IJMRA

International Journal of Management, IT & Engineering

(ISSN: 2249-0558)

CONTENTS

Editorial Board

Title

SIMULATING COMPLEX ENVIRONMENTAL PHENOMENA USING CUBEMAP MAPPING TECHNIQUE

Author(s)

Movva. N.V. Kiran Babu
Associate Professor,
Mother Teresa Institute of Science & Technology,
Sathupally, Khammam

Ch. Siva Rama Krishna
Mother Teresa Institute of Science & Technology,
Sathupally, Khammam

M. Hanumantha Rao
Mother Teresa Institute of Science & Technology,
Sathupally, Khammam

V. Venu Gopal
Mother Teresa Institute of Science & Technology,
Sathupally, Khammam

Abstract:

concepts of simulation of rain falling effects using: Delta3D Particle System Editor, which renders rain falling effect using two static raindrop textures, and Physical Properties-Based Method which renders rain falling effect by mapping the background scene onto raindrops according to their refraction property. We observe that the reflection property of a raindrop is also important since it contribute to its visibility in the dark regions. Hence, we propose a new method that takes into account both refraction and reflection property of raindrop. increase in the computational speed of graphics hardware in recent times. This power afforded by modern graphics cards enables the possibility of simulating complex environmental phenomena.

1. Introduction:

Special Effects, especially atmospheric effects such as rain, cloud, snow and other outdoor scenes in interactive applications (video games, training systems...) are important in creating realistic environments. As current graphics hardware's computation speed is improving, their high degreeof realism is also required to immerse the user in a visually convincing environment. However,rendering these effects is a hard problem, especially in real time.Rain is a complex atmospheric physical phenomenon and consists of numerous visual cues. There are different forms of rainfall like light, moderate, heavy, and extremerainfall. The falling of raindrops is commonly and easily seen in the environment, than its effectafter it hits the ground. Whenever falling raindrops hit the surface of the ground, it causes twoeffects: water rippling and splashing of raindrop. During rain, all raindrops are merged togetherand form a stream of water on the streets and objects.

1.1 Rain Rendering

In general, Particle Systems are used to generate rain in many video games by using particles falling vertically from the sky. Each particle is a translucent white streak. This may be a segment geometrically composed of two points, or approximated by a texture plated on a rectangle stretched vertically commonly called billboard). The second technique has an advantage of allowing different drop sizes depending on the depth relative to observer, unlike the use of segments which have a fixed width whatever their position in the scene. Langer et al. [1] presented interactive an image-based spectral synthesis method to render snow and rain. The proposed method that is a combination of a Particle System and a spectral analysis technique, which creates a dynamic rainy or snowy environment texture. The spectrum of the falling snow or rain texture is defined by a dispersion relation in the image plane. presents a result obtained by this methodWang and Wade [2] presented a technique that maps textures onto a double cone which is placed around the observer as shown by the left image in The orientation of cones is determined by the position and speed of camera movement. Several textures of rain and snow are simultaneously scrolled on the double cone with different speed to give a motion impression.

This technique is faster than Particle System, but does not allow any kind of interaction between particles and their environment, whether physical interaction (bounces and other forces applied to particles) or optical (refraction and reflection of the scene visible in the drops). Besides, the textures must be drawn by artists for all the conditions shown in the scene taking into account the density of precipitation and level of brightness of the scene in which the textures will be used.

ATI and Nvidia have integrated programs including a simulation of rain in their software development kits to illustrate the possibilities of their hardware. The figures present the comparison of the results obtained by ATI and Nvidia respectively. ATI was the first to publish a demo, called Toyshop Demo, of a rainy night in a city. This demo contains various effects detailed in [3], [4], and [5]. Among these effects, the rain simulation interests us the most. The technique is based on a post-processing image-space technique, in which the rain is added once the rest of the scene was rendered. Several layers of raindrops are simulated with different speeds at varied depths in a single rendering layer in order to get a feeling of raindrop motion parallax (a strong visual cue in any dynamic environment). The fact that the animation takes place at night, and therefore with a low intensity enables not to bother with simulating refraction and reflection in the drops. Hence, the used textures which contain a constant stain and intensity are modulated by the intensity of a global luminous scene. This enables a realistic interaction of rain with the lightning which shortly lightens the scene from times to times. The demo also contains a simulation of raindrops falling off objects, for example raindrops pouring from a gutter pipe or falling off the rooftop ledge. Drops are simulated as independent particles and animated using graphics hardware of the method described in [6]. Finally, this demo takes into account the splashes coming from drops collision with a wet surface. The positions of the collisions are random and uncorrelated with the positions of the parallax planes that contain the falling drops. The rendering of the splashes is done by pre-rendering a high-quality splash sequence for a milk drop. This sequence is then deformed to avoid obvious similarities between apparent splashes, and afterward played step by step at each splash position. Nvidia also proposed a demo of rain with its software of development kit for DirectX 10 to illustrate the possibilities of its models Geforce 8. The demo is detailed in [7] and uses databaseof images created by [8]. Some parameters have been eliminated and the database was restricted to 300 textures. Some of the textures, the angle of viewpoint and the light direction are selected and interpolated to render raindrop particles. This demo uses a night scene and only the interaction with light is simulated. It does not take into account the refraction and reflection of the raindrops appearance.

1.2 Problem Statement

The main objective of the project is to create realistic rain falling effect taking into account the refraction and reflection of raindrop by modeling the world as a cube map, and using environment matting techniques. Unfortunately, this takes a lot of computation to render an image. Hence, We also focus on optimization running time of this technique in order make this effect run in real-time (which means the number of image or frame generated per second (frame-rate) is greater than 20).

1.3 Our Contributions

Our contributions toward this project are :

1. We implemented a new method of creating realistic rain falling effect in Delta3D. It isa physically-based method and mainly based on cube maps and environment mappingtechniques.

2. We optimized Matlab's codes of environment mapping technique and converted them intoC++ code. This causes realistic rain falling effect run in real-time with more than 10000raindrops in the scene.

3. We also implemented a new method of making streak raindrops based on our raindropmodel. This makes our rain effect look more realistic.

4. We generated maps which can be used in Delta3D, and also for other purposes of creatingrealistic rain falling in animation movies, training systems, PC games, PS3 games, or othersimulations.

5. The proposed method above can be considered as an extension to existing environmentmapping technique [9] as follows:

(a) Many objects can be moved in the scene

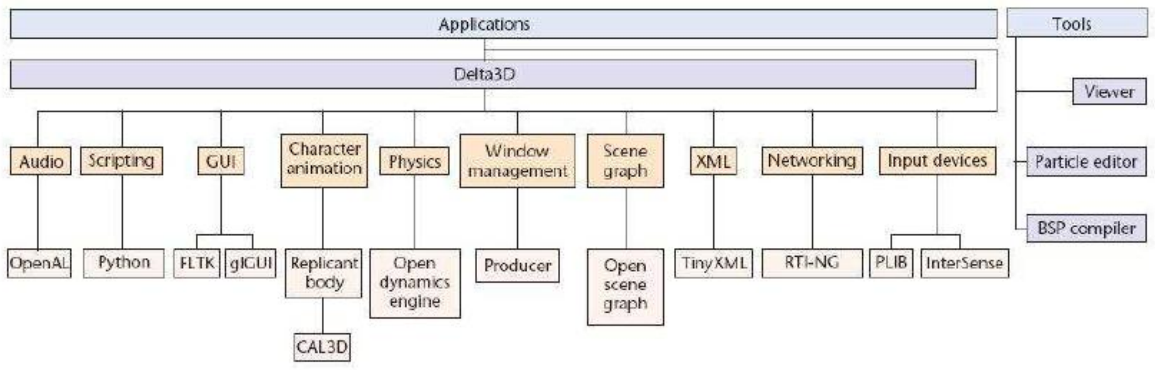
(b) View point (camera) can freely be rotated in the scene

(c) View point can be moved in the scene

(d) Rendering a photorealistic image of resolution 512x512 takes less than 0.05 second,thus rendering time of moving objects in the scene can be done in real-time.

2. Delta3D:

Delta3D [10][11] is an Open Source engine which can be used for games, simulations, or othergraphical applications. Its modular design integrates other well-known Open Source projectssuch as Open Scene Graph [12], Open Dynamics Engine [13], Character Animation Library [14],and OpenAL [15] as well as projects such as Trolltech'sQt, Crazy Eddie's GUI (CEGUI), Xerces-C, Producer, InterSense Tracker Drivers, HawkNL, and the Game Networking Engine (GNE).It has a high-level, cross-platform (Win32 and Linux) C++ API designed with programmers inmind to soften the learning curve, but always makes lower levels of abstraction available to thedeveloper. Programmers can develop content through the level editor-they can write Pythonscript to the Delta3D API or to the underlying tools directly. Delta3D uses the standard LesserGNU Public License (LGPL) [16]. It's completely modular and allows a best-of-breed approachwhereby any module can be swapped out if a better option becomes available. Below figure showsthe Delta3D architecture. All the products in the bottom layer are existing open source projects.Delta3D unifies them into one consistent API with associated tools.



2.1 Effect in Delta3D

Delta3D contains many effects such as motion models (Fly, UFO,Walk, Orbit, and First Person),smoke, explosion, animation blending, and particularly Graphical Particle Effect Editor. Thiseditor allows developers to use graphical tool to change the properties of a particle system andsee the effects immediately in real time.

2.2 **Using Delta3D Particle System Editor**

The principal concept of this method is to first create a texture of a raindrop and map it on asmall billboard (particle) that will always face the viewer (camera). Then, a rectangular emitteris made to shoot randomly particles straight down from a particular height. Some propertiesof a particle have to be set, such as, life of drop (for example 2 seconds, it’s best to have raindropsdisappear when they hits the ground), size of drop (or particle, for example from 0.15mto 0.25m) , various numbers of particles creation (for example from 150 particles/sec to 200particles/sec), and initial velocity rang of drops (for example from 6 m/s to 10 m/s) so thatraindrops don’t fall at regular rates, numbers, and patterns. Even though the particles aresupposed to be random, most people will notice a pattern after a while. So finally, to further

stop this from happening, emitters (or layers) have to be created by putting different numbersinto these emitter’s properties and especially making different raindrop textures.Whenever a raindrop hits the wet surface of flat area such as the surface of the water in pool,circular of ripple (water ring) will occurs. This effect can be made by this particle system editorin a similar manner. At first, texture of water ring has to be mapped to a particle that alwayslay flat on the ground. Then a rectangular emitter is made to shoot randomly particles butthe velocity shooting has to be set to zero in order to keep the particle from floating upwards.Lastly, the size of particle (or water ring) has to be set from 0m to 0.16m so that enlargementof water ring can be seen.These textures and a simple terrain are used in the particle system editor whose properties areset with the example values mentioned aboveThis method allows user to have the impression of a rainy environment, but it’s not very realistic.This is because of the rain is rendered using two static texture of raindrops, whose colorsare independent to the environment.

2.3 **Using Physical Properties-Based Method**

This method is based on physical properties (geometrical, dynamic and optical) of raindrops.Ross [19] proved that falling raindrops look more like ellipsoids. Small raindrops are almostspherical, and bigger raindrops get fattened at the bottom. Beard and Chuang [20] [21] gave a10th order cosine distortion on a sphere as a means to reproduce raindrop shapes:

$$r(\theta) = a\left[1 + \sum_{n=0}^{10} c_n \cos(n\theta)\right]$$

The basic concept is capturing an image of the background scene to a texture. Later, this textureis mapped onto the raindrops according to optical laws (Only Snell’s law). Since it takes into account the refraction of raindrop,its result is much more realistic than the previous method. However, since thismethod doesn’t take care of the reflection of raindrop, raindrops will not be seen in the darkarea, for example, when the scene camera looks at the dark sky while the ground is lit. In thiscase, raindrops are visible because the appearance of raindrops is influenced by the reflectionlaw that causes the color on the border of the drops look like the ground

A photograph of the original scene was used as a backgroundimage for the simulated drop. The bottom images show a close view of the original andsimulated drops. As the real drop just left the tap, its shape is not yet stabilized and is notperfectly spherical, and so it does not behave exactly as the simulated one. The image that wesee inside the drop is upside down or inverted image because drops act like convex lens.

3. Using Cube Map and Environment Mapping:

Environment mapping is a rendering technique used to create photorealistic images of an objectby mapping the environment on the object according to its property. This technique can bedone by using a cube map that contains six faces which are mapped by six images captured fromthe scene surrounding the object. It is useful because of the fact that rendering a transparentobject, which is reflective and refractive, by ray tracing can be very expensive, whereas environmentmapping render it more cheaply with little loss of accuracy. So the main point of thismethod is to map the environment on each raindrop. This takes into account both refractionand reflection of raindrop.The following subsections present three primary functions of the environment mapping [9]:Pattern Generation, Map Generation and Relighting and Compositing, our main method ofmaking realistic rain falling, making a streak raindrop, and the final algorithm.

3.1 Three Primary Functions

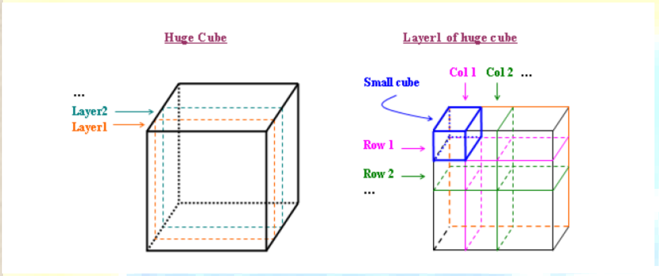
Pattern Generation, this function is used to make unique color code for each pixel of the cubemap by generating patterns. Patterns are projected on six faces of the cube for finding the map.Number of patterns depend on the number of pixels in the cube and number of colors used forcoding the pixels. They are generated by the following formula:

$$\left\lceil \frac{\log(|\text{Pixels}|)}{\log(|\text{Colors}|)} \right\rceil = \left\lceil \frac{\log \text{K x K x 6}}{\log \text{c}} \right\rceil$$

whereKxK is the resolution of a cube face. For example, the number of patterns required for acube map (with each face) of resolution 512x512 and 3 colors for coding is 13.Map Generation, this function generates a map, stores the location and fraction of the contributingpixel on cube map, by checking each pixel color of the input images. These input images

are created by rendering of object A, at location X, in the cube whose six faces are mappedwith generated patterns. Since patterns contains unique color codes, so does the created inputimages. According to these codes, contribution pixels on cube map can be found.Relighting and

Compositing, this function use the generated map to get new good quality imagesby rendering object A at location X in the cube whose six faces are mapped by six imagescaptured from any scenes.Once we get the map of object A, we use that map to render it in a new real world environmentto see how the beautiful illumination effects created by refraction or reflection causedby object A. Top left and right are renderedimage without zooming. Bottom left and right are rendered image with 50 times zoom in.



3.2 **Our Technique**

Our technique is primarily based on the aforementioned three functions, Pattern Generation,Map generation, and Relighting and Compositing. It has two steps: pre-processing step andprocessing step.In pre-processing step, we make an imaginary ”huge” cube and set the camera (the viewpoint)in the middle of it. The huge cube is then divided into L layers and each layer are divided intoR rows and C columns, so we can imagine that we have L*R*C small cubes inside that hugecube. Afterwards, we use Pattern Generation and Map Generation togenerate maps of a raindrop put in the middle of the small cube one by one. Eventually, weget M*R*C maps which are used in processing step. All theses maps are indexed from one toM*R*C according to the position of small cubes so that we can easily find them.In processing step, we capture six images of the scene and mapped these images on six facesof the huge cube. Then we check position of each raindrop and used the correspond map torender those raindrops. For example if a raindrop positions in a small cube numbered three thanthe map numbered three is used by Relighting and Compositing function to render that raindrop.

Capturing six imges of the scene: we use a second camera which is positioned at the samelocation as the primary camera (observer) and directed parallel to the ground. Field Of

View(FOV) of the camera is set to 90o. Front, left, right, top, down, and back image are generatedby capturing the scene after the camera is rotated 0o, 90o to left, 90o to right, 90o to up, 90o todown, and 180o to up respectively.Movement of the camera: wherever the camera moves (forward, backward, leftward, rightward,upward, downward), the cube is also moved along with it since we want that camera tobe always located in the middle of the cube. This is because all the maps are generated whenthe camera is only positioned in the middle of the huge cube in pre-processing step. Besides,when the camera is changed from a position A to a new position B, six images of the scene haveto be recaptured so that raindrops are looked reasonable in the scene by rendering with thesenew images. But, when it is rotated, the scene needs not to be recaptured.Thescene contains a blue object, a red object, and a raindrop. In reality, if the raindrop is seen fromthe position A, it appears red; if that raindrop is seen from the position B, it appears blue.

our technique shows the movement of camera from positionA to B. In bottom left image, the camera is positioned at A and the raindrop looks redsince it is primarily contributed by the red color which is on the huge cube. In bottom rightimage, when the camera is moved to B, the huge cube is also moved along with it and the sceneis recaptured so that the raindrop looks blue.

3.3.3 **Making Streak Raindrop**

Making rain falling effect as mentioned above is a physically-based method, thus all the raindropslook like spherical or ellipsoidal. Even though these raindrops are physically correct, theyseem to lack realism since human eye actually see raindrops like streaks. This is because ofhuman's retina which takes about 70 ms to form an image and raindrop moves very quickly.During that time, the raindrop keeps falling, and all its different positions compose a continuoussequence of images on the retina.Since the streak is the result of one moving drop, we used an image of our raindrop model

to make a new vertical streak image as follows (Fig 3.11):

1. Compute a mean of each pixel column of a raindrop to get a row of mean values

2. Use this row to make a streak

3. Lower alpha value in order to blur the streak

4. ALGORITHM

Pre-processing step:

```
1: Generate patterns
2: Map these patterns on the huge cube
3: for Each small cube in the huge cube do
4: place a raindrop in the middle of the small cube
5: camera direction is looked at the raindrop
6: camera field of view is decreased to see only the raindrop
7: render an input image
8: end for
9: Generate L*R*C maps using these input images
```

Processing step:

```
1: Load maps generated in pre-processing step
2: Capture 6 images of surrounding environment
3: for Each particle that is visible to camera do
4: Get the particle position
5: Check the particle if it is inside which small cube using its position
6: if The small cube is not already used then
7: Compute raindrop texture (raindrop color) using captured images
and a map correspond to the small cube
8: if Streak raindrop is enable then
9: Compute streak raindrop texture using raindrop texture above
```

```
10: end if
11: Store this texture in memory
12: Mark that small cube as used
13: else
14: Load raindrop texture (raindrop color) from memory
15: end if
16: Map this texture to the particle
17: end for
```

4.1 **Optimization**

After checking the existing Matlab codes of three primary functions Pattern Generation, MapGeneration and Relighting and Compositing, it's seemed that the map structure and some partsof the codes can probably be optimized to gain computation speed. Hence we restructure themap and modify some codes, particularly Relighting and Compositing function's codes. As aresult, running time of modified function improves much over the original function. Since these three functions are needed for our implementation in C++, we converted them to C++ codes.Interestingly, the new result of computation time is pretty fast. Both output result (outputimage) of Matlab and C++ functions are the same. Detailed result of our experiment is shownbelow:

The following are our assumption which is used by original Matlab, new Matlab, and c++ codes.

The number of reflection + refraction is 4 because we did the same experiment by varying thatnumber from 1 to 10 and the result images of setting number from 4 to 10 are look exactly thesame.

• Resolution of each cube face, input images, and output image: 512x512

• Number of colors for making unique color codes: 3

• Number of refraction + reflection: 4

The following is the result of original Matlab code:

• Map computation : 70 seconds (average)

• Relighting and Compositing : 30 seconds (average)

We modified the original Matlab codes using two steps:

1. By changing the map format and partly modifying codes in Map Generation and Relighting and Compositing function in order to use the new map, we got the following result:

• Map computation : 80 seconds (average)

• Relighting and Compositing : 20 seconds (average)

2. By optimizing the first step, that is modifying main part of the codes in Relighting and Compositing function, we got the following result:

• Map computation : 75 seconds (average)

• Relighting and Compositing : 1 seconds (average)

Finally, we converted Malab codes into c++ codes and got the following result:

• Map computation : 2 seconds (average)

• Relighting and Compositing : 0.04 seconds (average)

4.2 Creating Rain Falling Effect

To facilitate our implementation, particle system of OpenSceneGraph integrated in Delta3D isused to make falling raindrops. Practically, no particle system generate raindrops, but particleswhose properties such as position, size, color, and visibility are controllable. So we can changetheir color to a reasonable raindrop color using cube map and environment mapping. We modified some properties of particle system to shoot randomly particles straight down from the top of huge cube. These particles are generated inside the huge cube, at least 2 meters awayfrom camera, and move along with camera.

4.3 **Result**

All the result images are rendered in Delta3D on a PC with Dual-Core Intel CPU 1.86GHzprocessor with 2GB RAM. Our rain falling effect can be run in real-time in a scene with 10000 drops or streak drops we can clearlysee different between randrops on the top and bottom corner because of the refraction propertyof raindrop. Since raindrop have reflection property, we can also see the light brown color, whichis contributed by brown color of the ground, on the border of raindrops. In the same situation,raindrops rendered by Physical Properties-Based Method will not be visible since it doesn't takeinto account the reflection property of raindrop.

5. Conclusion and Future Work:

We created rain falling effect using Delta3D Particle System Editor that allows the user to havethe impression of a rainy environment, but it's not very realistic. We proposed a new method togenerates visually convincing results of rain falling using Cube Map and Environment Mapping,which is a rendering technique used to create photorealistic images of an object by mapping theenvironment on the object. Results of this method are more realistic than the other two methodsusing: Delta3D Particle System Editor, which renders rain falling using two static textureof raindrops whose colors are independent to the environment, and Physical Properties-BasedMethod, which renders rain falling by mapping the background scene onto raindrops accordingtheir refraction property. We highly optimized the original Matlab codes of environmentmapping by modifying map format and main part of the codes in Relighting and Compositingfunction. We also converted the optimized Matlab codes into c++ whose computation time ispretty fast. We proposed a new method of making streak raindrops based on our raindropmodel. Finally, we implemented our new methods in Delta3D modifying its particle system torender reasonable raindrop textures instead of particles.In future work, our method can be implemented in Graphics Processing Unit, which is veryefficient at manipulating and displaying computer graphics, in order to render more than 10000raindrops in real-time. It can also be improved to handle collisions of the raindropswith the ground or objects in the scene.

6. Bibliography:

- M. S. Langer, L. Zhang, A. W. Klein, A. Bhatia, J. Pereira, and D. Rekhi.A spectral-particlehybrid method for rendering falling snow.In Rendering Techniques 2004 (EurographicsSym-posium on Rendering). ACM Press, june 2004.
- N.Wang and B.Wade. Rendering falling rain and snow.In ACM SIGGRAPH 2004 TechnicalSketches Program, 2004.
- N. Tatarchuk. Artist-directable real-time rain rendering in city environments.In Proceedingsof Game Developers Conference, 2006.
- N. Tatarchuk. Artist-directable real-time rain rendering in city environments. In Proceedingsof the 2006 Symposium on Interactive 3D graphics and games Poster, page 30, New York,NY, USA, 2006. ACM Press.
- N. Tatarchuk and J. Isidoro.Artist-directable real-time rain rendering in city environments.In Eurographics Workshop on Natural Phenomena, New York, NY, USA, 2006.ACM Press.
- P. Kipfer, M. Segal, and R. Westermann.Uberflow : A GPU-based particle engine. In Pro-ceedings of ACM SIGGRAPH/EUROGRAPHICS Symposium on Graphics hardware, pages115-122, 2004.
- S. Tariq. Rain.Technical report, Nvidia, 2007.
- K. Garg and S.K. Nayar.Photorealistic Rendering of Rain Streaks.ACM Trans. on Graphics(also Proc. of ACM SIGGRAPH), July 2006.
- BiswarupChoudhury, DeepaliSingla, SharatChandran. Fast Color-Space decompositionbased Environment Matting. In Proceedings of the 2008 symposium on Interactive 3D graph-ics and games, pages 1-1. ACM, 2008.
- Delta3D. www.delta3d.org, last viewed on June, 05 2008.
- Rudy Darken, Perry McDowell, and Erik Johnson. Projects in VR: the Delta3D open sourcegame engine. Computer Graphics and Applications, IEEE, Vol.25, Iss.3, May-June 2005.
- Open Scene Graph. http://www.openscenegraph.org/, last viewed on June, 05 2008.
- Open Dynamics Engine. http://www.ode.org/, last viewed on June, 05 2008.

- 3D Character Animation Library. https://gna.org/projects/cal3d/, last viewed on June, 052008.
- OpenAL,Cross-Platform 3D Audio. http://www.openal.org/, last viewed on June, 05 2008.
- GNU Lesser General Public License. http://www.gnu.org/licenses/lgpl.html, last viewed onJune, 05 2008.
- Falling Rain Particle Effect. http://www.delta3d.org/filemgmt/visit.php?lid=80, last viewedon June, 05 2008.
- Pierre Rousseau, Vincent Jolivet, DjamchidGhazanfarpour. Realistic real-time rain rendering.Computers & Graphics 30, 4 (2006), 507-518. special issue on Natural PhenomenaSimulation.
- Ross, O.N. (2000) Optical Remote Sensing of Rainfall Microstructures, FreieUniversit¨atBerlin, FachbereichPhysik, Diplom Thesis, 134pp.
- Chuang C, Beard KV. A new model for the equilibrium shape of raindrops. Journal ofAtmospheric Science, 44(11):1509-1524, 1987.
- Chuang C, Beard KV. A numerical model for the equilibrium shape of electrified raindrops. Journal of Atmospheric Science, 47(11):1374-89, 1990.