光线追踪总结

除了刚才讨论过的，关于光线追踪还有很多话题： 优化，折射。最终再梳理一遍至今の所有伪码。

```
CanvasToViewport(x, y){
  return (x * Vw/Cw, y * Vh/Ch, d)
}

ReflectRay(R, N) {
  return 2*N*dot(N, R) - R
}

ComputeLighting(P, N, V, s) {
    i = 0.0
    for light in scene.Lights {
        if light.type == ambient {
            i += light.intensity
        } else {
            if light.type == point {
                L = light.position - P
                t_max = 1
            }
            else {
                L = light.direction
                t_max = inf
            }

            # shadow check
            shadow_sphere, shadow_t = ClosestIntersection(P, L, 0.001, t_max)
            if shadow_sphere != NULL
                continue

            # diffuse
            n_dot_l = dot(N, L)
            if n_dot_l > 0
                i += light.intensity*n_dot_l/(length(N)*length(L))

            # specular
            if s!= -1 {
                R = 2*N*dot(N,L) -L
                r_dot_v = dot(R, V)
                if r_dot_v > 0
                     i += light.intensity*pow(r_dot_v/length(R)*length(V)),s)
            }
        }
    }
    return i
}

ClosestIntersection(O, D, t_min, t_max){
  closest_t = inf
  closest_sphere = NULL
  for sphere in scene.Spheres {
    t1, t2 = IntersectRaySphere(O, D, sphere)
    if t1 in [t_min, t_max] && t1 < closest_t
      closest_t = t1
      closest_sphere = sphere
    if t2 in [t_min, t_max] && t2 < closest_t
      closest_t = t2
      closest_sphere = sphere
  }

  return closest_sphere, closest_t
}

TraceRay(O, D, t_min, t_max, depth){
  closest_sphere , closest_t = ClosestIntersection(O, D, t_min, t_max)

  if closest_sphere == NULL
    return BACKGROUND_COLOR

  # local color
  P = O + closest_t * D #交点P的位置
  N = P - closest_sphere.center #计算P处的法向量
  N = N / length(N) #normalize 法向量
  local_color = closest_sphere.color * ComputeLighting(P, N, -D,
sphere.specular)

  # If we hit the recursion limit or the object is not reflective, we're done
  r = closest_sphere.reflective
  if depth <= 0 or r <= 0:
    return local_color

  # reflected color
  R = ReflectRay(-D, N)
  reflected_color = TraceRay(P, R, 0.001, inf, depth - 1)

  return local_color * (1 - r) + reflected_color*r
}

O = <0,0,0>

for x in [-Cw/2, Cw/2]{
  for y in [-Ch/2, Ch/2]{
    D = CanvasToViewport(x, y)
    color = TraceRay(O, D, t_min, t_max)
    canvas.putPixel(x, y, color)
  }
```

```
}
```

这是所用的所有参数:

```
viewport_size = 1 x 1
projection_plane_d = 1

sphere {
    center = (0, -1, 3)
    radius = 1
    color = (255, 0, 0)  # Red
    specular = 500  # Shiny
    reflective = 0.2  # A bit reflective
}
sphere {
    center = (-2, 1, 3)
    radius = 1
    color = (0, 0, 255)  # Blue
    specular = 500  # Shiny
    reflective = 0.3  # A bit more reflective
}
sphere {
    center = (2, 1, 3)
    radius = 1
    color = (0, 255, 0)  # Green
    specular = 10  # Somewhat shiny
    reflective = 0.4  # Even more reflective
}
sphere {
    color = (255, 255, 0)  # Yellow
    center = (0, -5001, 0)
    radius = 5000
    specular = 1000  # Very shiny
    reflective = 0.5  # Half reflective
}


light {
    type = ambient
    intensity = 0.2
}
light {
    type = point
    intensity = 0.6
    position = (2, 1, 0)
}
light {
    type = directional
    intensity = 0.2
    direction = (1, 4, 4)
}
```