심화전공실습 (CGL)

HW10_Viewing

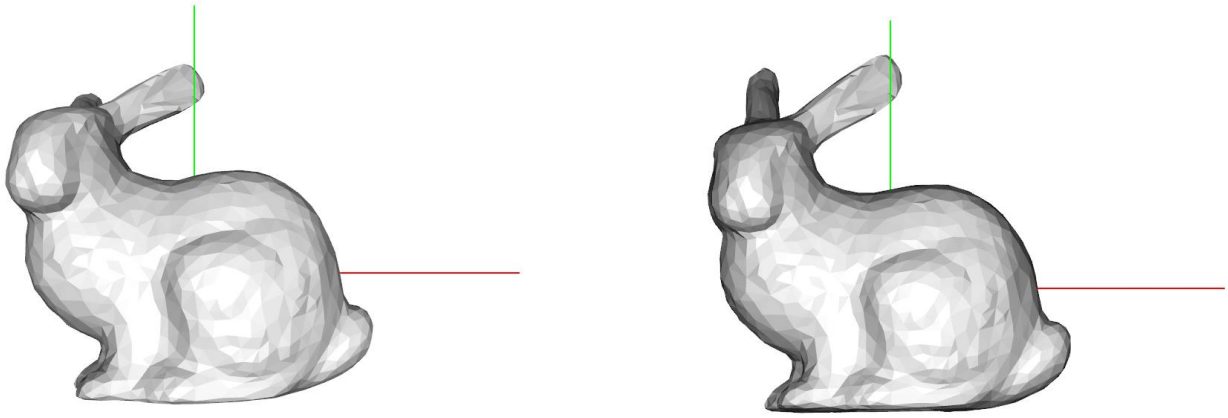
Self-scoring table

	P1	P2	E1	E2	Total
Score	**1**	**1**	**1**	**1**	**4**

2018707068 김경환

KwangWoon University

Practice01 Snapshot, Explanation:



왼쪽 사진은 Perspective projection이고, 오른쪽 사진은 Orthographic projection이다.

Perspective projection의 경우에는 COP가 bunny model의 귀에 비해 아래쪽에 위치해 있기 때문에 model의 귀 부분이 젖혀진 것처럼 보이는 것을 볼 수 있다.

Orthographic projection의 경우에는 projection plane이 Front View face와 평행함으로 Perspective projection과 달리 원근감 없이 bunny model이 표현되는 것을 볼 수 있다.

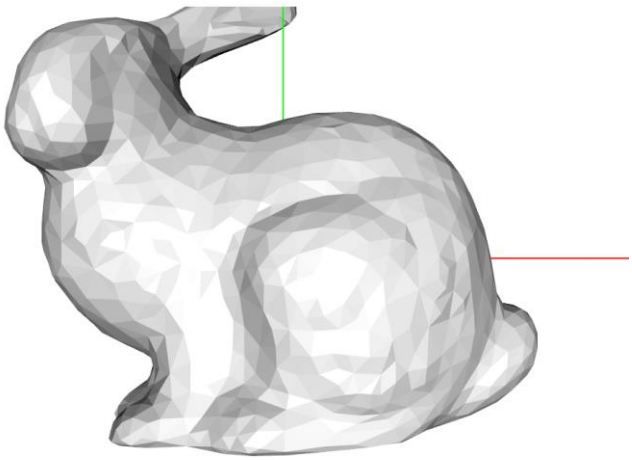
Practice02 Snapshot, Explanation:

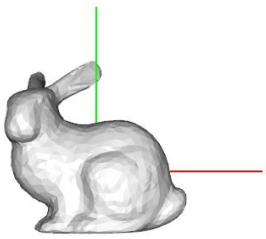
왼쪽 사진은 Perspective projection에서 오른쪽으로 camera 이동을 한 모습이다. 그리하여 파란색의 z 축이 왼쪽으로 뻗어나가듯이 보이는 것을 확인할 수 있다.

오른쪽 사진은 Perspective projection에서 왼쪽으로 camera 이동을 한 모습이다. 그리하여 파란색의 z 축이 오른쪽으로 뻗어나가듯이 보이는 것을 확인할 수 있다.

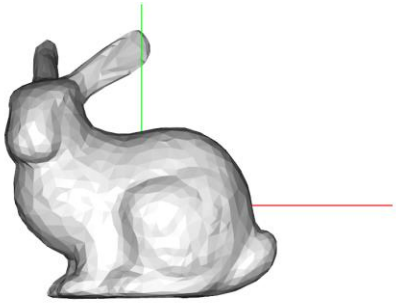
또한, 이는 Camera의 eye position과 at position 좌표에 right direction의 좌표값을 적절히 더해 구현할 수 있다. Ex) LEFT : C.e -= 0.1f * C.r; C.a -= 0.1f * C.r;, RIGHT : C.e += 0.1f * C.r; C.a += 0.1f * C.r;

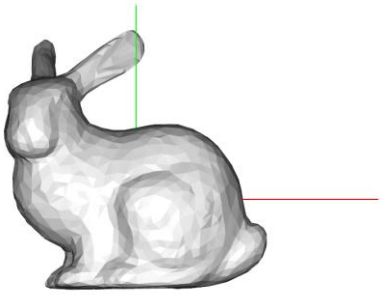
그리고 Perspective projection이기 때문에 camera의 이동에 따라 bunny model의 모양이 바뀌는 것을 볼 수 있다.



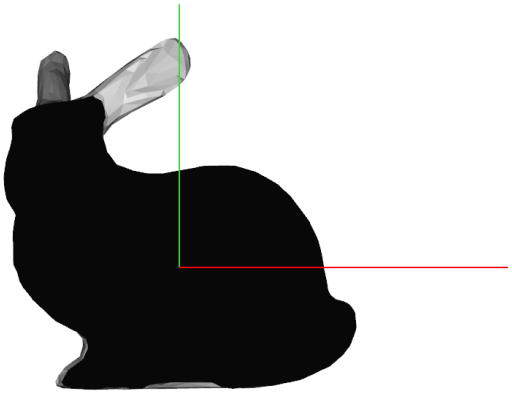


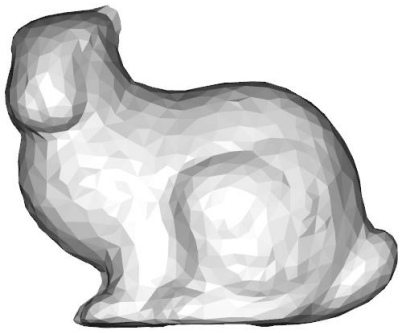
왼쪽 사진은 Perspective projection에서 앞쪽으로, 오른쪽 사진은 뒤쪽으로 camera를 이동한 모습이다. 그리고 이는 Camera의 eye position과 at position 좌표에 forward direction의 좌표값을 적절히 더해 구현할 수 있다. Ex) UP : C.e += 0.1f * C.f; C.a += 0.1f * C.f;, DOWN : C.e -= 0.1f * C.f; C.a -= 0.1f * C.f;
또한, Perspective projection이기 때문에 camera의 이동에 따라 bunny model의 모양이 바뀌는 것을 볼 수 있다.





왼쪽 사진은 Orthographic projection에서 오른쪽으로, 오른쪽 사진은 왼쪽으로 camera를 이동한 모습이다. 그리고 이는 Perspective projection에서 camera를 왼쪽, 오른쪽으로 이동한 방법과 같다.
하지만, Orthographic projection이기 때문에 camera의 이동에 따라 bunny model의 모양이 바뀌지 않고 같은 모습을 보이는 것을 볼 수 있다.



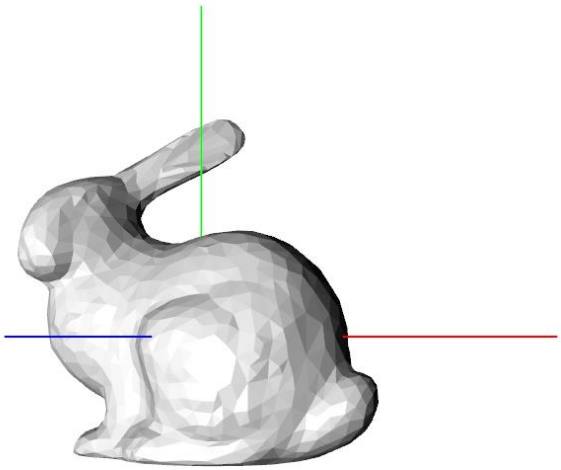


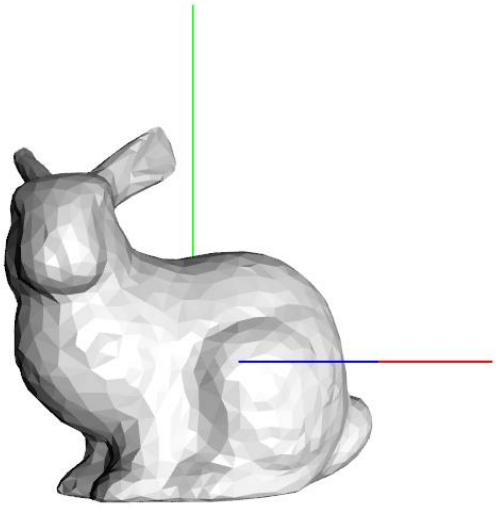
왼쪽 사진은 Orthographic projection에서 앞쪽으로, 오른쪽 사진은 뒤쪽으로 camera를 이동한 모습이다. 그리고 이는 Perspective projection에서 camera를 왼쪽, 오른쪽으로 이동한 방법과 같다.

하지만, 여기서 주의할 점은 glSetup.h에서 설정한 nearDist = 1.0f와 farDist = 20.0f에 의해 z값을 기준으로 -1~-20 사이의 값인 View volume을 벗어나면 bunny model을 잘리는 것을 볼 수 있다.

그리고 위에서 이야기 한 것처럼 Orthographic projection이기 때문에 z값이 (-1, -20) 사이의 값인 View volume에서는 camera의 이동에 따라 bunny model의 모양이 바뀌지 않는다.

Exercise01 SnapShot, Explanation:





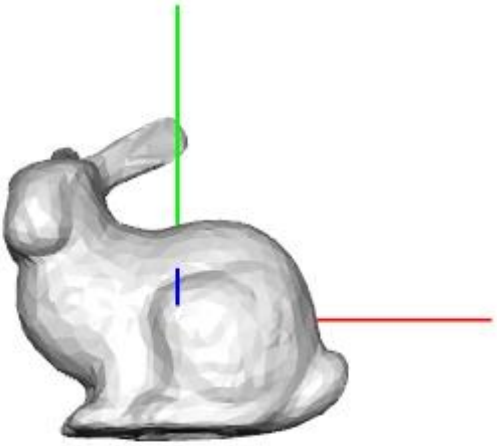
왼쪽과 오른쪽 사진은 각각 MOD_SHIFT key와 같이 LEFT, RIGHT arrow key를 누른 횟수 당 Camera의 up vector를 기준으로 CCW에 따라 10°, -10°를 한 모습이다. 또한, 회전된 모습이 잘 나오도록 camera를 적절히 조절하였다.

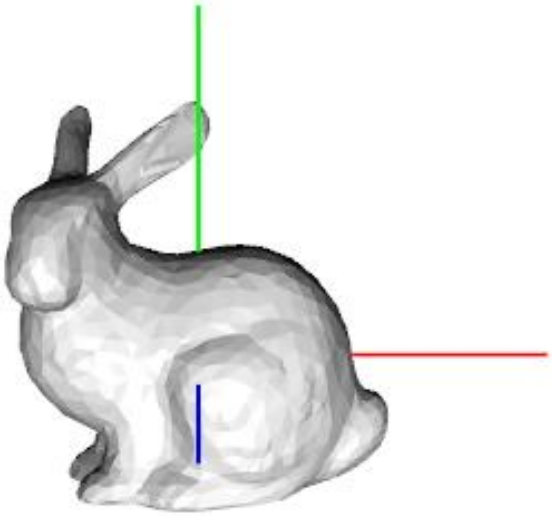
이를 구현한 방법으로는 MOD_SHIFT key와 같이 LEFT or RIGHT arrow key를 누르는 것에 따라 camera의 at position을 eye position을 중심으로 10° or -10°씩 pivot rotate를 하였다.

그리고 forward direction과 right direction을 재계산해야 하기 때문에 이들도 각각 camera의 eye position을 중심으로 10° or -10°씩 rotate를 하였다.

여기서 주의할 점은 direction의 경우 at position과 다르게 point가 아닌 direction이므로 pivot rotate가 아닌 rotate를 해줘야 한다.

Exercise02 SnapShot, Explanation:





왼쪽 사진은 Perspective projection에서 위로, 오른쪽 사진은 아래로 camera를 이동한 모습이다. 그리하여 파란색의 z축이 아래쪽과 위쪽으로 뻗어나가듯이 보이는 것을 볼 수 있다.
그리고 이는 위 Practice 2.에서와 같이 camera의 eye postion과 at position에 up vector의 값을 더해 구현할 수 있다. Ex) CTRL + UP : C.e += 0.1f * C.u; C.a += 0.1f * C.u;, CTRL + DOWN : C.e -= 0.1f * C.u; C.a -= 0.1f * C.u;

※여기서 주의할 점이 하나 있는데 온전한 camera의 위아래 이동을 위해서 앞에서 (action == GLFW_PRESS || action == GLFW_REPEAT) && !(mods & GLFW_MOD_SHIFT)일 때의 KEY_UP, KEY_DOWN에 && !(mods & GLFW_MOD_CONTROL) 조건을 추가해야한다.
그렇지 않으면 앞으로 이동하면서 위로 같이 이동하거나 뒤로 이동하면서 아래로 같이 이동한다..