OBJ Viewer

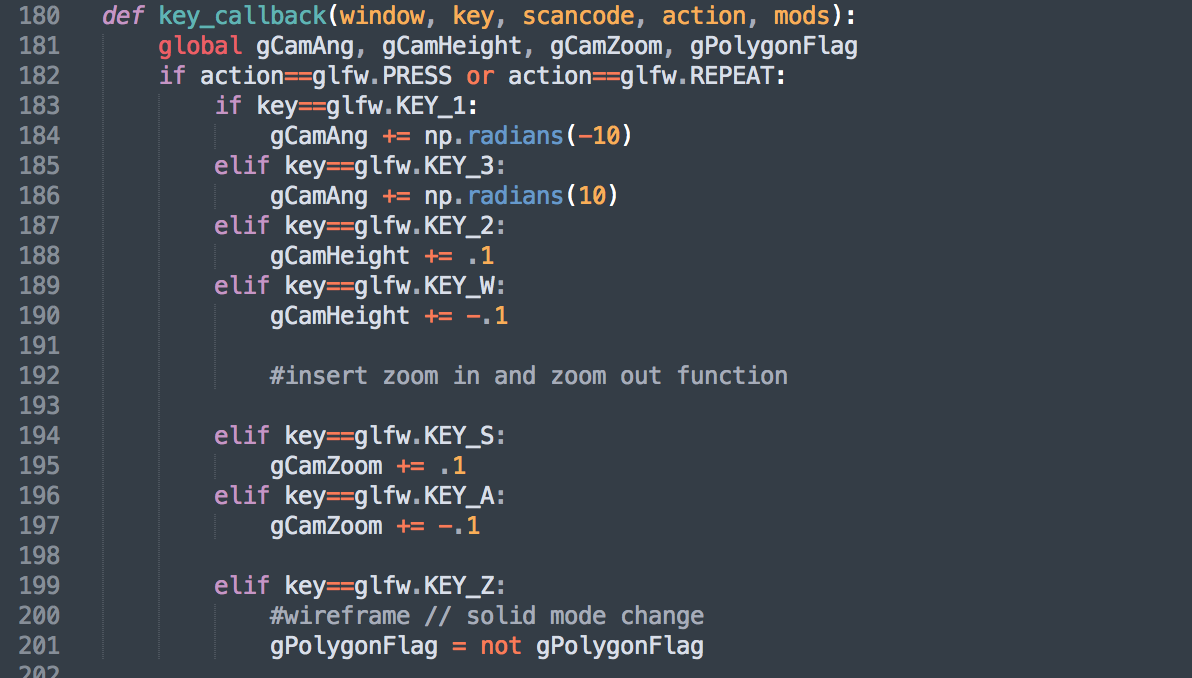
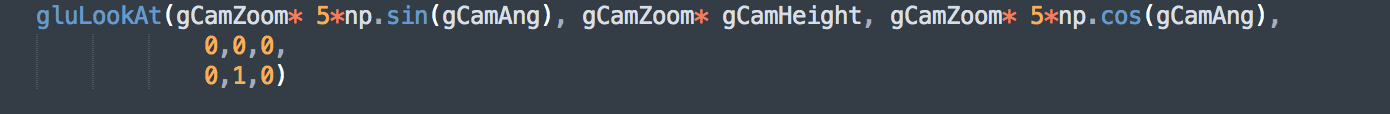
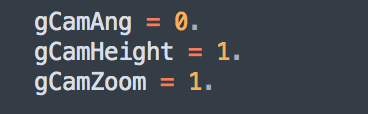
computer graphics assignment3

2016026026 컴퓨터 전공 최솔비

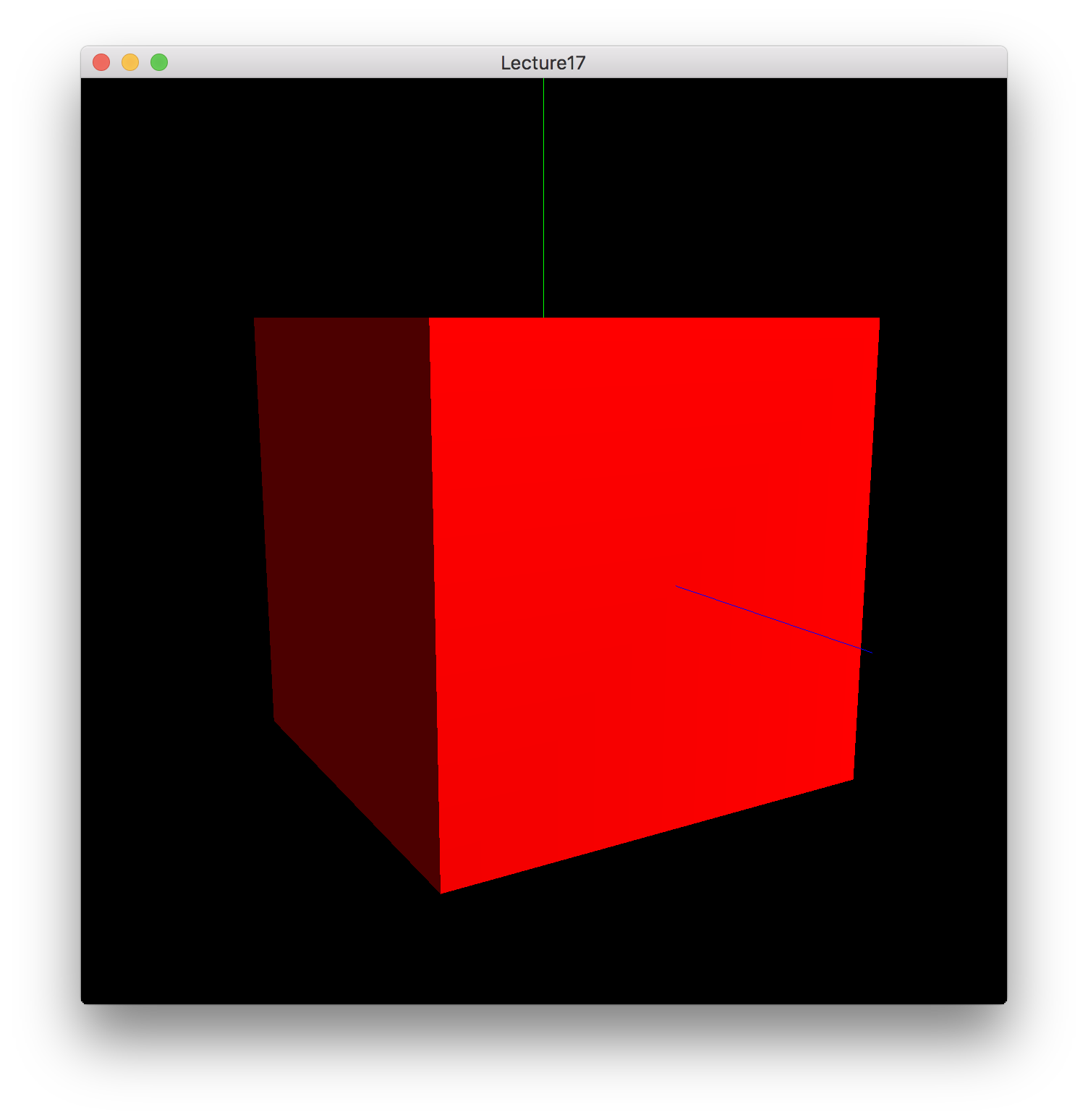
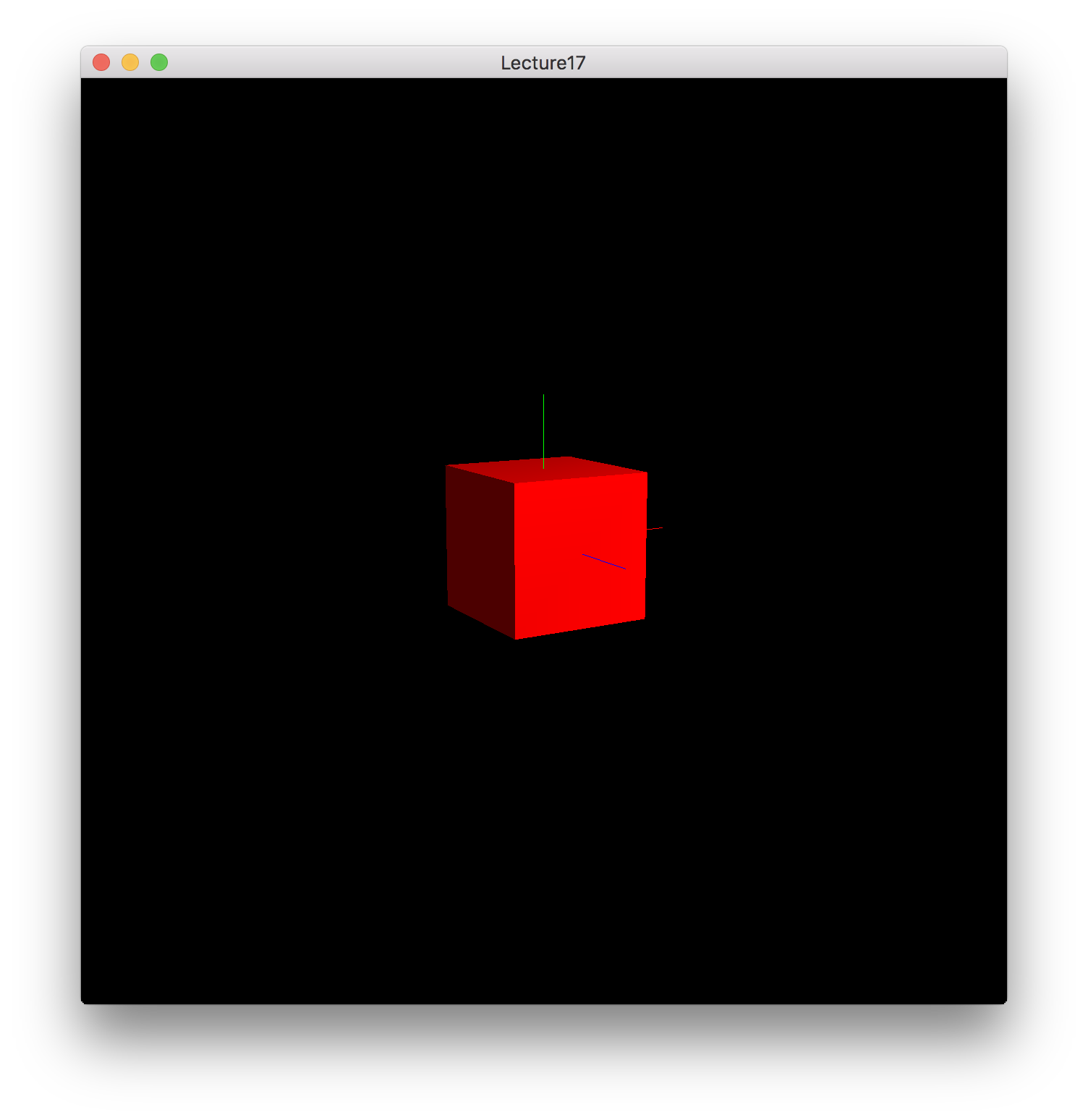
1. 과제 개요

.obj 파일을 프로그램에서 읽어 들여 파싱하여 필요한 정보를 저장합니다. 저장한 정보를 바탕으로 face들을 triangle mesh로 만들어 glDrawArrays 함수를 사용하여 렌더하도록 합니다. 파일은 main.py 한개 이며 shell에서 실행후 .obj 파일을 드래그앤 드롭으로 가져올 수 있습니다.

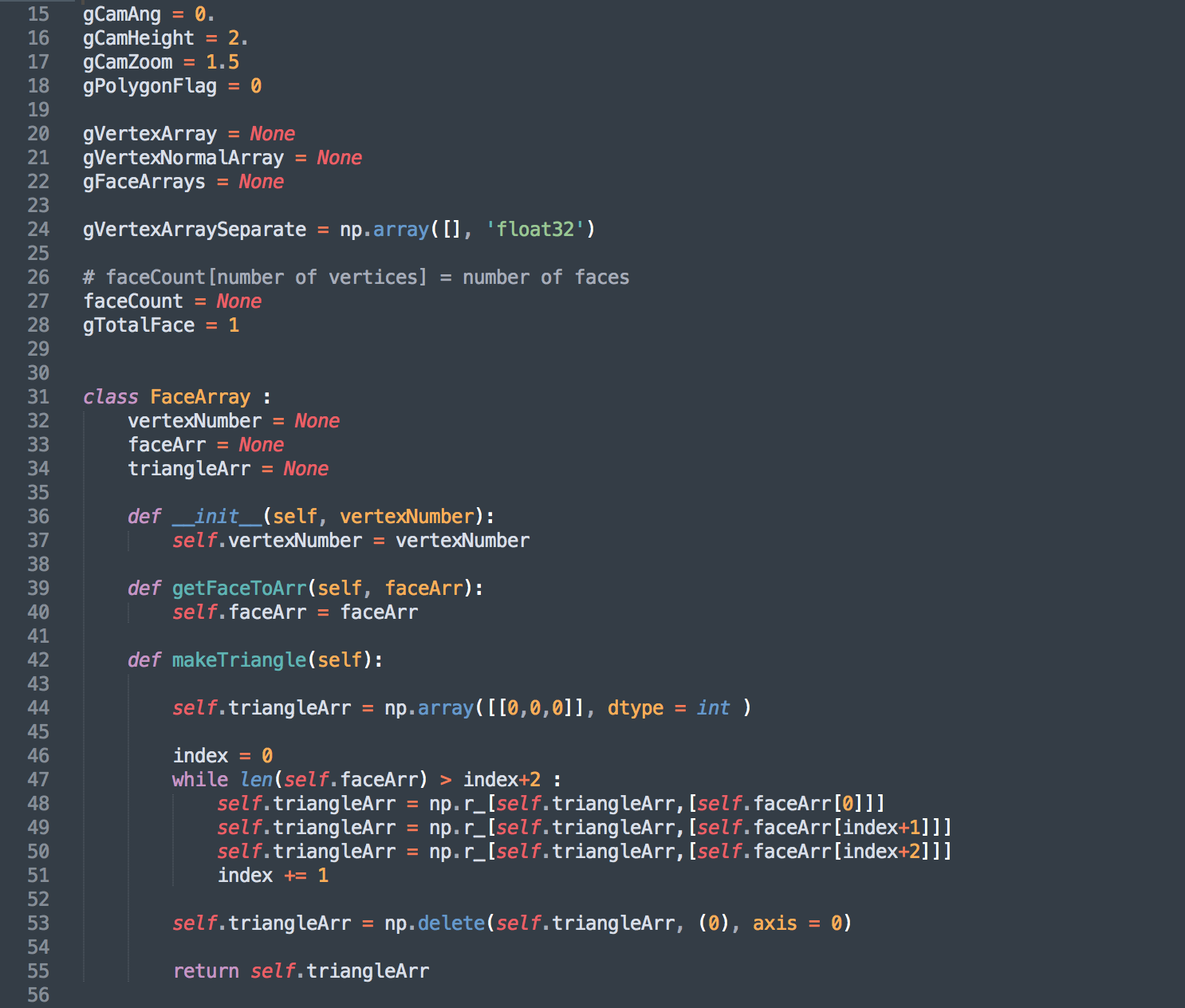
1. 구현한 내용 (코드와 출력화면)
2. Camera control 및 zoom in , out



주어진 코드에 전역 변수 gCamZoom 을 추가하여 이를 key callback function 으로 조작할 수 있게 하였습니다. 조작된 변수 gCamZoom 은 render 함수 내부에서 gluLookAt()의 각각의 인자에 곱해집니다. 이는 대상을 바라보는 카메라의 거리를 움직입니다. 출력 결과는 다음과 같습니다.

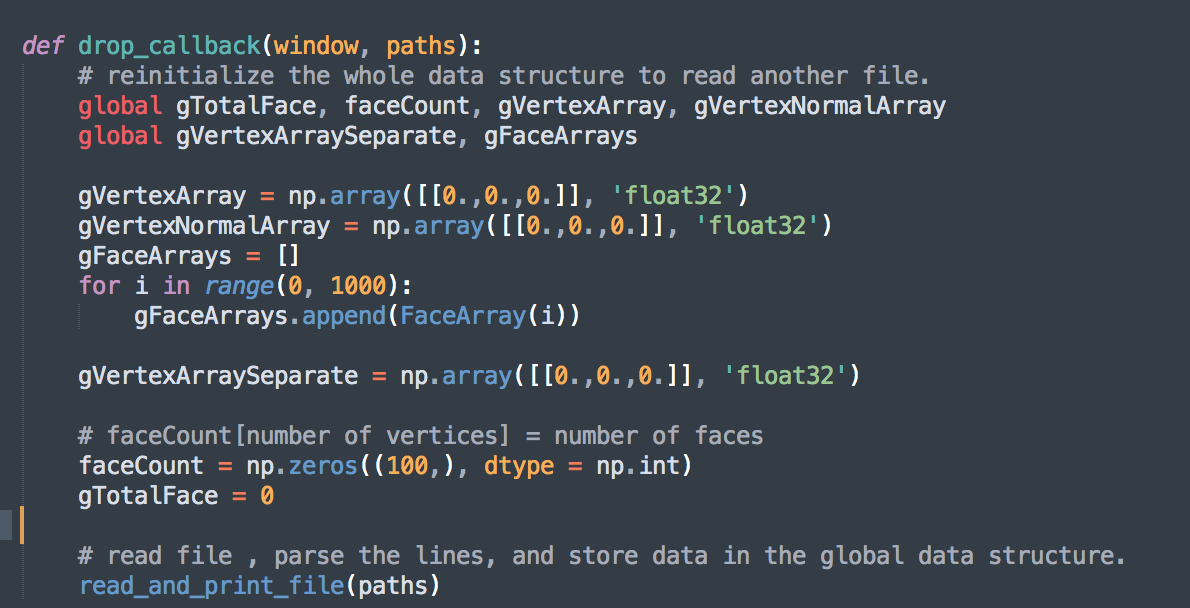


각각 S 와 A 키를 눌러 줌 아웃, 줌 인 한 결과입니다.

1. .obj 파일 로드 후 파싱하기

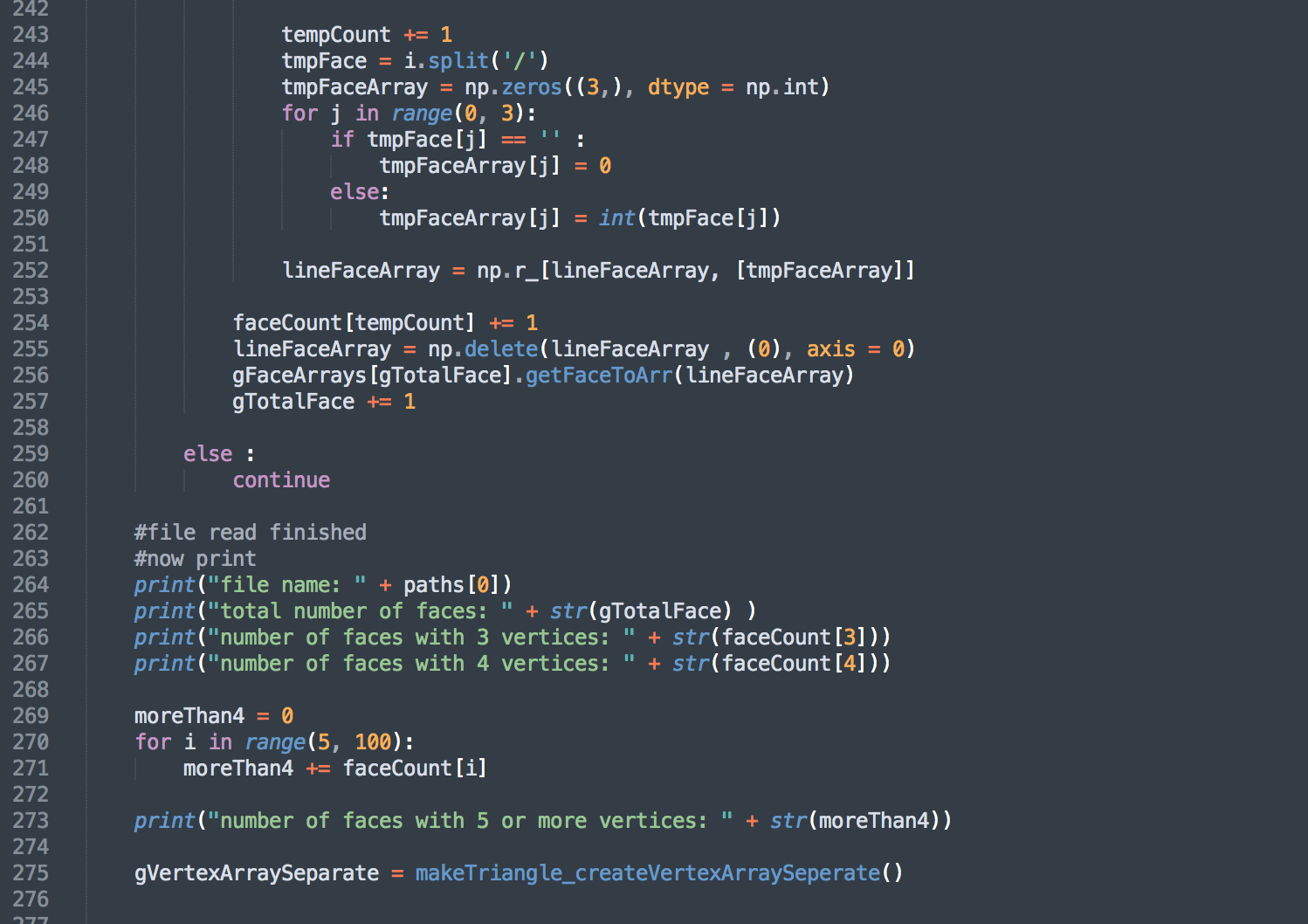
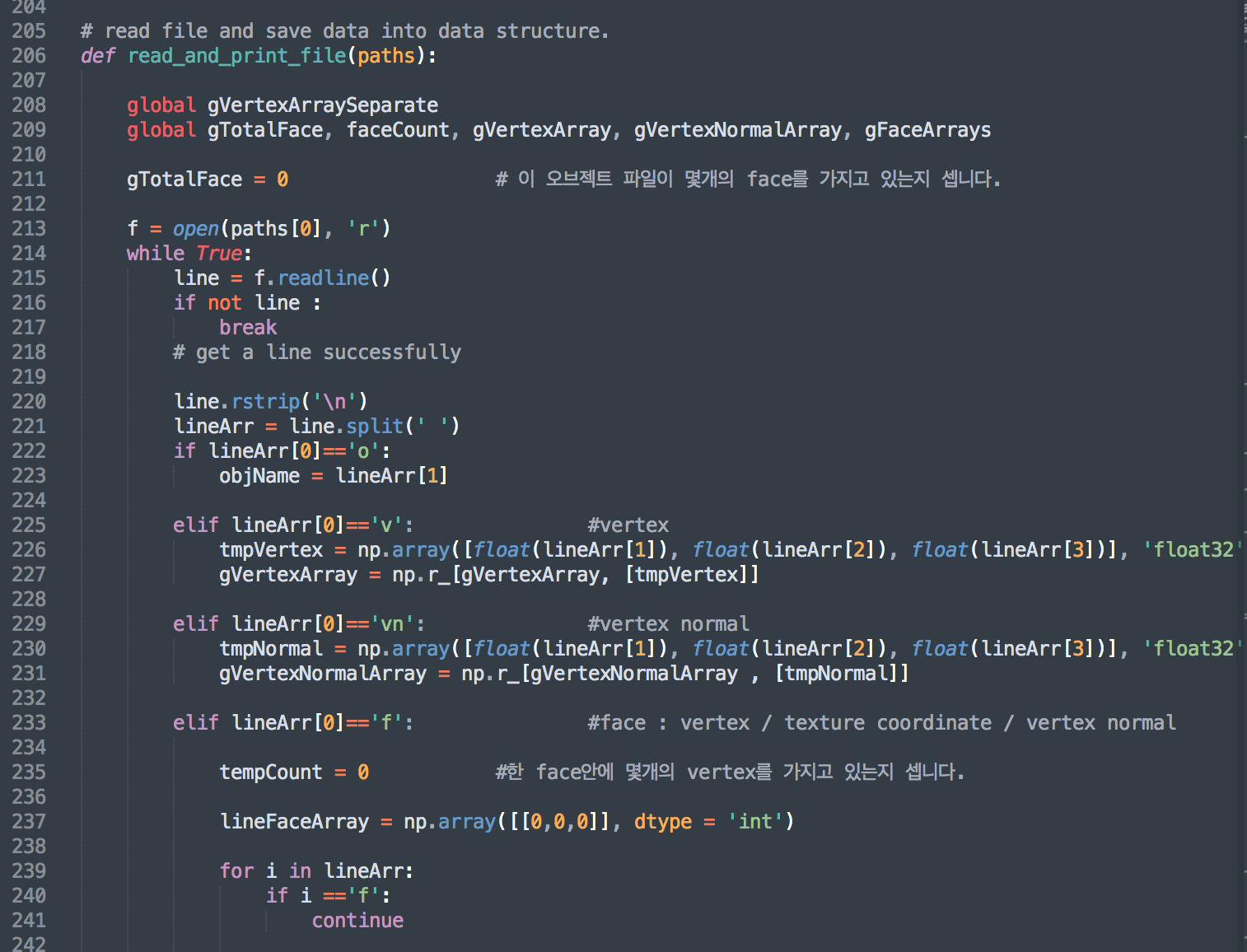
.obj 라는 특정 포맷의 파일 형식을 open() 하여 읽고 파싱합니다. 파싱한 정보는 적절한 자료구조로 관리해주어야 합니다. V로 시작하는 line 에 담긴 vertex 정보를 저장하기 위한 gVertexArray , vn 으로 시작하는 line에 담긴 vertex nomal vecter 정보를 저장하기 위한 gVertexNormalArray, 그리고 f로 시작하는 line에 있는 각각의 face 정보를 저장하기 위한 gFaceArrays 를 글로벌 변수로 선언해주었습니다. 각각의 face는 n개 vertex를 가지는 polygon일 수 있으며 이에 따라 저장하게 되는 정보의 크기가 달라지므로 class FaceArray를 선언하여 해당 face에 대한 data를 관리합니다. gFaceArrays 는 FaceArray 클래스타입의 어레이입니다.

4개 이상의 vertex를 갖는 polygon을 여러개의 삼각형으로 나누기 위해 클래스 안에 mkaeTriangle함수를 정의하였습니다. 해당 함수에서는 index 기준 (0,1,2) (0,2,3) (0,3,4)… 과 같은 형태로 face array를 조작하여 새로운 triangleArr 를 만들어 리턴합니다.



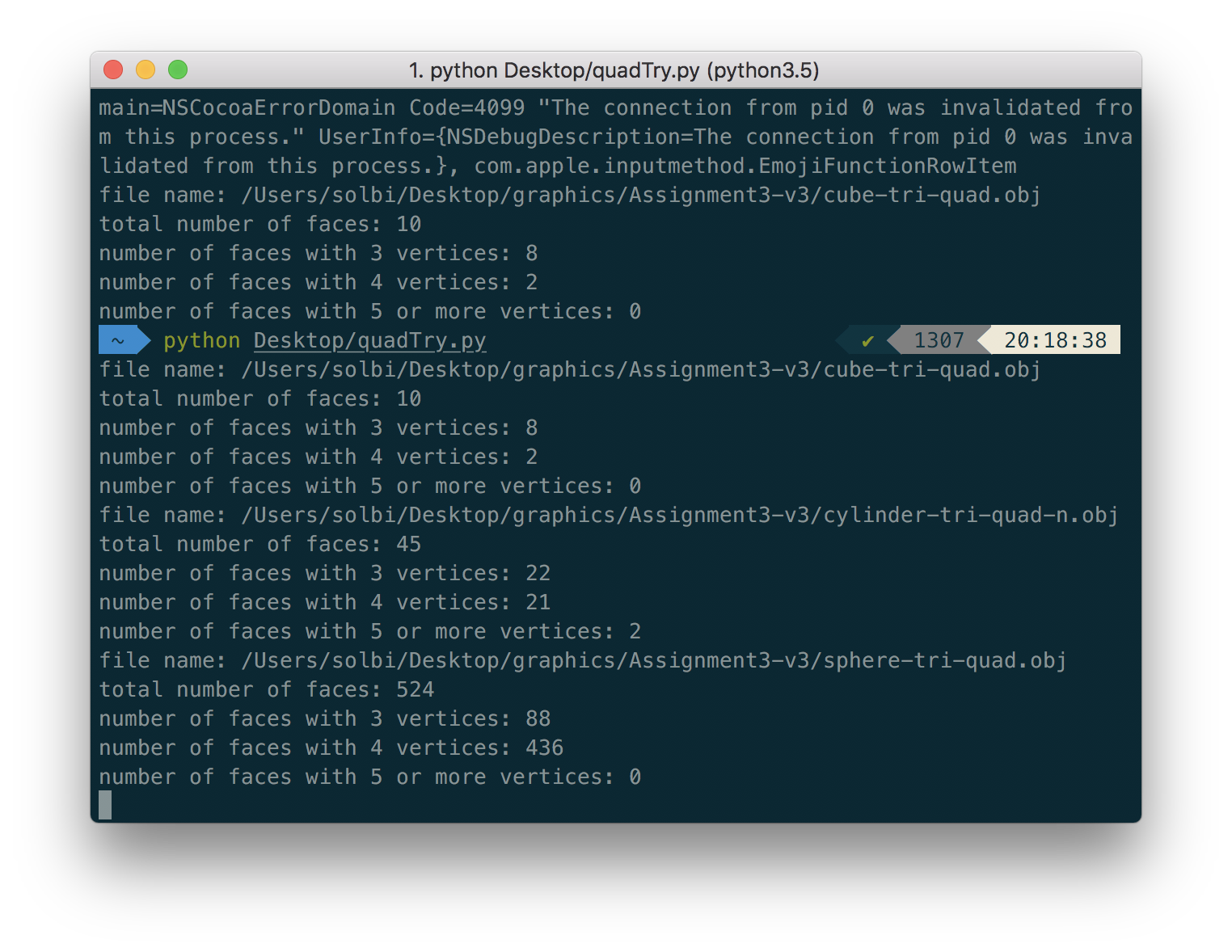
이는 .obj 파일을 드래그앤 드롭 하였을때 수행하게 될 콜백 펑션 drop\_callback()입니다. 해당함수 안에서 자료구조를 위한 전역 변수들을 초기화 합니다. 이는 하나의 오브젝트 파일을 렌더하고 있을 때 다른 오브젝트 파일에 대한 드래그 앤 드롭이 수행되었을 때 자료구조의 정상적이 초기화를 위함입니다. gFaceArray 변수를 초기화하는 과정에서 for문의 range를 수정하여 최대 face개수를 조절할 수 있습니다. 모든 자료구조의 초기화가 끝나면 read\_and\_print\_file() 함수를 호출 합니다. (다음페이지에 코드)

Read\_and\_print\_file 함수에서는 파일을 오픈한 뒤 readline() 함수를 통해 라인 단위로 읽어가며 파싱합니다. 첫글자가 v이면 vertex array 에, vn 이면 vertexNormal array 에 , f 이면 faceArrays (face class로 이루어진 array)에 넣습니다. 동시에 총 face의 개수와 vertex 개수마다의 face 개수를, gTotalFace 와 faceCount[] 에 저장합니다. 모든 라인의 읽기가 끝나면 저장된 count들을 적절히 출력한 후 makeTriangle\_createVertexArraySeperate() 함수를 호출합니다.

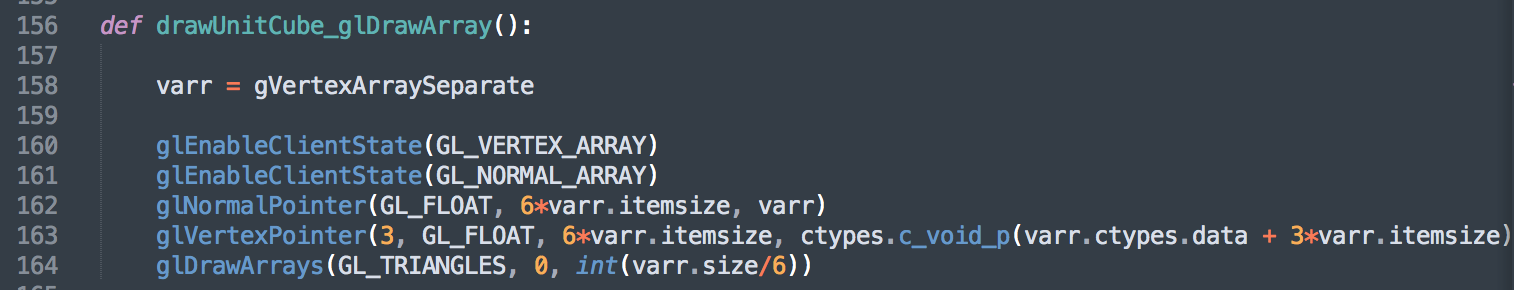




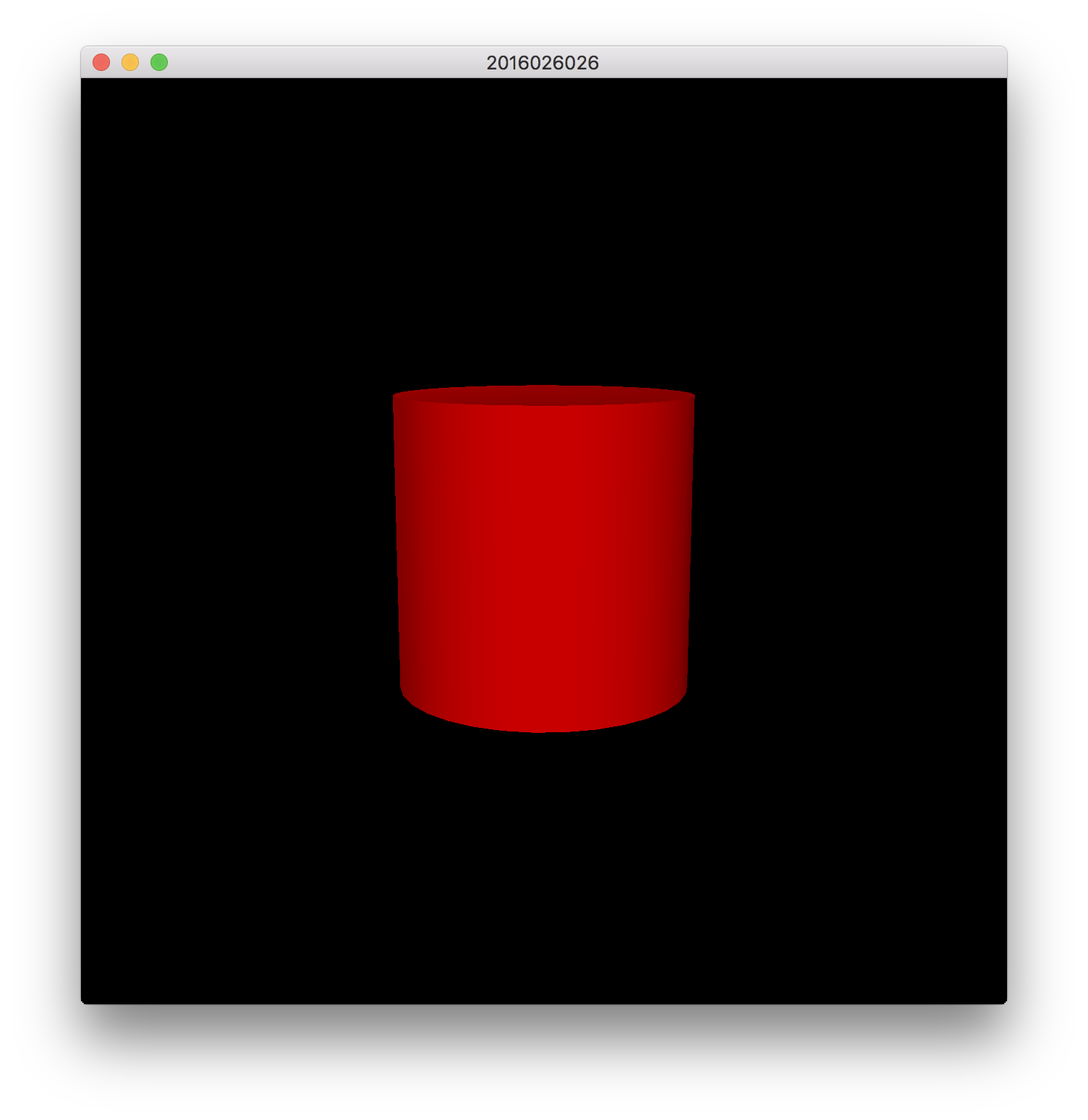
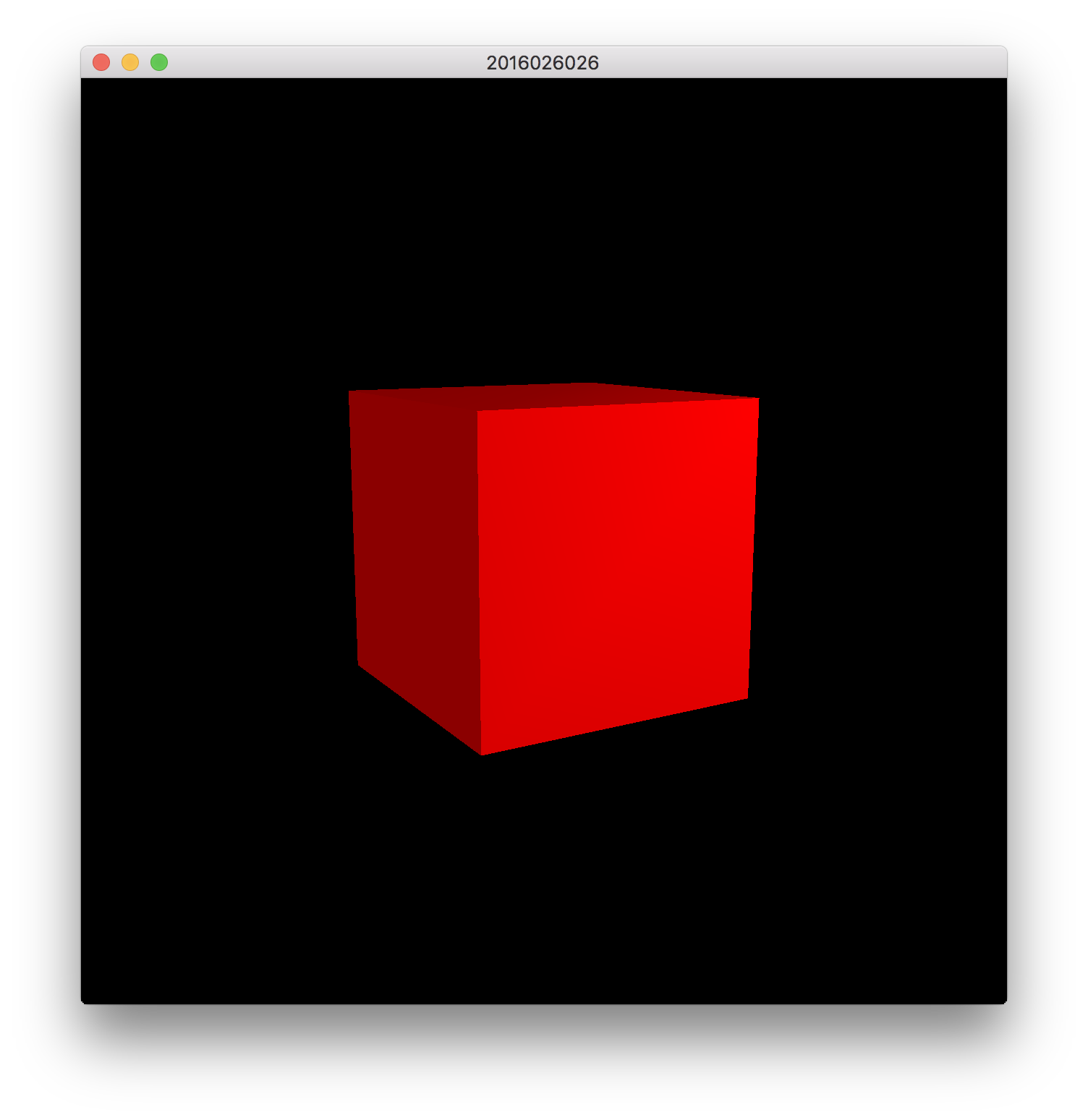
MakeTriangle\_createVertexArraySeperate 함수는 gFaceArrays 배열을 구성하는 FaceArray 클래스 인자들에게 차례대로 클래스 내부 멤버 함수 make Triagle 을 시행합니다. MakeTriangle 은 삼각 메쉬를 리턴하게 되고, 리턴받은 어레이는 모두 하나의 어레이로 합쳐져 함수 makeTriangle\_createVertexArraySeperate 의 최종 리턴값이 됩니다. 이는 전역변수 gVertexArraySeperate 에 저장되어 render 함수에서 glDrawArrays 함수에서 그리게 될 행렬로 넘겨받습니다.

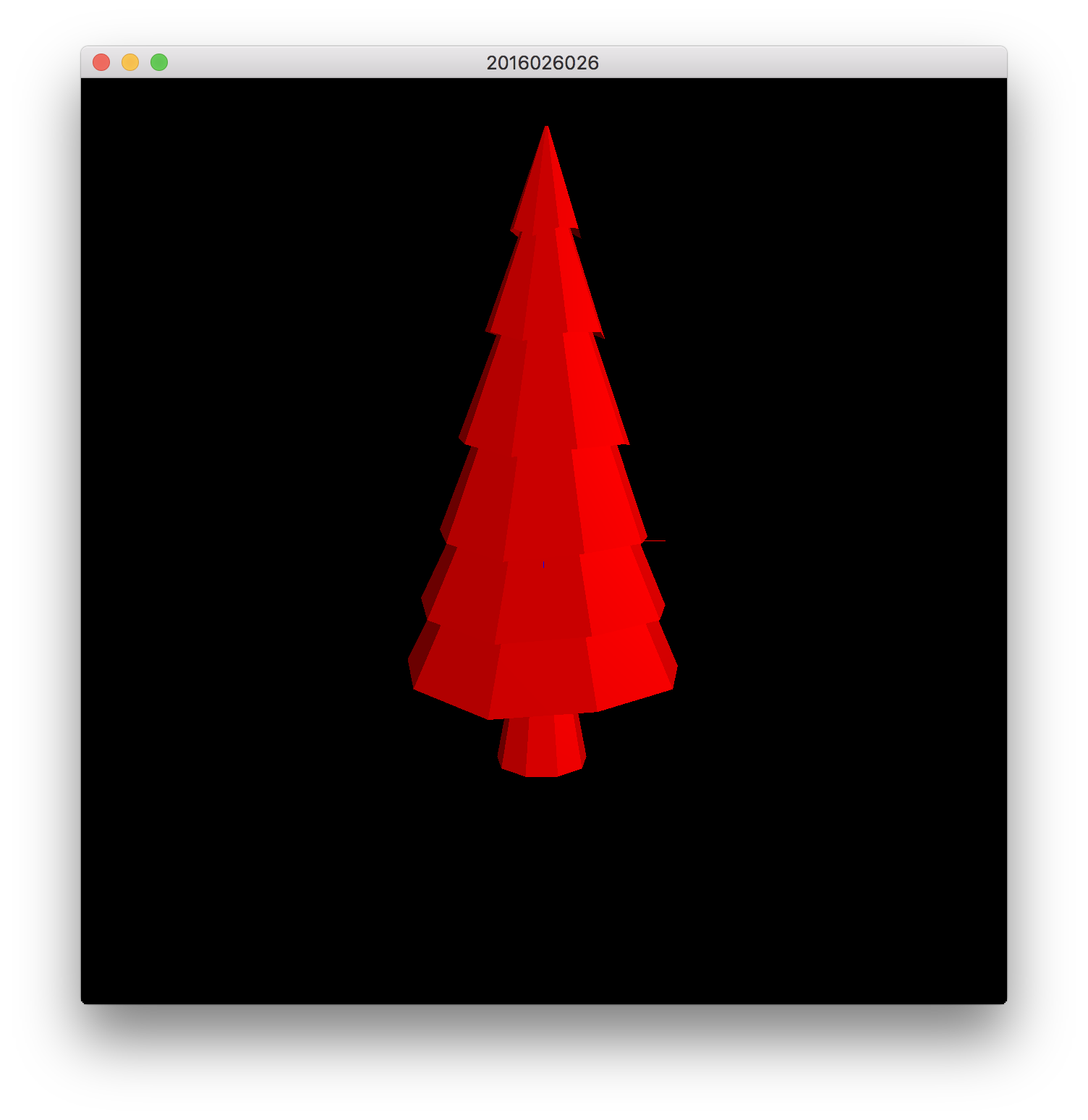
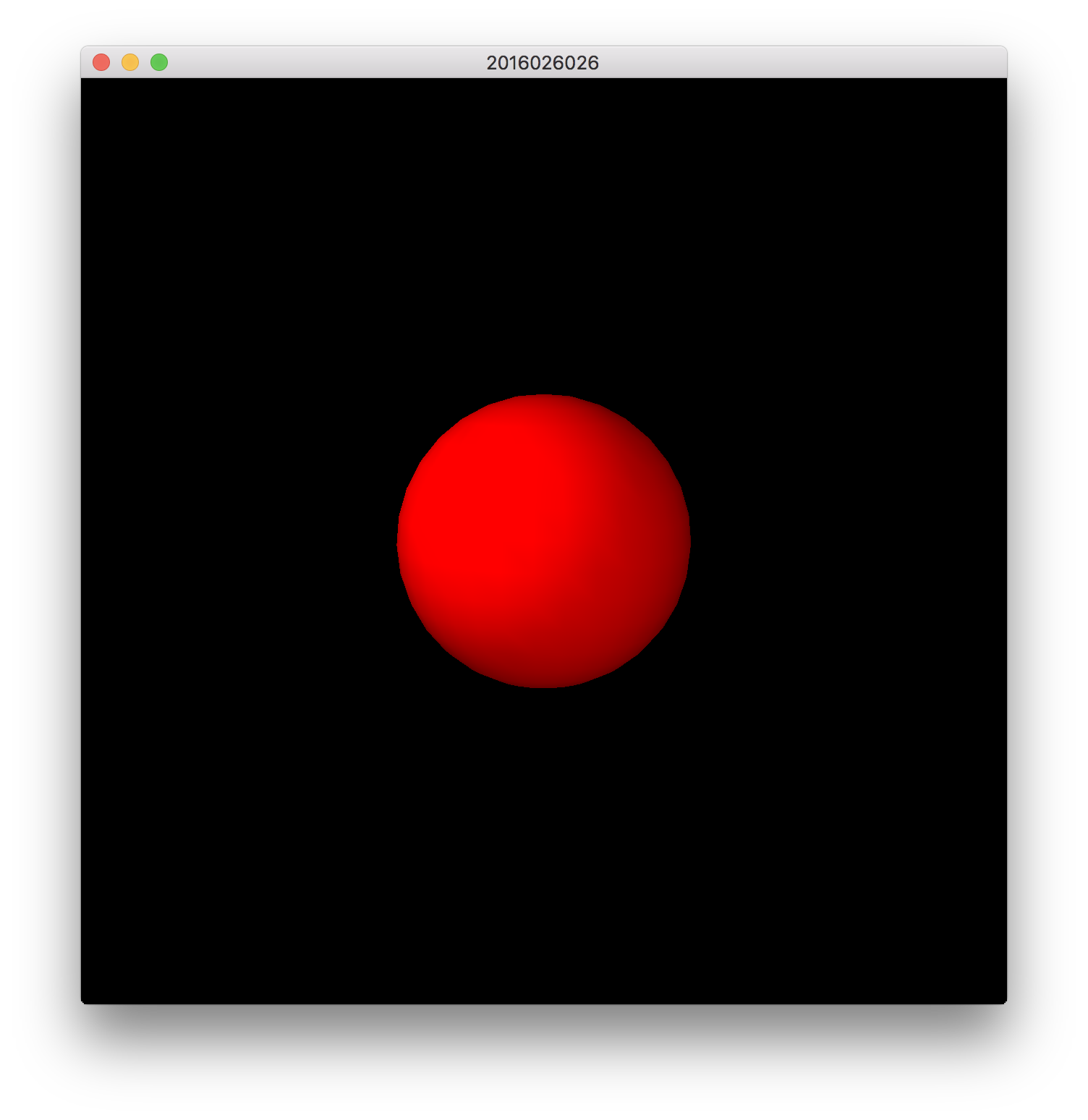


위 화면은 drop callback 함수를 set 한 후 .obj 파일을 드래그 앤 드롭하였을 때 shell에 출력되는 결과값입니다. 파일 이름과 face의 총 개수, face내부 vertex의 개수에 따른 face의 개수를 출력하고 있습니다.



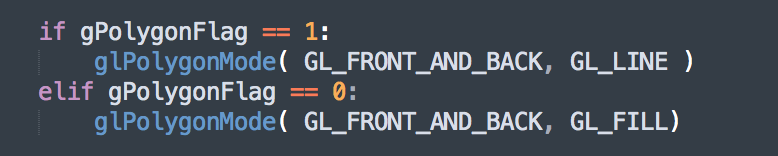
최종적으로 render 함수 내부에서 drawUnitCube\_glDrawArray 함수를 호출하게 되면, 함수 내부에서 전역변수 gVertexArraySeparate 를 상대로 glDrawArrays 를 실행하게 됩니다. gVertexArratSeparate에는 미리 조작된 삼각형의 mesh만 존재하게 됩니다.

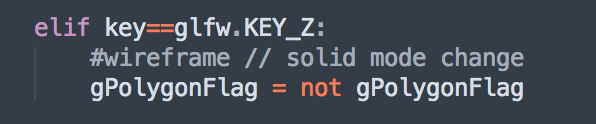




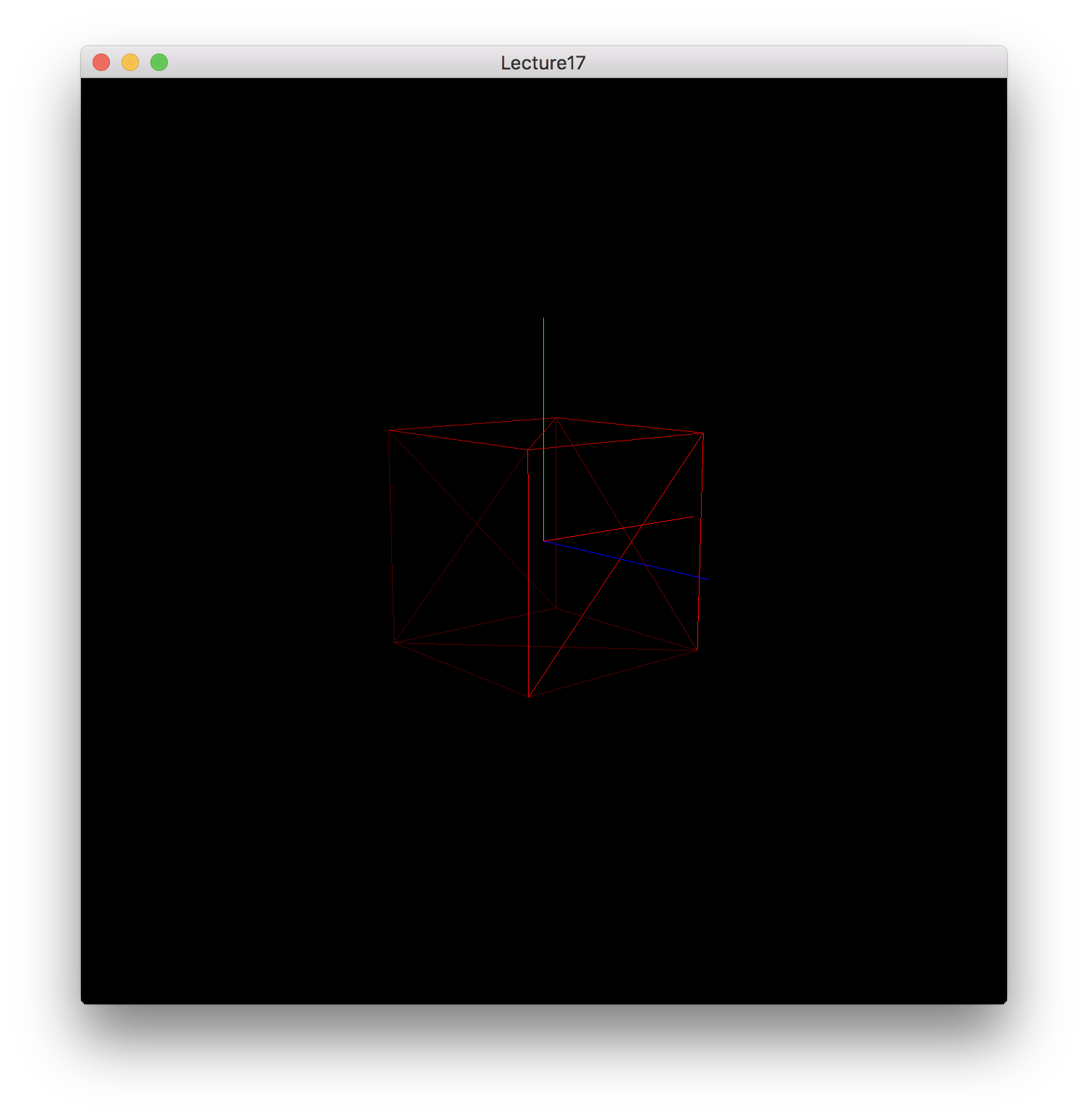
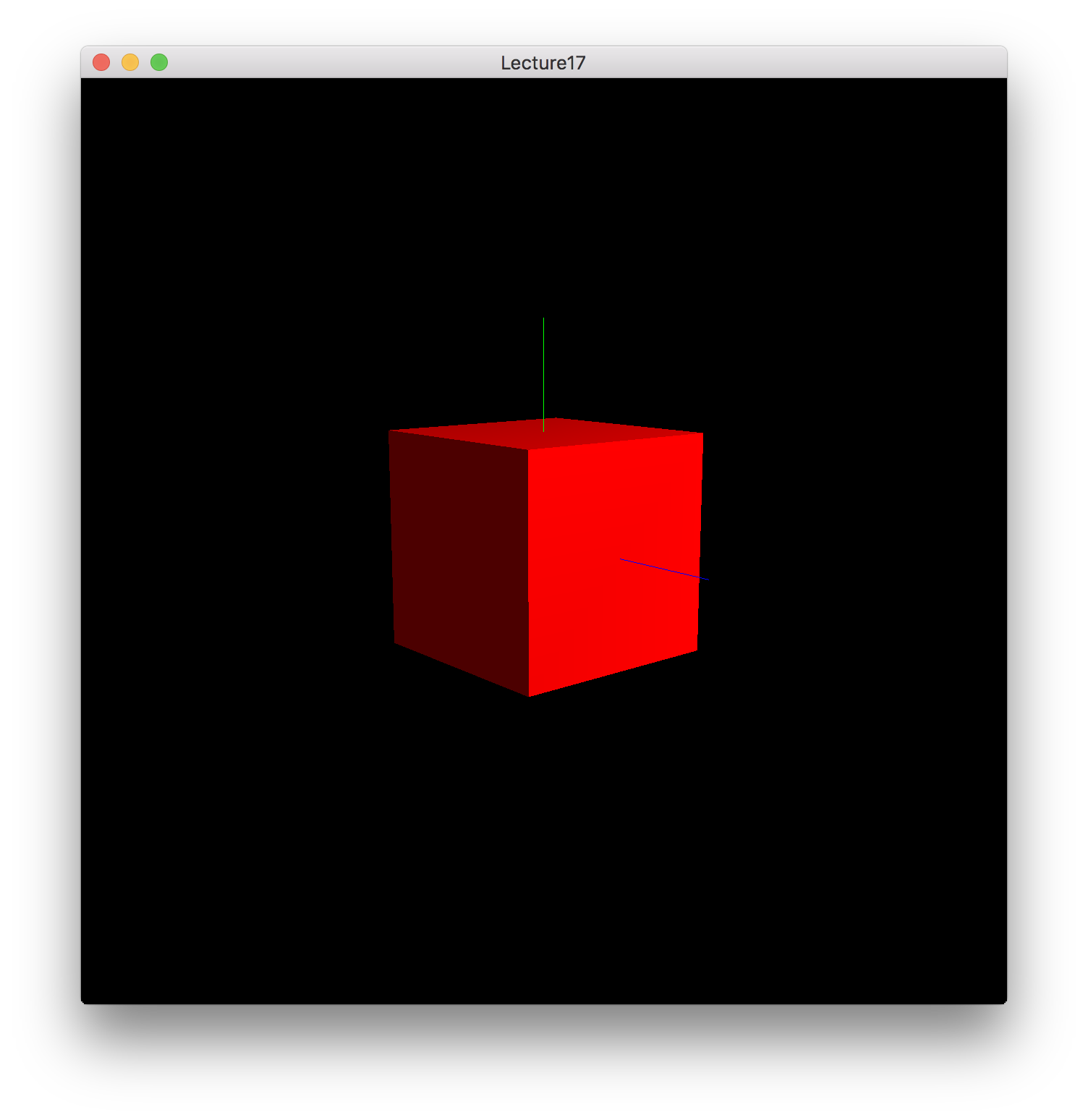
각각 cube-tri-quad.obj , cylinder-tri-quad-n.obj, sphere-tri-quad.obj 와 샘플 오브젝트 파일을 렌더링한 모습입니다. 삼각형이 아닌 face도 정상 처리하는 것을 확인할 수 있습니다.

1. Z키를 눌러서 Wireframe /solid mode 구현하기





Key\_callback 함수에서 전역변수 gPolygonFlag 를 조작하여 프레임 모드와 솔리드 모드로 바꿀수 있도록 합니다. 결과 화면은 다음과 같습니다.



1. Lighting

Render() 함수 내부에서 광원에 대한 설정을 합니다. 현재 광원은 2개 이면 한 개는 고정, 다른 한 개는 회전하고 있습니다. Ambient 와 diffuse 값을 낮춰 2개의 광원에도 너무 밝지 않도록 조정하였습니다. 빛 관련 코드는 다음과 같습니다.

