컴퓨터 그래픽스 HW5

2017-11522 컴퓨터공학부 박종석

과제 구현사항 및 구현방법

Ray Tracing Spheres / Polygons
제출한 이미지에서 확인할 수 있듯이 sphere과 polygon에 대한 ray tracing을 구현했습니다.

Recursive Reflection / Refraction
depth를 5로 설정해서 recursive하게 reflection과 refraction이 계산되고 있습니다.

Phong Illumination
물체마다 Ka, Ks, Kd를 주어서 diffuse reflection을 하는 물체에 Phong Illumination을 구현했습니다. 빛의 입사각, 반사각과 시야의 방향에 따라서 각각의 빛의 세기를 계산합니다. 그 결과 이미지에서 볼 수 있듯이 opengl과 거의 흡사한 형태로 사물들이 렌더링되는 것을 볼 수 있습니다.

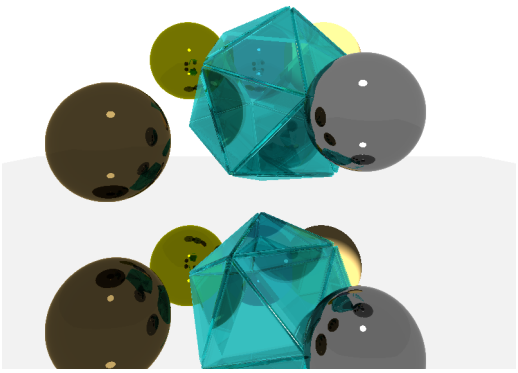
Export Image Files
make run을 통해서 이미지를 그리면 result.ppm으로 이미지가 export됩니다.

Spacial Partitioning
폴리곤들을 더 빠르게 렌더링하기 위해서 BSP Tree를 이용해서 spacial partitioning을 하였습니다. 이를 통해 폴리곤 약 7000개가 포함된 탁자 하나를 100*100 이미지에 렌더링하는데 2분 11초에서 6초로 시간이 단축되는 것을 확인하였습니다.

Extras
reflection과 refraction된 빛의 세기를 계산할 때 fresnel equation을 사용해서 최대한 실제와 비슷한 형태로 그려질 수 있도록 구현했습니다. 물체마다 굴절률을 주어서 같은 모양의 물체여도 다르게 빛이 굴절되는 것을 확인할 수 있습니다. 또, 물체마다 reflection과 transmission 값을 주어서 reflection과 refraction된 빛이 얼마나 강하게 맺힐 것인지 조절할 수 있습니다. 이 값들을 통해 그림자가 투명한 물체에 통과할 때 그림자가 더 약하게 맺히는 등의 특징도 구현할 수 있었습니다.

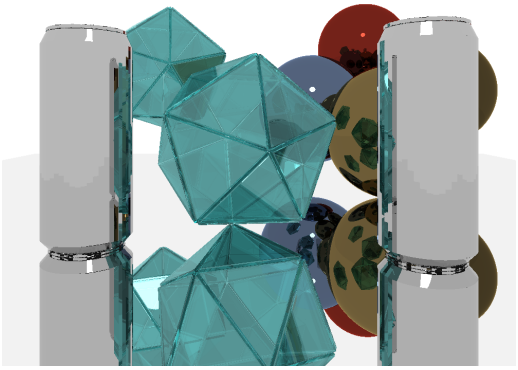
과제를 통해 생성한 이미지들

result1



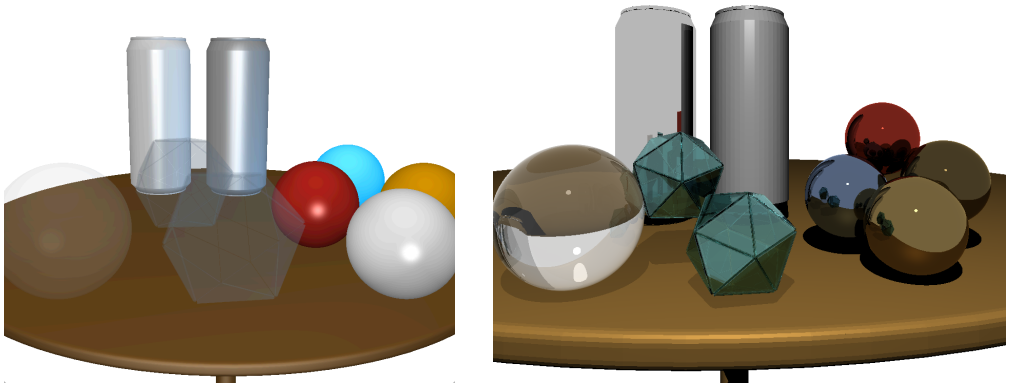
이 이미지를 통해 phong illumination이 적용된 구와 recursive reflection / refraction이 구현된 icosphere와 구를 확인할 수 있습니다. 바닥은 plane하나로 이루어진 폴리곤으로 구성되어 있습니다.

result2



이 이미지를 통해서 polygon이 다수 포함된 scene도 bsp tree를 통한 최적화를 이용해 고화질로 그릴 수 있음을 알 수 있습니다.

result3



opengl로 그린 scene과의 비교입니다. 비교를 통해서 ray tracing을 이용해서 렌더링한 scene이 그림자와 투명한 물체의 reflection과 refraction이 더 사실적으로 묘사되었음을 알 수 있습니다. 그림자가 투명한 물체에서는 더 약하게 맺히는 것도 역시 확인할 수 있습니다. 또한 phong illumination을 통해서 탁자나 캔에 specular light과 diffuse light이 재질에 따라서 다르게 나타나는 것도 확인할 수 있습니다. 왼쪽 구의 굴절률이 높아 상이 거꾸로 맺히는 것도 확인할 수 있습니다.

과제 컴파일 및 실행 방법

과제는 c++로 작성하였고 맥에서 작성하고 실행시켰습니다. makefile은 맥과 리눅스 모두에서 돌아가도록 고쳤습니다. makefile을 이용하여 파일들을 컴파일합니다.

1. make all
2. make run

이 두 명령어를 통해 과제를 실행시킬 수 있습니다. 제출된 코드는 result3의 이미지를 1000*1000로 그리는 코드로 이를 렌더링하는데 맥북 프로에서 1시간 정도 소요됨을 확인했습니다.