A multiscale model for rain rendering in real-time

Yoann Weber[a], Vincent Jolivet[a], Guillaume Gilet[a], Djamchid Ghazanfarpour[a]

[a]*XLIM - UMR CNRS 7252, University of Limoges, France*

Abstract

This paper presents a coherent multiscale model for real-time rain rendering which takes into account local and global properties of rainy scenes. Our aim is to simulate visible rain streaks close to the camera as well as the progressive loss of visibility induced by atmospheric phenomena. Our model proposes to correlate the attenuation of visibility, which is due in part to the extinction phenomenon, and the distribution of raindrops in terms of rainfall intensity and camera's parameters. Furthermore, this method proposes an original rain streaks generation based on spectral analysis and sparse convolution theory. This allows an accurate control of rainfall intensity and streaks appearance, improving the global realism of rainy scenes.

Keywords: Natural phenomena, Rain, Physically-based modeling, Real-time rendering, Attenuation of visibility, Extinction

1. Introduction

Rain rendering has been widely studied in computer graphics. Rainy scenes include several effects such as streaks due to the velocity of raindrops and camera's exposure time or retinal persistence. It also includes among other features a progressive loss of visibility, due to suspended raindrops in the air spreading light in a specific way. Thus, the realism of a rainy scene in computer graphics depends on the raindrops appearance as well as the global properties of this meteorological phenomenon. These problems can be tackled by simulating the mesoscopic and macroscopic views of a rainy atmosphere. In this paper, we take into account these two scales.

The existing methods mainly focus on streaks rendering. The most recent ones, such as [1] and [2], use a particle system and a textures database to render streaks. Nevertheless, such a database has a high memory footprint and also requires complex mechanisms to control particles, in order to preserve a constant and physical distribution of raindrops in space during the rain simulation. Another technique for rain rendering [3] consists in using one or several rainfall textures positioned at different distances from the observer. These textures are defined in a finite space. Such methods cause a repetition of rain patterns as textures are moving and repeated over and over again. All of these methods usually use fog models in an empirical way to simulate the loss of visibility naturally present in rainy scenes. Their attenuation is neither based on physically realistic extinction coefficients, nor correlated with rainfall intensity. This paper addresses these issues and proposes a real-time multiscale method relying on image-space post-processing (i.e. independent of scene complexity) computations. The main contributions of our method are:

- A procedural generation of streaks in the image space, based on spectral analysis of real rain patterns. The sparse convolution noise technique is used to control precisely the rainfall intensity, and to manage only visible streaks. This generation also enables a constant and physical distribution of raindrops during the simulation.
- A density function to quantify the visible streaks in the view frustum in terms of rainfall intensity and camera's parameters.
- A physical correlation between the density of visible streaks and the attenuation of visibility. Only once distribution is used to evaluate the density of raindrops and also to render the loss of visibility in the scene.

2. Previous works

We will focus on methods which propose streaks generation and attenuation of visibility. These two phenomena are the key elements of our multiscale model.

2.1. Rain streaks rendering

We can classify papers in two categories: the ones based on scrolling rainfall textures and the others based on a particle system.

The papers [3] and [4] fall into the first category of methods. The advantage lies in its simplicity and reduction of the computation time. Nevertheless there are two recurrent problems with the rainfall textures. The repetition of the patterns prevents the rain effect from being visually realistic while the simulation is running. Indeed, those textures are finite and usually artistically-generated. This results in an empirical distribution of raindrops. Moreover the user cannot change the rainfall intensity in real-time during the simulation, and it is impossible to evaluate collisions between raindrops and objects in the scene.

Meanwhile, the papers [1], [2], [5], [6], [7], [8] and [9] achieve streaks rendering by using a particle system. Those methods are more computationally expensive. Generally, they either extract textures from video which are then mapped onto particles, or use some precomputed ones. In this regard, Garg and Nayar [2] are the first to propose a complete precomputed streaks database. This method has been used by Tariq [7] and more recently by Creus and Patow [1], yielding really convincing results. Textures are classified in terms of three different angles depending on the position of the observer and the light sources. The main downside is that a such database has a very important memory footprint. Moreover a particle system costs a lot of GPU ressources, being fill rate based, and rain animation is not a trivial task, managing only visible particles in the view frustum while avoiding repetitive behaviour and having an invariant distribution of raindrops. The random respawn of particles does not guarantee a physically realistic spatial distribution of raindrops. This implies an adapted implementation.

2.2. Attenuation rendering

Most methods use an empirical approach to simulate the attenuation of visibility. Using an arbitrary fog model yields to a non-physically realistic rendering. For instance, Tatarchuk and Isodoro [3] and Tariq [7] propose an attenuation without using a specific value for the so-called extinction coefficient. However our research shows that this coefficient must be correlated with the rainfall intensity and the distribution of raindrops.

Changbo et al [10] offer the possibility to simulate the attenuation of visibility taking into account different sizes of raindrops in the air. They made computations for every type of raindrops, and considered a uniform phase function to evaluate how light is scattered. It is however difficult to know which data must be precomputed, and set precisely the extinction coefficient in order to obtain a correlation with the distribution of raindrops.

A different approach for snow or rain rendering is proposed by Langer et al [11], using an infinite density function based on a sum of 2D waves cosine to simulate the density of precipitation. The density function is expressed in the spectral domain. The Fourier Transform and its inverse are computed. The evaluation of this function in the spectral domain is not based on a physical distribution of raindrops in the spatial domain.

These methods achieve an attenuation rendering by using a macroscopic approach. Indeed, increasing the number of particles does not simulate the continuity of the meteorological phenomenon. Generating more particles induces visual artifacts due to the resolution, and does not naturally generate attenuation of visibility. Nevertheless, most of these methods overlook the distribution of raindrops to get a physically realistic solution. This prevents a correlation between the rainfall intensity and the attenuation of visibility.

3. New multiscale model

A rainy scene is typically composed of many raindrops - each with a size much larger than visible wavelengths - which causes the light to spread in a specific way. Closer raindrops, which have a projected size larger than a pixel, are individually visible, and further ones can only be seen in a macroscopic view as a participating media. As a result, visibility decreases until details blur into a fog like appearance. Taking into account these two rain features enables to propose a physically based rain simulation. The main contribution of this paper is to propose a correlation between the density of close and visible raindrops and the attenuation of visibility.

According to the preceding observations, we decompose a rainy scene in two areas. The first one corresponds to the region of the scene where streaks have a projected size larger than a pixel or thus are individually visible. We will refer further at this region as the Visible Streaks Region (VSR). In this region, our rain streaks generation has to take into account the problems inherent to particle systems and finite rainfall textures, as we discussed in the section 2. Beyond this rain region, streaks cannot be individually seen because it would require a subpixel precision. Moreover, that would imply simulating a large number of raindrops. The idea is to develop a scattering method specifically for this Scattering Rain Phenomena (SRP) in order to indirectly simulate remote raindrops. This phenomenon is continuous in the whole scene from the camera to the furthest visible object in the scene. It simulates the raindrops too small to be represented even in the Visible Streaks Region. There is actually an attenuation effect in the whole scene, even in the VSR where it could be assumed to be very close to zero. All raindrops cannot be actually simulated even in the VSR. This implies rendering this phenomenon with a macroscopic approach to simulate how light behaves in contact with raindrops in the air. Such a multiscale method enables to represent each phenomenon of rainy scenes with an adapted model, and to correlate each model according to physically based parameters. This part of the paper will discuss which models to use to simulate the VSR and SRP, and which physical parameters will allow us to correlate each model and control our rain simulation in real-time.

We propose in the subsection 3.1 a generation of visible rain streaks. This first model simulates only visible and individual streaks close to the camera. The second model, presented further in the subsection 3.2, takes into account the distribution of far raindrops in the scene with a single scattering method, to progressively attenuate the visibility.

3.1. The Visible Streaks Region (VSR)

Section 2 highlights the recurrent problems inherent to existing methods based on particle systems and finite rainfall textures. Particle systems offer the possibility to precisely control rain streaks but require complex mechanisms to manage only visible raindrops. Indeed, some of those raindrops may be visible after projection and some others may not, depending on their size and distance from the observer. Complexity also lies in both simulating a large number of particles with a constant distribution over space and usually in storing a texture database with a high memory footprint.

Meanwhile, finite rainfall textures reproduce the global appearance of this meteorological phenomena. Few calculations are necessary but the number of available rainfall textures is usually limited and therefore does not allow a precise control over the rainfall intensity in real-time. Moreover, repetitions appear due to finite resolution issues while rainfall textures are scrolling down. To tackle these problems, we propose a middle ground method to combine these two approaches. This technique is controlled by a density function in real-time, evaluated in terms of rainfall intensity and camera's parameters, and enables to manage only visible rain streaks. It provides the following advantages:

- A control over the density of raindrops through physical parameters,
- A physically based distribution of raindrops.

We consider rain as a set of raindrops, distributed in space. Their size varies, and they overlap on screen once projection is done. Therefore we can consider the color of a pixel of a rainfall texture as a contribution of different overlapping projected streaks. We can thus rely on sparse convolution theory. Sparse convolution techniques [12] have been widely used in texture synthesis to produce non-repetitive content for infinitely large surface. It provides a continuous definition of a given phenomenon that can be evaluated on-the-fly at any resolution. Sparse convolution techniques consist in uniformly sampling a spatial domain and associating a local kernel function to each sample. Each point of the spatial domain can then be evaluated as the sum of all distributed kernel functions. Sparse convolution noise [12] is usually defined using a uniform Poisson distribution of kernels [13]. As recently introduced by Gilet et al [14], simpler distribution schemes, such as regular or jittered grids, can also be used in conjunction with various different kernels, thus providing an accurate control over local and global features. As our goal is to represent an controllable rainfall texture, we propose to define the Visible Streaks Region as a convolution of rain streaks kernels distributed along a simple regular grid. Furthermore, to avoid the memory issues introduced by texture databases, we propose to express rain streaks kernels as a sum of cosine waves, extracted from real streak patterns from a real rain video using Fourier theory. This theory explains that any signal can be decomposed and reproduced by summing all frequency components. Thresholding the Fourier spectrum comes down to extracting fundamental frequencies, and reproducing a signal close to the original one with the following formula:

$$\sum_{i=1}^{N} \alpha_i \times cos(2\pi(f_i u \times g_i v) + \phi_i) \quad (1)$$

where N is the number of fundamental frequencies to reproduce a specific analyzed streak pattern, α is the magnitude, ϕ is the phase, u v are the spatial coordinates in the range [0 ; 1], and f g are the spatial frequencies.

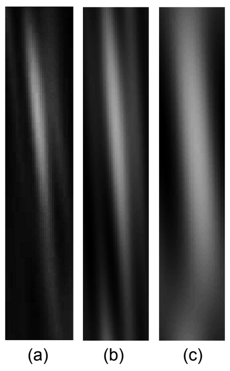

Figure 1: (a) Original streak pattern. Reconstruction of streak pattern with (b) 6 frequencies and with (c) 3 frequencies.

According to the theory, all frequencies should be used in Equation 1 to exactly reproduce the input signal. However, as shown in Figure 1, using 6 frequencies is sufficient to reproduce a visually similar rain streak (a lower number of frequencies deteriorates the original signal, as shown in the right of the figure). A finite rain pattern function is associated to each distributed kernel. To reproduce accurate global results, the density of kernels must be determined. The widely used Marshall and Palmer's distribution function [15] gives the number of raindrops in volume unity for a given rainfall and a given raindrop diameter. We have already explained that some raindrops may be visible after projection and others may not. In the Visible Streaks Region, we will focus on knowing the density of streaks which projected size happens to be larger than a pixel. Besides, the projected size of small and large raindrops may be the same depending on their distance from the camera, as shown in Figure 2.

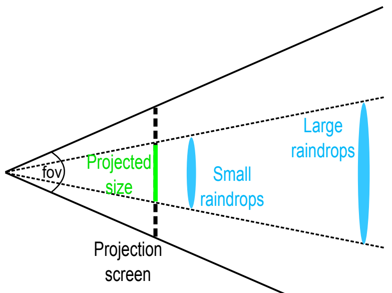

Figure 2: 2 different size of raindrops may have the same projected size depending on their distance from the observer.

Thus the idea is to compute the density of streaks for a size after projection sap and for a given raindrop diameter D, in terms of camera's parameters and rainfall intensity R. A closer look at projection calculations shows that the final projected size depends on camera's field of view fov and y-axis resolution h. Moreover, exposure time T makes streaks height vary in the scene. Taking all these factors into account enables to simulate the entire region close to the observer where streaks are individually visible. To do so, for a given projected size, we use a δsap to simulate a part of the view frustum, as shown in Figure 3. We assume that raindrops in this volume will have the same projected size. By uniformly sampling the projected size range and taking into account every raindrop diameters, the entire Visible Streaks Region may be simulated (Figure 7).

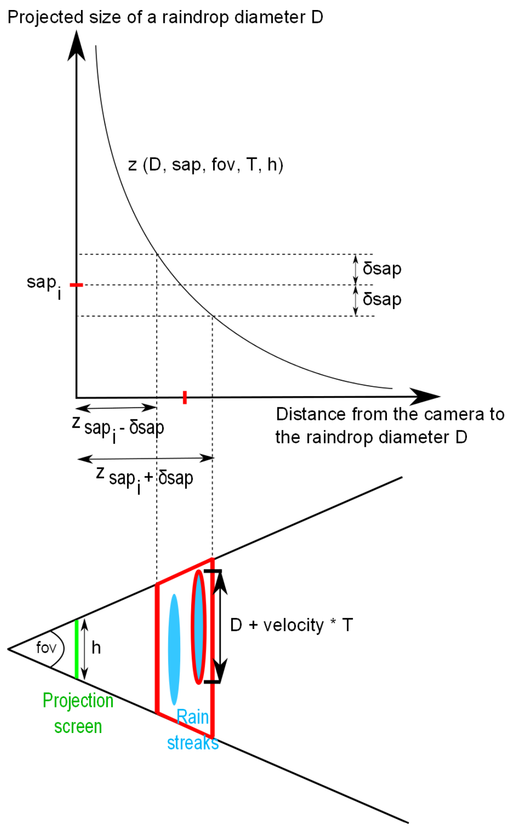

Figure 3: Simulation of a part of the view frustum for a given projected size *sap* and for a given raindrop diameter D, in terms of camera's parameters fov and T.

Figure 3 introduces the z function defining at which distance a raindrop is positioned from the observer to project a size equal to *sap*. Figure 4 shows that the z function curves profile changes in terms of raindrop diameter D. These raindrop diameters are presented in Table A.5 in the appendix.

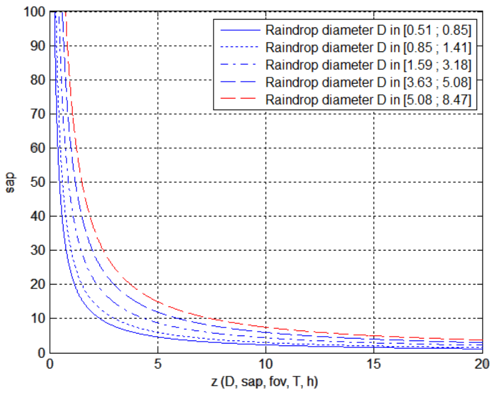

Figure 4: Curves profile depends on raindrop diameter D. Camera's parameters fov and T are set, respectively, at 120 degrees and 20 ms. The y-axis resolution is set at 1080 pixels.

The z function is expressed as follows:

$$z(D, sap, fov, T, h) = \frac{sbp(D, T) \times h}{2 \times sap \times tan(fov \times 0.5)} \tag{2}$$

where $sbp(D, T)$ is the size before projection function, expressed in m, computed as follows:

$$sbp(D, T) = \frac{D}{1000} + v(D) \times \frac{T}{1000} \tag{3}$$

where $v(D)$ is the velocity function for a raindrop diameter D. As it was recalled by Garg and Nayar [16], the velocity of a raindrop can be expressed as:

$$v(D) = 200 \times \sqrt{\frac{D}{2000}} \tag{4}$$

where D is in mm and T in ms. $v(D)$ is expressed in $m.s^{-1}$.

Using the z function presented above, a volume is computed in the view frustum as shown in Figure 3, and the visible streaks density function f is then deduced as follows:

$$\begin{aligned} f(D, sap, \delta sap, fov, T, h, R) = \\ V(D, sap, \delta sap, fov, T, h) \times N(D, R)\delta D \end{aligned} \tag{5}$$

where $N(D, R)\delta D$ is a number of raindrops expressed in m^{-3} for a raindrop diameter range δD. $V(D, ..., h)$ is a function to compute the volume of a part of the view frustum using Equation 2. According to the Marshall and Palmer's distribution of raindrops [15]:

$$N(D, R) = N_0 \ e^{-\Lambda_R D} \tag{6}$$

where

$$N_0 = 8000 \ m^{-3} \ mm^{-1} \tag{7}$$

and

$$\Lambda(R) = 4.1 R^{-0.21} \ mm^{-1} \tag{8}$$

Regarding the rain animation, this consists in translating the kernels to get the desired rain effect. The distribution of raindrops remains constant during the simulation. The velocity function 4 is also used to compute the scrolling velocity for each projected size *sap* and each raindrop diameter D. The rendered streaks will then be blended with the scene using an artistic coefficient after the SRP computations in the next subsection. This coefficient aims to simulate the refraction of the scene through each streak in the VSR, and thus how streaks influence the overall lighting of the scene.

3.2. The Scattering Rain Phenomenon (SRP)

In the previous subsection we dealt with our streaks generation method to only simulate close and visible raindrops after projection. We are now going to indirectly simulate further raindrops by rendering the complex interactions between light and raindrops. Raindrops behave indeed like a participating media and this results in fog appearance. Light spreads in a specific way that causes a visibility attenuation effect. We will describe in this subsection the behaviour of light when it hits raindrops in the air. Single scattering is commonly used for simplicity in real-time simulations and to avoid too many computations. We suggest a simple single scattering model to simulate the attenuation of visibility in a rainy atmosphere, according to the same distribution of raindrops as the one described in the previous subsection to deal with the density function.

Light is scattered by air molecules and aerosols particles in the atmosphere. During sunny day, they are often assumed nonexistent even for realistic rendering in computer graphics. They actually induce a participating media which cannot be ignored in a rainy atmosphere. Raindrops are much larger than the particles usually found in the air. Thus this causes light to be spread in a particular way decreasing visibility in the scene. Garg and Nayar [17] explain light behaviour in contact with raindrops. Depending on the size of particles, light is scattered backwards or forwards. Concerning rain, light is rather diffused in the forward direction because raindrops are much larger than visible wavelengths. Figure 5 shows that the size of particles influences light scattering.

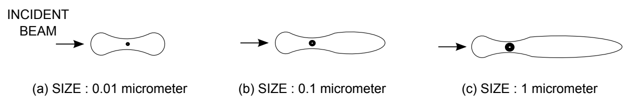

Figure 5: Light behaviour in contact with particles depending on the size from [17]

4 types of interactions can be defined in a participating media. Light absorption and out-scattering decrease radiance in the incident direction, whereas light emission and in-scattering increase it. The sum of absorption and out-scattering is called extinction. The extinction phenomenon has to be estimated during a rainy day to compute how light decreases in terms of distance and rainfall intensity, according to the distribution of raindrops. Extinction is generally considered constant through the participating media for simplicity in real-time simulations. We made this assumption by considering the atmosphere to be homogeneous.

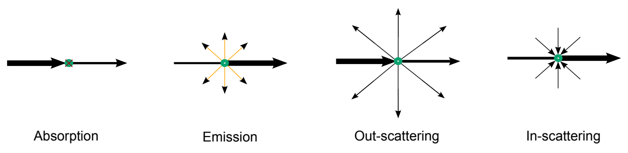

Figure 6: Types of interactions in a participating media from [18]

The extinction coefficient can be computed, as in [19], in terms of rainfall intensity according to the Marshall and Palmer's distribution of raindrops [15]:

$$\beta_{ext} = 0.312\ R^{0.67} \tag{9}$$

In the subsection 3.1, the density function is evaluated according to the same distribution, thus providing a correlation between the Visible Streaks Region and the Scattering Rain Phenomenon. The extinction coefficient usually depends on wavelength. Raindrops being far larger than visible wavelengths, we assume that the extinction coefficient is the same for each one of them.

Although raindrops cause extinction and therefore decrease radiance in the incident direction, light in-scattering has to be also considered, using the appropriate phase function. This enables to take into account how light from the sun reaches the observer's eyes, through the rain, to reproduce rainy sky and also aerial perspective. This function replaces the so-called BRDF for atmospheric rendering. Contrary to the BRDF, the phase function performed the integration over the whole sphere. Moreover there is no cosine term since no surface is present. For the given incident light and viewing directions, the phase function provides the quantity of light that is scattered, in contact with particles, toward the camera. Jarosz [18] dealt with the commonly used Henyey-Greenstein phase function when light is in contact with larger spherical particles than air molecules. We assume, for simplicity, that raindrops are spherical to respect this constraint. The Henyey-Greenstein phase function is much simpler to use than the original Mie phase function, and represents a good approximation according to most of the papers. The Henyey-Greenstein phase function is expressed as follows:

$$\beta_{HG}(\theta) = \frac{(1-g)^2}{4\pi(1+g^2-2g\cos(\theta))^{3/2}} \tag{10}$$

where g is the eccentricity or more precisely the scattering distribution. The value must be in the range $[-1\ ;\ 1]$. By varying this parameter, users can choose if the light distribution is rather in the backward or the forward direction. Considering rain, g has to be greater than 0.

As Hoffman and Preetham [20] and Cozman and Krotkov [21], the Beer-Lambert law is commonly used to simulate extinction in participating medias. This law is based on the idea that light transmission in a participating media decreases exponentially with pathlenght.

$$F_{ext}(s) = e^{-\beta_{ext} s} \tag{11}$$

where s is the distance between the object and the observer, and β_{ext} is the extinction coefficient of the media. As the Mie theory deals with large spherical particles, meanwhile the Rayleight theory attends to simulate very small particles such as air molecules. The Rayleight coefficients could be expressed to reproduce light absorption and out-scattering by those particles. Nevertheless, it are ignored in this model. The reason is that the Rayleight coefficients turn out to be far smaller than the previous computed rain extinction coefficients, and thus they do not strongly contribute to the rainy atmosphere. The contribution of in-scattering can be expressed as an additional term as follows:

$$L_{in}(s, \theta) = \beta_{HG}(\theta)\ E_{sun}\ (1 - F_{ext}(s)) \tag{12}$$

where E_{sun} is the solar irradiance, $\beta_{HG}(\theta)$ is the Henyey-Greenstein phase function and θ the angle between the light and viewing directions.

By adding the extinction function F_{ext} and the light in-scattering function $L_{in}(s, \theta)$, the final equation for the attenuation rendering is obtained:

$$L(s, \theta) = L_0 F_{ext}(s) + L_{in}(s, \theta) \tag{13}$$

This is finally not a basic fog with an arbitrary fog color, only based on the distance between observer and objects, as it was often suggested in previous papers. It takes into account the light source and the type of particles in the air to render how light is scattered in the scene. Users can set and update in real-time the sun's position, the solar irradiance E_{sun} and the scattering distribution g to adjust the simulation.

The rainfall intensity parameter R is the common parameter of the Visible Streaks Region (Equation 5) and the Scattering Rain Phenomenon (Equation 9). Varying this parameter does not only change the density of kernels, but also makes the attenuation of visibility increase or decrease depending on its value. The two rain models introduced in this section simulate each one a region of a rainy scene, and the rainfall intensity parameter R gives us a correlation between the loss of visibility and the density of visible streaks.

4. Implementation and results

We will describe in this section our CPU and GPU implementations for streaks and attenuation rendering. User has the possibility to update in real-time the simulation by using the three main parameters : the camera's field of view fov, the exposure time T and the rainfall intensity R. The extinction coefficient is computed from the rainfall intensity to render light spreading. The single scattering model enables to change the scattering distribution or eccentricity g and the solar irradiance E_{sun} to control more precisely the simulation.

4.1. CPU

All of these calculations are done in CPU before the simulation starts. Some of them are recalculated in real-time if the rain parameters are updated.

The magnitude of the extracted streak patterns is computed through the 2d discrete Fast Fourier Transform. Only the fundamental frequencies are preserved and stored with their magnitude and phase information in files. The data are then sent only once to GPU for the rendering pass.

The density of streaks is precomputed for several projected sizes and for each raindrop diameter (Table A.5) according to rainfall intensity and camera's parameters.

Algorithm 1 Procedural generation of the rainfall texture

```
for each frame do
  recover the database of frequencies
  recover the number of kernels for each projected size and
  for each raindrop diameter
  for each pixel do
    for each projected size and each raindrop diameter do
      retrieve index of current cell
      initialize the pseudo-random number generator with
      the corresponding seed
      distribute kernels inside the cell
      for each kernel do
        compute parameters (depth, pattern, etc.)
        evaluate contribution to the pixel (reproduce the
        streak pattern by summing cosine waves using the
        database of frequencies)
        use a gaussian window to avoid edge effects
      end for
    end for
    sum and display the contribution of every projected size
    and every raindrop diameter
  end for
end for
```

4.2. GPU

A single post-processing GPU pass is used to generate the rainfall texture and another one for the attenuation of visibility.

The streaks generation reproduces a rainfall texture from the database of frequencies. Each streak is recreated by adding cosine waves. The density function gives the density of streaks in terms of the rainfall intensity and the camera's parameters.

As in texture synthesis by sparse convolution noise, the image is decomposed in a regular grid to avoid closest kernels search. Each grid cell has a unique seed number determined from texture coordinates. The pseudo-random number generator offers the possibility to always obtain the same random values for kernels inside a given cell.

The discretization of our screen in grid cell avoids evaluating the contribution of streaks too far from the current kernel. Examples of the rainfall textures are presented in Figure 7. Algorithm 1 presents the method.

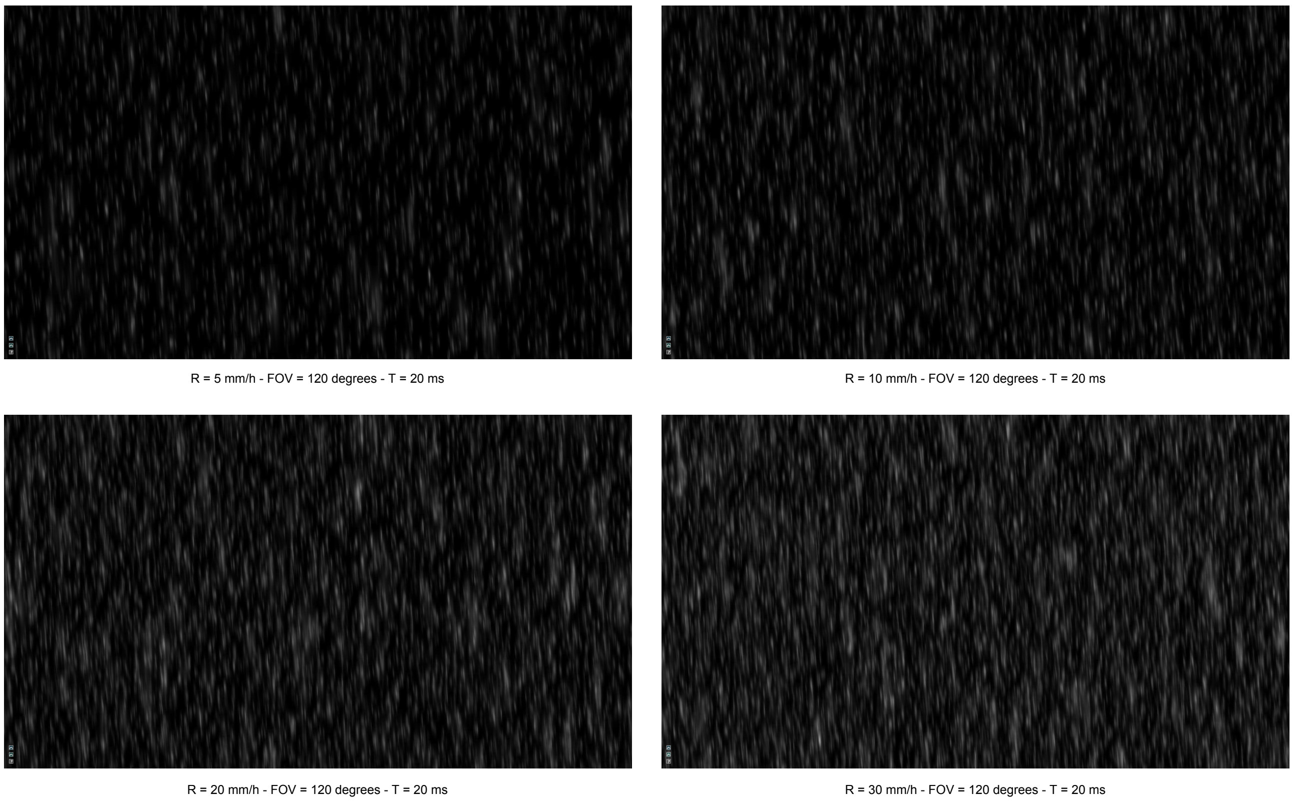
R = 5 mm/h - FOV = 120 degrees - T = 20 ms
R = 10 mm/h - FOV = 120 degrees - T = 20 ms
R = 20 mm/h - FOV = 120 degrees - T = 20 ms
R = 30 mm/h - FOV = 120 degrees - T = 20 ms

Figure 7: Generation of streaks using the sparse convolution noise technique, varying the rainfall intensity.

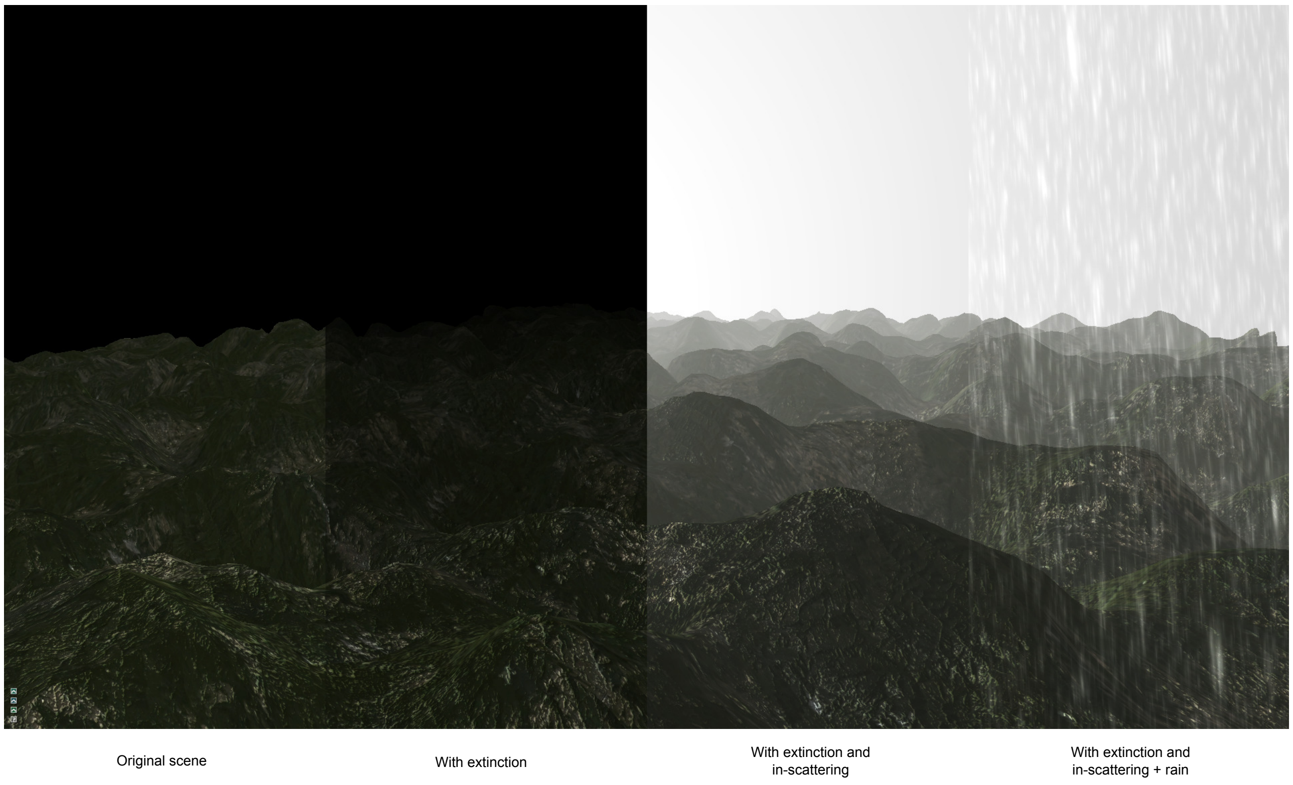
Original scene
With extinction
With extinction and in-scattering
With extinction and in-scattering + rain

Figure 8: Steps for attenuation rendering with a strong extinction.

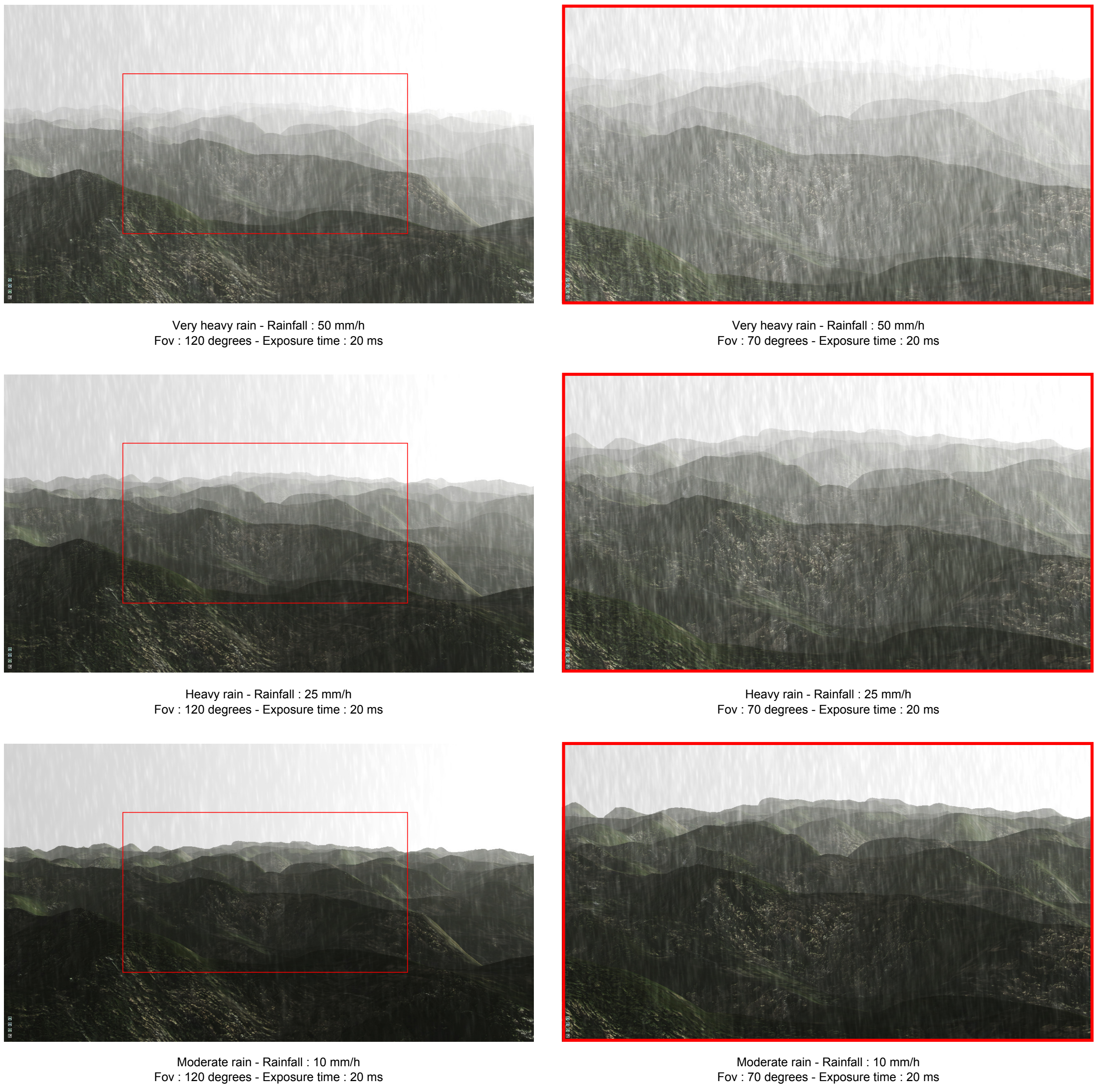
Very heavy rain - Rainfall : 50 mm/h
Fov : 120 degrees - Exposure time : 20 ms
Very heavy rain - Rainfall : 50 mm/h
Fov : 70 degrees - Exposure time : 20 ms
Heavy rain - Rainfall : 25 mm/h
Fov : 120 degrees - Exposure time : 20 ms
Heavy rain - Rainfall : 25 mm/h
Fov : 70 degrees - Exposure time : 20 ms
Moderate rain - Rainfall : 10 mm/h
Fov : 120 degrees - Exposure time : 20 ms
Moderate rain - Rainfall : 10 mm/h
Fov : 70 degrees - Exposure time : 20 ms

Figure 9: Rain and attenuation rendering varying the rainfall intensity and the field of view.

The attenuation rendering consists in using Equation 13. That equation is composed by two terms. The first term is related to the extinction phenomenon, including light absorption and out-scattering. Meanwhile, the second term corresponds to light in-scattering. Because our method uses a post-process approach, the scene and its depth map are required and stored in a buffer. The void pixels in the original scene correspond to the sky pixels. For sky rendering, we consider the atmosphere as a sphere around the scene. This simplification enables to avoid using a complex mesh. The shader computes the intersection between the viewing direction and the virtual sphere to evaluate the distance, and thus to simulate our atmosphere without adding complex calculations. Algorithm 2 describes the scene rendering, including sky. We present in Figure 8 the results that we obtained with the single scattering model to render the loss of visibility.

4.3. Results

The results are presented in Figure 9. The scene represents a vast mountainous terrain in order to perceive the attenuation of visibility far from the observer with various parameters. The more the rainfall intensity increases, the more the visibility decreases far from the observer and the more the number of visible streaks increases. In this figure, we can easily observe how raindrops scatter light, and thus produce a loss of visibility depending on the rainfall parameter. Moreover, the results show

Algorithm 2 Attenuation of visibility rendering

```
for each frame do
  store scene and depth map in buffers
  for each pixel do
    recover the depth of the pixel
    if current pixel = sky pixel then
      compute distance $s$ between the observer and the sky
    else
      compute pixel's position in the space world
      compute distance $s$ between the observer and the object
    end if
    compute the angle $\theta$ between the light direction $L$ and the viewing direction $V$
    recover the pixel color $L_0$
    compute $L_0 F_{ext}(s) + L_{in_scattering}(\theta, s)$
  end for
  display the scene
end for
```

that decreasing the field of view results in increasing the density of visible streaks.

A phenomenological approach makes us decide to represent streaks between 30 and 200 pixels. We made this empirical choice which offers visually convincing results. However the user could represent streaks with a larger or a lower projected size. That would mean simulating very close or very far raindrops. Regarding the raindrops under 30 pixels, we consider that they will be rendered indirectly by the attenuation of visibility. Moreover, this avoids simulating raindrops which are not visually significant.

The procedural streaks generation in Figure 7 provides non-repetitive and infinite rain patterns. The density function expresses accurately the density of visible streaks depending on rainfall intensity and the camera's parameters. Only visible raindrops are handled and occlusion by close objects can be taken into account by using on-screen depth information.

The memory footprint depends on the number of extracted streaks and the number of chosen frequencies. We do not treat precisely how many different extracted rain patterns are necessary to achieve a realistic final rainfall texture. In practice, we empirically observed that about a dozen streaks is sufficient to get visually interesting results. Considering a dozen streak patterns, each one requiring about 6 2D frequencies with amplitude and phase, memory requirements average around 1 Kb.

The performance of the system are presented in Table 1, depending on the field of view fov and rainfall intensity R. We set the exposure time T to 20 ms. Our GPU computations are made with a Nvidia Geforce GTX 980. The scene is composed by about 300k triangles and is running at 380 fps without the streaks and the attenuation of visibility. The performance of the system shows that a very heavy rain with an usual field of view value, 120 degrees, runs in real-time at 30 fps. This corresponds to 10 thousand distributed kernels. A light rain with the same field of view runs at 60 fps and corresponds to 1200 kernels. The SRP computations remain unchanged whatever the rainfall intensity and camera's parameters. Only the number of rendered streaks makes the number of FPS vary. Moreover the number of rendered streaks does not vary linearly with the number of FPS.

Table 1: Performance of the system in FPS.

fov \ R	2	5	25	75
120	60	48	37	30
100	55	45	35	27
80	50	42	31	24
60	45	39	28	22

4.4. Comparison with a particle system

We provide a rain simulation using a particle system for a comparison. Our extracted streak textures are mapped onto the particles which are simulated around the camera. We show in Figure 10 that the results are visually quite similar. The major feature of our rain model is its intrinsic quality to render only visible streaks as well as keeping a physical distribution of raindrops. This is not trivial by using a particle system, and thus requires a specific implementation to be achieved. Our method has also the advantage over particle systems that it does not directly depend on the number of particles rendered. Contrary to the vertex-bound methods, we are not limited in the number of streaks that we can render. The computation cost of the VSR is defined in a per-pixel basis according to the number of rendered streaks, computed from the field of view value and rainfall intensity. Moreover we propose a post-process technique in the image space with all the consequential benefits. This rain model is compute based while a particle system is rather fill rate based. The overdraw can be a limitation in usual real-time simulations such as video games using many particle system based effects. Our paper has the advantage of offering an alternative compute based method to render streaks and visibility.

5. Conclusion and future works

We introduce the first multiscale rain model which takes into account the local and global physical properties of a rainy scene. The streaks generation and the attenuation rendering, depending on the rainfall intensity and the camera's parameters, are based on the same distribution of raindrops in order to correlate the density of visible streaks and the loss of visibility in the scene.

However, some limitations remain in our method. Concerning the rain animation, we do not match the rainfall texture with lateral movements of the camera. Furthermore, we do not preserve coherency of raindrops position while the camera is moving. However, this does not impact the visual quality of our method as the velocity of raindrops is an order of magnitude faster than the camera movements.

A difficulty of the method lies in rendering the streaks in the VSR when the user looks upward or downward. Streaks cannot

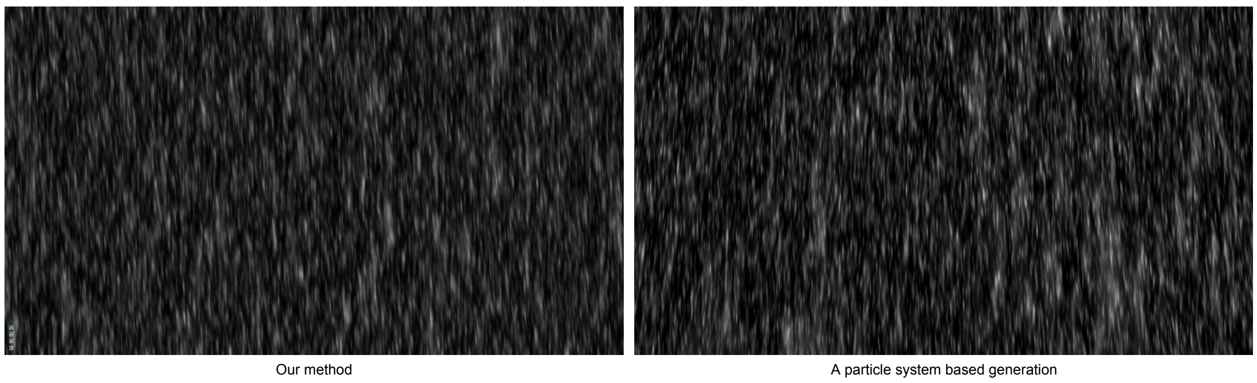

Figure 10: A visual comparison of particle system based methods and our image space streaks model. We simulate a heavy rain with about 8000 visible streaks. The particle system implementation runs at 200 FPS and our image space model runs at 30 FPS. The field of view fov and the exposure time T are respectively set to 120 *degrees* and 20 *ms*.

be generated and animated in the same way. This would imply to see the streaks far from the observer falling toward him. For the moment, we do not address this problem. We think that a specific analysis has to be done in order to evaluate how streaks are perceived according to the orientation of the camera, and perhaps build a new database of frequencies to reproduce and animate streaks according to the elevation angle.

The simulated cloud covered sky may be a little too uniform regarding the single scattering model. Thus it would be interesting to break this homogeneity. This could be achieved by adding real clouds in the scene, e.g. clouds could be correlated with the type of rain.

In this paper, we only focused on rain rendering in daylight, in a diffuse scene. The rain video was captured under a cloud covered sky, considering the illumination as globally diffused and so independent of the position of the sun. Our method renders a rain simulation under a similar illumination. The position of the sun in our real-time simulation enables to compute how light is scattered toward the camera, and so enables to physically simulate the behaviour of light in contact with particles in the air. Moreover, the artistic coefficient mixing the streaks and the scene approximates the refraction for each streak. It is actually necessary to capture streaks patterns under a covered sky to guarantee a correlation between both models.

Our method may be able to hide streaks occluded by closer objects. A distance in the range $[z_{sap_i-\delta sap}; z_{sap_i+\delta sap}]$, figure 3, could be computed for each generated streak, for each projected size and each size of raindrop. This would result in a depth information for each streak computed on-the-fly. Moreover, we think that we could take account for spaces with cover from rain as Rousseau et al [6]. This is a visibility problem that could be solved by using a global depth map, as in shadow maps techniques.

Some previous works [2] [10] take into account several light sources in the scene. As this increases the realism of the scenes, such phenomenon must be integrated in our future rain model. This will lead to more varied climatic conditions. Moreover, numerous effects such as water splashes on the ground, change of appearance of wet objects and light sources could be added to our model to increase realism. The challenge will be to solve these problems using a coherent multiscale approach.

Appendix A. Distribution of raindrops in m^{-3}

Different values of rainfall intensity are proposed in Table A.2.

Table A.2: Rainfall intensity R.

Rainfall intensity	R ($mm\ hr^{-1}$)
Light	2
Moderate	5
Heavy	25
Very heavy	75

By using Equation 9, Table A.4 gives the extinction coefficient β_{ext} in km^{-1} for each rainfall intensity R in $mm\ h^{-1}$.

Table A.4: Extinction coefficients β_{ext} in km^{-1} for each rainfall intensity R

Rainfall	R ($mm\ h^{-1}$)	β_{ext} (km^{-1})
Light	2	0.50
Moderate	5	0.92
Heavy	25	2.70
Very heavy	75	5.63

We propose Table A.3 in case that the user wants to only use specific rainfall intensities, hence avoiding CPU calculations to evaluate the distribution of raindrops. The distribution in Table A.3 is computed using Equation 6 from rainfall intensity in Table A.2 and the size of raindrops in Table A.5 according to Bentley [22].

Table A.3: Distribution of raindrops in terms of rainfall intensity and size of raindrops, computed from Equation 6. The values are expressed in (m^{-3}).

	Very small	Small	Medium	Large	Very large
Light	448	225	46	0	0
Moderate	613	380	122	0	0
Heavy	939	772	463	6	1
Very heavy	1168	1111	917	29	6

Table A.5: Diameter of raindrops D according to Bentley [22].

Size of raindrops	D (mm)
Very small	[0.51 ; 0.85[
Small	[0.85 ; 1.41]
Medium	[1.59 ; 3.18]
Large	[3.63 ; 5.08]
Very large	]5.08 ; 8.47]

Appendix B. Distance function $z(D, sap, fov, T, h)$

The velocity of a raindrop can be expressed as:

$$v(D) = 200 \times \sqrt{\frac{D}{2000}} \tag{B.1}$$

where D is the diameter in *mm*. $v(D)$ is expressed in $m\ s^{-1}$. Using Equation B.1, we compute the size of a streak as:

$$sbp(D,T) = \frac{D}{1000} + v(D) \times \frac{T}{1000} \tag{B.2}$$

where the function $sbp(D,T)$ gives the height of a streak in meters taking into account the exposure time of the camera.

A typical OpenGL projection computation can be expressed as follows:

$$\begin{bmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{zFar+zNear}{zNear-zFar} & \frac{2\times zFar\times zNear}{zNear-zFar} \\ 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x_{clip} \\ y_{clip} \\ z_{clip} \\ w_{clip} \end{bmatrix} \tag{B.3}$$

where *aspect* is the aspect ratio, *znear* and *zfar* define the frustrum view and:

$$f = \frac{1}{tan(\frac{fov}{2})} \tag{B.4}$$

We are only interested in the computations in the y dimension. Indeed, only the height of the streaks is important to us. We will deduce the width by a simple ratio.

$$y_{clip} = f \times y \tag{B.5}$$

$$w_{clip} = z \tag{B.6}$$

Then the normalized coordinates in the viewport are obtained by dividing each dimension by w_{clip}.

$$y_{viewport} = \frac{y_{clip}}{w_{clip}} \tag{B.7}$$

sap corresponds to the size of a streak after projection in pixels. We have to remind that the viewport coordinates are in the range of $[-1\ ;\ 1]$.

$$sap = \frac{y_{viewport} \times h}{2} \tag{B.8}$$

sap is reformulated as:

$$sap = \frac{f \times y \times h}{z * 2} \tag{B.9}$$

y can be replaced with the function $sbp(D,T)$ to obtain the size of a streak before projection. Thus we can express the distance z as:

$$z(D, sap, fov, T, h) = \frac{sbp(D,T) \times h}{2 \times sap \times tan(fov \times 0.5)} \tag{B.10}$$

where h is the y-axis resolution, fov is the camera's field of view and T is the exposure time.

[1] Creus C, Patow G. R^4: Realistic rain rendering in realtime. Computers & Graphics 2013;37(1-2):33–40.

[2] Garg K, Nayar SK. Photorealistic rendering of rain streaks. In: ACM SIGGRAPH 2006 Papers. SIGGRAPH '06; New York, NY, USA: ACM. ISBN 1-59593-364-6; 2006, p. 996–1002. doi:10.1145/1179352.1141985.

[3] Tatarchuk N, Isidoro J. Artist-directable real-time rain rendering in city environments. In: Chiba N, Galin E, editors. NPH. Eurographics Association. ISBN 3-905673-38-X; 2006, p. 61–73.

[4] Wang N, Wade B. Rendering falling rain and snow. In: ACM SIGGRAPH 2004 Sketches. SIGGRAPH '04; New York, NY, USA: ACM. ISBN 1-58113-896-2; 2004, p. 14–. doi:10.1145/1186223.1186241.

[5] Wang L, Lin Z, Fang T, Yang X, Yu X, Kang SB. Real-time rendering of realistic rain. In: In ACM SIGGRAPH (Sketch. 2006,.

[6] Rousseau P, Jolivet V, Ghazanfarpour D. Gpu rainfall. J Graphics Tools 2008;13(4):17–33.

[7] Tarik S. Rain. Nvidia White Paper 2007;.

[8] Puig-Centelles A, Ripolles O, Chover M. Creation and control of rain in virtual environments. Vis Comput 2009;25(11):1037–52. doi:10.1007/s00371-009-0366-9.

[9] Puig-Centelles A, Sunyer N, Ripolles O, Chover M, Sbert M. Rain simulation in dynamic scenes. IJCICG 2011;2(2):23–36.

[10] Changbo W, Wang Z, Zhang X, Huang L, Yang Z, Peng Q. Real-time modeling and rendering of raining scenes. Vis Comput 2008;24(7):605–16. doi:10.1007/s00371-008-0241-0.

[11] Langer MS, Zhang L, Klein AW, Bhatia A, Pereira J, Rekhi D. A spectral-particle hybrid method for rendering falling snow. In: Keller A, Jensen HW, editors. Rendering Techniques. Eurographics Association. ISBN 3-905673-12-6; 2004, p. 217–26.

[12] Lewis JP. Algorithms for solid noise synthesis. SIGGRAPH Comput Graph 1989;23(3):263–70. doi:10.1145/74334.74360.

[13] Cook RL. Stochastic sampling in computer graphics. ACM Trans Graph 1986;5(1):51–72. doi:10.1145/7529.8927.

[14] Gilet G, Sauvage B, Vanhoey K, Dischler J, Ghazanfarpour D. Local random-phase noise for procedural texturing. ACM Transactions on Graphics 2014;33(6):N/A. doi:2661229.

[15] Marshall JS, Palmer WM. The distribution of raindrops with size. J Meteorol 5 1948;:165–6.

[16] Garg K, Nayar S. Vision and Rain. International Journal on Computer Vision 2007;:1–25.

[17] Nayar S, Narasimhan S. Vision in Bad Weather. In: IEEE International Conference on Computer Vision (ICCV); vol. 2. 1999, p. 820–7.

[18] Jarosz W. Efficient monte carlo methods for light transport in scattering media. Ph.D. thesis; UC San Diego; 2008.

[19] Atlas D. Optical extinction by rainfall. J Meteor 1953;(Volume 10):486488.

[20] Hoffman N, Preetham AJ. Rendering outdoor light scattering in real time 2002;.

[21] Cozman FG, Krotkov E. Depth from scattering. In: CVPR. IEEE Computer Society; 1997, p. 801–6.

[22] Bentley WA. Studies of raindrops and raindrop phenomena. Mon Wea Rev 1904;:450–56.