FOLAR: A FOggy-LAser Rendering Method for Interaction in Virtual Reality

FOLAR: En FOggy-LAser Rendering Metod för Interaktion i Virtual Reality

TIANLI ZHANG
Sammanfattning

FOLAR: A FOggy-LAser Rendering Method for Interaction in Virtual Reality

Tianli Zhang

ABSTRACT

Current commercial Virtual Reality (VR) headsets give viewers immersion in virtual space with stereoscopic graphics and positional tracking. Developers can create VR applications in a working pipeline similar to creating 3D games using game engines. However, the characteristics of VR headsets give disadvantages to the rendering technique particle system with billboard sprites. In our study, we propose a rendering technique called FOggy-LAser Rendering method (FOLAR), which renders realistic laser in fog on billboard sprites. With this method, we can compensate for the disadvantages of using particle systems and still render the graphics in interactive performance for VR. We studied the characteristics of this method by performance benchmarks and comparing the rendered result to a baseline ray-casting method. User study and image similarity metrics are involved in the comparison study. As a result, we observed a satisfying performance and a similar rendering result compared to ray-casting. However, the user study still shows a significant difference in the rendered result between methods. These results imply that FOLAR is an acceptable method for its performance and correctness in the rendered result, but still have inevitable trade-offs in the graphics.

KEYWORDS

Computer Graphics, Volumetric Rendering, Virtual Reality, Particle System, Participating Media, Illumination, Real-time Rendering

1 INTRODUCTION

1.1 Motivation

Virtual Reality (VR) technology has been highly commercialized during recent years, while enormous numbers of applications are created for commercial VR applications. Current commercial VR headset devices give the viewers practical immersion mainly by stereoscopic graphics and positional tracking. Application creators for VR could benefit from the experience of creating 3D computer games by using game engines such as Unity or Unreal with minor changes to the content of the application. However, this is not always true for all cases.

In our case, a well-known method called “particle fog” presents significant disadvantages in displaying the same experience in VR. It produces the undesirable effect that the fog is “rotating” or “flipping” with the motion of the viewer’s head. This undesirable effect is due to the use of billboard sprites. Billboard sprites are 2D surfaces that follow the user camera in the 3D space, so the user always sees the front face of the surface. This effect interferes with immersion in stereoscopic rendering and positional tracking, both features of VR head-mounted headsets.

To compensate for this problem, we created a technique called FOggy-LAser Rendering method (FOLAR). FOLAR takes advantage of particle system rendering methods and displays a more realistic fog illuminated by laser beams in VR. It approximates the rendered result from ray-casting. The method makes opportunistic uses of the way to construct an icosahedron [31] to model the volumetric features of a spherical group of fog volume to avoid ray-casting in real-time rendering, as well as light participating models based on the distance of a ray to the laser and the position on the ray. As a result of relying on particle systems, FOLAR still allows displaying the fog in an interactive performance for VR applications, and also allows the fog to be illuminated by laser beams with a convincing effect in VR that is similar to the result from ray-casting.

This study aims to introduce FOLAR as a novel method in creating a particle-system-based laser-illuminated fog effect. In the study, we determine how FOLAR could compensate for the disadvantages of particle fog and to measure the difference in visual characteristics of FOLAR and a simple volumetric ray-marching method described in section 3.5.
1.2 Research Question
Given our new real-time rendering method for laser-illuminated fog in VR named FOLAR, what are the main visual characteristics of the algorithm that allow people to perceive a realistic looking laser cast into the fog when compared to volumetric ray-casting as comparatively analyzed by running performance metrics and by image similarity metrics, both computed or perceptually-based?

2 RELATED WORK

2.1 Virtual Reality
Virtual Reality (VR) has been a well-known technique in immersive visualization and interaction in educational, industrial, and commercial usages. In recent years, commercial head-mounted VR display devices (or VR headsets) such as the HTC Vive [20] and the Oculus Rift [17] has been well accepted in the current market with numerous commercial or educational software applications developed for those platforms. For those VR headsets, content creators merely care for the VR displaying technology but consider more in the design of the contents to fit the methodology that VR headsets present the contents to the viewer.

Current commercial VR headsets give the viewers practical immersion by combining multiple methods in giving a human a sense of the virtual world, such as stereoscopic graphics and positional tracking. The sense of immersion from those two methods are identified as binocular disparities and motion perspective in early research in 1997 [8]. In order to achieve this, modern computers use a deviation of projection-rasterization-based graphics pipelines called stereo multi-projection [13]. This technique applies projection and rasterization twice to the rendered scene with different camera parameters that match the physical position of the user’s head measured by positional tracking sensors.

As a result of VR being well-accepted, game engines such as Unity [2] and Unreal Engine [10] often embed native support in the creation of VR content. They provide content creators a similar working pipeline as creating ordinary 3D games, as seen in [18] and [15], which is reasonably a benefit from using stereo multi-projection graphics pipeline. This working pipeline allows application creators to use existing materials and experience in creating 3D computer games for creating VR applications.

Despite all the advantages of current commercial VR headsets, those display systems and game engines are not immediate solutions for application creators. Inappropriate content or misuse of the game engine may lead to the lack of presence of viewers or even worse, such as the unpleasant experience from motion sickness [27]. In order to overcome this problem, there have been researches regarding presence [26] and user experience [28] in VR, as well as measurements of motion sickness caused by specific methods in VR contents [19].

2.2 Particle System

2.2.1 Definitions and Usages of Particle Systems. Particle system is a well-described technique in computer graphics that essentially makes use of a group of individually animated graphic objects, namely particles, to “render a ‘fuzzy’ object or view of a volume”, as introduced by William T. Reeves [25]. In most usages of a particle system, often billboard sprites are utilized as the particles, under the assumption that each particle has the same appearance from any perspective of view.

Particle systems are well-known in simulating a volume of participating media in real-time applications, such as clouds [14], fogs [1], smokes [4], and explosion effects [30]. Sometimes those billboard sprites are rendered with noised textures instead of rotational-symmetric textures, such as for particle fog. The noise texture introduces more stochastic visual effects for the fog, which significantly reduced the number of particles to be used in the rendering process to achieve a convincing fog effect, thus increasing the performance of rendering, as mentioned in [1]. Despite breaking the assumption of the same appearance from any perspective of view, this technique usually generates well-looking and convincing results on flat-screen displays. As a result, many commercial applications, such as 3D games, adapted this method to create a high-performance stochastic fog effect.

2.2.2 Particle System in VR. Opposed to applications made for flat displays, the use of stereo multi-projection [13] in VR headset displays denies the advantages of billboard images in VR, making particle systems a less potent tool in creating immersive contents for VR. Since stereo multi-projection uses the same vertices of the scene to render the image for both eyes, viewers perceive billboard sprites as ordinary flat objects in the virtual space. This difference makes billboard images impossible to provide the same appearance for any perspective of view. Application creators often avoid the use of particle system at close distances to the viewer in virtual space, or intentionally design the particles to be actual flat objects.

2.3 Ray-casting and Volumetric Rendering
Ray-casting (or ray-tracing) is a generic name of a set of well-accepted methods in computer graphics to render realistic images by imitating the physics of light rays in the real world, which sacrifices the rendering time for a better result. According to the pinhole camera model, the 3D scene is projected onto a plane in front of the camera. The projection process maps each sample on the image to a corresponding direction towards the camera. Ray-casting renders those samples by calculating the color of light on the ray that shoots towards the camera by that corresponding direction. Those rays are called primary rays, and any rays that influence the primary rays indirectly, such as reflection, refraction, and scattering of light, are called secondary rays.

With ray-casting, we can achieve volumetric rendering by calculating the amount of radiance that comes through the volume of the participating media. The factors of the participating media’s influence on the amount of radiance are categorized into three kinds: absorption, emission, and scattering [23]. Different from absorption and emission, light scattering inside a volume of participating media is a non-trivial problem that has been broadly researched and discussed in computer graphics. Studies have proposed advanced methods such as stochastic simulation of light participation [33] and direct rendering of the volume with modeled data [32]; the principle under volumetric rendering has also been well-discussed [29]. A simple method to achieve volumetric ray-casting is ray-marching,
which approximates the continuous changes in the amount of radiance along the ray by taking small steps on the ray and calculates on each of the sample points between steps [29].

2.3.1 Ray-casting in VR. Because of the power of ray-casting in rendering realistic images, by appropriately adapt it into VR application, we can produce convincing visuals and fascinating graphics for VR users. One obvious issue with this, though, is the rendering performance. By longer rendering time, it means to delay the visual further behind the desired image corresponding to the physical viewing position.

There have been efforts to make ray-casting more efficient to be acceptable at an interactive performance for specific cases. In the case of cloud rendering, some study proposed screen-space scattering as a part of an efficient physically-based rendering method [9]. There is also a method that models the cloud into volumes of spheroids that allows the calculation to be done separately on CPU and GPU [16]. In the second example, they simulated the light scattering between spheroids on CPU, then sorted the spheroids according to the distance to the viewing position, and finally calculated the amount of radiance on rays by applying the changes in the amount from the spheroids on GPU in the sorted order. Furthermore, the thought of rendering volumetric features of the fog onto billboard sprites are already proposed in [5]. The method described in the article shares a similar methodology with FOLAR, where both methods use fragment shader that renders the fog onto billboard sprites in order to achieve real-time performance.

3 MATERIAL AND METHODS
In this section, we will describe all the steps and materials for FOLAR and the parameters that we used for the study, as well as the ray-casting method we used as a baseline result in comparisons with FOLAR.

3.1 Delimitation and Assumptions in FOLAR
For Virtual Reality applications, we have introduced some delimitation and made several assumptions in the design of FOLAR in order to achieve interactive performance.

In order to make the best use out of the hardware, we designed FOLAR based on one usage of particle systems named particle fog, as described in [1]. With particle systems, we render large dynamic textured transparent billboard sprites that always face the camera or the position of the viewer’s head in virtual space. This methodology gives FOLAR several limitations in fog dynamics. There will be no simulation on how air flows effects the texture of the fog; the fog can be animated only by moving in large groups and with animated textures. Those limitations are the characteristics of particle fog addressed in [16].

In order to convince the viewers that the fog is volumetric, we designed FOLAR as an approximation of ray-casting methods. We implemented FOLAR mainly as a fragment shader applied to the particles, with each particle being rendered as the appearance of the corresponding volume. As we display the rendered result on individual particles, we may expect lousy approximation when the volumes of the particles collide, but in terms of visual correctness in variable viewing angle, this should still be a considerable approximation. This method can also be seen in [16] as well, and this issue is also mentioned in their works.

In aspects of the illumination process and ray-related approximations, in order to achieve real-time performance, we assume that the fog has the following properties, which applies to both FOLAR and the ray-casting method:

1. The fog scatters all of the radiance that it absorbs;
2. Light absorbed at any position in the fog is scattered evenly among all directions;
3. For light from the back of any particle, only out-scattering will be considered;
4. The fog is overall evenly distributed, so that in some cases by using an average value for the whole fog is most likely to be considered a valid approximation, such as using the same attenuation cross-section over the whole volume of the fog for the simulation of light scattering;
5. All light scattering or attenuation calculations are limited to Mie Scattering and Beer-Lambert law so that the wavelength of light does not have any effect in the calculations;
6. The fog is evenly illuminated by ambient lights, such as sunlight.

3.2 Useful Mathematical Functions
Throughout the whole description of FOLAR, we may make use of some mathematical functions frequently for simplicity and their frequent use in computer graphics. Preferably, we keep our definitions as close as possible to the definitions from the reference manual of the shader language Cg [7] from NVIDIA Corporation.

3.2.1 Clamping Function. Clamping function is a helper function that limits a scalar variable to a certain range \([a, b]\), written as:

\[
\text{clamp}(x, a, b) = \max(a, \min(b, x))
\] (1)

Note that by definition the result of clamping is \(a\) if \(a > b\), but we will avoid this case in any usages by all means.

3.2.2 Linear Interpolation Function. Linear interpolation function is a fast and useful way to find a linear transition between two values. For two variables \(a\) and \(b\), both scalars or vectors of the same dimension, and a scalar value \(f\), we define the linear interpolation function as

\[
\text{lerp}(a, b, f) = (1 - f)a + fb
\] (2)

Notice that when \(f\) equals 0 or 1, the result of the function is exact \(a\) or \(b\) even with floating point errors considered, based on how modern computers implements multiplication; otherwise, when \(f \not\in [0, 1]\), this function actually behaves as a linear extrapolation function.

3.2.3 Two-argument Arctangent. This function is useful to convert Cartesian coordinates \((x, y)\) to polar coordinates \((r, \theta)\) of the same point by definition:

\[
\theta = \text{atan2}(y, x)
\] (3)

noticing that \(y\) needs to be the first argument of \text{atan2}.
3.3 Pre-defined and Pre-rendered Models

Before real-time rendering, we prepared several materials under our ad-hoc definitions to allow real-time rendering at the expected performance. The materials allow us to avoid the use of ray-marching or similar algorithms that depends on repeated iteration along every ray, which could cause potential drops in rendering performance.

3.3.1 Fog Spheroid and Fog Density with Noise. We define a fog spheroid (or simply spheroid) as a volume of fog with noise textures and the approximated rendering of it based on its volumetric features. We will build our fog volume with spheroids and render each spheroid onto a billboard sprite as a particle in the particle system. The spheroid is not defined for real-time simulated fog or smoke, nor do we introduce any dynamics for the spheroid. However, the dynamics of fog may be achieved by using animated textures and the particle system itself.

For simplicity, we assume that the amount of light attenuated on a ray through its volume is scattered by a fixed proportion and distributed evenly among all directions. The rate of light attenuation and the amount of light scattered to all directions on any position are determined by the attenuation cross-section and the albedo. In our case, we set albedo as a constant value and let a fog density function serve as the normalized attenuation cross-section of the fog spheroid.

The density value is defined as a function Density(\( \mathbf{v} \)) of position \( \mathbf{v} \) with the origin at the center of the fog particle, which is further subdivided as the product of two functions, Texture(\( \mathbf{v} \)) and Distrib(\( \mathbf{v} \)), i.e.: 

\[
\text{Density}(\mathbf{v}) = \text{Texture}(\mathbf{v}) \cdot \text{Distrib}(\mathbf{v})
\]  

(4)

where Texture(\( \mathbf{v} \)) is a function to introduce a random texture to the distribution of density with the expectation value of the function \( E[\text{Texture}] = 1 \), and Distrib(\( \mathbf{v} \)) is a function that scales the noise texture Texture(\( \mathbf{v} \)) with a quadratic distributed value in order to give the final result of Density(\( \mathbf{v} \)) a smooth transition in the values from the center to the edge of the spheroid.

The main usage of Density(\( \mathbf{v} \)) is to find the attenuation cross-section at certain position, defined as:

\[
\sigma(\mathbf{v}) = \hat{\sigma} \ast \text{Density}(\mathbf{v})
\]  

(5)

where \( \hat{\sigma} \) is a constant that marks the average attenuation cross-section of the fog that is desired in real applications, referred as fog density constant or fog density.

**Noise Texture from Fractional Brownian Motion and Perlin Noise.** For the noise texture used for natural random distribution of density, we adapt a fractional Brownian motion (fBm) method described by Inigo Quilez [24] on gradient noises such as Perlin noise [22] and Simplex noise [21]. This method is introduced in a technique to generate random texture for cloud rendering [16] to introduce the roughness detail in noise because of its non-differentiable property in general.

Firstly, we define a gradient noise function Noise(\( \mathbf{v} \)) as a function of a 3-dimensional position \( \mathbf{v} \) with the expectation of the noise \( E[\text{Noise}] = 0 \); in concern of the directional artifacts that Perlin noise introduces as described by Gustavson [12], we choose to use Simplex noise as the gradient noise function and adapted Kenjiro Takahashi’s implementation in Cg adapted from Ashima’s implementation for this study.

Secondly, we adapted Inigo Quilez’s fBm method [24] in generating the a intermediate value noise with suggested parameters, i.e.: 

\[
\text{fBmNoise}(\mathbf{v}) = \sum_{i=1}^{n} w^i \text{Noise}(s^{-i} \mathbf{v}), \quad w = 0.5, \quad s = 2, \quad n = 5
\]  

(6)

knowing that the weighted summary of Noise(\( \mathbf{v} \)) still give an expectation value of \( E[\text{fBmNoise}] = 0 \).

Finally, with \( \text{fBmNoise}(\mathbf{v}) \), parameter \( a \) to control the scale of the randomness, and parameter \( s_0 \) and \( \mathbf{v}_0 \) as the scale and offset of positions in noise generation, we can produce a decent noise texture with the following definition:

\[
\text{Texture}(\mathbf{v}) = \text{clamp}(a \cdot \text{fBmNoise}(s_0 \mathbf{v} + \mathbf{v}_0) + 1, 0, 2)
\]  

(7)

in which we added a scale factor \( a \) to control the scale and range of the values in the noise texture while we clamped the value in [0, 2] to avoid negative values. For normal cases, \( a = 1 \) could generate a strong noise and \( a = 0 \) completely removes the noise texture from the rendering process; for our study, we used a fixed value of \( a = 1 \) for simplicity.

**Quadratic Distribution of Expected Density.** To specify the range of fog for one particle, also to provide a soften effect to individual particle, i.e. gives the particle a transition on fog density from the center to the edge, we scale the values from the random texture Texture(\( \mathbf{v} \)) generated in the last section by a certain amount Distrib(\( \mathbf{v} \)).

In need of calculating light attenuation for the later illumination steps, as shown in figure 2, we chose the positive part of a vector-form centered quadratic function as Distrib(\( \mathbf{v} \)) with an apparent spherical range:

\[
\text{Distrib}(\mathbf{v}) = \text{intensity} \cdot \max(1 - ||\mathbf{v}||_{\text{range}}^2, 0)
\]  

(8)

By giving the parameter range, we are able to scale the visual size of the particle and the scope of Distrib(\( \mathbf{v} \)) as Distrib(\( \mathbf{v} \)) \( = 0 \) when \( ||\mathbf{v}||_{\text{range}} \geq \text{range} \); with parameter intensity we simply scale the result value up at will without changing the visual size of the particle.

The reason for choosing a quadratic function is that the value of the function on a line is still a quadratic function of linear position on that line, as visualized in figure 2.

3.3.2 Icosahedron Interpolated Ray-mapped Texture. To eliminate the process of iterated calculations in real-time rendering, we created a unique method to model the volumetric features of a fog spheroid to allow us to find intermediate values to get the final result of the volumetric rendering trivially.

Since a fog spheroid is intuitively a spherical volume of participating media, it has a certain degree of symmetry or similarity over rotations to the center of the spherical volume. This character allows us to use a ray-positioning method similar to the spherical coordinate system, which specifies rays relative to the center of the


\[\text{https://github.com/ashima/webgl-noise, retrieved June 18, 2019}\]
fog spheroid. One ray could be determined by its direction vector and components of the position of the origin that is perpendicular to the direction vector, since adding any ratio of the direction vector to the origin point leads to the same ray as before.

In aware of this, we designed the method called Icosahedron Interpolated Ray-mapped Texture. The model maps certain sets of parallel rays (which are called parallel-groups) with the same direction vector onto the coordinates on textures as ray-mapped textures. Specifically, 12 groups of parallel rays are paired with the 12 textures, with the direction vectors of each group of parallel rays determined by vectors of the 12 vertices of a regular icosahedron.

Under this model of positioning of rays, we are able to store the volumetric information of a fog spheroid onto 2D textures, with each sample on the texture mapping to a specific ray. In real-time rendering, the volumetric information is extracted by interpolating among samples mapped to rays close to the ray in the calculation. Determined by the icosahedron, the modeled rays only have 12 possible directions; in our design, only 3 of those are "close to" any calculated ray. This is due to our specific icosahedron construction. With a weighted average of three samples from the textures depending on differences between the directions, we can successfully extract the volumetric information from the model.

In this section, we will specify the mathematical details of this model and the volumetric information that we stored in the model for real-time rendering in FOLAR. The real-time usage of the model will be described later in section 3.4.2 along with the method to extract the volumetric information from the model.

Parallel-groups of Parallel Rays. Since the amount of attenuation is a result of continuous out-scattering along the ray direction towards the camera, the values of attenuation are mapped to rays, which contain both positional and directional information, rather than positions or directions only.

For our usage, we determine the ray with an origin position and the direction of the ray. As shown in the figure below, we determine ray $ray_{n, v}$ by a direction vector $n$ and the positional offset $v$ that is perpendicular to $n$, which is also the distance from the particle center $O$ to the ray; for any point $P$ on ray $ray_{n, v}$, $OP = v + \lambda n$ for some real number $\lambda$.

Figure 2: The value of the quadratic function $f$ (in red) on 2D line $L$ (in blue) is a quadratic function $g$ (in green) of linear position on that line.

As shown in figure 4, we specify a set of special rays by grouping the rays into multiple parallel-groups, that rays in the same group are towards the same direction; for a group $k$, we set the common direction vector to be $n_k$. To find the positional offset $v$ of any ray in the group, we specify an up vector $i_k$ that is perpendicular to $n_k$; with the up vector $i_k$ and a right vector $j_k = n_k \times i_k$, any positional offset $v$ can be represented with a combination of scaled up vector and right vector, i.e. $v_{x,y,k} = (x,y)_k = u i_k + v j_k$. Notice that $n_k$, $i_k$, and $j_k$ are all normalized vectors.

Ray-mapped Texture. With the parallel-groups defined in the previous sections, we will sample useful values for each ray in a manner that respects the parallel-groups and store the values onto 2D textures, while the rays to be sampled is chosen according to the texture coordinates of the color samples on the texture.

The values of every ray in each parallel-group are mapped onto one single ray-mapped texture. Given the normal vector $n$ of the parallel-group, the value of any ray $val_{ray_{n, v}}$ is mapped to a scaled
position on the texture with the coordinate of $k\mathbf{v} + \mathbf{v}_0$, where $k$ is an arbitrary number that allows all useful values to be fit into the texture at any 2D coordinate system for the texture (e.g. UV coordinates), and $\mathbf{v}_0$ is the coordinate of the center of the texture.

**Light Attenuation Values on Rays.** For every ray mapped onto a sample on the ray-mapped texture, we specifically calculate two values for each ray and encode them into 32-bit floating-point samples that is stored on 2D texture images, sharing the same method for interpolations between samples (e.g. bi-linear interpolation) when looking for inter-sample values.

The 3 integrated values represents the optical thickness along the ray path through one fog spheroid. According to the light out-scattering formula [23], the total transmittance of light through the fog spheroid (despite of the presence of in-scattering) are calculated given assumption (1) that light is absorbed at the same rate among all directions:

$$T_{ray} = e^{-\int_{-\infty}^{\infty} \sigma_{ray}(t) dt} = e^{-\int_{ray} \sigma \text{Density}(\mathbf{v}) ds}$$

(9)

By taking the negative of the natural log value of the transmittance value and setting the fog density constant $\sigma$ to 1, we find our first value for the samples, which is treated as a normalized (in terms of setting $\sigma = 1$) optical thickness value:

$$\tilde{t}_{ray} = -\ln T_{ray} = \int_{ray} \text{Density}(\mathbf{v}) ds$$

(10)

Since the function Density $\mathbf{v}$ are calculated from fBm noise with limited iteration depth, which is in fact still a fractal noise, the integral can be calculated in a trivial fix-step summation method with the step small enough; we used a step size of 0.02 in our implementation.

After that, we could add an additional value, the radius or the half length of the ray path inside the spheroid, into the sample just for convenience, despite that it can also be calculated real-time. The radius is represented in a normalized way by setting the radius of the spheroid to 1:

$$r_{ray,\mathbf{v}} = \sqrt{2} r_{\text{spheroid}} - ||\mathbf{v}||^2 = \sqrt{1 - ||\mathbf{v}||^2}$$

(11)

Finally, we put the two values together in a vector form,

$$S = (\tilde{t}_{ray}, r_{ray})$$

(12)

encoded as 32-bit floating-point samples that could be easily stored in popular image formats (for our case, we used OpenEXR format [6] since we need to store values that is beyond range [0, 1]).

For rays with no collision with the volume of the modeled spheroid, the calculation of all four values are skipped and zeros are assigned to the sample.

**Ray Orientations Based on Specific Icosahedron Construction.** To specify the direction vector of each parallel-group, we constructed a regular icosahedron in a specific way with its center at $(0, 0, 0)$ and the distance from the center to each vertex equals 1, and set the direction vectors of each group to be matching the vector from the center to each vertex. In this way, the number of parallel-groups is fixed 12.

We adopted a unique method to construct the icosahedron, as indicated by Andrew [31]. As shown in figure 5, all the vertices of the icosahedron are on axis-aligned planes, forming three identical rectangles perpendicular to each other. We specifically call this method **3-rect icosahedron construction**. The orientation of icosahedrons constructed in this way allows us to use a simple way to sample the values from the Icosahedron Interpolated Ray-mapped Texture, described in the next section. The exact coordinates of the vertices are listed in table 1, mapped with the normal vectors and up vectors of each parallel-group.

With normal vectors and up vectors defined, we prepared a set of 12 ray-mapped textures corresponding to the 12 parallel-groups. By assigning an arbitrary ratio of the size of the texture, the size of the modeled fog spheroid (in our cases, the spheroid is set 80% wide as the square-sized texture), and a useful resolution for the texture ($512 \times 512$ in our cases), the positions of all rays can be as well determined.

**Facet Tracing for Real-time Rendering.** In real-time rendering, we use a specific interpolation method to find approximated values for an arbitrary ray through a fog spheroid, which will be described in later sections. In the interpolation, only values from three of the ray-mapped textures will be sampled and averaged with weights. The three ray-mapped textures are selected according to the direction of the ray: we shoot a ray with the same direction from the center of the icosahedron to the surface of it, knowing that it will hit one of the facets of the icosahedron, then we decide on the three ray-mapped textures corresponding to the three vertices of the facet the ray hits by looking up the indices of the vertices. We especially name this process **Facet Tracing** in FOLAR.

As a benefit of our specific construction of the icosahedron, we are able to find the facet by judging a few conditions and look up a predetermined table with the conditions, namely the facet table.

Since the icosahedron shown in 5 is symmetrical over all three axis-aligned planes, we first decide which octant contains the ray. In order to refer to each octant, we name the octants with + or - marks after the three axes showing that if the octant is positive or negative for each axis, e.g. **X+Y-Z-**.

After that, in any of the octants, there are explicitly 4 facets that is possible for the ray to hit. As shown in the figure 6 for octant **X+Y-Z+**, there are three right triangles, i.e. $\Delta AA'C$, $\Delta BB'A$, and $\Delta CC'B$, around a equilateral triangle, i.e. $\Delta ABC$, with a rotational symmetric layout. By connecting $AO$, $BO$, and $CO$, we get three distinct volumes that is separated by the three triangles $\Delta ABO$, $\Delta BCO$, and $\Delta ACO$, so that we can simply determine which facet a ray from the origin $O$ hits by looking at the dot product between the direction of the ray and the normal vector of the three triangles. Specifically, we set the normal vectors to:

$$n_x = (-1.618, 2.618, 1),$$

$$n_y = (1, -1.618, 2.618),$$

$$n_z = (2.618, 1, -1.618)$$

(13) (14) (15)

and if the dot product between the direction of the ray $d$ and either of the normal vectors above is negative (if there are multiple negatives by accident, choose either one), we mark the conditions as $X-$, $Y-$, or $Z-$; otherwise $N$ for all positives.

Given the conditions of octants and directions, the facet table can be represented below as a look-up table with pre-assigned numbers for each facet of the icosahedron:
Given the index of the facet assigned, the index of the vertices can be listed in another table, namely the vertex table. In our case, the indices have to be put in a particular order such that the same index in different rows (for different facets) is seen at the same column in the rows to avoid artifacts in the rendered result, as seen in figure 7. This issue might be due to inconsistent behaviors when neighboring fragments are using different facet to sample the model as a result of using integer indices on GPU. We are not sure how this comes to be an issue on our hardware, nor does it serve as the primary purpose for this study, but given our condition of the hardware, we insist on this specific way of specifying the vertex table. As a result of this, each row has to contain at least four columns, leaving one “null” indices for each row. The exact table we used is shown below:

### 3.3.3 Light Scattering Models

In this section, we prepared two models to allow us to approximate the amount of light that the laser contributes to the surround fog volume, in order to sufficiently remove any of the iterated illumination steps in real-time. As assumed in section 3.1, we treat the fog density, which is proportional to the attenuation cross-section, to be the same as a whole by ignoring the noise texture of the fog. In consequence, we model the values as a function of the distance to the laser and the attenuation cross-section.

In the first section, we model the amount of radiance that any position receives in an infinite volume of homogeneous fog illuminated by the laser as *Light Scattering Model*. The calculation is
done by iterating the calculation on the total amount of radiance from the surrounding volume received on every sampled position. The initial condition of the iteration is a one-step illumination from the laser, with the amount of radiance on the laser to any direction to be exactly 1. Since the amount of radiance on any position and direction is in proportion to the result of the calculation from this initial condition, we can control the lightness of the laser in the rendered result by scaling the illumination value contributed by the laser.

Figure 6: The 4 facets that is clipped into one octant, with the three normal vectors used to determine which facet the ray from the center of the icosahedron hits.

Table 3: The vertex table

<table>
<thead>
<tr>
<th>Facet Index</th>
<th>Vertex Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4 8 - 0</td>
</tr>
<tr>
<td>1</td>
<td>4 - 2 9</td>
</tr>
<tr>
<td>2</td>
<td>1 8 6 -</td>
</tr>
<tr>
<td>3</td>
<td>3 - 6 9</td>
</tr>
<tr>
<td>4</td>
<td>- 5 10 0</td>
</tr>
<tr>
<td>5</td>
<td>- 5 2 11</td>
</tr>
<tr>
<td>6</td>
<td>1 7 10 -</td>
</tr>
<tr>
<td>7</td>
<td>3 7 - 11</td>
</tr>
<tr>
<td>8</td>
<td>1 8 - 0</td>
</tr>
<tr>
<td>9</td>
<td>3 - 2 9</td>
</tr>
<tr>
<td>10</td>
<td>1 - 10 0</td>
</tr>
<tr>
<td>11</td>
<td>3 - 2 11</td>
</tr>
<tr>
<td>12</td>
<td>4 5 - 0</td>
</tr>
<tr>
<td>13</td>
<td>4 5 2 -</td>
</tr>
<tr>
<td>14</td>
<td>1 7 6 -</td>
</tr>
<tr>
<td>15</td>
<td>3 7 6 -</td>
</tr>
<tr>
<td>16</td>
<td>4 8 - 9</td>
</tr>
<tr>
<td>17</td>
<td>- 8 6 9</td>
</tr>
<tr>
<td>18</td>
<td>- 5 10 11</td>
</tr>
<tr>
<td>19</td>
<td>- 7 10 11</td>
</tr>
</tbody>
</table>

Figure 7: Line-shaped artifacts in the rendered result when facet indices are assigned in random orders in the rows of the vertex table.

In the second section, we integrate the values from the first section along rays that is perpendicular to the laser as Integrated Light Scattering Model. As a result, we can avoid iterated calculations in real-time rendering such as ray-marching, but instead find the desired value from this integrated model.

Light Scattering Model. Given the amount of received radiance \( L_\sigma(x) \) as a function of the distance to the laser \( x \) with the parameter of attenuation cross-section \( \bar{\sigma} \) and a constant albedo \( \rho \), the iterated calculation can be expressed with a recursive formula of additional radiance received from all surrounding volume \( L_\sigma^{(\eta)}(x) \) in the \( \eta \)-th iteration, which is a summation of radiance over the volume by dividing the volume into \( N \times N \times M \) unit cubic cells of width \( l \):

\[
L_\sigma^{(\eta+1)}(x) = \sum_{i=-N}^{N} \sum_{j=-N}^{N} \sum_{k=-M}^{M} \rho L_\sigma^{(\eta)}(X_1) \cdot \frac{e^{-\bar{\sigma}X_2}}{4\pi X_2^2},
\]

\[
X_1 = ||l \cdot (i,j,0)||, X_2 = ||l \cdot (i,j,k) - (x,0,0)||, \eta \geq 0 \quad (16)
\]

with its initial condition as a summation of radiance from the laser:

\[
L_\sigma^{(0)}(x) = \sum_{k=-M}^{M} L_{laser} \cdot \frac{e^{-\bar{\sigma}X}}{4\pi X^2}, X = ||(x,0,kl)||, L_{laser} = 1 \quad (17)
\]

The final result of \( L_\sigma(x) \) is a summation of all intermediate results of the iterated calculation:

\[
L_\sigma(x) = \sum_{\eta=0}^{H} L_\sigma^{(\eta)}(x) \quad (18)
\]

In our implementation, we stored the values onto a 2D texture with each pixel on normalized coordinate \( (u,v) \) mapped to certain values of \( x \) and \( \bar{\sigma} \), that is:

\[
x = X(u) = e^{20u - 10} \quad (19)
\]

\[
\bar{\sigma} = D(v) = 10^4v^{-2} \quad (20)
\]

which allows us to model the illumination values on fog densities from 0.01 to 100 and a maximum distance of \( e^{10} \) to the laser with...
dense samples near the laser and sparse samples far from the laser. In between of the samples, the value of $L_\lambda(x)$ is found by linear interpolation on the texture with corresponding $u$ and $v$ with clamping to range $[0, 1]$; in our study, the average value of $\bar{\sigma}$ is always in $[0.01, 100]$ and the value of $L_\lambda(x)$ when $\sigma > \epsilon^0$ in our model is very close to 0 so that it does not cause noticeable differences in visuals by clamping.

For our study, we stored $L_\lambda(x)$ and the intermediate result $L^{(\eta)}_\lambda(x)$ in 32-bit floats on a 2D texture of resolution 256 × 256. The parameters that we used for the study are: $H = 10$, $\rho = 1$, $N = 30$, $M = 1$, and $l$ as a function of $v$ as $l = 0.5e^{-4v^2}/256$, which gives us decent results that we could conduct our study with.

**Integrated Light Scattering Model.** As described earlier, we prepare the integrated light scattering model by integrating $L_\lambda(x)$ over rays perpendicular to the laser with a distance of $x$ in between; with this model we can approximate the amount of radiance on any ray gained in any range when the ray is not perpendicular to the laser.

![Figure 8: The argument $\lambda$ is defined as the offset of some position $A$ on the ray to the position on the ray with minimum distance to the laser $O$ on the direction of the ray, so that $OA = \lambda d_{ray}$ for the case in the figure.](image)

By adding a new argument $\lambda$ representing the amount of offset on the ray as shown in figure 8, the integrated light scattering model are defined as function:

$$L_{\sigma, x}(\lambda) = \int_0^{\lambda} L_{\sigma}((x, \lambda))e^{-\bar{\sigma}d\lambda}$$

(21)

so that the given fog density constant $\bar{\sigma}$ as the attenuation cross-section in average and a ray perpendicular to the laser with distance $x$ in between, the amount of radiance $L_{c,a\rightarrow b}$ on the position $\lambda = c$ gained from the range $\lambda \in [a, b]$ of the ray can be calculated by:

$$L_{\sigma, x}(a, b) \rightarrow c = L_{\sigma, x}(b) - L_{\sigma, x}(a)e^{-\bar{\sigma}c}$$

(22)

For rays with an arbitrary angle $\theta$ between its direction and the laser, the amount of radiance is approximated by scaling $a, b$, and the result by a trigonometric function value:

$$L_\theta((a, b) \rightarrow c) = \frac{L_{\sigma, x}(b \sin \theta) - L_{\sigma, x}(a \sin \theta)}{\sin \theta} e^{-\bar{\sigma}c}$$

(23)

In our implementation, we stored $L_{\sigma, x}(\lambda)$ in 32-bit floats on a 3D texture of resolution 256 × 256 × 256, mapping the parameters and the argument $\lambda$ the normalized coordinate $(u, v, w)$ as:

$$x = X(u), \bar{\sigma} = D(v), \lambda = X(w)$$

(24)

with definitions of the functions from equations 19 and 20; since the values of $L_{\sigma, x}(\lambda)$ may exponentially increase or decrease, we actually stored the logarithm value of $L$ in the texture.

The integration is approximated by a 10-step summation per sample on coordinate $w$, i.e.:

$$\tilde{L}_{\sigma, x}(X(n/256)) = \tilde{L}_{\sigma, x}(X((n - 1)/256)) + \sum_{i=0}^9 (L_{\sigma}((x, \lambda_i)) + L_{\sigma}((x, \lambda_{i+1}))) e^{-0.5\bar{\sigma}(\lambda_{i+1} - \lambda_i)} - \lambda_i = X(n + 0.1i)/256), n \in \mathbb{Z}, n > 0$$

(25)

with an approximated value for the initial condition:

$$\tilde{L}_{\sigma, x}(X(0)) = L_{\sigma}(x) \cdot X(0)$$

(26)

### 3.4 Real-time Rendering

With the materials prepared as described in the previous section, real-time rendering can be implemented with shader programming. We made use of the shader language Cg with additional functionality and rendering support from Unity to allow us to visualize the rendered result on flat display or in VR. The shader is applied on a ParticleSystem component in Unity with the billboard sprites facing the camera (or the virtual head of the viewer in VR).

The process of the real-time rendering can be shown in the task graph 9, with the dependency between tasks drawn as arrows. In our study, we implemented the whole process in Unity’s Vertex and Rendering.
**Fragment Shader** [3], with the grammar of Cg. We calculated the attenuation of light from the background first and the amount of radiance from the laser second, finally coloring the fog in desired ways.

In the following text, we will describe the implementation of the vertex shader in the next section, how to determine the attenuation of light of rays from the background in section 3.4.2, how to determine the amount of radiance that the laser contributes to each ray in section 3.4.3, and the way we visualized the light values to allow us viewing the result in section 3.4.4.

### 3.4.1 Vertex Shader

In the vertex shader, certain arguments needs to be calculated before being interpolated by the hardware and passed to the fragment shader, which contains but not limited to:

1. The size/scale of the spheroid rendered on the particle.
2. The projected coordinate of the fragment in screen space, in order to find the position of the ray hitting the background by sampling the depth texture; can be replaced with arguments for the same purpose.
3. The relative position of the perspective camera to the fragment; when normalized, it serves as viewing direction or ray direction.
4. The relative position of the perspective camera to the center of the spheroid.
5. The position of the fragment in world space.
6. The relative position of the fragment to the center of the spheroid in world space.

There are multiple ways to prepare the parameters in the vertex shader in Unity, and it would differ if this is implemented on different rendering platforms even if the shader is still programmed in Cg. Our implementation of the vertex shader function `vert()` and the definition of the input and output structure of the vertex shader are exactly as below, notice the numbers in the comments and the definition of the input and output structure of the vertex.

```cpp
struct appdata
{
    float4 vertex : POSITION;
    float4 center : TEXCOORD0; // Assign "Center(xyz)"
    // "Size(w)" to the custom vertex stream
    // of ParticleSystem
};

struct v2f
{
    float4 vertex : SV_POSITION;
    float4 clipPos : TEXCOORD1; // (1)
    float4 clipPos : TEXCOORD2; // (2)
    float3 viewDir : TEXCOORD3; // (3)
    float3 viewDir0 : TEXCOORD4; // (4)
    float3 worldPos : TEXCOORD5; // (5)
    float3 worldOffset : TEXCOORD6; // (6)
};

class v2f

v2f vert(appdata v)
{
    v2f o;
    o.vertex = UnityObjectToClipPos(v.vertex);
    o.color = v.color;
    o.size = v.center.w;
    o.clipPos = o.vertex;
    o.viewDir = WorldSpaceViewDir(v.vertex);
    o.viewDir0 = WorldSpaceViewDir(
        mul(UNITY_MATRIX_M, float4(v.center.xyz, 0)));
    o.worldPos = mul(UNITY_MATRIX_M, v.vertex);
    o.worldOffset = o.worldPos - v.center.xyz;
    return o;
}
```

### 3.4.2 Attenuation of Light from the Background

In this section, we will describe the method to find the optical thickness of the fog by taking samples from the Icosahedron Interpolated Ray-mapped Texture. This is done in three steps, firstly find the offset of the primary ray to the center of the spheroid, secondly do the facet tracing with the direction of the primary ray as specified in section 3.3.2 to determine which three ray-mapped textures do we take samples from, also find the UV coordinates of the samples, and finally interpolate between the samples from the three ray-mapped textures. The sample from the Icosahedron Interpolated Ray-mapped Texture are in form of $S = (\bar{t}, r)$, to be further used in section 3.4.3.

**Ray Offset to the Center of the Spheroid.** Before making use of our prepared models, we need to first calculate the offset of the primary ray to the center of the fog spheroid, which is defined as the shortest vector from the latter to the former. This offset represents the distance from the primary ray to the center of the fog spheroid, as well as showing the position on the ray nearest to the center.

Given argument (3) as $l$ and (4) as $l_0$, that is the relative position of the perspective camera to the fragment and to the center of the particle, the offset vector $o$ and the direction vector of the ray $d$ is found by:

$$o = \vec{l} - \vec{l_0} = \frac{l}{||l||} - \vec{l_0}, d = \frac{l}{||l||}$$

(27)

With the offset vector, we are able to find the UV coordinates to sample on certain ray-mapped textures in later steps.

**Facet Tracing and Texture Sampling.** In this section, we sample the values modeled in the ray-mapped textures given the direction vector of the primary ray $d$ and its offset to the center of the fog spheroid $o$, then interpolate between the samples from ray-mapped textures that are mapped to different parallel-groups.

Using the facet table and the vertex table specified in table 2 and 3 in section 3.3.2, we are able to find out three indices $k_i, i = 0, 1, 2$ as the choice of ray-mapped textures from which we will take samples, skipping the null indices for the mathematical representations (e.g. $k_i = 4, 2, 9$ for facet index 1). As described for Facet Tracing, the vertices of these indices on the icosahedron forms a triangular facet that a ray of direction $d$ from the center of the icosahedron intersects with the facet.

Known the indices, we will find three rays $r_{n_k}, \mathbf{v}_{k_i}$ in the parallel-groups of indices $k_i$ that is close enough to the primary ray that we are rendering. The positional offset $\mathbf{v}_k$ of $r_{n_k}, \mathbf{v}_{k_i}$ as shown in figure 8 is specified in polar coordinate $(r_k, \theta_k)$ for parallel-group $k_i$ (notice that $\mathbf{v}_k$ is defined as a 2D vector as shown in figure 8). In our implementation, the angle $\theta$ is set to be the angle between $\mathbf{u}_k$ and the up vector of the camera $\mathbf{u}_{\text{cam}}$ projected onto a plane with normal vector $d$ (the same as the direction vector of...
With the three positional offset vectors \( v \) weights depending on the distance between the direction weighted average of the three samples:

\[
\begin{align*}
    \mathbf{v}_i = (r_{k_i} \cos \theta_{k_i}, r_{k_i} \sin \theta_{k_i}), \\
    r_{k_i} = ||\mathbf{o}||/\text{size}, \theta_{k_i} = \arctan 2((\mathbf{u}_{ik} \times \mathbf{u}_{cam}) \cdot \mathbf{d}, \mathbf{u}_{ik} \cdot \mathbf{u}_{cam})
\end{align*}
\]  

(28)

With the three positional offset vectors \( \mathbf{v}_k \) we can easily take samples from the three ray-mapped textures with corresponding indices, as specified in the Ray-mapped Texture part of section 3.3.2.

Interpolation in Facet. After we found the three samples from the ray-mapped textures, namely \( S_i, i = 0, 1, 2 \), we need to interpolate between them to find a usable sample for fog coloring and illumination. As shown in figure 10, this interpolation is interpreted as a weighted average of the three samples:

\[
S = \frac{\sum_{i=0}^{2} w_i S_i}{\sum_{i=0}^{2} w_i}
\]  

(29)

The weight \( w_i \) is determined by how close the direction vector of the ray \( \mathbf{d} \) is to the three direction vectors of the parallel-groups \( \mathbf{n}_{k_i} \), assuming they are normalized; in our implementation, we simply determined the weight by a clamped linear function:

\[
w_i = \max\{1 - x_i/0.99, 0\}, x_i = ||\mathbf{d} - \mathbf{n}_{k_i}||
\]  

(30)

Since the weight is determined by distance, the approximation introduces “dead areas” when the direction vector of a ray is nearly parallel to the direction vector of one parallel-group (since the weights of any other parallel-groups are 0 because of the distance). The “dead areas” can be avoided by using triangle-based interpolations, however in forms of rendered results, this “dead areas” does not introduce any visible artifacts as the weights are continuous over the directions. It would generate a satisfactory result without more expensive calculations, such as cross-products.

3.4.3 Illumination Value from the Laser Beam. In this section, we will prepare a set of intermediate variables in a specific before sampling the integrated light scattering model, and then make use of the value sampled from the model.

To simplify the process, we need to do the calculations in laser space, i.e. a rotated and translated variant of the world space where the laser is precisely on Z-axis, with a direction vector of \((0, 0, 1)\), so that some calculations can be done by simply switching the vector components.

Find Range of Ray Inside Fog Spheroid. To find the radiance value of a range from the Integrated Light Scattering Model, we need to calculate the offset of the two ends of the range of the primary ray inside the fog spheroid, as specified in the definition of the Integrated Light Scattering Model in section 3.3.3. The range of the primary ray will be given in the form of offset value \( \lambda \) as described in figure 8.

Given the sample \( S_{ray} = (\hat{r}_{ray}, r_{ray}) \) from the Icosahedron Interpolated Ray-mapped Texture, we know that the range of the ray inside spheroid is in form of \( \lambda \in [\lambda_{min}, \lambda_{max}] = [c - \hat{r}, c + \hat{r}] \) with \( c \) as the offset from the nearest point \( N \) to the mid-point of the range of the ray inside spheroid \( M \) as seen in figure 11.

To find \( c \), we project the direction vector of the primary ray \( \mathbf{d} = (d_x, d_y, d_z) \) onto the X-Y plane as vector \( \mathbf{d}_{XY} \), and then find the ratio of \( \mathbf{OM} \) to \( \mathbf{d} \) by projecting \( \mathbf{d} \) and \( \mathbf{OM} \) onto \( \mathbf{d}_{XY} \) (any components parallel to the laser or \( N'N \) is removed by doing the projection), which is exactly \( c \):

\[
c = \frac{\mathbf{OM} \cdot \mathbf{d}_{XY}}{\mathbf{d} \cdot \mathbf{d}_{XY}} = (d_x, d_y, 0)
\]  

(31)

while \( c \) can also be used to find the position of \( N \) when given the position of \( M \) in world space:

\[
\mathbf{N} = \mathbf{M} - c \mathbf{d}
\]  

(32)

Considering the camera might be in the volume of the fog spheroid or the spheroid is partial inside the opaque background geometry, the actual range of the ray inside spheroid should be \( [\lambda_{min}, \lambda_{max}] = [c+\text{clamp}(-\hat{r}, -r_{cam}, -r_{by}), c+\text{clamp}(r, r_{cam}, -r_{by})] \), in which \( r_{cam} \) is the distance from the camera to the fragment and
\( r_{by} \) is found by scaling \( r_{cam} \) by the ratio of the depth value of the background to the one of the fragment:

\[
r_{by} = r_{cam} \cdot \frac{\text{depth}_{\text{background}}}{\text{depth}_{\text{fragment}}} - r_{cam} \quad (33)
\]

In our implementation, we assigned \( M \) with the position of the fragment as an approximation, which makes \( OM \) exactly the position of the fragment in laser space; since we specified the billboard sprites of the particle system to be always facing the camera instead of aligning to the viewing direction as mentioned at the beginning of section 3.4, the position of the fragment is close to the precise position of \( M \). This can be replaced with the precise position with further calculations, but we choose using an approximation for this study nonetheless.

**Find the Nearest Position Inside Spheroid on the Ray to the Laser.**

In order to render a better detail of the laser passing through the fog (where the laser dimmer when the fog density is thinner and vice versa), we sampled the noise texture once at the nearest position on the ray to the laser to introduce this effect to the rendered result.

The nearest position is found by clamping the offset of \( N \) in figure 11, the nearest position on the ray to the laser ignoring the spheroid, into the range of the ray inside the spheroid. Since by definition \( N \) is always of an offset of 0, the offset of the nearest position inside the spheroid is precisely:

\[
\lambda_{\text{inside}} = \text{clamp}(0, \lambda_{\text{min}}, \lambda_{\text{max}}) \quad (34)
\]

which allows us to find the position of the nearest position in the spheroid in world space, similar to equation 32:

\[
N_{\text{inside}} = M_{\text{world}} + \lambda_{\text{inside}} \cdot d \quad (35)
\]

where \( M_{\text{world}} \) is the position of \( M \) in world space.

**Sample the Integrated Light Scattering Model.** As the final step, we will take samples from the Integrated Light Scattering Model and turn it into the amount of radiance from the laser as the rendered result.

First of all, we need to find the distance from the laser to the primary ray. Given laser-space direction vector of the primary ray \( d = (d_{x}, d_{y}, d_{z}) \) and also the laser-space position of the fragment \( p \), we firstly find the direction of the vector of minimum magnitude between he laser and the primary ray \( \hat{l} \) by rotating the projection of \( d \) on X-Y plane by 90\(^\circ\):

\[
\hat{l} = \left| l' || l' \right|, l' = (-d_{y}, d_{x}, 0) \quad (36)
\]

After that, the distance \( l \) is found by projecting an arbitrary vector from the laser to the primary ray onto \( \hat{l} \); the laser-space position of the fragment \( p \) serves well for this, since the origin in laser space is on the laser and the fragment itself is on the primary ray:

\[
l = |p \cdot \hat{l}| \quad (37)
\]

Also, we have to find the angle between the ray and the laser as two skew lines in responsible to the way we sample the Integrated Light Scattering Model according to equation 23:

\[
\sin \theta = \sqrt{\frac{d_{x}^{2} + d_{y}^{2}}{d_{x}^{2} + d_{y}^{2} + d_{z}^{2}}} \quad (38)
\]

With all of the values we have found previously, the amount of radiance is calculated by sampling the Integrated Light Scattering Model according to equation 23 and scale the value by the noise texture as well as the optical thickness \( \tilde{t}_{\text{ray}} \) from the Icosahedron Interpolated Ray-mapped Texture:

\[
L_{\text{ray}} = \tilde{L}_{\text{ray}} \cdot \left[ \frac{\tilde{\sigma}_{\text{ray}}(\lambda_{\text{max}} \sin \theta)}{\sin \theta} \cdot \tilde{t} \cdot \text{Density}(\frac{N_{\text{inside}} - C}{\text{size}}) \right] \quad (39)
\]

with \( C \) as the position of the center of the spheroid.

**3.4.4 Fog Coloring.** With the sample \( S_{\text{ray}} = (\tilde{t}_{\text{ray}}, r_{\text{ray}}) \) from the Icosahedron Interpolated Ray-mapped Texture and the amount of radiance gained by the laser \( L_{\text{ray}} \) from the previous section, we are able to present our visuals in either raw-but-informative or realistic coloring; the coloring method are applied to both FOLAR and the ray-casting method (which will be introduced in section 3.5) for the comparison tests in our study.

**Raw Coloring.** Raw coloring is done by simply outputting the optical thickness \( t_{\text{ray}} = \tilde{t}_{\text{ray}} \) in sample \( S_{\text{ray}} \) to the red channel, the amount of radiance from the laser \( L_{\text{ray}} \) to the green channel, and the natural logarithm of \( L_{\text{ray}} \) to the blue channel purely for visualization purposes (not to be studied). To be precise, the output of the fragment shader is in form of:

\[
C_{\text{raw}} = (r_{\text{raw}}, g_{\text{raw}}, b_{\text{raw}}, a_{\text{raw}}) = (t_{\text{ray}}, L_{\text{ray}}, \ln L_{\text{ray}}, 1) \quad (40)
\]

with HDR (high dynamic range) color enabled so that values beyond [0, 1] will not be clamped by the rendering pipeline.

**Realistic Coloring.** We did the realistic coloring in a relatively simply way by mixing certain colors together according to the values from the previous sections. Given the albedo color of the fog \( C_{\text{fog}} = (r_{\text{fog}}, g_{\text{fog}}, b_{\text{fog}}, a_{\text{fog}}) \) and the color of the laser beam \( C_{\text{laser}} = (r_{\text{laser}}, g_{\text{laser}}, b_{\text{laser}}, 1) \), we colored the fog by the following equations:

\[
C_{\text{realistic}} = C_{a} \cdot a_{a} + C_{r} \cdot a_{r}, \quad (41)
\]

\[
a_{a} = \text{lerp}(a_{\text{fog}}, 0, e^{-r_{\text{ray}}}), \quad (42)
\]

\[
C_{a} = (r_{\text{fog}}, g_{\text{fog}}, b_{\text{fog}}, a_{\text{a}}), \quad (43)
\]

\[
a_{r} = 1 - (1 + \text{clamp}(a \cdot L_{\text{ray}}, 0, 1))^{-1}, \quad (44)
\]

\[
C_{r} = \text{lerp}(r_{\text{laser}}, g_{\text{laser}}, b_{\text{laser}}, \alpha_{r}), (1, 1, 1, \alpha_{r}), \quad (45)
\]

\[
\text{clamp}(1 - (a \cdot L_{\text{ray}})^{b}, 0, 1)) \quad (46)
\]

in which we introduced parameters \( a \) and \( b \) to control how much color of the laser we mix into the final output and how bright the color is when the amount of radiance times \( a \) is greater than 1; we kept on using \( a = 0.1 \) and \( b = 1 \) for the whole study.

**3.5 Ray-casting Method.**

In order to compare the results of FOLAR to the results from other methods as a reference, ray-casting methods are an obvious choice for us, since our method is designed to mimic the ray-casting methods with the usage of VR in mind.

The ray-casting method we used for this study is a simple ray-marching method with fixed marching step size and constant render distance. The effect of both out-scattering and in-scattering along
the primary ray are calculated without tracing secondary rays caused by light scattering.

For each marching step, we add the attenuation cross-section, i.e. \( \sigma(\psi) = \bar{\sigma} \times \text{Density}(\psi) \), of every particle multiplied by the step size to the total optical thickness \( \tau_{ray} \) which is to determine how much the light from the background is attenuated. The amount of scatter-in radiance from the laser during each marching step is added to an intermediate variable along the ray for each marching step and attenuated by the average optical thickness (mean of attenuation cross-section of the two steps times the step size) between the steps. After the marching, the intermediate variable is treated as the total amount of radiance from the laser \( L_{ray} \) received by the camera.

To determine the amount of scatter-in radiance from the laser, we used the iteratively simulated Light Scattering Model from section 3.3.3 as an approximation of it, without tracing along the secondary rays.

The step size for all tests in our study (including the benchmark study and image-based study) is fixed 0.01 with a constant rendering distance of 5, which covers all the tasks in the comparative studies.

4 STUDY DESIGN

4.1 Benchmark Test

In order to validate the method for the desired time-efficiency, we conducted benchmark tests based on mean rendering time on per-frame basis (counting in both CPU and GPU in the measurements). We fixed the configuration of the scene and vary certain parameters one at a time for each measurement.

We conducted benchmark tests for two configurations of the scene. Both scenes were rendered under \( N \times N \) resolution, where we varied \( N \) to control the number of rendered fragments. The camera was positioned exactly at \((0, 1.472, 0.910)\) with direction vector \((0, -0.851, -0.526)\) and up vector \((0, -0.526, -0.851)\). The size of particles are fixed to be 2, so that it fills the whole field of view of the camera.

- **Scene A**: Exactly one particle is rendered at \((0, 0, 0)\).
- **Scene B**: The particle in **Scene A** is rendered 10 times per frame to mimic the usage of particle fog in actual applications.

The number 10 in **Scene B** is selected based on our assumption: to make a dense and dynamic particle fog, in average 10 fragments rendered on each pixel is enough.

For both scenes, we measured the performance of the method under a set of render resolutions ranged from \(50 \times 50\) to \(3000 \times 3000\). This allow us to control the number of fragments rendered every frame. Since we want to fill the whole field of view of the camera, we set the width and the height of any resolution to be identical, making the field of view spherical. The width used in the measurements for the two scenes are:

- **Scene A**: 500, 600, 800, 1000, 1500, 2000, 2500, 3000
- **Scene B**: 50, 80, 100, 120, 150, 200, 300, 500, 600, 800, 1000, 1200, 1500, 2000, 2500, 3000

To measure the mean rendering time, we run the two methods one at a time under certain resolution for 20 consecutive frames and record the time stamps in the end of each frame (including the frame before the measured first frame) with the time stamp variable \( \text{time.realtimeSinceStartup} \) provided by Unity. The rendering time of each frame is found by subtracting the neighboring time stamps, and the mean rendering time is found by subtracting the last time stamp by the first time stamp then divided by the total number of frames, i.e. 20.

The hardware we conducted the benchmark tests consists of an Oculus Rift and a computer with graphic card Radeon RX 580, CPU Ryzen 5 1600X, and two 8GB DDR4 2134c8 RAM cards in dual-channel.

4.2 Image Similarity

4.2.1 Scene Configuration. In the following study, we used always the same configuration of the particle and the laser. It can be reproduced by positioning the particle at \((0, 0, 0)\), the origin of the laser at \((-0.310, -0.252, 0)\) and the direction vector of the laser to be \((0, 0, 1)\). All images are rendered in \(512 \times 512\) resolution, with the field of view of the perspective camera to be \(60\degree\), the distance from the camera to the particle always \(1.73\), the direction of the camera to be always towards the particle, and the width of the square-shaped billboard sprite of the particle to be 2, so that the particle fills the whole field of view of the camera. Under the conditions above, we will specify the camera for each test by giving the direction vector and the up vector of the camera, since the parameters of the camera can be reproduced with these vectors.

4.2.2 Quantitative Method. As FOLAR is designed to imitate the images from ray-tracing methods with the same scene configuration, for the quantitative study, we took measurements based on a image similarity metric called sum of squared error (SSE) with normalization on the image.

Before calculating SSE, we render the images of particles one pair at a time with the scene configuration mentioned above in raw coloring as mentioned in section 3.4.4 and store the rendered result in a 3-channel-color image, preferably in OpenEXR format which essentially allows the storing of HDR float-point values.

For each pair of those images, SSE is measured separately on the red and blue channel (corresponds to \(\tau_{ray}\) and \(L_{ray}\)) with normalization. Specifically, we firstly normalize the intensity of values in the image per-channel by dividing the values by the standard deviation of each channel:

\[
\hat{I}^C = \frac{I^C - \bar{I}^C}{\sigma(I^C)} , c := L \text{ or } \tau
\]

Then we calculate the SSE for each channel:

\[
\text{SSE}(c) = \sum (\hat{I}^C_{\text{FOLAR}}[i] - \hat{I}^C_{\text{RayCasting}}[i])^2 , c := L \text{ or } \tau
\]

We evaluated three sets of images, with the parameter of each group exactly as below:

- **Group 1**: Camera direction parallel to the direction of one of the parallel-groups and vary the fog density:
  - Direction vector \((0, -0.851, -0.526)\) and up vector \((0, -0.526, -0.851)\),
  - Fog density \(\bar{\sigma} \in \{0.2, 1.0, 5.0\}\),
  - Marked as test image pair 1-x, with x chosen from 0.2, 1.0, and 5.0 representing each fog density (e.g. image pair 1-0.2 or 1-5.0),
Each image in the pair is marked with either prefix -f or -r, representing the result from FOLAR or ray-tracing (e.g., image 1-0.2-f or 1-5.0-r), the same below for all groups.

- Group 2: Camera direction parallel to the center of a triangle with the vector of its vertices the same as the direction vectors of parallel-groups 2, 5, and 11, varied fog density:
  - Direction vector (0.577, -0.577, 0.577) and up vector (0.408, 0.816, 0.408),
  - Fog density $\sigma \in \{0.2, 1.0, 5.0\}$,
  - Marked as test image pair 2-x, with x chosen from 0.2, 1.0, and 5.0 representing each fog density.

- Group 3: Camera direction parallel to the laser, fixed fog density:
  - Direction vector (0, 0, -1) and up vector (0, 1, 0),
  - Fog density $\sigma = 1.0$,
  - Marked as test image pair 3.

4.2.3 Qualitative Method. Since we are also interested in how humans perceive the images from FOLAR and ray-tracing, we designed a test based on surveys of image similarity that is conducted on computer screens with a survey software. The test has the following major steps for each test participant:

- We first show the introduction of the study and the survey software to the test participants, where an image of the survey software is shown to allow the test participants to be prepared for the test under the introduction text:
  "Thank you for your help with this study. The goal of this study is to compare two computer graphics techniques for producing images. During the test you will be presented with pairs of images. For each image pair, you will be asked two questions. Please make sure your screen in as clean as possible and do not change its settings during the study. Your answer can be processed purely based on your own judgement. To enter your answer, simply drag the horizontal slider to your response. At the end of the test, you will be asked to write down some of your demographic information. We are asking for your age and gender."

- During the test, the test participant is given pairs of images on screen and is asked the following two questions:
  "How much do these two images differ as a whole, for example difference in lightness and saturation?"
  "How much do these two images differ in their details, ignoring aspects such as lightness and saturation?"

- Under the pair of images and on the right of each question, there is a slider that allows the test participant to enter their response as a rating. The knots on sliders can be dragged horizontally onto evenly distributed 10 positions, with texts "no differences" and "completely different" on each side; the values of the slider are recorded as the answer of rating from test participants, with 1 representing the least amount of differences and 10 representing the most amount of differences.

- After the test, the test participant is asked to leave some demographic information, that is the age and gender of the test participant. This is to better understand the data in case that the data is showing significant characteristics in interpersonal differences.

All instruction and question texts are given in other languages, such as Chinese, as well.

The images for the test are chosen from the images rendered with the parameters below:

- Group 4: Random directions, fixed fog density, the same coloring as in the quantitative method:
  - Direction vectors of (-0.394, 0.324, -0.860), (0.909, -0.318, -0.269), (0.705, -0.009, -0.709) and up vectors of (-0.260, -0.937, -0.234), (0.388, 0.883, 0.266), (-0.296, 0.905, -0.305) respectively,
  - Fog density $\sigma = 2.0$,
  - Marked as test image pair 4-1 to 4-3;

- Group 5: Camera direction "in the middle of" the direction of three parallel-group "next to each other", fixed fog density:
  - Direction vectors of (-0.981, -0.149, 0.126), (-0.165, -0.973, 0.164), (-0.243, -0.825, -0.510) and up vectors of (0.013, -0.696, -0.718), (-0.957, 0.118, -0.266), (0.563, 0.308, -0.767) respectively,
  - Fog density $\sigma \in \{3.0, 4.0, 5.0\}$ for each direction,
  - Marked as test image pair 5-1-x to 5-3-x, with x chosen from 3.0, 4.0, 5.0 representing each fog density (e.g. 5-5-3.0 or 5-1-5.0)

The survey software shows the following sets of image pairs in a shuffled order and show the pairs of images at a random order from left to right, with the purposes described in the text:

- Test Set 1-1: The image pairs in group 4, to allow us to evaluate the result between FOLAR and ray-tracing without a realistic coloring. In total 3 pairs of images are shown.
- Test Set 1-2: The same image twice, either FOLAR or ray-tracing, from the image pairs in group 4. This is to find a baseline result for set 1-1 when the test participants are facing the same images. In total 6 pairs of images are shown, twice as much as in set 1-1.
- Test Set 2-1: The image pairs in group 5 with fog density 4.0 (5-1-4.0 to 5-3-4.0), to allow us to evaluate the result between FOLAR and ray-tracing in a realistic coloring. In total 3 pairs of images are shown.
- Test Set 2-2: The same image twice, either FOLAR or ray-tracing, from the images used in set 2-1. This is to find a baseline result for set 2-1, similar to 1-2 but instead with a realistic coloring. In total 6 pairs of images are shown.
- Test Set 3-1: The image pairs with different fog density values and different rendering methods, by pairing the image from one rendering method (e.g. FOLAR) of fog density 3.0 with the image from the other rendering method (e.g. ray-tracing) of fog density 5.0. This is to evaluate the result by intentionally differentiate the density of fog, so that we can see how the details of the images are perceived by human. This make pairs of images such as pair 5-1-3.0-f with 5-1-5.0-r and pair 5-1-5.0-f with 5-1-3.0-r and etc., which is in total 6 pairs of images.
- Test Set 3-2: The image pairs with different fog density values and the same rendering method, using fog density 3.0 and fog density 5.0, to provide a baseline result for 3-1. This make pairs of images such as pair 5-1-3.0-f with 5-1-3.0-r and etc., which also makes 6 pairs of images in total.
The exact images used for each pair is listed in table 10 in the appendix section.

5 RESULTS

In this section, we will show our evaluations of the rendering results and the data in different tests. Raw data of the study can be found in appendix B.

5.1 Benchmark Test

In our benchmark tests, FOLAR is generally keeping running at an expected performance. According to measurements for scene A, the mean rendering time of FOLAR keeps stable at around 6ms under all of our measured resolutions, as seen in figure 12(a) and 13(a). In scene B, the rendering time starts to increase at the resolution of $1000 \times 1000$, as seen in figure 12(b) and 13(b), and it keeps in ratio to the number of fragments rendered, which can be seen in figure 13(c).

At the same time, the rendering time of the ray-casting went out of control in our measurements. We only took measurements under resolutions with width not over 2000 in scene A and not over 300 in scene B. As seen in the figures, the rendering time becomes greater than 1 second quickly upon increasing the render resolution.

In figure 12(c) and 13(c), we can also see that the standard deviation of the rendering time per frame are larger under higher resolutions. This outcome could be due to low precision on the timestamps that we used to calculate the rendering time when the hardware is under heavy work. Still, this does not mean our data are invalid; by measuring the rendering time of 20 consecutive frames, we effectively lowered the error introduced by this low precision.

From the data, we can speculate that FOLAR is able to produce interactive graphics on our hardware at 100 FPS (frames per second) on flat displays under the render resolution around $1200 \times 1200$ when 10 fragments are rendered on each pixel on average. The frame rate becomes 50 FPS if we increase the resolution to around $2000 \times 2000$.

5.2 Image Similarity Metrics Test

Under the configurations specified in the study design, we rendered all the images that will be applied to the image similarity metrics. The images can be seen in figure 14, with their marks under the images. The central red parts of images (c), (f), (i), and (l) does not represent the real rendering result since the color values from the images are clamped to range $[0, 1]$ due to the limitations in printing technology.

The result of SSE can be seen in figure 15. As shown in figure (a), $SSE(\tau)$ is significantly higher than $SSE(L)$. The cause of this can be seen in the images in figure 14: the rendered result from the ray-casting method, e.g. image 1-1.0-r, has a greater range of ‘flare’ colored in yellow compared to the result from FOLAR, e.g. image 1-1.0-f.

By looking at $SSE(L)$ only, we can see that the amount of difference rises when the fog density is increased. Since we normalized the images before calculating SSE, this increase might be due to FOLAR less focused on changed details with different fog density, while it is naturally covered by the ray-casting method.

5.3 User Study

In our user study, there are in total of 16 participants involved in the test. We invited 2 test participants in a pilot test, and then recruited 14 high school students in reward of movie tickets, snack boxes, or equivalent. In all test participants, there are 9 male, 4 female, and 3
The purpose of recruiting high school students is that we believe teenagers to be more sensitive to the difference in graphics. The results from them should serve better for our study. Also, teenagers are more open to different displaying methods in providing interactive graphics; they will accept the use of VR devices naturally since this technology has already appeared during their early life.

The rating between the images in test set 1 shows a significant difference between 1-1 and 1-2. Single-tailed paired t-test between 1-1 and 1-2 (either FOLAR only or ray-casting only) gives p-values that are all less than 0.0001. The rating for test set 2 shows a similar result to test set 1, with p-values less than 0.0001 as well. For test set 3, although we differentiated the fog density, the rating is still showing a significant difference between samples from different methods (test set 3-1) and samples from the same method (test set 3-2), with p-values less than 0.0001. This indicates that test participants recognize the difference between rendered results from FOLAR and ray-casting.

All p-values are listed in table 4. The distribution of ratings can be seen in figure 17, 18, and 19, notice that in the legend we used
Figure 15: SSE between rendered results from FOLAR and ray-casting, notice that a different scale is used for figure (b) to show differences of the small values better

Table 4: P-values from single-tailed paired t-tests between the results from test sets

<table>
<thead>
<tr>
<th>Test Set</th>
<th>Question</th>
<th>&quot;Total&quot;</th>
<th>&quot;Detail&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>1-2 (FOLAR only)</td>
<td>$&lt; 10^{-10}$</td>
<td>0.000008</td>
</tr>
<tr>
<td>1-1</td>
<td>1-2 (ray-casting only)</td>
<td>$&lt; 10^{-10}$</td>
<td>0.000009</td>
</tr>
<tr>
<td>2-1</td>
<td>2-2 (FOLAR only)</td>
<td>$&lt; 10^{-10}$</td>
<td>$&lt; 10^{-6}$</td>
</tr>
<tr>
<td>2-1</td>
<td>2-2 (ray-casting only)</td>
<td>$&lt; 10^{-10}$</td>
<td>$&lt; 10^{-6}$</td>
</tr>
<tr>
<td>3-1</td>
<td>3-2</td>
<td>0.000002</td>
<td>0.000008</td>
</tr>
</tbody>
</table>

Figure 16: Demographic information about test participants

Figure 17: Rating for images in raw coloring ($p < 0.0001$), with examples of image pairs

6 DISCUSSION
6.1 Benchmark Test

Based on our measurements in the benchmark test, we are confident to conclude that FOLAR fulfills an interactive-level performance with certain conditions.

From the data, we have seen that FOLAR is able to keep the frame rate at 100 FPS on flat displays with $1200 \times 1200$ resolution on Scene B. Since FOLAR is heavily based on the fragment shader in real-time rendering, we expect the mean rendering time to scale with the number of fragments that are rendered, as shown in figure 13(c).

For real applications, those numbers may have more complex meanings.

In the study design, we assumed that "to make a dense and dynamic particle fog, in average 10 fragments rendered on each pixel is enough". In real applications, content creators might not
need too many particles on the scene to get an impressive graphics. If fewer particles are used to make the fog effect, fewer fragments on average will be rendered for each pixel, which should scale the rendering time down in expectation.

In the aspect of the rendering pipeline, although the multi-projection technology allows the scene to be rendered for the displays at both eyes simultaneously, the fragment shader will still need to be executed separated for both eyes. The reason for this is, FOLAR uses viewing-direction-dependent parameters to find the rays mapped to each rendered fragment in order to compute the graphics. Since the real-time calculation of FOLAR is heavily based on the fragment shader, multi-projection does not allow us to reduce the rendering time by half. We expect that the rendering time is of the same proportion to the number of rendered fragments on VR than it on flat displays.

To overcome the performance issues, we propose to combine other general rendering methods with FOLAR to achieve the most suitable result for real applications. One plausible method is to intentionally render fewer fragments for the particles where the color is expected to be less dynamic, such as particles that are far from the laser and the viewing position.

Counting all the factors mentioned above, we believe that FOLAR is able to render at a sufficient performance level by itself on our specific hardware used for the benchmark test. As proposed by Oculus, it’s possible to achieve a smooth graphics at a frame rate of 45 FPS with Asynchronous SpaceWarp [11]. Since Oculus Rift displays at 1080 x 1200 resolution per-eye, by default scene B can be rendered at around 50 FPS on average with our hardware. For a different VR headset and hardware configuration, the situation may differ.

\[
\begin{align*}
\text{(a) Test Set 2-1} & \quad \text{(b) Test Set 2-2 (both FOLAR)} \\
\text{(c) Test Set 2-2 (both ray-casting)}
\end{align*}
\]

Figure 18: Rating for images in realistic coloring \((p < 0.0001)\), with examples of image pairs

\[
\begin{align*}
\text{(a) Test Set 3-1 (}\bar{\sigma}_{\text{FOLAR}} = 3.0, \bar{\sigma}_{\text{Ray Casting}} = 5.0) & \quad \text{(b) Test Set 3-1 (}\bar{\sigma}_{\text{FOLAR}} = 5.0, \bar{\sigma}_{\text{Ray Casting}} = 3.0) \\
\text{(c) Test Set 3-2 (both FOLAR, } \theta = 3.0, 5.0) & \quad \text{(d) Test Set 3-2 (both ray-casting, } \theta = 3.0, 5.0)
\end{align*}
\]

Figure 19: Rating for images with different fog density in realistic coloring \((p < 0.0001)\), with examples of image pairs; notice that \(\bar{\sigma}\) stands for fog density constant and \(\sigma\) stands for standard deviation

### 6.2 Image Similarity Study

#### 6.2.1 Quantitative Method

As mentioned in the study design, we used SSE between the rendered images from FOLAR and ray-casting to measure the similarity between the images as the result of the two rendering methods. As a result, we observed relatively lower SSE values on the optical thickness of the fog, \(\tau\), and relatively higher SSE values on the amount of radiance from the laser, \(L\).

By looking at the images used for SSE shown in figure 14, we can clearly see a similar pattern for the optical thickness between the images from the two methods. For example, we can see similar T-shaped high-density areas on image 1-1.0-f and 1-1.0-f by only viewing the red channel of the image, which is shown in figure 20. The intensity of the optical thickness from the two methods seems to differ, which could be a result of a series of approximations. This could be solved by adjusting constants in the calculations of FOLAR, but since we normalized the data before calculating SSE, this is not an issue for our study.

The amount of radiance on the images, however, shows considerably more differences between the two methods. By looking at images in figure 14, we can notice that the images from ray-casting always show a broader range of illuminated fog by laser, but the ones from FOLAR does not. Also, the effect of uneven lightness of the laser that uneven fog density gives also seems to differ between the two methods, which can be seen on image (h) and (k) in the...
Figure 20: Example of similar patterns in optical thickness between the images from FOLAR and ray-casting.

The effect that laser looks dimmer when further inside the fog volume is roughly the same between the two methods, which can be seen on image (c) and (f). On image (m), there are noticeable artifacts around the laser, curving to the disappearing point of the laser, also shown in figure 21. This curving artifact is unexpected and might confuse the viewer. However, given the condition that the laser is parallel to the viewing direction and close to the center of the field of view, this effect should be rare to happen in real applications. Nonetheless, this is still an obvious downside of FOLAR.

Figure 21: Curving artifacts in the rendered result from FOLAR when the laser is parallel to the viewing direction and close to the center of the field of view.

Given all visible differences in the images used in the quantitative study, we can confirm that the result from SSE are meaningful in showing the difference in rendered results between FOLAR and ray-casting.

6.2.2 Qualitative Method. In our user study design, we required the test participants to rate the amount of difference between a set of designed pairs of images in two aspects: the difference as a whole ("total") and the difference in detail ("detail"). According to the result of the user study, there is an extremely significant difference between ratings on images from the same method and ratings on images from different methods. From this, we can speculate that the difference in the rendered result between the two methods is always noticeable for teenagers at ages from 15 to 17.

By looking at the distribution of ratings, we can see that for images from different methods, the ratings from test participants tend to distribute evenly among the values, while the average rating for difference as a whole is always higher than it for the difference in detail. This outcome might indicate that those test participants are unsure about their criteria in rating the amount of difference, given that we explicitly indicated "no difference" for the minimum value and "totally different" for the maximum value of the rating in the survey software.

On the other hand, in test set 3, we intended to differ the fog density in the images. As a consequence of this, we can see a more substantial difference in the average ratings between the two questions. This outcome might also indicate that test participants are more confidence in deciding the rating under different aspects as designed for the two questions. In the result of the rating for test set 3, we can confirm that those test participants are able to spot the difference between the rendered results from FOLAR and ray-casting.

6.3 Summary

From the result of SSE, we can see that FOLAR was not able to achieve the same effect as ray-casting do, especially for the amount of radiance. Although we can see that the laser effect is rendered in the correct direction and length, the shape and range of the flare around laser were not perfect. The user study also confirms this, as our test participants recognize rendered results of FOLAR from that of ray-casting. Despite all the downsides, FOLAR is still able to achieve a satisfactory rendering performance. This performance allows the volumetric laser-illuminated fog to be presented in VR on an interactive level, while the trivial volumetric ray-marching method failed to do so.

In the aspect of visuals, although FOLAR is not perfect as we have tested, it is rendering the effect in a physically-correct methodology as designed. As a consequence of this, FOLAR is still able to overcome the issue of billboard sprites for the particle system. Since FOLAR is designed based on particle systems, content creators are still able to make use of their experience in creating particle fog to create a similar fog effect for VR. In conjunction with the advantages that FOLAR gives, the trade-off in visuals is acceptable for VR usages.

7 CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

In this thesis, we introduced a brand-new method, FOLAR, in rendering laser-illuminated fog effect at interactive performance for VR. We measured the performance of FOLAR with a benchmark test, and we assessed the similarity of result images from FOLAR and a trivial ray-casting method with image similarity metrics and
a user study. Discussions on the results from the benchmark test and the image similarity study have been provided. The main work of the degree project is to design and optimize FOLAR, and to study the main visual characteristic of this new method. Our contribution to the method is mainly categorized into several parts. We defined several properties of a standard spherical volume of fog and named the standard volume as a fog spheroid. We introduced a novel method named Icosahedron Interpolated Ray-mapped Texture to model the volumetric features of a fog spheroid. This is achieved by projecting the volume onto 12 textures of different directions, with each direction corresponding to the vertex vectors of a regular icosahedron. We adopted a particular method to construct the icosahedron, namely 3-rect construction of the icosahedron, to help us make use of the model simpler in real-time rendering with a method called facet tracing. We simulated the result of light-scattering in an infinite-volume homogeneous fog under different attenuation cross-sections, then modeled the result as an integrated function over orthogonal paths as the Integrated Light Scattering Model. Finally, given all the materials we prepared, we are able to render fog spheroids as particles in a particle system in real-time. In the study, we conducted a benchmark test to measure the performance of FOLAR, as well as to show the difference in rendering time between FOLAR and ray-casting. As a result, FOLAR shows significant advantages in performance, allowing the volumetric laser-illuminated fog to be rendered at interactive performance for VR. However, tests in image similarity still show recognizable differences in visuals between FOLAR and ray-casting. Still, the difference in visuals is acceptable as a trade-off for the performance.

7.2 Critiques and Future Work

There are few places that this study about FOLAR and the design of FOLAR itself could be or could have been improved. Those places include aspects that the study could not cover, and also regrets and possible improvements if we can conduct further studies based on this work.

First of all, FOLAR itself has a large proportion of original works based on continuous developments. Sunken costs have been paid in the development of the method before it comes to the state that we feel sufficient for study. Still, FOLAR has several distinct places that could be improved, but we choose not to make more efforts into the development of it.

In detail, approximations were made at several places to give FOLAR a better performance. Those approximations directly lead to the majority of differences in the rendered result between FOLAR and the ray-casting method. However, we had several alternative methods for those approximations, which could lead to better results. If there were more works on it, it would be meaningful to conduct studies to compare the difference among those alternative methods.

Furthermore, FOLAR consists of many novel models and steps to render the final visual effect. Some of the models or steps have not been studied in depth separately, such as the Icosahedron Interpolated Ray-mapped Texture. Although we can certainly get a decent result with all the models and steps involved in FOLAR, the characteristics of those separate ones were not covered in this work. If given the opportunity, studies could be conducted for those separate steps or models; those studies could also be combined with the alternative methods mentioned earlier.

There are also some details in the design of FOLAR that uses ad hoc solutions to individual problems. One major one is the width of the facet vertex table, as shown in table 3. In the description of it, we said that there are inconsistent behaviors when the table is specified arbitrarily. We could not explain this behavior in this study, and we decided on using the ad hoc solution to this. We do not expect this to be solved for FOLAR, but further studies should be welcomed. With further studies, we could learn better about this behavior on different configurations of hardware.

In the aspect of the study design of this thesis, we several points and some considerations on what other study designs could be better for us to answer the research question. One consideration is to make a VR application with FOLAR and let the test participant involve in using the application. Specific contents and tasks could be made for the test, such as pointing at virtual objects with lasers or judging distances through the fog. Those tasks would help us to learn better about the user’s behavior in interacting with the application built with our method. By rating their performance on specific tasks, we could have a direct view of the visual characteristics of FOLAR.

Finally, for the quantitative test, we admit that SSE was not the best method to show the characteristics of FOLAR in the rendered result. During the literature study, we learned that there are methods to assess the quality of compressed stereoscopic graphics. By referring to similar studies designed explicitly for stereoscopic graphics or better VR, we can let the data tell more information about the visual characteristics of FOLAR.

REFERENCES


## A TEST IMAGES

![Figure 22: Test images for image similarity metrics, group 1](image1)

![Figure 23: Test images for image similarity metrics, group 2](image2)

![Figure 24: Test images for image similarity metrics, group 3](image3)
Figure 25: Test images for image similarity metrics, group 4

(a) 4-1-f  
(b) 4-2-f  
(c) 4-3-f

(d) 4-1-r  
(e) 4-2-r  
(f) 4-3-r

Figure 26: Test images for image similarity metrics, group 5, test set 1

(a) 5-1-3.0-f  
(b) 5-1-4.0-f  
(c) 5-1-5.0-f

(d) 5-1-3.0-r  
(e) 5-1-4.0-r  
(f) 5-1-5.0-r

Figure 27: Test images for image similarity metrics, group 5, test set 2

(a) 5-2-3.0-f  
(b) 5-2-4.0-f  
(c) 5-2-5.0-f

(d) 5-2-3.0-r  
(e) 5-2-4.0-r  
(f) 5-2-5.0-r

Figure 28: Test images for image similarity metrics, group 5, test set 3

(a) 5-3-3.0-f  
(b) 5-3-4.0-f  
(c) 5-3-5.0-f

(d) 5-3-3.0-r  
(e) 5-3-4.0-r  
(f) 5-3-5.0-r

Figure 29: Test images for image similarity metrics, group 5, test set 4

(a) 5-4-3.0-f  
(b) 5-4-4.0-f  
(c) 5-4-5.0-f

(d) 5-4-3.0-r  
(e) 5-4-4.0-r  
(f) 5-4-5.0-r

Figure 30: Test images for image similarity metrics, group 5, test set 5

(a) 5-5-3.0-f  
(b) 5-5-4.0-f  
(c) 5-5-5.0-f

(d) 5-5-3.0-r  
(e) 5-5-4.0-r  
(f) 5-5-5.0-r
B RAW DATA
B.1 Performance Benchmark
Table 5: Rendering time of FOLAR (seconds), 1 fragment per pixel, part 1

<table>
<thead>
<tr>
<th>Frame</th>
<th>Height or Width of Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>500</td>
</tr>
<tr>
<td>1</td>
<td>0.0103</td>
</tr>
<tr>
<td>2</td>
<td>0.0052</td>
</tr>
<tr>
<td>3</td>
<td>0.0047</td>
</tr>
<tr>
<td>4</td>
<td>0.0049</td>
</tr>
<tr>
<td>5</td>
<td>0.0101</td>
</tr>
<tr>
<td>6</td>
<td>0.0048</td>
</tr>
<tr>
<td>7</td>
<td>0.0047</td>
</tr>
<tr>
<td>8</td>
<td>0.0053</td>
</tr>
<tr>
<td>9</td>
<td>0.0054</td>
</tr>
<tr>
<td>10</td>
<td>0.0052</td>
</tr>
<tr>
<td>11</td>
<td>0.0052</td>
</tr>
<tr>
<td>12</td>
<td>0.0052</td>
</tr>
<tr>
<td>13</td>
<td>0.0048</td>
</tr>
<tr>
<td>14</td>
<td>0.0052</td>
</tr>
<tr>
<td>15</td>
<td>0.0061</td>
</tr>
<tr>
<td>16</td>
<td>0.0056</td>
</tr>
<tr>
<td>17</td>
<td>0.0053</td>
</tr>
<tr>
<td>18</td>
<td>0.0050</td>
</tr>
<tr>
<td>19</td>
<td>0.0052</td>
</tr>
<tr>
<td>20</td>
<td>0.0050</td>
</tr>
</tbody>
</table>

Table 6: Rendering time of ray-casting (seconds), 1 fragment per pixel

<table>
<thead>
<tr>
<th>Frame</th>
<th>Height or Width of Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>500</td>
</tr>
<tr>
<td>1</td>
<td>0.0810</td>
</tr>
<tr>
<td>2</td>
<td>0.0050</td>
</tr>
<tr>
<td>3</td>
<td>0.0579</td>
</tr>
<tr>
<td>4</td>
<td>0.0599</td>
</tr>
<tr>
<td>5</td>
<td>0.0598</td>
</tr>
<tr>
<td>6</td>
<td>0.1425</td>
</tr>
<tr>
<td>7</td>
<td>0.0818</td>
</tr>
<tr>
<td>8</td>
<td>0.0051</td>
</tr>
<tr>
<td>9</td>
<td>0.0578</td>
</tr>
<tr>
<td>10</td>
<td>0.0602</td>
</tr>
<tr>
<td>11</td>
<td>0.0595</td>
</tr>
<tr>
<td>12</td>
<td>0.1424</td>
</tr>
<tr>
<td>13</td>
<td>0.0829</td>
</tr>
<tr>
<td>14</td>
<td>0.0047</td>
</tr>
<tr>
<td>15</td>
<td>0.0580</td>
</tr>
<tr>
<td>16</td>
<td>0.0597</td>
</tr>
<tr>
<td>17</td>
<td>0.0601</td>
</tr>
<tr>
<td>18</td>
<td>0.1425</td>
</tr>
<tr>
<td>19</td>
<td>0.0817</td>
</tr>
<tr>
<td>20</td>
<td>0.0049</td>
</tr>
</tbody>
</table>
Table 7: Rendering time of FOLAR (seconds), 10 fragments per pixel, part 1

<table>
<thead>
<tr>
<th>Frame</th>
<th>Height or Width of Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>0.0107</td>
</tr>
<tr>
<td>2</td>
<td>0.0060</td>
</tr>
<tr>
<td>3</td>
<td>0.0054</td>
</tr>
<tr>
<td>4</td>
<td>0.0058</td>
</tr>
<tr>
<td>5</td>
<td>0.0056</td>
</tr>
<tr>
<td>6</td>
<td>0.0063</td>
</tr>
<tr>
<td>7</td>
<td>0.0055</td>
</tr>
<tr>
<td>8</td>
<td>0.0054</td>
</tr>
<tr>
<td>9</td>
<td>0.0059</td>
</tr>
<tr>
<td>10</td>
<td>0.0058</td>
</tr>
<tr>
<td>11</td>
<td>0.0056</td>
</tr>
<tr>
<td>12</td>
<td>0.0057</td>
</tr>
<tr>
<td>13</td>
<td>0.0063</td>
</tr>
<tr>
<td>14</td>
<td>0.0061</td>
</tr>
<tr>
<td>15</td>
<td>0.0066</td>
</tr>
<tr>
<td>16</td>
<td>0.0063</td>
</tr>
<tr>
<td>17</td>
<td>0.0059</td>
</tr>
<tr>
<td>18</td>
<td>0.0058</td>
</tr>
<tr>
<td>19</td>
<td>0.0061</td>
</tr>
<tr>
<td>20</td>
<td>0.0056</td>
</tr>
</tbody>
</table>

Table 8: Rendering time of FOLAR (seconds), 10 fragments per pixel, part 2

<table>
<thead>
<tr>
<th>Frame</th>
<th>Height or Width of Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>600</td>
</tr>
<tr>
<td>1</td>
<td>0.0106</td>
</tr>
<tr>
<td>2</td>
<td>0.0055</td>
</tr>
<tr>
<td>3</td>
<td>0.0059</td>
</tr>
<tr>
<td>4</td>
<td>0.0054</td>
</tr>
<tr>
<td>5</td>
<td>0.0059</td>
</tr>
<tr>
<td>6</td>
<td>0.0063</td>
</tr>
<tr>
<td>7</td>
<td>0.0056</td>
</tr>
<tr>
<td>8</td>
<td>0.0059</td>
</tr>
<tr>
<td>9</td>
<td>0.0064</td>
</tr>
<tr>
<td>10</td>
<td>0.0063</td>
</tr>
<tr>
<td>11</td>
<td>0.0060</td>
</tr>
<tr>
<td>12</td>
<td>0.0061</td>
</tr>
<tr>
<td>13</td>
<td>0.0062</td>
</tr>
<tr>
<td>14</td>
<td>0.0057</td>
</tr>
<tr>
<td>15</td>
<td>0.0061</td>
</tr>
<tr>
<td>16</td>
<td>0.0065</td>
</tr>
<tr>
<td>17</td>
<td>0.0068</td>
</tr>
<tr>
<td>18</td>
<td>0.0240</td>
</tr>
<tr>
<td>19</td>
<td>0.0055</td>
</tr>
<tr>
<td>20</td>
<td>0.0063</td>
</tr>
</tbody>
</table>
Table 9: Rendering time of ray-casting (seconds), 10 fragments per pixel

<table>
<thead>
<tr>
<th>Frame</th>
<th>Height or Width of Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td>1</td>
<td>0.1656</td>
</tr>
<tr>
<td>2</td>
<td>0.1601</td>
</tr>
<tr>
<td>3</td>
<td>0.1702</td>
</tr>
<tr>
<td>4</td>
<td>0.1714</td>
</tr>
<tr>
<td>5</td>
<td>0.1701</td>
</tr>
<tr>
<td>6</td>
<td>0.1692</td>
</tr>
<tr>
<td>7</td>
<td>0.1585</td>
</tr>
<tr>
<td>8</td>
<td>0.1584</td>
</tr>
<tr>
<td>9</td>
<td>0.1708</td>
</tr>
<tr>
<td>10</td>
<td>0.1587</td>
</tr>
<tr>
<td>11</td>
<td>0.1705</td>
</tr>
<tr>
<td>12</td>
<td>0.1589</td>
</tr>
<tr>
<td>13</td>
<td>0.1587</td>
</tr>
<tr>
<td>14</td>
<td>0.1716</td>
</tr>
<tr>
<td>15</td>
<td>0.1577</td>
</tr>
<tr>
<td>16</td>
<td>0.1597</td>
</tr>
<tr>
<td>17</td>
<td>0.1589</td>
</tr>
<tr>
<td>18</td>
<td>0.1696</td>
</tr>
<tr>
<td>19</td>
<td>0.1706</td>
</tr>
<tr>
<td>20</td>
<td>0.1587</td>
</tr>
</tbody>
</table>
B.2 User Study
Table 10: Question number in test sets with corresponding image marks

<table>
<thead>
<tr>
<th>Test Set</th>
<th>Question</th>
<th>Images</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>1-1-1</td>
<td>4-1.0-f 4-1.0-r</td>
</tr>
<tr>
<td></td>
<td>1-1-2</td>
<td>4-2.0-f 4-2.0-r</td>
</tr>
<tr>
<td></td>
<td>1-1-3</td>
<td>4-3.0-f 4-3.0-r</td>
</tr>
<tr>
<td>1-2 FOLAR</td>
<td>1-2-1f</td>
<td>4-1.0-f 4-1.0-f</td>
</tr>
<tr>
<td></td>
<td>1-2-2f</td>
<td>4-2.0-f 4-2.0-f</td>
</tr>
<tr>
<td></td>
<td>1-2-3f</td>
<td>4-3.0-f 4-3.0-f</td>
</tr>
<tr>
<td>1-2 Ray-casting</td>
<td>1-2-1r</td>
<td>4-1.0-r 4-1.0-r</td>
</tr>
<tr>
<td></td>
<td>1-2-2r</td>
<td>4-2.0-r 4-2.0-r</td>
</tr>
<tr>
<td></td>
<td>1-2-3r</td>
<td>4-3.0-r 4-3.0-r</td>
</tr>
<tr>
<td>2-1</td>
<td>2-1-1</td>
<td>5-1.4-0-f 5-1.4-0-r</td>
</tr>
<tr>
<td></td>
<td>2-1-2</td>
<td>5-2.4-0-f 5-2.4-0-r</td>
</tr>
<tr>
<td></td>
<td>2-1-3</td>
<td>5-3.4-0-f 5-3.4-0-r</td>
</tr>
<tr>
<td>2-2 FOLAR</td>
<td>2-2-1f</td>
<td>5-1.4-0-f 5-1.4-0-f</td>
</tr>
<tr>
<td></td>
<td>2-2-2f</td>
<td>5-2.4-0-f 5-2.4-0-f</td>
</tr>
<tr>
<td></td>
<td>2-2-3f</td>
<td>5-3.4-0-f 5-3.4-0-f</td>
</tr>
<tr>
<td>2-2 Ray-casting</td>
<td>2-2-1r</td>
<td>5-1.4-0-r 5-1.4-0-r</td>
</tr>
<tr>
<td></td>
<td>2-2-2r</td>
<td>5-2.4-0-r 5-2.4-0-r</td>
</tr>
<tr>
<td></td>
<td>2-2-3r</td>
<td>5-3.4-0-r 5-3.4-0-r</td>
</tr>
<tr>
<td>3-1</td>
<td>3-1-1a</td>
<td>5-1.3-0-f 5-1.5-0-r</td>
</tr>
<tr>
<td></td>
<td>3-1-2a</td>
<td>5-2.3-0-f 5-2.5-0-r</td>
</tr>
<tr>
<td></td>
<td>3-1-3a</td>
<td>5-3.3-0-f 5-3.5-0-r</td>
</tr>
<tr>
<td></td>
<td>3-1-1b</td>
<td>5-1.5-0-f 5-1.3-0-r</td>
</tr>
<tr>
<td></td>
<td>3-1-2b</td>
<td>5-2.5-0-f 5-2.3-0-r</td>
</tr>
<tr>
<td></td>
<td>3-1-3b</td>
<td>5-3.5-0-f 5-3.3-0-r</td>
</tr>
<tr>
<td>3-2 FOLAR</td>
<td>3-2-1f</td>
<td>5-1.3-0-f 5-1.5-0-f</td>
</tr>
<tr>
<td></td>
<td>3-2-2f</td>
<td>5-2.3-0-f 5-2.5-0-f</td>
</tr>
<tr>
<td></td>
<td>3-2-3f</td>
<td>5-3.3-0-f 5-3.5-0-f</td>
</tr>
<tr>
<td>3-2 Ray-casting</td>
<td>3-2-1r</td>
<td>5-1.5-0-r 5-1.5-0-r</td>
</tr>
<tr>
<td></td>
<td>3-2-2r</td>
<td>5-2.5-0-r 5-2.5-0-r</td>
</tr>
<tr>
<td></td>
<td>3-2-3r</td>
<td>5-3.5-0-r 5-3.5-0-r</td>
</tr>
</tbody>
</table>
Table 11: Rating from test participants on the difference in total

<table>
<thead>
<tr>
<th>Question</th>
<th>Participant Number</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1-1</td>
<td></td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>1-1-2</td>
<td></td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>1-1-3</td>
<td></td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>8</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>10</td>
<td>3</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>1-2-1f</td>
<td></td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1-2-2f</td>
<td></td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1-2-3f</td>
<td></td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1-2-1r</td>
<td></td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1-2-2r</td>
<td></td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1-2-3r</td>
<td></td>
<td>1</td>
<td>3</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2-1-1</td>
<td></td>
<td>9</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>9</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>10</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>2-1-2</td>
<td></td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>2-1-3</td>
<td></td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>2-2-1f</td>
<td></td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2-2-2f</td>
<td></td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2-2-3f</td>
<td></td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2-2-1r</td>
<td></td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2-2-2r</td>
<td></td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2-2-3r</td>
<td></td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3-1-1a</td>
<td></td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>9</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>5</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>3-1-2a</td>
<td></td>
<td>2</td>
<td>4</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>8</td>
<td>10</td>
<td>10</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>10</td>
<td>4</td>
<td>8</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>3-1-3a</td>
<td></td>
<td>3</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>9</td>
<td>5</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>3-1-1b</td>
<td></td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>6</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>3-1-2b</td>
<td></td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>7</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>3-1-3b</td>
<td></td>
<td>9</td>
<td>7</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>6</td>
<td>10</td>
<td>9</td>
<td>7</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>3-2-1f</td>
<td></td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>7</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3-2-2f</td>
<td></td>
<td>2</td>
<td>7</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>10</td>
<td>4</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>3-2-3f</td>
<td></td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>10</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>8</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>3-2-1r</td>
<td></td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>7</td>
<td>3</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td>10</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>3-2-2r</td>
<td></td>
<td>2</td>
<td>1</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>10</td>
<td>7</td>
<td>8</td>
<td>6</td>
<td>10</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>3-2-3r</td>
<td></td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>8</td>
<td>8</td>
<td>4</td>
<td>10</td>
<td>4</td>
<td>7</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>Question</td>
<td>Participant Number ('Participants from the pilot test')</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>1-1-1</td>
<td>10</td>
<td>8</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>1-1-2</td>
<td>10</td>
<td>2</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1-1-3</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1-2-1f</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2-2f</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2-3f</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2-1r</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2-2r</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1-2-3r</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-1-1</td>
<td>10</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>10</td>
<td>7</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2-1-2</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>10</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>10</td>
<td>6</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-1-3</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>9</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2-2-1f</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-2-2f</td>
<td>7</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-2-3f</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-2-1r</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-2-2r</td>
<td>9</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2-2-3r</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-1-1a</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>10</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>3-1-2a</td>
<td>8</td>
<td>2</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>10</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3-1-3a</td>
<td>8</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td>9</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3-1-1b</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>10</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3-1-2b</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-1-3b</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>10</td>
<td>7</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>3-2-1f</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-2-2f</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-2-3f</td>
<td>9</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-2-1r</td>
<td>4</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-2-2r</td>
<td>9</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>10</td>
<td>7</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3-2-3r</td>
<td>4</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>7</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>