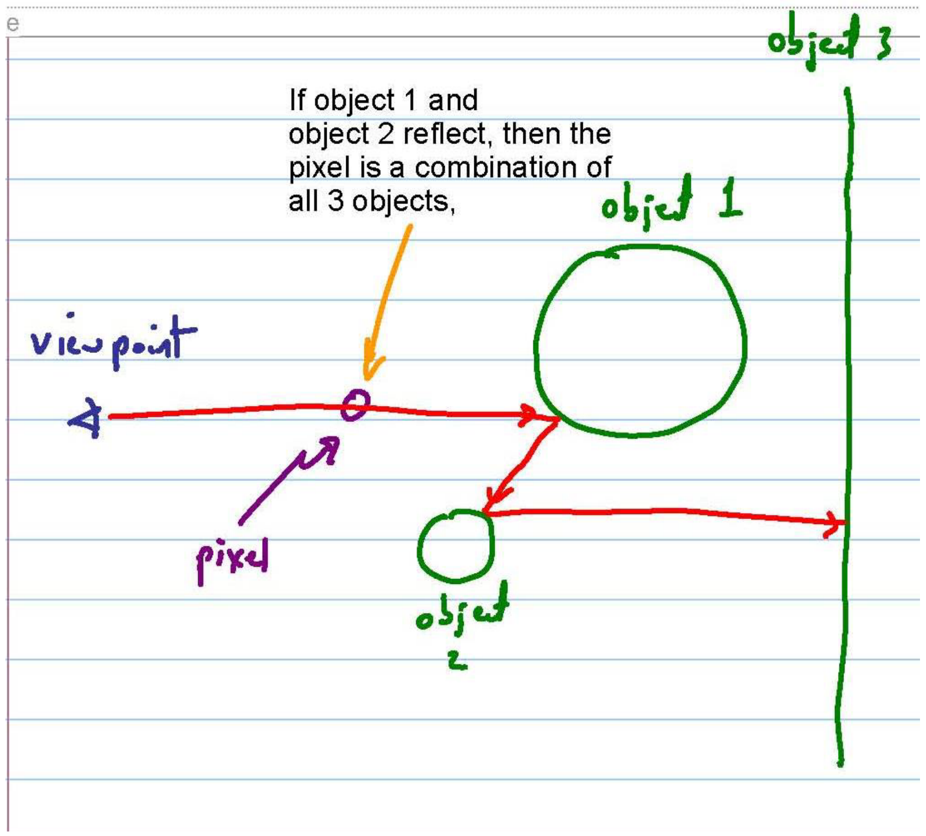
Reflections (specular lighting)

Another property of an object is whether its surface is reflective or not. In the material structure in addition to ambient and diffuse lighting values, there is an additional set of values for specular lighting. If these values are all 0, then the surface is not reflective. A non-zero value would indicate that that color is reflected.

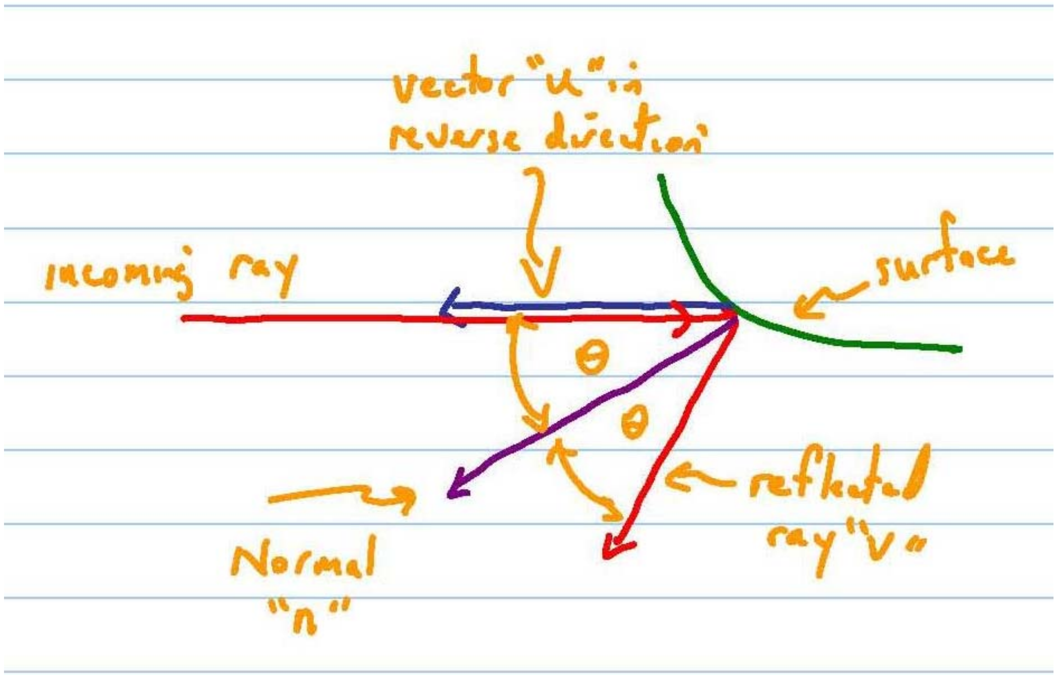
Assuming an object's surface is reflective, the determination of what it reflects is surprising simple.

Consider the following picture:


e
object 3
If object 1 and object 2 reflect, then the pixel is a combination of all 3 objects,
object 1
viewpoint
pixel
object 2

In looking at the ray between object 1 and object 2, note that this is really the equivalent of "what do we see from object 2 if we shoot a ray from the hit point on object 1 along a vector that follows the angle of reflection from object 1's surface". If we can determine the vector (i.e. the red line above between object 1 and object 2), then we pretty much know how to do this – this is just a matter of recursively calling "raytrace()", but in this case using the hit location on object 1 as the new base (note that up until now we've called raytrace() with the viewpoint as the base).

So how do we determine the vector from the hit point to the next object? Consider the following diagram:



Remember we know the normal to the surface at the hit point. The angle theta between the normal and the incoming ray is also the angle between the reflected ray and the normal. In the following let "u" be a unit vector in the OPPOSITE direction of the incoming ray (i.e. if "r" is the incoming ray, then "u = -r"), "v" be a unit vector in the direction of the reflected ray, and "n" be a unit vector in the direction of the normal.

Then the vector u + v is in the same direction as the normal. Furthermore

$$(u + v)/2 = n \cos(\Theta)$$

where

$$\cos(\Theta) = u \text{ dot } n \quad (\text{or } v \text{ dot } n).$$

In other words,

$$u + v = 2\,(u \text{ dot } n)\, n.$$

So

$$v = 2\,(u \text{ dot } n)\, n - u.$$

So "v" is the vector we want to follow from the hit point on the object. If the ray hits another object (find_closest_obj()), then that object also contributes to the pixel's color. If the reflected ray does not hit anything, then the pixel is not affected by reflections.

Note that the process repeats all over again if the next object hit also reflects.

So the algorithm raytrace() now becomes:

```
raytrace() {
        Test each no-light object to see if it is hit by the ray and set "closest" to point to
                the nearest object hit;
        If closest is NULL return;
        Add the distance from the base of the ray to the hit point to total_dist;
        Add the ambient reflectivity of the object to the intensity;
        Add the diffuse reflectivity of the object at the hit point to the intensity;
        Scale the intensity by 1/total_dist;
        If  one or more specular values for the object is non-zero {
                Determine the vector "v" for the reflected ray;
                Call raytrace() using the hit point as the new base and "v" as the direction;
                Multiply the returned intensity by the specular values for the original
                        object;
                Add the result to the intensity
        }
}
```

Notes:

1. when raytrace() is called for specular lighting, total distance should reflect the total distance up to this point (i.e. use the current value of total distance, don't call raytrace() with zero this time),
2. you will need to make use of the "last_hit" parameter to raytrace() in this case and find_closest_obj(). Your find_closest_obj() function should ignore the object pointed to by last_hit (otherwise we will find the "closest object" is the object that contains the hit point – which isn't very interesting).
3. Be sure that "u", "v" and "n" are all unit vectors.