Parallel Visualization of Snow

Ingar Saltvik, Anne Cathrine Elster, and Henrik R. Nagel

Norwegian University of Science and Technology, Trondheim, Norway,
{saltvik|elster|hrn}@idi.ntnu.no

Abstract. This paper presents methods for using parallel computer systems to optimize performance of previously published methods for snow modelling, to achieve real-time performance. These methods have not been created with real-time use in mind, and therefore require much computational resources. The method includes a model and parallelization scheme for snowflake motion in the air, simulation of a wind field, and accumulation of snow on the ground. Focus is set on efficient parallelization on multicore processor systems.
Keywords: Snow, Parallel Computing, Computer Graphics, Simulation.

1 Introduction

Snow is a natural phenomenon in big parts of the world for long periods of time, particularly in the Nordic countries. With its presence, it has the ability to completely change the atmosphere and appearance of a scene. From a computer graphics point of view, snow can therefore be used to create evocative and realistic looking scenes. Despite this great visual significance, real time snow modelling has been given little attention in computer graphics research.

In a scene with heavy snowfall, there may be hundreds of thousands of snowflakes falling from the sky. Every single snowflake follows its own complex, chaotic path down to the ground, influenced by wind, air vortices, snowflake geometry and gravity. When snowflakes hit the ground, they pile up and form snowcover on objects in the scene. To realistically model this behavior, massive amounts of computing power is needed.

This paper presents methods for optimization and parallelization of previously published models for snow modelling in computer graphics. Previous attempts on snow modelling have either been to computationally heavy for real-time use, or sacrificed much of the realism to achieve a sufficient framerate. This work aims to use parallel computing to enable the use of more realistic models in a real-time setting. With the recent growth of dualcore and multicore processors, use of parallel methods is becoming more relevant in applications that traditionally have been restricted to only one processor.

2 Previous Work

Previous work on snow visualization can be categorized as falling and fallen snow. The most widely used methods for these categories are billboards and flat textures, respectively, and they often do not give a realistic impression.

Fearing (2000) [1] presents a method for modelling fallen snow, made up of two components; accumulation and stability. The accumulation model determines how much snow each surface should receive, and the stability model determines how snow moves from physically unstable areas to surfaces underneath in small avalanches. Nishita et al. (1997) [5] uses metaballs with a snow intensity distribution and volume rendering to display scenes with fallen snow. These methods are unfortunately too computationally intensive to be suitable for real time usage.

Haglund et al. (2002) [2] present a method for simulating and rendering snow accumulation in real time, by employing 2D matrices to store the snow depth of areas where snow could fall. When a snowflake hits the ground, the height value at that area is increased. Rendering is done by creating triangulations of the height matrices. Because of the real time requirement, this method sacrifices physical and visual realism for speed. It also requires some work by the modeller to place the height matrices. Ohlsson and Seipel (2004) [6] present a technique similar to a shadow map to render accumulated snow, and use the GPU to achieve real time performance.

For falling snow, there are mainly two approaches; image-based and particle systems. Image-based methods composite a texture on the scene to mimic snowfall. Langer et al. (2004) [7] use a spectral approach to create the texture. The other approach uses a particle system to simulate the behavior of falling snow. Particle systems can be parallelized efficiently, because the snowflakes move independent of each other. Reeves (1983) [8] deals with particle systems and Sims (1990) [9] shows how they can be parallelized. The influence of wind on falling snow can be handled by using computational fluid dynamics. Wei et al. (2003) [10] uses the Lattice Boltzmann Model (LBM) to simulate the effect of wind on light, mildly deformable objects, which should also be applicable to snowflakes. They use a feather as an example, using the model presented by Belmonte et al. (1998) [11] for falling paper strips. Stam (1999) [13] presents an unconditionally stable method for using computational fluid dynamic (the Navier-Stokes equations) for simulating fluid (air) flow.

Moeslund et al. (2005) [3] and Aagaard and Lerche (2004) [4] presents a comprehensive model for modelling of snow in computer graphics, including snowflake generation, snowflake motion, wind, and accumulation of snow. The methods are mainly based on work already mentioned here. Their aim is the movie industry, and do not try to develop the methods for real-time use.

3 Methods for Parallel Simulation of Snow

[3] and [4] identifies and handles three main aspects of snow modelling that are chosen for parallelization in this work; movement of falling snowflakes, the influence of wind on falling snowflakes, and accumulation of snow on the ground. In addition, rendering of snowcover and generation and rendering of snowflakes is highly relevant, but is not given priority here.

Two models for the motion of falling snowflakes are compared. The first one is the rather simple model of [3] where the snowflakes follow a spiral path down to the ground. The second one is a model for a falling paper strip ([11] and [12]). The latter method is found not to be worth the additional computationally complexity compared to the former. In addition, it is only defined in two dimensions, and does not easily generalize to three dimensions. Parallelization is easily achieved in a data-parallel manner, where snowflakes are assigned to processors.

To realistically account for the effect of wind on falling snowflakes, a wind field has to be simulated. This is implemented by using the method presented by [13] to solve the Navier-Stokes equations for fluid flow numerically, and parallelized by using methods presented in [14] and [15].

The approach to snow accumulation is adapted from [3] and [1], with certain simplifications. To enable surfaces to accumulate snow, triangulations are created of them. When snow falls, the triangles are checked for collisions. If a snowflakes hits a triangle, the snow height for that triangle is increased. To make a connected representation of the surfaces for rendering with OpenGL, the height values at triangle vertices are calculated as the average of the heights of all neighboring triangles. The original model includes algorithms for redistribution of snow from unstable to stable places (small avalanches), and refinement of the triangulations for a better representation of the snowcover at detailed areas. These algorithms has been left out. Detection of collisions between snowflakes and triangles is the most time consuming operations in this part of the model, and is parallelized in the same data-parallel manner as falling snow.

A task-parallel approach where rendering with OpenGL is performed in one thread, and the simulation is executed in another thread, is also examined. This may allow smooth navigation within the scene even with a slow simulation.

4 Further Work

Because this is work in progress, it must be expected that some of these suggestions for further work will be carried out in the final version of this paper.

The implementation and parallelization of the involved algorithms still has to be further optimized for performance. This will include thorough performance testing with special attention to parallel efficiency.

In the snow accumulation model, two important components, refinement of the triangulations and stability check, have been left out. These could be implemented to achieve more realism.

Another item for further work is how to render falling snowflakes and snowcover on the ground. The geometry of snowflakes is well-known to be extremely complex, as “no two snowflakes are alike”. A reasonable simplification may be to use triangles or rectangles, possibly with textures for snowflakes. For snowcover, some shading model should be applied, and for instance bump mapping could be used to imitate the small scale details on the snow surface.

The current implementation does only support shared memory architectures. The feasibility of other parallel architectures for this application, in particular PC clusters, could be investigated further.

5 Conclusion

A method for real-time visualization of snow was presented, based on previous work by [4] and references therein. To be able to handle large scenes with heavy snowfall with a real time framerate, the methods were parallelized to utilize the power of parallel computer systems. As this highly is work in progress, further work which can be expected to be elaborated, was presented.

References

1. Fearing, P.: Computer Modelling Of Fallen Snow, Proceedings of SIGGRAPH 2000, pages 37–46.
2. Haglund, H., Andersson, M., and Hast, A.: Snow Accumulation in Real-Time, Proceedings of SIGGRAD 2002, pages 11–15.
3. Moeslund T.B., Madsen C.B., Aagaard M., and Lerche D.: Modeling Falling and Accumulating Snow, Vision, Video and Graphics, 7-8 July 2005 Heriot Watt University, Edinburgh.
4. Aagaard M., and Lerche D.: Realistic Modelling of Falling and Accumulating Snow, Master thesis, Aalborg University, Denmark, 2004.
5. Nishita T., Iwasaki H., Dobashi Y., and Nakamae E.: A Modelling and Rendering Method for Snow by Using Metaballs, Eurographics 1997, Vol 16, No. 3.
6. Ohlsson P. and Seipel S.: Real-time Rendering of Accumulated Snow, SIGRAD 2004.
7. Langer M. S., Zhang L., Klein A.W., Bhatia A., Pereira J., and Rekhi D.: A spectral-particle hybrid method for rendering falling snow, Eurographics Symposium on Rendering (2004).
8. Reeves, W. T.: Particle Systems—A Technique for Modelling a Class of Fuzzy Objects, ACM Transactions on Graphics, Vol. 2, No. 2, April 1983, Pages 91–108.
9. Sims K.: Particle Animation and Rendering Using Data Parallel Computation, SIGGRAPH 1990: Proceedings of the 17th annual conference on Computer graphics and interactive techniques, Pages 405–413.
10. Wei X., Zhao Y., Fan Z., Li W., Yoakum-Stover S., and Kaufmann A.: Blowing in the Wind, Eurographics/SIGGRAPH Symposium on Computer Animation (2003).
11. Belmonte, A., Eisenberg, H., and Moses E.: From Flutter to Tumble: Inertial Drag and Froude Similarity in Falling Paper, Physical Review Letters, Vol. 81, No. 2, Pages 345–348 (1998).
12. Tanabe, Y. and Kaneko, K.: Behavior of a Falling Paper, Physical Review Letters, Vol. 73, No. 10, Pages 1372–1377 (1994).
13. Stam J.: Stable Fluids, SIGGRAPH 1999: Proceedings of the 26th annual conference on Computer graphics and interactive techniques, Pages 121–128.
14. Vik, T.: Real-time Visual Simulation of Smoke, Master's thesis, IDI, NTNU, Norway, 2003.
15. Bryborn, M., Klein, R., May, S., Schneider, S., and Weber, A.: A portable, parallel, real-time animation system for turbulent fluids. International Conference on Parallel and Distributed Computing and Systems '00, pages 394–400.