컴퓨터 그래픽스 과제5

18011683 조현우

그림자

TO-DO: Check for shadows

- 만약 hit한 지점과 광원과의 거리 사이에 구와 intersect한 부분이 있다면 그것은 차폐된 부분이라고 볼 수 있습니다.

TO-DO: If not shadowed, perform shading using the diffuse color only

- 차폐가 되지 않았을 경우에 diffuse값을 N, L을 이용하여 조절해 명암을 나타내줄수있습니다.

- color += uLIntensity * (NL* mtl.k_d);

Ray와 구의 교차

TO-DO: Test for ray-sphere intersection

- 구의 방정식을 $(P-C)\cdot(P-C)=r^{\wedge}2$ 으로 볼수 있습니다. 그렇기 때문에 광선이 이 구와 닿는 지점을 P(t)라고 하면 P(t) = A + tb (A: ray의 좌표, b: ray 방향벡터) 라고 볼수 있고 만족하는 t의 개수가 0이면 구와 만나지 않는 것, 1이면 광선과 구는 한점에서 만나는 것, 2이면 광선이 구를 뚫고 두개의 점에서 만나는 것으로 볼수 있습니다.
  즉, $(P(t)-C)\cdot(P(t)-C)=r^{\wedge}2$ 은 $(A+tb-C)\cdot(A+tb-C)=r^{\wedge}2$ 로 나타낼수 있고
  이를 정리하면 $t^{\wedge}2b\cdot b+2b\cdot(A-C)t+(A-C)\cdot(A-C)-r^{\wedge}2=0$ 의 방정식을 얻게 됩니다. t 에 대한 근의 공식을 통해서 근의 개수가 이전에 말했듯이 1이상이면 구와 교차한다고 볼 수 있습니다.

TO-DO: If intersection is found, update the given HitInfo

- 결국 판별식인 (b(A – C))^2 – b^2 * ((A-C)•(A-C) - r^2) >= 0 에 의해 교차하는지 판별이 가능합니다. 만약 교차한다면 hit한 지점의 값을 업데이트 해주게 됩니다.

레이트레이싱

TO-DO: Initialize the reflection ray

- 광선이 튕길때마다 어느 부분으로 튕기게 되는지 찾아주며 튕긴지점을 업데이트해주어야 합니다. 그래서 reflection ray에 대해서
  r.pos = hit.position.xyz;
  r.dir = normalize(reflect(view, hit.normal)).xyz;

와 같이 r.pos 는 hit한 지점의 좌표를, r.dir은 reflect함수를 이용하여 바라보는 방향에서 hit.normal을 이용해 반사되는지점을 알아내 이를 normalize하여 방향을 구하게됩니다.

TO-DO: Hit found, so shade the hit point

- 반사되어 다른 구에 맞았다면 clr (컬러) 값에 hit.mtl.k_s * 쉐이드된 diffuse값을 넣어줍니다.

TO-DO: Update the loop variables for tracing the next reflection ray

- 반사된 ray 에 대해서 업데이트를 해주어야하므로 view의 방향은 r.dir의 방향으로 바뀌게 되고 k_s는 반사되어 맞은 구의 material의 specular값을 곱해주어 업데이트합니다.

시행착오

- 근의 공식을 통해 구한 hit한 지점이 두개일 때 어떤 것을 선택해야하는지에 대해서 정확히 이해 할 수 없었는데 이를 직접 두가지 경우를 실행해보면서 가장 가까운 지점에 대해서 hit한 점이 되어야겠다는 판단을 하였습니다.

- Update the loop variables for tracing the next reflection ray 부분에서 k_s의 값을 업데이트 할 때 specular값이 계속 누적되어야 된다고 생각을 해서 덧셈으로 처리를 했었는데 반사 표면이 너무 밝게 나와서 어차피 specular의 값이 전부 1보다 작으므로 곱하기 누적으로 해줘 레이트레이싱이 반사될 때마다 점점 어두워지게 처리를 하였습니다.

- 처음에 구와 ray가 hit하는지에 대한 판단을 할 때 ray.pos에서 구와 한점에서 만날때의 방향벡터와 ray.pos에서 구의 중심으로의 방향벡터 이 두가지의 벡터 사이의 각도보다 ray.dir과 ray.pos에서 구의 중심으로의 방향벡터의 각도가 작거나 같다면 hit한다고 봐도 된다고 생각을 했습니다. 하지만 이후에 어느지점에서 정확히 만나는지에 대한 값을 구하지 못하여 이는 구글링을 통해서 방법을 알아내었고 구의 방정식에 ray.pos + ray.dir * t 을 대입해 근의 공식을 통하여 구할 수 있다는 것을 알았습니다.
(참고:
https://raytracing.github.io/books/RayTracingInOneWeekend.html#addingasphere/ray-sphereintersection)

느낀점

- 한 학기의 마지막과제를 하며 전체적인 개념들을 다시 정리해볼 수 있었고 많이 어려웠지만 성공했을 때 굉장한 성취감을 느꼈고 막힐때 구글링을 통해 찾아보기도 하고 분석하는 과정들이 앞으로의 코딩생활에 큰 도움이 될 경험이였으며 평생 잊지 못할 과제가 된 것 같습니다. 감사합니다.