**Computer Graphics Term Project Report**

**3D filter**

-Post Processing-

**알파고알파고 조**

**20140908 컴퓨터공학과 박경원(pkw1012)**

**20140735 컴퓨터공학과 백우정(pcsewj14)**

**목차**

1. **프로젝트 개요**
2. **프로젝트 환경**
3. **프로젝트 설계 및 구현**
4. **프로그램 실행 방법**
5. **토론**
6. **결론**
7. **개선 방향**
8. **역할 분담**
9. **참고 문헌**
10. 프로젝트 개요

우리는 한 학기 컴퓨터 그래픽스 수강의 결과물로 수업에서 배운 것들을 활용하는 내에서 프로젝트를 진행하고 싶었다. shader를 적극 이용할 수 있는 주제를 찾고 싶었고, 그렇게 주제를 정하게 된 것이 3D filter 였다. 각 필터마다 shader를 다르게 쓰면서 fragment가 어떻게 다루어 지는지 더 잘 이해할 수 있을 것이라 생각하였다.

3D filter는 기존과 같이 사진에 적용하던 필터와는 달리 animation이 있는 3D modeling에 filter효과를 적용한다는 의미를 담고 있다. 이를 위해서는 animation이 있는 modeling을 해야했고, 구현할 필터를 선정하고 각 필터를 위한 shader를 구현해야 했다.

이 프로젝트의 modeling은 ‘효자시장으로 가는 길’을 기반으로 하였고, 구현한 필터는 아래와 같이 8가지이다.

* 1. 흑백 효과 (graycolor)
  2. 반전 효과 (inverting)
  3. 모자이크 (mosaic)
  4. 블러(흐림) (blur)
  5. 어안렌즈 (fish eve lens)
  6. 카툰 (cartoon)
  7. 유리창 효과 (embossed glass)
  8. 충격파 효과 (shake wave)

1. 프로젝트 환경

프로젝트는 assignment와 같은 환경인 Visual Studio 2015와 GLSL을 이용하였다.

1. 프로젝트 설계 및 구현

이 프로젝트의 구현은 크게 modeling과 shader로 나눌 수 있다.

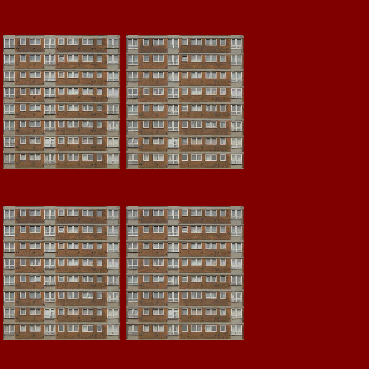
* 1. modeling

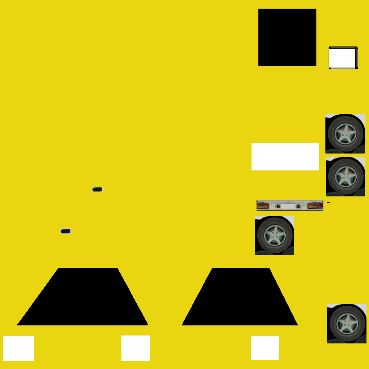
인터넷에 있는 오브젝트 파일 자료를 쓰려고 했지만, ‘효자시장 가는 길’을 직접 modeling을 해보는 것도 나쁘지 않다고 판단하였다.

modeling에는 metasequoia라는 프로그램을 사용하였다.



직접 프로그램을 사용하여 modeling을 하여 texture 역시 직접 만들어 mapping을 하였다. texture file은 대부분 인터넷의 자료를 받아 사용하였다.





그 중 animation을 구현에 이용되는 model은 자동차이다. model을 만드는 것은 상당히 손이 많이 갔지만, model을 불러와서 정해진 위치에 놓는 것은 assignment에서 많이 해봤던 것이라 어렵지 않았다.



modeling의 결과는 위와 같다.

참고 https://github.com/IlhwanHwang/CSED451\_assn2/blob/master/assn2/model.cpp

* 1. shader

shader는 우리 프로젝트에서 가장 중요한 부분이라고 할 수 있는데, 그 이유는 적용되는 필터마다 각자 다른 shader를 적용해야했기 때문이다.

shader는 uniform으로 받는 int값에 따라서 적용될 것을 구분했기 때문에 한 shader안에서 모두 구현할 수 있었다.

* + 1. 흑백 효과 (graycolor)

void main() {

color = texture(screenTexture, TexCoords);

float sum = color.x + color.y + color.z;

float ave = sum / 3;

color = vec4(ave, ave, ave, 1);

}

위의 코드를 보면 먼저 fragment에 색을 입히고 난 후, 그 색의 RGB값을 평균 내어 세 값을 모두 동일하게 만든 후, 다시 fragment에 집어 넣는 것을 알 수 있다. 그 결과로 아래와 같이 색이 모두 무채색으로 변한 화면을 볼 수 있다.



* + 1. 반전 효과 (inverting)

void main() {

color = texture(screenTexture, TexCoords);

float r = 1 - color.x;

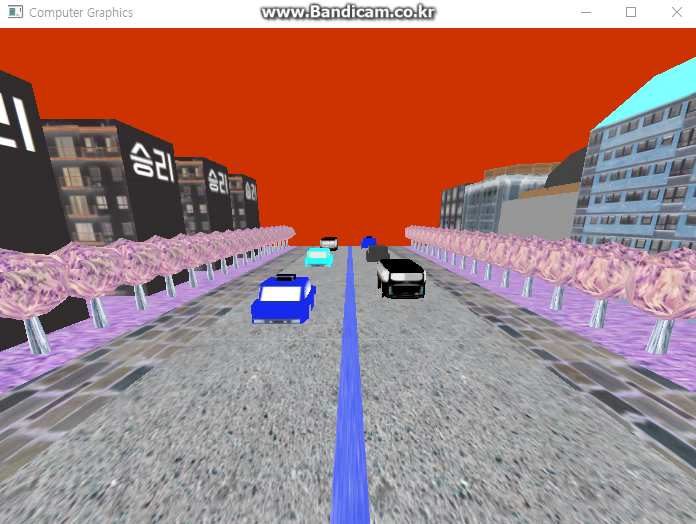
float g = 1 - color.y;

float b = 1 - color.z;

color = vec4(r, g, b, 1);

}

반전은 흑백과 RGB값을 건드린다는 점에서 매우 비슷하다. 다만 연산이 조금 다른데, fragment의 RGB값을 각각 1에서 빼 다시 fragment에 넣어주면 반전 효과를 나타낼 수 있다.



* + 1. 모자이크 (mosaic)
    2. 블러(흐림) (blur)
    3. 어안렌즈 (fish eve lens)
    4. 카툰 (cartoon)
    5. 유리창 효과 (embossed glass)

const float DIV = 50.0;

vec2 distort(vec2 v)

{

float theta = atan(v.y, v.x);

float radius = sqrt(v.x \* v.x + v.y \* v.y);

radius = pow(radius, 2);

v.x = radius \* cos(theta);

v.y = radius \* sin(theta);

return 0.5 \* (v+1.0);

}

void main() {

vec2 uv;

int ix = int(TexCoords.x \* DIV);

int iy = int(TexCoords.y \* DIV);

float x = float(ix) / DIV;

float y = float(iy) / DIV;

vec2 xy = vec2(TexCoords.x - x - (1/DIV), TexCoords.y-y-(1/DIV));

xy = 2\* DIV \* xy - 1.0;

float dist = sqrt(xy.x \* xy.x + xy.y \* xy.y);

uv = distort(xy) / (2 \* DIV);

uv.x += x+0.025;

uv.y += y+0.025;

color = texture(screenTexture, uv);

}

* + 1. 충격파 효과 (shake wave)

void main() {

vec2 center = vec2(upleft.x/700.0, (500.0 - upleft.y)/500.0);

vec2 uv = TexCoords.xy;

vec2 texCoord = uv;

float distance = distance(uv, center);

if( (distance <= (time + shockParams.z)) &&

(distance >= (time - shockParams.z)) )

{

float diff = (distance - time);

float powDiff = 1.0 - pow(abs(diff\*shockParams.x), shockParams.y);

float diffTime = diff\*powDiff;

vec2 diffUV = normalize(uv - center);

texCoord = uv + (diffUV \* diffTime);

}

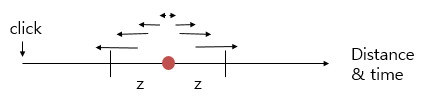
color = texture(screenTexture, texCoord);

}

충격파는 시간을 이용한다는 점에서 다른 필터들과 구분된다.

center는 마우스가 클릭된 fragment coordinate 좌표를 texture coordinate 좌표로 바꾼 값이다.

color는 기본적으로는 해당 좌표에 해당하는 texel값이지만, 먼저 아래와 같이 거리와 시간이 같은 스케일이라 가정하고, 어떤 좌표가 시간으로부터 주어진 parameter z의 범위 내에 있을 때 uv에 time에 해당하는 distance와의 거리에 비례하는 vector를 더해서 퍼진 듯한 모양이 나오도록 한다.



그 결과는 아래와 같다.

캡쳐 상으로는 잘 보이지 않지만 가운데 아래쪽에 크게 도넛 모양의 wave가 존재한다는 것을 알 수 있다.



* 1. mouse

마우스를 이용하는 방법을 생각해 보았다. 마우스의 event callback 함수를 만들어서 마우스가 click/release 한 좌표를 uniform으로 shader에 넣어주었다. 그리고 그 값으로 영역을 만들어서 그 곳에만 filter를 위한 연산을 하도록 하였다.

다만 어안 렌즈의 경우에는 선택된 영역을 전체로 보고 다시 값을 계산하도록 하였다. 때문에 항상 선택된 영역의 중심에 구 모양으로 왜곡된 형상이 존재하게 되었다.



만화 효과가 선택된 영역에만 적용된 것을 볼 수 있다.



또한 승리 아파트 쪽을 영역으로 선택했을 때, 영역의 중심인 아파트가 둥글게 왜곡된 것을 확인할 수 있다.

1. 프로그램 실행 방법

먼저 ctrl + F5를 누르면 빌드가 된다. 이 때 헤더파일 Angel.h, vec.h, mat.h와 shader program을 불러와주는 InitShader.cpp가 반드시 있어야한다. 이 파일들은

<http://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SIXTH_EDITION/CODE/include/> 에서 얻을 수 있었다.

이후 m을 사용해서 필터를 바꿀 수 있으며, 마우스로 필터가 적용되는 영역을 바꿀 수도 있다. 또한 방향키를 이용하면 시점을 바꿀 수 있다.

마우스로 영역을 지정할 때 드래그를 하지 않으면 넓이가 0인 region을 선택한 것으로 인식하기 때문에 filter가 적용되지 않은 모습으로 나오게 된다. 그럴 경우에는 마우스 오른쪽 버튼을 클릭하면 filter가 다시 적용된 것을 볼 수 있다.

1. 토론

이번 프로젝트는 처음 설계부터 우리가 시작한다는 점에서 assignment와는 확실히 달랐다. 주제를 정하고 구현하는 과정에서 많은 생각을 해야했다. 먼저 프로젝트로서 장점이자 단점은 filter를 구현하는 것이어서 매우 병렬적인 코딩이 가능했다. shader끼리는 영향을 주고받지 않기 때문에 어떤 filter를 구현하는 것이 막혀도 다른 filter에는 전혀 영향을 주지 않았다.

그리고 구현에 있어서 먼저 modeling의 경우는 간단하지만 상당히 시간이 많이 가는 일이었다. 그래픽스 수업을 듣고 modeling을 직접 해서 object 파일을 만들어보는 것도 의미 있을 것 같아 처음 배워서 해보았는데, 꽤 재미있었다. 프로젝트 파일에서 main.cpp를 제외한 모든 파일은 다 object 파일의 load를 위한 것이다. module화가 덜 이루어져서 code reuse가 많지만 중요한 건 shader라고 생각하여 큰 신경을 쓰지 않았다.

shader는 먼저 구현할 filter를 결정한 뒤, 구현을 했는데, filter를 몇 개 구현할지, 무슨 filter를 구현해야 할지 고민을 많이 했다. filter 선정은 먼저 ‘가장 익숙한 것’, 그리고 ‘구현해보고 싶은 것’을 고르다 보니 구현한 8가지의 filter가 나오게 되었다. 구현하면서 assignment에서는 사용하지 않았던 frame buffer나, edge detection, posterization 그리고 mouse callback function 등을 사용해볼 수 있어 좋았다. 여기 shader관련해서 좀 더 써줘

1. 결론

프로젝트는 직접 제작한 modeling에 filter를 입히는 식으로 진행되었고, modeling은 metasequoia라는 프로그램을 사용하여 직접 제작하였다. 3D filter에서 구현한 필터는 총 8개로 각 필터는 모두 fragment shader에서 구현 되었다. 그 외에 마우스를 이용해서 영역을 지정하는 것을 구현해보았다.

1. 개선 방향

우리가 흥미 있는 필터 위주로 구현하다 보니 난이도를 크게 생각하지 못한 것 같아 아쉽다. 만약 좀 더 고려를 했다면 더 challenging한 filter에 도전할 수 있지 않을까 한다.

1. 역할 분담

박경원 – 대부분의 shader 구현 (50)

백우정 – modeling 담당, shader 일부를 구현, 마우스 동작 구현 (50)

1. 참고 문헌

<http://www.cs.unm.edu/~angel/BOOK/INTERACTIVE_COMPUTER_GRAPHICS/SIXTH_EDITION/CODE/include/>

https://github.com/IlhwanHwang/CSED451\_assn2/blob/master/assn2

<http://www.geeks3d.com/shader-library/>

[http://learnopengl.com/#!Advanced-OpenGL/Framebuffers](http://learnopengl.com/)

<http://3dapi.com/dv16_shader4/>