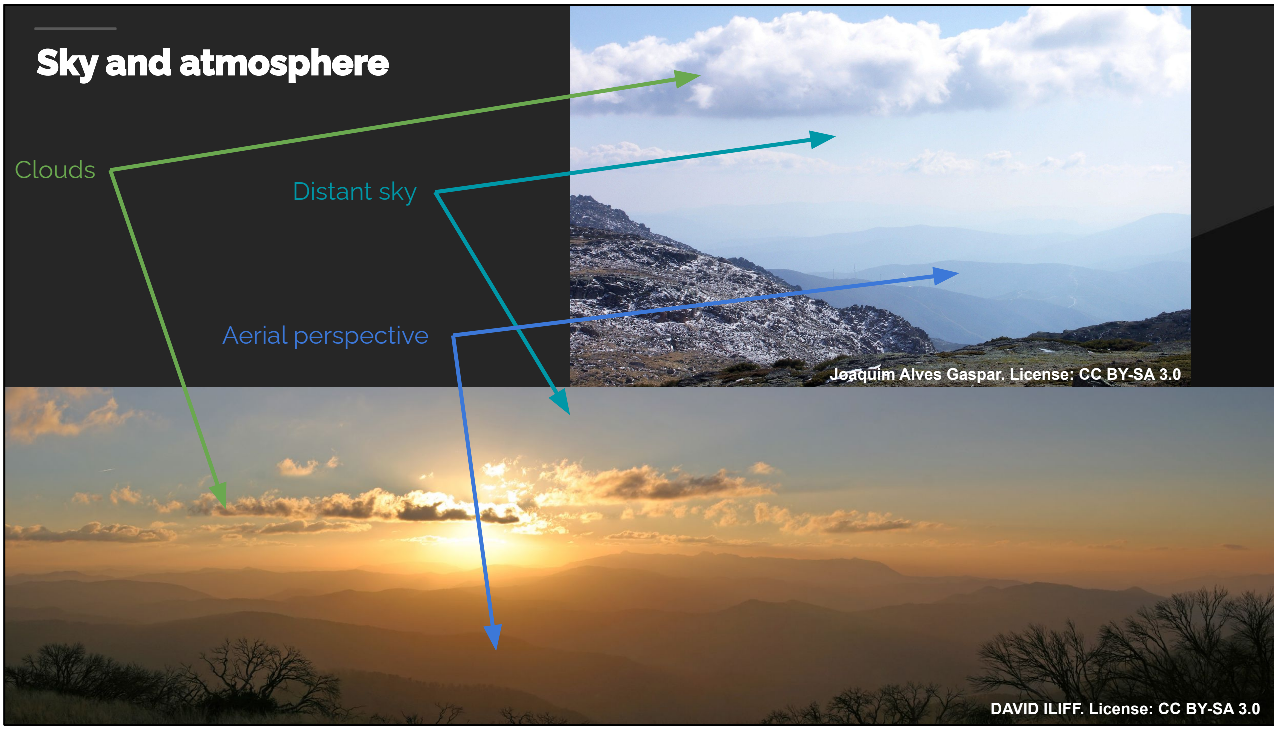
Physically Based and Scalable Atmospheres in Unreal Engine

Sebastien Hillaire
Epic Games

#UE4 | @UNREALENGINE

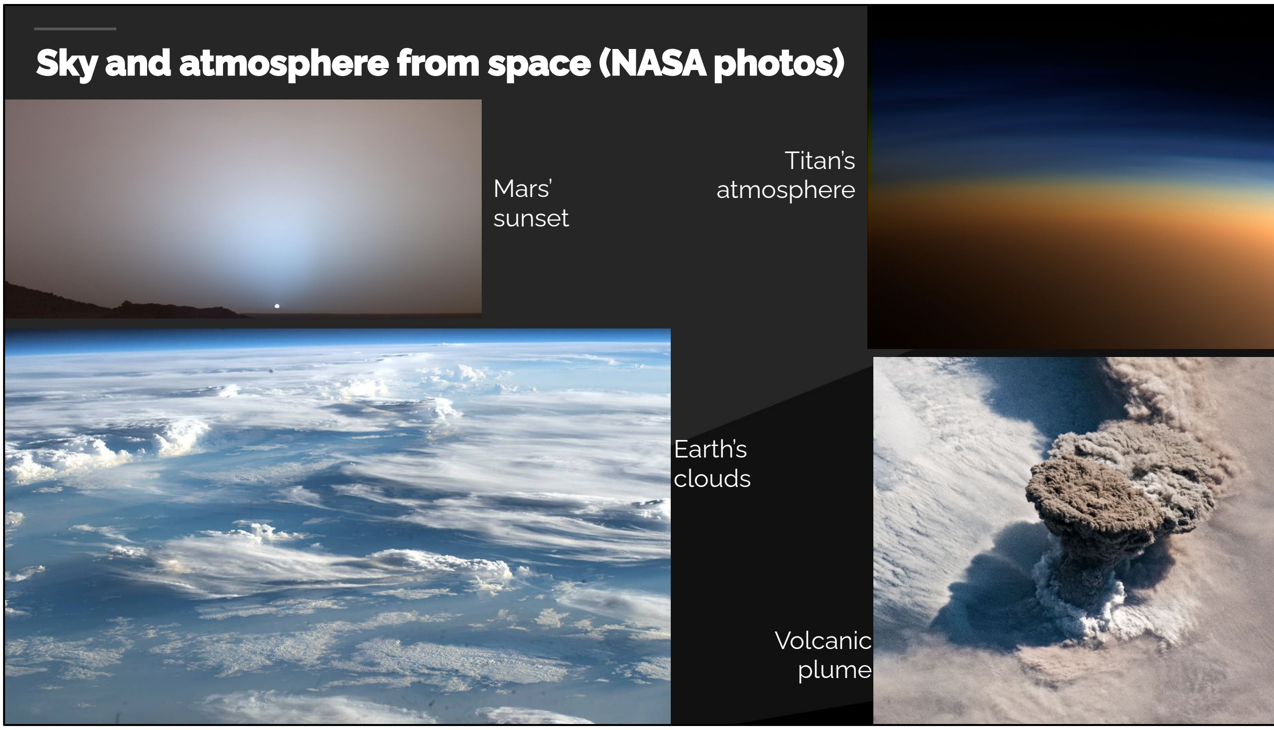
SIGGRAPH 2020 - Physically Based Shading in Theory and Practice

Hello everyone
I am Sebastien Hillaire and I am going to present to you the atmosphere rendering techniques we have developed in Unreal Engine.



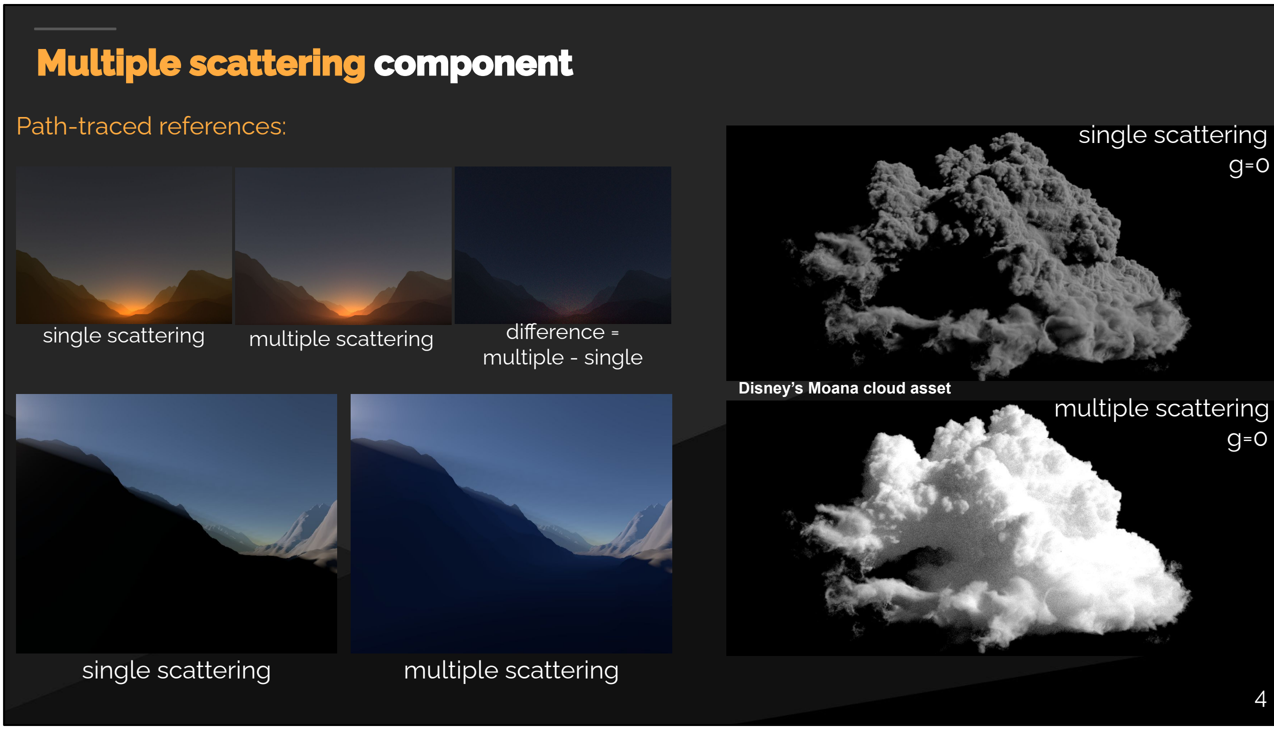
So what are we talking about?

[click]
The sky is the result of complex light scattering in particles within the atmosphere. On Earth, it has a rich blue color during the day and features more complex yellow, green and red tones at sunset.
[click]
When light scattering happens between objects and your eyes, this physical interaction is called aerial perspective. Those objects then look like they are in a fog.
[click]
Last but not least, clouds are also part of the atmosphere, also interacting with light and casting volumetric shadows within the atmosphere.



Many other different types of atmosphere exist in space.
For example, Mars and its blue sunset or Titan's complex atmosphere.

Moreover, other cloud formations can happen within a planet atmosphere such as cyclones, tornadoes or volcanic plumes.



So, the light will scatter several times within the atmosphere before reaching your eyes, and this is an important effect to simulate to achieve rich atmospheric visuals. We cannot take pictures of the real world with and without this phenomenon so here are some artificial examples achieved using path tracing.

You can see that on Earth, during sunset, multiple scattering is critical to achieving believable results and avoid a yellow-ish atmosphere look.

[click]
At the bottom, images show that multiple scattering is important for the light to fill up the space between the camera and landscape when inside a volumetric shadow. In this case, this is the result of light scattering around mountains.

[click]
Clouds participating media usually have an albedo close to 1. As such, light can bounce thousands of time before going out of the cloud at any position.
Thus, multiple scattering is a requirement to render clouds that do not look like dark smoke.

What do we want?

Artistic freedom and easy to use in the Unreal editor

High quality

Get close to the ground truth

Atmospheric effects for any time of day and weather

Support views from space

Scalable (Fortnite runs on mobile ES3.1, decoupled from screen resolution)

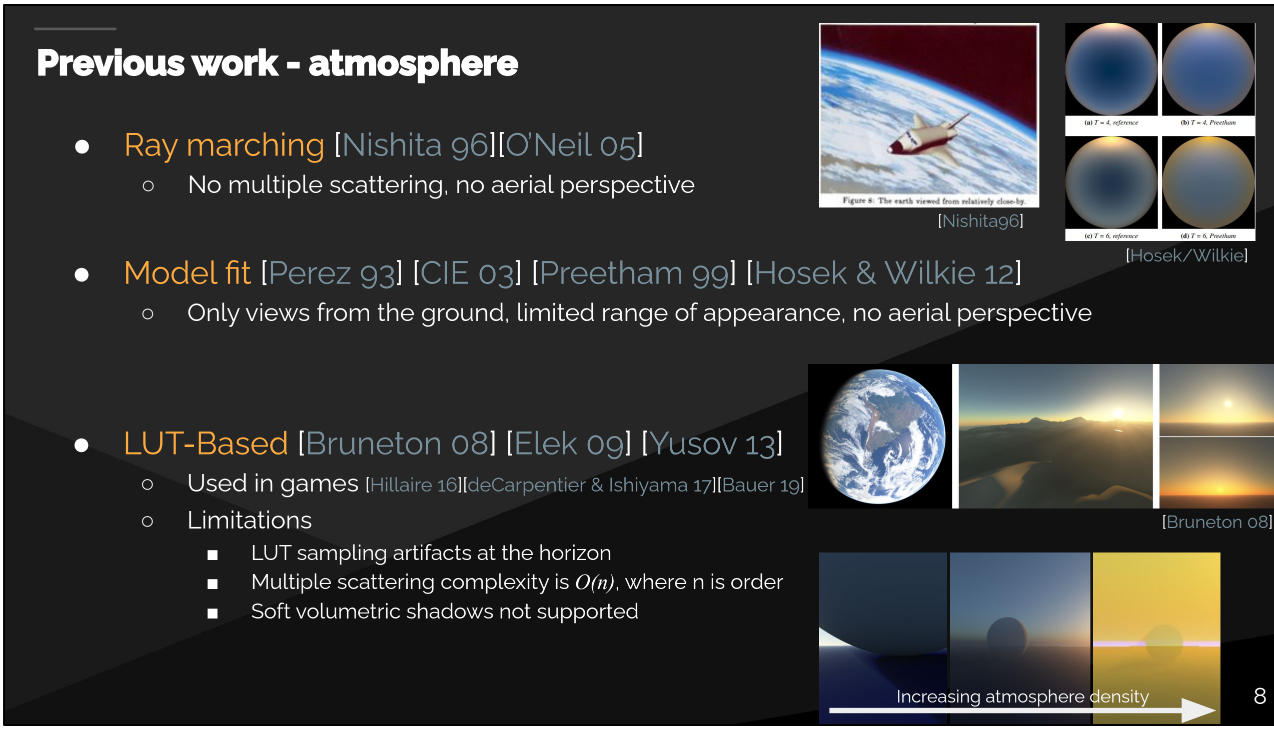
So

- [click] We wanted artistic freedom and be sure the tech is easy in Unreal editor
- [click] We wanted a high visual quality without objectionable artifacts
- [click] We wanted to be close to the ground truth in the way we represent and render the atmosphere
- [click] While supporting dynamic time of day and weather.
- [click] We wanted to support views from space, because many industrial applications or games require such a possibility for visualization or science fiction.
- [click] Last but not least, we wanted the technique to be scalable because our game Fortnite runs on mobile. And we also wanted the atmosphere rendering performance to be decoupled from the screen resolution (for example crazy 4K).

So here are some results we now have in Unreal Engine.

The first video shows the atmosphere being rendered in real time in the editor. The sky is updated each frame, so the atmosphere can be tweaked to match Mars without any delay.
The second video shows a view from space of planet Earth.
The third video shows that one can seamlessly fly from ground to space and even represent tiny planets with soft volumetric shadows from clouds.
The last video shows some custom cloud material representing some cyclone with exotic colors.

Sky Rendering



Several methods have been proposed to render skies.

For instance, it is possible to ray march the atmosphere or fit a mathematical model on sky color itself. However, such models are not taking into account multiple scattering, not providing solution to render the aerial perspective, and do not support views from space.

[click]
A more successful approach to sky rendering are the *lookup table* based models. They have been used successfully in many games with a few simplifications.

However, they do have some limitations:
- Some visual artifacts can sometimes be seen at the horizon, and it becomes unavoidable for thick atmosphere.
- The high dimensional scattering look up table is expensive to update and requires N iterations to compute the result of N orders of scattering
- Also, soft volumetric shadows from clouds is not directly supported

Definition of a planet atmosphere (Earth like)

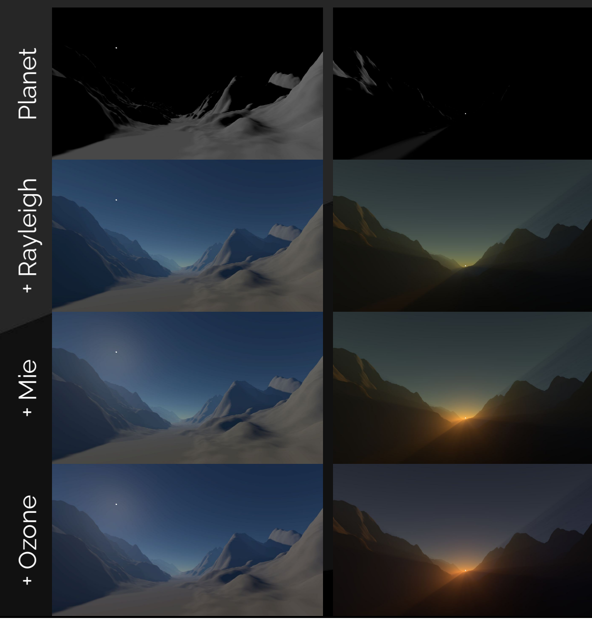
Assumes a perfect terrestrial planet's atmosphere [Bruneton 17]:

- Rayleigh scattering,
- Mie scattering & absorption,
- Ozone absorption,

More details in our EGSR2020 paper:

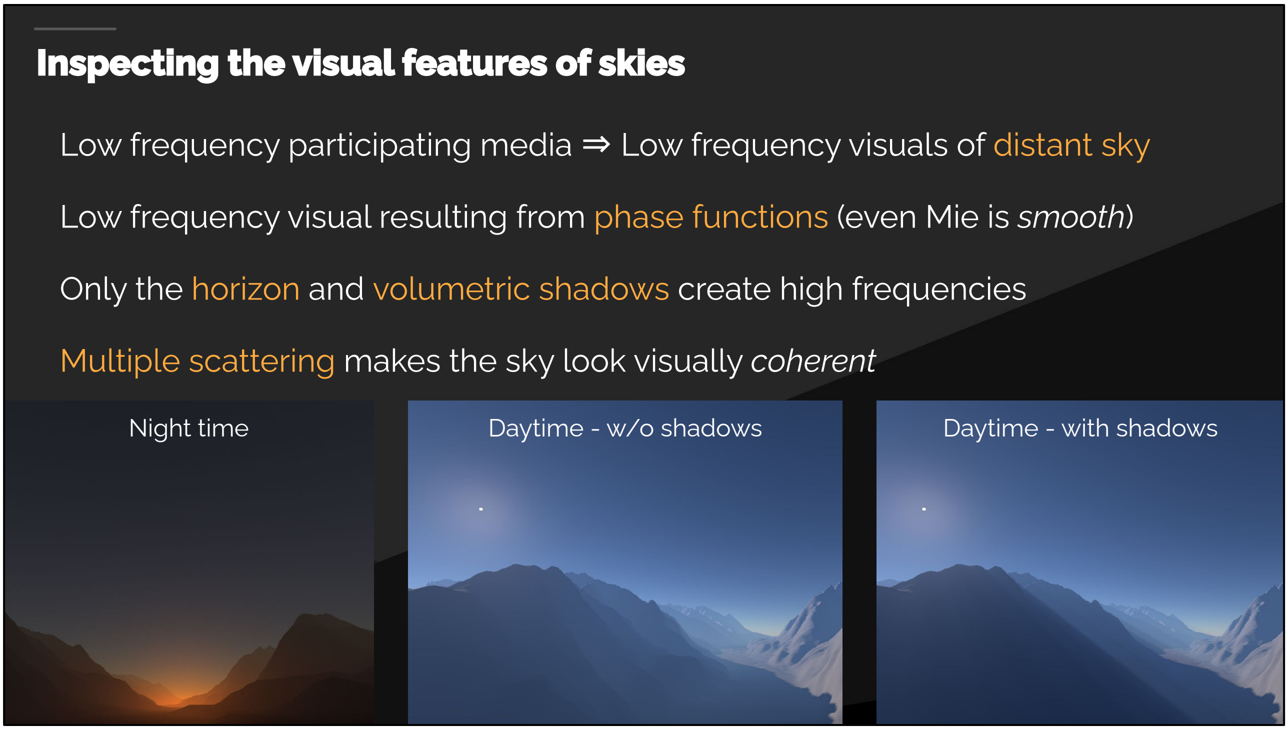
Type	Scattering $(\times 10^{-6}\mathrm{m}^{-1})$	Absorption $(\times 10^{-6}\mathrm{m}^{-1})$
Rayleigh	$\sigma_s^r = 5.802, 13.558, 33.1$	$\sigma_a^r = 0$
Mie	$\sigma_s^m = 3.996$	$\sigma_a^m = 4.40$
Ozone	$\sigma_s^o = 0$	$\sigma_a^o = 0.650, 1.881, 0.085$

- Height distributions,
- Ground albedo, etc.



Before we dive in the rendering technique, I wanted to mention that we represent the atmosphere material the same way as it has been done in previous work.

Please refer to our EGSR paper for more details.
https://diglib.eg.org/bitstream/handle/10.1111/cgf14050/v39i4pp013-022.pdf

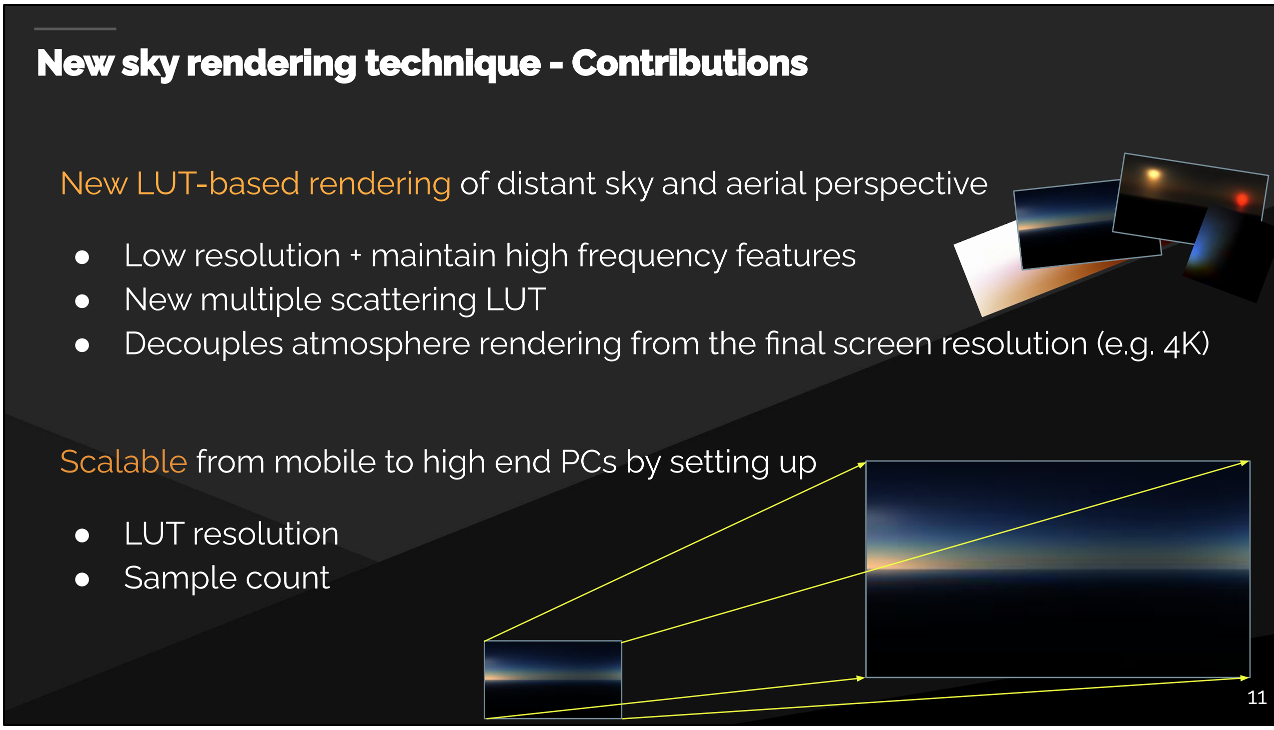


So we started by looking at what is really required to render the sky. A bit like when you want to best represent visual features of a BRDF: you have to identify the representative characteristics having a high impact in order to reproduce a target visual.

[click]
One can easily notice that the distant sky is of very low frequency.
[click]
Even the Mie scattering lobe is a soft blob around the sun.
The aerial perspective is also very smooth on screen and in depth.

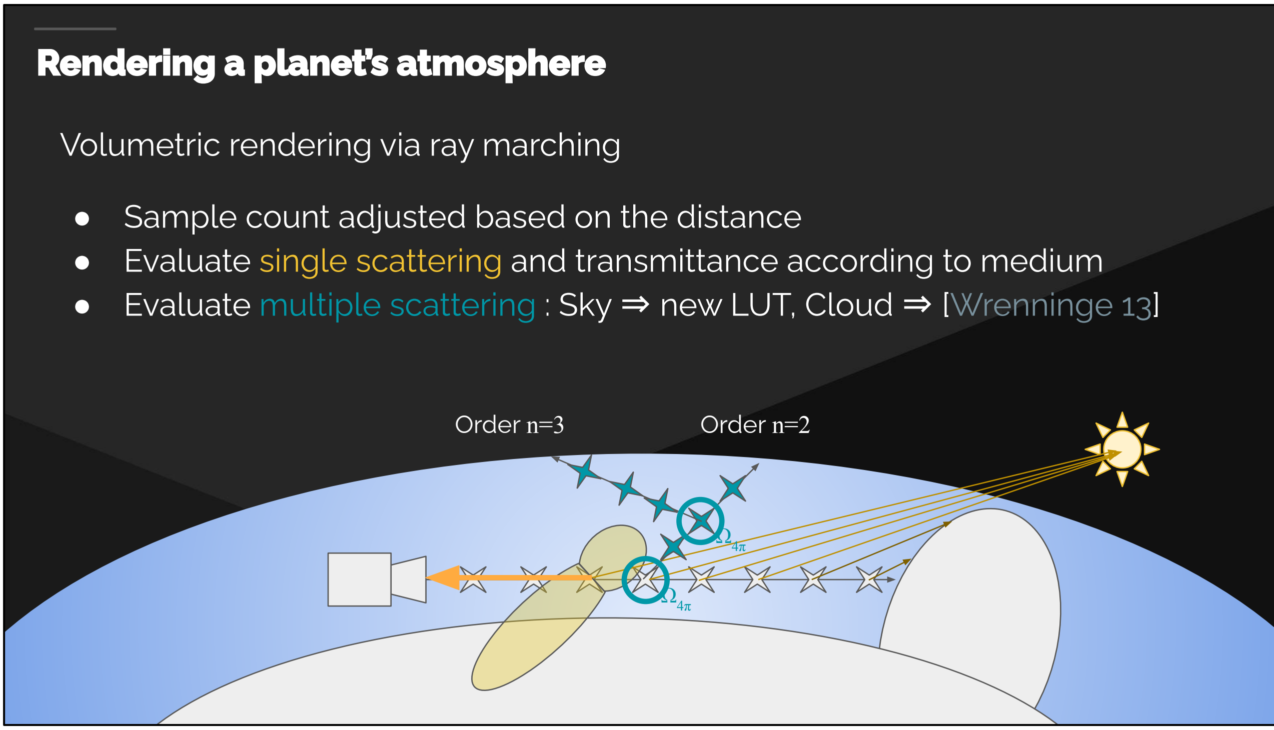
[click]
The only high frequencies we can see are the rapid change of atmosphere color near the horizon and the variations due to volumetric shadows.

[click]
On top of that we also recognize the importance of multiple scattering in order to faithfully represent atmospheric scattering and achieve believable visual results.



From this simple visual analysis, we propose a way to render all these important visuals details using a new set of low resolution LUTs maintaining high frequency features.

[click]
These LUTs allows us to decouple the atmosphere rendering performance from the screen resolution.
It makes the technique scalable from mobile to high-end PCs by simply tweaking the LUT resolution and ray marching sample count.

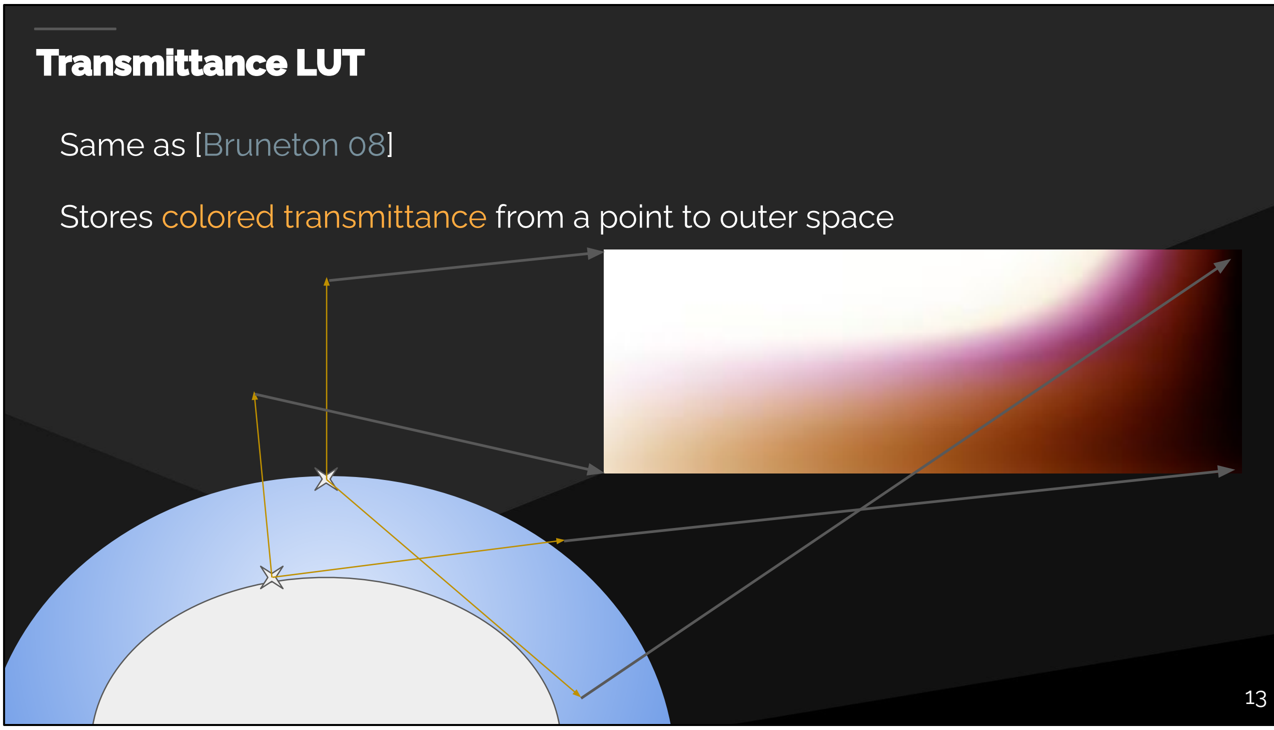


Now, how can we render an atmosphere?

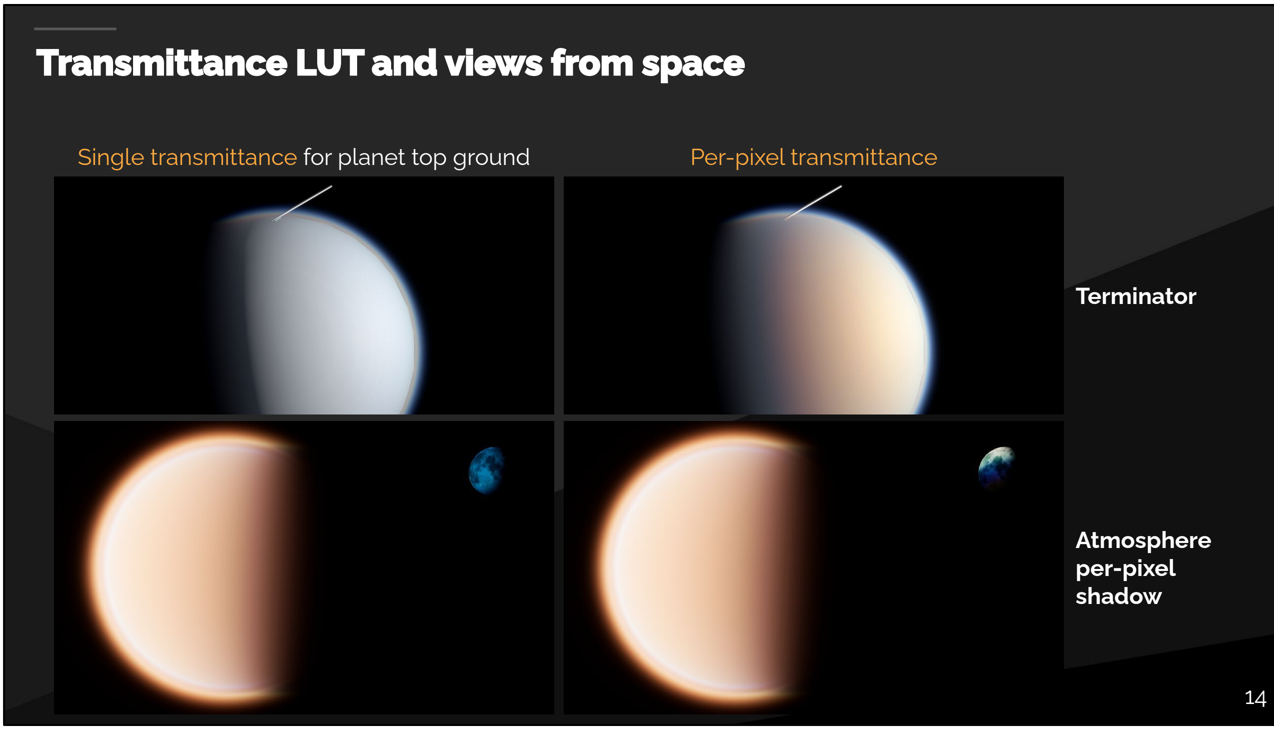
[click]
We will use typical volumetric ray marching with sample count adjusted based on distance.

[click]
For each sample, we evaluate the light transmitted through the atmosphere,
[click]
phase function and atmosphere material and
[click]
From that we can deduce the amount of light scattered toward the camera and the transmittance to the background.
[click]
and so on

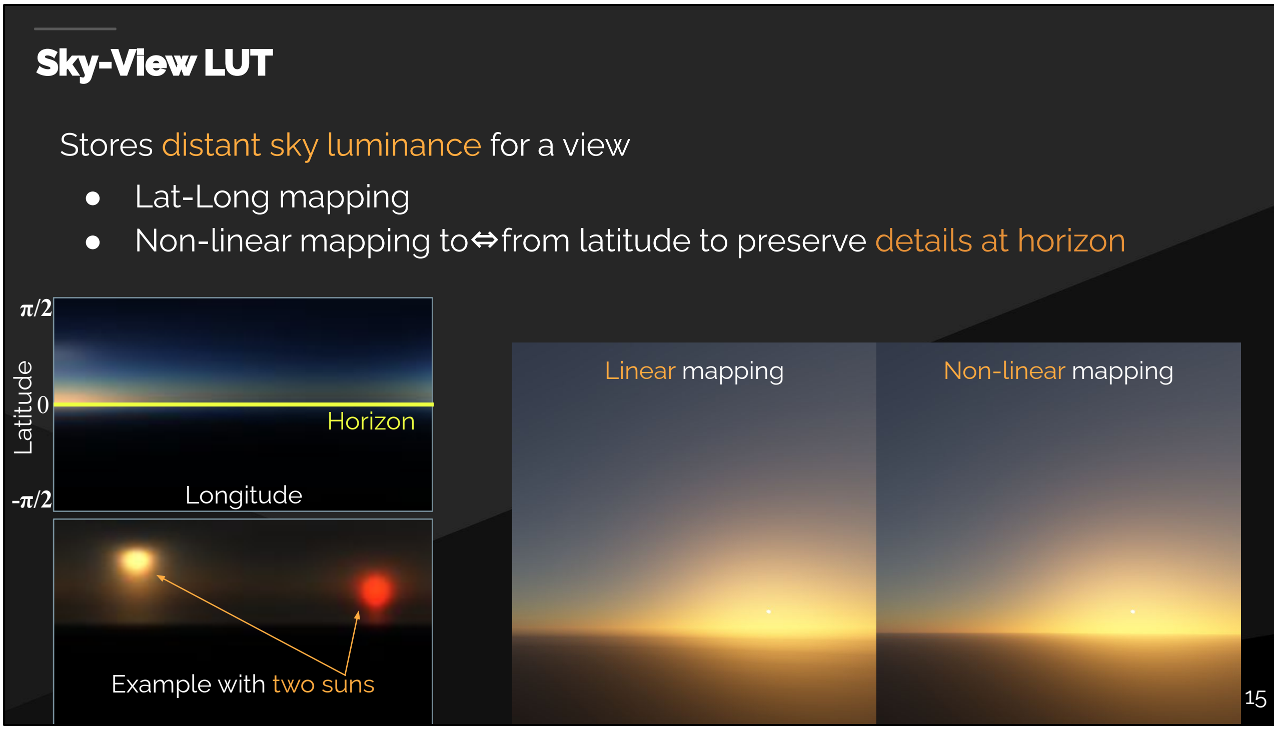
[click]
Multiple scattering is typically too expensive to evaluate this way so approximations will be used.



As mentioned previously, we often need to evaluate the light transmitted through the atmosphere toward a point.
Instead of secondary raymarching, we use the same look up table proposed by Bruneton storing colored transmittance.



By default, our rendering technique is optimized for views form the ground. In this case, we compute a single transmittance value for the top of the planet to the ground. It is applied on all entities.
[click]
We also give an option to apply that function per pixel in order to achieve more realistic space view of a planet and its terminator region.
[click]
or have the atmosphere itself cast shadow onto moons for instance

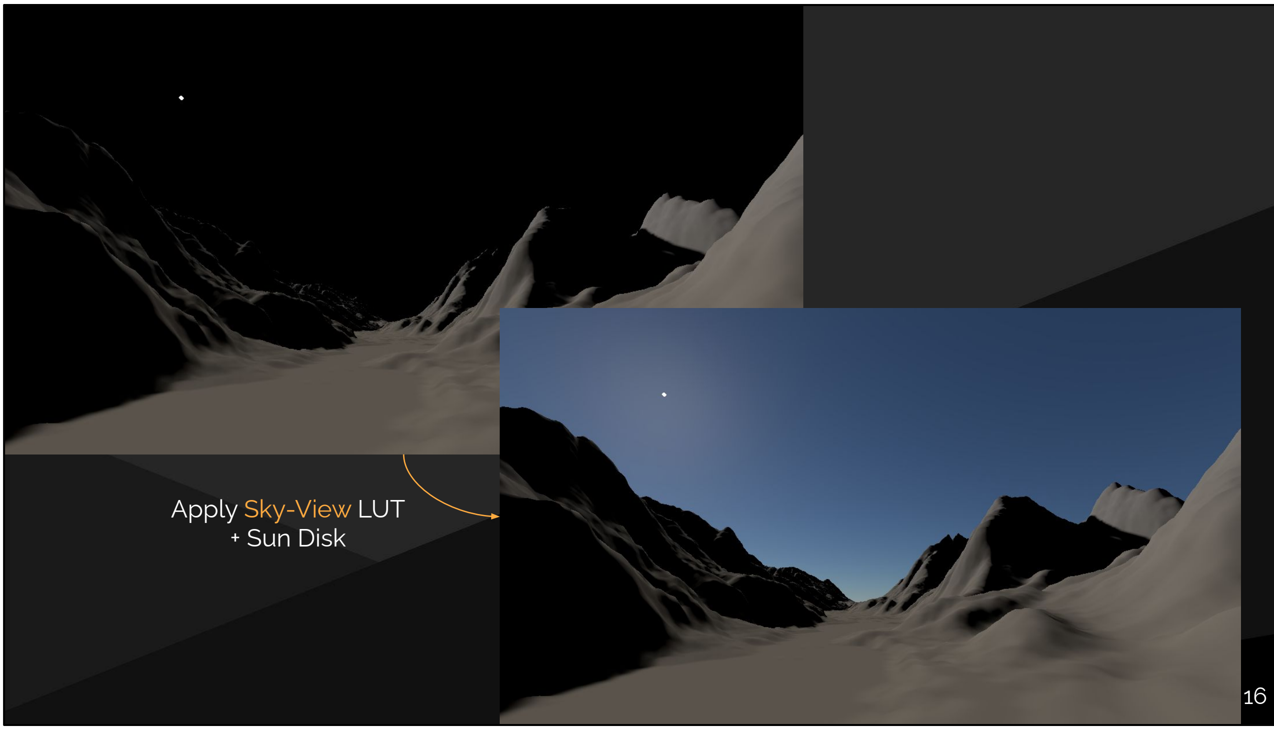


The Sky-View is one of the new lookup table we propose in order to render the distant sky.

[click]
It stores the ray marching result using a lat/long mapping.

[click]
[click toggle]
Please note that the latitude mapping is non linear in order to maintain the high frequency colors at the horizon while reducing linear interpolation artifacts.

[click]
This lookup table can also store the contribution of any number of suns at once.

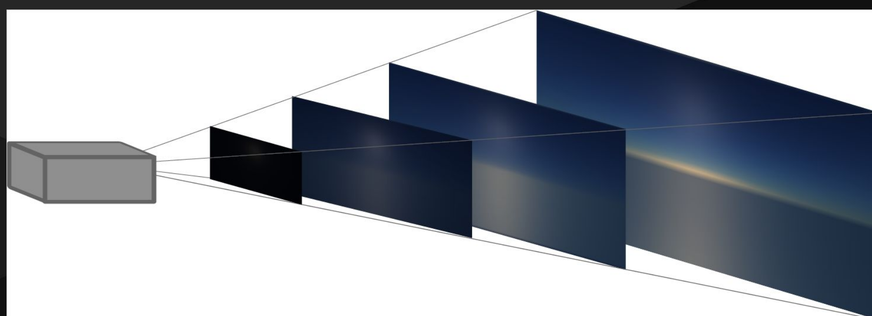


So we can now render the distant sky as seen here.
[click]
And the sun disk is composited at this stage.

Aerial Perspective LUT

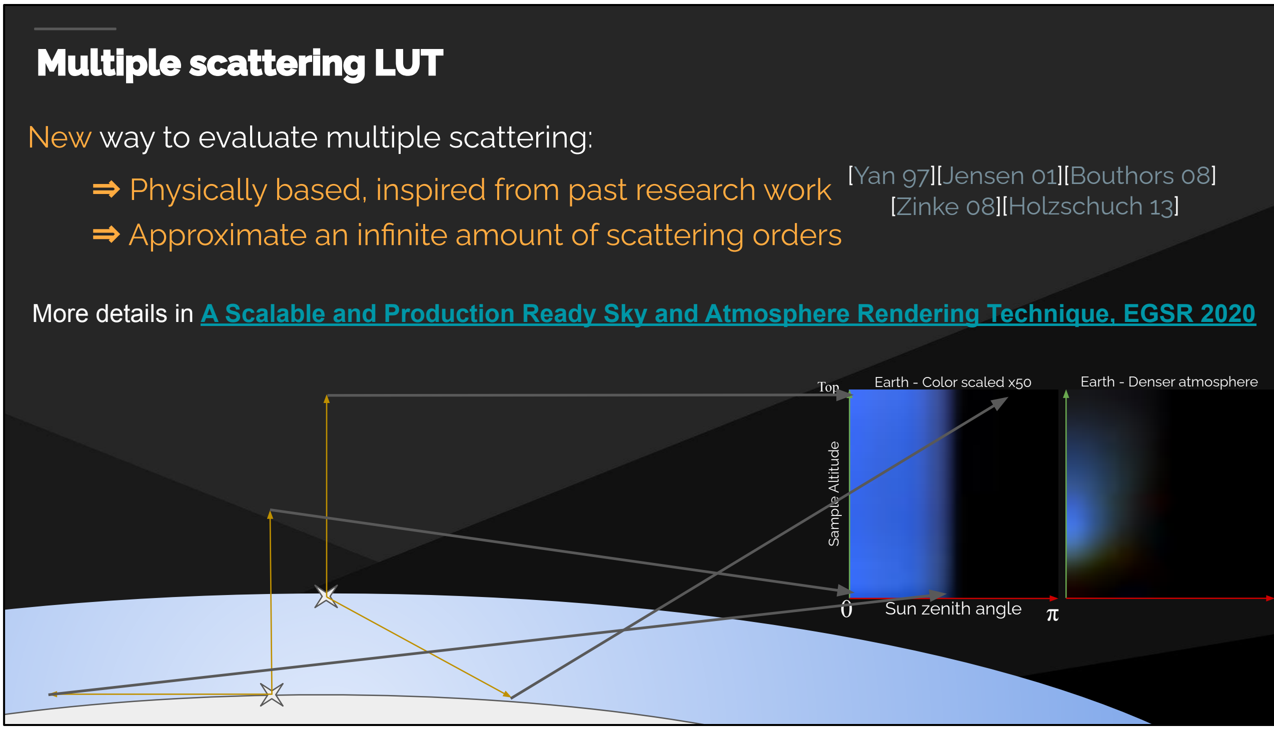
Froxel volume texture mapped onto camera frustum [Wronski 14, Hillaire 16].

- RGB stores luminance reaching the camera from the froxel center
- Alpha stores greyscale transmittance from camera to froxel center
- Applied on opaque and translucent surfaces



The aerial perspective luminance and transmittance is evaluated and stored in a volume texture mapped onto the camera frustum, as proposed in some previous work.
This can then be applied on opaque and translucent surfaces.

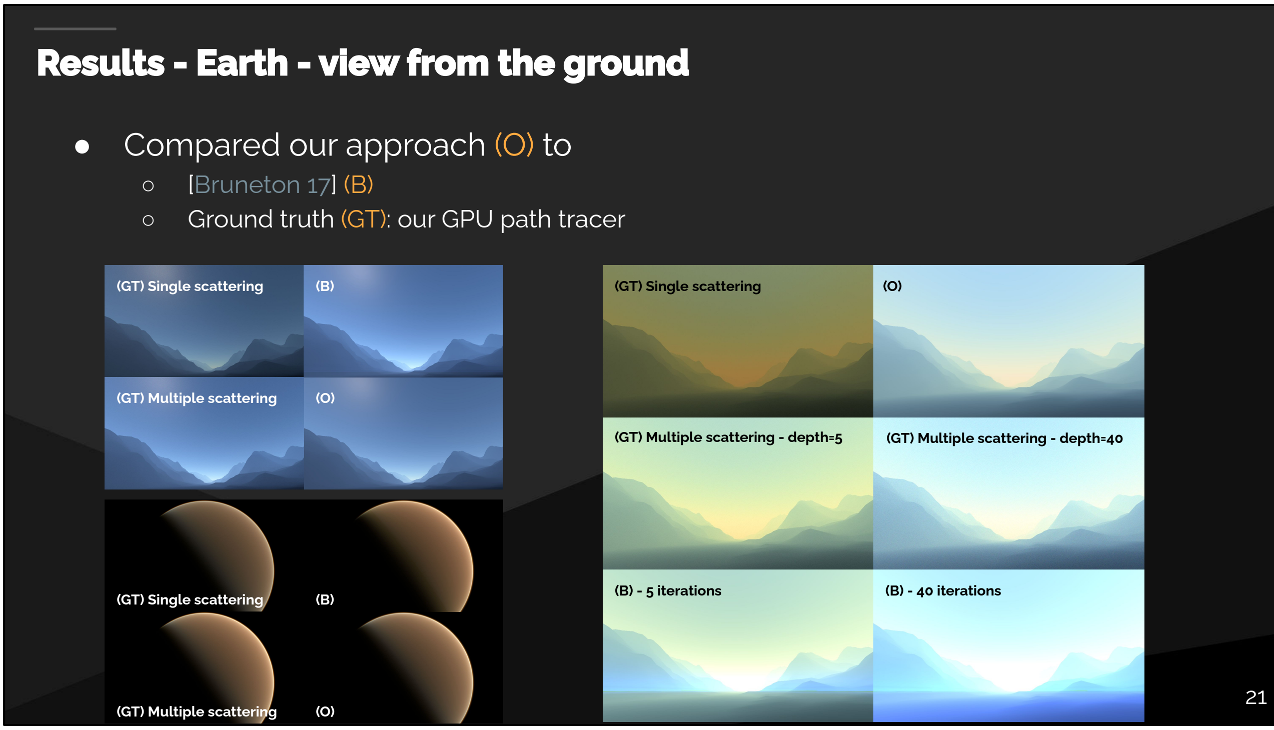
To recap: this is with the distant sky only
[click]
and this is now with the aerial perspective applied.



[click]
The evaluation of the luminance resulting from multiple scattering is simplified by gathering ideas from previous work from light transport papers for participating media and hair rendering. But this time adapted to atmospheric rendering.
Our physically based approach can approximate the evaluation of an infinite number of scattering orders.
For the sake of time, please refer to our EGSR paper for more details, https://diglib.eg.org/bitstream/handle/10.1111/cgf14050/v39i4pp013-022.pdf

[click]
Using this technique, we end up with a 2 dimensional lookup table storing the isotropic multiple scattering contribution as a function of the sample altitude and sun direction.

Ok, let’s have a look at some results now



We have compared our approach to the state of the art technique from Bruneton, and a volume path tracer we have developed specifically to be our ground truth.

In short
[click]
The comparisons show that our model as close to the ground truth as the state of the art model
[click]
And it is the only model that can faithfully reproduce the effect of infinite multiple scattering in a denser extra-terrestrial atmosphere.

Please refer to our EGSR paper for more a more in depth analysis,
https://diglib.eg.org/bitstream/handle/10.1111/cgf14050/v39i4pp013-022.pdf

Performance - PC/Mobile

Can be lowered for increased performance if visual difference is acceptable

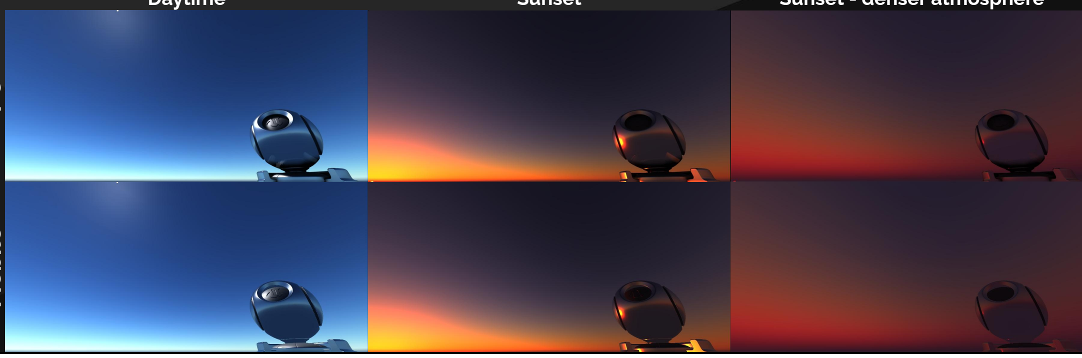
PC (NVIDIA 1080)			
LUT	Resolution	Step count	Render time
Transmittance	256×64	40	0.01ms
Sky-View	200×100	30	0.05ms
Aerial perspective	32^3	30	0.04ms
Multi-scattering	32^2	20	0.07ms

Render sky & atmosphere @720p = 0.14ms

⇒ Total cost = 0.31ms

Mobile (iPhone 6s)			
LUT	Resolution	Step count	Render time
Transmittance	256×64	40	0.53ms
Sky-View	96×50	8	0.27ms
Aerial perspective	$32^2 \times 16$	8	0.11ms
Multi-scattering	32^2	20	0.12ms

Lut cost = 1.03ms



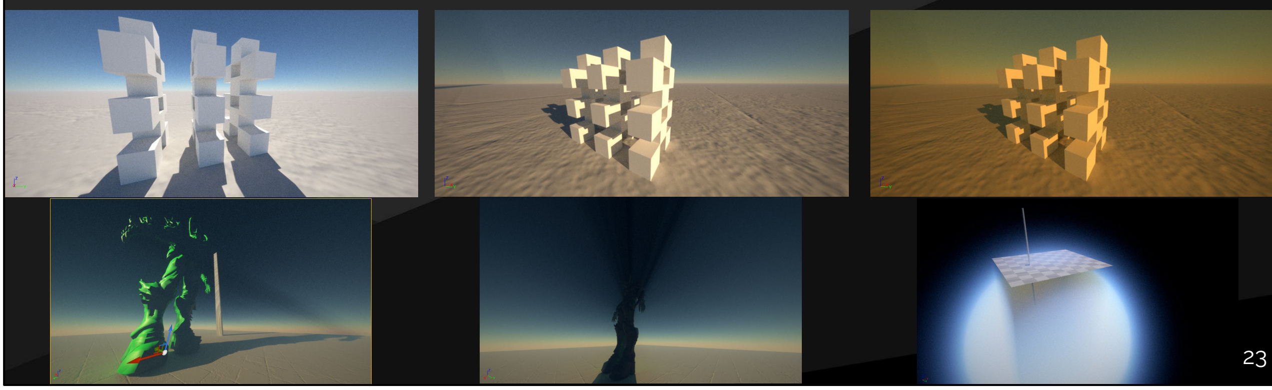
Here you can see the performance when building those LUTs on pc or mobile.
[click]
Please note that the transmittance LUT could have an even lower resolution on mobile and use less samples if you are willing to accept minor visual differences.
[click]
And you can see that we get a nice match between high-end PC and the lower-end mobile platforms we support.

Another level of scalability: Reference path tracing

Prototype

In the Unreal DXR path tracer

- Path integral of the Radiance Transfer Equation [Chandrasekhar 50]
- Delta tracking for heterogeneous participating media [Fong 17]
- Spectral tracking [Kutz 17], Ratio tracking [Novak 14]



At the other hand of the complexity spectrum, we have a work-in-progress prototype of a reference for atmosphere rendering in Unreal's path tracer.
Global illumination here is coming from light scattering on particles within the atmosphere layer, not from a distant cube map.
From this, we get correct global illumination within the atmosphere as well as proper volumetric shadows and multiple scattering.

Ok. Now let’s chat a bit about the rendering of clouds, because they are key to achieving believable skies.

Previous work - real time volumetric clouds

Ray marching procedural volumes

- [Schneider 15, 17]
  - Convincing procedural shapes based on Perlin and Worley noise
- [Bauer 19]
  - Unified cloud and fog ray marching, atmosphere composited separately

[Schneider 15, 17]

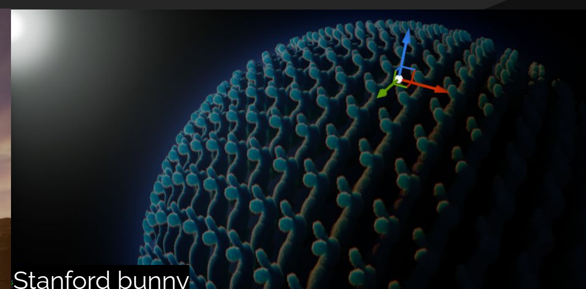
[Bauer 19]

Recently, beautiful real-time cloud rendering implementations have successfully shipped in games.
Schneider proposed a way to assemble noise primitives to render visually convincing volumetric clouds.
[clic]
Then Bauer presented a method improving this approach using a unified model, rendering nearby volumetrics, fog and cloud altogether.

[Schneider 15, 17]
Update 1 pixel per 4x4 tile ⇒ 16 frame latency but looks full-res
[Bauer 19]
Trace at quarter resolution with TAA upsampling ⇒ Instant result but looks blurrier

Cloud rendering

- Remove the static cloud shape/noise restriction from [Schneider 17][Bauer 19]
  - Clouds are defined using a volume material graph
  - Freedom to achieve anything: any shapes, tornadoes, cyclones, etc
  - Custom Unreal blueprint workflow

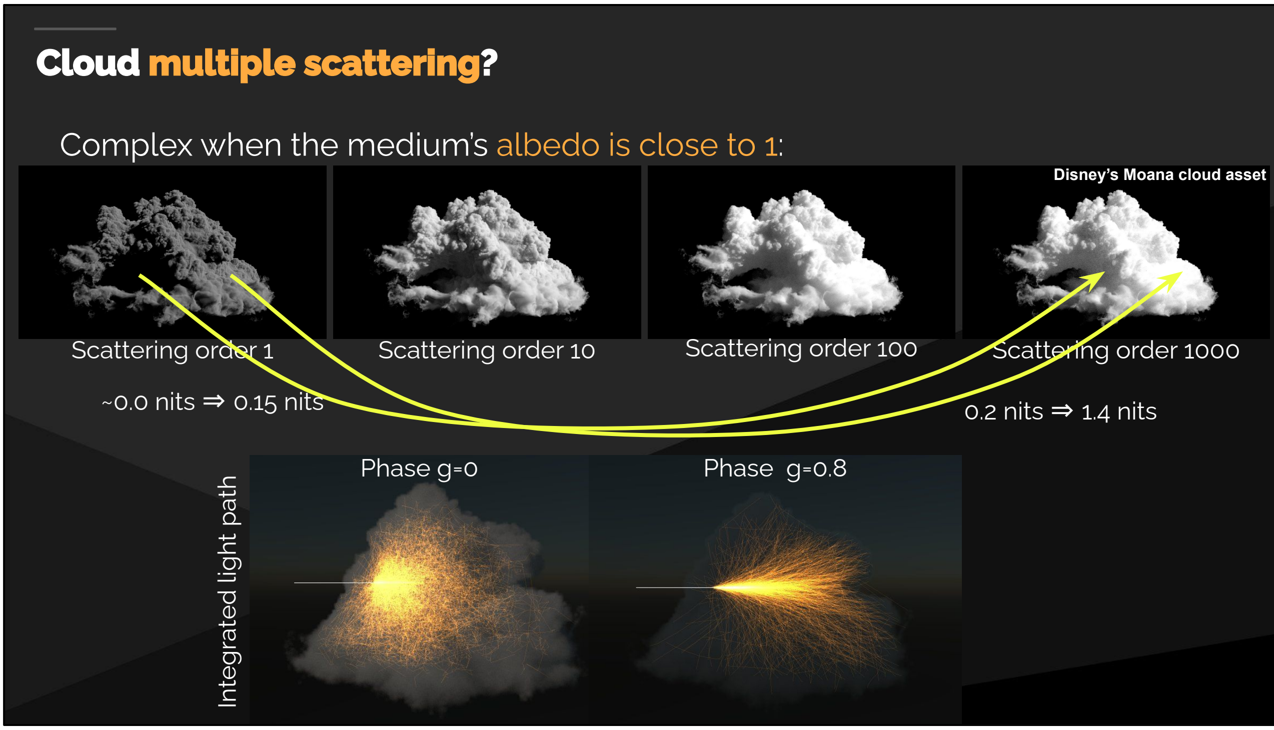


Stanford bunny

- Physically based multiple scattering
- Better occlusion representation: *Beer Shadow Map*
- Other visual features

However, these methods are relying on static ways to combine noise to represent clouds. They still gives artists a lot of flexibility through the exposed parameterization, but we wanted to lift this limitation.
In Unreal, the cloud layer is a volumetric material graph that is authored by tech artists and the workflow can be customized using Unreal's blueprint.
[click]
With that, one can render any cloud shape, for instance tornadoes, or bunny shaped clouds if you feel like it.

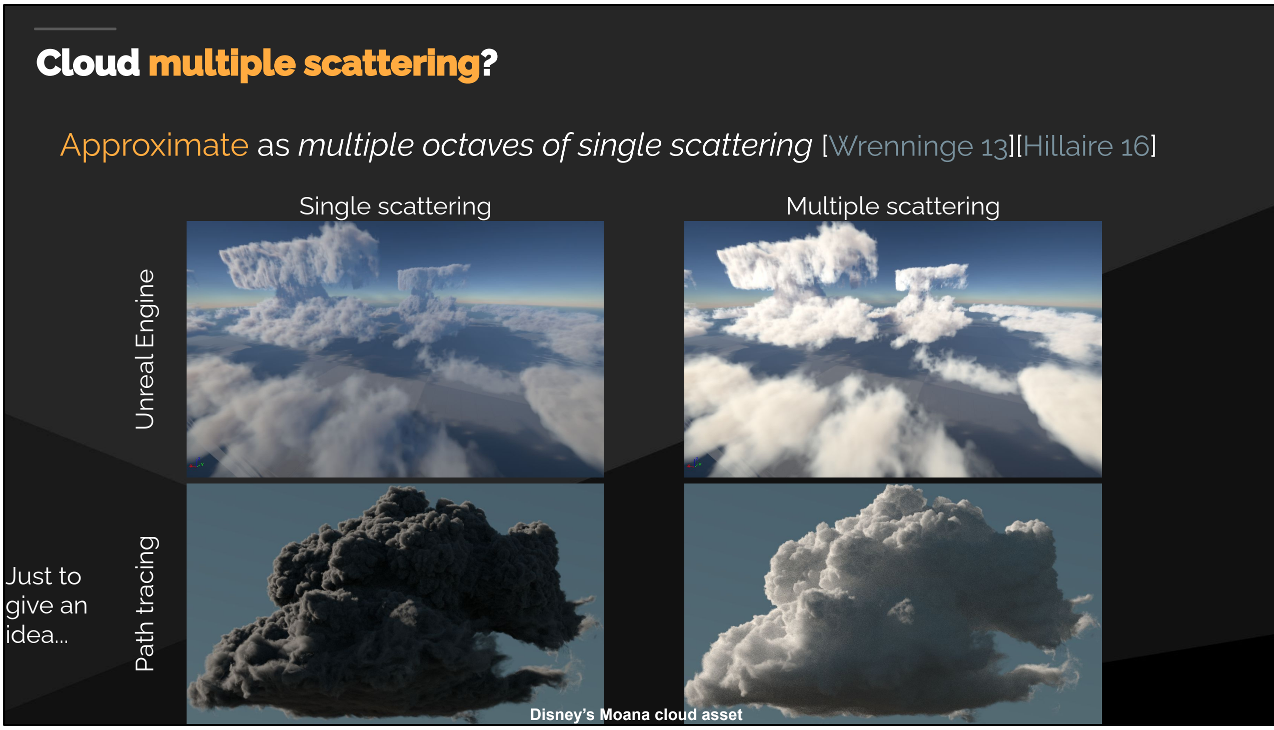
[click]
Later, I'll also talk about how we handle physically based multiple scattering, a new way to represent occlusion using a *Beer shadow map* and other visual features that can be important to render.



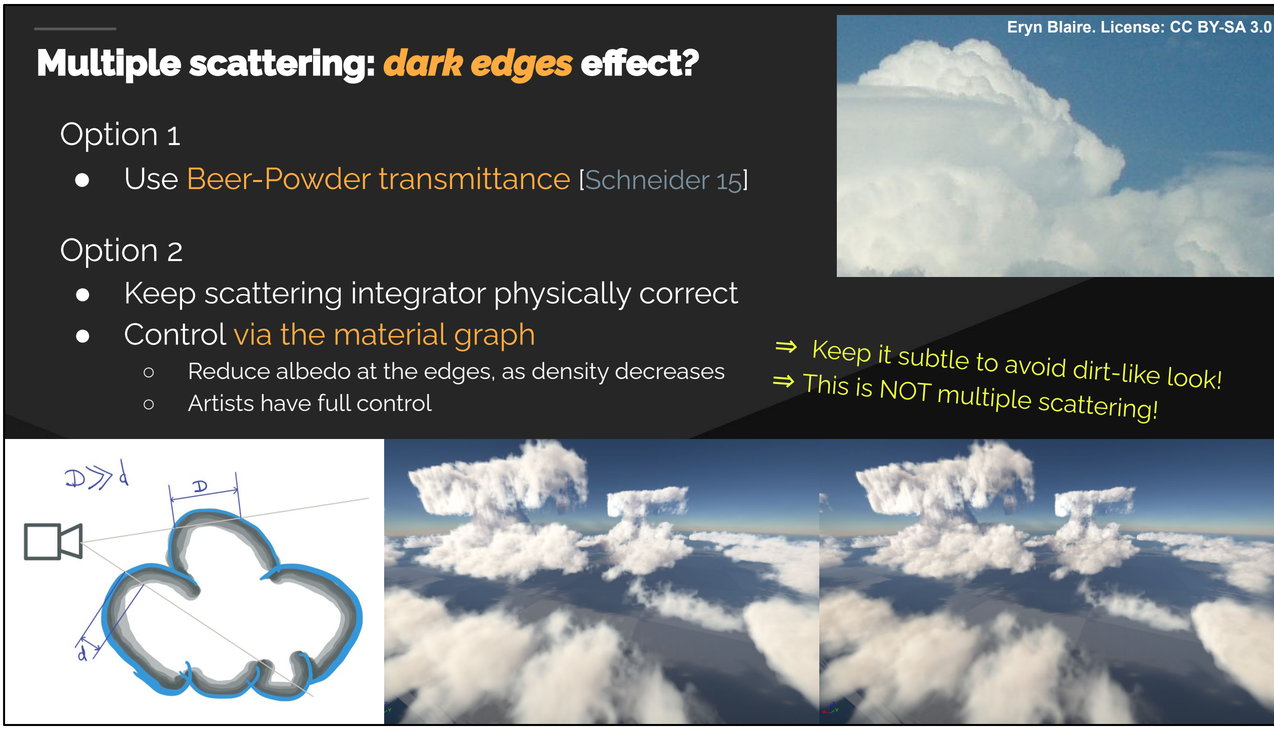
So clouds are rendered using ray marching, but how can we evaluate the multiple scattering?

As mentioned before, that phenomenon really defines the distinctive appearance of a cloud.
[click]
Without multiple scattering, a huge part of the energy is lost because the participating medium albedo is usually very close to 1, meaning that light is almost never absorbed.

[click]
As you can see in these debug representations, paths that need to be integrated to gather scattered luminance according to different phase functions are complex. There has been some work done to solve this in real time but nothing that seems shippable while respecting the complex visuals of clouds.



We settled on using the *multiple octaves of single scattering* approach proposed by Wrenninge.
[click]
This is single scattering only. And I show a path traced result just to give to you a visual idea idea of the ground truth
[click]
And this is the result when using two octaves of single scattering: you can see that the light penetrates deeper into the medium, achieving a brighter and more cloud-like appearance.



However, this is not a true multiple scattering simulation and as such lacks another defining visual feature: dark edges being the result of low probability of light scattering towards the eye in those region.
Multiple scattering in a high albedo situation is challenging even for offline rendering, so we have to cheat a bit here:

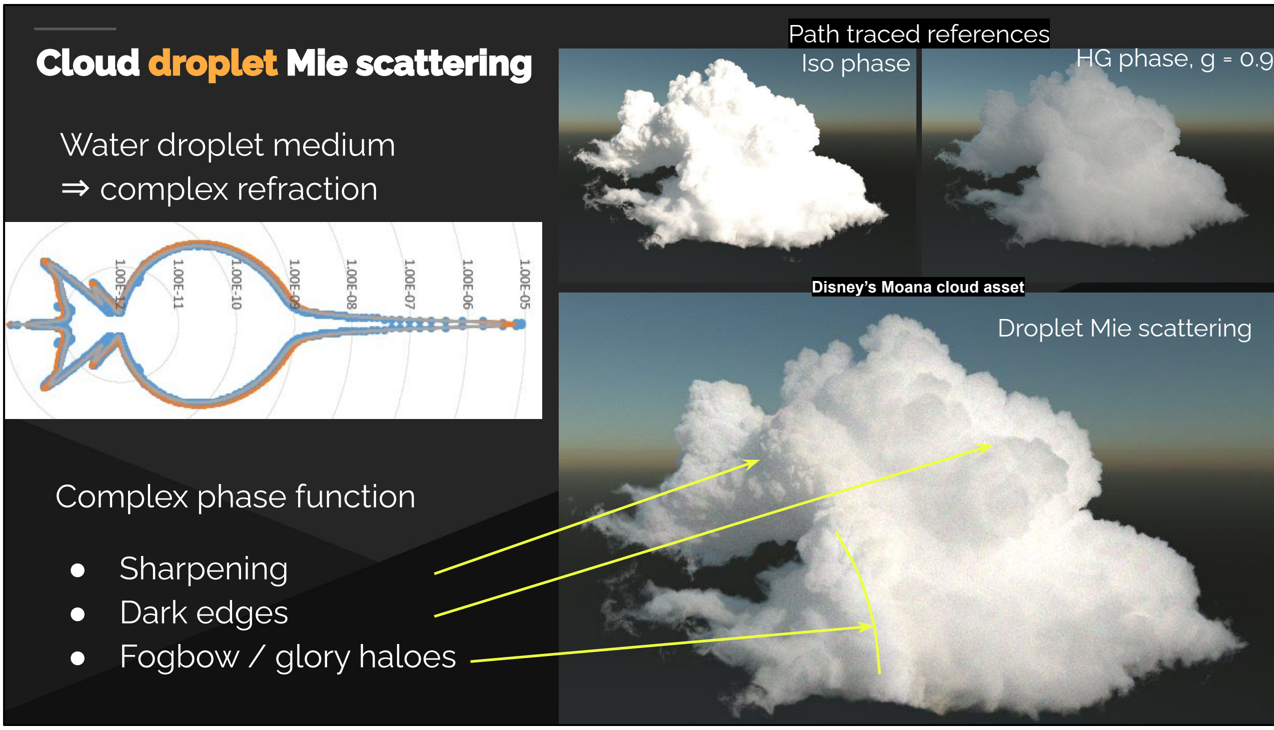
[click][click]
One option is to use the custom transmittance function presented by Schneider.

[click][click]
But we prefer to stay physically based and
[click][click][click]
let artists control such effect from the material graph by simply lowering the albedo near the edges of clouds.
[click]
As you can see on this sketch, rays will travel more in low albedo regions at the edges of the volume and that will automatically reveal details at edges.

[click]
[click back and forth to show diff]
You can see this subtle effect here on the right image

[click]
However, this is like ambient occlusion: it is wrong but it works and helps visually. But

be careful to not overdo it in order to avoid a *dirty cotton* like look.

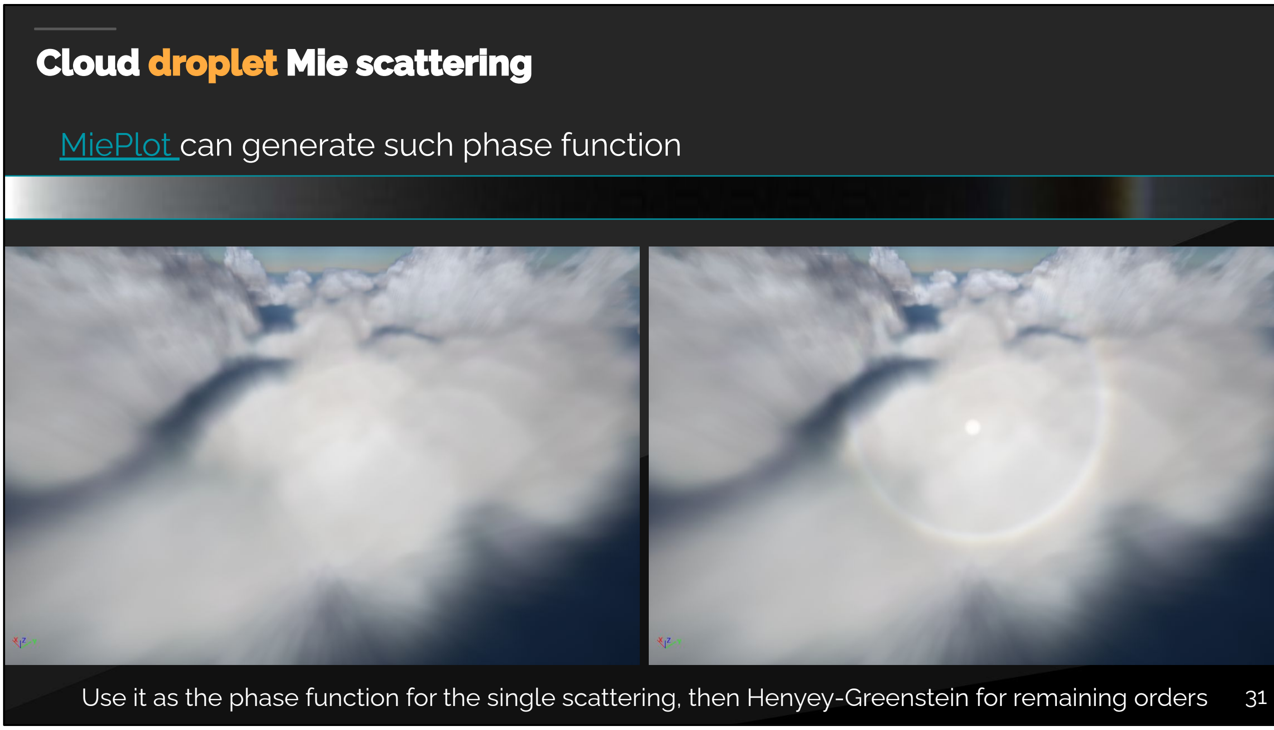


Just to put another nail in the coffin: clouds are not composed of dust or air molecules. They are made of many tiny water droplets resulting from condensation. This results in refraction events in turn
[click] resulting in a complex phase function that is wavelength dependent.
[click] It produces many visual features such as sharpening, glory halo and the dark edges we discussed before.

So if you use
[click] an isotropic phase function, it can look too bright or
[click] too dark when using a single strongly forward lobe phase function.
And you will never be able to realistically render clouds like this
[click]. I have been able to achieve this render using my personal volumetric path tracer running on GPU,
sampling the realistic Mie scattering water droplet phase function shown on the left.

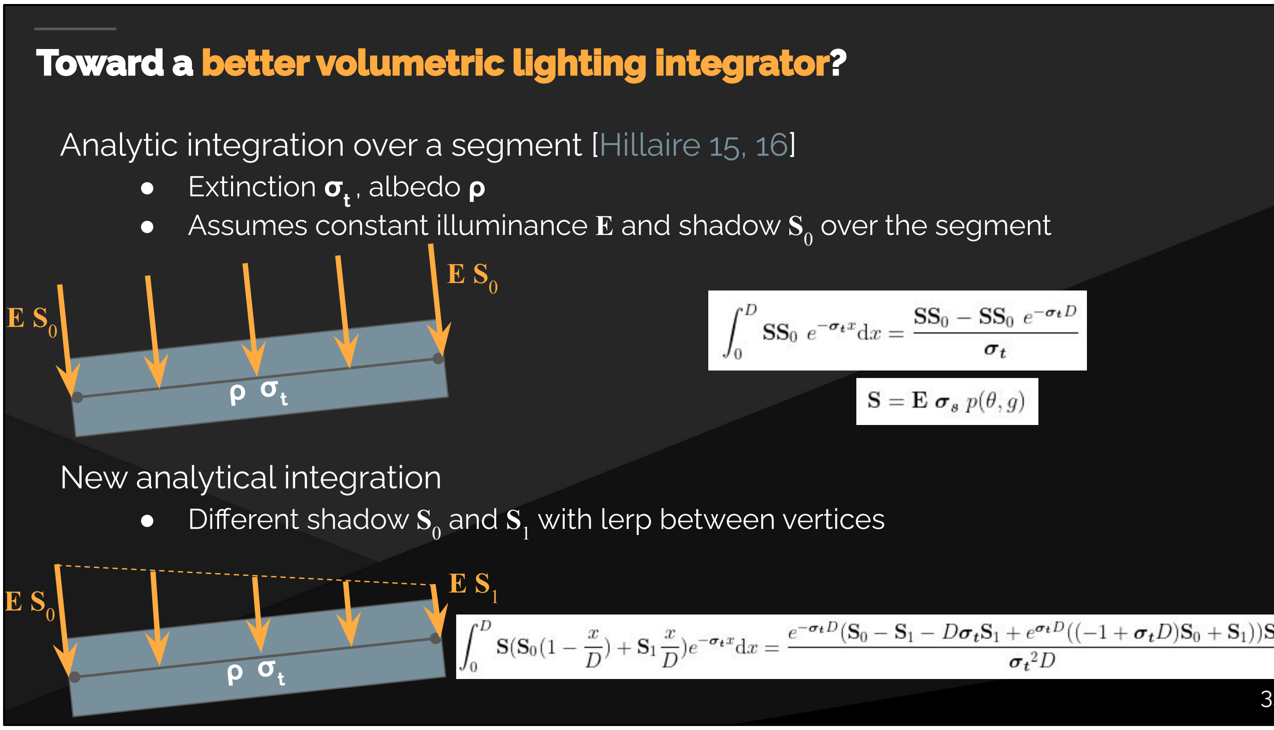
[click]
And it automatically has all of these important cloud visual features.

There is still research to conduct to get this result in real time, but today I do not have any nice solution to give you apart from the albedo trick I mentioned before.



However, it is possible to do the following:

- [click] Generate a complex phase function using the MiePlot software.
- [click] Then, for the sake of real time performance, that phase function can be used on the single scattering path only. That at least allows us to recover the fogbow and glory halo visual features.
- Multiple scattering is then simply evaluated using the previously mentioned *multiple octave of single scattering* approach.



So, when ray marching such volumetric data, you need to consider how you integrate the lighting.
[click]
A few years ago I have presented an analytical integration better than Simpson or trapezoidal integration because it respects Beer's law over the considered segment. However it was only supporting a single shadow value over the segment, here S0.
[click]
I considered improving this by taking into account a different shadow value per vertex that are linearly interpolated over the segment.

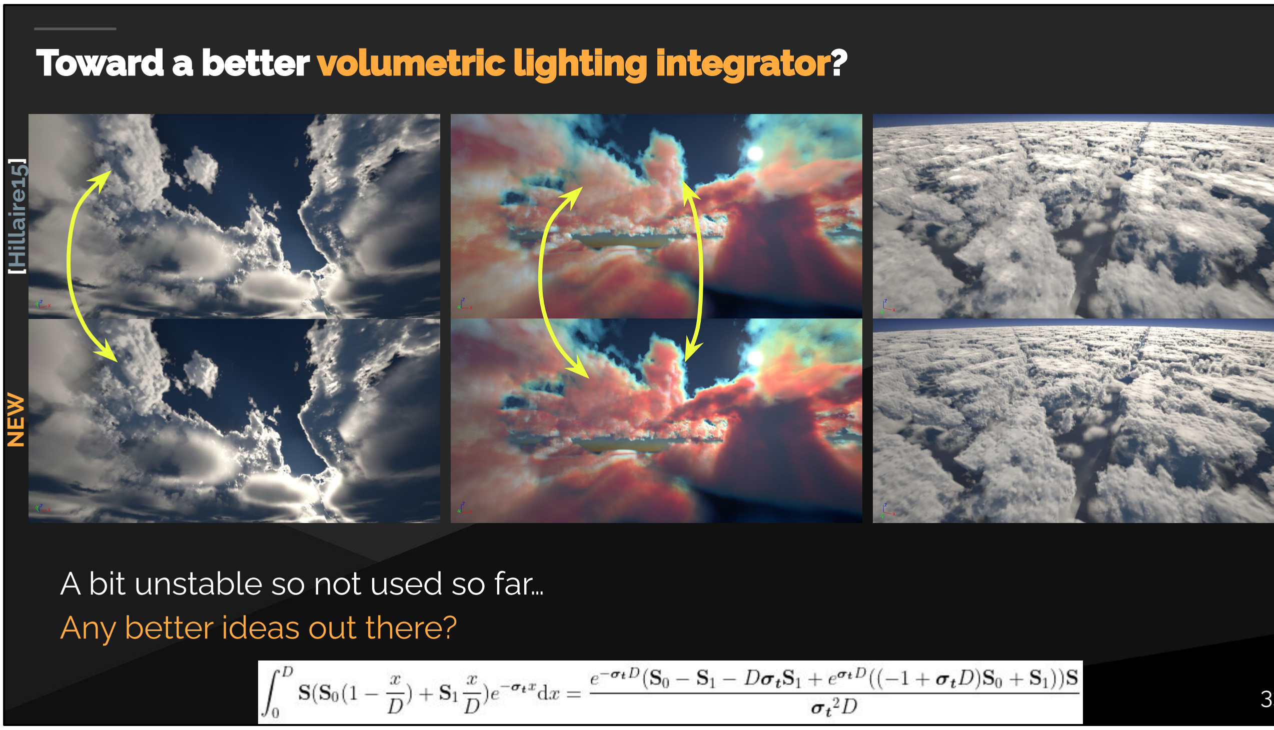
Google slides does not support equations :(
https://www.codecogs.com/latex/eqneditor.php

\int_0^D{\mathbf{S}\mathbf{S}_0 \ e^{-\boldsymbol{\sigma_t} x} \textup{d}x} = \frac{\mathbf{S}\mathbf{S}_0 - \mathbf{S}\mathbf{S}_0 \ e^{-\boldsymbol{\sigma_t} D}}{\boldsymbol{\sigma_t}}

\int_0^D { \mathbf{S} (\mathbf{S}_0 (1-\frac{x}{D}) + \mathbf{S}_1 \frac{x}{D}) e^{-\boldsymbol{\sigma_t} x} \textup{d}x}

$$
=
\frac{ e^{-\boldsymbol{\sigma_t} D} (\mathbf{S}_0 - \mathbf{S}_1 - D\boldsymbol{\sigma_t}\mathbf{S}_1 + e^{\boldsymbol{\sigma_t} D} ((-1+\boldsymbol{\sigma_t}D) \mathbf{S}_0 + \mathbf{S}_1)) \mathbf{S}}{\boldsymbol{\sigma_t}^2 D}
$$

$$
\textbf{S} = \textbf{E} \ \boldsymbol{\sigma_s} \ p(\theta,g)
$$



You can see here the before/after difference. It does help to better define the cloud shapes and shadow.
[click toggle in and out]

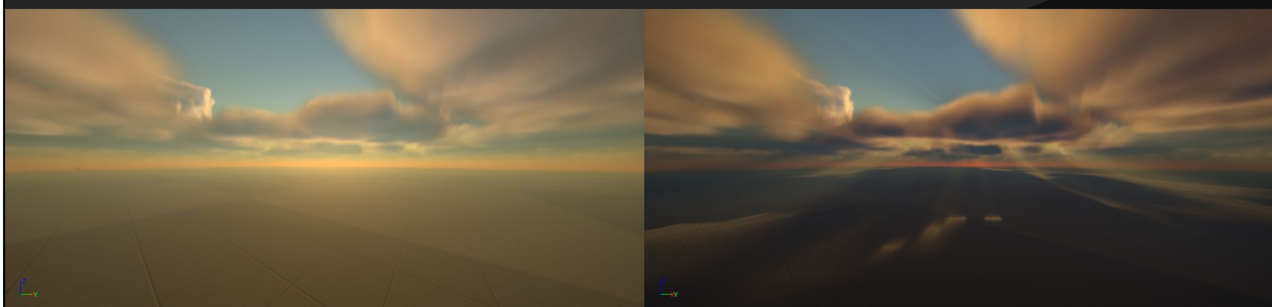
However, we are not using it yet because it is a bit more expensive,
and also a bit unstable due to the division by the squared extinction, which requires clamping in order to avoid numerical precision issues.

I look forward to seeing if any of you use this or if there are even any better ideas out there.

Cloud volumetric occlusion

Cloud occlusion is important [Hillaire 16][Bauer 19]

- Sun light shafts
- Sky top/down ambient occlusion



[click toggle continuously]

Aside from lighting, cloud occlusion is important to evaluate to render sun light shafts within the atmosphere, as shown here on the right.

Cloud volumetric occlusion

Can use Exponential Shadow Map (ESM) [Annen 08][Bauer 19]

- ESM is Beer's Law!
  - with a constant extinction $\sigma_t = c$
  - between ray sample at distance d and occluder at distance z
- When d-z increases, transmittance will always converge to 0

$$f(d,z) = \boxed{e^{-c(d-z)}}$$
$$= e^{-cd} e^{cz}$$

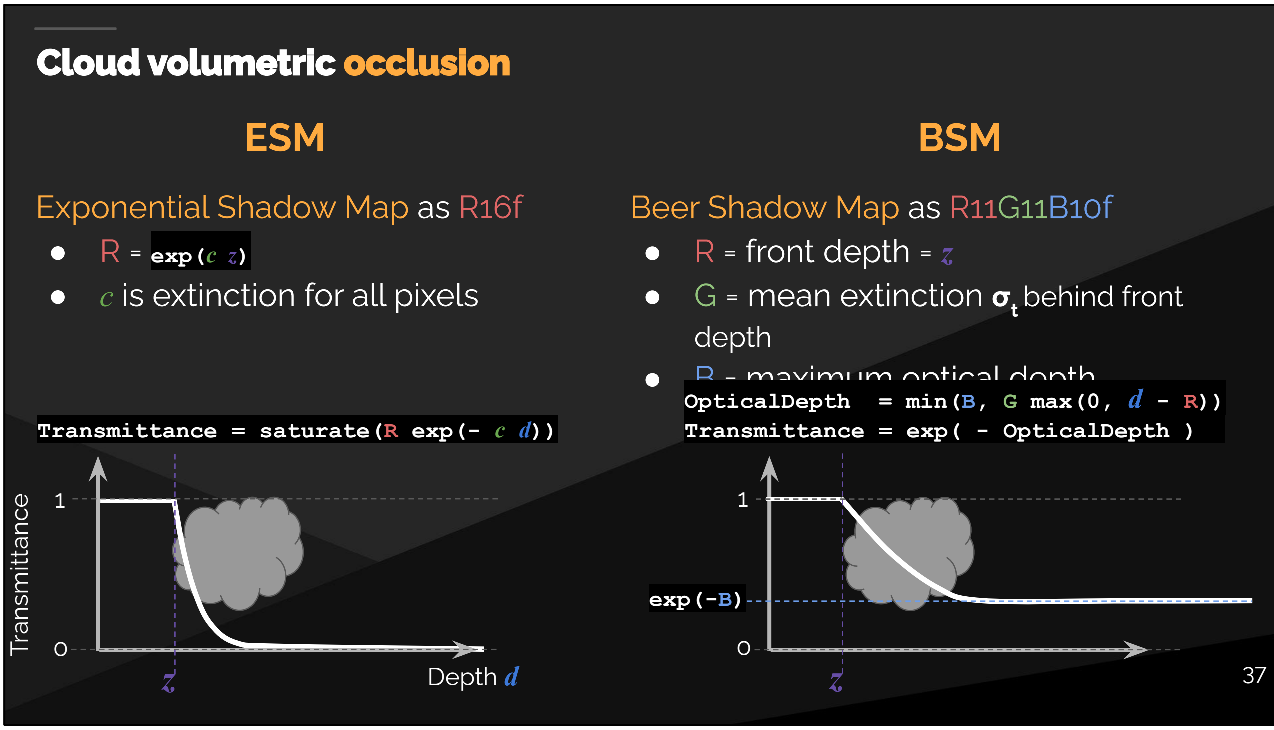
One can use exponential shadow maps, like Bauer.

In case you never noticed, ESMs are exactly Beer's law but with only one constant extinction value for all the texels.
The problem is that it will result in an occlusion that will always converge to a transmittance of 0 over a large distance.

This problem is illustrated here on this almost vertical plane receiving cloud shadows.

[click]
Shadows are getting darker with distance from the cloud layer.
And this is a problem for long distance occlusion when the sun is at the horizon.
And since extinction constant c is the same for all texels, there is no right answer.

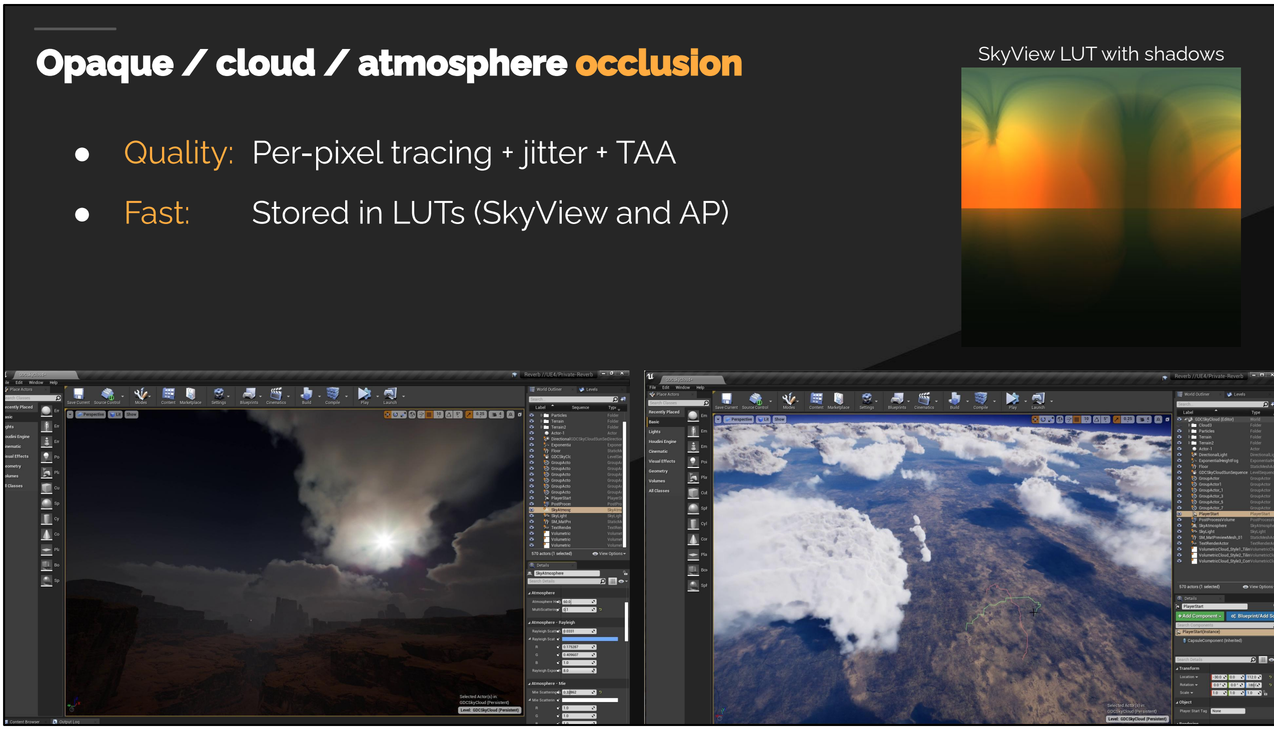
[click]
Instead we propose a new occlusion representation we call *Beer shadow map* that is more consistent and avoids the problems I just mentioned, as you can see here.



So here is a comparison between the ESM and BSM approaches

Beer shadow map is basically the transmittance curve of an homogeneous medium with extinction varying per texel.
It starts at depth Z and has a clamp on the optical depth to stop the transmittance converging toward 0.
All of this data is generated while ray marching to generate the Beer shadow map.

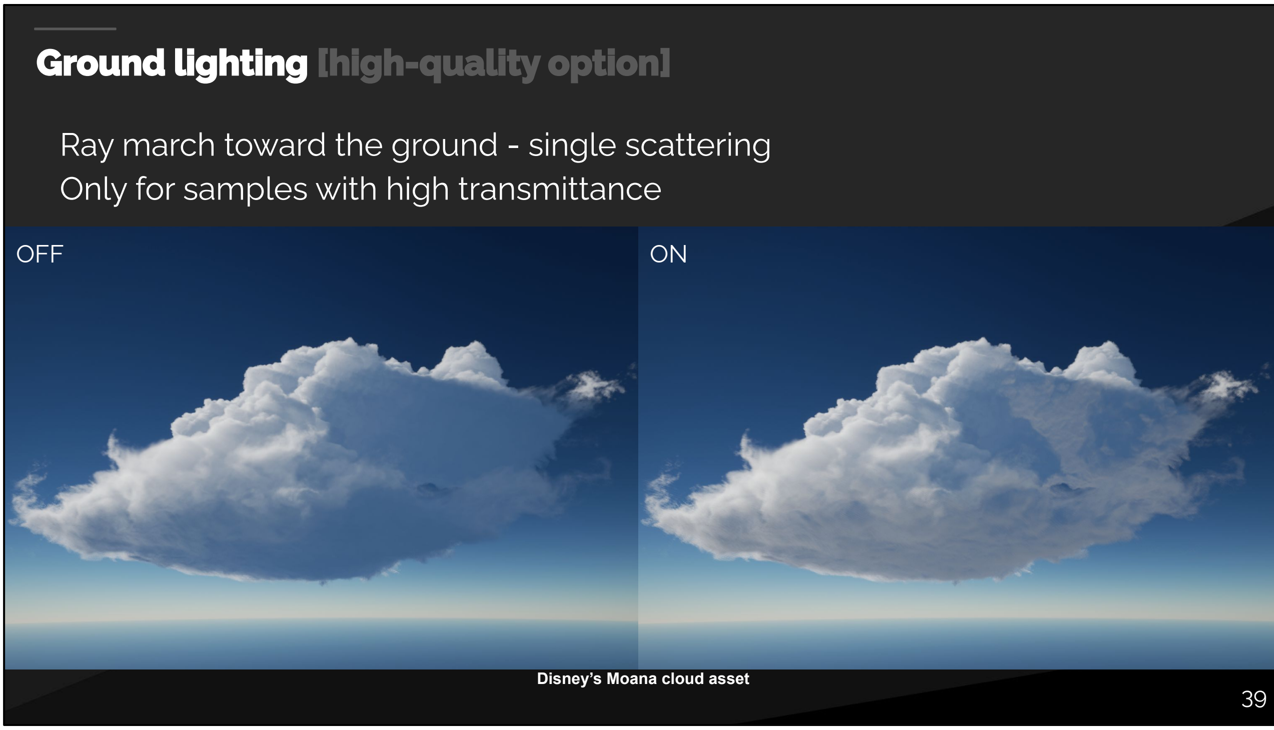
For clouds, this is better than exponential shadow map because it is a better match for media with spatially varying density.
And it avoids the large memory footprint of coefficients that would be required when using transmittance or Fourier opacity maps, for instance.
You can also use a bilateral filter to blur the mean extinction and max optical depth while keeping the front depth untouched, to ensure that light shafts match the cloud shape.



We can now evaluate volumetric shadow using beer shadow maps. We can use per-pixel tracing with sample jitter and TAA to achieve sharpness but at the cost of full resolution rendering.

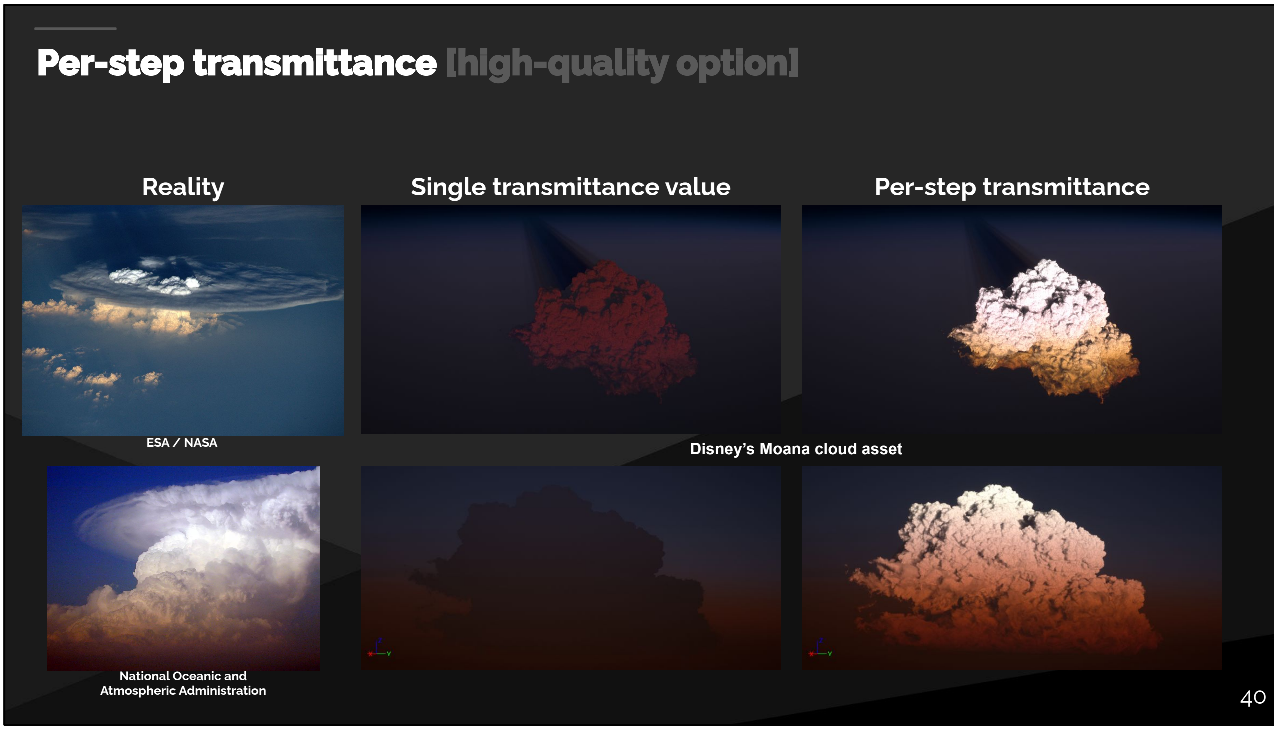
[click]

And if this per-pixel tracing cost is prohibitive for your use case, it is possible to simply store the shadows as part of the Sky-View and aerial perspective LUTs. And you will have to play with their resolution and sample count to achieve good-looking results with your content.

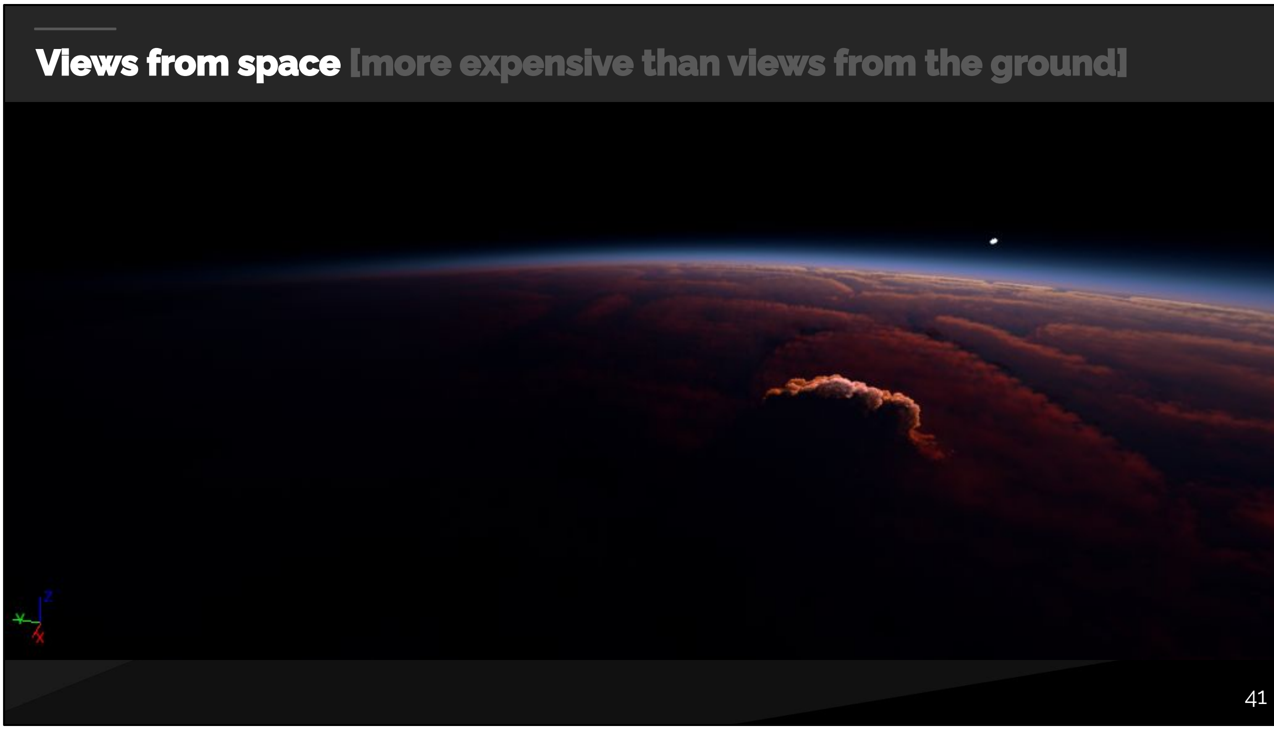


We also have a lot more optional visual features that can be enabled if the user's budget allows it.

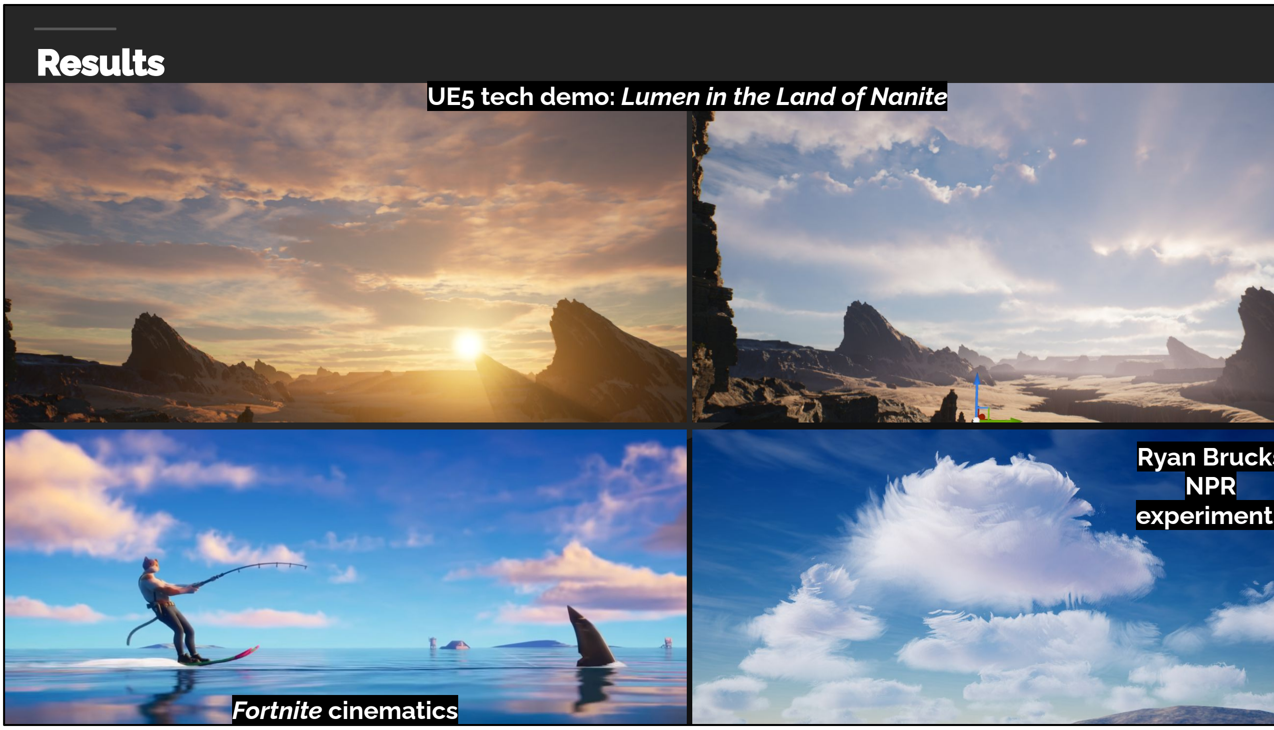
[click toggle to show different]
For instance, it is possible to run a secondary trace towards the ground to evaluate its contribution to the cloud lighting.
This can be very important to help with the perception of the cloud shape.



It is also possible to have the atmospheric transmittance evaluated for each step we take when integrating clouds volumetric lighting. It results in a more complex and realistic look at sunset.



So to recap,
first we have the atmosphere participating media
[click] it can be occluded by clouds
[click] then we have the cloud participating media
[click] that can cast shadow onto itself
[click] and the atmosphere is also applied on the clouds.
[click] and just as an example, this is how it looks if you make the atmosphere thicker:
it remains consistent.



You can see here the tech used in the Unreal Engine 5 real time demo: *Lumen in the Land of Nanite*.
Physically based does not only mean realistic: You can see at the bottom the result of stylised clouds rendering in a *Fortnite* cinematic, as well as an experiment that Ryan Brucks has conducted.

Conclusion

Conclusion

An efficient atmosphere and cloud rendering technique

- Dynamic atmosphere, weather and time of day
- Multiple scattering
  - Sky: physically based approach
  - Clouds: convincing approximation
- Supports view from space and ground

Scalability	Mobile	XB1/PS4	XBX/PS5	Low/High PC	Reference
Atmosphere	✓	✓	✓	✓	Prototype
Cloud	?	✓	✓	✓	Prototype

Available in : Epic Games - UNREAL ENGINE source
Sky only + path tracer : https://github.com/sebh/UnrealEngineSkyAtmosphere

To conclude, I have presented to you the atmosphere rendering technique available in Unreal Engine.
It can scale from high performance to high fidelity rendering and from low-end mobile to high-end platforms.
Only cloud rendering is not reasonably achievable on mobile. For now.

It supports dynamic atmosphere and time of day, approximates multiple scattering using physically based approaches and can be rendered from ground and space view points.

References

[Annen 08] *Exponential Shadow Maps*, Proceedings of Graphics Interface 2008
[Bauer 19] BAUER, FABIAN. *Creating the Atmospheric World of Red Dead Redemption 2: A Complete and Integrated Solution*. Advances in Real Time Rendering, ACM SIGGRAPH 2019 Courses. 2019
[Bouthors 08] BOUTHORS, ANTOINE. *Realistic rendering of clouds in realtime*. PhD thesis. Université Joseph Fourier, 2008
[Bruneton 08] BRUNETON, ERIC and NEYRET, FABRICE. *Precomputed Atmospheric Scattering*. Proceedings of Eurographics. 2008
[Bruneton 17] BRUNETON, ERIC. *A Qualitative and Quantitative Evaluation of 8 Clear Sky Models*. IEEE Transactions on Visualization and Computer Graphics 23.12 (2017)
[deCarpentier & Ishiyama 17] de Carpentier, Giliam and Ishiyama, Kohei. *The Real-Time Volumetric Cloudscapes of Horizon: Zero Dawn*. Advances in Real Time Rendering, ACM SIGGRAPH 2017 Courses. 2017
[Chandrasekhar 50] *Radiative transfer*, 1950
[CIE 03] *Spatial Distribution of Daylight––CIE Standard General Sky*. Standard CIE S 011/E:2003
[Elek 09] ELEK, OSKAR and KMOCH, PETR. *Real-time spectral scattering in large-scale natural participating media*. Proceedings of the Spring Conference on Computer Graphics (SCCG). 2010
[Fong 17] Julian Fong, Magnus Wrenninge, Christopher Kulla, Ralf Habel, *Production Volume Rendering*, SIGGRAPH 2017 Course Notes.
[Kutz 17] , Peter Kutz, Ralf Habel, Yining Karl Li, Jan Novák, *Spectral and Decomposition Tracking for Rendering Heterogeneous Volumes*, ACM Trans. Graph. 31.4 (2017)
[Hillaire 15] HILLAIRE, SÉBASTIEN. *Physically Based and Unified Volumetric Rendering in Frostbite*. SIGGRAPH 2015 Course: Advances in Real Time Rendering
[Hillaire 16] HILLAIRE, SÉBASTIEN. *Physically Based Sky, Atmosphere and Cloud Rendering in Frostbite*. SIGGRAPH 2016 Course: Physically Based Shading in Theory and Practice
[Hillaire 20] HILLAIRE, SÉBASTIEN. *A Scalable and Production Ready Sky and Atmosphere Rendering Technique*, EGSR 2020
[Holzschuch 13] HOLZSCHUCH, NICOLAS and GASCUEL, JEAN-DOMINIQUE. *Double- and Multiple-Scattering Effects in Translucent Materials,* IEEE Computer Graphics and Applications (2013)

All the references for you later...

References

[Jensen 01] JENSEN, HENRIK WANN, MARSCHNER, STEPHEN R., LEVOY, MARC, and HANRAHAN, PAT. *A Practical Model for Subsurface Light Transport*. Proceedings of the ACM on Computer Graphics and Interactive Techniques. 2001

[Nishita 96] NISHITA, TOMOYUKI, SIRAI, TAKAO, TADAMURA, KATSUMI, and NAKAMAE, EIHACHIRO. *Display of the Earth Taking into Account Atmospheric Scattering*. Proceedings of the ACM on Computer Graphics and Interactive Techniques. 1993

[Novak 14] NOVÁK, JAN, SELLE, ANDREW, and JAROSZ, WOJCIECH. *Residual Ratio Tracking for Estimating Attenuation in Participating Media*. ACM Trans. Graph. 2014

[O'Neil 05] O'NEIL, SEAN. *Accurate Atmospheric Scattering*. GPU Gems 2. 2007

[Hosek & Wilkie 12] HOSEK, LUKAS and WILKIE, ALEXANDER. *An Analytic Model for Full Spectral Sky-dome Radiance*, ACM Trans. Graph. 31.4 (2012)

[Perez 93] Perez, R., Seals, R., Michalsky, J, *All-weather model for sky luminance distribution––preliminary configuration and validation*, Solar Energy, 1993

[Preetham 99] PREETHAM, A. J., SHIRLEY, PETER, and SMITS, BRIAN. *A Practical Analytic Model for Daylight*. Proceedings of the ACM on Computer Graphics and Interactive Techniques. 1999

[Schneider 15] SCHNEIDER, ANDREW. *The Real-Time Volumetric Cloudscapes of Horizon: Zero Dawn*. Advances in Real Time Rendering, ACM SIGGRAPH 2015 Courses. 2015

[Schneider 17] SCHNEIDER, ANDREW. *Nubis: Authoring Real-Time Volumetric Cloudscapes with the Decima Engine*. Advances in Real Time Rendering, ACM SIGGRAPH 2017 Courses. 2017

[Wrenninge 13] WRENNINGE, Magnus, Kulla Chris, Lundqvist Viktor, *Oz: The Great and Volumetric*, ACM SIGGRAPH talk, 2013

[Wronski 14] Wronski, Bart, *Volumetric fog: Unified, compute shader based solution to atmospheric scattering*. Advances in Real Time Rendering, ACM SIGGRAPH 2014 Courses. 2014

[Yan 97] YANOVITSKIJ, EDGARD G. *Light Scattering in Inhomogeneous Atmospheres*. Springer-Verlag Berlin Heidelberg, 1997

[Yusov 13] YUSOV, EGOR. *Outdoor Light Scattering*, Game Developers Conference. 2013

[Zinke 08] ZINKE, ARNO, YUKSEL, CEM, WEBER, ANDREAS, and KEYSER, JOHN. *Dual Scattering Approximation for Fast Multiple Scattering in Hair*. ACM Trans. Graph. 27.3 (2008)

References

Noise generators

- Tileable volume noise
  https://github.com/sebh/TileableVolumeNoise
- Nubis noise generator
  https://drive.google.com/file/d/0B-D275g6LH7LNF93dW02MkZSYWM/view

Volumetric assets

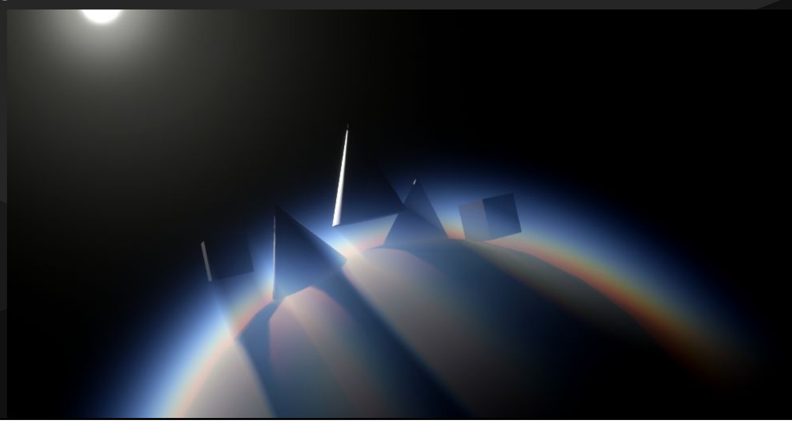
- Disney's Moana cloud asset
  https://www.technology.disneyanimation.com/clouds
- OpenVDB
  https://www.openvdb.org/download/

Thanks!

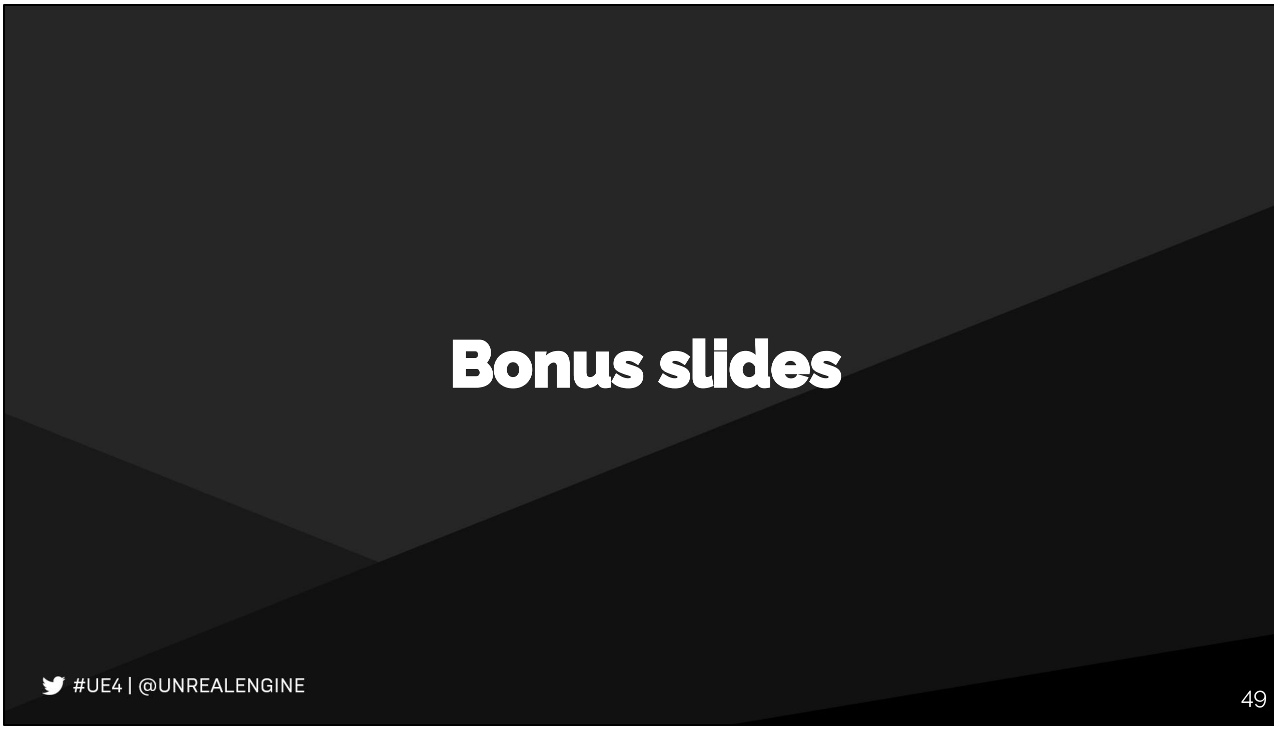
Thanks to the Unreal Engine rendering and art teams!

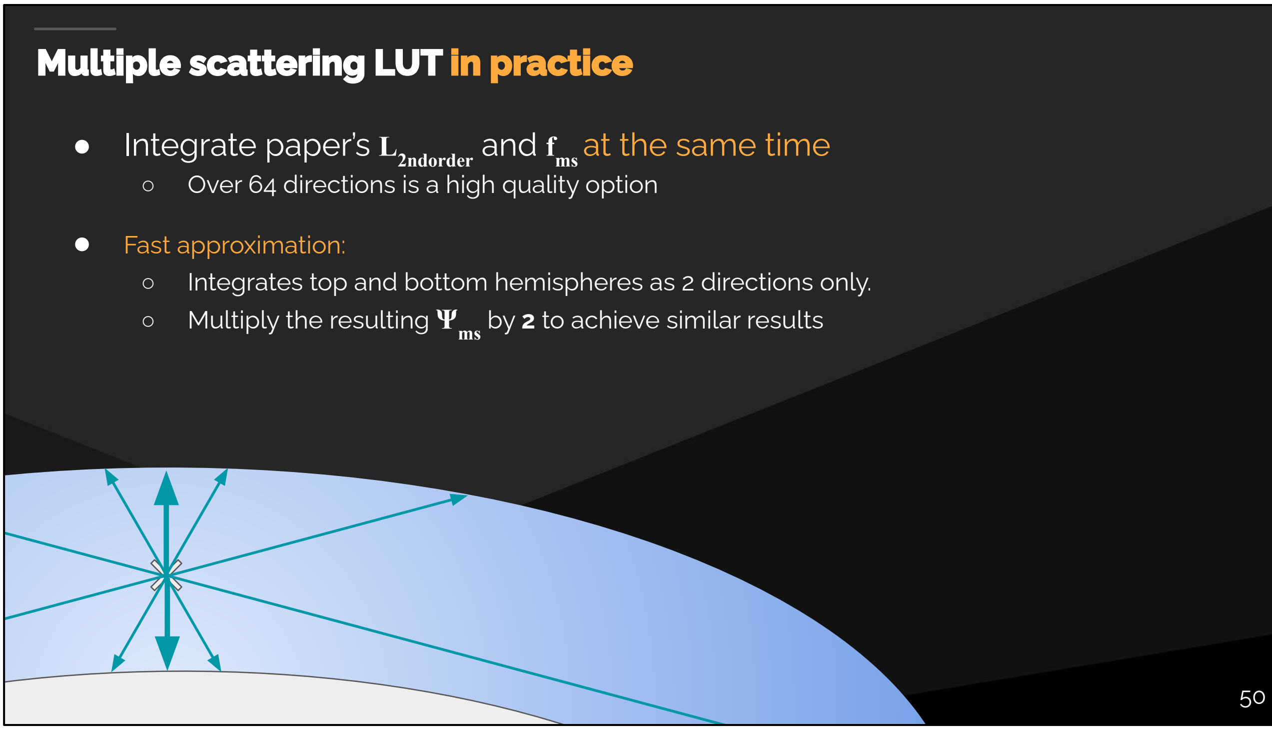
Thanks to course organizers Stephen Hill And Steve McAuley

We are hiring! Epic Game - Unreal Engine



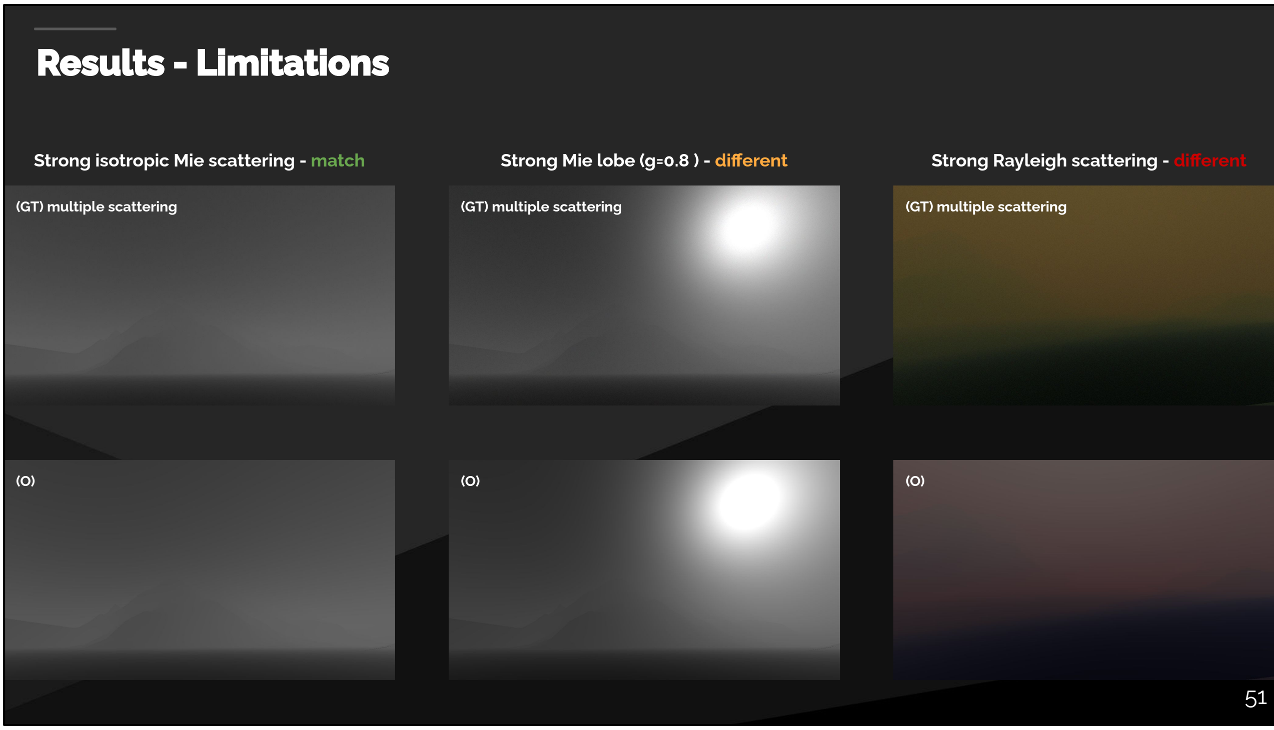
And that is it! Thank you very much for listening.


Bonus slides
#UE4 | @UNREALENGINE
49



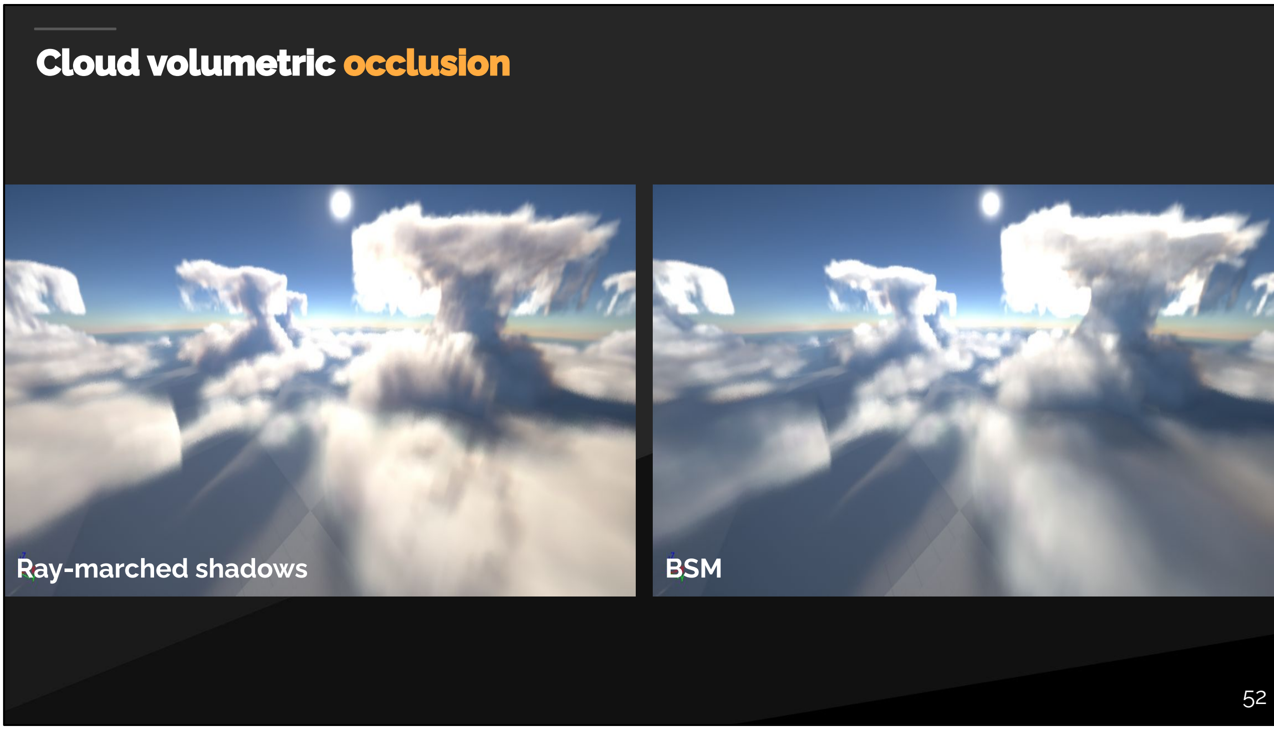
In practice, this multiple scattering contribution is computed from some values such as the scattering second order evaluated by integrating single scattering over the unit sphere according to 64 uniform directions.

This is the high quality version. A faster option interesting to use when running on low end platforms is to only evaluated the hemisphere using two sample: top and bottom only. This is a nice and fast approximation that is close to the ground truth if the contribution is multiplied by a factor of 2.



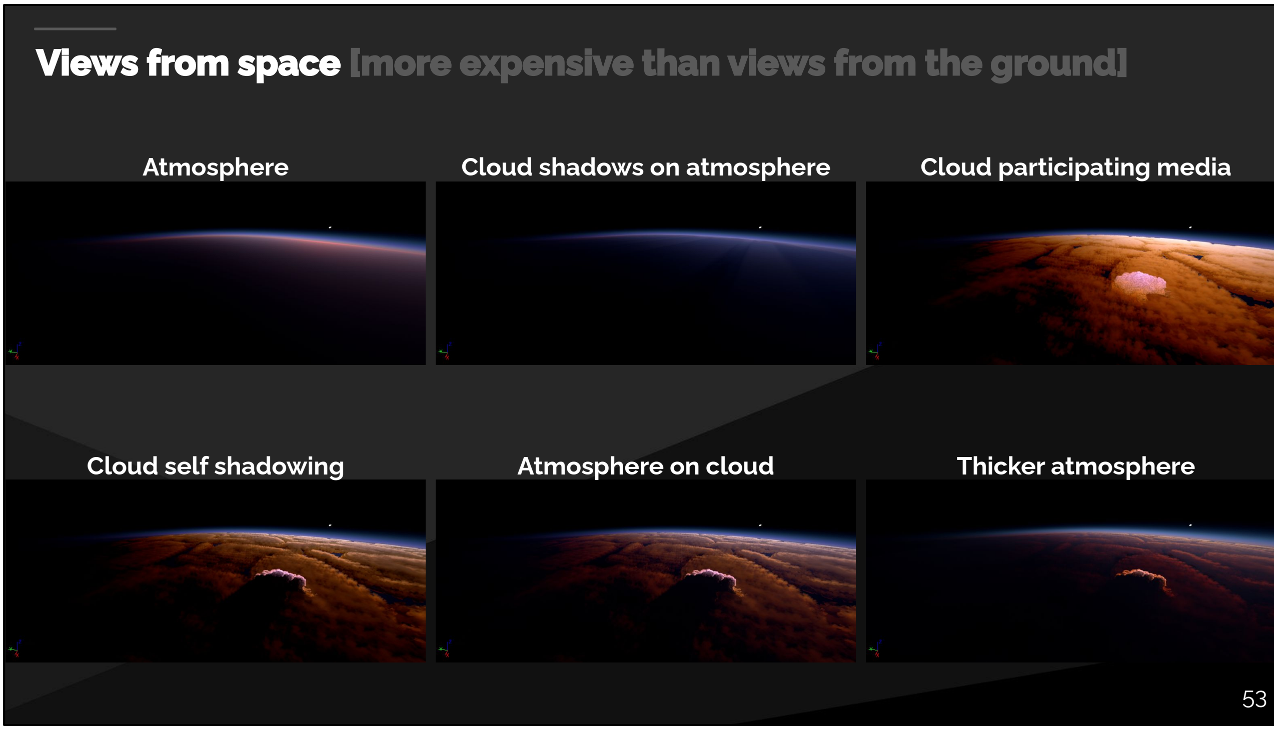
[continuously click back and forth]

There are some caveats to be aware of when using our new multi scattering LUT: it does work nicely for isotropic phase functions but it will slowly diverge when it is made anisotropic.
Last but not least, for very high scattering coefficients that are colored, the multi scattering luminance hue can start drifting as compared to the ground truth as you can see there on the right.



Beer shadow map are better for volumetric shadow but also they can also be used for cloud self shadowing instead of secondary ray marching.

Self shadow on cloud can be evaluated in two ways:
[click] using secondary ray marching to get sharp and colored shadows, but this is a limited shadow tracing distance.
[click] or using Beer shadow map, which is less accurate and not colored, but it is a lot faster and supports a huge shadow distance.



So now let’s recapitulate the important visual components