Publishing Time Dependent Oceanographic Visualizations using VRML

Dr. Jonathan C. Roberts

Computing Laboratory,
University of Kent at Canterbury,
Kent, UK, CT2 7NF

J.C.Roberts@ukc.ac.uk

Proceedings of the Fifth UK VRSIG Conference, Exeter 5th September 1998
Edited by Simon Grange.

Abstract:

Oceanographic simulations generate time dependent data; thus, visualizations of this data should include and realize the variable 'time'. Moreover, the oceanographers are located across the world and they wish to conveniently communicate and exchange these temporal realizations. This publication of material may be achieved using different methods and languages.

VRML provides one convenient publication medium that allows the visualizations to be easily viewed and exchanged between users. Using VRML as the implementation language, we describe five categories of operation. The strategies are determined by the level of calculation that is achieved at the generation stage compared to the playing of the animation. We name the methods: 2D movie, 3D spatial, 3D flipbook, key frame deformation and visualization program.

Key words: VRML, Visualization, Oceanography, Animation

1. Introduction

Oceanography simulations generate inherently time dependent datasets describing, for example, the movement of water, mineral concentrations and fish-stock over time. Thus, such simulation models necessarily include many variables, generate large amounts of data, and may take many hours to calculate. It is this data that our oceanographic clients require to view. The concerned oceanographers are located all over the world, and wish to manipulate and interrogate the time dependent visualizations to better understand their complicated multivariate data. The web thus provides a convenient publication medium.

Traditionally, the publication of these visualizations have tended to be by video or image-based animations, that may be easily generated, published and repeatedly viewed. However, these animations inevitably have fixed viewpoints, inhibiting the user from directly manipulating the information, and the user consequently misses out on the 'richness of the information content'. Essentially, oceanographers require methods to encode the time dependent information into a visualization with unspecific or alterable viewpoints that may be readily published.

Within this context, we define a publication to describe the act of making something, in this case oceanographic visualizations, publically known and accessible. The publications operate independent of any remote client, and display a specific (pre-determined) realization. Thus, we exclude both visualization systems and the visualization services, such as the Web visualization server of Wood et al [1] and the 'cthru' ocean visualization program [2] using the CAVE virtual environment.

The VRML97 language [3] provides an ideal format for describing visualizations [1][4][5]; indeed it is an ideal format in which to publish time dependent oceanographic visualizations. We have discovered different methods that enable the time dependent oceanographic simulation data to be visualized and published in three dimensions with this VRML format. In essence, the problem becomes one of 'not only describing the motion of the object', but also of 'visualization modelling' to effect and change the shape of the objects over time.

2. The categorization

We categorize the different methods into five groups. In order to generate a three dimensional time dependent visualization the simulation data is first visualized, generating a time-dependent model that may be rendered into the displayed realizations. The categories are determined by the level of evaluation that is achieved when the visualization is re-played by the client, compared to when it is generated by the publisher, see Table 1.

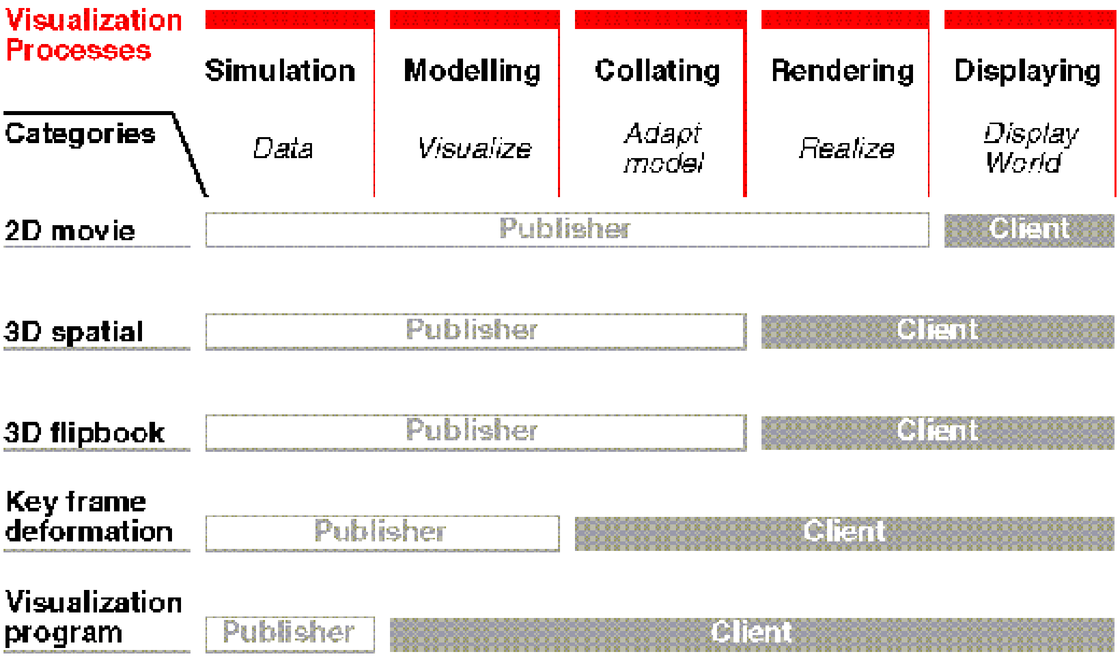

Table 1 *The five categories are classified by the level of evaluation achieved by the Publisher compared to that at the client-side.*

The table depicts the five publication strategies on the left, with five processes from the visualization and display pipeline along the top. The *simulation* process generates the data to be visualized. The *modelling* part provides the visualization process and includes methods to select and filter data points of interest, and map graphic objects (such as hue, transparency, position and geometry) onto each data value. The modelling process generates multiple models that are collated into one world and then *rendered* to generate the images that are finally *displayed*.

The 2D movie category, for example, provides a two dimensional animation that is simply re-played. This necessitates in creating, rendering and collating each frame of the model to provide the animation. Conversely, another method incorporates a script with the publication to describe the 3D visualization by deforming the realized object over time. This is represented by the ‘key frame deformation’ category and relies on an ability to encode the visualized motion as a model behaviour.

We describe each category, include some information about how the published visualization may be generated, discuss their relative advantages and disadvantages, and present example implementations.

These ideas are illustrated using data from the marine simulation of the University of Bergen and Institute of Marine Research. The model is used to study the response of marine systems to variations in nutrient supplies [6]. Our visualizations depict the growth and movement of plumes of algae in an ideal fjord. IRIS Explorer is used to both visualize the oceanographic data and to generate the VRML models, and the animations are displayed and rendered using CosmoPlayer.

3. 2D movie

This category includes the traditional two dimensional animations. The frames are individually rendered and sequentially collated to produce the animation.

3.1 Implementation

Many visualization systems provide mechanisms to store and save two-dimensional images of the realizations. Thus, animations of the data are easily generated. For example, in IRIS Explorer a FOR loop may be used, connected to other modules, to iterate through the data frames, render the realization and either save a rendering of each time-frame (to generate animations using post-production techniques) or directly compose an animation file.

Most browsers support MPEG and GIF animation types and many other types can be viewed using an external application or plugin. Moreover, it is possible to include such animations into VRML scenes. This may be achieved by generating an MPEG animation and defining a *MovieTexture* node. Additional programing is required to control the speed, direction and size of the animation.

3.2 Advantages

The movie may be described in many different formats and easily replayed; the animations may be readily exchanged between oceanographers and played outside the confines of a web browser.

By including the movie within a VRML scene the animation may be related to other objects in the world. For example, two animations depicting different variables from the simulation may be synchronised together to allow the information to be related. Additionally, it is possible to compare and correlate the animations to three dimensional objects placed in the scene. For example, Risch et al [7] present a virtual environment that displays and links together multiple forms of multimedia. These ‘multiple representations’ of the same information are a useful technique to correctly understanding complex data [8]. Such a mixture of video and VRML is used by Katkere et al [9] who display video frames, taken from multiple camera angles, with corresponding interactive virtual-worlds; to dynamically depict three dimensional representations of people as they move across the camera's view.

If the animations are being used to 'teach' or show specific aspects of the simulation, then these fixed viewpoints may be more beneficial, as the instructor and user would always be viewing the same information through the same viewport.

3.3 Disadvantages

These two dimensional animations have fixed pre-calculated viewpoints, this may generate a misunderstanding of the visualization as parts of the information may be obscured, thus, generating a misinterpretation of the underlying information. Incidentally, after the images have been generated from the fixed-viewpoints it is possible to post-produce additional 'inbetween' images from neighbouring viewpoints using morphing and interpolation techniques [10].

3.4 Examples

The fjord data may be easily presented as a 2D animation. Figure 1a,b shows two visualizations of the fjord, with the mouth of the fjord to the left of each picture. The pictures represent different snapshots in time. The left picture is realized using contours, with an isosurface round a plume of algae. The right picture uses a volume rendering algorithm, with a non-linear transparency mapping, to depict the algae concentrations. Figure 1c depicts the animations within a VRML world.

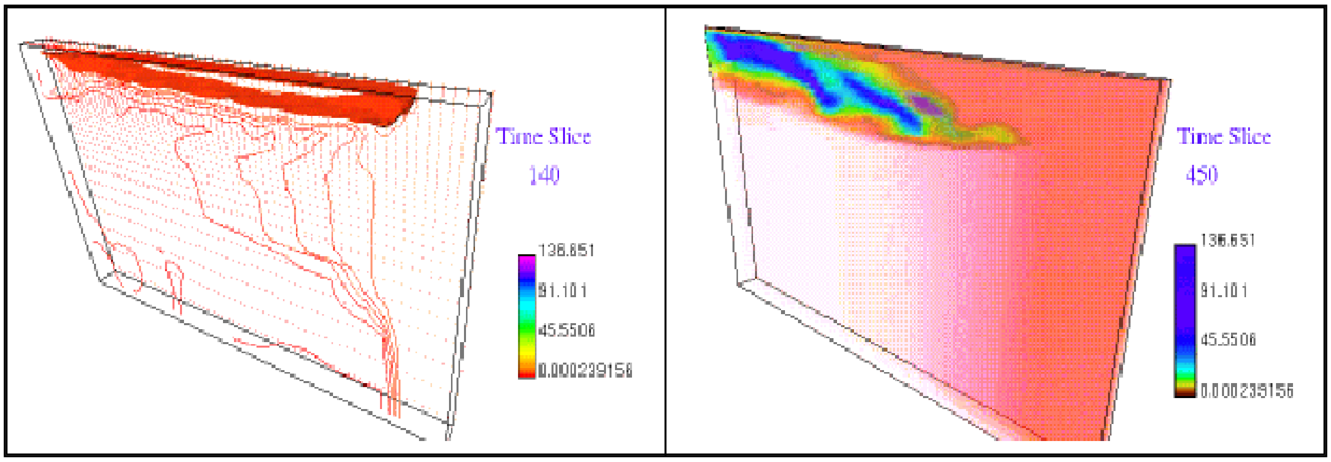

Figure 1a,b *Two dimensional animations of the fjord simulation data.*

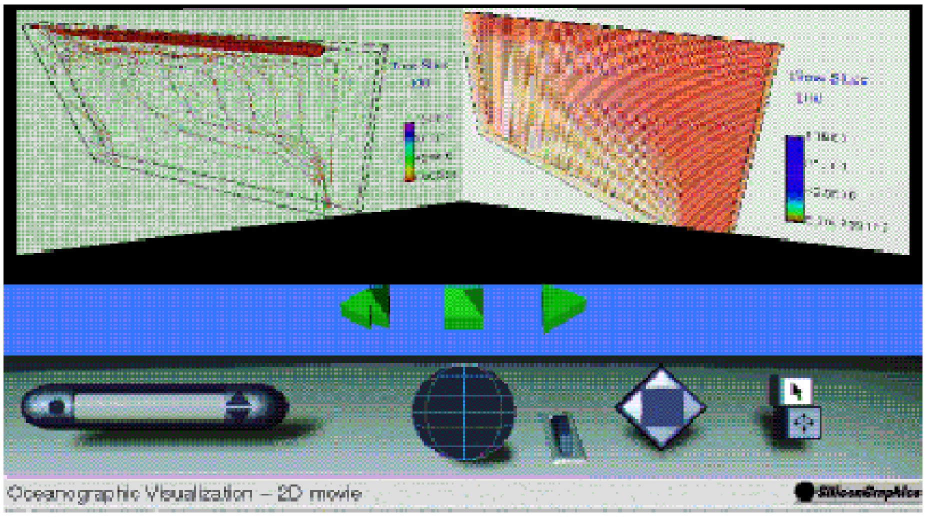

Figure 1c *Two 2D movies within a VRML world.*

4. 3D spatial

The time dependent part may be mapped into the spatial domain. For example, a stack of three dimensional models may be generated, such as created by the visualization server of Wood et al [1] in their environmental visualization example.

4.1 Implementation

This method maps the variable ‘time’ into some representation other than time! For example, time may be represented by the graphics attributes of position, colour or size, or realized perhaps by sound. One of the best mappings of the temporal domain (other than using animation techniques) is to map the temporal domain into a position: an offset along a time-line. Thus, the time slices could be displayed as an adjacent stack of three dimensional images.

IRIS Explorer is used to generate the time-stack. The three dimensional models are individually generated and then collected together to form one world.

4.2 Advantages

By displaying the temporal variable as positional information the whole dataset may be viewed and compared in one visualization. This allows the user to gain a better understanding of how the information varies over time, by comparing the models generated at different time frames. The individual models are easily generated by many visualization packages and simply collated to generate the animation. Indeed, this visualization process may be automated and placed on the web as a ‘visualization service’ [1].

4.3 Disadvantages

The information in one model may obscure some information shown in a neighbouring frame. One solution is to generate disparate models for each frame, thus presenting each snapshot in time in a separate view. However, it is still possible for the user to misinterpret the information due to an unfamiliar representation: with the variable ‘time’ being represented by (say) position. This may be resolved by displaying the temporal data in different visualization forms: representing the information in ‘multiple views’ [11].

4.4 Examples

Figure 2 depicts two individual visualizations, both representing time as positional changes. As the user touches the cursor over a model that time frame is highlighted and its frame number is displayed. Additionally, the separated models (on the right part of the figure) can be individually or jointly rotated.

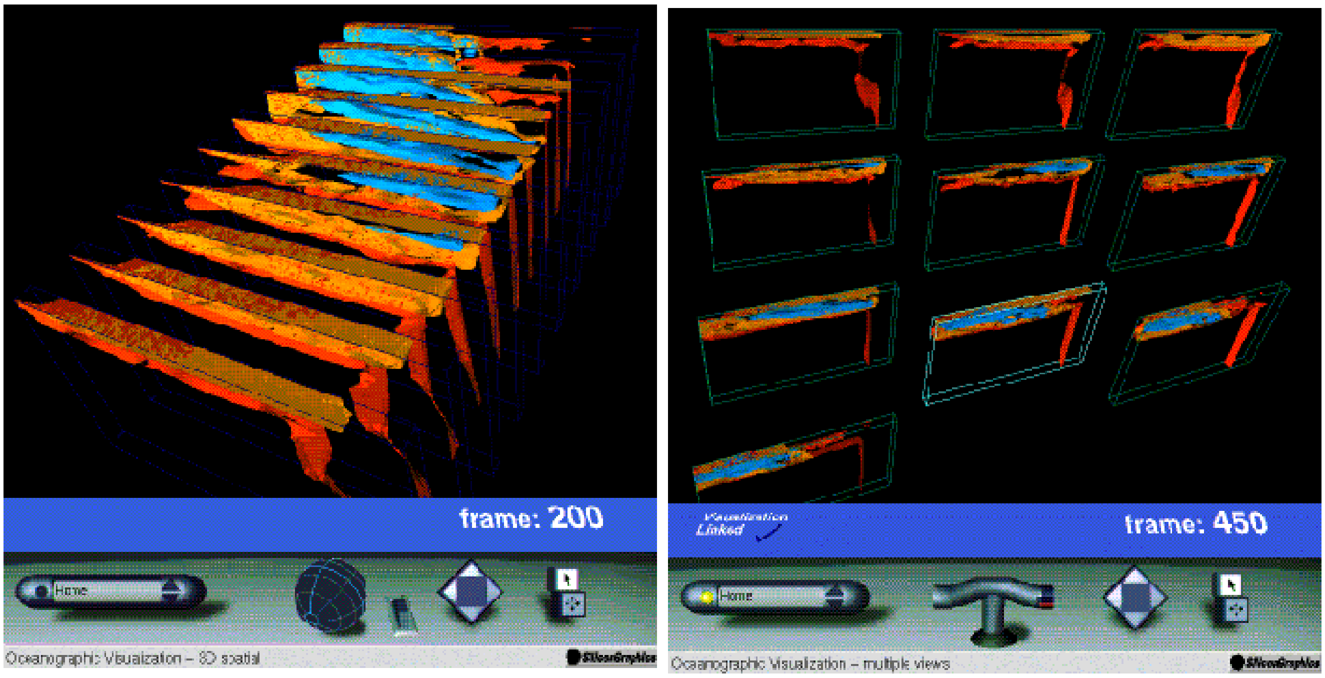

Figure 2 *The ‘time’ variable is visualized using a positional change. The individual models on the right can be individually or jointly rotated.*

5. 3D flipbook

Each frame may be represented by a three dimensional model that is sequentially rendered on playback, to generate the appearance of motion.

The temporal information may be represented by *switching* between different rendered frames, as of the 2D movie. The rendering is achieved as each frame is played, thus, allowing the viewport to be dynamically altered as the model is being played.

5.1 Implementation

Visualization systems such as IRIS Explorer can be ‘programmed’ to generate the individual frames and to directly output the visualization as VRML files. The flipbook animation method may be generated by grouping individual time frames of the visualization together and loading the frames as children of a Switch node. Then, the behaviour of the animation is described by ‘routing’ a timer to sequentially switch between the different scene options. Incidentally, the browser may cache the models, so a smooth flipbook animation is possible.

It is best to add a user interface to control the *start* and *stop* time of the Timer. These controls could be included in a model-based Head Up Display, enabling the user interface to remain in constant *view* while the user navigates through the world.

5.2 Advantages

Three dimensional animations, such as these, are useful for oceanography as the investigator can *fly through* a model that is moving. The individual models are easily generated by many visualization packages and simply collated to generate the animation.

The individual models, possibly containing many complex parts, are accurately displayed. There is no loss of visualized information and no mathematical assumptions are made (such as, when using linear interpolation) when the model is re-played.

It is easy to re-publish and share such three dimensional models. This allows the oceanographers to discuss different visualized aspects of a particular frame and share ideas with colleagues. Indeed, it is possible to *stream* individual models to the viewer; thus, saving on download time and the size of the stored model. For example, Moezzi et al [12] present a system that generates individual three dimensional dynamic models from surface representations of people from video sequences within VRML.

5.3 Disadvantages

This animation technique relies on a fast playback speed, it is easy to generate a *jerky* animation if the model is large and the rendering is slow: no quality of service is guaranteed.

The usual implementation is to completely replace the current scene with the new time frame. But often, parts of the scene remain unchanged between frames. It is possible to update only the parts that change. This ‘temporal coherence’ may save storage size and rendering calculations. One implementation method is to split each time frame into two, with the static model being first displayed, and the dynamic parts being added and removed from the scene.

5.4 Example

Figure 3 depicts three frames of the animation. The animation may be played, stopped, rewound, and the viewpoint altered as the animation is executing.

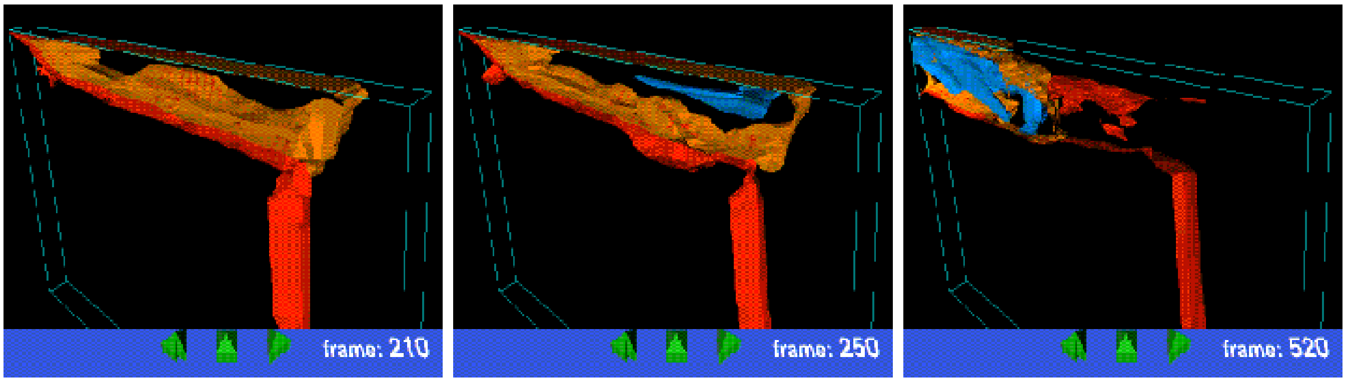

Figure 3 *Three frames from a flipbook animation, depicting three isosurfaces round a plume of algae.*

6. Key frame deformation

The visualization may be described as a deformation between key-frames, where the objects are morphed into the next frame.

One key frame method is achieved by moving the vertices of a polygonal model or adapting the control points on a parametric one [13]. Thus, the realization contains both representations of the objects and methods of how to move and deform the model to generate the motion.

6.1 Implementation

The problem here becomes one of ‘how to describe the visualization animation in terms of object deformations’. In essence, this deformation calculation may happen before or after the creation of the geometry; before being mapped to geometry the objects themselves may be tracked, see Silver [14], whereas, after the geometry has been generated issues such as splitting and merging vertices need to be addressed.

It is not easy to generate these deformation models. The objects are usually complex and many problems exist to describe efficient models [15]. This is because the geometric model, from the visualization, contains multiple vertices, the number of vertices probably change between *time frames* and often objects appear and disappear between frames. Further, it is possible to reduce the vertices using decimation techniques [16], so, it should be possible to ‘fix’ the number of vertices between a subset of adjacent frames to generate a behaviour for this reduced set of frames.

Moreover, the deformation can be used to animate specific objects within a scene. For example, this technique could be used to visualize particle trajectories through a vector field, with the objects being animated along the direction of the fluid flow.

6.2 Advantages

This generates a smoother 3D animation than the flipbook version. It is easier to incorporate ‘static parts’ within this method, because, the animation behaviour is directed to specific parts of the model.

Simplified or abstract views are more readily described by this style of animation. They generate a smaller model than one encoded in the 3D flipbook method, as this behaviour description would be smaller than the size of equivalent replicated geometry.

6.3 Disadvantages

It is difficult to describe an animation using a deformation technique that maintains the accuracy of the visualization. In essence the model has been 'averaged' over a set of frames: with vertices moved, geometry reduced and errors may accumulate through each morphed frame. Thus, the visualization may be 'less exact' than some of the other methods, as some parts of the model may have been augmented to describe an 'efficient behaviour'. Moreover, the behaviour description itself may generate a larger file compared with the flipbook method.

6.4 Example

We use the VRML interpolators to generate a linear interpolated animation. Figure 4 depicts an abstract visualization of different concentrations of algae, a 'bounding area' surrounds different algae concentrations and the boxes are morphed using *CoordinateInterpolator* nodes.

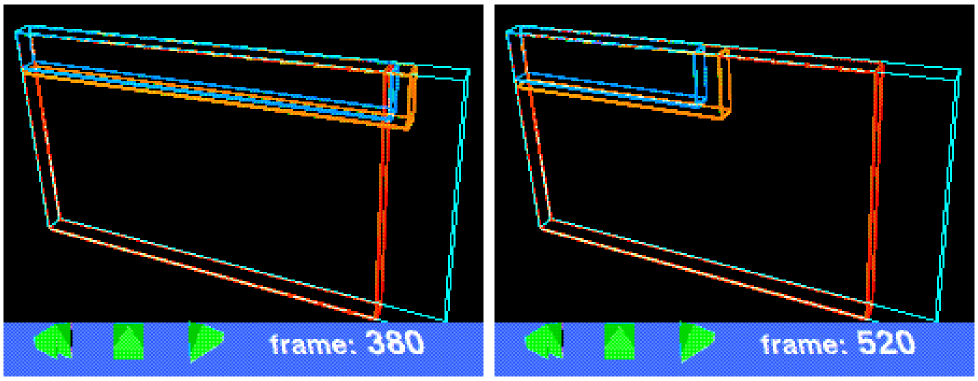

Figure 4 *The extents of three algae concentrations are depicted by bounding boxes; interpolator nodes are used to control their positions.*

7. Visualization program

This publication strategy includes some simulation data with the publication. Thus, specific visualization algorithms and parts of the simulation (or stored) data are bundled with the publication.

The data is used to dynamically generate the geometry of a specific visualization. For example, Java, through the External Authoring Interface, may be used to generate the geometry, animate and modify the realized VRML world. The difference between this 'visualization program' and other web visualization tools, such as VizWiz [17], is that this category implies that the effort of 'visualization' has been achieved by the publisher. However, the next obvious scenario is indeed a visualization system!

7.1 Implementation

This method may be implemented using a model replacement technique; where, the visualization program, operating on the client side, updates the current model with the geometry of the next frame. Moreover, only some parts of the model, those that have altered, need to be updated. Thus, to publish such a 'visualization program' the important data is also published, along with a 'behaviour description' of how to generate the geometry for a specific realization.

7.2 Advantages

Many different realizations could be incorporated in the same publication, and the same *data* used for a number of published scenarios. It is possible to download the data once and import new visualization methods to display alternative representations, thus saving some download time.

The realization can be displayed in different ways, depending on certain user criteria, such as 'their location' or 'who they are'. Moreover, it is possible to program the publication so that the user is able to change some visualization (or animation) parameters to generate an alternative representation of the same information [18] or be able to generate a more detailed version of the visualization that focuses on a particular aspect of the realization.

7.3 Disadvantages

The contained data is often very large and will take sometime to download. Additionally, the publisher often does not wish to release the data to the client, however encoded or cryptic the data is stored.

The whole fact of publishing something that may generate a slightly different realization (and perhaps a new understanding) of the information itself is a disadvantage, because, the tutor and user may be looking at different realizations of the same information.

8. Conclusion

Data, generated from oceanographic simulations, is often complex, with multiple variables, is described on non-regular grids, displayed with uneven (ocean-floor) boundaries and is inherently time dependent. Thus, it is prudent to visualize and publish this data with the temporal variable.

VRML is an ideal 3D publication medium, indeed, VRML97 is designed to support 'moving worlds' [3]. Thus, VRML may be used to publish and realize 3D temporal visualizations of oceanographic data.

There are many ways to publish the realizations within VRML, we have categorized the methods into five groups, discussed their relative merits and provided some example implementations. No one method is better: they each have their advantages and disadvantages. However, within the context of oceanography, we believe that the methods, such as the '3D flipbook' that display the realization as an animation and without a fixed viewport are very important. Often, the oceanographic simulations take many hours to execute, even on todays fast architectures, so these visualization methods with alterable views allow the oceanographer to better interrogate and understand the temporal aspects of the simulation.

Acknowledgements:

I thank Jarle Berntsen (University of Bergen, Norway) for the oceanographic simulation data, Tim Hopkins (University of Kent at Canterbury) for his help and Duncan Langford (University of Kent at Canterbury) for proof reading the paper.

References:

[1] "Visualization over the World Wide Web and its Application to Environmental Data" J. Wood and K. Brodlie and H. Wright, Proceedings IEEE Visualization '96, pages 81-86. 1996.

[2] "Interactive Visualization of Ocean Circulation Models", Scott Nations et al, Proceedings IEEE Visualization '96, pages 429-432, October 1996, ISBN 0-89791-864-9.

[3] VRML 97, ISO/IEC 14772-I:1997, Standards and Specifications, < http://www.vrml.org/Specifications/>

[4] "Rock'n' Roll: Using VRML 2.0 for Visualisation", Jeremy Walton and David Knight, NAG Ltd, Oxford OX2 8DR, UK. <http://www.nag.co.uk/doc/TechRep/HTML/tr2_97.html>

[5] "Publishing in a 3D World: Data Sharing with VRML", Jeremy Walton, BCS International Conference on 3D and Multimedia on Internet, WWW & Networks, 16-18 April 1996, Bradford, UK. (NAG technical report: TR7/96 NP3067) < http://www.iec.co.uk/doc/TechRep/NP1513.html>

[6] "Fresh Water driven primary Production in a Fjord", Jarle Berntsen and Dag L. Aksnes and Arne Foldvi, Department of Applied Mathematics, University of Bergen, Norway, March 1997, No. 107.

[7] "A Virtual Environment for Multimedia Intelligence Data Analysis", John S. Risch, Richard A. May, Scott T. Dowson and James J. Thomas, IEEE Computer Graphics and Applications, vol 16(6), pages 33-41, November 1996.

[8] "Waltz - An Exploratory Visualization tool for Volume data, using Multiform Abstract Displays", Jonathan C. Roberts. In Robert F. Erbacher and Alex Pang, editors, Visual Data Exploration and Analysis V, Proceedings of SPIE, volume 3298, pages 112-122. (IS&T and SPIE), January 1998.

[9] "Interactive video on WWW: Beyond VCR-like interfaces", A. Katkere, J. Schlenzig, A. Gupta and R. Jain, Fifth International World Wide Web Conference, Elsevier Science Publishers B.V.. Computer Networks and ISDN Systems. Vol 28(7-11), pages 1559-1572. May 1996. < http://vision.ucsd.edu/~deborah/infra/jain3/mpiv/mpiv.html>

[10] "View Interpolation for Image Synthesis", Shenchang Eric Chen and Lance Williams, Computer Graphics (SIGGRAPH '93 Proceedings), pages 279-288, vol 27, August 1993

[11] "On Encouraging Multiple Views for Visualization", Jonathan C. Roberts, Proceedings of the International Conference on Information Visualization IV'98, IEEE Computer Society, London, pages 8-14, 29-31 July 1998.

[12] "Reality Modeling and Visualization from Multiple Video Sequences", Saied Moezzi, Arun Katkere, Don Y. Kuramura and Ramesh Jain, IEEE Computer Graphics and Applications, vol 16(6), pages 58-63. November 1996.

[13] "Advanced Animation and Rendering Techniques - Theory and Practice", A. Watt and M. Watt, Addison-Wesley, 1992. ISBN 0-201-54412-1.

[14] "Object-Oriented Visualization", Deborah Silver, IEEE Computer Graphics and Applications, 15(3) May 1995, pages 54-62.

[15] "Interactive Computer Animation", Edited by Nadia Magnenat Thalmann and Daniel Thalmann, Prentice Hall, 1996. ISBN 0-13-518309-X.

[16] "A Topology Modifying Progressive Decimation Algorithm", IEEE William J. Schroeder. In Roni Yagel and Hans Hagen, editors, Visualization ’97, IEEE, pages 205-212. November 1997.

[17] "VizWiz: A Java Applet for Interactive 3D Scientific Visualization on the Web", Cherilyn K. Michaels and Michael J. Bailey. In Roni Yagel and Hans Hagen editors, Proceedings of the 8th Annual IEEE Conference on Visualization, Visualization ’97. pages 261-268. October 1997.

[18] "Alternative Archaeological Representations within Virtual Worlds", Jonathan C. Roberts and Nick Ryan. The Fourth UK VR-SIG Conference, Brunel University, pages 179-188, November 1997.
< http://www.cs.ukc.ac.uk/people/staff/jcr/archaeology/vrsig.html>