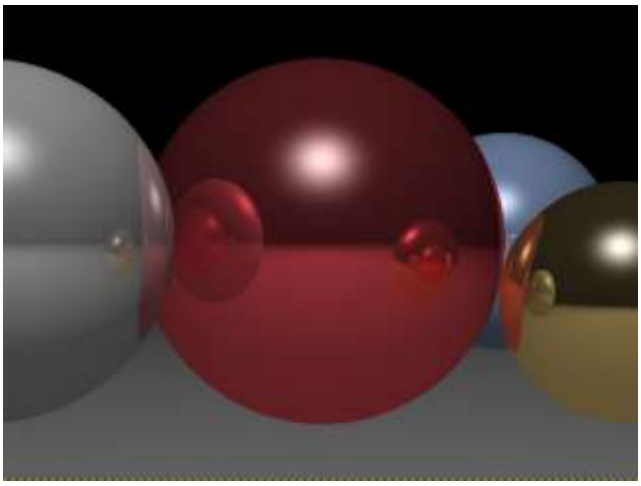
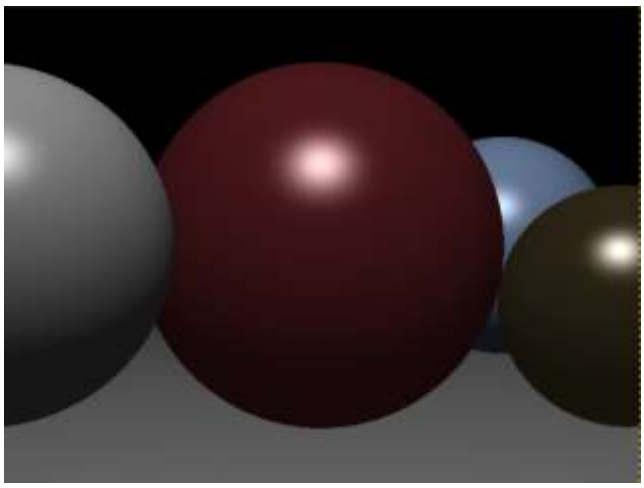
Advanced Graphics Programming

Workshop Five – Ray Tracing (Recursive Reflection)

This workshop involves implementing reflections as illustrated below:



Before you start this workshop you should already have already completed the Phong shading in workshop 2. In that workshop you added ambient, diffuse and specular light. The ambient intensity $\mathbf{i_a}$ is constant for the scene and the diffuse coefficient $\mathbf{k_d}$ and specular coefficient $\mathbf{k_s}$ can be set for each sphere to create dull or glossy spheres, so you should start with something similar to the image below (for this workshop I have changed the background to black and lightened the large grey sphere that is acting as the ground plane):



```
// position, radius, surface colour,
specular power (p), specular
coefficient (ks), diffuse coefficient
(kd)
// Ground sphere
(0, -10004, -20), 10000, (0.4, 0.4,
0.4),0, 0, 1
// Red sphere
(0, 0, -20), 4, (1.00, 0.32, 0.36),20,
0.8, 0.2
// Yellow sphere
(5, -1, -15), 2, (0.90, 0.76, 0.46),
20, 0.9, 0.1
// Blue sphere
(5, 0, -25), 3, (0.65, 0.77, 0.97),
20, 0.5, 0.5
// Gray sphere
(-5.5, 0, -15), 3, (0.90, 0.90,
0.90),20,0.5, 0.5

// Ambient (ia)
0.1
```

The recursive reflection algorithm uses a similar reflection ray $\mathbf{r}$ to the one you calculated in workshop 3 (step 4) but this time you are reflecting the primary ray $\mathbf{v}$.

$$\mathbf{v}' = \mathbf{v} - 2 * (\hat{\mathbf{v}} \cdot \hat{\mathbf{n}}) * \hat{\mathbf{n}}$$

And $\mathbf{n}$ is the normal at the intersection point $\mathbf{p}$ of your sphere with centre $\mathbf{c}$:

$$\mathbf{n} = \mathbf{p} - \mathbf{c}$$

The reflection ray origin $\mathbf{p}'$ is the point of intersection but you need to add a small amount to it for the same reason you did for shadow rays, so that the reflection ray does not hit the sphere that generated it.

$$\mathbf{p}' = \mathbf{p} + \mathbf{n} * \varepsilon$$

Where $\varepsilon =$ 1e-4.

In the real world, some energy is lost during reflection, this can be implemented by adding a recursive call in your ray tracing function and multiplying it by the specular coefficient $\mathbf{k_s}$ of your sphere:

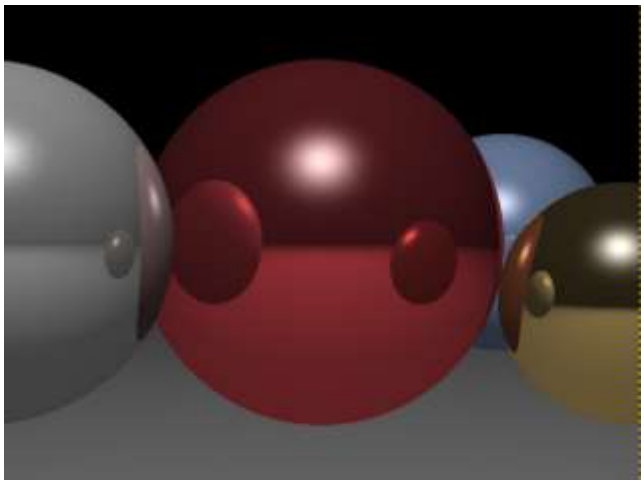
$$\mathbf{c}' = \mathbf{c} + \mathbf{k_s} * \text{trace}\,(\mathbf{p}', \mathbf{v}', d + 1\,)$$

Where d is a new variable you will need to make sure that you do not keep making recursive calls forever, remember to check if this exceeds your MAX_DEPTH near the start of the trace function. MAX_DEPTH is a fixed value you set to represent the maximum number of reflections.

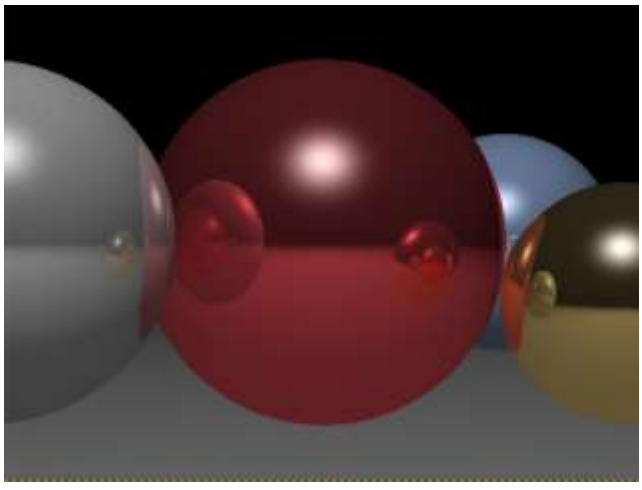
For efficiency only trace the reflective ray if the specular coefficient $\mathbf{k_s}$ is greater than zero.

The basic idea is that reflection is calculated recursively. So you need to restructure your existing method(s) to enable this. The trace method (or whatever you have called it) represents the method that takes a ray and finds the colour of the pixel at the point of intersection. Your trace method should already do the calculations for diffuse, specular and ambient light. To make this function recursive, it needs to call itself probably using something similar to the line of pseudo code above somewhere near the end of the trace function. This line of pseudo-code illustrates the concept that you call the trace function again but this time with the reflected ray $\mathbf{v}'$.

Important: you must set a maximum depth for the recursion otherwise your algorithm will never terminate.



1 reflection ray



5 reflection rays

1 reflection ray

5 reflection rays