Real-Time Rendering of Atmospheric Clouds

Parker Ford

Advisor: Prof. Kelvin Sung

Committee Members: Clark Olson, Yusuf Pisan

BOTHELL

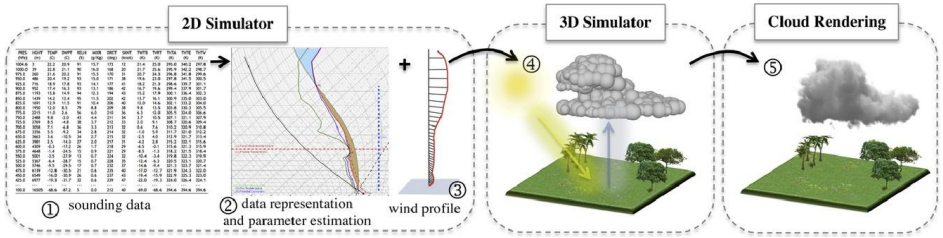
Problem

- Rendering realistic clouds is an important aspect of creating believable virtual environments
- Due to their detailed shapes and complex light interactions, rendering clouds in real time is a difficult task
- The task can be broken into two sub problems:
  1. Cloud Modeling
  2. Cloud Rendering

Related Work – Cloud Modeling

- Particle Based Method:
  - Simulating cloud formation where particles represent air parcels with water molecules that converge under atmospheric conditions [1]
  - Provides realistic results based on physics; is computationally expensive
- Procedural Based Method:
  - Modeling clouds through the use of several procedural noise functions [2]
  - Realistic appearance with efficient run-time; not based in physics

Particle Based Example [3]:



[1] T. Hädrich, M. Makowski, W. Pałubicki, D. T. Banuti, S. Pirk, and D. L. Michels, "Stormscapes: simulating cloud dynamics in the now," *ACM Trans. Graph.*, vol. 39, no. 6, pp. 1–16, Dec. 2020, doi: 10.1145/3414685.3417801.

[2] A. Schneider, "Nubis: authoring real-time volumetric cloudscapes with the Decima Engine," SIGGRAPH Advances in Real-Time Rendering in Games Course. ACM, pp. 619–620, 2017.

[3] R. P. M. Duarte and A. J. P. Gomes, "Real-time simulation of cumulus clouds through SkewT/LogP diagrams," Computers & Graphics, vol. 67, pp. 103–114, Oct. 2017, doi: 10.1016/j.cag.2017.06.005.

Related Work – Cloud Rendering

- Texture Billboard:
  - Pre-rendered images of clouds onto textures in place of 3D clouds [4]
  - Very efficient, not dynamic
- Volumetric:
  - Rendering volumetric material by simulating light transport [5]
  - Dynamic shapes and lighting, more expensive

Texture Billboard Example [4]:

[4] M. J. Harris and A. Lastra, “Real-time cloud rendering for games,” presented at the Proceedings of Game Developers Conference, 2002, pp. 21–29.

[5] J. Fong, M. Wrenninge, C. Kulla, and R. Habel, “Production volume rendering: SIGGRAPH 2017 course,” in ACM SIGGRAPH 2017 Courses, Los Angeles California: ACM, Jul. 2017, pp. 1–79. doi: 10.1145/3084873.3084907.

Goals

- We aim to provide a cloud rendering system that:
  - Model's the various detailed shapes that clouds can take
  - Simulate light transport in a realistic appearing way
  - Dynamically change shape and lighting based on their environment
  - Be efficient enough for real-time applications

- Due to these goals our solution implements:
  - A procedural approach for cloud modeling
  - A volumetric approach for cloud rendering

Roadmap

I. Solution Design
II. Implementation
III. Results
IV. Conclusion

I. Solution Design

- Cloud Modeling Design
- Cloud Rendering Design

Cloud Modeling Design

- Based on the procedural method as described by Schneider [2]
- The goal of the modeling approach is to describe where in the atmosphere our clouds will form.
- This is accomplished with several lookup textures that map 3D positions to cloud density values.

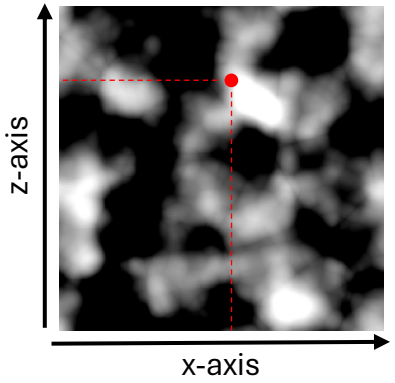
[2] A. Schneider, “Nubis: authoring real-time volumetric cloudscapes with the Decima Engine,” SIGGRAPH Advances in Real-Time Rendering in Games Course. ACM, pp. 619–620, 2017.

Cloud Maps

- Cloud Coverage Map: Defines the potential density of clouds based on the longitude and latitude in the world.
- Cloud Type Map: Defines the type of cloud formation based on the longitude and latitude in the world.
- Cloud Height Gradient: Defines the potential density of clouds based on cloud type and altitude.

Cloud Coverage

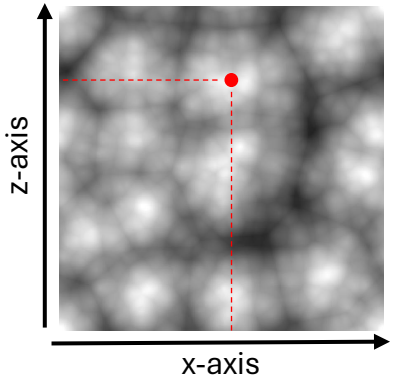
Cloud Coverage Map



Cloud Coverage
0.75

3D position
x = 550 , y = 600 , z = 800

Cloud Type Map



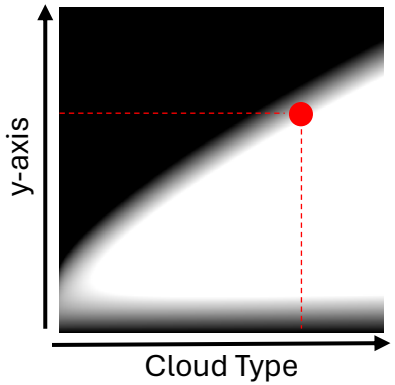
Cloud Type
0.8

Vertical Profile

3D position
x = 550 , y = 600 , z = 800

Cloud Type
0.8

Cloud Height Gradient



Vertical Profile
0.9

Dimensional Profile

Cloud Coverage
(Potential density based on longitude and latitude)
0.75

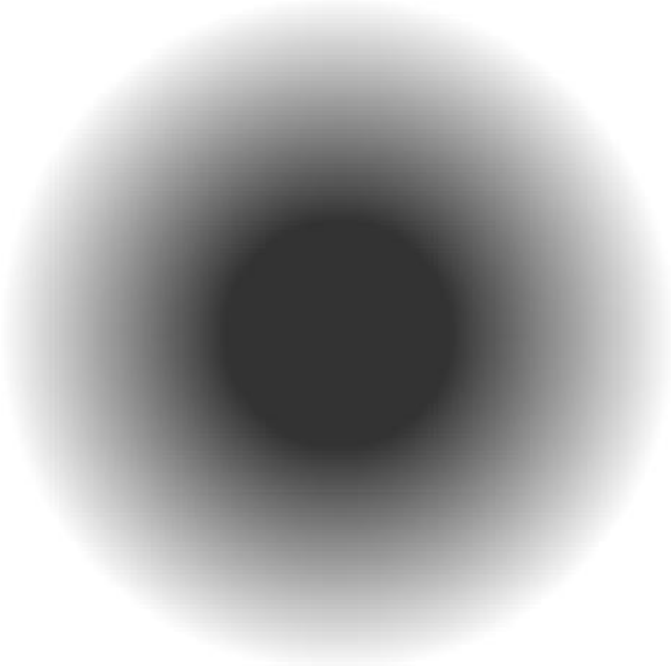
×

Vertical Profile
(Potential density based on altitude)
0.9

=

Dimensional Profile
(Total potential density)
0.675

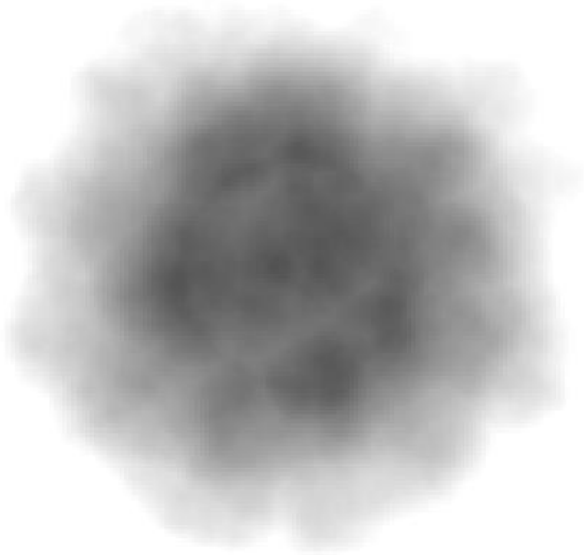
Dimensional Profile



Cloud Density

Noise Erosion

$$\rho = \left(N - \left(1 - d_p\right)\right)$$



Cloud Rendering

- We follow Fong's volumetric rendering framework which simulates the interaction of light rays through a volumetric medium [5]
- Additionally, we integrate other's research into Fong's volumetric framework to capture additional lighting characteristics of clouds

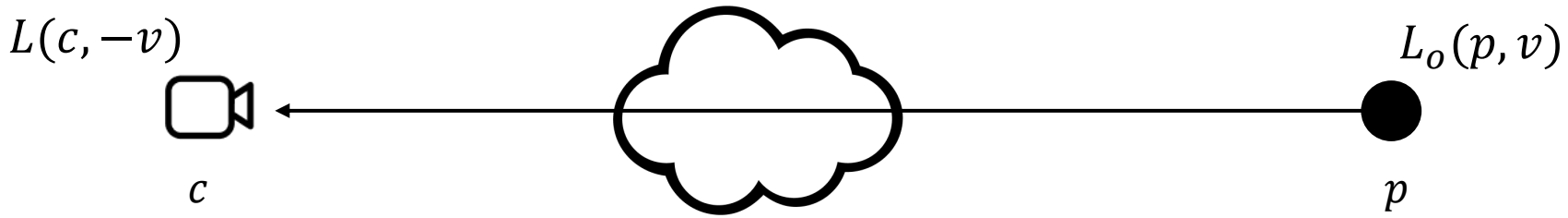
[5] J. Fong, M. Wrenninge, C. Kulla, and R. Habel, "Production volume rendering: SIGGRAPH 2017 course," in ACM SIGGRAPH 2017 Courses, Los Angeles California: ACM, Jul. 2017, pp. 1–79. doi: 10.1145/3084873.3084907.

Volumetric Rendering

- Radiance is the measurement of light intensity along a ray.
- Transmittance is the ratio of light that is capable of passing through some interval.
- σ_s: Medium's potential to scatter light
- σ_a: Medium's potential to absorb light
- σ_t: Summation of σ_s and σ_a
- σ_s , σ_a , and σ_s are scaled by density (ρ)

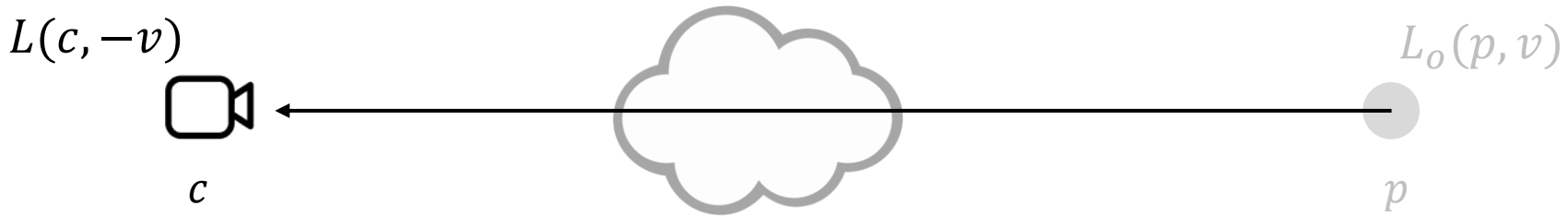
Radiance Equation

$$L(c,-v) = Tr(c,p)L_o(p,v) + \int_{t=0}^{||p-c||} Tr\,(c,c-vt)L_{scat}(c-vt,v)\sigma_s dt$$



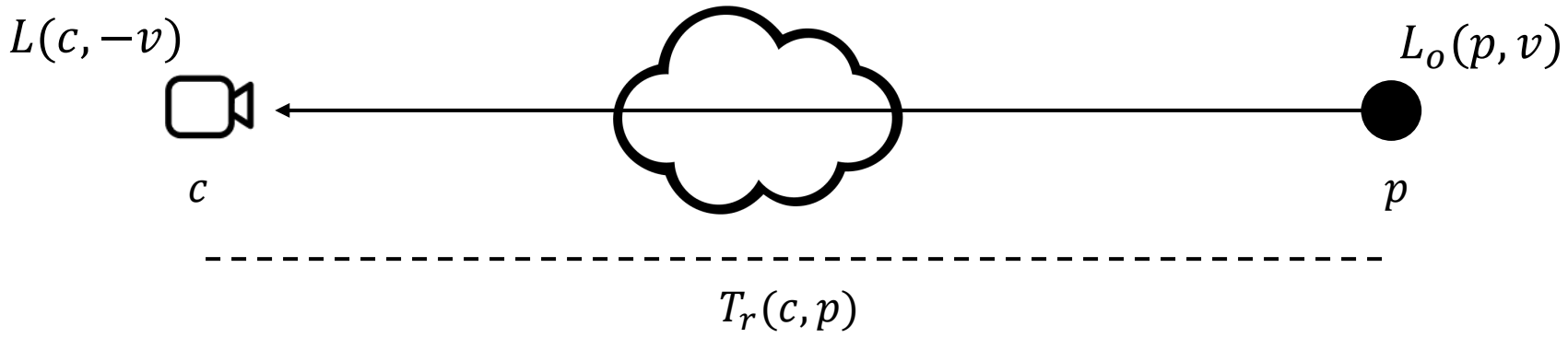
Radiance Equation

$$L(c,-v) = Tr(c,p)L_o(p,v) + \int_{t=0}^{||p-c||} Tr\,(c,c-vt)L_{scat}(c-vt,v)\sigma_s dt$$



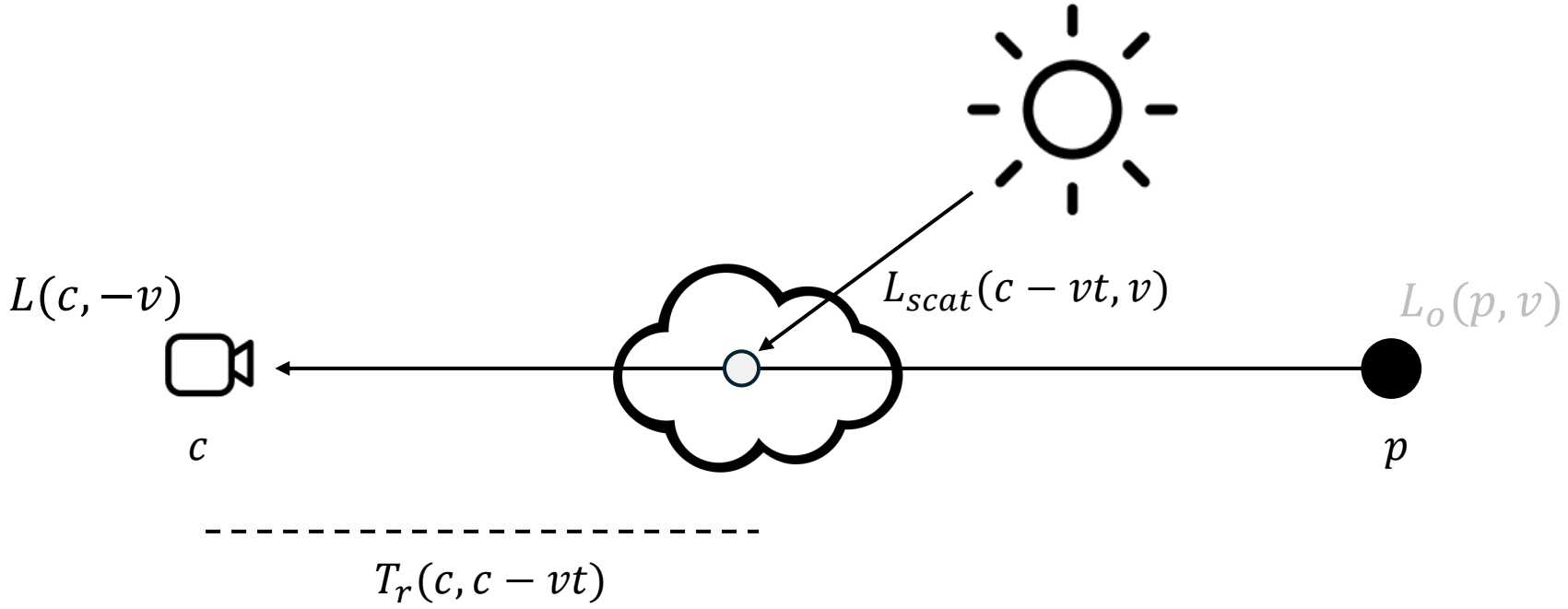
Radiance Equation

$$L(c,-v) = Tr(c,p)L_o(p,v) + \int_{t=0}^{||p-c||} Tr\,(c,c-vt)L_{scat}(c-vt,v)\sigma_s dt$$



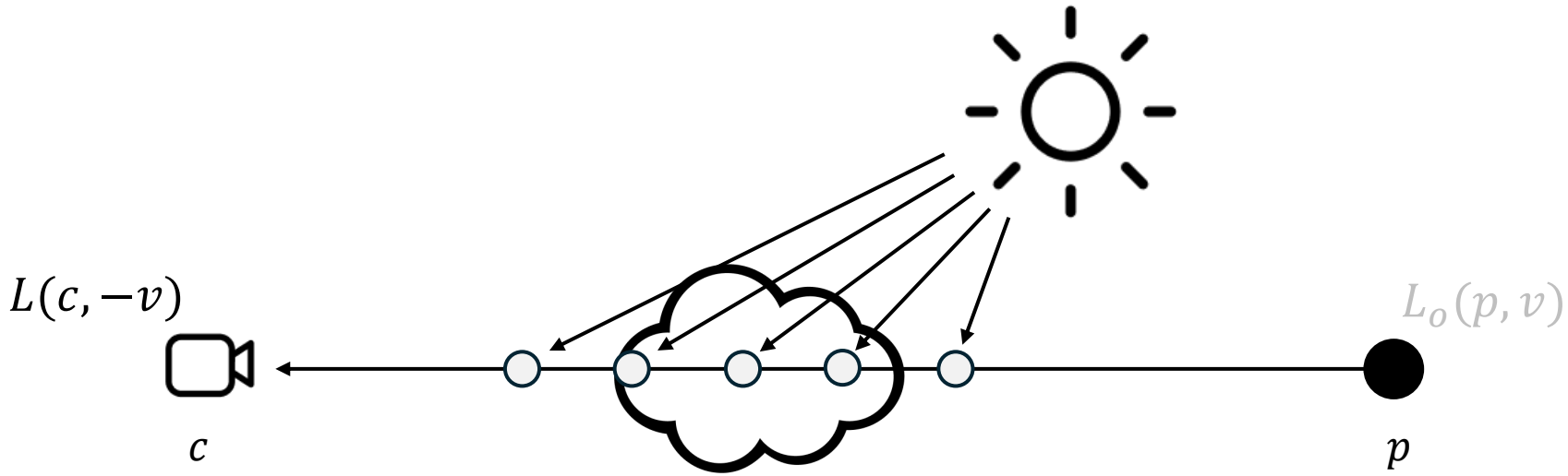
Radiance Equation

$$L(c,-v) = Tr(c,p)L_o(p,v) + \int_{t=0}^{||p-c||} Tr\,(c,c-vt)L_{scat}(c-vt,v)\sigma_s dt$$



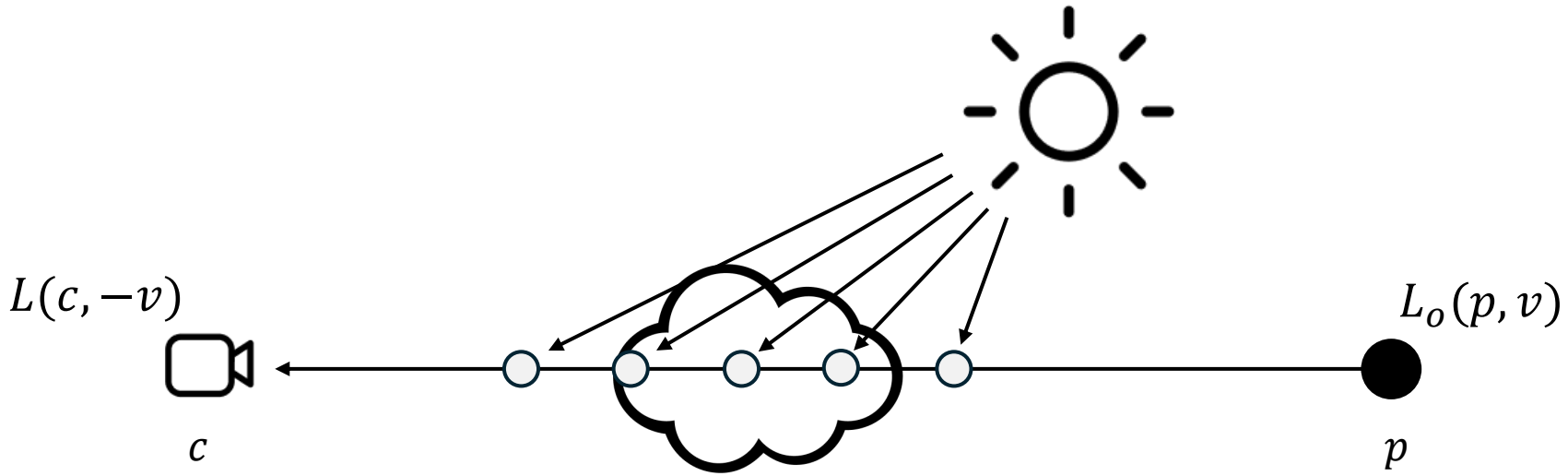
Radiance Equation

$$L(c,-v) = Tr(c,p)L_o(p,v) + \int_{t=0}^{||p-c||} Tr\,(c,c-vt)L_{scat}(c-vt,v)\sigma_s dt$$

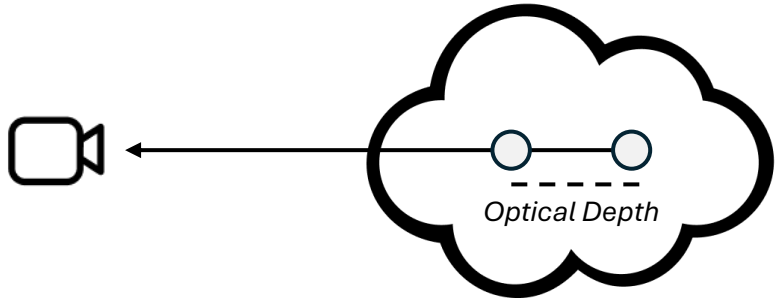


Radiance Equation

$$L(c,-v) = Tr(c,p)L_o(p,v) + \int_{t=0}^{||p-c||} Tr\,(c,c-vt)L_{scat}(c-vt,v)\sigma_s dt$$



Transmittance



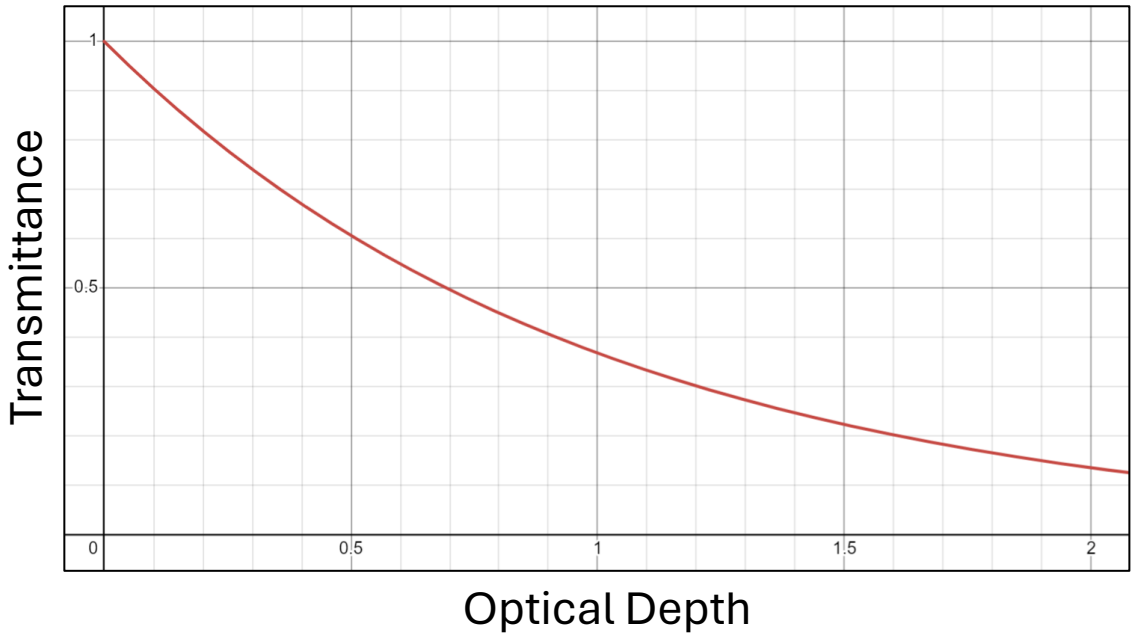
Transmittance

$$T_r(x_a, x_b) = e^{-\tau}$$

Optical Depth

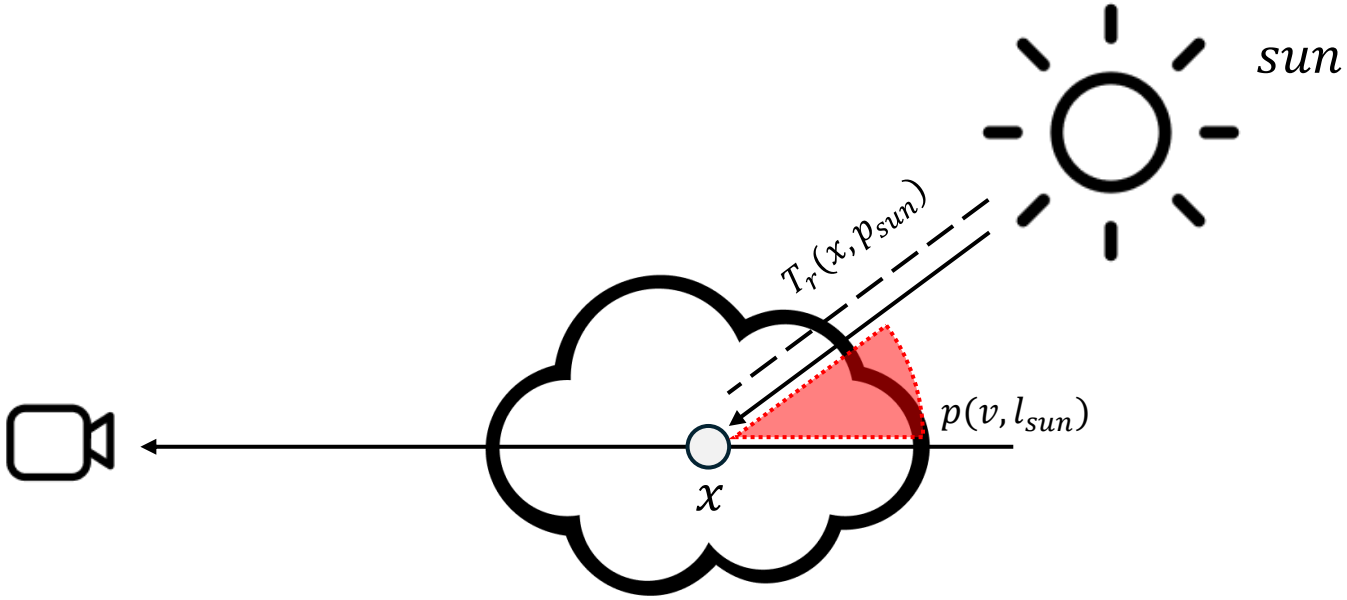
$$\tau = \int_{x=x_a}^{x_b} \sigma_t \, dx$$

Beer-Lambert Law



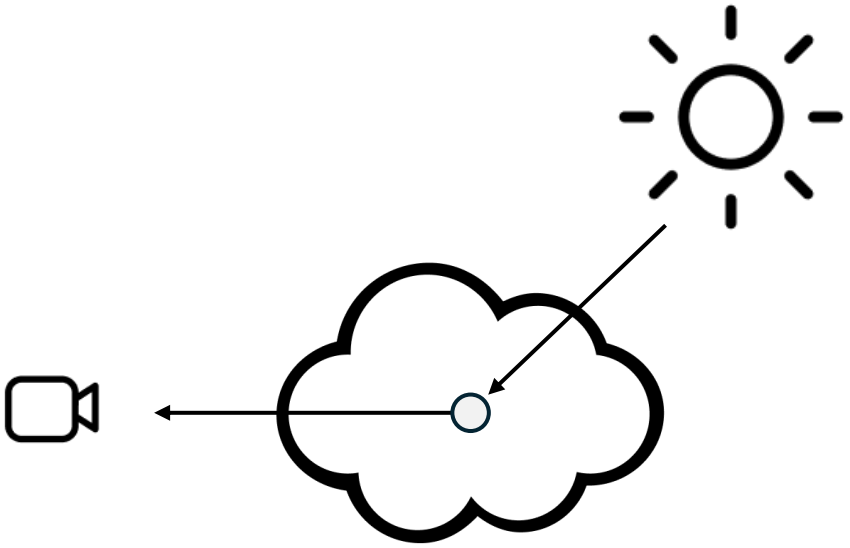
Scattering

$$L_{scat}(x, v) = p(v, l_{sun})T_r(x, p_{sun})$$

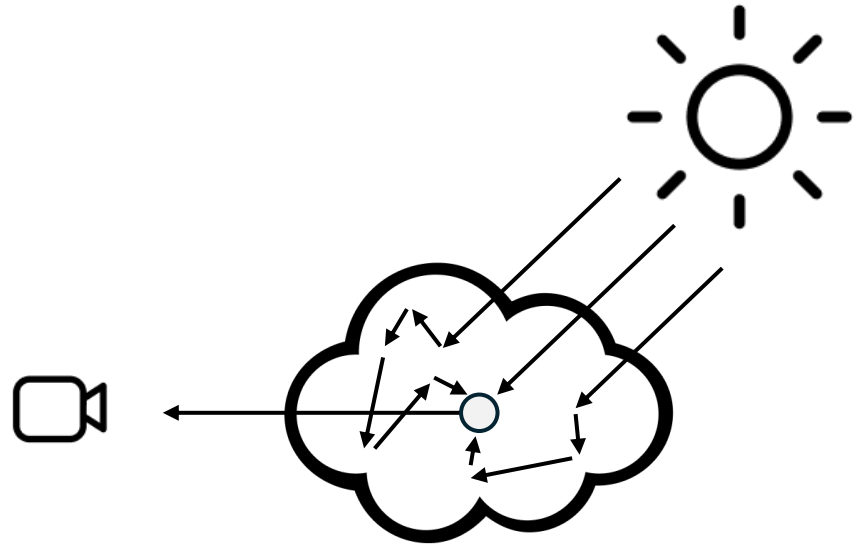


Single vs Multiple Scattering

Single Scattering



Multiple Scattering



Multiple Scattering Approximation

$$L(c,-v) = Tr(c,p)L_o(p,v) + \int_{t=0}^{||p-c||} Tr\,(c,c-vt)L_{scat}(c-vt,v)\sigma_s dt$$

$$L(c,-v) = Tr(c,p)L_o(p,v) + \int_{t=0}^{||p-c||} Tr\,(c,c-vt)L_{mult}(c-vt,v)\sigma_s dt$$

Multiple Scattering Approximation

Single Scattering Equation

$$L_{scat}(x, v) = p(v, l)e^{-\tau}$$

Wrenninge's Multiple Scattering Equation [6]

$$L_{mult}(x, v) = \sum_{i=0}^{N-1} L_i(x, v)$$

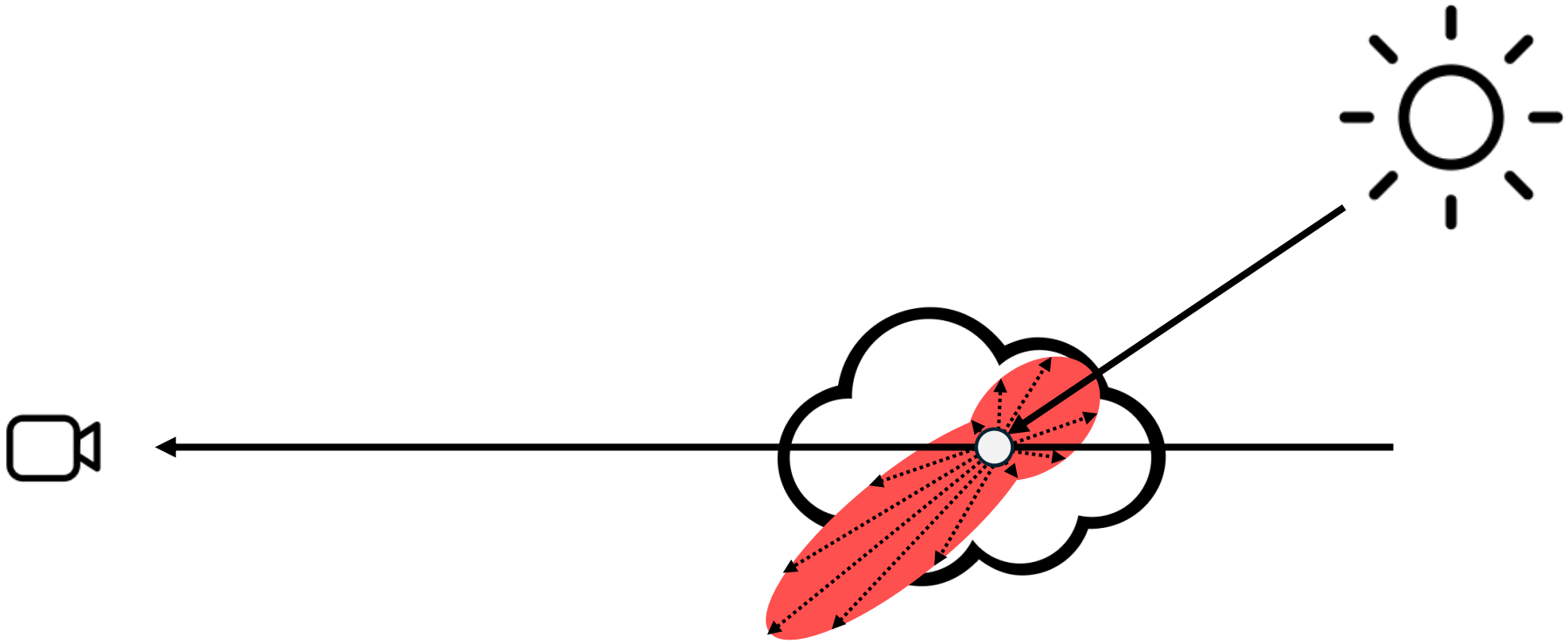
$$L_i(x, v) = b^i p(v, l)e^{-a^i \tau}$$

$$0 \leq a < 1$$

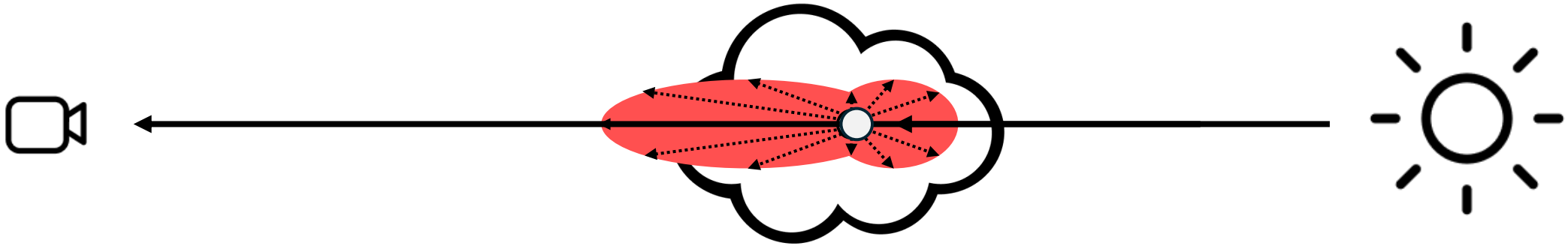
$$0 \leq b < 1$$

[6] M. Wrenninge, "Art-directable multiple volumetric scattering," in ACM SIGGRAPH 2015 Talks, Los Angeles California: ACM, Jul. 2015, pp. 1–1. doi: 10.1145/2775280.2792512.

Phase Function

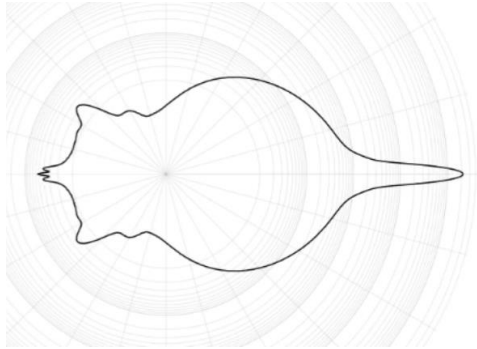


Phase Function



Mie Scattering

- Real physical phenomena can be modeled by Mie Scattering
- For a given droplet size, wavelength, and angle, you can derive directional probability.
- Requires large lookup tables



Phase Functions

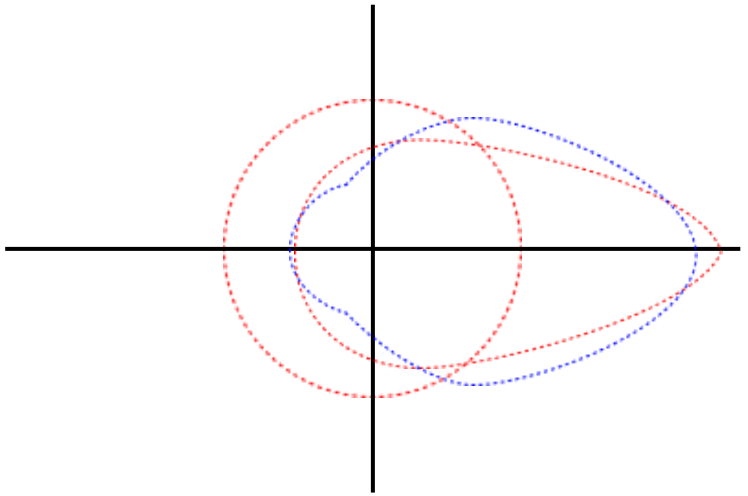
- Several phase functions have been developed to approximate Mie Scattering without lookup tables:

Henyey-Greenstein:

$$p_{hg}(\theta, g) = \frac{1 - g^2}{4\pi\left(1 + g^2 - 2gcos(\theta)\right)^{1.5}}$$

Draine:

$$p_d(\theta, \alpha, g) = \frac{1 - g^2}{4\pi(1 + g^2 - 2g\cos(\theta))^{1.5}} \frac{1 + \alpha\cos(\theta)^2}{1 + \frac{\alpha(1 + 2g^2)}{3}}$$

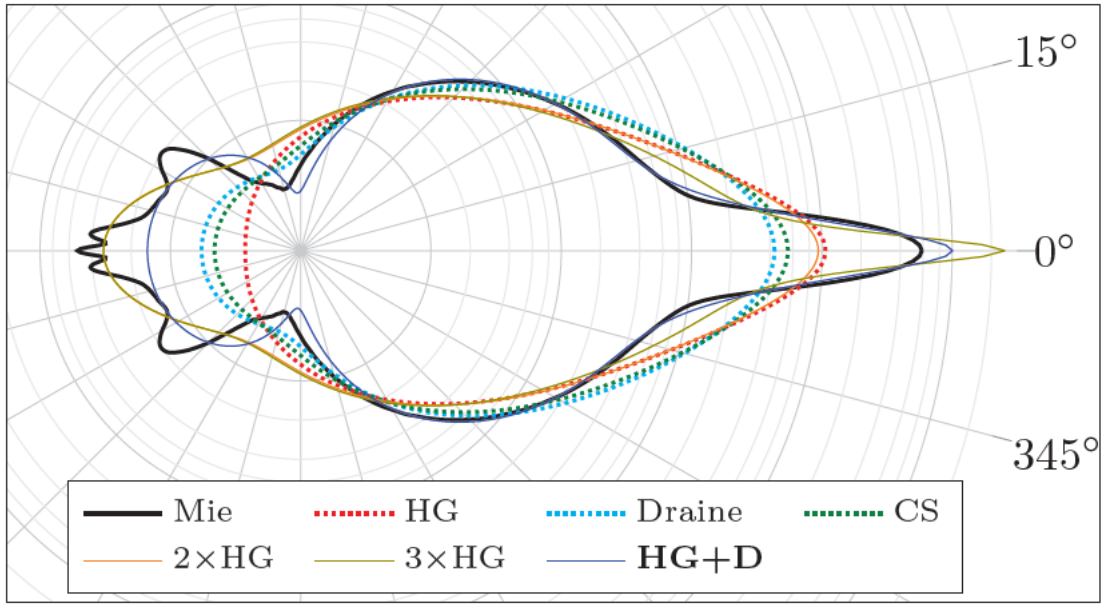


Phase Functions

[7] J. Jendersie and E. d'Eon, "An Approximate Mie Scattering Function for Fog and Cloud Rendering," in ACM SIGGRAPH 2023 Talks, Los Angeles CA USA: ACM, Aug. 2023, pp. 1–2. doi: 10.1145/3587421.3595409.

Jendersie & d'Eon's Phase Function [7]:

$$p(\theta, \alpha, g_{hg}, g_d, w) = (1 - w)\, p_{hg}(\theta, g_{hg}) + w\, p_d(\theta, \alpha, g_d)$$



Parameters based on diameter:

$$g_{hg}(20) = 0.9881$$

$$g_d(20) = 0.5567$$

$$\alpha(20) = 21.9955$$

$$w(20) = 0.4824$$

Summary

- Our modeling approach is a straightforward adaptation of Schneider's approach
- Our Rendering approach follows the volumetric framework provided by Fong and integrates others research to capture additional lighting characteristics of clouds:
  - Wrenninge's approach to account for multiple scattering
  - Jendersie & d'Eon's phase function to account for the angular distribution of light scattering within clouds

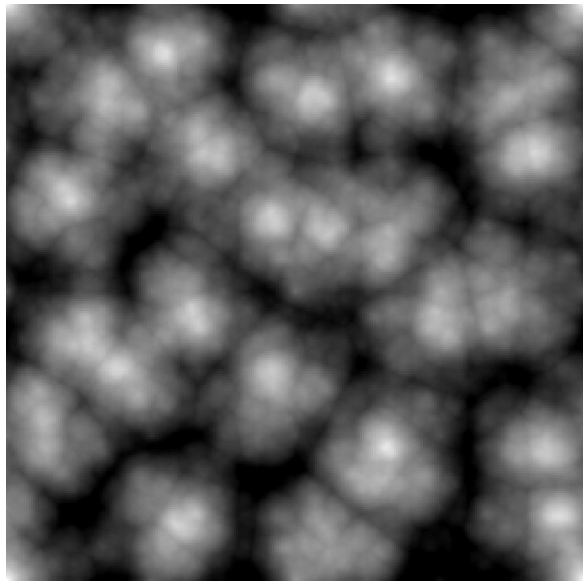
II. Implementation

- Cloud Texture Generation
- Ray Marching Algorithm
- Anti-Aliasing

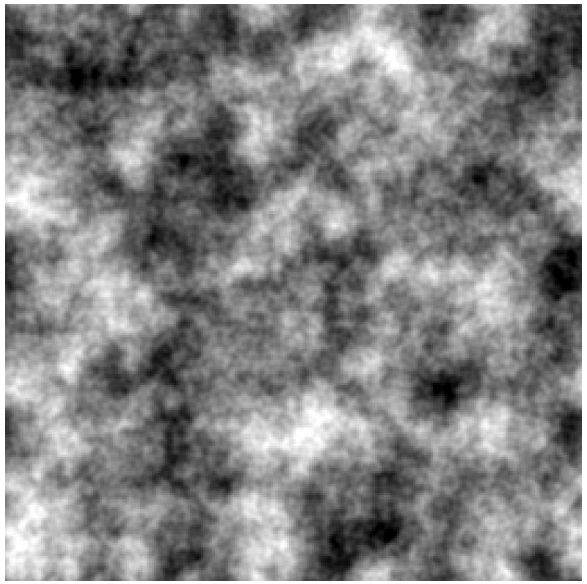
Fractional Brownian Motion (FBM)

- Technique to incrementally add detail to noise textures

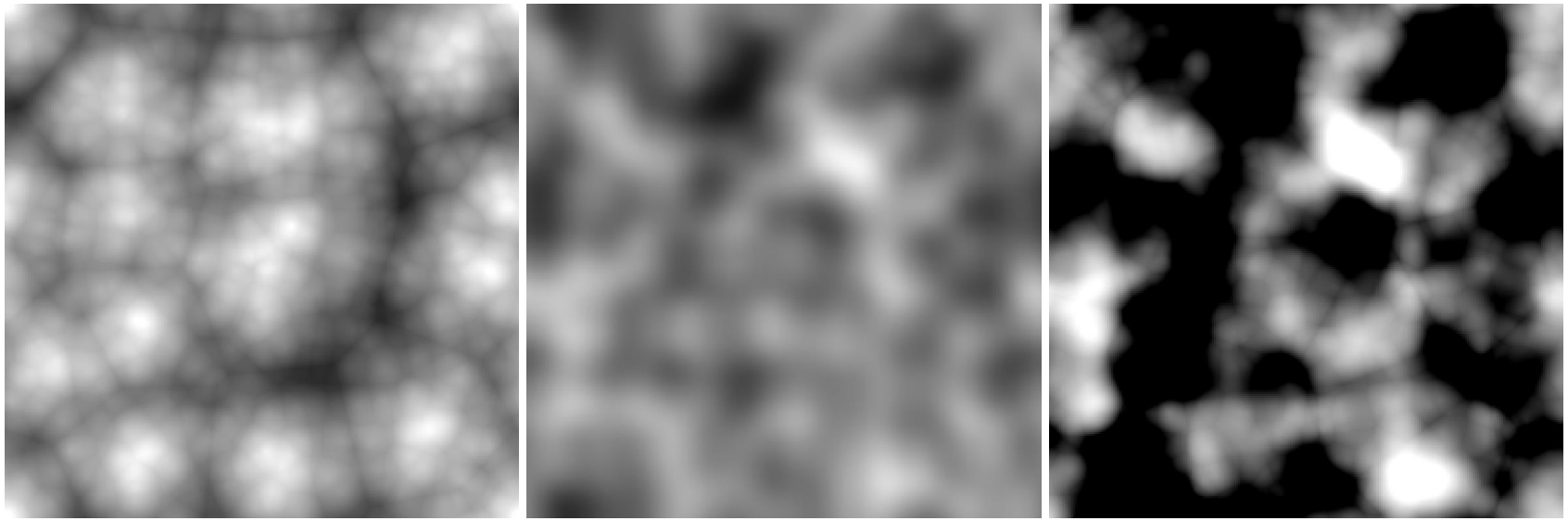
Inverted Worley Noise



Perlin Nosie



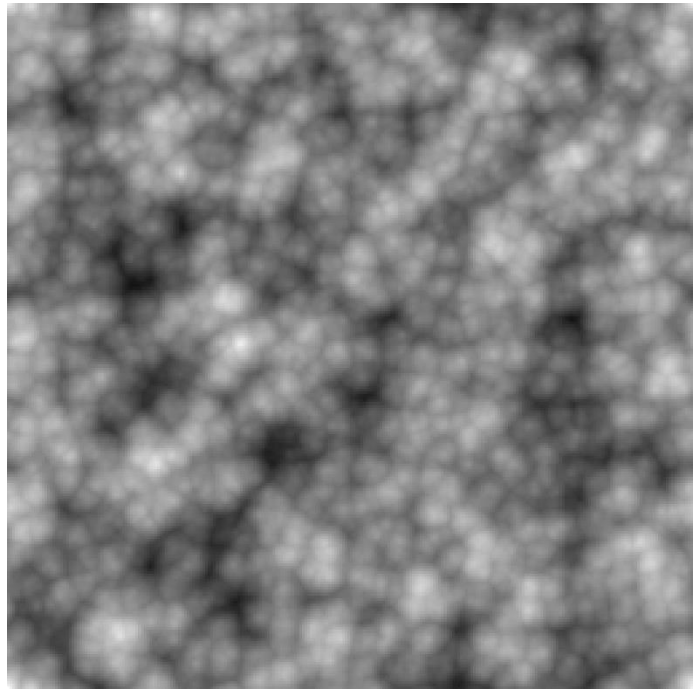
Cloud Coverage Map



1. Inverted Worley FBM
2. Perlin Noise
3. Remapped Worley Noise

Cloud Type Map

Global Type Value +



Inverted Worley FBM

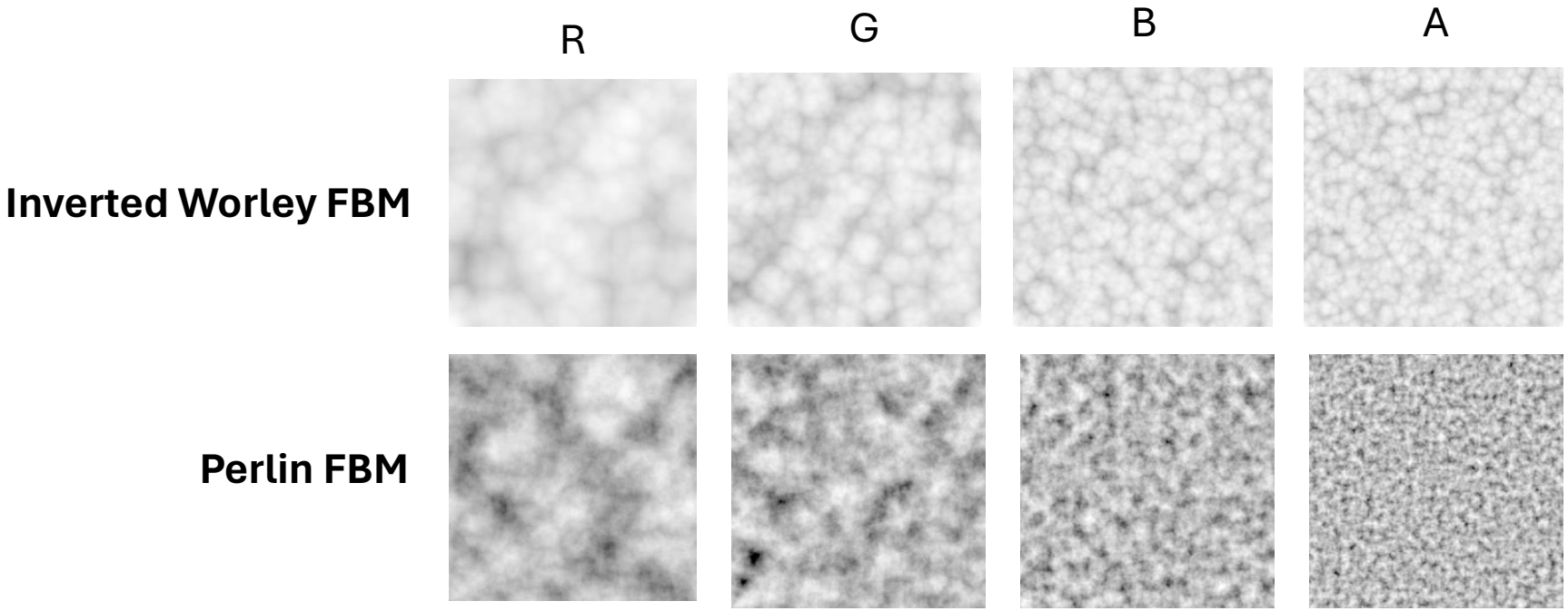
Cloud Height Gradient

Stratus Gradient

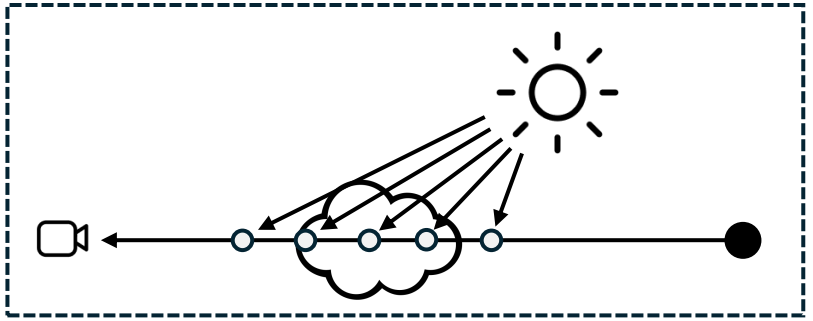
Cumulus Gradient

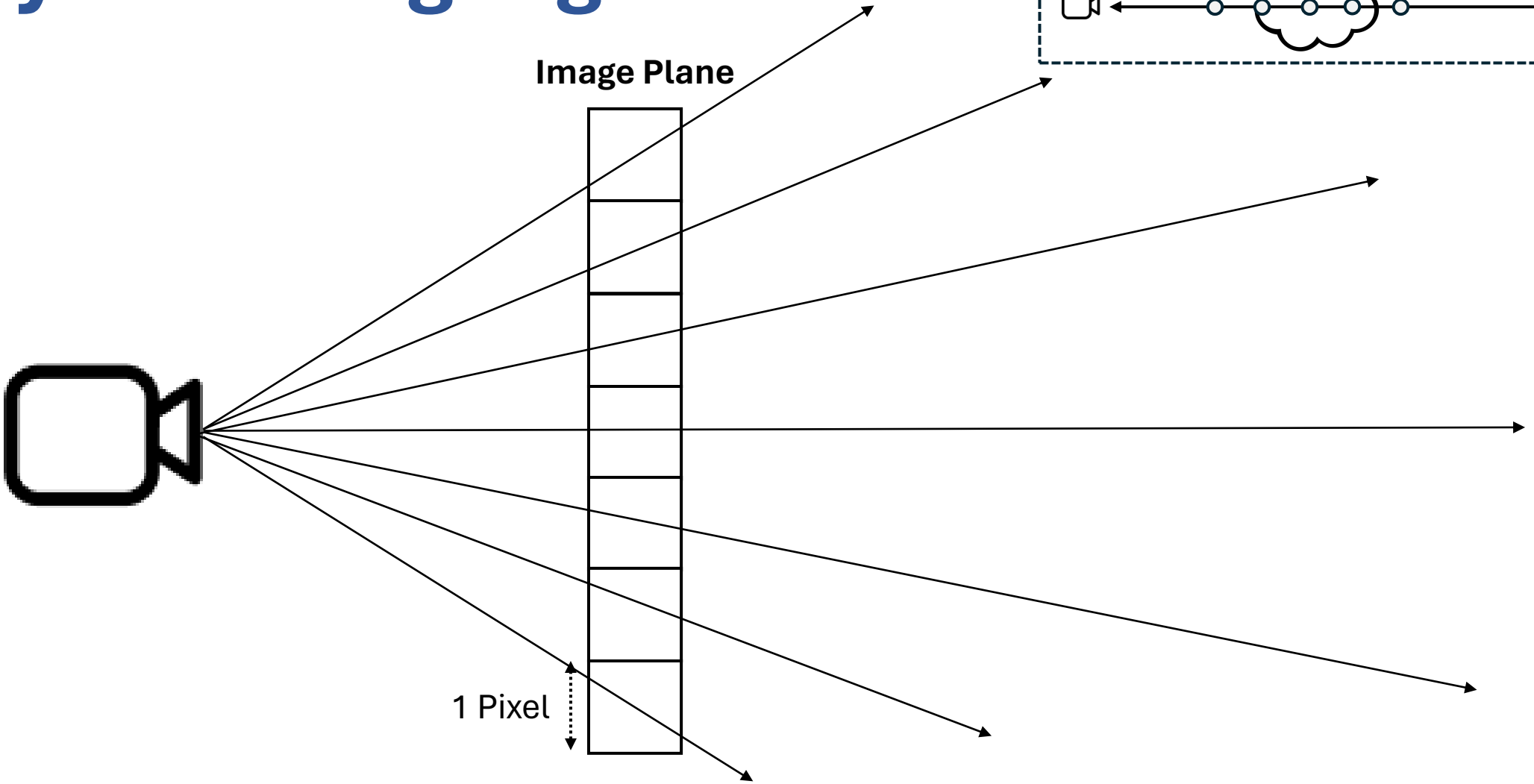
Interpolated Gradients

Cloud Nosie 3D Textures

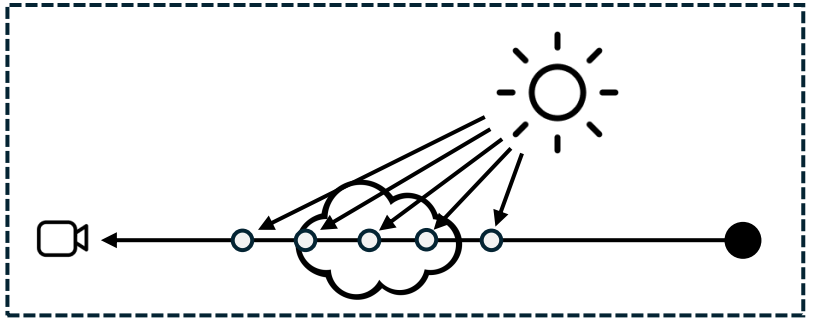


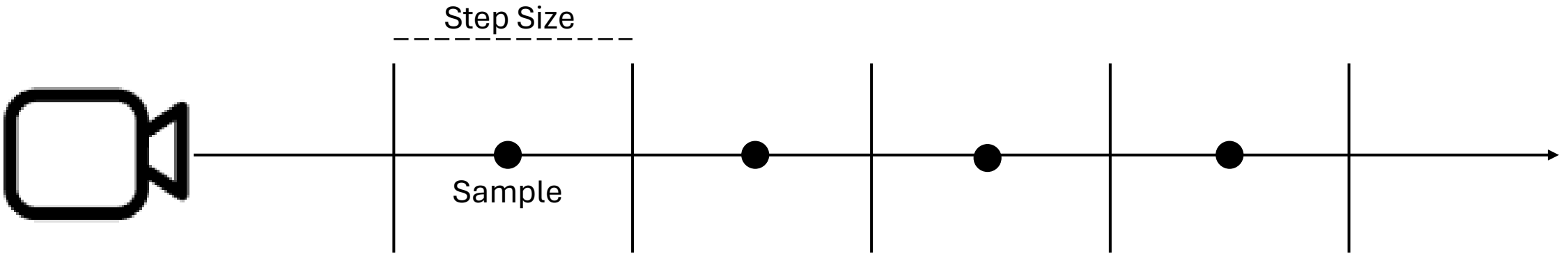
Ray Marching Algorithm



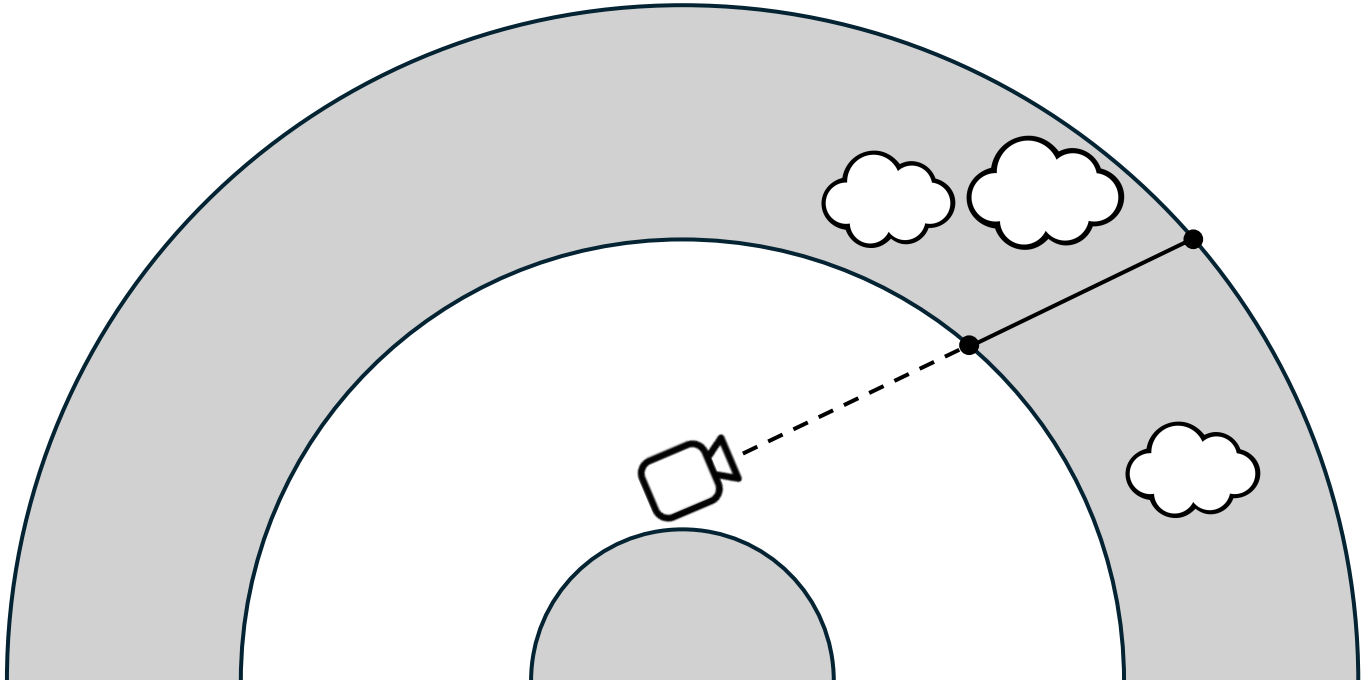


Ray Marching Algorithm



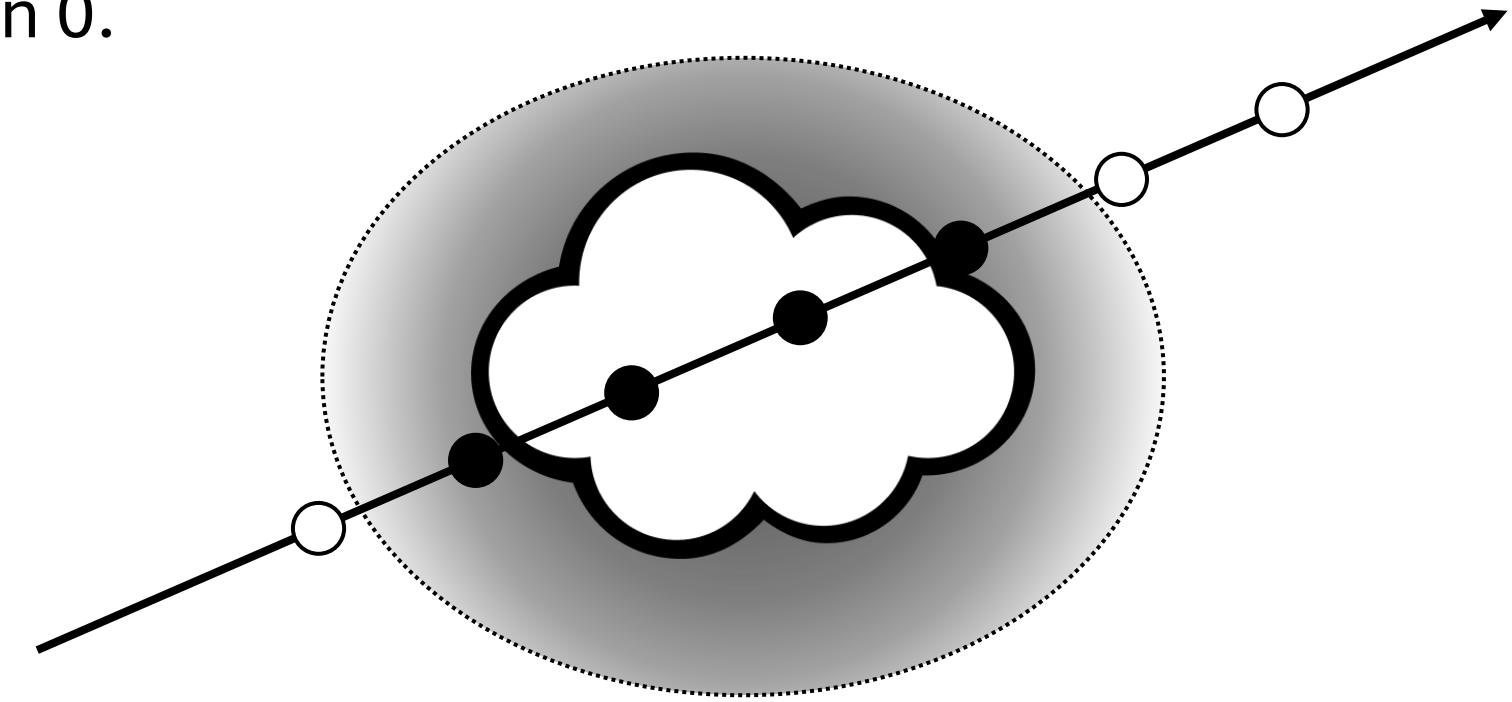


Ray Marching Algorithm



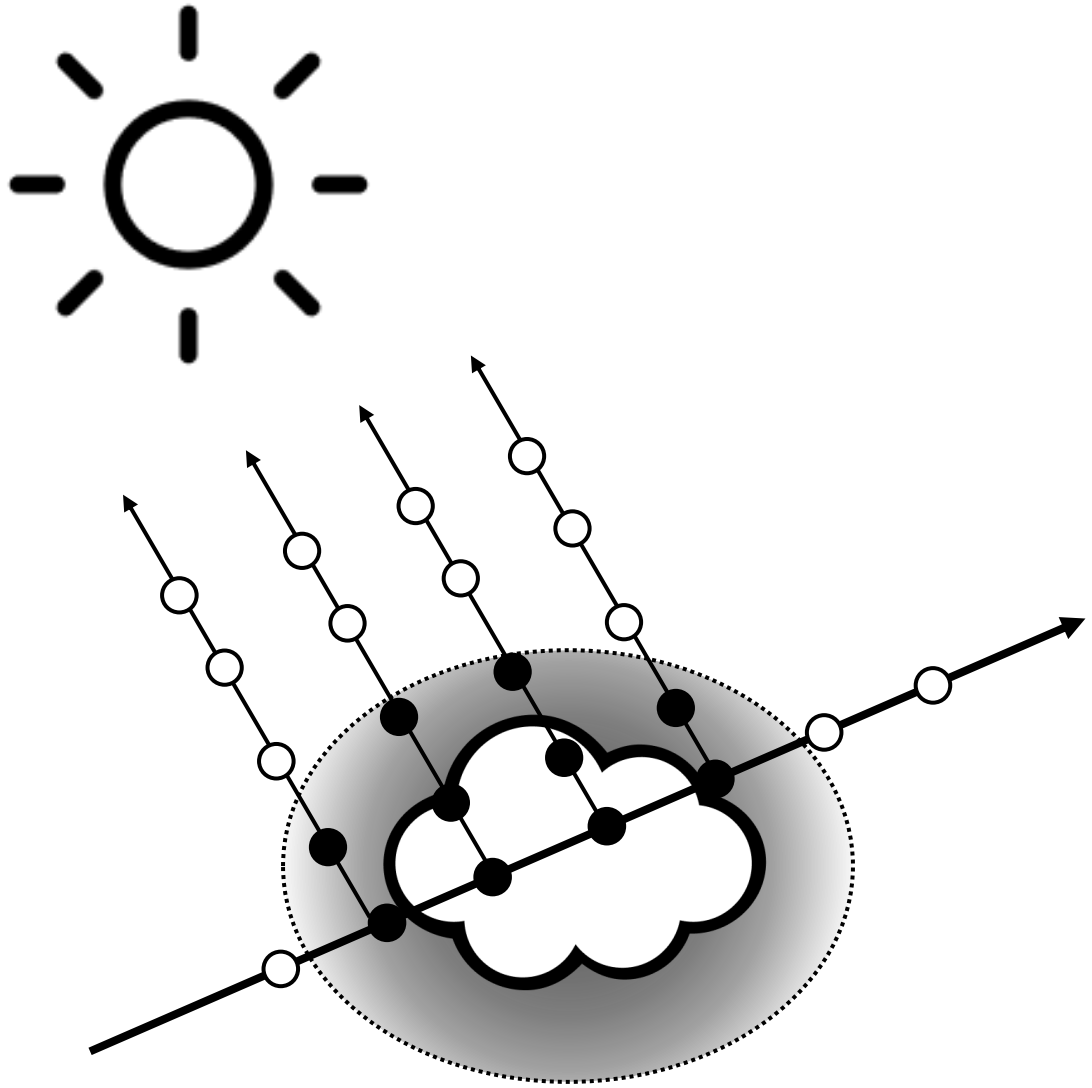
Ray Marching Algorithm

Sample Transmittance value where dimensional profile is greater than 0.



Ray Marching Algorithm

Scattering samples are calculated by performing another ray march from the sample position towards the light source.



Improved Scattering Integration

- The typical approach for integrating scattered light through the volume is:

$$e^{-\int_{x_a}^{x_b} \sigma_t \, dx} \, L_{scat}(x, v) \, \sigma_s \, dx$$

- This however leads to inaccuracies when the length between x_a and x_b is large.
- Hillaire proposes an alternate integration approach [8]:

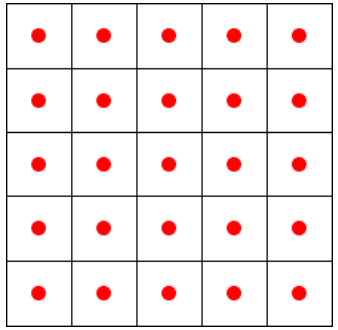
$$e^{-\int_0^d \sigma_t \, dx} \, L_{scat}(x, v) \, \sigma_s \, dx \approx \frac{L_{scat}(x,v) \, \sigma_s \, (1 - e^{-d})}{\sigma_t}$$

[8] S. Hillaire, “Physically Based Sky, Atmosphere and Cloud Rendering in Frostbite”.

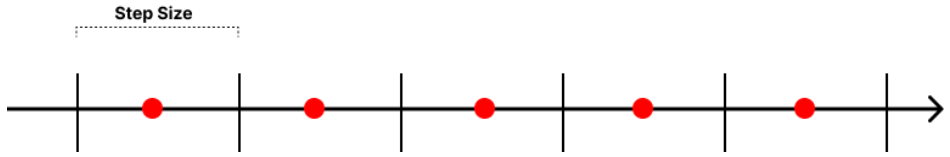
Aliasing

- Aliasing artifacts will be seen in our renders due to sampling at the center of each pixel as well as the center of each ray march segment.
- We mitigate this by introducing randomness into our sample positions.

Pixel Sampling



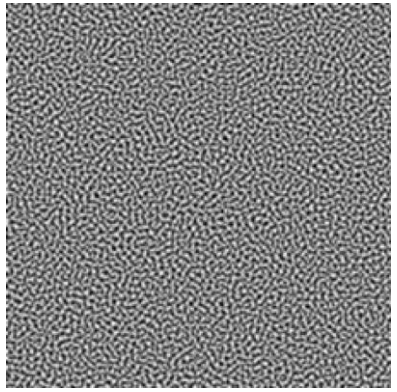
Segment Sampling



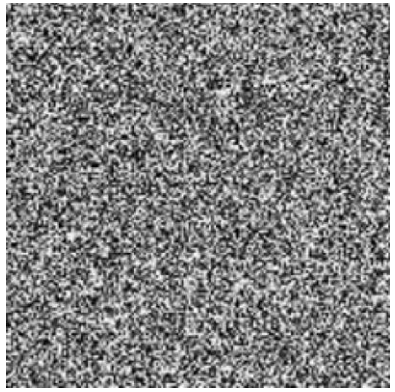
Randomization

- For our randomization strategy, each pixel samples from a series of 3 blue noise textures.
- Blue noise provides randomization without the introduction of as much visual noise as white noise.

Blue Noise

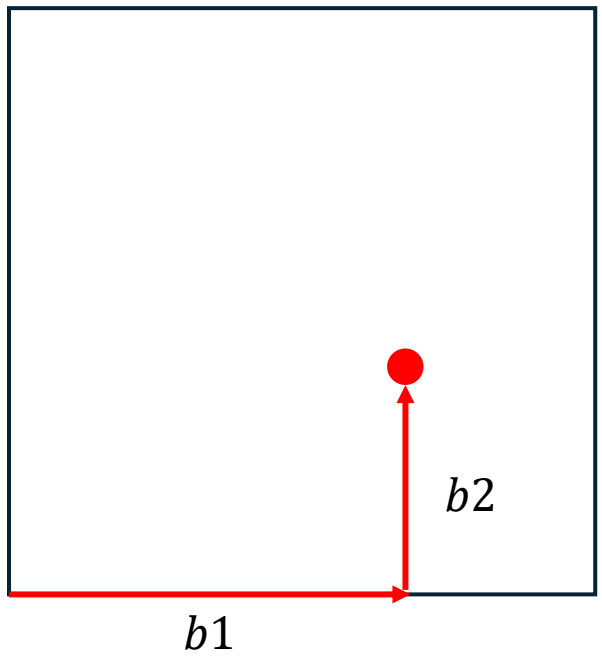


White Noise

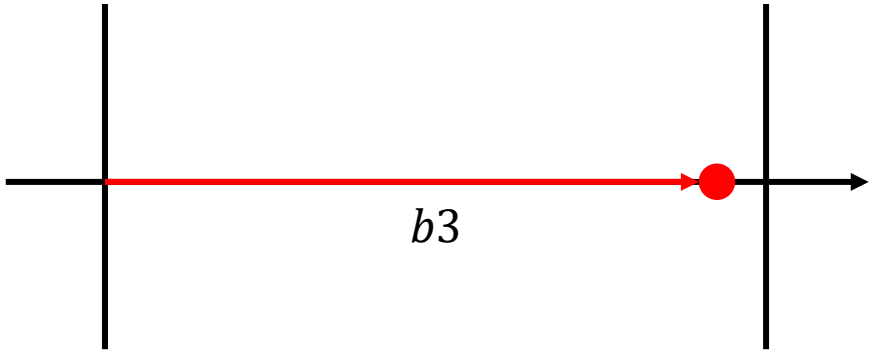


Anti-Aliasing

Pixel Sampling

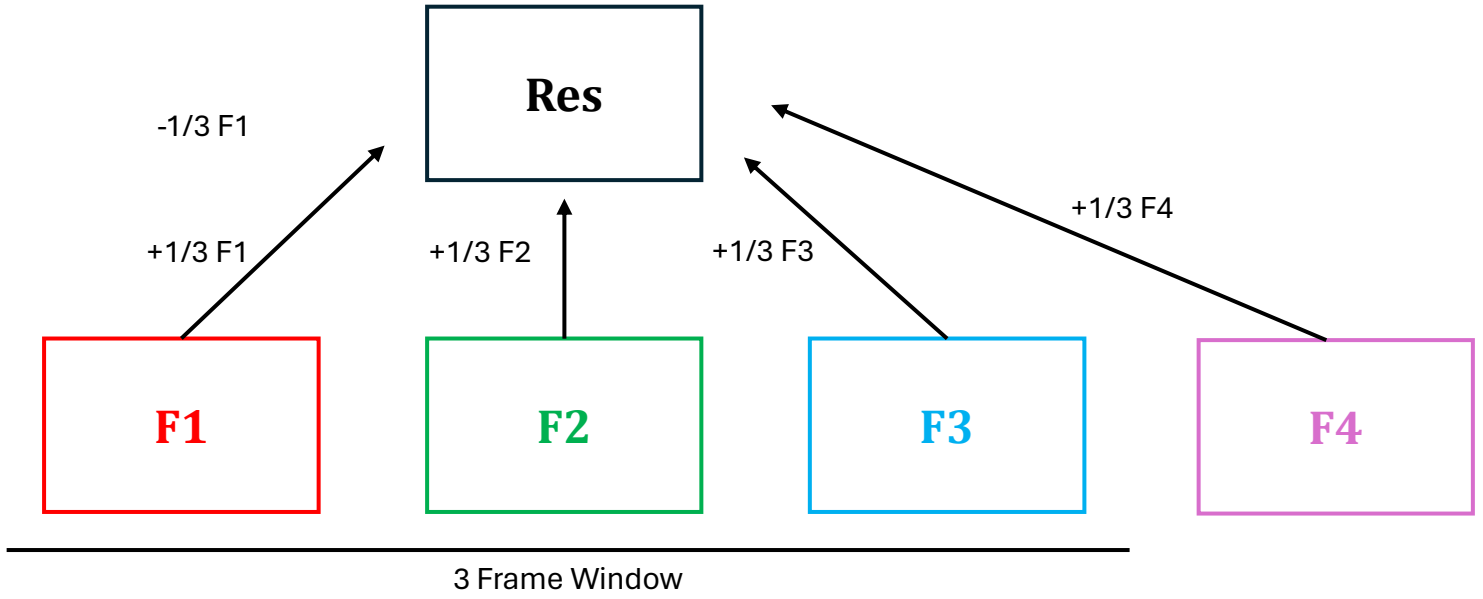


Segment Sampling



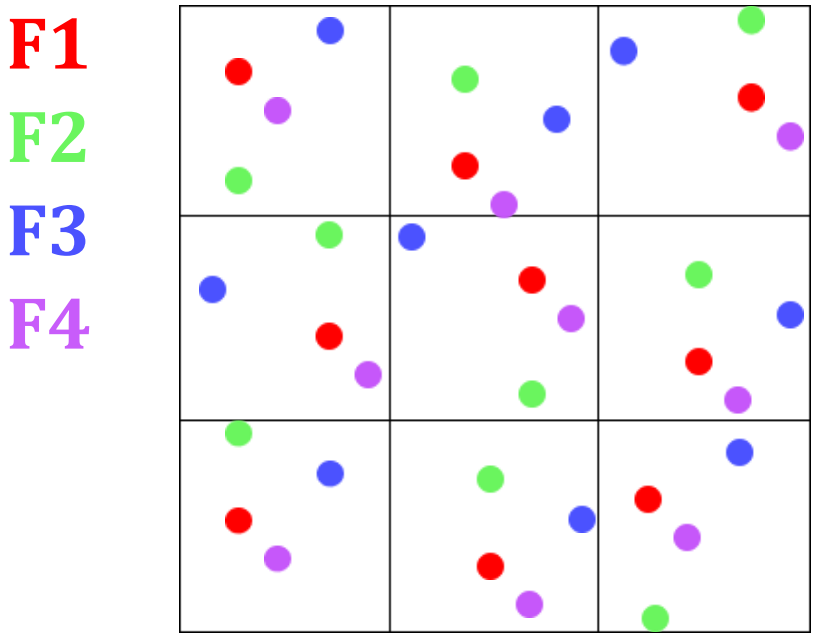
Unified Temporal Anti Aliasing

- Temporal Anti Aliasing refers to the combination of frames over the time domain.
- We accomplish this using a sliding window approach



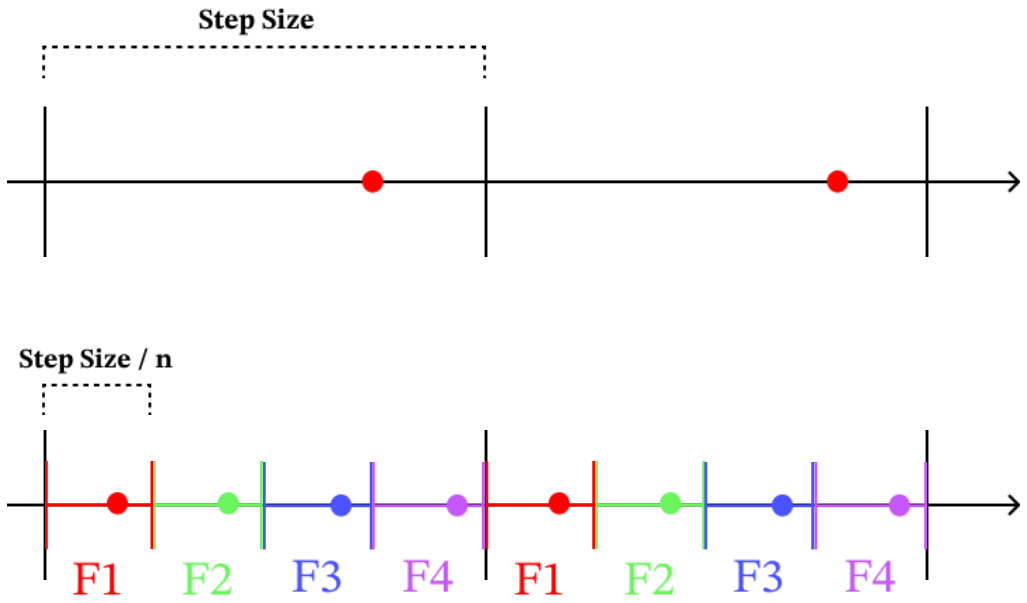
Pixel Area TAA

- Each pixel sample position begins at its blue noise offset, then is shifted according to pre-compute N-Rooks offsets



Volumetric TAA

- Each segment is split into smaller frame segments.
- Each frame samples within its corresponding frame segment.



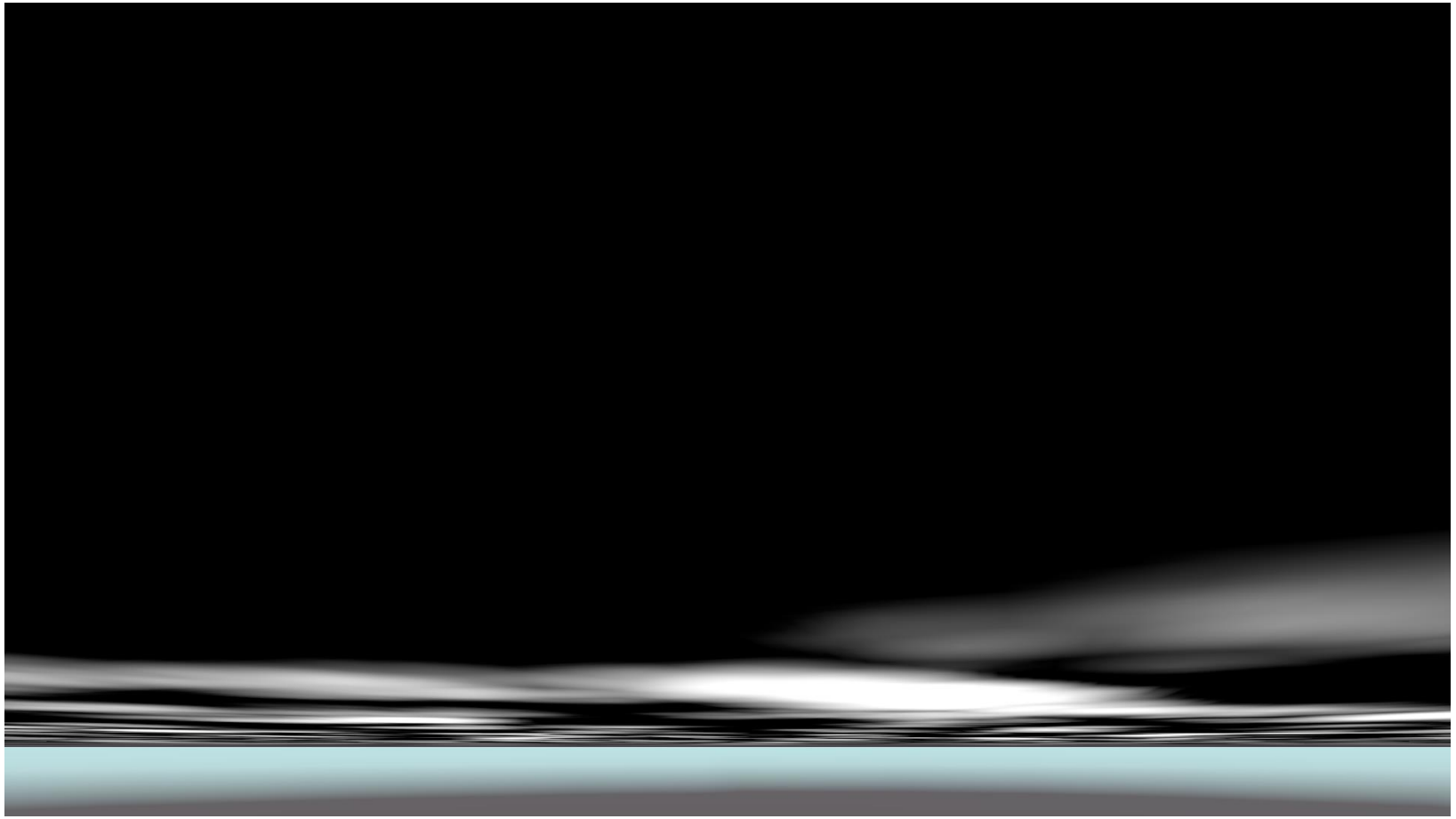
Summary

- We have presented our:
  - Generation of cloud map textures based on observations
  - Ray-march algorithm for approximating the radiance equation
  - Unified Temporal Anti-Aliasing solution that:
    - Unify pixel area and volumetric sampling
    - Capture additional volumetric detail without incurring additional computational cost
  - Use of Hillaire's improved scattering integration equation for more accurate scattering approximations over larger step sizes.

III. Results

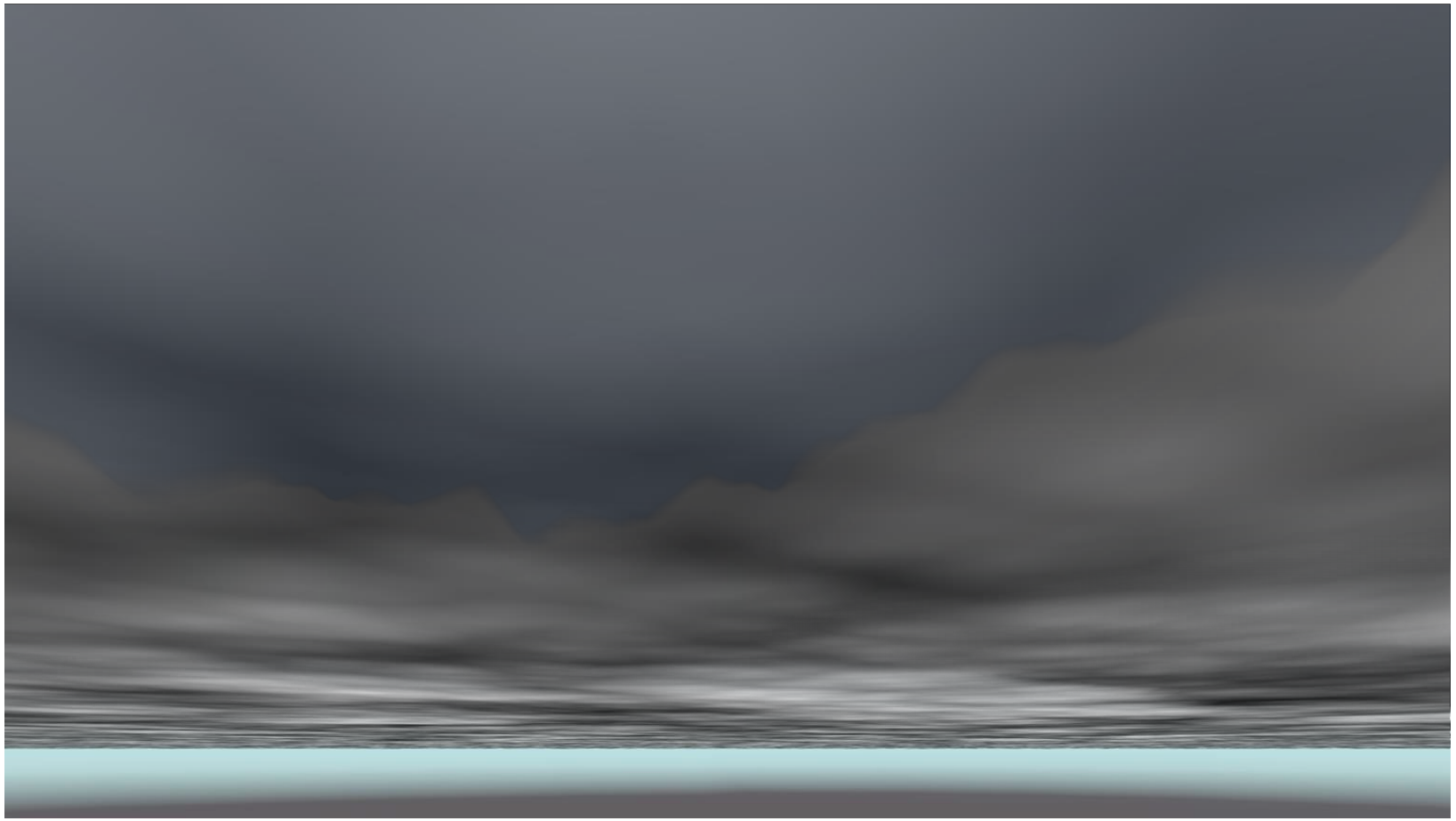
- Modeling
- Rendering
- Anti-Aliasing
- Optimizations

Coverage Map

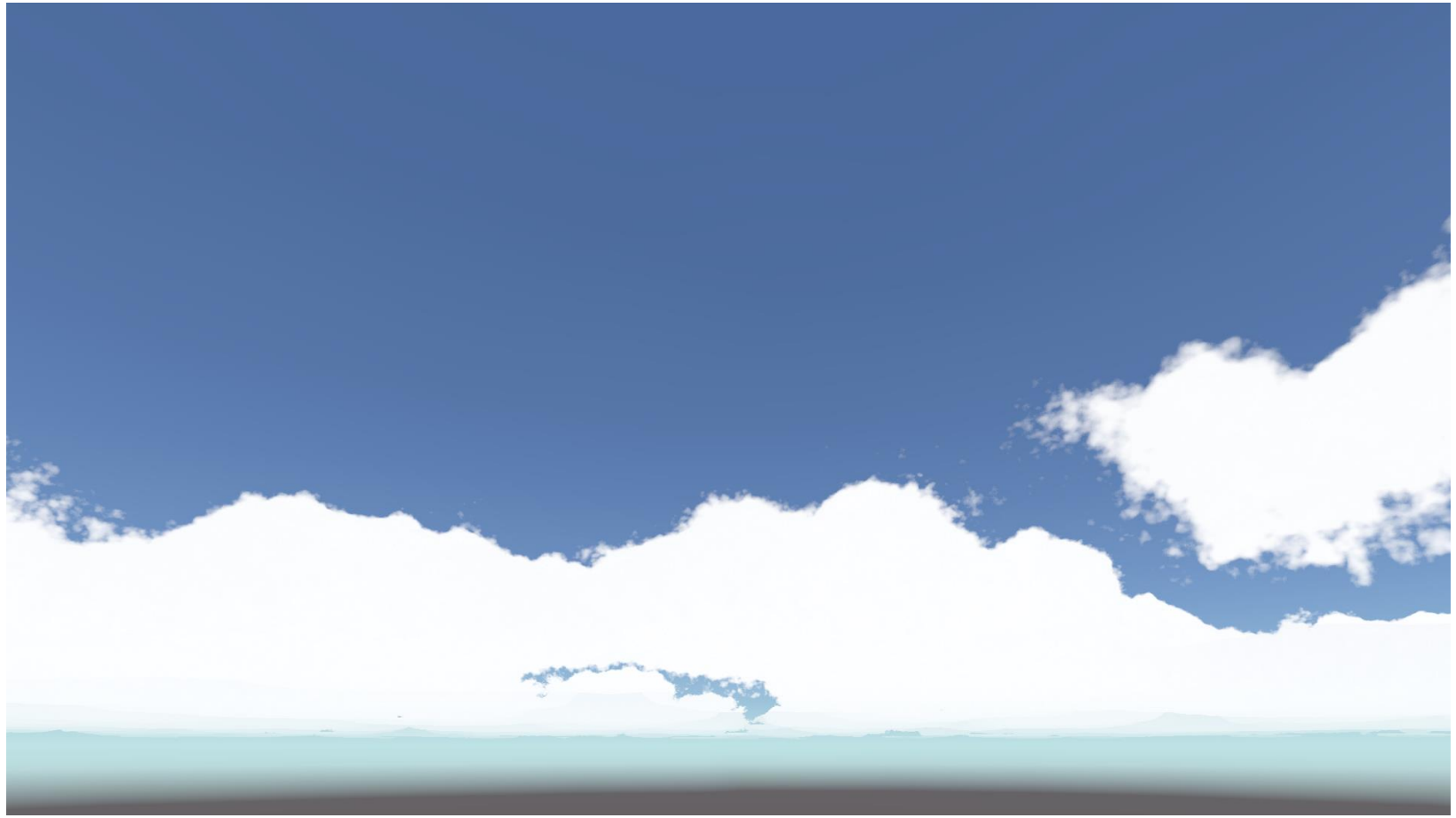


Type Map

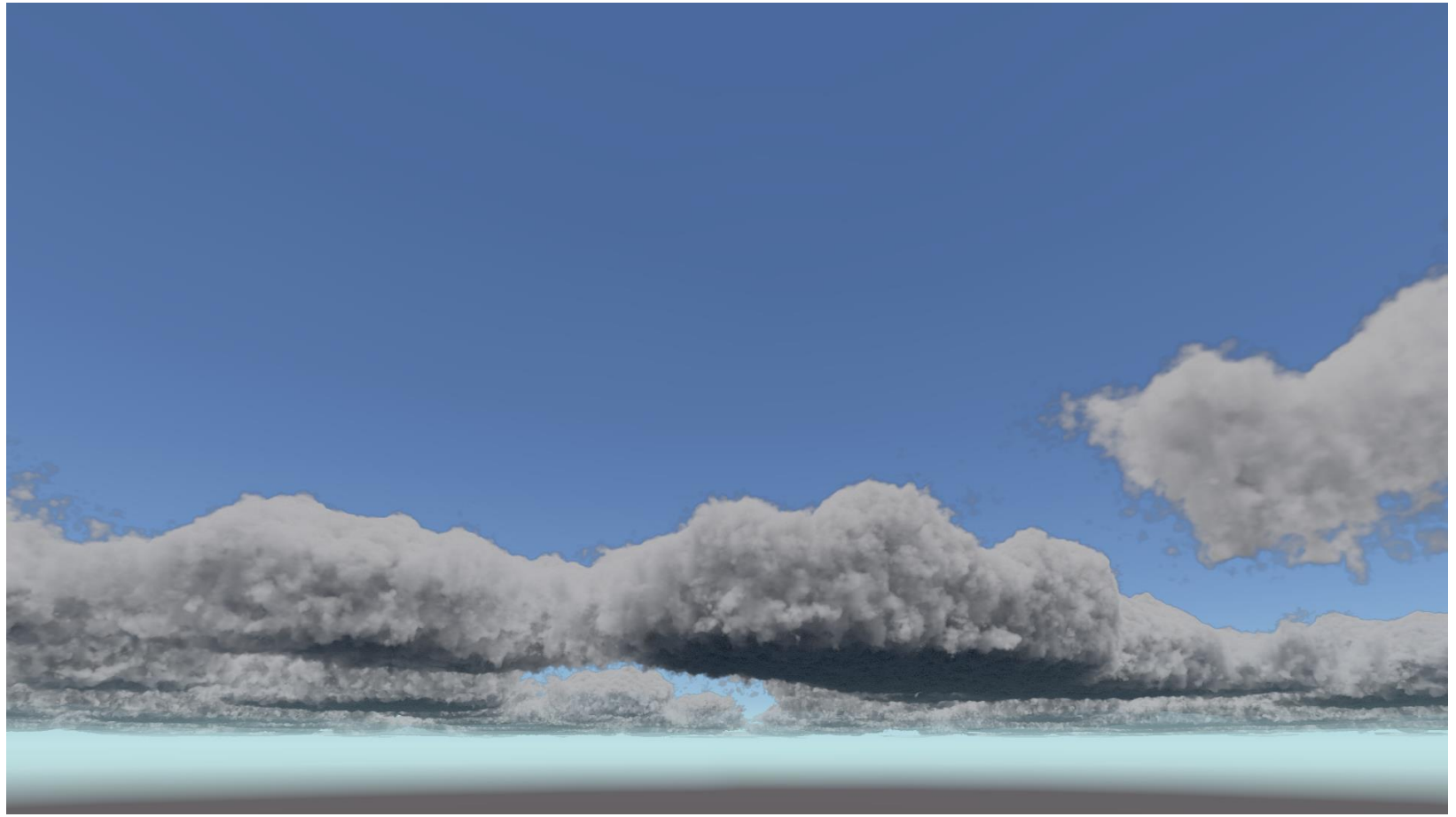
Dimensional Profile



Cloud Density



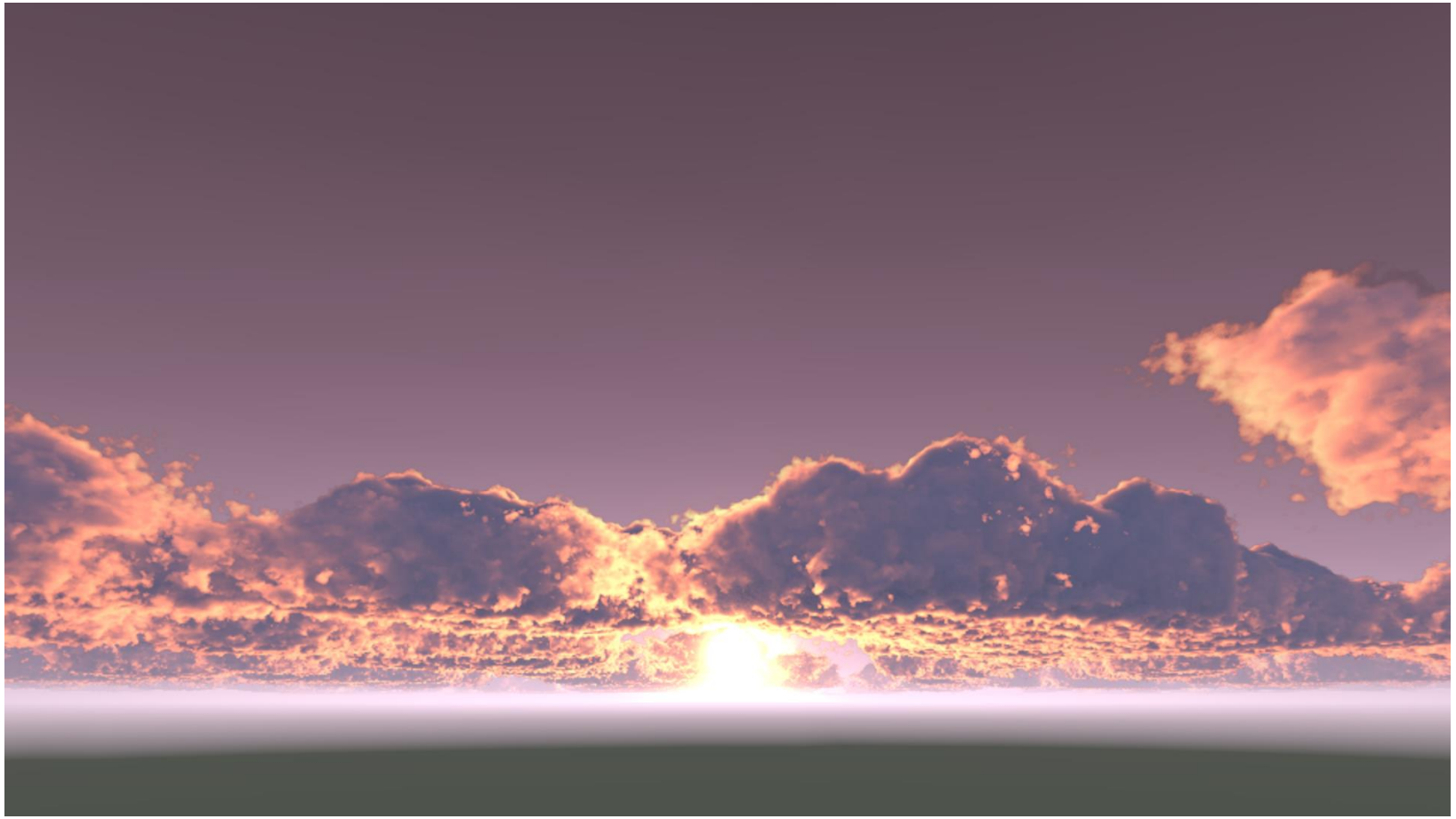
Single Scattering



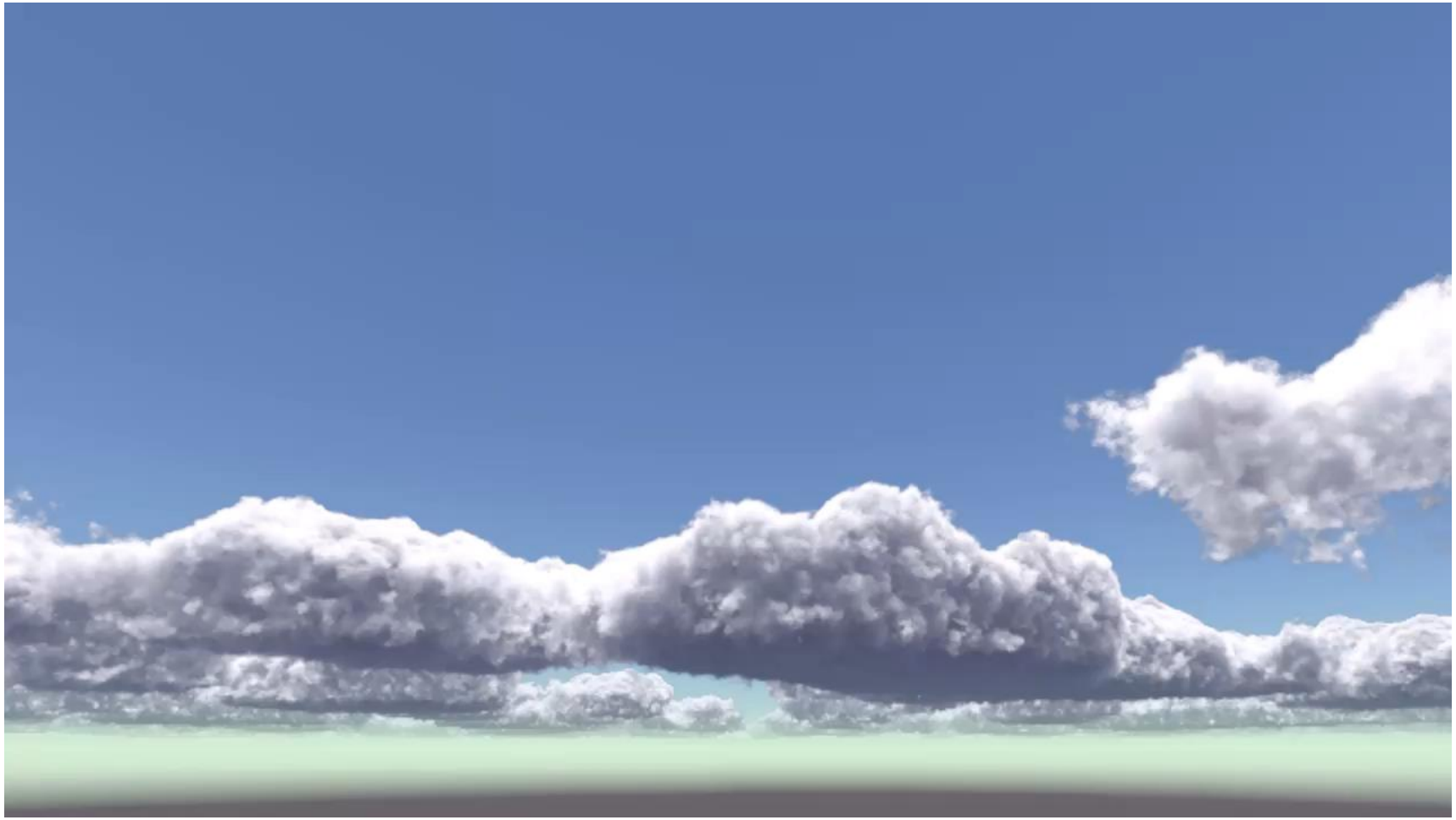
Multiple Scattering

Ambient Lighting

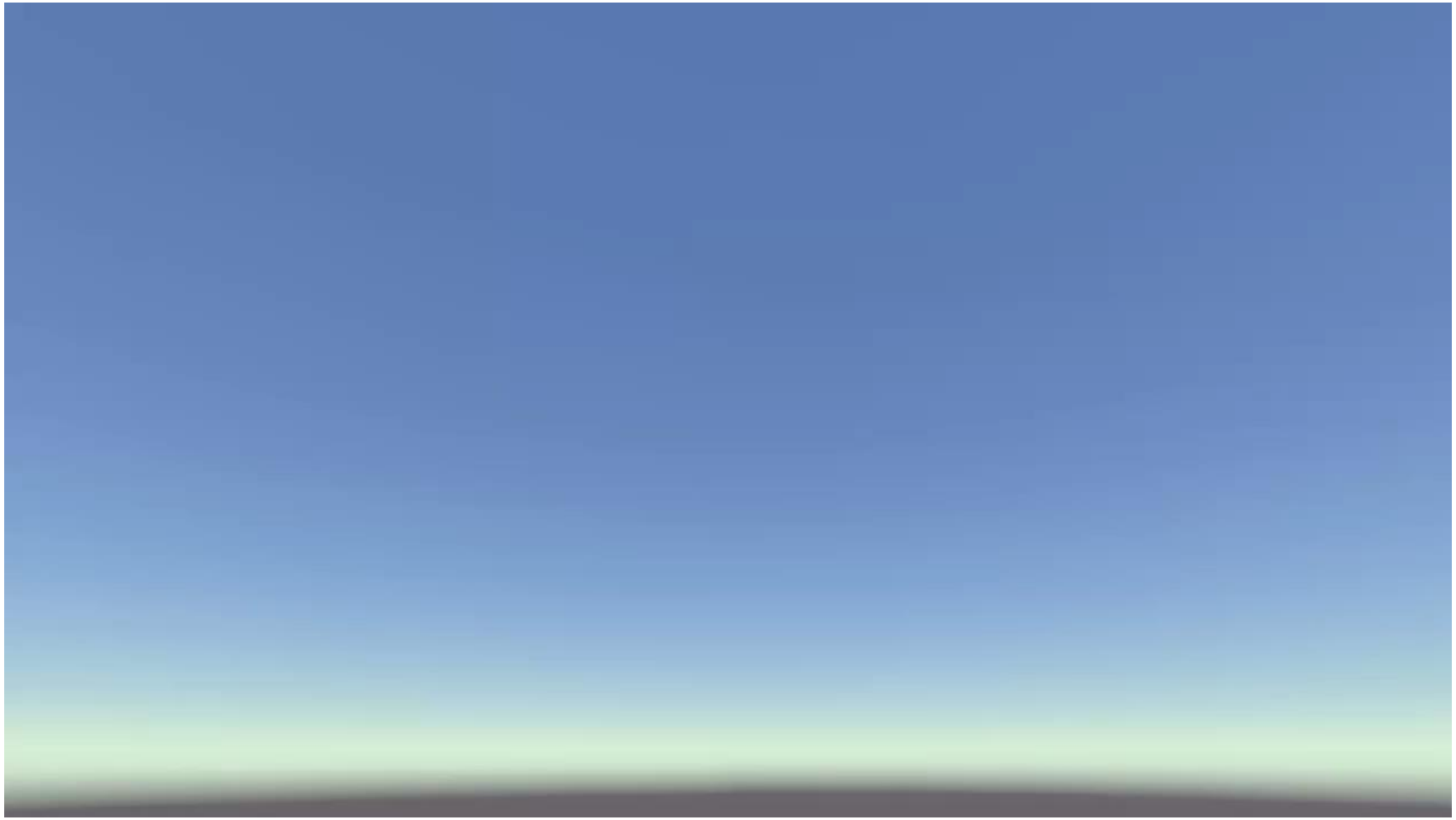
Phase Function



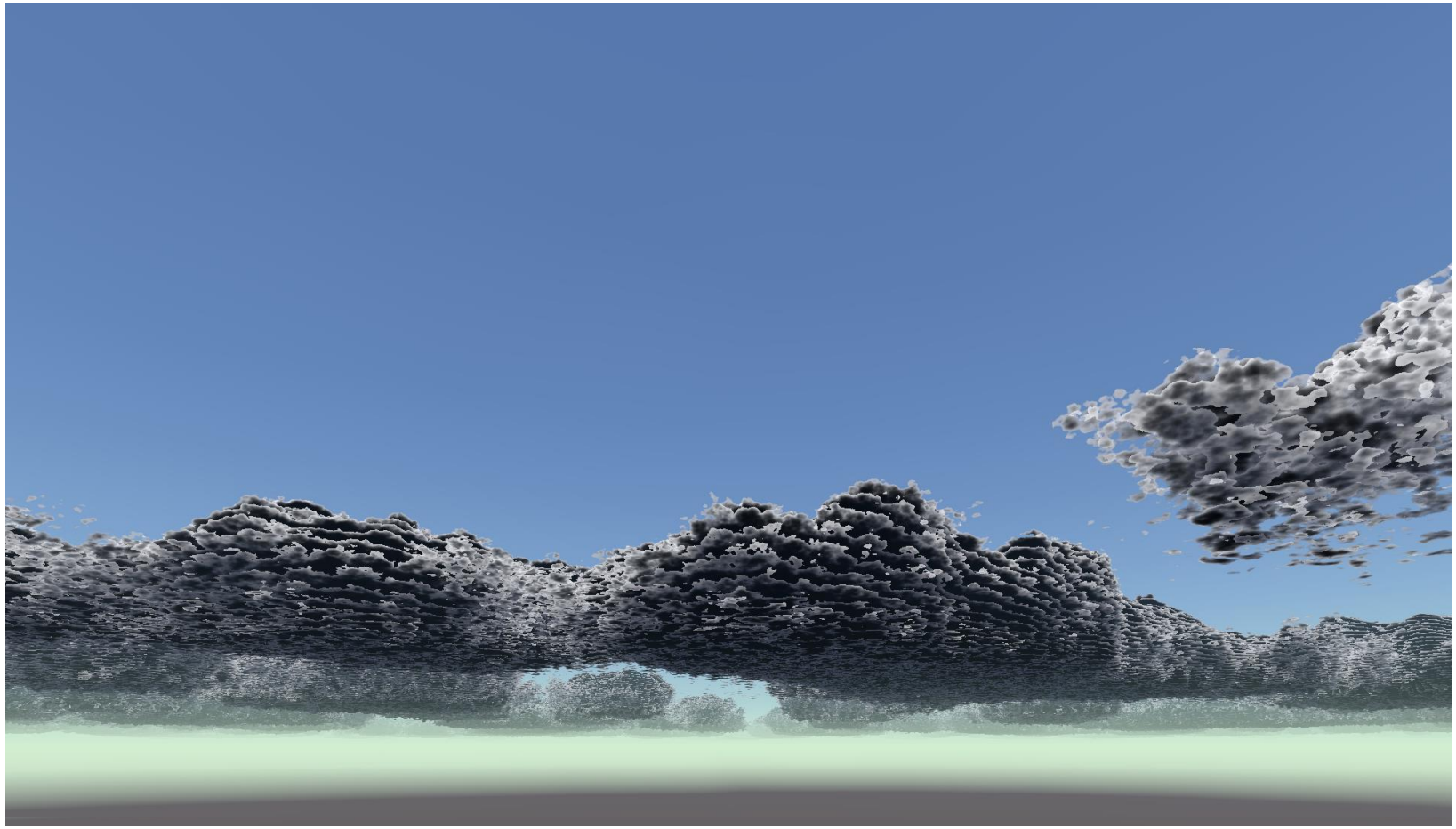
Dynamic Lighting



Cloud Animation

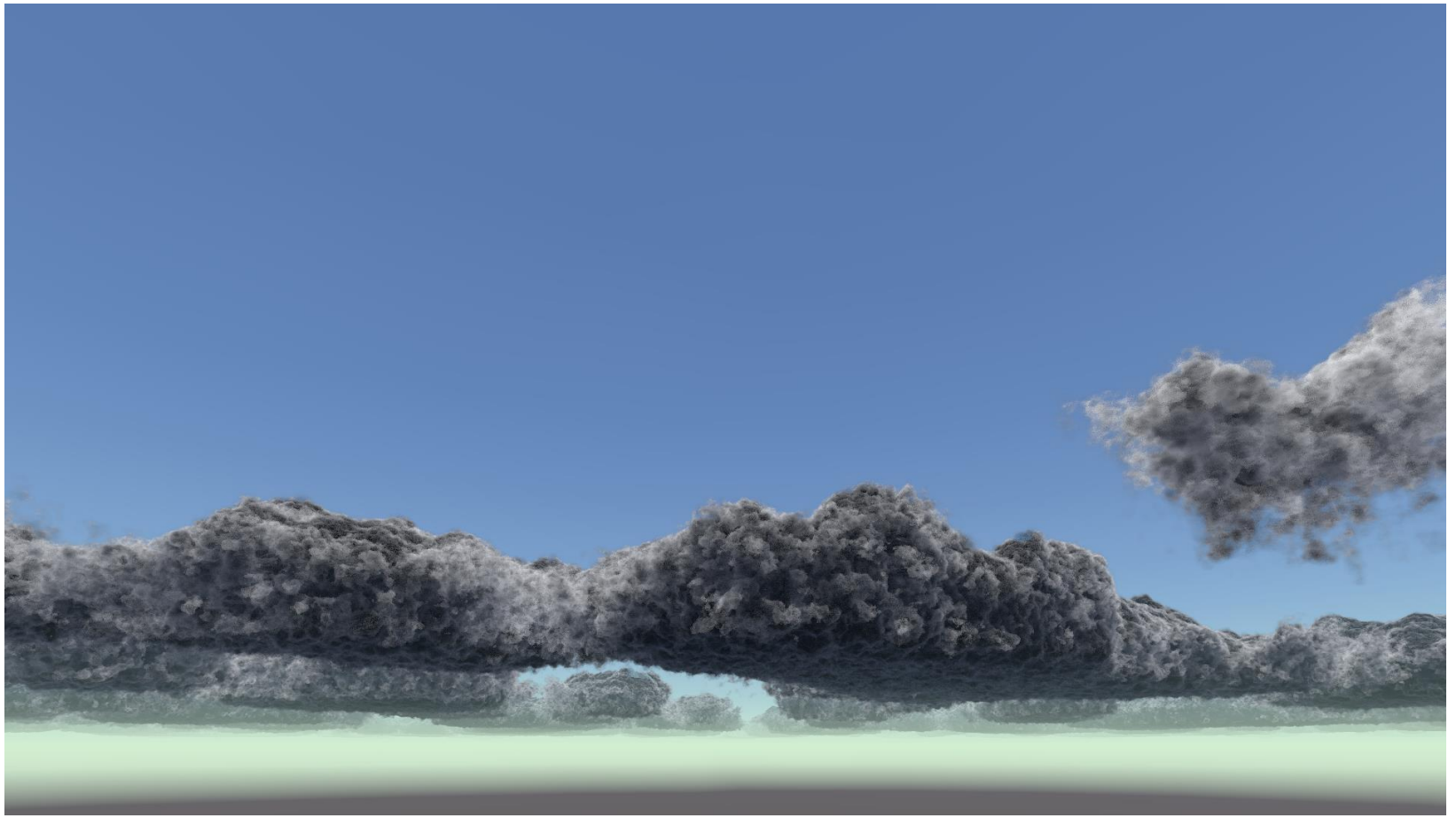


Anti-Aliasing

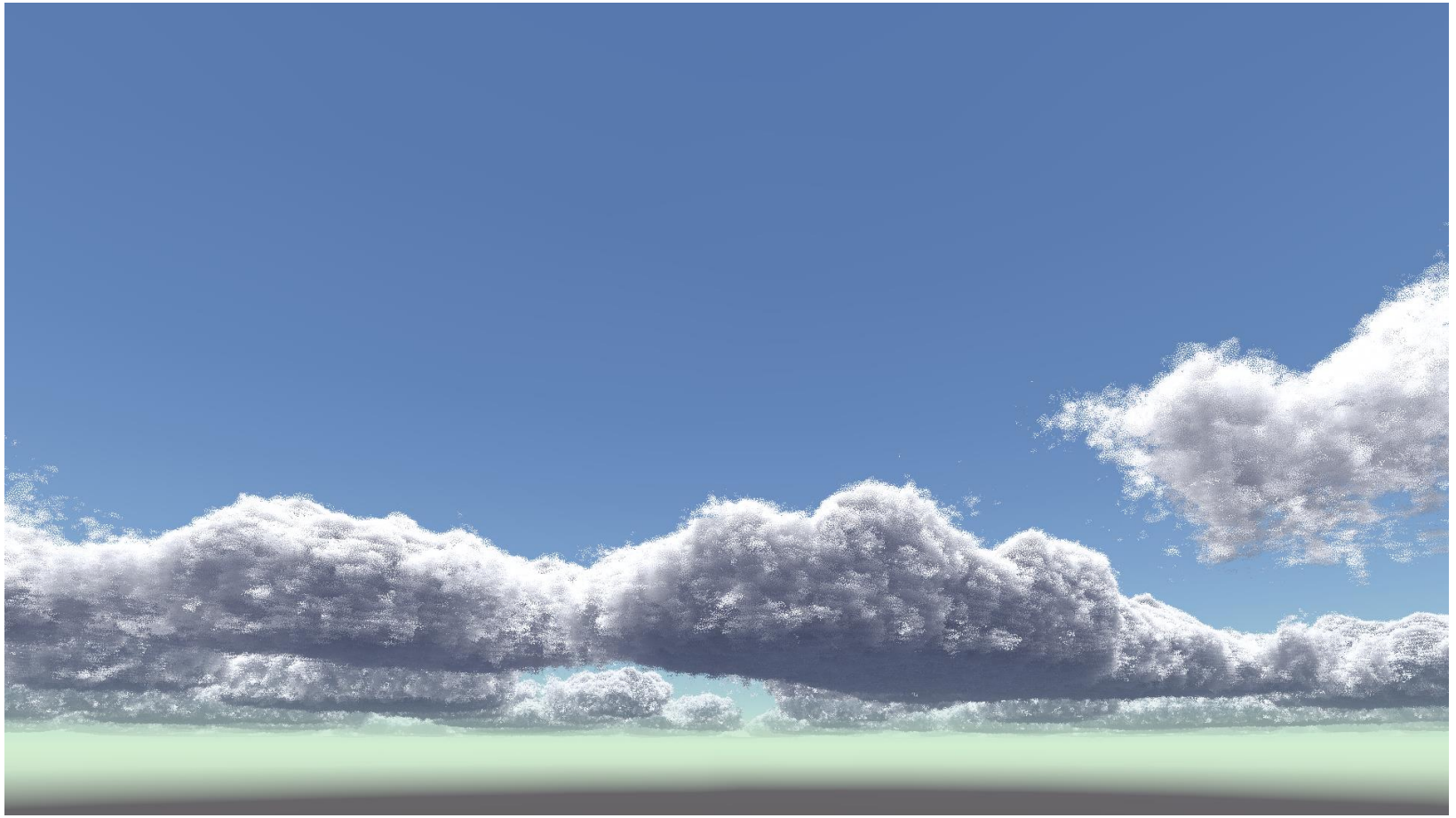


Anti-Aliasing

Anti-Aliasing

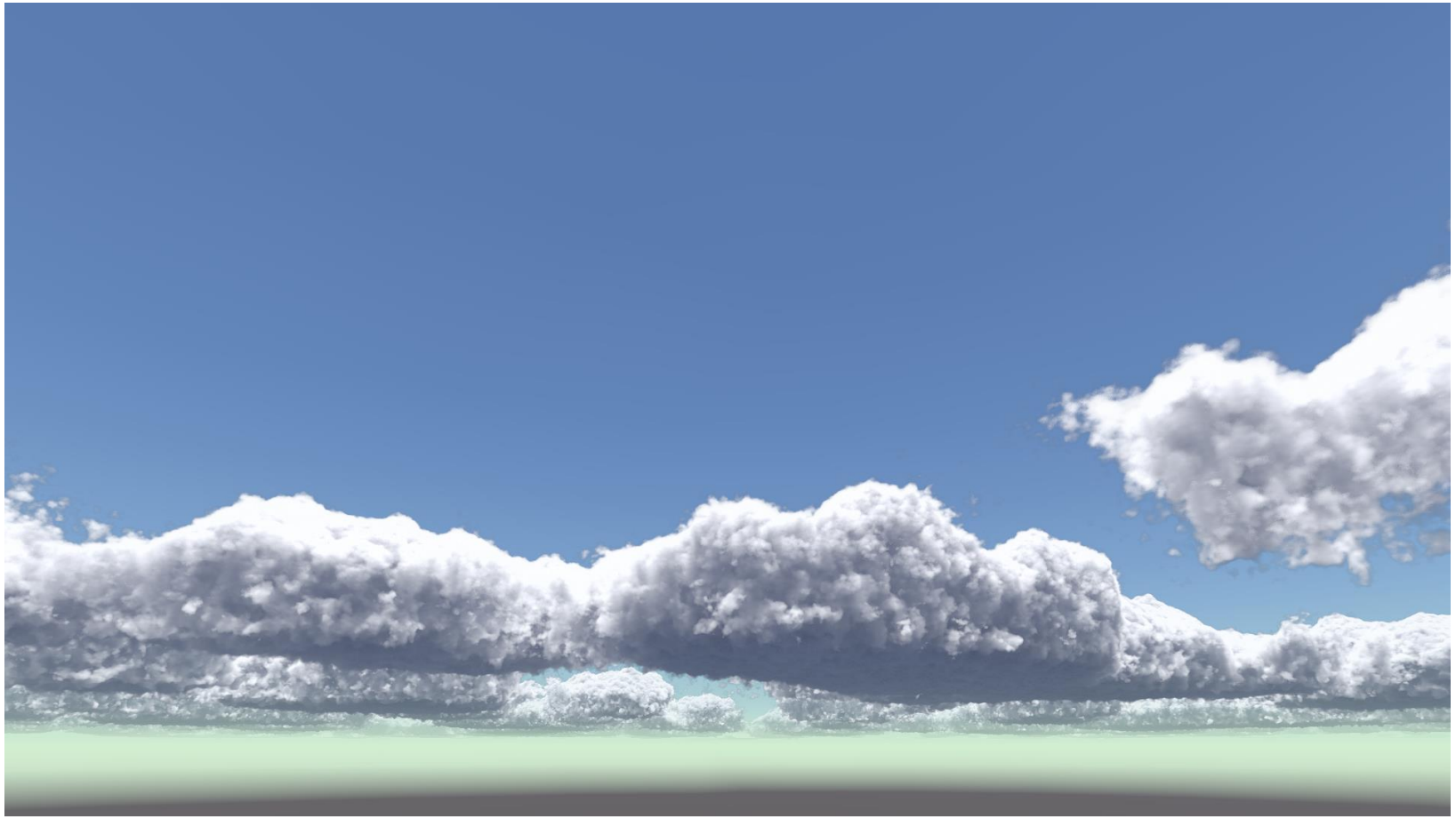


Improved Scattering Integration



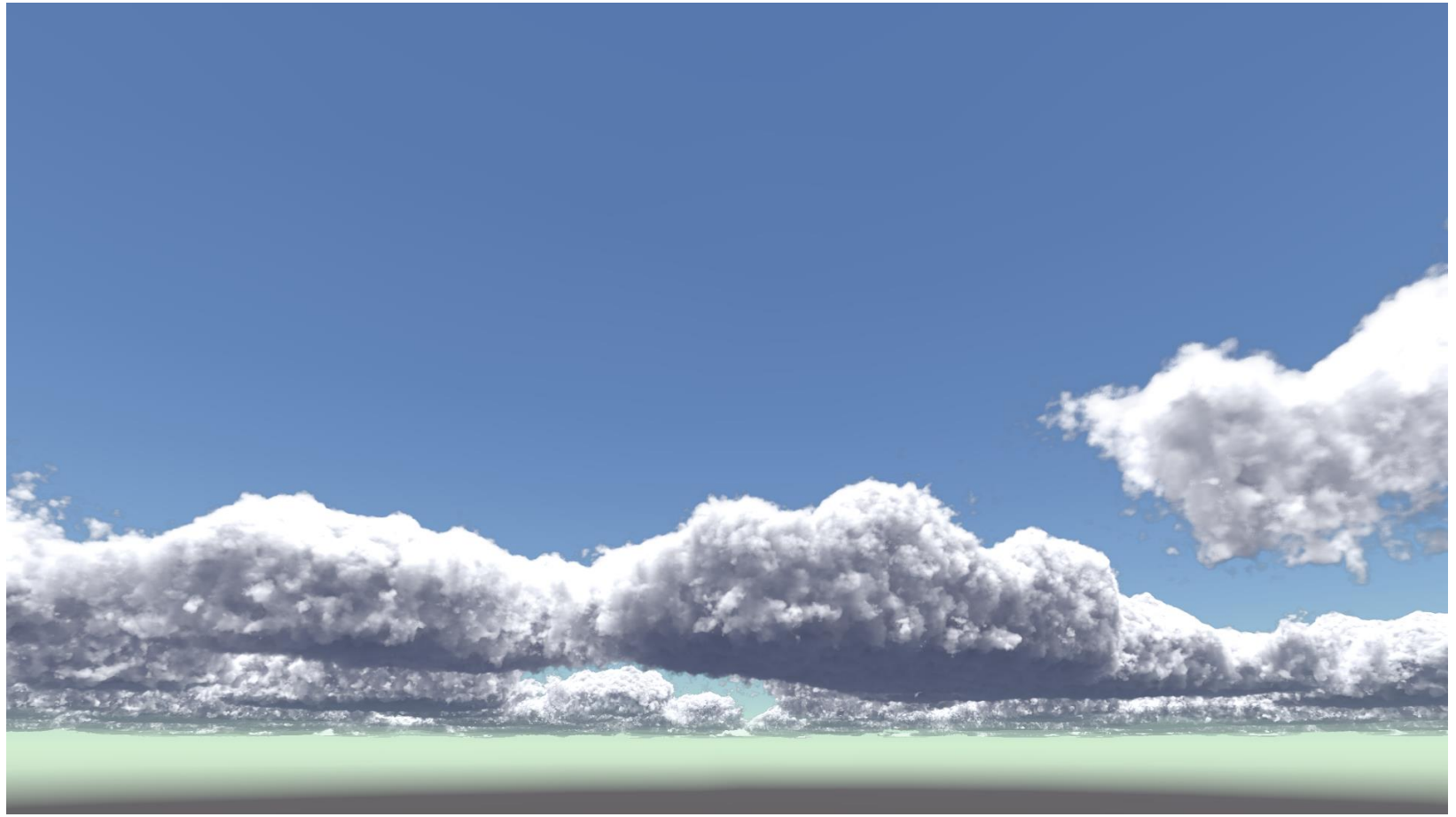
Temporal Anti-Aliasing

Temporal Anti-Aliasing



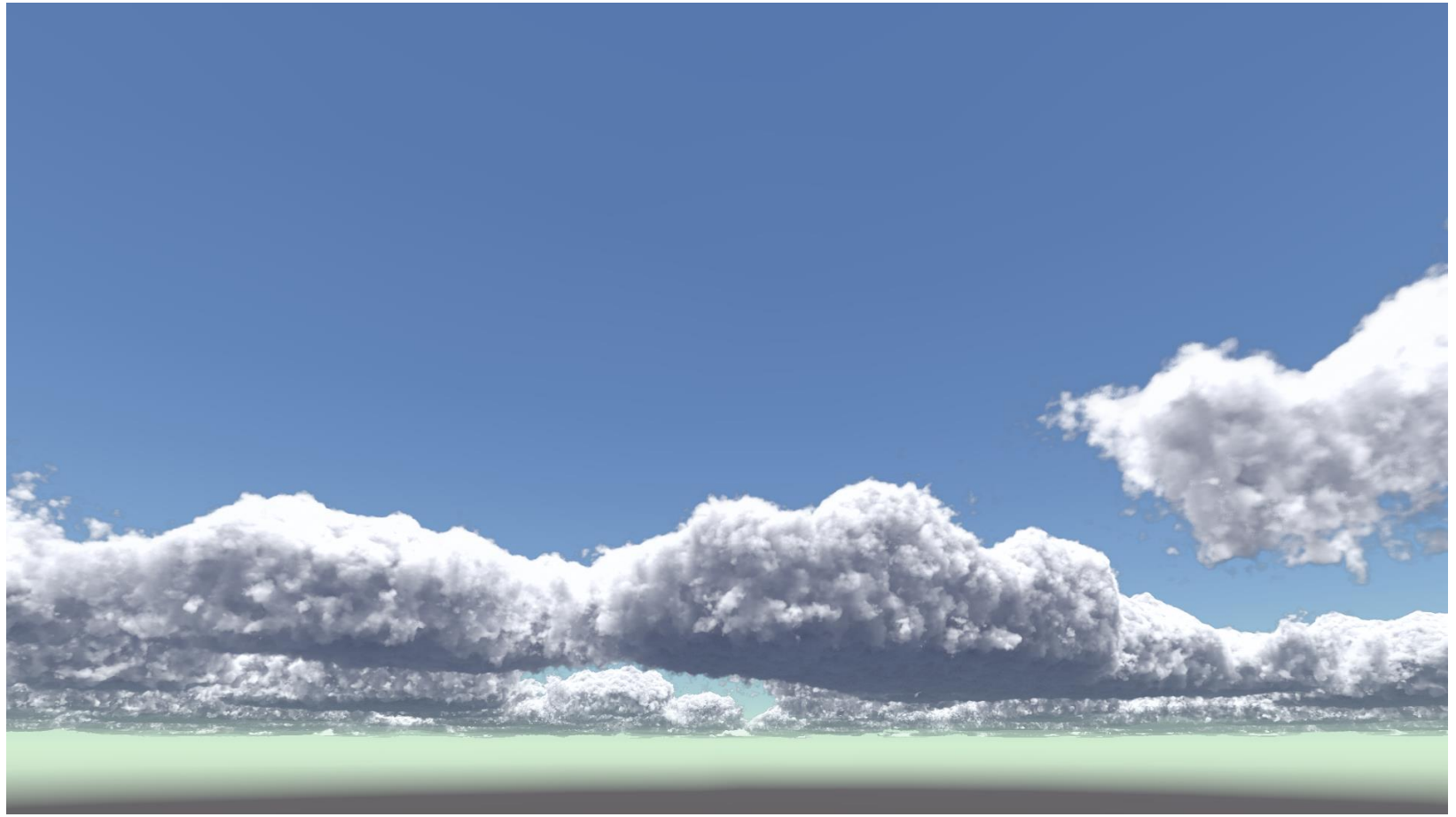
Reference Image

Step Size: 9
Run Time: 1558ms



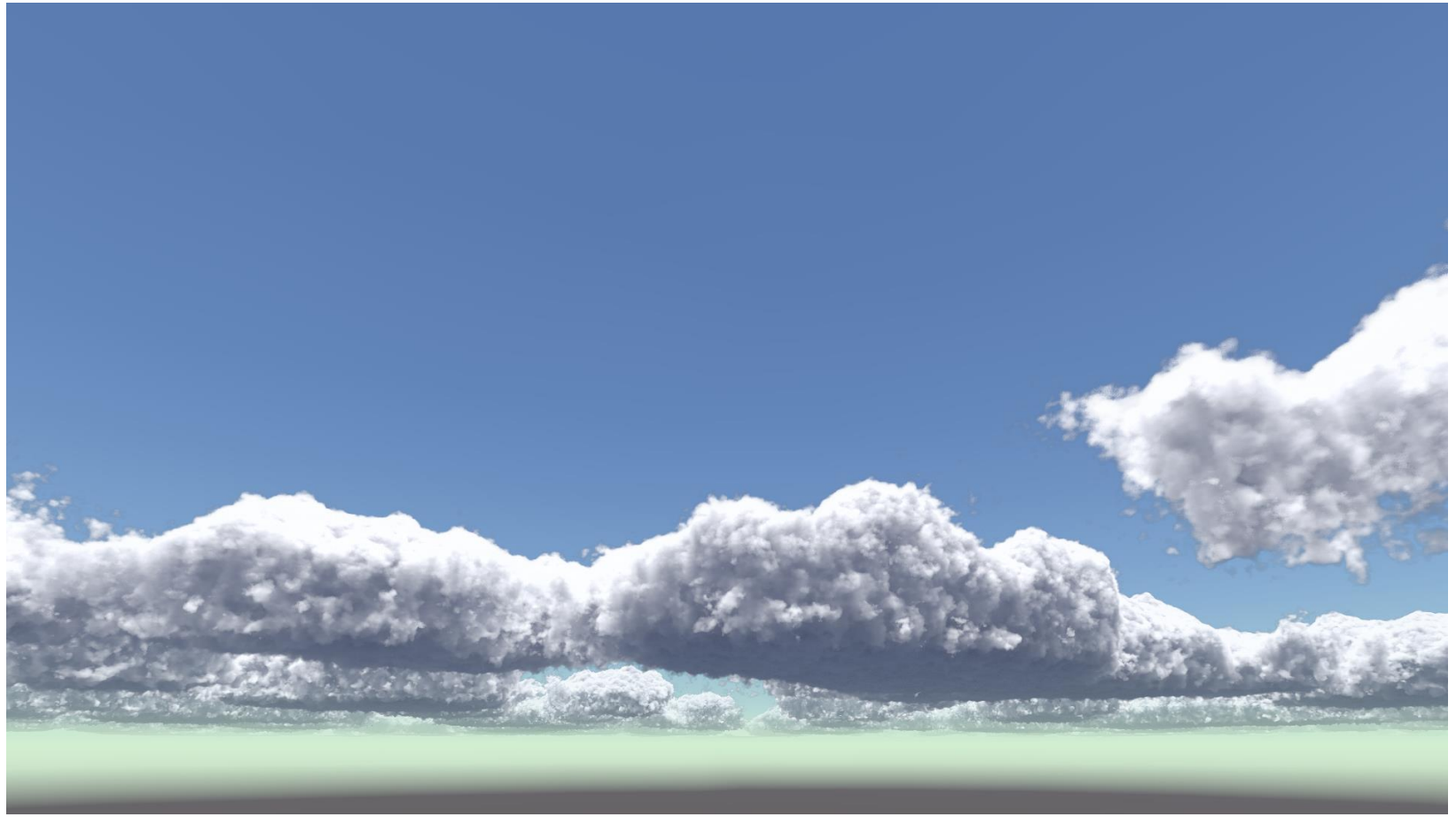
Early Exit

Step Size: 9
Run Time: 373ms
-1185ms



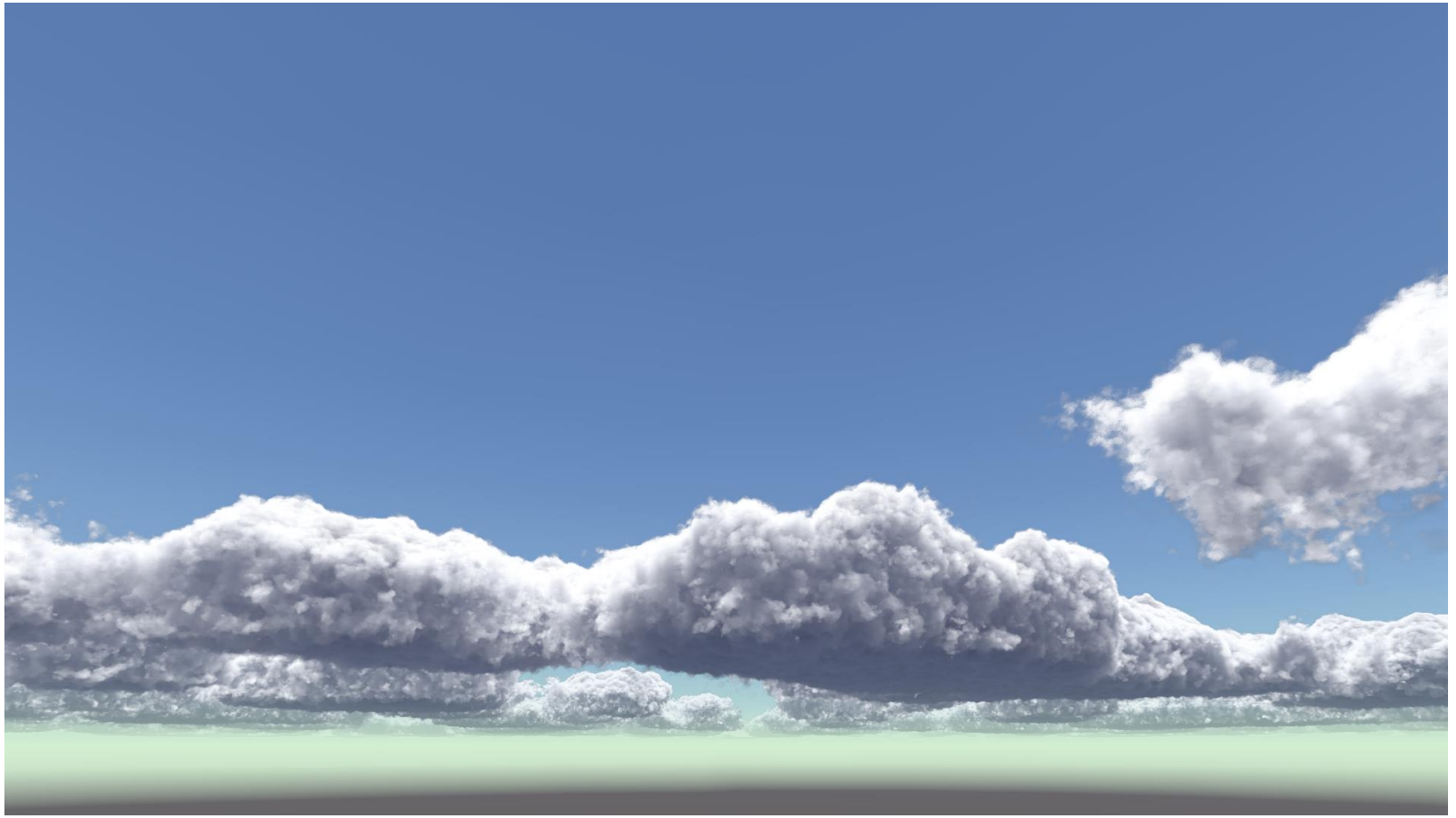
Skip Low Density Samples

Step Size: 9
Run Time: 34.6ms
-338.4ms



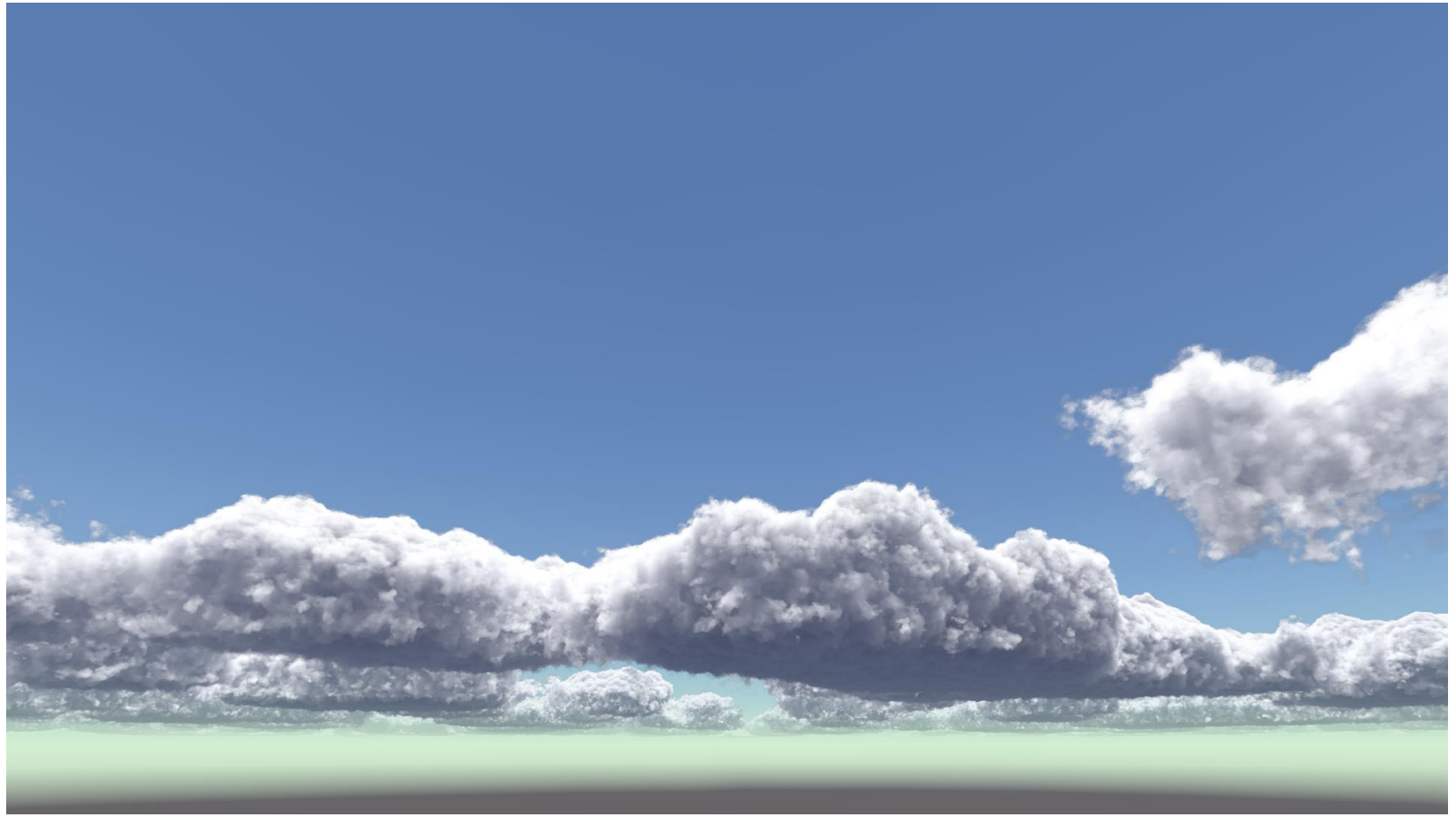
Temporal Anti Aliasing

Step Size: 100
Run Time: 6.5ms
-28.1ms



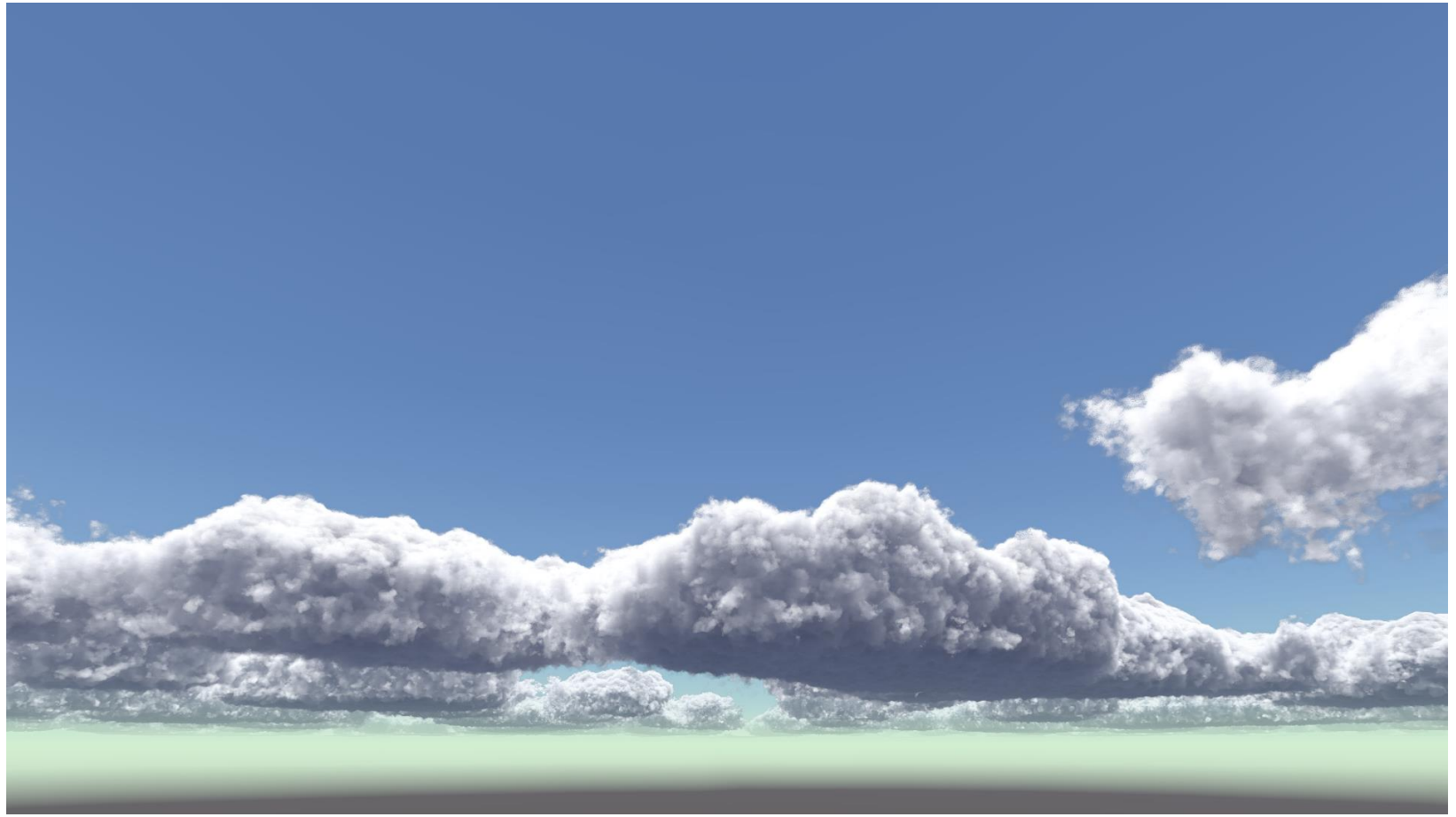
Variable Step Size

Step Size: 100
Run Time: 6.0ms
-0.5ms



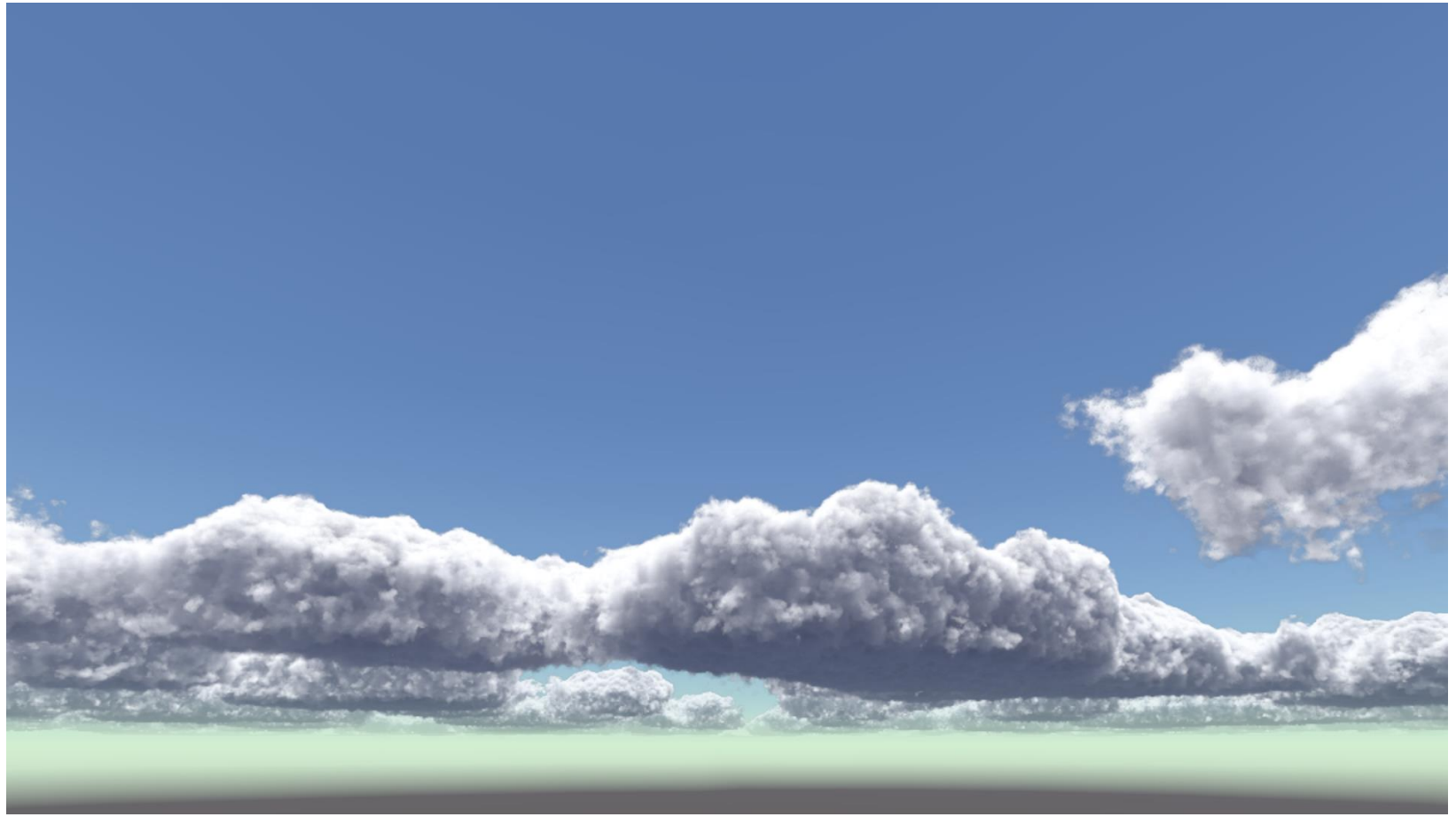
Full Resolution

Step Size: 100
Run Time: 6.0ms



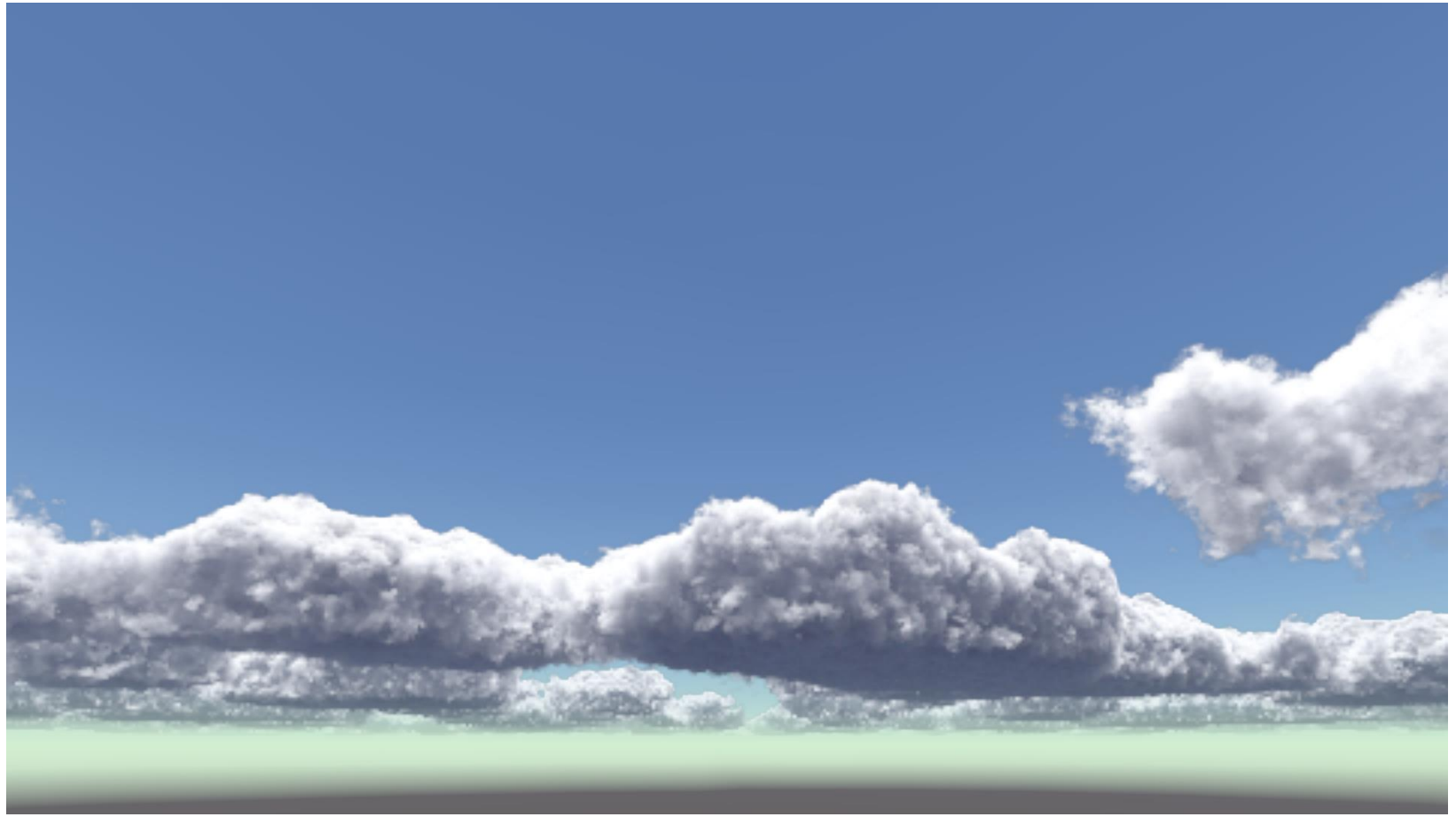
Half Resolution

Step Size: 100
Run Time: 2.7ms
-3.3ms



Quarter Resolution

Step Size: 100
Run Time: 1.8ms
-0.9ms



IV. Conclusion

- Goal Assessment
- Future Work

Conclusion

- We have provided a cloud rendering solution that:
  - Is capable of modeling the shape and complex details of cloud surfaces
  - Captures the lighting characteristics observable in clouds
  - Produces clouds that can dynamically change shape and react to lighting changes in their environment
  - Is capable of running at frame rates suitable for real-time applications
- Our work:
  - Case study for those implementing their own cloud rendering solution
  - A novel Unified Temporal Anti Aliasing strategy for volumetric rendering

Future Work

- Improvements to Cloud Map generation:
  - Maps that are generated based on cloud formation physics
  - Maps that realistically animate over time
- Interaction with environment:
  - Show clouds with other elements of environment
  - Clouds that cast shadows onto the environment

Thank you!

Questions?