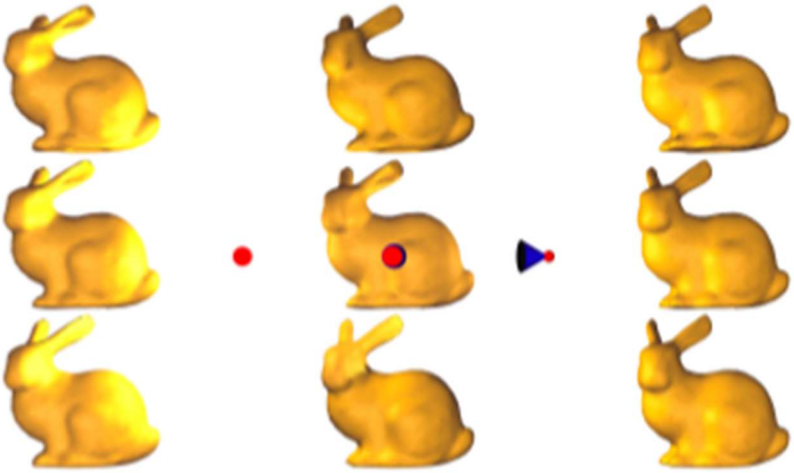
Computer Graphics



Alpha Blending

HW14

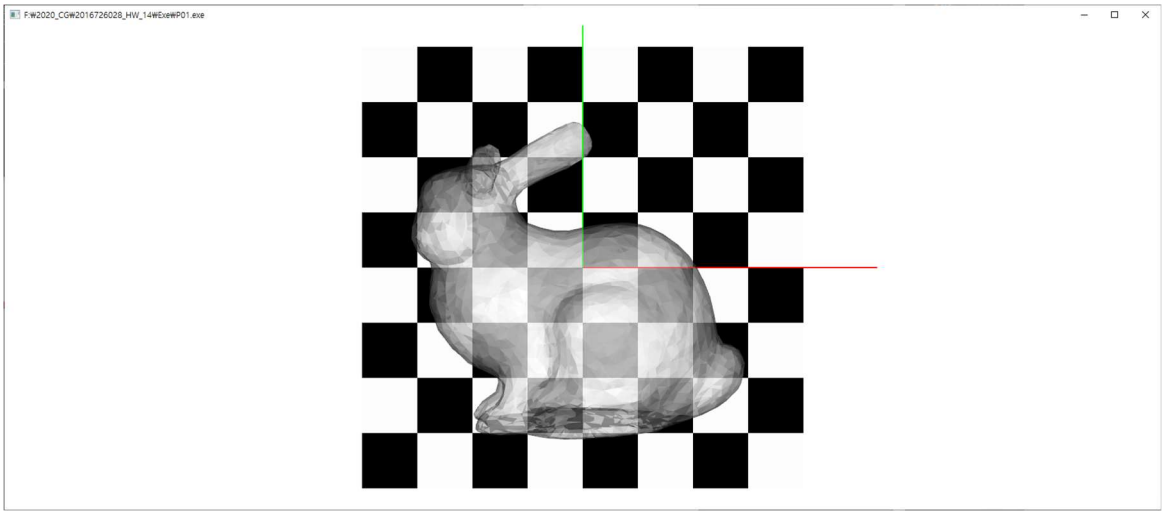
이민재 | Computer Graphics [심화전공실습 1] | 2020/12/08

	P01	P02	P03	P04	Total
SCORE	1	1	1	1	4

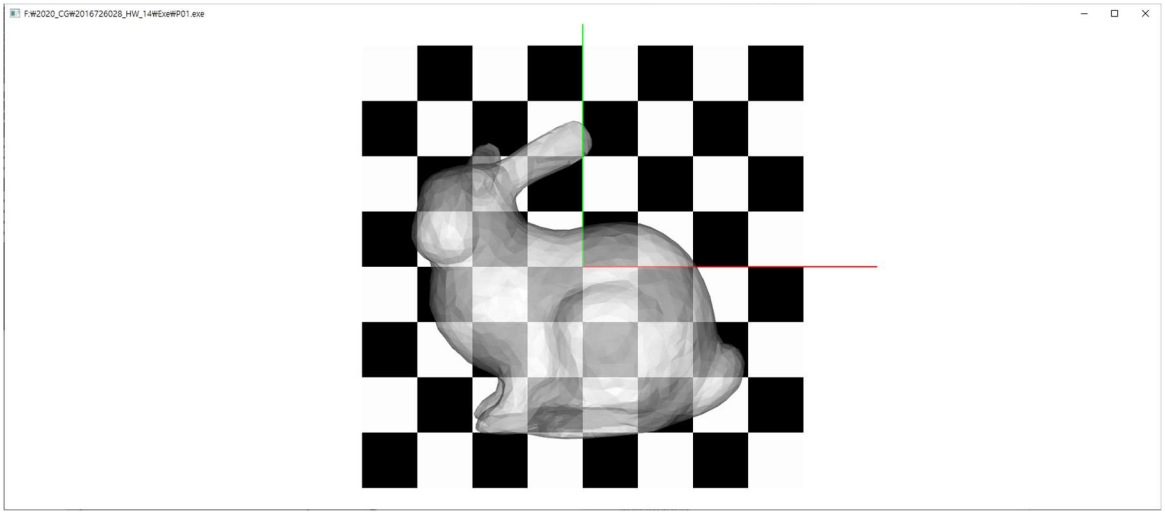
P01 (Turn on/off depth sorting to a translucent flat bunny)

<SNAPSHOT>

DEPTHSORTING = OFF



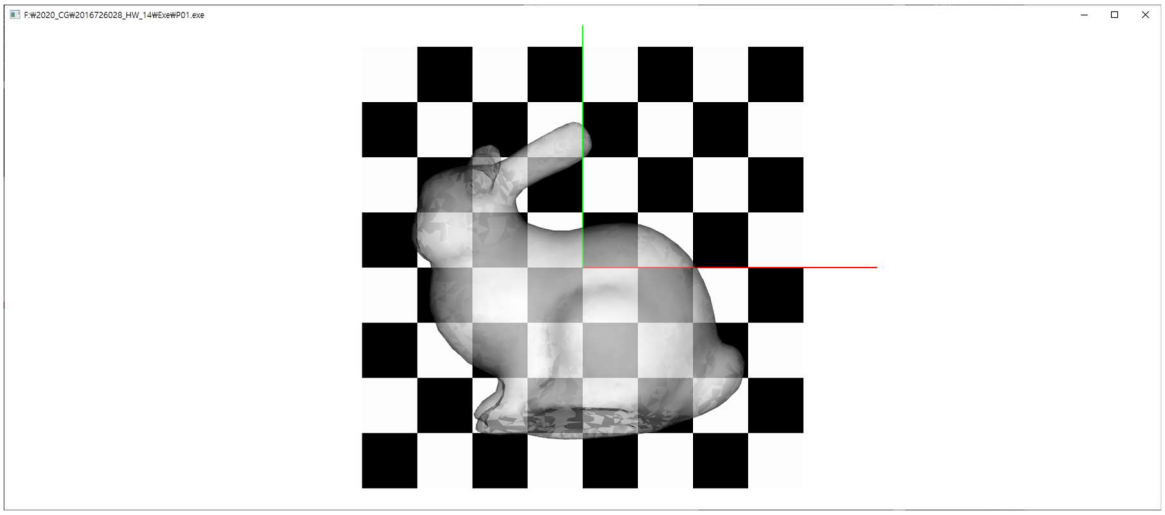
DEPTHSORTING = ON



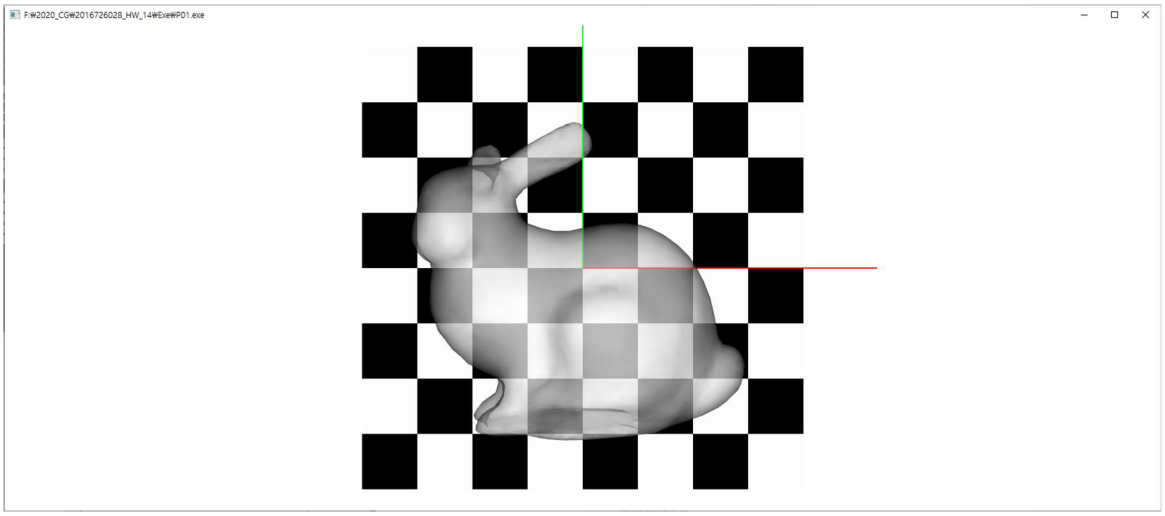
P02 (Turn on/off depth sorting to a translucent smooth bunny)

<SNAPSHOT>

DEPTHSORTING = OFF

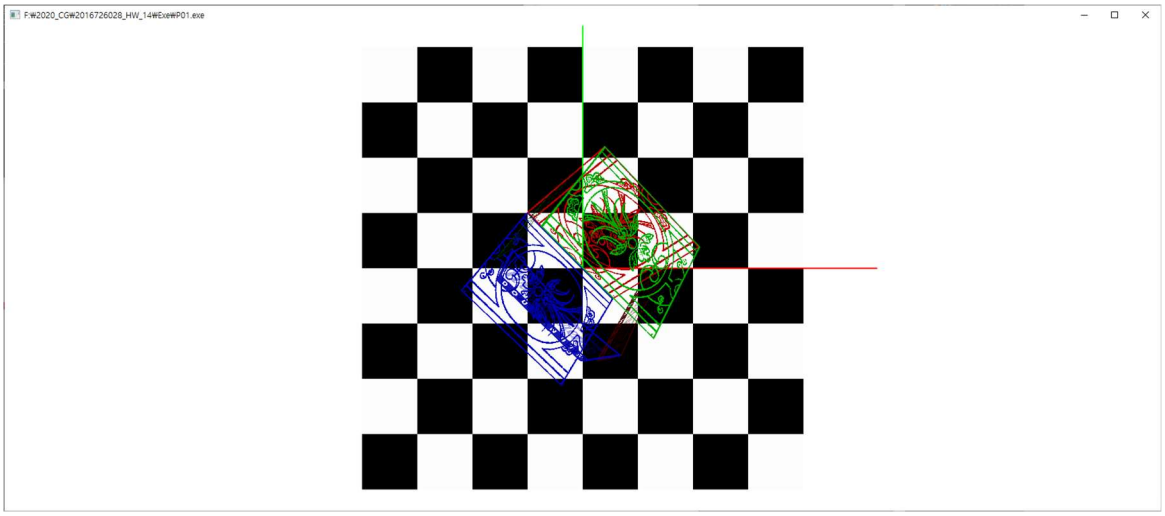


DEPTHSORTING = ON



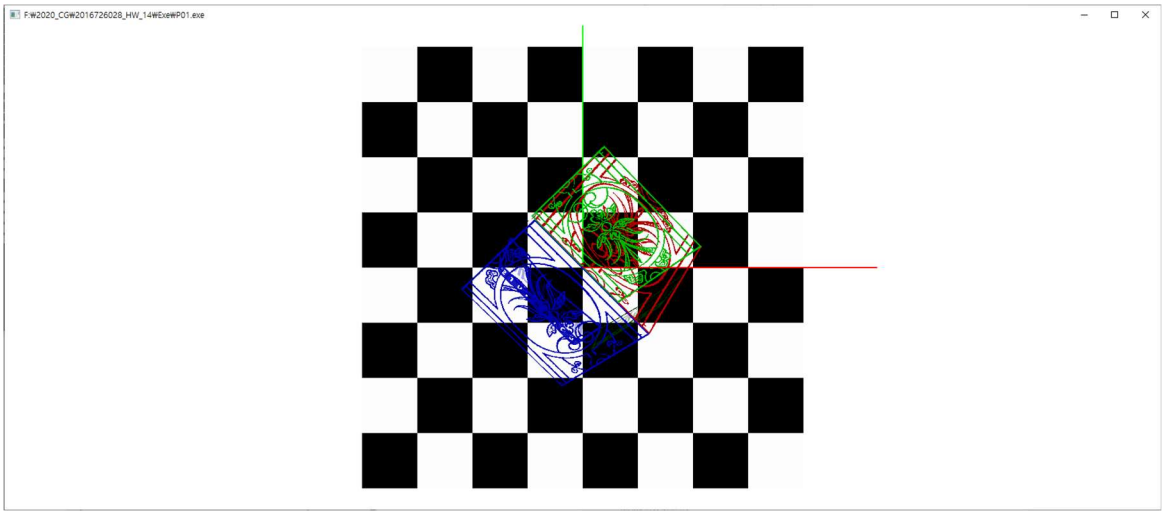
P03 (Alpha textured cube without depth sorting)

<SNAPSHOT>

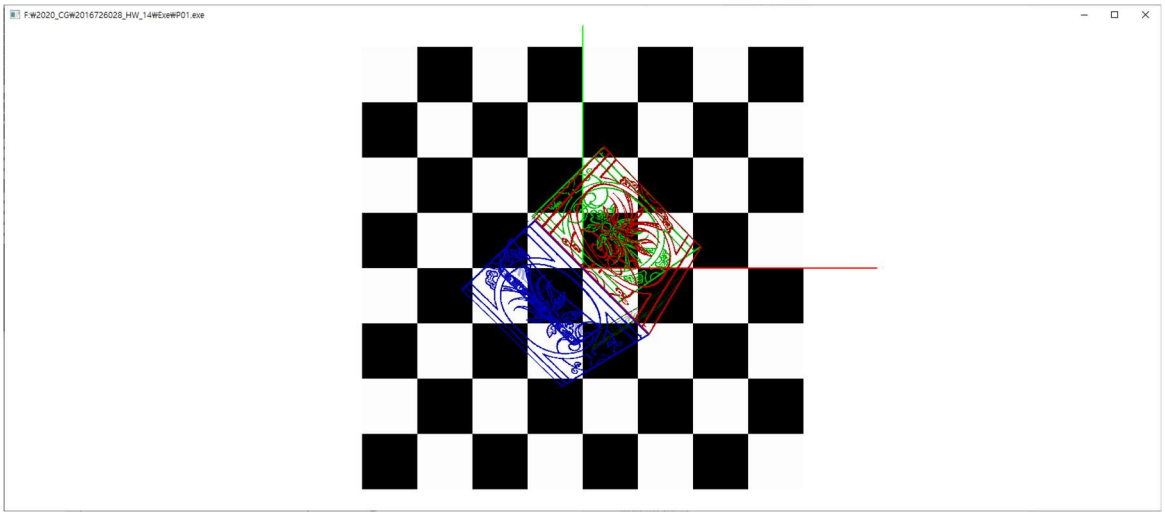


E01 (Alpha textured cube with/without depth sorting)

<SNAPSHOT>
DEPTHSORTING = OFF



DEPTHSORTING = ON



<EXPLANATION>

Depth sorting 을 한 cube 의 경우 붉은색면이 초록색 면보다 나중에 그려질 때 위에 분명하게 붉은색 텍스처가 표시되지만 off 상태의 경우 초록색과 붉은색이 겹치는 부분을 보면 어느것이 관찰자에 더 가까운지 모호해보인다.

Cube 에 depth sorting 을 적용하기 위해서 다음과 같은 처리를 해주었다. 기본적인 코드 구조는 DepthSortData struct 를 사용하는 bunny 예제를 참고하였다.

```
vec3 center;

for(int i=0; i<6; i++)
{
    //Depth sorting Data
    center = vertex[i][0] / 4.0f;
    center += vertex[i][1] / 4.0f;
    center += vertex[i][2] / 4.0f;
    center += vertex[i][3] / 4.0f;

    fcbe[i].i = i;
    fcbe[i].center = vec4(center, 1.0f);
}

//sortfaces
sortCubeFace();
// Cube
for (int i = 0; i < 6; i++)
{
    int iFace = fcbe[i].i;
    glBindTexture(GL_TEXTURE_2D, texID[iFace + 1]);
    glBegin(GL_QUADS);



    glNormal3fv(value_ptr(normal[iFace]));
    for (int j = 0; j < 4; j++)
    {
        glTexCoord2fv(value_ptr(texcoord[j]));
        glVertex3fv(value_ptr(vertex[iFace][j]));
    }
    glEnd();

}
```

먼저 지정한 각 면에 대해서 center 값을 구한다음, DepthSortData 구조인 scbe 에 저장하고 이 저장한 값을 sortCubeFace 함수를 호출하여 정렬한다. sortCubeFace 는 sortMeshFace 와 인자 크기만 다를뿐이지 완전히 동일하여 설명을 생략한다. 이후 정렬된 fcbe[i].i 의 값을 이용하여 먼 depth 순에 따라 순차적으로 텍스쳐를 입히고 사각형을 출력한다.