OpenGL Visualization of Hurricane Isabel

Greg P. Johnson * Christopher A. Burns †

Texas Advanced Computing Center

ABSTRACT

Our visualization solution is an OpenGL application we designed and developed specifically to visualize this data set. The code is written in C/C++ and uses the GLUT toolkit for user input and frame buffering. We have also made use of the FLTK library for GUI controls, and an implementation of the Pthreads API for multithreading. We chose this strategy so that we would best be able to customize the visualization features to suit this particular data set, and to maximize the interactivity of the visualization. Multithreading is used to effectively perform I/O and sorting operations in parallel, while GUI controls give the user an easy way to set various parameters related to visual quality, performance, and the transfer functions used to represent data.

CR Categories and Subject Descriptors: I.3.3 [Computer graphics]: Picture/Image Generation; I.3.6 [Computer Graphics]: Methodology and Techniques

Additional Keywords: scientific visualization, weather model, cloud rendering, volume data, volume rendering

1 INTERACTIVITY

Real-time performance of our solution was a design goal from the beginning. Our program by default will downsample the precipitation data by a factor of four in each dimension, the wind data by a factor of ten in the X and Y dimensions, and the cloud data is downsampled by a factor of three. However, the user can dynamically downsample the data further to improve performance if necessary (the cloud data can be sampled at full resolution). Our hardware consists of a Dell system with two Intel Xeon processors running at 2.8 GHz, 2.0 GB of memory, and an NVIDIA GeForce 6800 GT graphics card. There are three different sets of variables which can be enabled or disabled by the user: clouds, wind, and precipitation. At a resolution of 800x600 and with all three sets enabled, we can achieve a frame rate of approximately 13 fps. When the clouds are disabled, frame rate improves to 30 fps. While the simulation is running and new data is constantly being streamed from disk and interpolated, these values drop to 5 fps and 13 fps respectively.

2 EXPLORATORY SUPPORT

The user is able to manipulate the camera view easily during any mode or phase of the simulation (while paused, or while the simulation plays through the timesteps). The movement of the camera is intuitive and therefore easy to use. The mouse controls the look direction and arrow keys on the keyboard control movement through space. The camera can be trucked along the view direction, from side to side, or along the camera's "up" axis. These features are sufficient to allow the user to examine any part of the visualization at any time during the simulation.

While the simulation automatically advances the visualization along the timeline by default, the user may pause at any time, or instruct the program to jump to any particular timestep. The data is interpolated between the given timesteps to ensure smooth transitions between them for intermediate frames. Thus the user

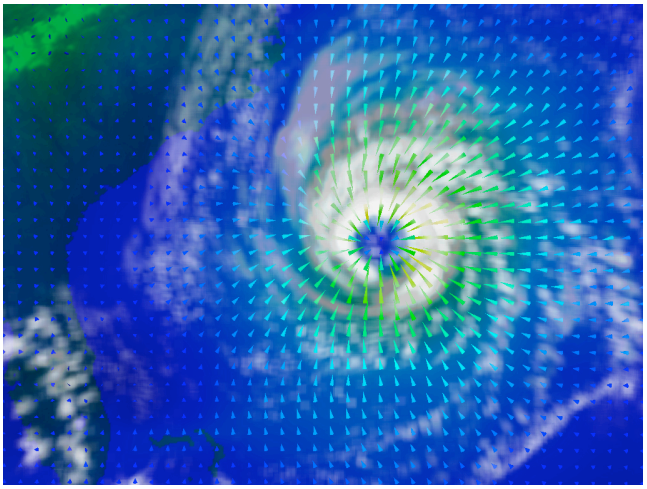

Figure 1. The vector field colors indicate wind magnitude while the clouds clearly show the structure of the hurricane.

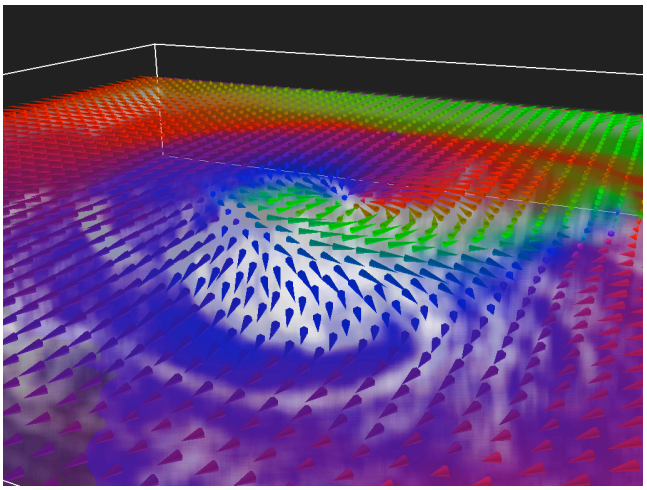

Figure 2. At higher altitudes the swirling pattern is more intense and easily visible than at sea level. The color of the vector field is a function of wind direction, with red mapped to north.

* gregj@tacc.utexas.edu
† cslugg@tacc.utexas.edu

has full flexibility to scan through the time-varying data automatically, or can inspect any particular timestep statically.

In addition to movement through time and space, we provided the user with a significant amount of flexibility in adjusting the various transfer functions for each of the data variables. The minimum and maximum data values to which the color gradients are mapped are all user configurable, as are the subsampling rates and the position of the wind vector slicing plane as shown in Figure 5.

3 Multiple Characteristics

Our system is capable of rendering multiple variables simultaneously in real-time and in a meaningful way. Figures 1 and 4 show the rendering of cloud data along with precipitation and wind velocities, giving the user a means to understand and analyze the relationship between the variables.

3.1 Clouds and Structure

To display the clouds we used a particle system where each cloud particle is rendered with a single billboarded quad. The saturation and transparency of the particles are linear functions of the cloud data and are set by default such that heavy clouds result in darker and more opaque particles, resulting in a cloud that appears to have thickness. As the camera travels through the rendered particles, the clouds appear to thin out, just as real clouds do. Sorting of the partially transparent quads is achieved by subdividing the volume into a series of 3D tiles which are sorted by their centroids' distance from the camera. The particles in each tile are then sorted in parallel. The tiling reduces the computational complexity of sorting the cloud particles, and since the arrangement of the data in memory reflects the tiling, our program achieves a more coherent memory access pattern which makes better use of the processors' cache memories.

3.2 Wind Velocity

The wind data can optionally be visualized simultaneously with the cloud and precipitation data. Using a horizontal slicing plane, the wind values are illustrated with a grid of colored vectors and a partially transparent surface. The colors are a mapping of the color wheel onto the compass directions, with red indicating north, shown in Figure 2. Alternatively, the colormap can show wind magnitude, with warmer colors indicating stronger winds. The vectors' length is a function of the wind magnitude at that voxel. The altitude of this plane can be freely adjusted by the user.

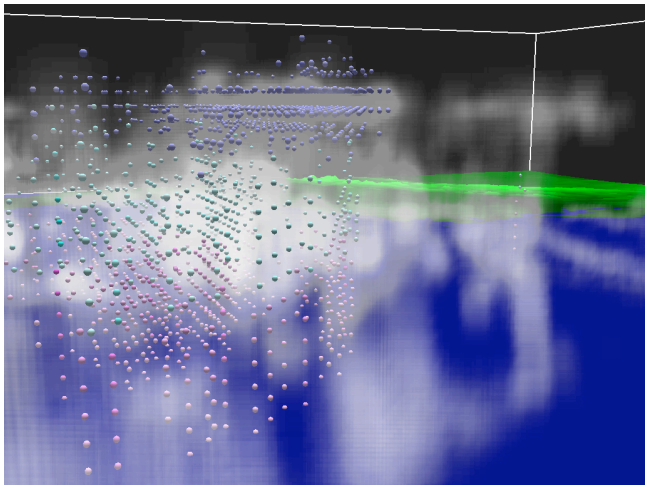

Figure 4. The particles show how the snow is concentrated at the higher altitudes, the groupel in the middle ranges, and the rain further towards sea level. This technique allows the user to clearly see the relationship between the clouds and the precipitation.

3.3 Precipitation

The precipitation variables are rendered using a simple particle technique that makes it easy to visualize the precipitation along with the cloud structure in the same frame. For a given voxel, the precipitation is represented by a colored sphere. Cyan spheres indicate graupel, magenta corresponds to rain, blue indicates snow, and grey represents the total precipitation. The saturation of a particle is determined by a linear transfer function of the variable's ratio data value. The discrete particles show up well against the background of the cloud structures, which are rendered in a way that looks more volumetric. It is important that the users be able to effectively visualize the cloud structure and the precipitation simultaneously as in Figure 4.

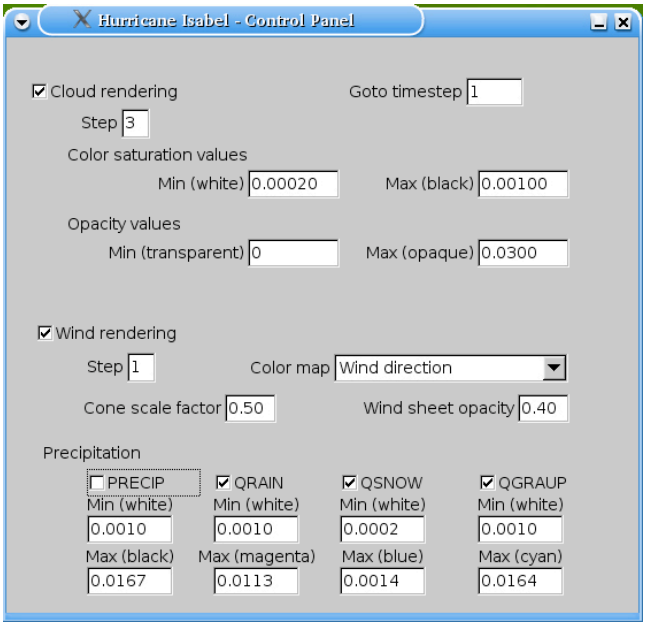

Figure 5. The user interface. The "Step" values refer to the down-sampling factor for those variables.

4 Conclusion

The sheer size of the data set presented the biggest obstacle to our goal of producing an interactive visualization that is both aesthetically pleasing and scientifically informative. We expended considerable effort optimizing the memory layout, reducing the frequency of I/O operations, and eliminating unnecessary geometric processing on the GPU. While our solution is not *complete* (due to subsampling), we compensate by allowing the user to adjust the degree of sub-sampling. Finally, we have not implemented any visualization of the temperature, pressure, or ice mixing ratio variables due to time constraints. However, the methods used to visualize other similar variables could easily be extended to illustrate these. Other improvements might include more user control over the transfer functions, slicing planes along other axes, and more precise control over the camera.