Rain and Snow Flurries

Esther de Jong

21st November 2014

Msc Computer Animation and Visual Effects

CGIT

Research Report

Bournemouth University

Contents

Abstract

This report will look into different methods for animating rain and snow flurries and derive a method that will be used for the project. Methods for animating rain and snow are discussed separately. Most methods use particles to simulate the snow and rain and this will also be done for the project. The wind will be simulated using the lattice-Boltzmann method, creating a grid of wind nodes. The wind nodes will create a dynamic vector field by communicating with their neighbours. The particles will act under the wind force, the gravity and the collision force.

1 Introduction

The mood of a scene can be set or changed by implementing snow or rain. This effect can be used in movies, animations or games. This report will discuss the methods found in literature to realisticly animate snow and rain their interaction with wind. Resulting in rain and snow flurries. First several techniques for snowfall and rainfall are discussed. In the second part the techniques that will be used later on will be discussed in more detail.

2 Literature Research

In this section several methods for rain and snow simulation will be described. Not all methods take the influence of the wind into account by focussing on realistic rendering, or create the whirling effect in another way.

2.1 Snow

Since snow flakes are small and separate entities they can be visualized using particles. This is done by Langer et al. [Langer et al. 2004], Muraoka and Chiba [Muraoka and Chiba 2000]and Wang et al. [Wang et al. 2006], but they all use different methods for simulating the wind.

Langer et al. [Langer et al. 2004] uses a particle system to generate a number of snowflakes. To keep the number of particles low and still simulate the flow pattern the rest of the snowflakes are generated using image-based spectral synthesis method producing a dynamic texture. A time-varying opacity function is created, representing the summed density of snowflakes that project to each pixel and frame, which is the sum of moving 2D sine waves. Which are then rendered in overlapping image tiles. This opacity function is then used to composite a white falling snow layer on top of a background image. The particles are given a constant velocity and a discrete position. The parameters of the spectral snow are consistent with the motion of the particles within the corresponding tile. The snowflakes will move faster and appear bigger, the closer they are to the camera.

Muraoka and Chiba [Muraoka and Chiba 2000] describes a method for falling snow, snow cover and snow melt. A dynamic vector field of air currents, that is influenced by obstacles, is created by the method of vortex fields. This method by combines laminar flows with turbulent flows. The turbulent flows are created by combining velocity fields of various vortexes of different sizes. The vortexes change over time. They are generated at random point in time, grow, move and eventually go extinct. Fixed vortexes and laminar flow components are used to create wind in a fixed direction and ascending and descending currents. The snowflakes are moved by the force received from the air current field, gravitation and a random force.

$$Ma(t) = fv(t) + fr(t) + Mg - Cv(t) \tag{1}$$

Where M denotes the mass of a snowflake, a(t) denotes the acceleration of a snowflake, fv(t) denotes the force received from the vortex field, fr(t) denotes a random force, g denotes the gravity, C denotes the air resistance, and v(t) denotes the velocity of the snowflake.

Wang et al. [Wang et al. 2006] describes a method between the wind and snow particles, using the lattice-Boltzmann method (LBM) to create the wind field. With 15 set directions. This method will be used and discussed later on.

2.2 Rain

Puig-Centelles et al. [Puig-Centelles et al. 2009] describes the rendering of rain streaks in real time where the user keeps moving around, using a particle system. The level of detail is decreased for objects further away from the viewer. These Level-of-detail techniques are used to control the number of particles that need to be simulated. This is done by managing rain zones and transitions between zones with different rain conditions. All of these techniques are used to cope with the main disadvantage of a particle simulation. Since a large number of particles, which are needed to create a realistic scene, requires significant computation power.

There are three different types of rain:

- Convection rain: this is heavy rain and it arrives rapidly and does not last long.
- Relief or orographic rain: has thick clouds and drizzly conditions.
- Frontal or cyclonic rain: rain that begins slowly and remains steady for a certain amount of time which can be between a few hours or days.

When the intensity of the rain increases the number of larger rain drops also increases. While small drops are almost spherical, bigger drops are more ellipsoidal due to flattening of the bottom by relation between the surface tension and the aerodynamic pressure. For a more realistic scene rain particles are reshaped into vertical streak, because the human eye often perceives rain drops as streaks. Rain can be viewed in two different ways from afar so the viewer is outside the raining area, called far rain, and from the inside when the viewer is in the rain area, called close rain. We will only look at the close rain method. The distribution of particles when using the Level-of-detail technique is linear as described by formula 2.

$$n = 1 - d \tag{2}$$

Where n denotes the number of particles, and d the distance from the particle to the observer. The size of the particles is a quadratic relation as described by formula 3.

$$s = d^2 \tag{3}$$

Where s denotes the size of the particle.

Rousseau et al. [Rousseau 2006] states that the shape of a raindrop comes from the fact that the surface tension tries to minimize the contact surface between the air and the raindrop. In addition the aerodynamic pressure tries to stretch the raindrop horizontally. The speed of a raindrop determines its radius. For the simulation of the light refraction Rousseau et al. [Rousseau 2006] take into account the disoriented texture or the scene in addition to the light sources. Although Rousseau et al. uses several special techniques to minimize

computation and optimize the appearance of the scene while remaining in real time they do not consider the influence of wind in their model.

Garg and Nayar [Garg and Nayar 2006] developed a model for rain streaks to capture the interaction between the lighting direction, the viewing direction and the raindrop oscillation. They render a database of rain streaks before running the scene. Rain drops are created using an uniform spatial distribution in a 3D volume above the field of vision and each drop is assigned a random oscillation parameter.

For the realistic rendering of rain the reflection and refraction within a rain droplet is also of interest. For this assignment the focus will be on the interaction with the snow and rain particles with the wind. Thus these two methods fall outside the scope of this project, but can be interesting for further development.

2.3 Wind

Wei et al. [Wei et al. 2003] uses the LBM, as described in Wang et al. [Wang et al. 2006], extended with a subgrid model to simulate a real-time dynamic wind field interacting with lightweight objects. The space is divided into grids and each grid has 19 vectors in set directions. For every time step the vectors are converted to the neighbouring nodes. This will be explained in more detail in the next section. Extending the LBM with a subgrid model makes it possible to use high Reynolds numbers. Higher Reynolds numbers are needed to model turbulent flows.

2.4 Interaction

As a particle intersects with a wind node, impulse is transferred. The snow particles will start at the top nodes and the vectors of the wind nodes will be added to the snow particle along with the gravity and air resistance.

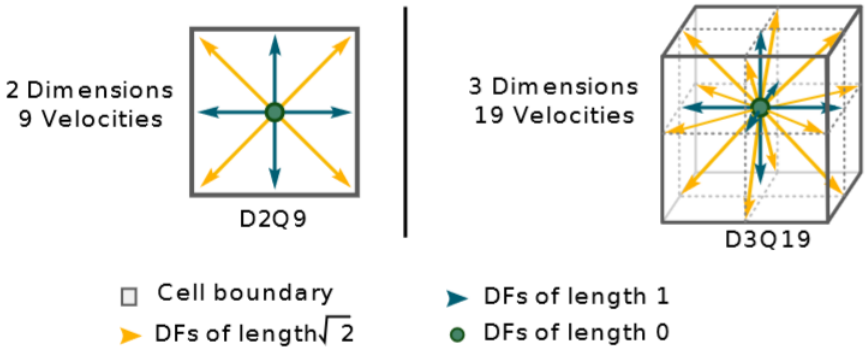

Figure 1: D3Q19 LBM [Muller et al. 2008]

3 Method

The method we will use will be based on the method described by Wang et al. [Wang et al. 2006], with some adjustments taken from the method described by Muller et al.[Muller et al. 2008]. Combining Wang et al. and Muller et al. results in a particle based simulation for the snowflakes and rain droplets. These particles will be influenced by a wind field which will be simulated using the LBM system. The LBM is a discretization of the Boltzmann equation which describes the behaviour of gas on the microscopic scale [Muller et al. 2008]. A discrete description of the Boltzmann equation is implemented by dividing the space into separate cells. The system is further discretized by allowing only certain discrete vectors inside the cells.

3.1 Wind field

The LBM is grid-based and samples the wind field at the grind nodes. Each node only interacts with adjacent nodes. The discretization and the explicit time steps of the LBM method make it possible to simulate the wind field in real-time. The LBM can be divided into two steps: the modelling of the wind field (the stream step) and the interaction of the wind field with the snow particles (collide step). Every gird node has 19 vectors in set directions (see image 1)(Denotes as D3Q19). A global wind field vector is constructed by the local vectors in the nodes. That is adding the local node vectors with discrete lengths results in a discretized representation of the global vector.

The D3Q19 method will be used as described by Muller et al. [Muller et al. 2008] and Wei et al. [Wei et al. 2003] and not the D3Q15 as described by Wang et al. [Wang et al. 2006]. This because Muller states that the model of 19 vectors is more stable than 15 and it also affects the 18 nodes that are closest to the

current node, with the distance being 1 or $\sqrt{2}$ instead of 1 and 2 for 3DQ15.

The size of those vectors determine the flow field. The velocity vectors have the following values: $e_1 = (0,0,0)^T$, $e_{2,3} = (\pm 1,0,0)^T$, $e_{4,5} = (0,\pm 1,0)$,$e_{6,7} = (0,0,\pm 1)^T$,$e_{8,9,10,11} = (\pm 1,\pm 1,0)^T$,$e_{12,13,14,15} = (0,\pm 1,\pm 1)^T$,$e_{16,17,18,19} = (\pm 1,0,\pm 1)^T$. Particle movement is represented by a floating point number $f_{1..19}$. This value needs to be stored for each of the velocities.

Assuming that the distance between the nodes is one, the wind nodes have the discrete vectors of zero for f_0, one for $f_{1,..,7}$, and $\sqrt{2}$ for $f_{8,...,19}$.

The lattice-Boltzmann stream step can be expressed as

$$f_i^*(x, t + \Delta t) = f_i(x + \triangle t\, e_{\bar{i}}, t) \tag{4}$$

$$i = 0, 1, 2, ..., 19$$

where $f_i$denotes the particle distribution function, $\triangle x$ denotes the size of a cell, Δt denotes the time interval during which particles travel along the grid, $e_{\bar{i}}$ denotes the inverse velocity vector of e_i, f_i^* denotes post-stream particle distribution function. Taking into account that$\triangle t/\triangle x = 1$. Formula 4 has to be extended with the collide step, which depends on the density and velocity of the wind particles. The density and velocity are computed by

$$\rho = \sum f_i,\ u = \sum e_i f_i \tag{5}$$

$$f_i(x, t + \triangle t) = (1 - \omega)(f_i^*(x, t + \triangle t) + \omega f_i^{eq} \tag{6}$$

Where ω denotes the parameter that controls the viscosity between the range of (0..2]. Values close to 0 give a very viscous flow and values close to 2 give a more turbulent flow. The equilibrium distribution function f_i^{eq}can be calculated with

$$f_i^{eq} = w_i \left[\rho + 3e_i * u - \frac{3}{2}u^2 + \frac{9}{2}(e_i * u)^2\right],\ where \tag{7}$$

$$w_i = 1/3\ for\ i = 1,$$

$$w_i = 1/18\ for\ i = 2..7,$$

$$w_i = 1/36\ for\ i = 8..19.$$

Formula 6 calculates the distribution function for each time step, each cell needs the distribution function from the previous time step of the surrounding cells to calculate the next step. Cells can also be set as obstacle cells, this means that no wind can pass trough them. When the neighbouring cell is an obstacle cell the inverse distribution function from the current cell is used. Resulting in

$$f_i^*(x, t + \triangle t) = f_i(x, t) \tag{8}$$

3.2 Falling Particles

The particles will start at the top of the screen randomly on one of the wind nodes. This way the particle will immediately have a vector from the wind. This vector together with the gravity will determine the translation of the particle. The force of collision with other snowflakes will not be taken into account due to the small effect of this event [Wang et al. 2006]. The gravity will be calculated by mg where m denotes the mass of a particle and g the gravitational acceleration of 9.78 m/s^2.

When the translation results in a position that corresponds to the position of a wind node the wind vector will apply to the particle. When a particle does not have the same position as a wind node the particle will inherit the vector from the closest wind node. If two wind nodes are at the same distance the vector will be inherited from the wind node that lies in the direction the particle came from. The wind vectors will change overtime to simulate effects like a wind gust. The snow particles will change direction when this happens and thus create a swirling effect.

4 Conclusion

The LBM will be used to create an interactive wind field. The D3Q19 as described by Wei and Muller will be used instead of D3Q15 as described by Wang. The snow and wind will be simulated using particles and those particles will receive the impulse from the wind nodes, gravity and air resistance. For this application the particles will have a simple colour, white for snow and blue for rain. The weight of the rain particles will be bigger than the weight of the snow particles thus their behaviour will differ.

References

[Garg and Nayar 2006] Garg, K. and Nayar, S. K., 2006. Photorealistic rendering of rain streaks. *ACM Transactions on Graphics*, 25(3), 996-1002.

[Langer et al. 2004] Langer, M. S., Zhang, L., Klein, A. W., Bhatia, A., Pereira, J., & Rekhi, D. 2004. A Spectral-particle hybrid method for rendering falling snow. *Rendering techniques*, 4, 217-226.

[Muller et al. 2008] Müller, M., Stam, J., James, D., & Thürey, N. 2008, August. Real time physics: class notes. *In ACM SIGGRAPH 2008 classes,* (p. 88). ACM.

[Muraoka and Chiba 2000] Muraoka, K., & Chiba, N. 2000, October. Visual simulation of snowfall, snow cover and snowmelt. *In Parallel and Distributed Systems: Workshops, Seventh International Conference* on, 2000 (pp. 187-194). IEEE.

[Puig-Centelles et al. 2009] Puig-Centelles, A., Ripolles, O., & Chover, M. 2009. Creation and control of rain in virtual environments. *The Visual Computer*, 25(11), 1037-1052.

[Rousseau 2006] Rousseau, P., Jolivet, V., & Ghazanfarpour, D. 2006. Realistic real-time rain rendering. *Computers & Graphics*, 30(4), 507-518.

[Wang et al. 2006] Wang, C., Wang, Z., Xia, T., & Peng, Q. 2006. Real-time snowing simulation. *The Visual Computer*, 22(5), 315-323.

[Wei et al. 2003] Wei, X., Zhao, Y., Fan, Z., Li, W., Yoakum-Stover, S., & Kaufman, A. 2003, July. Blowing in the wind. *In Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (pp. 75-85). Eurographics Association.