**Project**

**Computer Graphics**

|  |  |
| --- | --- |
| 프로젝트 명 | Magazine-style Photo Viewer |
| 문서 제목 | Final Report |

|  |  |
| --- | --- |
| **제출일** | 2014-06-23 |

|  |  |
| --- | --- |
| **팀원** | 김 성 근 |
| 김 현 준 |
| 김 연 상 |
| 최 준 석 |

**CONFIDENTIALITY/SECURITY WARNING**

이 문서에 포함되어 있는 정보는 국민대학교 전자정보통신대학 컴퓨터공학부 및 컴퓨터공학부 개설 교과목 캡스톤 디자인I 수강 학생 중 프로젝트 “ Magazine-style Photo Viewer ”를 수행하는 팀 “ Francis ”의 팀원들의 자산입니다. 국민대학교 컴퓨터공학부 및 팀 “ Francis ”의 팀원들의 서면 허락 없이 사용되거나, 재 가공 될 수 없습니다.

**목 차**

[1 개요 4](#_Toc347414841)

[1.1 프로젝트 개요 4](#_Toc347414842)

[2 개발 내용 및 결과물 5](#_Toc347414844)

[2.1 목표 5](#_Toc347414845)

[2.2 연구/개발 내용 및 결과물 6](#_Toc347414846)

[2.2.1 연구/개발 내용 6](#_Toc347414847)

[2.2.2 시스템 기능 및 구조 설계도 6](#_Toc347414848)

[2.2.3 활용/개발된 기술 6](#_Toc347414849)

[2.2.4 결과물 목록 6](#_Toc347414851)

[3 자기평가 7](#_Toc347414853)

[4 참고 문헌 7](#_Toc347414854)

[5 부록 7](#_Toc347414855)

[5.1 사용자 매뉴얼 7](#_Toc347414856)

# **개요**

## **프로젝트 개요**

본 설계과제는 4번의 과제를 통하여 총 5주 이상 진행되며 단계적인 일정에 의해 진행된 프로젝트입니다.

**[과제 #01] 잡지 스타일의 레이아웃 구성.**

* 사각형들을 이용하여 잡지 스타일의 photo-viewer의 프레임들을 보여주는 기본적인 OpenGL 프로그램 구현한다.

**[과제 #02] 상호작용이 가능한 프로그램 구현.**

* 1번 과제에 기능을 추가하도록 하며, 사용자의 마우스 상호작용에 따라 책장을 넘기는 효과 (Animation)와 사진의 확대 축소, 드래그 & 무브를 할 수 있는 기능을 구현한다.

**[과제 #03] 잡지 스타일의 Photo Viewer 완성.**

* 2번 과제에 기능을 추가하도록 하며, 시스템의 특정 Directory에 있는 사진들을 Texcture Mapping 기능을 이용하여 보여주고 사각형 프레임 안에서 사진이 떠다니는 효과를 구현한다.

위의 3개의 과제를 종합하여 [과제 #04] 웹 브라우저에서 구동되는 잡지 스타일의 Photo viewer를 완성 하는 것입니다.

# **개발 내용 및 결과물**

## **목표**

1. **웹 브라우저에서 구동되는 잡지 스타일의 레이아웃 구성.**

* 사각형들을 이용하여 잡지 스타일의 photo viewer의 프레임들을 구현.

1. **웹 브라우저에서 구동되는 상호작용이 가능한 프로그램 구현.**

* 사용자의 마우스 상호작용에 따라 책장을 넘기는 효과를 구현.
* 사진의 확대 축소, Drag & Move를 할 수 있는 기능을 구현.

1. **웹 브라우저에서 구동되는 잡지 스타일의 photo viewer 완성.**

* 시스템의 특정 Directory 에 있는 사진들을 Texture Mapping 기능을 이용하여 구현.
* 사진이 떠다니는 효과를 구현.

1. **과제 요구사항 이외의 추가적 기능 구현.**

* Lighting 기능
* Intrinsic 속성 변환기능
* 여러 개의 Texture Mapping 기능
* 사진이 떠다니는 속도 변환 기능
* Texture Filter 변환가능 기능

## **연구/개발 내용 및 결과물**

### **연구/개발 내용**

1. 웹 브라우저에서 구동되는 잡지 스타일의 레이아웃 구성.  
   가장 기본적인 color shader를 구현하여 정상적으로 잡지 스타일이 그려지는 지를 먼저 판단하였습니다. 가정 처음에 했던 과제#01에서와 같이 많은 수의 Vertex를 미리 선언해두고 화면 Click 시 Page Frame이 변경되는 지 여부를 확인하였습니다.  
     
   -Fragment color shader  
     
   -Vertex color shader  
     
   - Matrix code

**precision** **mediump** *float*;

*void* main(*void*) {

**gl\_FragColor** = *vec4*(1, 1, 0, 1);

}

**attribute** *vec3* a\_vertex;

**uniform** *mat4* mvp\_matrix;

*void* main(*void*) {

gl\_Position = mvp\_matrix \* *vec4*(a\_vertex, 1.0);

}

mat4.ortho(projection\_matrix,

-webgl.attribute.orthoWidth-0.1, webgl.attribute.orthoWidth+0.1,

-webgl.attribute.orthoHeight, webgl.attribute.orthoHeight+0.2,

0.1, 100.0);

mat4.lookAt(view\_matrix,

[webgl.attribute.lookX, webgl.attribute.lookY, 1],

[webgl.attribute.lookX, webgl.attribute.lookY, 0],

[0, 1, 0]);

……

mat4.multiply(view\_model\_matrix, view\_matrix, model\_matrix);

mat4.multiply( projection\_view\_model\_matrix,

projection\_matrix, view\_model\_matrix);

1. 사용자의 마우스의 상호작용에 따라 책장을 넘기는 기능 구현  
   Canvas에 들어오는 Mouse Event에 대해서는 javascript DOM을 직접접근 하여 Handle Function을 연결하도록 구현하였습니다. ‘eventHandler.js’ 파일에 Canvas에 발생하는 Event에 대하여 모두 작성하였습니다.  
     
   책장을 넘기는 효과는 따로 timer를 주는 것이 아닌 계속해서 Canvas가 업데이트 되는 것을 이용하여 구현 하였습니다. 아래의 코드에서 보이는 것처럼 C++ glutMainLoop()로 계속 도는 것과 같이 계속 그리게 되는데 그 부분에 회전을 시키는 함수를 주어 클릭으로 인한 회전이 발생할 시 실행될 수 있도록 구현하였습니다.  
   - Webgl Animate  
     
     
   Texture Mapping을 이용한 회전을 위해서 Buffer를 상단과 하단을 구분하여 그렸습니다. 따라서 x 축을 중심으로 그림이 접히듯이 회전하게 됩니다. 회전은 Rotate를 이용하였습니다.  
   -Webgl Rotate

webgl.animate = **function**(){

requestAnimFrame(webgl.animate);

webgl.rotationEffect();

webgl.drawScreen();

};

mat4.rotateX(

model\_matrix, model\_matrix, webgl.attribute.topAngle\*Math.PI/180);

1. 사진의 확대 축소, Drag & Move를 할 수 있는 기능 구현  
   2.번에서 설명했듯이 Canvas에 들어오는 Mouse Event는 DOM에 Handler Function을 연결하였습니다.  
     
   Drag를 시키기 위해서 확대를 먼저 구현하였습니다. 2가지 방식으로 구현하였는데 첫번째 방법은 Ortho에 보이는 화면의 크기를 변경하는 것과 Buffer Vertex에 Scale 화 시켜주는 방법을 모두 구현 하였습니다. 1번 Matrix-code 의 ‘webgl.attribute.orthoWidth’ 이 있는 부분이 Ortho에서의 화면 확대를 위해서 입니다.  
     
   -Webgl Scale

mat4.scale(

model\_matrix, model\_matrix, [webgl.attribute.zoom, webgl.attribute.zoom, 1.0])

1. Texture Mapping 기능  
   Texture를 초기화 하고 Texture를 Mapping 시키는 부분은 모두 ‘texture.js’에 작성하였습니다. 기본적인 Texture Mapping Code는 아래와 같으며, TEXTEURE\_FILTER를 사용자가 정의 할 수 있도록 구현하였습니다. 또한 아래에 코드에는 나와 있진 않지만 Texture가 변경 될 수 있도록 하였습니다.  
   - Texture Mapping  
     
     
   또한 이전까지 사용되었던 Color Shader가 아닌 Texture Shader를 이용하였습니다. Texture Shader 코드는 아래와 같습니다.  
   - Texture Fragment shader  
     
   - Texture Vertex shader  
     
     
   Texture Shader를 구현하고도 color shader를 다른 파일로 계속 이용하였는데 그 이유는 Texture Mapping 없이 흰색 Color로만 선을 그리기 위해서 사용하였습니다.

gl.bindTexture(gl.TEXTURE\_2D, tex);

gl.pixelStorei(gl.UNPACK\_FLIP\_Y\_WEBGL, **true**);

gl.texImage2D(gl.TEXTURE\_2D, 0, gl.RGBA, gl.RGBA, gl.UNSIGNED\_BYTE, image);

gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_WRAP\_S, gl.CLAMP\_TO\_EDGE);

gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_WRAP\_T, gl.CLAMP\_TO\_EDGE);

**if**(webgl.attribute.textureFilter == webgl.TEXTURE\_FILTER.LINEAR){

gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_MAG\_FILTER, gl.LINEAR);

gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_MIN\_FILTER, gl.LINEAR\_MIPMAP\_LINEAR);

webgl.errorHandler('Linear');

}**else** **if**(webgl.attribute.textureFilter == webgl.TEXTURE\_FILTER.NEARST){

gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_MAG\_FILTER, gl.NEARST);

gl.texParameteri(gl.TEXTURE\_2D, gl.TEXTURE\_MIN\_FILTER,

gl.NEARST\_MIPMAP\_NEARST);

webgl.errorHandler('Nearst');

}

gl.generateMipmap(gl.TEXTURE\_2D);

gl.bindTexture(gl.TEXTURE\_2D, **null**);

**precision** **mediump** *float*;

**varying** *vec2* v\_texcoord;

**uniform** *sampler2D* uTexture;

*void* main(*void*) {

**gl\_FragColor** = texture2D(uTexture, v\_texcoord);

}

**attribute** *vec3* a\_vertex;

**attribute** *vec2* a\_texcoord;

**uniform** *mat4* mvp\_matrix;

**varying** *vec2* v\_texcoord;

*void* main(*void*) {

gl\_Position = mvp\_matrix \* *vec4*(a\_vertex, 1.0);

v\_texcoord = a\_texcoord;

}

1. 사진이 떠다니는 효과 구현  
   사진이 떠다니는 효과는 Buffer의 Texture 부분을 증가시키거나 감소시키고 다시 Buffer를 불러오는 식으로 구현하였습니다. 또한 계속해서 값을 바꾸기 위하여 javascript의 setTimeout을 이용하였습니다.  
   - Flooding Code

webgl.onYincrease = **function**(){

**if**(webgl.attribute.scaling){

**return**;

}

webgl.attribute.yMin += webgl.attribute.fSpeed2;

webgl.attribute.yMax += webgl.attribute.fSpeed2;

buffer.texture();

**if**(webgl.attribute.floating){

**if**(webgl.attribute.yMax > 0.95 ){

setTimeout(webgl.onYdecrease, 500);

}**else**{

setTimeout(webgl.onYincrease, 30);

}

}

}

1. Lighting 기능 구현

프로젝트에 마지막으로 Light 기능을 구현 하였습니다. 가장 먼저 Shader를 구성하였고 그 Shader에 맞게 Javascript를 조작하였습니다. Light는 Phong Reflection Model을 기반으로 구성하였습니다.   
- Light Vertex Shader  
  
  
- Light Fragment shader  
  
Light 기능은 모두 페이지 안에서 수정 가능하게 구성하였으며, Light 모드를 사용하거나 하지 않는 것을 지정하였습니다.

**attribute** *vec3* a\_vertex;

**attribute** *vec3* a\_vertexnormal;

**attribute** *vec2* a\_texcoord;

**uniform** *mat4* mv\_matrix;

**uniform** *mat4* p\_matrix;

**uniform** *mat3* n\_matrix;

**varying** *vec2* v\_texcoord;

**varying** *vec3* v\_transformednormal;

**varying** *vec4* v\_position;

*void* main(*void*) {

v\_position = mv\_matrix \* *vec4*(a\_vertex, 1.0);

gl\_Position = p\_matrix \* v\_position;

v\_texcoord = a\_texcoord;

v\_transformednormal = n\_matrix \* a\_vertexnormal;

}

**precision** **mediump** *float*;

**varying** *vec2* v\_texcoord;

**varying** *vec3* v\_transformednormal;

**varying** *vec4* v\_position;

**uniform** *bool* uShowSpecularHighlights;

**uniform** *bool* uUseLighting;

// Material

**uniform** *float* uMaterialShininess;

// Ambient

**uniform** *vec3* uAmbientColor;

// Point

**uniform** *vec3* uPointLightingLocation;

**uniform** *vec3* uPointLightingSpecularColor;

**uniform** *vec3* uPointLightingDiffuseColor;

**uniform** *sampler2D* uTexture;

*void* main(*void*) {

*vec3* lightWeighting;

**if** (!uUseLighting) {

lightWeighting = *vec3*(1.0, 1.0, 1.0);

} **else** {

*vec3* lightDirection = normalize(uPointLightingLocation - v\_position.xyz);

*vec3* normal = normalize(v\_transformednormal);

*float* specularLightWeighting = 0.0;

**if** (uShowSpecularHighlights) {

*vec3* eyeDirection = normalize(-v\_position.xyz);

*vec3* reflectionDirection = reflect(-lightDirection, normal);

specularLightWeighting = pow(max(dot(reflectionDirection, eyeDirection), 0.0), uMaterialShininess);

}

*float* diffuseLightWeighting = max(dot(normal, lightDirection), 0.0);

lightWeighting =

uAmbientColor

+ uPointLightingSpecularColor \* specularLightWeighting

+ uPointLightingDiffuseColor \* diffuseLightWeighting;

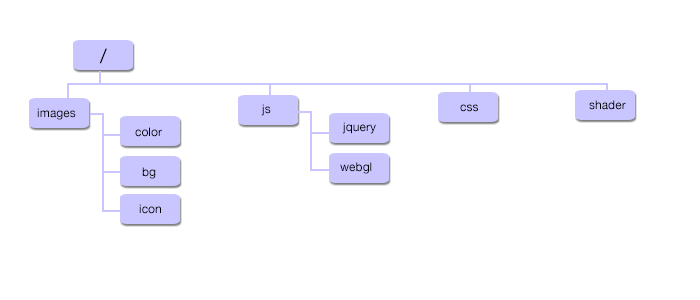
}

*vec4* fragmentColor = texture2D(uTexture, *vec2*(v\_texcoord.s, v\_texcoord.t));

**gl\_FragColor** = *vec4*(fragmentColor.rgb \* lightWeighting, fragmentColor.a);

}

### **시스템 기능 및 구조 설계도**



시스템의 구조는 위에 보이는 사진과 같습니다. images 폴더는 texture Mapping에 필요한 이미지를 모아둔 곳이고. js에는 javascript 파일, css는 css 파일 shader에는 shader파일을 모아 두었습니다.

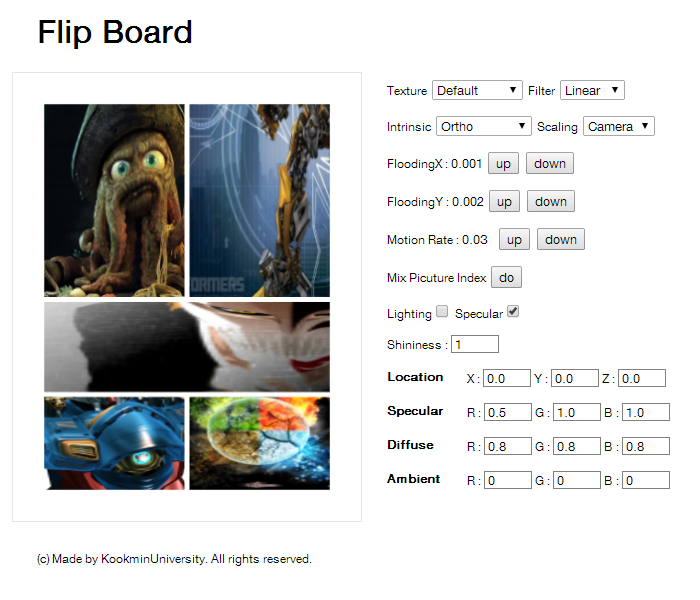
### **활용/개발된 기술**

DOM 의 Event 활성화와 Element를 선택하는 것에 편하게 하기 위해서 Jquery를 이용하였습니다.

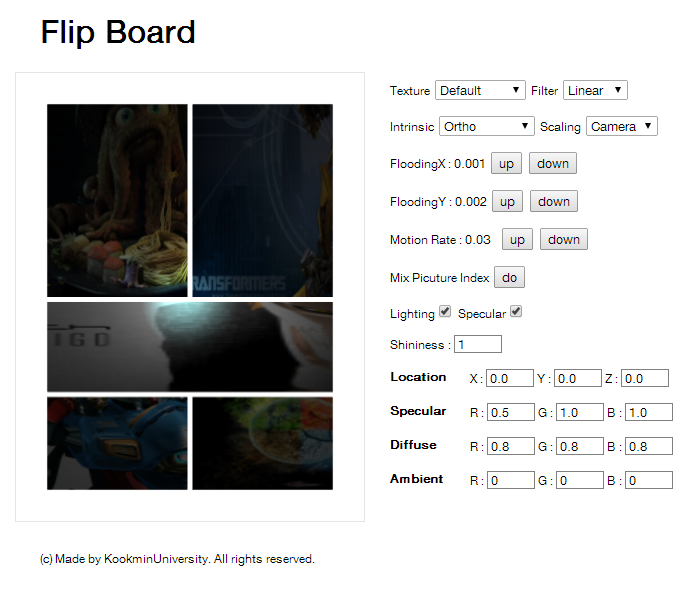
또한 Shader 파일을 외부 스크립트로 읽기 위해서 ShaderLoader를 인터넷 상에서 받아서 사용했으며 기본적인 원리는 Ajax를 이용하여 script file을 불러와 string으로 저장시켜주는 역할을 합니다.

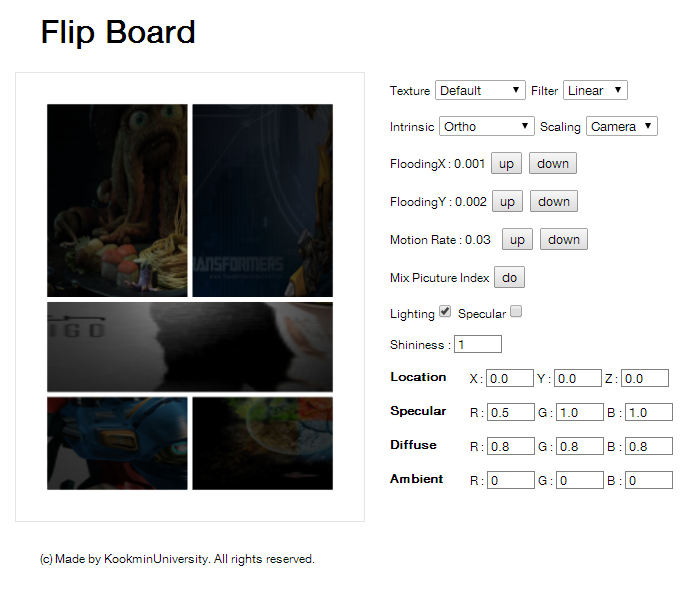
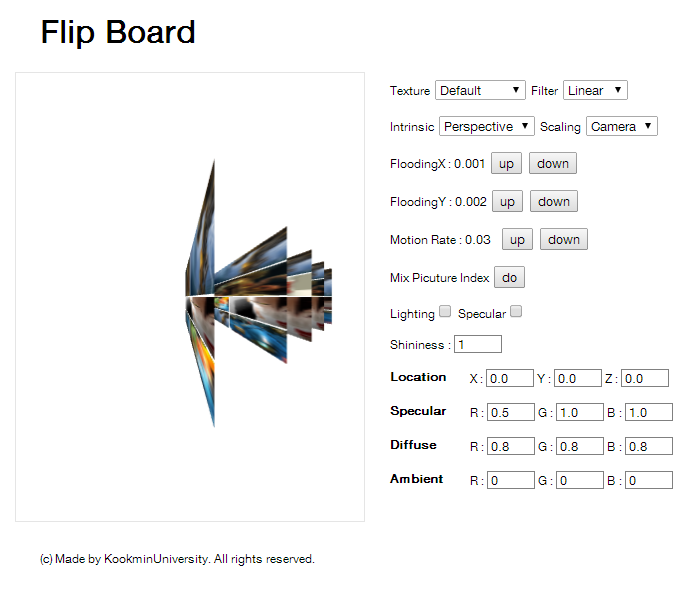
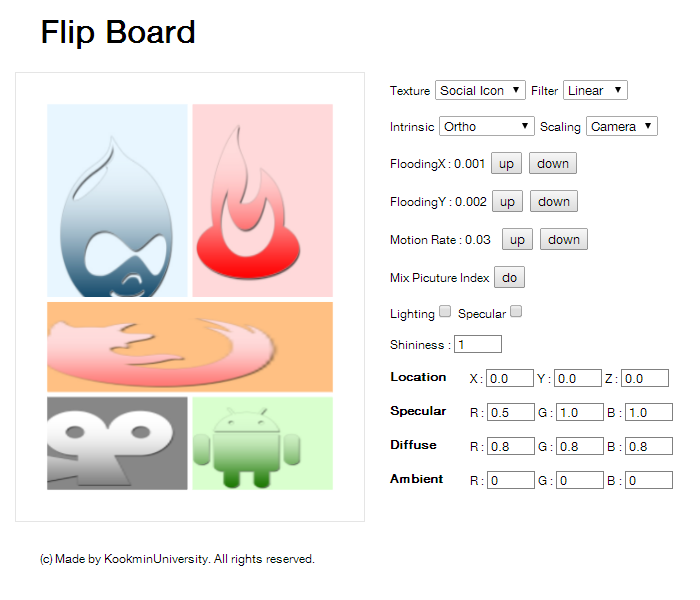
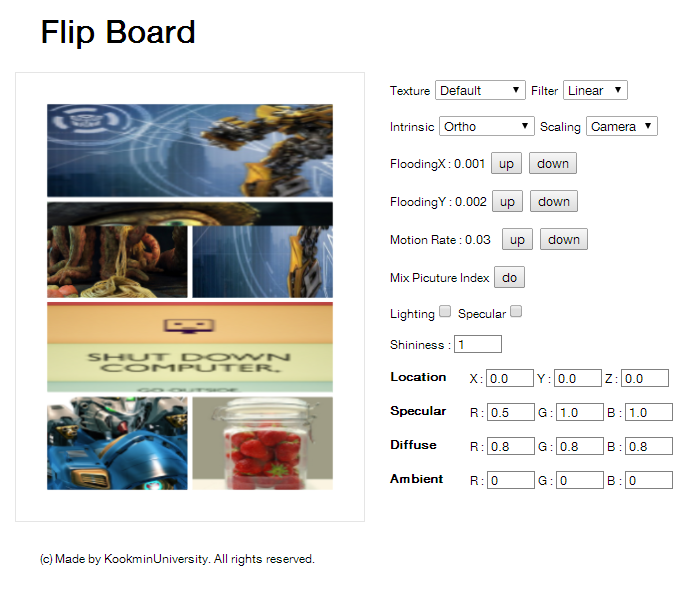
### **결과물 목록**

- Default



- Light & Highlight



- Light  
  
- Perspective View  
  
  
  
  
- Other Texture  
  
- Rotation  


# **자기평가**

2014학년도 그래픽스 과목의 설계 과제를 통해 그래픽스의 기본적인 개념을 openGL과 webGL을 활용하여 설계 과제를 수행하였다. 과제를 수행하기 앞서 우선적으로 Rendering Pipeline, Synthetic Camera Model, Synthetic Object, Shading, texture mapping등의 주요 그래픽스 개념들의 학습이 필요하였다. 첫 번째 과제를 통해 물체가 어떠한 방식으로 그려지는 지를 확인, 두 번째 과제를 통해 마우스의 상호작용에 따라 물체가 움직이는 애니메이션과 이벤트에 대한 구현, 세 번째 과제를 통해 그려진 프레임 범위 내에서의 텍스쳐 맵핑, 네 번째 과제를 통해서는 webGL을 활용하여 앞서 세 번째의 과제를 하는 동안 사용되었던 C언어를 사용하지 않고 스크립트 언어를 사용하여 웹에 대한 전반적인 개념도 다시 한번 더 확인 할 수 있는 계기가 되었다. 평소 관심이 많던 분야라 더 많은 기능적 요소를 추가하고 싶었으나, 여러 가지의 제한사항에 의해 많은 기능을 구현하지 못한 것이 아쉬움으로 남는다.

# **참고 문헌**

|  |  |  |
| --- | --- | --- |
| 번호 | 종류 | 출 처 |
| 1 | website | http://learningwebgl.com/blog/ |
| 2 | website | http://glmatrix.net/ |
| 3 | website | http://andre-cruz.com/webgl-shader-loader-js/ |
| 4 | website | http://jquery.com/ |

# **부록**

## **사용자 매뉴얼**

마우스 이벤트

|  |  |
| --- | --- |
|  | 프레임의 상단이 하단으로 내려오는 효과를 보이며 다음 프레임이 나타남. |
| 프레임의 상단 클릭 |

|  |  |
| --- | --- |
|  | 프레임의 하단이 상단으로 올라가는 효과를 보이며 이전 프레임이 나타남. |
| 프레임의 하단 클릭 |

|  |  |
| --- | --- |
|  | 마우스 휠 조정 값에 따라 확대/축소. |
| 마우스 휠 조작 |

|  |  |
| --- | --- |
|  | 사용자가 원하는 texture를  선택할 수 있다.  (Default, Social Icon, Color) |
| Texture tab |

|  |  |
| --- | --- |
|  | 사용자가 원하는 카메라 속성값으로 전환이 가능하다.  (Ortho, Perspective) |
| Intrinsic tab |

|  |  |
| --- | --- |
|  | 사용자가 원하는 antialiasing값을  지정할 수 있다.  (Linear, Nearst) |
| Filter tab |

|  |  |
| --- | --- |
|  | Scaling의 속성을 물체의 Scaling 또는 카메라의 이동으로 확대/축소 속성을 지정한다.  (Camera, Scale) |
| Scaling tab |

|  |  |
| --- | --- |
|  | 떠다니는 효과의 이동속도와 방향의 속성을 조작할 수 있다. |
| FloodingX or Y / Motion Rate |

|  |  |
| --- | --- |
|  | Do key를 누르면 texture값이 랜덤으로 섞이는 효과가 나타난다. |
| Mix Picuture index |

|  |  |
| --- | --- |
|  | Lighting속성 값을 지정할 수 있고 Lighting 우측 체크박스로 Lighting 사용여부를 확인 할 수 있다. |
| Lighting |