The Technical University of Denmark

02562 Rendering - Introduction

Assignment 2

Written by

Anders Bo Sørensen s125010

Supervisor

Jeppe Revall Frisvad

Hand-in date

September 26, 2017

1 Results

On a sidenote, I saw the feedback for last week's hand-in after I had taken all the pictures. The triangle intersection has been fixed afterwards.

The following results each correspond to an exercise.

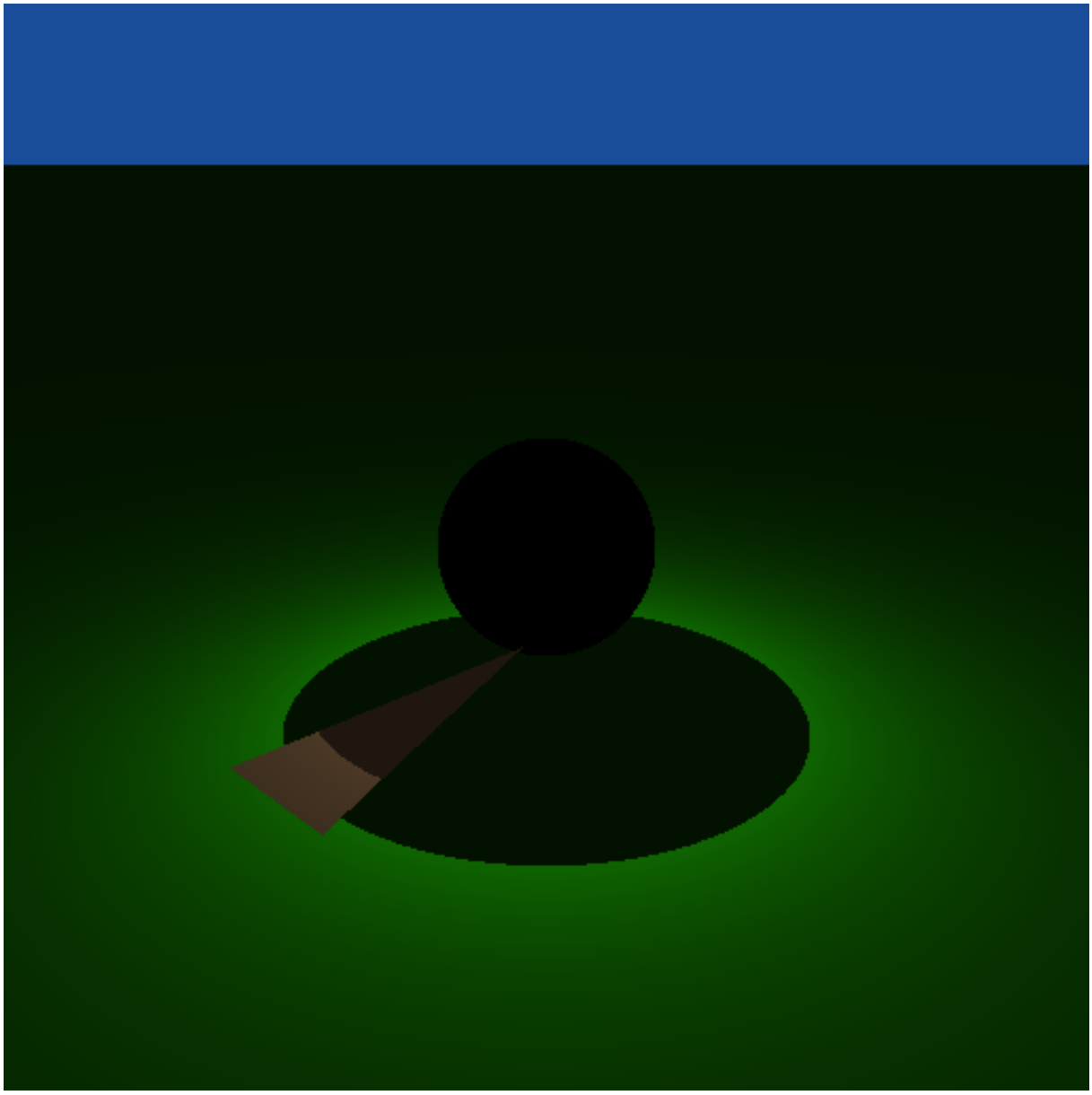

Figure 1: Hard shadows

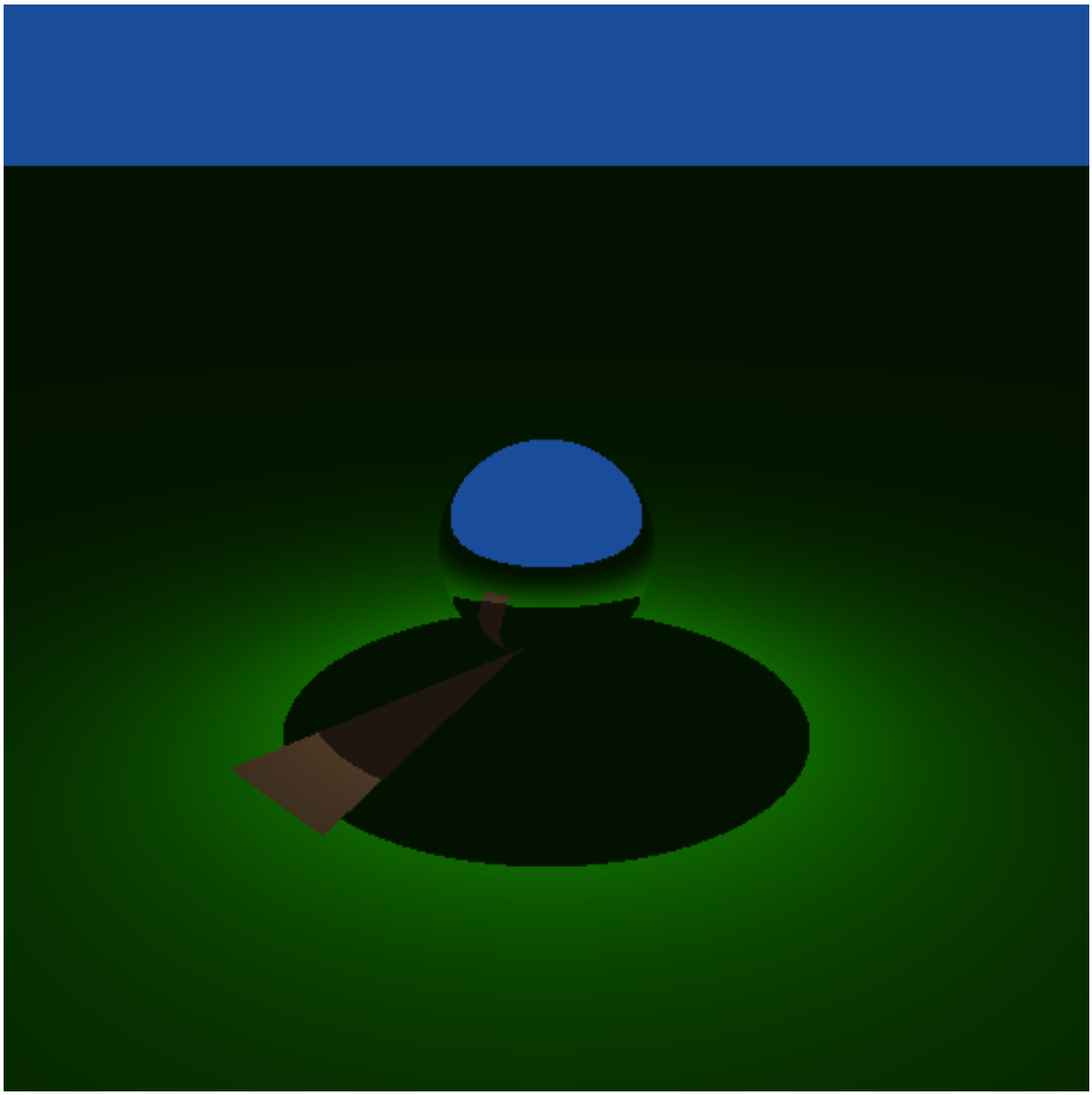

Figure 2: Mirror reflection

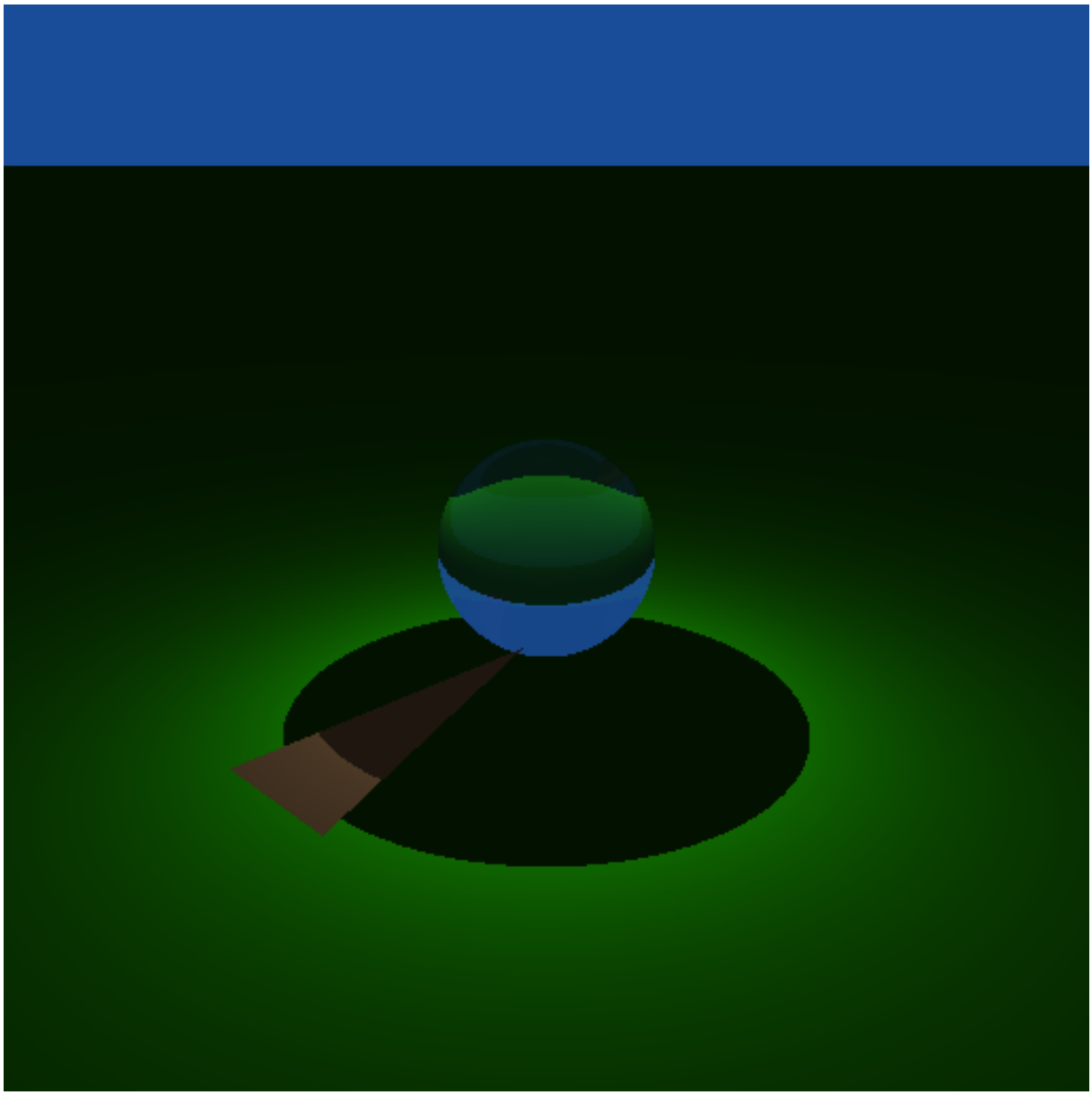

Figure 3: Refraction

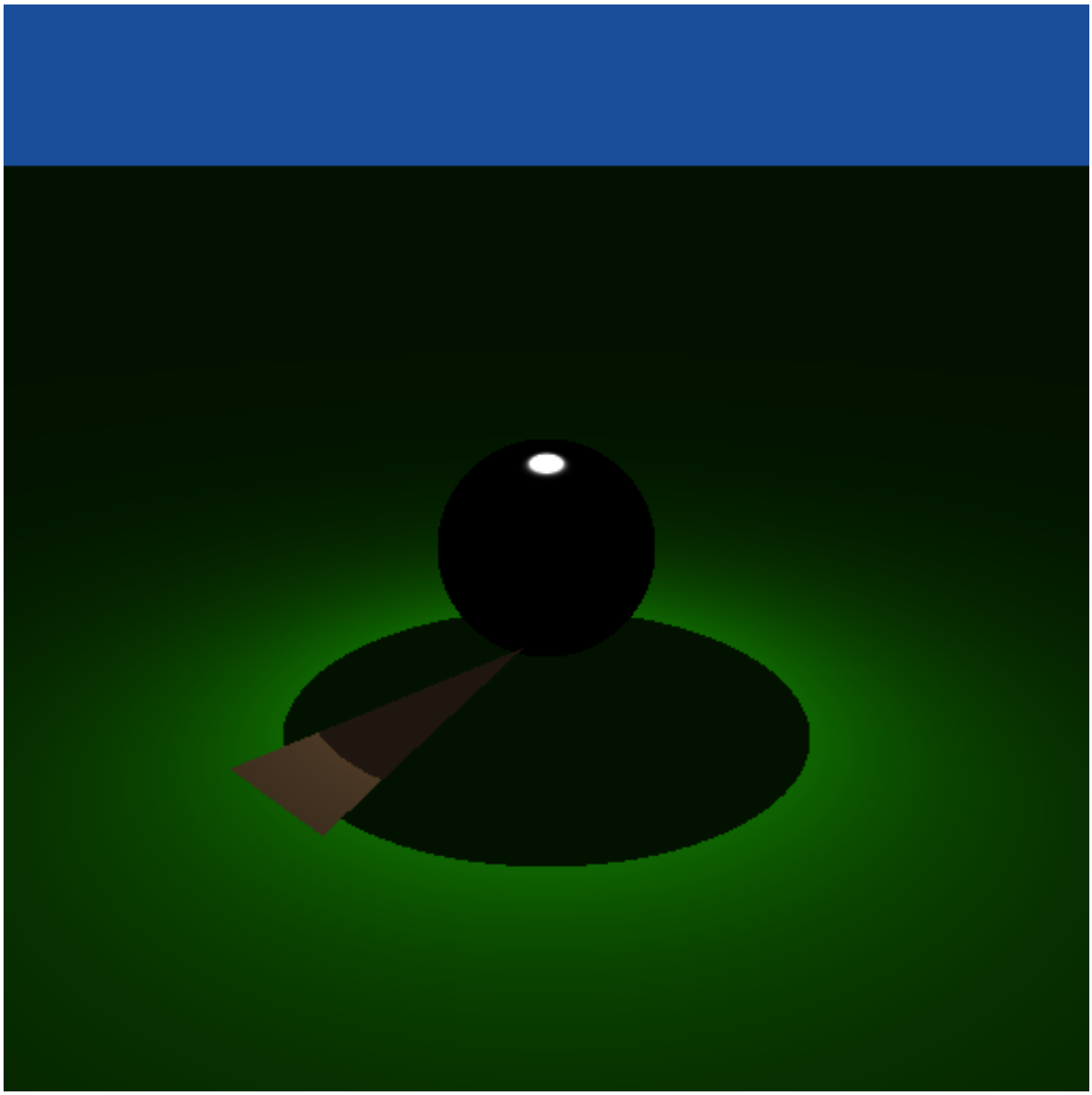

Figure 4: Phong highlight

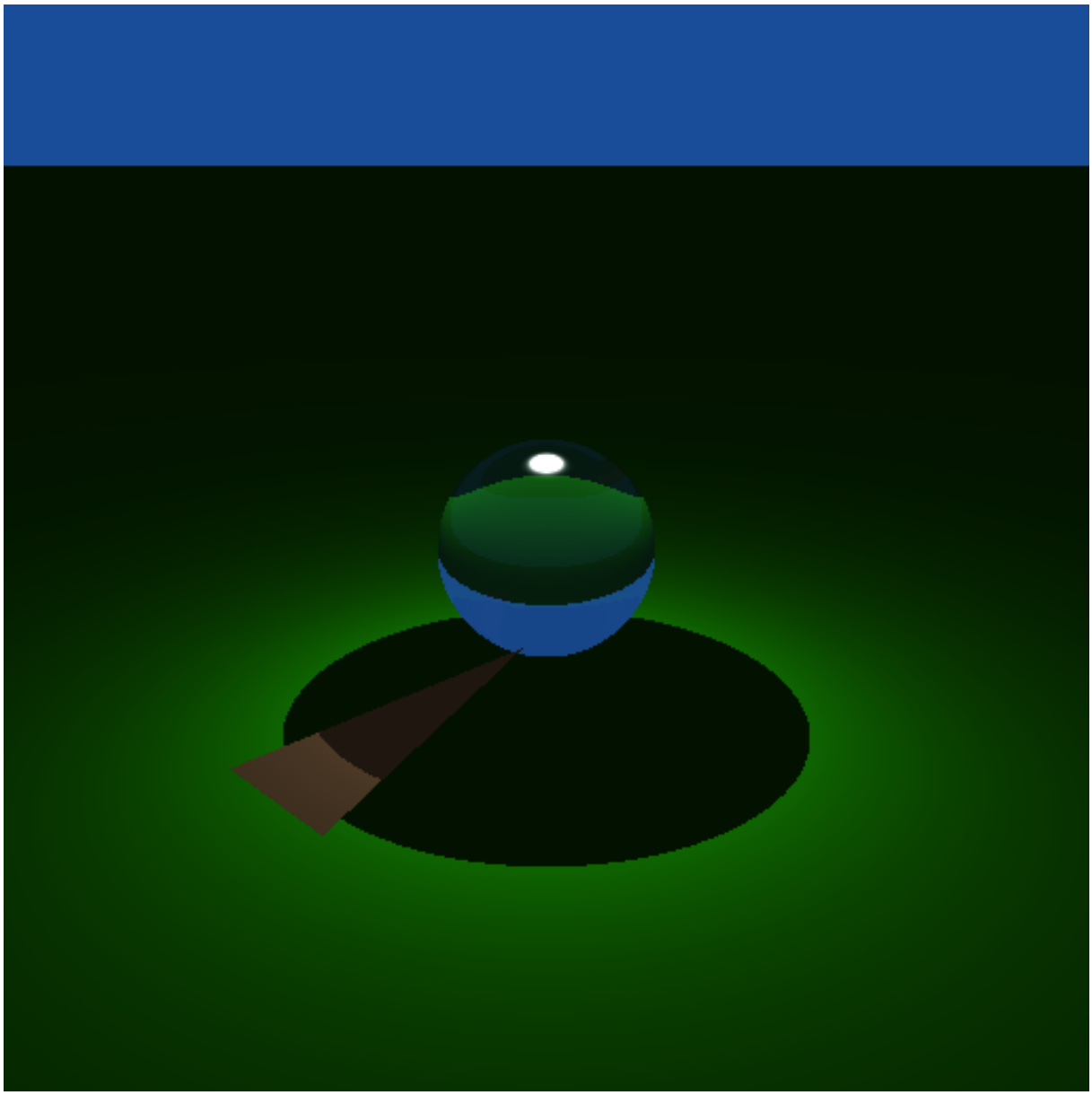

Figure 5: Reflection, refraction and Phong.

The following code was written
sample in PointLight.cpp

```
if (shadows) {
        Ray& shadowRay = Ray(pos, normalize(light_pos - pos), 0, 1.0e-4f, length(light_pos -
        HitInfo hit;

        if (tracer->trace_to_any(shadowRay, hit)) {
                return false;
        }
}

const float3 ab = light_pos - pos;
dir = normalize(ab);
L = intensity / pow(length(ab),2);
return true;
```

trace_reflected in RayTracer.cpp

```
out = Ray(in_hit.position, optix::reflect(in.direction, in_hit.shading_normal), 0, 0.01f);

if (this->trace_to_closest(out, out_hit))
{
        out_hit.trace_depth = in_hit.trace_depth + 1;
        out_hit.ray_ior = in_hit.ray_ior;
        return true;
}

return false;
```

trace_refracted in RayTracer.cpp

```
float theta;
float3 outDir, outNorm, refractedDir;

out_hit.ray_ior = get_ior_out(in, in_hit, outDir, outNorm, theta);

if (optix::refract(refractedDir, -outDir, outNorm, out_hit.ray_ior / in_hit.ray_ior))
{
        out.origin = in_hit.position;
        out.direction = refractedDir;
        out.tmin = 1e-4f;
        out.tmax = 999999;

        if (this->trace_to_closest(out, out_hit))
        {
                out_hit.trace_depth = in_hit.trace_depth + 1;
                return true;
        }
}

return false;
```

get_ior_out in RayTracer.cpp

```
//The normal for the point that was hit is the same
normal = in_hit.shading_normal;

//New direction is the opposite of the ingoing
dir = -in.direction;

//Angle between normal and direction
cos_theta_in = dot(normal, dir);

//If under the surface reverse the normal and the angle
if (cos_theta_in < 0.0)
{
        normal = -normal;
        cos_theta_in = -cos_theta_in;
        return 1.0f;
}
const ObjMaterial* m = in_hit.material;
return m ? m->ior : 1.0f;
```

shade in Phong.cpp

```
float3 result = make_float3(0.0f);
float3 L, dir, R;

for (uint i = 0; i < lights.size(); ++i)
```

```
{
        if (lights[i]->sample(hit.position, dir, L))
        {
                R = reflect(-dir, hit.shading_normal);
                result += rho_d*L*fmaxf(dot(hit.shading_normal, dir), 0) + rho_s*L* powf(fmaxf(dot(

        }
}

return result;
```

shade in Glossy.cpp

```
      if (hit.trace_depth >= max_depth)
              return make_float3(0.0f);

      float R;
      Ray reflected, refracted;
      HitInfo hit_reflected, hit_refracted;
      tracer->trace_reflected(r, hit, reflected, hit_reflected);
      tracer->trace_refracted(r, hit, refracted, hit_refracted, R);
      return (1.0f - R)*Phong::shade(r,hit,emit) + R*shade_new_ray(reflected, hit_reflected) + (1.0
```