Raytracing Pseudocode
function traceImage (scene):
    for each pixel (i,j) in image
        S = PointInPixel
        P = CameraOrigin
        d = (S - P)/|| S – P||
        I(i,j) = traceRay(scene, P, d)
    end for
end function
function traceRay(scene, P, d):
    (t, N, mtrl) ← scene.intersect (P, d)
    Q ← ray (P, d) evaluated at t
    I = shade(mtrl, scene, Q, N, d)
    R = reflectDirection(N, -d)
    I ← I + mtrl.k_r * traceRay(scene, Q, R)
    if ray is entering object then
        n_i = index_of_air
        n_t = mtrl.index
    else
        n_i = mtrl.index
        n_t = index_of_air
    if (mtrl.k_t > 0 and notTIR (n_i, n_t, N, -d)) then
        T = refractDirection (n_i, n_t, N, -d)
        I ← I + mtrl.k_t * traceRay(scene, Q, T)
    end if
    return I
end function
Thinking About Refraction

Remember Snell’s law?
• \( \eta_i \sin \theta_i = \eta_t \sin \theta_t \)

When does light bend?
• Must account for entering and leaving!
• How do we know if we’re entering or leaving? (hint: all geometry has a “front face” and a “back face”)

\[ \text{Diagram showing incidence and refraction angles.} \]
function shade(mtrl, scene, Q, N, d):
    I ← mtrl.k_e + mtrl. k_a * scene->I_a
    for each light source l do:
        atten = l -> distanceAttenuation(Q) * l -> shadowAttenuation(scene, Q)
        I ← I + atten*(diffuse term + spec term)
    end for
    return I
end function
function PointLight::shadowAttenuation(scene, P)
    d = (l.position - P).normalize()
    (t, N, mtrl) ← scene.intersect(P, d)
    Q ← ray(t)
    if Q is before the light source then:
        atten = 0
    else
        atten = 1
    end if
    return atten
end function
Some Additional Notes

The raytracer skeleton code is extensive but largely undocumented

- Taking time to look through the code to understand what it does is essential
- Mathematical elegance doesn’t mean there’s a simple codebase
Passing by Reference

Many important values are passed by reference!

• Look carefully to determine where/how values are being updated
• Very common in C and C++ codebases
tmax and tmin

Parametric values that define the bounding box around the scene

• Returned t values are within this range

Scene can be further subdivided for additional intersect optimizations!
Debugging Visually: What Happened?
Casting Shadow Rays

at what $t$ does the ray hit an object?
Casting Shadow Rays

at what $t$ does the ray hit an object?

if lucky: $\{-1.2, 0.0\}$
if unlucky: $\{-1.2, 1e-12\}$
Shadow Rounding Error

Classic fix: move slightly in normal direction before shooting shadow ray

- RAY_EPSILON provided for this
But Shadows Don’t Look Like This!
Hard vs Soft Shadows
Calculate Penumbra

Use full lighting equation or calculate geometrically (not required for A1!)