐 | 문자열에 텍스쳐 효과를 줌 |
| 그라데이션 | 문자열에 여러 색상을 이용해 그라데이션 효과를 줌 |
| 그림자 효과 | 문자열의 그림자를 표현함 |
| 소멸 점 효과 | 문자열의 소멸 점 효과를 줌 |
| 곡선/곡면 설정 | 곡선/곡면 효과를 주어 표현함<br>종류 : 반원, 나선효과, sin파 |

각 면을 렌더링하면서 각각 다른 효과를 주어 면 단위 설정이 가능하도록 구현하였다. 이와 비슷하게 각 글자의 순서를 변수로 두어 낱글자마다 위치를 비롯한 글자 설정을 할 수 있도록 하였다. 테두리는 Outset을 변수로 두어 Outset의 값의 크기에 따라 테두리의 굵기가 달라지도록 Outset만큼 떨어진 곳에서부터 옆면을 폴리곤으로 연결하여 표현하였다. 텍스쳐는 OpenGL ES의 텍스쳐 매핑 기능을 이용해 텍스쳐를 각 면에 덮어서 표현하였고, 여러 색상을 그라데이션으로 표현하여 텍스쳐를 만든 후 텍스쳐 매핑을 이용해 그라데이션 효과를 표현하였다. 그림자는 문자열의 위에 가상의 광원을 설정하고, 문자들 밑에 바닥을 설정하여 가상 광원으로부터 문자열을 바닥에 프로젝션 시켜서 그림자를 표현하였다. 그림자 프로젝션 행렬은 (그림 2)를 이용하여 구한다.

만약 광원의 점이 $L(l_0, l_1, l_2)$, 평면의 벡터가 $N(n_0, n_1, n_2)$, 평면의 한 점이 $E(e_0, e_1, e_2)$일 때

$d = N \circ L$ ( $\circ$ 은 벡터의 내적을 의미 함)

$c = E \circ N - d$

$K$: $K_{44}$ 값만 c이고 나머지는 0인 4×4 행렬

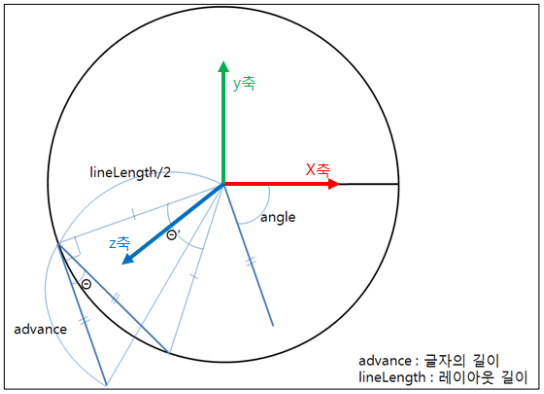
$$S = -(N|(c+d))(L|(1))^T + K$$

$$= -\begin{bmatrix} n_0 l_0 & n_0 l_1 & n_0 l_2 & n_0 \\ n_1 l_0 & n_1 l_1 & n_1 l_2 & n_1 \\ n_2 l_0 & n_2 l_1 & n_2 l_2 & n_2 \\ (c+d) l_0 & (c+d) l_1 & (c+d) l_2 & d \end{bmatrix}$$

(그림 2) 그림자 프로젝션 행렬

이렇게 구한 S 행렬을 객체에 곱하면 객체의 모습이 평면에 프로젝션 되어 나타나게 되고, 이를 투명한 회색으로 표현하여 그림자를 만들었다.

소멸 점은 뒷면을 없애고, 앞면의 점 들을 깊이(depth) 만큼 뒤에 있고 레이아웃의 중앙인 점에 연결하여 표현하였다. 또한 각 글자는 원의 안쪽에 배치하여 둥근 효과를 주었다. (그림 3)은 소멸 점에서 문자들의 배치 개념도이고, (그림 4)는 이에 대한 알고리즘이다.




(그림 3) 소멸 점에서 글자 배치 개념도

수식

$\theta = 90 - [acos(advance/lineLength)]/\pi \times 180$

$\theta' = [acos((1 - advance^2) * 2/lineLength^2)]/\pi \times 180$

(acos 함수 결과는 radian값)

알고리즘

레이아웃의 중심(lineLength/2,0,0)으로 평행

y 축을 기준으로 angle 만큼 회전

z 축으로 lineLength/2 만큼 평행 이동

y 축을 중심으로 $\theta$만큼 회전

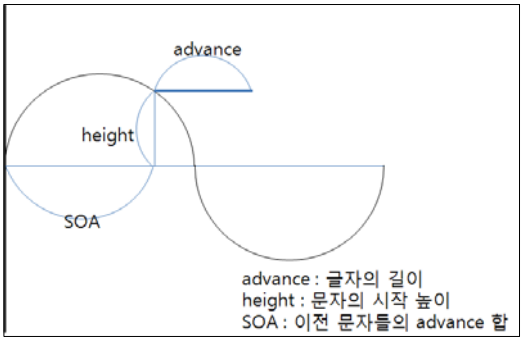
angle에 $\theta'$만큼 더함

문자 렌더링

(그림 4) 소멸 점에서 글자 배치 수식 및 알고리즘

곡선/곡면은 공간상에서 글자들을 수학적 모델링에 따라 배치하는 것을 의미한다. 종류에는 반원, 나선, sin파 효과가 있지만, 엄밀히 말하면 공간상에서 반원처럼 표현되는 소멸 점도 여기에 포함 된다. 반원과 나선효과에서 글자 배치 방법은 소멸 점의 배치 방법과 비슷하다. 반원은 소멸 점의 배치 방법에서 y축과 z축을 바꾼 상태에서 같은 알고리즘으로 계산하면 xz 평면위에 있는 원 위에 문자들이 표현된다. 나선효과는 소멸 점 알고리즘에서 레이아웃 중심에서 z축으로 이동한 후 문자 크기의 일정 비율만큼 높여주면 나선효과가 나타내게 된다.

(그림 5)는 sin파의 글자 배치 개념도로 (그림 6)의 알고리즘을 이용해 sin파 위에 각각의 단어들을 배치하였다.




(그림 5) sin파에서 글자 배치 개념도

수식

$height = facesize \times \sin(30 \times (index)/180 \times \pi)$

(sin 함수의 파라미터는 radian 값임)

알고리즘

y축으로 height 만큼 평행이동

x축으로 SOA(Sum Of Advance) 만큼 평행이동

문자 렌더링

(그림 6) 소멸 점에서 글자 배치 수식 및 알고리즘

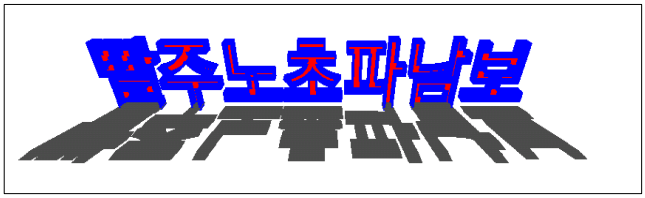
## 4. 실행결과

우리의 3D 폰트 라이브러리는 <표 1>과 <표 2>의 각 기능이 API로 제공된다. (그림 7)부터 (그림 10)까지는 각 기능을 이용해 문자들을 표현한 실행 결과이다.

(그림 7)은 <표 1>의 모든 기능과 <표 2>의 몇 가지 기능을 이용해 문자열을 표현한 실행 결과이다. 안드로이드 기본 내장 폰트를 이용해 표현하고, 면 단위 설정을 이용해 앞면은 검정, 옆면에 그라데이션 효과를 넣고, 테두리를 넣고 양각 표현을 하였다. 이후에 시점을 위쪽으로 옮겨서 비스듬한 문자열 표현하였다.

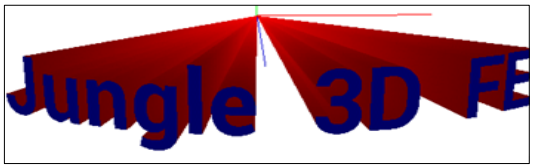

(그림 7) 테두리 및 그라데이션이 표현된 양각 문자열

(그림 8)은 앞면을 렌더링 하지 않고, 옆면은 파랑, 뒷면은 빨강으로 표현하여 음각 표현을 하였다. 이후에 가상의 광원을 문자열 뒤쪽 위에 두고 문자열 바로 밑에 평면을 만들어 그림자를 표현하였다.



(그림 8) 그림자가 표현된 음각 문자열

(그림 9)는 앞면은 파랑 옆면은 빨강하고 소멸 점 효과를 주었다. 또한 OpenGL ES에서 제공하는 실제 광원을 소멸 점에 적용하여 문자열이 소멸 점에 가까울수록 밝아지도록 표현하였다. 실행결과 문자열들이 원 처럼 둥글게 배치 된 것을 볼 수 있었다.




(그림 9) 소멸 점이 표현된 문자열

(그림 10)은 낱글자 설정을 이용해 글자색, 크기, 위치, 테두리 굵기 등을 다양하게 적용한 실행 결과이다. '주'와 '초'의 경우 테두리 굵기가 너무 커서 가독성이 떨어졌다.


(그림 10) 낱글자 설정이 표현된 문자열

## 5. 결론

본 논문에서는 널리 존재하는 2D 폰트들을 이용해 3D 타이포그래피를 쉽게 표현할 수 있는 라이브러리를 제안했다. 안드로이드에서 사용이 불가능했던 FTGL ES를 안드로이드에서 사용 가능하도록 포팅하고 기존의 FTGL ES의 폰트, 모드, 색상, 텍스쳐, 크기, 레이아웃 길이, 정렬 설정 뿐 아니라 양각, 음각, 면 면단위 설정, 곡선/곡면 등등 다양한 문자열, 레이아웃 효과들을 주어 3D 타이포그래피를 쉽게 표현할 수 있도록 구현하였다.

본 논문에서는 3D 타이포그래피의 주요 속성인 아름다움을 만족하기 위해 다양한 문자열과 레이아웃 효과들을 구현하였고, 편리함과 자동화를 만족하기 위해 각각의 기능을 API로 구현하였다. 각각의 기능들을 수학적으로 모델링하고, 이를 구현하여 표현함으로써 논문의 효율성을 검증하였다. 본 논문에서 제안한 API들은 주로 레이아웃 위주로 배치 및 조화를 위한 기능들이다. 3D 타이포그래피는 레이아웃 뿐 아니라 문자 각각을 색다르게 표현하는 다양한 효과들도 필요로 한다. 이에 우리는 보다 다양하게 3D 폰트를 표현하는 방법에 대해 연구를 계획 중이다. 이를 통해 보다 쉽고 빠르게 3D 타이포그래피 뿐 아니라 다양한 3D 콘텐츠에서 사용가능한 3D 폰트 라이브러리를 만들려고 한다.

## 참고문헌

[1] 이기현 외 3인, 2013년 콘텐츠산업 전망(서울:한국콘텐츠진흥원, 2013)

[2] 정훈동, "3D 타이포그래피의 공간적 활용: 입체적 공간상의 실험을 중심으로"(디자인학 연구 통권 제 80호 (Vol. 21 No. 6), pp 53-58

[3] 조진환, "TEX: 조판, 그 이상의 가능성", The Asian Journal of TEX, Vol. 1, No. 1, 2007, pp 1-16

[4] 박재형 외 3명, "모바일 3D 기술", TTA저널 No. 97, 2007, pp 114-149

[5] *"FontLab Studio"*, FontLab Ltd.(Jul 2012), <http://www.fontlab.com/font-editor/fontlab-studio/>

[6] "About FTGL", Sourcefoi(22 Dec 2009), <http://sourceforge.net/apps/mediawiki/ftgl/index.php?title=Main_Page>