GPU-based Dynamic Flow Visualization for Climate Research Applications

Nicolas Cuntz*
Computer Graphics Group
University of Siegen, Germany

Martin Leidl†
Telecooperation
TU Darmstadt, Germany

Andreas Kolb*
Computer Graphics Group
University of Siegen, Germany

Christof Rezk Salama*
Computer Graphics Group
University of Siegen, Germany

Michael Böttinger‡
Deutsches Klimarechenzentrum, Hamburg, Germany

Abstract

Climate models simulate the most important processes of the Earth System, including the circulation of the atmosphere and the ocean. For the the visualization of the resulting large time dependent data sets, many different techniques are used. In order to analyze the fluid flow, interactive flow visualization techniques are important. Various commercial and free software packages used for climate data visualization have rather limited support for the interactive exploration of large flow data sets. For example, some of them are designed to generate pre-calculated animations only.

On the other hand, recent developments in visualization techniques exploiting programmable features of current graphics processing unit (GPU) have proven to be very powerful. This is especially true for particle-based techniques. However, there is still a functional gap between these particle-based flow visualization techniques utilizing GPU processing power and the requirements in the context of geophysical fluid flows, which is the focus of this paper.

We present a complete GPU-based framework for interactive visualization of time-dependent climate flow data. The framework includes a proper data work flow from the simulation to the visualization, the handling of non-uniform data grids and, finally, a proper support for the interactive data-exploration feasible to climate researchers.

We demonstrate the framework using the flow data of a simulation of a typhoon. This work is still in progress and the framework's design is open for future incorporation of more particle-based visualization techniques.

*{nicolas.cuntz,andreas.kolb,christof.rezk}@uni-siegen.de
†leidl@tk.informatik.tu-darmstadt.de
‡boettinger@dkrz.de

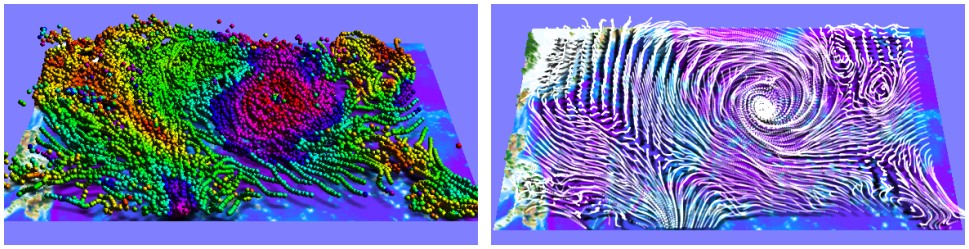

Figure 1: Interactive, particle-based flow visualization of a typhoon (Data courtesy of Max-Planck-Institute for Meteorology), incorporating shadows casted onto a height map of the respective geographical region. The left image shows a color-coding of flow magnitude, the right image uses additive blending in order to obtain smooth visual effects.

1 Introduction and Motivation

For the simulation of long term changes of our climate, coupled three dimensional models of the most important compartments of the earth system are used: atmosphere and ocean, including models for sea ice and land cover. These so called general circulation models (GCMs) simulate the dynamics of fluids, i.e. liquids and gases. The partial differential equations, which describe these processes, can only approximately be solved for discrete computational grids. The resulting multivariate and time dependent data consist of numerous scalar and vector quantities, which are stored for all grid points in regular time intervals. Due to limitations in storage capacity, it is only feasible to use storage intervals (e.g. 6 hours) which are much larger as the computational time step (e.g. 20 minutes) used for the simulation.

Although the interactive visualization of the climate data is absolutely essential for the analysis and communication of the results, only some suitable visualization systems are available, and in most cases they offer only a limited subset of the desired functionality [HBSB02]. In respect to the interactive exploration of climate simulation data, a visualization system should fulfil a set of basic requirements, such as geographical mapping and support of the time dimension. Although the system should ideally support the original grids used for the simulations, at least rectilinear grids must be supported to account for the aspect ratio of the physical space and the non linear vertical grid spacing used in most models. Most systems use arrows and/or stream lines for the visualization of vector quantities, i.e. wind or ocean currents. The transports of the fluids, and hence of energy, momentum, chemical compounds etc., which are of great importance for the large scale processes in the climate system, are directly linked to the vector fields.

Both, vector arrows and stream lines, fail to visualize the transports within time dependent flow fields. Arrows show only the actual velocity components per time step while stream lines are only applicable for stationary flows. Trajectories show the paths of particles transported by the flow field, but the simultaneous display of actual and past positions can be misleading - especially when other scalar quantities are visualized in the same context.

Animated short trajectories or particles allow for correct visualization of the transports in

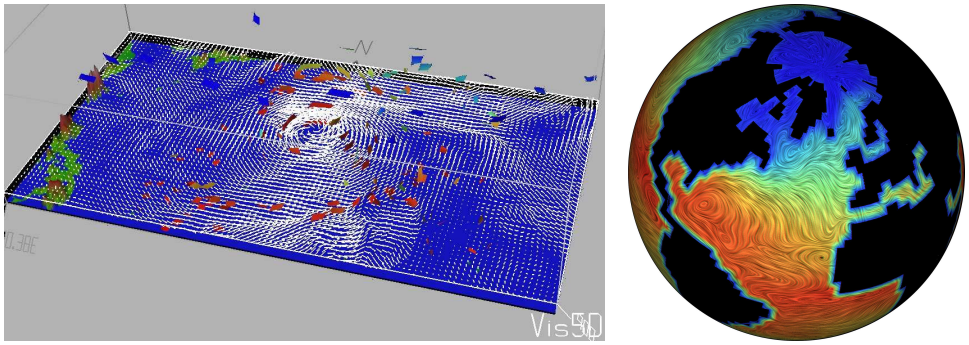

Figure 2: Left: Vis5d visualization using arrows and particles; Right: Global ocean circulation using Line Integral Convolution (LIC) visualizing the top layer stream pattern colored by temperature.

3D flow fields. Some of the available visualization systems support trajectories and particles. Fig. 2, left, shows a screen-shot of a visualization of a meteorological data set with vis5d [HS90] with arrows and particles. After the initial particle placement the advection is done on the CPU, the positions are stored for all time steps, before the results can interactively be visualized. It is not possible to move the emitter around in order to explore the flow field interactively. Other visualization systems like the commercial AVS/Express [AVS06] may support the interactive emitter placement, but lack the desired interactive performance, especially for larger numbers of particles. A larger number of particles allows for a dense representation of the flow field, comparable to LIC (see Fig. 2, right, and [CL93]), in order to observe small scale features and critical regions. Traditional LIC-based visualization techniques, however, can only be used for 2D data and are not suitable for interactive visualization.

The development of programmable graphics hardware introduced a large number of very powerful techniques in the visualization context. Examples are real-time volume graphics [EHK+06] and illustrative visualization [VKG05]. Based on purely GPU-based particle tracing techniques, introduced by Latta [Lat04], Kipfer et al. [KSW04] and Kolb et al. [KLRS04], Krüger et al. proposed the usage of GPU-based particle systems for flow visualization [KKKW05]. Other works realize the flow simulation on the GPU, either using grid-based techniques [Har03, KW05] or particle methods [KC05, HCM06]. However, none of these techniques have been adopted to the specific context of climate research and its specific requirements concerning functionality and usability.

The goal of this work is to leverage the power of GPU-based particle systems for specific applications in climate research. We focus on the question, if GPU-based particle systems can overcome the limitations of existing visualisation systems in respect to the use of dense 3D respresentations of unsteady flows for interactive data exploration. The major contribution of this paper is the concept of an open framework for particle-based visualization of climate flow data. This incorporates major technical and algorithmic aspects like handling non-uniform grids and unsteady 3D flow data on the GPU. The developed framework en-

ables the end-user, i.e. the climate researcher, to interactively explore his simulation results utilizing the graphics hardware from his commodity PC. Beside the interactive placements of particle sources, various visualization techniques have been integrated in the framework, including particle shadows on surface height-field and semi-transparent as well as solid sphere particle rendering with color-coded mapping of scalar values.
The remainder of the paper is structured as follows. After a presentation of related topics in the context of climate simulations and GPU-based particle tracing (Sec. 2), Sec. 3 deals with data conversion and non-uniform data grids. Sec. 4 discusses technical aspects of the framework. Implementation details are given in Sec. 5. Results and conclusions are presented in Sec. 6.

2 Related Topics

In this section, the major requirements to flow visualization implied by specific climate and earth-system research application are discussed (Sec. 2.1). Additionally, the current state in particle based flow visualization, especially on GPUs, is presented in Sec. 2.2.

2.1 System Requirements

In the context of climate research a large set of visualization task occur. Focus on the major task of flow visualization, the following system requirements are of high interest.

Data Sets: In general, climate simulations result in large sets of unsteady flow data, associated with various physical flow attributes like temperature or salinity.

We address the requirement by integrating existing conversion tools in the data preprocessing (see Sec. 3.1).

Grids: Typically, the simulation is performed on three-dimensional grids covering a specific section of the earth. In case of the simulation of global climate phenomena, these grids are curvilinear in horizontal direction and rectilinear in vertical direction.

Currently, our system supports rectilinear and, for specific cases, curvilinear grids (see Sec. 3.2).

Data Formats: Due to the specific setup of climate simulation quantities, the simulation data cannot directly be used for visualization. The usage of *standard formats*, however, is necessary in order to allow for the application of data handling tool at hand.

Our system does not introduce any further constraint on the data management. Thus, the integration of existing tools for data handling suffices to guarantee a proper data handling (see Sec. 3.1).

Flow Primitives, Geographical Mapping and Visualization Effects: The investigation of the flow can also be significantly improved by advanced visualization methods. For example, arrows are frequently used to indicate the direction of the flow, stream

lines give a better impression of the evolution of the flow, and complex glyphs can be used to store more information about the visualized data.

Furthermore, visual effects like shadows and lighting effects in combination with the a proper geographical mapping significantly improves the perception of 3D objects and thus of the fluid behaviour.

Currently, only particles are used for the flow visualization. The particle rendering is combined with the visualization of heightfields, representing the corresponding topographic features, and various rendering and lighting effects (see Sec. 4.3 and Sec. 4.4).

Interactivity and Precision: The visualization needs to provide as much interactivity as possible in order to allow fast exploration and comparison of large flow data sets.

In order to assess the quality of climate models, simulations of the past and the current climate are made and compared with observations. The mean state and the variability of the observed patterns should be met as good as possible. This includes processes like storm events. For this task, not only interactive rendering is essential, but the online control over visualization process itself, e.g. the placement of flow primitives, is of great importance.

On the other hand the precision of the visualization, i.e. the precision of the numerical integration in our case, should be under control of the user. Ideally, one would like to control the error due to numerical inaccuracy. At least, the visualization system should provide a mechanism to check whether errors occur, and where errors are located.

Utilizing the GPU's processing power in online particle tracing, our system can handle a large number of particles in real-time (see Sec. 2.2 and 4.2).

Computational Resources: Many systems, especially real-time tools, require high-end hardware in order to cope with the large amount of data. Performing interactive flow analysis on commercial, general purpose PCs highly decreases the costs for the hardware and helps to reduce the time for development and evaluation of climate models.

2.2 GPU-Based Particle Systems

Although various techniques for the visualization of vector fields have been proposed, particle systems play an important role in the analysis of flow mechanics. The mass-less particle p is tracked within the time-dependent flow field $\vec{\mathbf{v}}(\mathbf{x}, t)$ using Newton's law of motion, i.e.

$$\mathbf{x}_p(t) = \mathbf{x}_p(t_0) + \int_{t_0}^{t} \vec{\mathbf{v}}(\mathbf{x}_p(s), s)\, ds,$$

where t_0 and $\mathbf{x}_p(t_0)$ represent the initial time and the initial particle position, respectively. This provides a basis for different kinds of derived visualization techniques for dynamic flow fields, like curve-based techniques (e.g. streak lines, stream lines or path lines) or surface-based methods (e.g. stream ribbons or stream tubes) [SM00, WE04].

Latta first presented a *state-preserving* particle tracing system, e.g. a system that stores the particle state variables for the last time step(s), completely realized on the GPU [Lat04]. Latta stores the particle states, e.g. position, velocity or life time in 2D textures. Textures storing varying state parameters are used in a double buffering setup, in order to deal with the fact that the stream processing paradigm of the GPU requires a strict separation of source and destination data storage. Only the particle initialization is done on the CPU, thus keeping the data transfer to the GPU at a minimum. As a result, this system can track 1 million particles at interactive frame rates using a graphics card with NVIDIA GeForce 5900 chip. Latta's system additionally integrates an approximate depth sorting mechanism in order to achieve proper blending results in case of non-commutative blending operators in the image composition stage.
A concept for the visualization of static 3D flow-data on uniform grids was presented by Krüger et al. [KKKW05]. This work aims to achieve a precise and interactive visualization of flow data. Beside the pure particle visualization, Krüger et al. integrated stream lines and stream ribbons into their system. The approach also analyzes the resulting errors of the integration, allowing the user to alter the fix step size accordingly. Since this approach works on a given static velocity field, a computation of the particle's velocities is not needed, e.g. only one 2D texture storing the varying particle positions is used. The velocity field is stored in a 3D texture and transferred into the video memory during the program initialization.
Weiskopf et al. [WSEE05] propose a hybrid particle and texture based technique for unsteady, meanly 2D flow data on uniform grids. Here the particle tracing is used to generate a visualization by means of a *Line Integral Convolution (LIC)* utilizing radial basis functions, thus the particles indirectly contribute to the final visualization. The system has been applied to 2D climate flow data and achieves interactive frame rates in a *Lagrangian-Eulerian Advection (LEA)* framework. Since LIC is a space filling technique, it can not be applied to the visualization of 3D flow data directly.

3 Processing Climate Flow Data

Adopting the known particle based techniques discussed in Sec. 2.1 to the requirements in climate and earth-system research involves several problems addressed in this section. Sec. 3.1 describes the processing of data resulting from climate simulations in order to be applicable to particle-based visualization on the GPU. The handling of non-uniform grids is discussed in Sec. 3.2.

3.1 Data Conversion

One of the data formats widely used for the storage, exchange and further analysis of climate data is the self describing netCDF-Format [Uni06]. Beside the velocity in horizontal direction in $[m/s]$ and in vertical direction in $[Pa/s]$ (pressure variation per second), the data includes additional quantities like temperature, relative humidity and surface pressure. In a preprocessing step, the vertical component of the 3D velocity vectors has to be converted to $[m/s]$. To perform this computation, a specific conversion routine is applied, resulting in a unified flow vector representation, again stored in netCDF-Format.

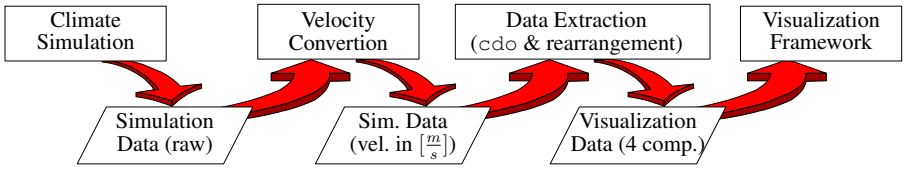

Figure 3: Data work flow from the climate simulation to the visualization.

Finally, the data relevant for visualization, i.e. the velocity vectors plus one additional scalar like temperature is extracted using the command line tool `cdo` (climate data operators), developed at MPI for Meteorology in Hamburg [MPI06]. The visualization data is then stored in a binary float format, including additional header information, i.e. size, resolution and number of time steps. After slight modifications, this format is ready to be used in the GPU-based particle visualization.

3.2 Data Grids

Current particle-based GPU-techniques for flow visualization use uniform grids, which resembles the usage of 2D- or 3D textures storing the flow fields. However, in climate research rectilinear, curvilinear or even unstructured grids are used, requiring additional functionality in particle tracing.

In general, one has to distinguish between the *physical space* $\mathcal{P}$, or *p-space*, e.g. a portion of the earth's troposphere, and the *computational space* $\mathcal{C}$, or *c-space*, in which the simulation results, i.e. all vector- and scalar quantities, are stored (see Sec. 2.1). Note, that the quantities are valid in $\mathcal{P}$ only. A location $\mathbf{x}^c$ in c-space can be mapped to a p-space location $\mathbf{x}^p$ via a bijective mapping $\Phi : \mathcal{C} \mapsto \mathcal{P}$ (see Fig. 4, left).

For local phenomena, the grids are defined in longitude-latitude coordinates and altitude. For visualization purposes, it is sufficient to assume the longitude-latitude coordinates to be orthogonal, i.e. Φ_{xy} is bi-linear in this case. For global phenomena, the simulation grids are defined more generally: $\Phi_{xy}(x^c, y^c) = (\phi_x(x^c, y^c), \phi_y(x^c, y^c))$. The visualization is done in longitude-latitude parameters. In both cases the altitude is parametrized using a list of fixed and ascending height values $\{z_0^p, \ldots, z_{z_{max}-1}^p\},\ z_i^p < z_{i+1}^p,\ i = 0, \ldots, z_{max} - 2$. For atmospheric simulations, the height spacing in p-space increases with larger altitudes (see Fig. 4, right).

During the particle tracing, it is necessary to transform a particle position $\mathbf{x}_p^p$ from p-space into c-space, i.e. $\mathbf{x}_p^c = \Phi^{-1}(\mathbf{x}_p^p)$, in order to determine any particle attribute, especially the particle velocity (see Fig. 4, left). Here, the assumption is made, that for a known location $\mathbf{x}_p^c$ in c-space, the corresponding attribute, e.g. velocity, temperature etc. can be obtained using tri-linear interpolation on the c-space grid (see also Sec. 4.2).

The altitude values ϕ_z are placed on a rectilinear grid, thus ϕ_z^{-1} is piecewise linear and can be determined using binary search in conjunction with a linear interpolation in c-space:

$$z^c = i + \frac{z^p - z_i^p}{z_{i+1}^p - z_i^p},\ \text{if } z^p \in [z_i^p, z_{i+1}^p[.$$

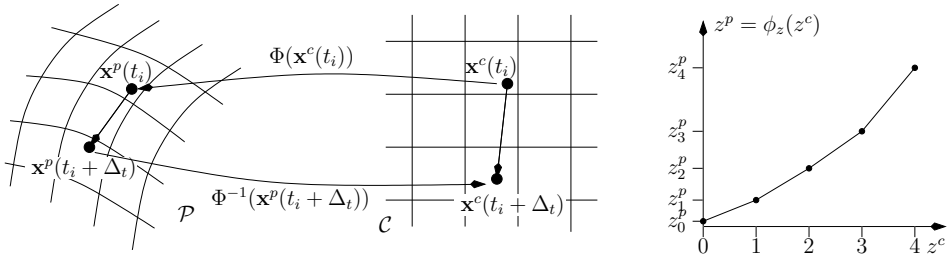

Figure 4: Data grids for climate simulations. Left, the mapping between computational space (c-space) and physical space (p-space) for a single time step in particle tracing. Right, the ascending altitudes with increasing spacing in the case of atmospheric simulations.

In the horizontal direction, our framework supports curvilinear grids with the restriction, that Φ_{xy}^{-1} is known analytically, thus $\mathbf{x}_p^c = \Phi^{-1}(\mathbf{x}_p^p)$ can be computed directly. However, in climate research this restriction is not always met. Efficient and robust algorithms to determine $\Phi^{-1}(\mathbf{x}_p^p)$ numerically will be integrated in the future.

4 Technical Aspects

In this section, techniques related to GPU-based particle visualization are described. This includes memory management (Sec. 4.1) particle tracing (Sec. 4.2) and the final visualization (Sec. 4.3 and 4.4).

4.1 Memory Management

There are two problems when working with large buffers stored on the GPU. First, the GPU usually disposes significantly less memory than the CPU. Climate visualization data comes along with particularly large buffers (e.g. flow data stored in large 3D textures) which cannot be hold completely in the GPU memory. The second problem occurs when data is initialized on CPU side and has to be transferred to GPU memory. The speed of this transfer is limited by the bandwidth of the graphics port, e.g. AGP (Accelerated Graphics Port) or PCI Express. The first problem implicates the second one, if only a portion of the data set can be hold in GPU memory at one time. Using the OpenGL extension `ARB_pixel_buffer_object`, it is possible to transfer data asynchronously, i.e. a buffer located in CPU memory is copied in parallel while control flow continues (see List. 1).
Note that proper rearrangement of the data for efficient transfer to the GPU is performed by the graphics driver only in the case of non-power-of-two textures (provided by the OpenGL extension `ARB_texture_non_power_of_two`).

```
void *pPBOMemory;

// bind and initialize pixel buffer object
glBindBuffer(GL_PIXEL_UNPACK_BUFFER_ARB,
             nTexBuffer[nBufferId]);
glBufferData(GL_PIXEL_UNPACK_BUFFER_ARB,
             nDataSize, NULL, GL_STREAM_DRAW);

// local copy from RAM to pixel buffer
pPBOMemory = glMapBuffer(GL_PIXEL_UNPACK_BUFFER_ARB,
GL_WRITE_ONLY);
memcpy(pPBOMemory, (void*)data, nDataSize);
glUnmapBuffer(GL_PIXEL_UNPACK_BUFFER_ARB);

// bind and transfer texture
glBindTexture(GL_TEXTURE_3D, uiTexNames[nTextureId]);
glTexSubImage3D(GL_TEXTURE_3D, 0, 0, 0, 0,
    uiResolutionX, uiResolutionY, uiResolutionZ,
    GL_RGB, GL_FLOAT, BUFFER_OFFSET(0));
```

Listing 1: OpenGL code sample for asynchronous bus transfer using pixel buffer objects.

4.2 Particle Tracing

The computation of the particle positions for a subsequent time step is done in a fragment shader triggered by rasterizing the particle texture. Thus, the shader code is processed for each particle independently.

Since we aim for an accurate particle integration, the velocity vectors stored in the discrete flow field and for discrete time steps are sub-sampled using linear interpolation. Essentially, this is a quadri-linear interpolation, where three spatial and one time components are involved. Unfortunately, no support for hardware accelerated tri-linear texture filtering for 3D texture is currently available. Thus, the interpolation has to be performed manually using Cg's linear interpolation `lerp`.

This interpolation code is run twice in order to interpolate between two subsequent time steps. The time integration responsible for particle motion can be done by an integration scheme like Euler or Runge-Kutta. The Runge-Kutta method is known to produce accurate results, provided that an appropriate time step is chosen. Traditionally [Stö95], an ideal time step can be obtained by means of the so-called Q-factor, which is defined as follows:

$$Q = \left| \frac{k_3 - k_2}{k_2 - k_1} \right|$$

Numerical errors are minimal for $Q \in (0.025, 0.1)$. Typically, the time step is adapted in an iterative algorithm, until this requirement is fulfilled.

```
float3 x1, x2, x3;
float  t1, t2, t3;
float3 k1, k2, k3, k4;

k1     = sampleVField(oldPos, time);
t1     = time   + timeStep * 0.5;
x1     = oldPos + timeStep * 0.5 * k1;

k2     = sampleVField(x1, t1);
t2     = time   + timeStep * 0.5;
x2     = oldPos + timeStep * 0.5 * k2;

k3     = sampleVField(x2, t2);
t3     = time   + timeStep;
x3     = oldPos + timeStep * k3;

k4     = sampleVField(x3, t3);
newPos = oldPos + timeStep/6.0 * (k1 + 2.0*(k2 + k3) + k4);

float q = abs((k3-k2)/(k2-k1));
float color = step(0.025, q)*step(-0.1, -q);
// color = 1: ok, color = 0: not
```

Listing 2: Cg code sample for particle tracing using 4-th order Runge-Kutta integration.

For real-time applications, nearly constant frame rates are required in order to produce a smooth animation. Thus, time-adaptivity is hard to realize, but it is possible to check accuracy during simulation in order to provide a feed-back to the user. In our system, particles for which the Q-factor fails to be in the desired bounds are color-coded, indicating the i occurrence and location of the numerical error. If desired, the user can adjust the time step iteratively until the error is eliminated. The code for the implemented Runge-Kutta 4 approach, together with accuracy check, is listed in List. 2.

4.3 Height Fields

A height field is used to represent the earth surface. A vertex buffer object (provided by the OpenGL extension `ARB_vertex_buffer_object`) stores the vertices of a regular 2D grid storing the z-displacement for each vertex. An additional texture, with potentially higher resolution, is used to visualize topographic details. The resulting visualization of the model topography shows the geographic registration of the flow field.

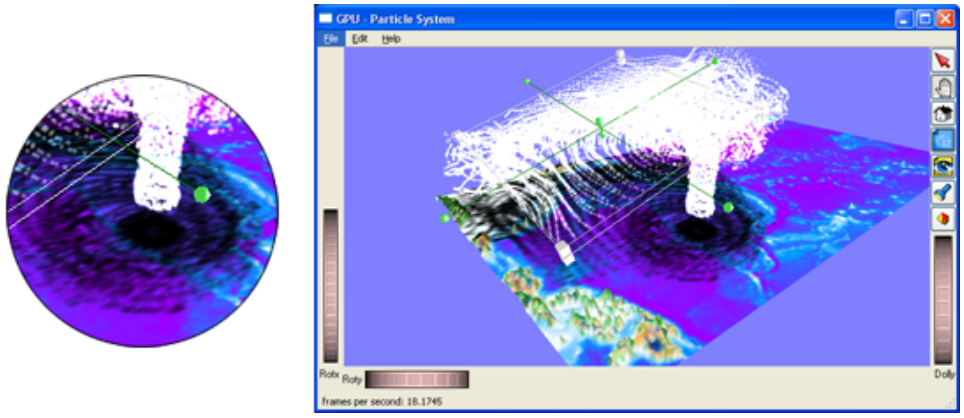

Figure 5: Shadow rendering using a shadow texture approach (left), the user interface is designed to facilitate the exploration of the visualized flow data. In the picture, the emitter plane is just being moved. (right)

4.4 Rendering Particles and Shadows

The particles can be rendered either as solid spheres or as semi-transparent sprites. Both approaches are realized using quadratic, screen-aligned point sprites (see Fig. 1). In case of the semi-transparent sprites, additional α-blending is used. Currently, this is restricted to commutative blend-operators, in order to prevent a particle depth sorting. However, there is no conceptual challenge to extend this to non-commutative blending operators (see [KLRS04, KSW04]. Rendering sphere is realized using a binary α-component in the point-sprite stenciling the sphere in conjunction with an α-test. The actual sphere-shape is generated in a fragment program using a Phong model. The color-coded visualization of the additional scalar quantity stored in the 3D flow texture can be applied only in case of spheres, since blending alters the color values leading to not meaningful results.
Along with the particle rendering, shadows facilitate the perception of the spatial location and orientation of displayed 3D objects. In our case, particles moving in the flow can be better associated with locations on the underground by means of projected shadows. This is especially the case where the flow is not restricted to stay in a plane, thus forcing the user to choose different points of view.
Shadows are rendered orthogonally onto the textured height field which visualizes the model topography. This is done by mapping the z component of the particle positions in a vertex shader while rerendering all particles. The advantage of orthogonal mapping in combination with height fields is that even for elevated regions, the shadows remain correct.

5 Implementation

This section discusses aspects related to the application interface. his includes the graphical user interface and the textual configuration of the visualization scenario.
The application can be controlled by means of two interfaces. Interactive control is responsible for the exploration of the visualized data. It is supported by the graphical user interface (GUI), which is based on the `SoExaminerViewer` provided by the scene graph library OpenInventor and the GUI toolkit Qt. This combination is known to be suited for scientific visualization tools (see [RSEH06]). This interface provides control elements for navigating in the scene. It supports moving, resizing and orienting of particle emitter planes, which is a very useful tool when exploring different areas in the flow (see Fig. 5).
The second interface is given in terms of a configuration file. The idea behind this mechanism is to provide a flexible method to the user in order to incorporate new visualization setups. For each new setup, all parameters are encapsulated in one single configuration file. This is a powerful tool in addition to the user interface. In the configuration file, technical aspects as integration time step, number of particles and the flow data set can be modified.

6 Results and Conclusions

The evaluation of the described system was performed using a prototype implementation running on an AMD64 3000+ system with 1 GB main memory and an NVidia Geforce 7800 GS (AGP 8x) and 256 MB local video memory.
For the tests of our method we used data simulated with the global AGCM (atmospheric general circulation model) ECHAM5 [RBB$^+$03] in T213 resolution, which corresponds to a horizontal grid spacing of 0.5625 degrees or approximately 60 km near the equator. For the visualization, the originally 31 vertical hybrid model levels have been interpolated onto 39 rectilinear height levels. We selected a time period and region where a strong tropical storm, a typhoon, occurred in the simulation. The resulting subset of the original data was $106 \times 53 \times 39$ grid points and 32 time steps.
To evaluate the performance of our framework, several grid resolutions have been tested. On a 50^3 grid (1.5 MB), the measured frame rate is 21.3 fps, for a 175^3 (64.3 MB) grid, the frame rate is 11.8 fps. Note that at least two time steps have to be in GPU memory for the particle tracing, while a third one is loaded asynchronously. In all cases, some $50,000$ particles are traced and visualized. The evaluation has shown that the described technique is capable of sustaining interactive frame rates for field resolutions of the order of those used in climate simulations. Due to the fact that the distance between two time steps is large (6 hours in the visualization of the typhoon), the amount of data transferred from CPU memory to GPU memory is relatively small. Thus, our code is limited by computation rather than bandwidth. Although the flow data for one time step is relatively small, texture cache misses are likely to happen due to the linear memory layout for non-power-of-two textures, which is rather inefficient for 3D textures. More efficient storage schemes exist for power-of-two 3D textures. In this case, however, the CPU is required to rearrange the texture data, which prohibits asynchronous texture transfer from host to local video memory.

An ongoing practical evaluation by the domain experts shows that the presented application is a valuable improvement compared to previous solutions for climate research. The users highly appreciate the possibility to interactively navigate their simulation data and explore the inherent information by selectively probing the vector field using real-time interaction with the particle emitters.
The presented framework should still be considered to be work-in-progress and provides the first step toward a fully functional real-time application for climate visualization. Future investigations must include more intuitive user interfaces and advanced means to interactively probe the data. The use of transfer functions will potentially be of help for the isolation of interesting structures in the flow. Particle visualization can be enhanced by encoding additional information by the use of vector or tensor glyphs. Higher-order primitives such as stream lines, ribbons and surfaces to further improve the perception of flow characteristics may be integrated using approaches similar to [KKKW05]. The extension of the presented techniques to allow for arbitrary curvilinear grid topologies is another challenge to be addressed in future work.

References

[AVS06] AVS. AVS - Advanced Visualiation Systems, 2006. `http://www.avs.com`.

[CL93] B. Cabral and L. Leedom. Imaging vector fields using line integral convolution. In *ACM Proceedings SIGGRAPH*, pages 263–270, 1993.

[EHK+06] K. Engel, M. Hadwiger, J. Kniss, A. Lefohn, C. Rezk Salama, and D. Weiskopf. *Real-time volume graphics*. AK Peters Ltd., 2006.

[Har03] M. Harris. *Real-time cloud simulation and rendering*. PhD thesis, CS Dep., University of N. Carolina at Chapel Hill, 2003. Supervisor: Anselmo Lastra.

[HBSB02] B. Hibbard, M. Böttinger, M. Schultz, and J. Biercamp. Visualization in earth system science. *ACM SIGGRAPH Comput. Graph.*, 36(4):5–9, 2002.

[HCM06] K. Hegeman, N. Carr, and G. Miller. Particle-based fluid simulation on the gpu. In *Proc. ICCS 2006*, volume 3994 of *LNCS*, pages 228–235. Springer, 2006.

[HS90] B. Hibbard and D. Santek. The vis-5d system for easy interactive visualization. *Proc. IEEE Conf. on Visualization*, pages 28–35, 1990.

[KC05] A. Kolb and N. Cuntz. Dynamic particle coupling for gpu-based fluid simulation. In *Proc. of the 18th Symposium on Simulation Technique*, pages 722–727, 2005.

[KKKW05] J. Krüger, P. Kipfer, P. Kondratieva, and R. Westermann. A particle system for interactive visualization of 3d flows. *IEEE Transactions on Visualization and Computer Graphics*, 11(6):744–756, 2005.

[KLRS04] A. Kolb, L. Latta, and C. Rezk-Salama. Hardware-based simulation and collision detection for large particle systems. In *Proc. Graphics Hardware*, pages 123–131. ACM Press, 2004.

[KSW04] P. Kipfer, M. Segal, and R. Westermann. UberFlow: A GPU-based particle engine. In *Proc. Graphics Hardware*, pages 115–122. ACM Press, 2004.

[KW05] J. Krüger and R. Westermann. Gpu simulation and rendering of volumetric effects for computer games and virtual environments. In *Proceedings*, pages 685–693. Eurographics, 2005.

[Lat04] L. Latta. Building a million particle system, 2004. `http://www.2ld.de/gdc2004/`.

[MPI06] MPI for Meteorology. CDO - Climate Data Operators, 2006. `http://www.mpimet.mpg.de/~cdo`.

[RBB+03] E. Roeckner, G. Baeuml, L. Bonaventura, R. Brokopf, M. Esch, M. Giorgetta, S. Hagemann, I. Kirchner, L. Kornblueh, E. Manzini, A. Rhodin, U. Schlese, U. Schulzweidea, and A. Tompkins. The atmospheric general circulation model echam5, part i. Technical report, Max Planck Institute for Meteorology, 2003.

[RSEH06] C. Rezk-Salama, K. Engel, and F. Higuera. The OpenQVis project, 2006. `http://openqvis.sourceforge.net/`.

[SM00] H. Schumann and W. Müller. *Visualisierung - Grundlagen und allgemeine Methoden*. Springer, 2000.

[Stö95] Horst Stöcker. *Taschenbuch mathematischer Formeln und moderner Verfahren*, chapter 18.12 Numerische Integration von Differentialgleichungen, page 635. Verlag Harri Deutsch, 1995.

[Uni06] Unidata. NetCDF - Network Common data format, 2006. `http://www.unidata.ucar.edu/software/netcdf/`.

[VKG05] I. Viola, A. Kanitsar, and E. Gröller. Importance-driven feature enhancement in volume visualization. *Proc. IEEE Conf. on Visualization*, 11(4):408–418, 2005.

[WE04] D. Weiskopf and G. Erlebacher. *The Visualization Handbook*, chapter 12 Overview of Flow Visualization, pages 261–278. Academic Press, 2004.

[WSEE05] D. Weiskopf, F. Schramm, G. Erlebacher, and T. Ertl. Particle and texture based spatiotemporal visualization of time-dependent vector fields. In *Proc. IEEE Conf. on Visualization*, pages 639–646, 2005.