Responding to environmental emergencies in real time at the Canadian Meteorological Centre: using "SPI" as a management and visualization tool for global to local scale emergencies

Jean-Philippe Gauthier, Environmental Emergency Response Division, Canadian Meteorological Centre, 18 April, 2006

Introduction

The Canadian Meteorological Centre (CMC) has specific international mandates in the area of environmental emergency response. The Centre is a WMO Regional Specialized Meteorological Centre (RSMC) for atmospheric transport modeling of radioactive material and an active participant on the Comprehensive Nuclear Test Ban Treaty (CTBT). It is also an ICAO Volcanic Ash Advisory Centre (VAAC). The CMC plays as well an active role in providing modeling dispersion guidance and scientific advice in the support to emergency response activities within Canada.

In order to meet these mandates in the operational context the Division has developed over the past 6 years a series of tools and graphical applications. These tools enable the operational meteorologist as well as specialized users to manage the response and the execution of dispersion models in a user-friendly way. In the early years, all of the informatics processes were managed using scripts running in batch mode. Very little interactivity was available or even possible in the entire process.

Of course, interactivity, timeliness in the quick response and user-friendliness are key elements in an emergency response tool in the 24/7 operational context. As well, the user needs to be able to modify the many input and output parameters (models, output formats, product definitions, etc.) in the most natural and easy way. Responding to emergencies for a wide range of geographical domains and resolutions is an added specialized requirement to meet our needs. The tool needed to be flexible and to operate on global scale events as in the case of volcano eruptions or large nuclear emergencies down to local fine scale for various materials (chemical, nuclear, biological, pollutants, etc.) released in the atmosphere. This specialized need required that the tool be able to integrate "on the fly" different types of data for analysis or visualisation.

This work led to the development of "SPI" (Spherical Projection Interface), a management and visualization tool for global to local scale emergencies. This tool enables users to run a series of transport and dispersion specialized models (CANERM eulerian model, Lagangian models, trajectory models) in a very user friendly way and to obtain quick graphical outputs to support environmental emergency requirements.

Requirements of the management and visualisation tool....

As in any other application, stability, robustness and efficiency are surely very important requirements; nobody wants their application to crash in an emergency response situation where time is the critical factor, but you do want to obtain the dispersion and final product within the shortest delays. Hence, lots of attention has been put into implementing fast and optimal algorithms.

Our needs were, and are still, evolving at an impressive rate. If we had to invest in a tool, we had to be able to make it evolve at our pace; thus be extensible and flexible. We had to be able to build on it and expand its capabilities. We needed not only an application but also a framework that would allow us to fill in new and previously unthought-of requests. It also needed to be easily supported and maintained. A good architecture and design should not complicate the support to the point were it becomes unmanageable by a few persons. The end result had to be kept in mind all along the creative process.

As we know, emergencies do not occur on a daily basis. Therefore, users do not have the opportunity to manipulate the specialized tools on a frequent basis, consequently, the tools needed to be as simple as possible to use.

Current workstations are extremely powerful and making good use of this power has to be a priority. One should exploit the hardware as best as can be. Sometimes, too much time is spent in architecture and design at the expense of simplicity and performance. The overhead of some coded management scheme can sometime be devastating when looking at the performance.

Our computing environment is quite varied and is typical of major modeling center. A wide variety of hardware and operating systems make it important to have tools that are portable. This is of course an important feature in the context of code redistribution.

Options towards meeting the requirements...

Looking at the existing internal software, we had found that they did not have the flexibility we needed neither the extensibility. They also had no concept at all of 3D visualization. On the other hand, external off-the-shelf or freely available scientific software was not an option as they did not meet our specific requirements. It became quite clear that the appropriate option was for us to invest in building our own specific tools. Internal knowledge of the software, from the architecture down to the code was an important factor that we considered. In the event of an emergency situation, it was essential that we had full control and knowledge of the application in order to react quickly in cases of software problems

Choosing the right development path...

The building process was the result of many iterations. Here were a few basic decisions that were taken along the way:

Since many open source projects already existed, we chose to use them instead of rewriting them. These were well built and efficient software and we had access to their code. We used GADL/OGR and PROJ4 to manipulate vectorial or raster geographical data. They allowed us to geo-reference different kind of data on multiple different projection and datum as well as manipulate them and merge the information with our dispersion and meteorological modelling data.

Some tools already developed in-house were also used such as interpolation routines from libraries developed by our Meteorological Research Branch. As well, functionalities of basic geographical information from another in-house project were used as the core geographical database of SPI.

Language was one of the most discussed and argued aspect of the project. Tcl/Tk might not have seemed a good choice to many but it had proven to be very efficient for this project. Its extension architecture is simple and efficient and the Tk package is well known and recognized for its simplicity and efficiency in creating user interfaces. Many extensions were already available and could be used freely. The only problem that might have been pointed out is the execution speed of Tcl scripts but this was simply solved by using C extensions to manage the processor hungry tasks. By using the Tcl/Tk framework, we only had to create the object and functions to manipulate our data and could profit from the already efficient, proven and stable qualities of Tcl/Tk as a control language. By developing the whole application around Tcl/Tk, we benefited from a full featured scripting language to interact with the application. It was also very easy for users to extend the application capabilities to their needs. We only had a single language to learn and use for the application itself, extensions and scripting.

As performance and high interactivity was one of our requirements, we also decided to use OpenGL as the rendering engine in order to take advantage of the hardware acceleration of today's GPU. Using it's counterpart from MESA would also allow us to use the same rendering code for batch mode. The goal here was interactivity. To be able to reach this goal, we fixed a target frame rate of 25 frames per seconds with a minimum under heavy load of 10 frames per second. Whatever the user asked to the application, we had to keep the minimum frame rate with the maximum information and resolution while panning, zooming, flipping or rotating. All of this had to happen on a reasonable system, not top of the line hardware, as it is often too easy to put the pressure on the hardware instead of working with it.

Having minimal human resources to apply on this project also proved to be very efficient since it minimized the size of the project while keeping a low project management overhead. Big and largely distributed project often do not get realized or get delayed since a large part of the resources are spent in management and synchronization between the different groups.

Architecture

Tcl/Tk is used as the control and interfacing language. Within it we built data object in an Object Oriented style through C extensions. Each object is then capable of managing and rendering itself. The idea is to hide the complexities of data input/output and manipulation through Tcl objects so as to be able to do numerous and usually complex tasks with one or two lines of code. For example

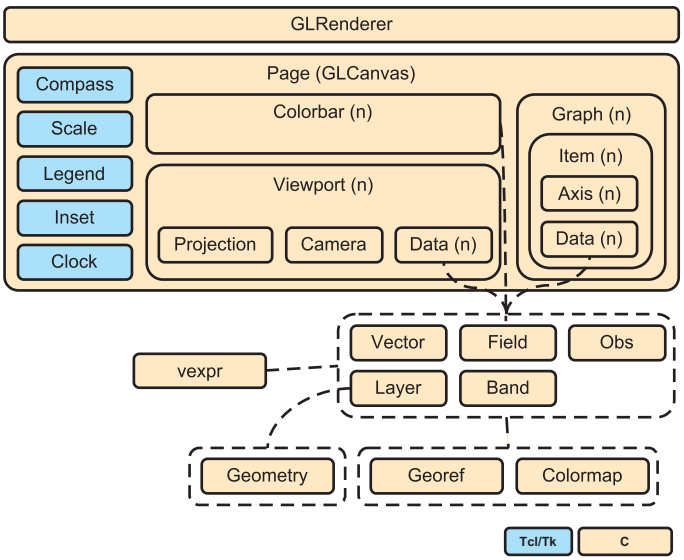
```
fstdfield configure FLD -rendercontour 2 -intervals { 10 20 30 } \
-label 1 -color black -font *-Courrier-*-12-
```

would configure the display of the data "FLD" with a contouring at the 10, 20 and 30 intervals using a line width of 2 in black and putting a label once per contour using Courrier font at 12 dpi. While

```
.page.vp itemconfigure VP1 -data FLD
```

would assign the data "FLD" for rendering in the viewport "VP1" within the page .page.vp

To be able to use hardware acceleration we had to recode the Tk Canvas so as to run in OpenGL. Everything in this new canvas is now using OpenGL for its rendering, the lines, polygons, images items, etc... This allowed us to create new canvas object, like the viewport and graph with hardware accelerated rendering. This conversion also had the effect of simplifying the Tk Canvas code a lot since the management of X Resources like graphical context and pixmaps were completely removed. We have also extended the standard canvas objects with some very useful functionality like transparency and rotation for the texts which could not be done by XWindows.

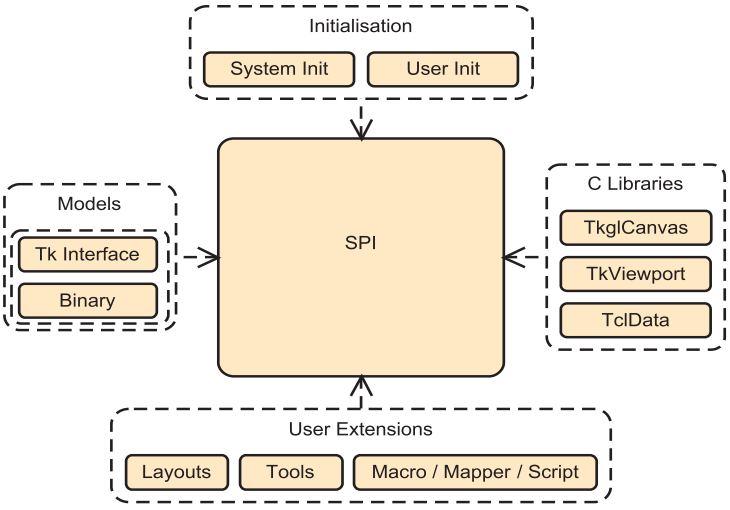


We have devised the concept of pages with the WYSIWYG approach. Everything happens within pages and the application can manage multiple pages. Within those pages, one can insert different objects like viewports, which are used to display geographically related information, or graphs, colorbars, scales, etc... Each of those objects can be placed and scaled interactively so as to create the needed product. Those page layouts can then be saved for later use either interactively or within a script in batch mode.

Each viewport has a projection object assigned to it which defines the geographical situation and information to display as well as functions for transformation from geographical coordinates to pixel space and vice-versa, path projection, distance calculation and more. Each viewport also has a camera object assigned to it. This camera places the viewer around. By using the camera concept, we can have an always 3D approach were you just place the camera on top to look at it in 2D and flip the camera to go 3D. We then have different data objects which can be configured for rendering and passed to the viewport which takes care of localisation and rendering. Each viewport within the same page has the same projection and camera. This allows for visualising different data at the same geographical location since rotating or zooming in one viewport will affect every viewport of the page.

Any viewport can also serve as input selection for different tools. For instance, in cases where the user has to enter the bounding coverage of three different vertical levels for several specific times, he only has to draw this bounding directly on the visualisation viewports for it to be automatically entered within a text bulletin in the specified format. It could also be the selection of a range for a graph or location searching tool.

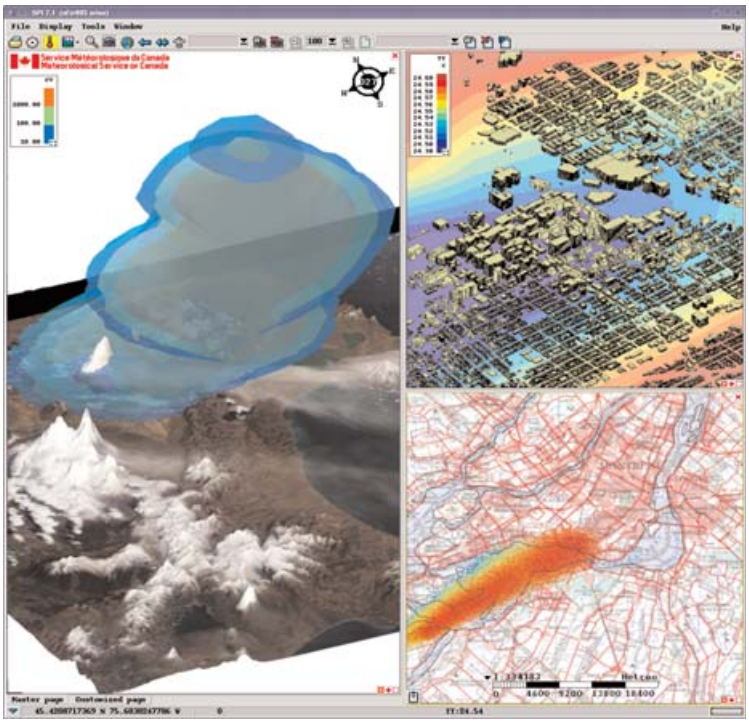
We did not use the common concepts often used in GIS applications which is to layer all the information on top of the other were the layering defines the displaying order. We instead used the dimensional approach were data is displayed were it should be displayed. Higher objects appear over lower objects (looking from the top). All we have to do then is give all of the data to the viewport in any order and it takes care of rendering them were they should be. Some rendering schemes although, have rendering priorities over others like contours, point data, wind vectors, streamlines and others, unless the zbuffer gets activated in which case they become obscured when they should like any other scheme.



Extensibility is provided through different mechanics. The “layout and script” capabilities can be used for product creations, either interactively or in batch processing, to initiate specific processing or anything else that can be thought of. The “tool” functionality defines a very simple architecture to be able to interact with SPI by using different selection mode (picking, range selection, drawing ...) to implement specific user needs. Finally there is the “macro” concept which is a more rigid specification which allows one to create simple processing tasks that can be registered within SPI and called on demand.

Visualisation functionalities

We can display standard scientific visualisations like contouring, labelling, colormap coloring and editing, wind barbs and arrows, trajectories, and others within different products often containing multiple viewports



We also have advanced 3D (Always) visualisation functionalities like iso-surfaces, streamlines, particles, 3D objects/modelling which can all be integrated in the same scene. We can even use vectorial data to do 2.5D extrusions along one of its parameters.

All of these can be animated over time, elevation and different other parameters. We can also do "flyby" animation with some predefined pattern or by entering a custom fly path through key points/frames on which the path will be interpolated.

There are also ways to insert interactive items like value point or picked streamlines which readjust according to the assigned data, thus be animated when animating the data itself, or do direct annotation with basic shapes, geo-referenced or not.

Analysis functionalities

We created a "graph" object which can be used to render different graphs within a page. Again, multiple graphs could also be placed within the page, side by side with any other object. It is actually used to plot time series, vertical profiles, time profiles, cross sections, scatter plots, comparison graphs and frequencies diagrams but could also be used for any other kind of graph. Those graphs, within the visualisation application, are linked to a viewport from which they take their data. Changing the data or animating a viewport, will automatically update the linked graphs and in this way animate the graph itself.

We also implemented a vectorial expression parser which allows us to do calculus on whole matrices with scalar, statistical and even logical operators. As for example, here's how we do the relative humidity calculus in one of our post-processing script

vexpr PV 10^(9.4041-2354.0/(TT+273.0))
vexpr HR (10^(9.4041-2354.0/((TT-ES)+273.0)))/PV

The different data manipulation functions are all created with the same pattern. A single Tcl object for a specific data type which allows for three set of commands is used to do everything one might need. The "configure" command is used to define the displaying parameters of the data when passed to a viewport or graph which could be intervals, contouring or colormap. The "define" command is used to query or set the data specification parameters like the date, the data type and size or even to get access to the data itself. Then there is the "stats" command which is used to manipulate the data as in interpolating the value at a specific point in grid or geographical coordinates, projecting and un-projecting a coordinate or even to recuperate the coordinates of a streamline going through a vectorial field from a specific point.

Managing model execution and outputs

Modelling capabilities are implemented as an extension were a Tk interface is built for each models which takes care of its specific input parameters as well as its execution. Resulting data is managed by the experiment management framework which basically follows the same concept has the "tools" described earlier.

Every model execution is managed in a per event basis. We first create the experiment by specifying the source(s) location and the type of event (volcano, nuclear, fire, etc ...) which will set default parameters for all of the available models, allowing the user to change the minimum number of parameters for a first answer. Later, one can refine the simulations by modifying numerous parameters like the source term, temporal and vertical distribution, etc. All models can use the meteorological input fields provided by the operational Numerical Weather Prediction system on various scales (global, regional, local). The models can all also be executed on different computing platforms such as the IBM super-computer or on Linux workstations and clusters.

Conclusion

Over the past 6 years, CMC's Environmental Emergency Response Division has developed SPI, a sophisticated graphical tool to manage and visualize a series of transport and dispersion models in support to environmental emergencies. This tool is critical to ensure timely and precise response in cases of emergencies.

We planned to build an emergency response tool and came up with more; an infrastructure to build with and which can be expanded and used to fulfill our constantly changing needs.

Although built for environmental emergency response needs the tool has functionalities that could be applied to many other applications among which, general meteorological visualization and GIS data manipulation needs.

The next major improvements will be to start using the GPU's shaders. These open a new era in scientific visualisation where previously impossible or slow techniques can be implemented at real-time interactive frame rate. We are also going to investigate the possibility of using them to do the actual modeling calculations, their performance on parallel arithmetic calculations being orders of magnitude above current CPU capabilities.