심화전공실습

HW #10

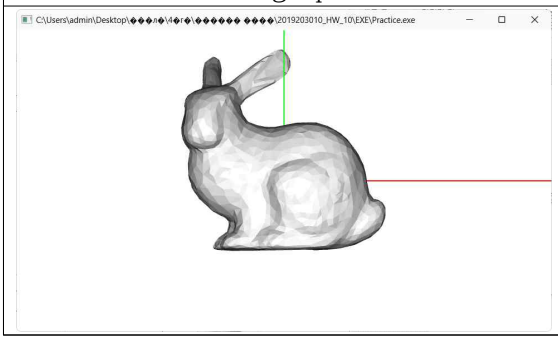
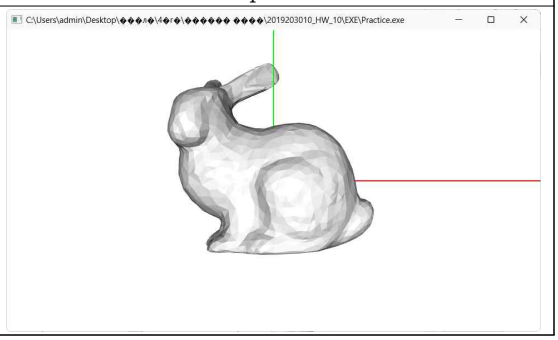
Self-scoring table

	P1	P2	P3	P4	Total
Score	1	1	1	1	4

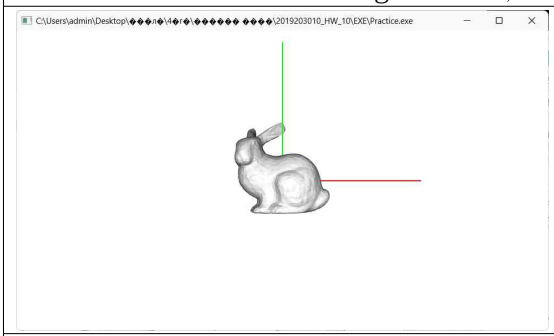
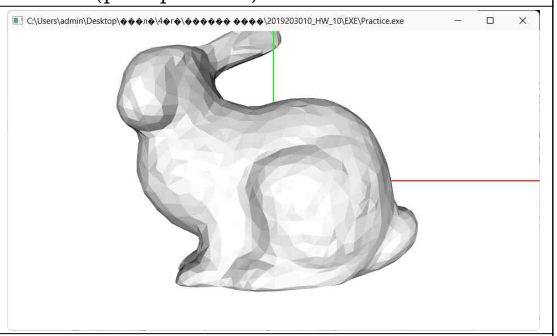
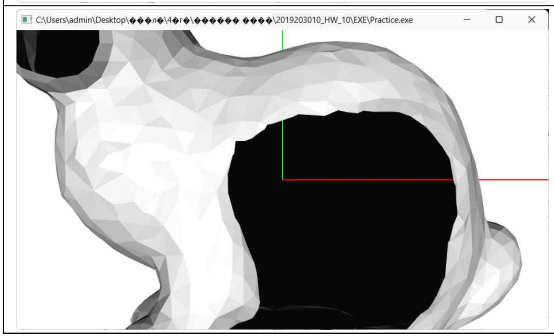
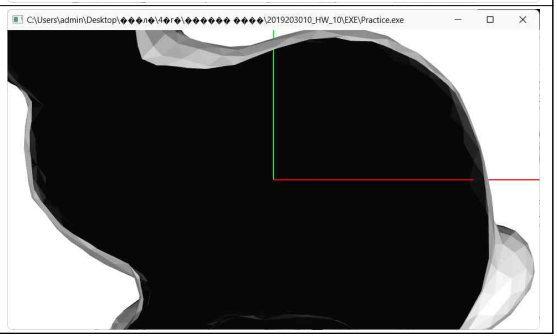
과목명	심화전공실습
학부	소프트웨어학부
학번	2019203010
이름	김민철
제출일자	2024년 11월 17일

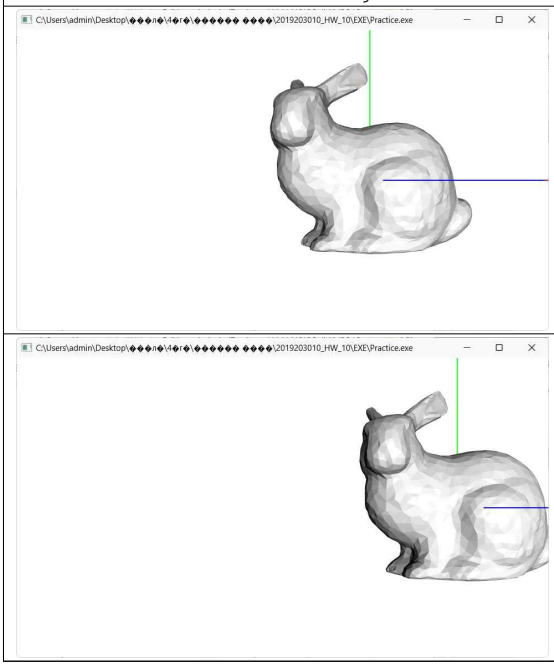
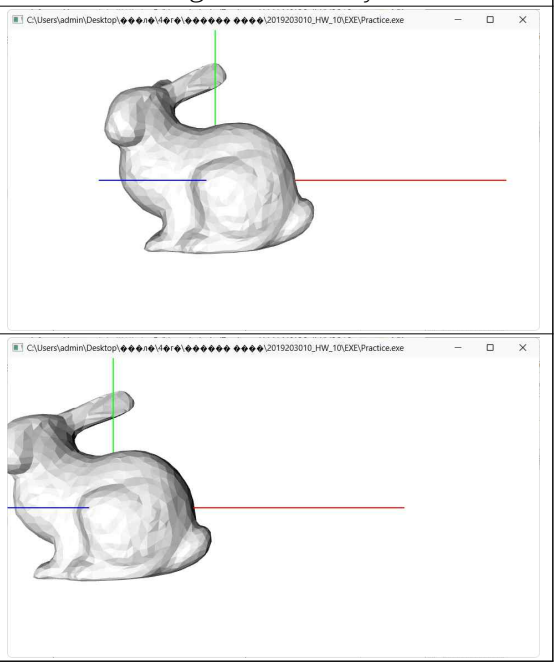
I. Practive

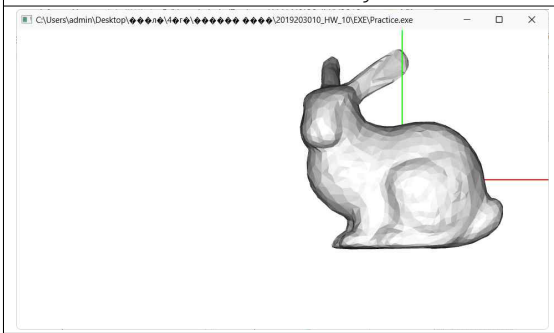
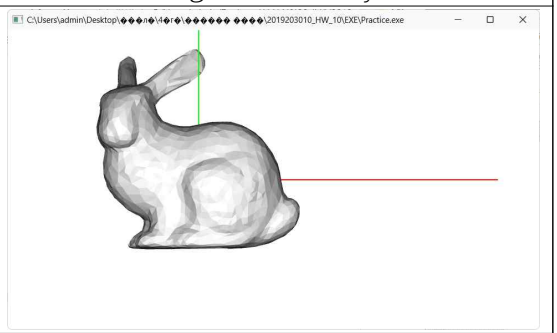
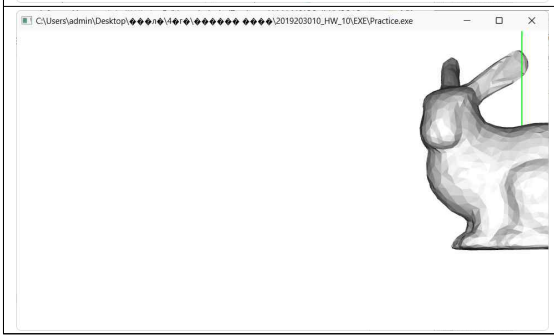
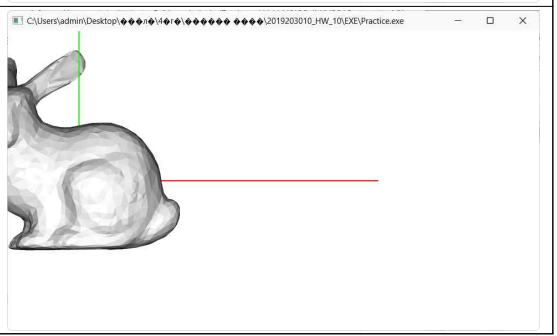
Practice 01. Orthographic and perspective projection

Orthographic	Perspectives
	

Practice 02. Interactive navigation with the arrow keys : forward, backword, left, right

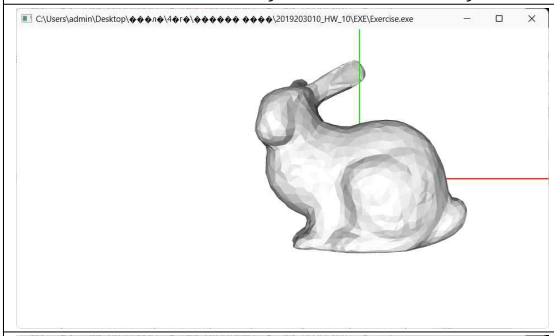
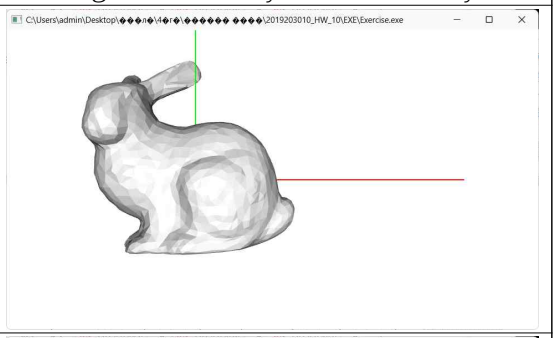
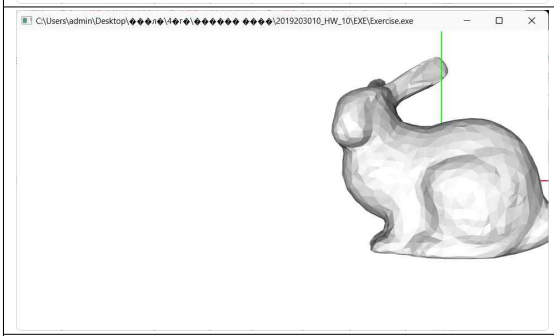
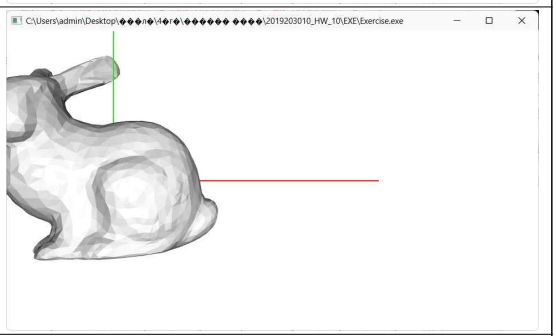
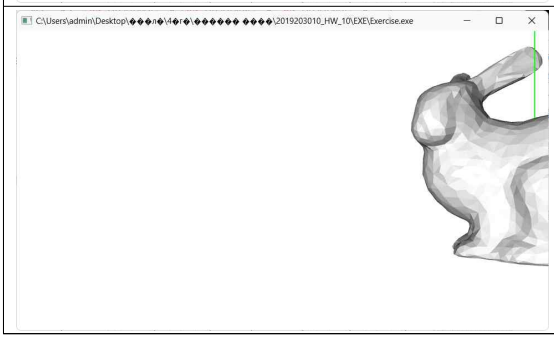
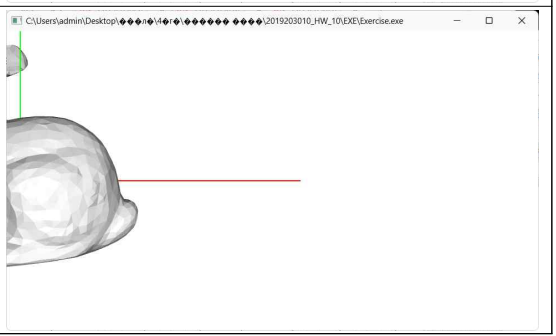
Moving forward, backward (perspective)	
	
	

Moving right, left (perspecive)	
left arrow key	right arrow key
	

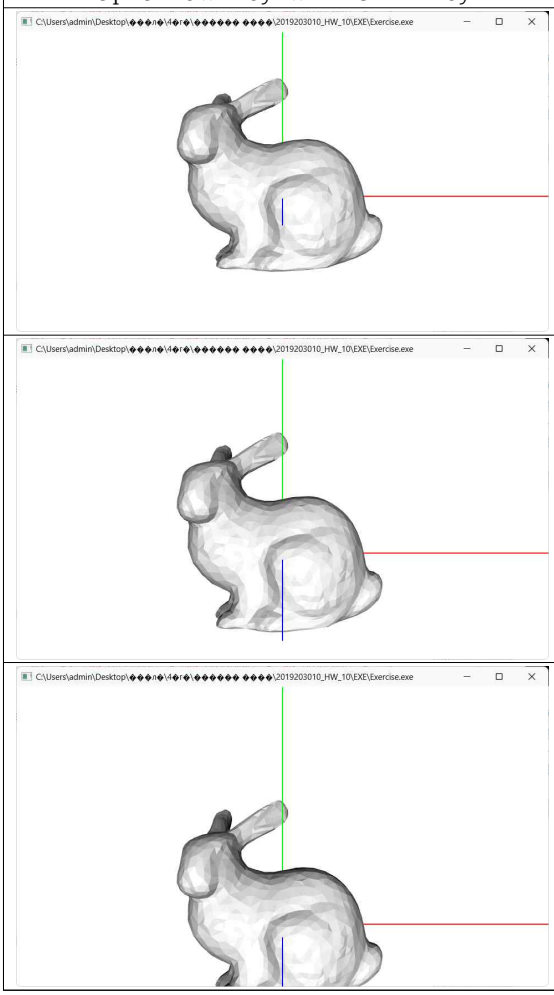
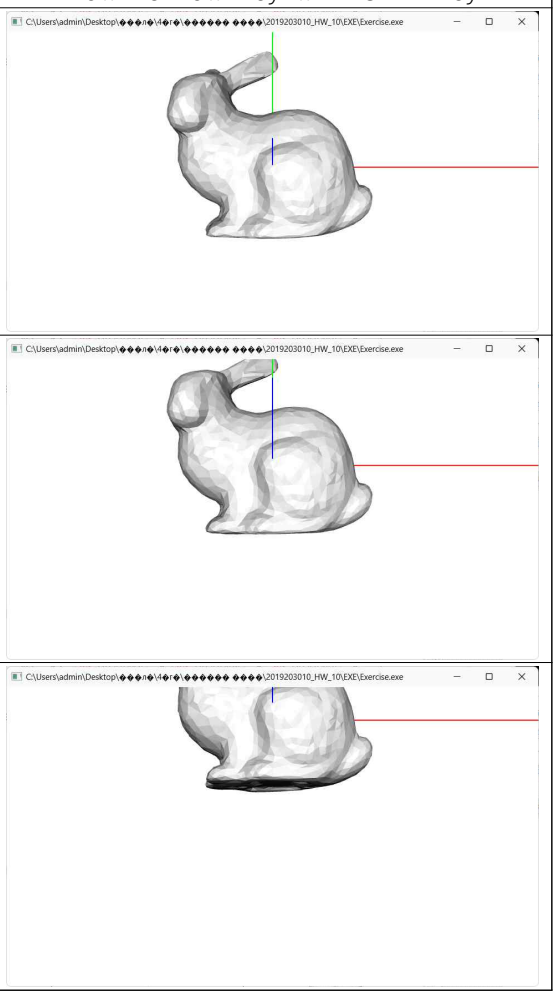
Moving right, left (orthographic)	
left arrow key	right arrow key
	
	

II. Exercise

Exercise 01. Tern left/right with the arrow and modifier keys

Turning legt, right (perspective)	
Left arrow key with Shift key	Right arrow key with Shift key
	
	
	

Exercise 02. Move up/down with the arrow and modifier keys

Moving up, down (perspective)	
Up arrow key with Shift key	Down arrow key with Shift key
	

Exercise 코드 설명

```
// changing view with shift key, turning / moving, a position이 바뀜, 이걸 e에 곱함
void turnLeft() {
    mat4 M(1.0);
    M = glm::translate(M, C.e);
    M = glm::rotate(M, radians(10.0f), C.u);
    M = glm::translate(M, -C.e);

    C.a = M * vec4(C.a, 1.0f);

    C.f = normalize(C.a - C.e);
    C.r = normalize(cross(C.f, C.u));
    C.u = normalize(cross(C.r, C.f));
}

void turnRight() {
    mat4 M(1.0);
    M = glm::translate(M, C.e);
    M = glm::rotate(M, radians(-10.0f), C.u);
    M = glm::translate(M, -C.e);

    C.a = M * vec4(C.a, 1.0f);

    C.f = normalize(C.a - C.e);
    C.r = normalize(cross(C.f, C.u));
    C.u = normalize(cross(C.r, C.f));
}
```

turnLeft() 함수:

목적: 시점을 왼쪽으로 10도 회전시킨다.

작동 방식:

벡터 u를 기준으로 10도 회전시키는 변환 행렬 M을 생성한다.

현재 변환 행렬 C.a에 M을 곱하여 새로운 변환 행렬을 얻는다.

(C.a = C.a * M)

업데이트된 C.a를 기반으로 C.f, C.r, C.u를 다시 계산한다.

turnRight() 함수:

목적: 시점을 오른쪽으로 10도 회전시킨다.

작동 방식:

벡터 u를 기준으로 -10도 회전시키는 변환 행렬 M을 생성한다.

현재 변환 행렬 C.a에 M을 곱하여 새로운 변환 행렬을 얻는다.

(C.a = C.a * M)

업데이트된 C.a를 기반으로 C.f, C.r, C.u를 다시 계산한다.

```
void moveUp() {
    C.e += 0.1f * C.u; C.a += 0.1f * C.u;
}

void moveDown() {
    C.e -= 0.1f * C.u; C.a -= 0.1f * C.u;
}
```

moveUp() 함수:

목적: 시점을 위쪽으로 이동시킨다.

작동 방식:

C.u 벡터 (위쪽 방향 벡터)를 0.1배 만큼 C.e (시점)에 더한다.

C.a (변환 행렬)에도 같은 값을 더하여 변환을 반영한다.

즉, 시점을 위쪽으로 약간 이동시키는 효과를 낸다.

moveDown() 함수:

목적: 시점을 아래쪽으로 이동시킨다.

작동 방식:

C.u 벡터를 0.1배 만큼 C.e에서 뺀다.

C.a에서도 같은 값을 빼서 변환을 반영한다.

즉, 시점을 아래쪽으로 약간 이동시키는 효과를 낸다.

```
if ((action == GLFW_PRESS || action == GLFW_REPEAT) && (mods & GLFW_MOD_SHIFT))
{
    switch (key)
    {
        //navigation
    case GLFW_KEY_LEFT:     cout << "left turn" << endl;    turnLeft();     break;
    case GLFW_KEY_RIGHT:    cout << "right turn" << endl;   turnRight();    break;
    case GLFW_KEY_UP:       cout << "move up" << endl;      moveUp();       break;
    case GLFW_KEY_DOWN:     cout << "move down" << endl;    moveDown();     break;
    }
}
```

Shift + 방향키: 카메라의 회전 또는 이동을 수행한다.

turnLeft, turnRight, moveUp, moveDown 함수: 각각 왼쪽 회전, 오른쪽 회전, 위쪽 이동, 아래쪽 이동을 담당한다.

keyboard() 함수: 키보드 입력을 감지하고, Shift 키와 방향키가 동시에 눌렸을 때 해당 함수를 호출한다.

III. 느낀점

3차원 공간에서의 카메라 조작이 생각보다 복잡하다는 것을 새삼 느꼈습니다. 단순히 시점을 옮기는 것처럼 보이지만, 실제로는 변환 행렬, 벡터 연산 등 다양한 수학적 개념이 복합적으로 작용한다는 것을 알게 되었습니다. 특히, turnLeft()와

turnRight() 함수에서 벡터 u를 기준으로 회전하는 부분은 직관적으로 이해하기 어려웠고, 이를 구현하기 위해 회전 행렬 생성 등 추가적인 학습이 필요했습니다.