GWVis: A Tool for Comparative Ground-Water Data Visualization

Daniel M. Best[a,*,1], Robert R. Lewis[b]

[a] *Pacific Northwest National Laboratory*
902 Battelle Boulevard
P.O. Box 999, MSIN K7-28
Richland, WA 99352
[b] *Washington State University*
School of EECS
2710 University Drive
Richland, WA 99354

Abstract

The Ground-Water Visualization application (*GWVis*) presents ground-water data visually in order to educate the public on ground-water issues. It is also intended for presentations to government and other funding agencies. *GWVis* works with ground-water level elevation data collected or modeled over a given time span, together with a matching fixed underlying terrain.

GWVis was developed using the Python programming language in conjunction with associated extension packages and application program interfaces such as *OpenGL*™ to improve performance and allow us fine control of attributes of the model such as lighting, material properties, transformations, and interpolation.

There are currently several systems available for visualizing ground-water data. We classify these into two categories: research-oriented models and static presentation-based models. While both of them have their strengths, we find the former overly complex and non-intuitive and the latter not engaging and presenting problems showing multiple data dimensions.

GWVis bridges the gap between static and research based visualizations by providing an intuitive, interactive design that allows participants to view the model from different perspectives, infer information about simulations, and view a comparison of two datasets. By incorporating scientific data in an environment that can be easily understood, *GWVis* allows that information to be presented to a large audience base.

Preprint submitted to Computers & Geosciences *February 8, 2011*

---

[*] Corresponding author

Email addresses: `daniel.best@pnl.gov` (Daniel M. Best)

[1] Tel: 509-372-6728 Fax: 509-375-4933

Key words: Ground-water, Hydrology, Visualization, Python

1. Introduction

Visualization of spatially-distributed data in general – and ground-water flow data in particular – is a growing field of research. Prior to three-dimensional graphical computer models, ground-water was most often represented as two dimensional maps. Hydrogeologists now have research tools that allow for analysis of complex problems in three dimensions such as Li and Liu (2006). These research models have been created for use by domain experts.

Compieta et al. (2007) point out that recently, interaction techniques for complex models have been advancing in usability. *GWVis* attempts to stand as an intermediary between what we will refer to as the classic static presentation-oriented (SPO) two dimensional maps and the dynamic research-oriented, (DRO) three dimensional visualizations. The following sections describe the purpose and the scope of work to build a solution to this problem. We then provide background on the current field of ground-water visualization, describe our solution to the problem, and finally present our results.

1.1. Purpose

This project provides a visual analytic tool that can be used for presentations and public dissemination. Visual analytic tools used to convey information, such as *GWVis*, are attractive because of the way human perception operates, allowing aspects of perceiving information to happen without conscious thought, as discussed by Card et al. (1999). Maps harness this natural human capability to present large datasets which would normally produce a cognitive overload.

Our aim is to produce a presentation-ready visual tool that allows comparisons of ground-water flow simulations. To be presentation-ready, the tool will be a simplified visualization that conveys ground-water concepts without the use of complex interaction requirements. Comparison is a key element of *GWVis*, one that sets our tool apart from other visual tools. The capability to compare two simulations is an advantage to ground-water researchers utilizing the U.S. Geological Survey modular finite-difference flow model (*MODFLOW*) system developed by McDonald and Harbaugh (2003).

Instead of reading text files and comparing numeric values, we provide the capability to immediately present differences visually. This includes the ability to compare time-varying data and simulations by animating the differences.

1.2. Scope of Work

GWVis's audience is comprised of individuals who are interested in the results of ground-water research. This interest is based on the desire to understand more about the environment: To know, for example, the result of adding more wells to an aquifer, or the need to perform further research when a model does not predict data accurately.

Initially, *GWVis* uses data from the the study conducted by Hsieh et al. (2007), which consists of aquifer elevation data for head and underlying terrain, along with a multi-attribute dataset for the Spokane River that includes depth, width, and inflow. This data uses the widely-used *MODFLOW* data format, but *GWVis* could easily be adapted to any other format that provides similar data.

In this particular case, the aquifer head elevation data consists of three aquifer layers on a 256 by 172 grid covering 326 square miles of the Spokane Valley-Rathdrum Prairie aquifer. All three layers have information recorded for 15 years and one month between September 1990 and September 2005.

The iconography used in GWVis was obtained through the interaction with a researcher of hydrology and by evaluating various methods of color encoding the model. While this has yielded a visually pleasing image, further work needs to be done to relate the information users are seeing to what they are expecting to see. A rich source of information is found in the field of cartography. The relationship between cartographic symbolism and 3D ground-water visualization is important, however it is not the focus of this work. Future enhancements based on such research can be incorporated into the system without much adaptation of the code (see Section 7).

2. Background

Technologies that allow for the creation of three dimensional visualizations were needed to implement *GWVis*. In addition, *GWVis*'s development benefited from "rapid deployment" technologies that reduce the time between the identification of the need for a feature by the domain expert and that feature's implementation. This participatory involvement of the domain expert was critical to its success, reducing the time needed between discussing additions with the domain expert and gaining insight into their effectiveness.

In this section, we will discuss all of these technologies.

2.1. OpenGL

OpenGL (see Shreiner et al. (2005)) is an application program interface (API) to graphics hardware that allows the programmer to produce interactive three dimensional applications. *OpenGL* is the standard for three dimensional graphics. Many popular three dimensional visualization tools, such as *OpenDX* (Thompson et al. (2004)), use *OpenGL* as their foundation.

As Shreiner et al. (2005) point out, *OpenGL* actually works on a very basic level of geometric primitives: points, lines, and triangles. Visualization systems build the shapes the user sees out of typically thousands of these primitives.

The API is not tied to a specific windowing or operating system, thus providing cross-platform capability. However, *OpenGL* is intentionally output-only: The user must provide the input part of the graphical user interface (GUI) by other means.

OpenGL has kept pace with the rapid advance of graphics hardware technology. Its most recent versions allow for such features as programmable shaders and vertex buffer objects. The latter are especially relevant for *GWVis*.

2.2. Python

Python (see, for example, Lutz (2007)) is a high-level programming language with attributes that make it attractive for use in visualization (see Langtangen (2006), for example). The pseudocode-like syntax of the language and built-in support for high-level constructs like lists, sets, and dictionaries results in highly-readable, elegant code that can be readily shared between programmers.

While easy to read, Python remains a compact language, which results in less code doing more work, as opposed to, say, a C implementation. For *GWVis*, this represents the ability to make rapid changes and extensions in response to domain expert comments.

Because of Python's popularity there are many modules to extend its functionality. These include "wrappers" for existing APIs written in other languages. One such wrapper, *pyOpenGL*, exists for *OpenGL* and *GWVis* uses it extensively.

2.3. NumPy

NumPy is a numerical extension to the Python programming language, and is a cornerstone of scientific computing in Python, as discussed by Langtangen (2006) and Varoquaux et al. (2008). A core component of *NumPy* is the `ndarray` object, which represents arrays of arbitrary size and dimensionality and a wide variety of base types (e.g. integer, single precision float, double precision float, etc.). *NumPy* also provides Python access to common scientific calculations such as vector arithmetic and linear algebraic operations over these arrays.

Also present in the code is the ability to "slice" an array: to extract or operate on a subarray of a larger array without explicit (and, in Python, expensive) looping (see NumPy (2006) and Oliphant (2007)). *NumPy* also allows one array to be used to map the values of another, again without looping.

Overall, *NumPy* allows Python developers to program numerically intensive programs while maintaining a reasonable level of time performance.

2.4. wxPython

wxPython is a GUI API built upon extensions created based on *wxWidgets*, a windowing framework API written in C++, as described in Rappin and Dunn (2006). Because of this, *wxPython* retains the cross platform capabilities of the base API.

Like many GUI implementations, *wxPython* is an event-driven environment where user inputs such as button clicks and mouse drags or system inputs such as redraw requests or timer expirations cause some action to be taken when the corresponding "event" occurs. *wxPython* calls user-specified "handler" functions to respond to such events asynchronously. In fact, after initialization of its internal data structures, *everything* a *wxPython*-based (or *wxWidgets*-based, for that matter) program does takes place in a handler of some kind.

wxPython includes support for *OpenGL* "canvases", which are rectangular areas in which *OpenGL* graphics output can take place with little or no overhead from the GUI.

3. Related Work

There are many applications and visualizations that cover aspects of viewing information associated with computational hydrology. Visualizations range from hand-drawn two dimensional models to complex interactive three dimensional models. The following selection of visualizations exhibit attributes that relate to our work.

3.1. Static, Presentation-Oriented (SPO)

Examples of ground-water maps can be seen in the U.S. Geological Survey (USGS) report by Hsieh et al. (2007). In the study there are many different views of the same imagery of the Spokane Valley-Rathdrum Prairie Aquifer. Each image shows different information regarding the study, such as areas of shallow bedrock, water purveyor service areas, areal distribution of water purveyor wells, and sewer hookup density. Figure 1 is one of these images. The drawing shows the difference between simulated and measured heads for the aquifer over time.

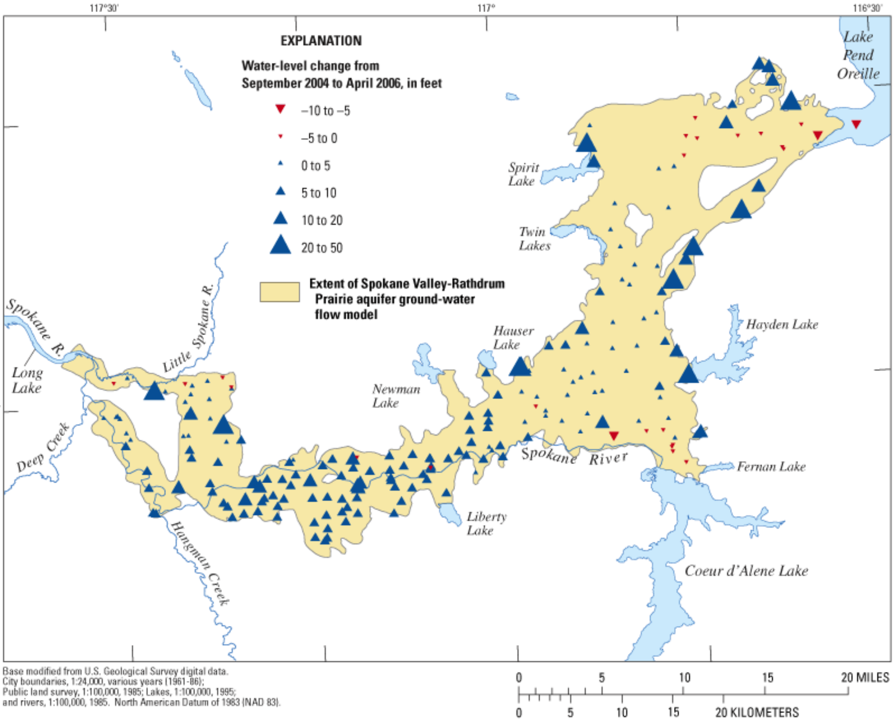
Multiple dimensions of data are difficult to represent on the same image, hence there are 51 different figures in the report, each of which has some important role in understanding the information being presented. While the information on any particular figure is being presented in an understandable manner, it is difficult to see correlations of the various attributes all at once. A reader of the report must turn to one particular view and then back to another to make any connections between them.

3.2. Dynamic, Research-Oriented (DRO)

A second domain of study for ground-water visualization centers around research. Analysts research topics such as pollution plumes and evaluating available resources.

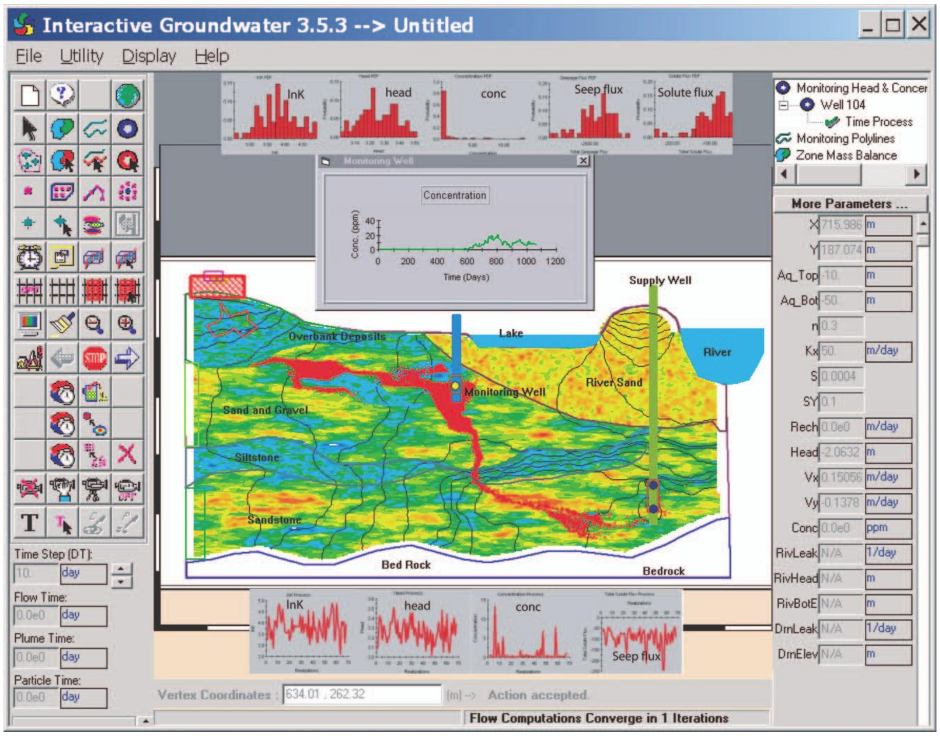
The performance of today's workstations makes feasible a new type of analytical tool. Research models that a user interacts with have been developed, such as *Interactive Ground-Water* (*IGW*) (Li and Liu (2006)), *Visual MODFLOW* (Schlumberger Water Services (2009)), *Groundwater Vistas* (Groundwater Vistas (2009)), and *Aquaveo* (formally *GMS*) (Aquaveo (2008)).

Let us consider *IGW* in particular. The model simulation can be stopped and adjusted at any time. It incorporates "level-of-detail" control: As a



Base modified from U.S. Geological Survey digital data.
City boundaries, 1:24,000, various years (1961-86);
Public land survey, 1:100,000, 1985; Lakes, 1:100,000, 1995;
and rivers, 1:100,000, 1985. North American Datum of 1983 (NAD 83).

Figure 1: Changes in Ground-Water Levels for Spokane Valley-Rathdrum Prairie Aquifer.



Time Step (DT): 10. day

Flow Time: 0.0e0 day

Plume Time: 0.0e0 day

Particle Time: 0.0e0 day

X 715.986 m

Y 187.074 m

Aq_Top -10. m

Aq_Bot -50. m

n 0.3

Kx 50. m/day

S 0.0004

SY 0.1

Rech 0.0e0 m/day

Head -2.0632 m

Vx 0.15056 m/day

Vy -0.1378 m/day

Conc 0.0e0 ppm

RivLeak N/A 1/day

RivHead N/A m

RivBotE N/A m

DrnLeak N/A 1/day

DrnElev N/A m

Vertex Coordinates : 634.01 , 262.32 (m) --> Action accepted.

Flow Computations Converge in 1 Iterations

Figure 2: *IGW* Modeling Interface.

user zooms in and out of the model, the amount of information presented is increased or decreased accordingly to avoid clutter.

To allow for general research use, *IGW* has been developed with eight modeling, analysis, and visualization engines. The system is a powerful analytic tool allowing the researcher to perform stochastic analysis, modify properties, and develop hierarchically-nested sub-models, among other tasks. Figure 2 shows *IGW*'s GUI for changing model parameters.

3.3. Spatially Distributed Data Interaction

Recent research in the field of spatially-distributed data visualization includes human factors. A major contributor to the development of this field is Google Earth. By default, Google Earth initially presents an image of the world. It is then left to the user to either type in a location or to use the mouse inputs to navigate. Navigation within the model is accomplished by having the user click, grab, and move the model with the mouse, producing an easily learned movement.

Compieta et al. (2007) look to eliminate the visual / cognitive overload possible with DROs by providing two interfaces. The first uses the Google Earth style interaction, while the other incorporates the analytic tools desired by a domain expert to data mine the information.

In that work, the human computer interaction (HCI) of the visualization is of special interest. The application looks to take advantage of the human's ability to perceive visual patterns and interpret them, and to provide a means of interaction that is intuitive. System interaction is developed through the use of Google Earth technology.

4. ***GWVis***

The implementation of *GWVis* took lessons learned from a prototype built in *OpenDX* (Thompson et al. (2004)) and expanded upon them. The program is written in Python using the *wxPython*, *pyOpenGL*, and *NumPy* packages heavily. Focus areas for *GWVis* to ensure its usability are the basic interaction mechanics, color encoding of the model, the ability to compare scenarios, and efficiency.

To provide a solution that is usable, informative, and easy to comprehend, *GWVis* incorporates attributes from SPO, DRO, and HCI domains. Some elements are shared by multiple domains.

Attributes *GWVis* uses from the SPO domain are:

- visualization of change in aquifer elevation information,
- simple layout of information using data encoding (referred to as thematic cartography), and
- geographical data that shows terrain, surface water, and aquifer location.

GWVis uses the following attributes from the DRO domain:

- incorporation of time and the ability to control the animation,
- ability to interact with the model during the animation,
- ability to pan and zoom into specific areas of interest, and
- displaying some analytic features.

Attributes included from HCI visualizations (which take cues from classic cartography) are:

- interaction mechanism,
- limiting cognitive overload, and
- usability-focused development.

4.1. Interaction

Beyond the initial rendering of the model, interaction is the attribute that most affects how people perceive the utility of *GWVis*. Interaction modes such as being able to zoom in to a particular section and how to move about the model were continuously adapted during development.

We started with classic navigational model (see, for example, Hill (2000)) which allows six degrees of freedom. Three of them are positional and are, in this case, longitude, latitude, and altitude. The other three relate to orientation: roll (rotation about the x-axis, taken to be the initial direction of travel), pitch (rotation about the y-axis), yaw (rotation about the z-axis).

Following Google Earth (when not in "flight simulator" mode, which we judged not appropriate for this application), we removed the "roll" degree of freedom: The vertical axis of the viewer is always perpendicular to the (geometric) horizon, except when looking straight down, when roll and yaw

Table 1: User Interaction

Interaction	**Effect**
mouse click and drag left or right	changes the compass bearing of the viewer (yaw)
mouse click and drag up or down	changes the elevation of the viewer's horizon (pitch)
mouse scroll wheel forward or backwards	moves the user forward or backward at a constant altitude
control button + mouse scroll wheel forward or backwards	moves the user towards or away from center of view (dolly)

are equivalent. Table 1 shows the various ways a user can interact with *GWVis*:

Interaction with the model using these methods allows a user to dolly in (move towards the center of view), shift the view left, right, up, and down, and move through the model at a constant altitude. All canvases are synchronized so that the user is viewing the same section of the aquifer at the same time in all of them, providing a mechanism for quick comparison of the images being shown.

Finally, the model is be animated month-by-month. The animation is controlled using typical media player control buttons (play, forward, backward, rewind, and pause). Playing the animation iterates through all data so a user can see which areas of the aquifer are changing and which are remaining the same.

4.2. Encoding Elevation with Histogram Equalization

The ground-water elevation for the *GWVis* development data has a fairly small range: from 1400 to 2600 feet above sea level compared to 66.25 miles (349,800 feet) in horizontal extent. Most of the data values are clustered closely together. However, values change rapidly at the aquifer extremities. Because of the nonlinear data distribution, the color encoding of the model based on elevation data should not be a simple linear interpolation. To eliminate this issue, we incorporated color histogram equalization.

Histogram equalization is a standard means to uniformly encode a set of data points having a non-uniform histogram, as seen in Bradsky and Kaehler (2008). We apply this technique to evenly distribute the elevation information and encode elevation as the saturation value of blue. This technique produces a range of color from white to blue. This allows the user to infer which sections of the aquifer are higher or lower than others.

4.3. Comparison

GWVis provides the ability to compare side-by-side scenario datasets, including their animations. SPO visualizations can provide this by showing two visualizations next to each other, but do not display differences as visualized quantities. DRO visualizations generally do not provide visualization for this purpose, either. We assert that showing the differences between two scenarios allows for basic analysis that would be beneficial to the intended audience.

Figure 3 shows *GWVis*'s layout: a small conventional pull-down menu and control area at the top, below which are two side-by-side “rate” canvases showing the two data sets individually. Below that, taking up approximately the bottom 2/3rds of the window, is the main “difference” canvas.

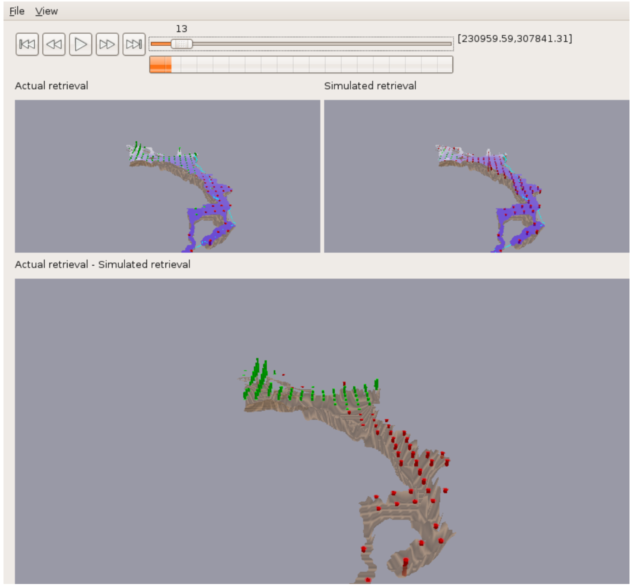
4.3.1. The Rate Canvases

In the rate canvases, *GWVis* uses color and position encoding to convey information about how the elevation for the visualization depicted on each canvas is rising or falling, as seen in Figure 4. The column is either green for increasing or red for decreasing elevation. An user can use this capability to determine which of the sets of data is changing the fastest where.

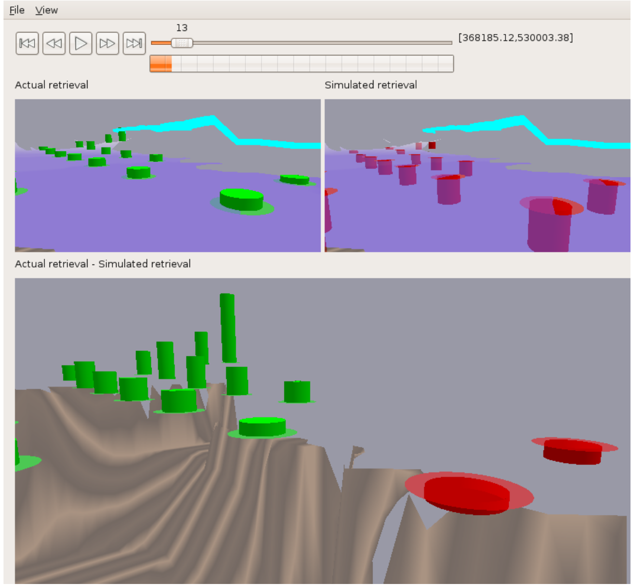
Because there are about 44,000 grid cells for each month, a large number of columns could be shown. However, the domain expert advised that we simplify the visualization by reducing the number of columns depicted. To accomplish this, we take the mean of each 7 by 7 block of data points and draw a column with that value in the center of the block. Reducing the number of columns simplifies the visualization, reducing information overload.

4.3.2. The Difference Canvas

This canvas shows the difference between the two scenarios at the current time step as one image (Figure 5). The right hand aquifer's elevations are subtracted from the left hand aquifer's. If, as we expect will be typical, the upper left quadrant is original flow data and the upper right quadrant is a



(a) Initial View



(b) Moved In

Figure 3: *GWVis* Visualization Interface.

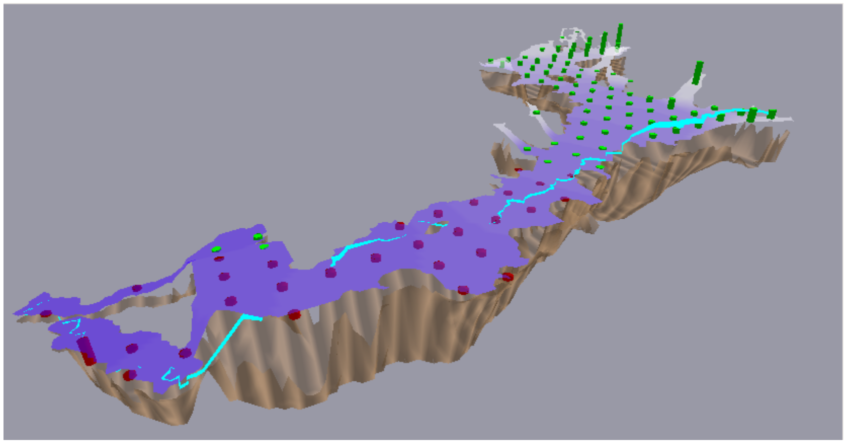


Figure 4: *GWVis* Rate Canvas.

simulation, this answers an important "what if" question for the user: "How would the changes being simulated affect the aquifer?"

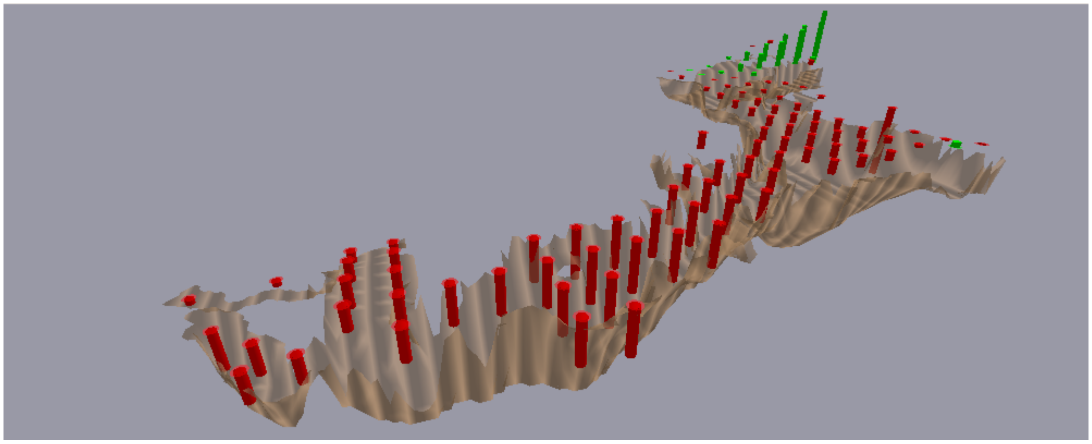


Figure 5: *GWVis* Difference Canvas.

4.4. Text

Currently, minimal text has been added to the tool. The intent is to convey the information needed using predominantly graphical means. Textual that has been added are titles describing the current scenarios and noting that the bottom panel is a comparison of the two rate panels. In addition, the latitude and longitude of the current mouse position on the canvas is shown. Finally, the current month is displayed above the animation progress bar.

4.5. Time Efficiency

From the beginning, we identified time efficiency, primarily in the form of the response time to user input, as the major criterion for *GWVis*'s acceptability. In this section, we will discuss changes we made during the program's design and implementation that most affected time efficiency.

4.5.1. Overcoming Interpreter Overhead

Python, being an interpreted language, is not intrinsically as fast as a compiled language. Considerable work has been done, however, in providing tools such as *NumPy* (see Section 2.3) whose effective use can minimize the interpreter penalty. We felt that the benefits of the language (see Section 2.2) justified the additional effort required to overcome the speed penalty.

4.5.2. Improving Response Time

Each input frame of data contains 256 × 172 = 44,032 values and there are 181 such frames in our target data set. Although there is less than 50% valid data in each frame (the aquifer does not cover the entire rectangular grid), it is still necessary to read in the full grid due to the file format. There is also terrain ("bottom") data on the same grid, but that needs to be read in only once, as it is constant for each frame.

To provide fast animation, we found it advantageous to read in all 181 frames at startup time, incurring a small penalty in startup time. Having all of the data in memory allowed *GWVis* to approach interactive rates, but obtaining real interactivity required taking advantage of improvements offered by *OpenGL*, "vertex buffer objects" in particular. This relatively new feature allows the program to share graphical memory with the graphics processing unit (GPU), leading to extremely fast interaction, both for user-directed inspection and animation.

Combining vertex buffer objects with *NumPy*'s slicing feature allowed us to re-use the same longitude and latitude grid coordinates while replacing only the elevations (and redrawing the bottom data) from frame to frame. This virtually eliminated the interpreter overhead, as all of the critical processing took place either within *NumPy* (which is written in C) or the GPU itself.

5. Results

The goal of *GWVis* is to produce a presentation-ready visual tool that allows comparisons of ground-water flow simulations. This is accomplished by incorporating elements of DRO and SPO visualization and bridging the two methods. Key attributes that set *GWVis* apart are its ability to compare simulations and the style of interaction that lends itself to presentations.

5.1. Comparison

Of all the visualizations mentioned in Section 3, none provide the capability to show a comparison between models. Many have the capability to run multiple simulations in the same workspace. However, this mechanism does not yield the same functionality that *GWVis* provides.

It is easier to come to a conclusion to a problem when the correct information is present in an easy-to-see format, as shown by Kirsh (2000). Figure 6 shows an example of this point. It is easier to sum a list of numbers using ruled paper rather than randomly placed on a page.

The significance of this is that *GWVis* presents differences visually so the information is apparent. Other visual tools rely on the observer to compare one simulation to the next by looking at both windows. Each canvas in *GWVis* has the same transforms applied to it so that the views are synchronized. DRO views however, are not synchronized. Keeping the view the same enforces the ability to compare simulations together, viewing areas of interest and only having to navigate to that area in one canvas.

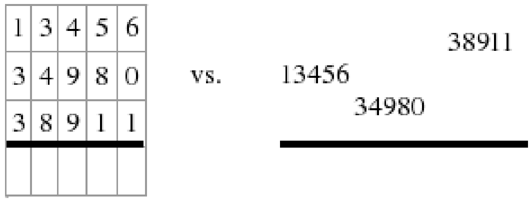


Figure 6: Addition Ease Comparison Concept. (from Kirsh (2000))

Evaluation of simulation differences provides functionality for *GWVis* that is useful beyond presentations. An analyst can run a *MODFLOW* simulation and compare its results to field data, or to those of another simulation.

5.2. Presentation

GWVis is intended for use as a presentation tool in addition to basic analytic capabilities. The information shown in the visualization is kept minimal due to the main audience for the tool. Kirsh (2000) explains that when a visualization adds more information than is needed, or the information is hard to find and comprehend then the viewer will suffer from information anxiety. This anxiety may lead to cognitive overload.

DRO models provide complex and powerful environments that allow analysis of flow simulations. Presenting a DRO model to the public or government and funding agencies will provide too much information for the purpose of telling a story about ground-water flow. There are buttons, labels, and capabilities that clutter the visualization. These are valid attributes of an analytic tool, however they can produce anxiety in individuals who are not trained in the environment.

SPO visualizations can be created in a way that minimizes information anxiety as *GWVis* does. However, the SPO models are unable to adapt their view to inquiries about specific areas of the aquifer. If the image was not created prior to the presentation, then the audience will be unable to view that requested information. *GWVis* can change its view and focus in on areas of interest in an aquifer. The navigation is learnable so if desired the viewer can interact with the tool themselves.

6. Conclusions

In conclusion, we visualize ground-water flow using a set of features that allow for analytic capability, while minimizing the effect of cognitive overload. Analytic capability is provided two ways. First, the rate of change canvases show a viewer how a simulation changes over time. The visualization include elements of ground-water, such as water elevation, river position, and underlying terrain, that are helpful in the decision making process. Rate of change allows a researcher to have an overall picture of what the aquifer is doing. Secondly, we provide a difference canvas which allows quick analysis of changes made in various simulations. Being able to compare two scenarios allows a researcher to analyze which changes to flow have affected a simulation in the ways they require for their study.

The visualization interface is kept simple, relying on user input to move around the model. Interacting with the model is developed in a way the users are able to learn quickly. Ease of interaction can be attributed to the use of standard mechanisms for input. While uncluttered with data, *GWVis* provides enough information to allow presentations and basic flow analysis.

We have adapted attributes of SPO and DRO visualizations and added a additional capabilities to produce a tool which allows comparison, is adaptable to *MODFLOW* scenarios, and is interactive.

7. Future Work

GWVis is an ongoing research project. As such, new topics continue to be thought of that have yet to be explored. Some key areas that would be beneficial to the model include:

1. Close the gap between the bottom of the aquifer and the head. Currently when viewing the model, a user sees a layer of water elevation data. It would be more accurate to represent the area between the head of the aquifer and the bottom as volume of water.
2. Predefined fly-through routes. The capability of the model to move anywhere in the three dimensional space is already part of the project. It may be beneficial for display purposes to have a custom "flight path" file that would start the user in a location on the model and move without the need of user input.
3. Animation generation. By combining flight path animations, comparison animations, and text display, an overall animation could be created. This type of animation could be used for presentations, informational web sites, or unattended video displays.
4. Addition of data dimensions. *GWVis* only has access to river, aquifer head, and aquifer bottom data. More data can be incorporated into the model such as the location of wells, active areas of the aquifer, and types of soil. It would benefit the model when done in a way that does not complicate the visualization further.
5. Integration with Google Earth. The aquifer information would be treated as a body of water and placed in the appropriate location. This would give the added benefit of utilizing functionality that is currently present in Google Earth such as satellite imagery.
6. Keyboard input. Currently input is through mouse movement and clicks. For ergonomic reasons it would be beneficial to accomplish the same tasks through keyboard input.

7. Incorporate methods from cartography. The relationship between the height field visualization in GWVis and maps is a close one. Elements from cartography can be incorporated into the model to allow for familiar and effective symbolism of information.

References

Aquaveo, 2008. GMS. `http://www.aquaveo.com/gms`, [accessed 19 April, 2009].

Bradsky, G., Kaehler, A., 2008. Learning OpenCV, O'Reilly Media, Inc., Sebastopol, CA, USA, 576pp.

Card, S. K., Mackinlay, J. D., Shneiderman, B., 1999. Readings in Information Visualization: Using Vision to Think, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 686pp.

Compieta, P., Di Martino, S., Bertolotto, M., Ferrucci, F., Kechadi, T., 2007. Exploratory spatio-temporal data mining and visualization. Journal of Visual Languages and Computing 18 (3), 255–279. doi:10.1016/j.jvlc.2007.02.006.

Groundwater Vistas, 2009. Groundwater Vistas. `http://www.groundwater-vistas.com/gwv/detailed_description.php?products_id=43`, [accessed 19 April, 2009].

Hill, F. J., 2000. Computer Graphics Using OpenGL, Prentice Hall PTR, Upper Saddle River, NJ, USA, 922pp.

Hsieh, P. A., Barber, M. E., Contor, B. A., Hossain Md., A., Johnson, G. S., Jones, J. L., Wylie, A. H., 2007. Ground-water flow model for the Spokane Valley-Rathdrum-Prairie aquifer, Spokane County, Washington, and Bonner and Kootenai Counties, Idaho. Scientific Investigations Report 2007-5004, U.S. Geological Survey, 78pp.

Kirsh, D., 2000. A few thoughts on cognitive overload. Intellectica 30, 19–51.

Langtangen, H. P., 2006. Python Scripting for Computational Science, Springer Verlag, Berlin, 780pp.

Li, S.-G., Liu, Q., 2006. A real-time, interactive steering environment for integrated ground water modeling. Ground Water 44 (5), 758–763. doi:10.1111/j.1745-6584.2006.00225.x.

Lutz, M., October 2007. Learning Python, 3rd edn., O'Reilly Media, Inc., Sebastopol, CA, USA, 1216pp.

McDonald, M. G., Harbaugh, A. W., 2003. The history of MODFLOW. Ground Water 41 (2), 280. doi:10.1111/j.1745-6584.2003.tb02591.x.

Numpy, 2006. Guide to NumPy. NumPy., Provo, UT, USA, 378pp.

Oliphant, T. E., 2007. Python for scientific computing. Computing in Science and Engineering 9 (3), 10–20. doi:10.1109/MCSE.2007.58.

Rappin, N., Dunn, R., 2006. wxPython in Action, Manning Publications Co., Greenwich, CT, USA, 620pp.

Schlumberger Water Services, 2009. Visual Modflow. `http://www.swstechnology.com/software_product.php?ID=29`, [accessed 19 April, 2009].

Shreiner, D., Woo, M., Neider, J., Davis, T., 2005. OpenGL Programming Guide, 5th edn., Addison Wesley, Upper Saddle River, NJ, USA, 928pp.

Thompson, D., Braun, J., Ford, R., 2004. OpenDX Paths to Visualization, 2nd edn., Visualization and Imagery Solutions, Inc., Missoula, MT, USA 206pp.

Varoquaux, G., Vaught, T., Millman, J. (Eds.), 2008. Proceedings of the 7th Python in Science conference. Pasadena, CA, USA, 78pp.