CS 3220
Root finding applications

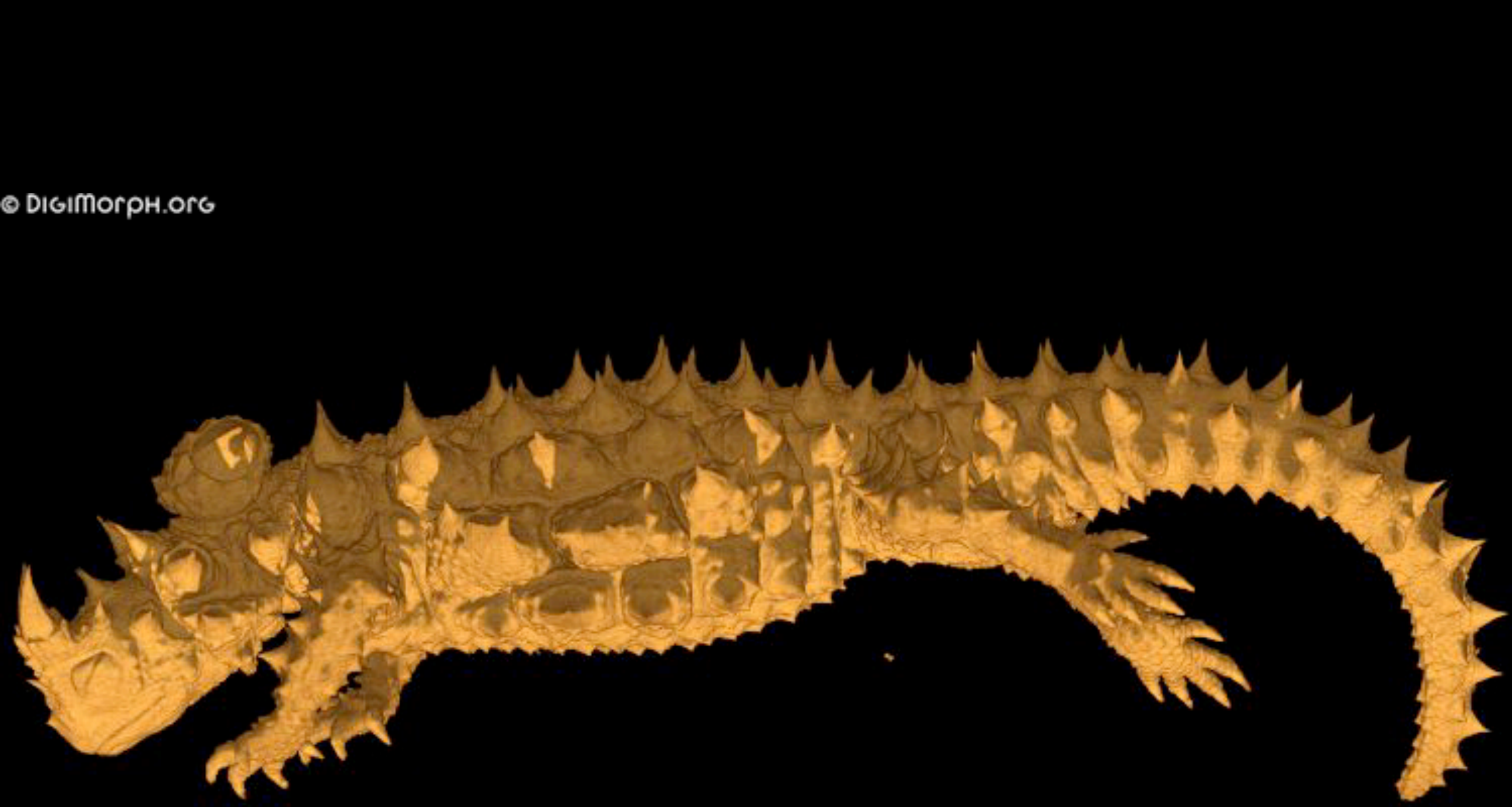
Steve Marschner
Spring 2010

1

© DIGIMORPH.ORG

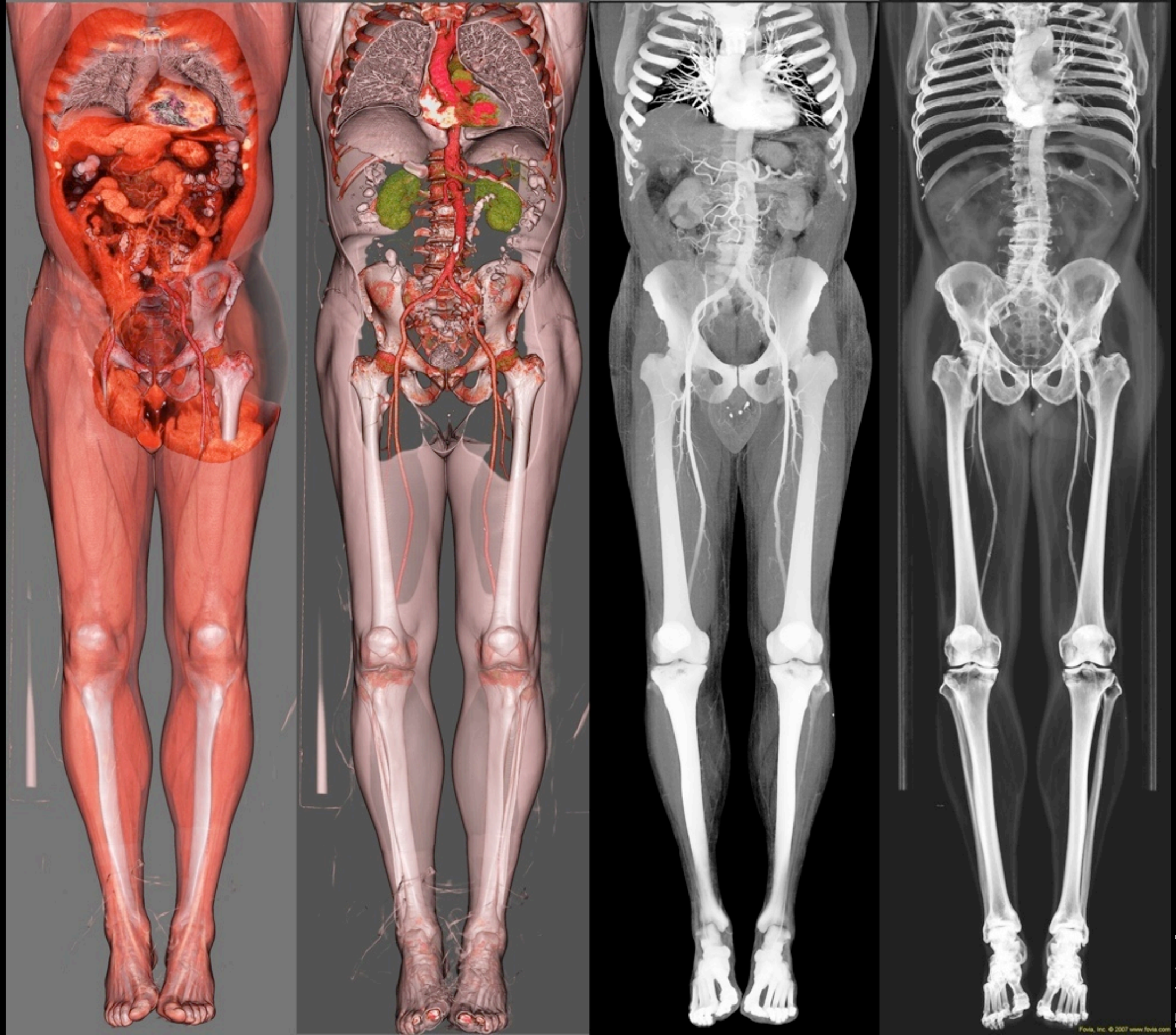
© DigiMorph.org

© DiGiMorpH.orG



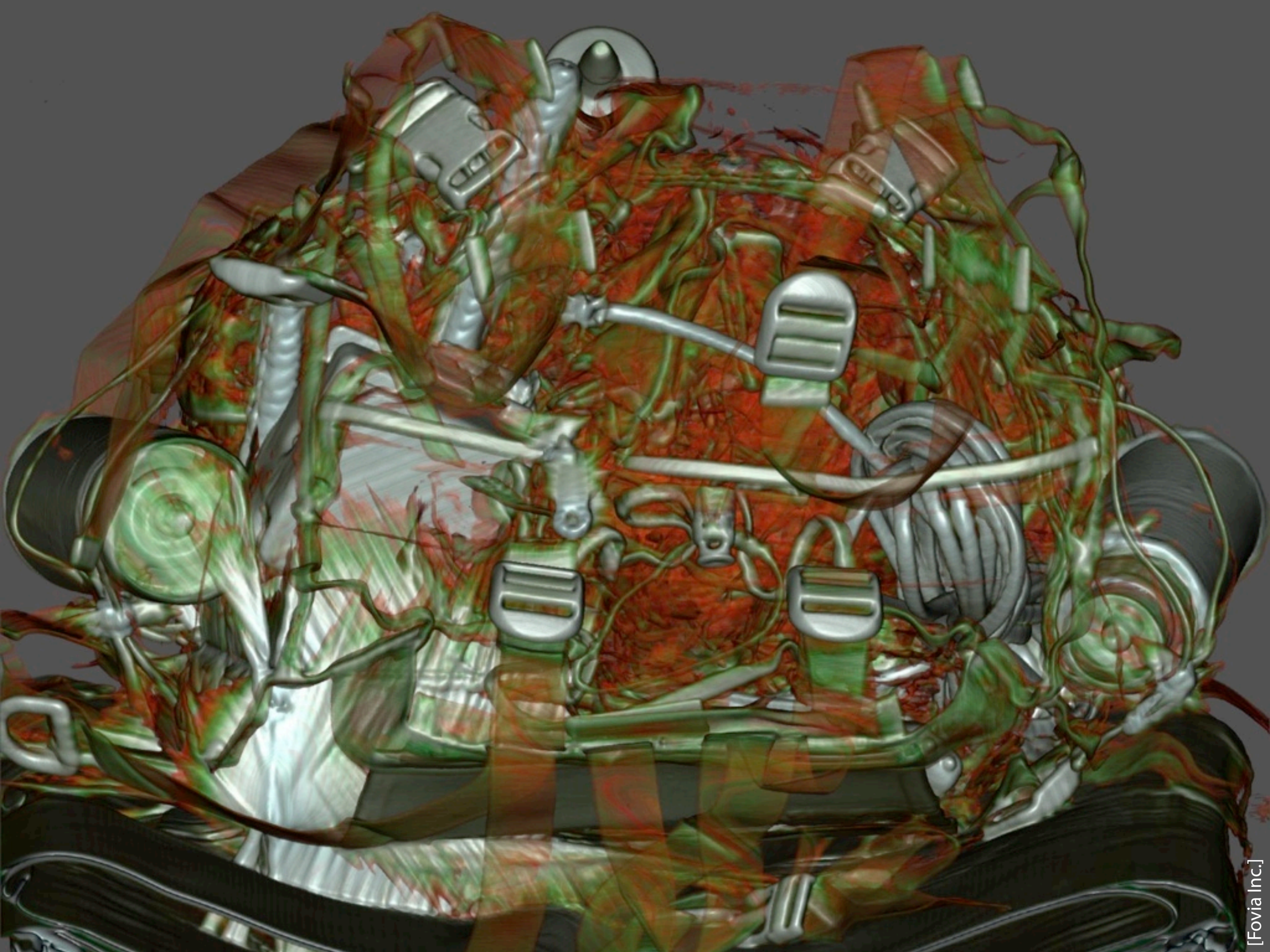
[Visible Human Project, Nat'l Library of Medicine]

[Fovia Inc.]

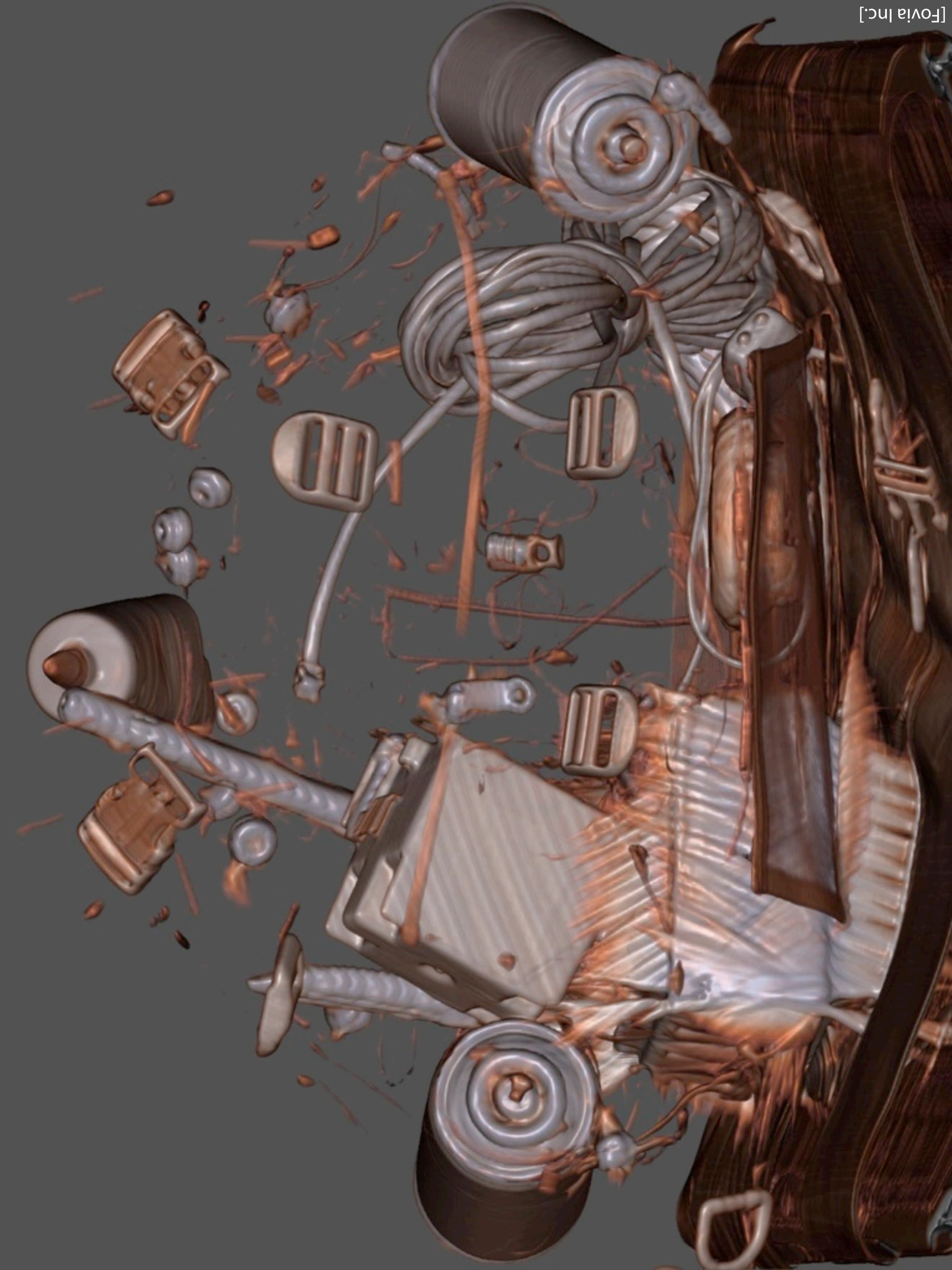


[Fovia Inc.]

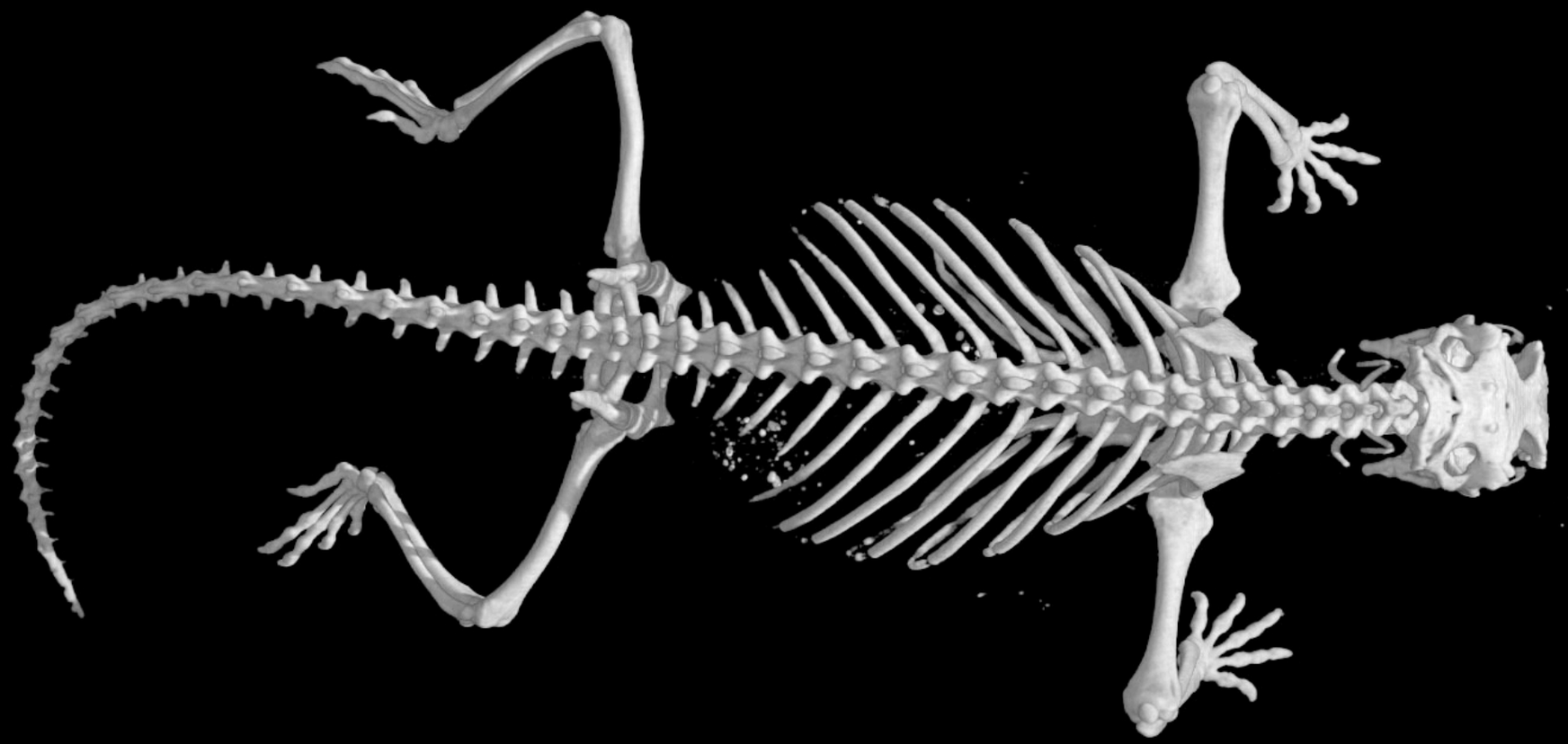
[Fovia Inc.]



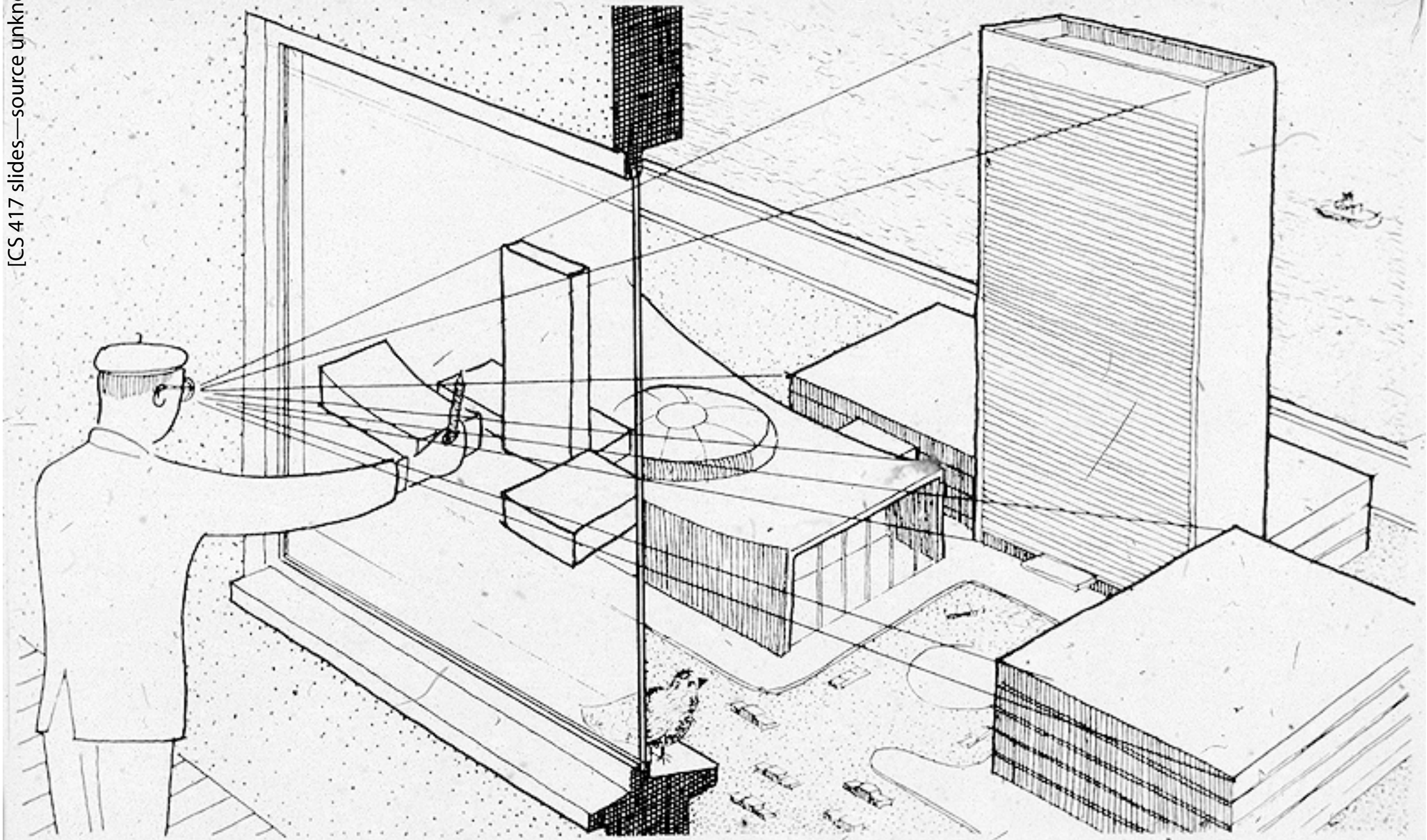
[Fovia Inc.]



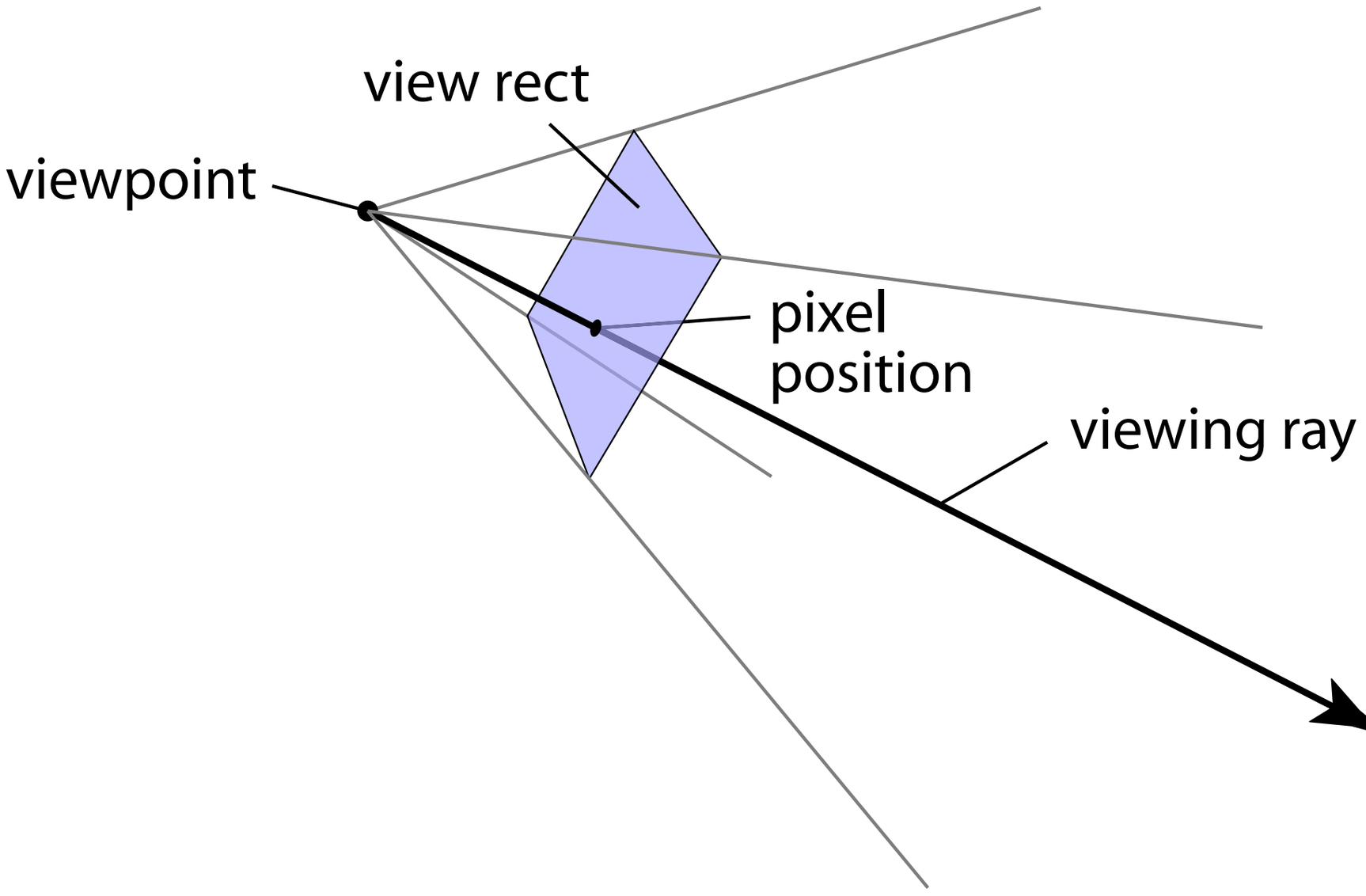
[Fovia Inc.]

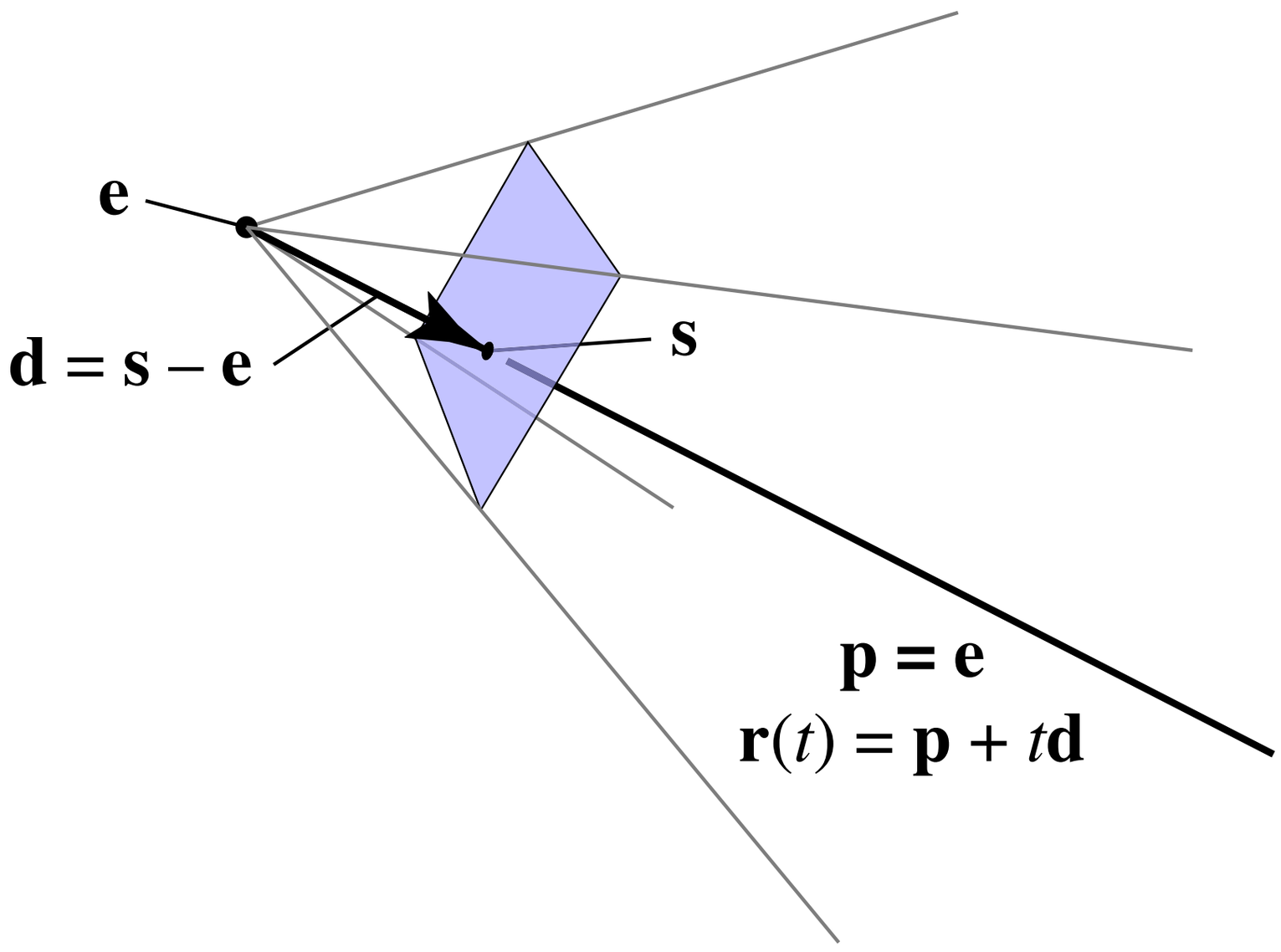


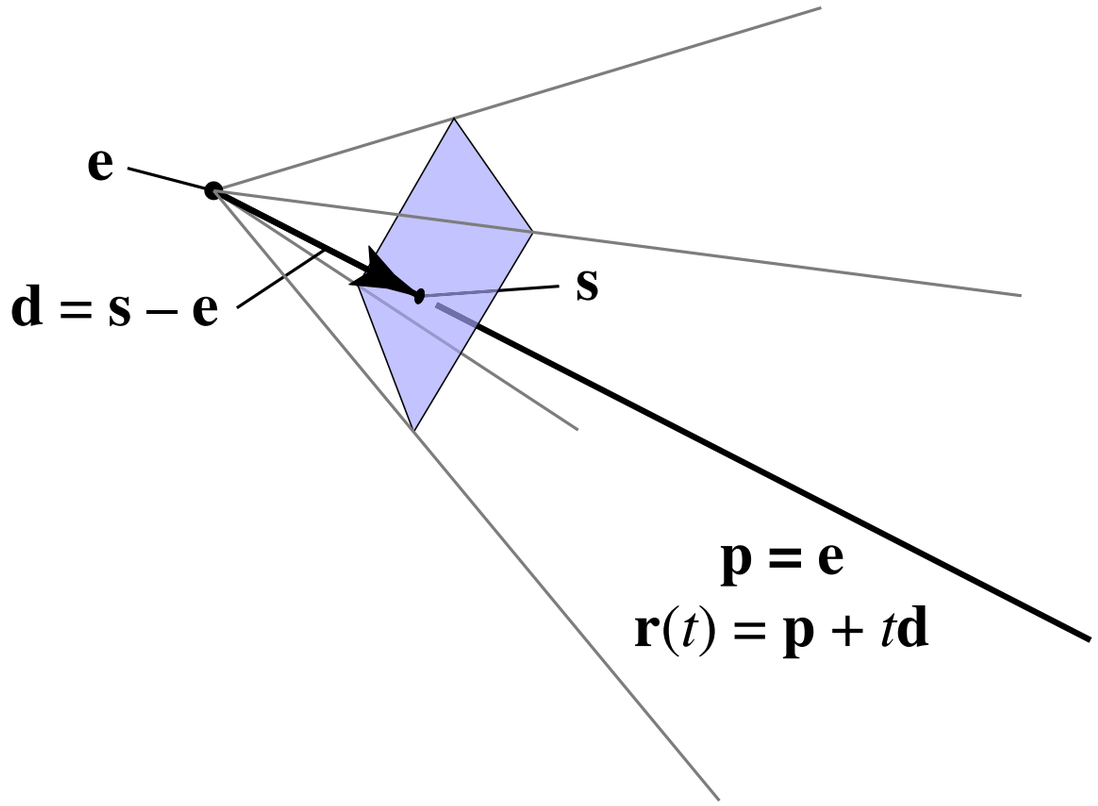
[CS 417 slides—source unknown]

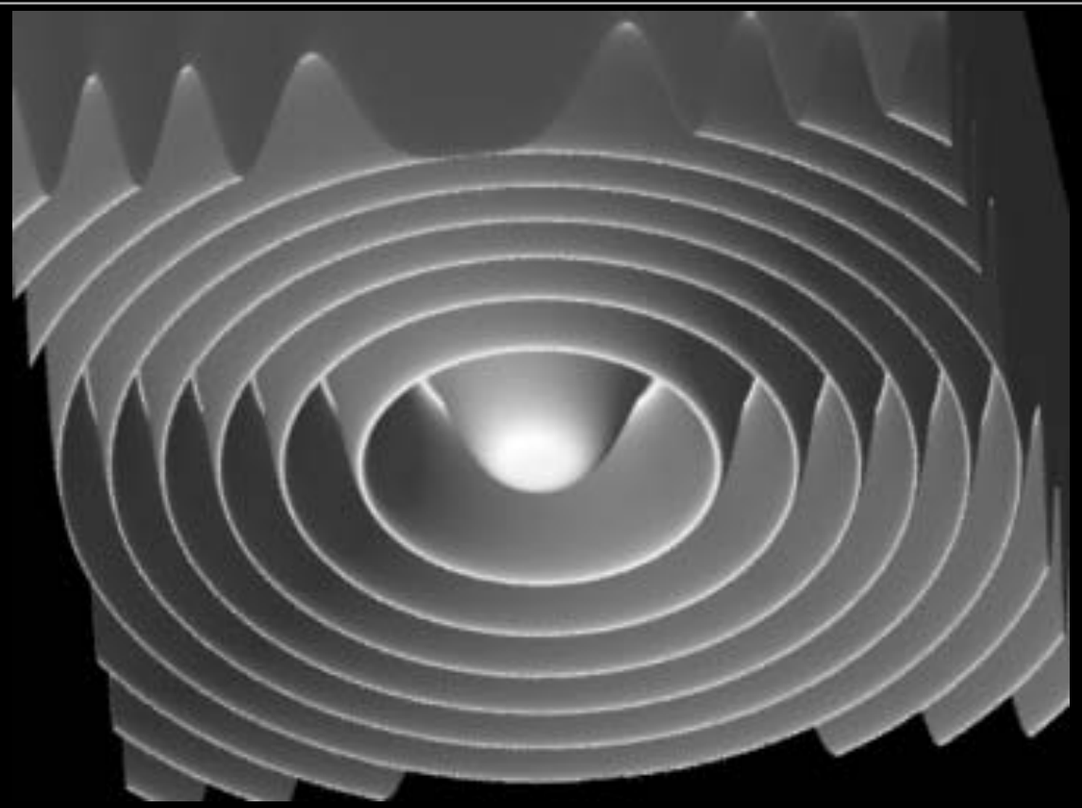


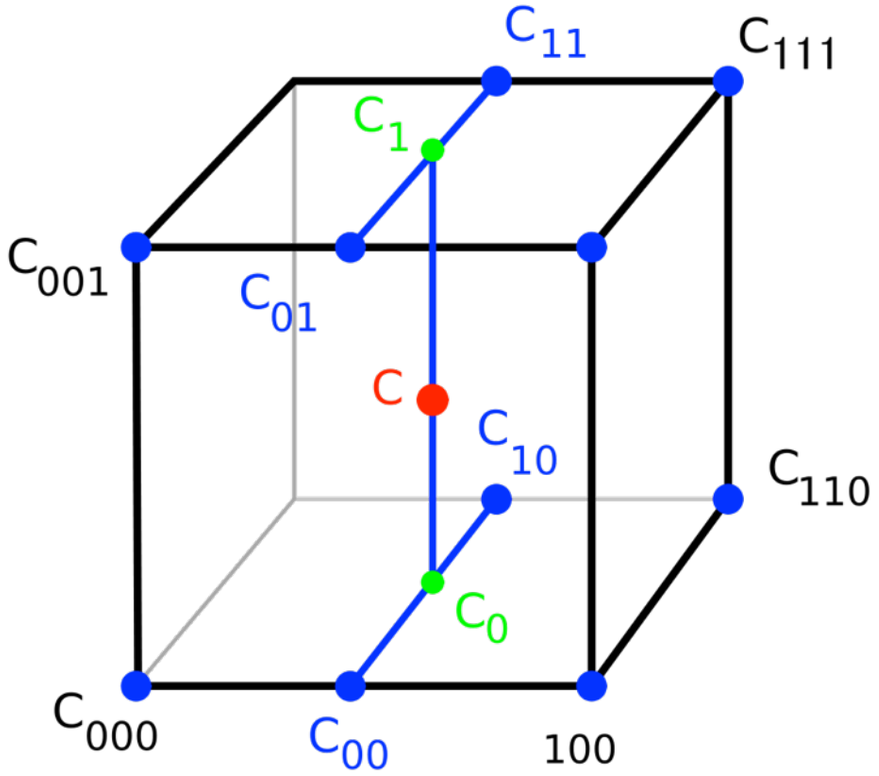
The concept of the picture plane may be better understood by looking through a window or other transparent plane from a fixed viewpoint. Your lines of sight, the multitude of straight lines leading from your eye to the subject, will all intersect this plane. Therefore, if you were to reach out with a grease pencil and draw the image of the subject on this plane you would be "tracing out" the infinite number of points of intersection of sight rays and plane. The result would be that you would have "transferred" a real three-dimensional object to a two-dimensional plane.

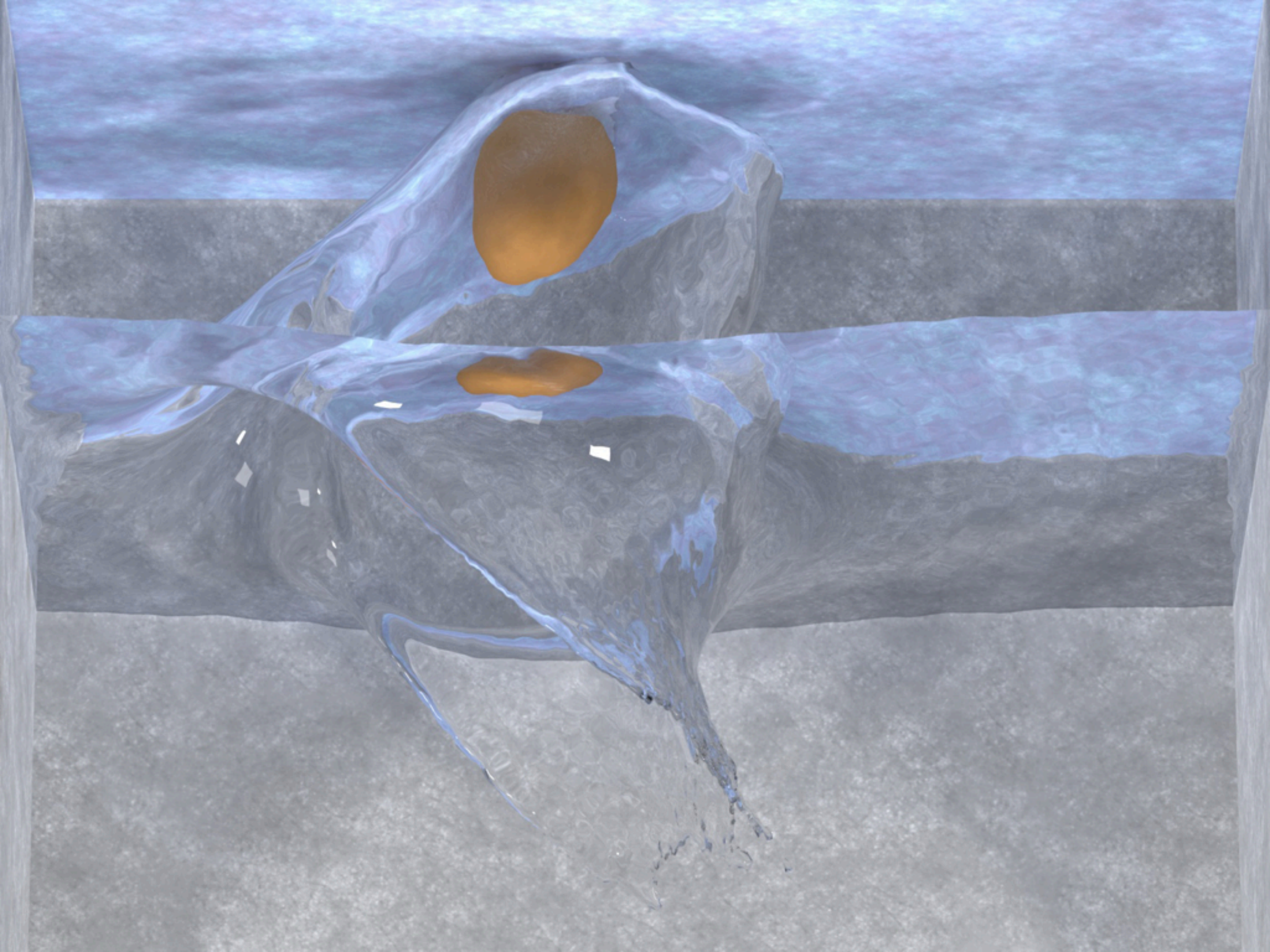

view rect
viewpoint
pixel
position
viewing ray

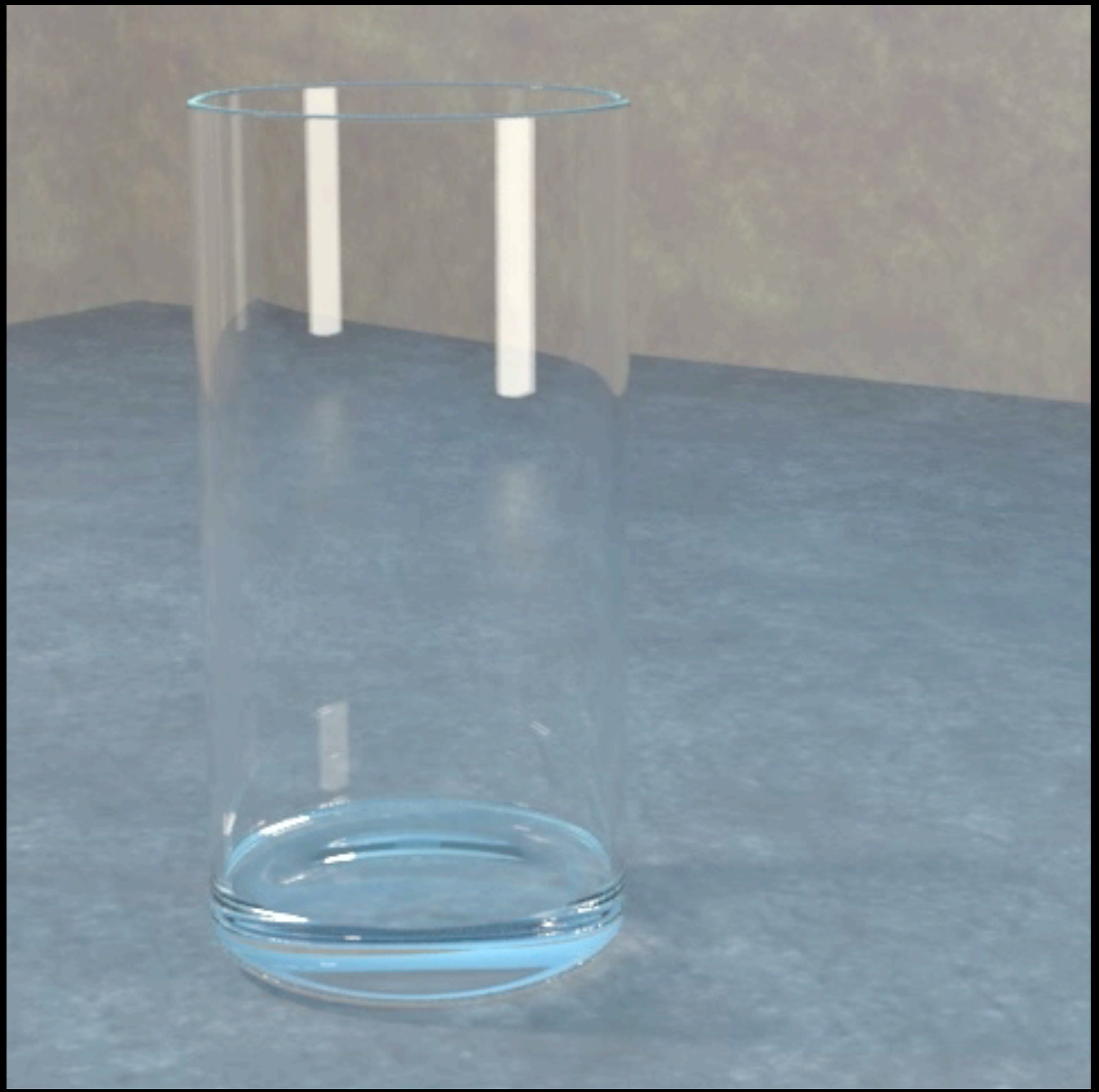

e
d = s – e
s
p = e
r(t) = p + td

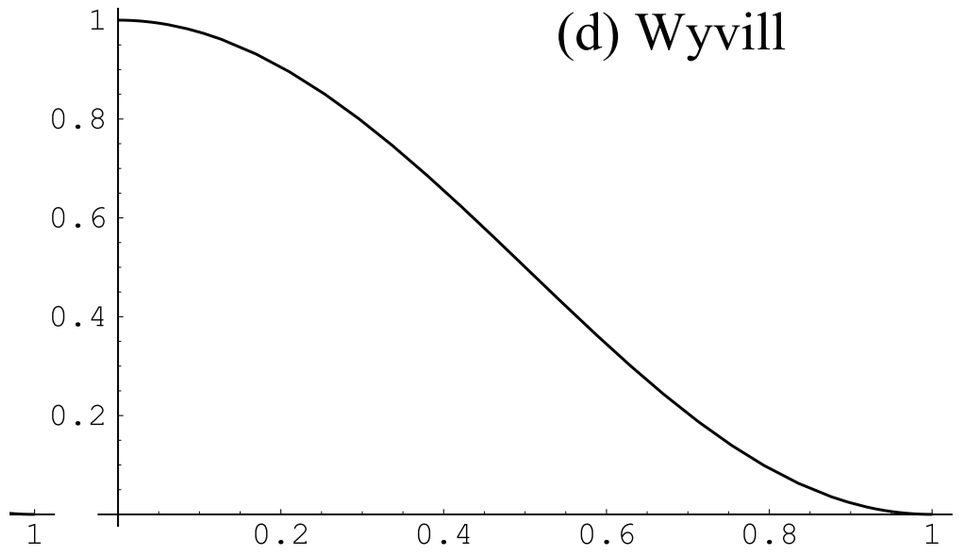

e
d = s – e
s
p = e
r(t) = p + td



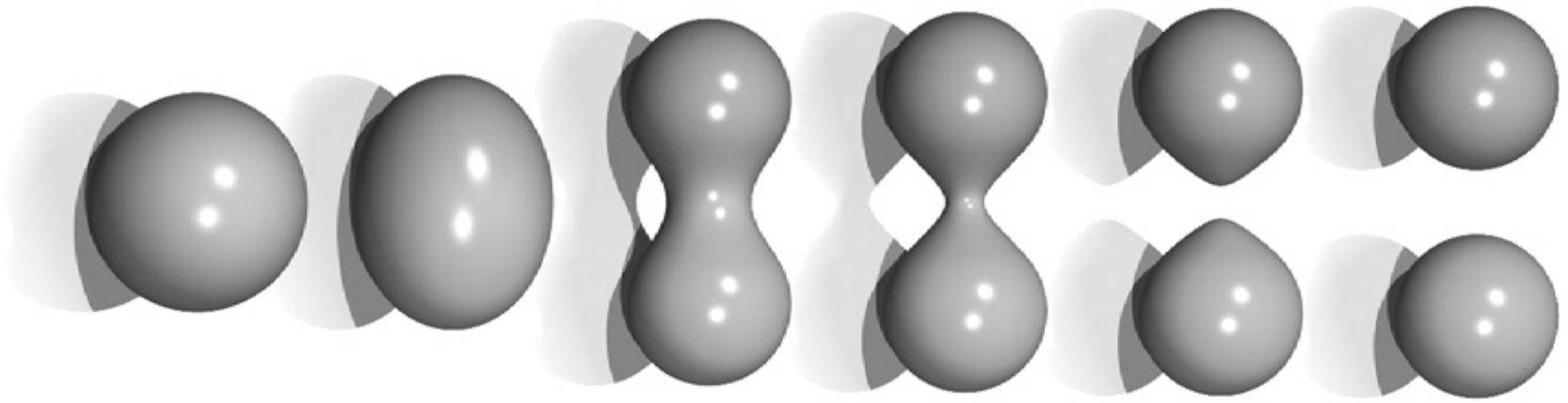

C11
C111
C1
C001
C01
C
C10
C110
C0
C000
C00
100





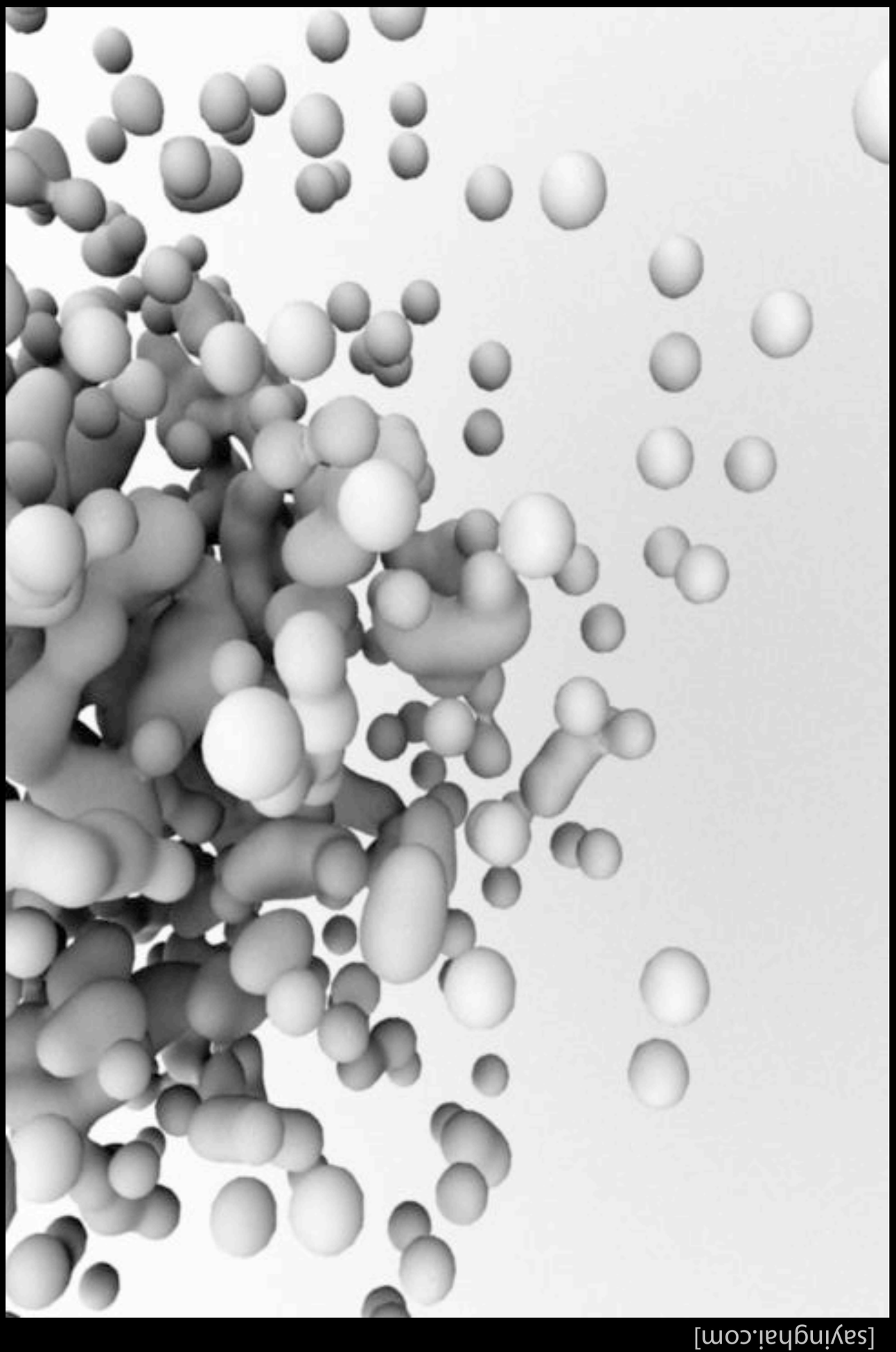


$$g(d) = \left(1 - \frac{d^2}{r^2}\right)^3$$



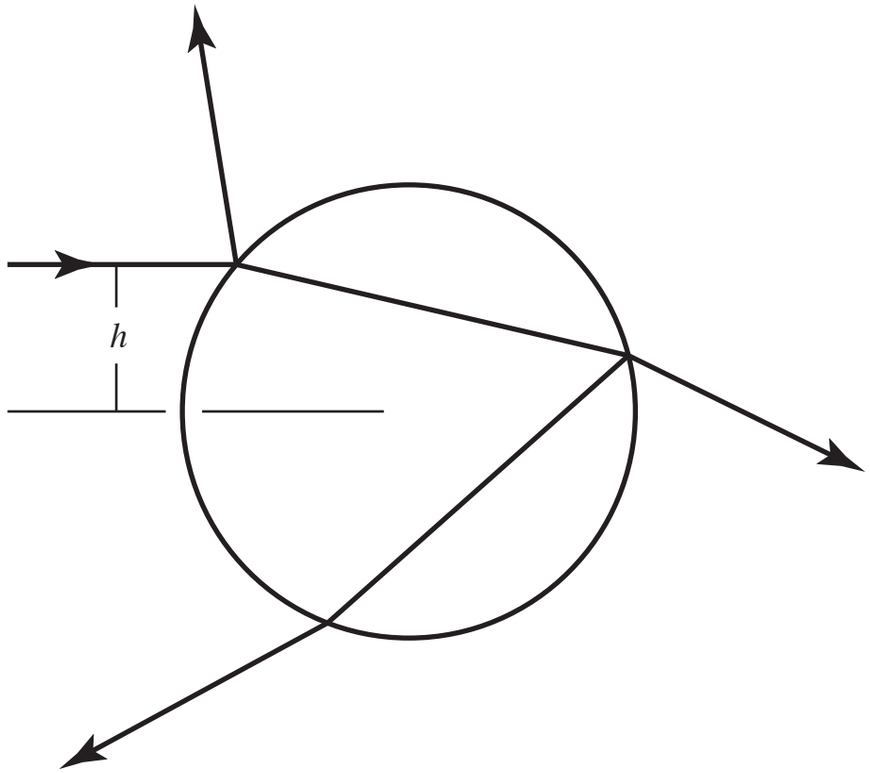
[Brian Wyvill]

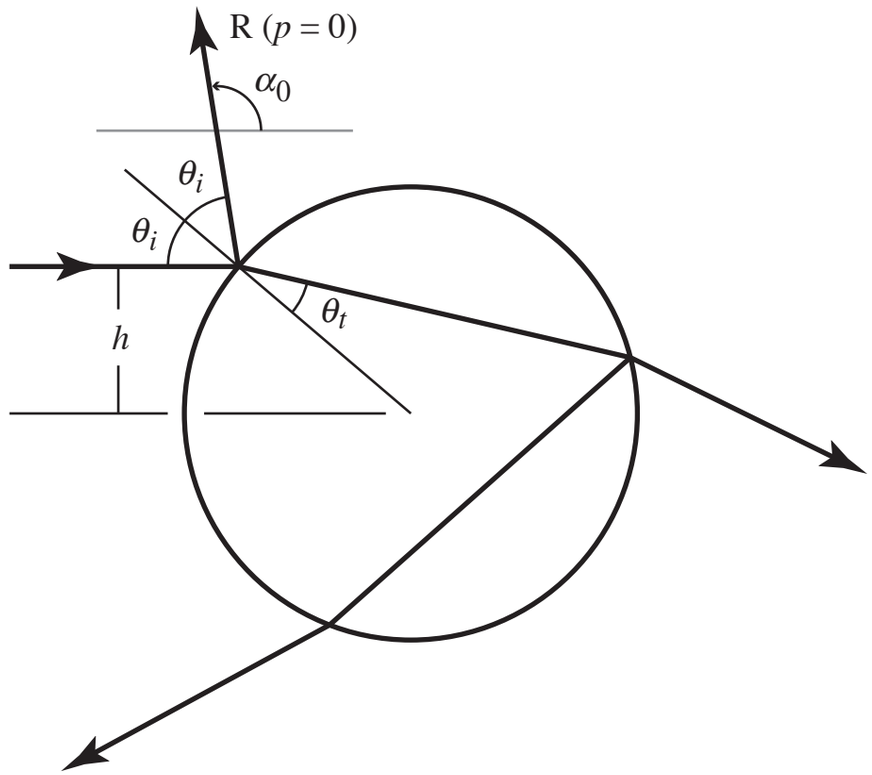
[sayinghai.com]

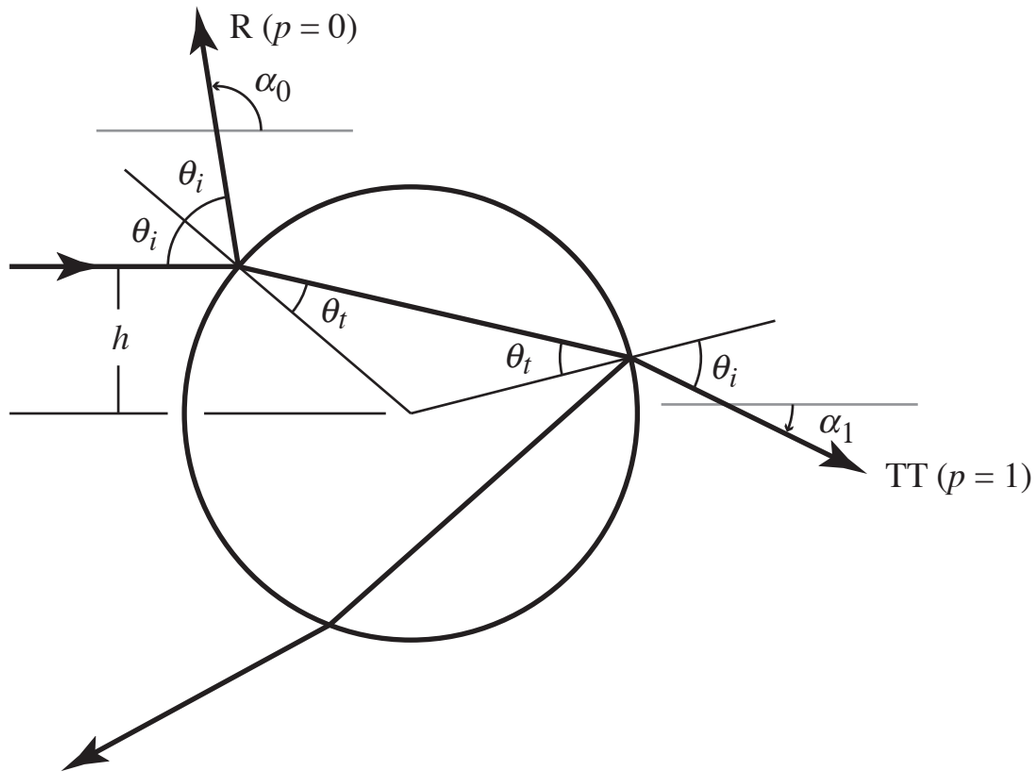


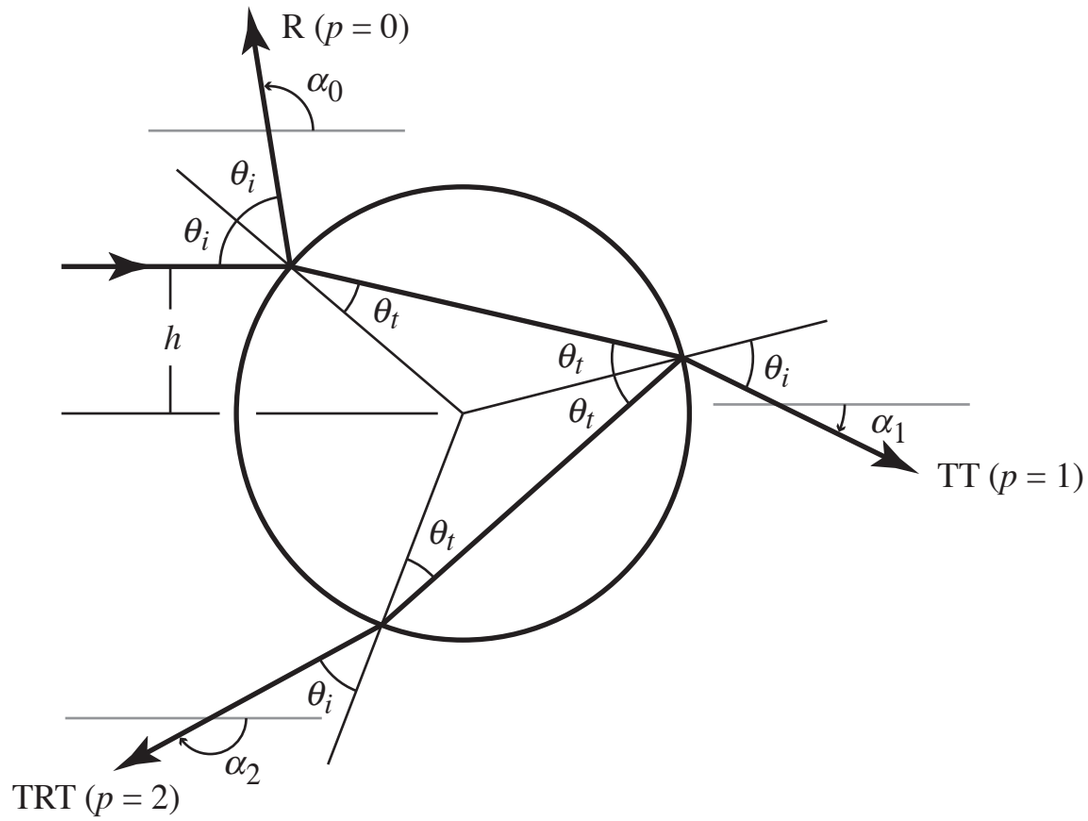
[sayinghai.com]

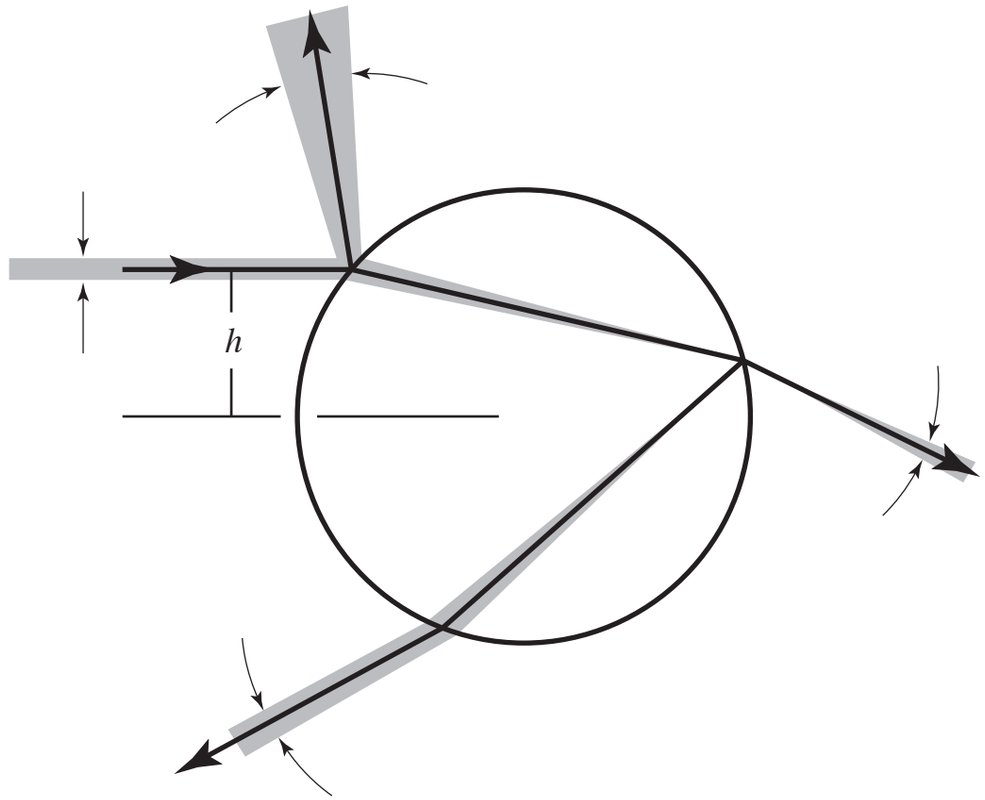
© KenRockwell.com

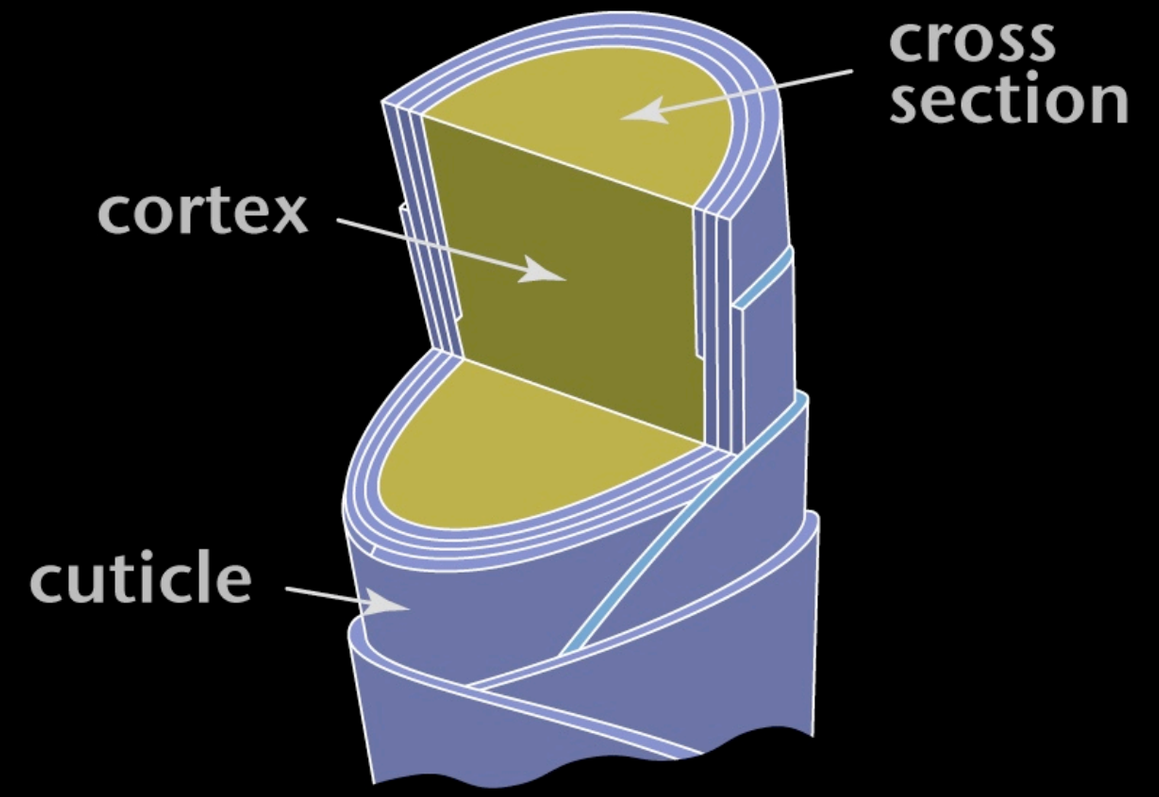

h

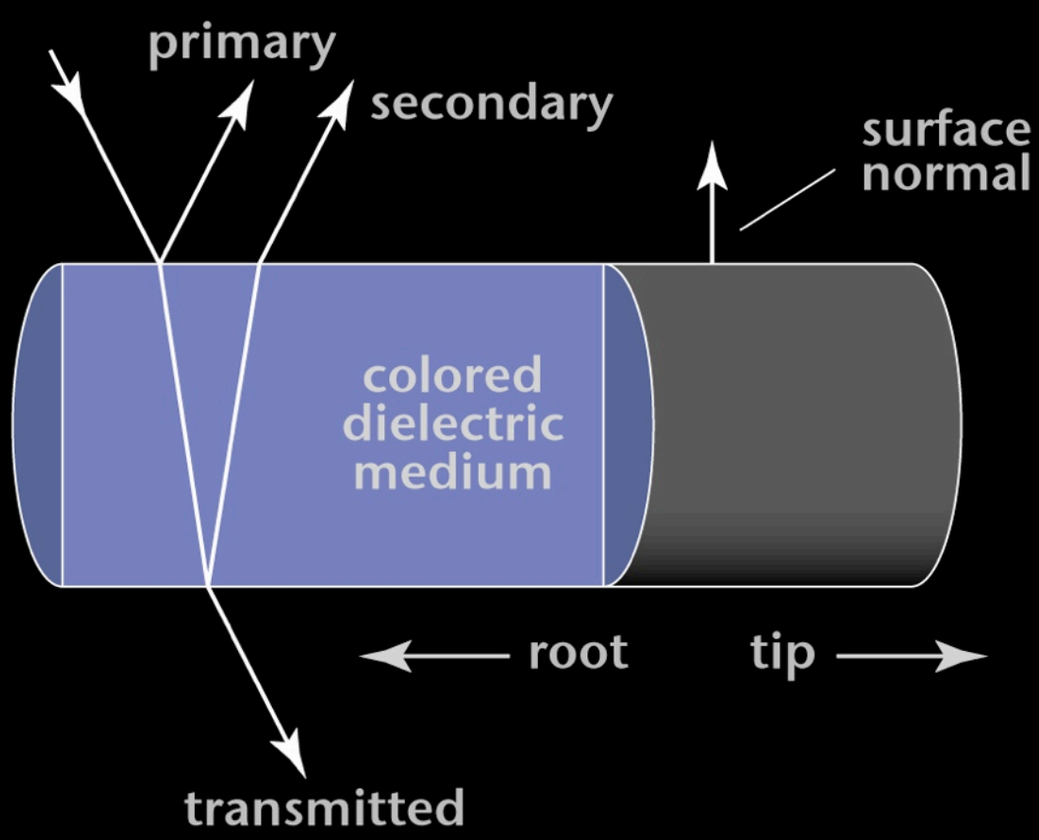

R ($p = 0$)
α_0
θ_i
θ_i
θ_t
h

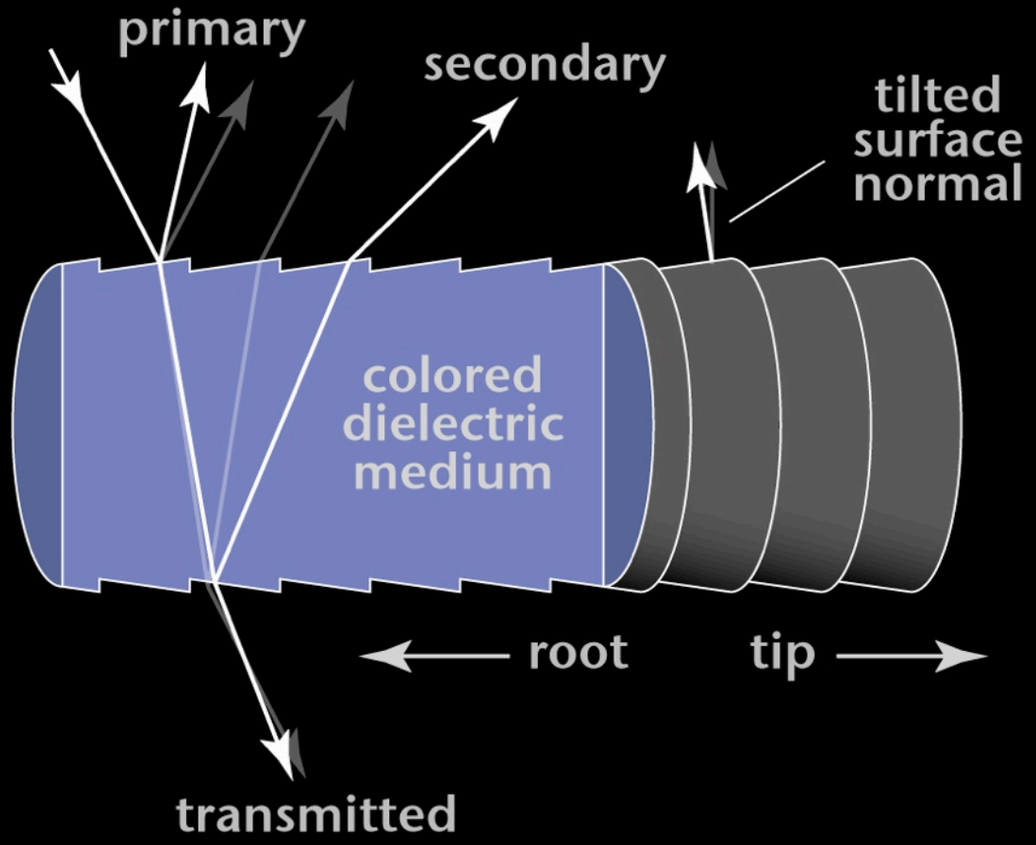

R ($p = 0$)
α_0
θ_i
θ_i
h
θ_t
θ_t
θ_i
α_1
TT ($p = 1$)


R ($p = 0$)
α_0
θ_i
θ_i
h
θ_t
θ_t
θ_t
θ_t
θ_i
α_1
TT ($p = 1$)
θ_i
α_2
TRT ($p = 2$)


h


cross
section
cortex
cuticle


primary
secondary
surface
normal
colored
dielectric
medium
root
tip
transmitted


primary
secondary
tilted
surface
normal
colored
dielectric
medium
root
tip
transmitted

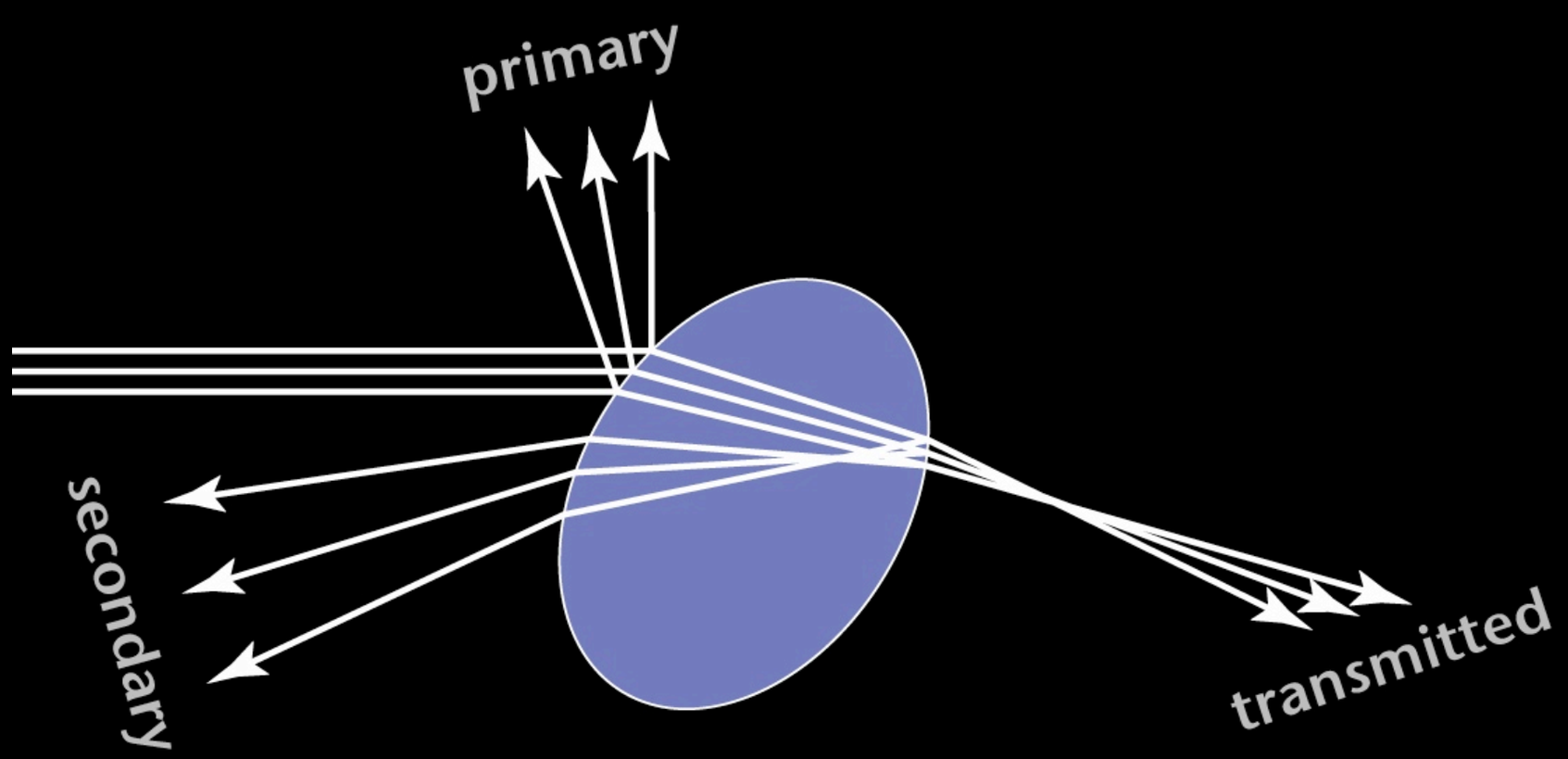
primary

secondary

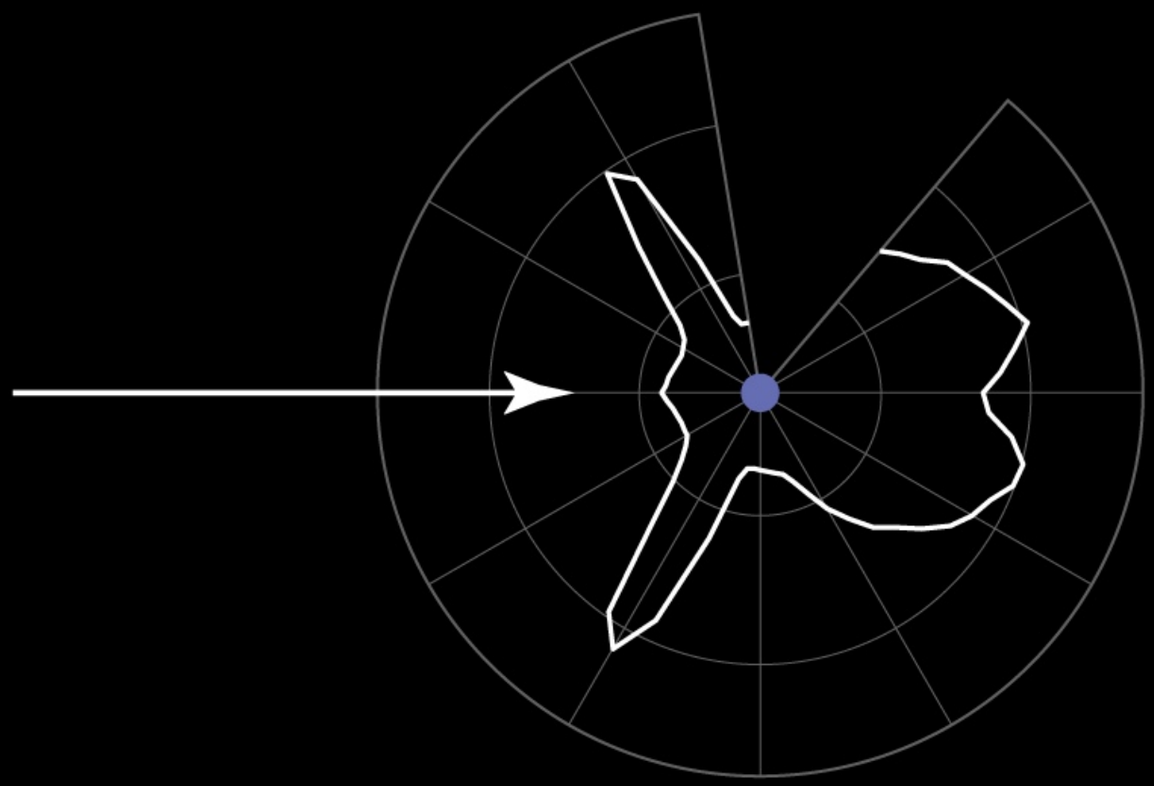
secondary
primary

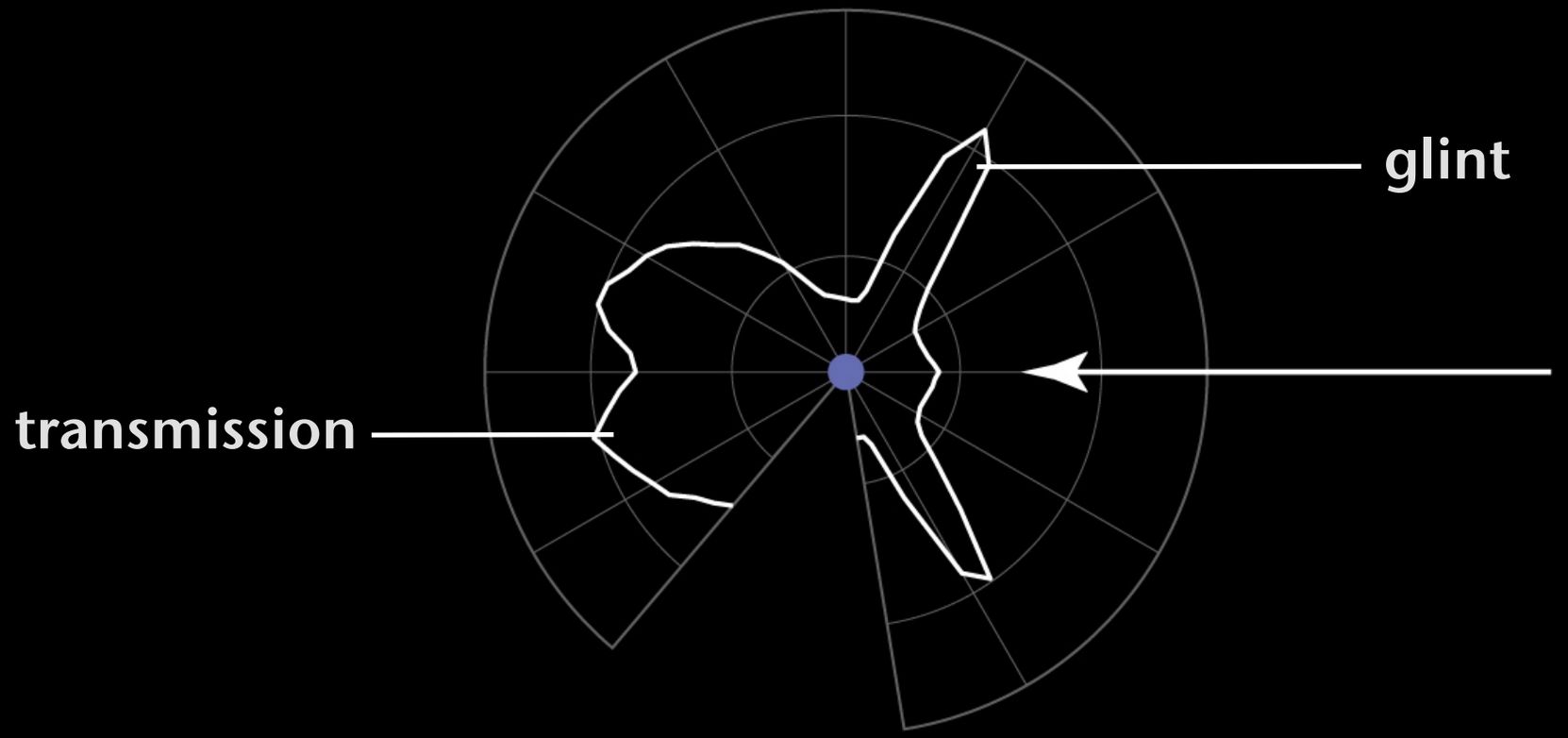
primary

secondary


primary
secondary
transmitted

glints




glint
transmission

```
float InvSqrt(float x) {
    float xhalf = 0.5f*x;
    int i = *(int*)&x;
    i = 0x5f3759df - (i>>1);
    x = *(float*)&i;
    x = x*(1.5f-xhalf*x*x);
    return x;
}
```

```
float InvSqrt(float x) {
    float xhalf = 0.5f*x;
    int i = *(int*)&x;  // get bits for floating value
    i = 0x5f3759df - (i>>1); // gives initial guess y0
    x = *(float*)&i;    // convert bits back to float
    x = x*(1.5f-xhalf*x*x); // Newton step, repeating increases accuracy return x;
}
```