Specular lighting

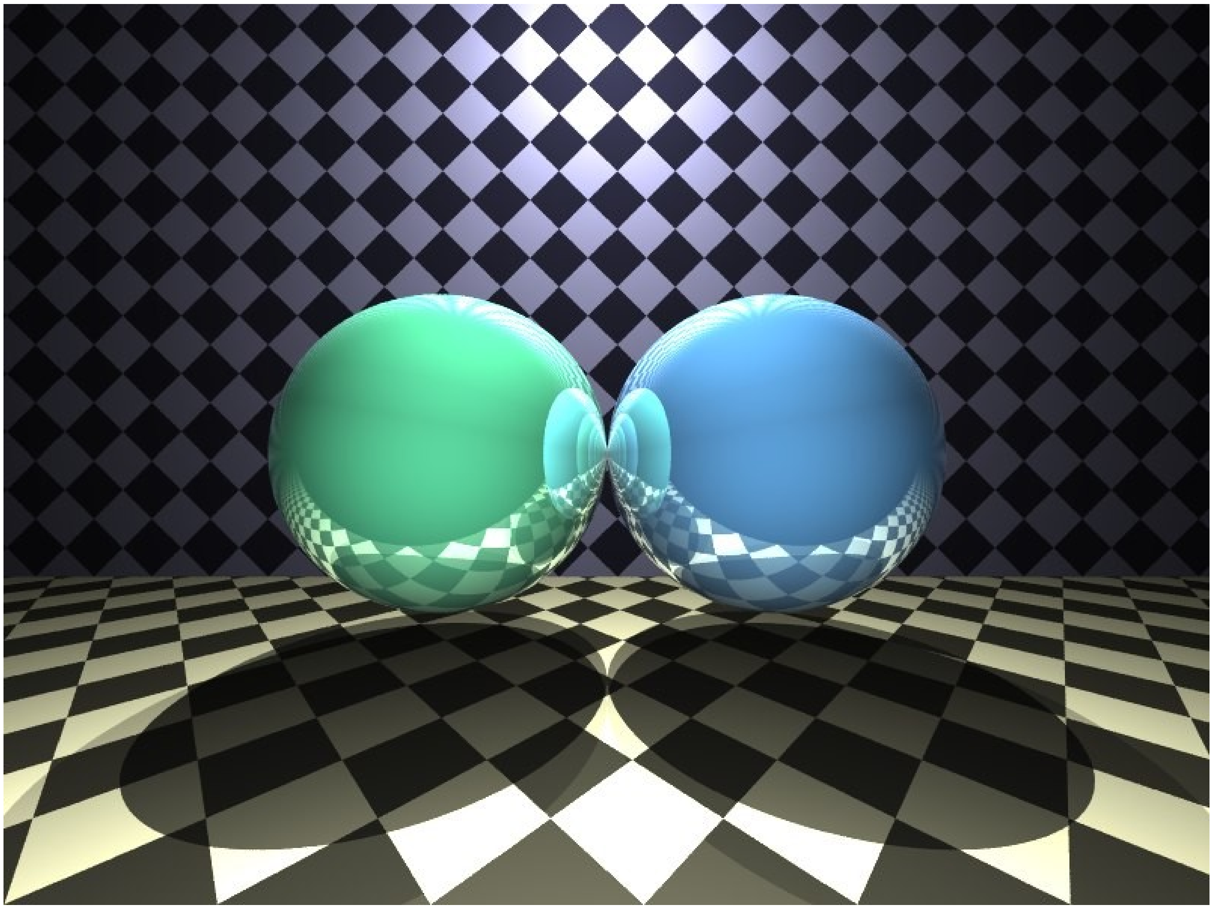
Specular light is which is coherently reflected *without scattering*. The best example of an object with no ambient or diffuse reflectivity but high specular reflectivity is a mirror.

When you look into a mirror, what you see is the reflection of light that has previously been reflected or emitted by other objects.

Therefore in a raytracing system, if a ray hits an object with a non-zero specular reflectivity it is necessary to reflect or *bounce* the ray to see what it hits next. If that object also has a non-zero specular reflectivity it is necessary to bounce the ray again.

This process continues until the bounced ray:

hits no object
hits an object with no specular reflectivity.
travels so far that the effect of further bounces is negligible

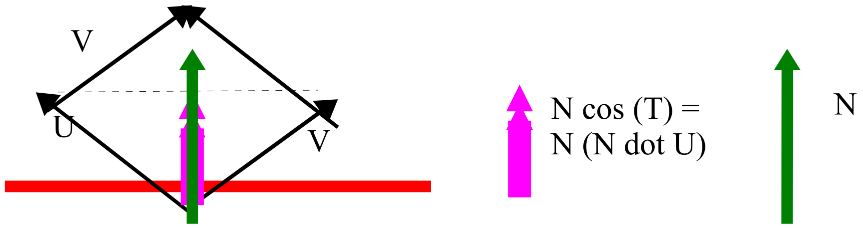


Reflecting a ray

Basic physics says: The angle of incidence (the angle the incoming ray makes with the normal at the hitpoint) is equal to the angle of reflection

```
/*
 * <SNIP>
 * unitin – unit vector in the incoming direction
 * unitnorm – outward surface normal
 * unitout – unit vector in direction of bounce
 * (veclib3d)
 */
```

*reflect3(double *unitin, double *unitnorm, double *unitout);*



Let

$U = -\mathit{unitin}$

$N = \mathit{unitnorm}$

Then

$U + V = 2\ N \cos(T)$ where T is the angle between U and N

$\cos(T) = U \text{ dot } N$

so

$U + V = 2\ N\ (U \text{ dot } N)$

and

$V = 2\ N\ (U \text{ dot } N) - U$

The updated raytrace function:

```
/*
 * <SNIP>
 *        lst
 *        *base          location of projer or previous hit
 *        *dir           unit vector in direction of object
 *        *intensity     intensity return location
 *        *total_dist    distance ray has traveled so far
 *        *last_hit      last obj hit if recursive call
 */
void ray_trace(list_t *lst, double *base, double *dir,
               double *intensity, double *total_dist,
               obj_t  *last_hit)
{
     obj_t  *closest;
     double  mindist;
     double specref[3] = {0.0, 0.0, 0.0};

     if (total_dist > MAX_DIST)
           return;

     Set  "closest" to point to the closest object hit by the rayt.
      If closest is NULL
          return;
     Add the distance from base of the ray to the hit point to total_dist
     Add the ambient reflectivity of the object to the intesity vector
     Add the diffuse reflectivity of the object at the hitpoint to the intensity vector
     Scale the intensity vector by 1 / total_dist.

     closest->getspec(specref);   /* see if object has specular reflectivity */
     if (vl_dot3(specref, specref) > 0)  {
          double specint[3] = {0.0, 0.0, 0.0};
          compute direction, ref_dir,  of the reflected ray.
          ray_trace(model, closest->hitloc, ref_dir, specint, total_dist, closest);
          multiply specref by specint leaving result in specref
     }
     vl_sum3(intensity, specref, intensity);
} .
```

Ensure that *specref[]* is a *local copy* of the specular reflectivity! You must not corrupt the reflectivity in the material structure!!

Specular lighting may also be used in combination with other effects such as texturing. In applications such as the *specref[]* values may be used to tune the blending of the base texture with the reflected image.