**< 컴퓨터 그래픽스>**

# PROJECT 최종 보고서



과목명 : 컴퓨터 그래픽스

교수명 : 송 인 식 교수님

학 과 : 소프트웨어학과

이 름 : 박서윤(32181618)

EMB0000524c3bca

**목차**

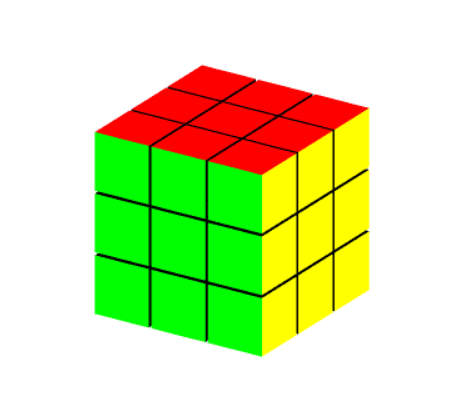
1 요약 ……………………………………………………2p

2프로젝트 배경 2p

3 최종 구현 기능 3p

4 제안서와 달라진 점 3p

1. 코드설명 4p
2. 실행화면 2p
3. 고찰 2p
4. 소스코드 2p
   1. **요약**

3d modeling를 통한 3\*3\*3의 rubik's cube 를 만든다. 생성된 rubik's cube는 각 면에 각각의 다른 물질로 mesh한다. 큐브는 마우스를 통해 360도 회전이 가능하다.

* 1. **프로젝트 배경**

WEBGL 수업을 들으면서 rotation 이나 3d modeling 또는 texture 와 같은 내용을 배웠다. 이런 이론과 예제를 복합적으로 활용할 수 있는 프로젝트를 생각해 보았고 3D rubik's cube를 이용해서 큐브를 맞출 수 있는 생각을 고안하게 되었다. 3\*3\*3 rubik's cube 를 사용하게 되면 큐브를 돌리기 위해서 rotation을 사용하고 정육면체를 통한 3d modeling 각각의 색깔을 입히기 위해서 texture를 사용해야한다. 그동안 수업을 통해 배운 실습코드중에 정육면체가 버튼을 누르면 x축 y축 z축으로 돌아가는 코드를 활용할 수 있고 새로운 코드를 추가해 더 좋은 결과를 낼 수 있다고 생각한다.

* 1. **최종 구현 기능**

1. **전체적인 큐브 회전 (마우스)**

기본적으로 주어지는 큐브는 3\*3\*3 정육면체 모양의 큐브이다. 사용자는 마우스를 이용하여(드래그 하여) 루빅스 큐브를 살펴 볼 수 있다.

* 1. **제안서와 달라진 점**

제안서에서는 실제로 큐브를 맞추는 것처럼 큐브의 단층 하나하나가 돌아가게끔 기능을 만들 예정이었으나, 시간부족으로 인하여 기능을 완성시키지 못하였다.

* 1. **코드 설명**

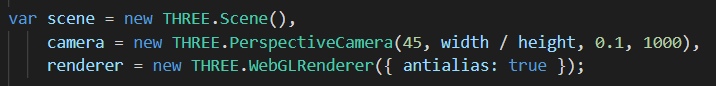
이용한 라이브러리

1. three.js : 웹지엘의 기능을 아주 쉽게 사용할 수 있는 자바스크립트 API를 제공한다.
2. jquery.js : 엘리먼트를 선택하고 그것들을 효율적으로 제어할 수 있는 다양한 수단을 제공하는 자바스크립트 라이브러리이다.
3. orbitcontrols.js : 마우스를 이용하여 카메라를 컨트롤 하기 위해서 넣은 라이브러리이다.
4. threex.domevents : 3D 장면 내에서 dom 이벤트를 제공하는 three.js 확장버전이다. Click, dbclick, mouseup, mousedown, mouseover, mouse out 등의 이벤트를 사용할 수 있다. 큐브의 개별 단층을 회전하기 위해서 추가한 라이브러리이다.

전체적으로 Rubik이란 함수를 만들었다.



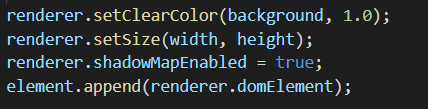
인자를 3개 받는다. element는 사용할DOM요소를 포함하는 jQuery객체이다. dimensions의 default값는 3이다. Background는 배경색이다. 0x444444(어두운 회색)으로 지정했다.



먼저 scene과 camera, renderer를 정의했다. scene객체는 렌더링할 모든 객체와 사용할 모든 광원을 저장하는 데 쓰이는 컨테이너다. THREE.Scene객체 없이는 Three.js는 아무것도 렌더링할 수 없다. camera객체는 장면을 렌더링 했을 때, 어떻게 보여질 것인지를 정의한다. 첫번째 인자로는 시야를 뜻한다. 45도에서 가장 자연스럽게 보인다. 다음인자는 카메라의 화면 종황비를 정의한다. 보통 rederer의 너비를 높이로 나눈 값이 들어간다. 마지막 숫자 두개(0.1,1000) 은 오브젝트가 카메라에 얼마나 가까워지고 멀어져서 그려질 수 있는지를 정의한다. Renderer 객체는 scene 객체가 camera객체의 각도에 따라 브라우저에서 어떻게 보이는지 산출하는 역할을 한다.



윈도우 창의 너비와 높이를 변수에 저장했다.



renderer의 여러함수들로 배경색과 사이즈를 설정할 수 있다.



카메라의 위치 조정.

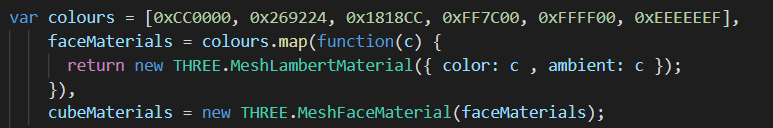


다음은 조명 설정이다. THREE.AmbientLight는 기본 광원 장면에 있는 객체의 현재 색상에 빛의 색상이 더해진다.

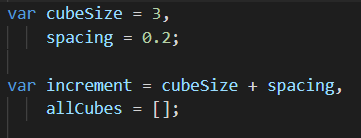


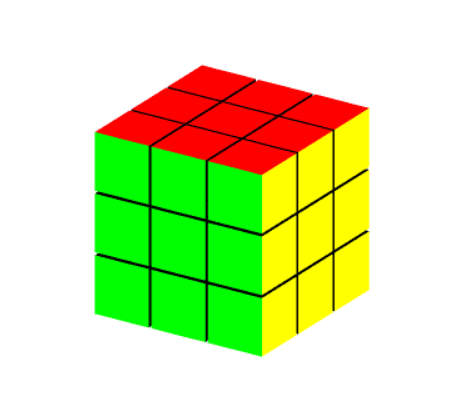
이 부분이 마우스를 이용하여 전체적인 큐브를 살펴볼 수 있게 하는 코드이다.

여기까지가 기본 셋팅이라면 이제는 오브젝트(큐브)를 만들어야한다. 먼저 보편적으로 루빅스큐브의 색으로 많이 쓰이는 6가지 색깔을(빨강, 노랑, 주황, 초록, 파랑, 하양) 배열에 넣었다. Three.js 에 대해 공부하다가 물질(material)에 대해 알게 되었다. 물질은 지오메트리의 외부 모양을 정의하는 일종의 스킨과 같다. Three.js는 다양한 물질을 제공하는데 그 중에서 MeshLambertMaterial을 사용했다. 이는 빛을 받아 반짝이지 않는 객체를 생성할 때 사용하는 물질이다.

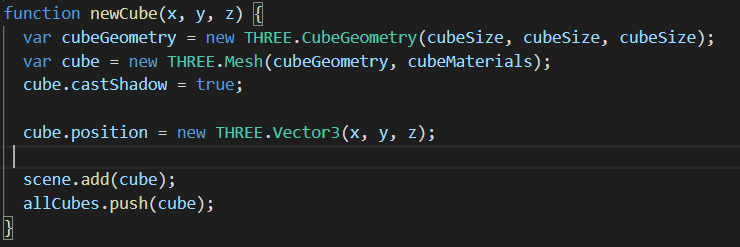


맨 마지막 줄의 THREE.MeshFaceMaterial은 다른 여러 물질들의 컨테이너에 가깝다. 지오메트리의 각 면에 서로 다른 물질을 지정할 수 있게 해준다. 이 점을 착안하여 큐브의 각각의 면에 다른 물질(여기서는 색상)을 지정하였다.



루빅스 큐브는 사실 조그만 정육면체 27개의 집합이라고 할 수도 있다. 

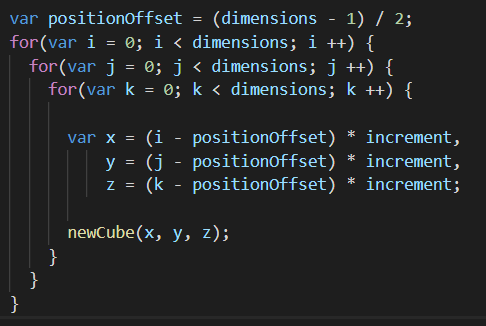
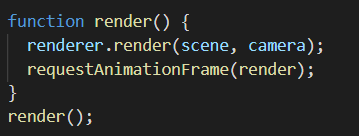
옆의 그림에 보이는 검은 선들을 빈 공간으로 생각하면 더더욱 잘 이해가 간다. 그래서 한 정 육면체의 사이즈를 3으로 잡고 공간(spacing)을 0.2라고 했으며 increment에 둘을 더한 값으로 지정했다.



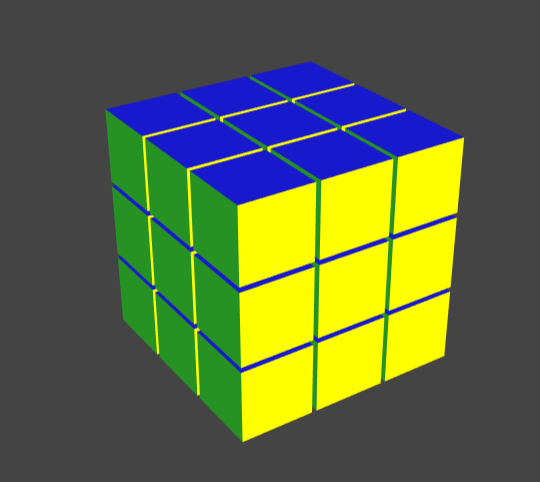
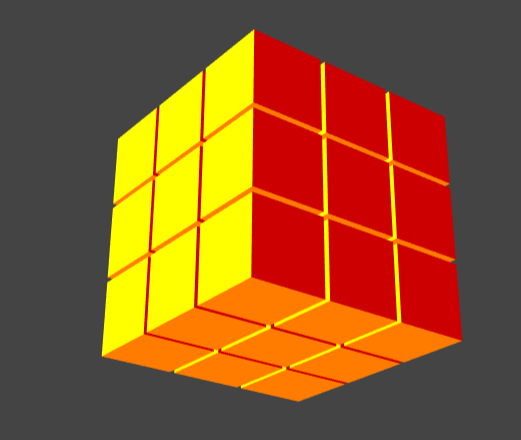
newCube 함수에선 3\*3\*3의 큐브 지오메트리를 만든다.(cubeSize=3 이므로) Mesh함수는 앞서 정의했던 물질과 객체가 어떻게 보이는지(cubeGeometry를 사용했으니 여기서는 당연히 정육면체로 보인다)에 대해 조합해주는 함수이다. position으로 위치를 잡아주고 scene에 더해준다. (참고로 CubeGeometry는 BoxGeometry와 거의 동일하다. 큐브를 만드는 프로젝트이므로 전자로 사용)

여기까지의 작업은 그저 정육면체를 만든것에 불과하다. 아까 말했듯이 가로, 세로, 높이 각 3개씩 총 27개의 정육면체가 일정한 공간을 두고 나열되어야 한다.

이 부분에서 많은 고민과 검색을 하다가, 단순하면서도 코드가 깔끔한(효율은 별로일지라도) 방법을 찾았다. 삼중 포문을 돌리는 것이다.

그냥 복잡할것 없이 하나 하나 줄 맞춰 쌓아가는 원리이다. 다른 방법도 많았지만 개인적으로 제일 이해하기 쉬워서 이 방식을 선택했다.

render함수를 이용하여 렌더링한다.

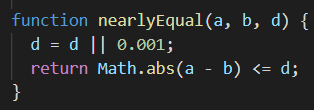
* 1. **실행 화면**

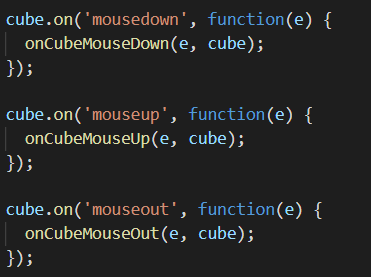
실행 첫 화면

마우스로 움직인 후의 화면

* 1. **고찰**

이런 개인적인 프로젝트는 한 적이 처음이라 제안서 쓸 때부터 고민이 정말 많았다. 특히 웹에는 전혀 자신이 없어서 처음 주제 정할 때는 그냥 엄청 쉬운걸로 할지도 많이 고민했다. 하지만 하지 못하더라도 해보고 싶은걸 해봐야 더 도움이 될 것 같아서 루빅스 큐브를 선택했다. 수업에서 three.js를 배우기 전부터 three.js를 알게되었고 이 것을 이용하여 코드를 만들고자 하였다. 처음엔 three.js 자체를 쓰는 것 부터가 너무 생소해서 인터넷과 책을 참고하여 기초 강의도 많이 들었다. 처음엔 3차원 큐브를 어떻게 구현할지 그 방식에 대해 많이 고민을 했다. 제일 처음 떠오른것은 그냥 모든 좌표를 배열에 넣은 후에 큐브를 돌릴때 마다 그 축을 바꿔주는 배열을 만드는 방법이었는데 시도 결과 너무 복잡하고 배열의 순서문제도 있고 무엇보다 코드가 너무 지저분해서 그 아이디어는 폐기처분 하였다. 그 다음 방법은 검색 해보다 3d 모델링을 통한 축을 만드는 데에서 착안한 아이디어인데, 3Dbuilder로 각 큐브의 회전 부품을 만든 후 그 파일을 가져와서 회전하는데 쓰는 방법이다. 하지만 이 방법은 첫째로, 부품이 사이즈까지 다 정교하게 맞아야하고 둘째, obj파일을 참조할 때 그 파일이 웹 상에 로드가 되어야 하는데 이 문제에 있어선 과제를 통하여 많은 시행착오를 겪었기에 이 방법도 포기해야했다. 방법론을 생각하는데에 너무 많은 시간을 뺏긴거 같아서 아쉽다. 일주일 넘게 고민하다 불현듯 그냥 단순하게 정육면체 27개를 위치만 맞춰서 쌓아보자 라는 생각이 이번 코드의 착안점이 되었다. 원래 의도했던 바는, 큐브가 층마다 어느 방향이든 다 돌아가게(실제 루빅스 큐브처럼)까지 구현하는것이었다. 그래서 원래는 코드가 엄청 길고 복잡했다. 아이디어만 적어보자면, 먼저 마우스로 지금 내가 어느 면을 돌릴것인지 판별해주는 함수를 만들고(마우스 커서의 좌표를 계산하여 그 절댓값으로 어느 면인지 판단)

그 값을 이용하여 마우스로 큐브를 돌린다. 이 때 마우스 이벤트를 위하여 jquery라이브러리를 사용하려 했다. 해당 라이브러리도 이번에 처음 알게 되었지만 널리 쓰이고 유명한 라이브러리 인 것 같았다. 사용하고 싶었던 마우스 이벤트는 mousedown(마우스 버튼을 누를 때 발생), mouseup(마우스 버튼을 뗄 때 발생), mousemove(마우스를 움직일 때 발생), mouseout(마우스가 element를 벗어날 때 발생)이다.

이 부분은 원래 newCube안에 있던 코드이다. 최종 코드에는 삭제되어있다.

여기 까지 구현을 했다면 사실 그 후에 몇 개의 기능을 더 추가 할 수도 있었다.(제안서에도 없던 여러 기능들) 좋은 아이디어가 있었는데 구현하지 못해 아쉽다. 원래는 실행취소, 랜덤으로 섞기, 초기화 기능까지 다 구현하고 싶었다. 사실 조금 더 시간이 있었으면 했을 수도 있었는데, 중간에 제안서도 한 번 바꾼데다 아이디어 구상과, 그리고 루빅스 큐브 모양으로 만들어 내는 과정에서 시간을 너무 뺏긴것이 패착요인인것 같다. 실행취소와 초기화에 대한 아이디어는 간단히 얘기 하자면, 모든 움직임에 대한 것들을 함수로 만든 후에, 그 움직인 좌표를 스택에 쌓고 undo를 하게 되면 top을 하나 제거 후 그 다음 좌표로 움직이게끔 하려고 했다. 스택은 LIFO구조라는 것에서 착안했다. 초기화도 마찬가지로 초기화 버튼을 누르면 스택을 하나 삭제 후 move, 다시 삭제 후 move 이런식으로 스택에 아무것도 없을 때까지 실행하면 가능 할 것 같다. 조금만 더 했다면 정말 완성도 높게 나올 수 있었을 것 같은데 많이 아쉽다. 이번 프로젝트는 완성 후 깃에 코드를 첨부해야한다. 깃에 조금 더 익숙했더라면 깃에서 작업을 했을텐데 아직 너무 생소해서 모든 작업은 vscode로 진행하였고 마지막 결과물만 업로드했다. 이 업로드 과정도 험난했다. 깃을 처음 써보아서 여기에도 시간이 많이 허비되었다. 여러모로 많은 아쉬움이 남지만 그래도 실행결과가 정말 실제 큐브처럼은 나와서 뿌듯하다. 이 프로젝트는 후에 계속 작업하여 완성시킬예정이다.

* 1. **소스 코드**

**Git Hub 주소 :** <https://github.com/S-YUN>