Enhanced Technique for Realistic Rain Rendering

Khushbu Sharma
Research Scholar, NITTTR, Chandigarh, India.

Dr. Maitrayee Dutta
Associate Professor, Department of Computer Science, NITTTR, Chandigarh, India.

Abstract- Rain simulation is a challenging problem due to the variety of different phenomena that need to be computed, all of them with complex physical evolutions. These phenomena include raindrops, splashes on the ground, fog, clouds, lightning, etc. Since water in rain is present in a wide variety of forms, different simulation algorithms present in order to adapt each condition. In this paper presented the design for rendering of rain towards considering the city environment in real time. A technique is being used to generate rain, which is being used in many video games, particles falling vertically from the sky in the form of translucent white streak. The proposed approach is both simpler and faster to animate, yet allow performing various kinds of adaptive sampling of the regions that need to be simulated and rendered. Interaction of the raindrops with the whole scene is also considered. It avoids drop's penetrating geometry and will also provoke splashes when collisions are detected, a phenomenon that gives good visual clues to identify rainy weather. Real-time rain rendering has become a major topic in natural phenomena simulation.

Keywords – Natural phenomena, Rain, Real-time rendering

I. INTRODUCTION

One important aspect of artistic outdoor scene filming is the need to choose specific weather conditions in order to properly trigger the audience reaction. Depending on which mood the artist desires to transmit, different weather conditions must be used. For instance, sunny days are typically associated with alert and cheerful states, while cloudy and rainy conditions are felt more oppressive and may enhance a sense of unrest. Special Effects, especially atmospheric effects such as rain, cloud, snow and other outdoor scenes in interactive applications (video games, training systems etc.) are important in creating realistic environments.

Rain is often used in movies and animations to express the mood of a scene. Filming rain scenes is, however, a laborious and expensive task that requires setting up sprinklers and light sources over a large physical area. The shooting of a single rain scene can take several days. Due to the high costs involved, it is often impractical to include rain scenes in small-budget movies. For the reasons, a simple computational method for rendering of rain is highly desirable. It would provide a convenient and inexpensive means to add rain effects in movies and animations. In addition, it would allow a film-maker to control the visual effects of rain during post-production. Rain rendering can also be used to add visual realism in other graphics applications such as games.

Rendering weather effect such as rain, fog, snow or mist on image or video is always a challenging task and requires state of the art hardware and software techniques. Rain rendering process on image or video is often linked to high cost, as they require special hardware such as profilometer and stereo cameras. Leaving apart the special hardware requirement to produce rain rendering effects, a rain rendering process requires high computational power as the process has to be a random process with proper accountancy for depth of the scene lighting angle and weather effects. Taking into account such computational and cost related challenges rain rendering process is often limited to trained professional with high performance computing systems.

In this paper, the developed methodology for rendering rain to a scene with minimum computational requirement has been discussed. The developed methodology is a low cost rain rendering technique that will simulate rain effect. The developed methodology requires very less computation power and almost eliminates the need of special hardware such profilometer and stereo cameras. The technique utilizes a rain strokes database to effectively render rain effect on the scene.

II. Related Work

Rain as a whole consists of a plethora of different phenomena, most of them related to water, how it moves and how it affects lighting. Since water in rain is present in a wide variety of forms, these phenomena require radically different simulation algorithms in order to adapt to each condition. Many different techniques have been developed in recent years to simulate rain, most of them focusing on a concrete subset of rain phenomena.

Kshitiz Garg and Shree K. Nayar have presented a rain streak simulation model and rain rendering algorithm. The first part consists of an analysis of a raindrop oscillation model developed in atmospheric sciences in order to infer the parameters that best match the reality. The analysis is done by capturing of real rain streaks and visually comparing them to the synthetic images produced by their software.

Kshitiz Garg, Gurunandan Krishnan and Shree K. Nayar have presented another rain phenomenon: splashing of water drops. They use a real setup to capture a wide variety of splashes in order to empirically infer a mathematical model. This model must depend on the inclination of the surface relative to the raindrop direction, the material of this surface and the size and velocity of the droplet. The rendering algorithm is also image-based and takes as input an image (or video), a coarse depth map of the image (video), a partition of the image (video) in different materials and the lighting and camera configurations. They also check collisions in order to add splashes using the probability distributions of their model.

Niniane Wang and Bretton Wade have presented a refined technique. Instead of a textured quad in front of the camera, two cones that encompass the observer are used and four artist-generated rain (or snow) textures for the cones. The cones are tilted to adjust for camera movement, the textures are elongated to simulate motion blur and also scrolled to mimic the rain fall. This scroll is done at different speeds for the 4 textures in order to simulate streak's depth with parallax.

Lifeng Wang, Zhouchen Lin, Tian Fang, Xu Yang, Xuan Yu and Sing Bing Kang have been analysed the optical properties of spherical raindrops. Even though real raindrops are not spherical, this simplification allows them to derive a closed formula for their algorithm. In a pre-processing step, they compress the environment map and the transfer function of their model using spherical harmonics. To render the raindrops they use real video analysis in order to extract real rain streaks that are later used as textures for the particle system. They also add to the output renders homogeneous fog, light glows approximated by Gaussian blurring and rain splashes designed by an artist. The raindrops and the splashes are uniformly distributed in the scene by their particle system.

Pierre Rousseau, Vincent Jolivet, and Djamchid Ghazanfarpour have been used raindrop refraction as the main defining factor of raindrop colouring. They start by analysing the optic properties of the raindrop and conclude that reflections are limited to its silhouette and thus can be neglected without reducing image quality. In order to model refraction in real time, at a pre computation step they generate a texture mask that determines the direction of the refracted viewing vector for a quasi-spherical raindrop. This vector is later used to index a texture storing a wide field of view render of the background. The particle system is stored in a texture where each Texel represents a particle's position, information that is updated per frame. During rendering time the particles are expanded to quad billboards and their colour is computed using the mask: either the quad's Texel lies outside the drop and the background is directly outputted, or the mask's refraction is used to perturb the direction of view and compute which part of the background is visible. To add motion blur they change the particles to vertical streaks and render a few drops along this streak, blending them together to simulate the camera's integration time.

Sarah Tariq presented that rain is rendered in real-time as quads textured with a simplified version of the image database presented in [5]. By doing so, they achieve realistically looking raindrops at the cost of texture lookups. The animation is done by using the graphics pipeline to update the raindrop positions, then GPU state is changed and the actual render proceeds. This render creates a quad for each particle and computes its lighting parameters needed to use the simplified database. To enhance the visualization, they also use fog to add glows around light sources and change the reflectance of the surfaces.

Anna Puig-Centelles, Oscar Ripolles, and Miguel Chover presented a new rain rendering algorithm based on a particle system. Its setting is a simplified raining area defined by an ellipse and a rain container where actual particles are simulated. This container is a semi-cylindrical volume whose size is big enough to allow the observer to rotate and move a certain amount without needing to compute new particles to populate the container. This is an important difference to the previous methods of [12] and [11], where recompilation is needed when the camera changes. The density of the particles far away from the observer is lower than that of the particles close to it. This adaptive scheme, along with a technique that changes the particle's size depending on the distance, allows simulating heavy rain with fewer particles than the previous methods.

Natalya Tatarchuk, presented the ATI R demo, is a set of techniques that approximate the rain phenomena without trying to use physically correct simulations. Rain streaks are animated with a particle system and rendered using an image-based algorithm that composes the final image with a single pass. Dripping raindrops are set up by an artist

and rendered as billboards using the same shading algorithm as normal rain streaks. Splashes are uncorrelated to the actual drop collisions, being instead randomly started on the ground. In heavy rain situations, objects receive a huge amount of drop collisions, each of them creating splashes of small particles that surround the object. This produces a halo like effect along the object's silhouette. To simulate this phenomenon a technique similar to the ones used to render fur is used: series of semi-transparent shells are rendered encompassing the object. The rendering includes reflection of the objects in front of the surface and refraction of the scene behind it. This part of the scene, the one behind the glass, receives shadows casted by the water and an approximation of caustic highlights. The technique used for this last phenomenon builds upon the algorithm of [13], but it is simplified and adapted to high performance GPU evaluation and rendering.

Here are the brief summary of the explained algorithms. They are defined as different sections, i.e. the off-line algorithms, the simple approach, particle systems and multiple rain phenomena methods. For each of them its main properties are highlighted; which rendering primitive is used for the simulation of raindrops, whether it is a real-time algorithm, use of the GPU to accelerate the rendering or the simulations, rendering of raindrops, raindrop particles removal when the ground is hit or when the raindrops are behind solid occludes, wind, rainbow and lightning simulations, reflection and refraction of the water (including wet surfaces or raindrops), participating media that modifies the illumination, simulation of splashes on the ground, adaptive schemes for Level-Of-Detail optimizations, and simulation of ripples and dripping water. None of the methods simulate at the same time all the phenomena. [5] is the most comprehensive one, but at the cost of a highly complex implementation and limiting camera movement. [10] is also extensive, but few details are given in the paper, so its evaluation is hardly possible. The simplistic approach in [4] only simulates the effect of raindrops, which is the only phenomenon common to all the presented algorithms. [8] is able to achieve good visual results, but requires as input real rain video in order to extract the streak textures. Interaction of the rain with the scene is one of our important objectives, since it avoids that rain penetrates geometry. In [11] they do not consider realistic streaks either; instead they use uniform transparent materials. Finally, let us note that the work done in [13] is similar to our approach, but, apart from the missing collisions, it also uses a simplified illumination computation.

III. Proposed Rain Rendering Method

Here discussed the proposed method of rain rendering technique which takes input as the weather parameter to develop realistic rain mask, to perform simulation, only wind is taken as a parameter that can distort and can change in the direction of fall of rain drops. Secondly the methodology removes noise and unwanted particle reflections from the database and prepares a plain reduced noise rain stroke. The strokes are truncated as per the ration of height to width of the captured scene on which the rain is to be rendered. When the strokes are ready a rain mask is prepared using random effects. The rain mask is prepared as to take into account the depth of the scene, the depth of the scene is passed manually. Once the mask is ready, the algorithm than shifts the histogram of the image to introduce cloudy effect to the scene and finally delivers a rain rendered scene. Detailed description of the methodology for rendering of rain has been Figure 1 shows the flow diagram of the methodology developed.

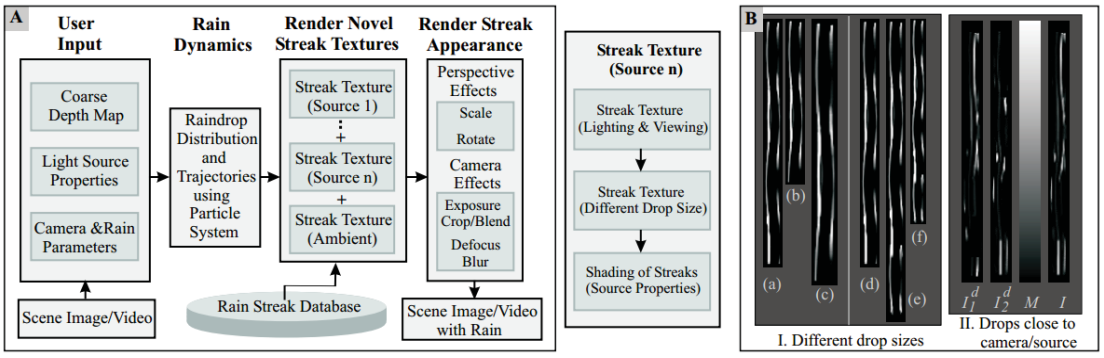

Figure 1. Process flow methodology for rain rendering

A. *Database Preparation*

The process of rain rendering starts with extraction of rain strokes from the image, to extract rain strokes from the scene requires camera parameter and lighting parameters which cannot be obtained in our case due to missing

hardware i.e. profilometer. Hence to compensate it, in our case the algorithm starts utilizing the already extracted rain strokes from a detailed library composed by Columbia university.

The library of strokes captures the effect of lightening parameters (directions), camera viewing direction and oscillation parameters on the appearance of rain strokes. 0 degrees and azimuthal angle of the source is varied from 10 degrees to 170 degrees both with steps of 20 degrees. To capture the effects of the viewing direction, we varied camera elevation angle from 0 degrees to 80 with steps of 20 degrees. For each light and view configuration we render 10 images to capture the appearance variation due to oscillations. Thus the database samples the streak appearance in the 4 dimensions.

The database streaks are rendered for drop of size 1.6mm and camera exposure time of 1/60 sec. It also includes streaks rendered under ambient lighting. The streaks are rendered as 16-bit monochrome images with maximum resolution of 32 times 1050 pixels. In total, the database includes 3150 streak images. The database also includes at 3 additional (lower) resolutions with streak widths of 16, 8 and 4, to avoid artifacts due to severe down-sampling when rendering streaks far from the camera. The complete database is about 50 MB in size. The streaks are in .png format. Figure 1 & 2 graphical representation of the initial steps to rain render using developed algorithm.

Figure 2. Preparation of Rain Stroke Database

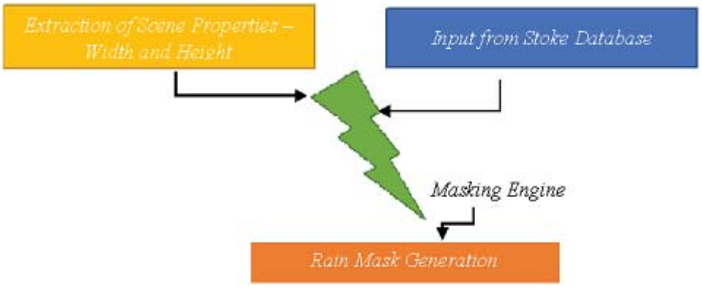

Figure 3. Rain Mask Generation Algorithm

As shown in figure 2, the rain strokes are directly picked up from database, but the templates are noisy and contains too many dark pixels i.e. DN value approaching zero. The template is converted in to "double" data format from Unsigned Integer of scale 0-255. The user is asked to enter the rotation parameter that will give an effect of wind parameter the range should be fed in optimized, too much angle of rotation on clockwise or anticlockwise will result in non-realistic scene and scene will appear as distorted form of rendering. An angle ranging from 30 – 0 - -30 is best suited for our developed algorithm.

Once the effect due to wind has been introduced, the template is ready for noise reduction and works on keeping the best suited one in the suit while removing the odd ones out. The pixels having DN value less than 0.2 (double data format) is removed while greater than 0.2 was kept the decision boundary was decided after studying the histogram of original as is templates and the majority of the pixels that belonged to darker region was ranging from DN values 0-0.2, as shown in figure 3a and 3b. Figure 4 shows the pixel after wind parameter was provided while Figure 5 shows the final template that will now be utilized by masking engine.

Figure 3a. Rain Drop Template from Point Source Database

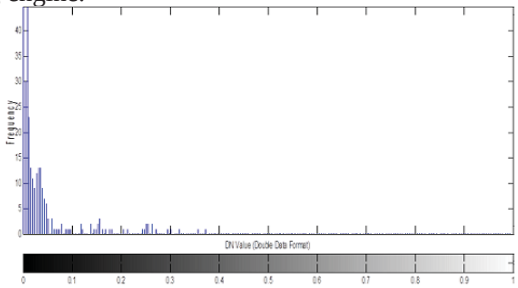

Figure 3b. Histogram of the Rain stroke in Figure 3a

Figure 4. Rotated Rain Stroke at an Angle of +5 degrees

Figure 5. Rain Stroke after Noise Removal.

The same process is also applied when rain is to be render using strokes from environment lightning conditions. As the wind parameter is passed, the masking engine is all setup to start for preparation of rain mask.

B. Mask Preparation

To effectively handle the memory limitation parameters the mask size is kept fixed to 170 Row and 210 Columns. This size is optimum and is determined by constant hit and trial method. The following number of rows and columns prevents the system to not to run out of memory while performing calculations as the rain has to be rendered on each frame hence the system has to develop a separate mask for each frame. A unique mask is required for each frame as rainfall is completely a natural phenomenon with high degree of randomness, hence to simulate rainfall conditions the rain mask needs to be completely unique and that can only be achieved by simulating random process. Figure 6 shows the pictorial representation of parameters of masking engine. The depicted parameters acts like gears of engine, hence right interlocking is necessary for smooth rendering.

The engine takes input as Rain Density Constant and Frame Constant. The random phenomenon is entirely dependent upon processor clock timing.

- Rain Density Constant: - The constant takes care of how much dense the rainfall should appear, in other words it can be defined as heaviness of rainfall in a particular scene.
- Frame Constant: - The constant takes care of how much rain mask frame has to be generated. Since video a frame per second, hence as rain to appear falling each frame has to be masked with the developed rain mask. Frame constant also determines the play time of the video.

Figure 6: Interlocking of essential parameters of masking generation

1) Masking Engine

The masking engine takes input as rain density constant and frame constant to perform mask generation task. Figure 7 depicts the flow diagram of working of a masking engine. The outer loop controlling the number of frames and passed frame constant while the inner loop takes rain density parameter and for each frame. By varying the rain density constant one can increase or decrease the heaviness of rain in scene. Presently the inner loop is by default set to run till number of columns present in image on which the rain effect is to render.

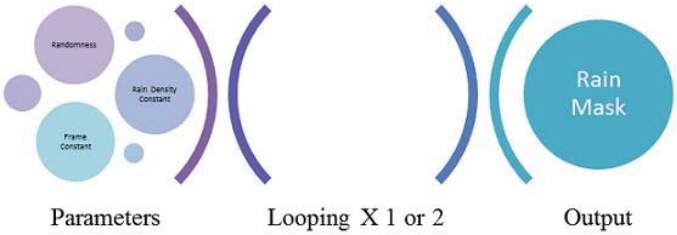

Figure 7: Generation of Rain Mask. Initials RD represents rain density

The output of the process is a rain mask that can be characterized as: Number of frames in mask equals to Frame Constant, Play speed will of the rain mask will depend upon selected frame rate or Frames Per Second abbreviated as FPS and X Rain Density Constant times differentiates each frame with one another and thus mimics closely to the natural rain phenomenon.

The Looping procedure can be performed for single time or for two times. If the looping procedure is performed only once, than the depth effect in the scene cannot be introduced. The rain strokes will be of the same length and will not take depth into account. Such rainy effect is useful for when introducing rendering on animated scenes. When the procedure is operated twice the depth effect can be introduced. After observing many video shots of natural rain it was deduced that the camera captures rain stroke nearer to the lens as long rain strokes with density of these long rain stroke as rare in nature. While rain strokes away from the camera lens become shorter in height and are more in number since the field of view (FOV) of the camera gets increased. The deeper the depth of the scene the shorter the height of the rain stroke with greater the rain density of these rain stroke.

C. Brightness Reduction

Brightness can be defined as the "attribute of a visual sensation according to which an area appears to emit more or less light. Rain is a natural phenomenon that is often accompanied by dark cloud that lowers the brightness of the scene making it dull and gloomy. Hence to introduce such effects into the scene the brightness of the scene needs to be reduced down significantly so as it would look like a naturally occurring rain phenomenon. In the developed methodology, two ways of reducing the brightness of the scene is adopted, namely:

1. Direct Reduction Approach: Features of the approach are as follows and block diagram shown in figure 8:
   a. Computationally less memory requirement
   b. Faster calculations
   c. Output seems to be realistic but dullness seems to affect the images in patches sometimes.

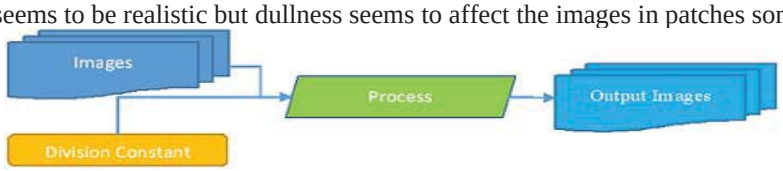

Figure 8: The direct reduction approach

2. HSV, Value (V) Reduction Approach: - Features of the approach are as follows and block diagram shown in figure 9:
   a. Computationally memory demanding
   b. Algorithm takes time to execute as transformation to HSV from RGB, followed by division of V by a certain specified constant and the converting HSV back to RGB.
   c. Realistic output, seems to affect the entire scene, making the scene gloomy and introducing a definitive dullness in the scene.

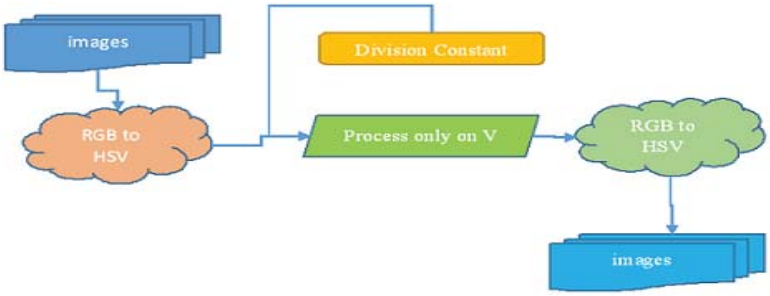

Figure 9: The HSV, Value (V) reduction approach

The process of reduction of brightness of the scene using the following techniques:

1. Direct reduction method: The color image consists of three bands that is red, blue and green. In the direct reduction method, the process tab directly divides the DN value of each of the pixel by 3. Thus reducing the DN of each band by a factor of three.
2. HSV, Value (V) Reduction: The color image consists of three bands that is red, blue and green. The bands are converted to HSV format. The H and S value matrix are kept as it is and under the process tab the V matrix is either scaled down by dividing it by a specified constant, generally kept "1.5–3" or by reducing a constant value from every matrix element present in V matrix. The obtained band is now considered as the new V matrix for the existing combination of H and S. Now these three bands are converted back to RGB format to obtain a naturally gloomy scene.

D. Mask Implementation

Implementation of rain mask over the obtained image is the final process of the rain rendering. The mask implementation process is shown in figure 10. The rain mask obtained is very high in intensity and if is applied as in on the scene that white strokes will be visible and thus making the scene look like an artificial scene. To render the mask so that the scene appears like a natural looking phenomenon the intensity of the rain mask is reduced using a mask reduction constant and which reduces the intensity of the rain mask. Once the rain mask intensity is reduced each rain mask pixel is then added to the scene pixel. The process makes the rain strokes in the scene to appear as a blurry transparent sheet thus allowing the background to appear as if we are looking through the rain stroke. Thus the scene appears to be an original one rather artificial one.

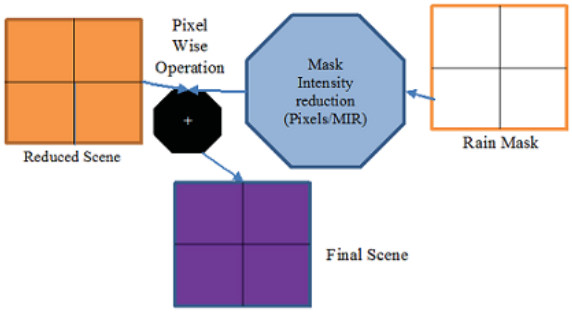

Figure 10: Final Mask Implementation Process

IV.CONCLUSION

Here focuses upon the implementation of Rain rendering on an ordinary scene. The process was implemented using a custom developed methodology and the available rain strokes from the library. We have presented a new approach that builds on existing state of the art methods ([27] & [7]) and improves them by considering extra significant phenomena. In particular, we are able to use the entire database of [27] instead of simplifying the illumination parameters as done in [7], thus achieving high quality images with complex illumination effects. We have proposed a model for rain streak appearance that captures the complex interactions between the lighting direction, the viewing direction and the oscillating shape of a falling drop. We have used this model to create a useful database of rain streaks that includes the wide variability in streak appearance. This database stands on its own as a contribution as it can be used by any image-based algorithm for rain rendering. Finally, we have developed a rain rendering algorithm that can be used to add realistic rain in images and videos. The algorithm requires minimal human input, provides a high degree of control, and is efficient. It provides an inexpensive way to add rain in movies and animations. The developed algorithm is fast and efficient and fully dependent on user input, but is constrained by smaller resolutions and is efficient to render rain on images or photograph of not more than 720X1024. Hence in future the simulation can be carried out using parallel processing to increase the size of the image on which rain can be rendered.

REFERENCES

[1] M. S. Langer, L. Zhang, A. W. Klein, A. Bhatia, J. Pereira, and D. Rekhi, "A spectral-particle hybrid method for rendering falling snow", *In Rendering Techniques 2004 (Eurographics Symposium on Rendering). ACM Press, june 2004.*

[2] N.Wang and B.Wade, " Rendering falling rain and snow," *In ACM SIGGRAPH 2004 Technical Sketches Program, 2004.*

[3] N. Tatarchuk, " Artist-directable real-time rain rendering in city environments," In Proceedings of the 2006 Symposium on Interactive 3D graphics and games Poster, page 30, New York, NY, USA, 2006. ACM Press.

[4] P. Kipfer, M. Segal, and R. Westermann, “Uberflow : A GPU-based particle engine ", In Proceedings of ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics hardware, pages 115-122, 2004.

[5] S. Tariq, "Rain", Technical report, Nvidia, 2007.

[6] K. Garg and S.K. Nayar," Photorealistic Rendering of Rain Streaks ", ACM Trans. on Graphics (also Proc. of ACM SIGGRAPH), July 2006.

[7] Biswarup Choudhury, Deepali Singla, Sharat Chandran, Fast Color-Space decomposition based Environment Matting, In Proceedings of the 2008 symposium on Interactive 3D graphics and games, pages 1-1. ACM, 2008.

[8] Delta3D. www.delta3d.org, last viewed on June, 05 2008.

[9] Rudy Darken, Perry McDowell, and Erik Johnson. Projects in VR: the Delta3D open source game engine. Computer Graphics and Applications, IEEE, Vol.25, Iss.3, May-June 2005.

[10] Open Scene Graph. http://www.openscenegraph.org/, last viewed on June, 05 2008.

[11] Open Dynamics Engine. http://www.ode.org/, last viewed on June, 05 2008.

[12] 3D Character Animation Library. https://gna.org/projects/cal3d/, last viewed on June, 05 2008.

[13] OpenAL,Cross-Platform 3D Audio. http://www.openal.org/, last viewed on June, 05 2008.

[14] GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html, last viewed on June, 05 2008.

[15] Falling Rain Particle Effect. http://www.delta3d.org/filemgmt/visit.php?lid=80, last viewed on June, 05 2008.

[16] Pierre Rousseau, Vincent Jolivet, Djamchid Ghazanfarpour. Realistic real-time rain rendering. Computers & Graphics 30, 4 (2006), 507-518. special issue on Natural Phenomena Simulation.

[17] Ross, O.N. (2000) Optical Remote Sensing of Rainfall Microstructures, Freie Universit¨at Berlin, Fachbereich Physik, Diplom Thesis, 134pp.

[18] Chuang C, Beard KV. A new model for the equilibrium shape of raindrops. Journal of Atmospheric Science, 44(11):1509-1524, 1987.

[19] Chuang C, Beard KV. A numerical model for the equilibrium shape of electrified raindrops. Journal of Atmospheric Science, 47(11):1374-89, 1990.

[20] K. V. Beard and C. Chuang. A new model for the equilibrium shape of raindrops. *J. Atmos. Sci.*, 44:1509–1524, 1987.

[21] A. C. Best. The size distribution of raindrops. *Quarterly Journal of the Royal Meteorological Society*, 76(327):16–36, 1950.

[22] George Borshukov. Making of the superpunch. In *SIGGRAPH ’05: ACM SIGGRAPH 2005Courses*, page 19, New York, NY, USA, 2005. ACM. doi: http://doi.acm.org/10.1145/1198555.

[23] Wang Changbo, Zhangye Wang, Xin Zhang, Lei Huang, Zhiliang Yang, and Qunsheng Peng. Real-time modeling and rendering of raining scenes. *Vis. Comput.*, 24(7):605–616, 2008. ISSN 0178-2789. doi: http://dx.doi.org/10.1007/s00371-008-0241-0.

[24] C. Chuang and K. V. Beard. A numerical model for the equilibrium shape of electrified raindrops. *J. Atmos. Sci.*, 47:1374–1389, 1990.

[25] Kshitiz Garg and Shree K. Nayar. Photorealistic rendering of rain streaks. In *SIGGRAPH ’06: ACM SIGGRAPH 2006 Papers*, pages 996–1002, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1179352.1141985.

[26] Kshitiz Garg, G. Krishnan, and Shree K. Nayar. Material Based Splashing of Water Drops. In *Proceedings of Eurographics Symposium on Rendering*, Jun 2007.

[27] Ross Gunn and Gilbert D. Kinzer. The terminal velocity of fall for water droplets in stagnant air. *Journal of Meteorology*, 6:243–248, 1949.

[28] Daniel Herman. Rainman: Fluid pseudodynamics with probabilistic control for stylized raindrops. In *Conference Abstracts and Applications ACM SIGGRAPH 2001*. ACM, 2001.

[29] Kazufumi Kaneda, Shinya Ikeda, and Hideo Yamashita. Animation of water droplets moving down a surface. *Journal of Visualization and Computer Animation*, 10(1):15–26, 1999.

[30] C. Magono. On the shape of water drops falling in stagnant air. *J. Meteorol.*, 11:77–79, 1954.

[31] H. R. Pruppacher and K. V. Beard. A wind tunnel investigation of the internal circulation and shape of water drops falling at terminal velocity in air. *Q. J. R. Meteorol. Soc.*, 96:247–256, 1970.

[32] Anna Puig-Centelles, Oscar Ripolles, and Miguel Chover. Creation and control of rain in virtual environments. *Vis. Comput.*, 25(11):1037–1052, 2009. ISSN 0178-2789. doi: http://dx.doi.org/10.1007/s00371-009-0366-9.

[33] Oliver N. Ross. Optical remote sensing of rainfall micro-structures. Freie Universität Berlin Master’s Thesis, 2000.

[34] Oliver N. Ross and Stuart G. Bradley. Model for optical forward scattering by nonspherical raindrops. *Applied Optics*, 41(24):5130–5141, 2002.

[35] Pierre Rousseau, Vincent Jolivet, and Djamchid Ghazanfarpour. Realistic real-time rain rendering. *Computers & Graphics*, 30(4):507–518, 2006.

[36] C. D. Stow and R. D. Stainer. The physical products of a splashing water drop. *Journal of the Meteorological Society of Japan*, 55:518–531, 1977.

[37] Natalya Tatarchuk. Artist-directable real-time rain rendering in city environments. In *SIGGRAPH ’06: ACM SIGGRAPH 2006 Courses*, pages 23–64, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1185657.1185828.

[38] Huamin Wang, Peter J. Mucha, and Greg Turk. Water drops on surfaces. *ACM Trans. Graph.*, 24(3):921–929, 2005. ISSN 0730-0301. doi: http://doi.acm.org/10.1145/1073204.1073284.

[39] Lifeng Wang, Zhouchen Lin, Tian Fang, Xu Yang, Xuan Yu, and Sing Bing Kang. Real-time rendering of realistic rain. In *SIGGRAPH ’06: ACM SIGGRAPH 2006 Sketches*, page 156, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: http://doi.acm.org/10.1145/1179849.1180044.

[40] Niniane Wang and Bretton Wade. Rendering falling rain and snow. In *SIGGRAPH ’04: ACM SIGGRAPH 2004 Sketches*, page 14, New York, NY, USA, 2004. ACM. ISBN 1-59593-896-2. doi: http://doi.acm.org/10.1145/1186223.1186241.