The 8th ACM/EG Expressive Symposium
C. Kaplan, A. Forbes, and S. DiVerdi (Editors)

Aesthetically-Oriented Atmospheric Scattering

Yang Shen[†1] Ian Mallett[2] and Konstantin Shkurko[2]

[1]Originate
[2]University of Utah

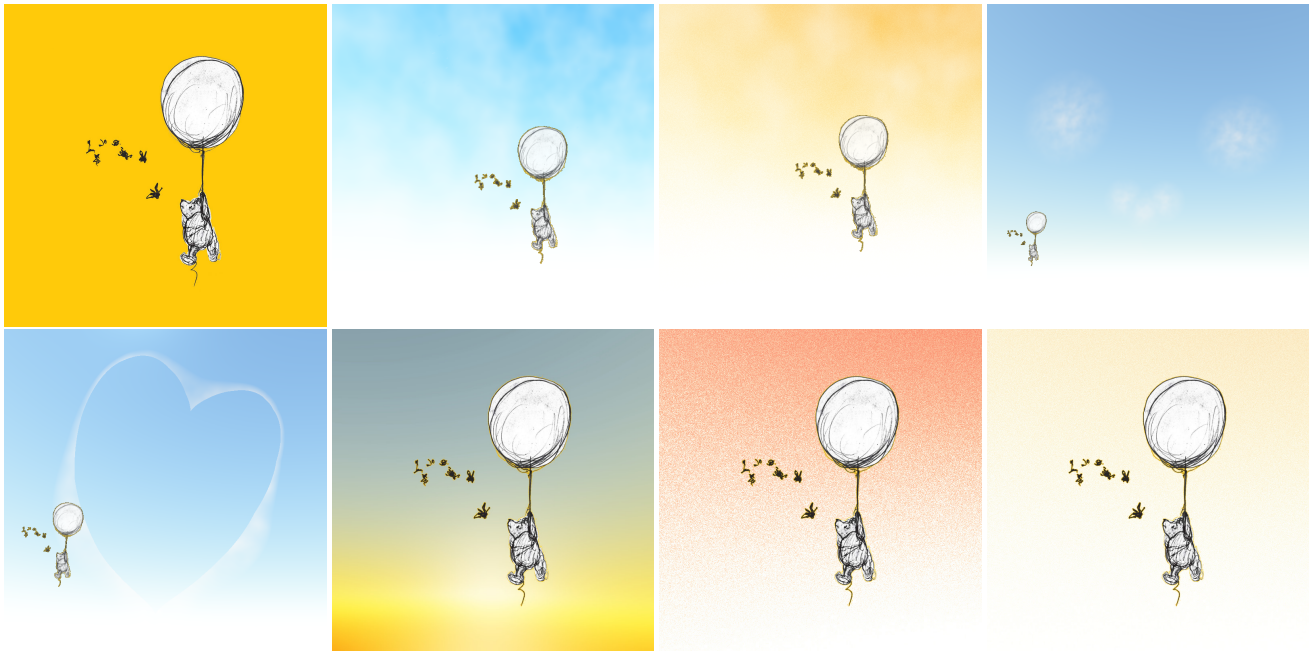

Figure 1: *Exploring Winnie-the-Pooh with AOAS: the reference image on the top left is the poster for the exhibition "Winnie-the-Pooh: Exploring a Classic" in the Museum of Fine Arts, Boston [Mus19]. We might wish to explore the overall feeling for Winnie flying over different sky styles. With AOAS, we are able to intuitively explore different sky styles with an immediate preview–be it a naturally beautiful sky (top second), a yellow sky, subtly turbulent, smiley-shaped, or heart-shaped clouds (top third, top fourth, bottom first), a sunrise (bottom second), or just letting Winnie float in cartoony styles with ambiguous coral or yellow skies (bottom third, bottom fourth).*

Abstract

We present Aesthetically-Oriented Atmospheric Scattering (AOAS): an experiment into the feasibility of using real-time rendering as a tool to explore sky styles. AOAS provides an interactive design environment which enables rapid iteration cycles from concept to implementation to preview. Existing real-time rendering techniques for atmospheric scattering struggle to produce non-photorealistic sky styles within any 3D scene. To solve this problem, first, we simplify the geometric representation of atmospheric scattering to a single skydome to leverage the flexibility and simplicity of skydomes in compositing with 3D scenes. Second, we classify the essential and non-essential visual characteristics of the sky and allow AOAS to vary the latter, thus producing meaningful, non-photorealistic sky styles with real-time atmospheric scattering that are still recognizable as skies, but contain artistic stylization. We use AOAS to generate a wide variety of sky examples ranging from physical to highly stylized in appearance. The algorithm can be easily implemented on the GPU, and performs at interactive frame rates with low memory consumption and CPU usage.

CCS Concepts

• ***Computing methodologies*** → ***Non-photorealistic rendering;***

† Research Is Supported by Originate 20% Time

GLOSSARY

Visual Characteristics of Sky: visual cues that form the appearance of physical sky. This paper discusses three visual characteristics: *sky hue*, *sky pattern*, and *sky gradient*. They are introduced in Section 3.1.

Aesthetic Principle: defines how a visual characteristic of the sky changes. A new sky style can be derived by modulating parameters of an aesthetic principle or combining several aesthetic principles together. For example, aesthetic principles of the sky hue could be:

- Sky hue could be any value in a custom domain of colors: the sky hue is the parameter of this aesthetic principle. Changing it derives a new sky style with a custom hue.
- Sun direction could be any value in the valid domain of sun directions: the sun direction is the parameter of this aesthetic principle. Changing it to correspond to a different time of day derives a different sky style.

1. Introduction

The work arose from our aspiration to render aesthetically pleasing skies, with a sky style that can be interactively configured. The sky, by its nature, affects mood with its variance and dynamism. Often, for example, we feel differently on a snowy day versus a sunny day. This indicates a great potential to explore the sky's aesthetic effect. If we evaluate the source of the sky's impact on the mood, then we find two *primary visual primitives* that constitute the sky's changes, as-perceived by the human visual system: *sky hue* and *sky pattern*. Styles of the sky hue and the sky pattern have been heavily applied in the visual arts to demonstrate the sky's aesthetic effect. For example, in animated films' colorscripts [Ami15] and game development concept art [Nav12], a common method to express the mood of a scene uses a non-photorealistic sky of a non-physically-based hue. In Mark Osborne's 2016 animated film *The Little Prince*, the sadness of the prince's departure is enhanced by the reddish sky, whose color evokes a semblance of departure sadness, especially in Eastern cultures [Zah16]. In Makoto Shinkai's 2016 animated film *Your Name*, when Taki and Mitshuha meet in their dreams, one's imagination cannot fall deeply into a sense of excitement about their fate without the influence of the dynamic sky patterns reflecting the emotions. Van Gogh's *The Starry Night* would feel incomplete without the elegant swirls of the sky.

Such examples motivated our development of the AOAS. We wanted to render the appearance of the sky in real-time with a range of styles, from physically-based to highly-stylized, while maintaining the technical flexibility of the sky style and the simplicity of compositing with any 3D scene. The user should be able to freely change the 3D scene and explore it from as many viewpoints and angles as possible, all with an immediate preview. Aesthetically, the user should be able to derive countless sky styles by intuitively configuring the parameters of the sky model. The sky hue can be configured by the time of day, be assigned to a non-physically-based color such as red or orange, or be otherwise tuned with varying brightness. Sky patterns can be configured to include as many *aesthetic properties* [JLZ05] as possible, such as the effects of *ambiguity* and *complexity*, or simply to contain custom shapes.

Real-time performance is paramount for interactivity, because the immediate feedback is crucial for drastic reductions in the time required for art-direction, as well as the facilitation and motivation of the creative flow for the users [MSRB17]. Unfortunately, existing real-time rendering techniques for atmospheric scattering struggle to produce non-photorealistic sky styles within any 3D scene. In the last two decades, the focus of academic research on real-time rendering of skies broadly falls into two categories. The first accurately simulates the sky color with varying atmosphere models, thereby producing only physically-based sky styles, and improves computational efficiency. The second category focuses on non-photorealistic sky styles, which are mostly seen in video games, with the Skybox being a popular technique [Val15]. Although the technique is capable of producing any sky style with any 3D scene in real-time, the style cannot be configured interactively because the Skybox relies on artwork typically created offline.

We are more curious about a fundamental theory to make any sky style meaningful. Are there common characteristics of the sky, without which the sky would not be recognizable or meaningful? What forms in the sky can be changed freely to make the sky highly artistic? In this paper, we address these two related questions with the observation of essential and non-essential characteristics of the sky. We then devise two technical solutions to flexibly configure compelling sky styles for any 3D scene. Our contributions can therefore be summarized as:

- An observation of sky characteristics. The essential characteristics constrain sky styles to be meaningful and recognizable. Non-essential characteristics can vary to create artistic styles.
- Simplification of the geometric representation of atmospheric scattering for flexible exploration and transfer of sky styles. We simulate the scattering within a single skydome, to benefit from the flexibility of using it when compositing the sky with any 3D scene, and the flexibility of previewing the sky style in a particular scene from any angle on the ground. Our simplification removes the inflexibility of geometric representations of existing atmospheric scattering techniques when compositing 3D scenes.
- Creation of transferable, meaningful, and non-photorealistic sky styles using atmospheric scattering. We create artistic, non-photorealistic styles by applying aesthetic principles to non-essential sky characteristics, resulting in a larger aesthetic domain of skies. The user is able to intuitively configure these artistic parameters to explore or transfer the sky styles whether they are physically-based on non-photorealistic in appearance.

In Section 2, we introduce related techniques of expressive sky rendering. In Section 3, we introduce AOAS and our approach to rendering atmospheric scattering in one skydome with non-physically-based sky styles. In Section 4, we demonstrate examples of sky style configurations both in a custom OpenGL project and in Blender Game Engine (BGE). In Section 5, we conclude with a discussion of limitations and future work.

2. Related Work

We combine the core ideas of the following techniques to render atmospheric scattering using a single skydome:

- The Skydome model of the Skybox technique [Val15]
- Single-scattering equations of Rayleigh and Mie scattering, as-described by Nishita et al. [NSTN93]
- A computational simplification of Nishita et al.'s single scattering equations by O'Neil [ONe05]

Nishita et al. propose scattering equations to display the Earth from the outer space including atmospheric color [NSTN93]. The scattering equations form the basis of more recent academic work on simulating the sky color in modern GPU shaders for two reasons. First is their accuracy in simulating essential characteristics of atmospheres with single-scattering. The second is their high computational performance derived from evaluating the scattering integral of the optical depth from any viewpoint in the atmosphere along a ray towards the outer atmosphere. We choose these single scattering equations to render the physically-based sky color because of these reasons. The method did not achiere veal-time performance on an IRIS Indigo Elan.

O'Neil describes a CPU algorithm based on Nishita el al.'s scattering equations [ONe04]. It is based on a precalculated 2D lookup table with four floating-point channels. Calculating the color for each vertex requires several lookups into the table, with extra calculations around each lookup. At the time, no GPU could support such operations in a shader in a single pass. In a followup paper, O'Neil eliminates the use of lookup tables with a mathematical approximation of optical depths, parametrized by any viewpoint in the atmosphere and any view direction toward the outer space [ONe05]. The algorithm can be implemented in a GPU shader and runs in real-time.

We found O'Neil's later method to be the most flexible on any platform supporting modern GPU shaders because it does not rely on lookup tables. However, the methods mentioned above rely on two concentric spheres to represent the Earth and its atmosphere, which is cumbersome to use with arbitrary 3D scenes placed on the ground. It is both tricky and inconvenient to transform the 3D scene to fit into the thin layer between the two spheres.

Bruneton and Neyret replace the two spheres with a screen-space quad to render both the ground and the atmosphere visible to the viewpoint [BN08]. Each quad fragment is shaded by sampling precalculated 3D textures, proposed by Schafhitzel et al. [SFE07], along the view ray. The texture stores scattering integrals of the multiple-scattering equations, an extension of Nishita et al.'s single-scattering formulation. This method performs highly on modern GPUs and supports rendering multiple-scattering effects with any static 3D scene on the ground and a dynamic aerial perspective. However, because the scene structure is taken into account, any changes to the 3D scene require a re-calculation of the 3D textures which is not fast enough for real-time preview. Changing the scene without the re-calculation results in a physically-incorrect appearance of the ground colors, shadows and light shafts.

Dobashi et al. render physically-based sky appearance by proposing atmospheric scattering equations also based on Rayleigh and Mie Scattering theory [DYN02]. The method allows for dynamic exploration of the scene in real time because it relies on texture memory for the lookup tables storing scattering intensities and attenuation ratios as functions of altitude, view direction, and the sun direction. However, because of the reliance on the texture memory, the method is more memory intensive, less flexible and less performance than the mathematical approximation proposed by O'Neil [ONe05]. Finally, because the method renders the sky using a screen-space quad, compositing the sky with 3D models is not as intuitive for the user.

The Skybox technique renders any sky style in real-time with any 3D scene [Val15]. The technique first bakes the sky color into a sky texture, then applies this sky texture onto a skybox or a skydome in the rendering loop. The advantages of the Skybox technique include rendering animated 3D scenes with user interaction in real-time with any sky style, which benefits from versatile visual forms enabled by varied sky textures, the flexibility and the simplicity of the skybox and skydome when including any 3D scene, and the low computational cost required to render a texture onto a skybox or a skydome. However, it is impossible to interactively configure the sky's style using the Skybox technique, because configuring the sky style requires creating a new sky texture. This process could be time-consuming, preventing real-time updates.

Between the geometric representations of atmospheres – screen-space quad used by Bruneton and Neyret [BN08]; the skydome or skybox used in the Skybox technique [Val15] – we found the latter to be flexible and simple-to-include with any 3D scenes, and to allow for exploration of the scene from any viewpoint towards any view direction. Since skydomes are more flexible than skyboxes due to the smoother surface, we chose skydomes for the sky color.

Reproducing the appearance of clouds which are integral to sky style is an ongoing area of research. Roden and Parberry blend cloud textures with noise to render dynamic clouds with customized shapes [RP05]. Because base cloud shapes are derived from from the real world pictures, it is difficult to reproduce highly-customized cloud shapes like drawings or animals. Using a more sophisticated method, Elek et al. simulate light transport within clouds at a low computational cost by utilizing a temporally-coherent illumination caching process and a novel representation of angular distribution of illumination within clouds [ERWS12]. Arbitrary cloud shapes are defined by a 3D density field, and performance can be an issue when too many are simulated. Although Perlin Noise is not designed for cloud rendering, it captures the random patterns in clouds and can be computed efficiently [Per85; Per02]. Perlin Noise can potentially support designing artistic clouds of arbitrary shape without noticeable performance penalties.

3. Aesthetically-Oriented Atmospheric Scattering (AOAS)

We evaluate the sky characteristics in §3.1, discuss how to render them with atmospheric scattering in §3.2, and how to change them to create non-photorealistic sky styles in §3.3.

3.1. Observation of Sky Characteristics

Evaluating the recognizable characteristics of the sky, we can find the three following characteristics:

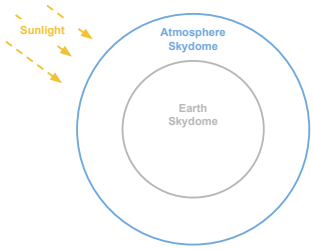

Figure 2: *The Earth (inner, gray) and the atmosphere (outer, blue) skydomes in 2D. In our method, they are concentric 3D spheres.*

Sky hue: captures hues appearing in the sky color. During the daytime of a sunny day, the blue color is the primary hue. During the sunset or sunrise, orange, yellow, and red colors dominate the hue. Determined by the scattering mechanisms of the Earth's atmosphere, the sky hue varies a lot based on the time of day, weather conditions, pollution, etc.

Sky pattern: captures the finer variance in the sky color. Clouds are a common example. Influenced by temperature, wind, etc., clouds give the impression of irregular changes in the form andthe color of the sky without recognizable patterns.

Sky gradient: captures the gradual change in the appearance of the sky from the horizon to the apex. The gradient is largely stable, since it is determined by the physics of light scattering within the Earth's atmosphere, and the direction and intensity of sunlight.

Based on our observations, we believe that the non-stable characteristics of the sky create an opportunity for artistic expression. More specifically, we believe that specifying non-photorealistic values for the sky hue and the sky pattern can generate artistically-driven sky styles, however the sky gradient must remain physically-based to make the sky recognizable. Despite this constraint, a variety of imaginative artistic styles are achievable.

3.2. Atmospheric Scattering

3.2.1. Single Skydome

Our approach renders the atmospheric scattering using a single atmosphere skydome, unlike the approach proposed by O'Neil that also requires the Earth skydome to be located within [ONe05]. The configuration is shown by Figure 2 with the inner sphere representing the Earth and the outer sphere representing the atmosphere.

Throughout this paper, we rely on the *skydome space*, the *Earth space*, and the transformation between them, as shown in Figure 3. The *Earth space* is the world space that combines both the atmosphere skydome and the Earth skydome centered at the origin of the atmosphere skydome. The *skydome space* is the model space of just the atmosphere skydome. The position of an object in the skydome space can be transformed into the Earth space with a linear translation. We calculate the sky color in the Earth space, similar to O'Neil's method, but with the ray origin within the skydome space.

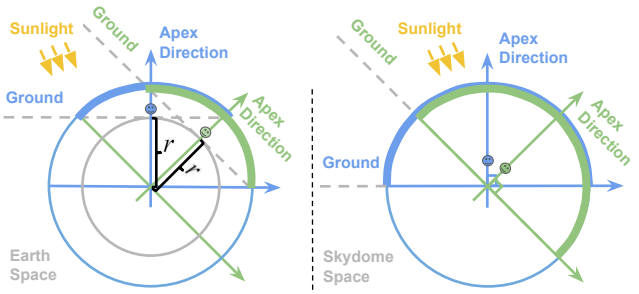

Figure 3: *Illustration of the Earth space and the skydome space. To transform from one space to the other: translate the viewer's position in the skydome space along the apex direction proportional to the ratio of the skydome radii. Two ground planes (blue and green) are shown as examples.*

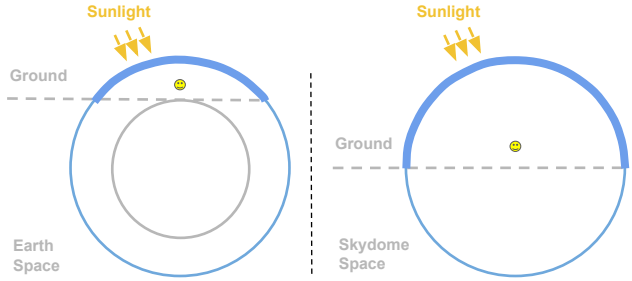

Figure 4: *Visualization of the ground plane and the sky color coverage as perceived by the viewer (**left**) with both skydomes present, and (**right**) with only the atmosphere skydome. The dotted gray horizontal lines represent the ground plane, and the thick blue arc represents the sky color coverage.*

3.2.2. Input Parameters

Removing the Earth skydome requires the following two changes in how the atmosphere skydome is treated, illustrated in Figure 4:

- Instead of being placed at the surface of the Earth skydome, the ground plane is centered at the atmosphere skydome.
- Because the ground plane has moved, the sky now covers the entire upper hemisphere of the atmosphere skydome, instead of the thin layer in between the sky and the Earth skydomes.

The two changes affect the input parameters of the O'Neil's method [ONe05] to calculate the sky color along a view direction by marching along a ray. The parameters are visualized in Figure 5 and include:

- View direction of the viewer in Earth space: $\vec{d}$
- Position of the viewer in Earth space: P_v
- Position of ray-marching sample in Earth space: P_i

Our approach shares only the position of the viewer in the skydome space, P_v^s. The other parameters must be rederived.

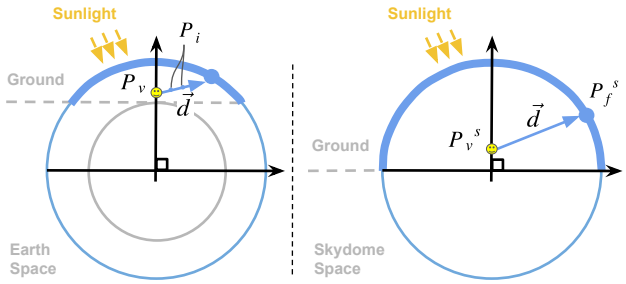

Figure 5: *Visualization of the input parameters to simulate atmospheric scattering.* ***Left:*** *viewer's position in Earth space P_v, viewer's direction in Earth space $\vec{d}$, and the position of ray-marching sample in Earth space P_i.* ***Right:*** *viewer's position in skydome space P_v^s, a fragment's position in skydome space P_v^s, and the viewer's direction towards the fragment $\vec{d}$.*

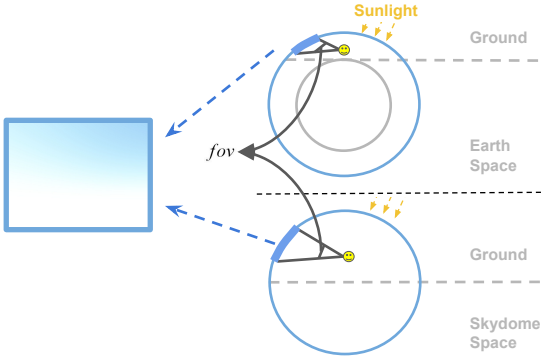

Figure 6: *The sky color along a view direction as perceived by the viewer remains unchanged (**left**) when the viewer is located above the center of the ground plane in either the Earth space (**right upper**) or the skydome space (**right lower**).* fov *is field of view.*

View direction in Earth space is the same as the direction of the viewer in the skydome space because the sky color along a view direction as perceived by the viewer remains unchanged, as illustrated in Figure 6. The transformation between the Earth and the skydome spaces ensures this. Consider a fragment of the atmosphere skydome located at P_f^s in the skydome space. The direction from the viewer located at P_v^s to that fragment in the skydome space is computed as a simple vector difference:

$$\vec{d} = \frac{P_f^s - P_v^s}{\left\| P_f^s - P_v^s \right\|} \tag{1}$$

Position of the viewer in Earth space. The geometric radii of the Earth with and without the Earth skydome are different. Without the Earth skydome, the geometric radius is set to 0. With the Earth skydome, the geometric radius of the Earth is set to the ratio r of the Earth radius to the atmosphere thickness which is assumed to be $T = 1$ in O'Neil's method [ONe05].

The difference in radii requires the position of the viewer to be transformed from the skydome space to the Earth space. To enable such a transformation, we must assume that the apex direction of the viewer remains the same in both spaces, as illustrated in Figure 3. The apex direction of the viewer is the vector from the Earth's origin towards the viewer position. We translate the viewer position along the apex direction to transform the position from the skydome space to the Earth space. The transformation relies on the ratio of the geometric radii:

$$P_v = P_v^s + r \cdot \frac{P_v^s - O}{\|P_v^s - O\|}\,, \tag{2}$$

where O is the Earth center, a zero vector.

Position of a ray-marching sample in Earth space is calculated based on the position of the previous ray-marching sample in Earth space P_{i-1}, the view direction of the viewer in Earth space $\vec{d}$, and the predefined ray-marching step t. The position of the first ray-marching sample, P_0, is set to the position of the viewer in Earth space, P_v:

$$\begin{aligned} P_i &= P_{i-1} + t \cdot \vec{d}, \\ P_0 &= P_v \end{aligned} \tag{3}$$

3.2.3. Full Solution

We can use the provided parameters in Earth space to compute the sky color within a fragment shader according to the following formulation:

$$I_v(P_v, \vec{d}) = \int_{P_a}^{P_b} I_{\text{Rayleigh}}(P_v, \vec{d}, P) + I_{\text{Mie}}(P_v, \vec{d}, P)\, dP\,, \tag{4}$$

where I_v is the sky color as perceived by the viewer. The integral accumulates the in-scattered light towards the viewer at each sample position P. A detailed explanation of light scattering is outside of the scope of this paper, and we refer the interested readers to see O'Neil [ONe05].

With our proposed approach, the coverage and the color distribution of the sky remains invariant to scaling transformations of the atmosphere skydome. The invariance of the skydome space offers the user the flexibility of scaling the atmosphere skydome as they see fit to composite with 3D scenes. We demonstrate this in Figure 7.

3.3. Aesthetic Principles

We notice that the atmospheric scattering varies less in reality than its presentation in the visual arts. For example, blue is the only possible sky hue in the middle of a sunny day, whereas other colors are sometimes present in artistic depictions, such as green and yellow in the video game Journey [Nav12] for example. For this reason, we apply a set of aesthetic principles to the sky hue and the sky pattern to enrich the sky styles.

Ambiguity in the Sky Pattern: "Ramachandran and Hirstein argue that under the right conditions ambiguity itself can be a source

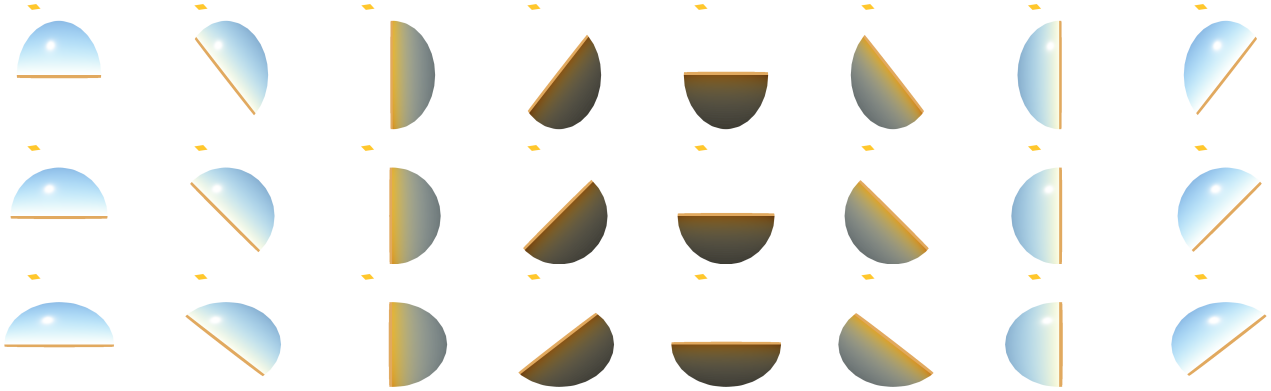

Figure 7: *Sky color and distribution perceived by the viewer stays invariant of the scaling transformation of the atmosphere skydome. Skies within the same column share the same viewer position in skydome space, which is different between columns.* ***Top to bottom:*** *tall, regular, and short skies. All skies share the same sunlight direction - from skydome center to yellow quad.*

of pleasure. For example the Mona Lisa's smile." [Goo02] We believe the impression of ambiguity can be achieved by rendering perturbations of reality.

To express the ambiguity, one can modify the view directions to produce a perturbation to the sky color, while maintaining a plausible color palette. The more the view directions are perturbed, the more ambiguous the sky color appears.

The basic process perturbs the viewing directions toward fragments of the atmosphere skydome. The perturbation range is configurable to produce different levels of ambiguity. When rendering atmospheric scattering in the atmosphere skydome, as discussed in §3.2, a fragment of the atmosphere skydome is shaded by the sky color along its corresponding view direction $\vec{d}$ calculated using Equation 1. Here, we first perturb the viewing direction for a fragment, thereby shading the fragment with the sky color along a related, but different, viewing direction.

We generate a new view direction using the following:

$$\vec{d} = \frac{P^s_{f'} - P^s_v}{\left\| P^s_{f'} - P^s_v \right\|}, \tag{5}$$

where $P^s_{f'}$ is a random point in a neighborhood area of the fragment P^s_f. The size of the neighborhood is configurable, determining how much the view direction is perturbed. We use l_{pt} to represent the size of the fragment's neighborhood:

$$P^s_{f'} = P^s_f + \langle \text{rand}(l_{pt}),\ \text{rand}(l_{pt}),\ \text{rand}(l_{pt}) \rangle \tag{6}$$

We illustrate the perturbation procedure in Figure 8 and demonstrate the effect of the resulting ambiguity in the sky pattern with different values of l_{pt} in Figure 9.

Cloud Complexity for the Sky Pattern: An appearance of a cloud can be achieved by modulating the saturation of a fragment's sky color using procedural noise. We choose the Improved Perlin Noise [Per02] for cloud effects because of its artistic value and the computational compatibility with fragment shaders. Artistically, the Perlin noise captures the beauty of randomness in nature. Computationally, it generates continuous noise values given continuous

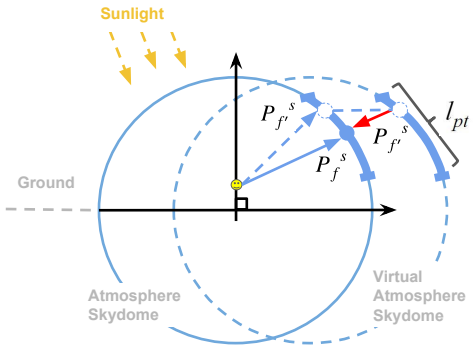

Figure 8: *Perturbing the view direction. The sky color along the view direction towards fragment P^s_f* ***(solid blue arrow)*** *is obtained by the unperturbed sky color of another fragment $P^s_{f'}$* ***(red arrow)****. $P^s_{f'}$* ***(dashed blue arrow)*** *is a random fragment from a neighborhood of P^s_f with configurable size l_{pt}.*

data as input, is controllable in terms of noise size, and can be computed efficiently and asynchronously at the fragment level [Per85].

The fragment's noise is computed based on the fragment position in the skydome space P^s_f and the configurable noise size S_{noise}. The size parameter controls the degree of the cloud complexity:

$$noise = \text{Perlin}(P^s_f, S_{noise}) \tag{7}$$

We can tune the saturation of the fragment's sky color by first converting the color from RGB into HSV color space [Ago05], then scaling the saturation component of the result by the noise, and finally converting the scaled sky color back to the RGB color space:

$$\begin{aligned} \vec{c}_{hsv} &= \text{RGB\_to\_HSV}(\vec{c}_{rgb}) \\ \vec{c}_s &= \vec{c}_s \times noise \\ \vec{c}_{rgb} &= \text{HSV\_to\_RGB}(\vec{c}_{hsv}) \end{aligned} \tag{8}$$

The effect is demonstrated in Figure 10.

Figure 9: *Sky patterns generated by applying the ambiguity principle, increasing from left to right.*

Figure 10: *Sky patterns generated by applying the cloud complexity principle, increasing from left to right.*

Cloud Shape for the Sky Pattern: It is possible to create clouds shaped by the user-specified sparse points or splines. Clouds can be rendered on the skydome based on distance to a particular point or spline, using linear interpolation to fade out with distance. Examples of user-defined clouds, one shaped like a smiley and another like a heart, are given in Figure 1.

More Colors for the Sky Hue: in atmospheric scattering, the sky hue is determined by the *scattering constants* of Rayleigh Scattering, which are fixed. Because they are constants, aesthetically they limit the sky color to physically-based styles. We introduce a new aesthetic principle: replacing the Rayleigh Scattering constants by a three-dimensional vector to derive sky hues which can lead to non-physically-based sky colors. We provide examples in Figure 11.

Day Cycle for the Sky Hue: the sun direction determines whether the sky appears blue (noon on a clear day), or orange (at sunset), for example. These styles evoke different feelings.Hence, we introduce another aesthetic principle to configure the sun direction. We demonstrate the sky styles produced by different time of the day in Figure 12.

3.4. Implementation Details

For completeness, we briefly describe our implementation from the perspective of the rendering pipeline. From the CPU, vertices of one skydome are passed to the GLSL shader program as a vertex buffer. At this step, the geometric model of AOAS differs from O'Neil's method, which would pass two skydomes. The GLSL shader program then simulates the atmospheric scattering for the fragments rasterized from the vertex buffer. This step also includes user-specified artistic transformations described in the previous subsections. The output of the atmospheric scattering is then post-processed as-desired to form the final sky color of the fragment.

4. Sky Examples

In-addition to the technical examples shown previously, we present some results demonstrating how AOAS was successfully used to

Figure 11: *Using color to drive the sky hue.* ***Left to right:*** *gray sky, yellow sky, blue sky, and coral sky.*

design both physically-plausible and stylized skies. In Figure 1, we explore sky styles for a 2D character, from physically-based to cartoony in appearance. The ability to rapidly configure the feeling of the sky was useful in helping the artist achieve their aesthetic goal. In Figure 13, we render the sky above a futuristic city at sunrise. The artist adjusted the time to midday, and then stylized the sky by adjusting the sky hue. We note that, although the sky would not be mistaken as photorealistic, it is still recognizable as a sky because it respects the gradient characteristic. At the same time, however, it is evocative, helping to convey the dystopian feeling of the city.

5. Conclusion and Future Work

We propose a simple technique, AOAS, for exploring sky styles of 3D scenes. By observing visual characteristics of the sky from an artistic perspective, we determine their importance to the sky being recognized as a sky. Moreover, we propose configuring non-essential sky characteristics to extend the style artistically while constraining the essential characteristics so as not to lose the sky's essence.

AOAS currently cannot faithfully tint the 3D scene based on the light reflecting from the sky, so as to fully integrate the sky style. This is caused by a lack of simulation of the global illumination in our method. Using real-time global illumination to fully integrate the sky style into the scene is an exciting future direction. Other avenues for future work include handling other atmospheric effects on the sky from rainbows, snow, rain, etc.

While our work is preliminary covering the needs of artists for designing sky styles, we hope our attempt at combining the aesthetic principles and physically-based rendering in real-time forms a reliable direction for creating expressive computer-generated art. While physically-based rendering greatly replicates reality, inner feelings have to be expressed through a different means. To our best knowledge, aesthetic principles lie at the very core of expressing emotions and incorporating them into rendering techniques could help derive artistic effects through formalized parameters.

6. Acknowledgements

Special thanks to M. Handler, K. Goslar, P. Pizzo, Z. Ren, and A. Stampoulis for encouragement, feedback and advice. We would like to thank the anonymous reviewers for their feedback. This research is supported by Originate 20% Time.

References

[Ago05] AGOSTON, M. K. *Computer Graphics and Geometric Modelling. Implementation & Algorithms*. Springer-Verlag London, 2005 6.

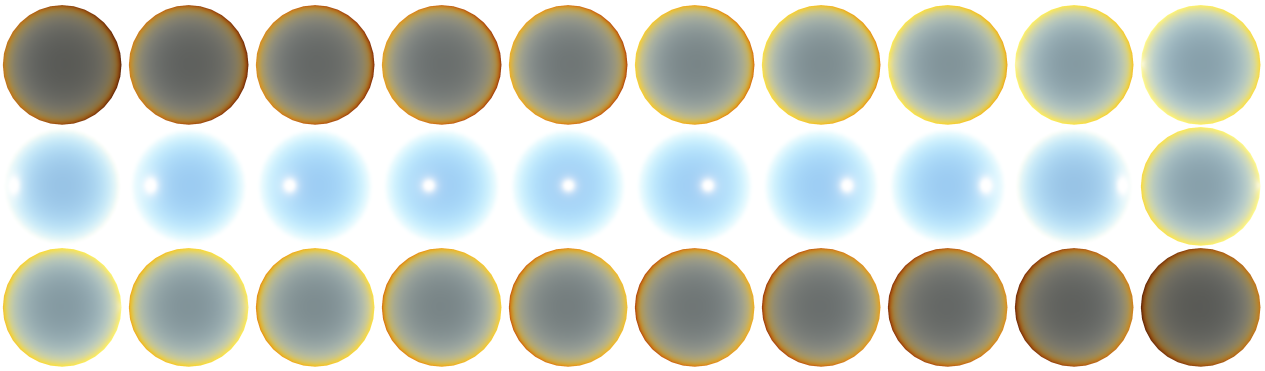

Figure 12: *Day cycle driving the sky hue.* ***Left to right:*** *increasing time of day.* ***Top row:*** *sunrise;* ***middle row:*** *day time;* ***bottom row:*** *sunset.*

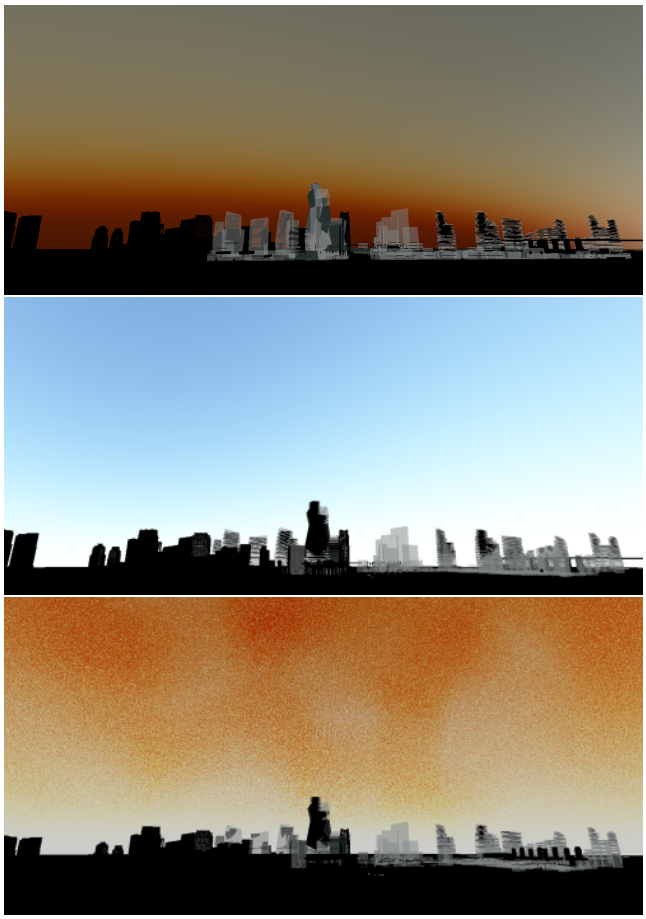

Figure 13: *Futuristic city rendered in BGE with highly stylized sky.* ***Top:*** *sunrise effect.* ***Middle:*** *crystal blue sky.* ***Bottom:*** *combination of a red sky and yellow sky, each with visible clouds.*

[Ami15] AMIDI, A. *The Art of Pixar: 25th Anniversary: The Complete Color Scripts and Select Art from 25 Years of Animation*. 2015 2.

[BN08] BRUNETON, E. and NEYRET, F. "Precomputed Atmospheric Scattering". *Proc. of EGSR '08*. 2008 3.

[DYN02] DOBASHI, Y., YAMAMOTO, T., and NISHITA, T. "Interactive Rendering of Atmospheric Scattering Effects Using Graphics Hardware". *Proc. of HWWS '02*. 2002 3.

[ERWS12] ELEK, O., RITSCHEL, T., WILKIE, A., and SEIDEL, H.-P. "Interactive Cloud Rendering Using Temporally-coherent Photon Mapping". *Proc. of GI '12*. 2012 3.

[Goo02] GOOCH, B. "Ramachandran and Hirstein's Neurological Theories of Aesthetic for Computer Graphics". *SIGGRAPH Courses* (2002) 6.

[JLZ05] JIANLI, L., LUGHOFER, E., and ZENG, X. "Aesthetic Perception of Visual Textures: A Holistic Exploration Using Texture Analysis, Psychological Experiment, and Perception Modeling". *Frontiers in Computational Neuroscience* 9.134 (2005) 2.

[MSRB17] MONTESDEOCA, S.E., SEAH, H.S., RALL, H.-M., and BENVENUTI, D. "Art-directed watercolor stylization of 3D animations in real-time". *Computers & Graphics* 65 (2017), 60–72 2.

[Mus19] MUSEUM OF FINE ARTS, BOSTON. *Winnie-the-Pooh: Exploring a Classic*. mfa.org/exhibitions/winnie-the-pooh. 2019 1.

[Nav12] NAVA, MATTHEW. *Art of Journey*. Bluecanvas, Inc., 2012 2, 5.

[NSTN93] NISHITA, T., SIRAI, T., TADAMURA, K., and NAKAMAE, E. "Display of the Earth Taking into Account Atmospheric Scattering". *SIGGRAPH '93*. 1993 3.

[ONe04] O'NEIL, S. *Real-Time Atmospheric Scattering*. www.gamedev.net/articles/programming/graphics/real-time-atmospheric-scattering-r2093. 2004 3.

[ONe05] O'NEIL, S. "Accurate Atmospheric Scattering". *GPU Gems 2: Programming Techniques for High-performance Graphics and General-purpose Computation*. 2005. Chap. 16, 253–267 3–5.

[Per02] PERLIN, K. "Improving Noise". *SIGGRAPH '02*. 2002 3, 6.

[Per85] PERLIN, K. "An Image Synthesizer". *SIGGRAPH '85*. 1985 3, 6.

[RP05] RODEN, T. and PARBERRY, I. "Clouds and Stars: Efficient Real-time Procedural Sky Rendering Using 3D Hardware". *ACM SIGCHI ACE '05*. 2005 3.

[SFE07] SCHAFHITZEL, T., FALK, M., and ERTL, T. "Real-time rendering of planets with atmospheres". *J. of WSCG* 15 (Jan. 2007), 91–98 3.

[Val15] VALVE DEV. COMMUNITY. *Skybox Basics*. developer.valvesoftware.com/wiki/Skybox. Aug. 2015 2, 3.

[Zah16] ZAHED, R. *The Little Prince: The Art of the Movie*. 2016 2.