# Participating media

UCSD CSE 272
Advanced Image Synthesis

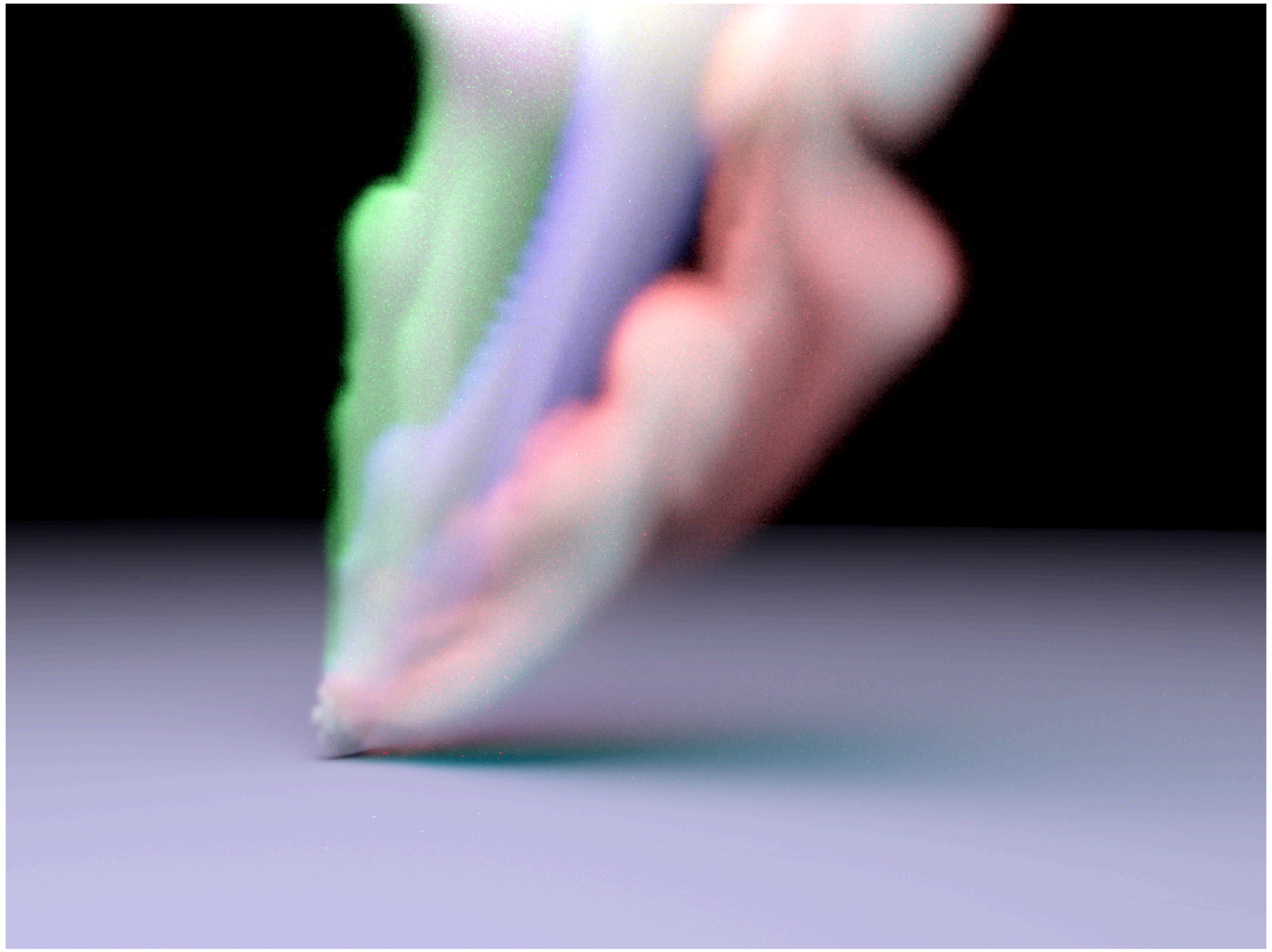
Tzu-Mao Li

*organization of the slides heavily borrowed from the SIGGRAPH course "Monte Carlo methods for physically-based volume rendering"*
https://cs.dartmouth.edu/~wjarosz/publications/novak18monte-sig.html

# HW2 is out

UCSD CSE 272 Assignment 2:

Volumetric Path Tracing



- START EARLY
- ASK QUESTIONS

# Today: foggy and transparent stuff

https://en.wikipedia.org/wiki/Sunbeam

*"... in 10 years, all rendering will be volume rendering."*

Jim Kajiya at SIGGRAPH '91

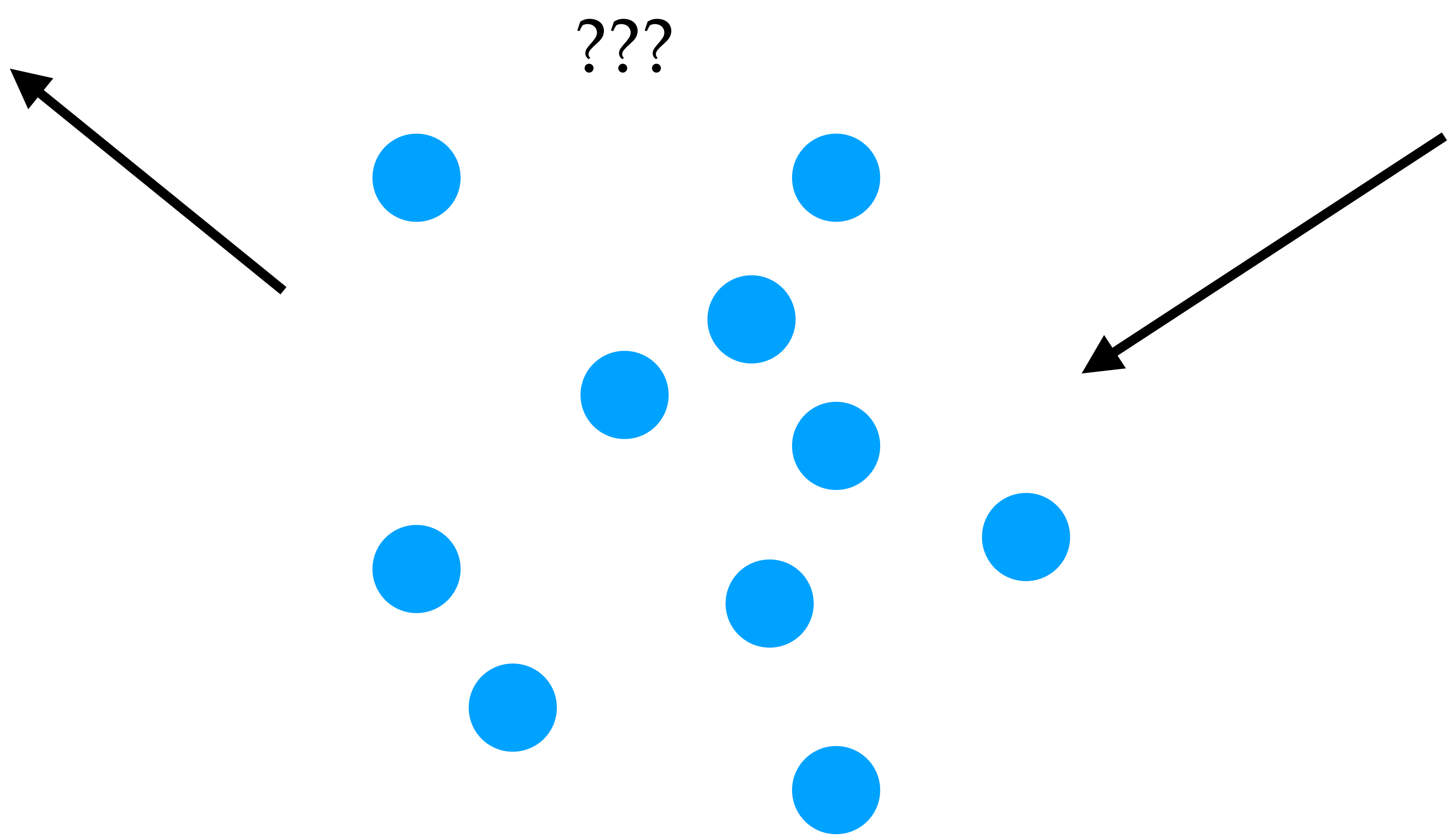
**A Survey of Algorithms for Volume Visualization**

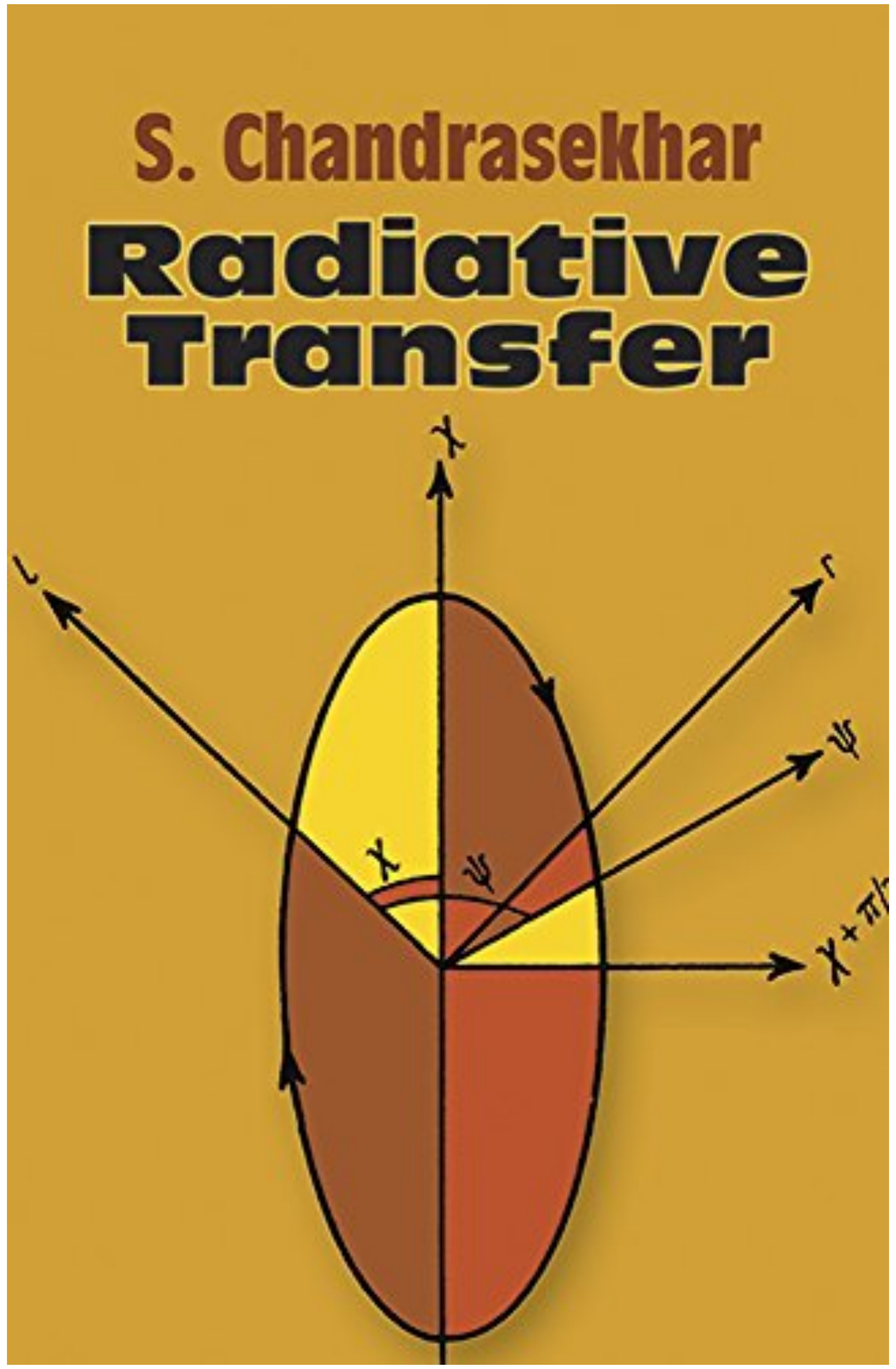
T. Todd Elvins

San Diego Supercomputer Center

# Foundation of modern rendering physics: radiative transfer [Chandrasekhar 1960]

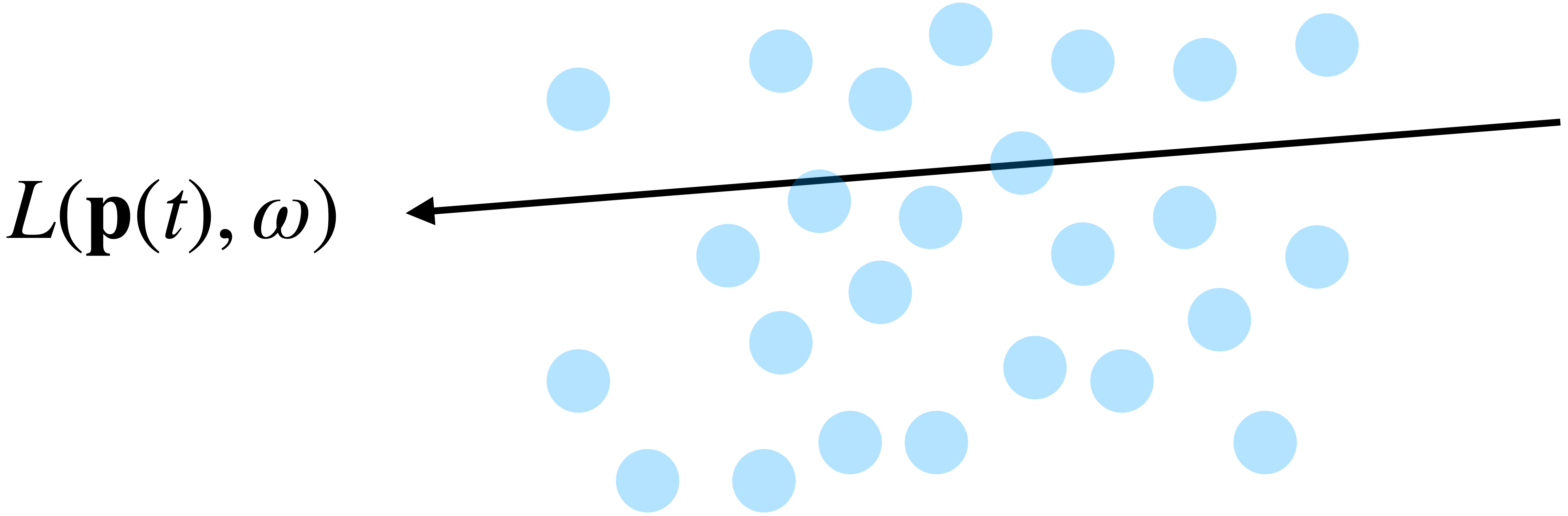
- what happens when light hits particles in the space?



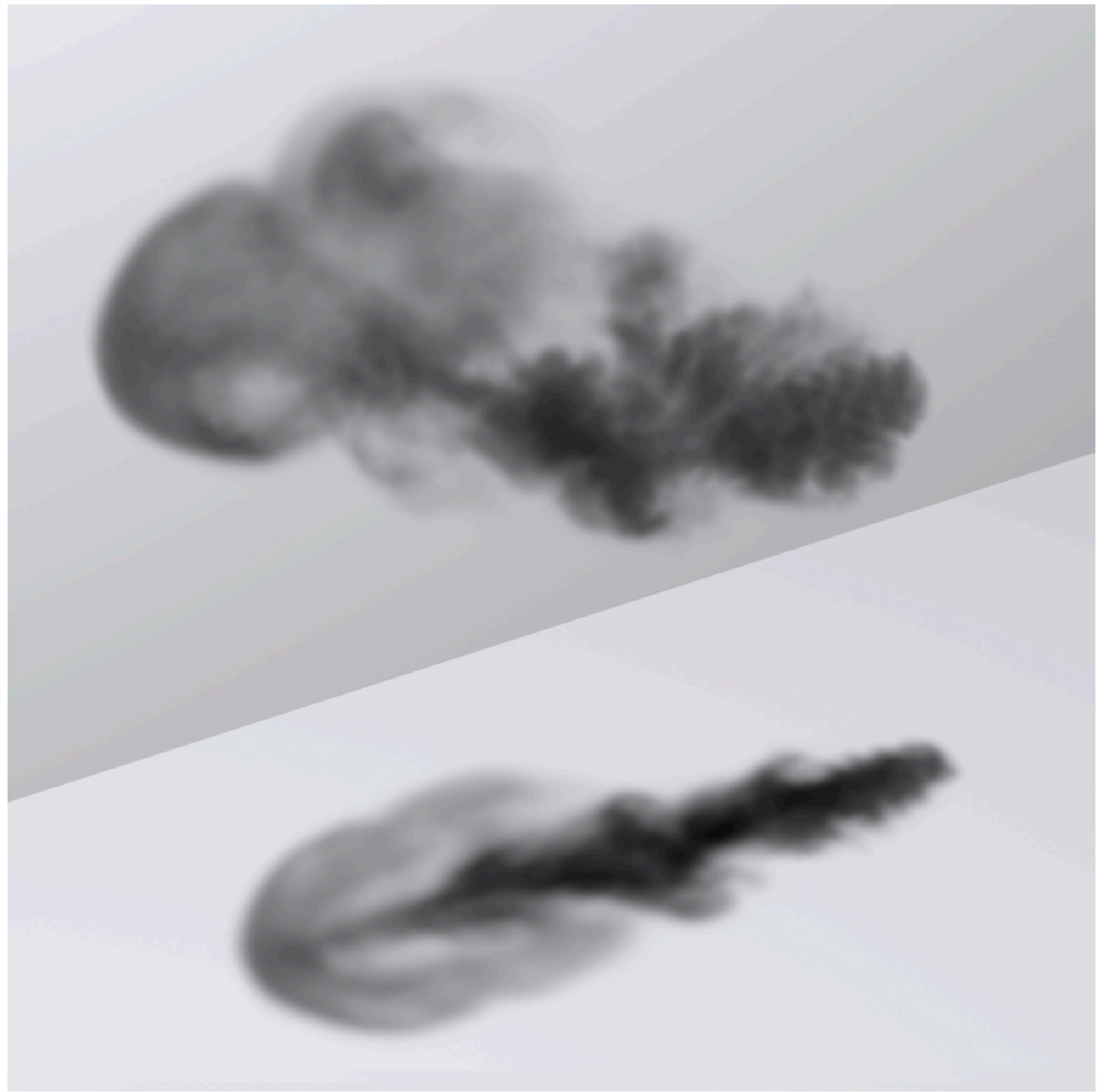



# Infinitely many particles:
# use ordinary differential equation to describe light's behavior

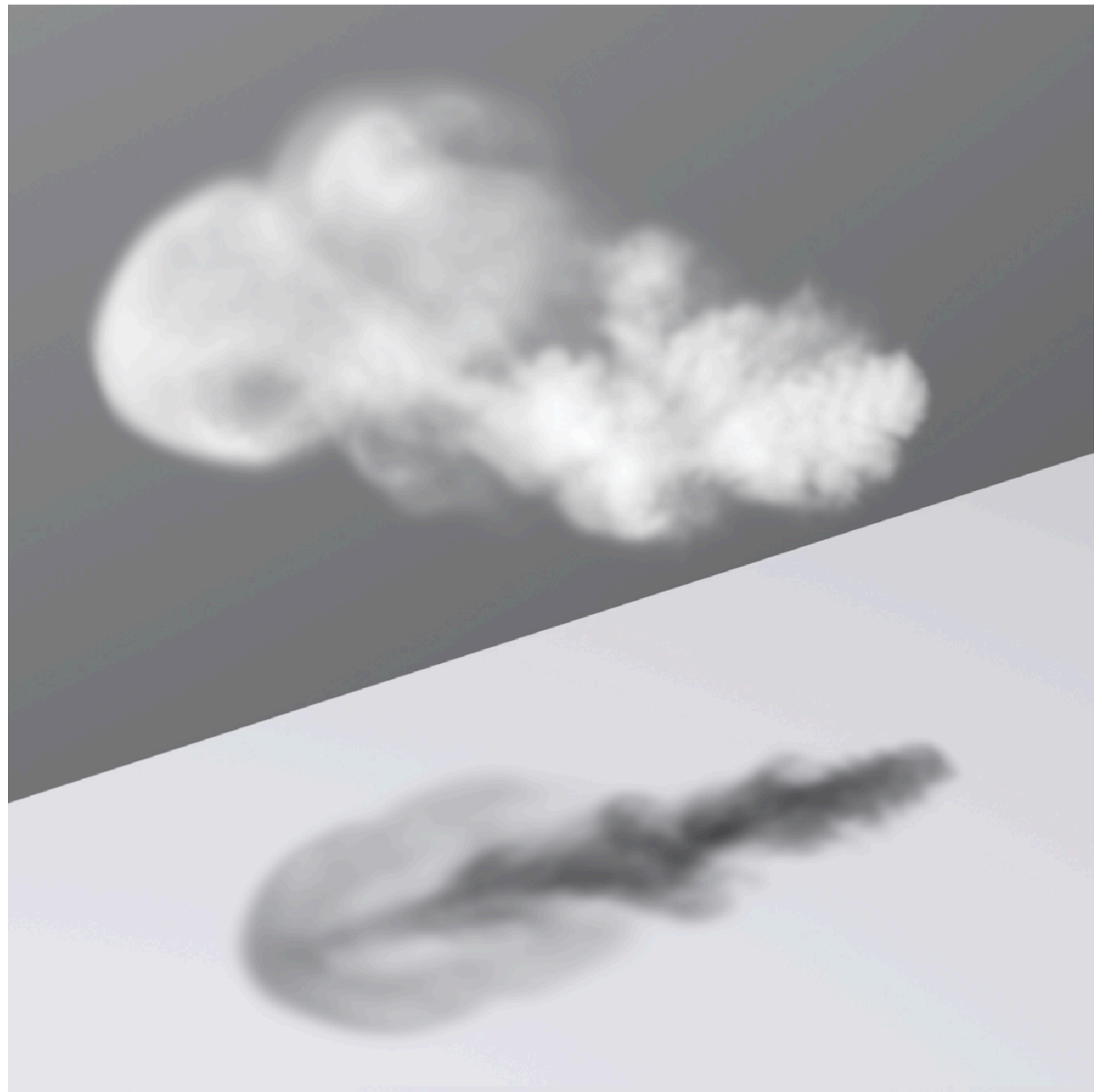
$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = ?$$



# Three volumetric phenomenon



absorption

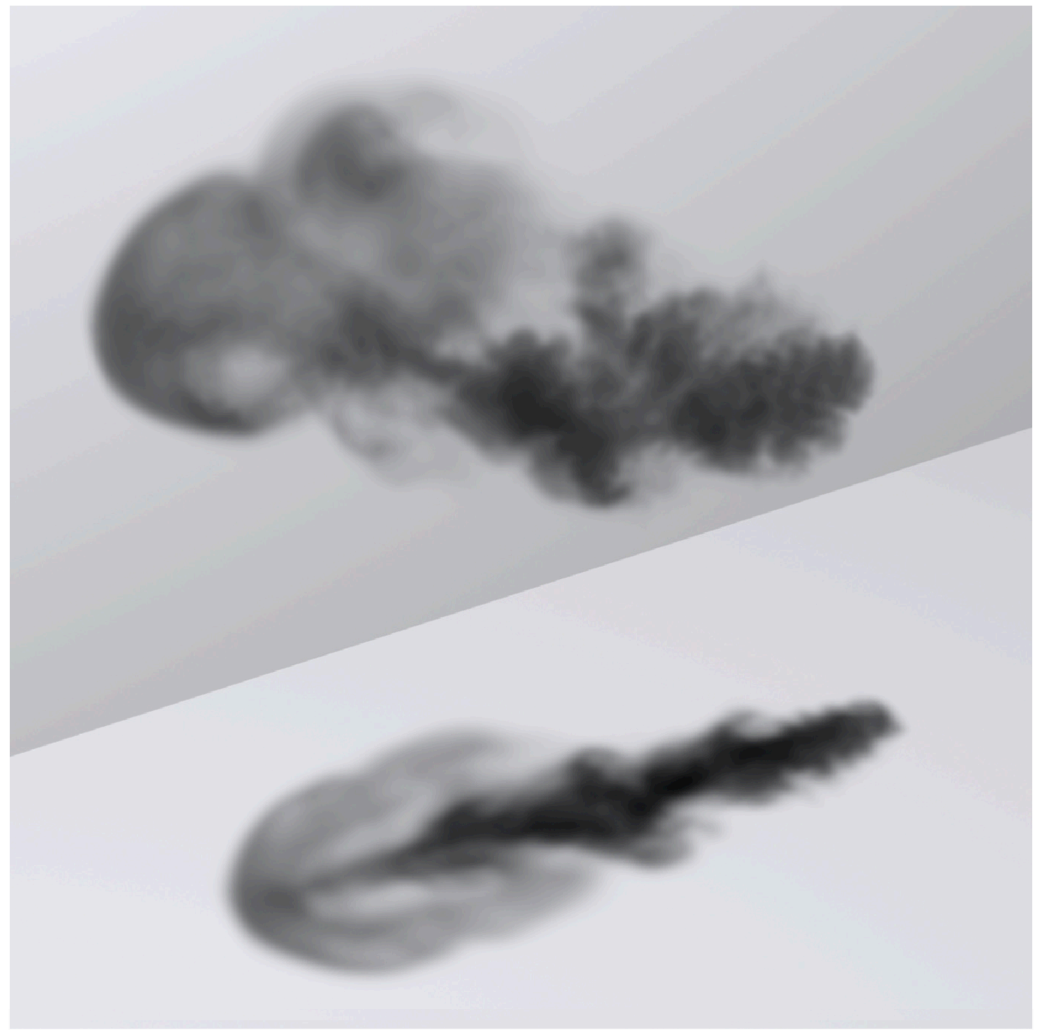


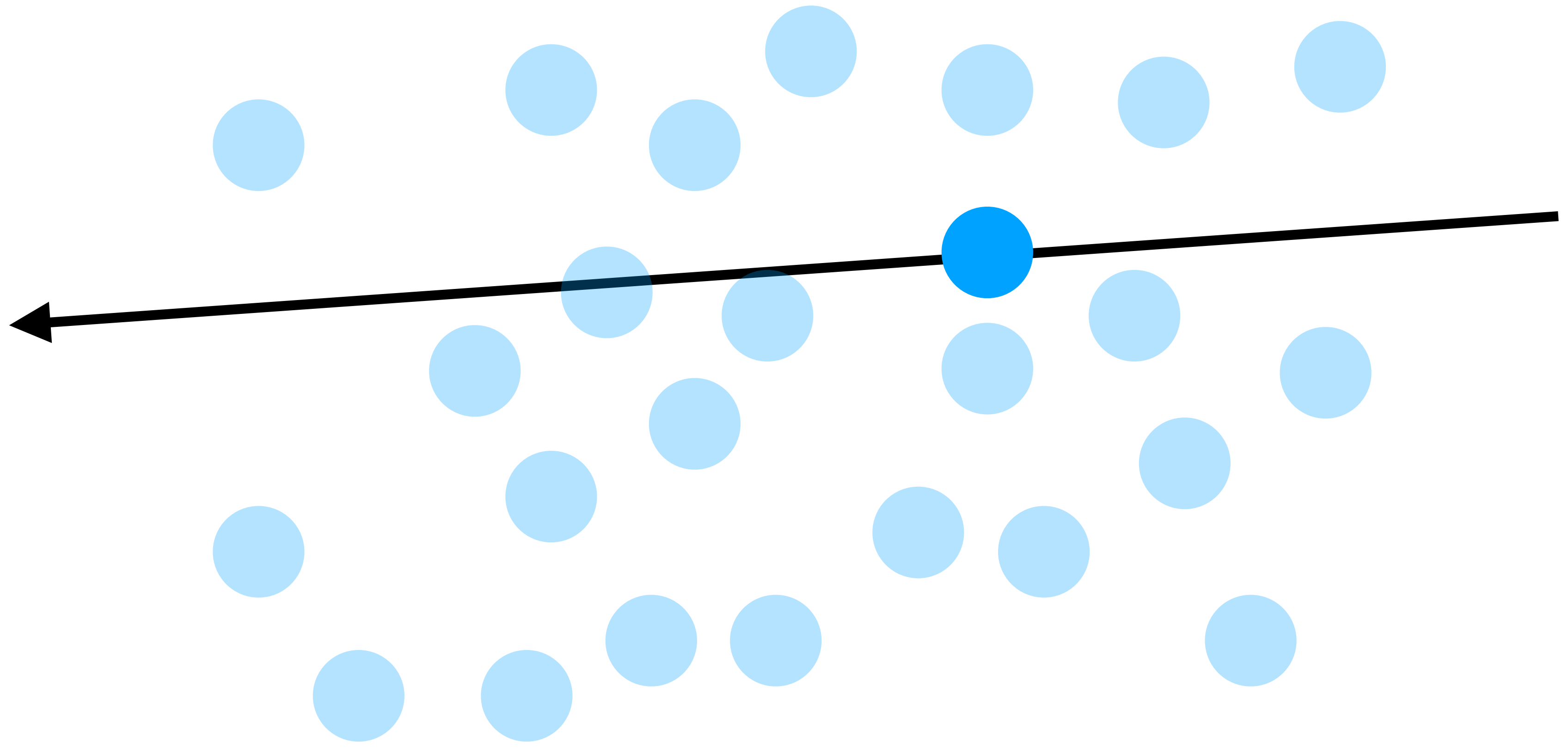
emission

scattering

https://www.pbr-book.org/3ed-2018/Volume_Scattering/Volume_Scattering_Processes
(smoke data from Duc Nguyen & Ron Fedkiw)

# Absorption

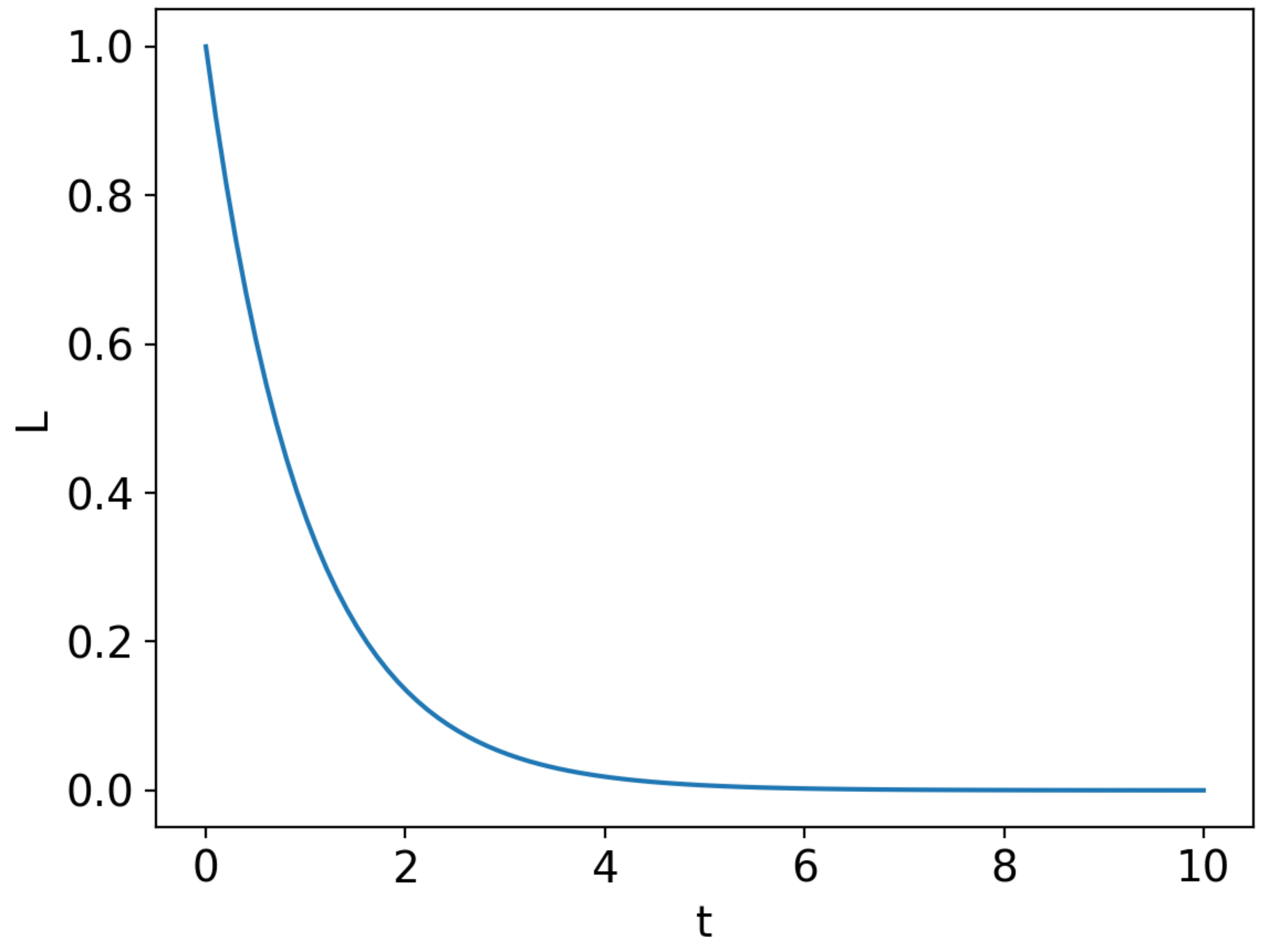




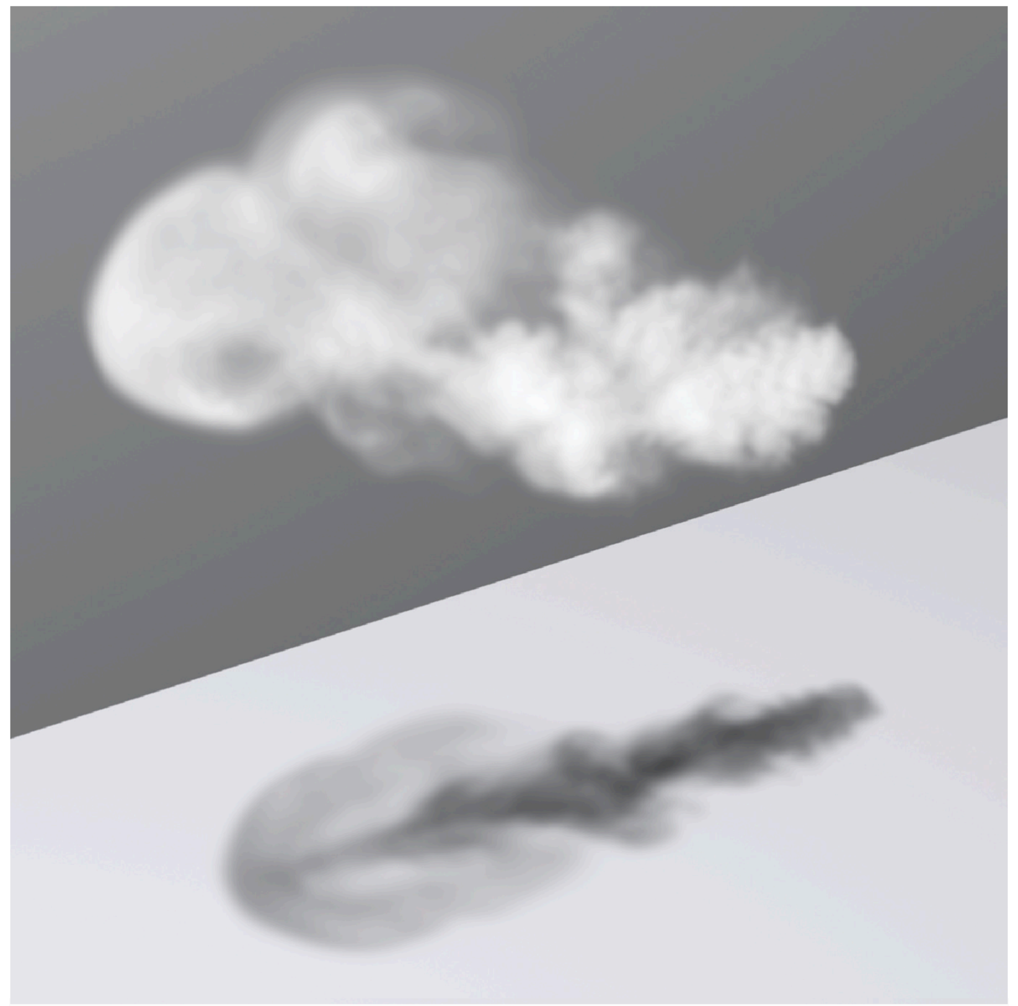
$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = -\sigma_a L(\mathbf{p}(t),\omega)$$

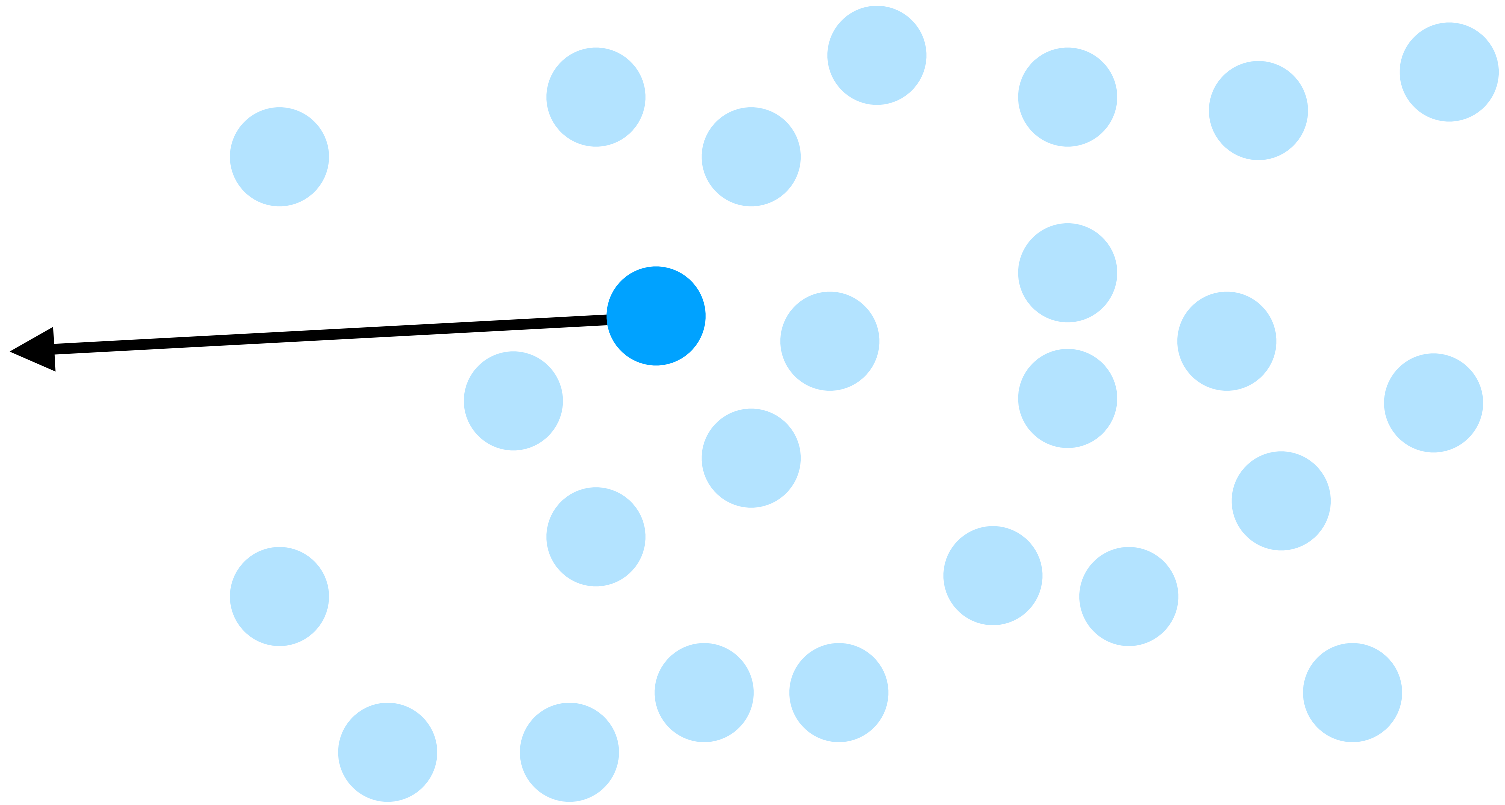
the particles absorb light's energy

(assumption: particles are independent to each other)



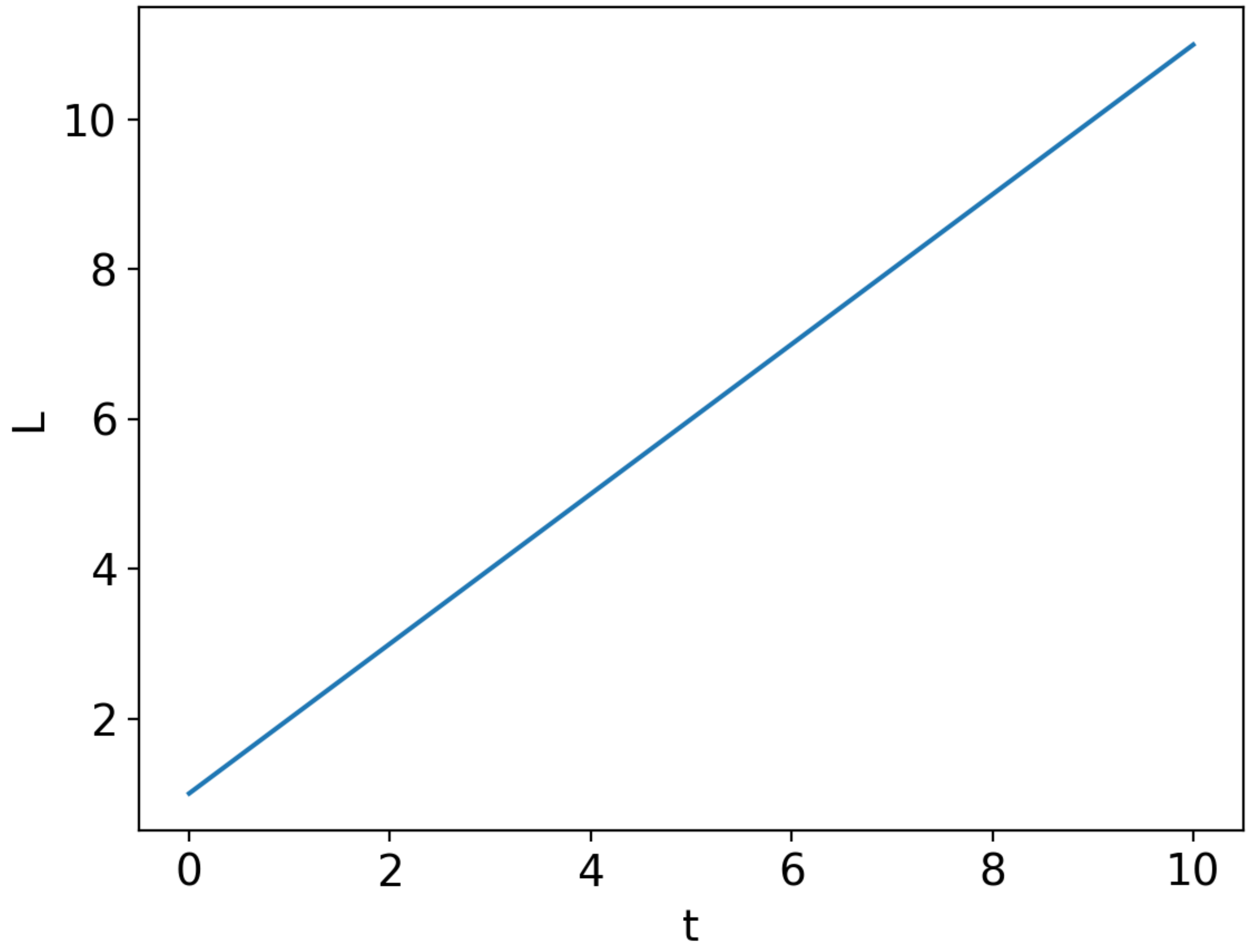
# Emission





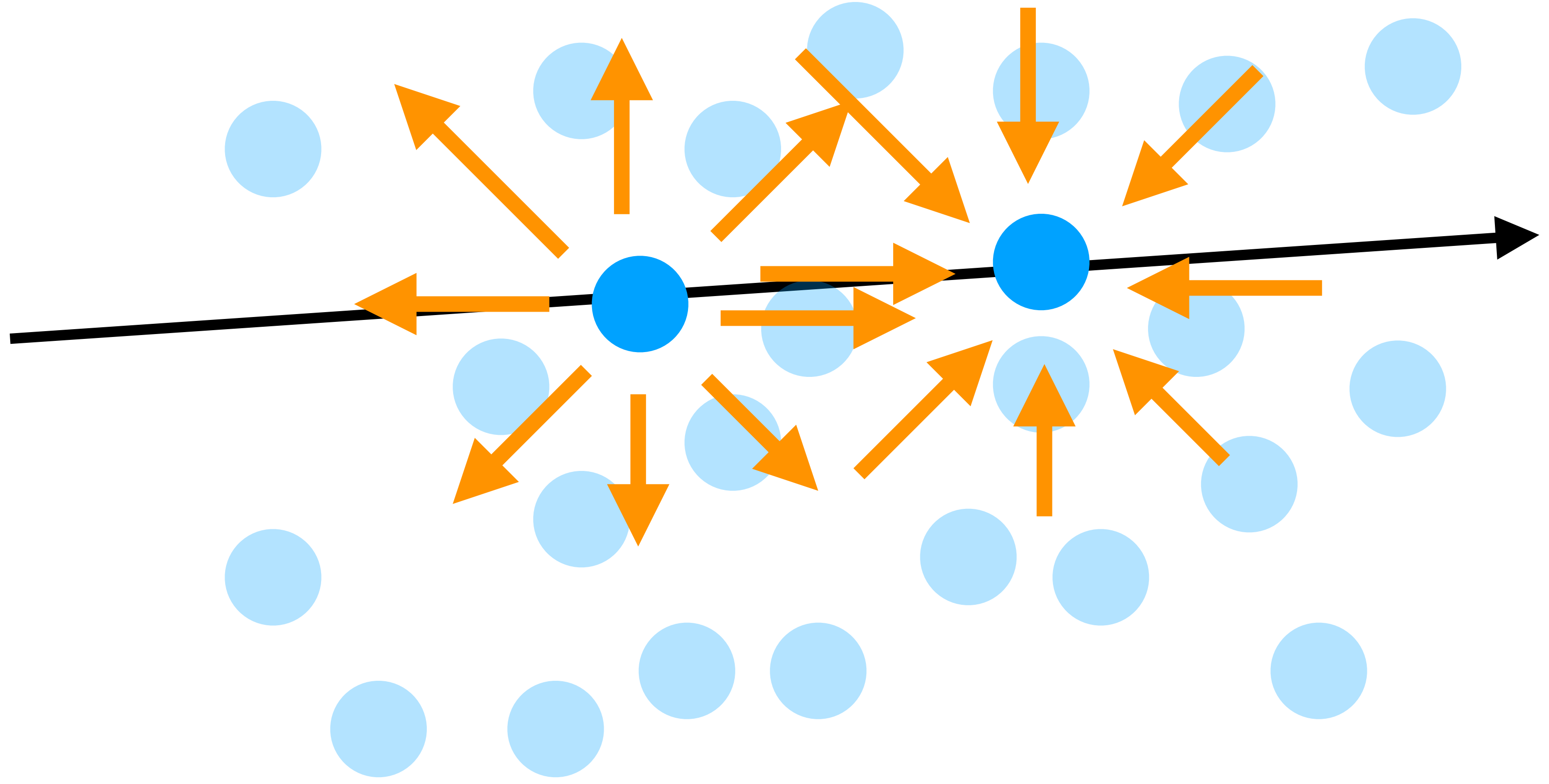
$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = L_e(\mathbf{p}(t),\omega)$$

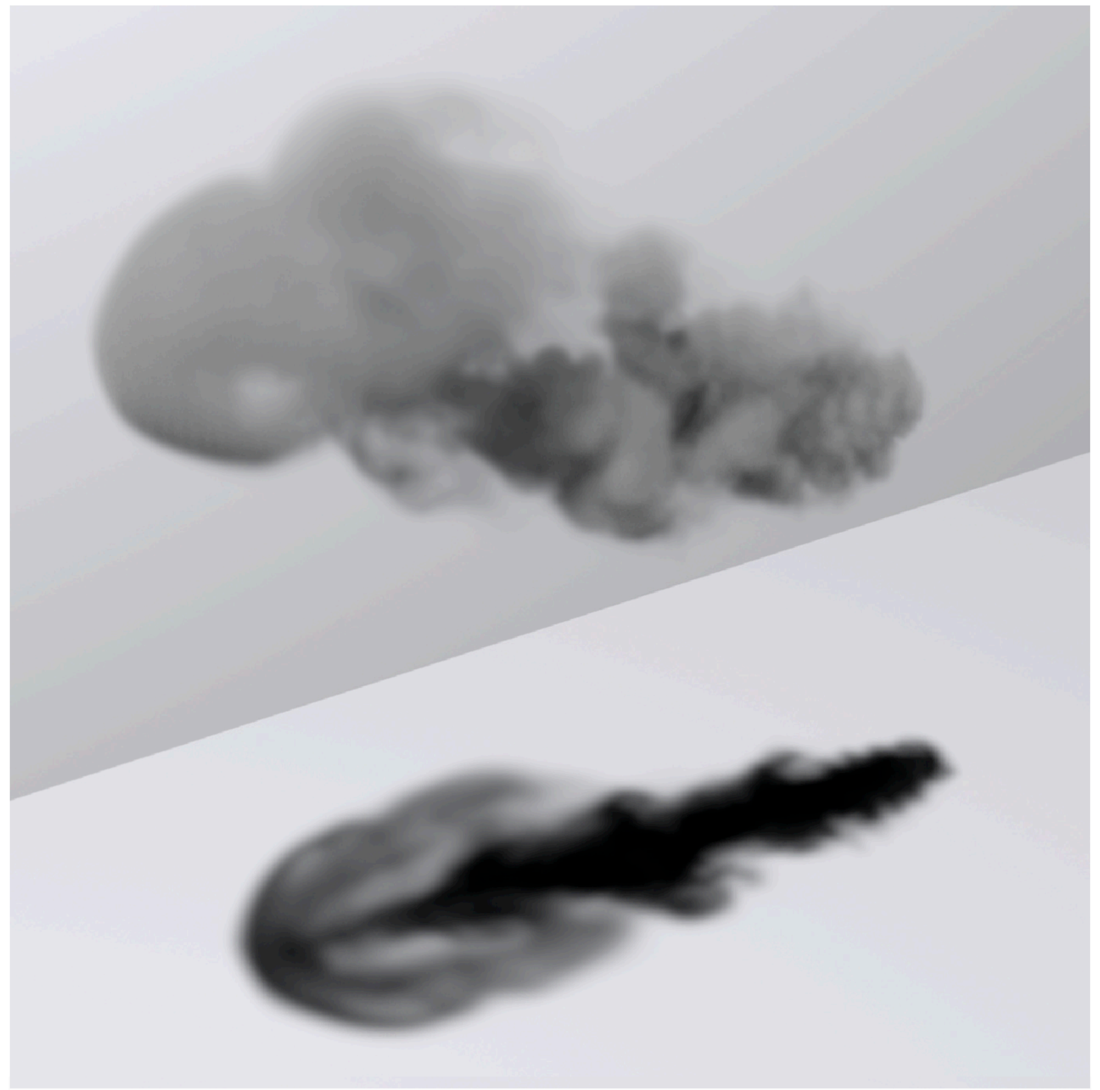
the particles add to light's energy




sometimes this is formulated as $\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = \sigma_a L_e(\mathbf{p}(t),\omega)$

# Scattering





out-scattering

$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = -\sigma_s L(\mathbf{p}(t),\omega)$$

in-scattering

$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = \sigma_s \int_{S^2} \rho(\omega,\omega')L(\mathbf{p}(t),\omega')\mathrm{d}\omega'$$

$\rho$: “phase function” (volume BSDF)

# Radiative Transfer Equation

absorption out-scattering

$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = \boxed{-\sigma_a L(\mathbf{p}(t),\omega) - \sigma_s L(\mathbf{p}(t),\omega)}$$

loss

$$\boxed{+L_e(\mathbf{p}(t),\omega) + \sigma_s \int_{S^2} \rho(\omega,\omega')L(\mathbf{p}(t),\omega')\mathrm{d}\omega'}$$

emission in-scattering

gain

# Radiative Transfer Equation

$$\sigma_t = \sigma_a + \sigma_s$$

extinction

$$\frac{\mathrm{d}}{\mathrm{d}t} L(\mathbf{p}(t), \omega) = \boxed{-\sigma_t L(\mathbf{p}(t), \omega)}$$

loss

$$\boxed{+L_e(\mathbf{p}(t), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega') L(\mathbf{p}(t), \omega') \mathrm{d}\omega'}$$

emission

in-scattering

gain

# A simpler case: volume without scattering

let $\sigma_s = 0$

$$\frac{\mathrm{d}}{\mathrm{d}t} L(\mathbf{p}(t), \omega) = -\sigma_t L(\mathbf{p}(t), \omega)$$

$$+ L_e(\mathbf{p}(t), \omega) + \cancel{\sigma_s \int_{S^2} \rho(\omega, \omega') L(\mathbf{p}(t), \omega') \mathrm{d}\omega'}$$

How would you solve for $L$?

# A simpler case: volume without scattering

let $\sigma_s = 0$

$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = -\sigma_t L(\mathbf{p}(t),\omega) + L_e(\mathbf{p}(t),\omega)$$

$$\frac{\mathrm{d}}{\mathrm{d}t}L(t) = a(t)L(t) + b(t)$$

it's a linear ODE that has an analytical solution (**quiz**: what is it?)

# A simpler case: volume without scattering

let $\sigma_s = 0$

$$\frac{\mathrm{d}}{\mathrm{d}t} L(\mathbf{p}(t), \omega) = -\sigma_t L(\mathbf{p}(t), \omega) + L_e(\mathbf{p}(t), \omega)$$

$$\frac{\mathrm{d}}{\mathrm{d}t} L(t) = a(t) L(t) + b(t)$$

$$L(t) = \int_0^t T(t) L_e(t) \mathrm{d}t \qquad T(t) = \exp\left( -\int_0^t \sigma_t(t') \mathrm{d}t' \right)$$

# A simpler case: volume without scattering

let $\sigma_s = 0$

$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = -\sigma_t L(\mathbf{p}(t),\omega) + L_e(\mathbf{p}(t),\omega)$$

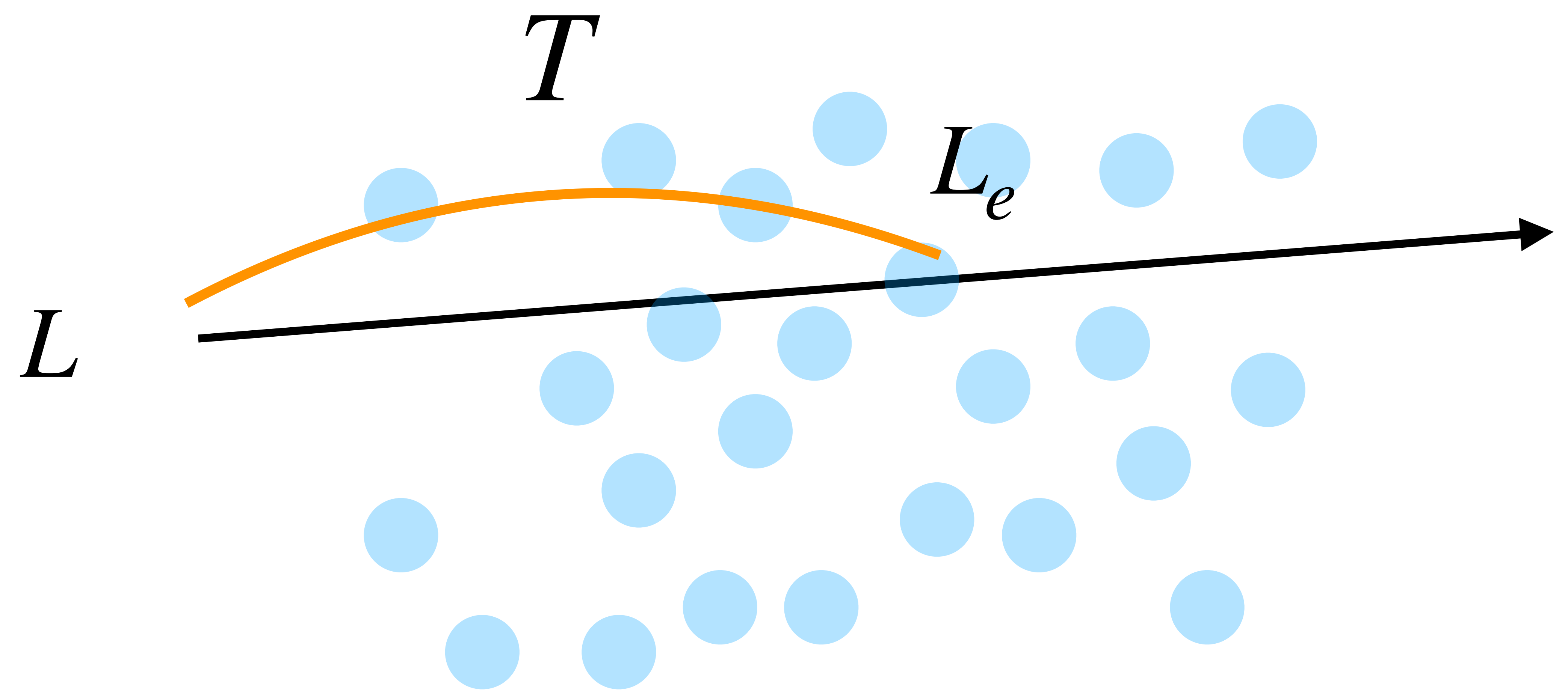
$$\frac{\mathrm{d}}{\mathrm{d}t}L(t) = a(t)L(t) + b(t)$$

$$L(t) = \int_0^t T(t)L_e(t)\mathrm{d}t$$

"transmittance"

$$\boxed{T(t) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)}$$

# A simpler case: volume without scattering




$$L(t) = \int_0^t T(t) L_e(t) \mathrm{d}t \qquad T(t) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

# The full radiative transfer equation is still a linear ODE

$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t), \omega) = -\sigma_t L(\mathbf{p}(t), \omega) + L_e(\mathbf{p}(t), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega') L(\mathbf{p}(t), \omega') \mathrm{d}\omega'$$

$$\frac{\mathrm{d}}{\mathrm{d}t}L(t) = a(t)L(t) + b(t)$$

# Integral form of radiative transfer equation

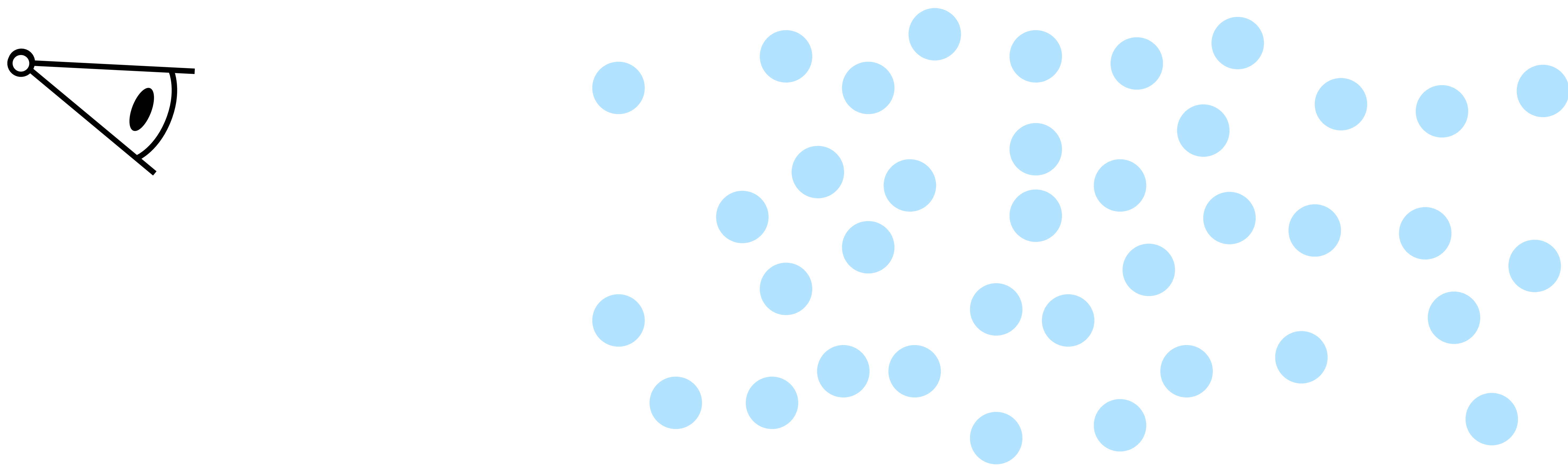
$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = -\sigma_t L(\mathbf{p}(t),\omega) + L_e(\mathbf{p}(t),\omega) + \sigma_s \int_{S^2} \rho(\omega,\omega')L(\mathbf{p}(t),\omega')\mathrm{d}\omega'$$

$$L(\mathbf{p}(0),\omega) = \int_0^t T\left(\mathbf{p}(0),\mathbf{p}(t')\right)\left[L_e(\mathbf{p}(t'),\omega) + \sigma_s \int_{S^2} \rho(\omega,\omega')L(\mathbf{p}(t'),\omega')\mathrm{d}\omega'\right]\mathrm{d}t'$$

$$T(\mathbf{p}(0),\mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

# Volumetric path tracing

- the inclusion of the transmittance is the main difference to surface rendering equation



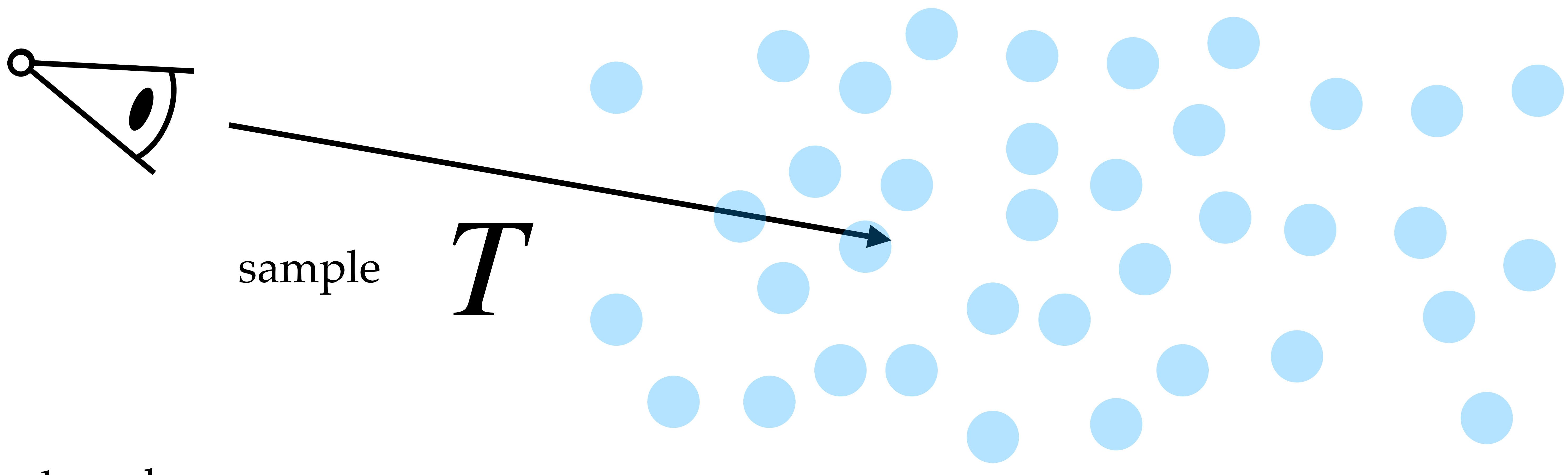
**quiz**: how would you do it?

$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

$$L(\mathbf{p}(0), \omega) = \int_0^t T\left(\mathbf{p}(0), \mathbf{p}(t')\right) \left[L_e(\mathbf{p}(t'), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega')L(\mathbf{p}(t'), \omega')\mathrm{d}\omega'\right] \mathrm{d}t'$$

# Volumetric path tracing

- the inclusion of the transmittance is the main difference to surface rendering equation



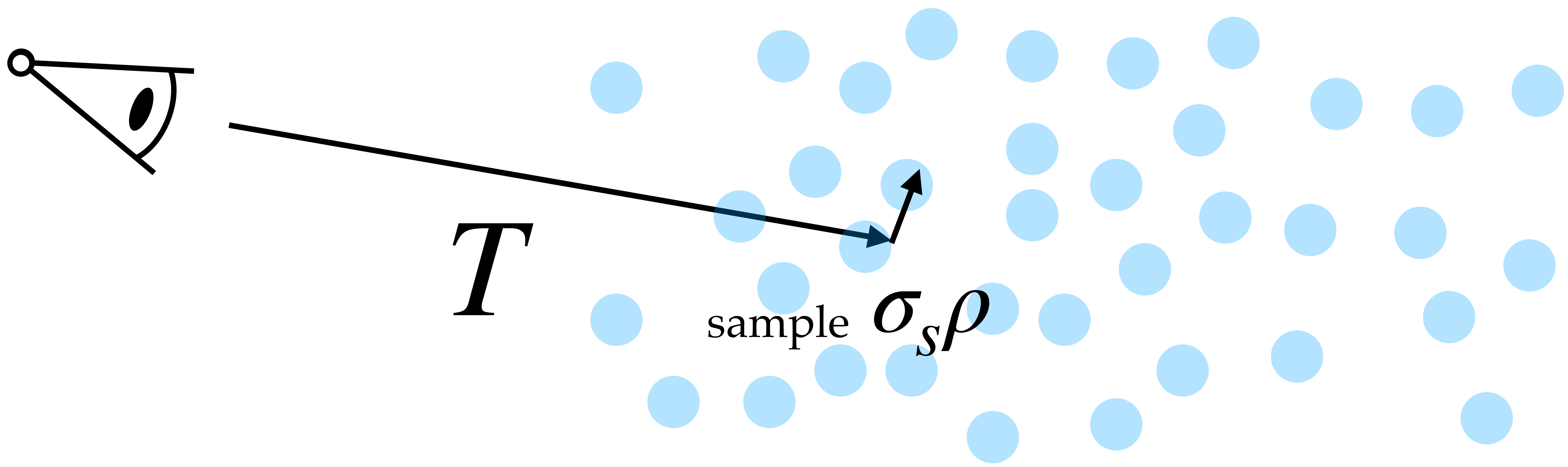

will talk about how to sample T next time

$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

$$L(\mathbf{p}(0), \omega) = \int_0^t T\left(\mathbf{p}(0), \mathbf{p}(t')\right) \left[L_e(\mathbf{p}(t'), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega')L(\mathbf{p}(t'), \omega')\mathrm{d}\omega'\right] \mathrm{d}t'$$

# Volumetric path tracing

- the inclusion of the transmittance is the main difference to surface rendering equation

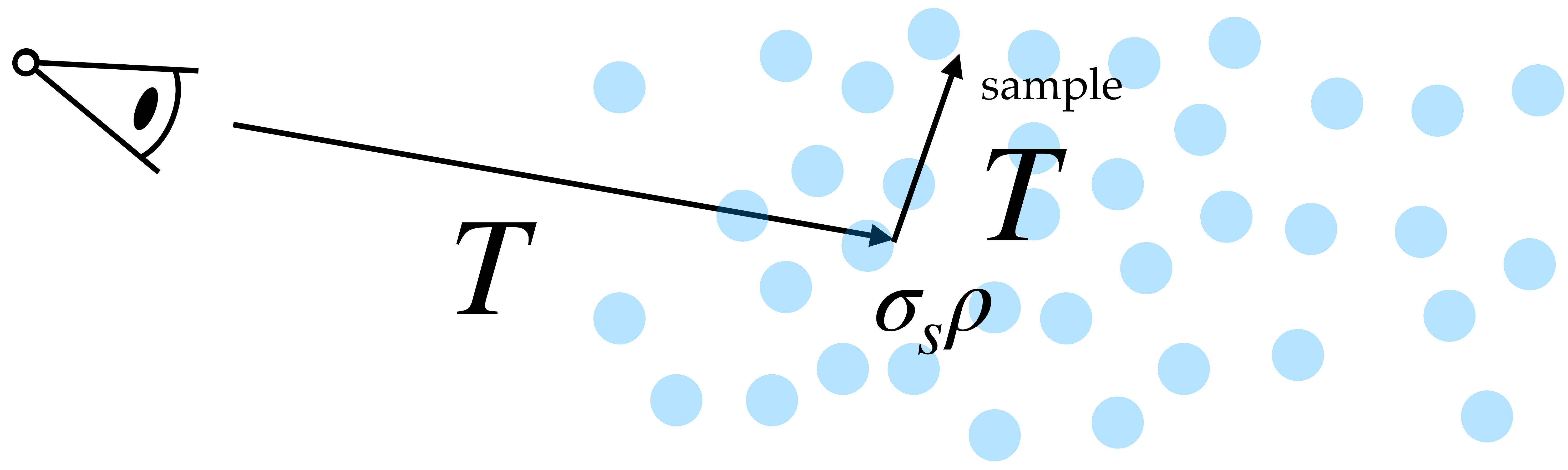



$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

$$L(\mathbf{p}(0), \omega) = \int_0^t T\left(\mathbf{p}(0), \mathbf{p}(t')\right)\left[L_e(\mathbf{p}(t'), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega')L(\mathbf{p}(t'), \omega')\mathrm{d}\omega'\right]\mathrm{d}t'$$

# Volumetric path tracing

- the inclusion of the transmittance is the main difference to surface rendering equation

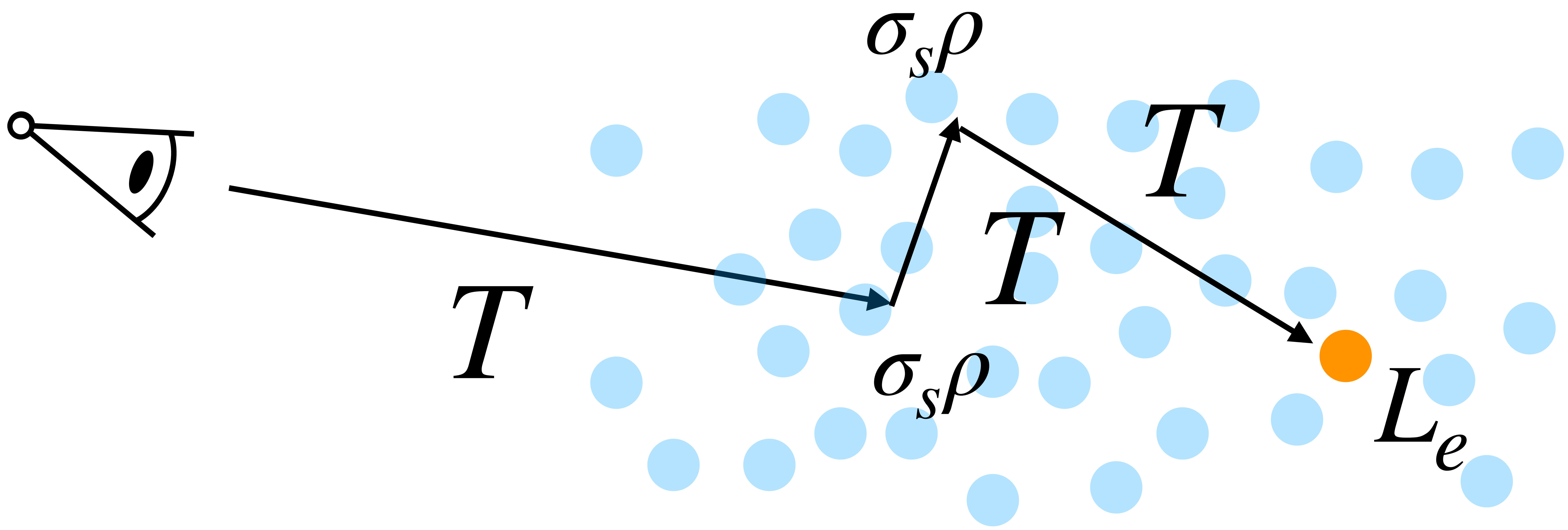



$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

$$L(\mathbf{p}(0), \omega) = \int_0^t T\left(\mathbf{p}(0), \mathbf{p}(t')\right)\left[L_e(\mathbf{p}(t'), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega')L(\mathbf{p}(t'), \omega')\mathrm{d}\omega'\right]\mathrm{d}t'$$

# Volumetric path tracing

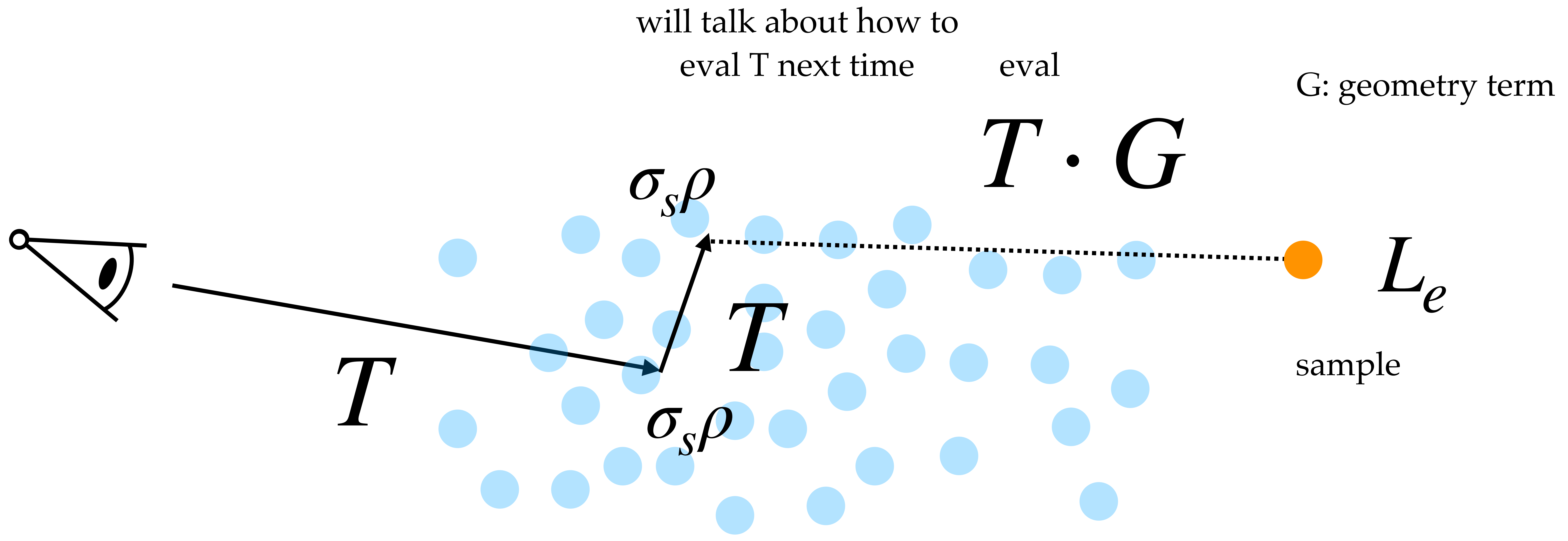
- the inclusion of the transmittance is the main difference to surface rendering equation




$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

$$L(\mathbf{p}(0), \omega) = \int_0^t T\left(\mathbf{p}(0), \mathbf{p}(t')\right) \left[L_e(\mathbf{p}(t'), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega')L(\mathbf{p}(t'), \omega')\mathrm{d}\omega'\right] \mathrm{d}t'$$

# Next event estimation in volumetric path tracing

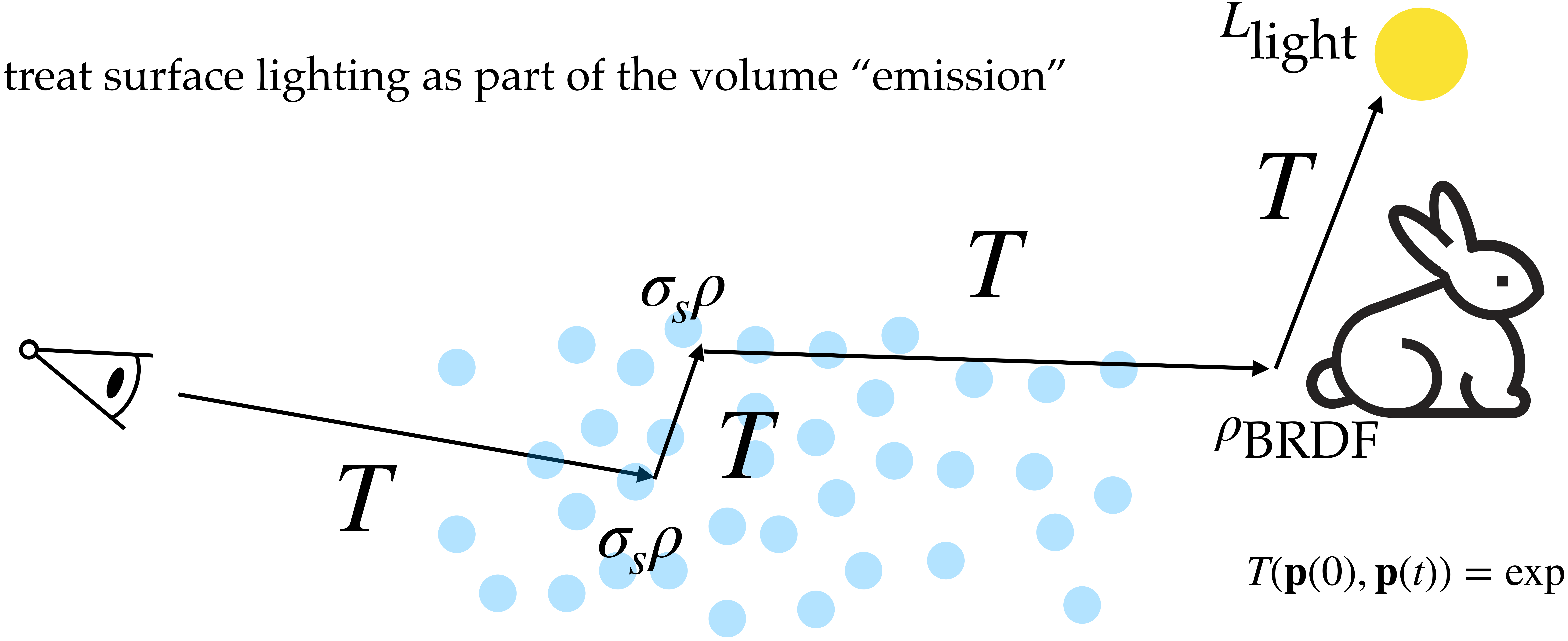



$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

$$L(\mathbf{p}(0), \omega) = \int_0^t T\left(\mathbf{p}(0), \mathbf{p}(t')\right)\left[L_e(\mathbf{p}(t'), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega')L(\mathbf{p}(t'), \omega')\mathrm{d}\omega'\right]\mathrm{d}t'$$

# Inclusion surface lighting in volume rendering

- treat surface lighting as part of the volume "emission"

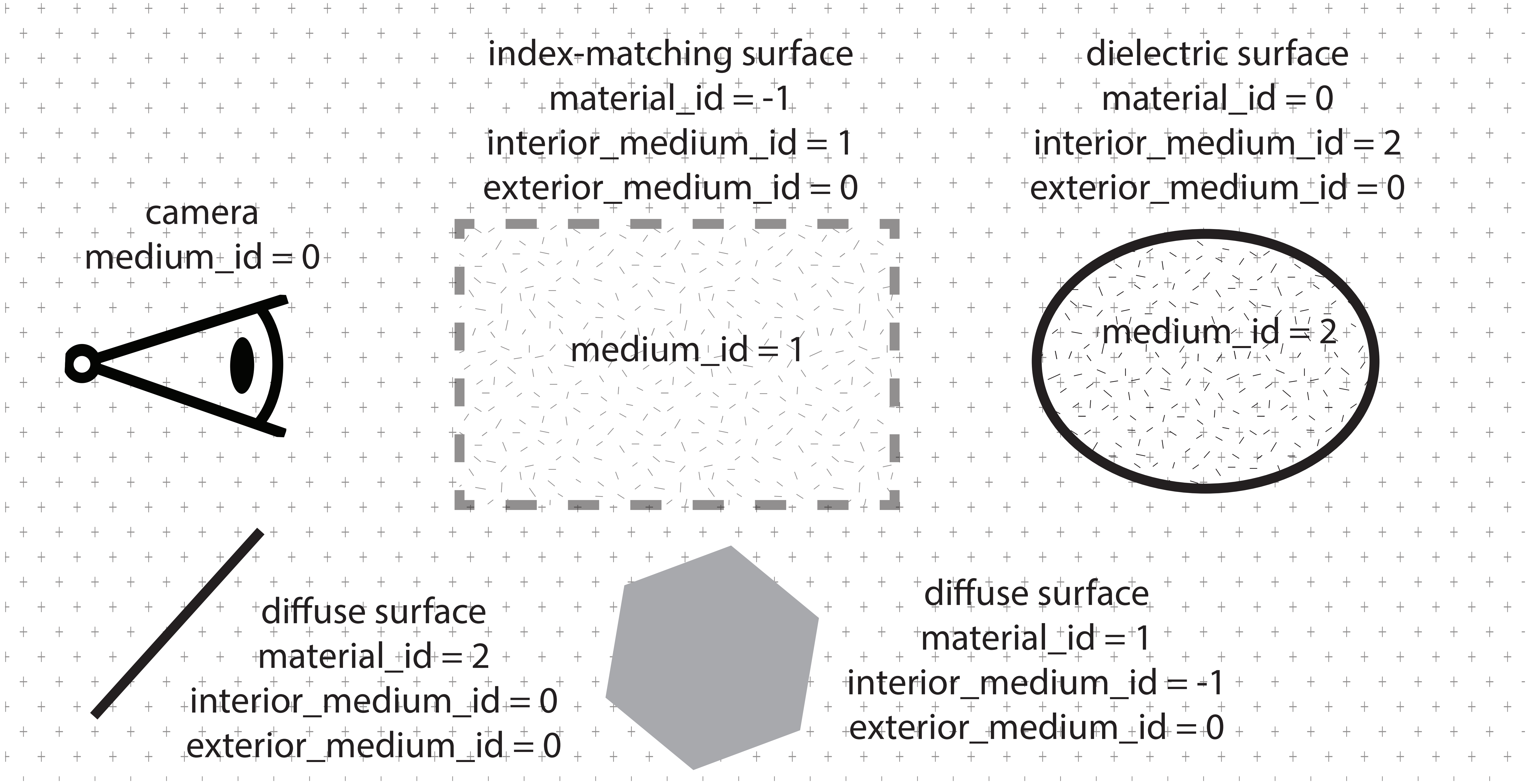



$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t'\right)$$

$$L(\mathbf{p}(0), \omega) = \int_0^t T\left(\mathbf{p}(0), \mathbf{p}(t')\right)\left[L_e(\mathbf{p}(t'), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega')L(\mathbf{p}(t'), \omega')\mathrm{d}\omega'\right]\mathrm{d}t'$$

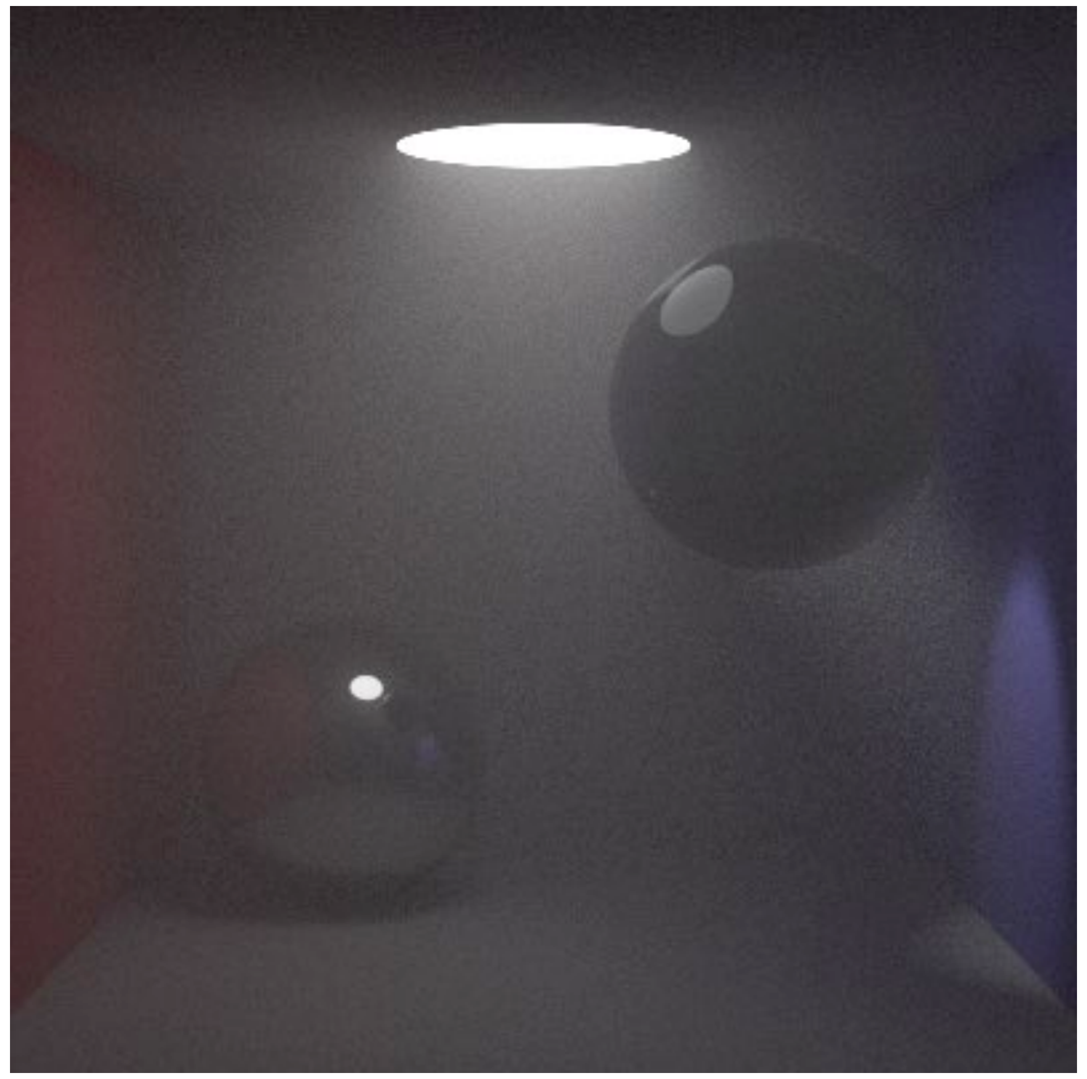
# Typical volume data structures in a path tracer

- use geometry as boundaries, store a medium inside each geometry



# Smallvpt: volume path tracing in 150 lines

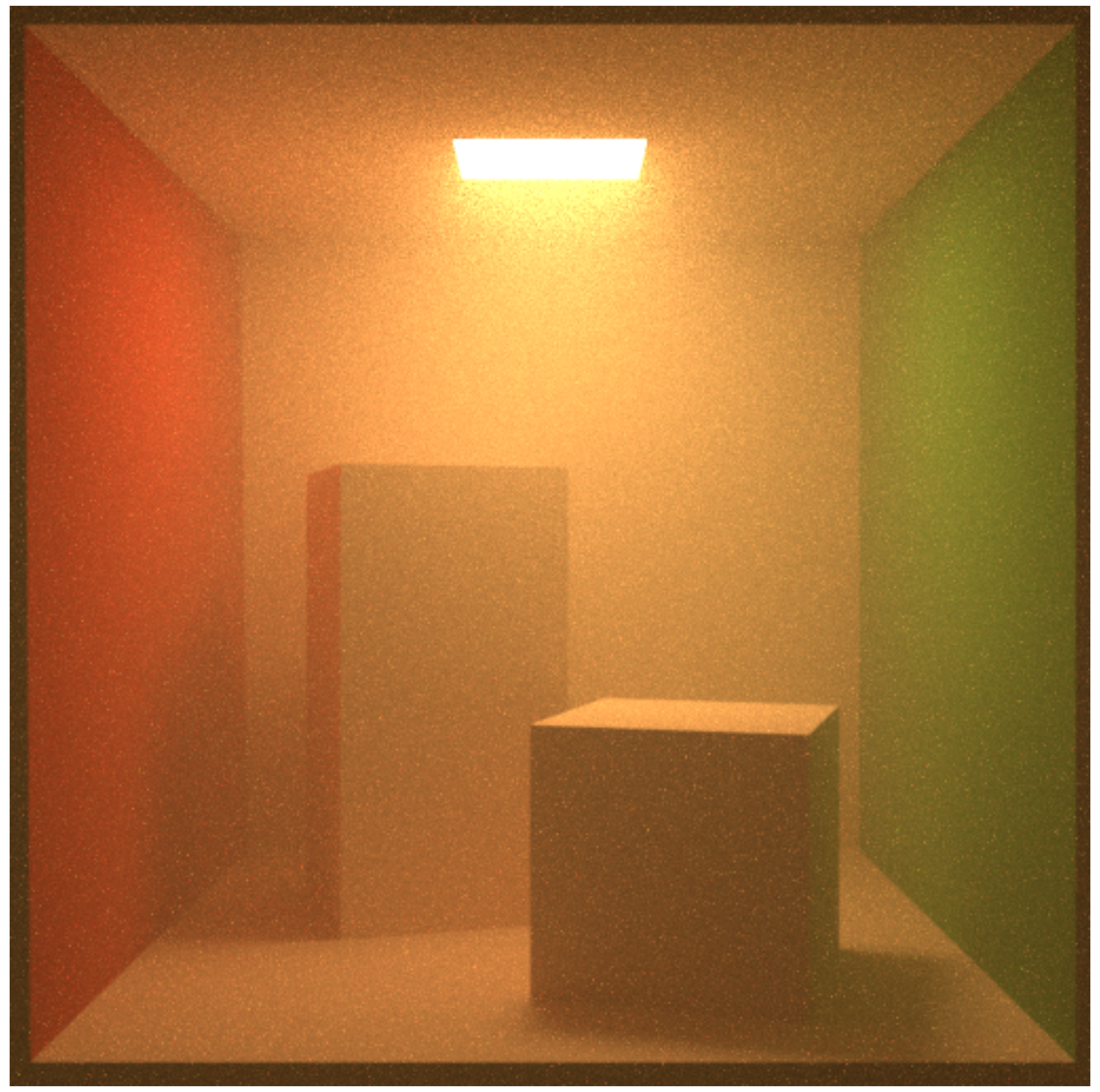
```cpp
#define _USE_MATH_DEFINES
#include <math.h>   // smallpt, a Path Tracer by Kevin Beason, 2008
#include <stdlib.h> // Make : g++ -O3 -fopenmp smallpt.cpp -o smallpt
#include <stdio.h>  //        Remove "-fopenmp" for g++ version < 4.2
#include <algorithm>
#pragma warning(disable: 4244) // Disable double to float warning
namespace XORShift { // XOR shift PRNG
    unsigned int x = 123456789;
    unsigned int y = 362436069;
    unsigned int z = 521288629;
    unsigned int w = 88675123;
    inline float frand() {
        unsigned int t;
        t = x ^ (x << 11);
        x = y; y = z; z = w;
        return (w = (w ^ (w >> 19)) ^ (t ^ (t >> 8))) * (1.0f / 4294967295.0f);
    }
}
struct Vec {        // Usage: time ./smallpt 5000 && xv image.ppm
    double x, y, z;                  // position, also color (r,g,b)
    Vec(double x_=0, double y_=0, double z_=0){ x=x_; y=y_; z=z_; }
    Vec operator+(const Vec &b) const { return Vec(x+b.x,y+b.y,z+b.z); }
    Vec operator-(const Vec &b) const { return Vec(x-b.x,y-b.y,z-b.z); }
    Vec operator*(double b) const { return Vec(x*b,y*b,z*b); }
    Vec mult(const Vec &b) const { return Vec(x*b.x,y*b.y,z*b.z); }
    Vec& norm(){ return *this = *this * (1/sqrt(x*x+y*y+z*z)); }
    float length() {return sqrt(x*x+y*y+z*z); }
    double dot(const Vec &b) const { return x*b.x+y*b.y+z*b.z; } // cross:
    Vec operator%(Vec&b){return Vec(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);}
};
struct Ray { Vec o, d; Ray() {} Ray(Vec o_, Vec d_) : o(o_), d(d_) {} };
enum Refl_t { DIFF, SPEC, REFR };  // material types, used in radiance()
struct Sphere {
    double rad;       // radius
    Vec p, e, c;      // position, emission, color
    Refl_t refl;      // reflection type (DIFFuse, SPECular, REFRactive)
    Sphere(double rad_, Vec p_, Vec e_, Vec c_, Refl_t refl_):
    rad(rad_), p(p_), e(e_), c(c_), refl(refl_) {}
    double intersect(const Ray &r, double *tin = NULL, double *tout = NULL) const { // returns distance, 0 if nohit
        Vec op = p-r.o; // Solve t^2*d.d + 2*t*(o-p).d + (o-p).(o-p)-R^2 = 0
        double t, eps=1e-4, b=op.dot(r.d), det=b*b-op.dot(op)+rad*rad;
        if (det<0) return 0; else det=sqrt(det);
        if (tin && tout) {*tin=(b-det<=0)?0:b-det;*tout=b+det;}
        return (t=b-det)>eps ? t : ((t=b+det)>eps ? t : 0);
    }
};
Sphere spheres[] = {//Scene: radius, position, emission, color, material
    Sphere(26.5,Vec(27,18.5,78),       Vec(),Vec(1,1,1)*.75, SPEC),//Mirr
    Sphere(12,Vec(70,43,78),       Vec(),Vec(0.27,0.8,0.8), REFR),//Glas
    Sphere(8, Vec(55,87,78),Vec(),  Vec(1,1,1)*.75, DIFF), //Lite
    Sphere(4, Vec(55,80,78),Vec(10,10,10),  Vec(), DIFF) //Lite
};
Sphere homogeneousMedium(300, Vec(50,50,80), Vec(), Vec(), DIFF);
const double sigma_s = 0.009, sigma_a = 0.006, sigma_t = sigma_s+sigma_a;
inline double clamp(double x){ return x<0 ? 0 : x>1 ? 1 : x; }
inline int toInt(double x){ return int(pow(clamp(x),1/2.2)*255+.5); }
inline bool intersect(const Ray &r, double &t, int &id, double tmax=1e20){
    double n=sizeof(spheres)/sizeof(Sphere), d, inf=t=tmax;
    for(int i=int(n);i--;) if((d=spheres[i].intersect(r))&&d<t){t=d;id=i;}
    return t<inf;
}
inline double sampleSegment(double epsilon, float sigma, float smax) {
    return -log(1.0 - epsilon * (1.0 - exp(-sigma * smax))) / sigma;
}
inline Vec sampleSphere(double e1, double e2) {
    double z = 1.0 - 2.0 * e1, sint = sqrt(1.0 - z * z);
    return Vec(cos(2.0 * M_PI * e2) * sint, sin(2.0 * M_PI * e2) * sint, z);
}
inline Vec sampleHG(double g, double e1, double e2) {
    //double s=2.0*e1-1.0, f = (1.0-g*g)/(1.0+g*s), cost = 0.5*(1.0/g)*(1.0+g*g-f*f), sint = sqrt(1.0-cost*cost);
    double s = 1.0-2.0*e1, cost = (s + 2.0*g*g*g * (-1.0 + e1) * e1 + g*g*s + 2.0*g*(1.0 - e1+e1*e1))/((1.0+g*s)*(1.0+g*s)), sint = sqrt(1.0-cost*cost);
    return Vec(cos(2.0 * M_PI * e2) * sint, sin(2.0 * M_PI * e2) * sint, cost);
}
inline void generateOrthoBasis(Vec &u, Vec &v, Vec w) {
    Vec coVec = w;
    if (fabs(w.x) <= fabs(w.y))
        if (fabs(w.x) <= fabs(w.z)) coVec = Vec(0,-w.z,w.y);
        else coVec = Vec(-w.y,w.x,0);
    else if (fabs(w.y) <= fabs(w.z)) coVec = Vec(-w.z,0,w.x);
    else coVec = Vec(-w.y,w.x,0);
    coVec.norm();
    u = w%coVec,
    v = w%u;
}
inline double scatter(const Ray &r, Ray *sRay, double tin, float tout, double &s) {
    s = sampleSegment(XORShift::frand(), sigma_s, tout - tin);
    Vec x = r.o + r.d *tin + r.d * s;
    //Vec dir = sampleSphere(XORShift::frand(), XORShift::frand()); //Sample a direction ~ uniform phase function
    Vec dir = sampleHG(-0.5,XORShift::frand(), XORShift::frand()); //Sample a direction ~ Henyey-Greenstein's phase function
    Vec u,v;
    generateOrthoBasis(u,v,r.d);
    dir = u*dir.x+v*dir.y+r.d*dir.z;
    if (sRay)       *sRay = Ray(x, dir);
    return (1.0 - exp(-sigma_s * (tout - tin)));
}
Vec radiance(const Ray &r, int depth) {
    double t;                               // distance to intersection
    int id=0;                               // id of intersected object
    double tnear, tfar, scaleBy=1.0, absorption=1.0;
    bool intrsctmd = homogeneousMedium.intersect(r, &tnear, &tfar) > 0;
    if (intrsctmd) {
        Ray sRay;
        double s, ms = scatter(r, &sRay, tnear, tfar, s), prob_s = ms;
        scaleBy = 1.0/(1.0-prob_s);
        if (XORShift::frand() <= prob_s) {// Sample surface or volume?
            if (!intersect(r, t, id, tnear + s))
                return radiance(sRay, ++depth) * ms * (1.0/prob_s);
            scaleBy = 1.0;
        }
        else
            if (!intersect(r, t, id)) return Vec();
        if (t >= tnear) {
            double dist = (t > tfar ? tfar - tnear : t - tnear);
            absorption=exp(-sigma_t * dist);
        }
    }
    else
        if (!intersect(r, t, id)) return Vec();
    const Sphere &obj = spheres[id];        // the hit object
    Vec x=r.o+r.d*t, n=(x-obj.p).norm(), nl=n.dot(r.d)<0?n:n*-1, f=obj.c,Le=obj.e;
    double p = f.x>f.y && f.x>f.z ? f.x : f.y>f.z ? f.y : f.z; // max refl
    if (++depth>5) if (XORShift::frand()<p) {f=f*(1/p);} else return Vec(); //R.R.
    if (n.dot(nl)>0 || obj.refl != REFR) {f = f * absorption; Le = obj.e * absorption;}// no absorption inside glass
    else scaleBy=1.0;
    if (obj.refl == DIFF) {                  // Ideal DIFFUSE reflection
        double r1=2*M_PI*XORShift::frand(), r2=XORShift::frand(), r2s=sqrt(r2);
        Vec w=nl, u=((fabs(w.x)>.1?Vec(0,1):Vec(1))%w).norm(), v=w%u;
        Vec d = (u*cos(r1)*r2s + v*sin(r1)*r2s + w*sqrt(1-r2)).norm();
        return (Le + f.mult(radiance(Ray(x,d),depth))) * scaleBy;
    } else if (obj.refl == SPEC)            // Ideal SPECULAR reflection
        return (Le + f.mult(radiance(Ray(x,r.d-n*2*n.dot(r.d)),depth))) * scaleBy;
    Ray reflRay(x, r.d-n*2*n.dot(r.d));     // Ideal dielectric REFRACTION
    bool into = n.dot(nl)>0;                // Ray from outside going in?
    double nc=1, nt=1.5, nnt=into?nc/nt:nt/nc, ddn=r.d.dot(nl), cos2t;
    if ((cos2t=1-nnt*nnt*(1-ddn*ddn))<0)    // Total internal reflection
        return (Le + f.mult(radiance(reflRay,depth)));
    Vec tdir = (r.d*nnt - n*((into?1:-1)*(ddn*nnt+sqrt(cos2t)))).norm();
    double a=nt-nc, b=nt+nc, R0=a*a/(b*b), c = 1-(into?-ddn:tdir.dot(n));
    double Re=R0+(1-R0)*c*c*c*c*c,Tr=1-Re,P=.25+.5*Re,RP=Re/P,TP=Tr/(1-P);
        return (Le + (depth>2 ? (XORShift::frand()<P ?   // Russian roulette
        radiance(reflRay,depth)*RP:f.mult(radiance(Ray(x,tdir),depth)*TP)) :
    radiance(reflRay,depth)*Re+f.mult(radiance(Ray(x,tdir),depth)*Tr)))*scaleBy;
}
int main(int argc, char *argv[]) {
    int w=1024, h=768, samps = argc==2 ? atoi(argv[1])/4 : 1; // # samples
    Ray cam(Vec(50,52,285), Vec(0,-0.042612,-1).norm()); // cam pos, dir
    Vec cx=Vec(w*.5135/h), cy=(cx%cam.d).norm()*.5135, r, *c=new Vec[w*h];
#pragma omp parallel for schedule(dynamic, 1) private(r)       // OpenMP
    for (int y=0; y<h; y++) {                        // Loop over image rows
        fprintf(stderr,"\rRendering (%d spp) %5.2f%%",samps*4,100.*y/(h-1));
        for (unsigned short x=0; x<w; x++)   // Loop cols
            for (int sy=0, i=(h-y-1)*w+x; sy<2; sy++)     // 2x2 subpixel rows
                for (int sx=0; sx<2; sx++, r=Vec()){        // 2x2 subpixel cols
                    for (int s=0; s<samps; s++){
                        double r1=2*XORShift::frand(), dx=r1<1 ? sqrt(r1)-1: 1-sqrt(2-r1);
                        double r2=2*XORShift::frand(), dy=r2<1 ? sqrt(r2)-1: 1-sqrt(2-r2);
                        Vec d = cx*( ( (sx+.5 + dx)/2 + x)/w - .5) +
                                cy*( ( (sy+.5 + dy)/2 + y)/h - .5) + cam.d;
                        r = r + radiance(Ray(cam.o+d*140,d.norm()),0)*(1./samps);
                    } // Camera rays are pushed ^^^^^ forward to start in interior
                    c[i] = c[i] + Vec(clamp(r.x),clamp(r.y),clamp(r.z))*.25;
                }
    }
    FILE *f = fopen("image.ppm", "w"); // Write image to PPM file.
    fprintf(f, "P3\n%d %d\n%d\n", w, h, 255);
    for (int i=0; i<w*h; i++)
        fprintf(f,"%d %d %d ", toInt(c[i].x), toInt(c[i].y), toInt(c[i].z));
}
```



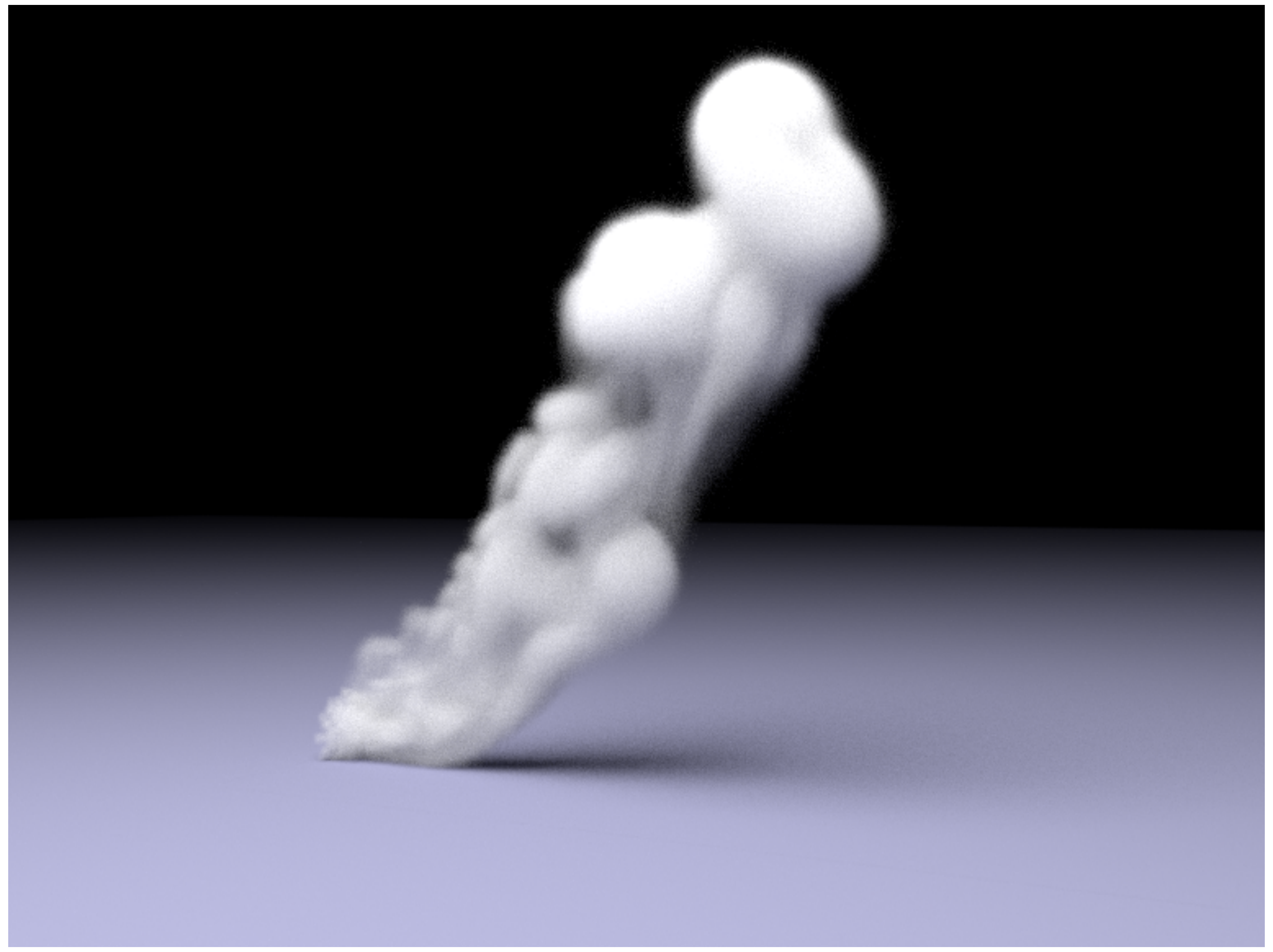
https://github.com/seifeddinedridi/smallvpt

# Types of media

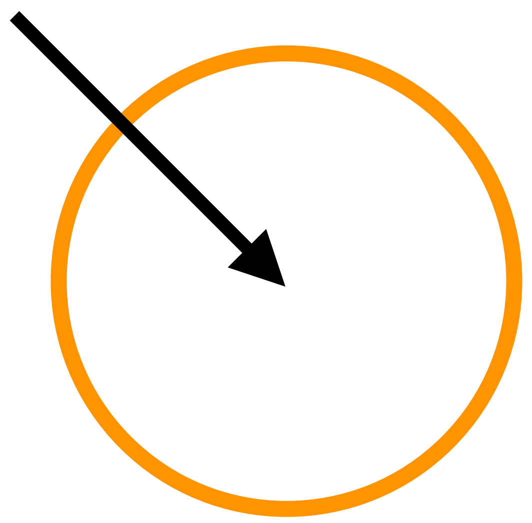
homogeneous medium



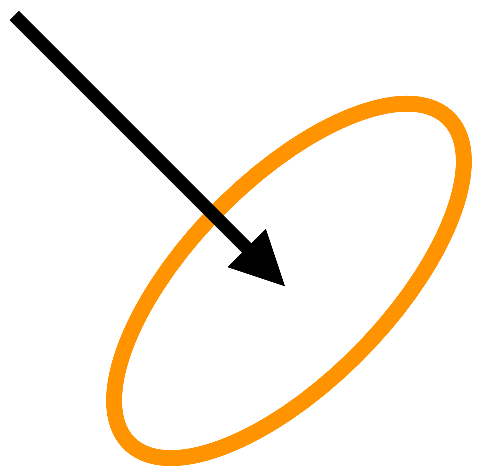
heterogeneous medium



isotropic phase function

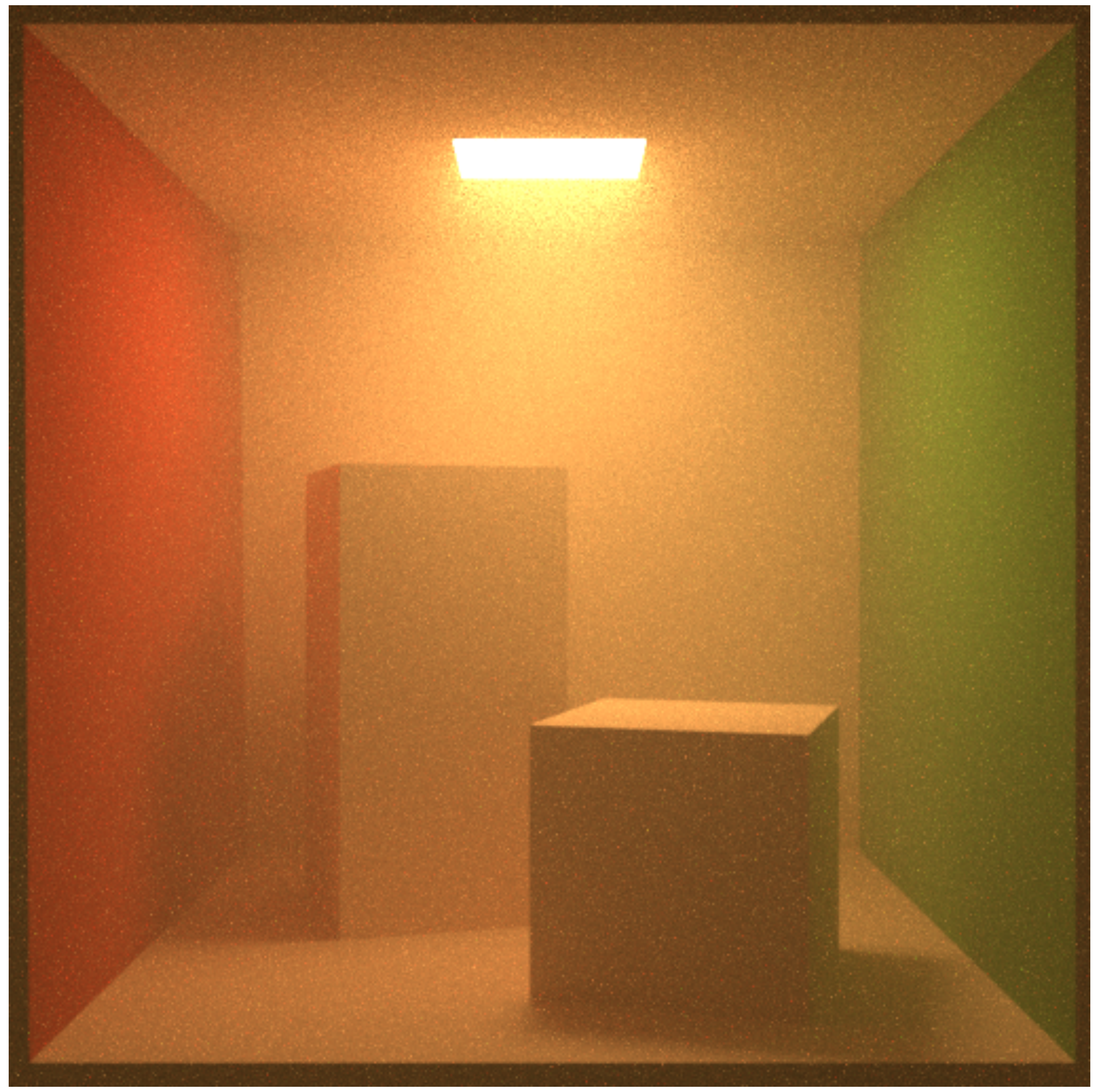
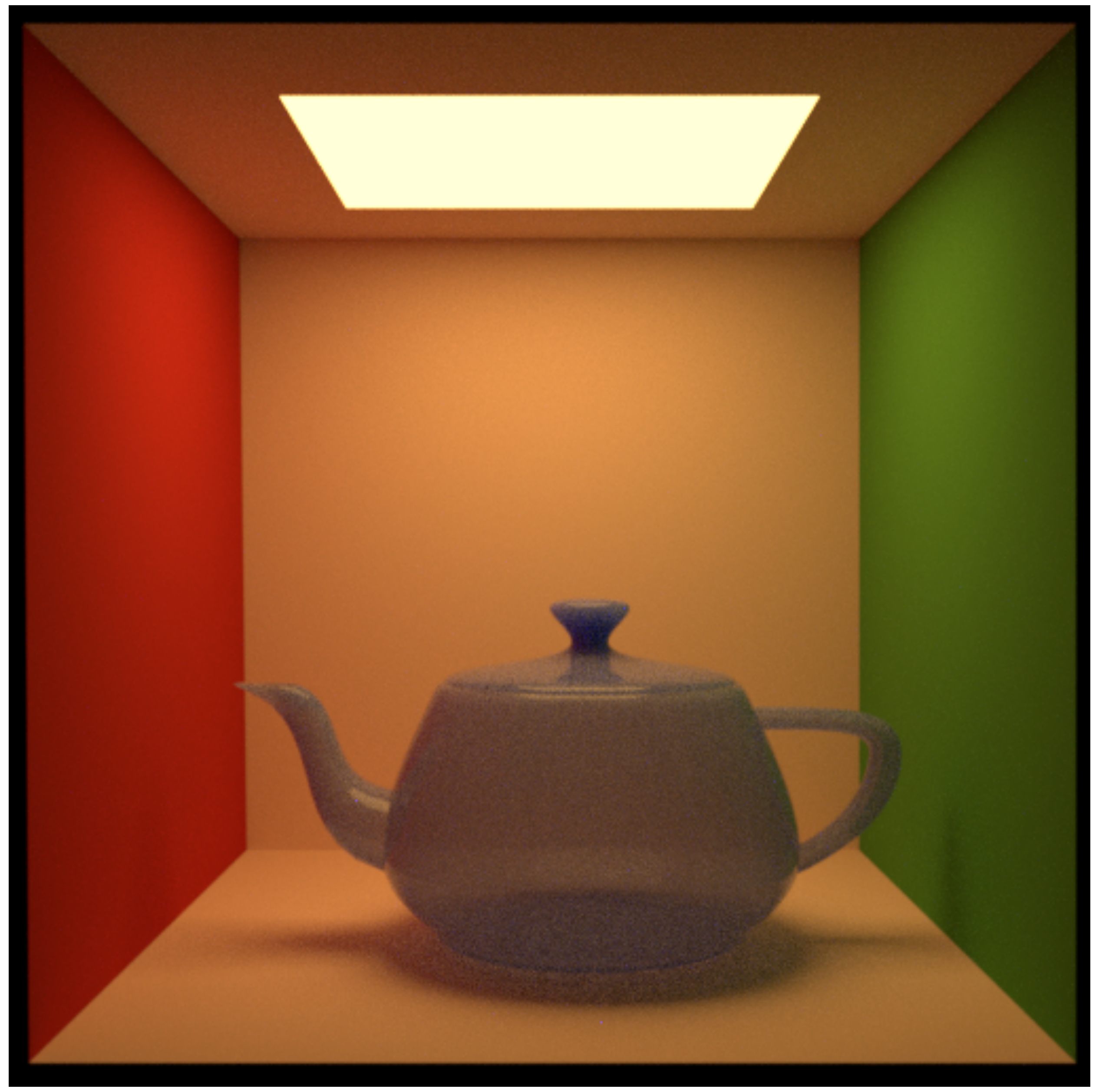


anisotropic phase function



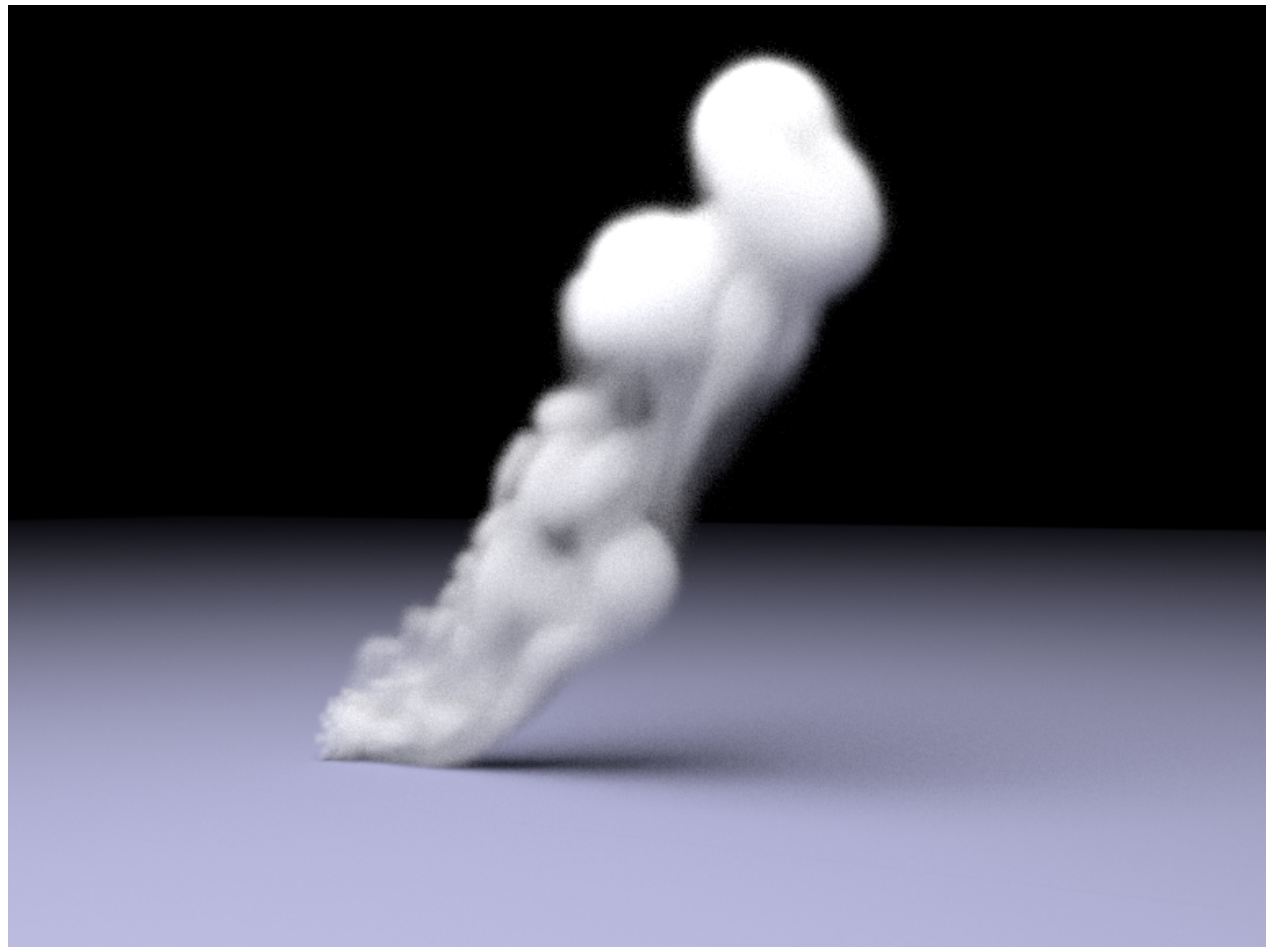
# Homogeneous medium: $\sigma_t$ is constant

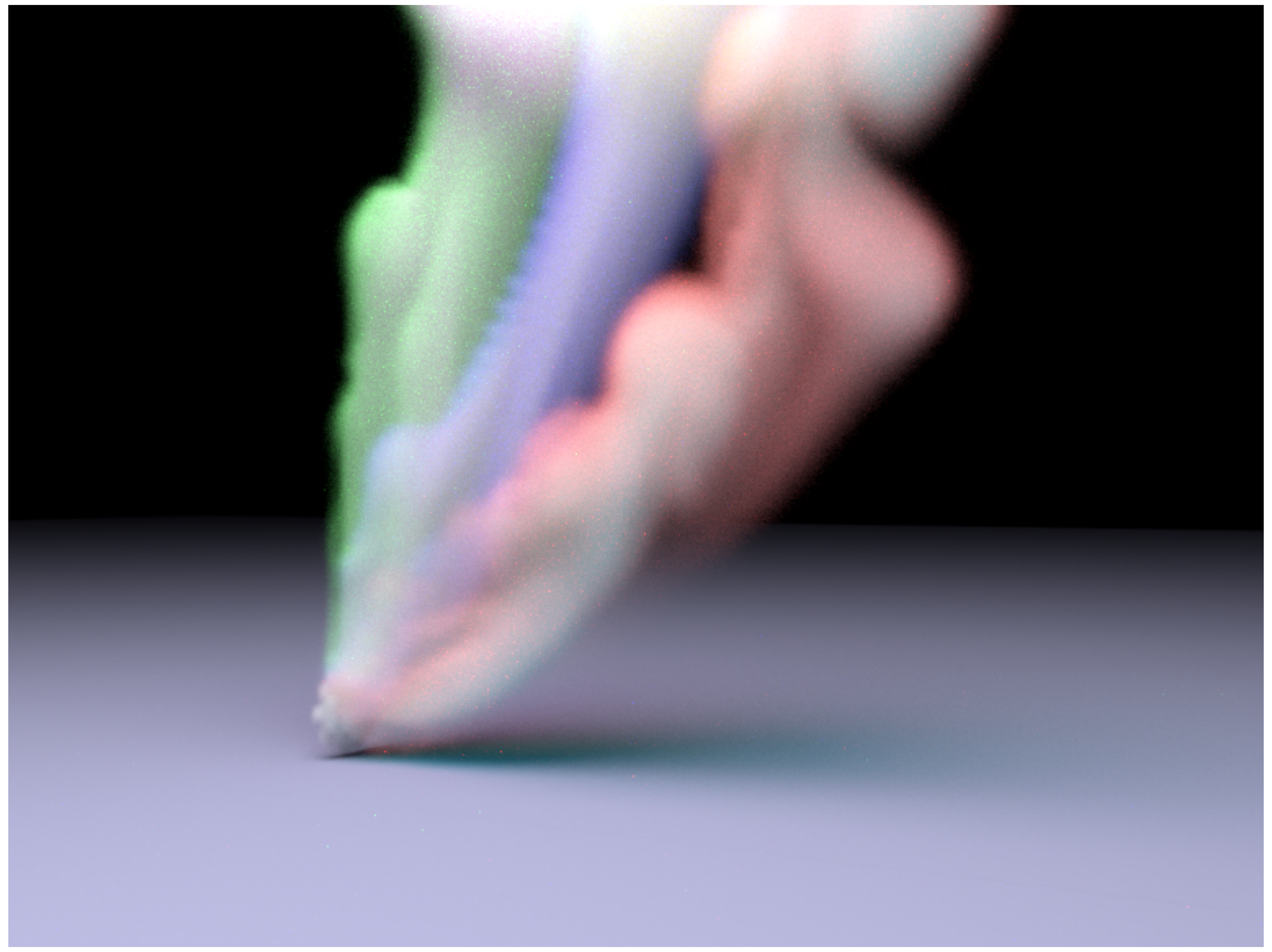
- significantly simplifies transmittance sampling/evaluation

$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(\mathbf{p}(t'))\mathrm{d}t'\right) = \exp(-t\sigma_t)$$

# Heterogeneous medium: $\sigma_t(\mathbf{p})$ varies spatially



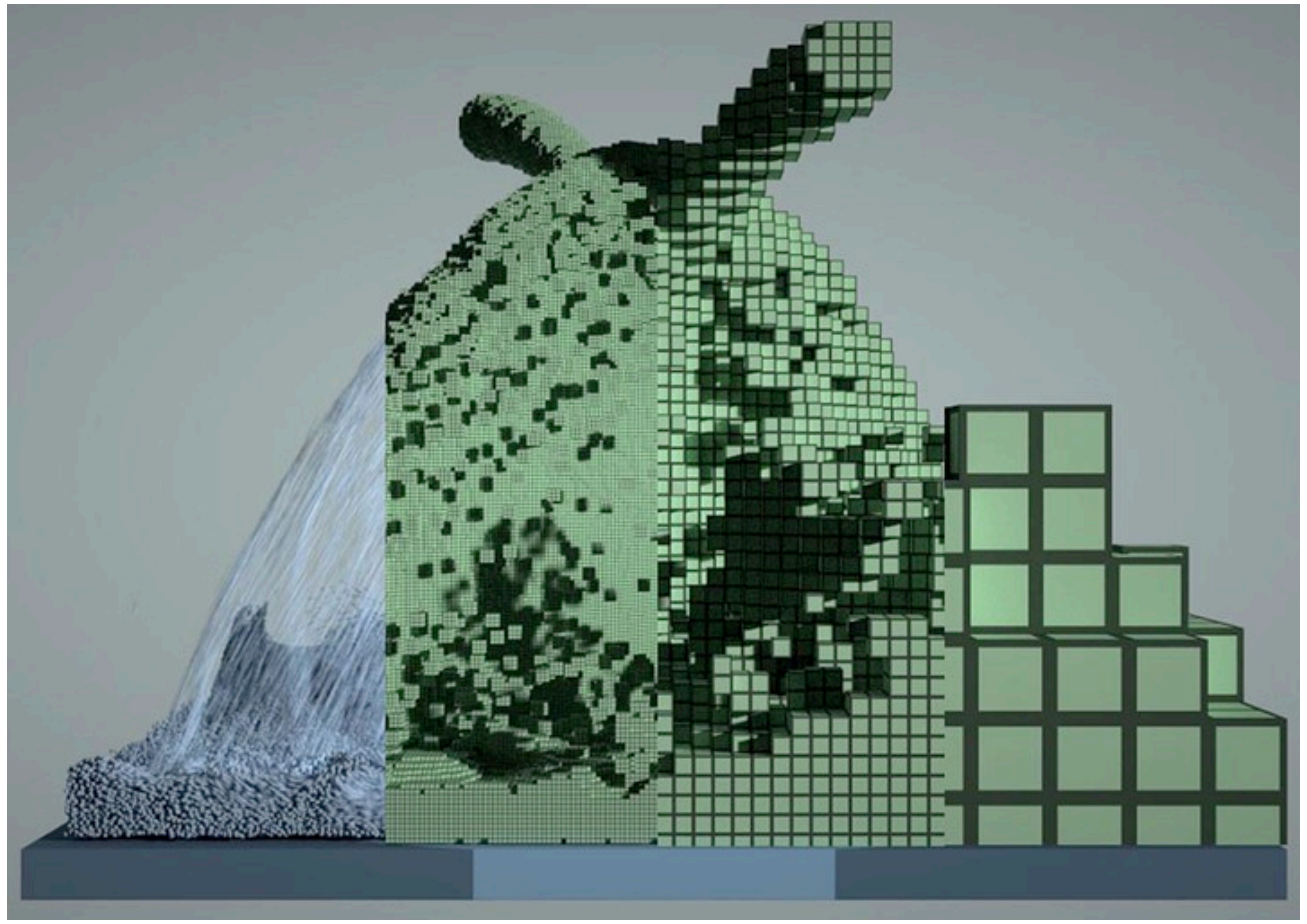


$$T(\mathbf{p}(0), \mathbf{p}(t)) = \exp\left(-\int_0^t \sigma_t(\mathbf{p}(t'))\mathrm{d}t'\right)$$

# Data structures for storing heterogeneous media

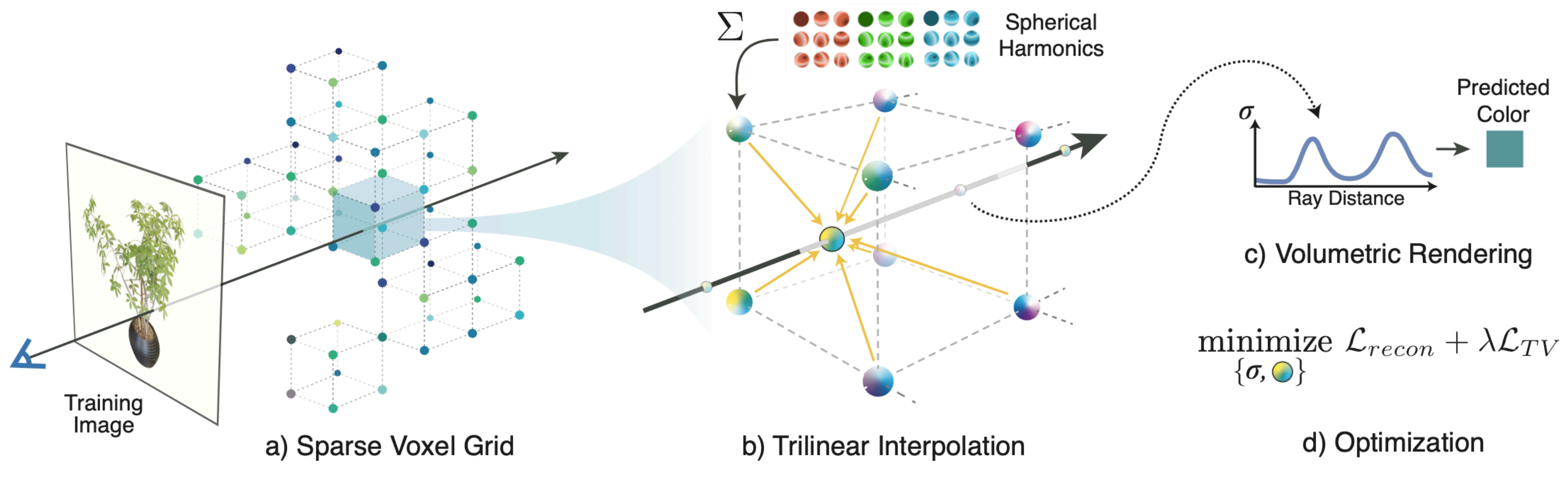
- hierarchical sparse arrays: exploiting spatial coherent sparsity

https://developer.nvidia.com/nanovdb

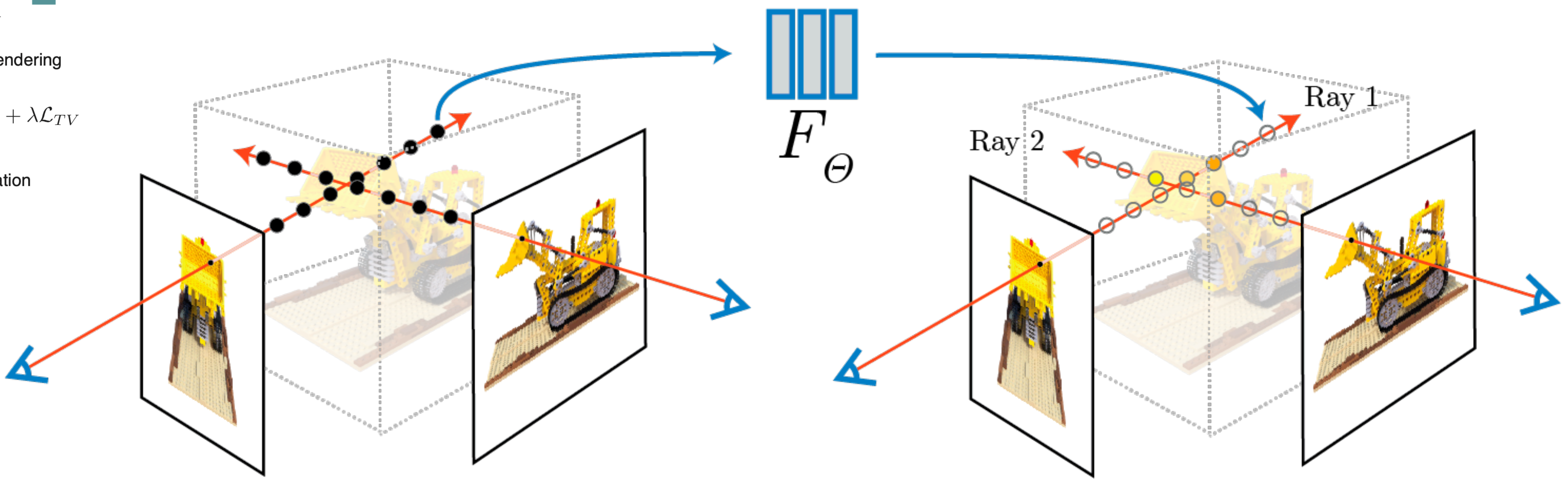


https://yuanming.taichi.graphics/publication/2019-taichi/

# NeRF: spatial-directionally varying emission-absorption only volumes (no scattering)




https://alexyu.net/plenoxels/?s=09




$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = -\sigma_t L(\mathbf{p}(t),\omega)$$

$$+L_e(\mathbf{p}(t),\omega) + \cancel{\sigma_s \int_{S^2} \rho(\omega,\omega')L(\mathbf{p}(t),\omega')\mathrm{d}\omega'}$$

neural networks (sparse grids are also good)

https://www.matthewtancik.com/nerf

# Me in 2019, after submitted the Taichi paper

- I still think this is a cool direction!!

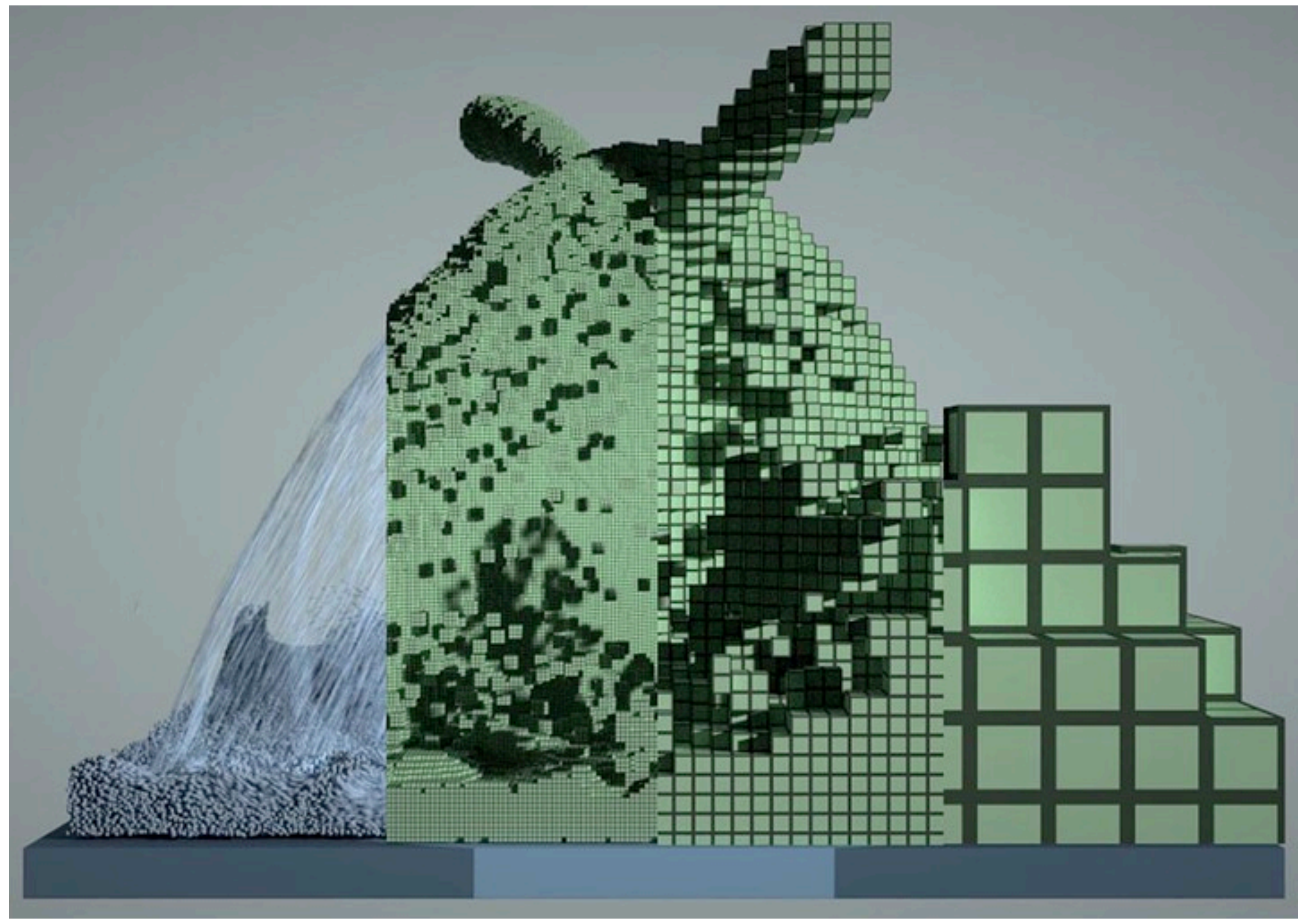
**Tzu-Mao Li** <bachi722@gmail.com>
to Yuanming, Luke, Fredo, Jonathan, Bill

Tue, 21 May 2019, 06:49

I start to feel that there is some opportunities for inverse rendering with volumetric path tracing, even in the surface lighting case.

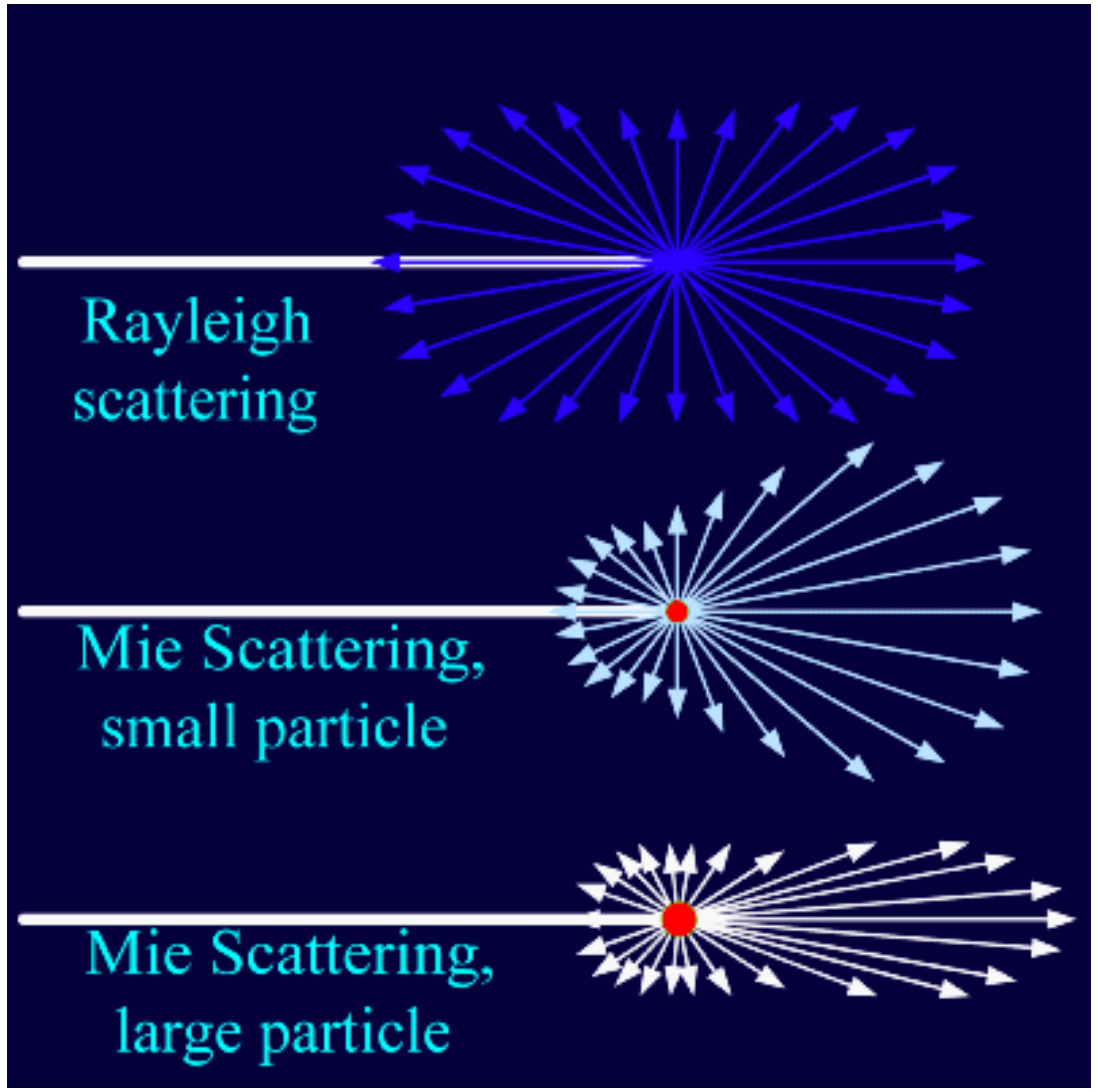
Combining the CNN and path tracing code in this project we already have a pretty efficient forward model for large scale rendering. The rest is to add an anisotropic phase function (E.g. microflake) and level of detail (e.g. sggx). Then we can autodiff the compiler. Efficient scatter to gather conversion is again an issue.

I am also thinking about the deep image prior thing applied to volume. A simple thing to try is to take a few measurement images and reconstruct the volumes using DIP.



# Phase function

- very little work on this!
- in physics:
  - very very small particles: Rayleigh scattering
  - very small particles: Mie scattering
  - large particles: just treat them as surfaces…
  - phenomenological model: Henyey-Greenstein



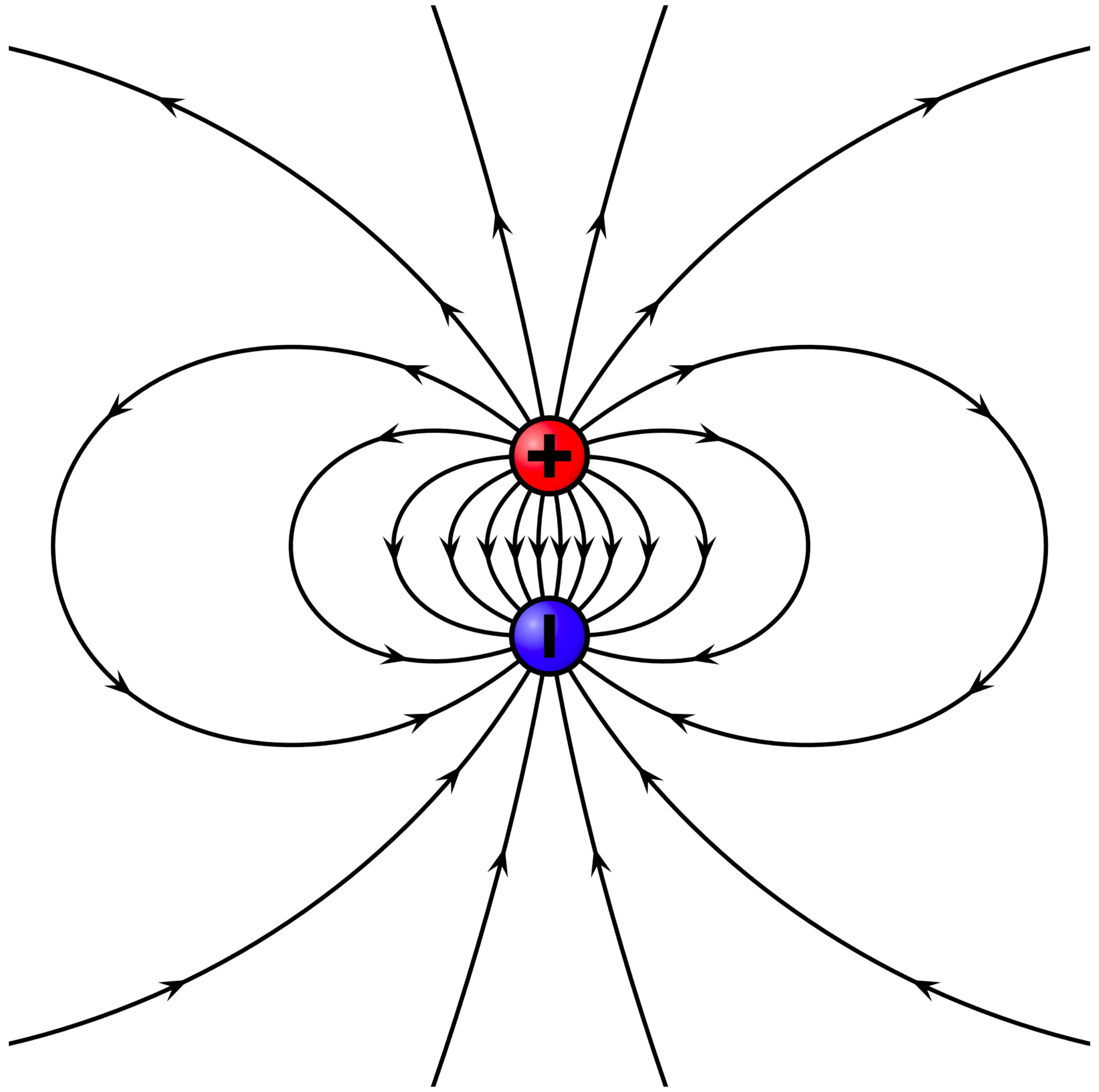

http://homework.uoregon.edu/pub/class/atm/scatter.html

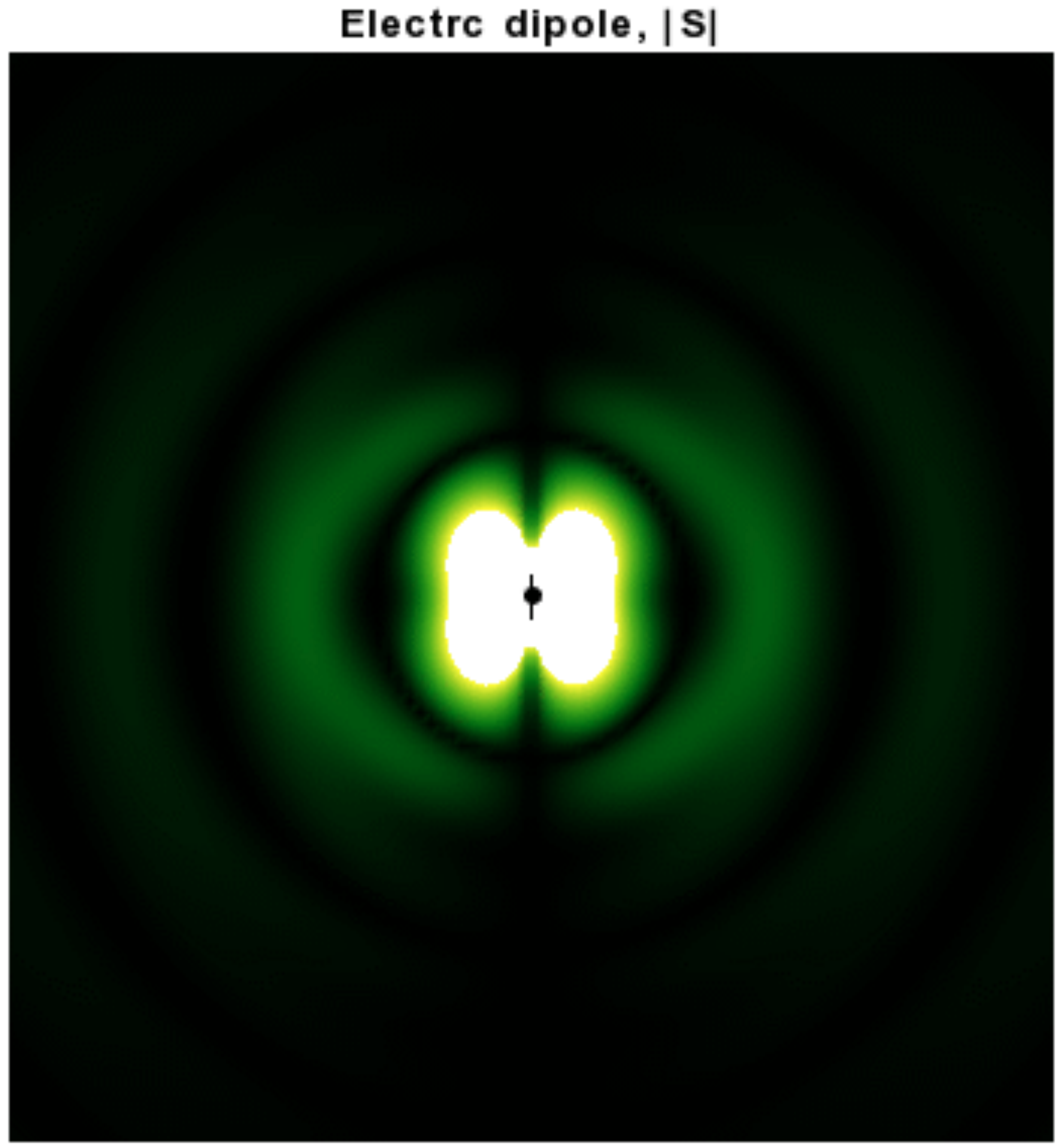
# Rayleigh scattering

- based on "dipole approximation" of electromagnetic fields

$$\rho(\omega, \omega') = \frac{8\pi^4\alpha^2}{\lambda^4}\left(1 + (\omega \cdot \omega')^2\right)$$

$\alpha$: "polarizibility" (https://en.wikipedia.org/wiki/Polarizability)
$\lambda$: wavelength

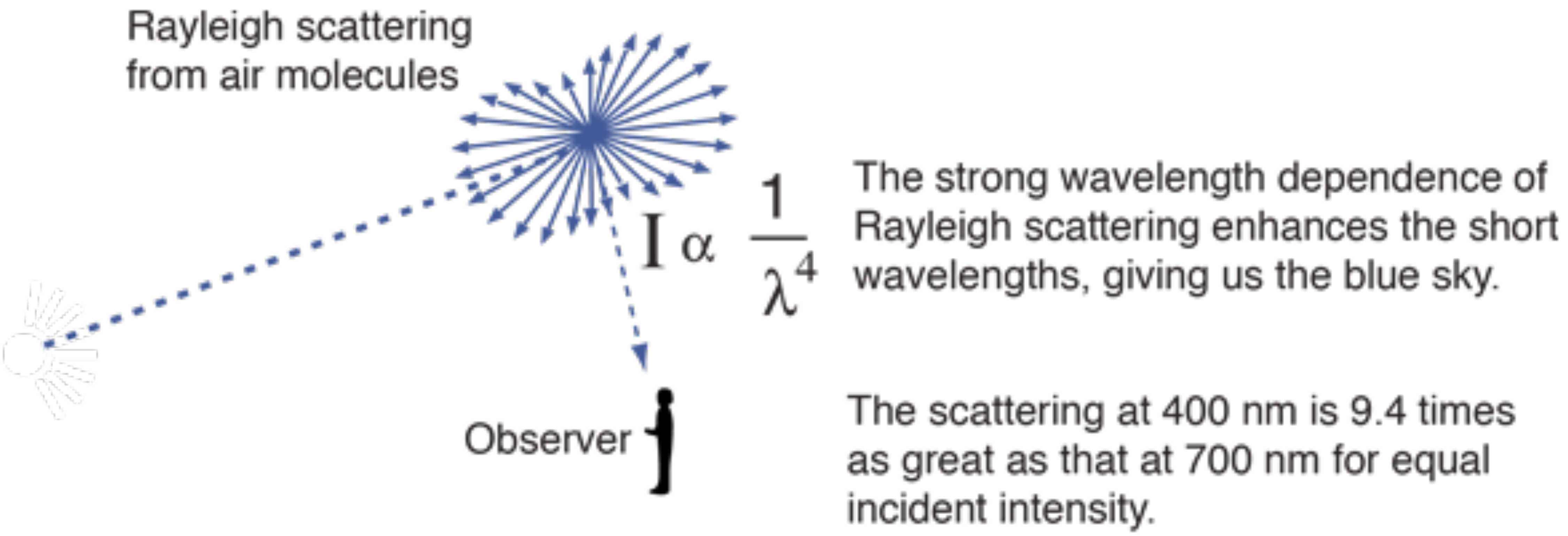





https://en.wikipedia.org/wiki/Dipole#Dipole_radiation

# Rayleigh scattering explains the color of sky

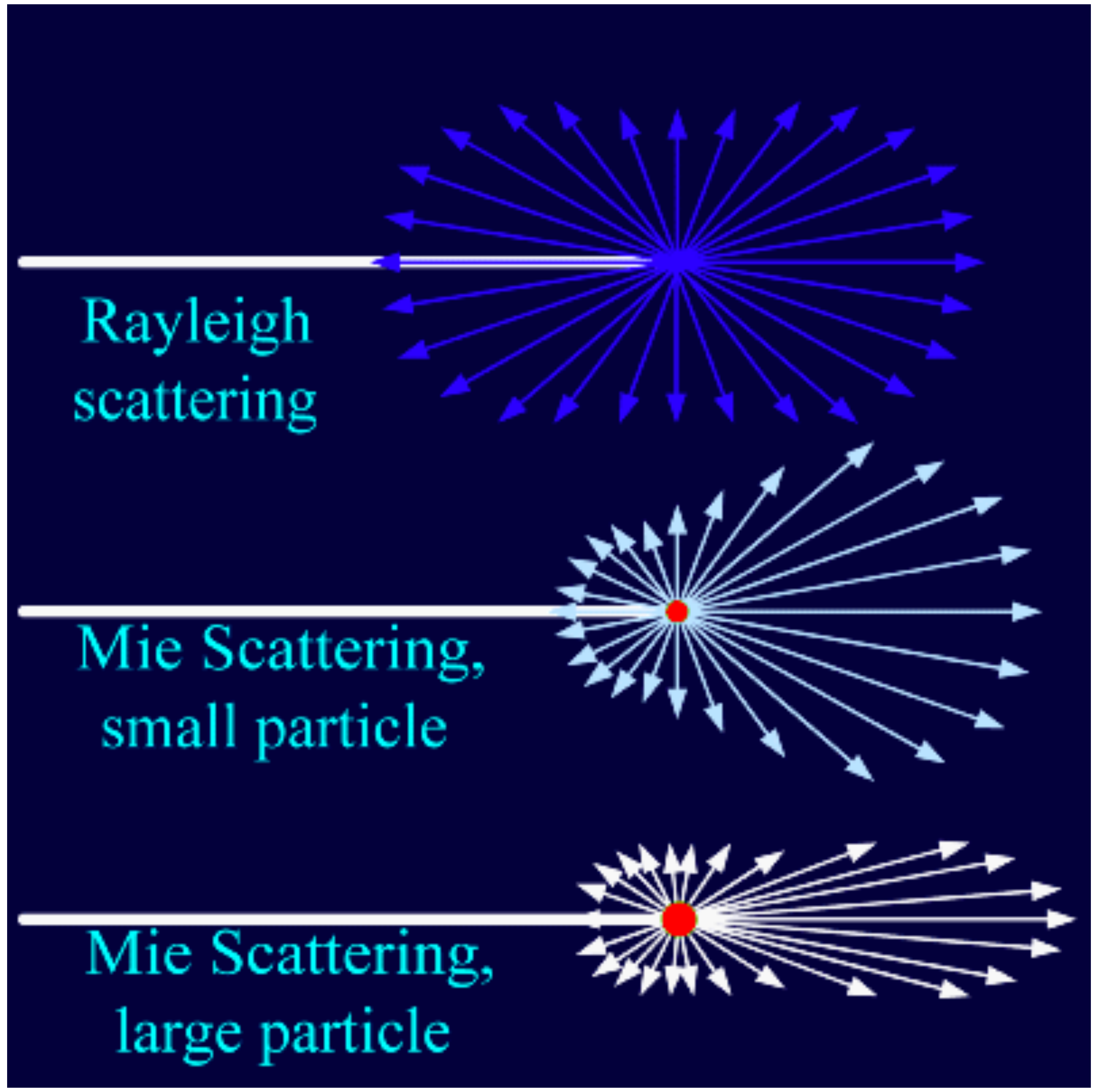
$$\rho(\omega, \omega') = \frac{8\pi^4\alpha^2}{\lambda^4}\left(1 + (\omega \cdot \omega')^2\right)$$




http://hyperphysics.phy-astr.gsu.edu/hbase/atmos/blusky.html

# Phase function

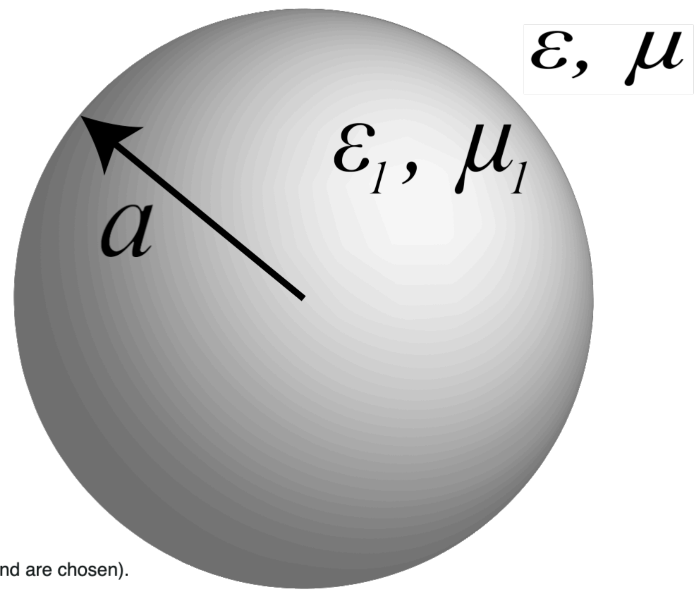
- very little work on this!
- in physics:
  - very very small particles: Rayleigh scattering
  - very small particles: Mie scattering
  - large particles: just treat them as surfaces…
  - phenomenological model: Henyey-Greenstein




http://homework.uoregon.edu/pub/class/atm/scatter.html

# Mie scattering

- derive the electric field by solving Maxwell's equation directly on a spherical particle
  - less wavelength dependent




### Mathematics [ edit ]

The scattering by a spherical nanoparticle is solved exactly regardless of the particle size. We consider scattering by a plane wave propagating along the *z*-axis polarized along the *x*-axis. Dielectric and magnetic permeabilities of a particle are $\varepsilon_1$ and $\mu_1$, and $\varepsilon$ and $\mu$ for the environment.

In order to solve the scattering problem,[3] we write first the solutions of the vector Helmholtz equation in spherical coordinates, since the fields inside and outside the particles must satisfy it. Helmholtz equation:

$$\nabla^2\mathbf{E} + k^2\mathbf{E} = 0, \quad \nabla^2\mathbf{H} + k^2\mathbf{H} = 0.$$

In addition to the Helmholtz equation, the fields must satisfy the conditions $\nabla\cdot\mathbf{E} = \nabla\cdot\mathbf{H} = 0$ and $\nabla\times\mathbf{E} = i\omega\mu\mathbf{H}$, $\nabla\times\mathbf{H} = -i\omega\varepsilon\mathbf{E}$. Vector spherical harmonics possess all the necessary properties, introduced as follows:

$$\mathbf{M}_{\substack{e\\o}mn} = \nabla\times(\mathbf{r}\psi_{\substack{e\\o}mn}) \text{ — magnetic harmonics (TE),}$$
$$\mathbf{N}_{\substack{e\\o}mn} = \frac{\nabla\times\mathbf{M}_{\substack{e\\o}mn}}{k} \text{ — electric harmonics (TM),}$$

where

$$\psi_{emn} = \cos m\varphi P_n^m(\cos\vartheta) z_n(kr),$$
$$\psi_{omn} = \sin m\varphi P_n^m(\cos\vartheta) z_n(kr),$$

and $P_n^m(\cos\theta)$ — Associated Legendre polynomials, and $z_n(kr)$ — any of the spherical Bessel functions.

Next, we expand the incident plane wave in vector spherical harmonics:

$$\mathbf{E}_{inc} = E_0 e^{ikr\cos\theta}\mathbf{e}_x = E_0\sum_{n=1}^{\infty} i^n \frac{2n+1}{n(n+1)}\left(\mathbf{M}^{(1)}_{o1n}(k,\mathbf{r}) - i\mathbf{N}^{(1)}_{e1n}(k,\mathbf{r})\right),$$
$$\mathbf{H}_{inc} = \frac{-k}{\omega\mu}E_0\sum_{n=1}^{\infty} i^n \frac{2n+1}{n(n+1)}\left(\mathbf{M}^{(1)}_{e1n}(k,\mathbf{r}) + i\mathbf{N}^{(1)}_{o1n}(k,\mathbf{r})\right).$$

Here the superscript $(1)$ means that in the radial part of the functions $\psi_{\substack{e\\o}mn}$ are spherical Bessel functions of the first kind. The expansion coefficients are obtained by taking integrals of the form

$$\frac{\int_0^{2\pi}\int_0^{\pi}\mathbf{E}_{inc}\cdot\mathbf{M}^{(1)}_{\substack{e\\o}mn}\sin\theta d\theta d\varphi}{\int_0^{2\pi}\int_0^{\pi}|\mathbf{M}^{(1)}_{\substack{e\\o}mn}|^2\sin\theta d\theta d\varphi}.$$

In this case, all coefficients at $m \neq 1$ are zero, since the integral over the angle $\varphi$ in the numerator is zero.

Then the following conditions are imposed:

1) Interface conditions on the boundary between the sphere and the environment (which allow us to relate the expansion coefficients of the incident, internal, and scattered fields)

2) The condition that the solution is bounded at the origin (therefore, in the radial part of the generating functions $\psi_{\substack{e\\o}mn}$, spherical Bessel functions of the first kind are selected for the internal field),

3) For a scattered field, the asymptotics at infinity corresponds to a diverging spherical wave (in connection with this, for the scattered field in the radial part of the generating functions $\psi_{\substack{e\\o}mn}$ spherical Hankel functions of the first kind are chosen).

Scattered fields are written in terms of a vector harmonic expansion as

$$\mathbf{E}_s = \sum_{n=1}^{\infty} E_n\left(ia_n\mathbf{N}^{(3)}_{e1n}(k,\mathbf{r}) - b_n\mathbf{M}^{(3)}_{o1n}(k,\mathbf{r})\right),$$
$$\mathbf{H}_s = \frac{k}{\omega\mu}\sum_{n=1}^{\infty} E_n\left(a_n\mathbf{M}^{(3)}_{e1n}(k,\mathbf{r}) + ib_n\mathbf{N}^{(3)}_{o1n}(k,\mathbf{r})\right).$$

Here the superscript $(3)$ means that in the radial part of the functions $\psi_{\substack{e\\o}mn}$ are spherical Hankel functions of the first kind (those of the second kind would have $(4)$), and $E_n = \frac{i^n E_0(2n+1)}{n(n+1)}$,

Internal fields:

$$\mathbf{E}_1 = \sum_{n=1}^{\infty} E_n\left(-id_n\mathbf{N}^{(1)}_{e1n}(k_1,\mathbf{r}) + c_n\mathbf{M}^{(1)}_{o1n}(k_1,\mathbf{r})\right),$$
$$\mathbf{H}_1 = \frac{-k_1}{\omega\mu_1}\sum_{n=1}^{\infty} E_n\left(d_n\mathbf{M}^{(1)}_{e1n}(k_1,\mathbf{r}) + ic_n\mathbf{N}^{(1)}_{o1n}(k_1,\mathbf{r})\right).$$

$k = \frac{\omega}{c}n$ is the wave vector outside the particle $k_1 = \frac{\omega}{c}n_1$ is the wave vector in the medium from the particle material, $n$ and $n_1$ are the refractive indices of the medium and the particle.

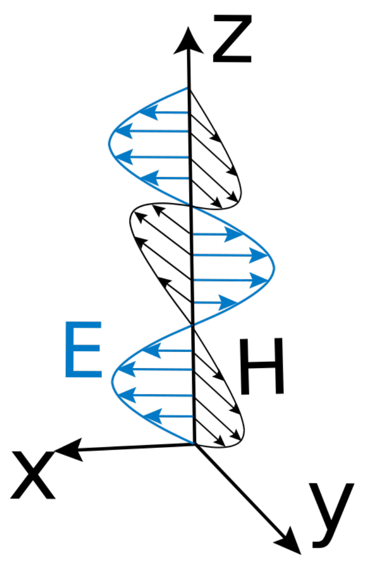
After applying the interface conditions, we obtain expressions for the coefficients:

$$c_n(\omega) = \frac{\mu_1[\rho h_n(\rho)]' j_n(\rho) - \mu_1[\rho j_n(\rho)]' h_n(\rho)}{\mu_1[\rho h_n(\rho)]' j_n(\rho_1) - \mu[\rho_1 j_n(\rho_1)]' h_n(\rho)},$$
$$d_n(\omega) = \frac{\mu_1 n_1 n[\rho h_n(\rho)]' j_n(\rho) - \mu_1 n_1 n[\rho j_n(\rho)]' h_n(\rho)}{\mu n_1^2[\rho h_n(\rho)]' j_n(\rho_1) - \mu_1 n^2[\rho_1 j_n(\rho_1)]' h_n(\rho)},$$
$$b_n(\omega) = \frac{\mu_1[\rho j_n(\rho)]' j_n(\rho_1) - \mu[\rho_1 j_n(\rho_1)]' j_n(\rho)}{\mu_1[\rho h_n(\rho)]' j_n(\rho_1) - \mu[\rho_1 j_n(\rho_1)]' h_n(\rho)},$$
$$a_n(\omega) = \frac{\mu n_1^2[\rho j_n(\rho)]' j_n(\rho_1) - \mu_1 n^2[\rho_1 j_n(\rho_1)]' j_n(\rho)}{\mu n_1^2[\rho h_n(\rho)]' j_n(\rho_1) - \mu_1 n^2[\rho_1 j_n(\rho_1)]' h_n(\rho)},$$

where

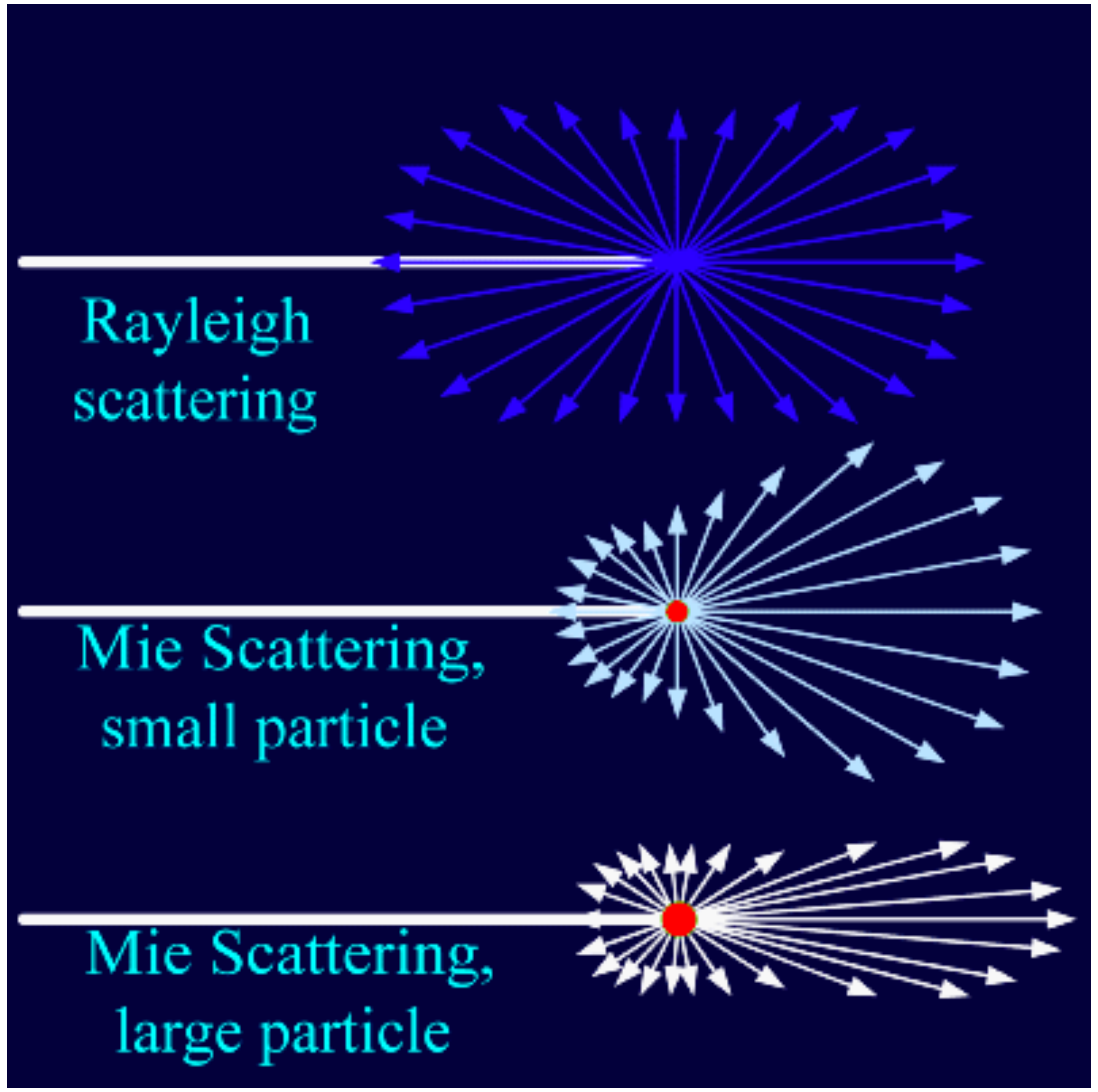
$$\rho = ka,$$
$$\rho_1 = k_1 a \text{ with } a \text{ being the radius of the sphere.}$$

$j_n$ and $h_n$ represent the spherical functions of Bessel and Hankel of the first kind, respectively.




https://en.wikipedia.org/wiki/Mie_scattering

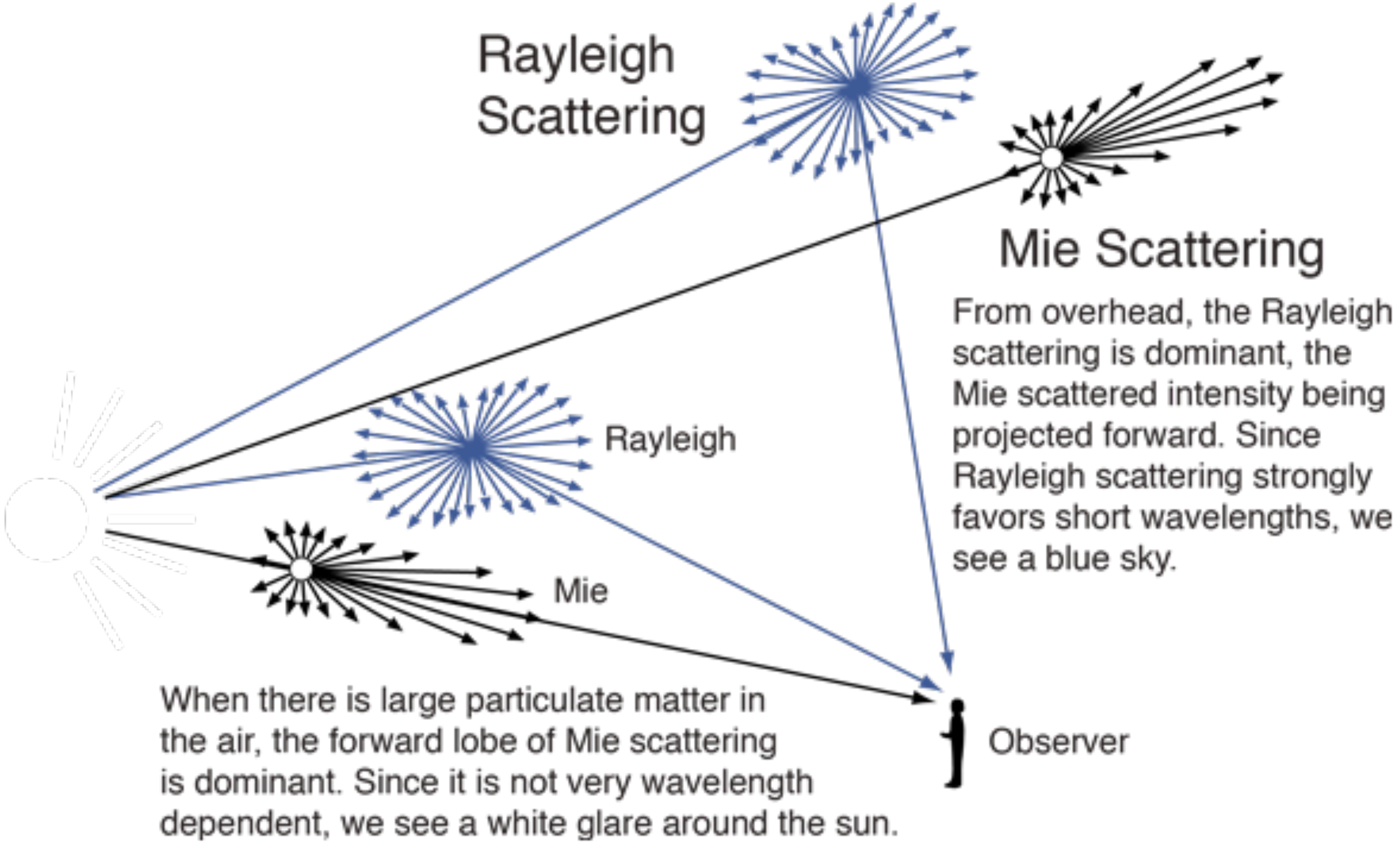
# Mie-scattering exhibits “forward scattering”


http://homework.uoregon.edu/pub/class/atm/scatter.html

https://cs.dartmouth.edu/~wjarosz/publications/novak18monte-sig-slides-2-fundamentals-notes.pdf

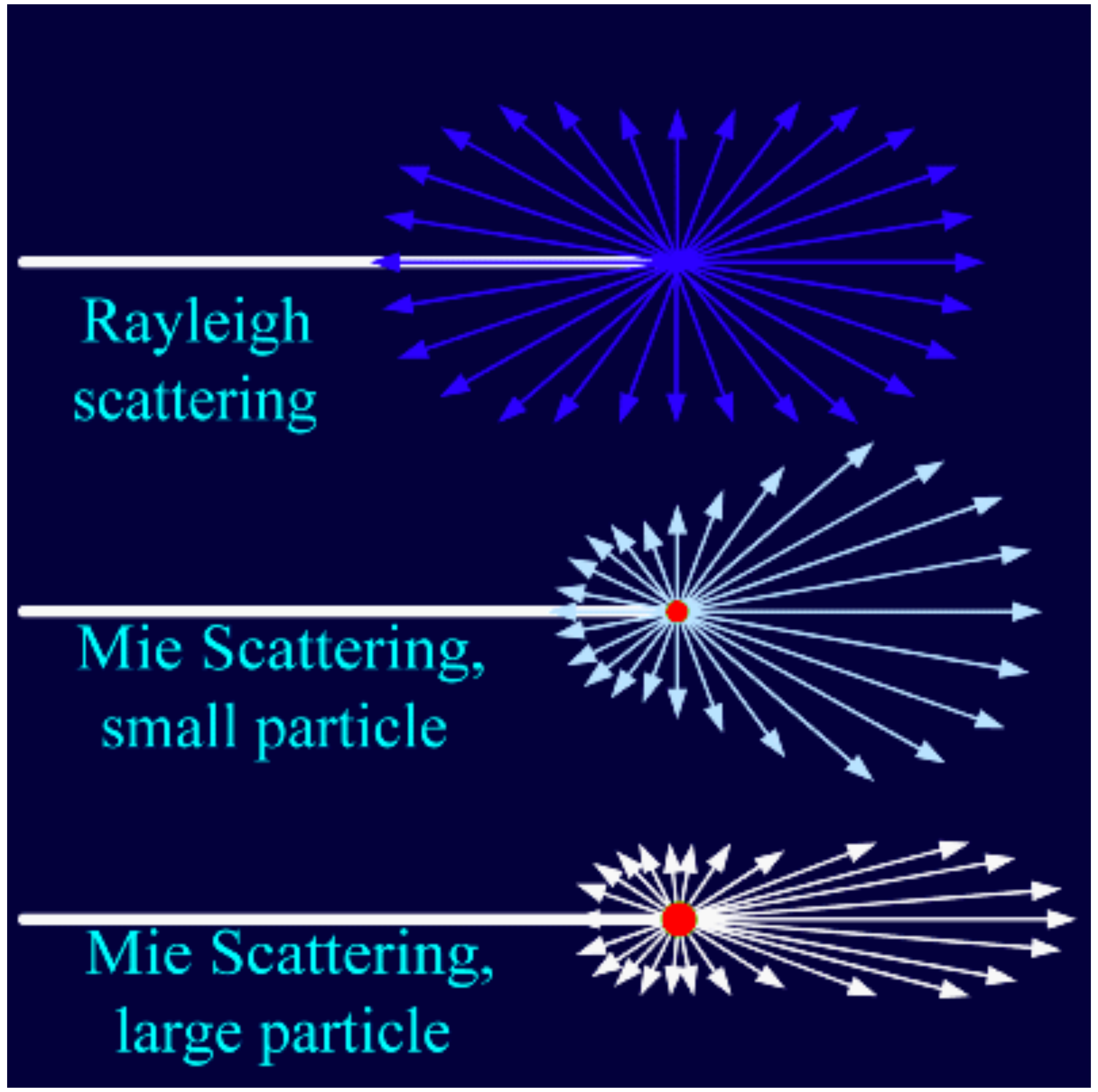
# Rayleigh and Mie scattering




http://hyperphysics.phy-astr.gsu.edu/hbase/atmos/blusky.html

# Phase function

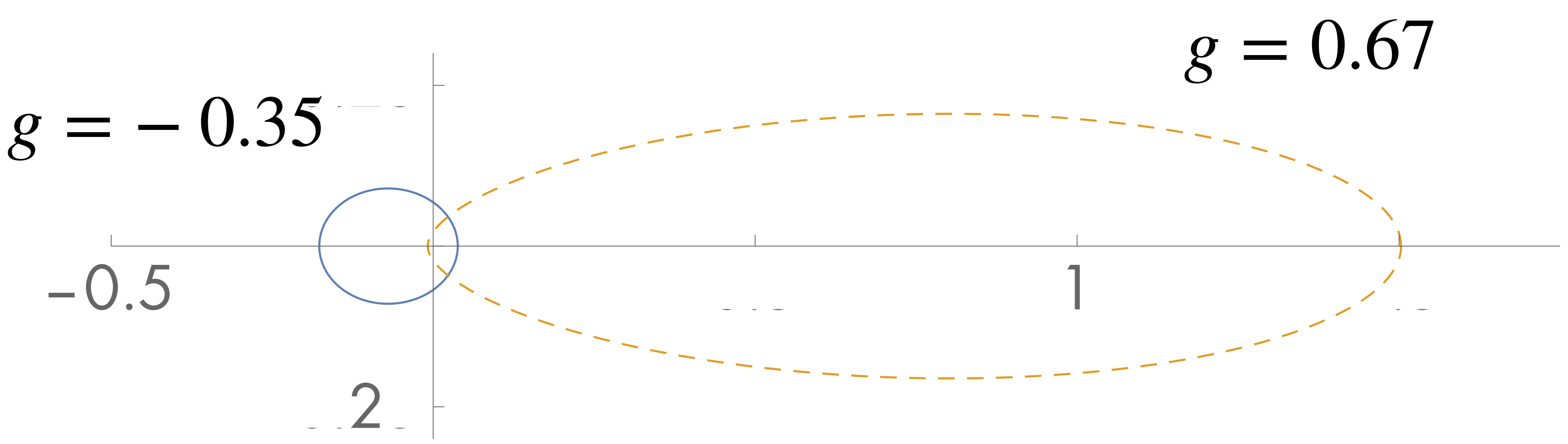
- very little work on this!
- in physics:
  - very very small particles: Rayleigh scattering
  - very small particles: Mie scattering
  - large particles: just treat them as surfaces…
  - phenomenological model: Henyey-Greenstein

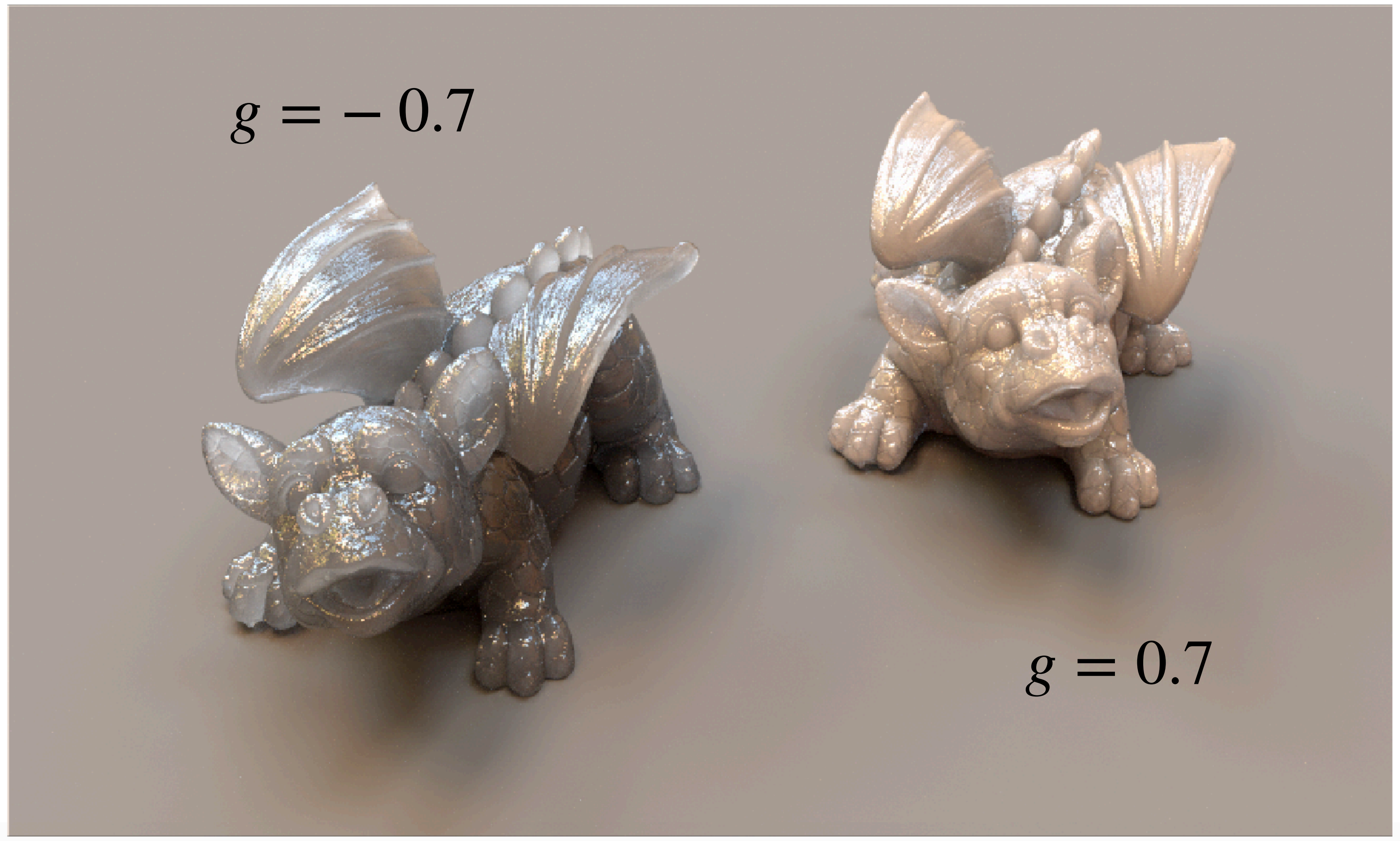



http://homework.uoregon.edu/pub/class/atm/scatter.html

# Henyey-Greeinstein phase function [1944]

$$\rho(\omega, \omega') = \frac{1}{4\pi} \frac{1 - g^2}{\left(1 + g^2 + 2g\omega \cdot \omega'\right)^{\frac{3}{2}}}$$




https://www.pbr-book.org/3ed-2018/Volume_Scattering/Phase_Functions

# Microflake: microfacets for phase functions

- more about it in the future

## A Radiative Transfer Framework for Rendering Materials with Anisotropic Structure

Wenzel Jakob   Jonathan T. Moon   Adam Arbree   Kavita Bala   Steve Marschner

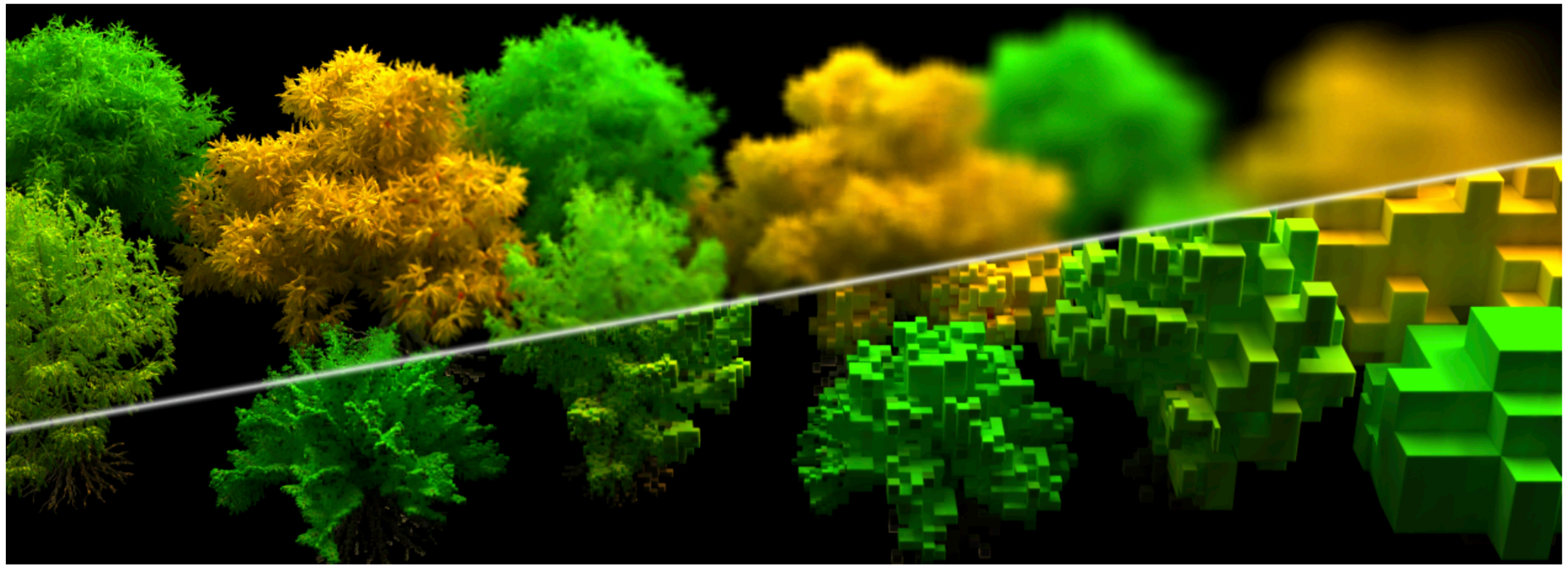
In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*

(a) Isotropic scattering   (b) Scattering by anisotropic micro-flakes   (c) Detail (isotropic)   (d) Detail (micro-flakes)

## The SGGX Microflake Distribution

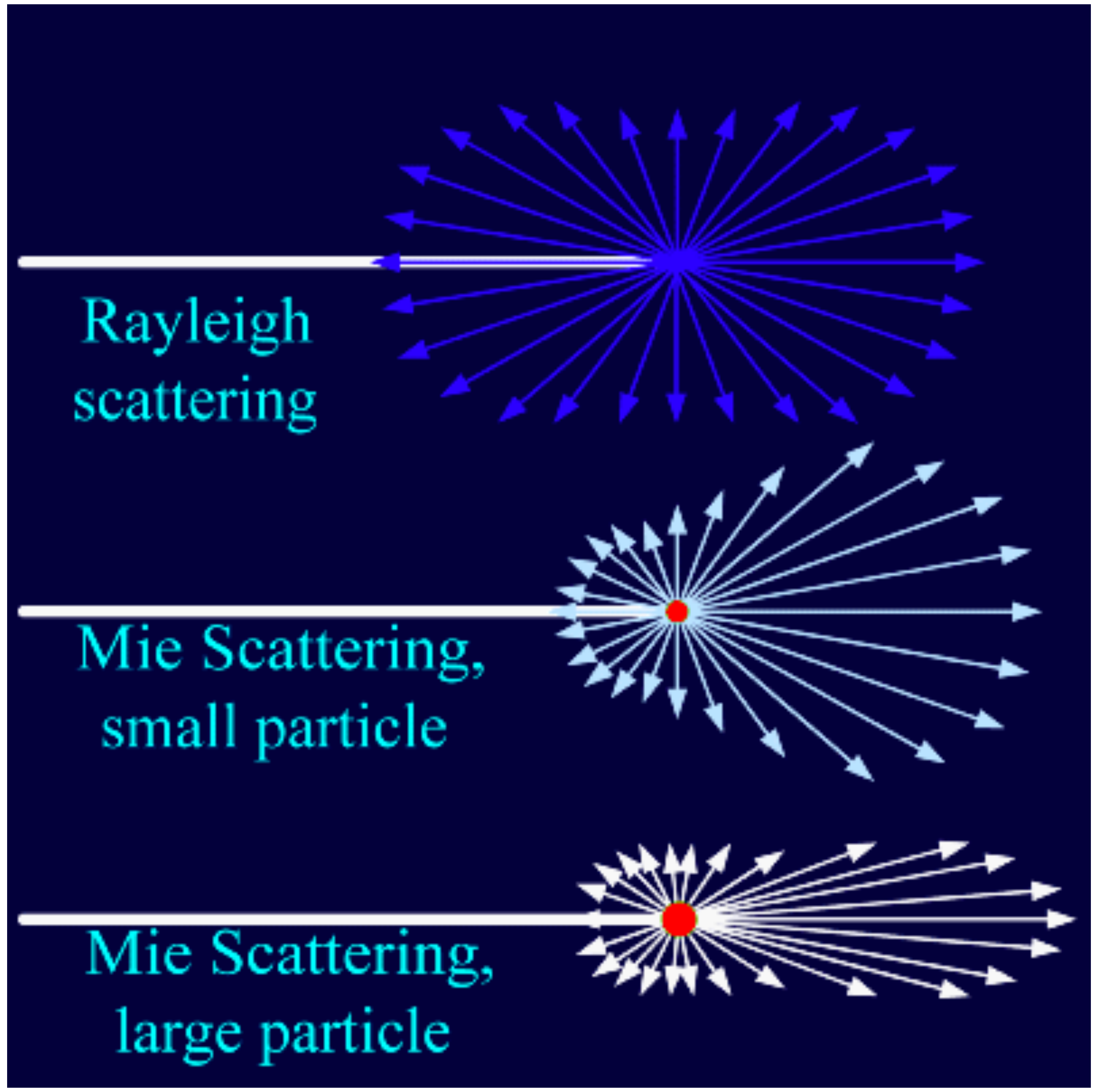
Eric Heitz[1,2]   Jonathan Dupuy[3]   Cyril Crassin[2]   Carsten Dachsbacher[1]

[1]Karlsruhe Institute of Technology   [2]NVIDIA   [3]Univ. Montréal; LIRIS, Univ. Lyon 1



# Phase function

- very little work on this!
- in physics:
  - very very small particles: Rayleigh scattering
  - very small particles: Mie scattering
  - large particles: just treat them as surfaces…
  - phenomenological model: Henyey-Greenstein




http://homework.uoregon.edu/pub/class/atm/scatter.html

# Shell tracing: precomputed phase function

- also more about this in the future, probably

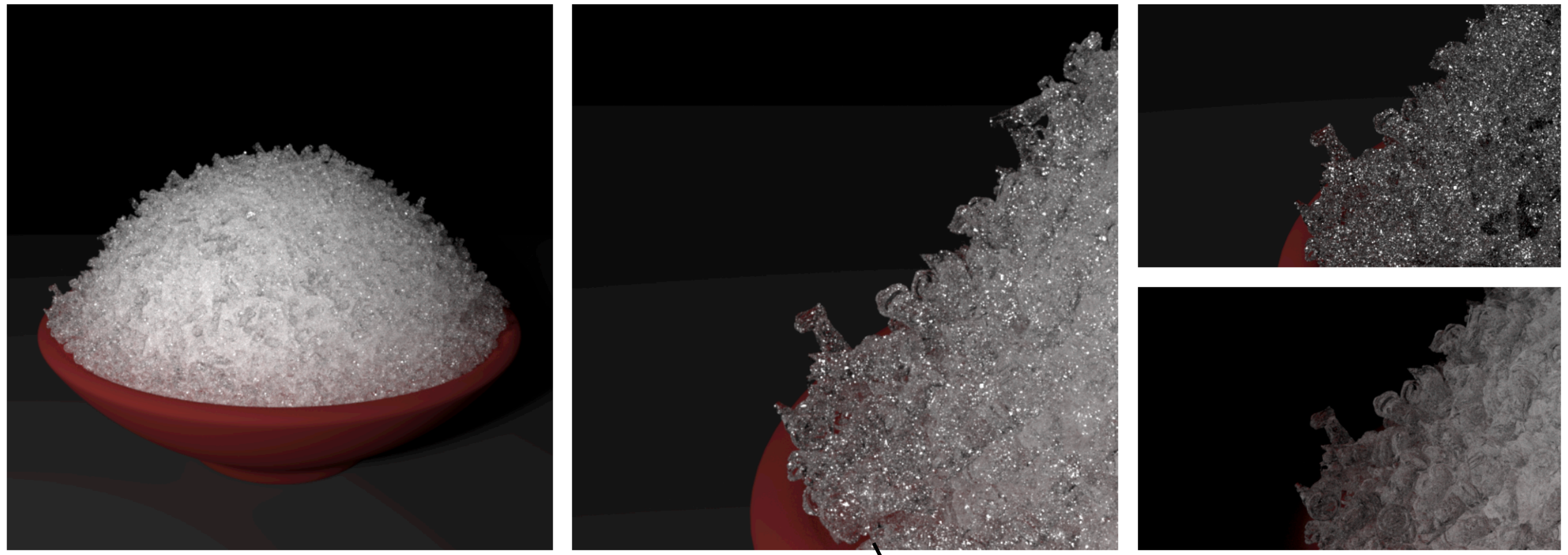
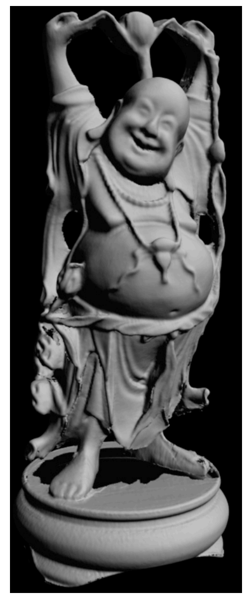
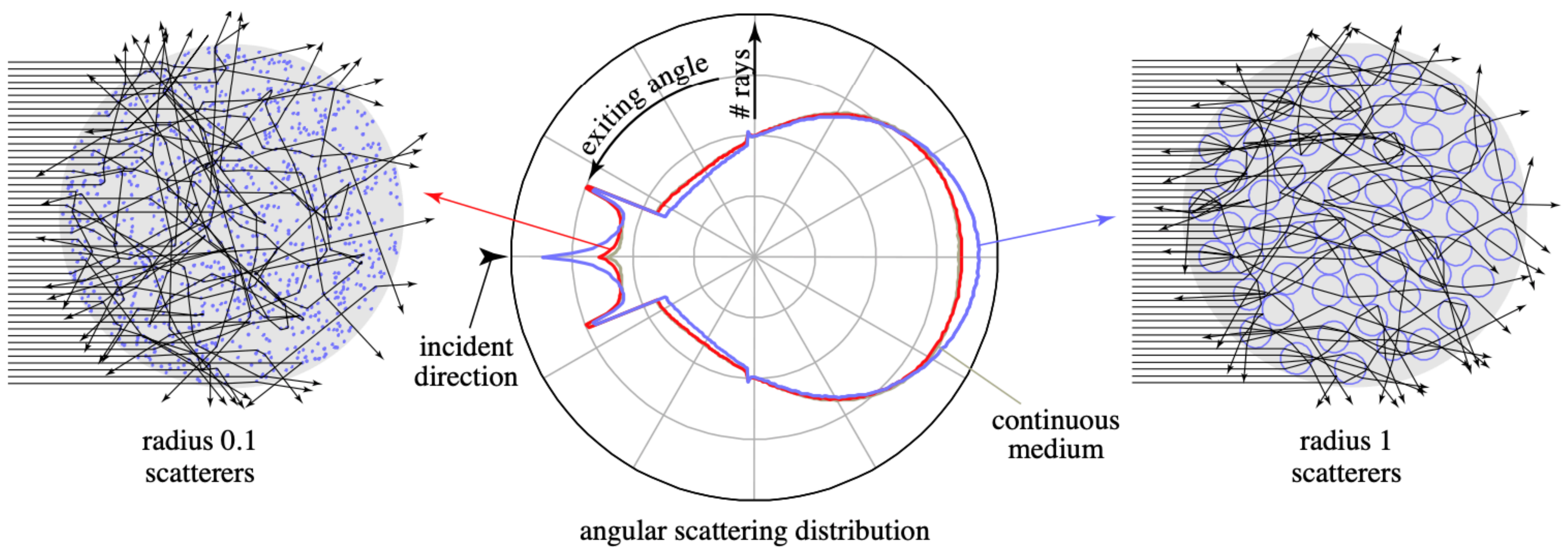


**Figure 13:** *Renderings of 10,000 glass Buddha meshes, each with 10,000 triangles, using our new method. Left: the full bowl of Buddhas, 133 minutes on the cluster. Center: inset of the left edge of the bowl of Buddhas, 175 minutes on the cluster. Top right: path traced low order scattering for the inset, in 135 minutes. Bottom right: high order scattering using shells, in 40 minutes. Precomputation time was 17 minutes on a single machine.*






**Rendering Discrete Random Media Using Precomputed Scattering Solutions**


Jonathan T. Moon, Bruce Walter, and Stephen R. Marschner

Department of Computer Science and Program of Computer Graphics, Cornell University

# Other fun research:
# non-exponential radiative transfer

traditional RTE assumes linear ODEs, can we use arbitrary ODEs?

$$\frac{\mathrm{d}}{\mathrm{d}t}L(\mathbf{p}(t),\omega) = -\sigma_t L(\mathbf{p}(t),\omega)$$

$$+L_e(\mathbf{p}(t),\omega) + \sigma_s \int_{S^2} \rho(\omega,\omega')L(\mathbf{p}(t),\omega')\mathrm{d}\omega'$$

# Other fun research: non-exponential radiative transfer

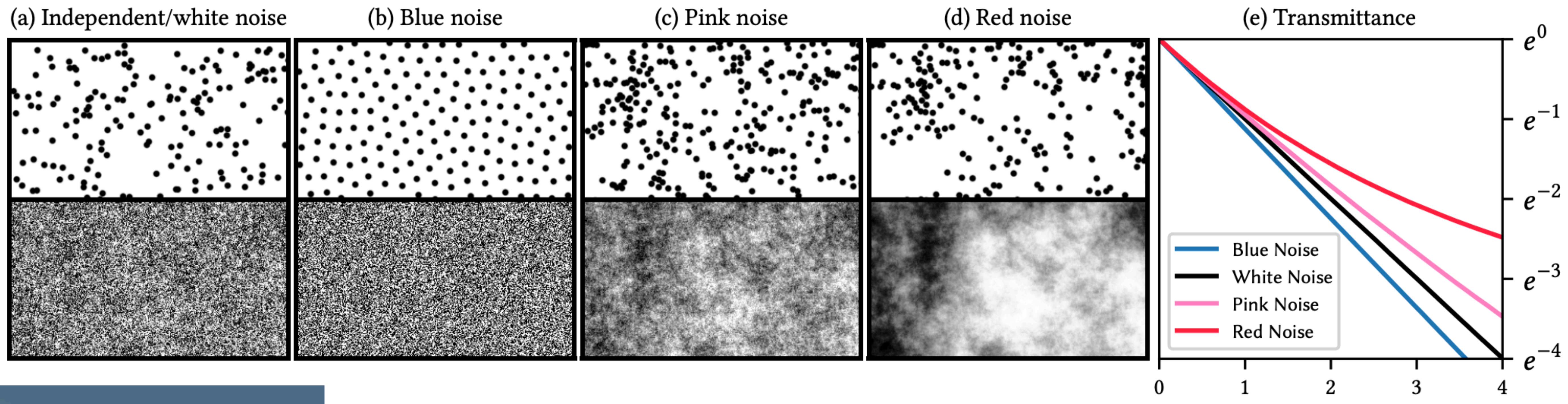
traditional RTE assumes linear ODEs, can we use arbitrary ODEs?

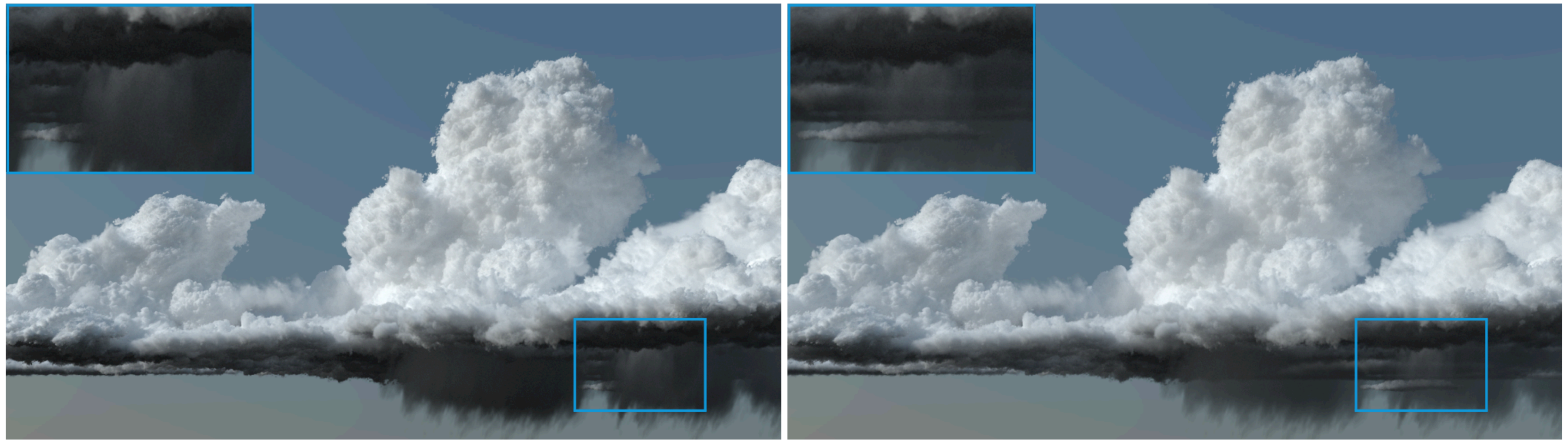
$$L(\mathbf{p}(0), \omega) = \int_0^t T\left(\mathbf{p}(0), \mathbf{p}(t')\right) \left[L_e(\mathbf{p}(t'), \omega) + \sigma_s \int_{S^2} \rho(\omega, \omega') L(\mathbf{p}(t'), \omega') \mathrm{d}\omega'\right] \mathrm{d}t'$$

T: arbitrary functions!

# Other fun research: non-exponential radiative transfer

non exponential T corresponds to correlated particles



exponential

non-exponential

A radiative transfer framework for non-exponential media


BENEDIKT BITTERLI, Dartmouth College, USA
SRINATH RAVICHANDRAN, Dartmouth College, USA
THOMAS MÜLLER, Disney Research, ETH Zürich, Switzerland
MAGNUS WRENNINGE, Pixar Animation Studios, USA
JAN NOVÁK, Disney Research, Switzerland
STEVE MARSCHNER, Cornell University, USA
WOJCIECH JAROSZ, Dartmouth College, USA

# Other fun research: beyond Mie scattering

**Computing the Scattering Properties of Participating Media Using Lorenz-Mie Theory**

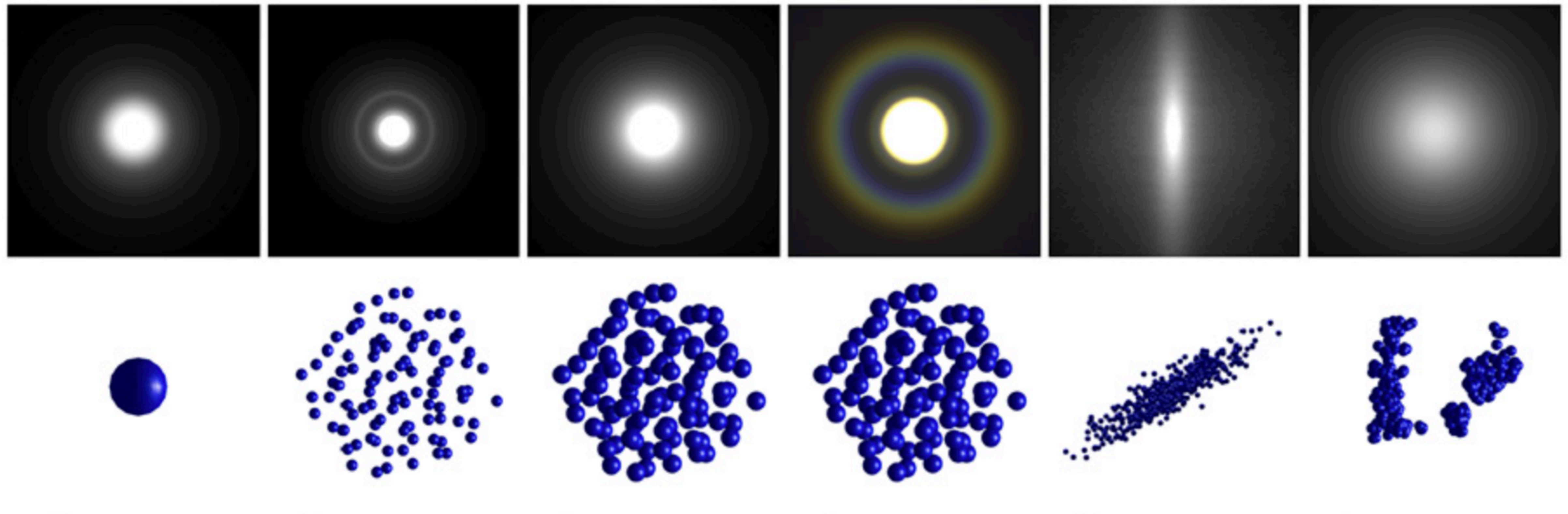
Jeppe Revall Frisvad[1] Niels Jørgen Christensen[1] Henrik Wann Jensen[2]

[1]Informatics and Mathematical Modelling, Technical University of Denmark
[2]University of California, San Diego

**Beyond Mie Theory: Systematic Computation of Bulk Scattering Parameters based on Microphysical Wave Optics**

Yu Guo[1], Adrian Jarabo[2], and Shuang Zhao[1]
[1]University of California, Irvine [2]Universidad de Zaragoza - I3A

ACM Transactions on Graphics (SIGGRAPH Asia 2021), 40(6), 2021



| | | | | | |
|---|---|---|---|---|---|
| $N^{\mathrm{cls}} = 1$, $a_i = 300$nm | $N^{\mathrm{cls}} = 100$, $a_i = 300$nm | $N^{\mathrm{cls}} = 100$, $a_i = 500$nm | $N^{\mathrm{cls}} = 100$, $a_i = 500$nm | $N^{\mathrm{cls}} = 100$, $a_i = 500$nm | $N^{\mathrm{cls}} = 100$, $a_i = 500$nm |
| Isotropic | Isotropic | Isotropic | Isotropic | Anisotropic | Postively correlated |
| $\lambda = 700$nm | $\lambda = 700$nm | $\lambda = 700$nm | Multi-spectral | $\lambda = 700$nm | $\lambda = 400$nm |

# Other fun research: phase functions for rainbows

## Physically-Based Simulation of Rainbows

IMAN SADEGHI
University of California, San Diego
ADOLFO MUNOZ
Universidad de Zaragoza
PHILIP LAVEN
Horley, UK
WOJCIECH JAROSZ
Disney Research Zürich, University of California, San Diego
FRANCISCO SERON and DIEGO GUTIERREZ
Universidad de Zaragoza
and
HENRIK WANN JENSEN
University of California, San Diego

(a) (b) (c) (d) (e)

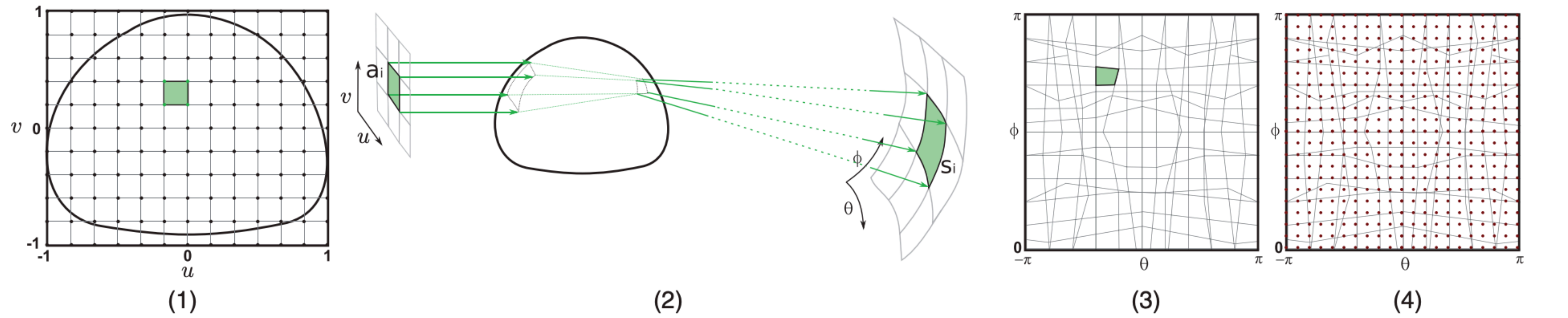
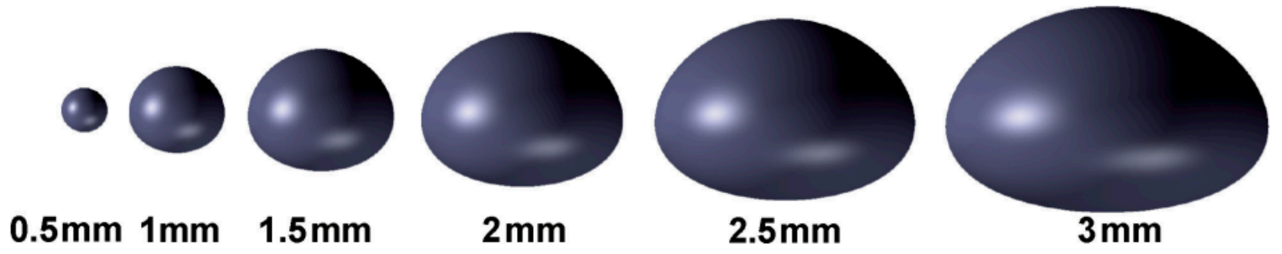



Fig. 8. Steps of the algorithm. (1) Casting the grid of rays towards the particle. (2) Rays are reflected and refracted towards the water drop, forming patches. (3) Outgoing patches are collected in an infinite collecting sphere. (4) The stored patches in the collecing sphere are queried at specific directions, sampling the phase function.

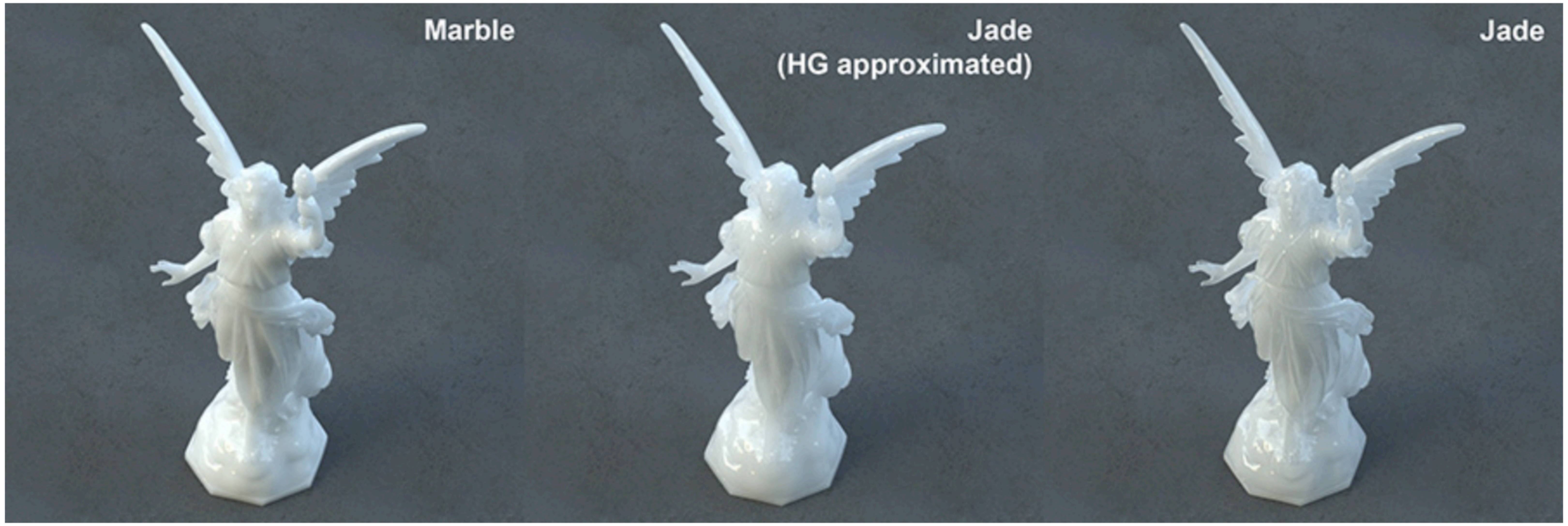


# Other fun research: perception study of phase functions

**Understanding the Role of Phase Function in Translucent Appearance**

Ioannis Gkioulekas[1], Bei Xiao[2], Shuang Zhao[3], Edward Adelson[2], Todd Zickler[1], and Kavita Bala[3]
[1]Harvard School of Engineering and Applied Sciences, [2]Massachusetts Institute of Technology, [3]Cornell University

**ACM Transactions on Graphics, 32(5), September 2013**



# Other fun research:
# acquiring phase functions

## Acquiring Scattering Properties of Participating Media by Dilution


Srinivasa G. Narasimhan (CMU)
Mohit Gupta (CMU)
Craig Donner (UCSD)
Ravi Ramamoorthi (Columbia University)
Shree Nayar (Columbia University)
Henrik Wann Jensen (UCSD)


(a) Acquired photographs (b) Rendering at low concentrations (c) Rendering at natural concentrations

Figure 1: *(a) Photographs of our simple setup consisting of a glass tank and a bulb, filled with diluted participating media (from top, MERLOT, CHARDONNAY, YUENGLING beer and milk). The colors of the bulb and the glow around it illustrate the scattering and absorption properties in these media. At low concentrations, single scattering of light is dominant while multiple scattering of light is negligible. From a single HDR photograph, we robustly estimate all the scattering properties of the medium. Once these properties are estimated, a standard volumetric Monte Carlo technique can be used to create renderings at any concentration and with multiple scattering, as shown in (b) and (c). While the colors are only slightly visible in the diluted setting in (b), notice the bright colors of the liquids - deep red and golden-yellow wines, soft white milk, and orange-red beer - in their natural concentrations. Notice, also the differences in the caustics and the strong interreflections of milk onto other liquids.*

## Inverse Volume Rendering with Material Dictionaries


Ioannis Gkioulekas[1], Shuang Zhao[2], Kavita Bala[2], Todd Zickler[1], and Anat Levin[3]
[1]Harvard School of Engineering and Applied Sciences, [2]Cornell University, [3]Weizmann Institute of Science


**ACM Transactions on Graphics (SIGGRAPH Asia 2013), 32(6), November 2013**

# Next time:
# Monte Carlo evaluation of transmittance

$$\exp\left(-\int_0^t \sigma_t(t')\mathrm{d}t\right)$$