Closed Form Transmittance in Heterogeneous Media Using Cosine Noise

Martin Bálint[*]
Supervised by: Rafał Mantiuk[†]

University of Cambridge
Cambridge / United Kingdom

Abstract

We present an analytically integrable noise function with similar computational cost and quality to Perlin's noise. We show how to evaluate transmittance integrals through our proposed noise function in a closed form. Such evaluation requires only two samples of our noise function. In contrast, previous methods require a number of samples that is proportional to the resolution of the noise to evaluate a transmittance integral. We also propose a distance importance sampling method for our noise function, which avoids the limitations of delta tracking. We compare our method to delta tracking. As the resolution of the noise increases exponentially with respect to the number of octaves of noise, our method becomes much faster. Additionally, with distance importance sampling, the probability density function of the samples can be calculated analytically, allowing for fast multiple importance sampling. We also discuss the limitations of our method regarding the shape of participating media and provide alternative approaches to overcome these limitations. Finally, we propose approximated solutions that allow for the use of our method in real-time applications at the cost of small bias.

Keywords: closed form, analytically integrable, heterogeneous media, transmittance, cosine noise, rendering

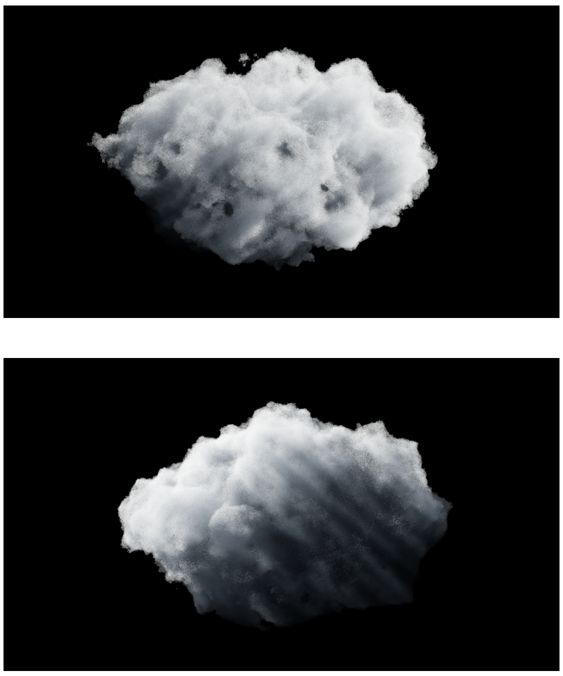


Figure 1: Cloud rendered using cosine noise. The scene is illuminated by a directional light source. Both renders took 0.8 seconds. Resolution: 450 by 800 pixels. Effective noise resolution of about 2000^3.

1 Introduction

In nature, many natural volumetric phenomena such as clouds, smoke and fog have small variations in density which need to be reproduced by rendering algorithms to give these phenomena a realistic look. Such variations are generally achieved by combining simulated volumes with procedural noise [2]. However, rendering participating media with procedural noise of such high-resolution is extremely expensive computationally as the time complexity of transmittance evaluations increases exponentially with respect to the number of octaves of noise. By octaves, we refer to the number of iterations or frequencies of noise used to produce fractal noise.

First, we will show how previous approaches fail to avoid this exponential complexity. Then we will show how we achieved linear complexity using an analytically integrable noise function and give an algorithm for distance importance sampling as well as compare our method to the widely used delta tracking. We will then show the limitations of our technique and detail alternative approaches, which can be used to avoid these limitations. Finally, we will give a short description of some optimisations, which let us use our method in real-time applications.

[*] mb2280@cam.ac.uk
[†] rafal.mantiuk@cl.cam.ac.uk

2 Related work

The radiance of the light scattered in a volume can be expressed with the rendering equation

$$L_v(x,\vec{w}) = \int_0^S T_r(x,x_t)L_s(x_t,-\vec{w})dt \qquad (1)$$

where L_v is the volumetric radiance seen from position x in direction $\vec{w}$, T_r is the transfer function and L_s is the radiance scattered out from position x_t in the direction $-\vec{w}$.

A simple method for the evaluation of Equation 1 is classical ray marching [9] along the ray with a fixed stepsize to evaluate the integrals. However as the lighting equation must be evaluated recursively, the time complexity of the algorithm is $O(N^B)$ where N is the average number of steps along a ray, and B is the number of bounces. Unfortunately as the resolution of the volume is increased with additional octaves of noise, N must be increased accordingly. The step size must be set smaller than the average of distances between local extrema of the noise function to avoid artefacts. Therefore, $N = O(a^M)$ where a is constant depending on the desired quality and the scaling factor between octaves of noise and M is the number of octaves of noise. Therefore, the time complexity of the lighting calculation is $O(a^{B\cdot M})$, which is unfavourable for rendering volumes with a high-resolution noise or many bounces.

Stochastic methods such as distance sampling are effective when used in homogenous volumes, but require ray marching in heterogenous volumes. Only one ray is marched for each bounce in each sampling iteration resulting in a time complexity of $O(B\cdot a^M)$. Although additional noise is introduced, the amount of computation time is distributed equally between bounces. Using the delta tracking algorithm, also called Woodcock tracking, [15, 10] the time complexity of distance importance sampling can be reduced to $O(B\cdot M)$. While delta tracking requires an accurate upper bound on the extinction coefficient, it is a fast method for distance importance sampling. However, when used for transmittance evaluations, it only gives stochastic results. Some methods can be adapted to use stochastic transmittance values, however, special care must be taken to ensure that the distribution of these transmittance values maps correctly when transformed non-linearly. With more specialised methods such as multiple importance sampling or artistic transmittance functions, the adaptation to stochastic transmittance values can be too complex or impossible for practical implementations. In these cases, an exact transmittance evaluation is needed, and with stochastic methods, an accurate transmittance evaluation requires many samples. Although techniques such as ratio tracking, residual tracking [7], free path sampling [14] and negative extinction [13] can significantly speed up stochastic transmittance evaluation, the associated computational cost is still much higher than with analytic methods. A comparison of ray marching and delta tracking is shown in figure 2.

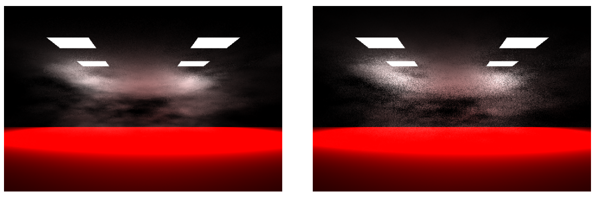

(a) Ray marching (b) Delta tracking

Figure 2: Comparison of ray marching and delta tracking from [10]. Delta tracking produces a noisier image within the same computation time.

Some caching methods such as decoupled ray marching [6] use binary search to importance sample the volume density wise, resulting in a time complexity of $O(B\cdot\log(a^M)) = O(B\cdot M)$ with no additional noise. However, this approach requires the storage and initialisation of cached data and acceleration structures. The exponential time complexity with respect to the resolution at the initialisation diminishes the advantages of fast samples for high resolutions in interactive applications, and the high memory usage makes most caching algorithms difficult to implement on GPUs.

Previous analytical methods mostly focus on rendering homogenous volumes and usually ignore shadowing. [8] Analytical integration of the transmission function has been explored recently in [5]. However, only exponential and spherical density functions were used. In physics, plane waves are commonly used and are analogous to the building blocks of cosine noise. Although linearly transformed cosine functions have been proposed to integrate BRDFs over polygonal domains analytically [3], analytically integrable noise non-piecewise functions have not been explored before. Many noise functions are piecewise integrable, however, as the number of sub-functions is linear to the frequency of the noise, their computational cost is similar to numerical methods.

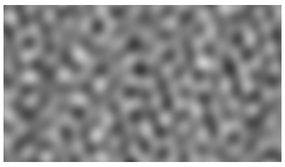
3 Noise definition

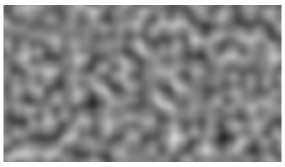
We propose a noise function defined as the weighted sum of longitudinal cosine functions along several vectors. The noise is described by the scalar field ϕ:

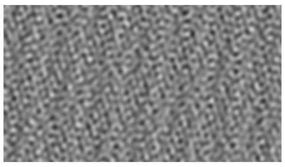
$$\phi(x) = c_{\text{offset}} + \sum_i \cos(x\cdot\vec{S}_i), \qquad (2)$$

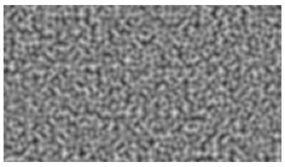
where S_i is the collection of vectors defining the scale and direction of the longitudinal cosines, c_{offset} is a constant defining the constant density of the noise and $x\cdot\vec{S}_i$ is the dot product of a position x and the ith vector of S. Although S_i and x can have arbitrary many dimensions, for rendering participating media, we will be using three dimensional vectors.

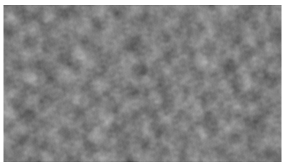
From now on we will refer to this proposed noise function as cosine noise. We have found that to achieve an even frequency, the vectors in S_i should be roughly the same length. To reduce the repetitiveness of the pattern, we used 24 quasirandom vectors. We have found vectors generated by distributing a two dimensional Sobol sequence [12] on a unit sphere and multiplying the results with uniformly distributed random values between 0.8 and 1.2, to be reasonably close to Perlin noise.

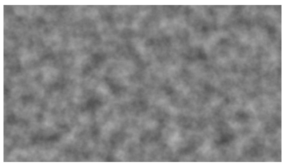

(a) Cosine noise

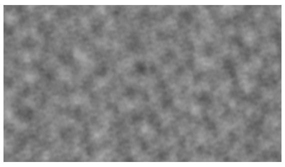

(b) Perlin noise


(c) Cosine noise over large area


(d) Perlin noise over large area


(e) Fractal cosine noise


(f) Fractal Perlin noise


(g) 9 octaves of cosine noise, 256 spp, 36.5 ms

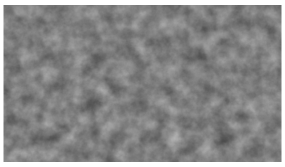

(h) 9 octaves of Perlin noise, 256 spp, 71.3 ms

Figure 3: Quality and performance comparison of cosine and Perlin noises.

As seen in figure 3c, cosine noise is repetitive when viewed over large areas, even with 24 vectors. However, this issue is much less noticeable when the noise is stacked to create fractal noise, as lower octaves mask the repetition. Also, the periodic area can be increased by adding more vectors to S.

As seen in figure 3g and figure 3h, cosine noise has similar computational cost to Perlin noise.

4 Integral, fractal noise

Calculating transmittance along the ray between points using previous methods such as ray marching or delta tracking takes up the vast majority of the computational time required by the rendering of participating media. The commonly used exponential transmittance function:

$$T_r(x,x_t) = \exp\left(-\int_x^{x_t} \sigma_t(z)dz\right), \tag{3}$$

where $\sigma_t(x)$ is the extinction at a given position x. To evaluate the exponential transmittance function, we need the integral of the extinction coefficient along the ray. The extinction coefficient is generally directly proportional to the value of the noise at a given position. If σ_t is the ratio between the noise and the extinction coefficient we get:

$$\sigma_t(x) = \sigma_t \cdot \phi(x). \tag{4}$$

Therefore when evaluating the exponential transmittance function along a ray starting from point x, with direction $\vec{w}$ we get:

$$x_t = x + t \cdot \vec{w} \tag{5}$$

$$T_r(x,x_t) = \exp\left(-\sigma_t \int_x^{x_t} \phi(z)dz\right), \tag{6}$$

$\int \phi(z)dz$ is an integral we can calculate analytically. The indefinite integral along a ray from position x with direction $\vec{w}$ is:

$$\int \phi(x,\vec{w},t) = c_{\text{offset}} \cdot t + \sum_i \frac{\sin((x+\vec{w}t)\cdot\vec{S}_i)}{\vec{w}\cdot\vec{S}_i}. \tag{7}$$

Equation 7 can be evaluated quickly on GPU's by grouping the vectors in S into a collection of three by three matrices S'. Algorithm 1 takes advantage of GPU's linear algebra and trigonometry hardware.

Algorithm 1 Cosine noise integral evaluator

```
 1: procedure EvalInt($\vec{w}, x, t$)
 2:     $p \leftarrow x + \vec{w}t$
 3:     $L \leftarrow 0$
 4:     for $i$ in $S'$ do
 5:         $p' \leftarrow pS'_i$
 6:         $\vec{w}' \leftarrow \vec{w}S'_i$
 7:         $L \leftarrow L + (1,1,1) \cdot \frac{\sin p'}{\vec{w}'}$
 8:     end for
 9:     return $L$
10: end procedure
```

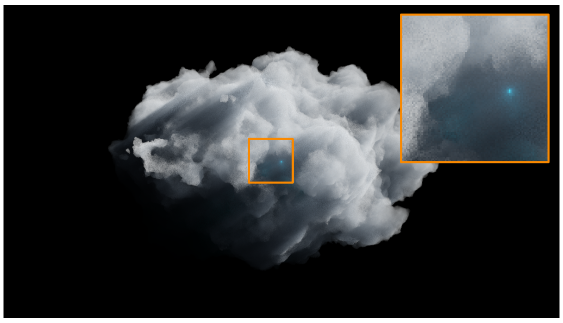
5 Density importance sampling

The transmittance function can be easily importance sampled. We can do this in two parts. In the first part, we will need to importance sample the transmittance function to get a target for the density integral, and in the second part, we will calculate the distance along the ray to get this density integral. First, we scale the indefinite integral of the transfer function so that its codomain is $[0,1]$. For exponential transmittance functions, we need to multiply by σ_t. Then we need to invert this scaled integral and evaluate it for a uniformly random variable in the range $(0,1)$. The resulting number is the target density integral. Now we can run a binary search on the definite integral of the cosine noise along the ray to find the distance along the ray which corresponds to the target density integral. The PDF at this sampling point is the evaluated transmittance function multiplied by the scalar used before as well as the density at the sampling point. This process is described by Algorithm 2.

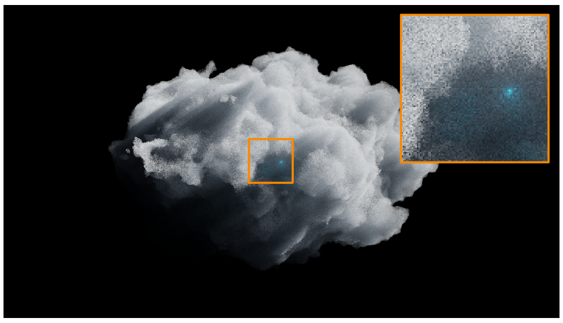
Algorithm 2 Binary search importance sampling

```
1: procedure DistanceSample($\vec{w}, x$)
2:     $\sigma_t \leftarrow (\text{Absorption} + \text{Scattering}) \cdot c_{\text{offset}}$
3:     $\zeta \leftarrow \text{rand}()$
4:     Target Density Integral $\leftarrow \frac{-\ln(1-\zeta)}{\sigma_t}$
5:     $t \leftarrow$ BinSearch(Bounds, Target Density Integral)
6:     PDF $\leftarrow \sigma_t \exp(-\text{Target Density Integral } \sigma_t)\phi(x_t)$
7: end procedure
```

Having an analytical PDF allows for efficient multiple importance sampling, for example, combining density-wise samples with equiangular samples [6].



(a) Closed form evaluation



(b) Delta tracking

Figure 4: MIS Cloud scene. Both renders took 5.5 seconds at a resolution of 675 by 1200 pixels. Effective noise resolution of about 2000^3.

To compare our method to delta tracking, we implemented a scene with a cloud using multiple importance sampling, illuminated by a directional light as well as a point light deep in the cloud. The only difference between the closed form and delta tracking versions of our demos is the method used to evaluate the transmittance integrals and the distance sampling method. Therefore the produced samples are identical, with the difference being the time required to produce these samples. Figure 4 shows the two demos rendered given a 5.5 second time limit at a resolution of 675 by 1200 pixels. Delta tracking produced an image with fewer samples resulting in much more noise. Table 1 compares the average time required by the most computationally intensive parts as well as the total time of the rendering process to render one sample for each pixel of the image. As seen in Table 1, our closed form demo is about 7 times faster, while transmittance evaluations and distance sampling are about 17 times faster.

Process	Closed form	Delta tracking
Ray tracing bounding surface	110ms	110ms
Transmittance evaluations and distance sampling	65ms	1136ms
Total	179ms	1250ms

Table 1: Performance comparison of our closed form evaluation and delta tracking.

All performance data has been recorded on a system with an Nvidia GeForce RTX 2080Ti and an Intel Core i9-9900K running OpenGL 4.5 on Ubuntu 18.10 using Nvidia's proprietary drivers.

6 Shaping cosine noise

In order to shape participating media, procedural noise functions are typically multiplied with other functions which contain information about the shape of the media such as a simulated volume. While this multiplication is trivial with numerical solutions, to get the performance benefits of cosine noise, we have to integrate the volume analytically. However, in this case, the product of functions generally will not have an elemental antiderivative. It is also common to shape participating media by composing the noise function with a shaping function. However, this composite usually does not have an elemental antiderivative either. While these limitations restrict the use of cosine noise in some special situations, there are other ways to shape participating media which do not prevent the analytical integration of cosine noise. We will detail these approaches in this section.

For multiplicand functions, in case we multiply cosine noise with a multiplicand function, a fast and straightforward approach is to sample the noise between bounding surfaces. We can use isosurfaces of the scalar field defined by the multiplicand function and multiply cosine noise with different c_i constants in-between bounds to approximate the product of cosine noise and a multiplicand function. Figure 5 illustrates this process.

Even with a single layer, we can get highly realistic results as seen in Figure 1. These surfaces can also be stored as meshes, enabling the use of the hardware rasterizer, making this approach optimal for real-time use. Real-time applications will be discussed in greater detail later.

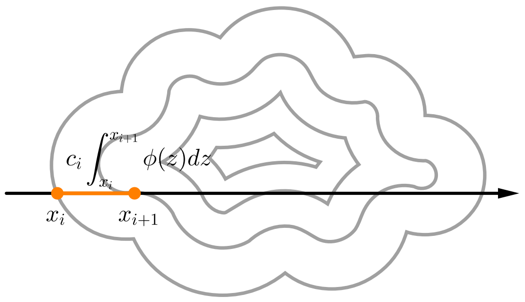


Figure 5: Illustration of isosurface traversal.

Another approach is to interpolate between samples of the multiplicand function. First, we sample the multiplicand function as we would previously, for example, ray marching with some step size, interpolate between these samples, and only multiply cosine noise with the interpolant. When using bicubic interpolation along a ray, the interpolant is given by a third-degree polynomial P. The antiderivative of the product of a polynomial interpolant and cosine noise is elemental. This approach is especially useful when cosine noise is combined with grid-based volume simulation, as interpolation is straightforward between voxels and voxel traversal is quite fast [1]. Figure 6 illustrates this process.

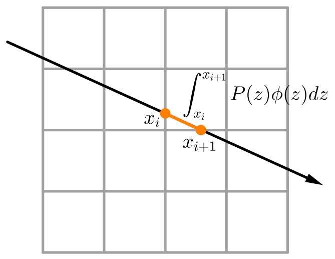
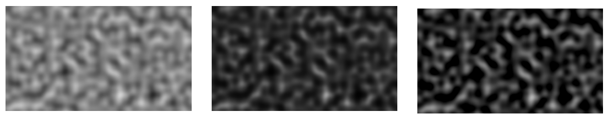


Figure 6: Illustration of voxel traversal.

Both of these approaches can be used together with density importance sampling. After determining the target extinction integral as well as the bounds or sample locations of the multiplicand function, we can iterate through these bounds along the ray, calculating the analytical extinction integral for each. After some iterations, we will find the two bounds between which we can use the previously mentioned binary search method to find the exact distance. This approach is used in Figure 1. Decoupled ray marching can be used to speed up this method similarly as described in [6]. First, we iterate through all bounds and store their extinction integrals in an array. Then we can use another binary search on this array for each subsequent sample to find the bounds that correspond to the target extinction integral. The combination of cosine noise and decoupled ray marching is especially powerful when rendering grid based simulated volumes, as a single ray intersects relatively few voxels, therefore, the size of the required array is small, while cosine noise can add a significant amount of detail, combining the best of both worlds.

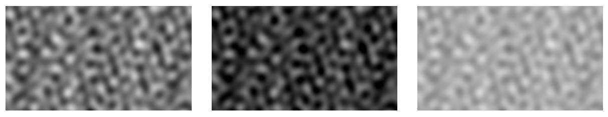
While multiplicand functions generally control the geometry of the noise, composites with shaping functions generally alter the shapes of variation in the noise, for example changing the density or making the transition between densities sharper or smoother. Figure 7 shows some examples.



(a) No shaping function applied (b) Raised to power of 2.5 (c) Clamping from 0.3 to 1.0

Figure 7: Comparison of different shaping functions applied to Perlin noise.

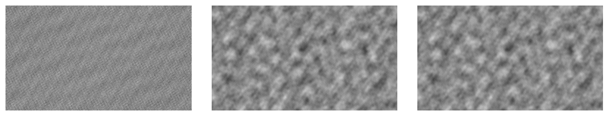
Multiplicand functions usually have a relatively low resolution and combining previous techniques with cosine noise is very effective in that case. However, if the composite of a shaping function and cosine noise does not have an elemental antiderivative, there is no simple solution. However, we have found that instead of using a shaping function, just tuning the parameters of cosine noise gives a similar amount of control. First, sparsity can be controlled by the c_{offset} parameter of the cosine noise as in Equation 2. Choosing a small offset may result in negative densities at particular coordinates, however, unless their magnitude is significant, clamping samples and integrals are enough. Large magnitudes of negative densities can cause anisotropic artefacts. Although, especially with animated volumes, small anisotropic artefacts are hard to discern. Figure 8 shows the effect of changing the offset parameter.



(a) Normal offset (b) Low offset (c) High offset

Figure 8: Comparison of cosine noises with different offsets

A second approach is to tune the parameters of the fractal noise. Changing the coefficient and the linear transformations between octaves can result in different looks. Figure 9 shows some examples.



(a) Blue noise (b) Grey noise (c) Pink noise

Figure 9: Examples of different fractal noises constructed using cosine noise.

Finally, unique noises can be produced by changing the vectors in S from the more general quasirandom vectors described in Section 3. Figure 10 shows an example.

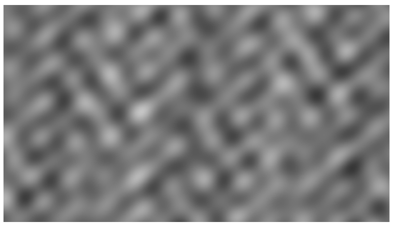


Figure 10: Cosine noise with artistic S.

When compared with shaping functions, these methods can also produce a variety of looks, satisfying many use cases.

7 Real-time application

Real-time volumetric renderers generally use ray marching to render heterogeneous media and to calculate lighting. However, even with temporal optimisation and shadow maps, only a low volume resolution can be achieved using these techniques because of the low number of samples. [4, 11] Cosine noise is especially useful here as the extinction integral towards the light can be evaluated analytically.

If we rely on bounding meshes to shape the noise, we can use the following method. First we need to draw the visible part of the bounding meshes similarly to transparency rendering techniques. For example, AMD's linked list method is a good option [16]. After this step, we should have a list of bounds for each pixel. We also need to create a multi-layer shadow map, which we can do by repeating the same process. Now we can use the importance sampling method described in Section 5. With just 16 quasirandom samples we can get a high level of detail. With such low sample counts, however, importance sampling based on only an average extinction reduces noise and allows for more samples as the binary search is not necessary. Figure 11 shows a screenshot of our implementation, as a single scattering volume bound by a sphere is illuminated by a directional light.

8 Future work

Real-time applications of cosine noise could be explored more in depth. The closed form integral of cosine noise could potentially be extended to support depth of field and motion blur by integrating over multiple domains. Shaping methods for cosine noise could also be explored further. We would also like to explore the use of other periodic functions in place of cosines. Cosine noise could be useful to compress 3D textures.

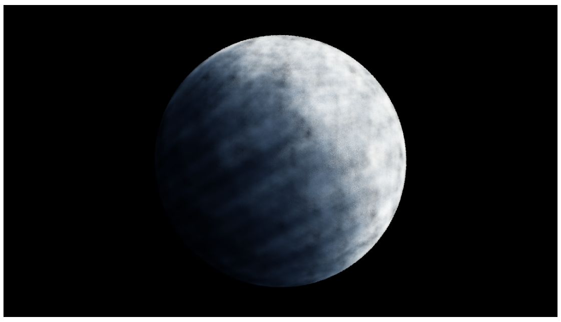


Figure 11: Real-time implementation of our method. Running at 80 frames per second at a resolution of 675 by 1200. Effective noise resolution of about 2000^3.

9 Conclusion

We proposed cosine noise as a solution to the high computational cost of rendering participating media with a high-resolution procedural noise using previous methods. We have shown that our method is significantly faster at evaluating transmittance in high-resolution noise than delta tracking. Our method is optimal for use on GPUs and offers a higher effective volume resolution in real-time applications than previous approaches. While our method is limited in terms of shaping cosine noise, we proposed several alternative approaches to mitigate this limitation.

References

[1] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10, 1987.

[2] Blender Foundation. Smoke Domain smoke high resolution, 2018. Available at `https://web.archive.org/web/20181118204606/https://docs.blender.org/manual/en/latest/physics/smoke/types/domain.html#smoke-high-resolution`.

[3] Eric Heitz, Jonathan Dupuy, Stephen Hill, and David Neubelt. Real-time polygonal-light shading with linearly transformed cosines. *ACM Transactions on Graphics (TOG)*, 35(4):41, 2016.

[4] Sébastien Hillaire. Physically based sky, atmosphere and cloud rendering in Frostbite. 2016. Available at `https://media.contentapi.ea.com/content/dam/eacom/frostbite/files/s2016-pbs-frostbite-sky-clouds-new.pdf`.

[5] Tomáš Iser. Real-time light transport in analytically integrable quasi-heterogeneous media. *Proceedings of CESCG 2018: The 22nd Central European Seminar on Computer Graphics (non-peer-reviewed)*, 2018.

[6] Christopher Kulla and Marcos Fajardo. Importance sampling techniques for path tracing in participating media. In *Computer graphics forum*, volume 31, pages 1519–1528. Wiley Online Library, 2012.

[7] Jan Novák, Andrew Selle, and Wojciech Jarosz. Residual ratio tracking for estimating attenuation in participating media. *ACM Trans. Graph.*, 33(6):179–1, 2014.

[8] Vincent Pegoraro and Steven G. Parker. An Analytical Solution to Single Scattering in Homogeneous Participating Media. *Comput. Graph. Forum*, 28(2):329–335, 2009.

[9] Ken Perlin and Eric M Hoffert. Hypertexture. In *ACM Siggraph Computer Graphics*, volume 23, pages 253–262. ACM, 1989.

[10] Matthias Raab, Daniel Seibert, and Alexander Keller. Unbiased global illumination with participating media. In *Monte Carlo and Quasi-Monte Carlo Methods 2006*, pages 591–605. Springer, 2008.

[11] Andrew Schneider and Nathan Vos. The real-time volumetric cloudscapes of horizon: Zero dawn. *Advances in Real-Time Rendering in Games, ACM SIGGRAPH*, 2015.

[12] Il'ya Meerovich Sobol'. On the distribution of points in a cube and the approximate evaluation of integrals. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 7(4):784–802, 1967.

[13] L. Szirmay-Kalos, I. Georgiev, M. Magdics, B. Molnár, and D. Légrády. Unbiased light transport estimators for inhomogeneous participating media. *Computer Graphics Forum*, 36(2):9–19, 2017.

[14] László Szirmay-Kalos, Balázs Tóth, and Milán Magdics. Free path sampling in high resolution inhomogeneous participating media. In *Computer Graphics Forum*, volume 30, pages 85–97. Wiley Online Library, 2011.

[15] E. Woodcock, T. Murphy, P. Hemmings, and S. Longworth. Techniques used in the GEM code for Monte Carlo neutronics calculation. In *Proc. Conf. Applications of Computing Methods to Reactors, ANL-7050*, 1965.

[16] Jason C Yang, Justin Hensley, Holger Grün, and Nicolas Thibieroz. Real-time concurrent linked list construction on the gpu. In *Computer Graphics Forum*, volume 29, pages 1297–1304. Wiley Online Library, 2010.