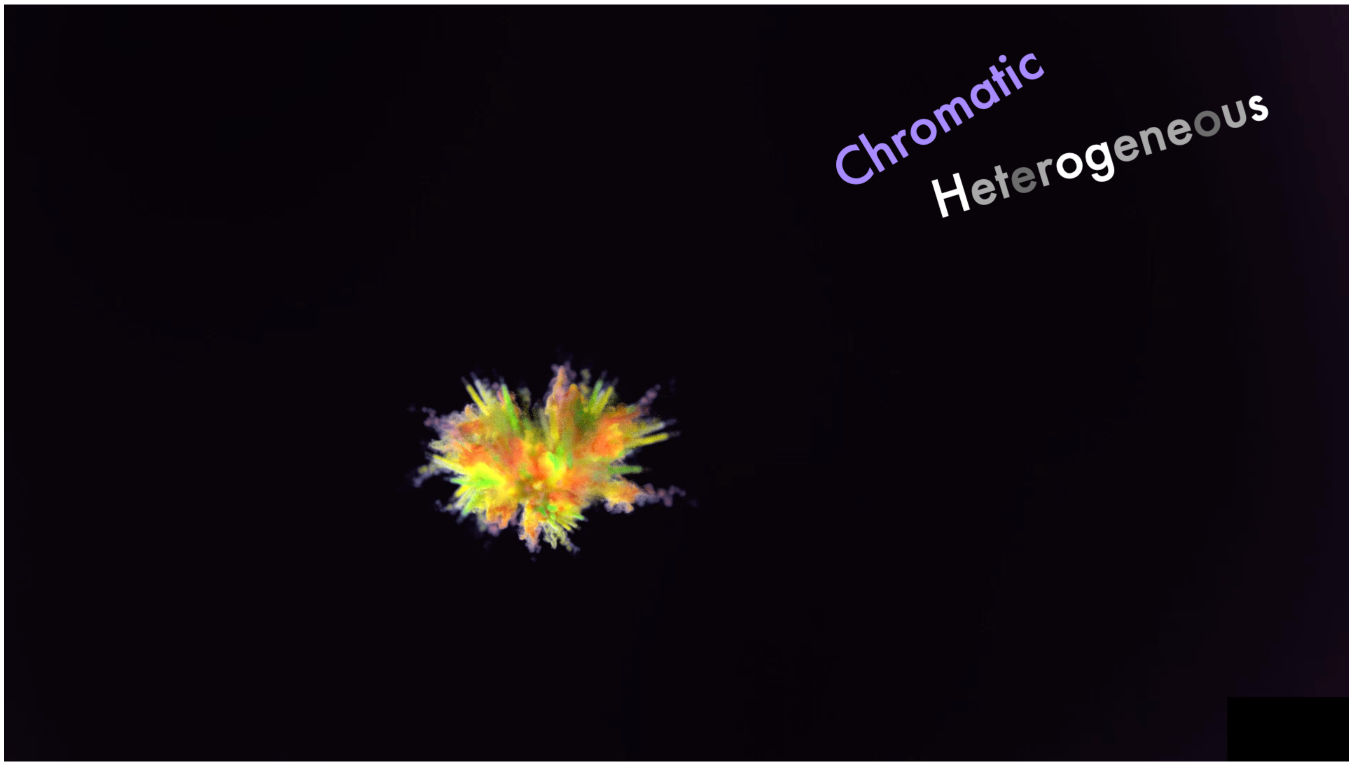
Walt Disney
ANIMATION STUDIOS
©Disney
Walt Disney ANIMATION STUDIOS

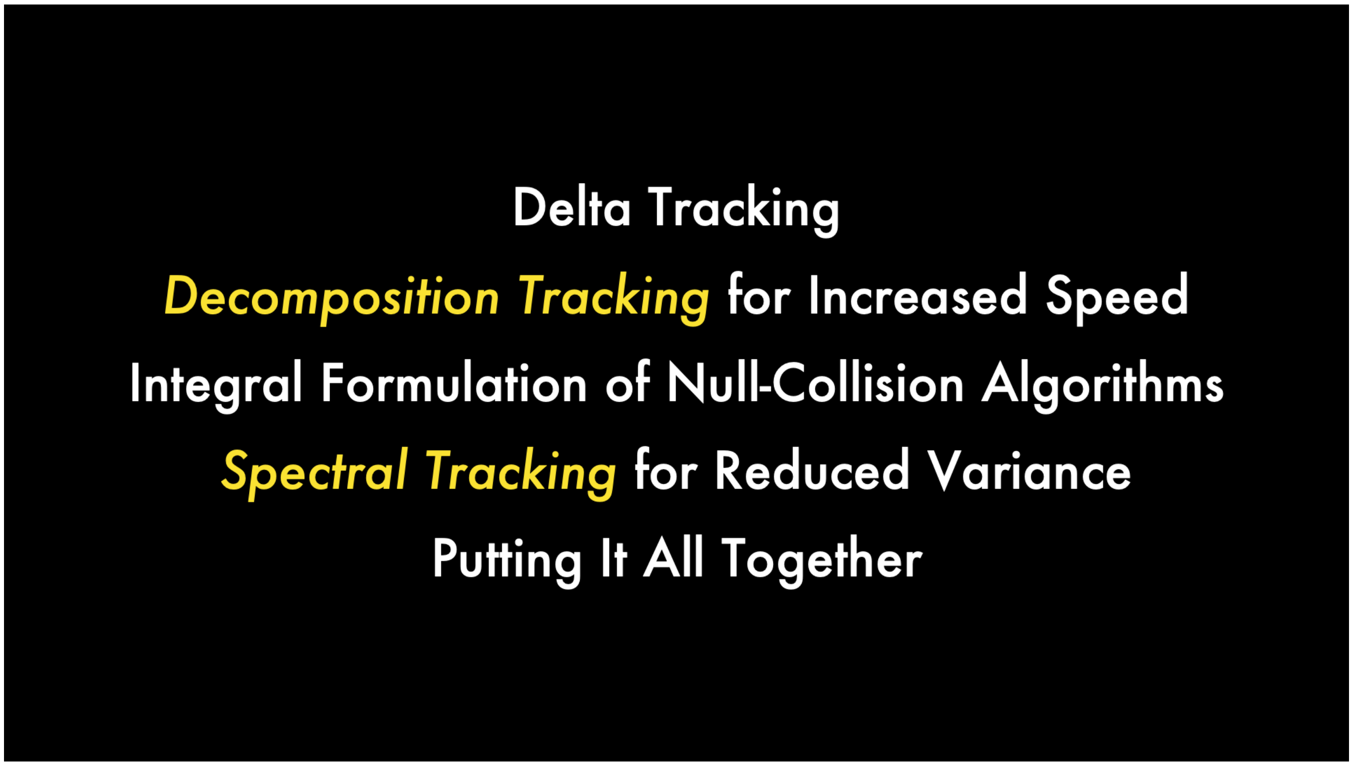
Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes

authored by **Peter Kutz**, **Ralf Habel**, **Karl Li**, and **Jan Novák**
and presented by **Peter Kutz**

Hi, I’m Peter Kutz and this talk is about the paper Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes.



In this work, our focus is more efficient unbiased rendering of volumes, in particular volumes that are chromatic and heterogeneous. Chromatic means that their scattering and absorption properties vary for different wavelengths of light, and heterogeneous means that their properties vary throughout space.



Delta Tracking

Decomposition Tracking for Increased Speed

Integral Formulation of Null-Collision Algorithms

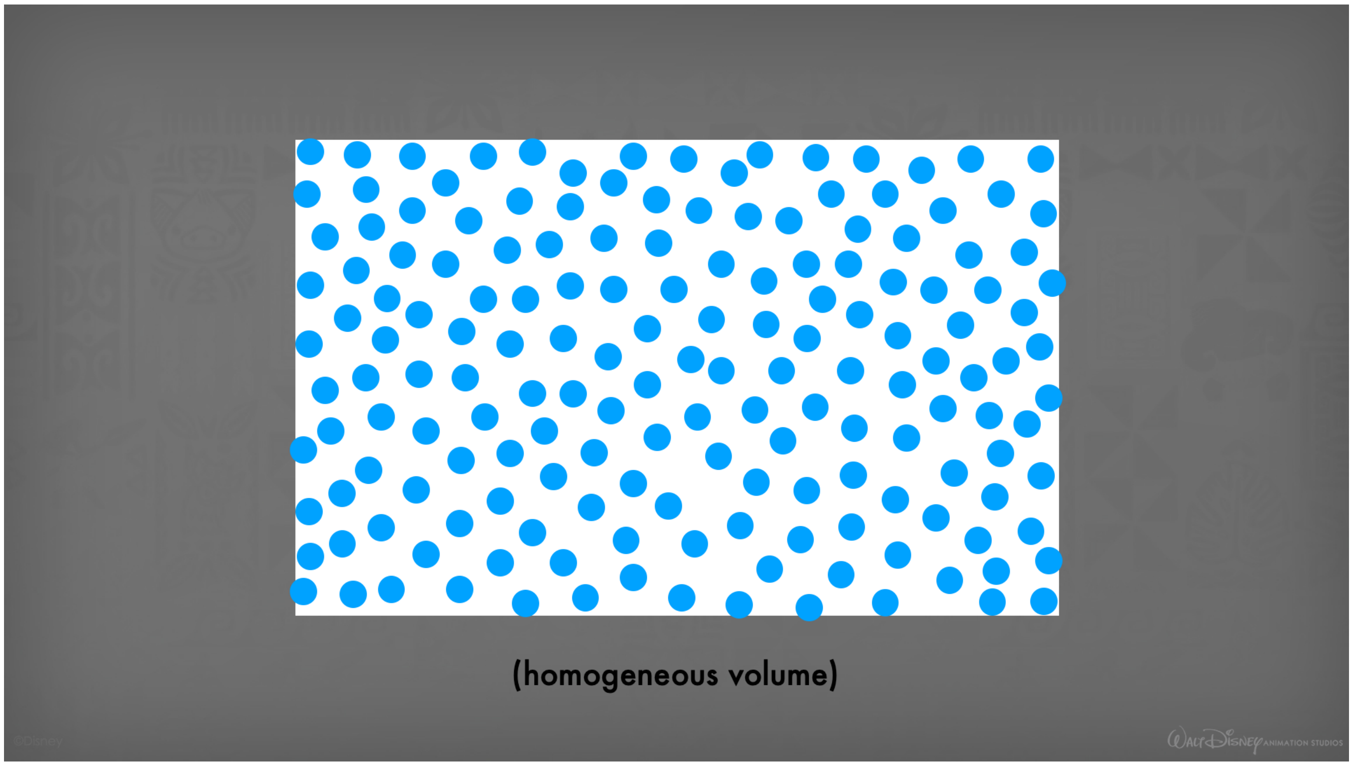
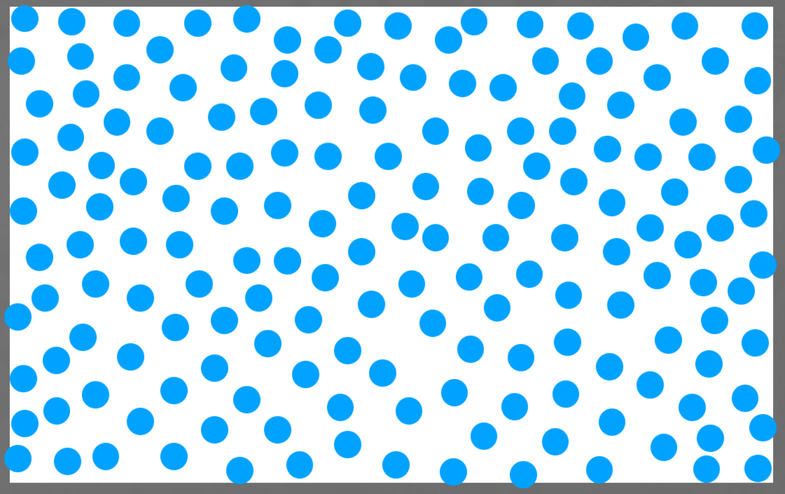
Spectral Tracking for Reduced Variance

Putting It All Together

First we will review Delta Tracking which is the foundation of our work.
Then we will introduce Decomposition Tracking.
We will also show the Integral Formulation of Null Collision Algorithms which we imported into graphics from physics literature.
Then we will also introduce Spectral Tracking.
Finally we will describe how our new algorithms can be combined.

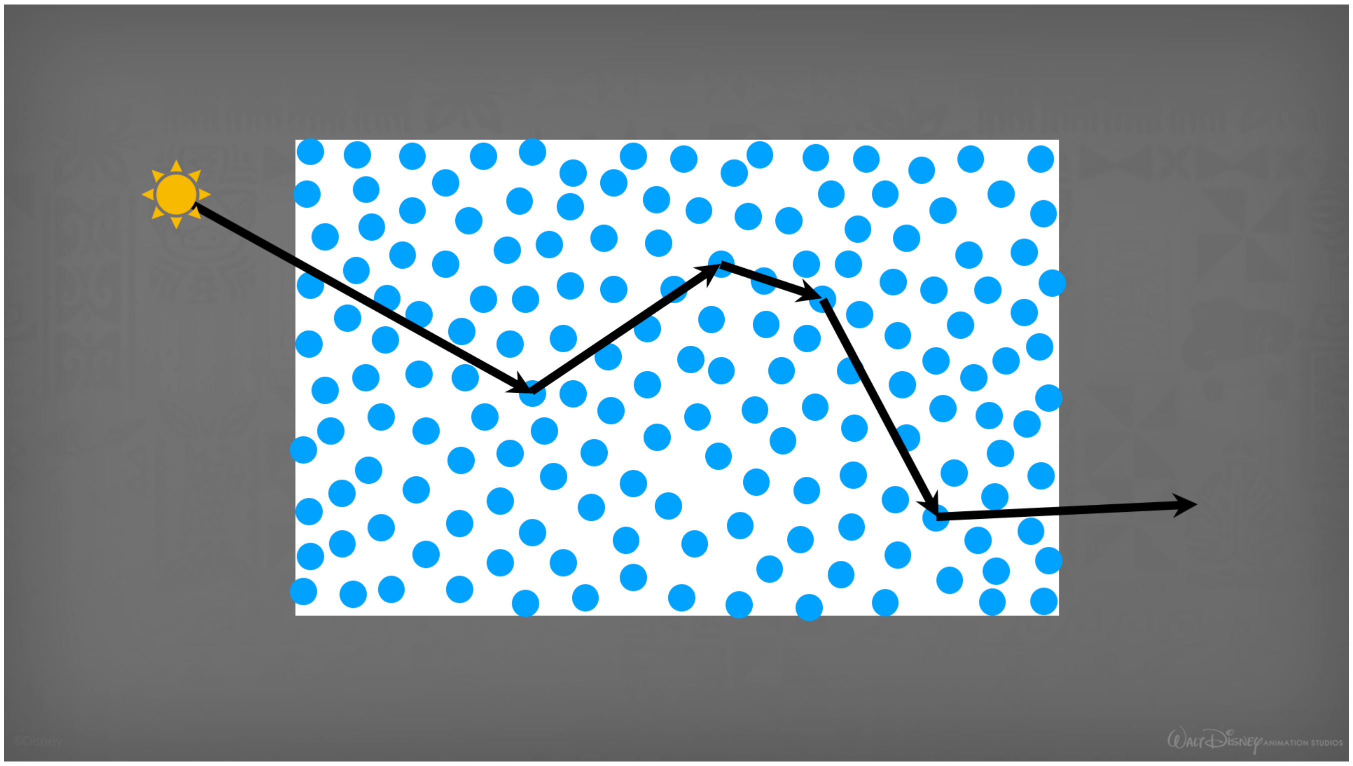
Delta Tracking

Let’s get started with delta tracking.

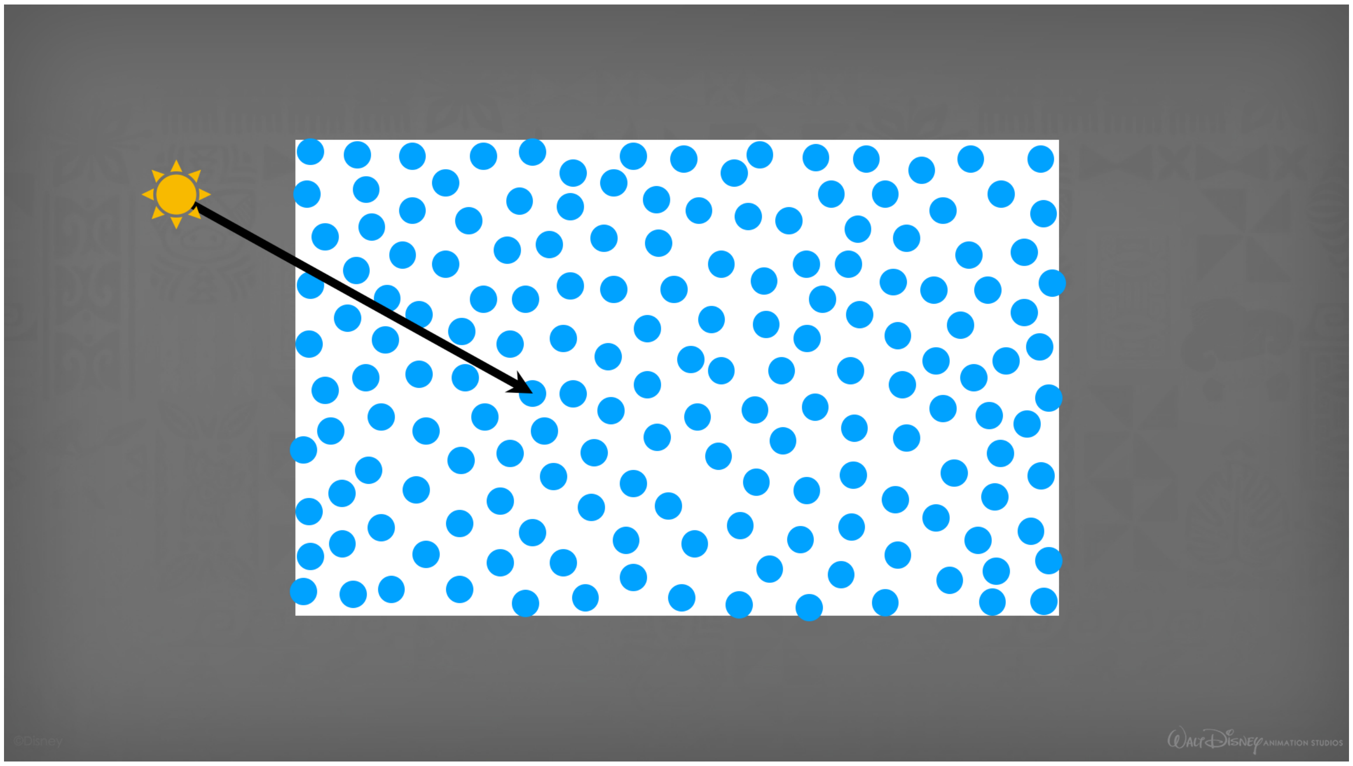



(homogeneous volume)

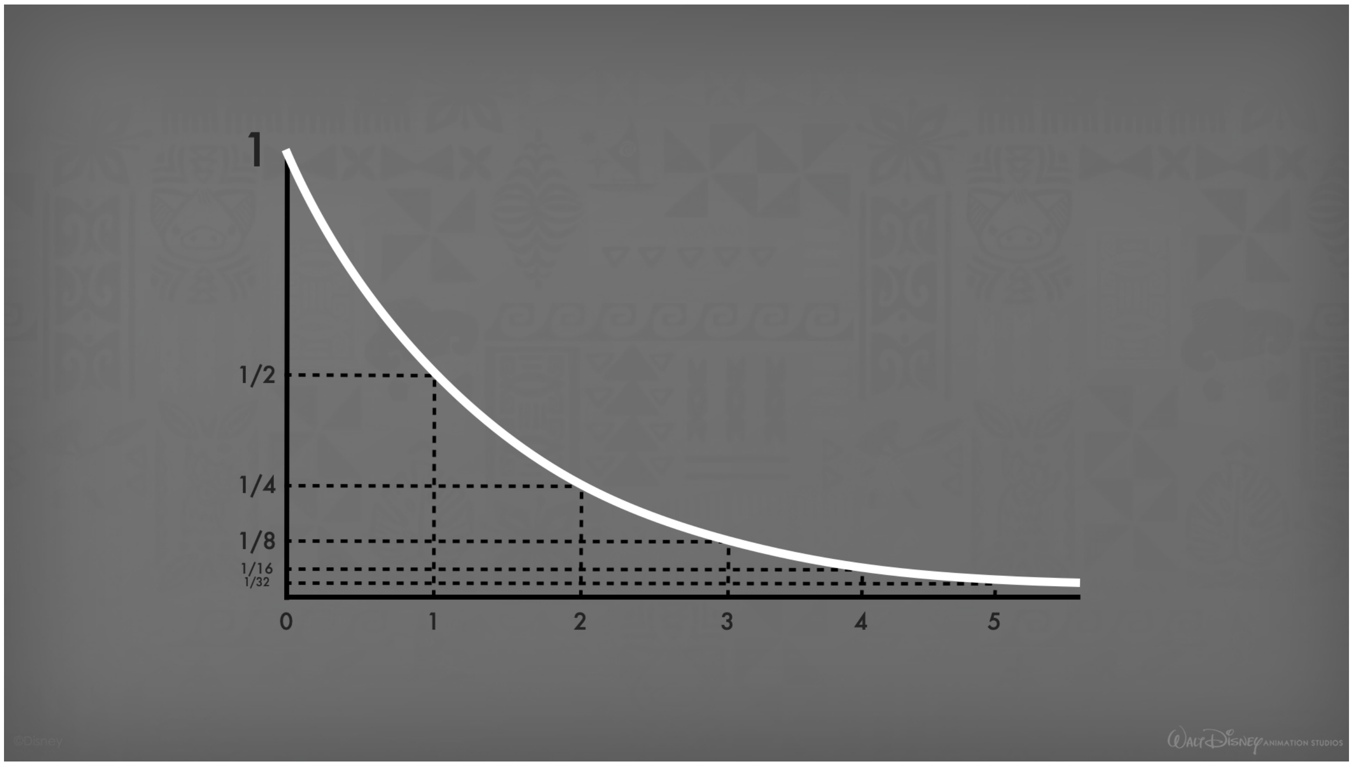
Here’s a schematic 2D diagram of a volume, which can be thought of as being composed of particles. This happens to be a homogeneous volume, one with the same properties throughout. It has a uniform density or extinction coefficient or scattering coefficient.



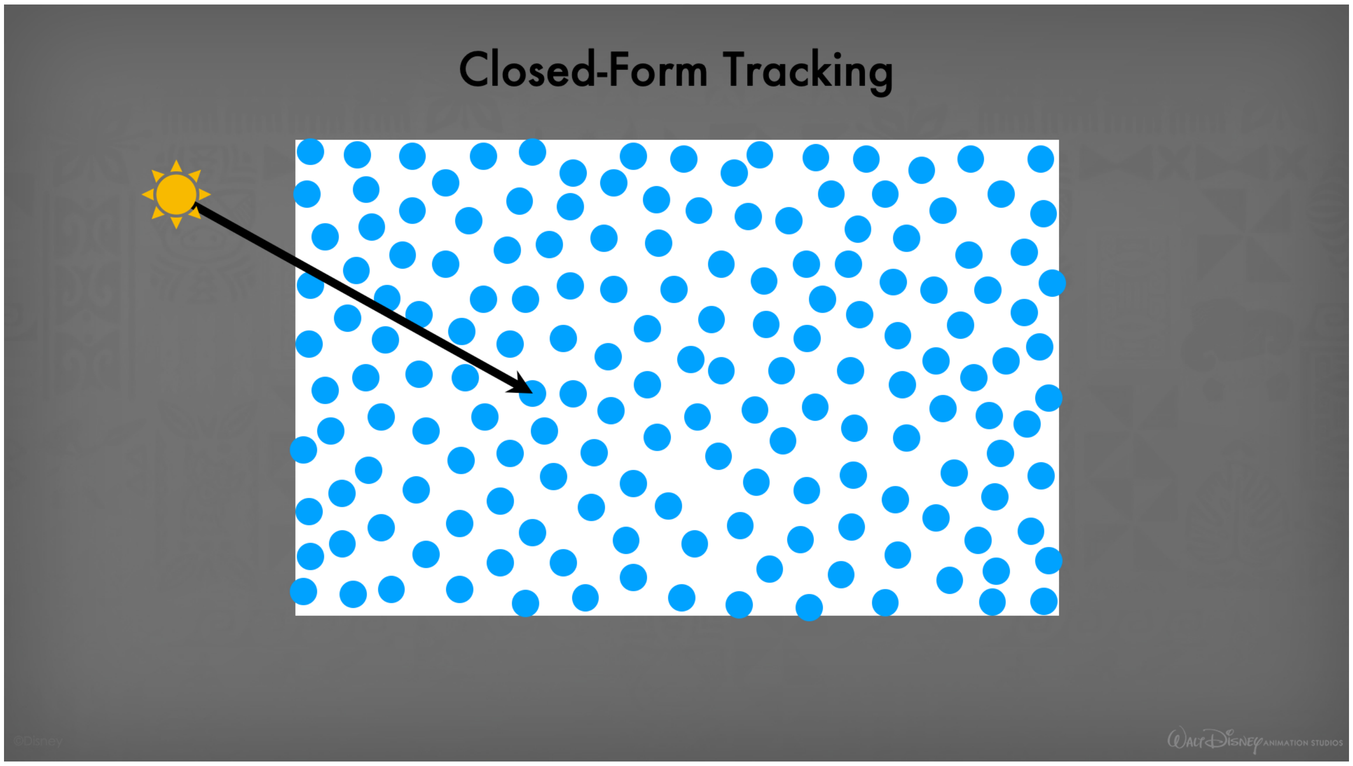
The ultimate goal of volume rendering and of our work is to generate paths of light between a light source and the camera.



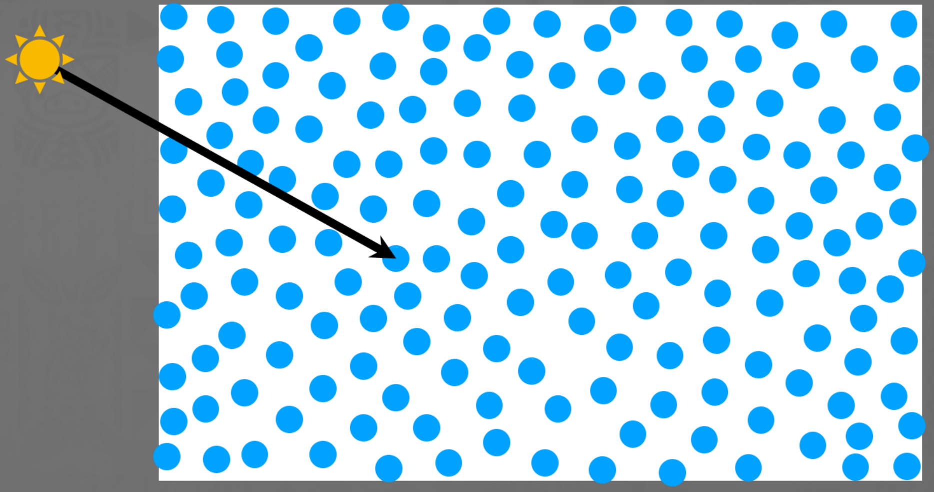
However our focus is the efficient generation of individual path segments, or free paths. For a homogeneous volume like this one this is relatively easy.



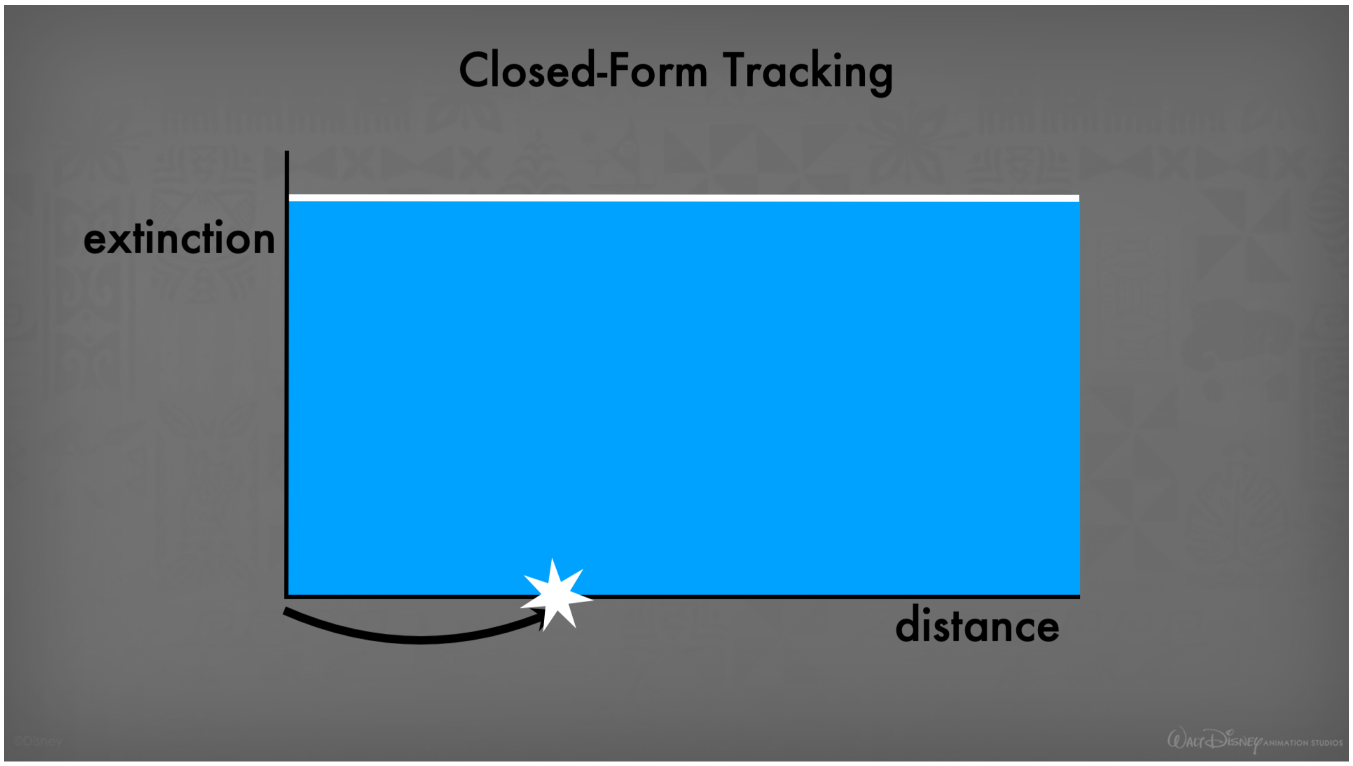
The distribution of scattering and absorption distances is exponential, proportional to the transmittance, and we can draw samples from an exponential distribution directly.



Closed-Form Tracking

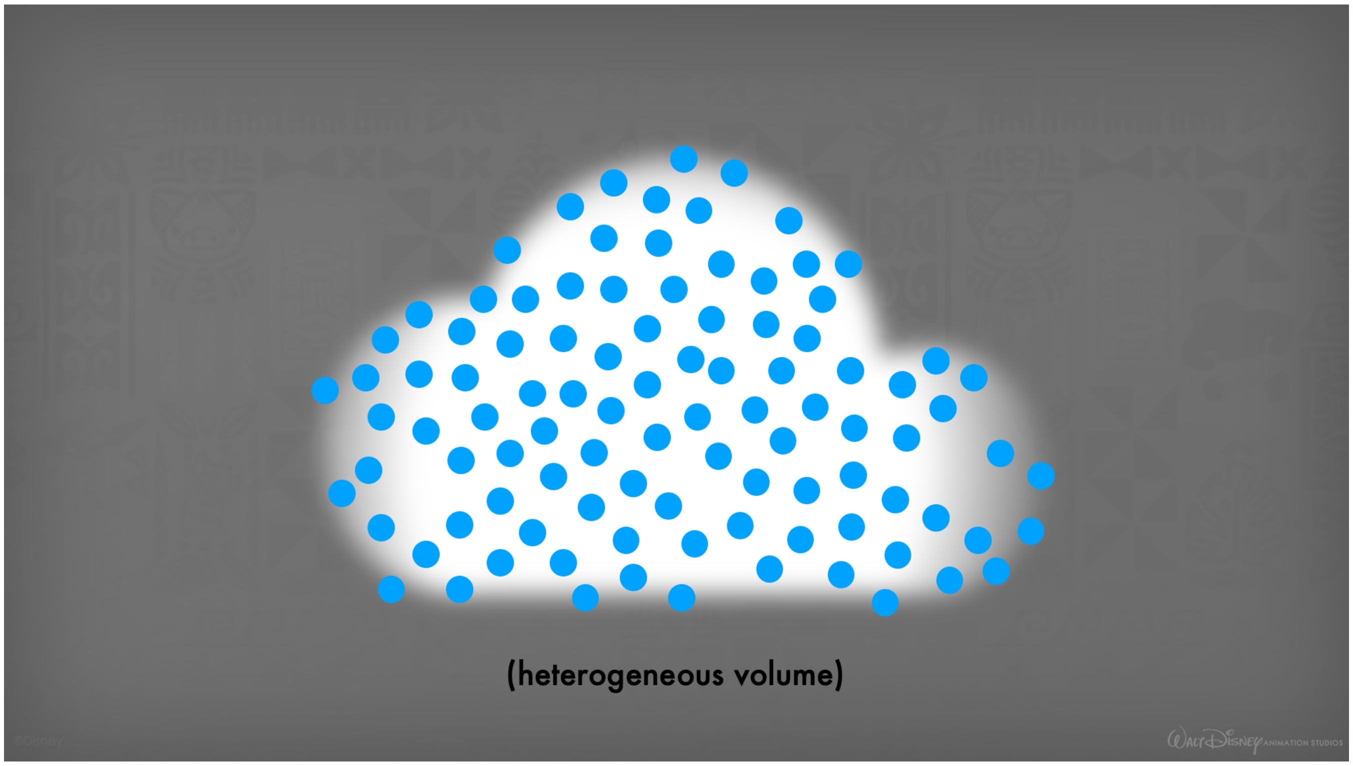


This sampling process is called closed-form tracking, and it immediately gives us a scattering or absorption location. For simplicity, we'll mainly just talk about scattering in this presentation.

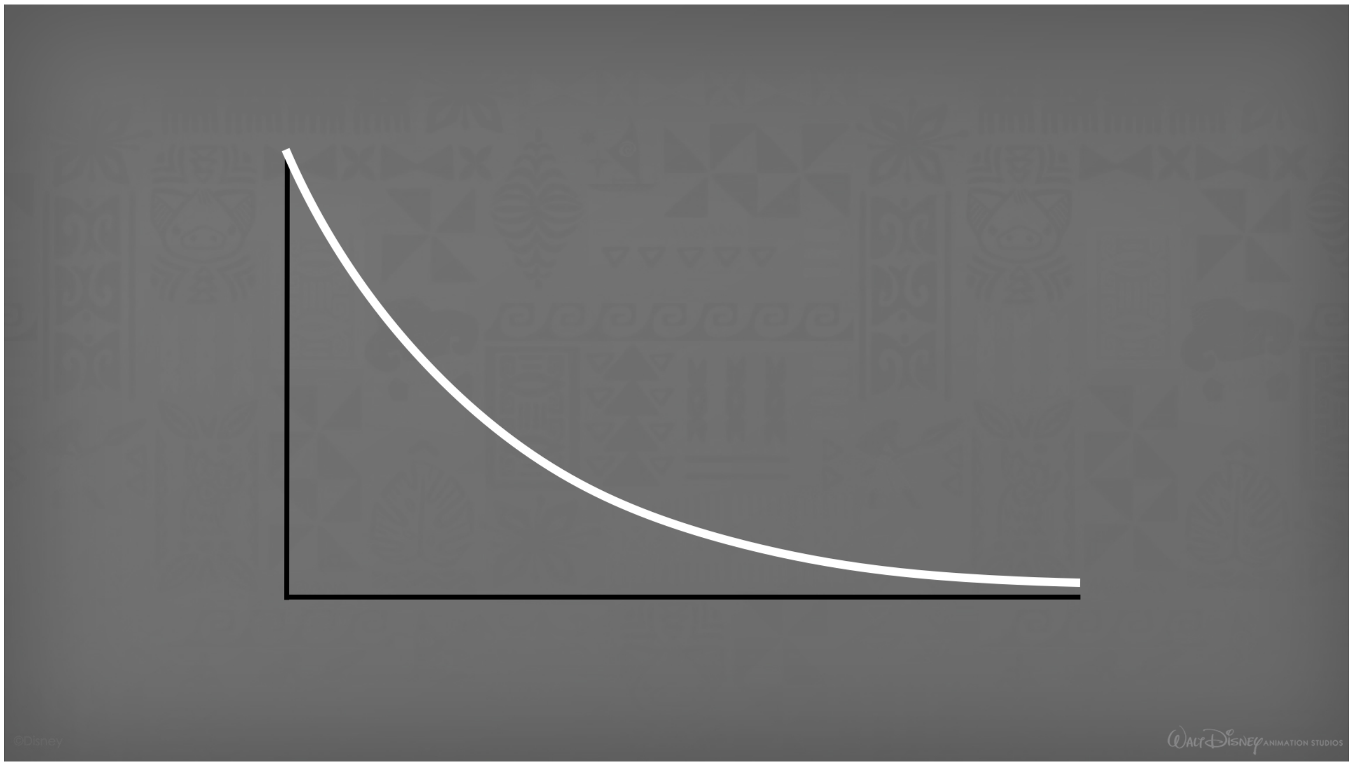


Closed-Form Tracking

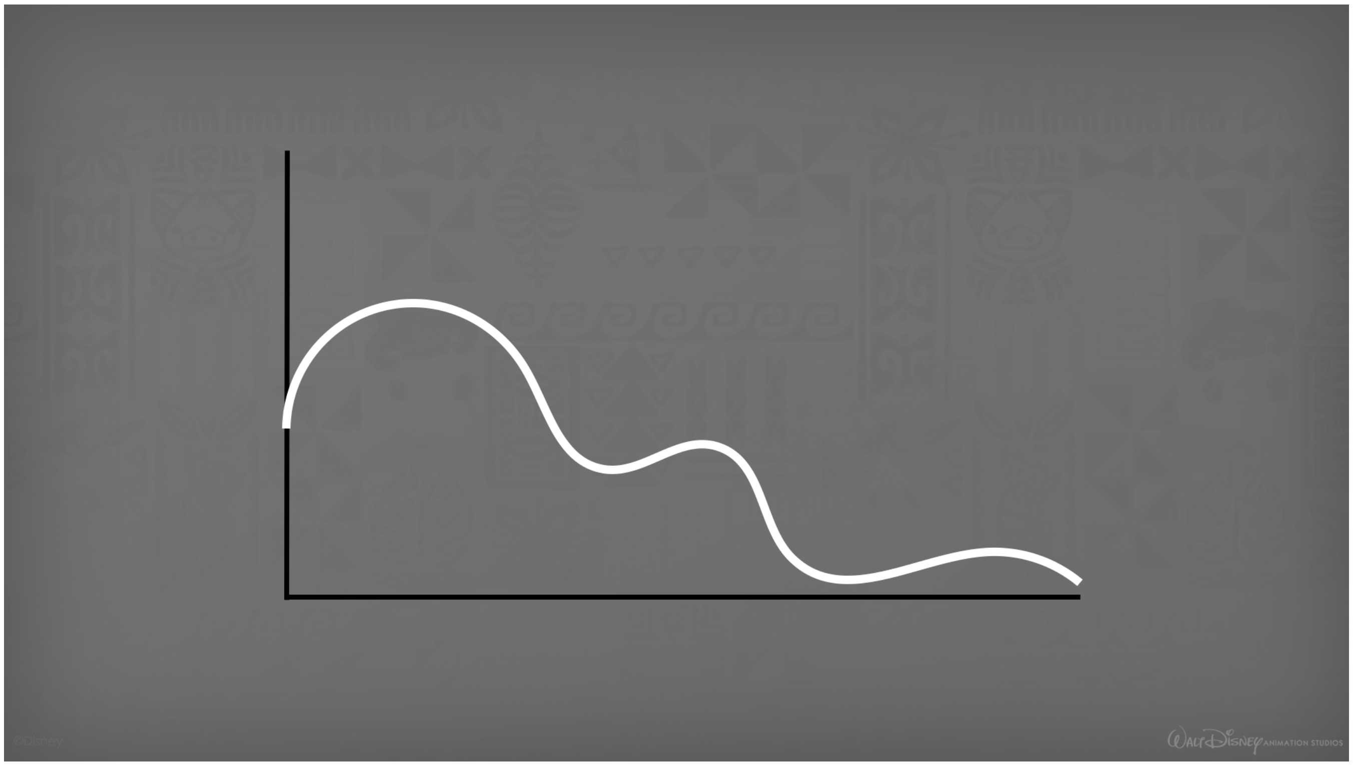
Here’s a different view of the process for a slice of the volume along the ray, with the distance along the ray on the horizontal axis and the extinction coefficient on the vertical axis. We simply sample a single distance from the exponential distribution and that tells us the scattering location.



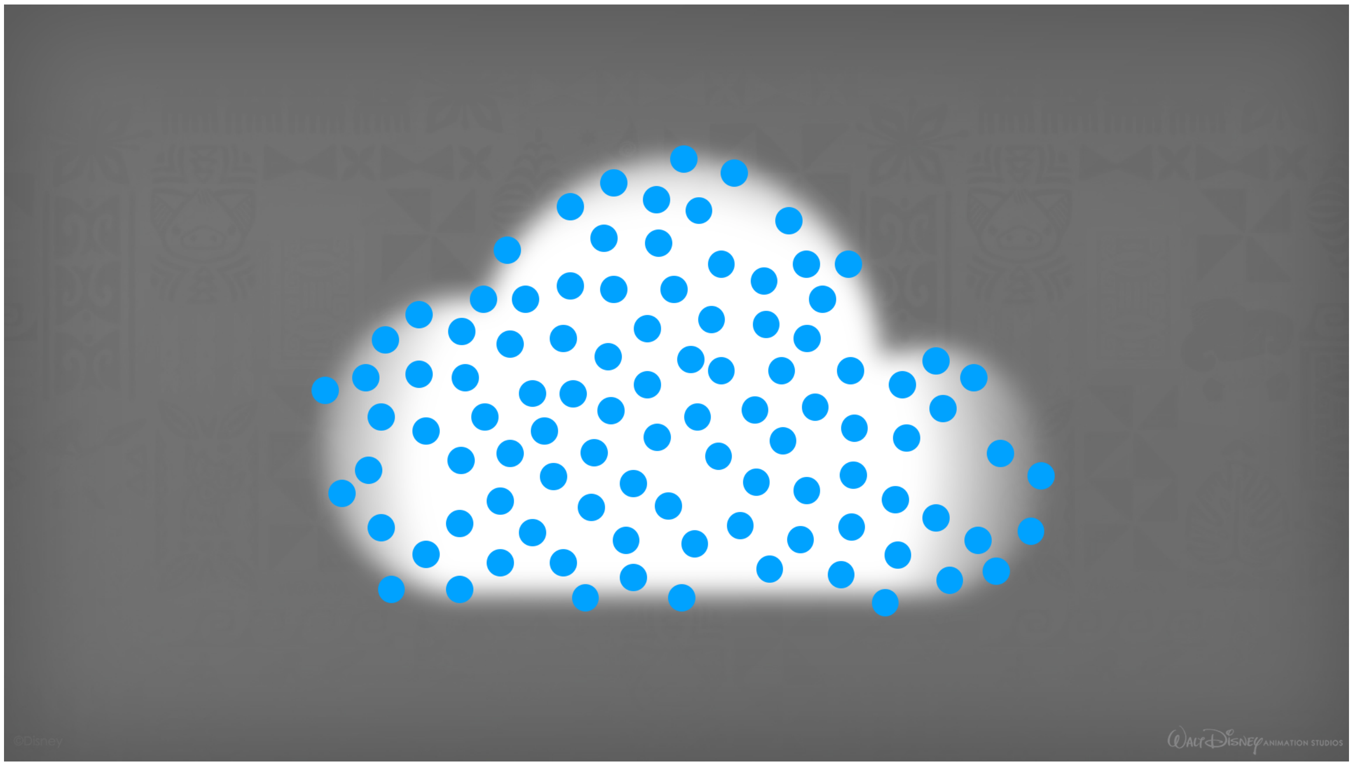
But what if we want to find a scattering location in a heterogeneous volume like this one? This volume has a strange shape and nonuniform density.



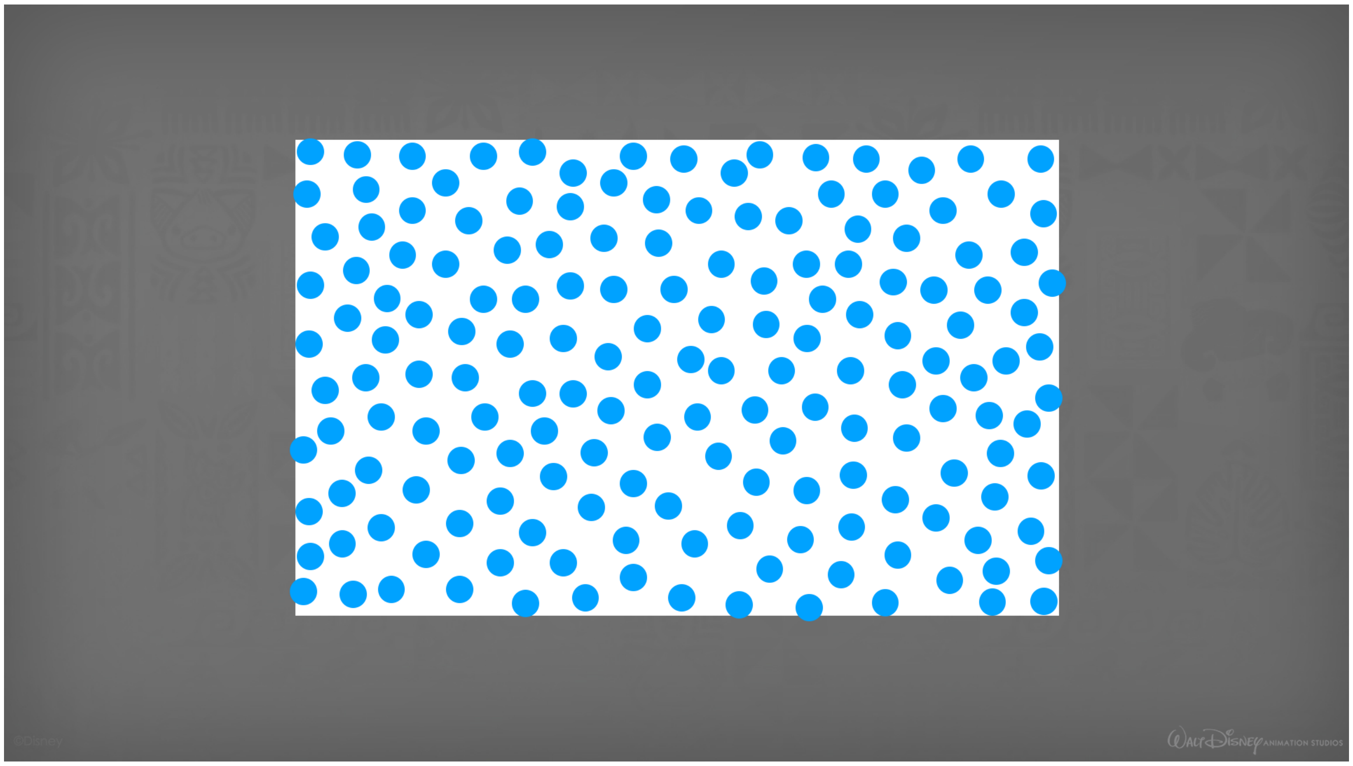
So instead of the distribution of scattering locations being exponential…



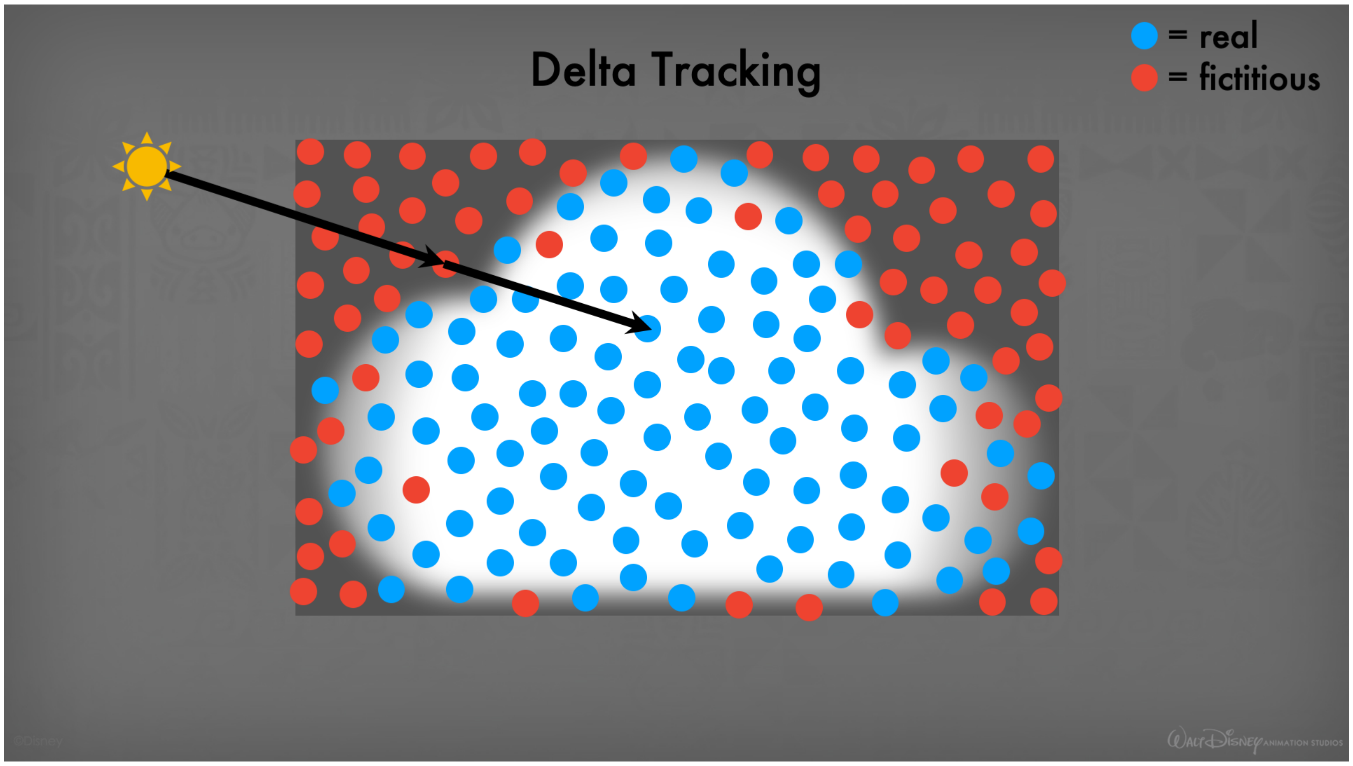
it can be some arbitrary shape.



But let’s observe that this cloud is actually a subset of the homogeneous box that we looked at before:



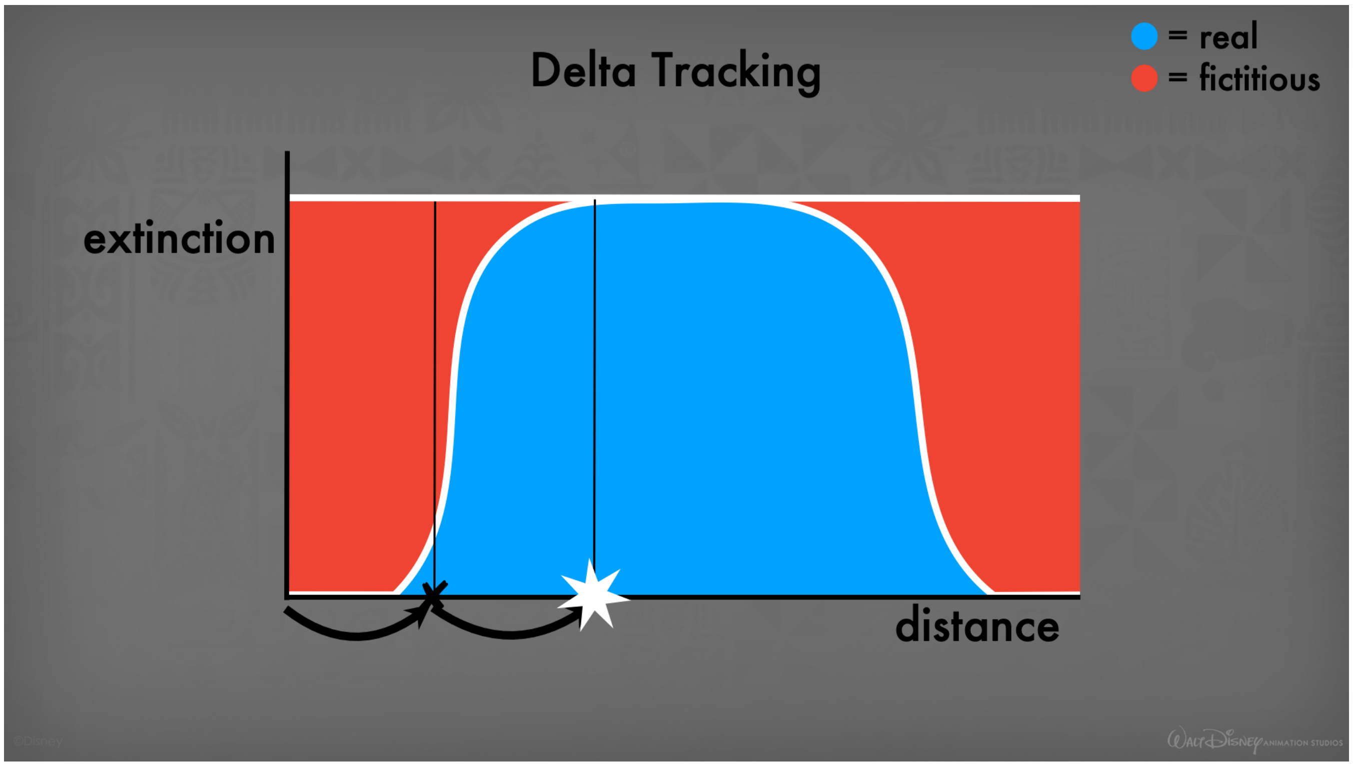
The idea of delta tracking is to use the closed-form homogeneous-volume sampling to sample a heterogeneous volume. This is done by introducing fictitious particles that homogenize the real heterogeneous volume:



Delta Tracking

● = real
● = fictitious

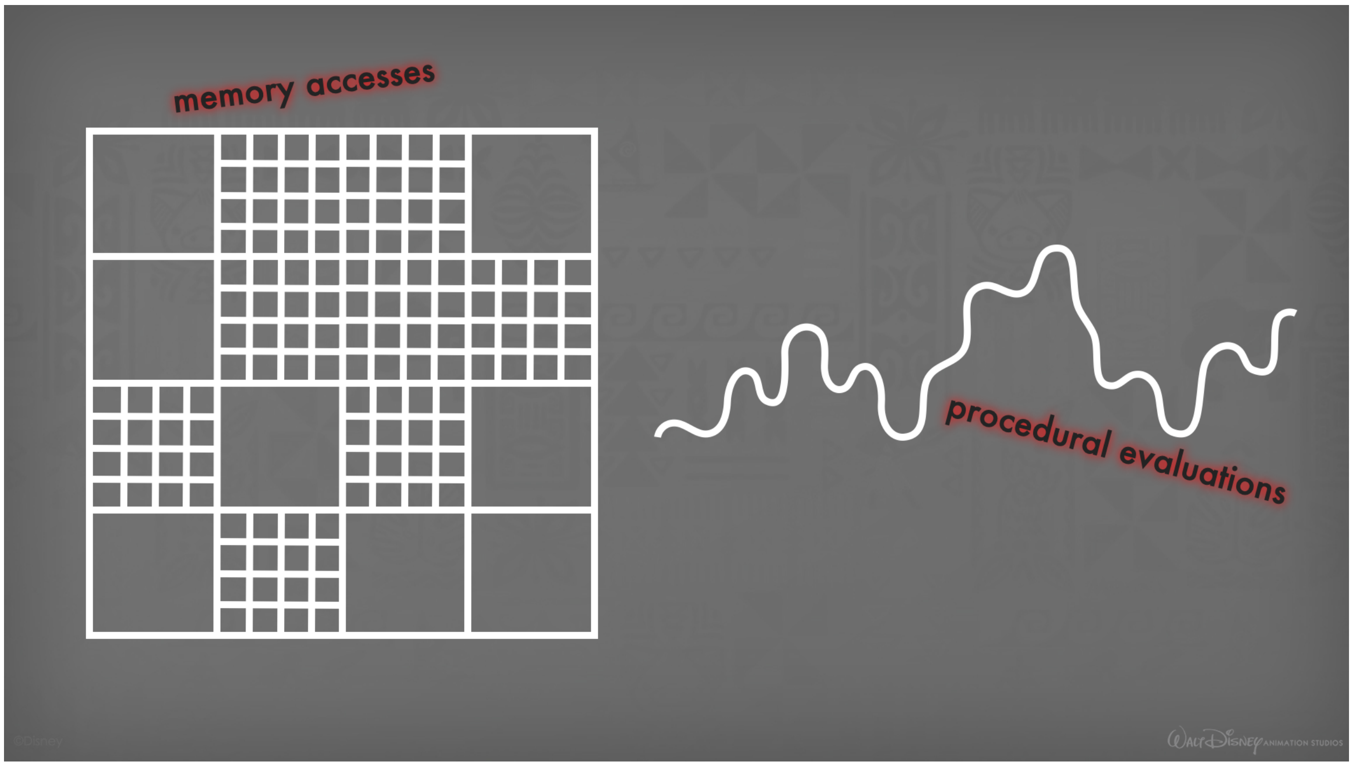
Delta tracking works by repeatedly sampling a distance from the combined volume, and then checking whether a real or fictitious particle was hit. Whenever a fictitious particle is hit, we just continue in the same direction. When a real particle is hit, we scatter. This algorithm perfectly importance samples the transmittance, producing the desired distribution of scatter distances.



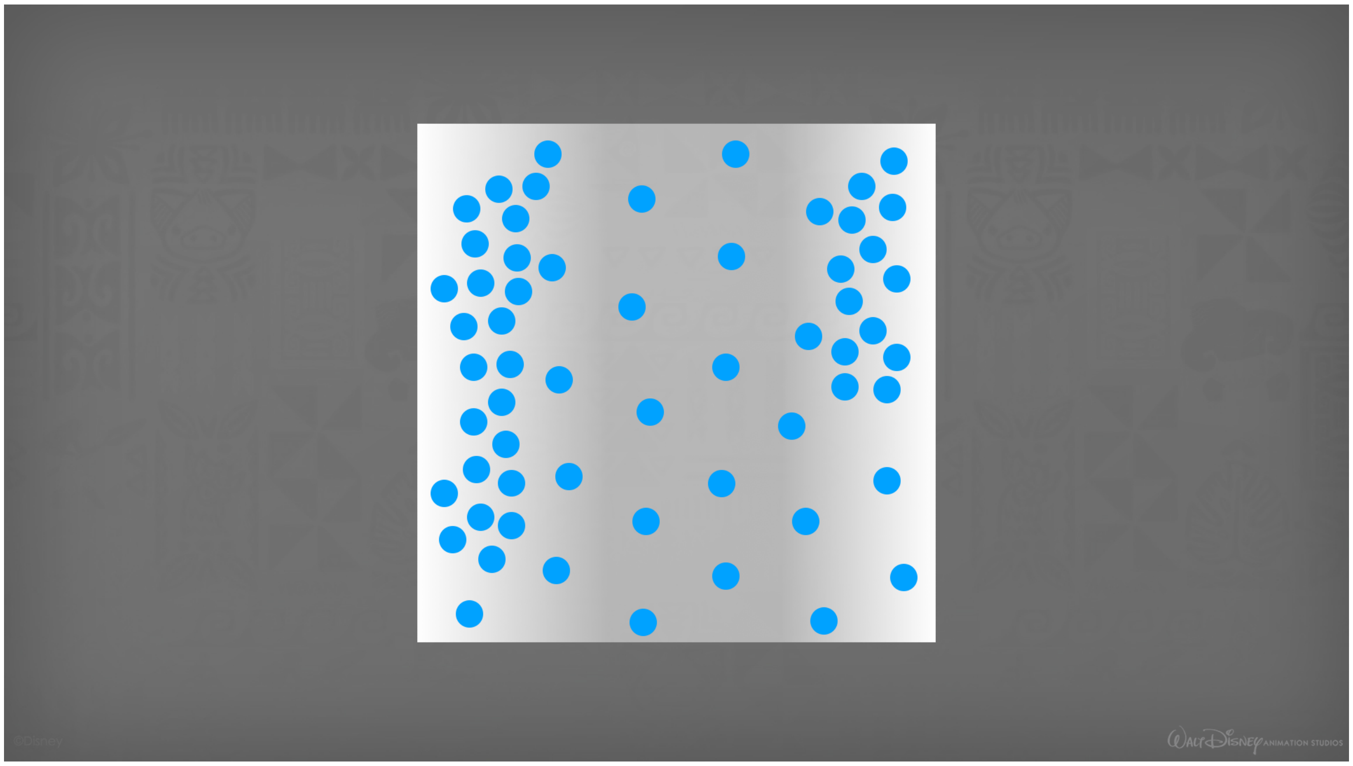
Here’s a plot of the extinction along the ray. We take a step forward according as if the volume were homogeneous with the maximum extinction coefficient of the combined volume, look up the ratio of real and fictitious media at the point, draw a random number and check whether it corresponds to the real or fictitious medium. We repeat this process until we hit the real medium.

Decomposition Tracking

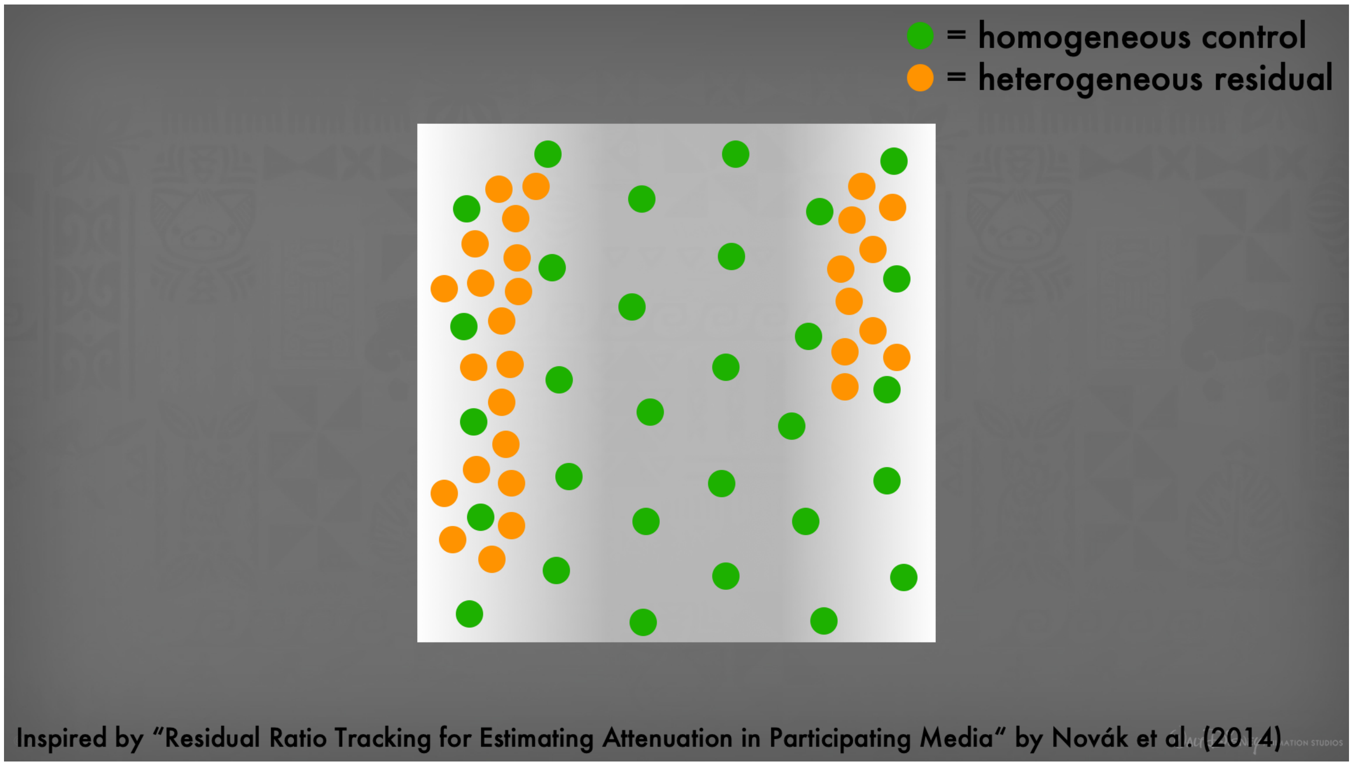
Now we have the background necessary to understand decomposition tracking.



Delta tracking requires a volume property lookup at each step of the algorithm. This usually consists of a memory lookup into a large volumetric data structure or a procedural evaluation. Especially for highly-scattering volumes, these volume property lookups can become a significant bottleneck. If we could reduce the cost of these lookups, we could reduce render times.

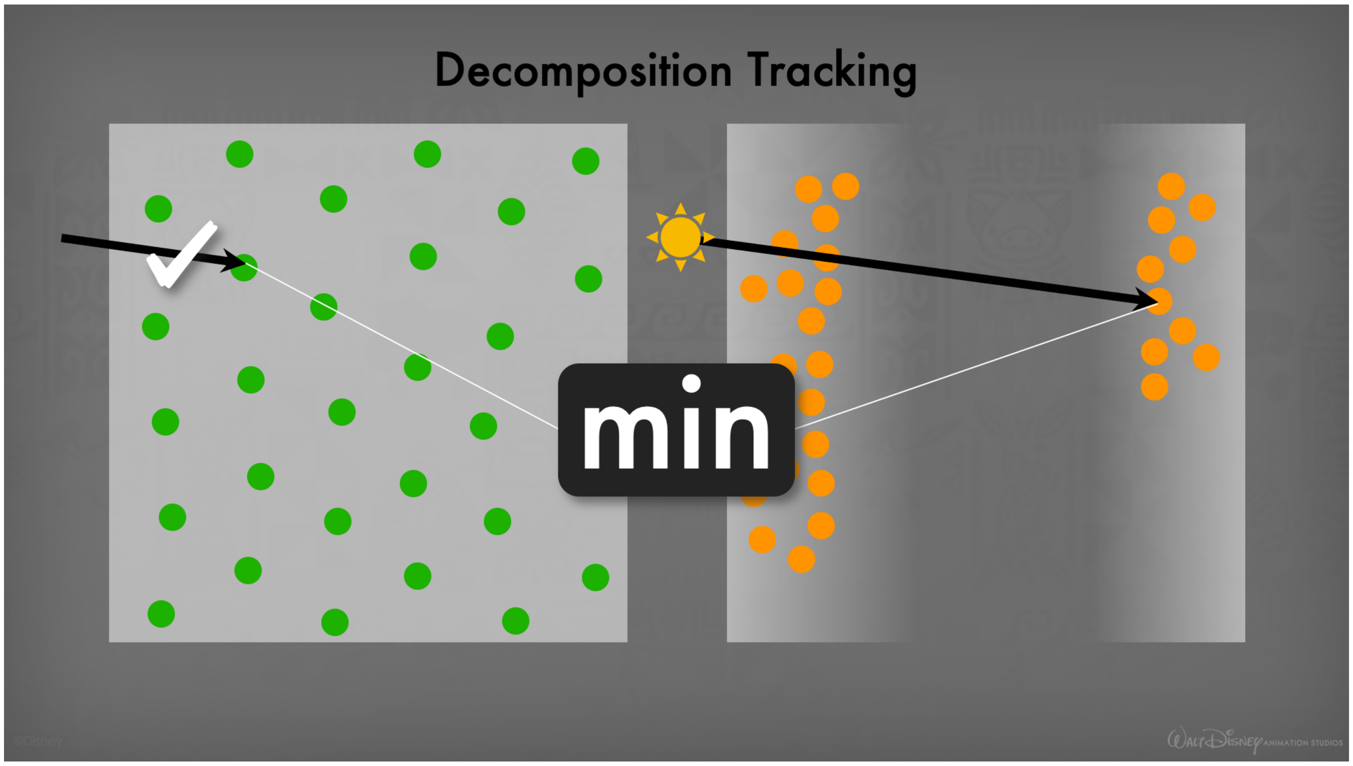


Here’s a diagram of a region of a heterogeneous volume. Along the lines of past work, we could split the volume into superimposed homogeneous and heterogeneous components:

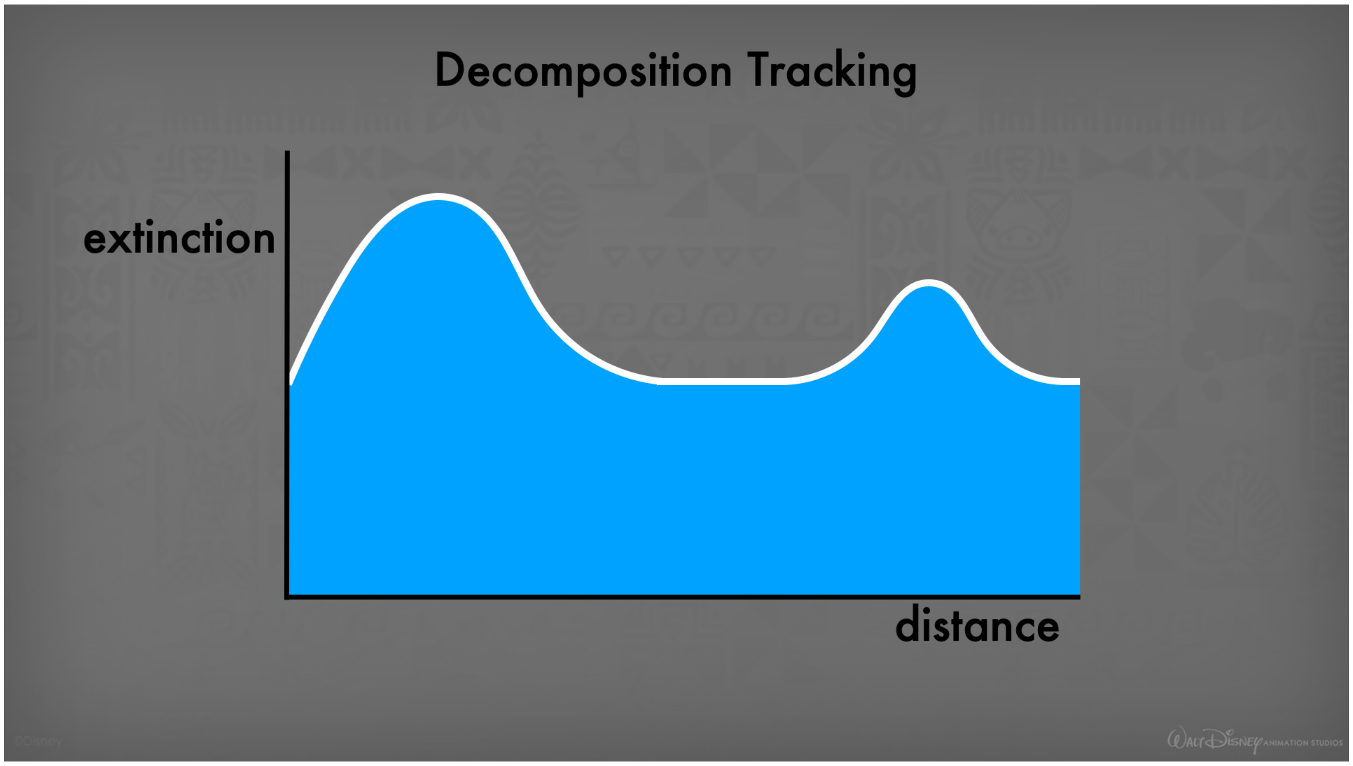

● = homogeneous control
● = heterogeneous residual
Inspired by "Residual Ratio Tracking for Estimating Attenuation in Participating Media" by Novák et al. (2014)

For calculating transmittance, the residual ratio tracking algorithm introduced in 2014 by Novák and colleagues calculates a transmittance in each component and multiplies them together to find the final value. Can we find an analogous approach that works for distance sampling?

Decomposition Tracking

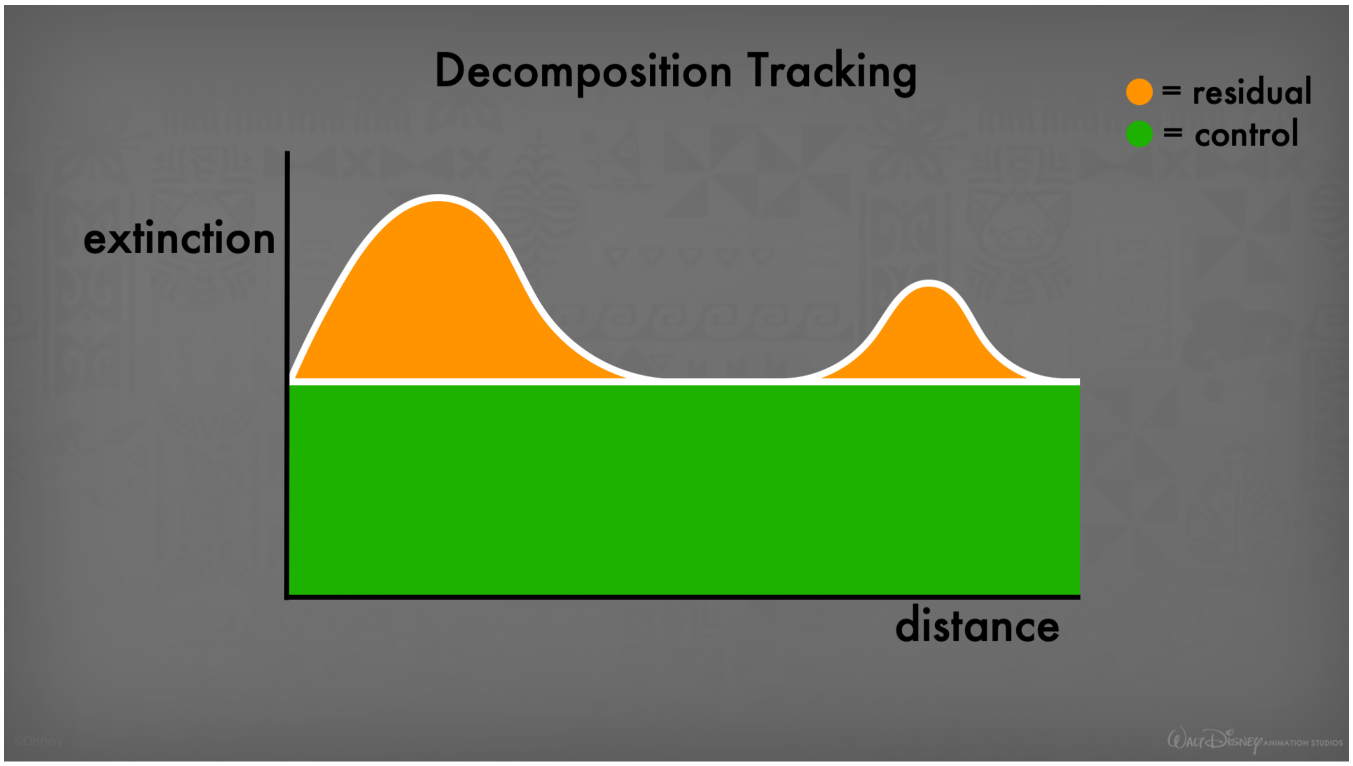


It turns out that we can, and this leads to the basic version of our decomposition tracking algorithm. We sample a separate distance in each component. Then to combine them into a single distance, we take the minimum of the two. As we prove in the paper, this produces the correct distribution of distances, the same distribution as delta tracking.



Decomposition Tracking

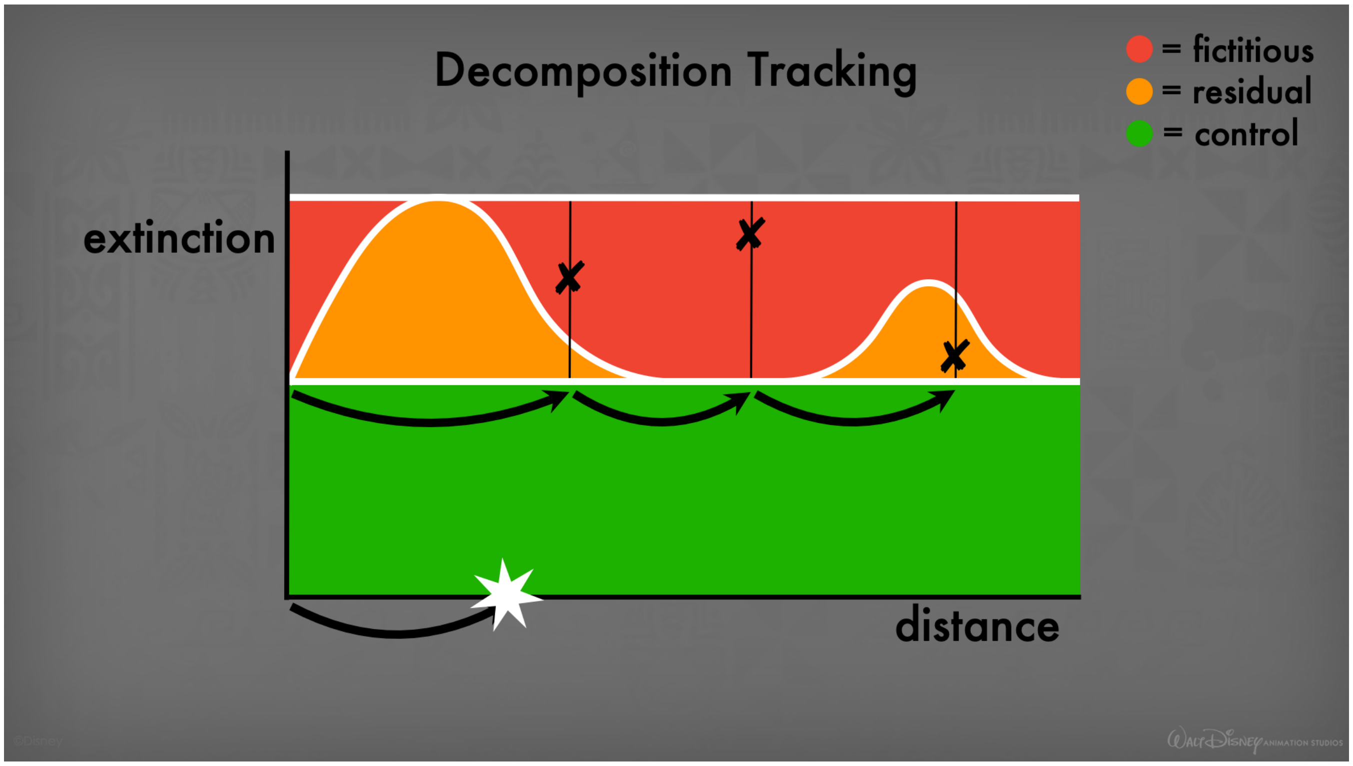
Here’s a 1D slice of that volume along the ray.



Decomposition Tracking

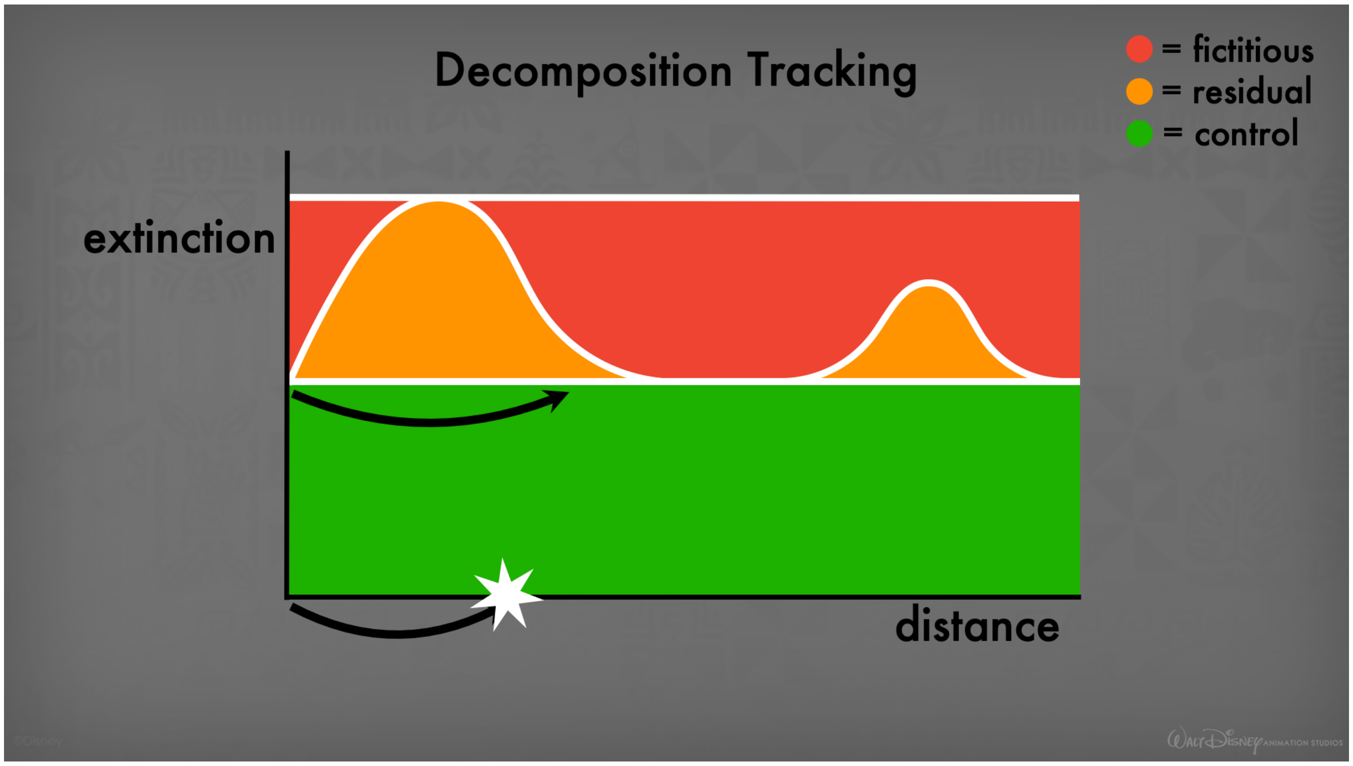
We split it into a homogeneous control and a heterogeneous residual.

Decomposition Tracking



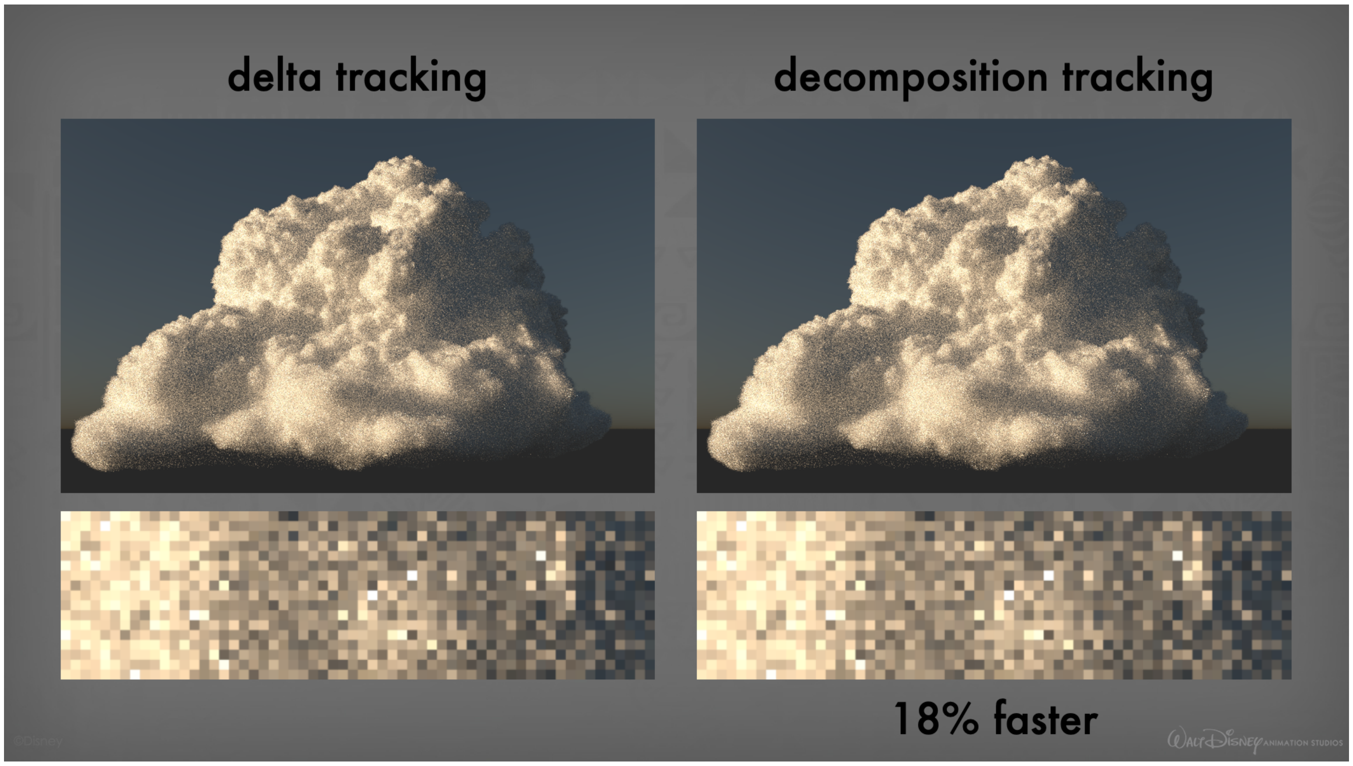
Then we homogenize it by adding a fictitious medium. We then sample a distance from the heterogeneous component using delta tracking. Take a step, null collision, take another step, null collision, take another step, real collision. Then we sample a distance from the homogeneous component using closed-form tracking. Finally, we take the minimum to find the final distance sample for the original volume. But how can this be cheaper than delta tracking? At first it looks like more work if anything.

Decomposition Tracking

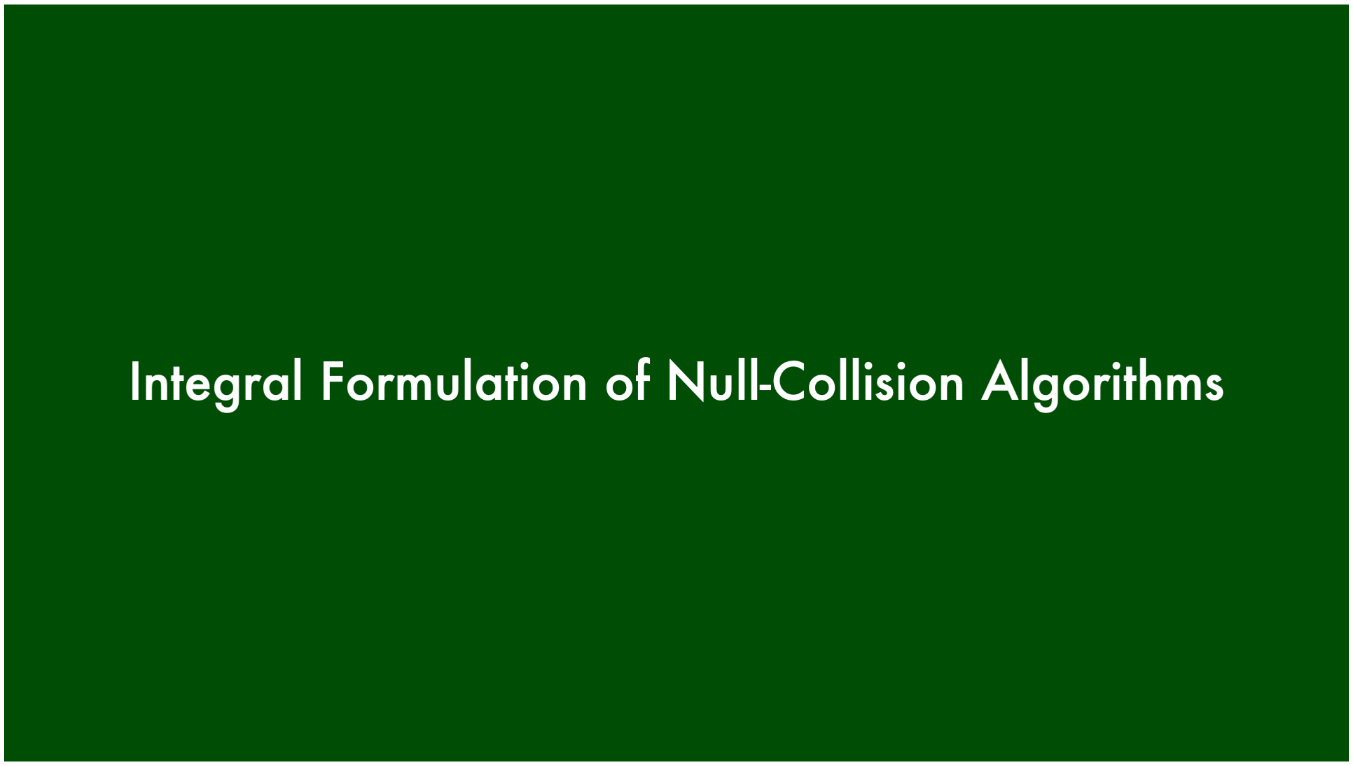


The key to making this performant is to sample the homogeneous component first. Here’s an illustration. We sample the distance from the homogeneous component using closed-form tracking. Then we begin delta tracking on the heterogeneous component. We take a step, as before. But now, we know the homogeneous distance in advance. And we know that as soon as we step beyond it, it doesn’t matter how far we go because we won’t use it anyway — we know the minimum will be the homogeneous distance. So in this case, we don’t even have to do a single lookup of the spatially varying volume properties.

By the way, in the paper, we show that there’s also another way to perform decomposition tracking that doesn’t require taking the minimum and makes it easier to combine decomposition with other algorithms such as spectral tracking.

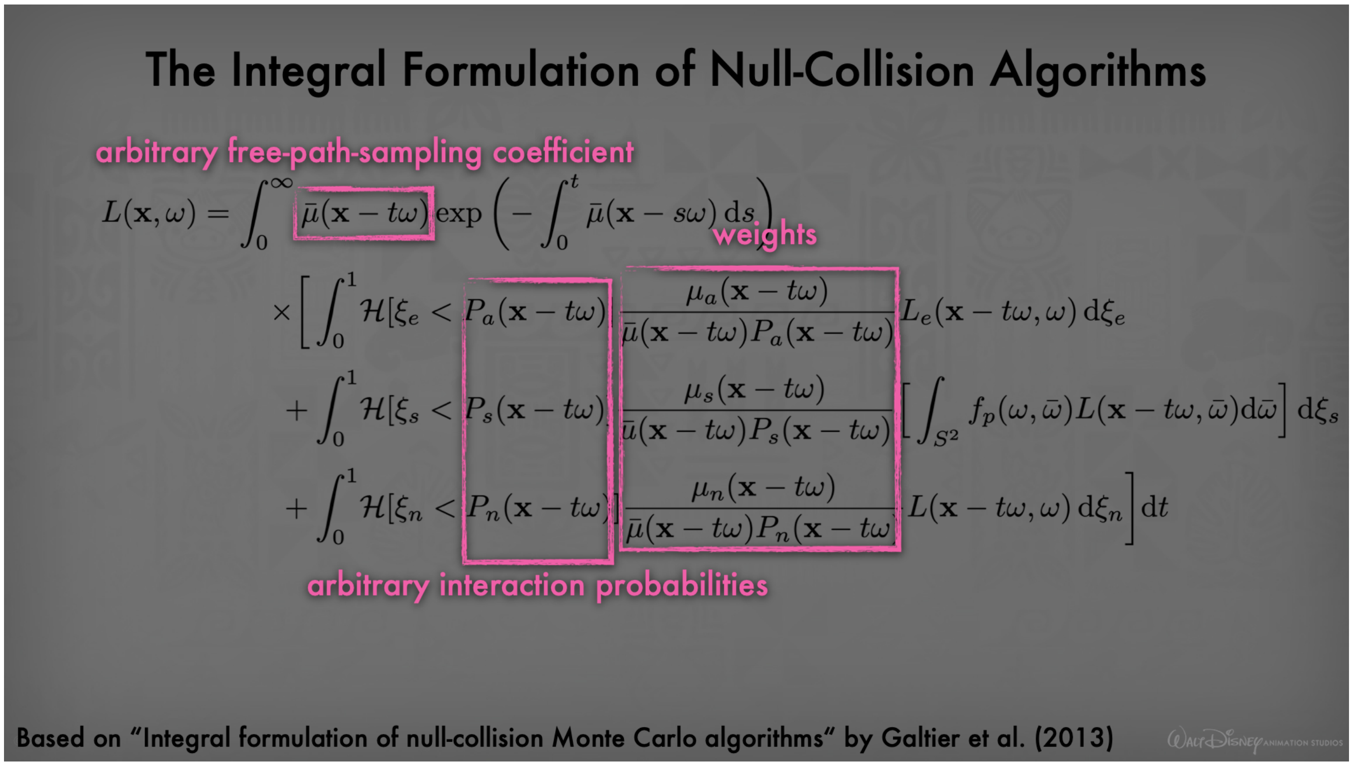


Decomposition tracking let’s us render the exact same image but faster, 18% faster in the case of this cloud. While it can’t do miracles, it can greatly reduce whatever cost is associated with the spatially varying lookups. The more homogeneity we can extract out of the volume, the more lookups we avoid.



Integral Formulation of Null-Collision Algorithms

Now let’s talk about the integral formulation of null-collision algorithms, which was introduced by Galtier and colleagues in 2013, and which encompasses delta tracking, as well as our algorithms and many others. This formulation is derived directly from the radiative transfer equation, a fundamental description of light transport in volumes, and is designed to be translated into code. We’ll look briefly at the full equation first, but don’t worry about the details of the math. The details are in the paper. For now, we just want to show what the overall structure of the equation looks like and use it to make a few points.



The Integral Formulation of Null-Collision Algorithms

arbitrary free-path-sampling coefficient

weights

$$L(\mathbf{x},\omega) = \int_0^\infty \boxed{\bar{\mu}(\mathbf{x}-t\omega)} \exp\left(-\int_0^t \bar{\mu}(\mathbf{x}-s\omega)\,\mathrm{d}s\right)$$

$$\times\left[\int_0^1 \mathcal{H}[\xi_e < \boxed{P_a(\mathbf{x}-t\omega)}] \boxed{\frac{\mu_a(\mathbf{x}-t\omega)}{\bar{\mu}(\mathbf{x}-t\omega)P_a(\mathbf{x}-t\omega)}} L_e(\mathbf{x}-t\omega,\omega)\,\mathrm{d}\xi_e\right.$$

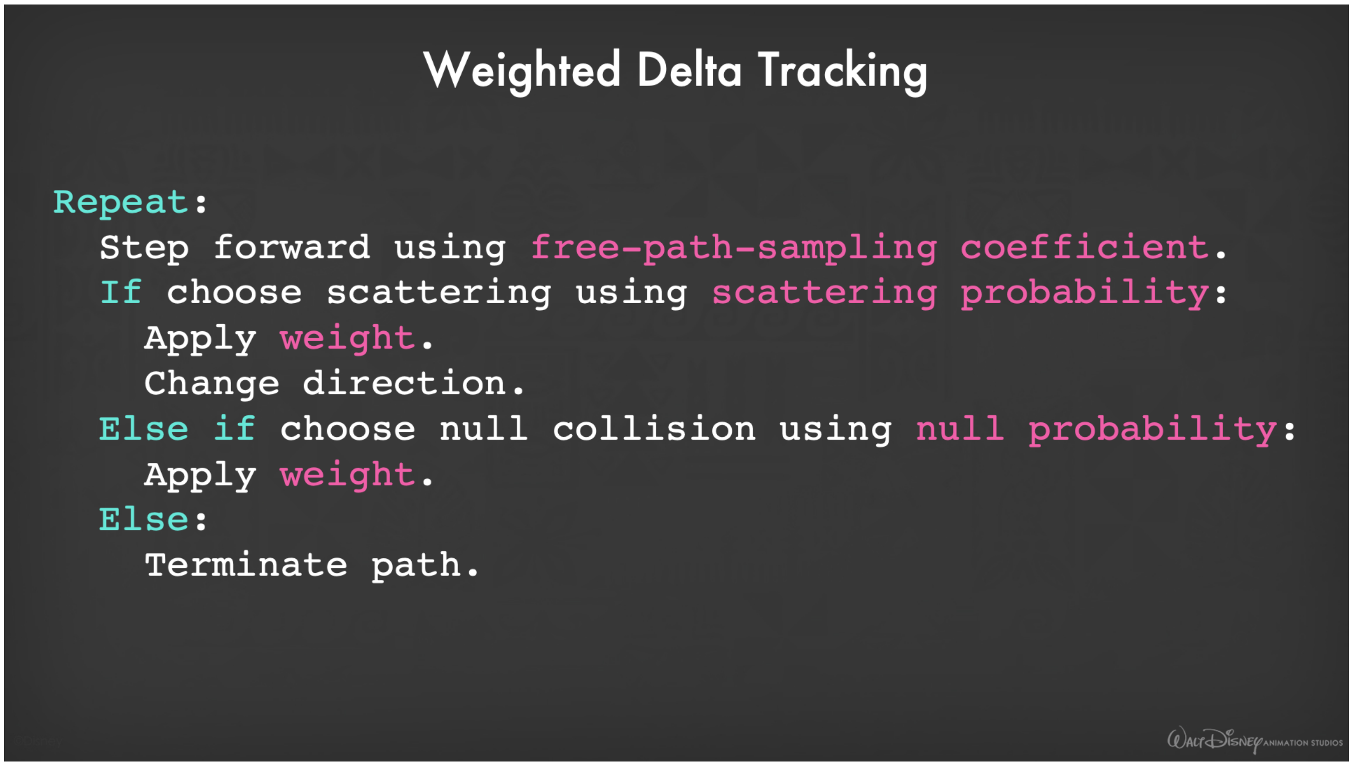
$$+\int_0^1 \mathcal{H}[\xi_s < \boxed{P_s(\mathbf{x}-t\omega)}] \boxed{\frac{\mu_s(\mathbf{x}-t\omega)}{\bar{\mu}(\mathbf{x}-t\omega)P_s(\mathbf{x}-t\omega)}} \left[\int_{S^2} f_p(\omega,\bar{\omega})L(\mathbf{x}-t\omega,\bar{\omega})\mathrm{d}\bar{\omega}\right]\mathrm{d}\xi_s$$

$$\left.+\int_0^1 \mathcal{H}[\xi_n < \boxed{P_n(\mathbf{x}-t\omega)}] \boxed{\frac{\mu_n(\mathbf{x}-t\omega)}{\bar{\mu}(\mathbf{x}-t\omega)P_n(\mathbf{x}-t\omega)}} L(\mathbf{x}-t\omega,\omega)\,\mathrm{d}\xi_n\right]\mathrm{d}t$$

arbitrary interaction probabilities

Based on "Integral formulation of null-collision Monte Carlo algorithms" by Galtier et al. (2013)

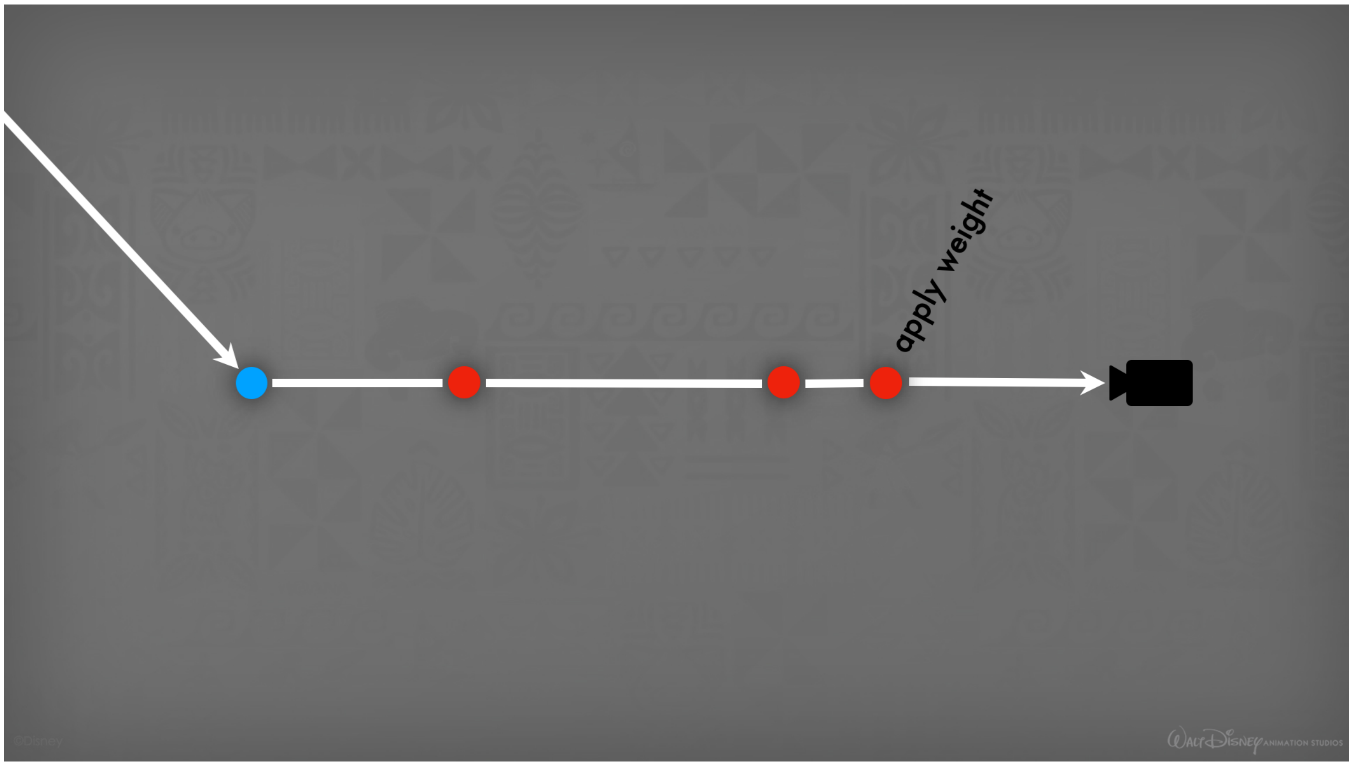
Here it is. I'll point out just a few key parts. This is an arbitrary "free-path-sampling coefficient" that is used for taking one step forward along the ray. These are arbitrary probabilities for selecting absorption, scattering, or null-collision at each step, and these are weights that compensate for these other arbitrary values. These three things lift the restrictions of delta tracking and allow us to do things that don't correspond to the physical process of photon scattering.



Weighted Delta Tracking

```
Repeat:
  Step forward using free-path-sampling coefficient.
  If choose scattering using scattering probability:
    Apply weight.
    Change direction.
  Else if choose null collision using null probability:
    Apply weight.
  Else:
    Terminate path.
```

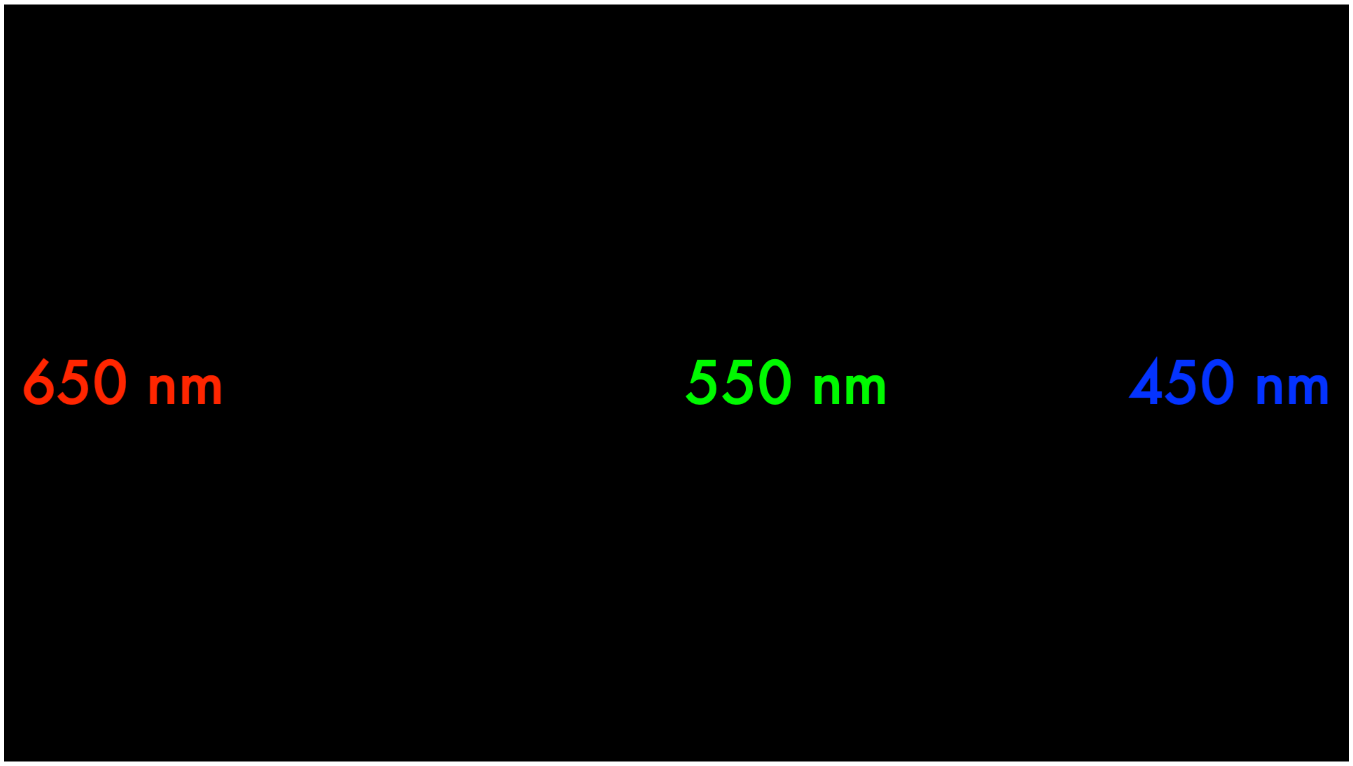
The integral formulation can be translated directly into the code of weighted delta tracking, which is like delta tracking but with arbitrary step sizes, arbitrary probabilities, and weights. At a high level, this code will play a role in the description of spectral tracking.



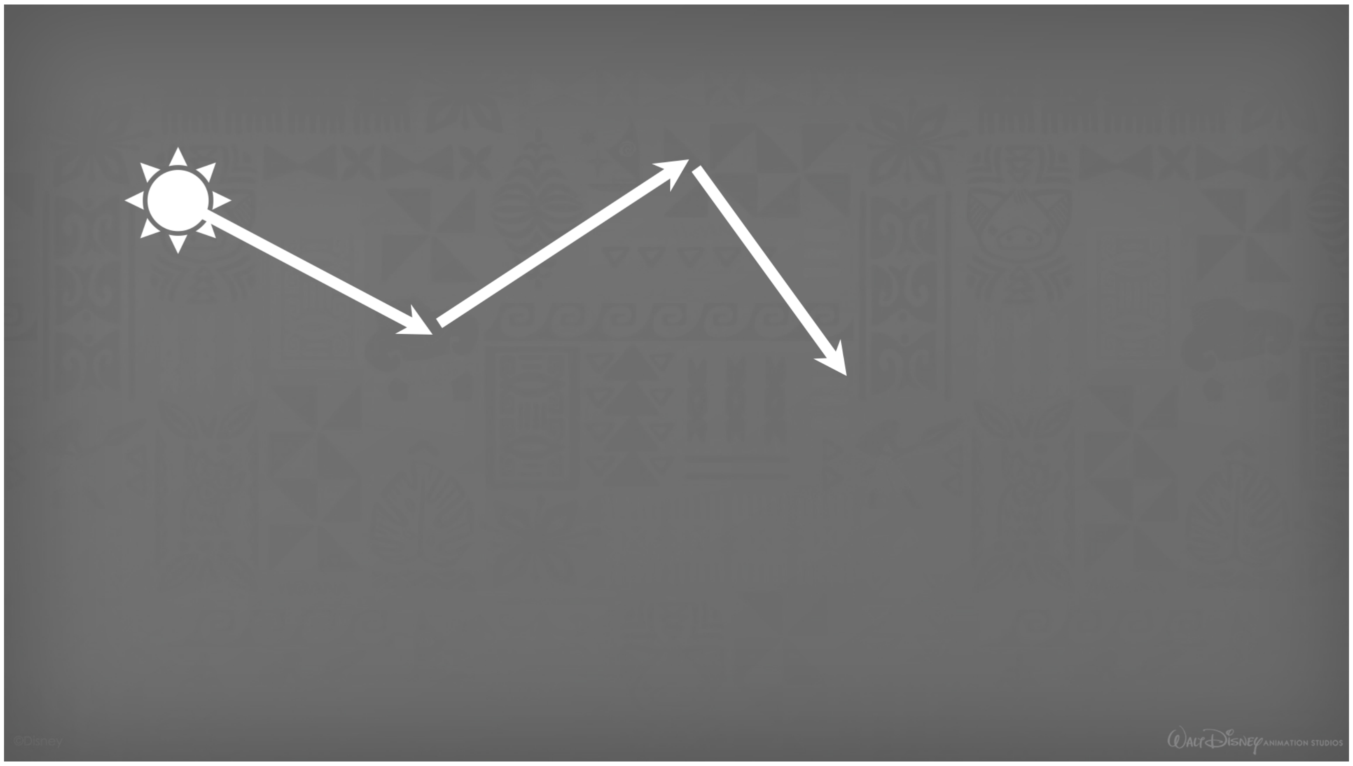
At every step of weighted delta tracking, including the null collisions, we apply a local collision weight to update the path weight, which we call the “throughput”.

Spectral Tracking

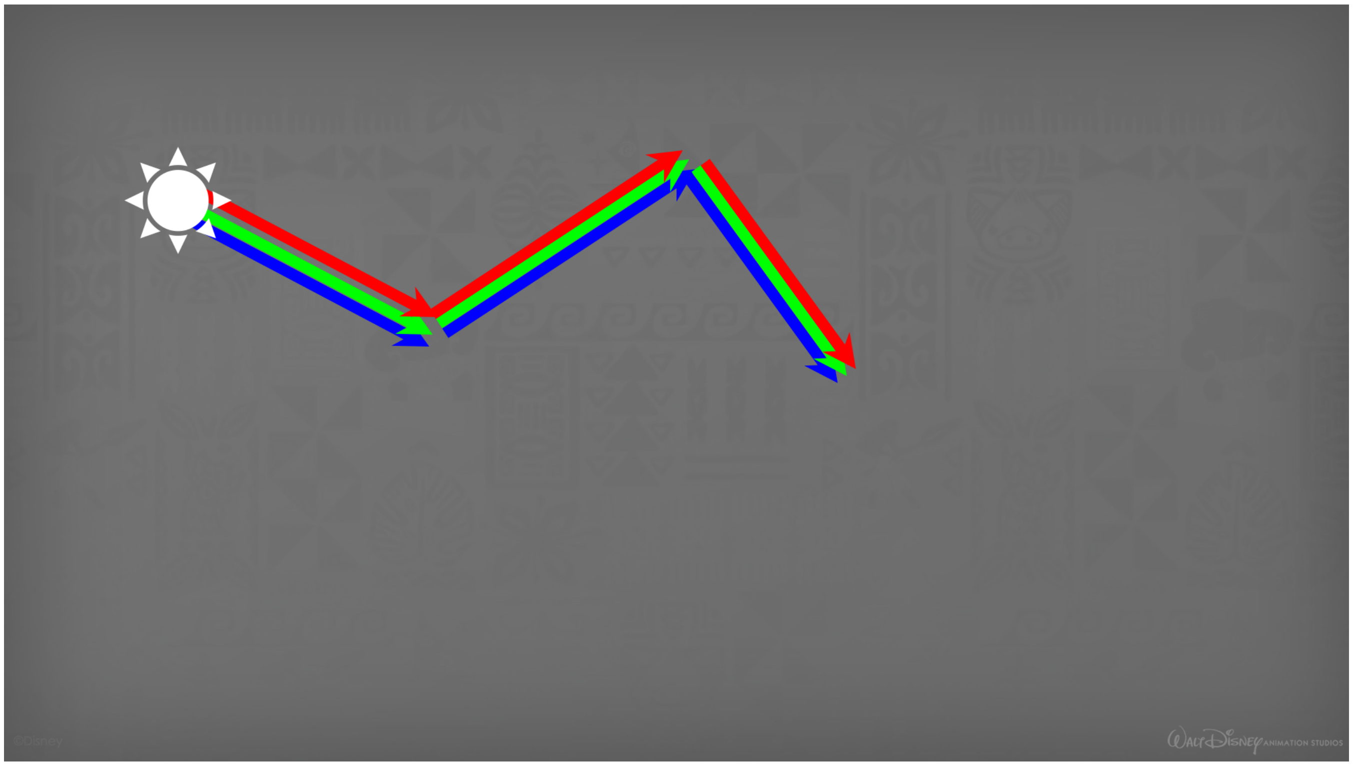
Now let’s look at spectral tracking.



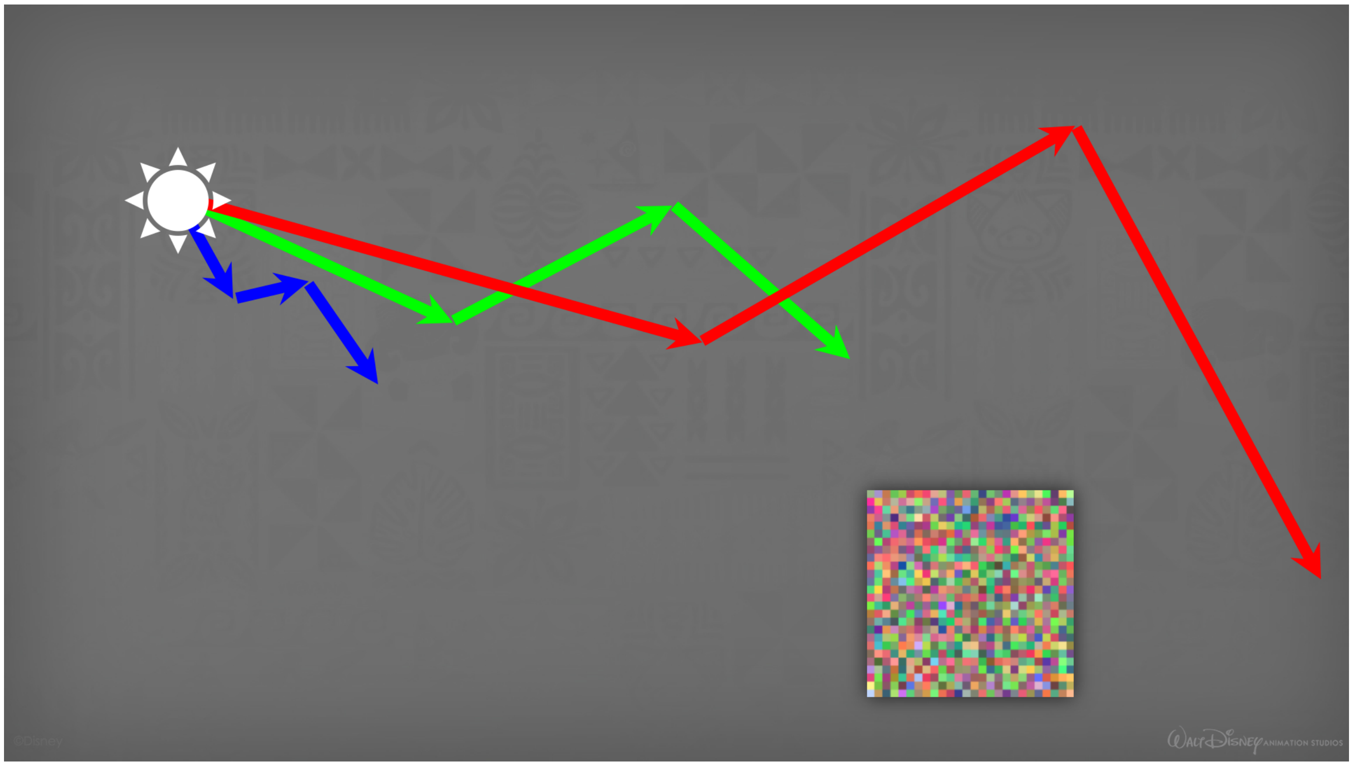
Let’s say we want to render the interaction of multiple different wavelengths of light with a volume that has different properties at each of those wavelengths. For simplicity, we’ll just look at red, green, and blue.



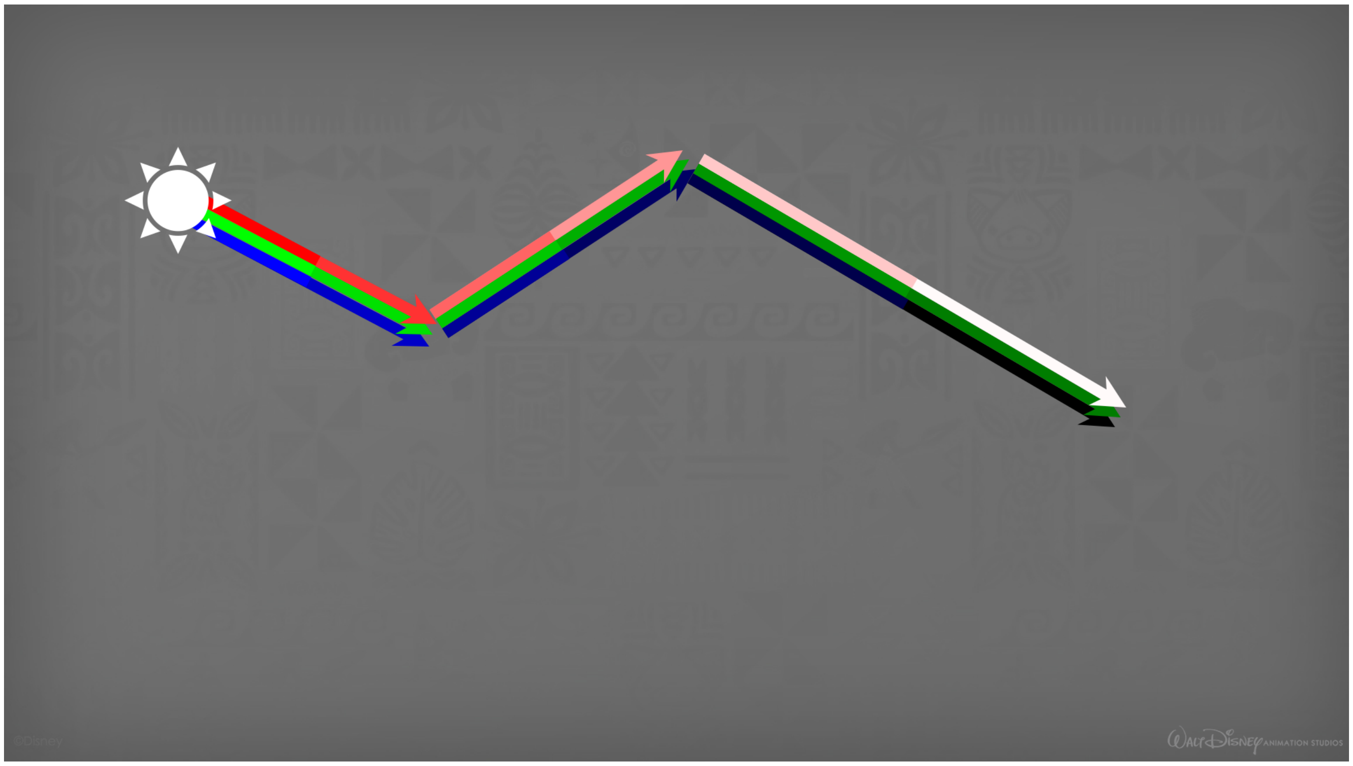
Unfortunately, regular delta tracking can't handle this situation. In regular delta tracking, we scatter around without regard for color.



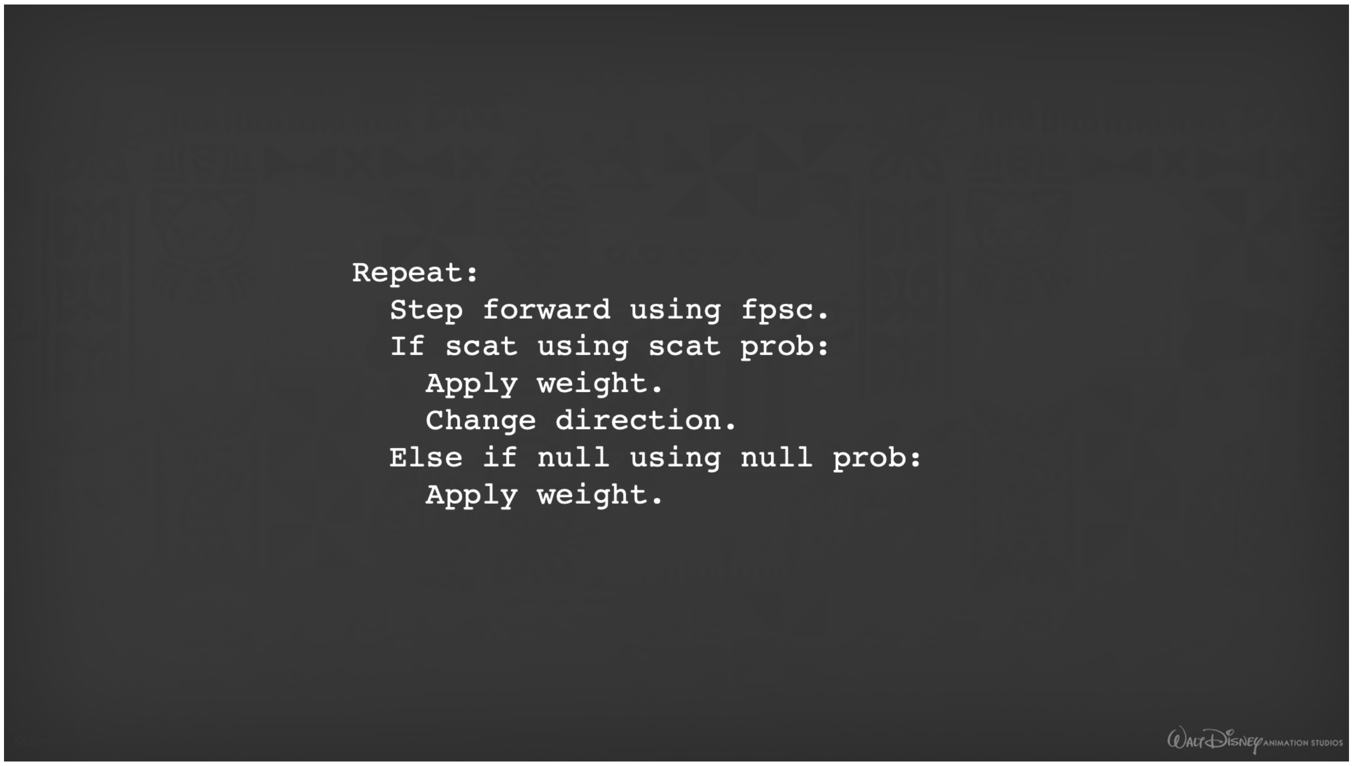
When the volume properties are not wavelength dependent, or when only certain properties are wavelength dependent, we can do regular delta tracking for all three wavelengths simultaneously.



But when the properties are wavelength dependent, we have to do delta tracking separately for each wavelength. In practice this results in more noise, multicolored noise in particular.

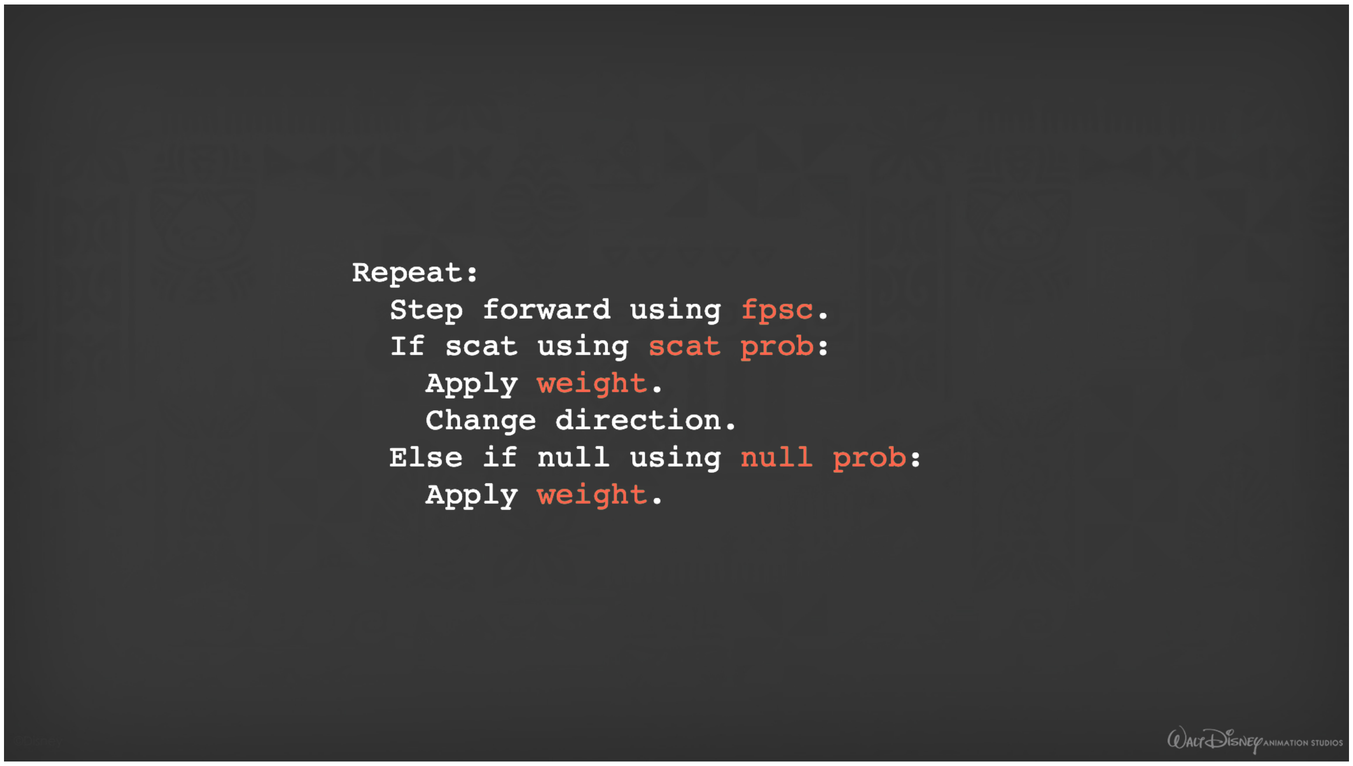


But with spectral tracking, we can trace a single path for multiple wavelengths and compensate with a special weighting scheme.

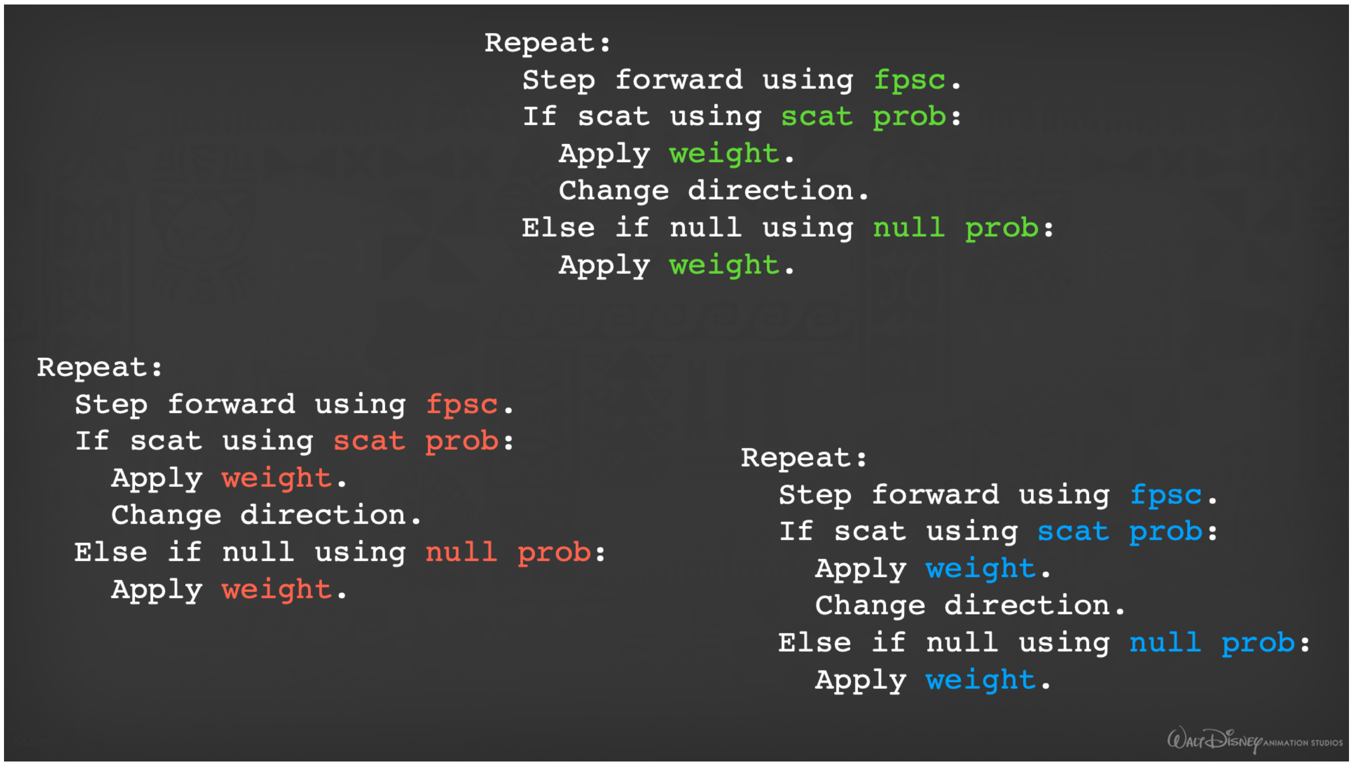


```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

How does spectral tracking work? Here's that weighted-delta-tracking pseudocode again, in an abbreviated form. Again, the basic way to render three different wavelengths is to trace three separate paths.



```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

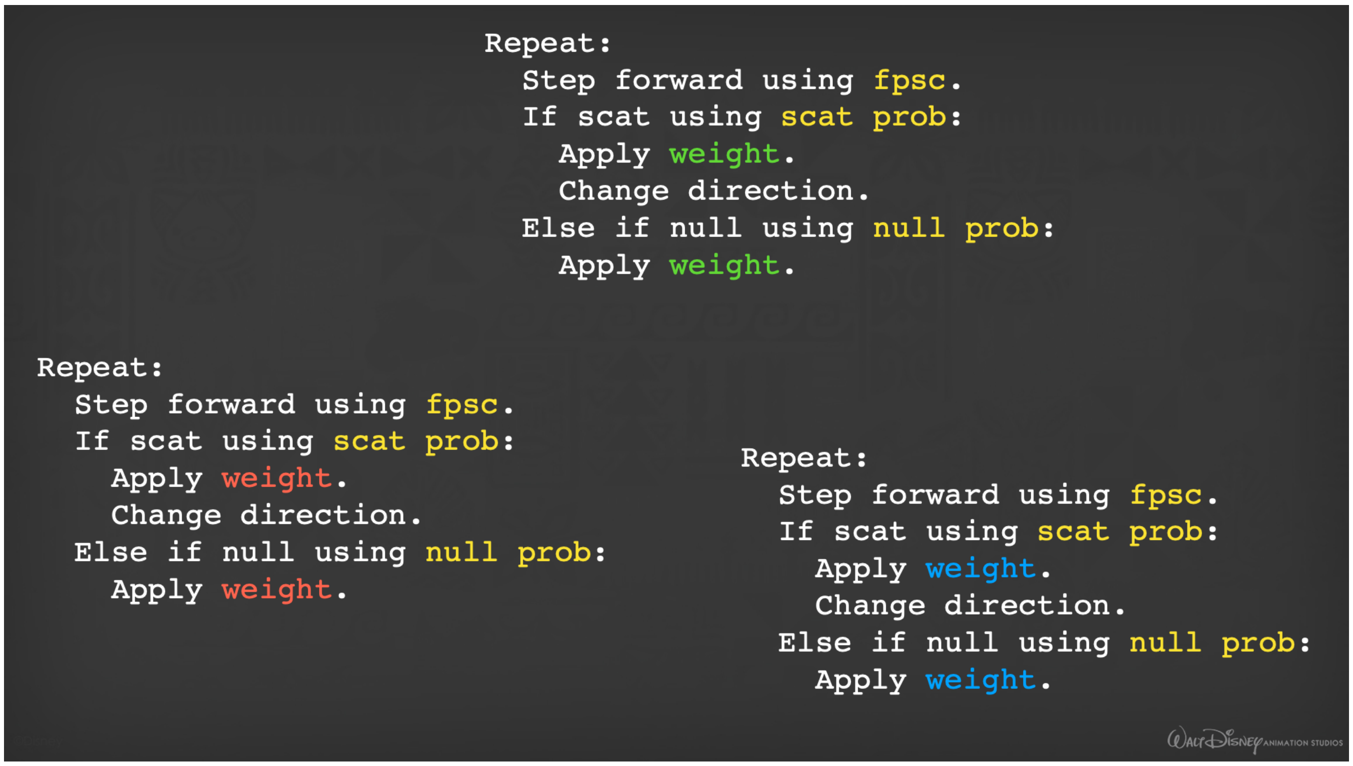


```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

Here you can see three instances of the pseudocode, one for each of the three wavelengths.

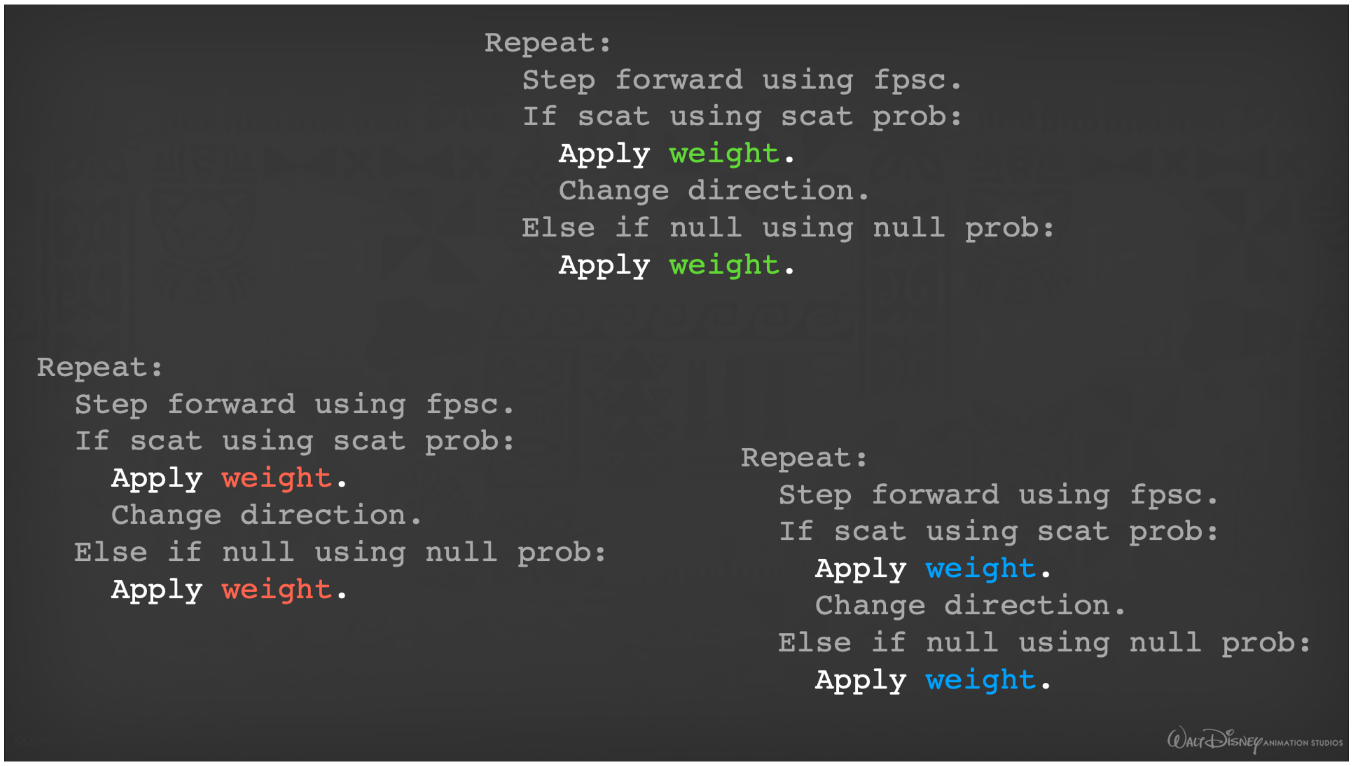


```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

Some of the variables are flexible, so we can set them to the same thing for each of the wavelengths.

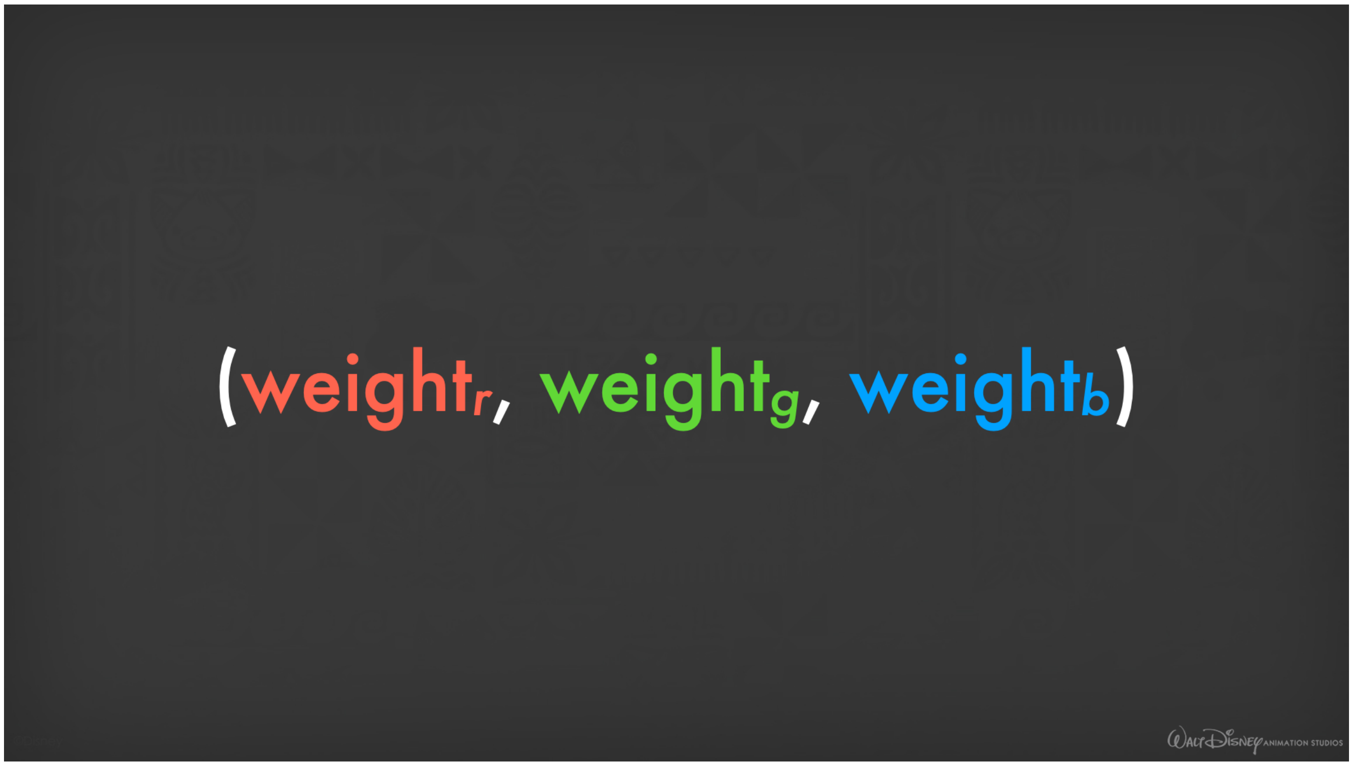


```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

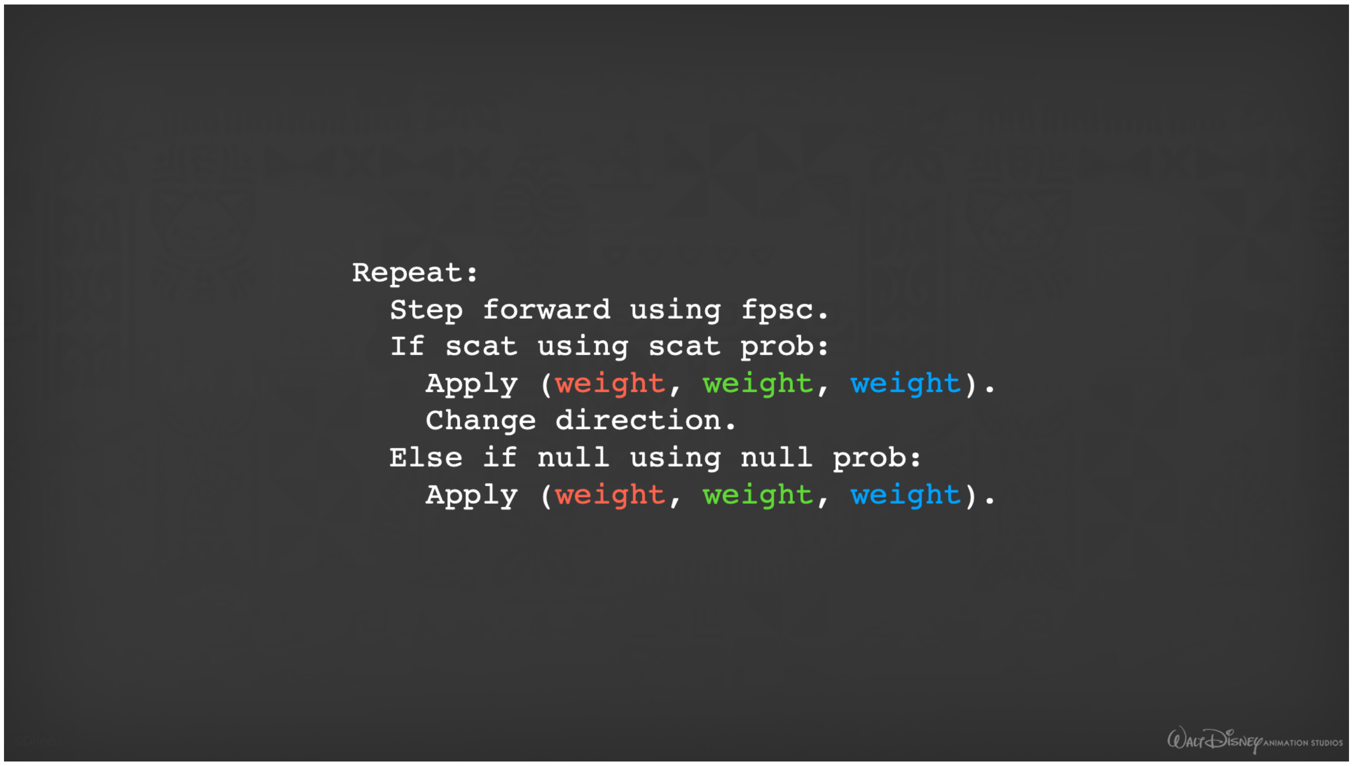
```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply weight.
    Change direction.
  Else if null using null prob:
    Apply weight.
```

The weights are then really the only thing that differ for the three wavelengths.

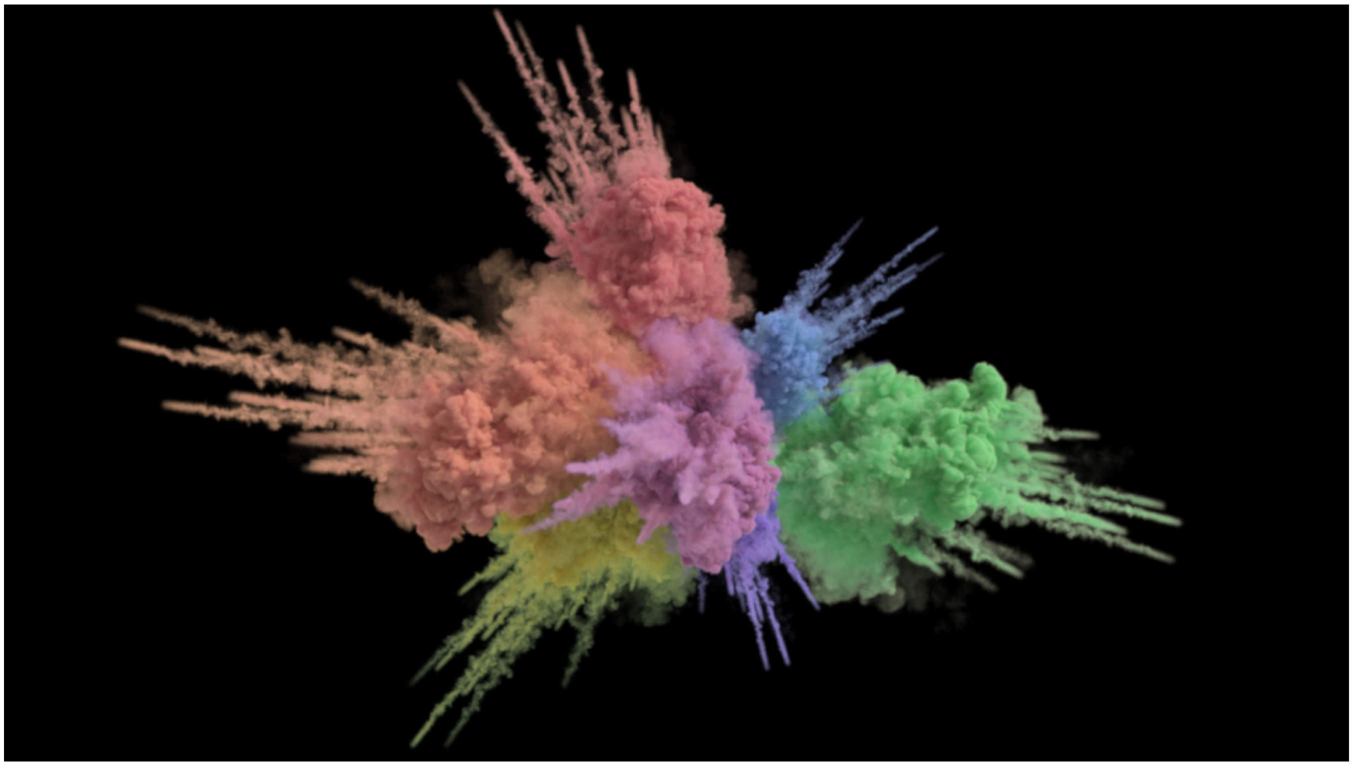

$(\text{weight}_r, \text{weight}_g, \text{weight}_b)$

So in spectral tracking we vectorize the local collision weights and the resultant path throughput.

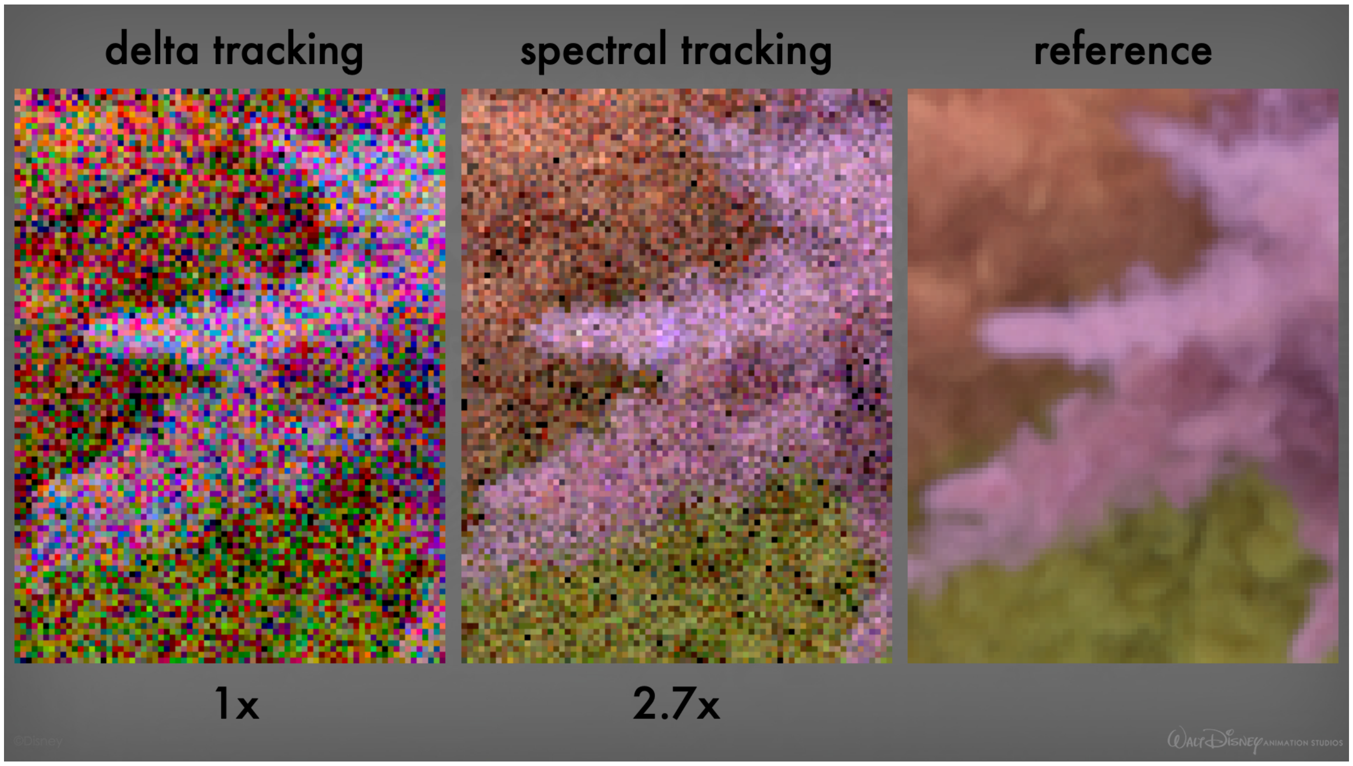


```
Repeat:
  Step forward using fpsc.
  If scat using scat prob:
    Apply (weight, weight, weight).
    Change direction.
  Else if null using null prob:
    Apply (weight, weight, weight).
```

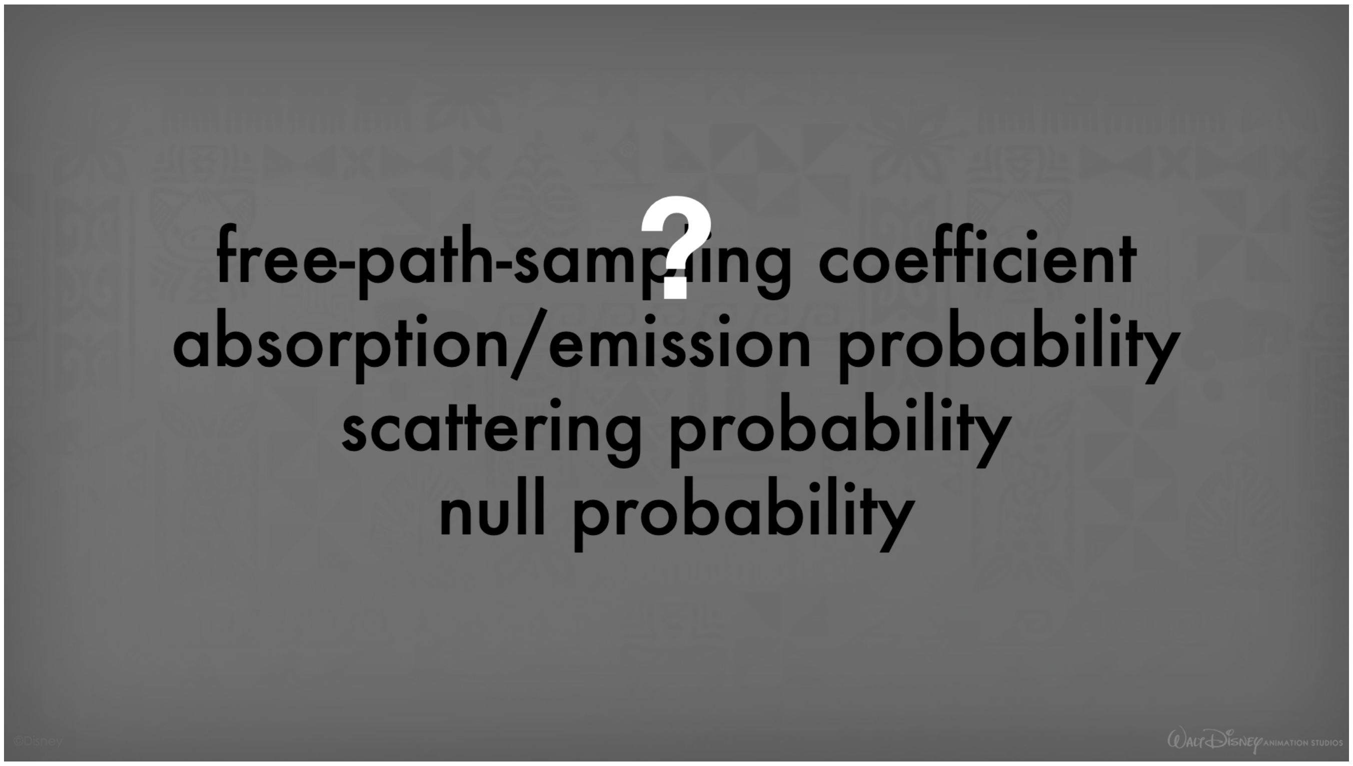
And then trace a single path for all of the wavelengths.



Here’s an example of a volume with wavelength-dependent scattering and absorption.



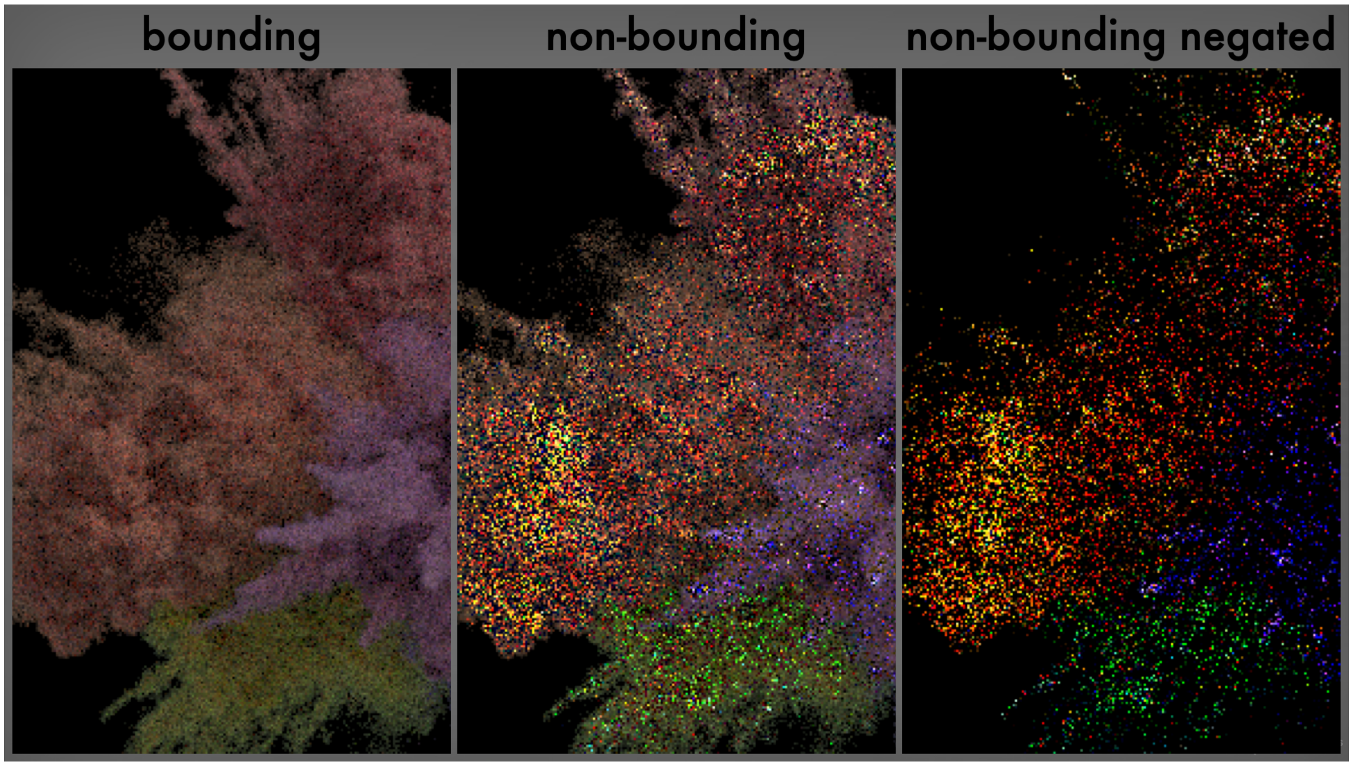
These equal-samples renders show that spectral tracking will achieve the same variance as delta tracking 2.7x faster.



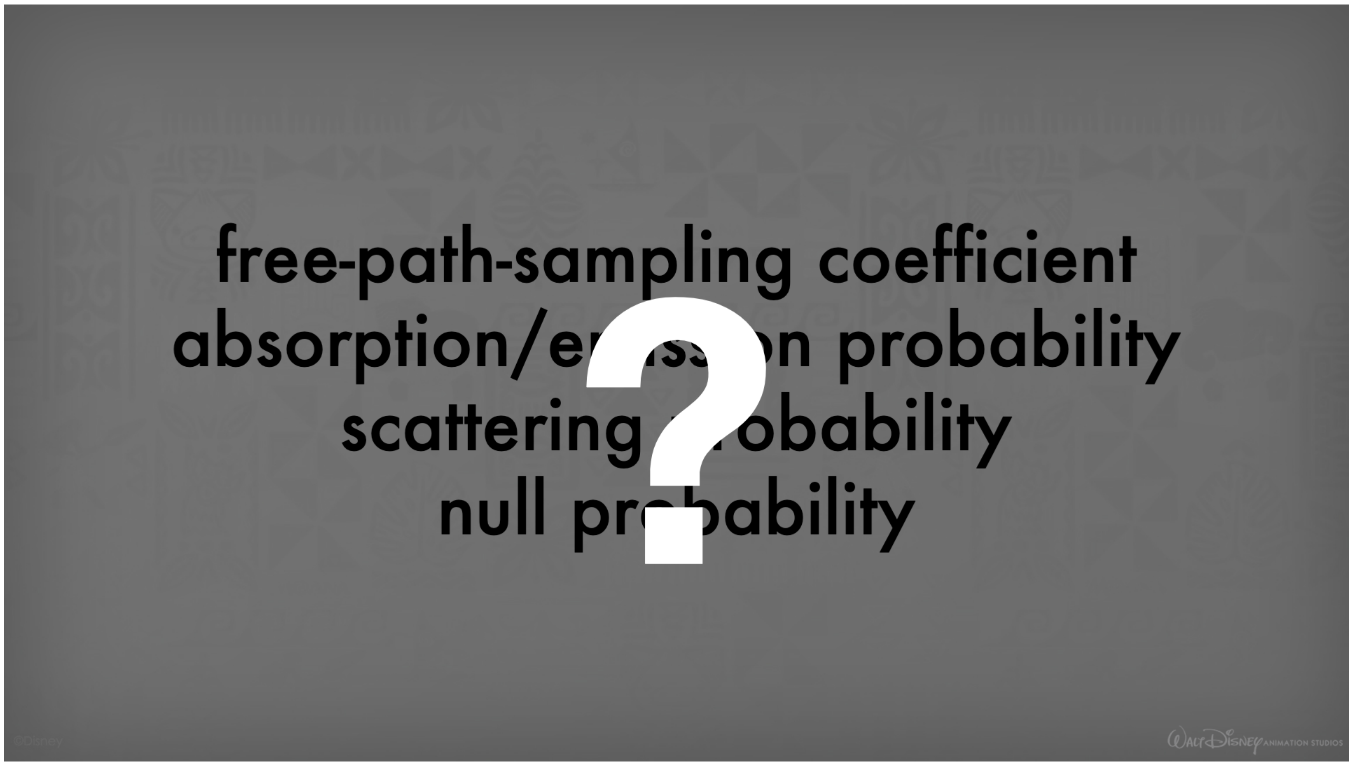
?

free-path-sampling coefficient
absorption/emission probability
scattering probability
null probability

So far we've described spectral tracking conceptually, but the details are important. What do we do with the degrees of freedom inherited from weighted delta tracking? In particular, how do we set these values? Let's start with the free-path-sampling coefficient, which is what determines the exponentially-sampled step sizes.



Well, this can be set to virtually anything, but to avoid exploding and oscillating weights it should be set to bound the maximum value of the extinction coefficient along the ray.

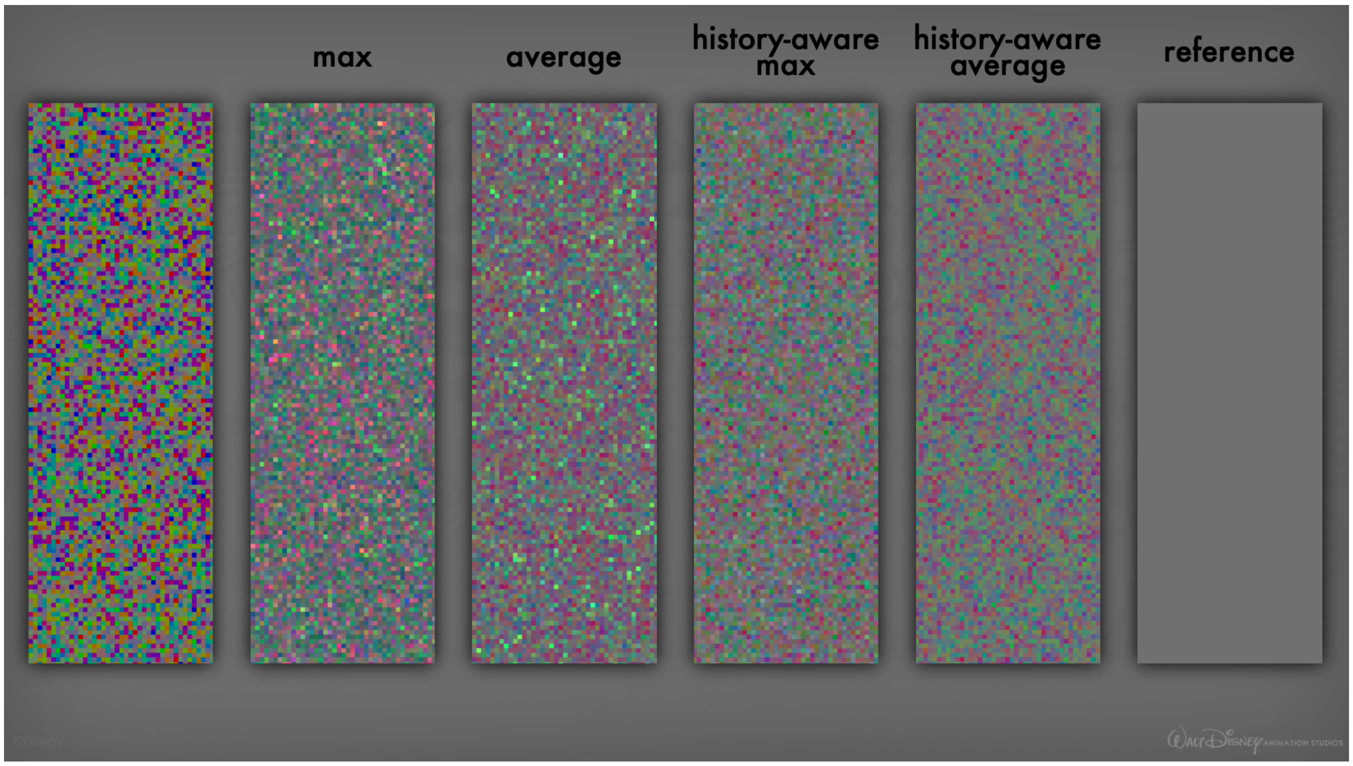

free-path-sampling coefficient
absorption/emission probability
scattering probability
null probability

?

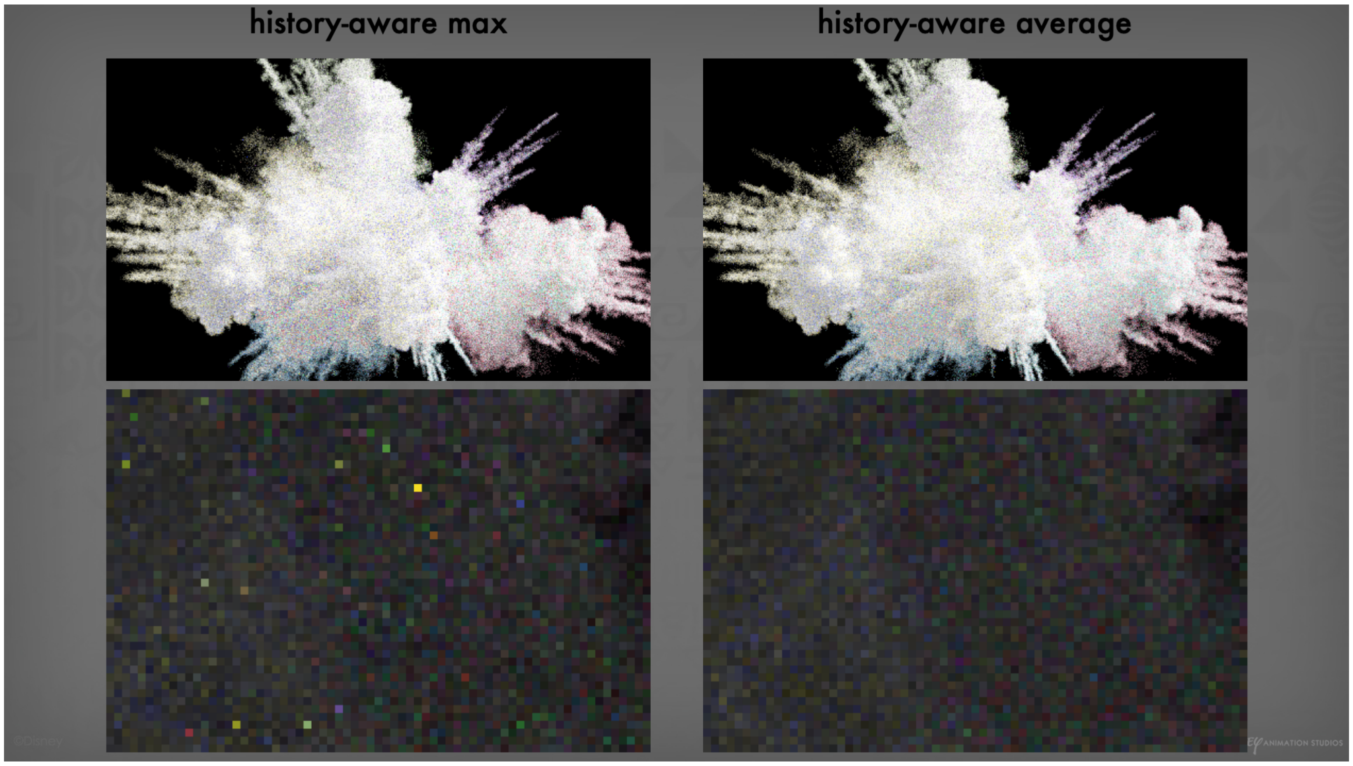
©Disney

Walt Disney Animation Studios

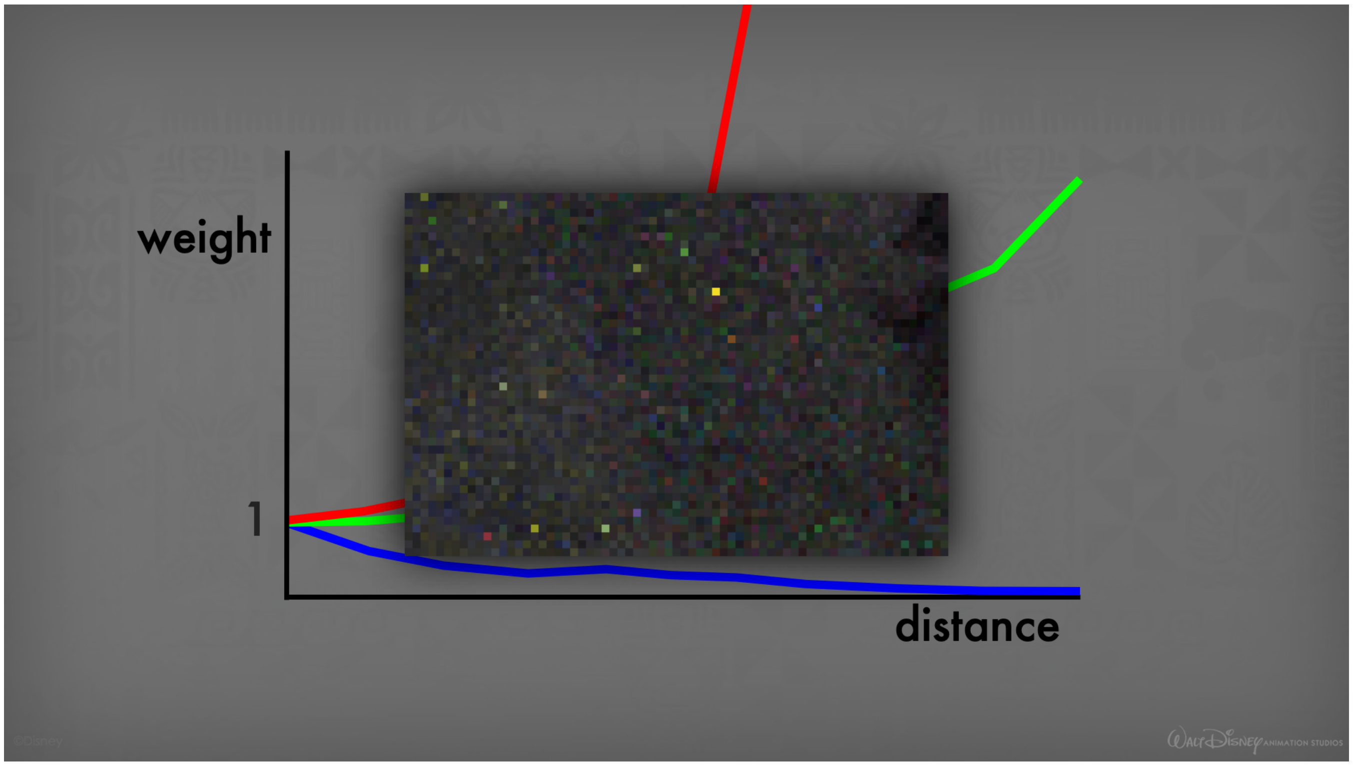
How about the interaction probabilities? This isn't as straightforward.



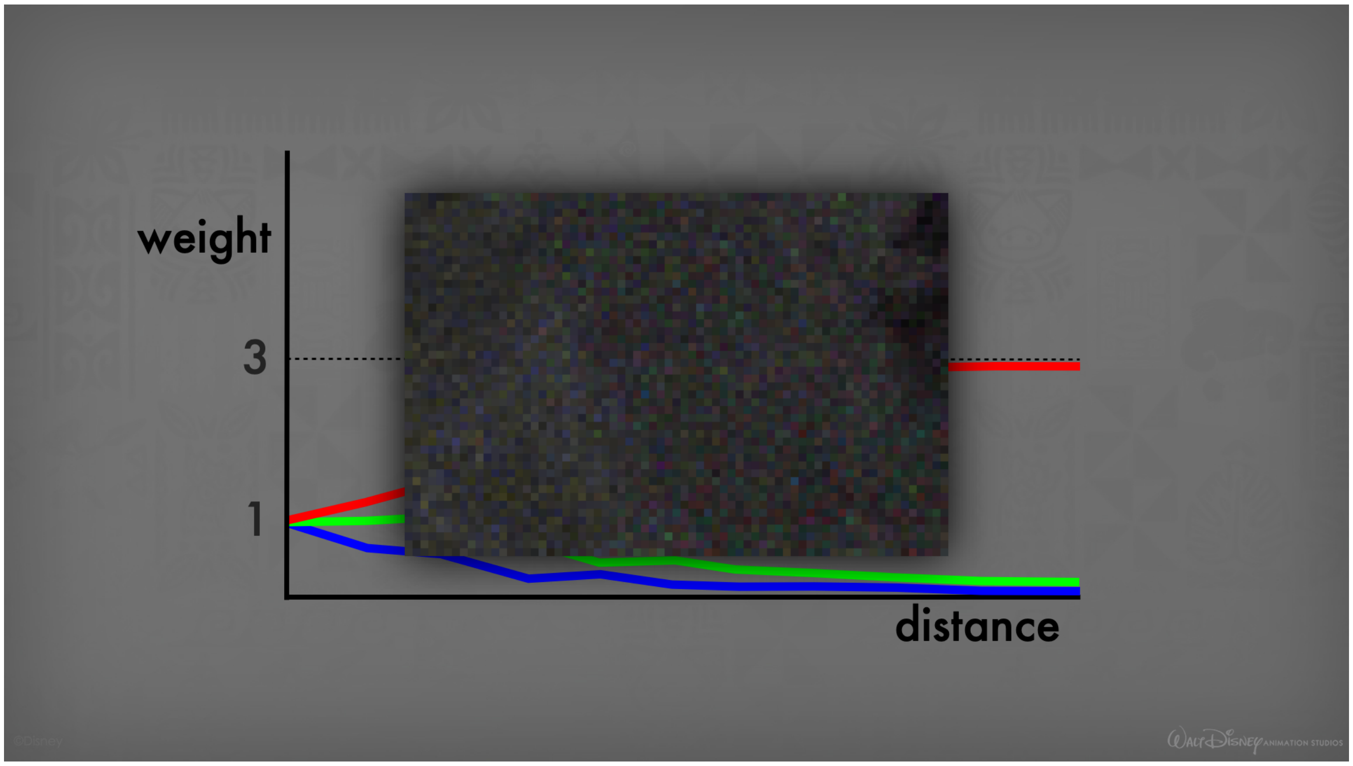
Different probability schemes produce very different results. The four images in the middle were generated using four of the most interesting possibilities that we investigated. You can see that the right two, history-aware average and history-aware max, both look the least noisy and pretty similar.



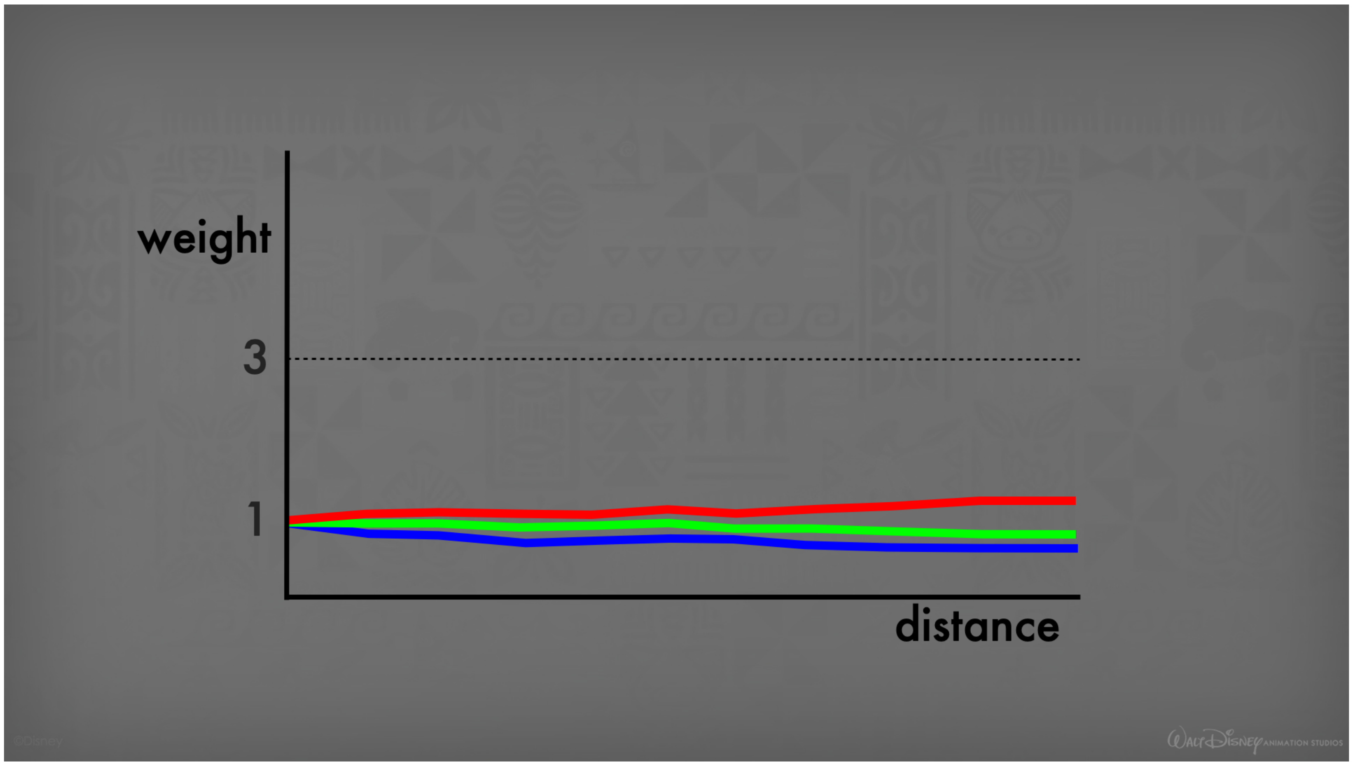
It's only in more extreme cases that it becomes clear which one is more robust. This is a volume with wavelength-dependent scattering and no absorption. All the images in this talk including these were rendered with unlimited multiple scattering. Here you can see that the history-aware-max scheme results in some fireflies whereas the history-aware-average scheme results in none. In fact, this scheme actually puts an upper limit on the throughput equal to the number of wavelengths being traced, which we prove in the paper's supplemental material. Out of the infinite possible probability schemes it's the only one we know of that has this desirable property.



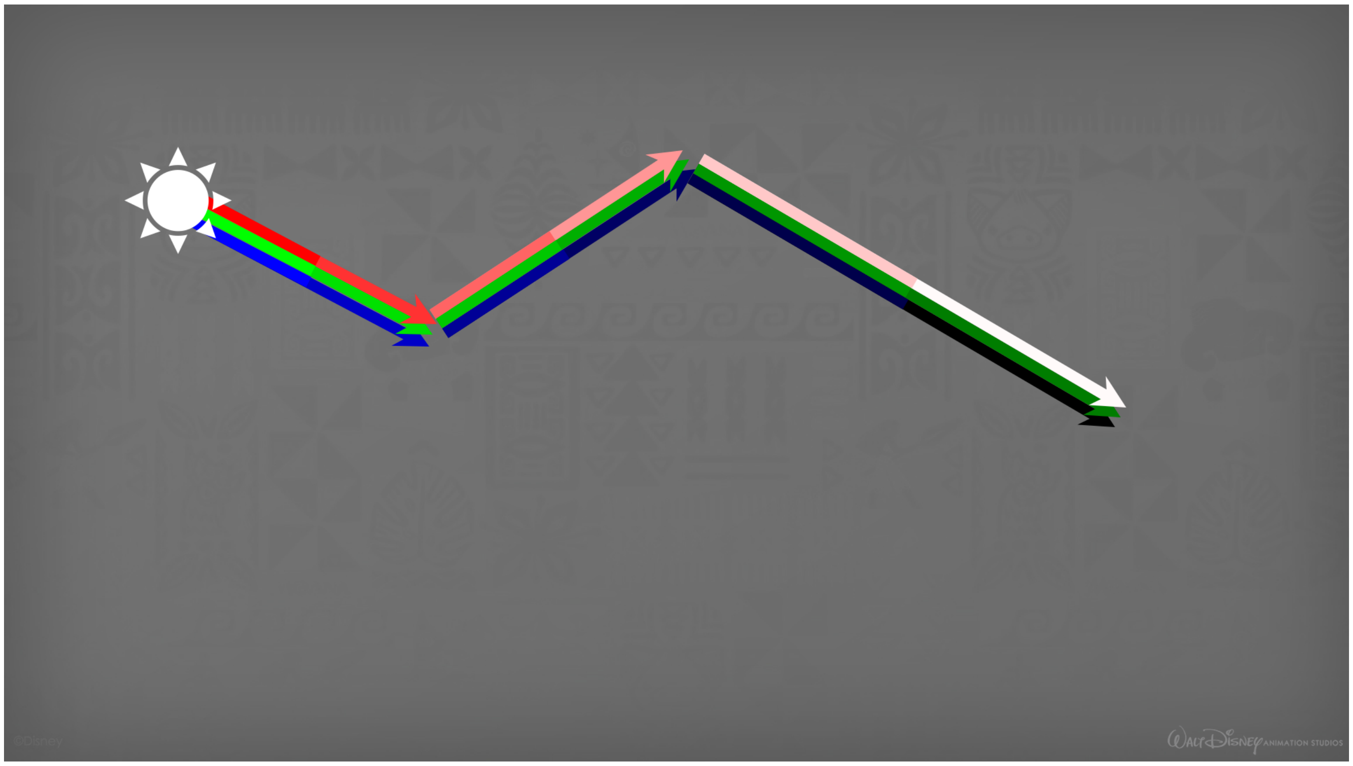
See, in general, since we can't perfectly importance sample all wavelengths simultaneously, the local weight can be above one, and we can experience geometric growth of the path throughput after multiple bounces. This can produce arbitrarily bright fireflies.



But the history-aware-average probabilities prevent this situation.
making them very robust and practical.



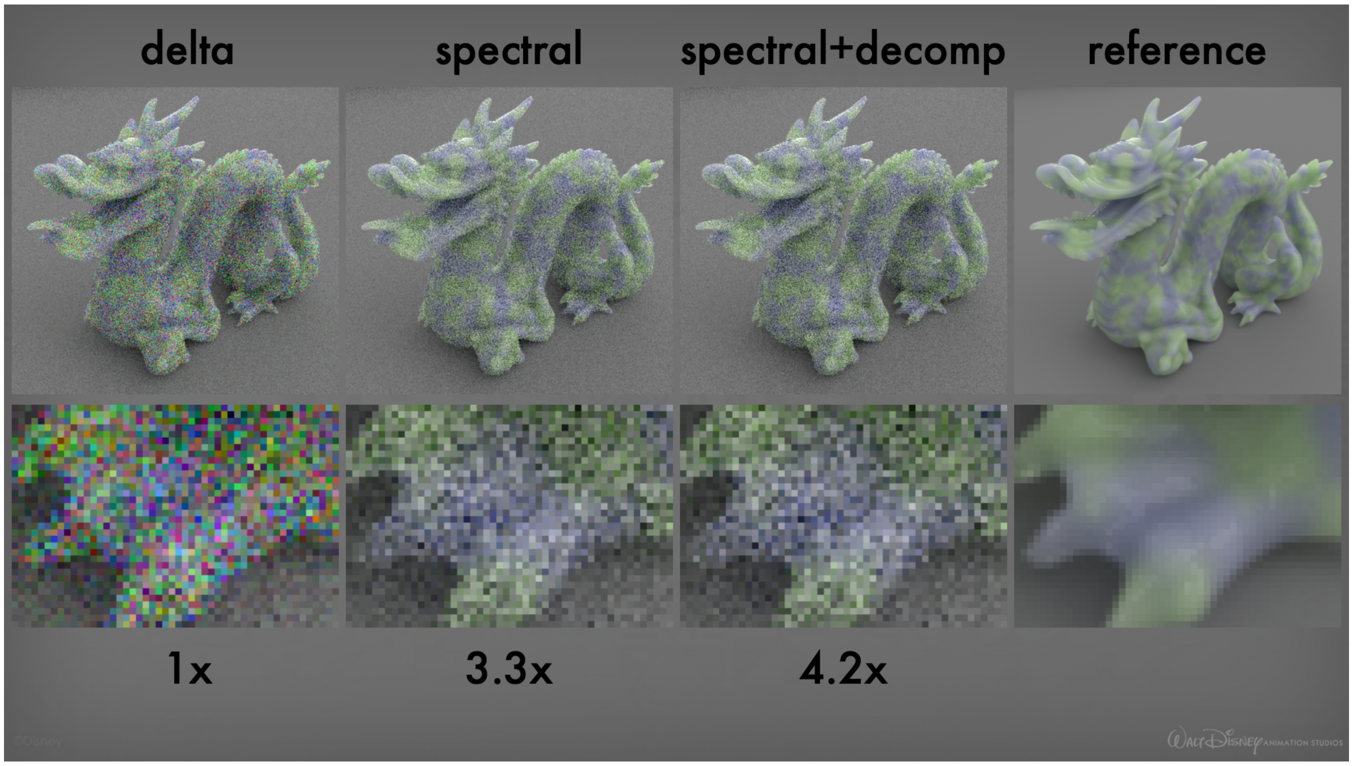
It’s worth noting that in practice, for less extremely chromatic volumes, the throughput for the wavelengths doesn’t diverge as quickly and the path can be shared more equally among the wavelengths.



If we revisit this diagram, we can see that, for this one particular path, the spectral tracking with the history-aware-average probabilities has drifted toward favoring the red wavelength, at which point the path construction is more heavily influenced by the red volume properties, resulting in longer distances between scattering events.

Putting It All Together

The integral formulation can be extended as described by Galtier in 2016 to include multiple components, which allows us to formulate both our decomposition and spectral tracking algorithms in one unifying mathematical framework.



Rendering this procedurally subsurface-scattering dragon with both techniques together produces less noise in less time than delta tracking each color channel separately. We hope that, in the future, others also use the integral formulation to derive other novel and useful algorithms, and to transfer ideas between different fields of study. That concludes this talk. Thanks for your attention.

Thank you for your attention!

The integral formulation can be extended as described by Eymet and colleagues to include multiple components, which allows us to formulate both our decomposition and spectral tracking algorithms in one unifying mathematical framework.

Walt Disney
ANIMATION STUDIOS
©Disney
Walt Disney ANIMATION STUDIOS