Reflection in a Ray Tracer

Mathematics

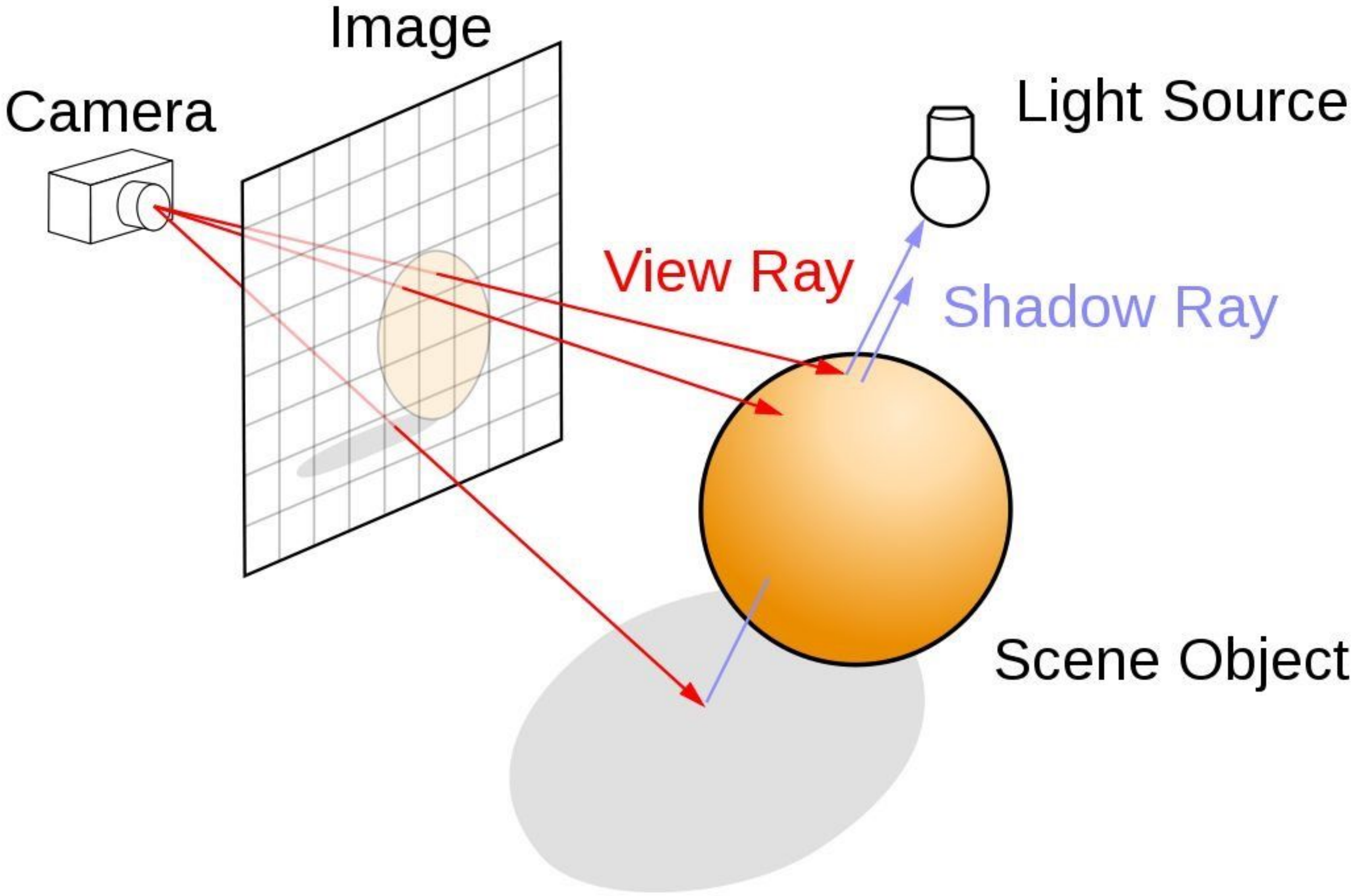
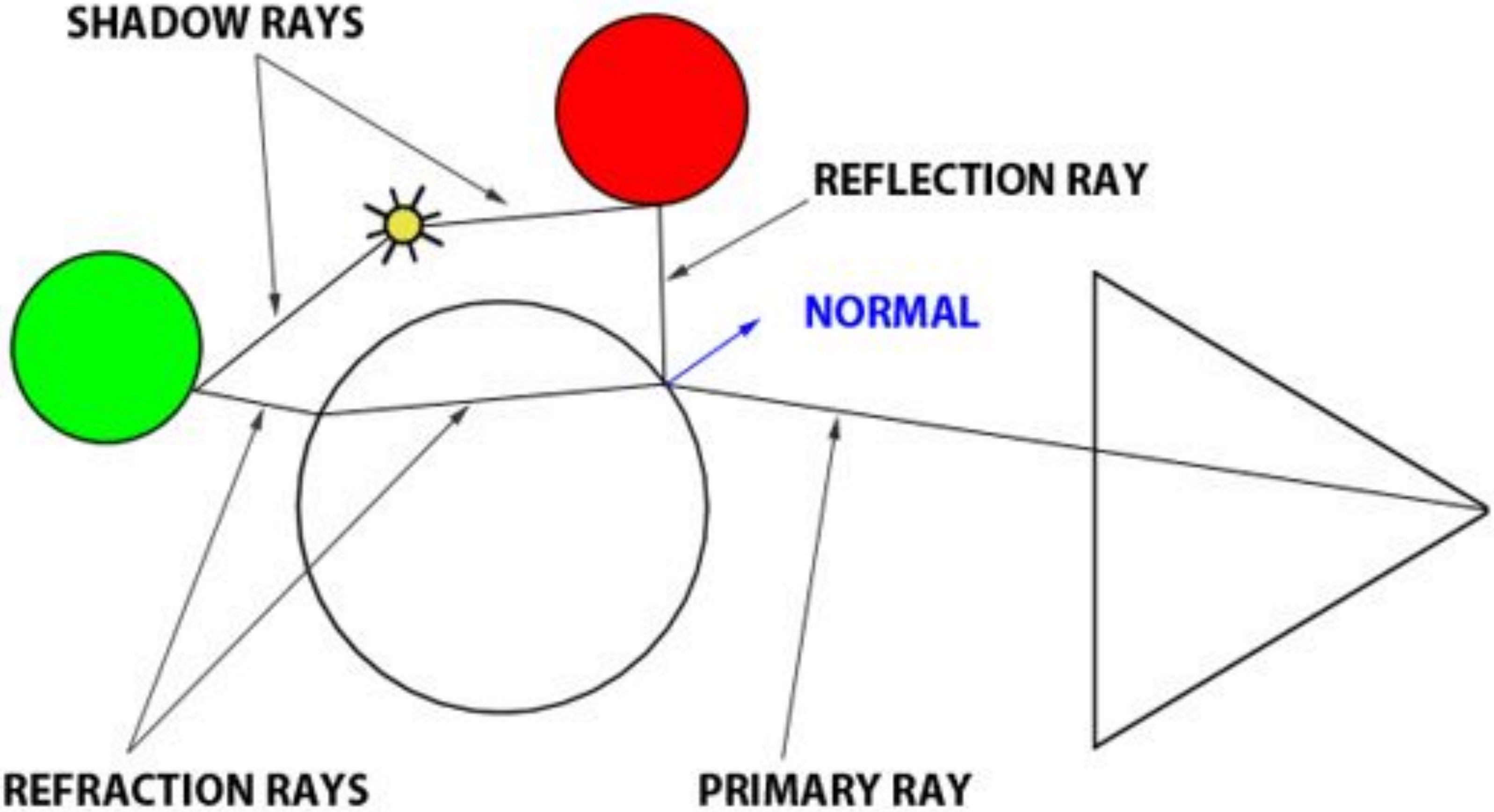


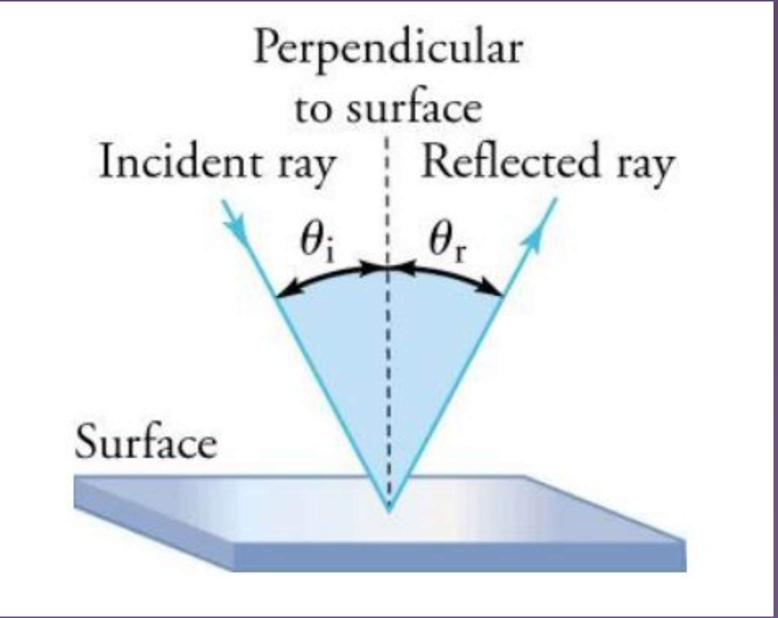
Image
Camera
Light Source
View Ray
Shadow Ray
Scene Object



SHADOW RAYS
REFLECTION RAY
NORMAL
REFRACTION RAYS
PRIMARY RAY

Reflection

$$R = 2(\hat{N} \cdot \hat{L})\hat{N} - \hat{L}$$



Breakdown

- Modify the traceForColor function from RT5
- Call function recursively
  - (scene, origin, directions)
  - Intersection with elements
  - Ambient, diffuse, specular
  - Shadows

Step by Step

STEP 1

traceForColor

Step 2

```
reflectionColors = np.zeros([3, nPixels])  # Initialize reflection color contribution

# Check if the surface is reflective
reflectiveMask = specularCoefficients[objects] > 0.0
```

- reflectionColors: contribution of reflected light for each pixel
- Specular coefficients are inputs in each scene

Step 3

```
# Check if there are reflections to compute
if np.any(reflectiveMask) and max_depth > 0:
    # Calculate reflected rays
    reflectDirs = normalize(reflections[:, reflectiveMask])
    reflectOrigins = hits[:, reflectiveMask] + 0.001 * normals[:, reflectiveMask]

    # Recursive call for reflected rays
    reflectionColors[:, reflectiveMask] = traceForColor(scene, reflectOrigins, reflectDirs, max_depth - 1)
```

- Recursive call of traceForColor
- Repeats max_depth times

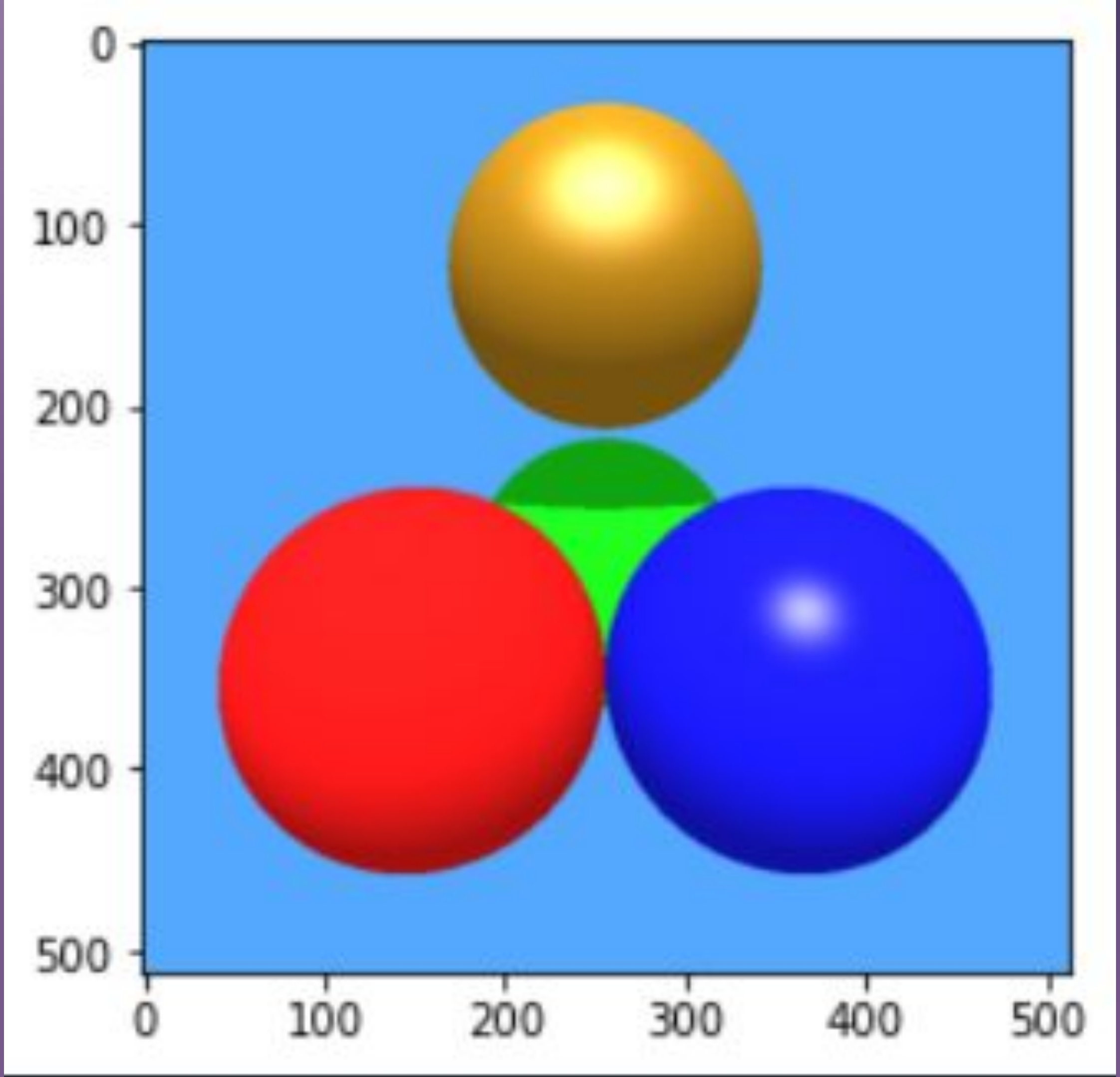
Step 4

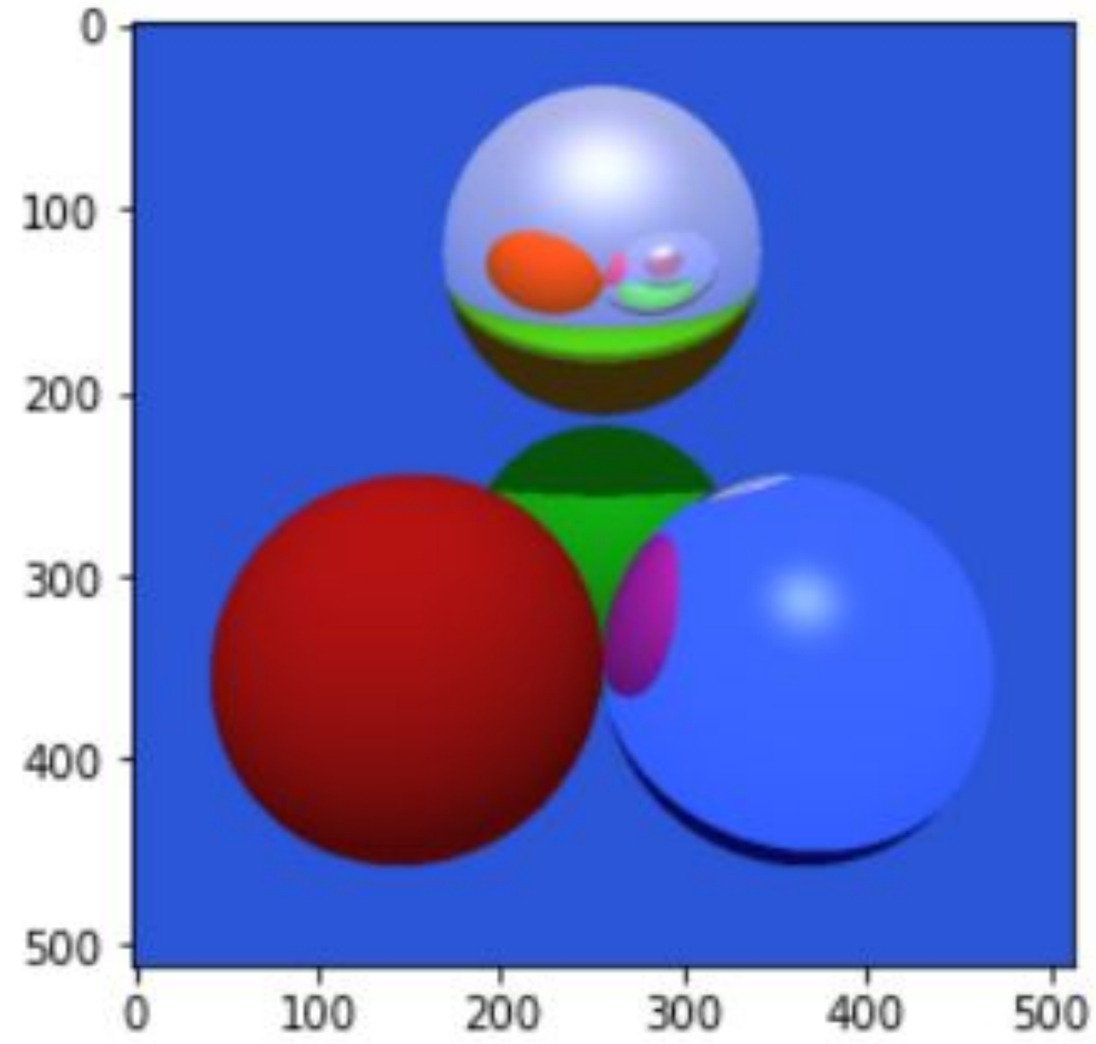
```
# Combine diffuse, specular, and reflection colors
colors[:, litMask] += (irradiance * lightCosines * diffuseColors[:, objects])[:, litMask]
colors[:, litMask] += (irradiance * specularCoefficients[objects] * phongCosines ** shineExponents[objects])[:, litMask]
colors[:, litMask] += reflectionColors[:, litMask]

return colors
```

- Adds contribution of diffuse, specular, and reflection colors to each lit pixel

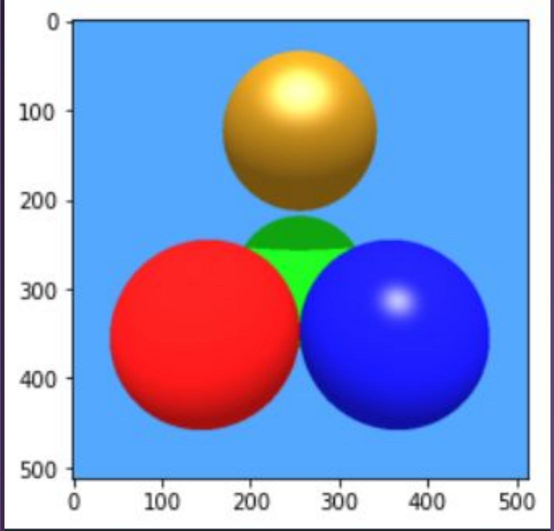
Results and Analysis


0
100
200
300
400
500
0
100
200
300
400
500

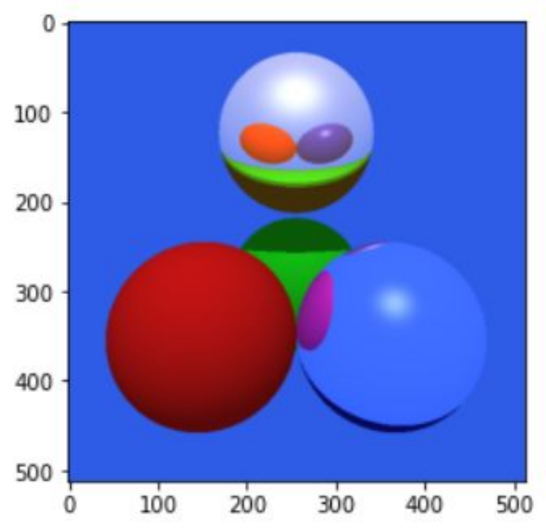

0
100
200
300
400
500
0
100
200
300
400
500

max_depth

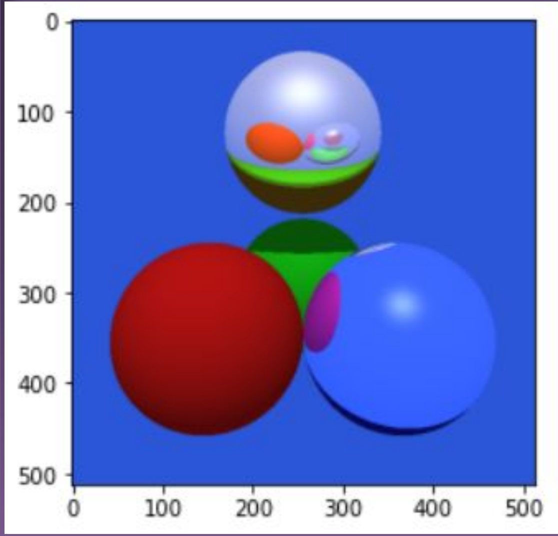
=0



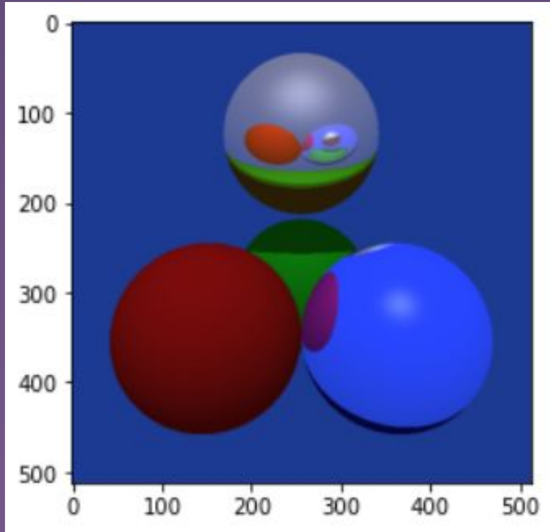
=1



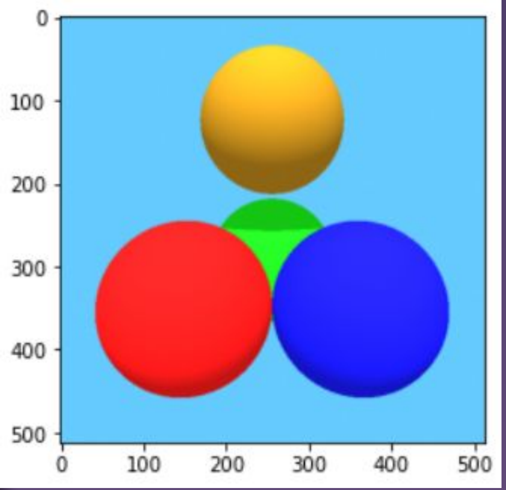
=2



=10

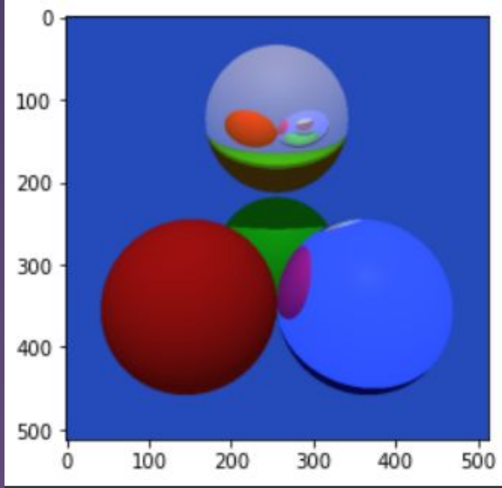


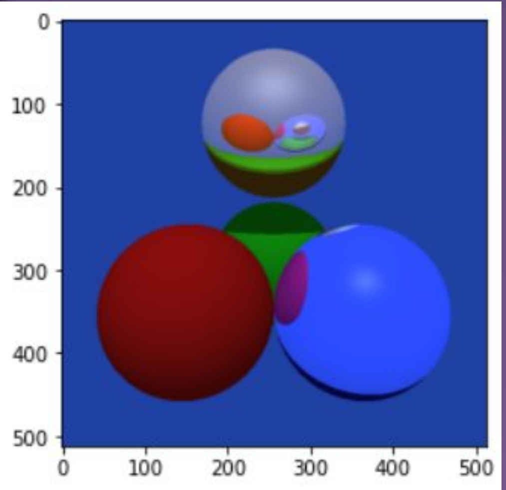
Specular Coefficients
max_depth=3



=0.0

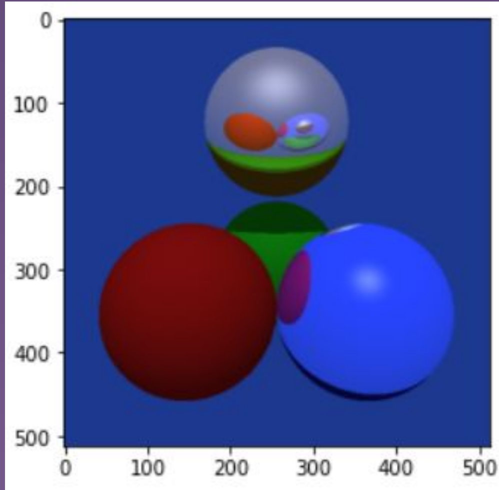
=0.1



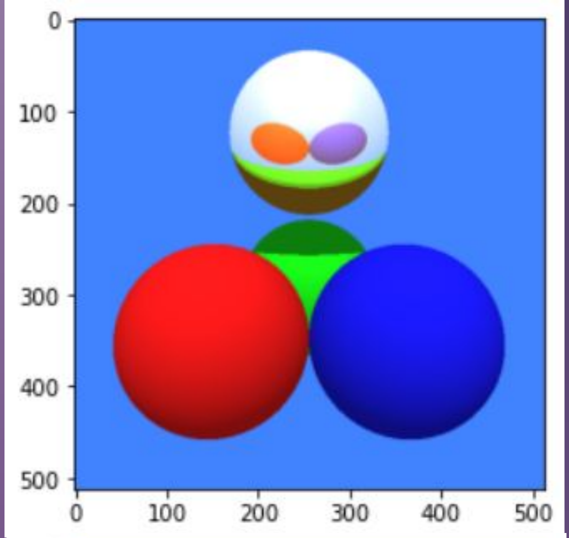


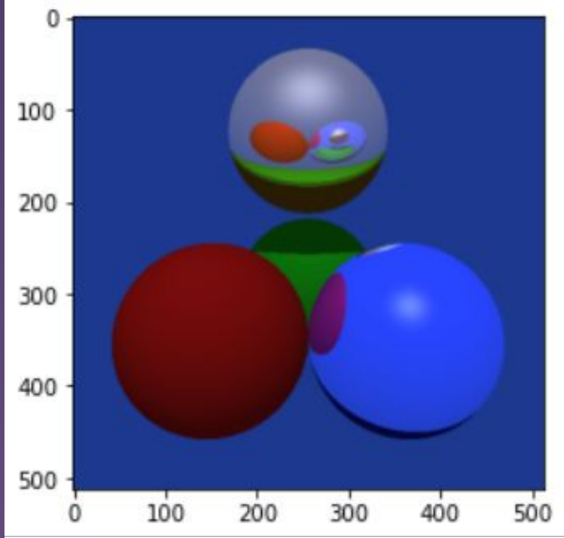
=0.5

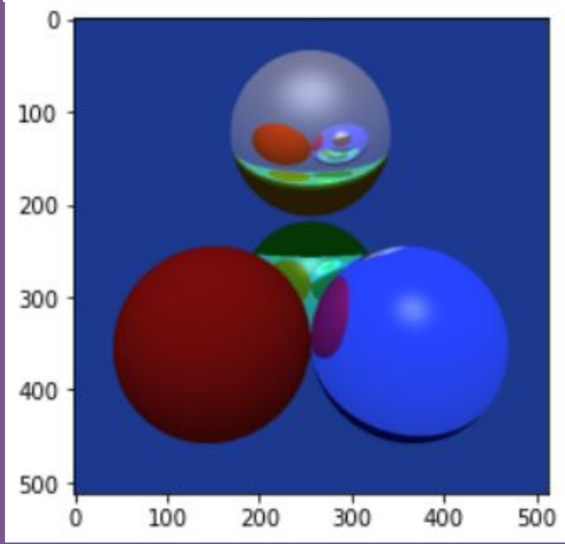
=1.0

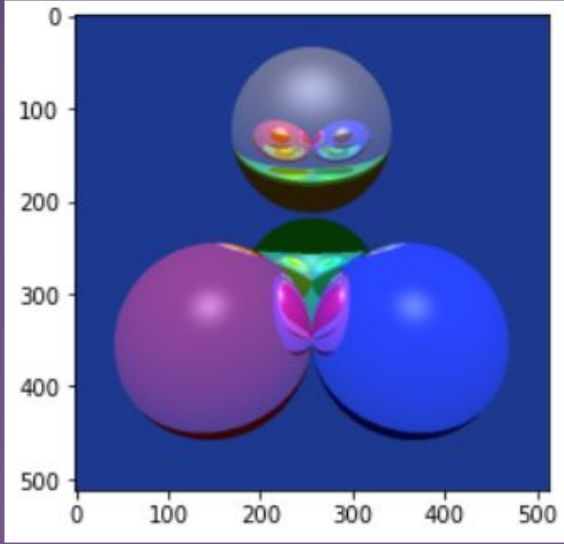


Specular Coefficients Cont. (=1.0)

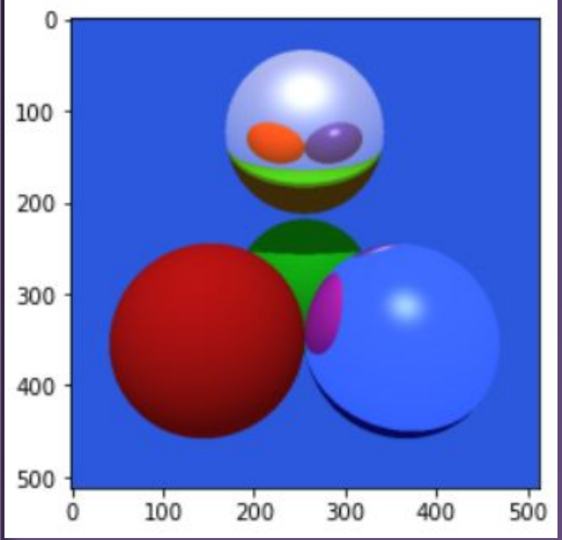


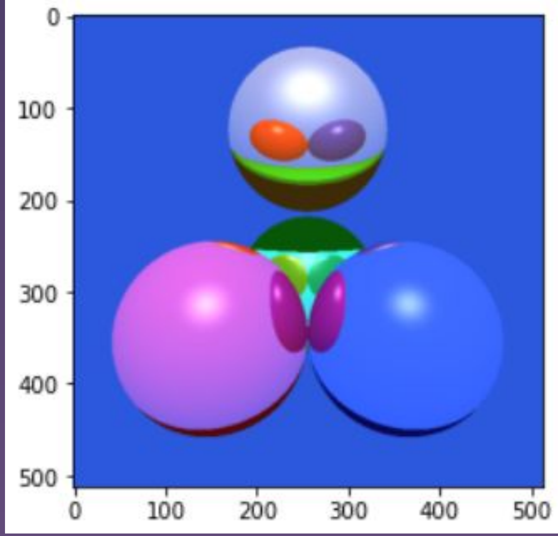






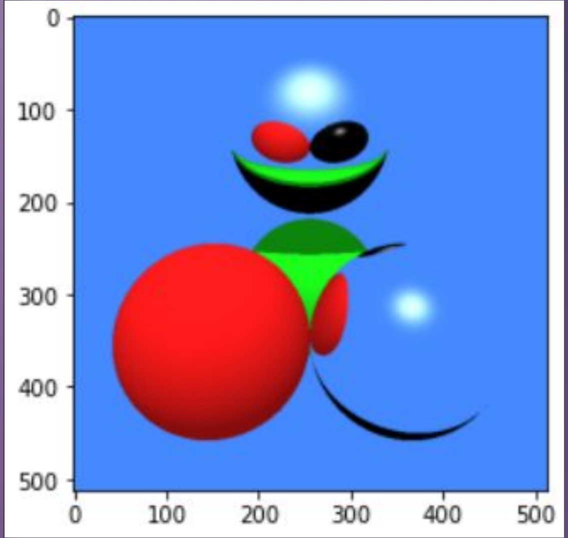
Color Mixing

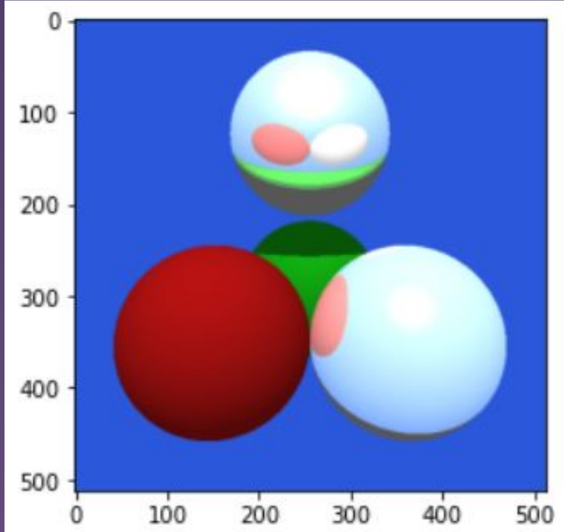




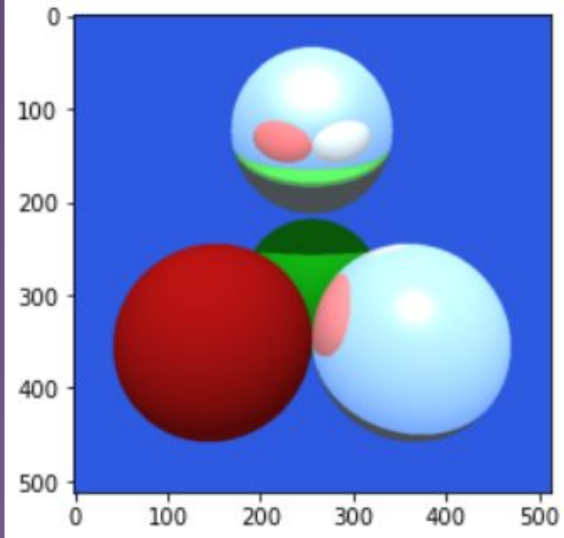
- Combining color contributions from diffuse and specular reflection with reflection colors

Color Mixing Cont.





- Most mirrors are white with a green tinge
- Black (top left)
- White (top right)
- “Mirror color”(bottom)
  - 0.9, 1.0, 0.9



Comparing with Turner Whitted

Phong Shading

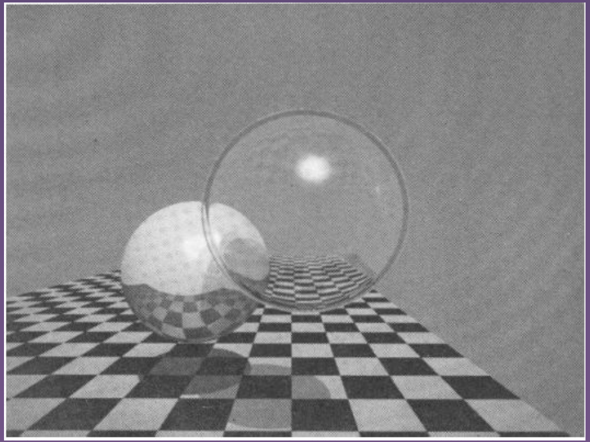
- Whitted, J. T. (1980). An improved illumination model for shaded display. *Communications of the ACM*, 23(6), 342-349.

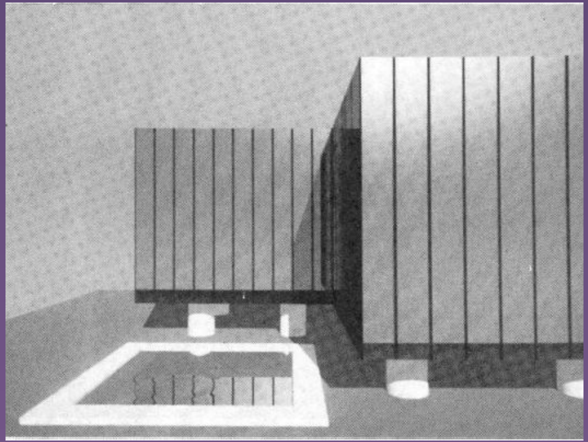
$$I = I_a + k_d \sum_{j=1}^{j=ls} (\bar{N} \cdot \bar{L}_j) + k_s S + k_t T$$

$$I = K_d I_a + K_d I_{in} \, (\vec{N} \cdot \vec{L}) + (\vec{V} \cdot \vec{R})^e \, K_s I_{in}$$

In Use

This illumination model draws heavily on techniques derived previously by Phong [8] and Blinn [3–5], but it operates recursively to allow the use of global illumination information. The approach used and the results achieved are similar to those presented by Kay [16].





Questions / Discussion

Sources

https://www.texasgateway.org/resource/161-reflection

https://omaraflak.medium.com/ray-tracing-from-scratch-in-python-41670e6a96f9

https://dl.acm.org/doi/pdf/10.1145/358876.358882