Decima Engine, 2023

NUBIS^3

Methods (and madness) to model and render immersive real-time voxel-based clouds.

Andrew Schneider / Atmospherics Lead / Guerrilla

Advances in Real-Time Rendering in Games Course

SIGGRAPH 2023
LOS ANGELES+ 6-10 AUG

Decima Engine, 2023

Nubis³ / A Brief (Nubis-Centric) History of Cloud Rendering

Clouds in the skies of *Rio*

Andrew P. Schneider   Trevor G. Thomson   Matthew S. Wilson
Blue Sky Studios*

Figure 1: *Blu and Jewel hangliding through the clouds over* Rio. Rio© *2011 Twentieth Century Fox Film Corporation. All Rights Reserved.*

1 Introduction

We faced four major challenges when creating the clouds and skies in *Rio*. The first challenge was making volumetric clouds that could be rendered in stereo. Previous approaches involved matte paintings and 2D cards, which lacked parallax and could not be lit correctly. The second challenge was creating an efficient workflow to quickly place clouds into lots of scenes. The third challenge, was producing highly-detailed clouds when they got close to camera. And the last major challenge was finding a way to efficiently ray trace all the clouds while keeping the memory footprint small and the render times short.

2 Synthesizing Clouds

We created three types of clouds: cumulus (heavy billowing clouds), stratocumulus (bandy, puffy, wispy clouds), and stratus (bandy and wispy). We started by building their general shape using spheres. These spheres were then converted to a into a volumetric grid of density values (i.e., voxels), and passed through a set of noise deformers that convected the data in billowy and wispy patterns. Once the desired look was achieved, we added a deformer that evolved the data based on the wind shear, density gradient, and turbulence. This allowed us to precisely control the look of the cloud and make adjustments based on notes from the art director.

3 Workflow

Once a cloud was approved by the director, its high resolution voxel representation and a RGB rendered turntable were committed into a library that we could draw from in three ways.

For close up shots, where a high level of detail was needed, we used the high resolution voxel representation. These voxel grids had resolutions of at least 300xNxN. We placed the cloud into the scene using Houdini, sometimes combining it with other clouds, according to art direction. Stretching, skewing and billowing evolution was added to the formation using proprietary volume deformers and solvers. Once the desired look was achieved, we wrote out different voxel files for each frame. The renderer would then use this sequence of files to render the clouds for the shot.

For medium to distant shots, where less detail was needed but parallax was still important, we imported the voxel representations of the clouds into a scene, modified them, and wrote a low resolution version of each cloud to disk. The resampled resolution was arrived at interactively inside of Houdini based on each clouds distance from camera. In this case, evolution was applied at render time by skewing each voxel grid and deforming the noise coordinates according to wind direction and speed. For long sequences like the one where Blu and Jewel fly over Rio (Fig. 1), we placed all of the clouds into a master set, and then adjusted as needed per shot.

Finally, for distant shots, where clouds were so distant that parallax was of no concern, RGB renders of assets from the cloud library were placed in 3D composite space and then relit in Nuke. These cloud cards were then combined with rendered versions of other 3D clouds in the shot into a ”sky set” that could be shared across multiple shots and sequences. We developed a number of tools to color the clouds in harmony with a proprietary, pseudo-volumetric sky generator, that allowed us to maintain tight control over art direction and time of day.

4 Volumetric Rendering

All of the cloud rendering for *Rio* was done using SmogVox, which is a part of to CGI Studio, our proprietary ray-tracing renderer. SmogVox, which renders many types of volumetric effects such as smoke, steam, and dust underwent a series of modifications and optimizations for *Rio*.

To improve the look and efficiency of the SmogVox procedure, we did four things. First, we modified the scattering model with an efficient algorithm to simulate the effects of multiple scattering at no additional cost to the render. Calculating all of the paths taken by refracted light in a cloud of water droplets would be much to expensive for production, but the illumination of the interior of a cloud is an essential part of its character. Second, we optimized the raymarcher to compute more optimal step sizes based on the volume's density and density gradient. This sped up ray tracking in situations where the ray passed through varying densities of cloud material. Third, we computed the most visually optimal points to throw shadow rays and calculate the lighting. This saved us time by not throwing shadow rays into regions of the clouds that we could not actually see. Finally we changed the way that we add detail to our volumetrics. For *Rio*, we rendered detail at a higher frequency than the voxel resolution via two user controlled noise functions that allowed us to sculpt the density based on position, density, density gradient, velocity, or other user-defined fields. These additional controls for noise were key to achieving the high-detail wispy and billowy regions of the clouds.

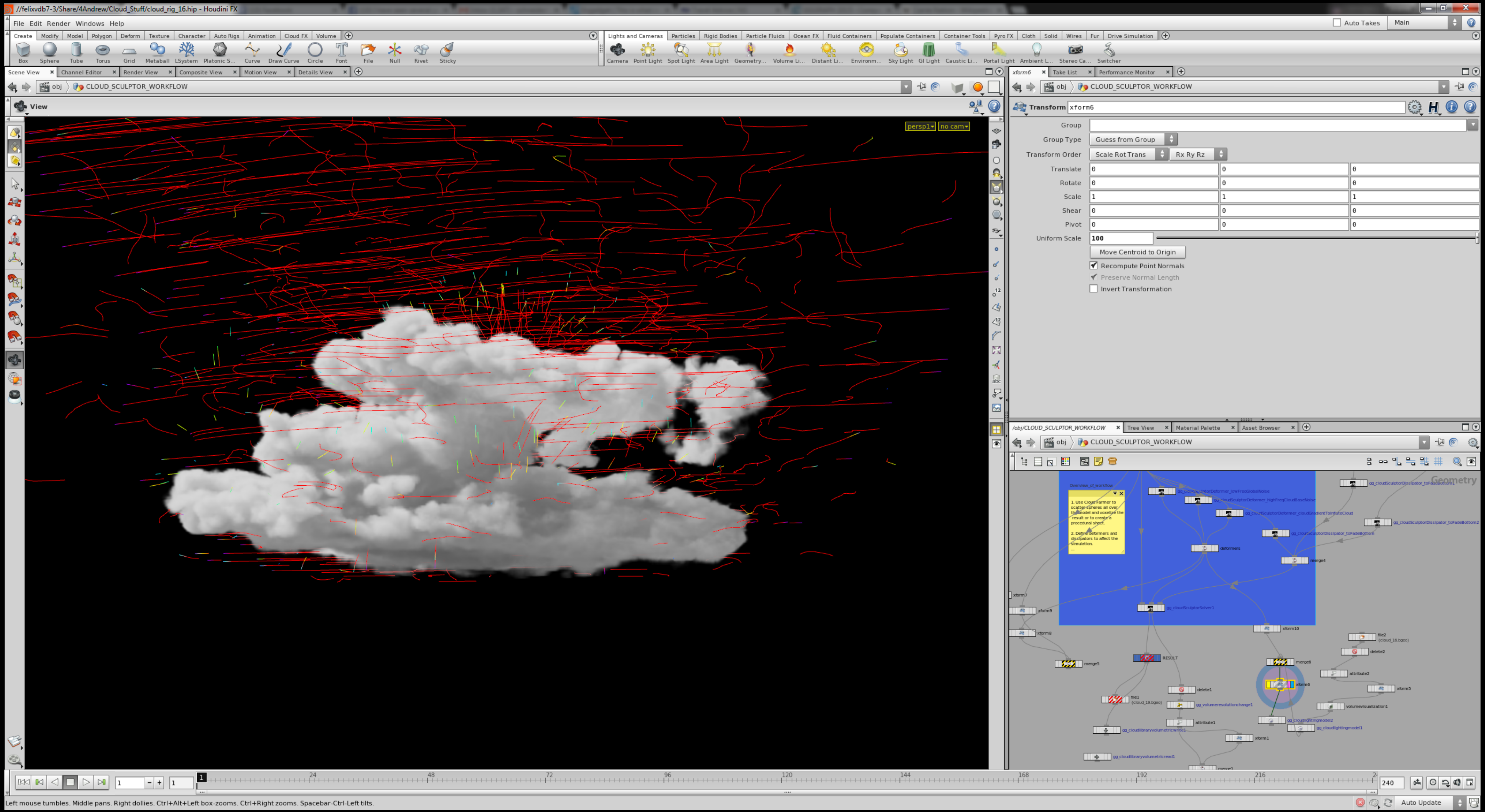
5 Conclusions

We completed over 534 shots requiring skies for *Rio*, 96 of which included 3D clouds. The most complicated cloud set included 20 3D and more than 30 2D clouds. Our multiple scattering technique added no additional time to the render. Render times for distant smaller clouds averaged 10 minutes for a 1920x1080 image and large high detail clouds averaged 4 hours at the same resolution. Memory usage averaged 500 MB for low resolution clouds and 4 GB for high resolution clouds.

*E-mail: {aschneider,tgt,mwilson}@blueskystudios.com

SIGGRAPH 2011, Vancouver, British Columbia, Canada, August 7 – 11, 2011.
ISBN 978-1-4503-0921-9/11/0008

Decima Engine, 2013

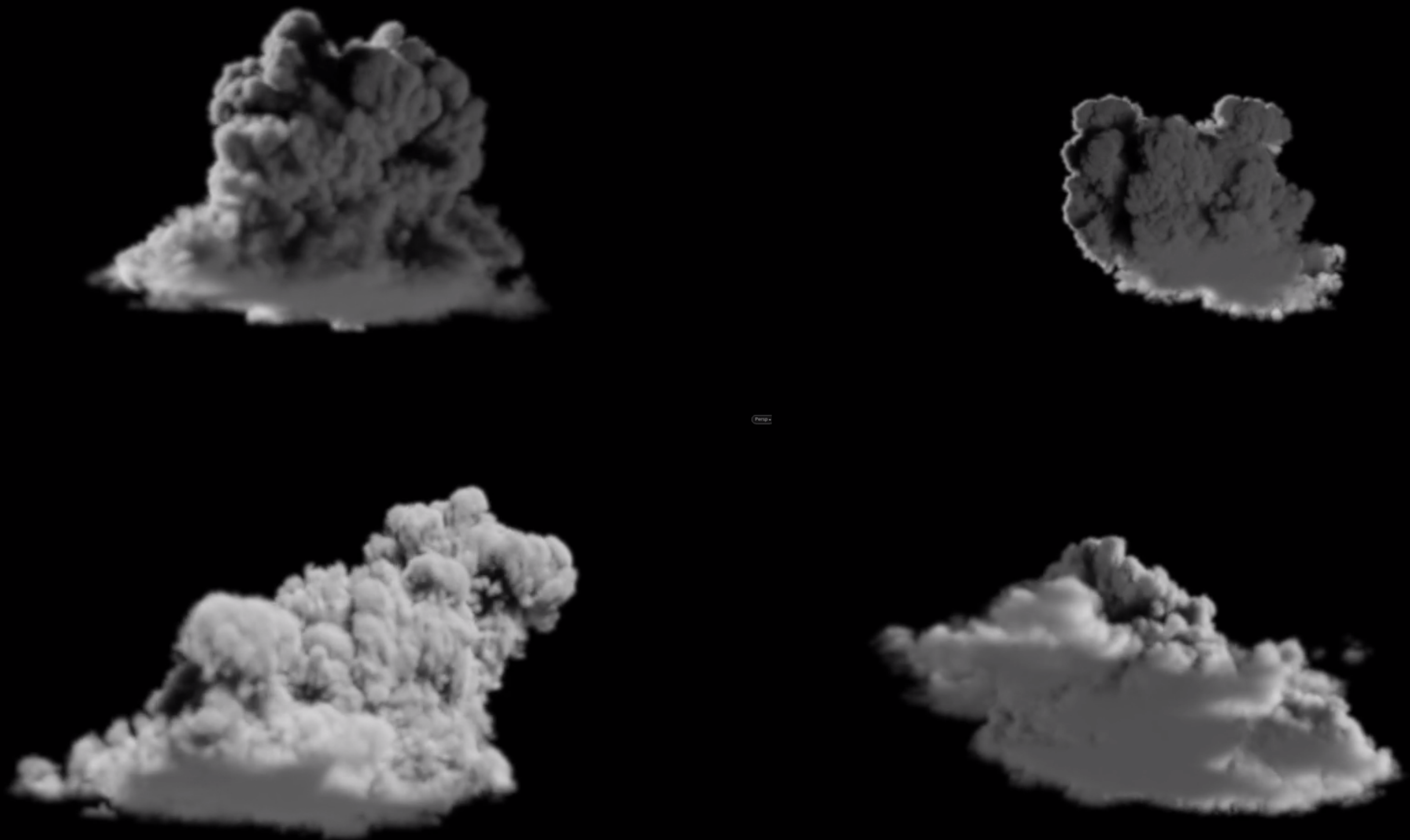


Aero Cloud Solver:

(one frame)

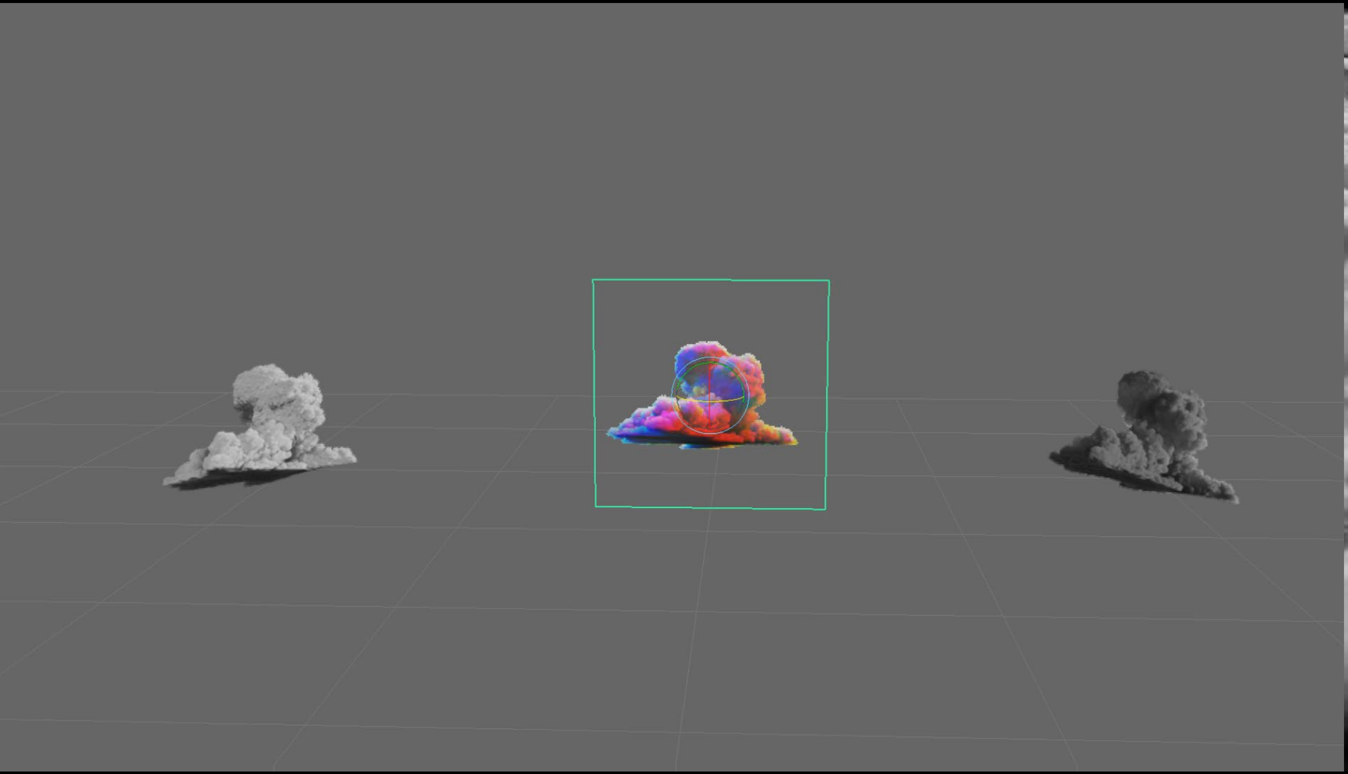
Apply Buoyant Force | Advect Density | Advect Velocity | Non-Divergence | Sourcing

→ Solve

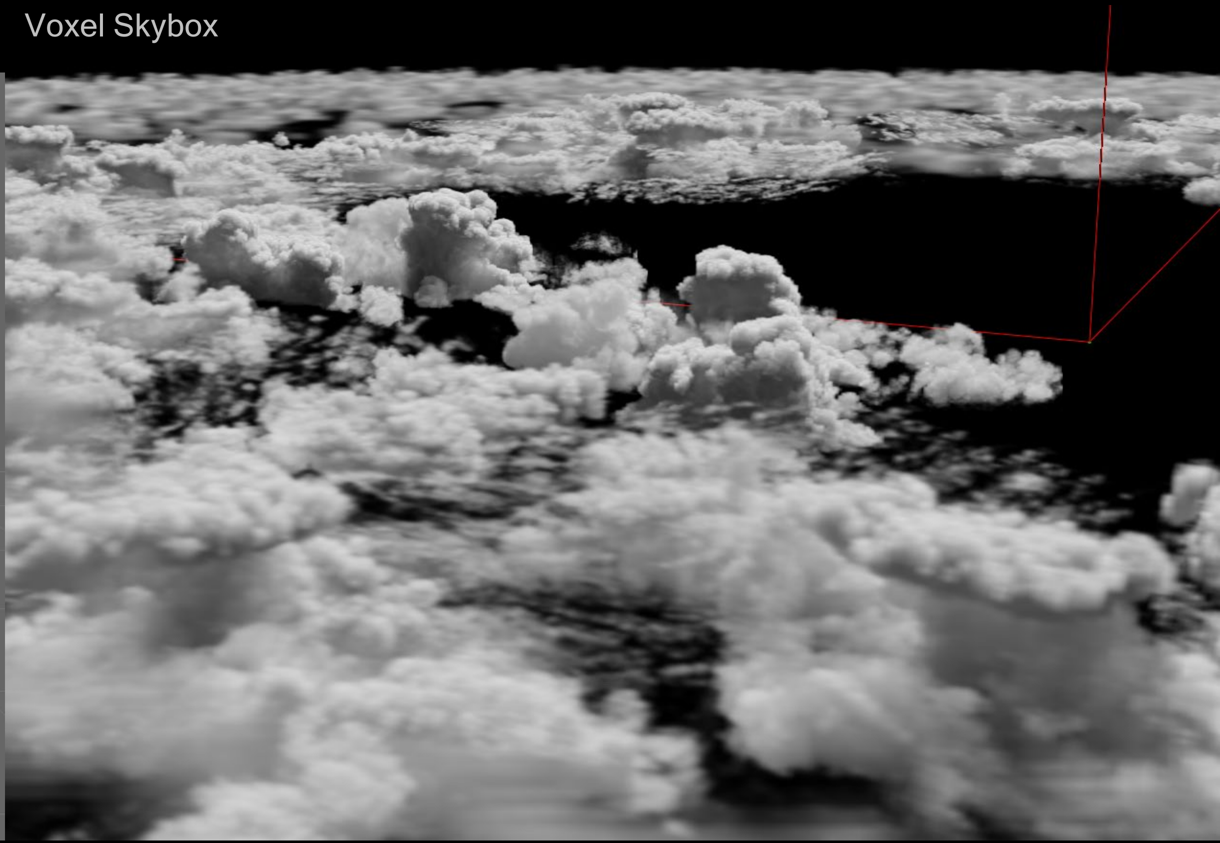


Aero Solver / Houdini, 2014

RGB Billboards



Voxel Skybox



Maya / Houdini, 2014

2.5D Clouds

Est. 2015

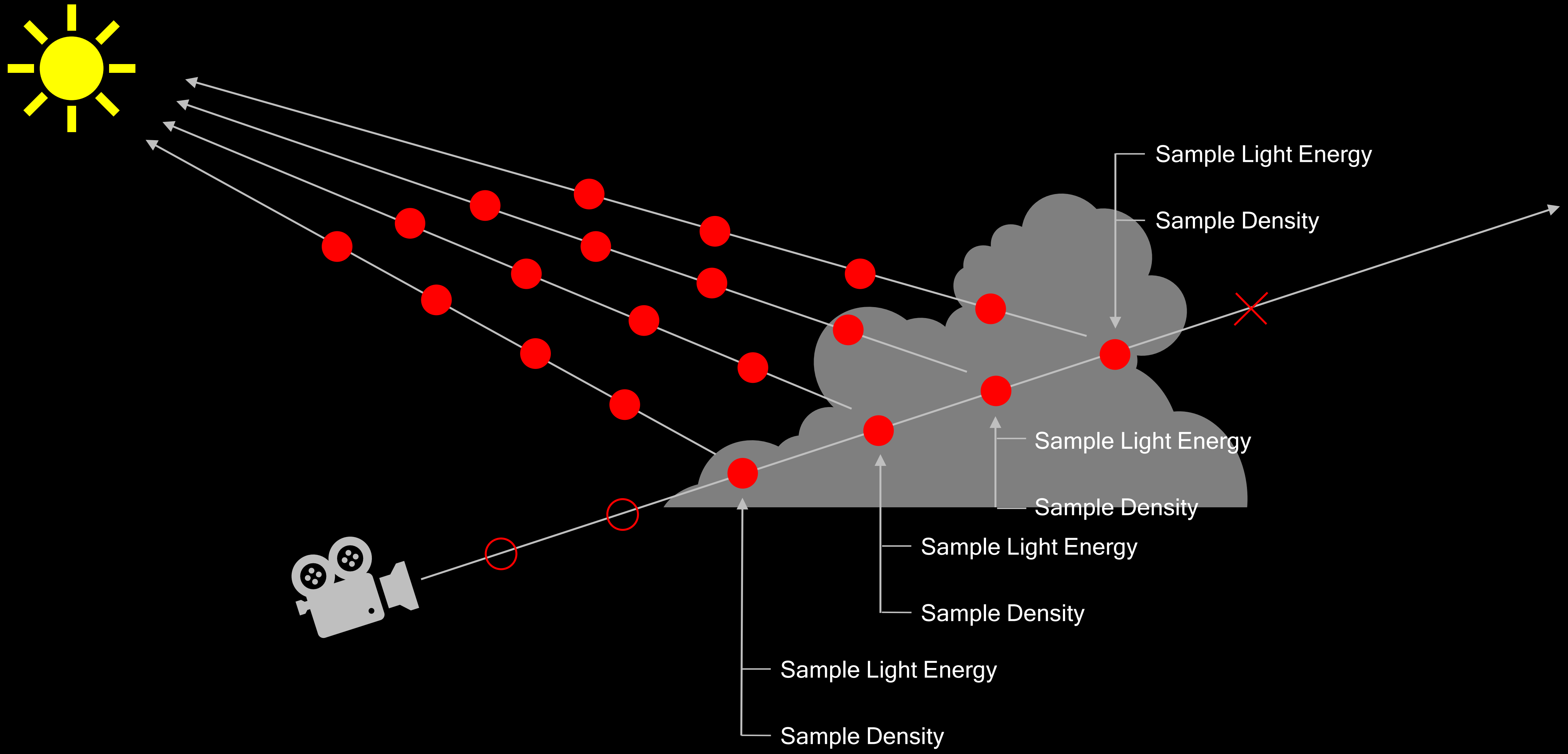
The Real-Time Volumetric
Cloudscapes of
Horizon Zero Dawn
(2015)

Real-Time Volumetric
Cloudscapes
for Games
(2016)

Nubis: Authoring Real-Time
Volumetric Cloudscapes
with the Decima Engine
(2017)

Nubis, Evolved:
Real-Time Volumetric Clouds for
Skies, Environments and VFX
(2022)

Decima Engine, 2022



Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”. *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

Nubis Volumetric
Ray-March Procedure

For each Pixel:

↳ Start a march along a ray:

↳ For each step along the ray:

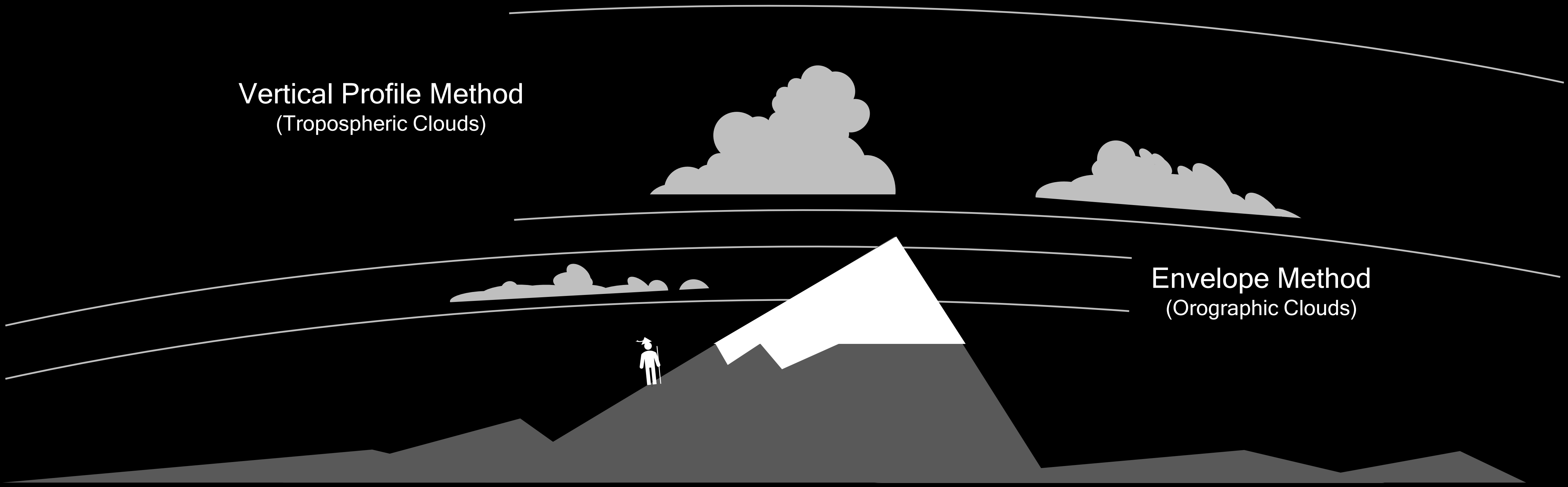
Sample Density

Sample Light Energy

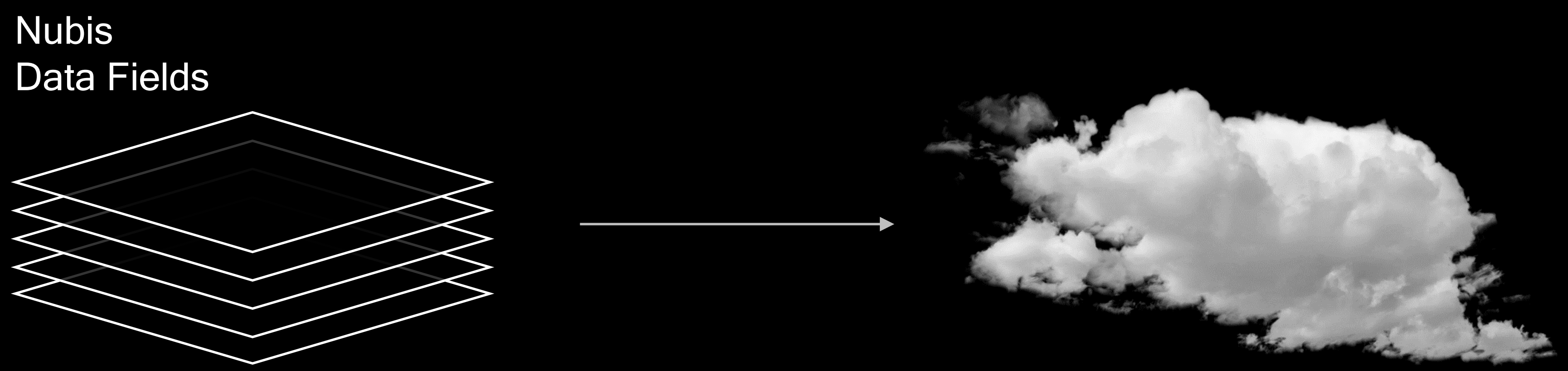
Integrate into Pixel data

Determine step size and take the next step

Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

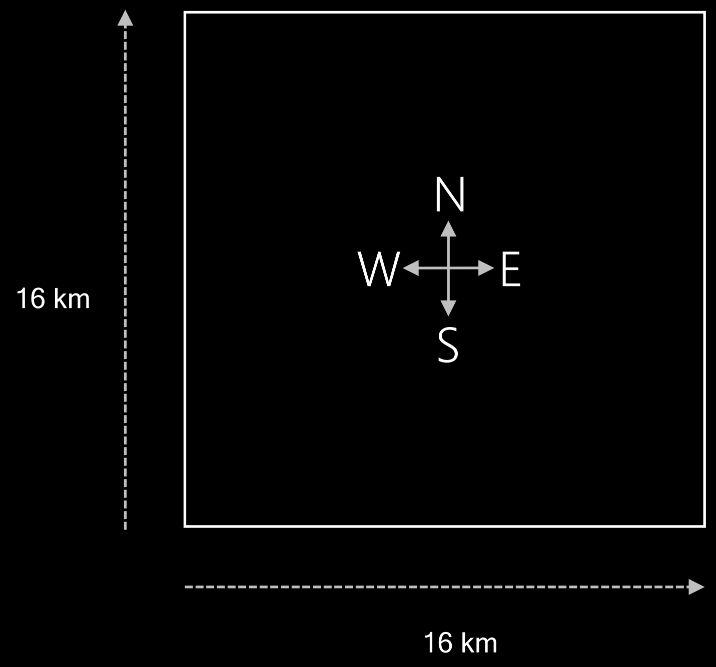


Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”. *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.



Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

2D NDF Mapping



Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

Vertical Profile Method

Envelope Method

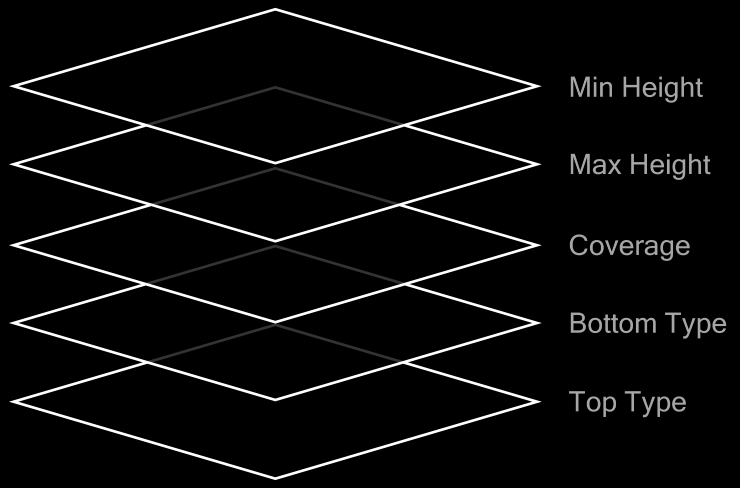
Decima Engine, 2022

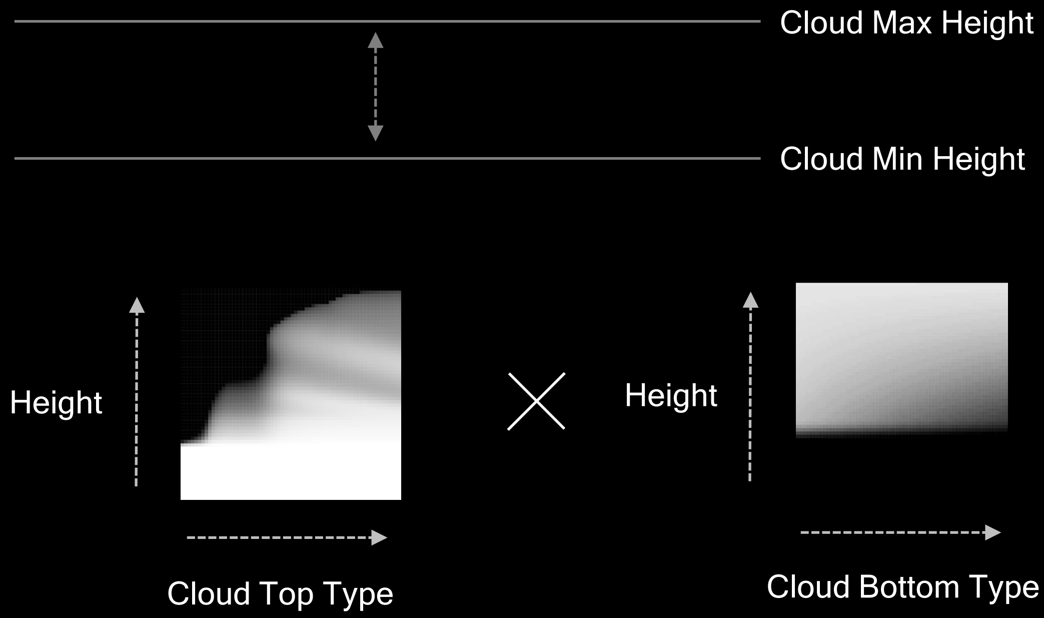
Vertical Profile Method

Envelope Method

Decima Engine, 2022

Vertical Profile Method
NDF's

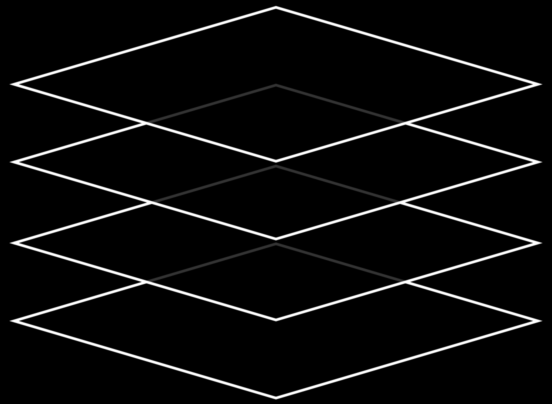


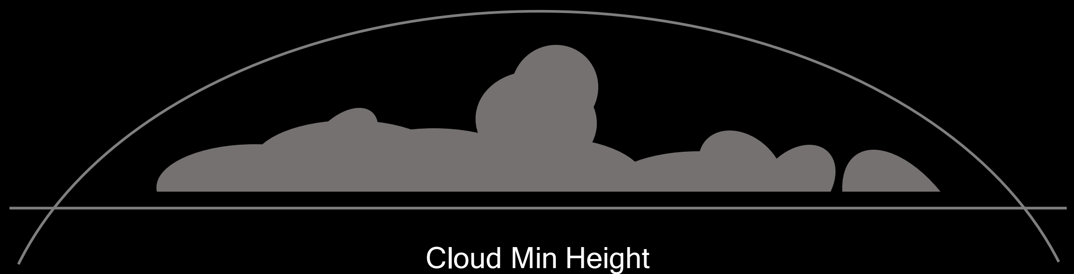



```
float dimensional_profile = vertical_profile * cloud_coverage;
```

Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

Envelope Method
NDF's






```
float dimensional_profile = bottom_gradient * top_gradient * edge_gradient;
```

Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX". *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

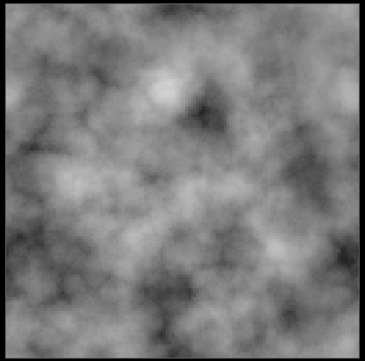
Vertical Profile Method

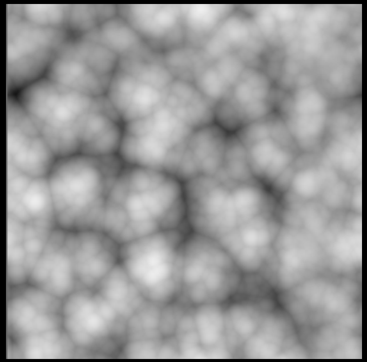
Envelope Method

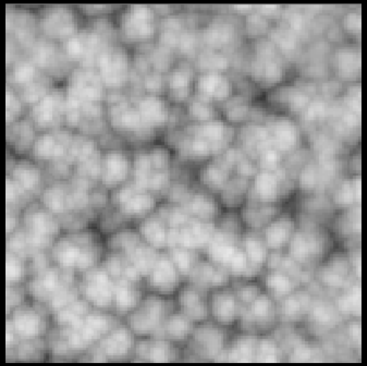
Decima Engine, 2022

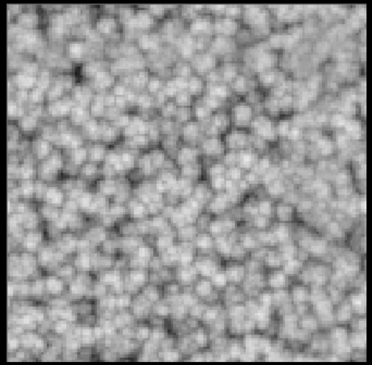
Nubis Noise

4 Channel
128 x 128 x 128 Voxels
Uncompressed, 2 Bytes / Texel
4.194 Megabytes


Low Freq “Perlin-Worley”


Low Freq Worley

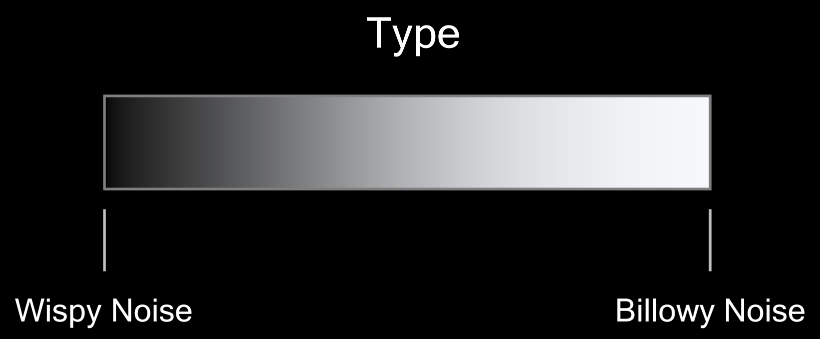

Med Freq Worley


High Freq Worley

“Wispy Noise Composite”

“Billowy Noise Composite”

Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”.
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.




```
// Define Noise composite - blend to wispy as the density scale decreases.
float noise_composite = lerp(wispy_noise, billowy_noise, detail_type);
```

Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

Vertical Profile Method

Envelope Method

```
// Define Cloud Density
cloud_density = saturate(noise - (1.0 - dimensional_profile));
```

Vertical Profile Method

Envelope Method

```
// Define Cloud Density
cloud_density = saturate(noise - (1.0 - dimensional_profile));
```

Decima Engine, 2022

Vertical Profile Method

Envelope Method

```
// Animate noise using a wind offset
float3 noise_sample_pos = sample_pos - wind_direction * scroll_offset;
```

Decima Engine, 2022

Nubis Volumetric Ray-March Procedure

For each Pixel:

↳ Start a march along a ray:

↳ For each step along the ray:

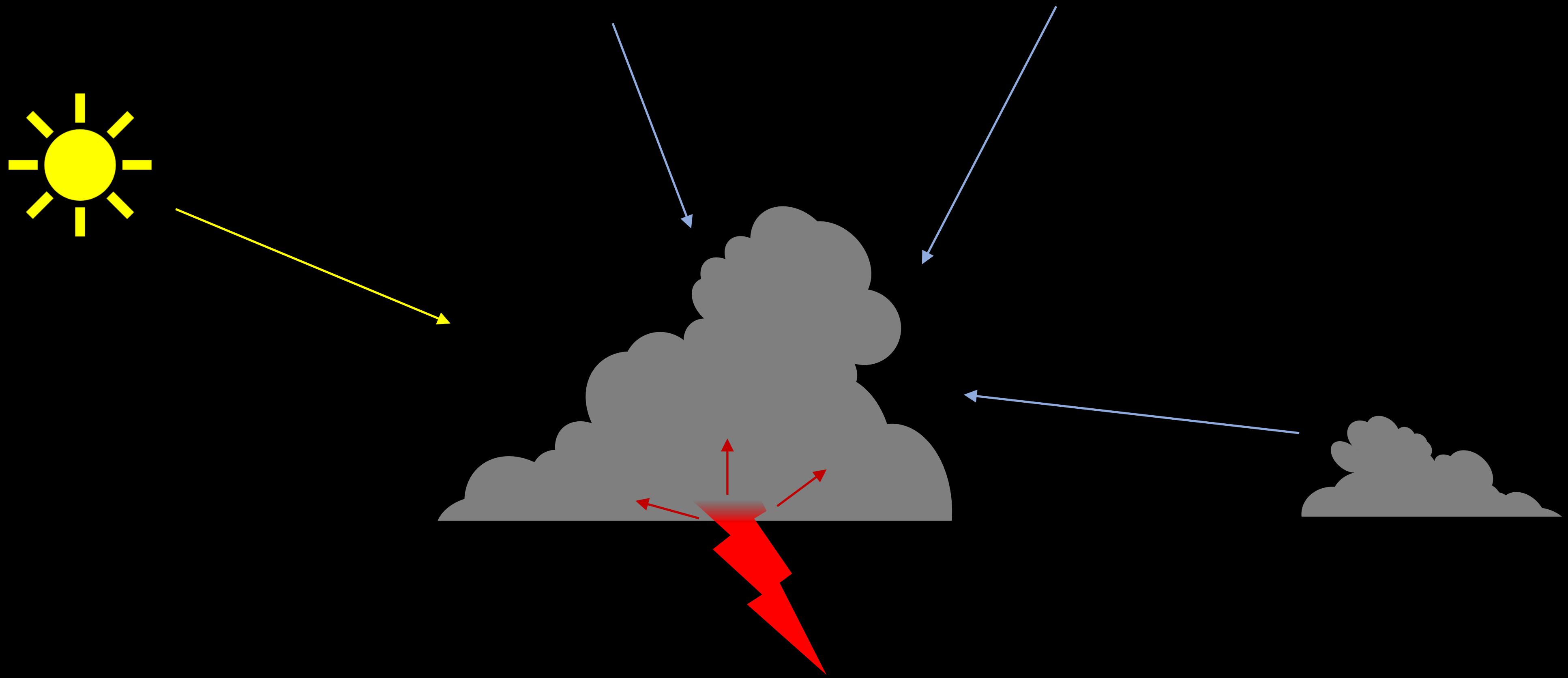
Sample Density

Sample Light Energy

Integrate into Pixel data

Determine step size and take the next step

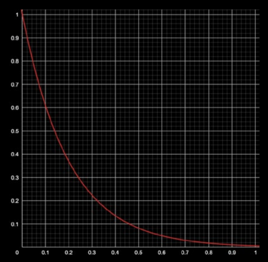
Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”. *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.



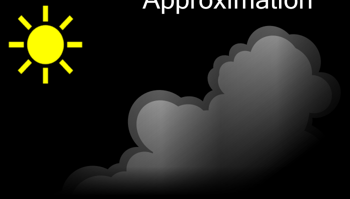
Light Energy = Direct Scattering + Ambient Scattering + Secondary Scattering

Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”. *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

The Beer-Lambert Law [1]

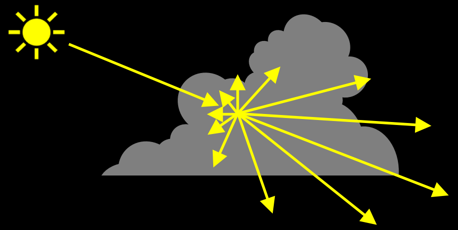


Nubis Multiscatter Approximation

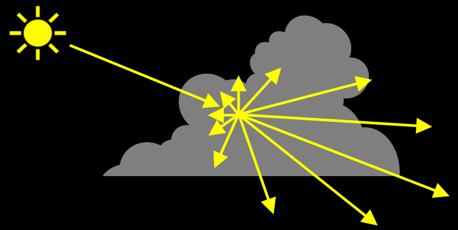


Direct Scattering = (Transmittance * Primary Scattering Phase) + (Multiple Scattering * Secondary Scattering Phase)

Henyey-Greenstein Phase Function [2]



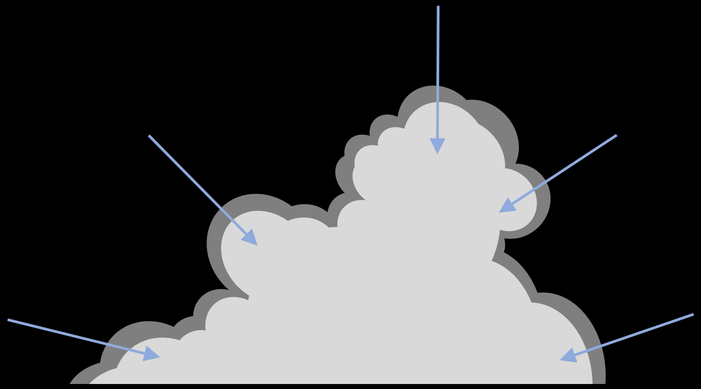
Henyey-Greenstein Phase Function [2]



1. Augustus Beer, "Bestimmung der Absorption des rothen Lichts in farbigen Flüssigkeiten" (Determination of the absorption of red light in colored liquids), *Annalen der Physik und Chemie*, vol. 86, pp. 78-88, 1852.
2. L. G. Henyey and J. L. Greenstein, "Diffuse radiation in the Galaxy," *Astrophysical Journal*, vol. 93, pp. 78-83, 1941.

Decima Engine, 2022

Decima Engine, 2022



Nubis Ambient Scattering Approximation

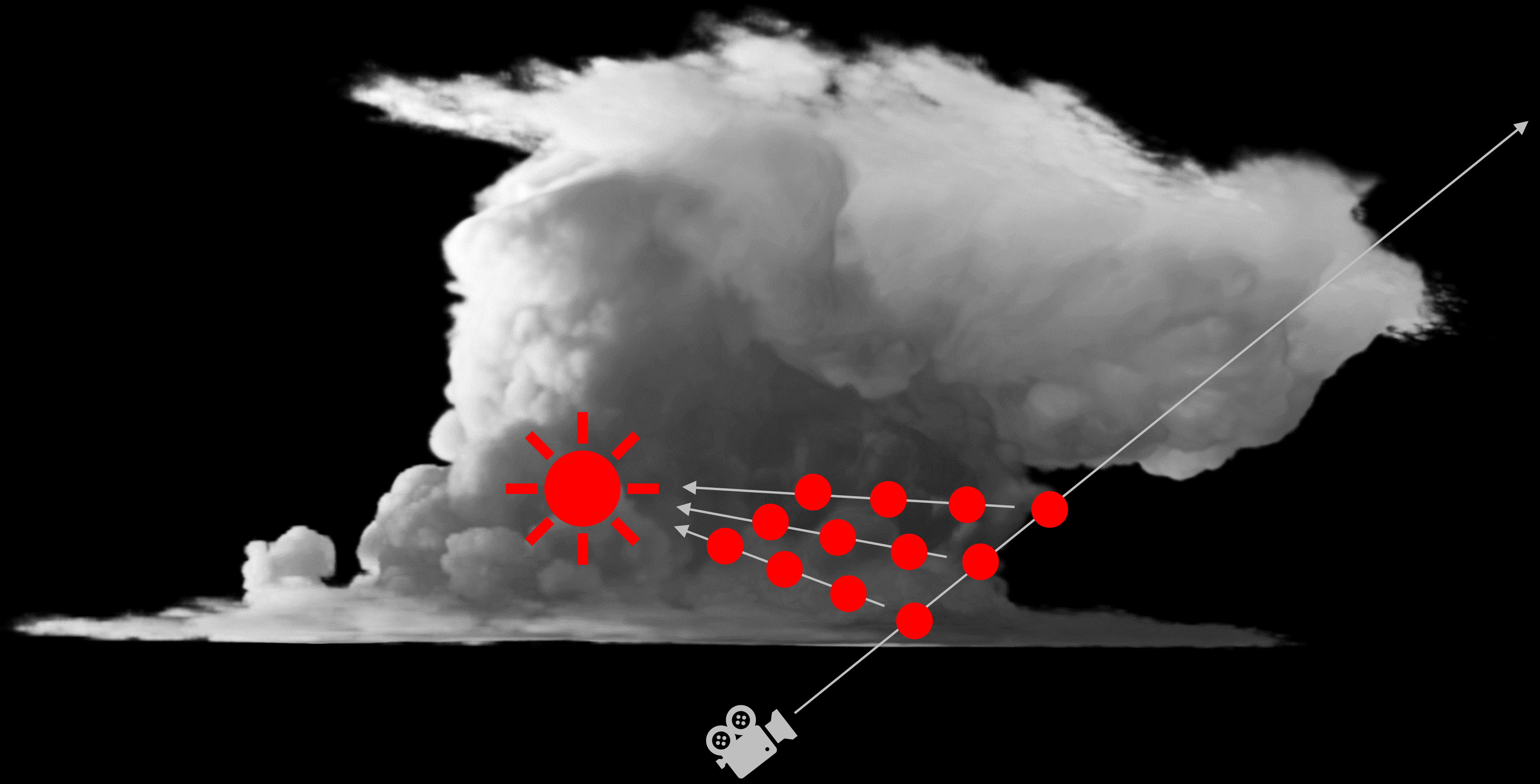
```
float ambient_scattering = pow(1.0 - dimensional_profile, 0.5);
```

Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

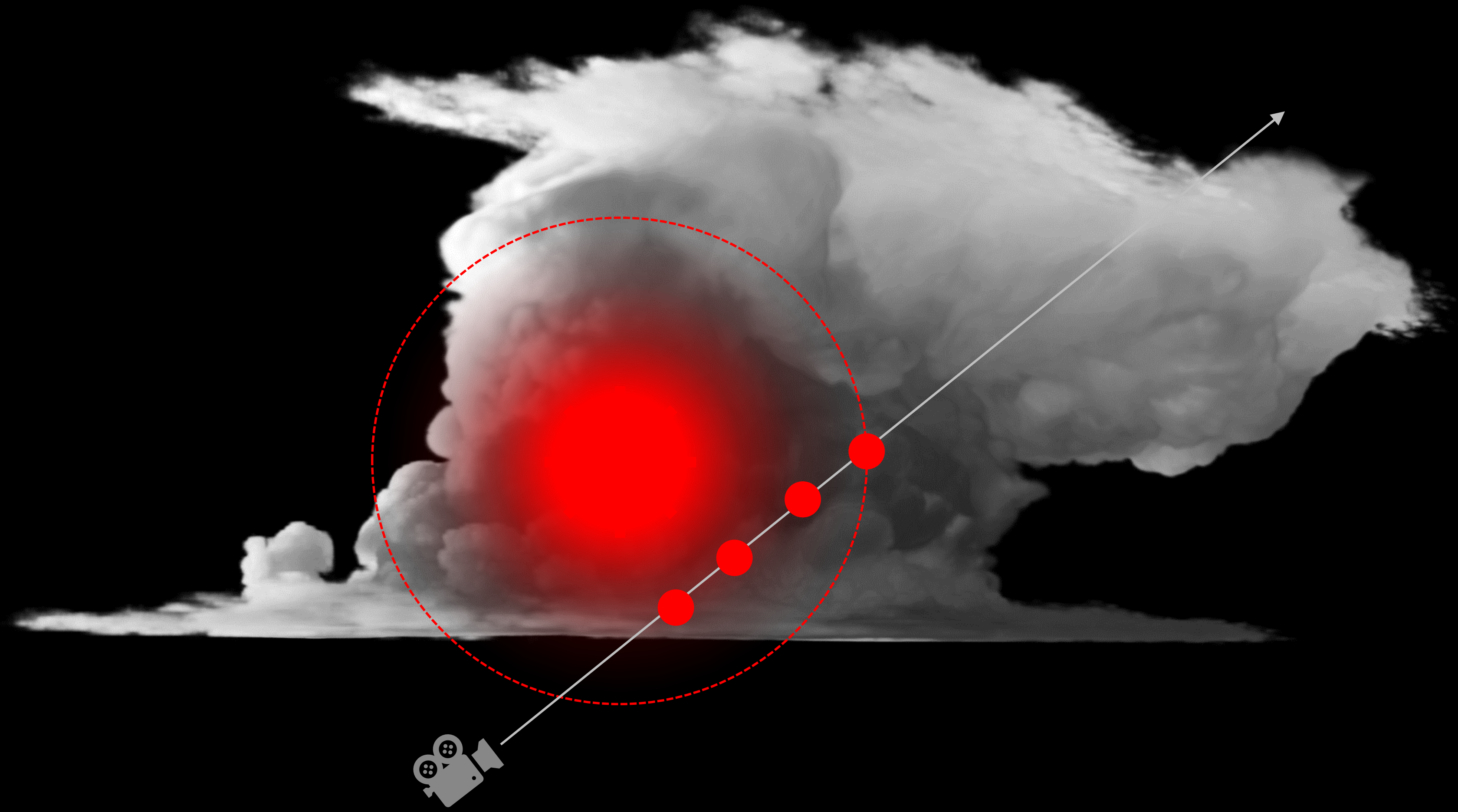
Decima Engine, 2022

Decima Engine, 2022

Decima Engine, 2022



Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”.
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.



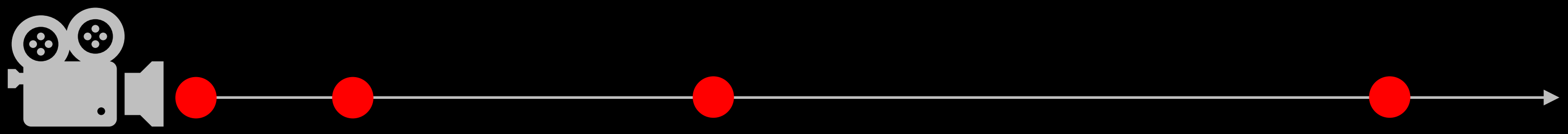
Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”. *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

Decima Engine, 2022

Nubis Volumetric Ray-March Procedure

For each Pixel:
- ↳ Start a march along a ray:
  - ↳ For each step along the ray:
    - Sample Density
    - Sample Light Energy
    - Integrate into Pixel data
    - Determine step size and take the next step

Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”. *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

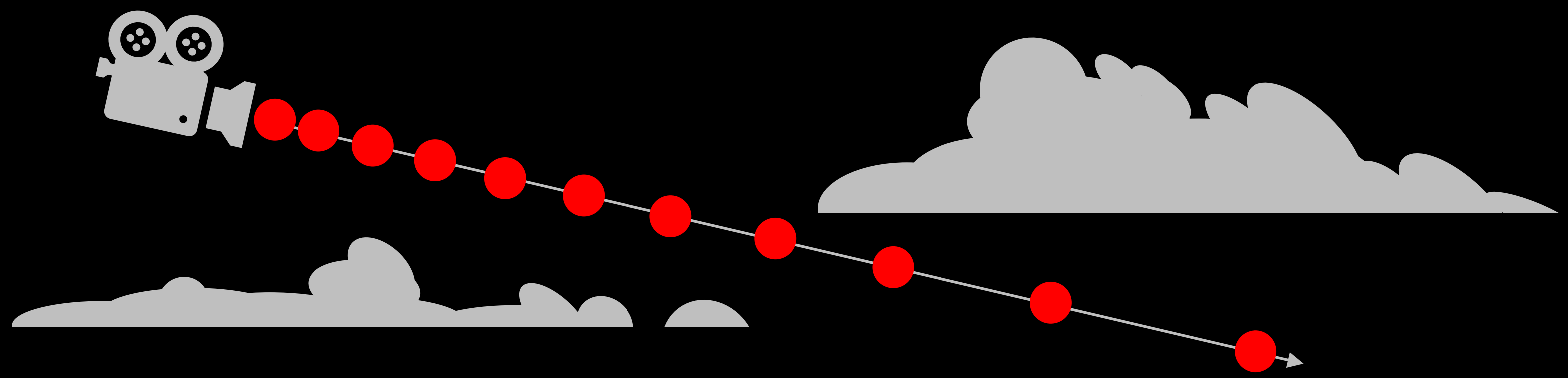


```
// Define step size constants
float near_step_size = 3.0;
float far_step_size_offset = 60.0;
float step_adjustment_distance = 16384.0;

// Calculate distanced-based step size
float step_size = near_step_size + ((far_step_size_offset * distance_from_camera) / step_adjustment_distance);
```

Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

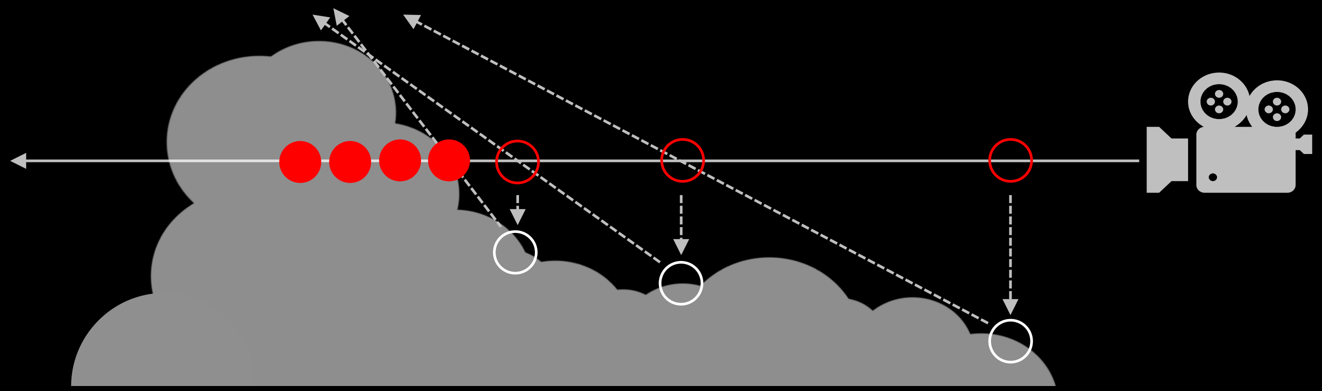
Decima Engine, 2022

Decima Engine, 2022

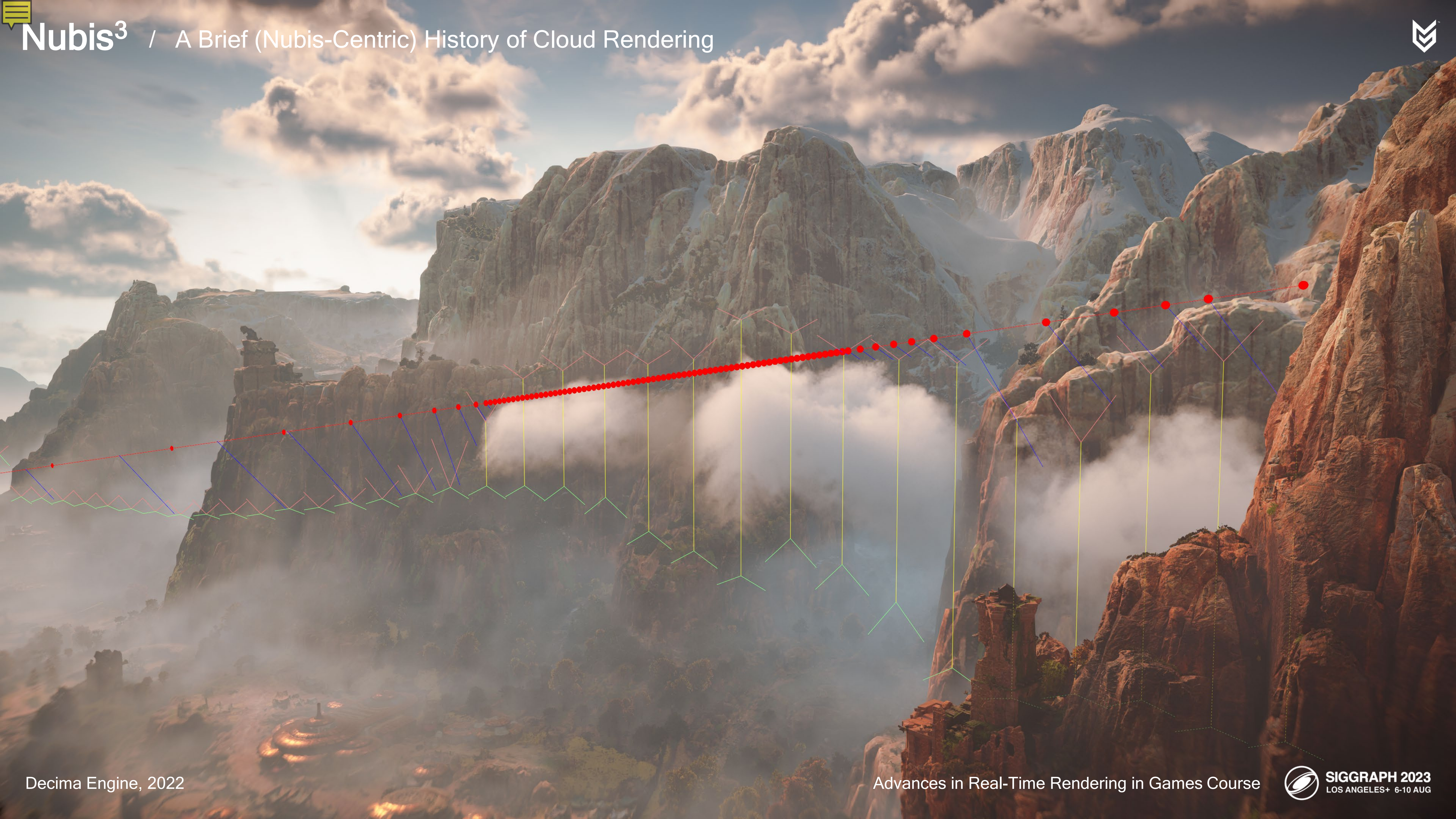


Cone Step Mapping + Distance Step Mapping

Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

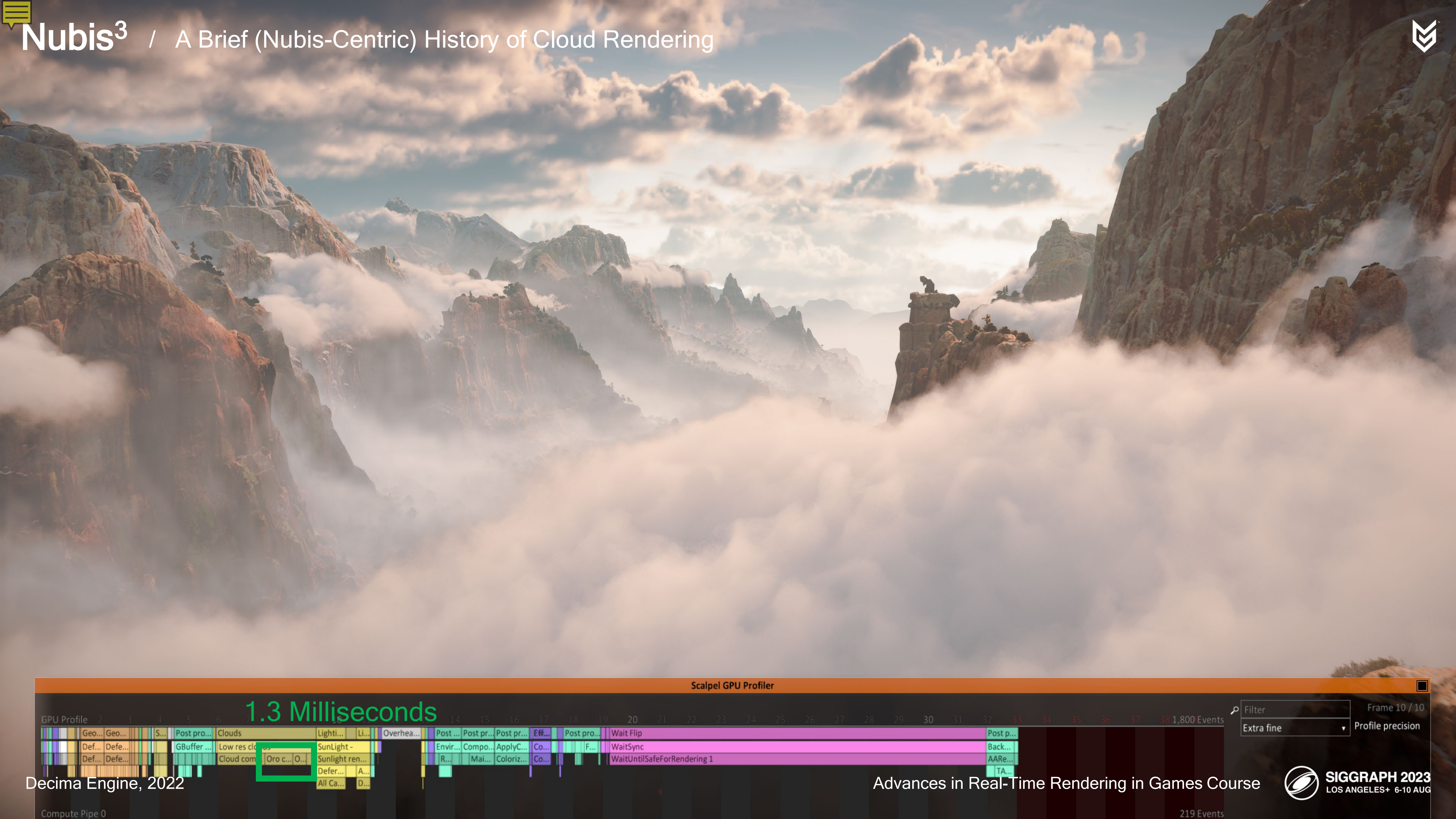


Jonathan “Lone Sock” Dummer, Cone Step Mapping: An Iterative Ray Heightfield Intersection Algorithm. 2006.



Decima Engine, 2022

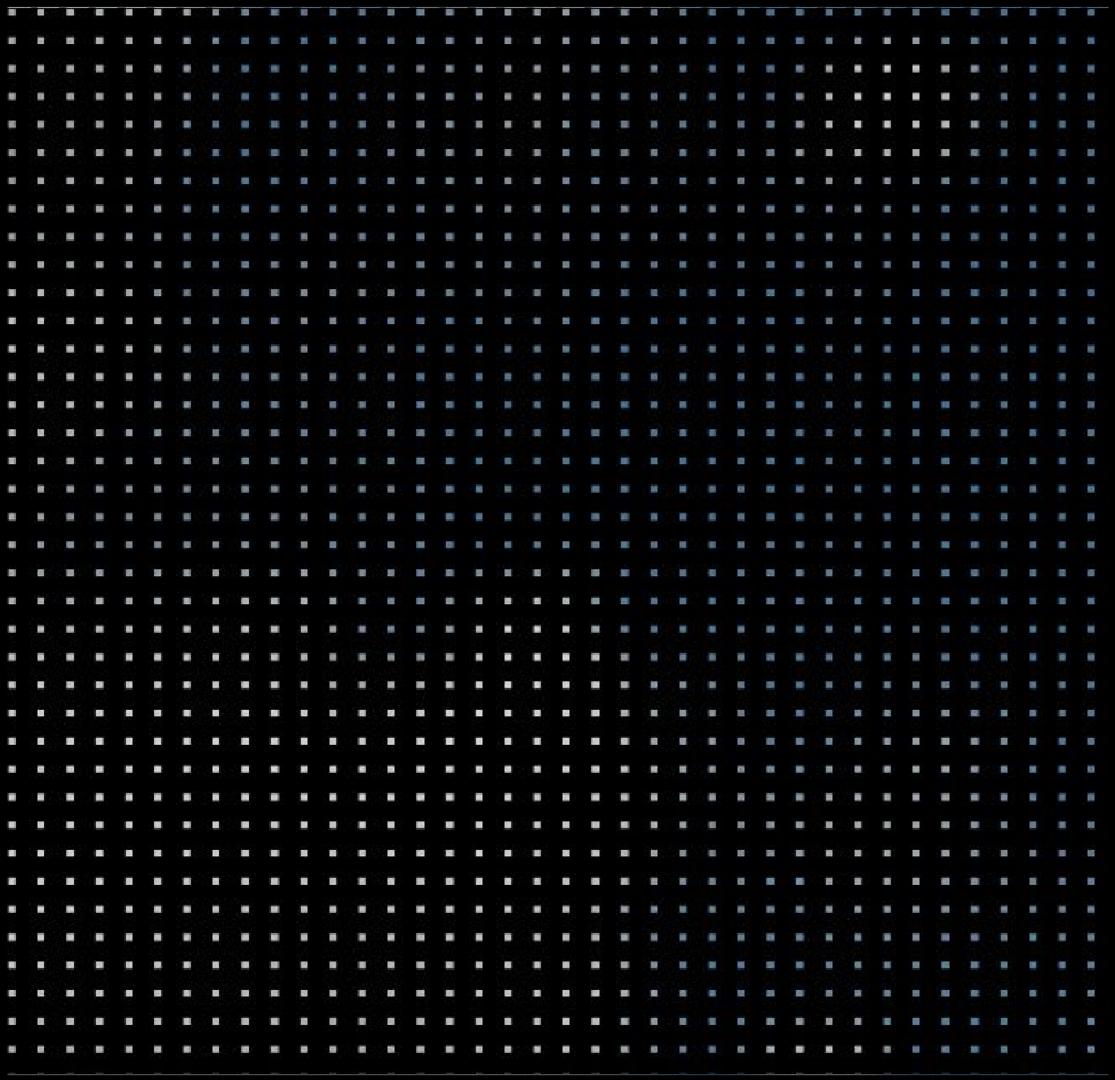
Decima Engine, 2022



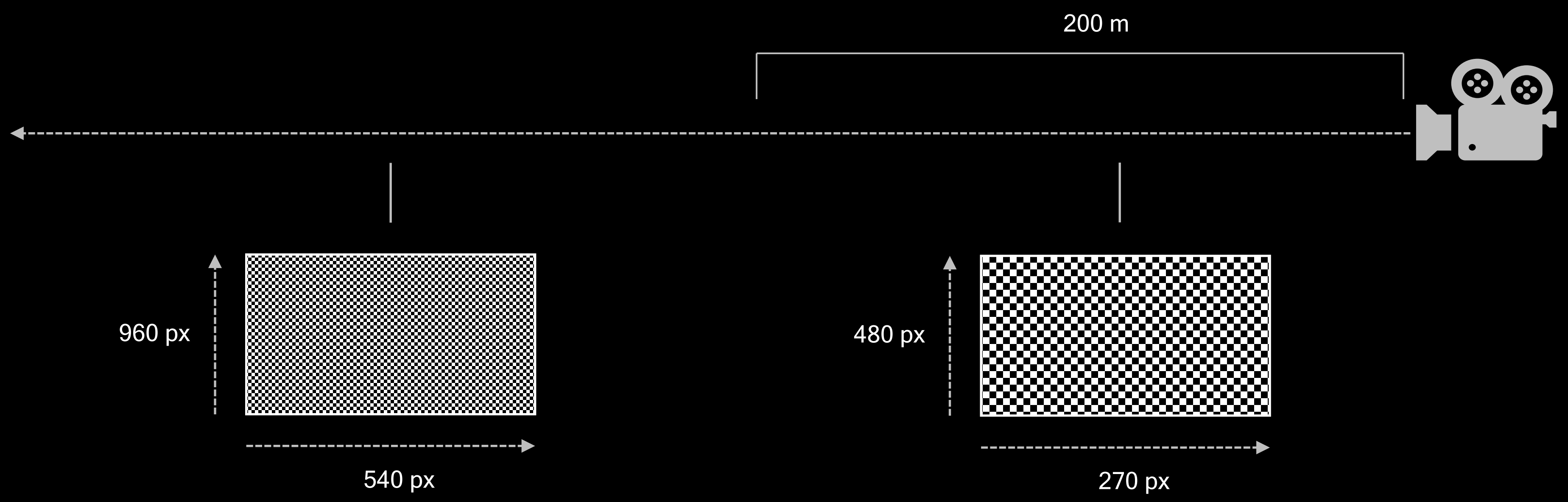
Decima Engine, 2022

Nubis Volumetric
Ray-March Procedure

For each Pixel:
↳ Start a march along a ray:
↳ For each step along the ray:
Sample Density
Sample Light Energy
Integrate into Pixel data
Determine step size and take the next step

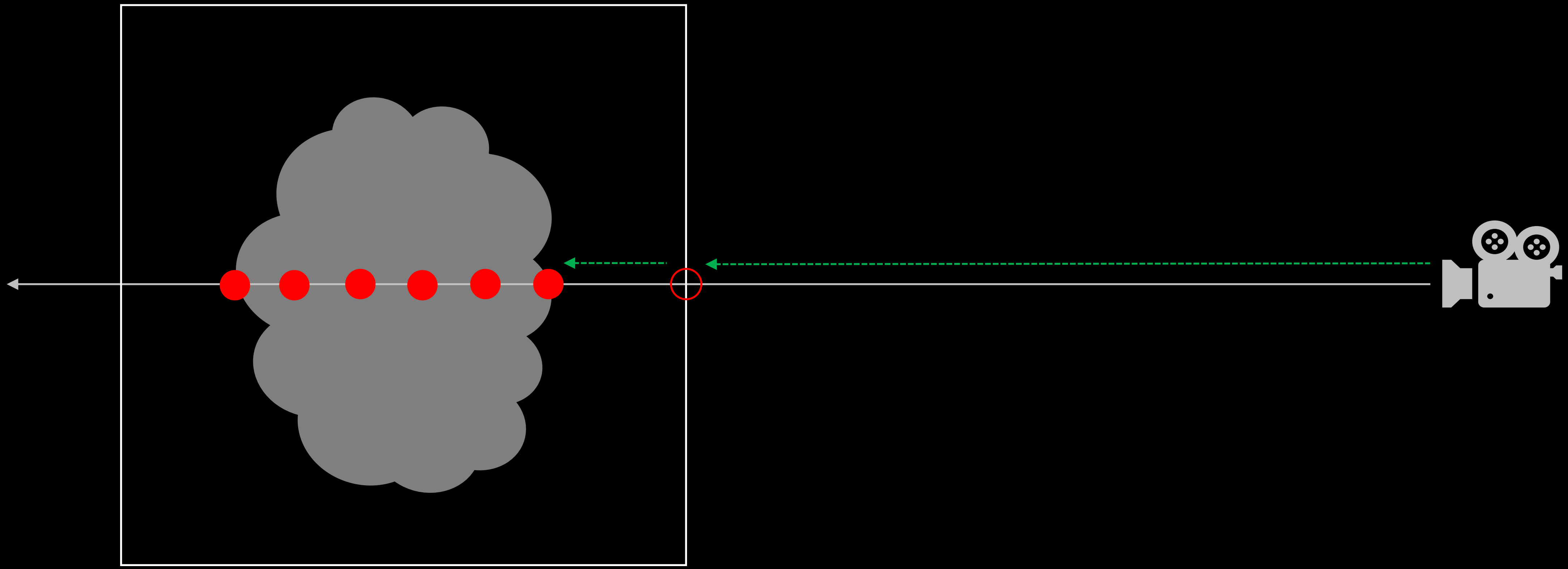


Andrew Schneider. “Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX”. *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.



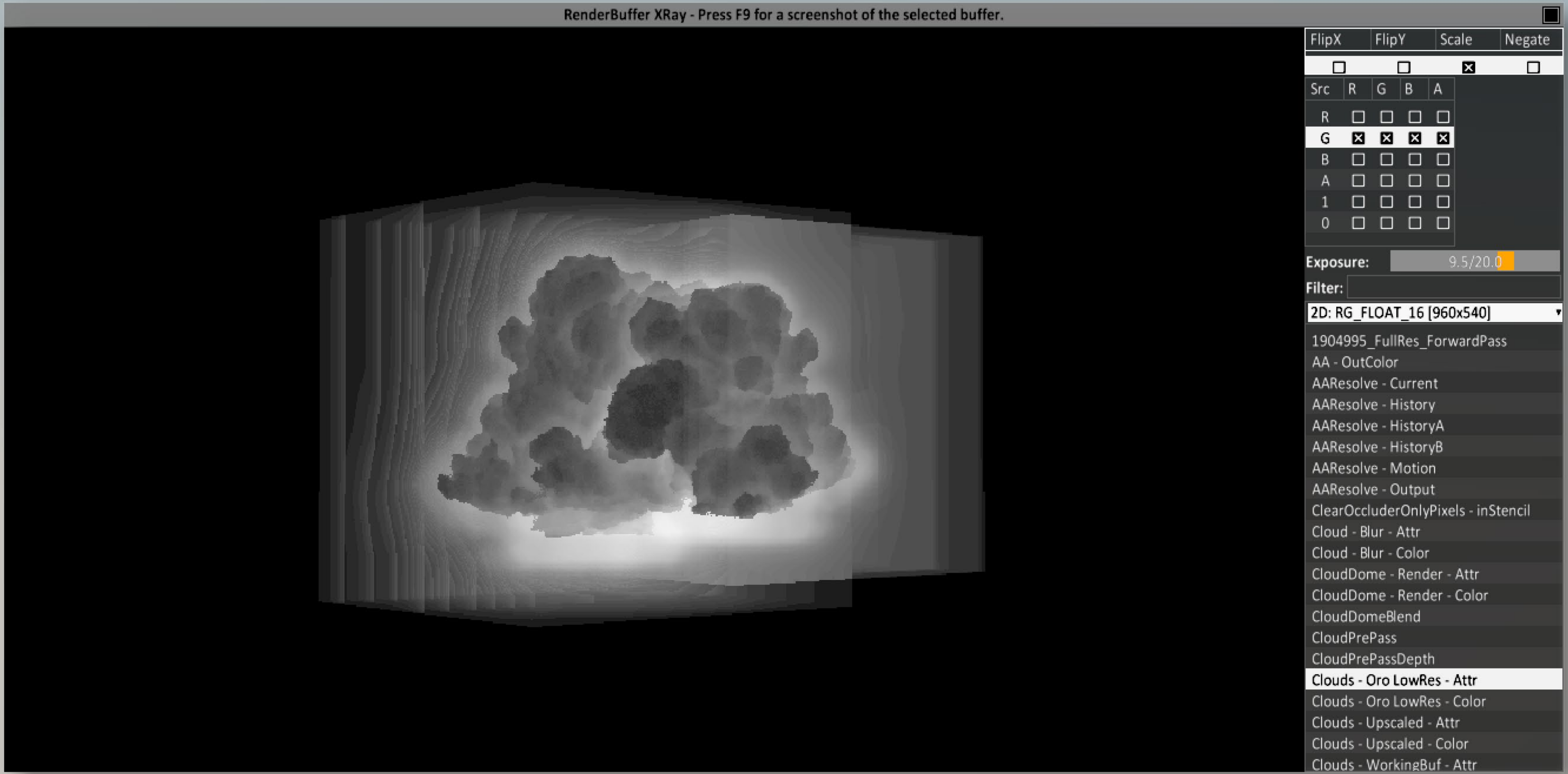
Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.

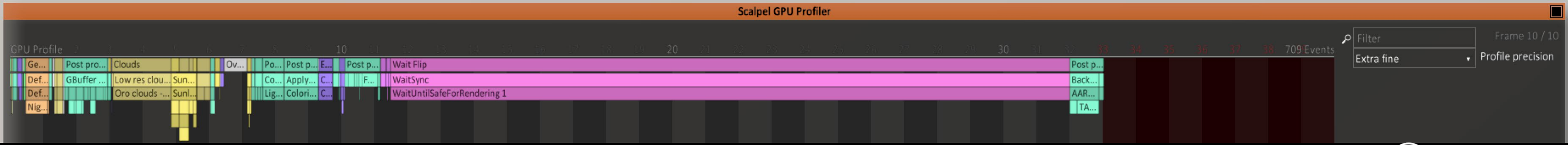
	Vertical Profile Method	Envelope Method
Evolution	Yes	Pseudomotion Only
Time Of Day	Yes	Yes
Lightning	Yes	No
High Frame-rates	Yes	Yes
Flight-Capable	No	Yes
Freeform Modeling	No	No



Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX".
ACM SIGGRAPH. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022. (Slides 192-205)

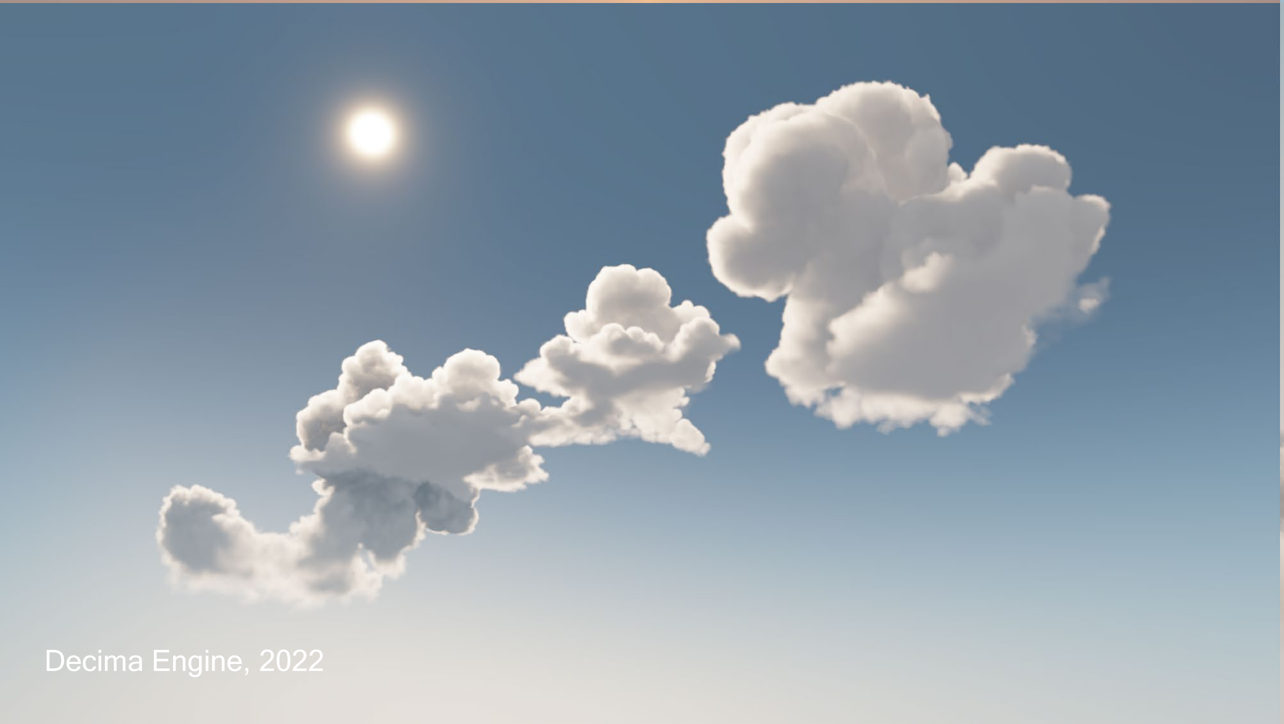
Decima Engine, 2022


RenderBuffer XRay - Press F9 for a screenshot of the selected buffer.
FlipX
FlipY
Scale
Negate
Src
R
G
B
A
1
0
Exposure:
9.5/20.0
Filter:
2D: RG_FLOAT_16 [960x540]
1904995_FullRes_ForwardPass
AA - OutColor
AAResolve - Current
AAResolve - History
AAResolve - HistoryA
AAResolve - HistoryB
AAResolve - Motion
AAResolve - Output
ClearOccluderOnlyPixels - inStencil
Cloud - Blur - Attr
Cloud - Blur - Color
CloudDome - Render - Attr
CloudDome - Render - Color
CloudDomeBlend
CloudPrePass
CloudPrePassDepth
Clouds - Oro LowRes - Attr
Clouds - Oro LowRes - Color
Clouds - Upscaled - Attr
Clouds - Upscaled - Color
Clouds - WorkingBuf - Attr


Scalpel GPU Profiler
GPU Profile
10
20
30
709 Events
Filter
Frame 10 / 10
Extra fine
Profile precision
Post pro...
Clouds
GBuffer ...
Low res clou...
Oro clouds -...
Wait Flip
WaitSync
WaitUntilSafeForRendering 1
Post p...
Apply...
Colori...
Back...
AAR...
TA...

Nubis^3 / A Multi-Voxel Cloud Renderer Prototype






Decima Engine, 2022

Nubis³ / The Burning Shores Trailer

Nubis³ / The Burning Shores Trailer

The Plan: <u>All</u>

Nubis^3 / The Burning Shores Trailer

Nubis^3 / The Burning Shores Trailer

Decima Engine, 2023

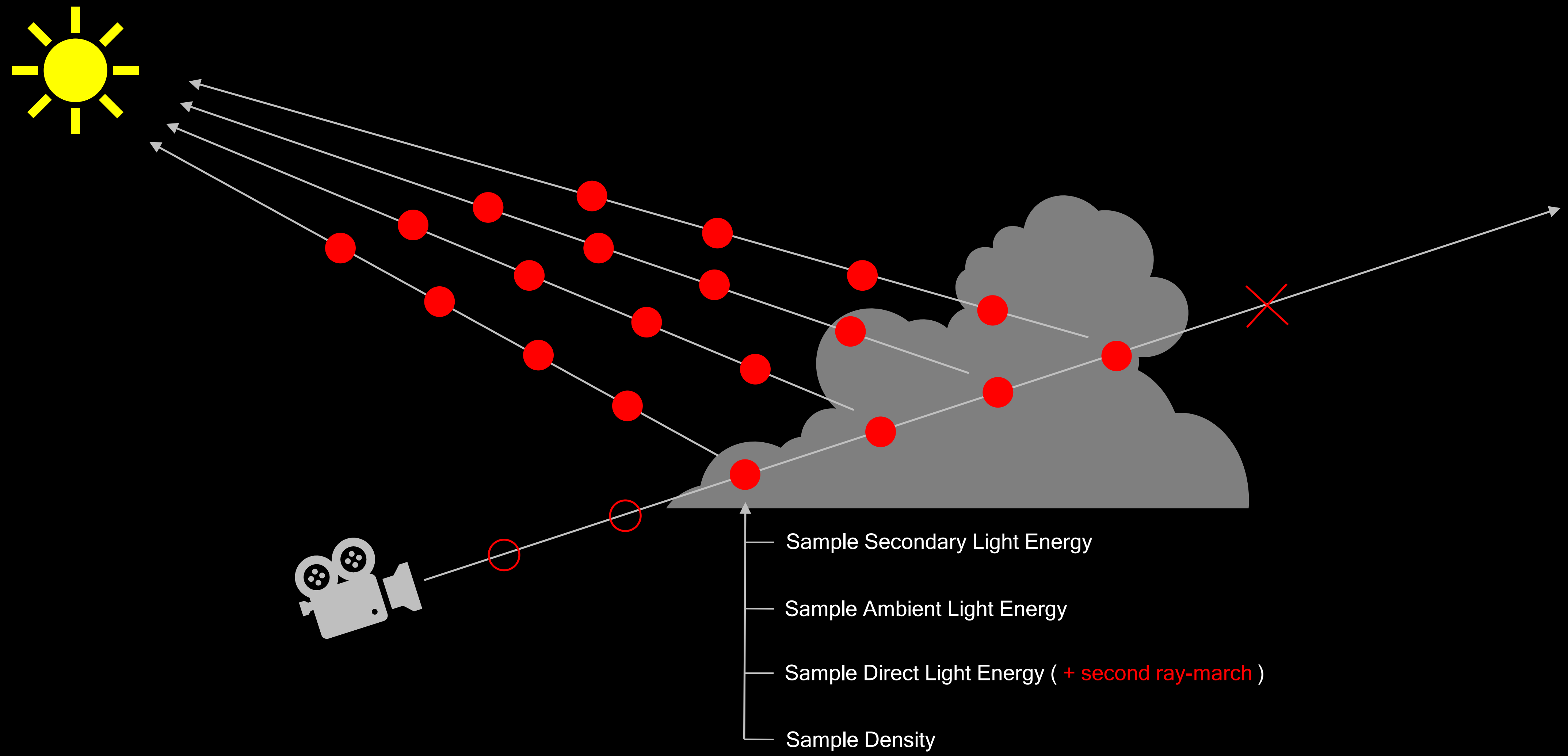
Nubis^3 / The Burning Shores Trailer

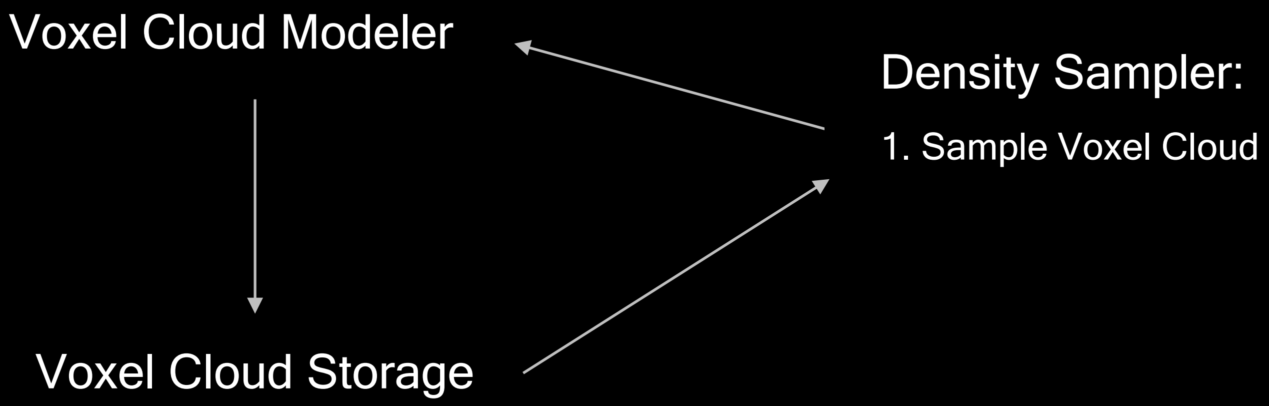
Nubis^3 / The Burning Shores

Guerrilla Viz Team, 2023

Voxel Clouds

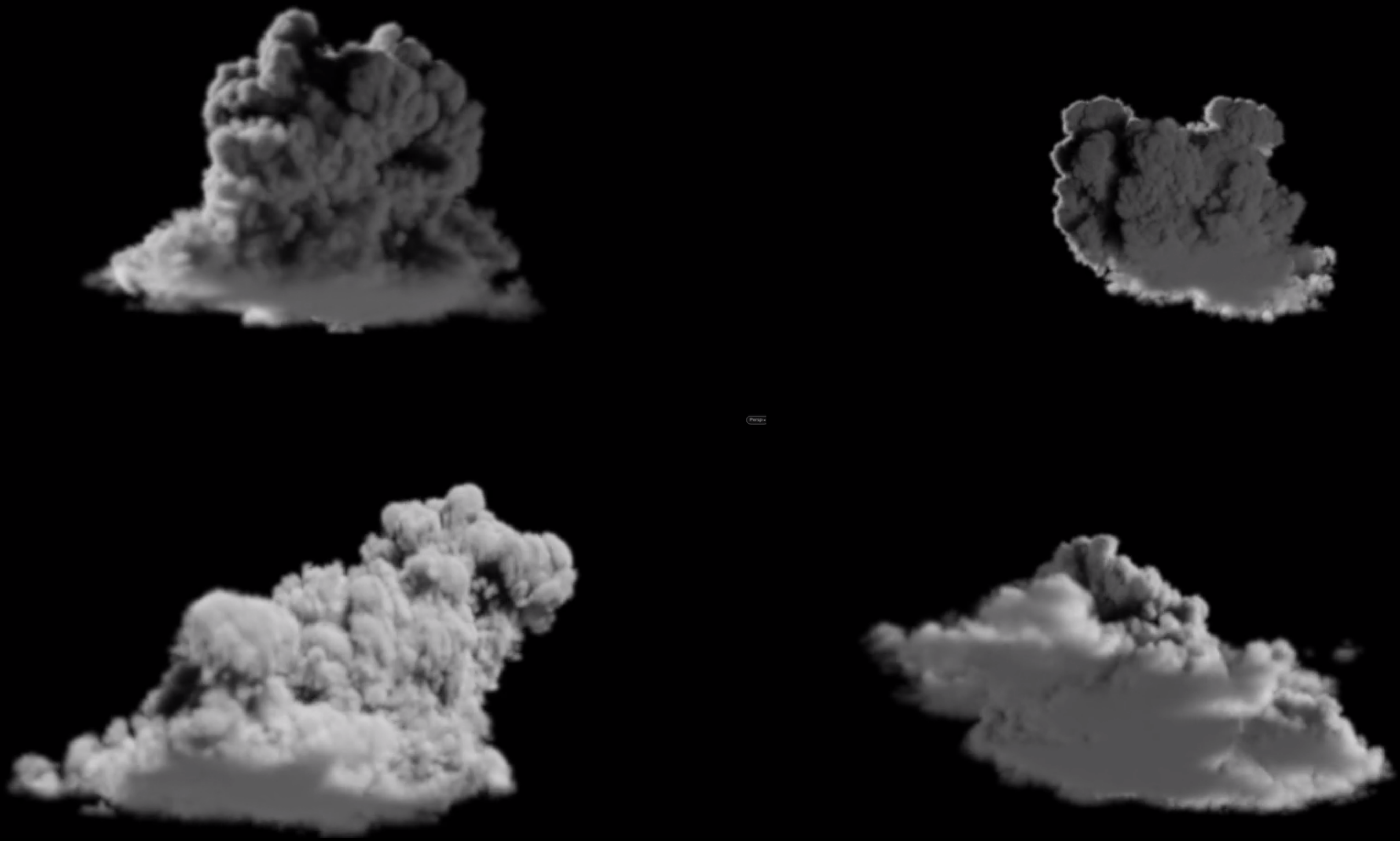
Est. 2023


Sample Secondary Light Energy
Sample Ambient Light Energy
Sample Direct Light Energy (+ second ray-march)
Sample Density


Voxel Cloud Modeler
Voxel Cloud Storage
Density Sampler:
1. Sample Voxel Cloud

Voxel Cloud Modeling Methods

- Metaballs
- Voxelized Meshes
- Particles
- **Fluid Simulation**



Atlas / Aero Solver, Houdini

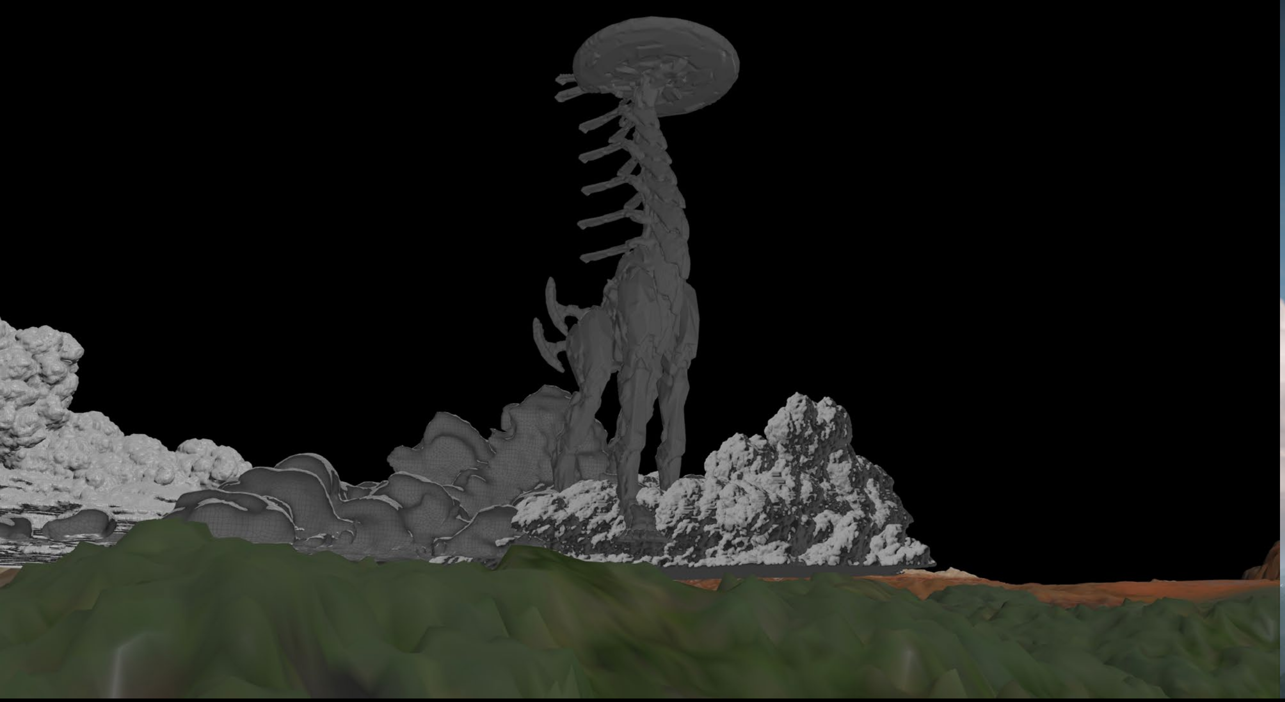
Atlas Tools (Houdini)

Voxel Compositing Tools

Cloud simulation

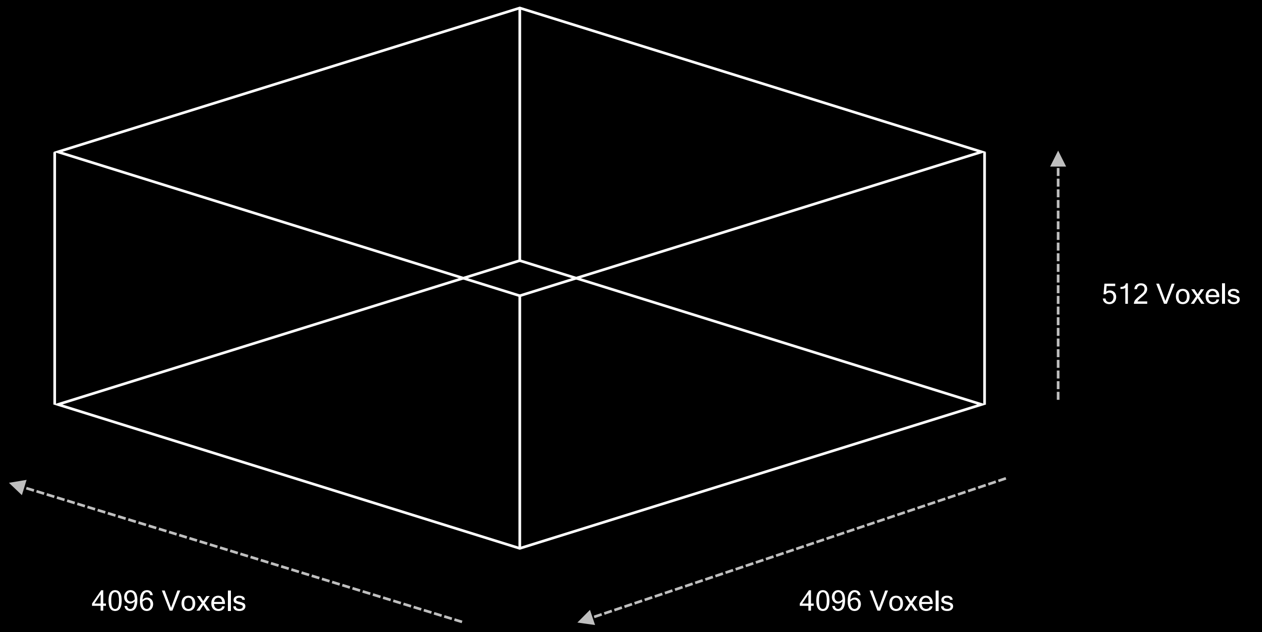
Sourcing:
- Voxels
- Point Clouds
- Meshes and
- 2.5D Authoring Data

Editing / Manipulation:
- Cutouts
- Erosion
- Squashing



Houdini / Decima Engine, 2023 (Artist: Bryan Adams)

Density Field



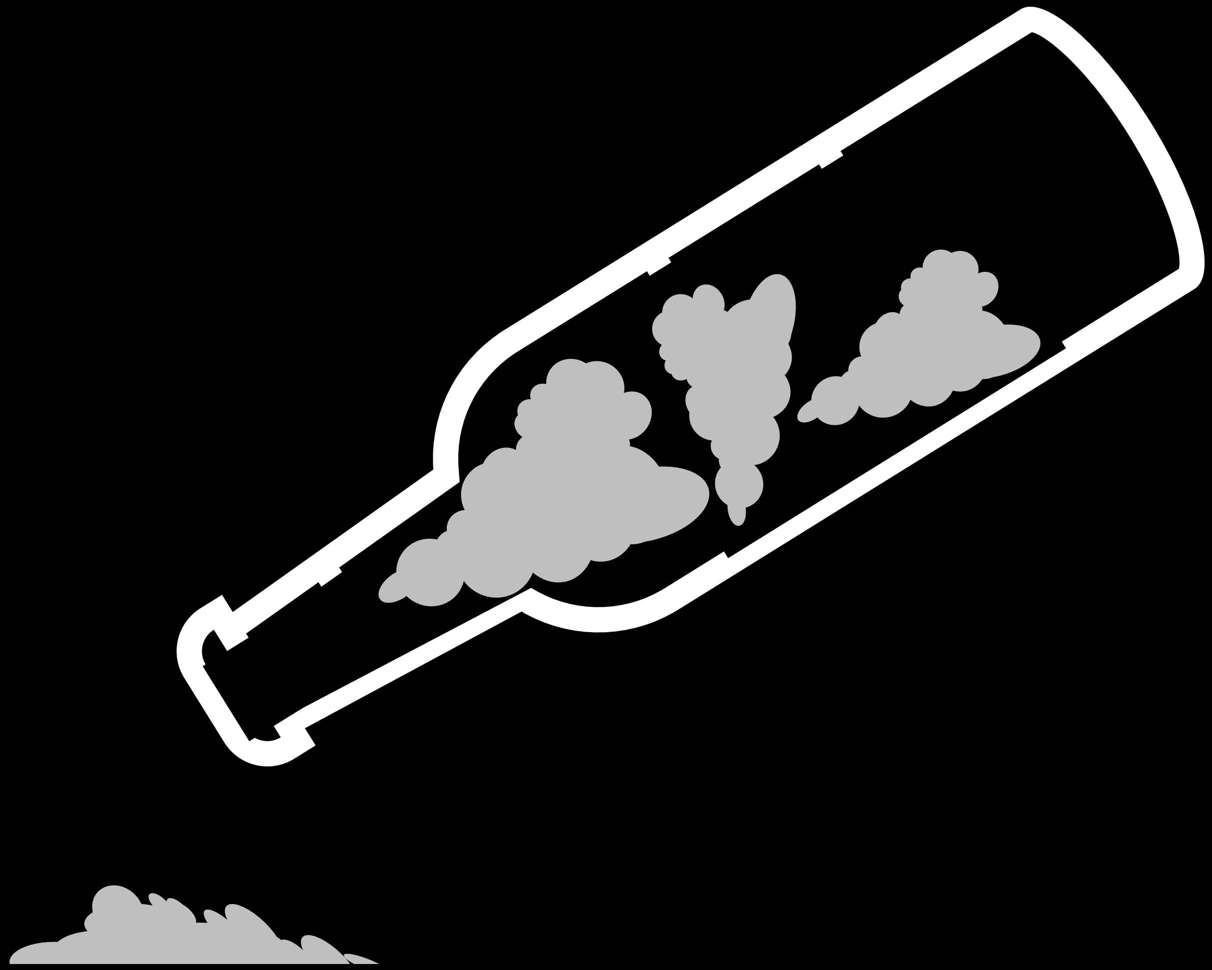
Voxel Cloud Modeler

↓

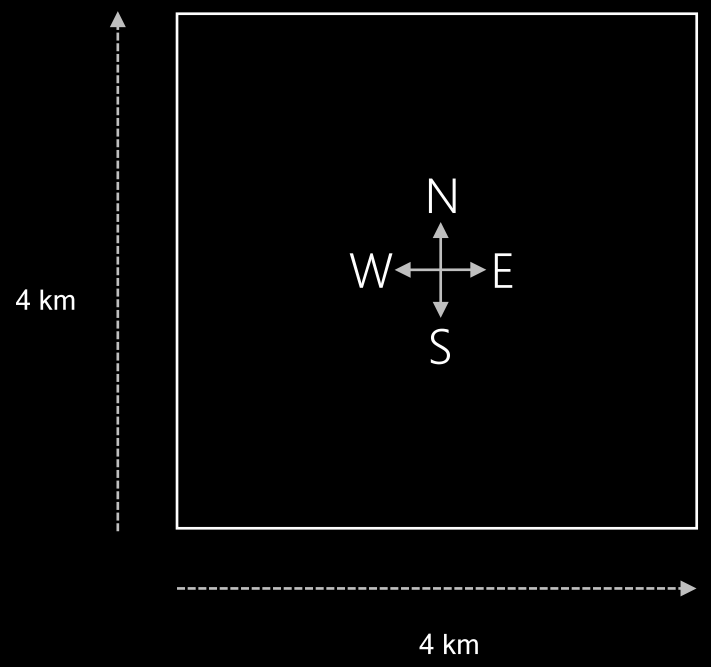
Voxel Cloud Storage →

Density Sampler:

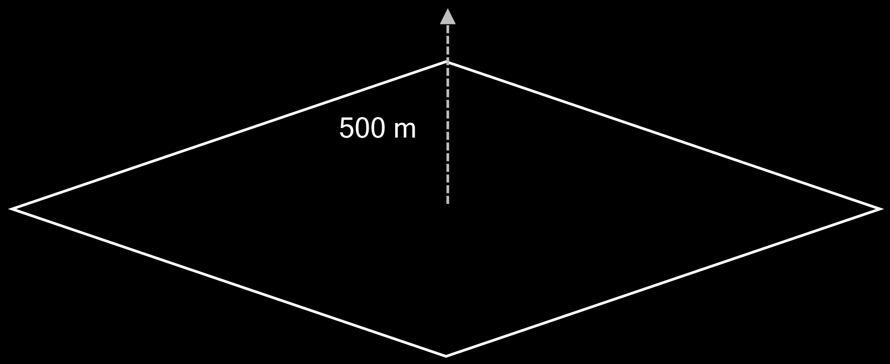
1. Sample Voxel Cloud



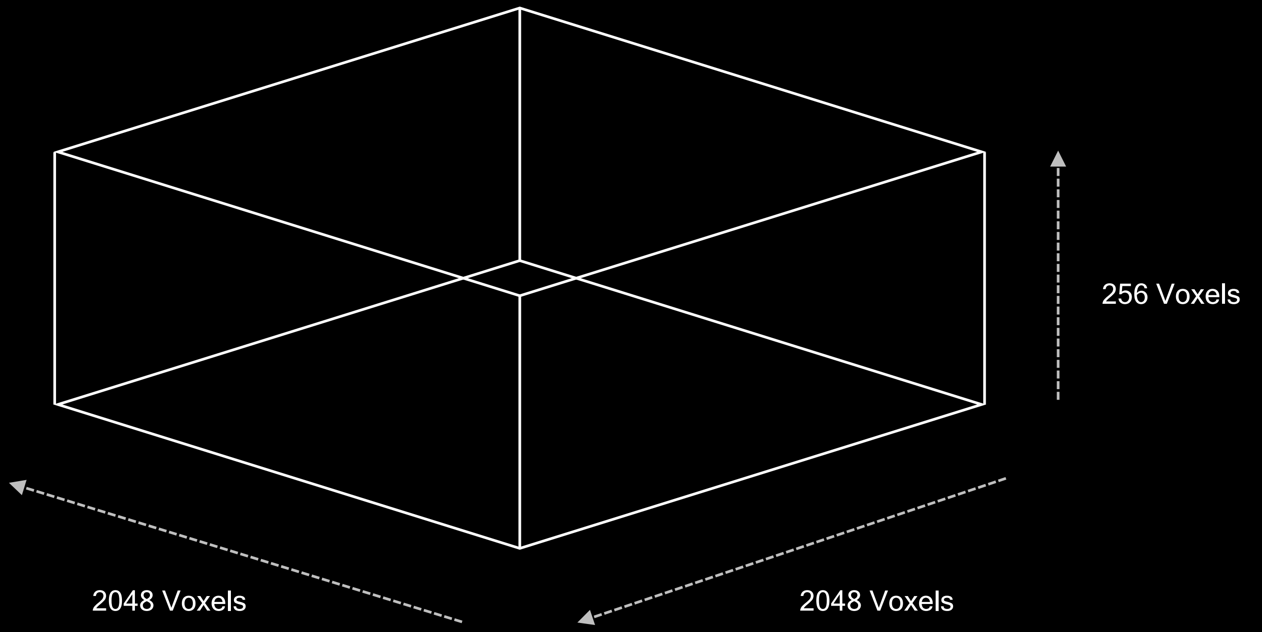
DLC Map Size



DLC Map Height



Density Field



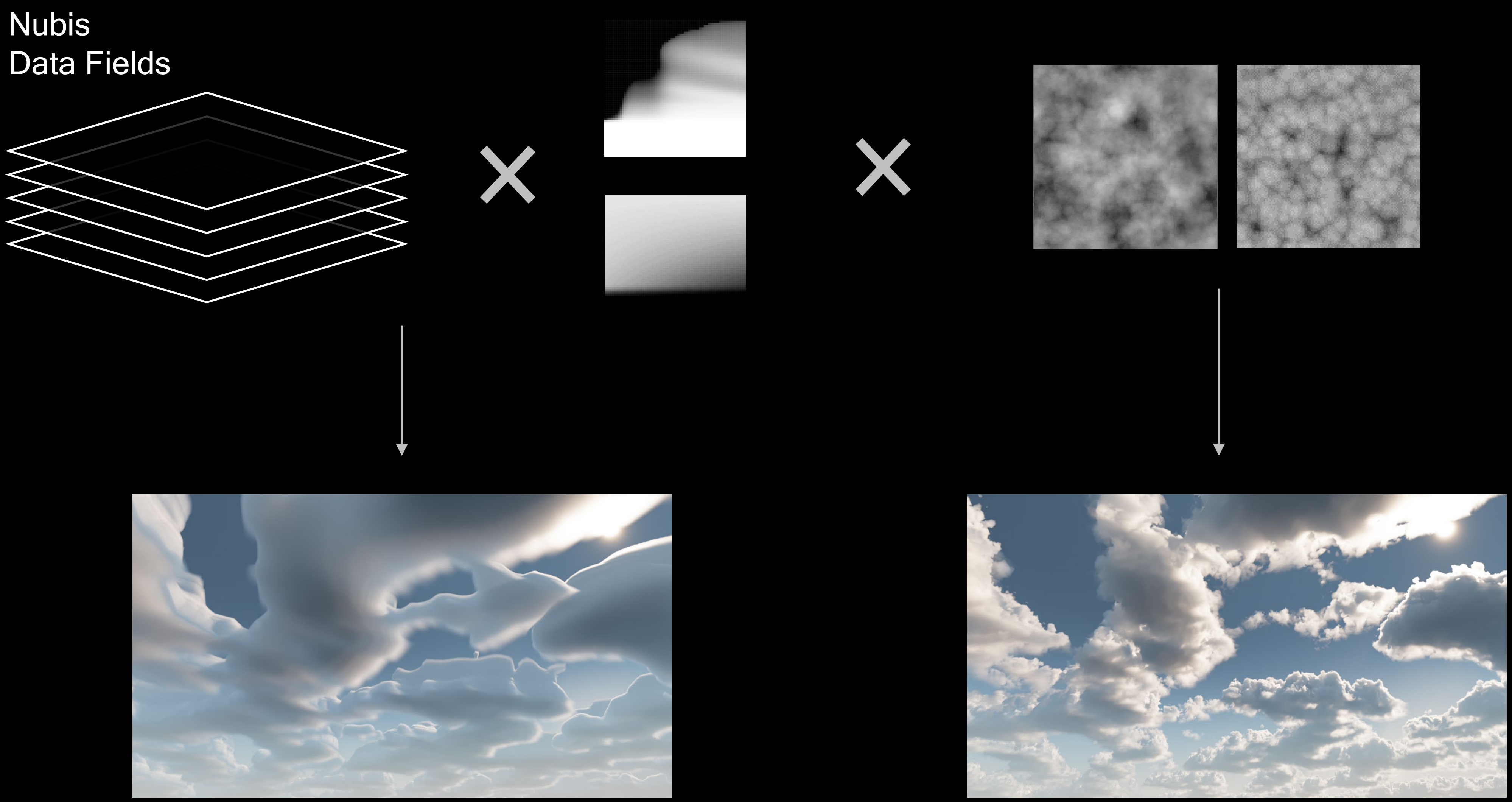
Resolution (voxels)	Uncompressed (2 bytes / texel)	BC4 (0.5 byte / texel)
4096 x 4096 x 512	17.178 Gigabytes	4.294 Gigabytes
2048 x 2048 x 256	2.146 Gigabytes	536.870 Megabytes
1024 x 1024 x 128	268.434 Megabytes	67.108 Megabytes
512 x 512 x 64	33.554 Megabytes	8.388 Megabytes

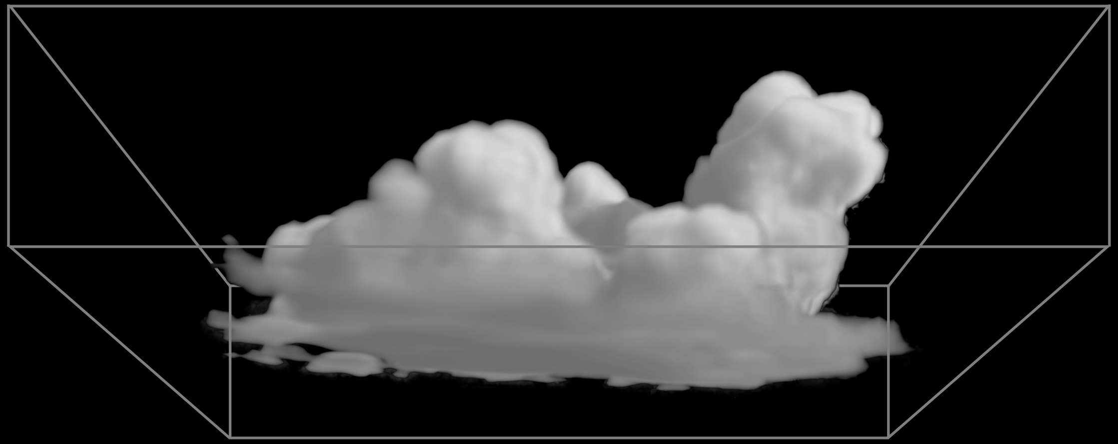
Voxel Cloud Modeling

Grow Clouds using Fluid Simulations.

Edit and composite them into “Frankencloudscapes.”

Store them in a voxel grid to be sampled at render time.


Nubis
Data Fields



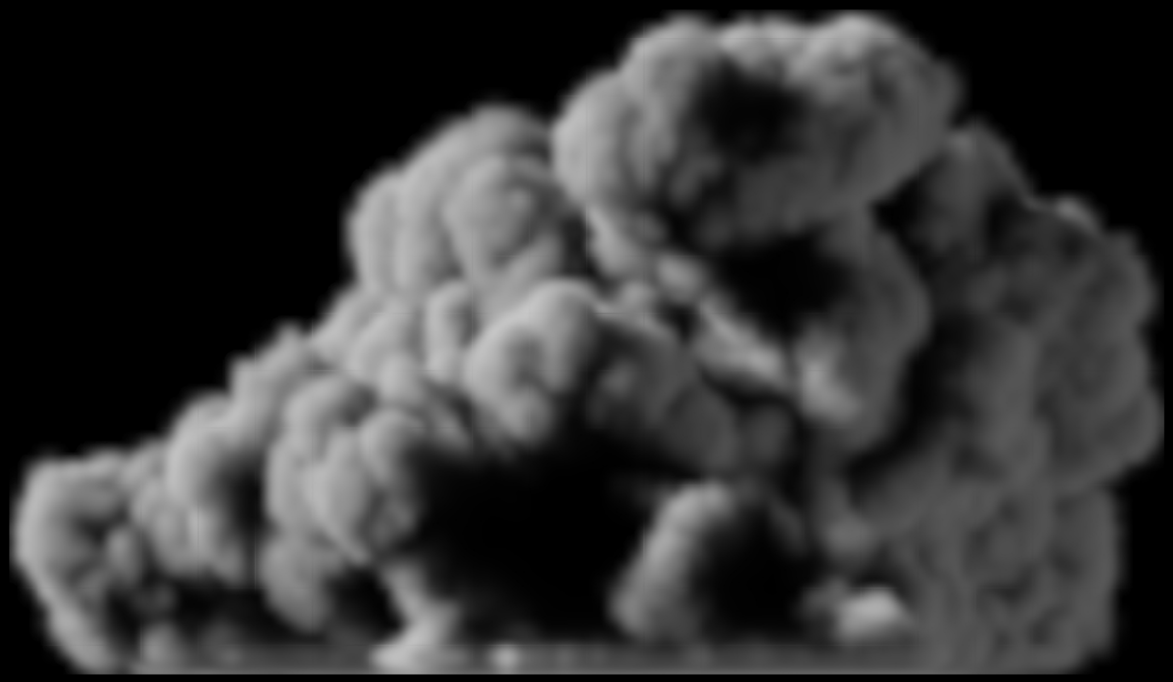
Dimensional Profile

Explicit Density

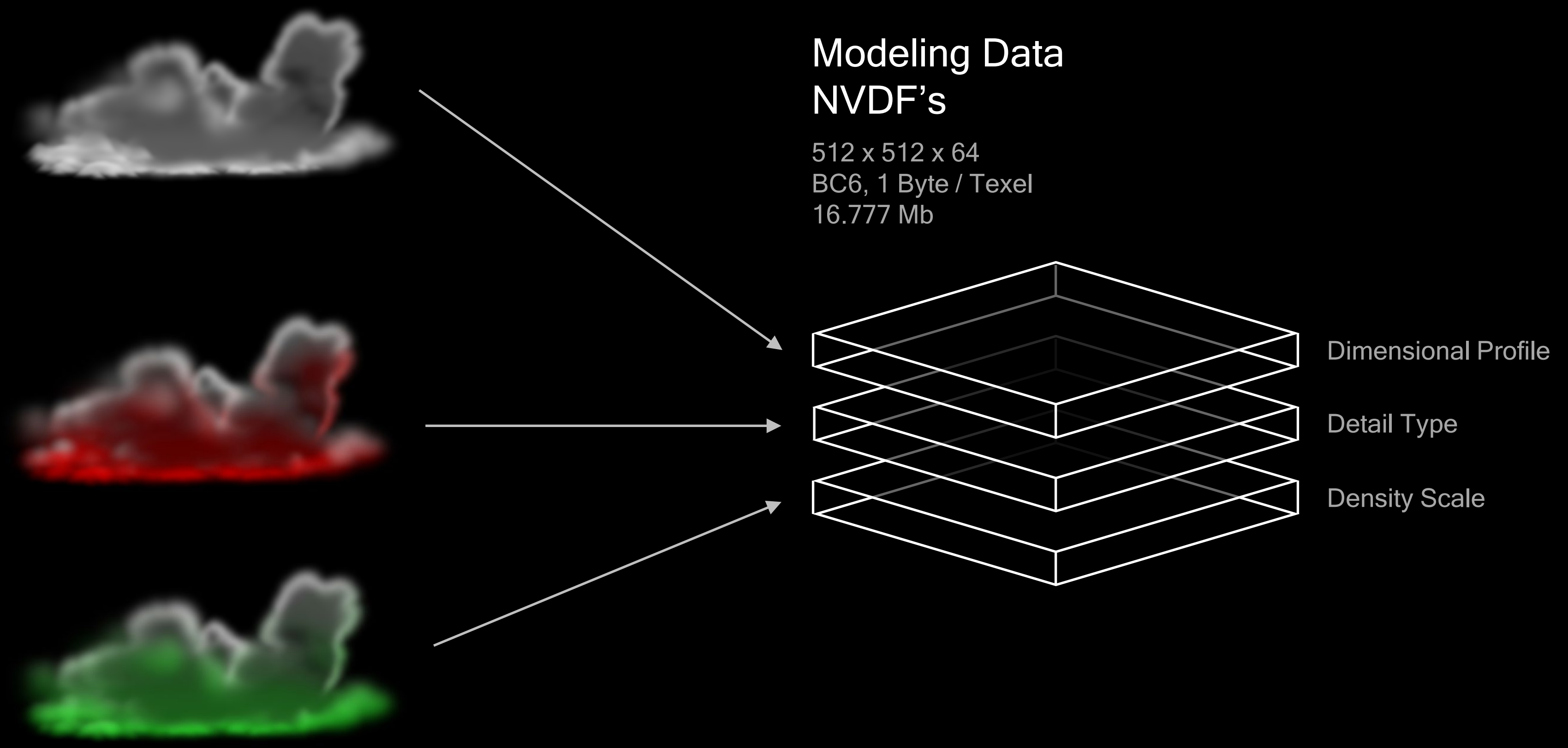
Voxel Size: 8 m

Voxel Size: 1 m

Dimensional Profile

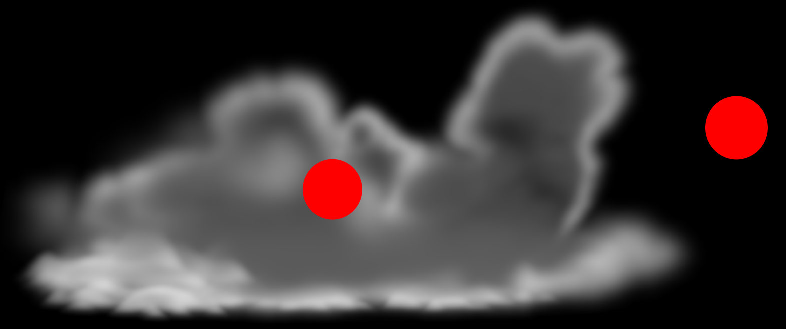


Dimensional Profile Cross Section


Modeling Data
NVDF's
512 x 512 x 64
BC6, 1 Byte / Texel
16.777 Mb
Dimensional Profile
Detail Type
Density Scale

Dimensional Profile

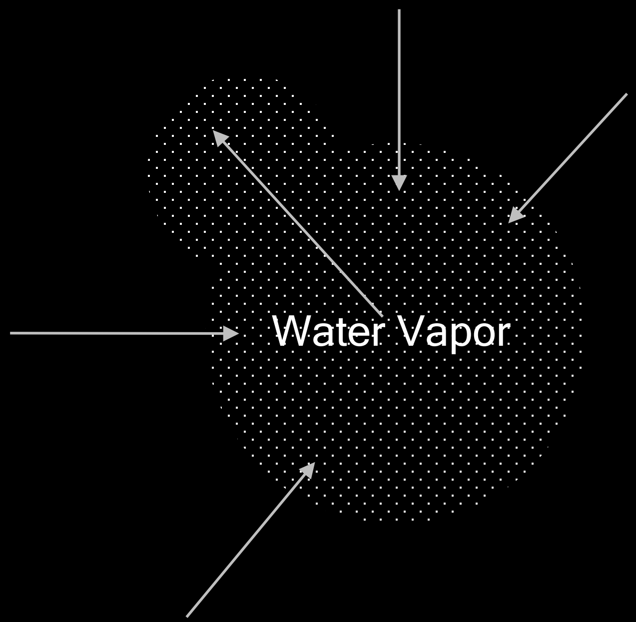
Voxel Size: 8 m



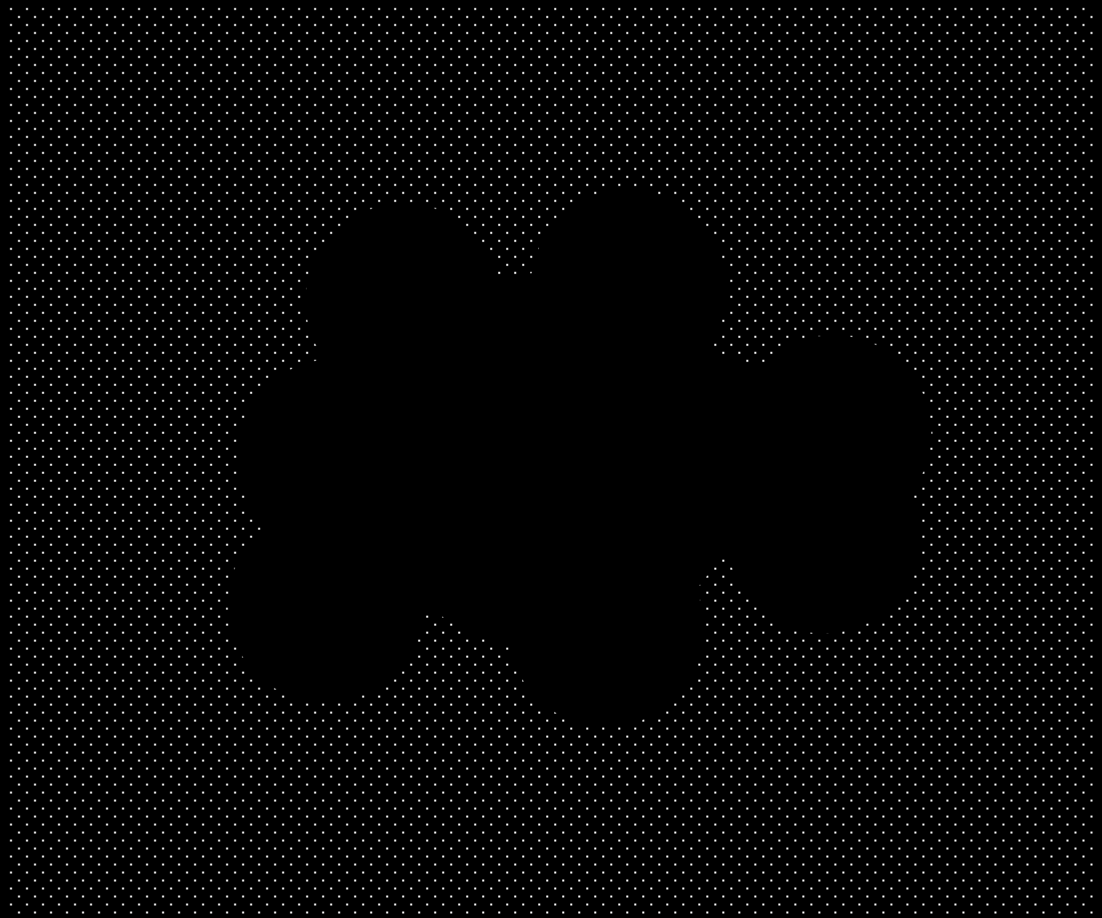
```
VoxelCloudModelingData modeling_data = GetVoxelCloudModelingData()

if (modeling_data.mDimensionalProfile > 0.0)
{
    cloud_density = GetUprezzedVoxelCloudDensity();
}
else
{
    cloud_density = 0.0;
}
```

Sony A7iii, 240mm ƒ6.3

Sony A7iii, 240mm ƒ6.3


Water Vapor

Nubis³ / Voxel Clouds / Sampling Density

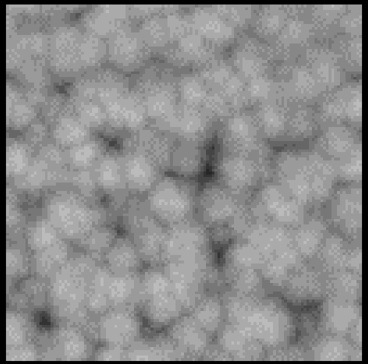
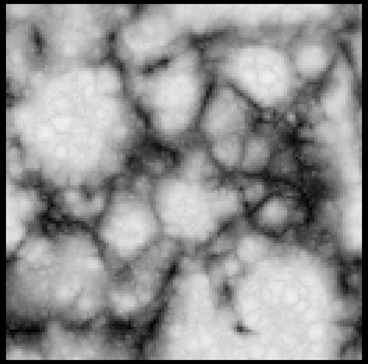
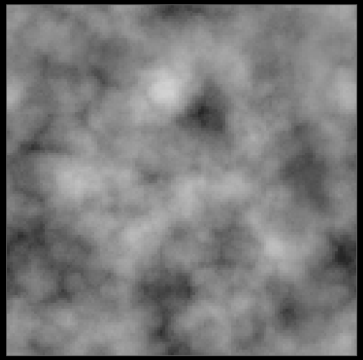
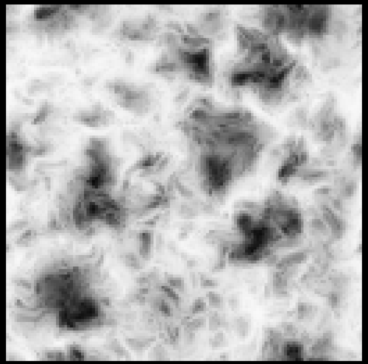



Sony A7iii, 240mm ƒ6.3

Decreasing Density = Curly Layered Wisps

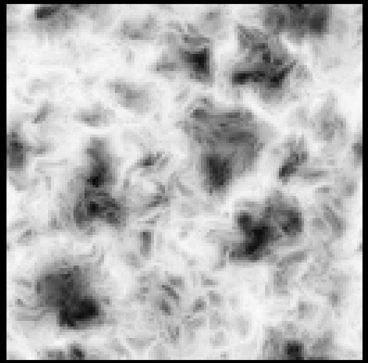
Increasing Density = Layered Billows

Nubis^3 / Voxel Clouds / Sampling Density

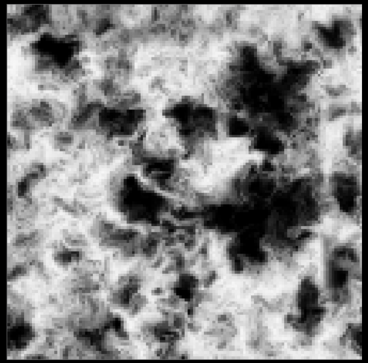
	Nubis Noise	Nubis^3 Noise
Billows	 Worley	 Alligator
Wisps	 “Perlin-Worley”	 “Curly-Alligator

Voxel Cloud Noise

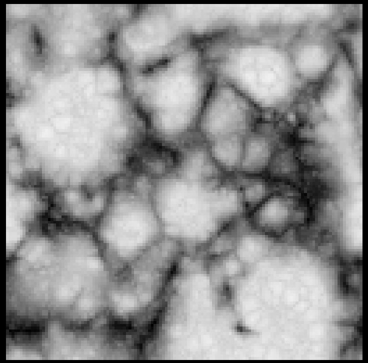
4 Channel
128 x 128 x 128 Voxels
Uncompressed, 2 Bytes / Texel
4.194 Megabytes



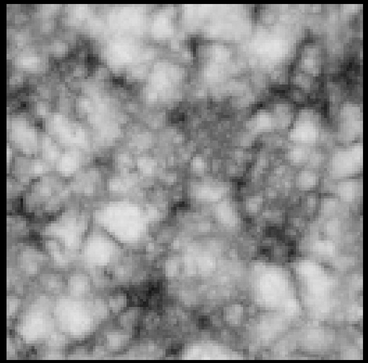
Low Freq “Curly-Alligator”



High Freq “Curly-Alligator”



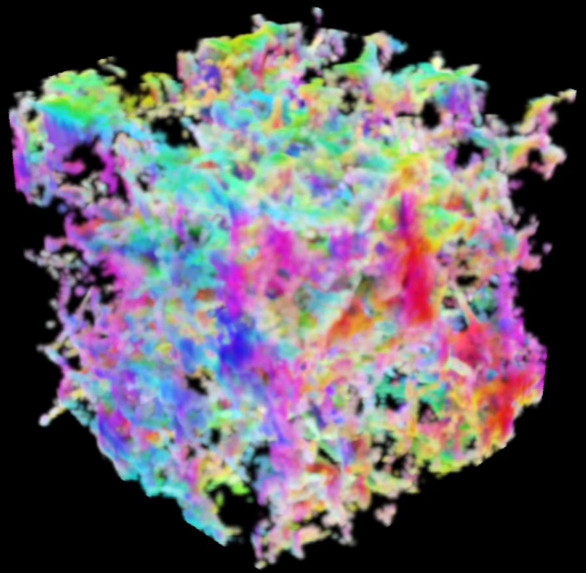
Low Freq Alligator



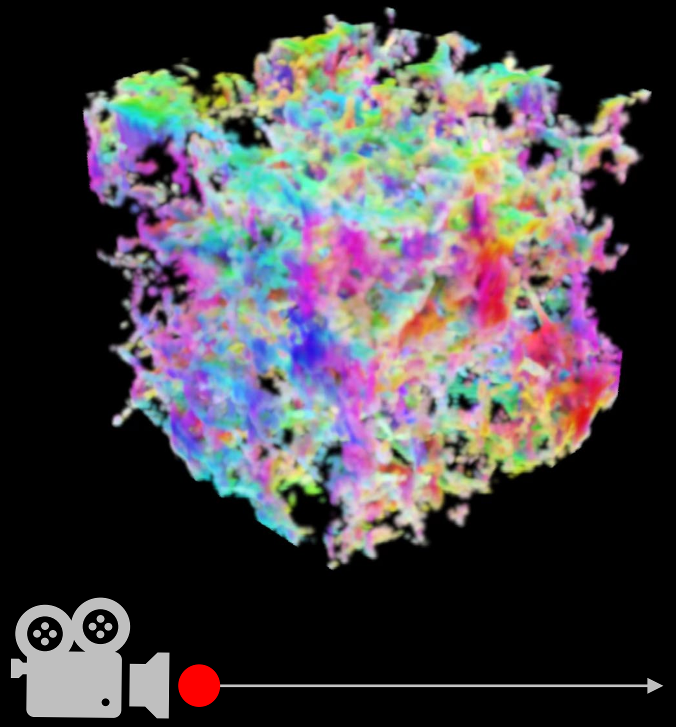
High Freq Alligator

Alligator Noise Code (via SideFX)

https://www.sidefx.com/docs/hdk/alligator_2alligator_8_c-example.html



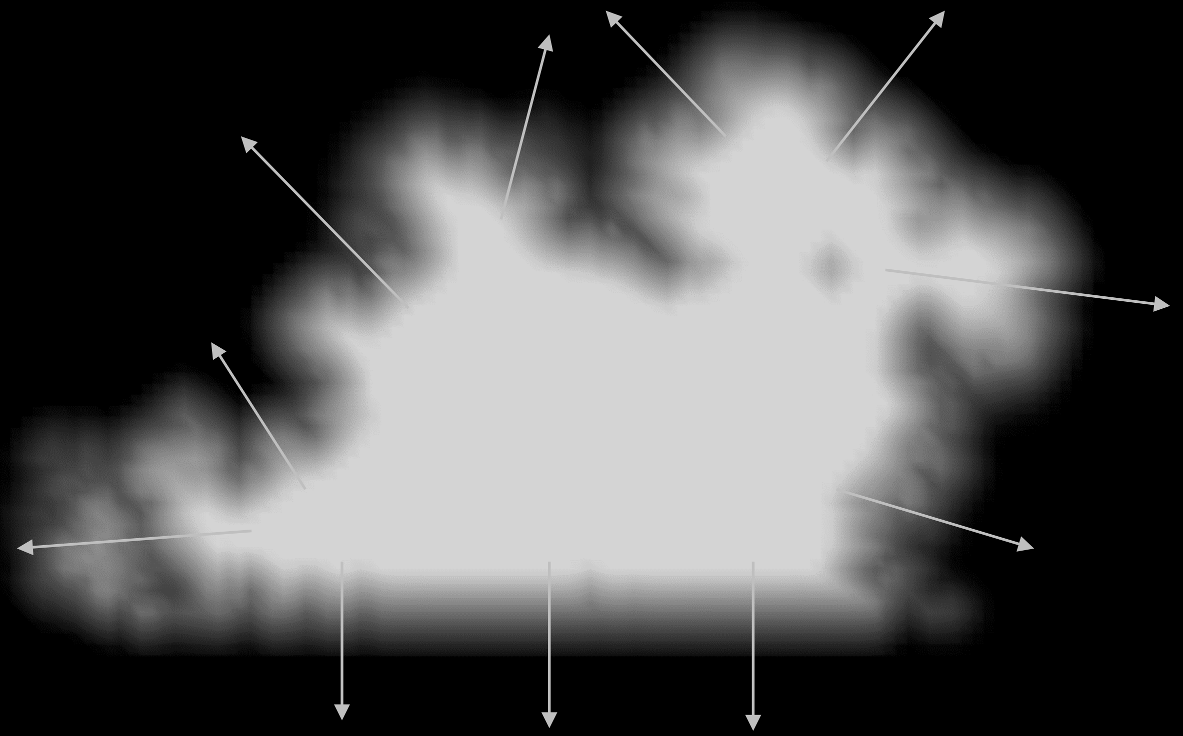
```
// Apply wind offset
inSamplePosition -= float3(cCloudWindOffset.x, cCloudWindOffset.y, 0.0) * voxel_cloud_animation_speed;
```



```
// Sample noise
float mipmap_level = log2(1.0 + abs(inRaymarchInfo.mDistance * cVoxelFineDetailMipMapDistanceScale));
float4 noise = Cloud3DNoiseTextureC.SampleLOD(Cloud3DNoiseSamplerC, inSamplePosition * 0.01, mipmap_level);
```

Sony A7iii, 240mm ƒ6.3

Dimensional Profile



Dimensional Profile

Low Frequency Wisps

High Frequency Wisps

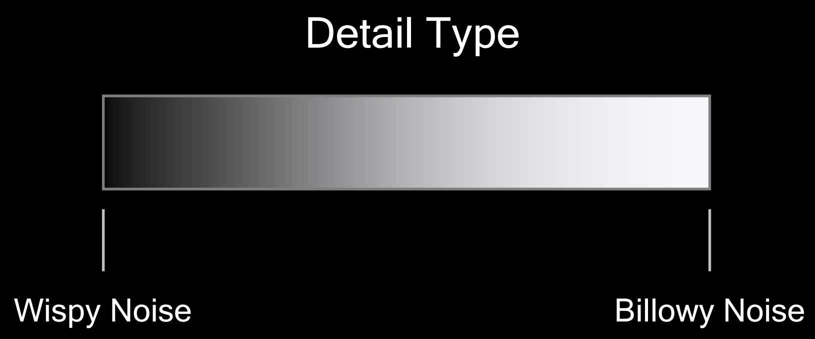
```
// Define wispy noise
float wispy_noise = lerp(noise.r, noise.g, inDimensionalProfile);
```

Canon KissX2, 80mm *f*6

Dimensional Profile

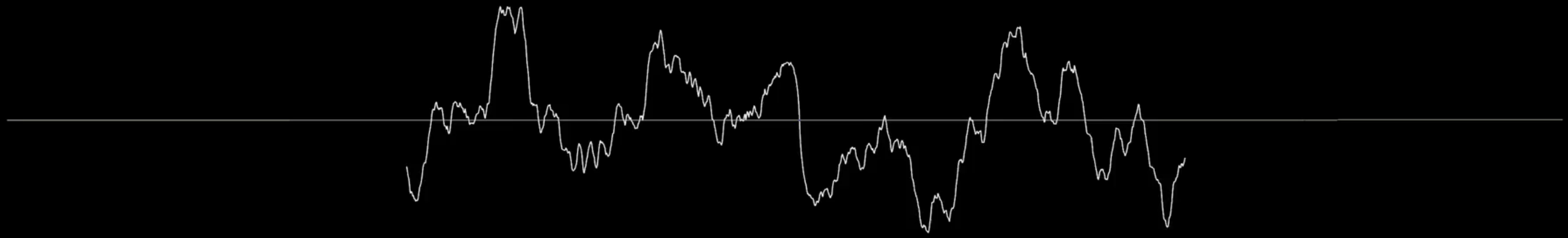

```
// Define billowy noise
float billowy_type_gradient = pow(inDimensionalProfile, 0.25);
float billowy_noise = lerp(noise.b * 0.3, noise.a * 0.3, billowy_type_gradient);

// Define Noise composite - blend to wispy as the density scale decreases.
float noise_composite = lerp(wispy_noise, billowy_noise, inType);
```

Nubis^3 / Voxel Clouds / Sampling Density




```
// Define Noise composite - blend to wispy as the density scale decreases.
float noise_composite = lerp(wispy_noise, billowy_noise, detail_type);
```

Nubis^3 / Voxel Clouds / Sampling Density



```
float hhf_wisps = 1.0 - pow(abs(abs(noise.g * 2.0 - 1.0) * 2.0 - 1.0), 4.0);
float hhf_billows = pow(abs(abs(noise.a * 2.0 - 1.0) * 2.0 - 1.0), 2.0);
```

Detail Type


High High Freq Wisps
High High Frequency Billows


150 meters
High High Freq Noise Composite
Noise Composite

```
// Composite Noises and use as a Value Erosion
float uprezzed_density = ValueErosion(inDimensionalProfile, noise_composite);

// Apply User Density Scale Data to Result
uprezzed_density *= inDensityScale;

// Sharpen result and lower Density close to camera to both add details and reduce undersampling noise
uprezzed_density = pow(uprezzed_density, lerp(0.3, 0.6, max(EPSILON, pow(saturate(inDensityScale), 4.0))));

// Return result
return uprezzed_density;
```

Voxel Cloud Density Sampler

Seamless high detail over distance.

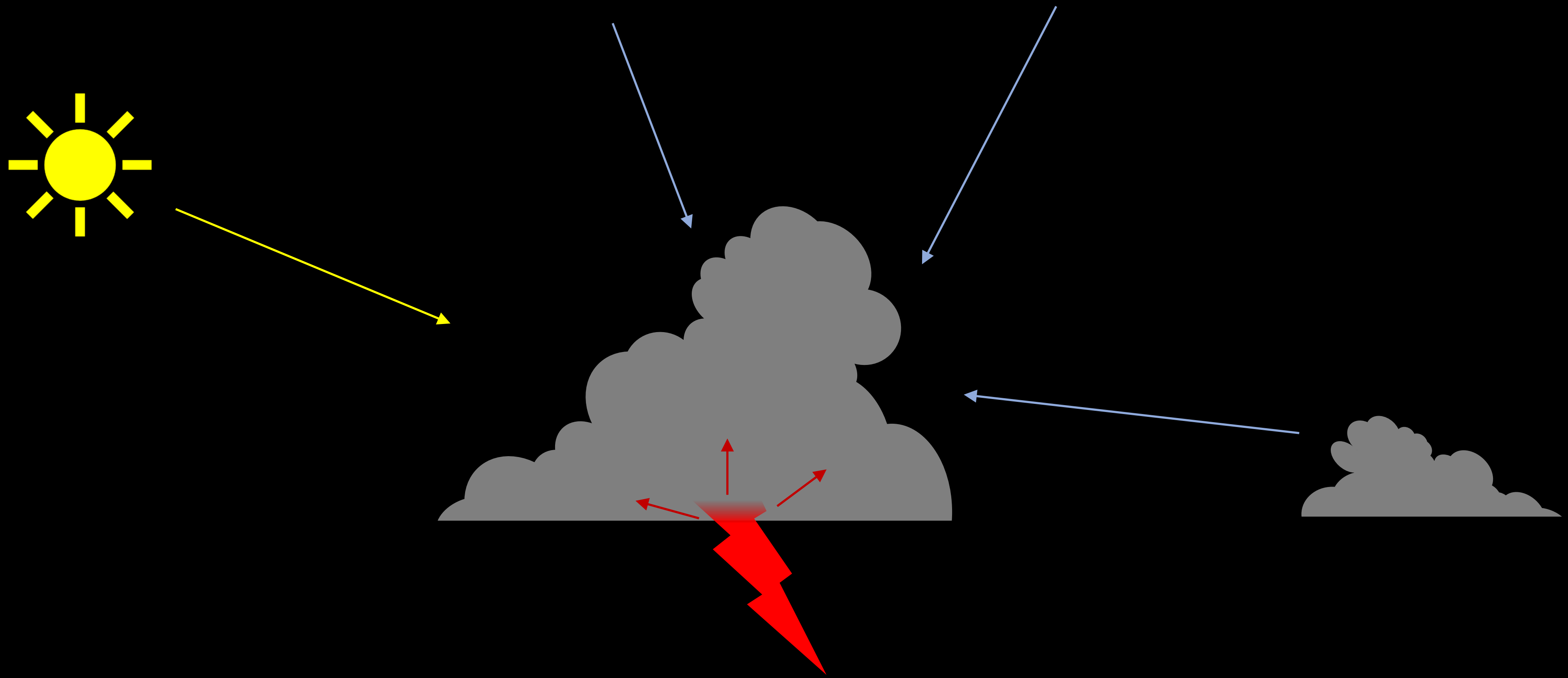
Up-rez avoids a memory bottleneck.

0.5 meter precision.

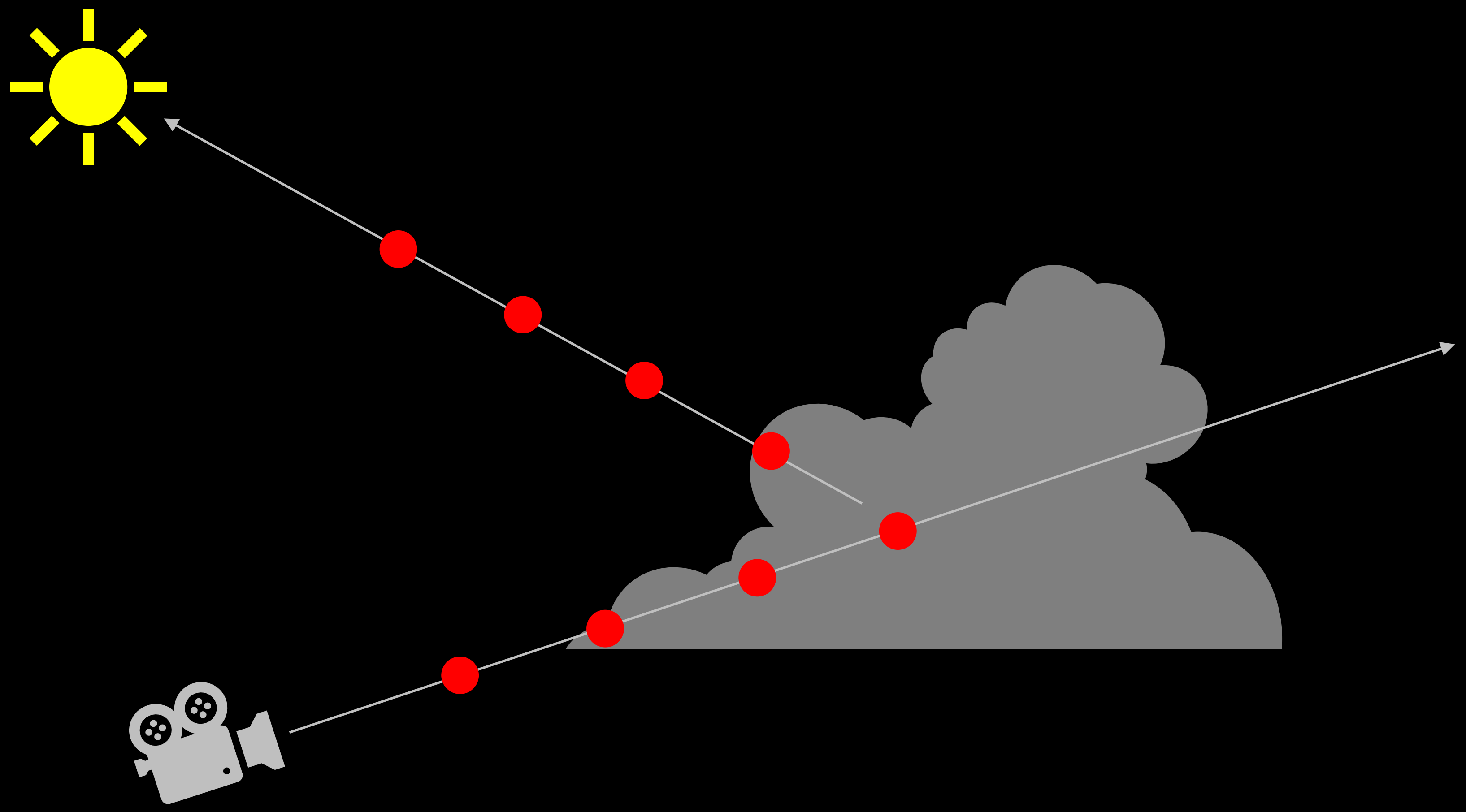
Balanced between memory and instructions

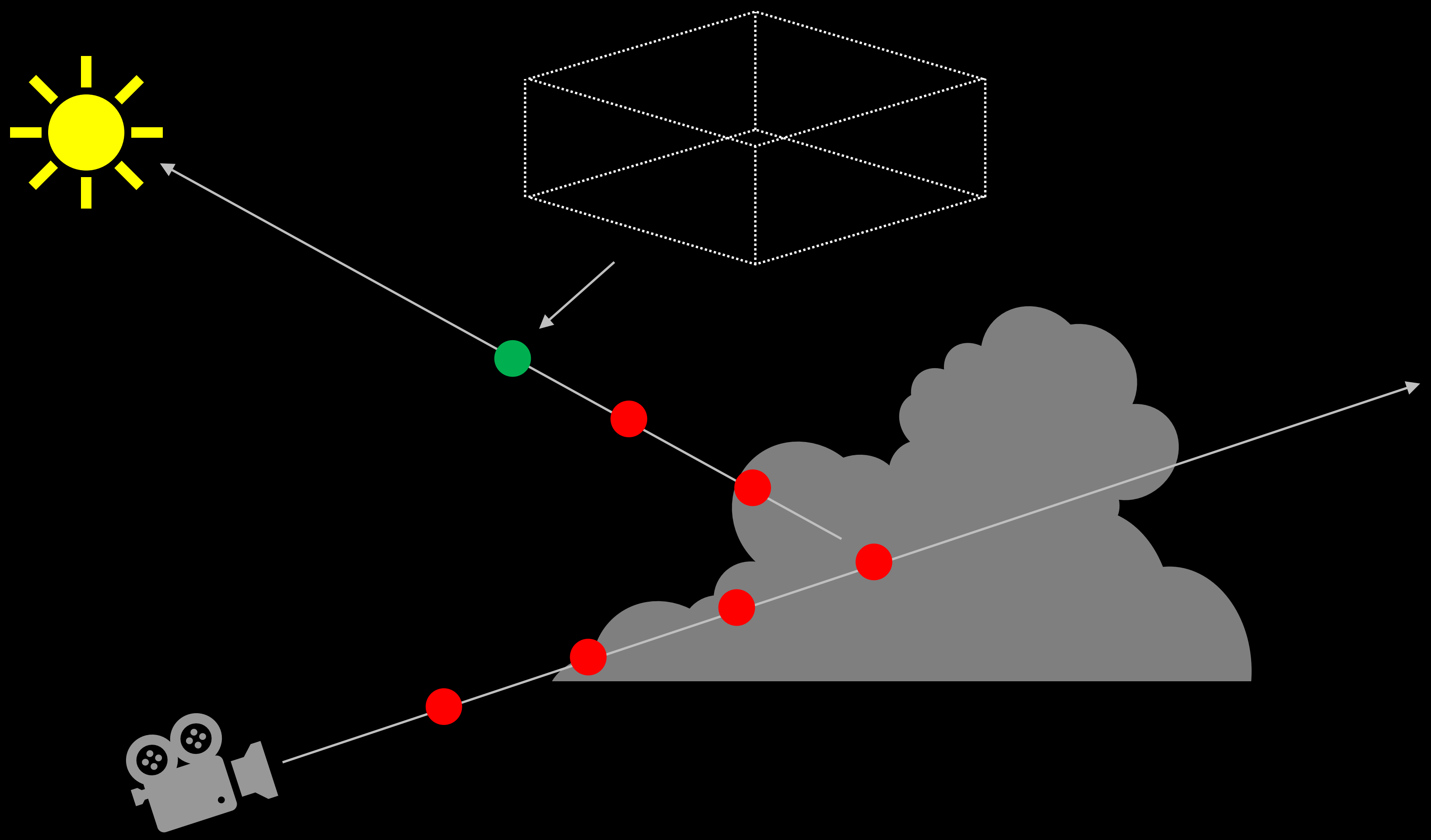
Current Cost: 10ms @ 960x540px

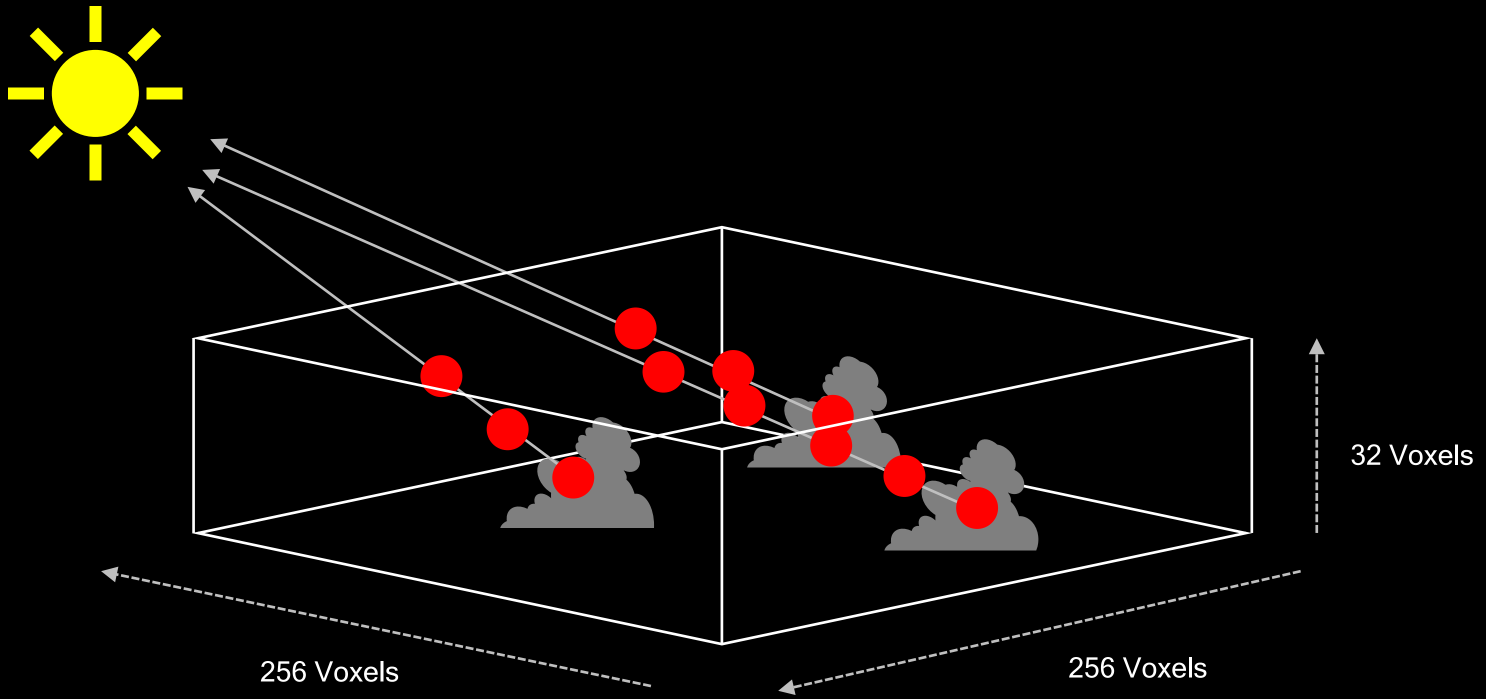
Dense Grid Sampling: 30+ms



Light Energy = Direct Scattering + Ambient Scattering + Secondary Scattering






32 Voxels
256 Voxels
256 Voxels

Decima Engine, 2023

Decima Engine, 2023

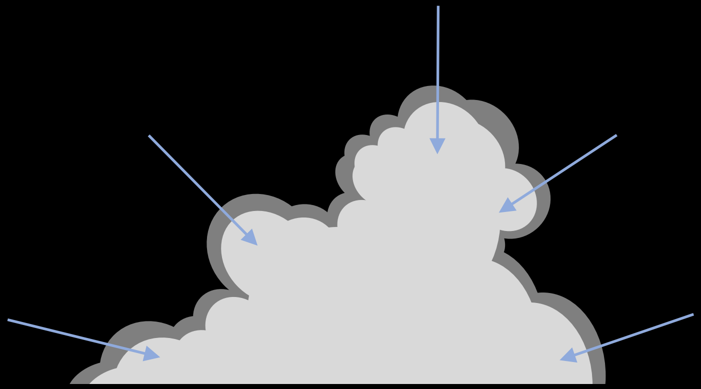
Sony A7iii, 240mm ƒ6.3

Sony A7iii, Konica Hexanon 50mm *f*2.8

Canon KissX2, 80mm *f*6

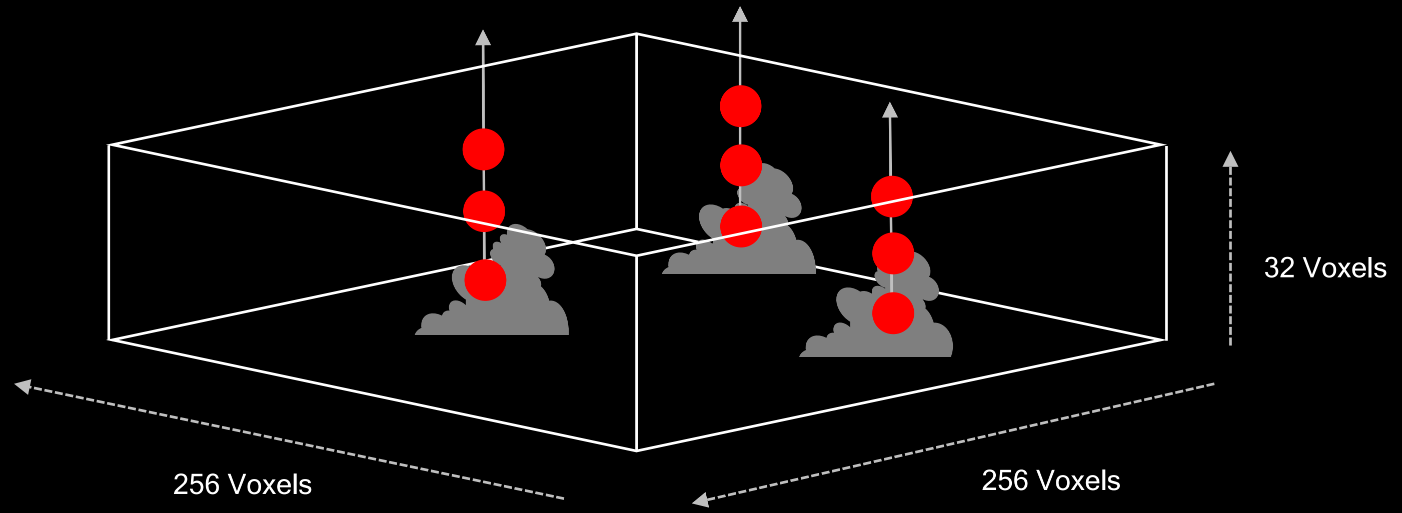
Sony A7iii, Konica Hexanon 50mm ƒ2.8

```
     ms_volume = dimensional_profile;
cloud_distance = GetVoxelCloudDistance(inSamplePosition);
    ms_volume *= exp(-inSunLightSummedDensitySamples * Remap(sun_dot, 0.0, 0.9, 0.25, ValueRemap(cloud_distance, -128.0, 0.0, 0.05, 0.25)));
```

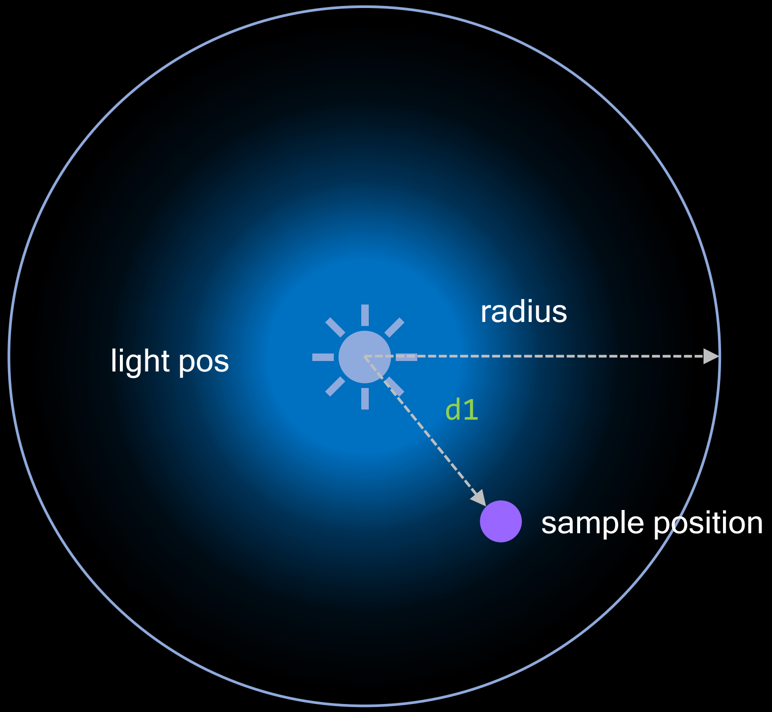


Nubis Ambient Scattering Approximation

```
float ambient_scattering = pow(1.0 - dimensional_profile, 0.5);
```




```
float ambient_scattering = pow(1.0 - dimensional_profile, 0.5) * exp(-summed_ambient_density);
```

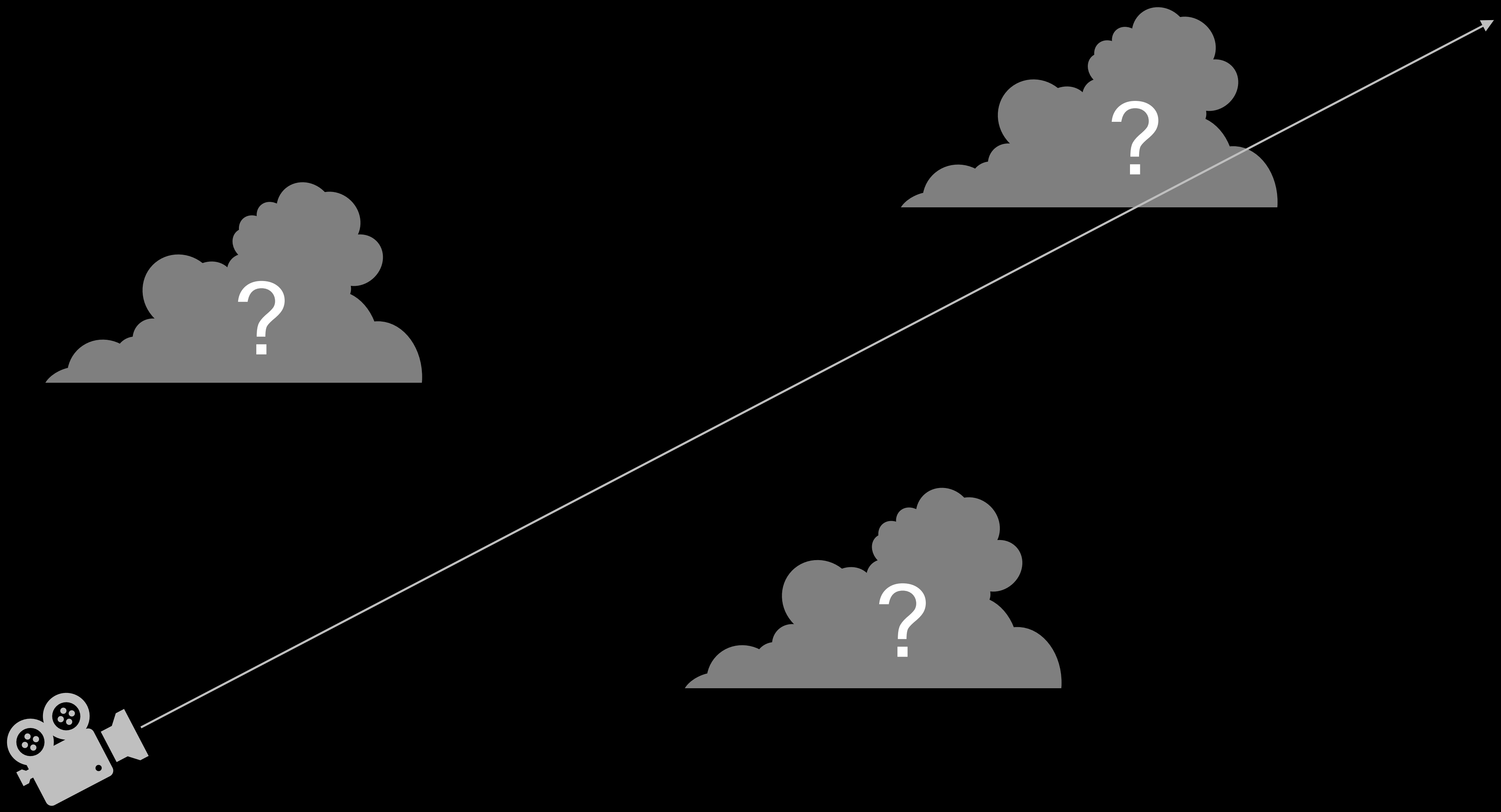



```
   potential_energy = pow( 1.0 – (d1 / radius), 12.0);

pseudo_attenuation = (1.0 – saturate(density * 5.0));

       glow_energy = potential_energy * pseudo_attenuation;
```

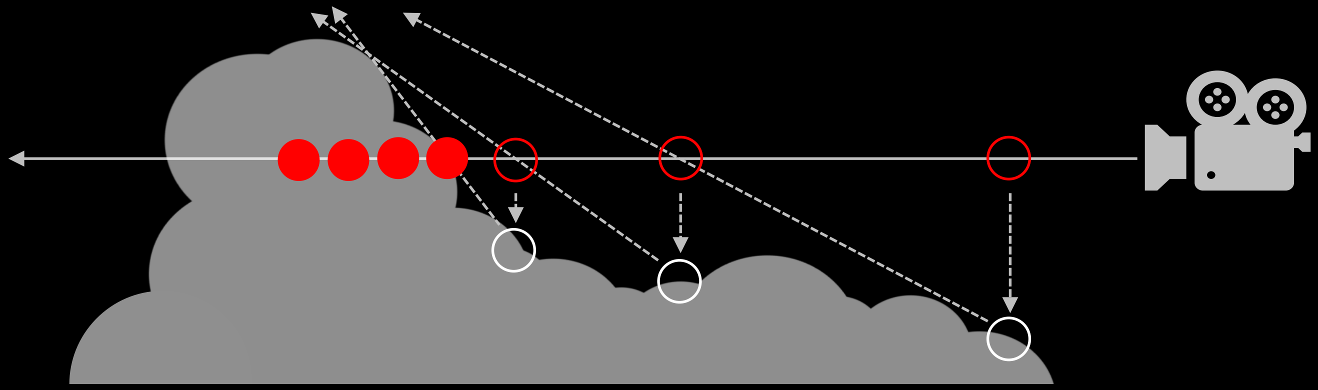
Decima Engine, 2023

Voxel Cloud Lighting

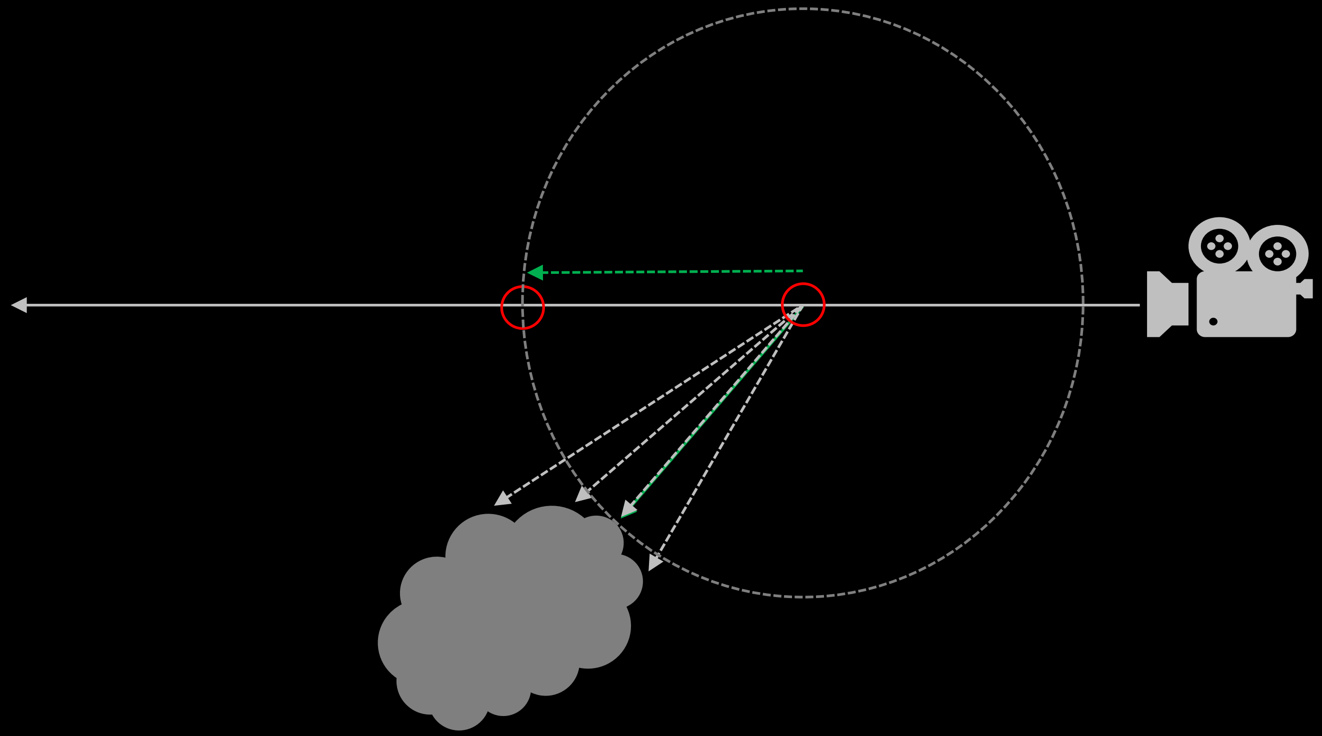
- Calculate and store summed density for some samples
- Amortized Cost: 0.1 to 0.2ms every 8 frames
- Reduces render time by 40%
- Better Results: Long Distance Shadows
- This approach was used to improve ambient scattering
- Many other approximations were simplified as a result of adopting voxels.



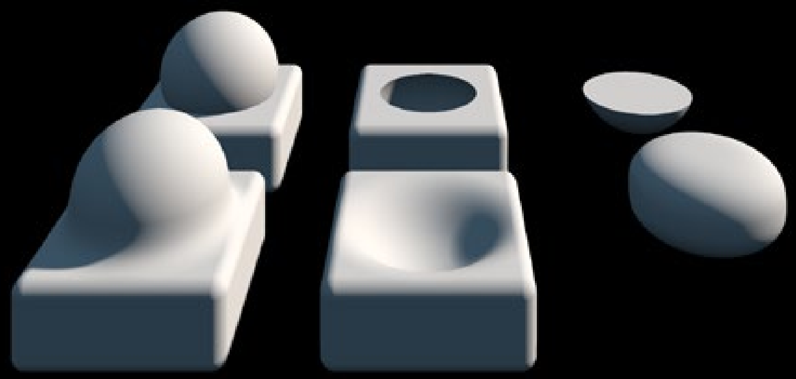
Distance Step Mapping + Cone Step Mapping



Jonathan “Lone Sock” Dummer, Cone Step Mapping: An Iterative Ray Heightfield Intersection Algorithm. 2006.



John Hart, “Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces”, *The Visual Computer*, June, 1995.

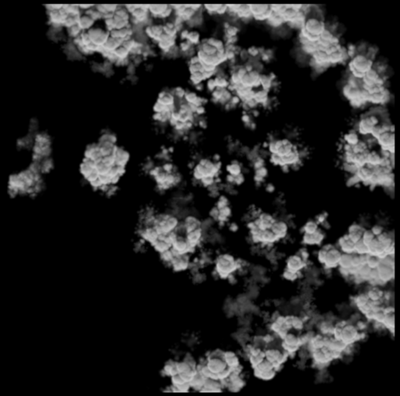


https://iquilezles.org/articles/

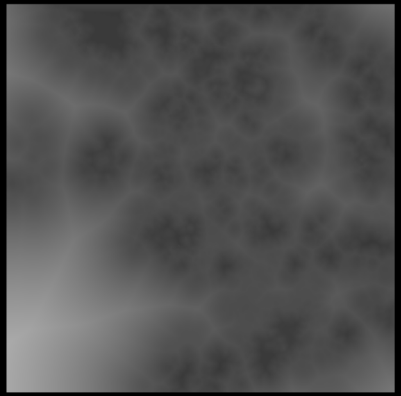
https://www.secondorder.com/publications

Nubis^3 / Voxel Clouds / Ray-Marching

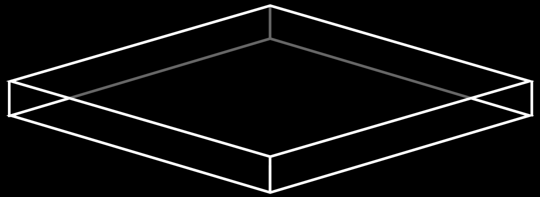
Frankencloudscape
Top View



SDF
Top View

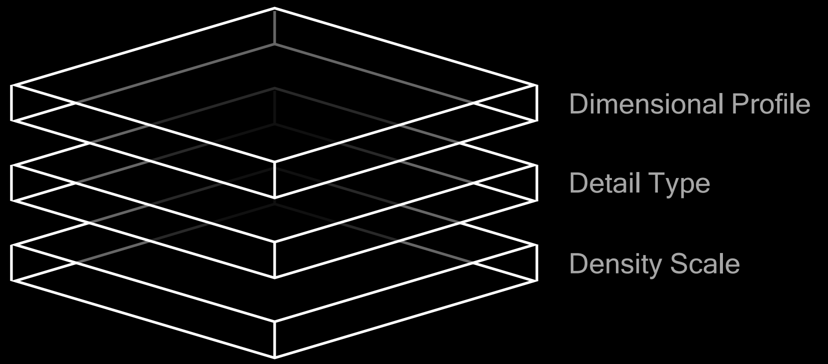


Field Data
NVDF
512 x 512 x 64 Voxels
-256 m to 4096 m



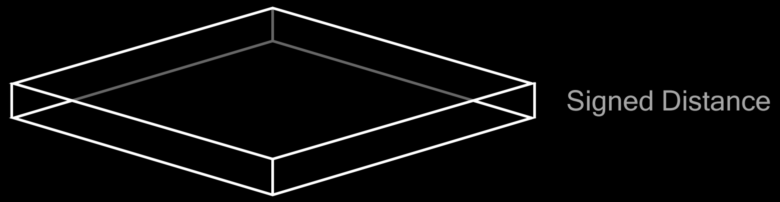
Modeling
NVDF's

512 x 512 x 64 Voxels
BC6, 1 Byte / Texel
16.777 Mb



Field Data
NVDF

512 x 512 x 64 Voxels
Compression ?



Source:

https://www.reedbeta.com/blog/understanding-bcn-texture-compression-formats

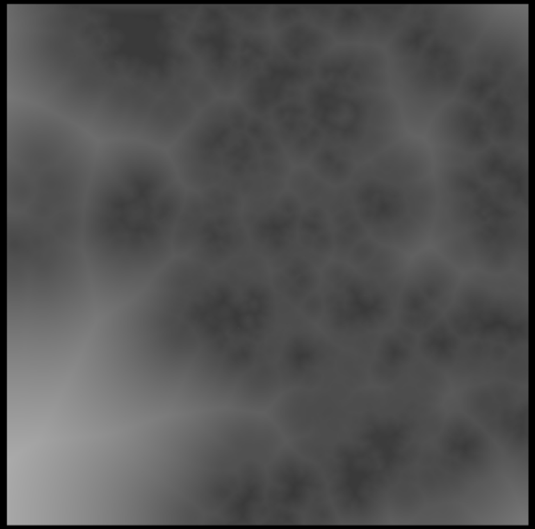
Type Of Data	Data Rate	Palette Size	Line Segments	Use For	
BC1	RGB + optional 1-bit alpha	0.5 byte/px	4	1	Color maps Cutout color maps (1-bit alpha) Normal maps, if memory is tight
BC2	RGB + 4-bit alpha	1 byte/px	4	1	n/a
BC3	RGBA	1 byte/px	4 color + 8 alpha	1 color + 1 alpha	Color maps with full alpha Packing color and mono maps together
BC4	Grayscale	0.5 byte/px	8	1	Height maps Gloss maps Font atlases Any grayscale image
BC5	2 × grayscale	1 byte/px	8 per channel	1 per channel	Tangent-space normal maps
BC6	RGB, floating-point	1 byte/px	8-16	1-2	HDR images
BC7	RGB or RGBA	1 byte/px	4-16	1-3	High-quality color maps Color maps with full alpha

Precision Loss Symptoms

- Too low = extra steps
- Too High = rendering artifacts

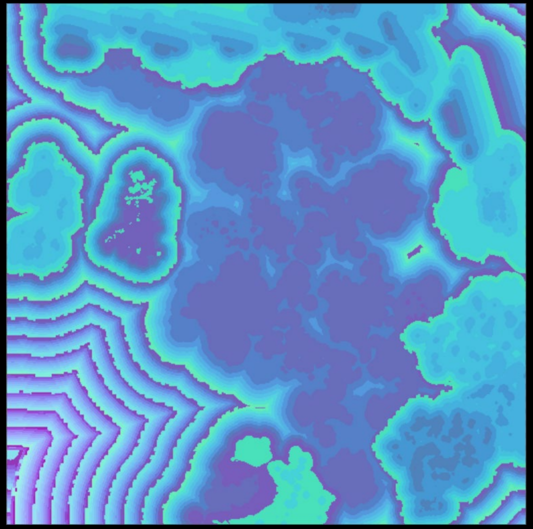
SDF

Uncompressed, 2 Bytes / Texel



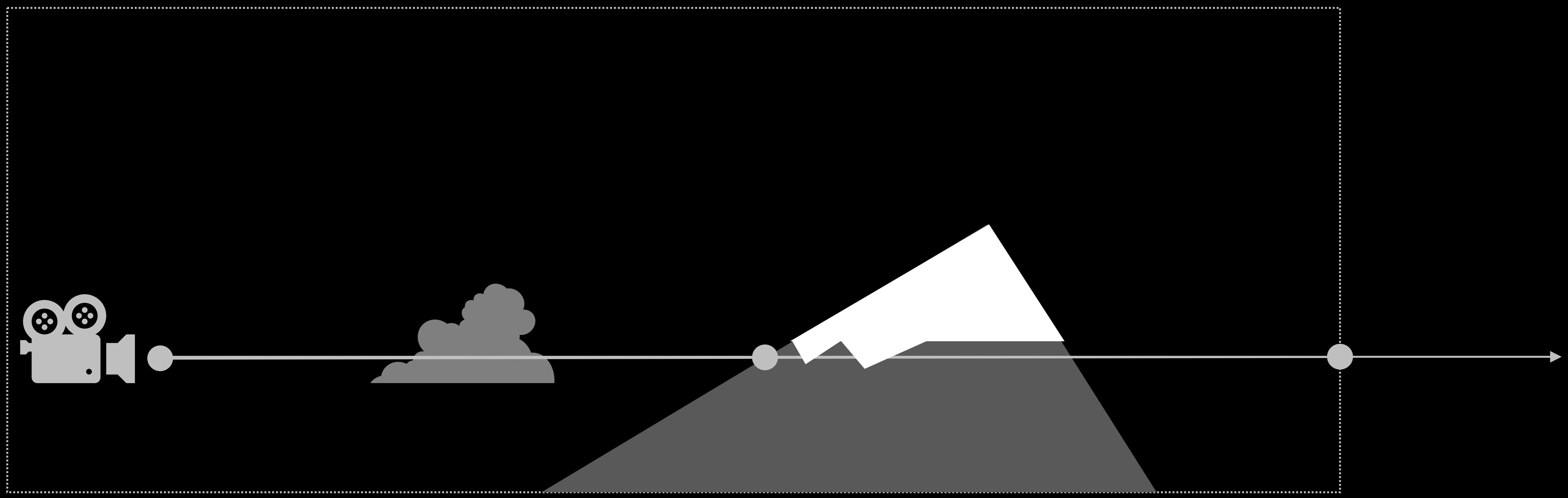
SDF

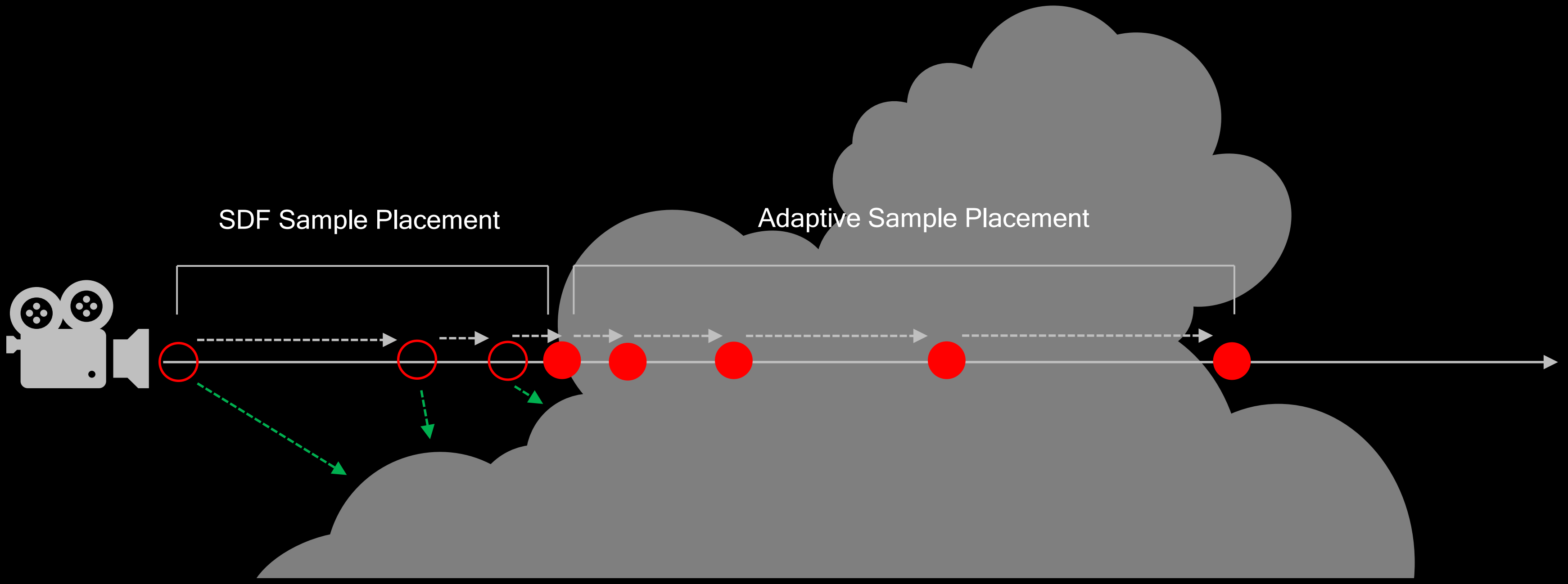
BC1, 0.5 Bytes / Texel



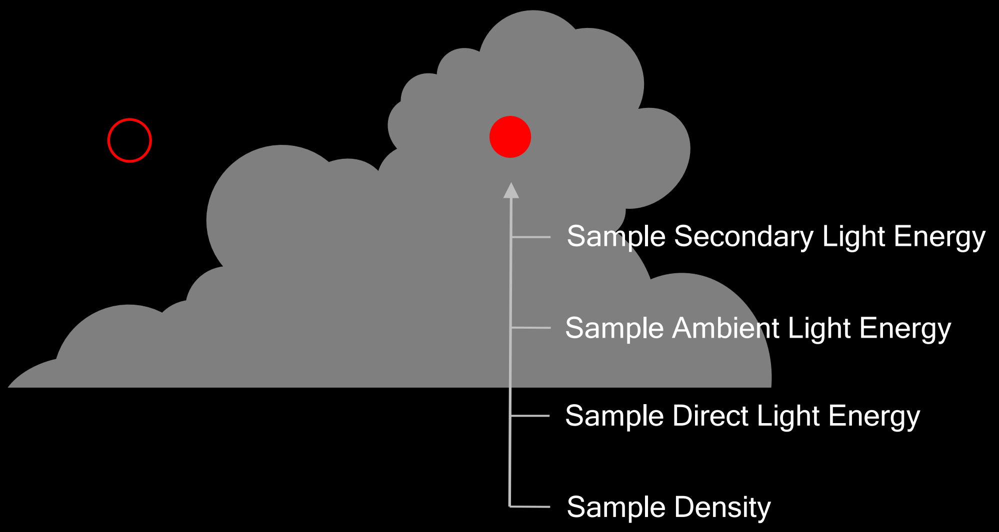
```
// Decompress BC1
dot(sampled_color.rgb, float3(1.0, 0.03529415, 0.00069204))
```

	Vertical Profile Method	Envelope Method	Voxel Method
Memory Per Cloudscape	0.541 Mb	9.437 Mb	25.166 Mb

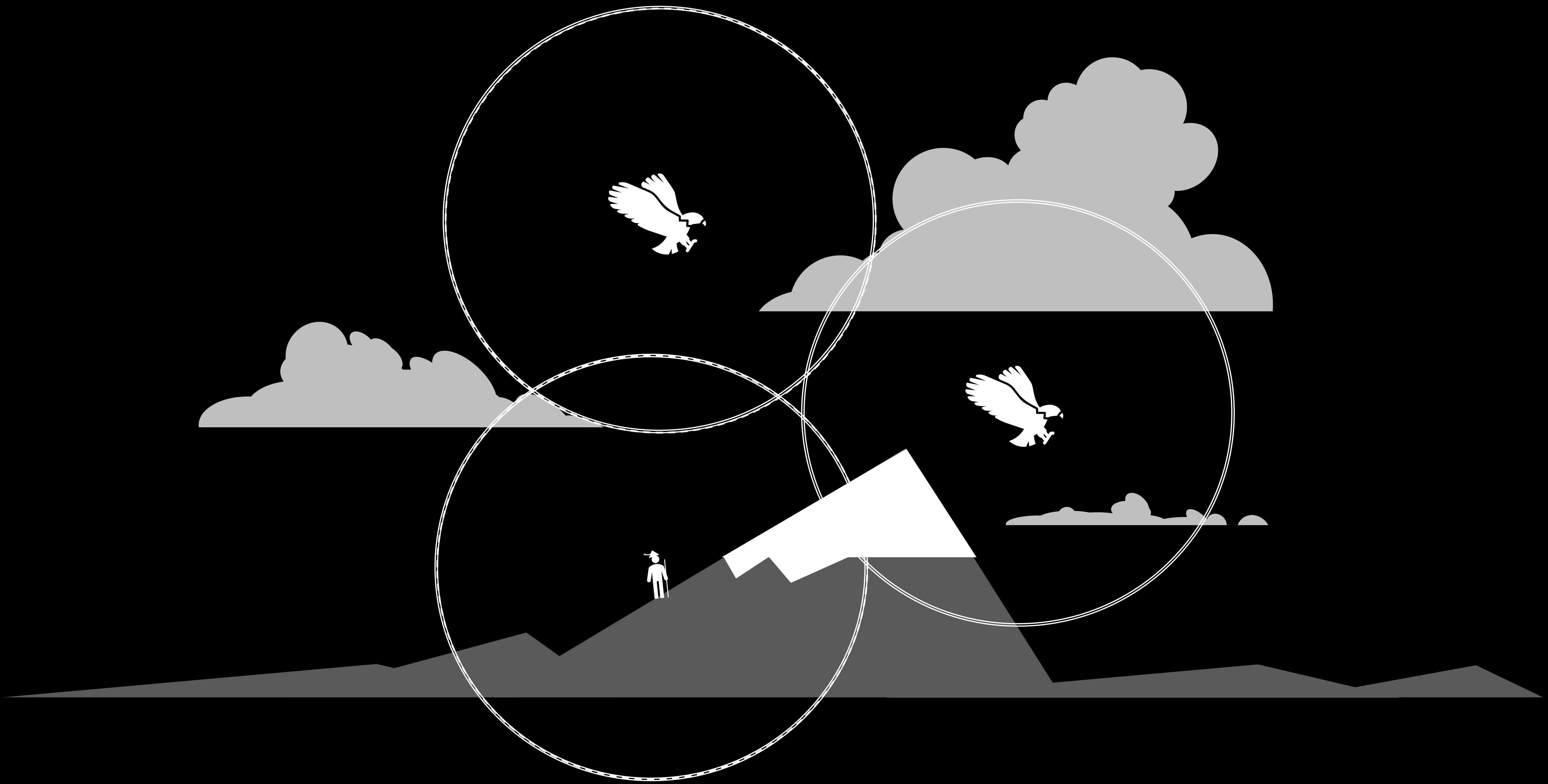





```
sdf_cloud_distance = GetVoxelCloudDistance(sample_pos);

adaptive_step_size = max( 1.0, max(sqrt(distance_from_camera), EPSILON) * 0.08);

     jitter_offset = distance_from_camera < 250.0 ? animated_hash : static_hash

         step_size = max(sdf_cloud_distance, adaptive_step_size) + jitter_offset;

        sample_pos = distance_from_camera + view_direction * step_size;
```




```
if (sdf_cloud_distance < 0.0)
{
    voxel_cloud_sample_data = GetVoxelCloudSampleData();

    IntegrateCloudSampleData(voxel_cloud_sample_data, pixel_data);
}
```



Optimizations	Cloud Cost	Geometry Cost
Base	10	7
+ Voxel-Based Lighting	6.1	7
+ Adaptive & SDF Ray-March	2.2	7

Decima Engine, 2023

Optimizations	Cloud cost	Geometry Cost
Base	12	5
+ Voxel-Based Lighting	8.2	5
+ Adaptive & SDF Ray-March	4.0	5

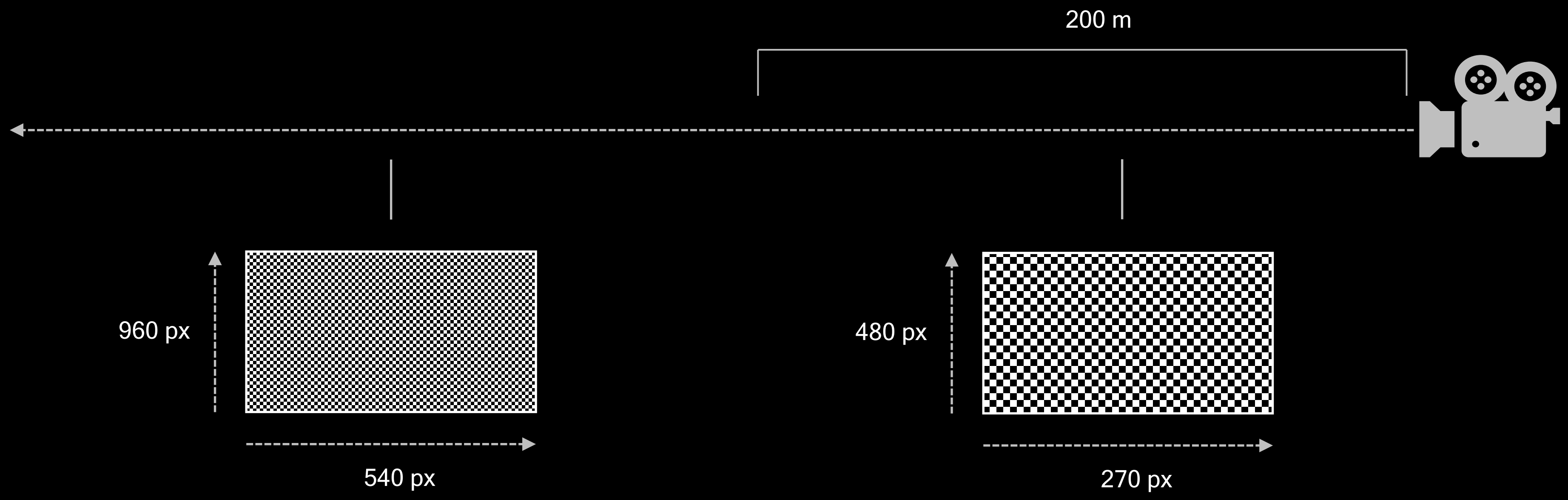
Optimizations	Cloud cost	Geometry Cost
Base	10	4
+ Voxel-Based Lighting	8.0	4
+ Adaptive & SDF Ray-March	4.0	4

Decima Engine, 2023

Voxel Cloud Ray-Marcher

- Uses Compressed SDF to avoid memory bottlenecks
- Hybrid SDF, Adaptive and Jittered samples
- Cost: 2.2 to 4 Milliseconds
- Performance scales

Decima Engine, 2023


200 m
960 px
540 px
480 px
270 px

Mode	Cost (in milliseconds)
30hz @ 960 x 540	4
40hz & 60hz @ 960 x 540 & 480 x 270	2.1

30hz Mode

60hz Mode

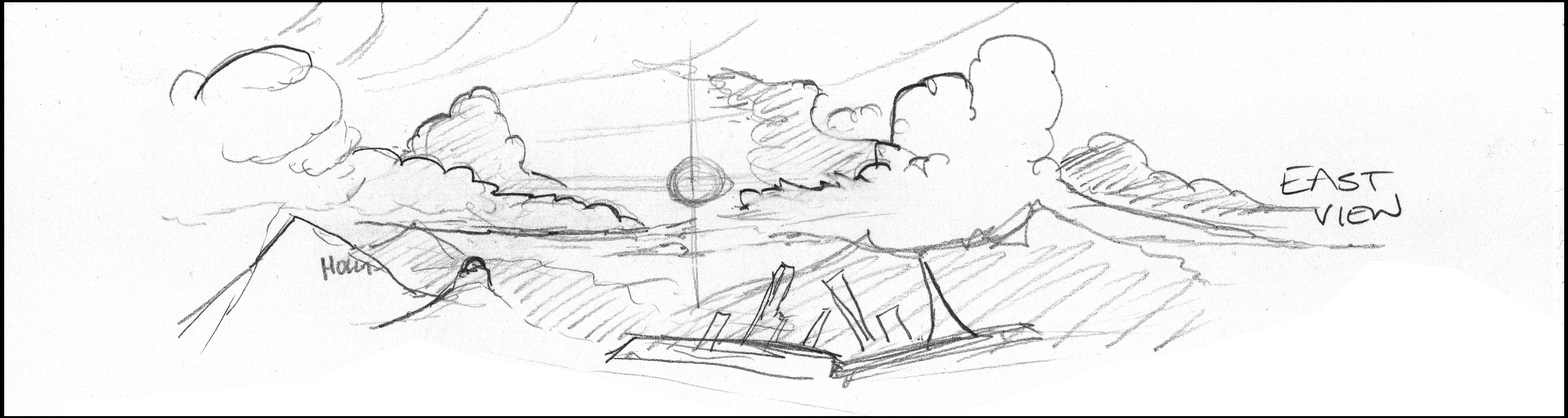
Voxel Cloud Renderer

Render Split into 2 passes:
- < 200 Meters: 480px * 270px
- > 200 Meters: 960px * 540px

Saves around 50%

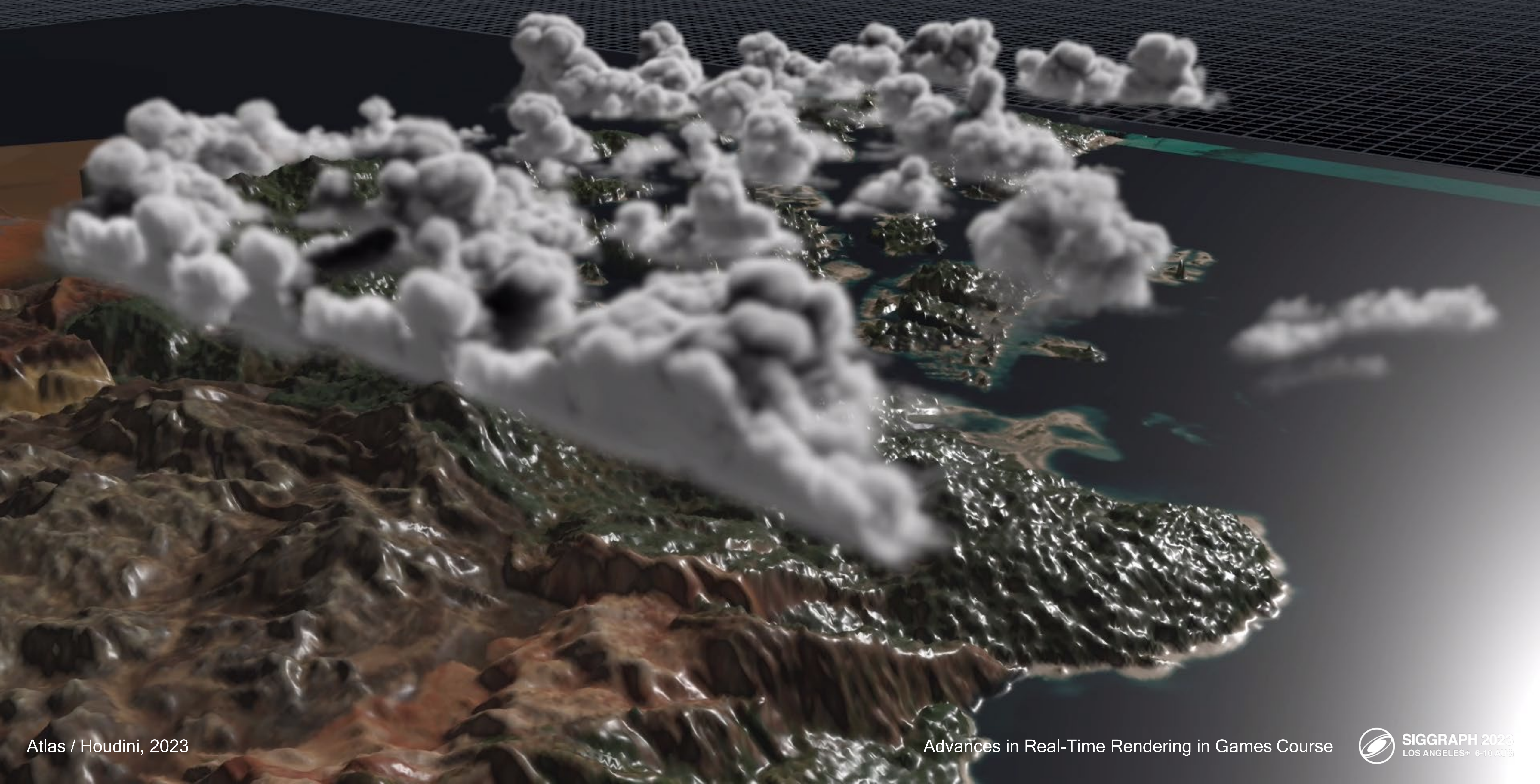
40hz Mode uses the same method

Development Build, Decima Engine, 2023



'Chickenscratch' Layout

Samsung Galaxy S21Ultra



Atlas / Houdini, 2023

Nubis^3 / Voxel Clouds / Production

Decima Engine, 2023

Houdini / Decima Engine, 2023

Voxel Clouds in Production

- Frankenclouding Works.
- Paradigm shift in terms of workflow
- Cinematics Memory benefits from Re-use
- Bespoke cloudscapes for Boss fights/etc are easy

	Vertical Profile Method	Envelope Method	Voxel Method
Evolution	Yes	Pseudomotion Only	Pseudomotion Only
Time Of Day	Yes	Yes	Yes
Lightning	Yes	No	Yes
Terrain-Cast Shadows	No	Yes	Yes
High Frame-rates	Yes	Yes	Yes
Flight-Capable	No	Yes	Yes
Freeform Modeling	No	No	Yes

Cloud Tasks

- Cloud Exterior + lightning
- Cloud Interior (Not Just a Cave)
- Dissipate After Stormbird Death

Decima Engine, 2022

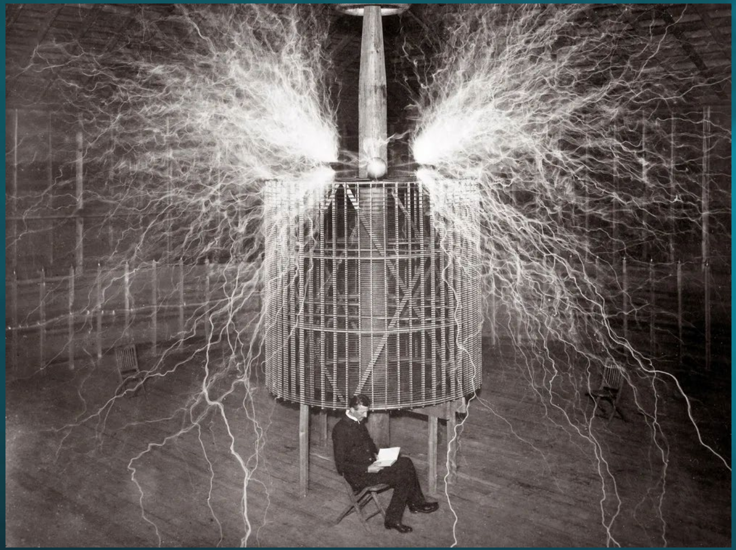
Nubis^3 / The Stormbird Encounter

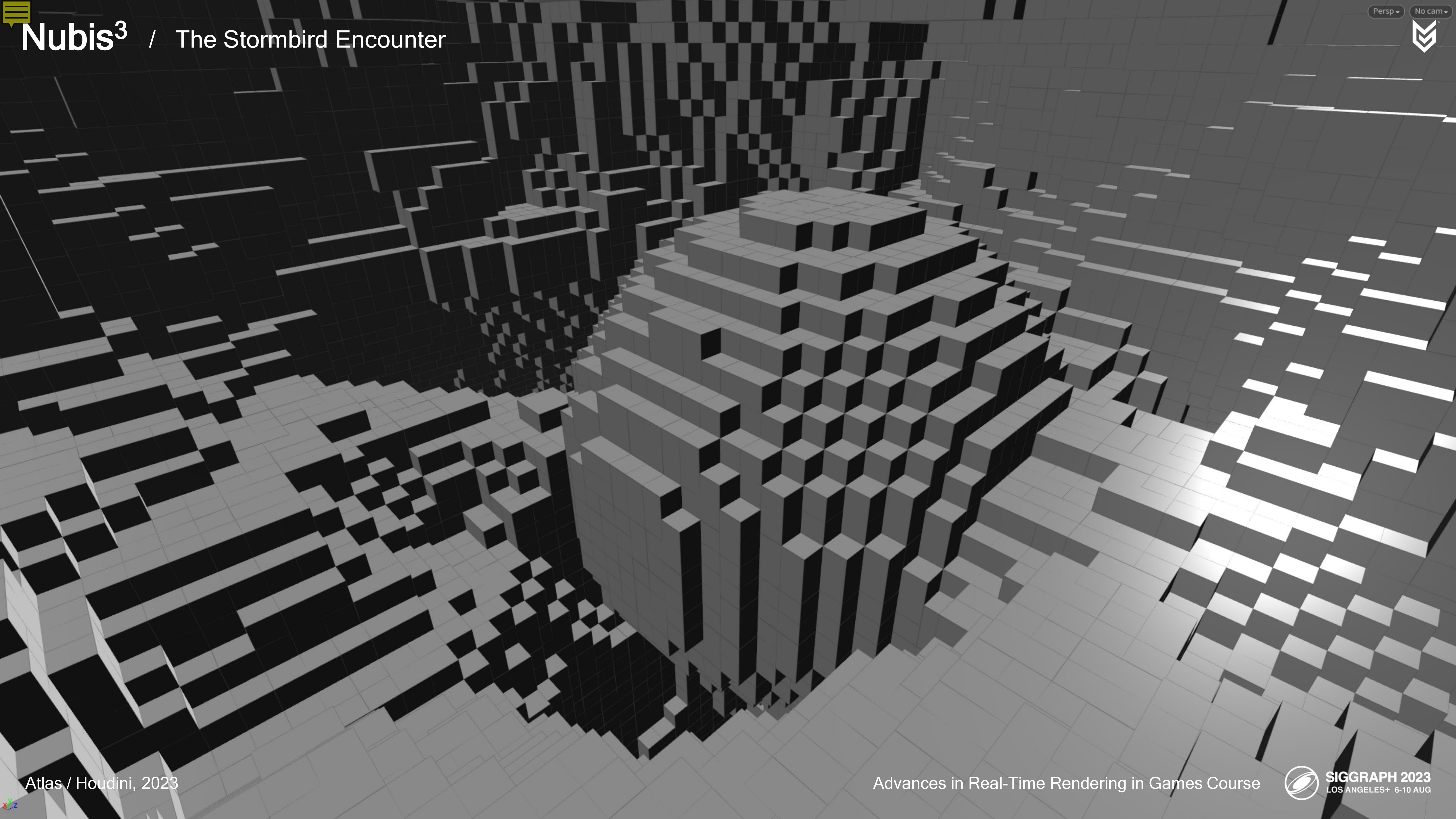
Decima Engine, 2023

Nubis³ / The Stormbird Encounter

Decima Engine, 2023

Nubis^3 / The Stormbird Encounter





Atlas / Houdini, 2023

The Stormbird Encounter

- Ease of 3d modeling
- Existing Lightning Tech
- Existing Local Weather System from Superstorms
- Collaboration with Quest Design and Audio Team

Nubis[3] / Conclusions

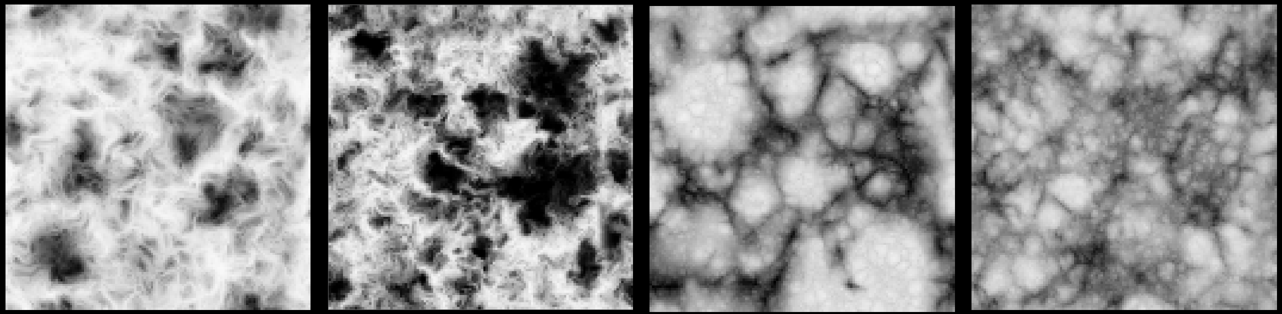
	Vertical Profile Method	Envelope Method	Voxel Method
Evolution	Yes	Pseudomotion Only	Pseudomotion Only
Time Of Day	Yes	Yes	Yes
Lightning	Yes	No	Yes
Terrain-Cast Shadows	No	Yes	Yes
High Frame-rates	Yes	Yes	Yes
Flight-Capable	No	Yes	Yes
Freeform Modeling	No	No	Yes
Support Quests	No	Ehhhh…	**Yes**
Potential for Growth	No	Not Really	**Very Yes**

NVDF's

Parkouring Cloud + Stormbird Cloud
TGA's + VDB

Voxel Cloud Noise

TGA's + VDB + Generator



http://bit.ly/NubisVoxelCloudPack

Development Team
Nathan Vos
Hugh Malan
James McLaren

Stormbird Team
Elijah Houck
Nick van Kleef
Bart van Oosten

Atmospherics
Bryan Adams
Michelle Tolentino

Art
Roderick van der Steen
Misja Baas
Jan-Bart van Beek

FX
Marijn Giesbertz
Elco Vossers
Mark van Berkel

Tech
Jeroen Krebbers
Michiel van der Leeuw
Benjamin Santerre

The Schneider Team
Rosa de Vries
Aidan Schneider
Liam Schneider
Amelia Schneider

Previous Talks
Andrew Schneider. "Nubis, Evolved: Real-time Volumetric Clouds for Skies, Environments, and VFX". *ACM SIGGRAPH*. Vancouver, BC: ACM SIGGRAPH, 2022. Web. 2022.
Andrew Schneider. "The Real-Time Volumetric Superstorms of Horizon Forbidden West". *GDC 2022*. SF, USA, Web. 2022.
Andrew Schneider. "Nubis: Real-Time Volumetric Cloudscapes in a Nutshell". *Eurographics*. Delft, NL, Web. 2018.
Andrew Schneider. "Nubis: Authoring The Real-Time Volumetric Cloudscapes Of Horizon Zero Dawn". *ACM SIGGRAPH*. Los Angeles, CA: ACM SIGGRAPH, 2017. Web. 2017.
Andrew Schneider, GPU Pro 7: *Real Time Volumetric Cloudscapes*. p.p. (97-128) CRC Press, 2016.
Andrew Schneider. "The Real-Time Volumetric Cloudscapes Of Horizon Zero Dawn". *ACM SIGGRAPH*. Los Angeles, CA: ACM SIGGRAPH, 2015. Web. 26 Aug. 2015.

References
Augustus Beer, "Bestimmung der Absorption des rothen Lichts in farbigen Flüssigkeiten" (Determination of the absorption of red light in colored liquids), *Annalen der Physik und Chemie*, vol. 86, pp. 78-88, 1852.
L. G. Henyey and J. L. Greenstein, "Diffuse radiation in the Galaxy," *Astrophysical Journal*, vol. 93, pp. 78-83, 1941.
John Hart, "Sphere Tracing: A Geometric Method for the Antialiased Ray Tracing of Implicit Surfaces", *The Visual Computer*, June, 1995.
Jonathan "Lone Sock" Dummer, Cone Step Mapping: An Iterative Ray Heightfield Intersection Algorithm. 2006.

Email:

andrew.schneider@guerrilla-games.com

andrew@schneidervfx.com

Twitter / X / Whatever:

@vonschneidz

Mastodon:

@AndrewSchneider@mastodon.gamedev.place