COMP3080 Computer Graphics
Ray Tracing

By Vu Luong
Student number: 15055014

1. **Plane and Cylinders:**
   Plane: Based on the ray equation **P + tV** where P is the initial point, V is the direction. I calculate the dot product of the plane normal with the ray origin plus the plane's d value(=n.p) and divide it by the dot product of the ray direction and the plane's normal to get -t.
   From there I can check whether t is in the range tMin and tMax then calculate the **hitPosition = ray.origin + t*ray.direction** to get where the ray should intersect the plane

   Cylinder: Similar to how to calculate intersection for spheres, I calculated a, b, c to then put into the quadratic formula and solve for two roots. However since the cylinders have a direction where they expand infinitely, I had to factor this information into my a,b,c calculations:
   a = D.D – (D.V)^2
   b = 2[D.X – (D.V)(X.V)]
   c = X.X – (X.V)^2 – r^2
   D = ray direction
   V = cylinder direction
   X = ray origin – cylinder position
   r = cylinder radius
   then with these information I proceed to calculate the HitInfo the same as Spheres.

2. **Materials:**
   Paper: White-ish grey, no specular, little glossiness, no refraction, no reflectiveness
   Plastic: dark yellow, low specular, low glossiness, no refraction, little reflectiveness
   Glass: no colour, high specular, high glossiness, refractive index about 0.9, high reflectiveness
   Steel mirror: no colour, low specular, low glossiness, no refraction, high reflectiveness

3. **Shadows:**
   I checked if the ray from hit position to light source intersects with anything in the scene, if yes then no visibility therefore cast shadow, if no then visibility = 1, no shadow
   Common problem: Shadow rounding errors
   Solution: Make shadow rays start a tiny distance from the surface, can be done by moving the start point or limiting the t range

4. **Reflection & Refraction:**
   Reflection: I calculate the reflected ray using the hitPosition as the origin and the *reflect()* function built in to glsl to calculate the direction of the reflected ray.
   Refraction: Similar to reflection, I use the hitPosition as the origin and the *refract()* function built in to glsl to calculate the direction of the refracted ray.

5. **Fresnel:**
   I used Schlick's Approximation to calculate Fresnel factor
   https://en.wikipedia.org/wiki/Schlick%27s_approximation
   Only this part of the code was helped created by one of my friends, Raymond Tan