A Waving Plane

Waving Color Plane에서 누구나 Plane을 먼저 만들 것이다. 이 Plane을 만드는 방법은 크게 2가지인것 같다.
1. xz-plane 위에 Plane을 만들고 y축 회전 → x축 회전
2. xy-plane 위에 Plane을 만들고 z축 회전 → x축 회전

두 방법 모두 비슷하게 동작할 것 같다. 나는 1번 방법을 사용했다.

참고로 회전행렬을 적용하는 순서는 중요하다. 입체적으로 잘 생각해보면 y축 회전 → x축 회전을 해야 실행 파일과 동일하게 나온다는 사실을 쉽게 알 수 있다. 또한 시간에 따라서 회전하는 것은 y축이다. 적절하게 θ_y을 $-t$로 두었다.

먼저 Plane을 만들었는데 WavingPlane 클래스를 만들어 생성자에서 동적할당 했다.

Division 수만큼 분할하는 것은 어렵지 않기 때문에 생략한다.

참고로 사각형을 삼각형 두개로 쪼개는 대각선의 방향에 따라서 모양이 다르다. 진폭이 크거나 Division의 수가 너무 작으면 부자연스러운 모습이 연출되는 이유이기도 하다.

하지만 어차피 90도씩 돌리면 대각선의 방향이 바뀌기 때문에 결국 한 방향으로 쪼갠다면 똑같다. 따라서 크게 신경쓰지 않았다.

다음은 Waving Color Plane에서 Waving이다. 이것도 여러가지 함수식을 접해봤다면 전혀 어렵지 않다.

관찰을 해보자면 멀어질수록 진폭이 작아짐을 알 수 있다. 원점으로부터의 거리가 주어지면 그 점에서의 y축으로의 최대 진폭을 구하는 함수를 찾아내야하는데 나는 Ae^{-kx^2}가 매우 적절해 보였다. 정규분포와 같은 곡선인데, 부드럽게 진폭이 줄어드는 것이 적절해 보인다.

그리고 구한 최대 진폭에 맞게 파동을 그리기 위해서 $y(d,t) = \sin(8\pi d - 4\pi t)$를 사용했다.

시간이 고정된다면, 거리에 따라서 파동이 형성된다.
거리가 고정된다면, 시간에 따라서 y축 위에서 진동한다.

실행파일과 최대한 비슷하게 만들기 위해서 상수를 잘 조절해본 결과, $k = 5.55$, $A = 0.3$ 이 적절해 보였다.

다음은 Waving Color Plane에서 Color이다. 매우 간단하게 Fragment Shader에서 Red 값에 $2y$를 더해주고 Blue 값에 $-2y$를 더해줬다.

시간이 많이 남아서 객체화를 더 잘해보자는 생각이 들어서 WavingPlane이라는 클래스를 사용하기 위해 필요한 것을 제외하고는 private 접근 지정자로 바꿨다.

그리고 다른 크기의 Plane 또는 진폭을 가졌을 때도 작동하도록 상수값들을 잘 조정하고 비율도 잘 조정했다.

원래대로하면 16ms 마다 0.016초를 더해주는 것이 맞지만 실행파일과 최대한 비슷하게 만들기 위해서 0.01초만 더해주었다.

내 프로그램이 완성된 것 같아서 Division을 2부터 200까지 늘렸다 줄였다 하고 있는데 갑자기 깜박깜박 하더니 비정상적으로 종료되었다.

OpenGL 공식 문서에 따르면OpenGL은 운영체제에 이미 들어있다고 한다. 여기서 GLEW는 도대체 왜 다운 받았는지 의문이 들었다. GLEW 공식 홈페이지에 가서 찾아본 결과, 수업시간에 배운 내용이랑 연결되어서 컨셉의 이해가 조금 더 선명해진 것 같다. 환경마다 다른 함수 이름들을 잘 매칭 시켜주는 것 같다.

궁금해서 내 기기안에 있는 OpenGL32.lib 파일을 찾아봤는데 존재했지만 딱히 할 수 있는 것은 없어 보였다.

문제가 무엇인지 생각해 보았는데, 아무래도 내가 짠 GLSL 코드나 OpenGL 관련 함수를 쓴 부분을 제외하고는 나의 로직에 문제가 없을거라는 확신이 들었다. 따라서 문제는 내가 모르는 함수들을 썼을 때 정확히 어떤 일이 일어나는지 모른다는 것이다. 내가 예상한 원인은 GPU에 계속 데이터를 보내는데 GPU 메모리에도 한계가 있을 것 같았고 Division을 바꿀 때마다 계속 메모리를 할당만하고 없애지는 않는 것 같았다. 그래서 OpenGL을 구현한 오픈소스 코드들을 살펴보았다. 문제는 glGenBuffers에 있는 것 같았다. Mesa 3D 구현

무언가 이름에 Gen이 들어가는데 생성을 계속하면 문제가 있을 것 같았다. 이를 해결하기 위해서 다음 코드는 생성자에만 존재한다.

```
glGenVertexArrays(1, &vao);
glGenBuffers(1, &vbo);
```

그릴때마다 다음을 한다.

```
glBindVertexArray(vao);
glUseProgram(prog);
```

이것이 필요한 이유는 만약 내가 WavingPlane 말고 다른 객체들을 함께 그린다면 그에 따른 그려야되는 정보들과 Shader 파일들이 다르기 때문이다. 실제로 Plane을 여러개 놓고 실험해본 결과 잘 작동하였다.