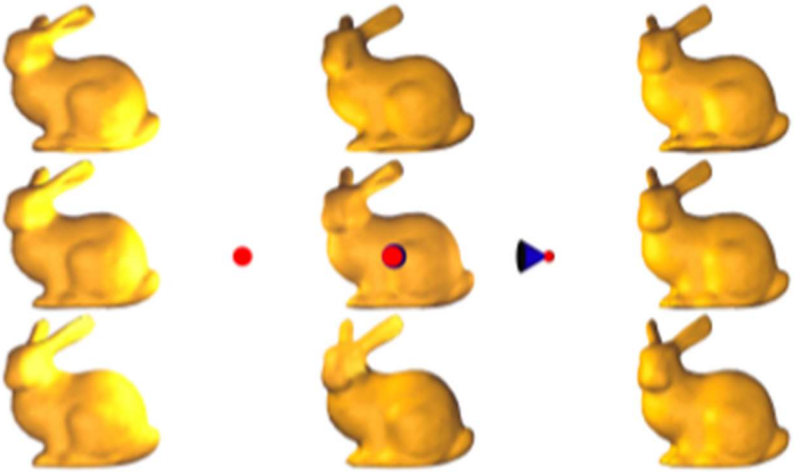
Computer Graphics

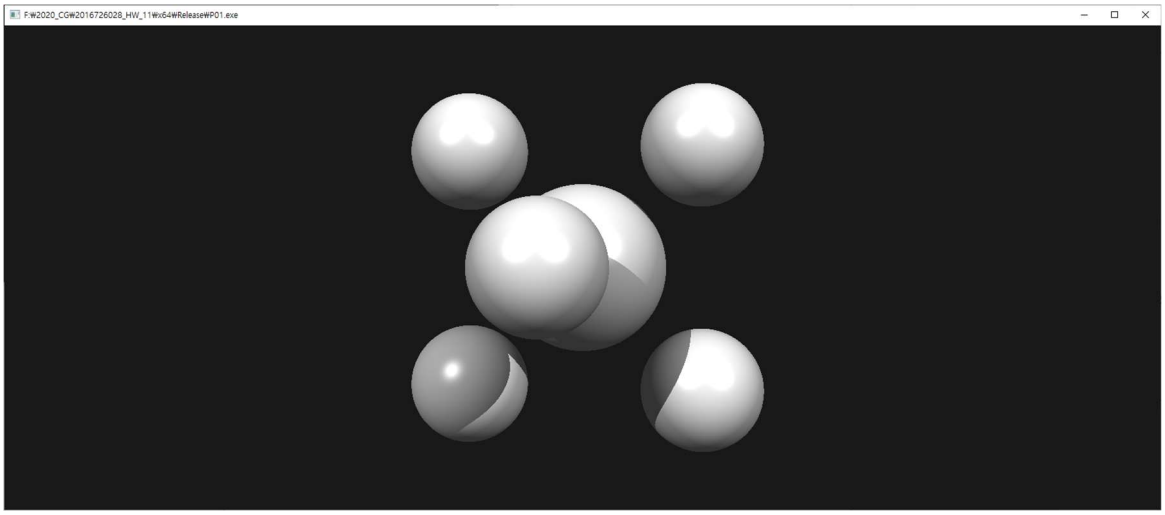


ray_tracing

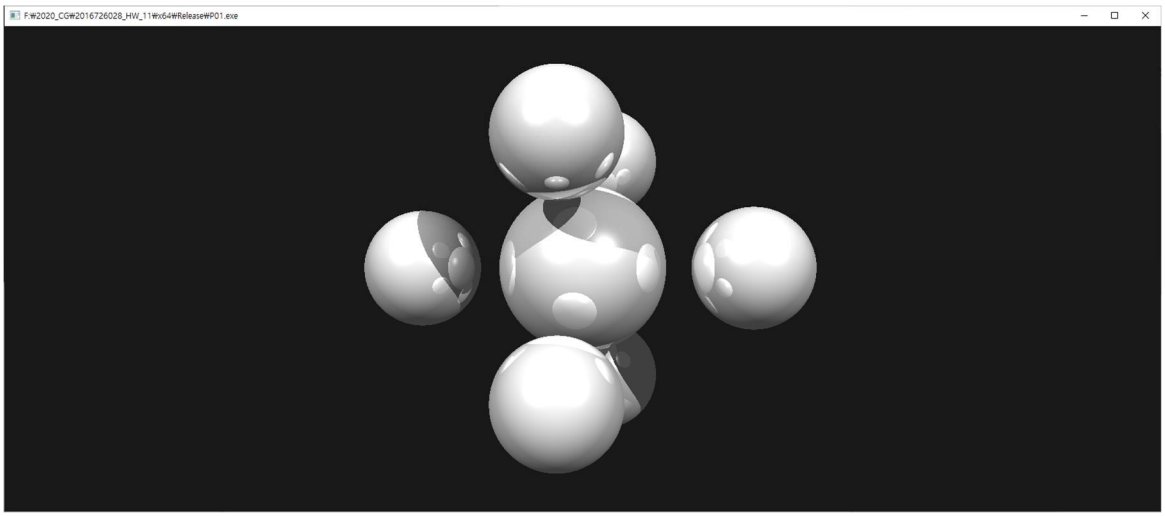
P01 (Ray tracing of spheres with various ray tracing depth)

<SNAPSHOT>

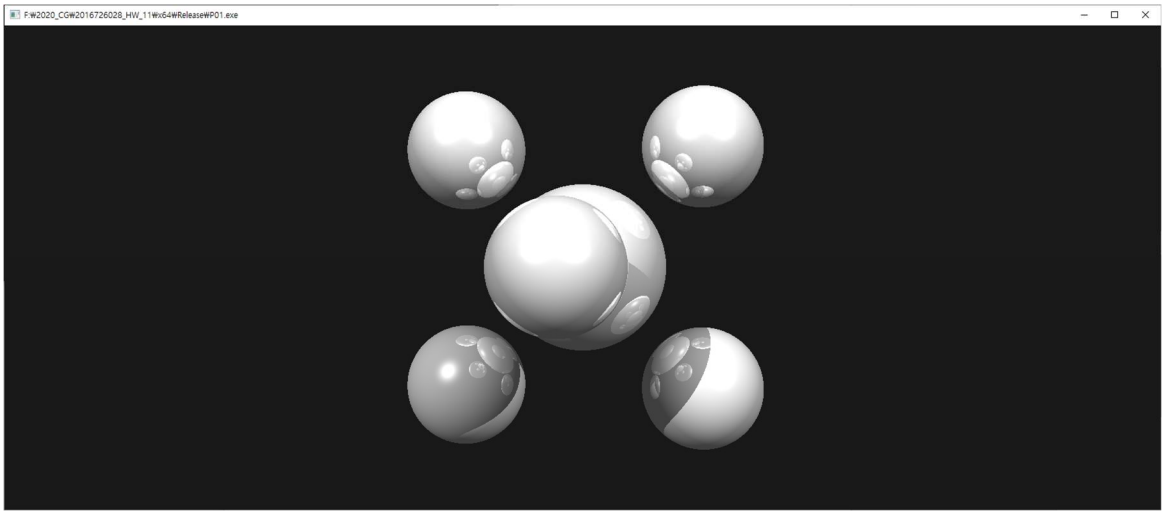
DEPTH =1



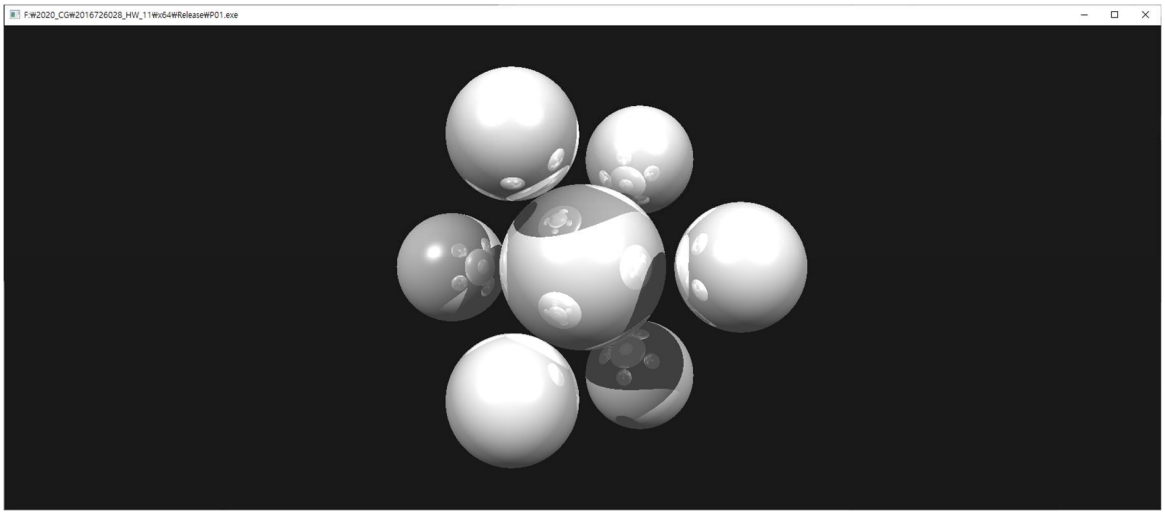
DEPTH =2



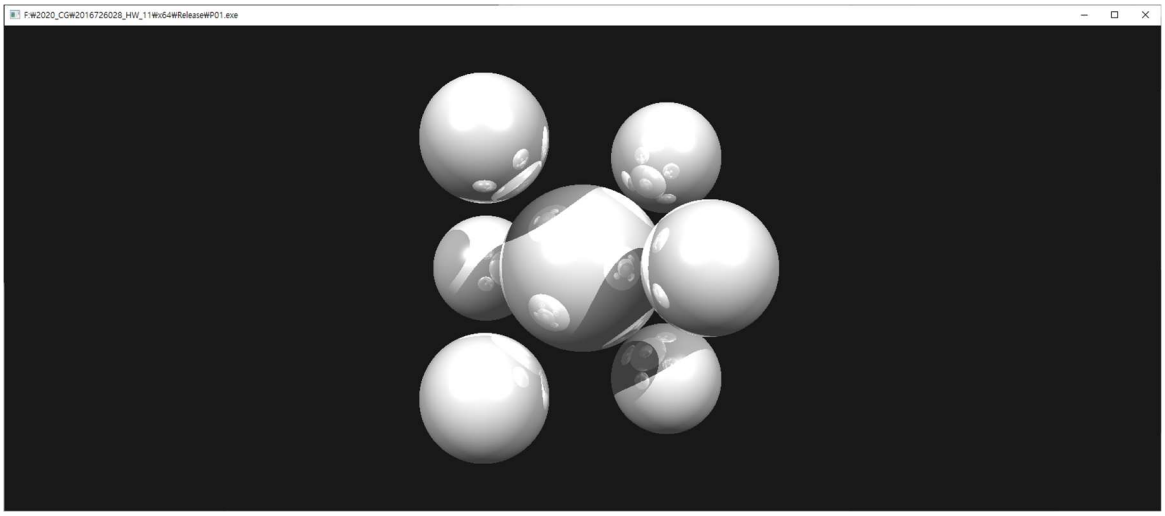
DEPTH =3



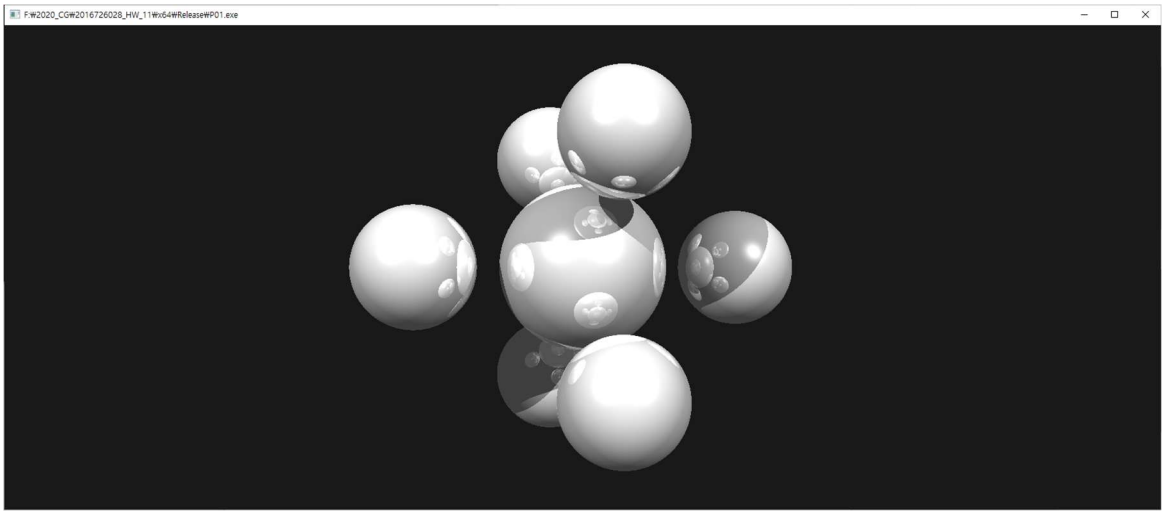
DEPTH =4



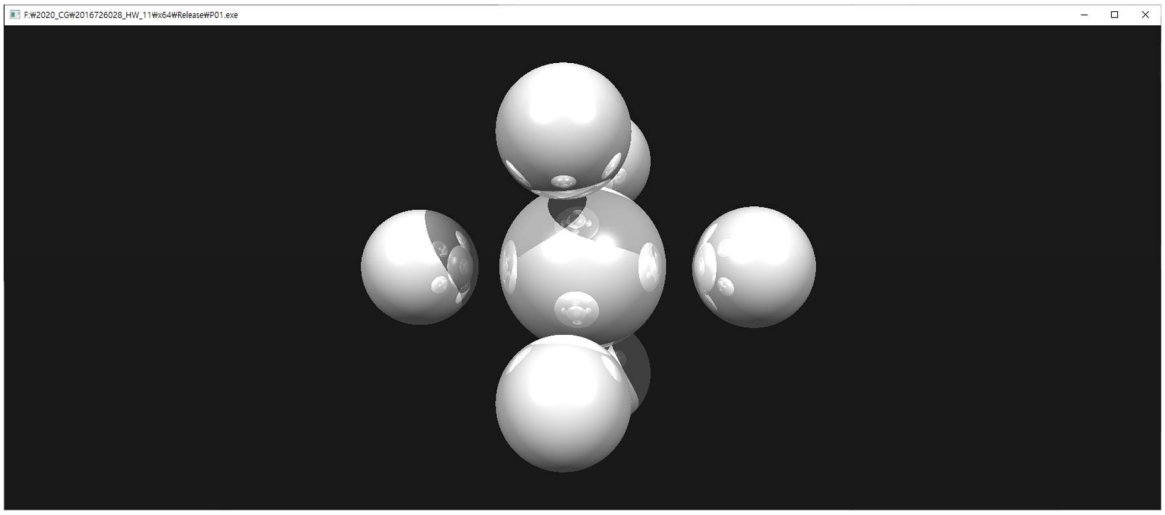
DEPTH =5



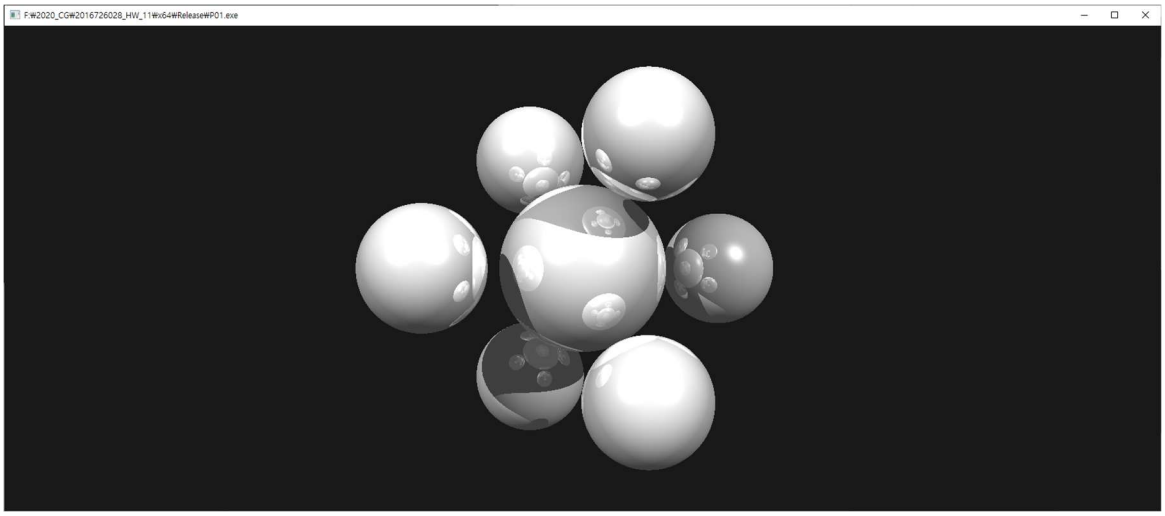
DEPTH =6



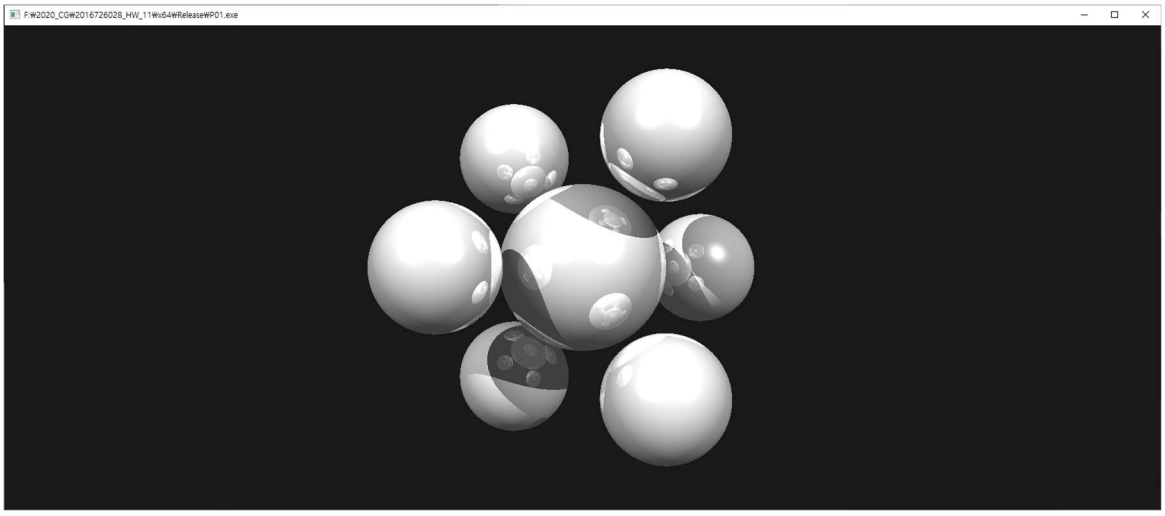
DEPTH =7



DEPTH =8



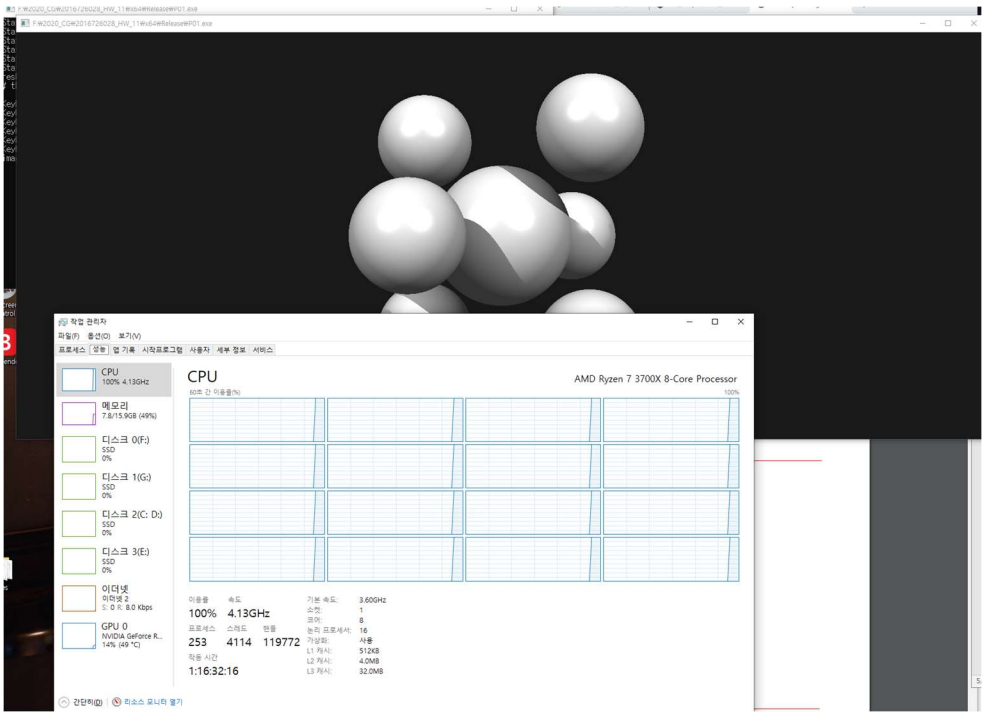
DEPTH =9



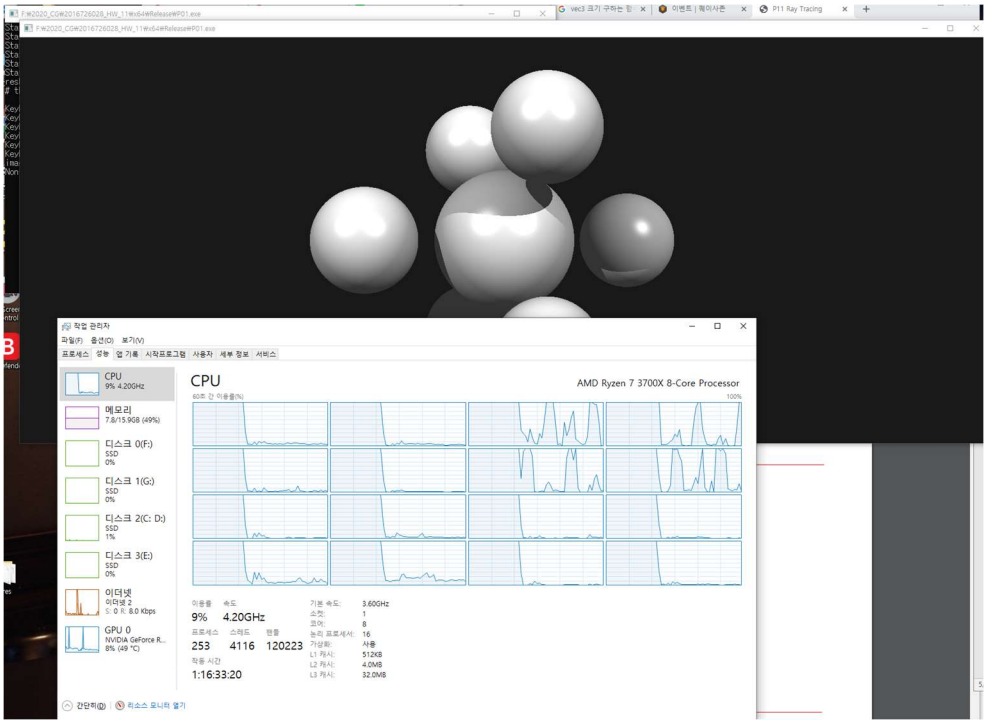
P02 (Parallel computing using OpenMP (show CPU usage))

<SNAPSHOT>

Parallel computing on

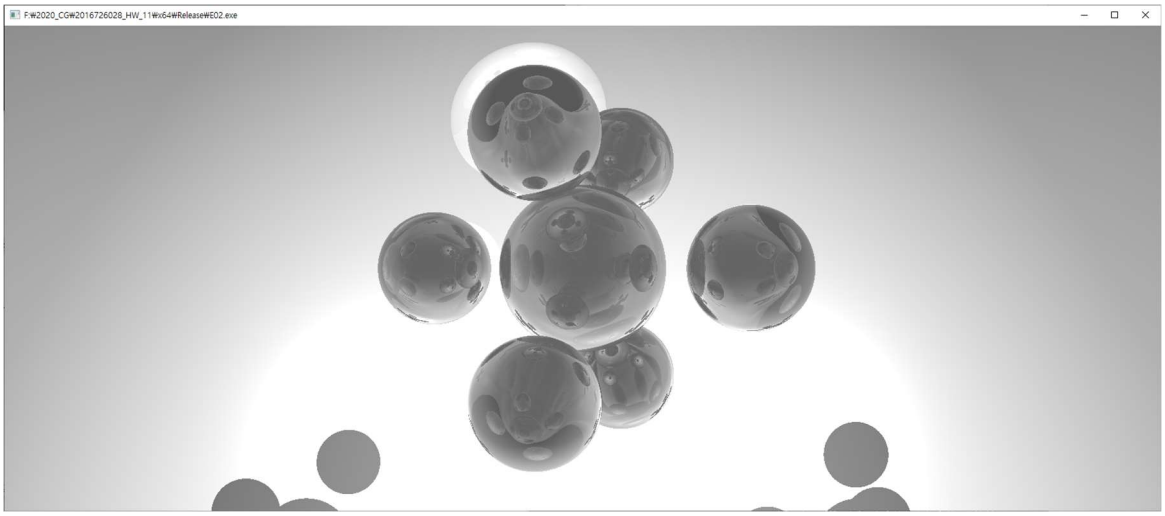


Parallel computing off



E01 (Enclose the scene with a sphere)

<SNAPSHOT>



<EXPLANATION>

먼저 영역을 뒤덮는 큰 구를 생성하기 위해 vec3 center_world 와 radius 컨테이너의 크기를 1 씩 늘리고 init 함수부에서 반지름을 8.5f 로 설정하여 구의 특성을 가지는 영역을 설정하였다.

그 다음, 구의 내부 반사를 구현하기 위해서 findIntersection 의 조건을 다음과 같이 수정하였다.

```
else if (t0 == 0 || t1 == 0)
{
    t = std::max(_Left: t0, _Right: t1);    // one of t = 0
    vec3 n_p1 = normalize(ray.p1);
    p = ray.p0 + n_p1 * 2.0f * x / y;   // The closest intersection point
    n = normalize(x: center - p);    // reverse Normal at the point
}
else {
    t = std::max(_Left: t0, _Right: t1);    // Beyond 0

    p = (1 - t) * ray.p0 + t * ray.p1;  // The closest intersection point
    n = normalize(x: center - p);    // reverse Normal at the point
}
```

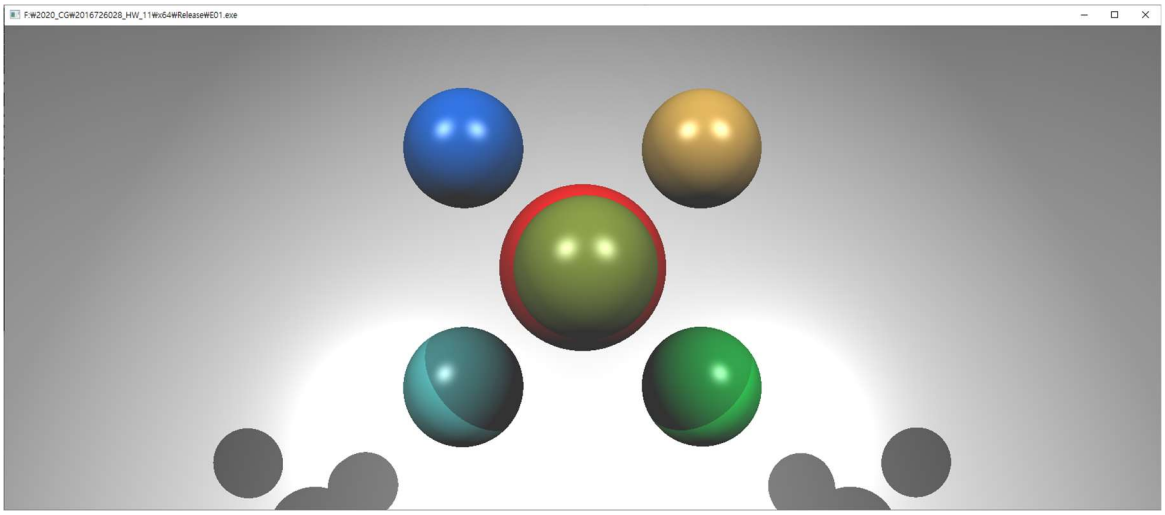
구의 내부 반사를 위한 부분은 else if 부분으로 x 와 y는 각각 `float x = dot(p10, pc0);`

`float y = sqrt(p10.x * p10.x + p10.y * p10.y + p10.z * p10.z);` 이다. $cos\theta = \frac{a \cdot b}{\|a\| \cdot \|b\|}$ 인 성질을 이용하여 pc0 과 p10 사이각을 구하고 이를 이용하여 p0 에서 p(t)로 향하는 벡터의 방향과 크기를 구해 p0 에 더해주었다. 또한, 구 내부의 반사에서 법선 벡터는 구의 중심방향을 향해야 함으로 normalize 한 벡터가 수직 반대방향으로 나오도록 center 에서 p 의 차를 방향으로 지정하였다.

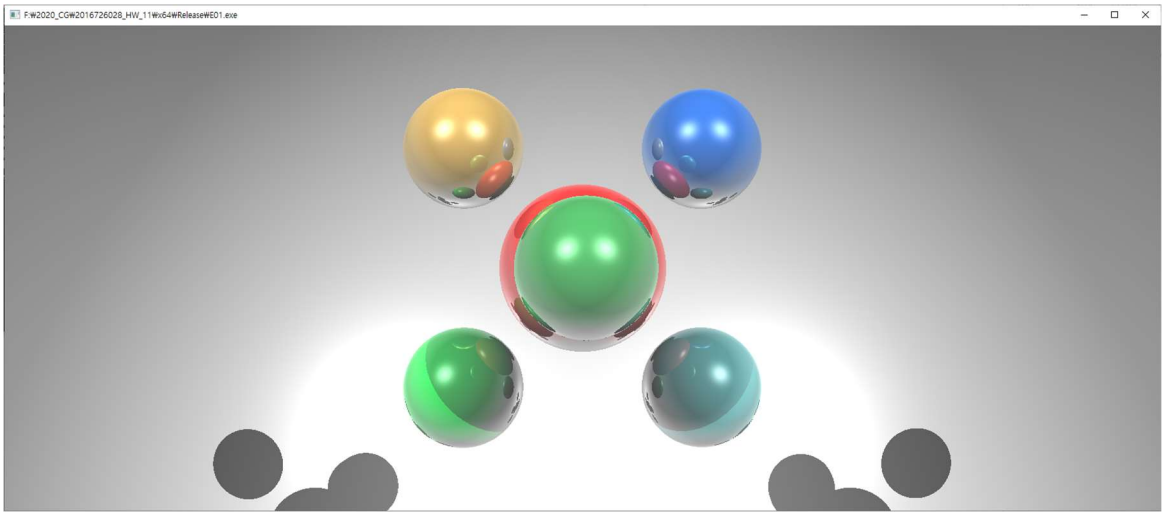
E02 (Employ a different color for each sphere)

<SNAPSHOT>

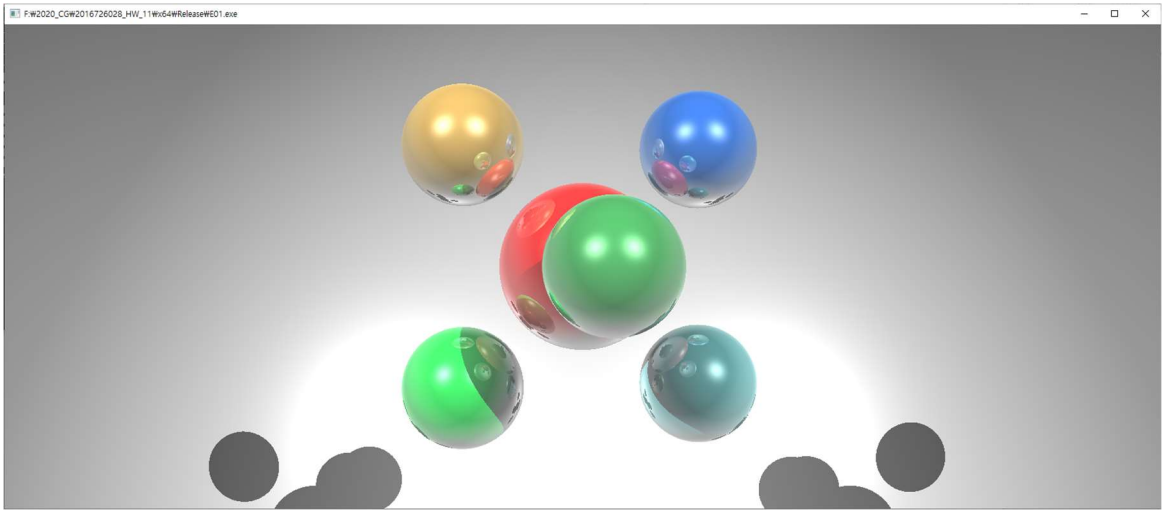
DEPTH=1



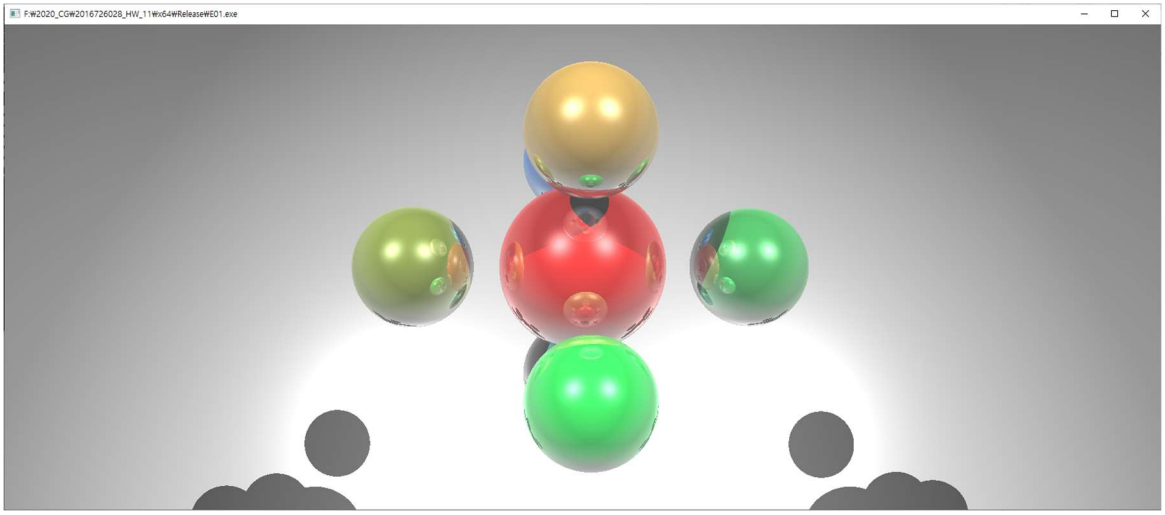
DEPTH=2



DEPTH=3



DEPTH=4

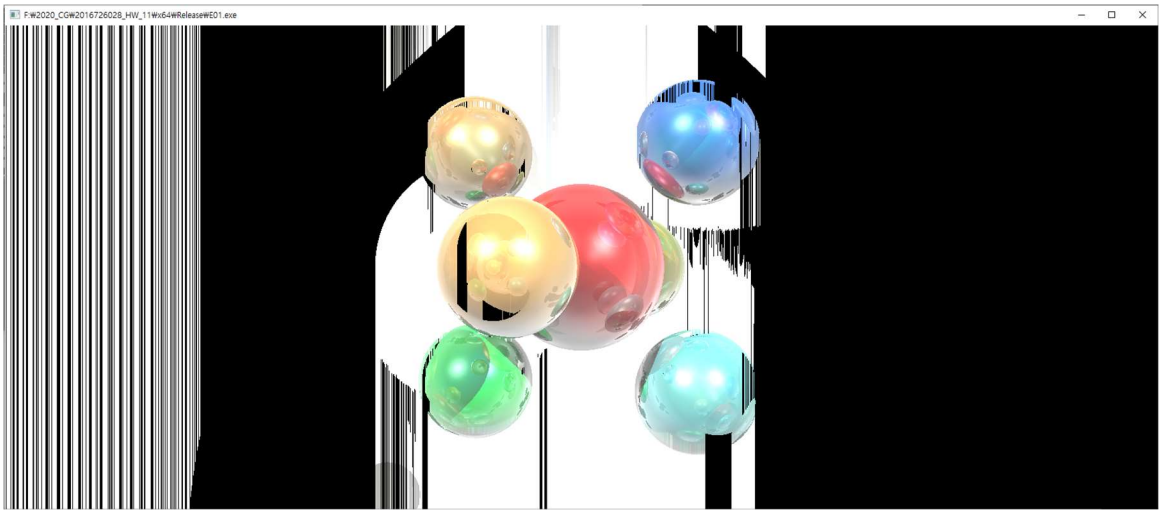


<EXPLANATION>

각 구의 색을 바꾸기 위해 m_diffuse 를 여러 종류를 만들어 iObject 인자를 전달받도록 수정한 phong 함수에서 다음과 같이 처리하도록 하였다.

```
switch(iObject)
{
case 7: I[i] += m_diffuse8[i] * lambertian * l.diffuse[i]; break;
case 0:I[i] += m_diffuse1[i] * lambertian * l.diffuse[i]; break;
case 1: I[i] += m_diffuse2[i] * lambertian * l.diffuse[i]; break;
```

이때 허공에 ray 가 발생하는 경우는 사라지고, 이가 case7 인 가장 큰 구에 부딛히는 경우 임으로 이를 위해 -1 로 설정하였던 기존 값들을 전부 object =7 의 수식으로 바꾸어 주었다. 이에 따라 , intensity 함수의 else 부도, 벽에서 벽으로 반사하는 경우를 고려하여 phong 모델이 작동하도록 수정하였는데, 다음과 같은 오류가 발생하였다.



벽에서 벽으로 (7 번구에서 7 번구로) 반사하는 경우의 reflection ray 를 계산하여 specular 값을 더해주면 위 화면과 같이 구의 반사의 품질은 원했던 것처럼 출력되나 ,전체 출력물이 검은색으로 깨지는 상황이 발생하였다. Recrusive Ray 를 계산하여 대입하는 과정에서 문제가 발생하는 것 같은데 정확한 원인을 찾지 못하여 최종 결과물에서는 reflection ray 부분을 제거 하였고, 상단의 4 개의예시 출력 결과물을 얻었다.