Computer Graphics Assignment 4

2019-18499 김준혁

1. 구현 내용

color.py

Color 클래스를 추가하여 r, g, b, a값을 표현할 수 있는 데이터 클래스로 사용하였습니다. 하위 __add__, __sub__, __mul__, __truediv__, __str__를 추가하여 Color 사이 또는 int, float 와의 연산이 가능하도록 하였습니다. 또한 최종적으로 0~255 사이의 값으로 색을 출력할 수 있도록 int 값으로 색깔을 변환해주는 int_color 함수를 추가해주었습니다.

materials.py

Material 클래스를 추가하여 color, ambient, diffuse, specular, shininess, reflectiveness, refractiveness, ior(Index of refraction)을 표현할 수 있는 데이터 클래스로 사용하였습니다.

Material 클래스를 상위로 하여, Plastic (반사, 굴절 없음), Matel (반사 있음, 굴절 없음), Glass (반사, 굴절 있음) 클래스를 추가하였습니다. 값들은 임의로 그럴듯하게 보이도록 지정하였습니다.

ray.py

발사 위치와 방향을 저장하는 ray 클래스를 추가하였습니다. 하위 함수로, 어떤 t일 때의 ray의 위치를 반환하는 point_at_parameter 함수를 추가하였습니다.

shapes.py

Shape 클래스를 추가하여 위치, 기본 색깔, material을 저장하며, 충돌을 체크하기 위한 intersect 함수와 해당 위치에서의 normal을 계산해주는 calculate_normal 함수를 선언하였습니다.

Hit 클래스를 추가하여 충돌한 개체가 있을 경우 충돌 위치, 해당 위치에서의 normal, 개체의 색깔, material을 저장하도록 하였습니다. getDist, getNormal, getDiffuseColor 함수를 추가하여 해당 속성을 가져올 수 있도록 하였습니다.

Light 클래스를 추가하여 빛의 위치, 색깔, intensity를 표현하도록 하였습니다. 이는 프로그램에서 사용되지는 않았습니다.

Shape를 상속받은 Sphere, Cube 클래스를 추가하였습니다. 이름과 같이 구, 큐브를 의미하며, 각 도형에 맞게 추가 parameter 입력과 intersect, calculate_normal 함수를 구현하였습니다. Sphere를 상속받는 SphereLight를 추가하여 구형의 빛을 표현할 수 있도록 하였습니다. 이에는 intensity가 추가 parameter로 들어갑니다.

main.py

전반적으로 슬라이드에서 제공된 pseudo code를 기반으로 구현하였습니다.

TraceRay 함수는 ray, depth, n (IOR)을 입력으로 받아 이에 따라 ray를 발사, 충돌 개체를 찾고, 그림자 체크 또한 하며 빛에 닿을 수 있는 경우에, LocalShade 함수(후술)에 따라 색상을 더해주었습니다. 이후 닿은 개체가 반사 가능한 material일 때 반사된 색 값(재귀적)을 추가, 굴절 가능의 경우, ior 값에 따라 굴절 방향을 구하여 굴절되어 구한 색의 값을 추가하도록 하였습니다. 이들을 모두 포함한 색 값을 반환합니다.

LocalShade 함수는 phong illumination model에 따라 해당 material의 속성을 이용하여 색의 값을 반환하는 함수입니다.

CheckIntersectionAll 함수는 빛을 포함한 모든 개체의 충돌을 체크하는 함수입니다.

CheckIntersectionAllShadow 함수는 그림자 표현을 위해 빛을 포함한 모든 개체의 충돌을 체크하면서, 'Glass' material의 경우는 무시하도록 하여 그림자가 표시되지 않도록 하였습니다.

AddShape 함수는 물체를 추가하는 함수입니다.

AddLights 함수에서는 빛을 추가합니다.

최종적으로 출력할 그림의 너비, 높이, fov 값을 지정하여 이에 따라 ray를 발사, 해당 픽셀의 색을 구하고, 최종 저장하여 output.png 로 나오도록 하였습니다.

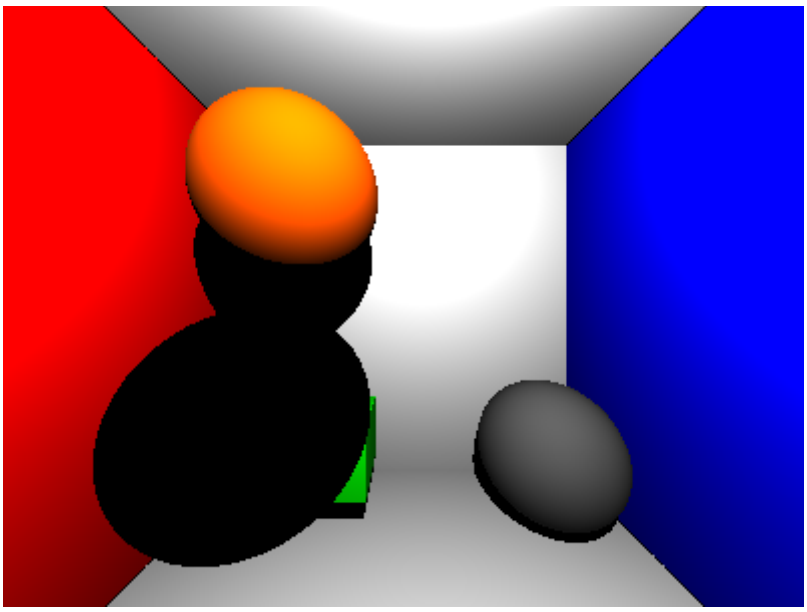
2. 실행 방법

기존 과제와 비슷하게 python3 main.py로 실행합니다.

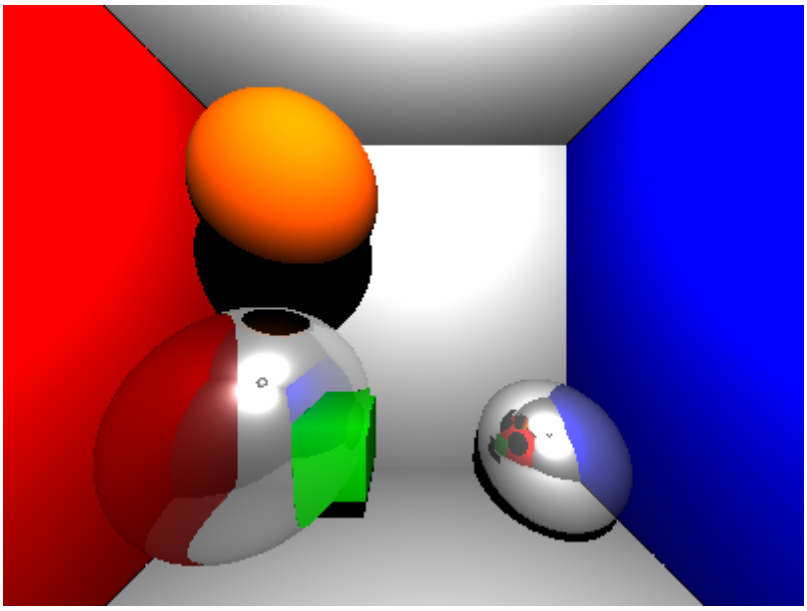
빛이나 개체를 수정할 경우, AddShape, AddLights 함수에서 수정합니다.

3. 실행 결과

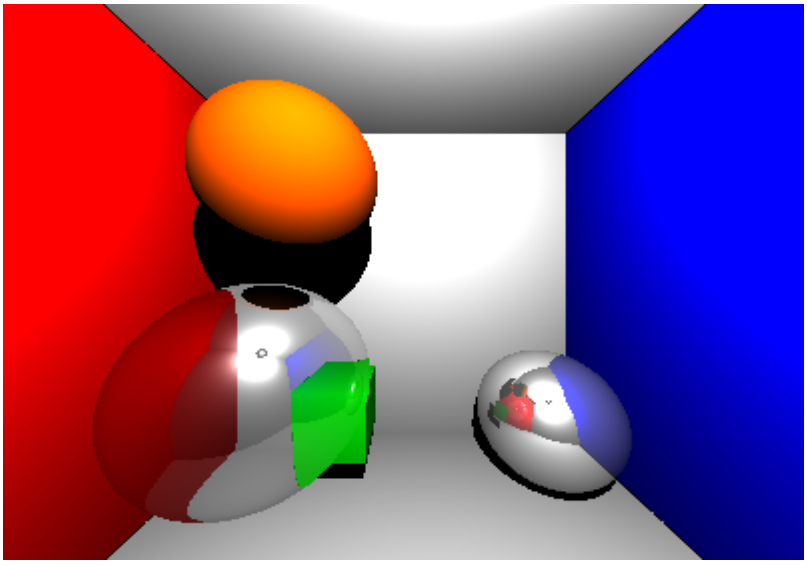
기본적으로 좌, 우벽은 빨강, 파랑의 색을 가지며, 나머지 벽은 하얀색으로 하였습니다. 왼쪽 위 개체는 주황색 Plastic 구, 우하단 회색 Metal 구, 좌하단 Glass 구가 있으며, Glass 구 뒷편에 녹색 Cube를 하나 두어 반사 및 굴절이 표현되는지 확인할 수 있도록 하였습니다.



1. Depth = 1



2. Depth = 2



3. Depth = 4

Depth가 1인 경우, 반사와 굴절된 빛이 계산되지 않아 Metal, Glass 물질은 LocalShade에 의한 결과만 반영되어 각각 회색, 검정으로만 표현되었습니다.

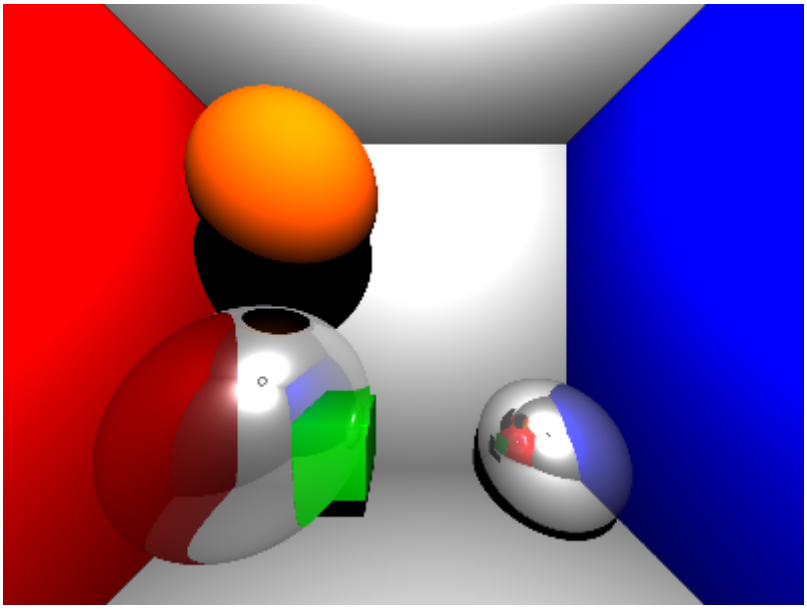
Depth가 2일 때 반사 및 굴절이 표현되는 것을 확인할 수 있었습니다. 하지만, Metal에 반사된 Glass 구가 검정으로 표현되는 등, 여러 번의 빛이 부딪혀 나아가는 것은 표현되지 않았습니다.

Depth가 4일 때, 반사되어 보인 Glass 구를 통해 빨간 벽이 표현되는 것을 확인할 수 있었습니다.

Depth를 상승시킬 때 이렇게 반사/굴절되어 나아간 빛의 표현이 잘 되는 것을 확인하였습니다.

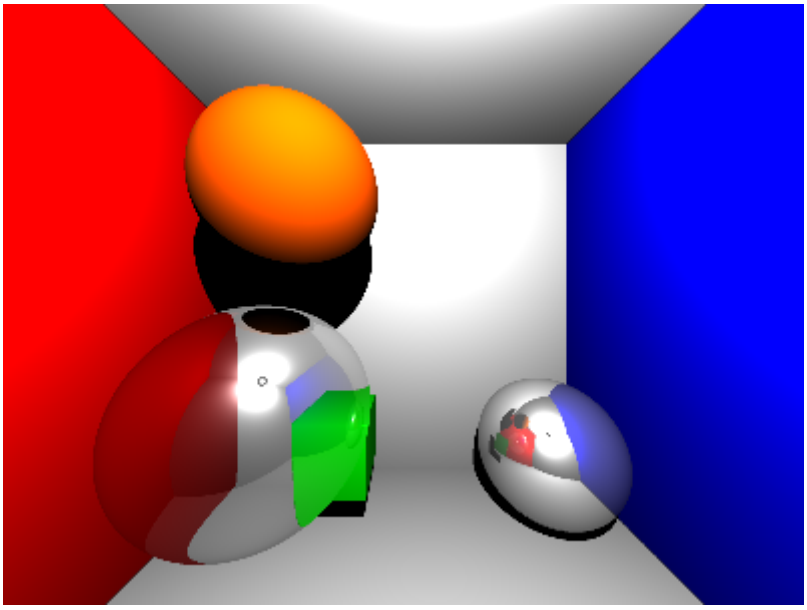
4 ray per pixel

본래 ray 1개만 이용할 때를 기준으로 대각선 4방향으로 총 4개의 ray를 발사하도록 하였을 때의 결과는 아래와 같습니다. Depth = 4일 때의 결과입니다. 1 ray per pixel일 때에 비해 약간의 Aliasing이 줄어드는 효과가 있는 것으로 판단됩니다.



4. Depth = 4, 4 rays per pixel

Result for Artistic score (Track 1)



Depth = 6, 4 rays per pixel

Depth = 6, 각 픽셀당 4개의 ray를 발사했을 때의 결과입니다.