Graphics HW5 Report

컴퓨터공학부 2013-11425 이창영

1. 사용 언어

Python 3.6

2. 사용 라이브러리

Pillow

3. 구현한 것

- Ray tracing spheres
- Ray tracing polygons
- Export image files
- Texture mapped shperes and polygons
- Phong illumination
- Recursive reflection
- Representative pictures
- Recursive refraction (extra)

4. 구현하지 못한 것

- Importing geometry files such as OBJ
- extra features..

5. 구현 내용, 방법

1) 객체 생성

(1) Vector

- 벡터를 나타내는 객체
- methods
  dot - dot product
  cross - cross product
  magnitude - 크기
  normal - 자신의 normalized 벡터
  add - 더하기
  sub - 빼기
  mul - scarlar 곱하기

(2) Ray

- 빛을 나나태는 객체
- 빛을 줄발점과 방향 벡터를 가지고 있다.

(3) Intersection
- 어떤 물체과 빛의 교점을 나타내는 객체
- 물체로부터의 거리로 표현한다.
- 그 점과 점에서는 normal 벡터를 가지고 있다.

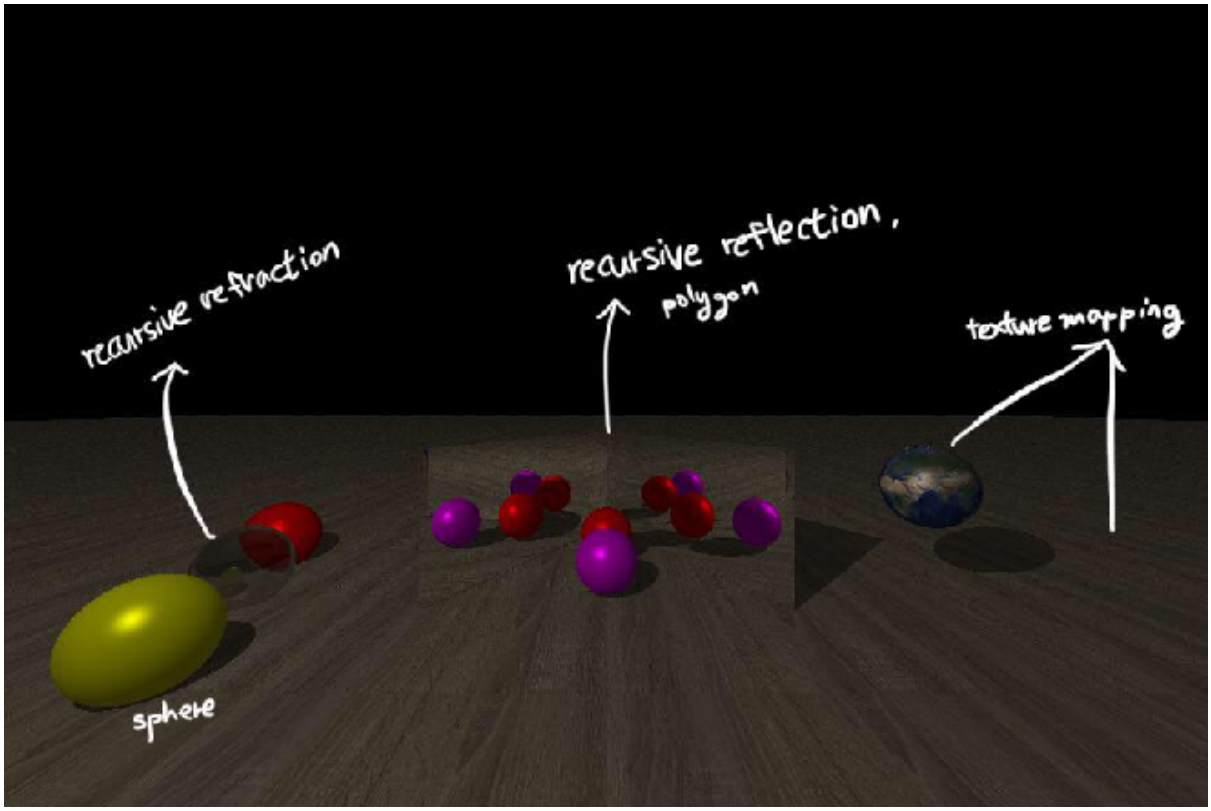
(4) Sphere
- 구를 나타내는 객체
- 구의 중심과 반지름을 가지고 있다.
- 색 또는 texture 정보를 가지고 있다.
- 'default', 'reflection', 'reflection_refraction' 세 가지 타입 중 하나이다.
- methods
  normal - 표면 위의 한 점에서의 normal 벡터
  getIntersection - Ray와 Sphere의 Intersection을 구한다.
  getColor - 표면에서의 기본 색을 구한다. 구가 색을 가지고 있을 경우에는 그 색을 그대로 return하고 texture을 가지고 있는 경우에는 texture이미지에서 mapping된 색 값을 return한다.

(5) Plane
- 평면을 나타내는 객체
- 한 점과 noraml 벡터로 이루어져 있다.
- 색 또는 texture 정보를 가지고 있다.
- 'default', 'reflection', 'reflection_refraction' 세 가지 타입 중 하나이다.
- polygon을 표현하기 위해서 x, y, z의 최솟값과 최댓값을 가지고 있다.
- methods
  normal - 표면 위의 한 점에서의 normal 벡터
  getIntersection - Ray와 Sphere의 Intersection을 구한다.
  getColor - 표면에서의 기본 색을 구한다. 구가 색을 가지고 있을 경우에는 그 색을 그대로 return하고 texture을 가지고 있는 경우에는 texture이미지에서 mapping된 색 값을 return한다.

2) 전체 알고리즘
(1) origin은 camera이고 pixel로 향하는 Ray를 생성한다.
(2) 모든 object에 대해 그 Ray와 Intersection이 있는지 확인한다.
(3) Intersection이 없다면 backgroundColor * ambient한 값이 그 색상이다.
(4) Intersection의 object가 'default' 타입이라면 그 지점으로부터 light source까지 다시 Ray를 만들어서 교점이 있다면 그 지점은 자신의 색 * ambient이고 교점이 없다면 그 점에서의 phone illumination식의 결과가 그 점의 색상이다.
(5) Intersection의 object가 'reflection' 타입이라면 그 지점에서 반사하는 방향으로 다시 Ray를 만든 뒤 (3)부터 다시 반복해서 얻어진 색상이다.

(6) Intersection의 object가 'reflection_refraction' 타입이라면 반사하는 방향, 굴절되는 방향으로 2개의 Ray를 만든다. 두 Ray를 가지고 (3)부터 다시 반복 한 뒤 그로부터 얻어진 색상을 굴절 0.7, 반사 0.3 만큼 더해서 색을 얻는다.
(7) recursive가 10회 이상 반복되면 recursive를 멈추고 background color * ambient 로 return하도록 했다.
(7) 이미지 라이브러리를 사용하여 얻어진 색들을 채워넣어 이미지 파일로 export한다.

3) 구현 내용 확인



(1) ray tracing spheres
그림에 존재하는 구 6개

(2) ray tracing polygons
그림에 존재하는 거울 2개 및 바닥을 나타내는 평면

(3) export image files
제출한 이미지

(4) texture mapped spheres and polygons
바닥 평면, 및 지구본 모양의 구
- 구현 방법
미리 texuter로 사용할 이미지 파일을 준비한 뒤 x, y, z coord 상에 있는 평면을 u, v 평면으로 옮긴 뒤 각 좌표를 준비한 이미지 크기에 대해 mod 연산하여 해당되는 픽셀의 색을 그대로 가져온

다.
평면인 경우 (x, y, z) -> (u, v)로 옮기는 식은 아래와 같다
normal = 평면의 noraml 벡터
width, height = 이미지의 크기
uV = (normal.y, normal.x, 0)
vV = uV X noraml
u = uV • (x,y,z) % width
v = vV • (x,y,z) % height
구인 경우 (x,y,z) -> (u, v)로 옮기는 식은 아래와 같다
width, height = 이미지의 크기
direction = center - (x,y,z)
uV = 0.5 + arctan2(direction.z, direction.x) / 2 * PI
vV = 0.5 - arcsin(direction.y) / PI
u = u * width
v = (1 - v) * height

(5) phong illumination
object의 타입이 'default'인 경우 모두 phong illumination 식을 이용하여 색상을 구하였다.

(6) recursive reflection
그림의 가운데 2개의 거울이 recursive reflection을 나타내고 있다.

(7) recursive refraction
왼쪽 가운데 있는 구는 유리구슬을 표현한 것으로 recursive refraction을 나타내고 있다.