# Basic Ray-Tracing (Algorithm)

## Ray Tracing Pipeline

1. Ray Generation
   - → shoot a ray from the camera (or point x) towards the scene
2. Ray Traversal
   - → traverse through the scene (follow the ray)
3. Intersection
   - → for each step of the traversal check for an intersection
4. Shading
   - → identify radiance of intersection point going back to the rays origin
   - → to compute the radiance, we need to figure out how much light is at this point → shoot rays into the scene
5. Framebuffer
   - → final color is stored here

## Ray Types

- **primary ray:** ray from camera towards scene
- **secondary ray:** reflected or refracted ray
- **shadow ray:** ray directly towards the lightsource
  - → check if point is occluded or illuminated

=> common raytracing uses next-event-estimate (shadow rays)

## Ray Tracing pseudo code

- simplest implementation (Lambertian Shading)
- image plane has dimension resX · resY

### function raytracer

```
MAX_DEPTH = 3
for y : 0 -> resY :                    } iterate over image plane
    for x : 0 -> resX :                }
        image[x,y] = 0
        for s : 0 -> #samplesperpixel      -> Samples per pixel
            ray = generateCameraRay(x,y)
            depth = 0
            radiance = trace(ray, depth) / MAX_DEPTH
            image[x,y] += radiance
        image[x,y] /= #samplesperpixel
```

### function trace(ray, depth)

```
radiance = 0
if depth != MAX_DEPTH :
    hitpoint = intersect(ray)
    direction = SampleRandomCosine(hitpoint)
    random_ray = Ray(hitpoint, direction)
    radiance += emit(hitpoint) + brdf(hitpoint) · trace(random_ray, depth + 1)
return radiance
```