4D Data Visualization

Incorporating Time into 3D Animations

By

Jonathan Boright

A research paper submitted to

**The Department of Geosciences,
Oregon State University**

In partial fulfillment of the
requirements for the
degree of

**Masters of Science
Geography Program**

June, 2003

Directed by
A. Jon Kimmerling

Acknowledgements

There are many people who have contributed in countless ways to my experience here at Oregon State University. To attempt to list them is certainly to do injustice to the many who will remain unlisted. That being said, there are several individuals without whose support my experience here would not have been possible, or at least would have been far less enjoyable.

I would especially like to thank my committee members, Dr. Wright, Dr. Rosenfeld, and Dr. Kimerling for their support and patience during the course of my studies at OSU, and Dr. Matzke for having faith in me in the first place. My gratitude also lies with Greg Gaston for being a soundboard for my many previous ideas for research projects, and with Michelle Frank for her undying encouragement and editing.

Finally, thanks also to my parents Gerlinde and John Boright who have always given me the support and inspiration needed to accomplish this goal as well as many others in my life.

Table of Contents

4D Data Visualization
Incorporating Time into 3D Animations

Abstract

ArcView 3D Analyst is a powerful geospatial tool, and has proven very useful in 3D visualization and analysis. In most cases, 3D data models are temporally static, representing only one instance in time. For many applications, the 'missing' time dimension is of great significance, and there is a need to incorporate this fourth dimension into data visualization techniques.

This paper discusses a method, designed by the author, of realizing four-dimensional or time-sensitive visualization of data. The method utilizes ESRI's ArcView 3.2 ® software, and integrates the surface generating ability of Spatial Analyst ®, the 3D visualization power of 3D Analyst ®, and the data compression and high quality graphics of Adobe's LiveMotion ® to generate high quality animation files accurately depicting changes in 3D space over time.

This paper will examine the development of this method and will showcase the manner in which it was applied to two different data sets: ground-water modeling and aerial dispersion modeling. Finally, the paper will discuss the challenges presented by this method, the limitations a user encounters, and possible directions for future work using this method of 4D data visualization.

Introduction

The use of Geographic Information Systems in academia, government and business is becoming increasingly widespread as scientists, urban planners, and corporate decision makers become aware of both the spatial analysis capabilities and the cartographic display abilities that GIS have to offer. While the ability to perform both analysis and display is one of the great and unique strengths of GIS, it is often overlooked. Frequently, GIS are used *either* for analysis (by calculating distances, areas, volumes, interpolating surfaces etc.) *or* display (by creating maps and graphics which are easily displayed, manipulated, and updated). One of GIS most overlooked abilities is the capacity to do analysis *through* display. This means combining the analysis and display into a dynamic and interactive process that can bring to light previously unexplored perspectives and suggest new paths for analysis.

Traditional forms of data display include charts, graphs, and 2-dimentional maps. More recent forms of data visualization, created in a large part by the emergence of GIS, are surface trend interpolation, 3D perspective maps, and 3D animated models. ESRI has encouraged these latest trends in data visualization through the introduction of its Spatial Analyst ® and 3D Analyst ® extensions. Together with ArcView ® software, these extensions create a powerful set of geospatial tools. In spite of ESRI's dominance in the GIS marketplace (The ArcView/ArcInfo suite of products is the most widespread GIS software package, and has become in many respects the de facto industry standard), this software is most often used to create temporally static graphics, representing only one instance in time. For many applications there is great utility in incorporating the fourth dimension, time, into data visualization techniques. Other software packages exist which are specifically designed to facilitate environmental modeling and visualization (such as RockWare, EVS, and Vertical Mapper) but these packages are costly both in initial price of the software, and in the training time required to use them successfully.

This paper looks at one method of realizing four-dimensional or time-sensitive data visualization using ESRI's ArcView 3.2a ® software and two of its extensions, Spatial Analyst ® and 3D Analyst ®. While this software is also expensive, it is already common among GIS users, and therefore does not demand an additional investment in purchase and training time. In this method ESRI's software is integrated with Adobe's LiveMotion ® animation builder, a relatively simple and inexpensive animation building software package, to generate animation files that accurately depict changes in 3D space over time.

This paper looks at the development of this method by showcasing its application to two different data sets: one set resulting from ground-water modeling and the other from aerial dispersion modeling. A compact disk containing these two animations is attached at the end of the paper. Finally, the paper discusses the challenges faced, limitations encountered, and possible directions for future work using this method of 4D data visualization.

Background (description of animations)

The method described in this paper has produced many successful animations. Two of them will be used as examples to illustrate points in this paper. The first animation takes place on an island in an estuarine environment, and illustrates the influence of tidal fluctuations on the groundwater surface. Hereafter it will be referred to as "the groundwater animation". The second animation graphically displays the results of an aerial dispersion model that uses meteorological data to predict the deposition pattern of emissions from a smelter plant. This animation will be referred to as "the aerial dispersion animation".

Groundwater animation:

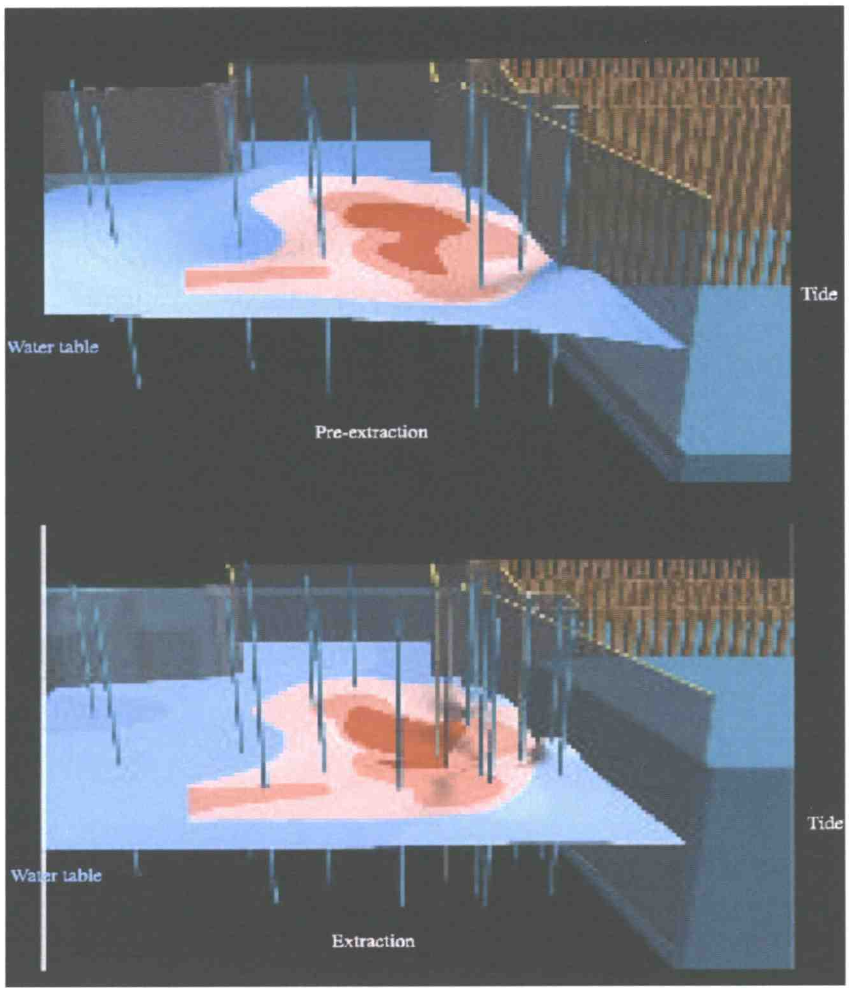

Figure 1.

The goal of the groundwater animation was to visually demonstrate tidal influence on the water table in this estuarine environment. This is of interest because there is a contaminant plume that rests on top of the water table, and to fully understand the fate of this contaminate plume one must first understand how it is affected by fluctuations in the water table. In an effort to remove the contaminant, the property owner has been pumping a mixture of contaminant and water from four "extraction" wells for several years. The question for the environmental scientist is "What effect is this extraction process having, if any, on the shape of the water table and the fate of the floating contaminant?" To take a closer look at these issues, two animations were created; one portraying the pre-pumping scenario, and one showing the situation during pumping.

The total modeled time span for these animations (pre and during extraction phases) is one tidal cycle, which is expressed in an eight second animation loop for each of the data sets. The two scenarios are shown simultaneously in the animation, one on top of the other. The animation will pause if the curser is placed on top of the image.

Aerial dispersion animation:

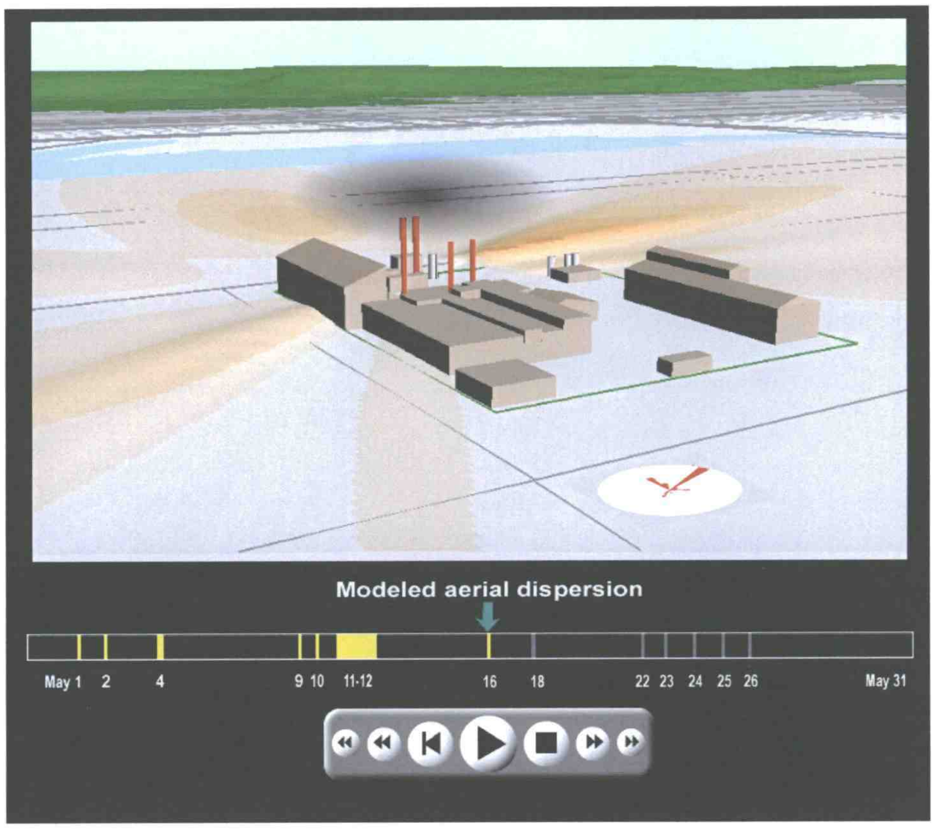

Figure 2. Aerial dispersion animation screen shot.

The goal of the aerial dispersion animation is to graphically demonstrate the deposition pattern of particulate produced from accidental emissions at a smelter plant. The aerial dispersion and deposition in the surrounding area was modeled by a statistical modeling package called ISCST3. The accident events occurred sporadically over a twelve-year time frame (1970 - 1982).

ISCST3 uses a variety of data inputs. These include accident type (fire or other failure), time and duration of accident, smoke stack height and location, and meteorological factors such as wind speed and direction. Once all of these factors have been determined and entered into the modeling program, the model is run and the deposition concentrations are predicted at points on a predetermined grid. This grid of concentration values becomes the input for the GIS animation building process.

The large span of the time portrayed in the aerial dispersion model, as well as the intermittent nature of the accident events, present a real challenge to the animation producer. On one hand, for the sake of accuracy, it is important to show the entire time interval at a constant speed, and still restrict the length of the animation to a viewable length (it is not realistic or desirable to have the viewer sit through an hour-long animation comprised mostly of inactivity). On the other hand, the actual events of interest shown at a time frame reasonable for the entire time interval would appear as almost instantaneous blips in the animation. The solution in this case was to produce multiple animations; one showing the entire time interval (all 12 years) at a relatively fast pace, another zeroing in on a shorter period of five years at a slower pace, and yet another zeroing in on an even shorter period of one month at a yet slower pace. In this manner, the scope of the overall time interval is retained, and the detail of the individual events are still observable.

The one-month animation will be used as an example in this paper.

Methods

The procedure for animation creation can be divided into three phases; data preparation, GIS use, and animation production.

Phase 1. Data Preparation

During the data preparation phase, the foundations of the animation are laid. The data-producing model is run, and data are generated and exported into a format readable in ArcView 3.2a. Important decisions are made in this step that will have a large effect on the quality and accuracy of the final product, as well as an impact on the level of effort involved. Careful consideration in making these decisions saves time and money later in the animation process.

The two most critical decisions in the data preparation phase are choosing the appropriate temporal and spatial resolution for the data set. Temporal resolution refers to the time interval between data points. Spatial resolution refers to the spacing between data points.

After these choices have been made and the data generated, the next step is to convert the data into an appropriate format. While this may sound trivial, the unusually vast size of many GIS data sets can make this step a significant challenge. Careful consideration of the existing computer hardware and software capabilities is warranted before taking this step.

Finally, an opportunity exists in the data preparation phase to gather "secondary data". Secondary data refers to data which are not critical to the creation of surfaces, but which are nevertheless useful or pertinent to the goal of the animation. An example of this kind of data is the total emission mass from the aerial dispersion. While this number might not relate to the animation itself, the number can be easily calculated in this phase and will be useful in the project in general.

Temporal resolution

Like many choices in computer technology, the choice of temporal resolution can be seen as a tradeoff between size and accuracy. Higher temporal resolution results in a smoother and more accurate animation. However, higher temporal resolution also means a larger data set. A larger data set, in turn, requires a computer system with larger storage capacity, faster processing speed, a more robust graphics accelerator, and longer data processing time. Processor speed and storage capacity can be purchased up to the limits of the current technology.

Another factor to consider when deciding on the size of a data set is the ultimate platform on which the animation will be run. An animation that runs well on a high speed GIS workstation will not necessarily fare well on a five year old laptop that cannot handle the graphics processing requirements.

To make decisions on temporal resolution, one must first consider the time-span to be simulated, the target length of the animation, and the degree of detail required from the final animation.

Example: In the groundwater model the total time being modeled is one tidal cycle, or approximately 24 hours. Well log data captured intermittently over a period of one full tidal cycle were used. From these data, water table levels were predicted for each well. The result was a data set with a temporal resolution of one hour. When the resulting animation was run, it was evident that the temporal resolution was insufficient. The animation, while small in size, appeared choppy and hard to comprehend.

The solution to this problem was to go back and resample the data at a higher temporal resolution. Specifically the temporal resolution was increased from one hour to one half hour. The resulting animation is smooth and effective and loops through the entire tidal cycle (48 time slices) in 8 seconds (six data frames per second).

It is important to understand the difference between the animation frame rate – the number of frames per second produced by the animation software – and the data frame rate – the rate of frames containing unique data. In this case the data frame rate is 6 frames per second, while the animation frame rate is twice as fast at 12 frames per second. There is no point in having a data frame rate faster than the animation frame rate because the data will only be displayed at the animation frame rate. On the other hand a high animation frame rate will be useless if the data it represents change at a much slower rate. The prime reason to have an animation frame rate faster than the data frame rate is to support secondary moving graphics in the final animation.

The temporal resolution selected should also be an easily understood increment of time. This is especially true if the final animation will have the option of "stepping" through the data frame by frame. Stepping can be accomplished by using an on-screen control button (as is the case in the aerial dispersion model) and is much more easily understood if the time step is a common unit, say an hour or half an hour, versus a non-standard unit (like 23/37ths of a day).

Spatial Resolution.

The second important decision in the data preparation phase is the choice of spatial resolution for the initial data set. In many cases this choice is limited by the availability of raw data. For example, in the groundwater animation the data set is limited to the number and location of the existing monitoring wells (barring the drilling of new wells). In other cases, such as the aerial dispersion model, the spatial resolution is not limited by data availability and must be specified. In this section we are speaking of the spatial resolution of the initial data points, and not the spatial resolution of the resulting surfaces or animation images. These latter two issues will be discussed in the phase 2 (GIS use) and phase 3 (animation production) sections, respectively.

Similar to the choice of temporal resolution, the choice of spatial resolution can be seen as a trade-off between accuracy and computational intensity. The smaller the distance between data points (higher spatial resolution), the more accurately the resulting surface will represent the model. On the other hand, the closer the data points are to each other, the more points will be required to cover the same area and the more computationally intense the process becomes. A good question to ask when deciding on the spatial resolution is, "What is the smallest feature that I wish this model to represent?" The data points should be spaced close enough to reflect the smallest features, but no closer. In some cases the solution might be to have multiple spatial resolutions in the same model.

Example: Aerial dispersion model

The aerial dispersion model discussed earlier represents a situation in which the modeler has the ability to choose the spatial resolution of the data. The nature of the model is such that a higher variability of concentration values is predicted near the source of emissions (the stacks). Consequently the modeler chose to space the data points closer together near the stacks (10 meter spacing), and farther apart (20 meter spacing) further away. This allowed for a more accurate surface representation near the center of the model, and a less computationally intensive process towards the edges where the data variability is significantly diminished.

Output format

Once the choices of spatial and temporal resolution have been made, the raw data are generated either by running the model (the aerial dispersion case) or collecting the field data (the groundwater case). What remains is to convert the data into a format that is understandable to the ArcView as well as the Avenue scripts. ArcView 3.2 can import data tables in *.dbf, *.csv, or *.txt formats. In both of the example animations presented here, the *.dbf format was chosen, primarily because it is easy to open and manipulate in both Microsoft Excel and Microsoft Access database packages, making the data easy to work with.

Aside from being readable to ArcView, the data table must be compatible with the "grid maker" script used later in the process. This script, discussed further in the next section, uses the data tables to interpolate a surface for each time increment. The format of the data tables needs to contain both location and data values. The location data can be in the form of coordinate pairs, or simply a name identifier that can be linked to locations later using the GIS. An example of the format used for both the groundwater and the aerial dispersion animations can be seen below. The first two columns in this format contain location data (x and y coordinates), and each remaining column contains data values for a single time slice. Because the data will be brought into ArcView without header information, it is important that the column names are understandable as time slices. In other words, the column names must read as "time1, time2…" or some other such distinguishable name.

X - coord	**Y – coord**	**Time1**	**Time2**	**Time3**
Value (meters)	Value (meters)	Value (data units)	Value (data units)	Value (data units)
Value (meters)	Value (meters)	Value (data units)	Value (data units)	Value (data units)
Value (meters)	Value (meters)	Value (data units)	Value (data units)	Value (data units)

Figure 3. data table format template

Secondary data

The final consideration in the data preparation phase is the collection of what I have termed "secondary data". These are data which are not used for calculating surfaces directly, but which may be useful for animation production or as corollary information. Examples of secondary data include the total deposition in the aerial dispersion model, and the tide height in the water table model.

Phase 2. GIS Use

Phase 2 can be broken down into three steps.

A. Bringing the data into ArcView.

B. Using the customized "Grid maker" script (and AV Spatial Analyst extension) to interpolate points, creating surfaces (grids).

C. Using the customized "Jpeg maker" script (and AV 3D Analyst extension) to produce and export a series of images (jpegs).

Bringing the data into ArcView

The data are brought into ArcView by importing the final table created in phase 1. The points are then plotted by either using the first two columns (if they are x,y data) or linking a station identifier to an already existing point theme.

"Grid maker" script

The "Grid maker" script is a customized Avenue script that streamlines the surface generating process. Use of this script requires the Spatial Analyst extension, also made by ESRI but sold separately from ArcView. A button is added to the ArcView "View" button bar that activates the "grid maker" script (The actual script is attached as appendix A). Upon activating the script, the user is led through a series of choices. The first is a choice of interpolation methods; inverse distance weighted (IDW) or spline. At the time the script was written, these were the only two options available in ArcView. With the emergence of ArcView 8.x and the Geostatistical Analyst extension, new possibilities exist, which will be discussed further in the "Discussion" section. After the interpolation method is chosen, the user is asked to define the extent and grid cell size of the new surface grids. After these parameters are selected, the user is asked to select, from a dropdown menu, time slices (column headers) for the surfaces that are to be interpolated. Finally, the user is asked to select a "clip" theme from the themes in the view. This clip theme will act as a boundary theme, outside of which the surfaces will be removed.

Once the clip theme is selected, the script will run and the surfaces will be created. It is important to note that the grids will be created in the project work directory, and will overwrite pre-existing grids of the same name in that directory. For this reason, it is very important to set the project work directory **before** creating the grids.

An important aspect of this phase is that the user has the opportunity to alter the interpolation methods should the surfaces turn out to be unsatisfactory. If the scripts are run again without changing the work directory, the older surfaces will be overwritten with the new ones of the same name. In practice, this has proven to be a very useful feature, as it is rare to select the optimal interpolation parameters on the first try.

If, like in the groundwater example, the user miscalculated the useful temporal resolution, it is not always necessary to go back to the data preparation phase. The option exists of using the software to interpolate between the surfaces already calculated. This should only be attempted if the nature and trend of the surfaces being modeled is known, and the shapes of the surfaces are mathematically predictable.

"Jpeg-maker" scripts

Once a suitable set of surfaces (one for each time slice) has been created, the project is ready to move into the jpeg creation phase. For this purpose, another Avenue script has been created (attached as Appendix B) which utilizes the 3D visualization features of ArcView's 3D Analyst extension. This script is run from a button added by the user to the 3D Scene button bar. As with the "grid maker" script, the "jpeg maker" script requires user input to set up the starting parameters of the sequence as described below.

To start with, the 3D scene is set up manually with the correct view angle, zoom factor, vertical exaggeration, sun angle and direction. Also, at least one surface must be present to use as a "legend key" for the time slice surfaces to be added. When developing a suitable legend key, it can be helpful to add to the 3D scene the surfaces with both the highest and lowest vertical relief values. This ensures that both extremes are covered by the legend. As a general rule, the legend should have a small enough initial increment to reflect changes at the smallest scale, but enough range to reflect the entirety of the surface with the largest vertical relief. An appropriate opacity should be chosen for the legend as well.

Once all the elements of the scene have been set, the scripts are activated by clicking a button on the button bar. In fact, there are two different scripts, one for adding one surface at a time and one for adding two surfaces at a time. For simplicity sake, the script for adding one surface at a time will be described here.

When the button activating the "jpeg maker" script is clicked, the user is first asked to select the set of surfaces to add to the scene from a dropdown menu, and then is asked to choose the legend key theme. Next, the user is asked to select the horizontal size (in pixels) of the images to be made, as well as the folder in which they will be stored. Once these choices have been made, the script will cycle through the following sequence for each time slice (column field) selected:

JPEG maker script sequence:

1. Bring in the first surface
2. Apply appropriate legend
3. Create a snapshot image in jpeg format
4. Name the image and save it to the location specified in the set up process
5. Remove the last surface
6. Update and redraw the scene and start from the top again.

When the script has finished this process for each time slice it will end. A series of consecutively numbered images will have been created in the specified folder, ready for import into the animation software.

Phase 3. *Animation Production Phase*

In this final phase of the animation process a separate animation software package is used to bring in the images produced in the GIS phase, add special effects, and create the final animation. Adobe's LiveMotion ® was chosen because it creates compressed animated files in the Shockwave format. This format is compatible with HTML, enabling the animations to be viewed on most web browsers.

Setting the global parameters in the animation software.

The first step in the animation production phase is to set the global animation parameters; namely the size (width and height in pixels) of the animated space, and the rate of the animation (frames per second). Both of these parameters correlate directly with the file size of the final animation; a relatively small animated area shown at relatively few frames per second will result in a significantly smaller animation file size than a large animated area with a high frame rate.

Importing the time series images (jpegs) and distributing them temporally

Once the global parameters of the animation have been set as described above, the series of images are brought into the animation-building software. This is accomplished by using the Live Motion's built-in import utility and is a relatively simple matter.

Distributing the images over the time scale of the animation can also be a simple matter, or it can be very time consuming and problematic. In some cases the subject to be animated is characterized by smooth and regularly moving surfaces like waves or steady groundwater flow. In such situations the interval between time slices is most likely a constant, which means that the resulting images should be distributed evenly throughout the time scale of the animation. In such cases simply defining the start and ending time of the series can create the correct temporal placement of the images. Live motion will distribute the images evenly between these two defined time points. The groundwater animation is an example of such a case.

In other cases the subject to be animated occurs at irregular time intervals. One example of this type is the aerial dispersion animation, which depicts sudden and accidental emissions from a factory. In such cases, the time interval between data frames is not regular and therefore should not be distributed evenly across the animation. Such cases require the user to place each event individually along the timeline within the animation. While this is not necessarily a problem for animations that have a small number of events (data frames), the effort required to place the events along the time line grows proportionally to the number of events. The fact that this process cannot be automated in LiveMotion is a major limitation of the software when used for this type of animation.

Adding secondary data and special animation effects

When the images have all been temporally placed in LiveMotion, other effects can be added to the animation. This is where LiveMotion really demonstrates its utility. A wide variety of effects and controls can be added to increase the communicative power of the animation. Examples include controls such as stopping, fast forwarding, or jumping back in time (as seen in the aerial dispersion model); visual effects like a smoky plume leaving a smokestack, a clock counting down (or up) time, or a scale bar growing in height; or sound effects such as the gurgle of water, howl of the wind, or boom of explosions.

Exporting the product into the final animation format.

The final step in the animation building procedure is to export the animation from the LiveMotion software into a Shockwave format (*.swf). The *.swf format is a compressed media format. When an animation is exported from LiveMotion, an HTML file is produced at the same time as the *.swf file. Opening the *.html file brings up a window in the default web browser, references and then runs the *.swf file which runs the animation on the users screen. It is also possible to run the *.swf file in several other multi-media players such as Apple's Quicktime or the Windows Media Player. It should be noted, however, that when running in formats other than *.swf (the Shockwave media format) performance may be sub-optimal. For example, when played on a Quicktime player, the *.swf files will not be able to play audio files associated with the animation.

Results

The two animations described in this paper have many similarities, but also differ in important ways. One of the major differences between the two models is that they occur over widely differing time scales. The groundwater animation loops through a simulated 24 hour tidal cycle, while the aerial dispersion animation takes place over one month. Another critical difference is that the time increments in the groundwater model are evenly spaced (one data frame every half an hour) while the data frames in the aerial dispersion model occur at random intervals. These differences have important implications for the animation building process.

What follows is a more detailed description of each animation.

Groundwater animation:

The groundwater animation is comprised of two separate animations running concurrently; one portraying the pre-pumping scenario, and one showing the situation during pumping. The goal was to visually demonstrate tidal influence on the water table in this estuarine environment in each scenario.

To create the data sets, groundwater elevation measurements were taken at each of 21 wells over a full tidal period in April of 1999 before pumping began in the extraction wells. Measurements were taken again in October of 2000, during the extraction phase. One initial problem for this animation was that not all data were taken simultaneously. Only in four wells was the data collected on a continuous basis. The remaining 16 wells were sampled periodically through the tidal cycle with an average of 10 samples per well location. From the behavior of the wells for which a full tidal cycle of data existed, predictions were made for the data values from the remaining wells. The results are two sets of complete cycle data for all 21 wells. This large set of data was sampled at half hour increments and written to the final two data sets.

For this model the “Spline” interpolation method was chosen. This method is true to each data point, and creates a smooth surface, similar to what we expect a water table surface to look like.

The total modeled time span for these animations (pre- and post- extraction phases) is one tidal cycle (approximately 24 hours). The data frame temporal resolution is one half hour, which results in 48 data frames per tidal cycle. A data frame rate of 6 frames per second over this time period creates an 8 second animation loop for each of the data sets.

In order for the movement of the water table to be easily discernable in the animation, the vertical exaggeration of the 3D scene was set to 10 times. The exaggeration of the tides (outside the bulkheads) was set at a lower number (5 times) in order to fit the entire tidal range in the animation area.

Care was taken in choosing the color scheme for this animation. The color of the ground water and the tides were set to green and blue so that they would be easily distinguishable from one another, yet both clearly represent water features. Also, the bulkhead is shaded gray and is partially transparent, allowing the viewer to see through to the tide behind. Other environmental objects such as pilings are added to give the viewer context that creates a sense of realism.

The result is a groundwater model that effectively demonstrates tidal influence on the water table. The animation also pointed to a breach in a bulkhead that was not previously suspected.

Aerial dispersion model:

As with the groundwater animation, the "Spline" interpolation method was chosen for the aerial dispersion model over an inverse distance weighted method to calculate the depositional surfaces. This, again, was due to the generally smooth nature of the "splined" surfaces. The spline interpolation technique has its drawbacks, however. The largest drawback is that splined surfaces tend to exaggerate peaks and troughs where surfaces reach maximums and minimums, especially where data points are widely dispersed.

The vertical exaggeration for the aerial dispersion model was then set to an appropriate level. In this case, the modeled data units are grams per meter squared (g/m2), which is a measure of concentration, not depth. As such, any three dimensional (vertical) appearance given to the surface in the animation is done purely for display purposes. Given this, the vertical scale is set so that the shading will appear at the earliest event in the animation, but will not cover the buildings or otherwise appear that there are drifts of product blanketing the site.

For the animation covering the 12-year period the animation rate is set at two weeks per second, resulting in a 5-minute animation covering the whole 12 years. A frame rate of 14 frames per second was set, resulting in a one frame per day temporal resolution. For the five year period the animation rate is set at 10 days per second, and the data frame rate is set at 10 frames per second (the temporal resolution is one day). This results in a 3 minute animation covering the five years from 1970 to 1974. For the one month animation, an animation rate of one day per second was chosen, resulting in a 31 second animation (the month of May 1972 was picked). For this animation, a data frame rate of 24 frames per second resulted in a one frame per hour temporal resolution.

It should be noted that because the data in the aerial dispersion model are not continuous (i.e. there are long periods of time where no events are occurring, thereby producing no data), it is only necessary to produce unique data frames for times when events are occurring. When events are not occurring, the surfaces do not change and the resulting images stay the same. Because of this fact, a high data temporal resolution can be achieved while keeping the final file size relatively small. While this temporal non-uniformity is handy in terms of file-size, the placement of data frames within the time line can be very time intensive. This subject will be discussed further in the discussion section.

The resulting aerial dispersion animation conveys to the viewer a sense of the frequency and relative magnitude of accident events as well as the spatial distribution of the resulting emissions that would have been difficult to portray by using static images and charts.

Discussion

The discussion will focus on the overall effectiveness of the animations, general limitations of the method, potential changes to the methodology, ideas for the future, and other software.

Overall effectiveness

This paper has described an effective process for creating animations for visualizing data in four dimensions using ESRI software in combination with Adobe's LiveMotion software. These animations enable the user to visualize and analyze data from a perspective not previously available. In the case of the groundwater model, the animation pointed to a breach in a bulkhead that was previously not suspected. In the air dispersion model, the animation was able to convey to the viewer a sense of the frequency and relative magnitude of accident events as well as the spatial distribution of the resulting emissions, which would have been difficult to portray by using static images and charts.

Limitations

The animation process described in this paper is not without limitations. Most significantly, the process of creating the animation can be very time intensive. Both the GIS and the animation software packages (ArcView 3.2a and LiveMotion) require large time commitments in setup, data preparation and data entry. LiveMotion is not 'scriptable' and therefore requires manual placement of individual events along the time-line (if the actual events are not regularly spaced in time). Also, the surface interpolation algorithms available in ArcView 3.2 are limited to inverse distance weighted (IDW) and spline. Consequently they are not powerful enough to interpolate accurately if surfaces are complex or data points are dispersed.

Another major limitation to this animation technique is that the current version of 3D Analyst is not able to model surfaces that double back on themselves in the vertical direction (Z axis). In other words, a surface my not have more than one z coordinate for any one x,y pair. The only method of displaying a solid object of limited horizontal extent is to “extrude” a surface vertically, which simply duplicates the surface at a different z value and connects the edges. In other words, ArcView can model a cylinder but not a sphere. This limits the users ability to model certain types of environmental phenomenon such as contaminate plumes consisting of relatively thick features of limited lateral extent, and fence diagrams.

Direction for future work.

There are several potential improvements in the animation procedure that should be mentioned here. Since the writing of the animation scripts, ESRI has put out a new version of ArcView (ArcView 8.x, part of the new ArcGIS system) with extensive modifications from the earlier versions. One of the most important modifications is the switch from Avenue scripting language to Visual Basic. This change could potentially facilitate communication between ArcView and other database programs such as Microsoft Access. Better communication between ArcView and standard databases could streamline the production process significantly, in effect joining Phases 1 and 2, as described in the methodology section. Another potential advantage of the new ArcView

8.x is the addition of new interpolation methods including “kriging”. This expanded functionality could help greatly in the interpolation of difficult data sets. Additionally, kriging is the interpolation method preferred by the EPA when dealing with chemistry data (Copsey 2002).

One of the best ways to further streamline the animation production phase (Phase 3) is be to use a scriptable animation software package. As of the time of this writing, I am not aware of such a package. A scriptable animation software package could have the ability to place events along the animation time line, perhaps by using a lookup table and assigning data frames to the animation according to a key field. The current manual placement of data frames along the timeline is the most time intensive aspect of phase 3. Some software packages that may have scripting ability are Adobe Premier and Macromedia Director Studio, but again, as of this writing I have not had the opportunity to investigate these programs.

Another potential improvement in this method would be to add the ability to change the viewer’s perspective at the same time that one changes the time slice. This would give the animation the effect of moving through space as well as time. ArcView 8.x allows the user to specify the viewer location using 3D analyst, overcoming a key obstacle in achieving this additional capability.

Other software.

Other software does exist for the production of 3D animations. Most of these packages, however, have only the ability to simulate the movement of the observer through space (so-called “fly-by” animations). Fly-by animations do not generally have the capability of moving the observer through time. The Baird software company has created such a product called Map Animator. ESRI has also added a fly-by animation builder to its latest 3D analyst functions in ArcView 8.2.

Other software packages that can create 3D visualizations are: EVS (Environmental Visualization System), Vertical Mapper, and RockWare OpenGL 3Dviewer. Of these, EVS come closest to creating the type of 4D animations produced with the method described in this paper. The base price of EVS-pro is close to $10,000. Thus, the method described herein remains one of the most cost-effective for creating 4D animations.

Summary

Geographical Information Systems (GIS) are in widespread use today in academia, government and business environments. While many GIS were originally set up to perform traditional cartographic display and analysis functions, GIS users have found new and innovative ways to use their systems. Alternative techniques for data visualization, such as 3D modeling, have become popular. The logical next step is to create 3D models that change over time, in effect making them 4D models.

This paper has examined one method of realizing four-dimensional or time-sensitive data visualization. We have seen how ESRI's ArcView 3.2a ® software and two of its extensions, Spatial Analyst ® and 3D Analyst ®, can be used in conjunction with Adobe's LiveMotion ® animation builder to generate animation files that accurately depict changes in 3D space over time. Two examples of this method have been shown: one animation showing groundwater modeling and the other depicting air dispersion modeling. Finally, the paper has discussed the challenges, limitations, and possible directions for future work using this method of 4D data visualization, along with other software packages that have similar functionality.

With a reasonable expenditure of time and energy, an intuitive visualization can be produced by the method described here, enabling people of different technical backgrounds to quickly understand a complicated set of environmental data.

References:

Berry, J.K., 1995. Spatial Reasoning for effective GIS, Fort Collins, CO. GIS World Books.

Fisher P., and D. Unwin, eds., 2002. *Virtual Reality in Geography*, New York, NY. Taylor and Francis

Hazen C., 1996. *The mapping of Eelgrass in Willapa Bay, WA, and an evaluation of C-CAP Mapping Guidelines*, Corvallis, OR, Oregon State University

McGuire, D. J., M. F. Goodchild, and D. W. Rhind. 1991. Geographic information Systems: Principals and applications, Essex, England, Longman Scientific and Technical

McCoy, J. and K. Johnston. 2001. Using ArcGIS Spatial Analyst. Redlands, CA, ESRI

Sun B., W. Shields, G. Brugger, and M. Bryant. 2002. *Forensic Significance of Accidental Releases from a Lead Smelter: Release Reconstruction, Dispersion and Deposition Simulation, and Impact on Soil Concentration.*

Copsey, R. 2002 *EVS release notes: Indicator kriging.* C Tech Development Corporation, Huntington Beach, CA

Appendix A "Grid Maker" Script

```
' Spatial.Grid_series
' created 4/10/01 by Jon Boright

'  This script allows the user to choose multiple fields within a point
'  theme, and creates multiple grid surfaces based on the chosen fields.



theView = av.GetActiveDoc

' SELECT Shapefile for analysis (first GTheme, TTheme, or point FTab found)
theTheme = theView.GetActiveThemes.Get(0)
t = theTheme
needDelete = FALSE

' obtain projection
aPrj = theView.GetProjection

theRlist = { }

' interpolate GRID if point FTab
if (t.GetClass.GetClassName = "FTheme") then

  ' obtain extent and cell size if not set
  ae = theView.GetExtension(AnalysisEnvironment)
  box = Rect.Make(0@0,1@1)
  cellSize = 1
  if ((ae.GetExtent(box) <> #ANALYSISENV_VALUE) or (ae.GetCellSize(cellSize) <> #ANALYSISENV_VALUE)) then
    ce = AnalysisPropertiesDialog.Show(theView,TRUE,"Surface Grid Specification")
    if (ce = NIL) then
      return NIL
    end
    ce.GetCellSize(cellSize)
    ce.GetExtent(box)
  end

  ' convert z or m shapes to non-z or non-m shapes,
  ' just create the empty shapefile with all fields
  ' here to populate interp dialog
  theFTab = t.GetFTab
  theClassName = theFTab.GetShapeClass.GetClassName
  if ((theClassName = "PointZ") or
     (theClassName = "MultiPointZ") or
     (theClassName = "PointM") or
     (theClassName = "MultiPointM")) then
    needDelete = TRUE
    theFieldList = theFTab.GetFields.Clone
    theShapeField = theFTab.FindField("Shape")
    theNewShapefile = av.GetProject.GetWorkDir.MakeTmp("cntmp","shp")
    theNewFTab = FTab.MakeNew(theNewShapefile,Point)
    theFieldList.RemoveObj(theShapeField) 'ignore the shape field
    theFieldCount = theFieldList.Count - 1
    theNewShapeField = theNewFTab.FindField("Shape")
    aShape = theFTab.ReturnValue(theShapeField,0)
    hasZcoord = aShape.HasZ
    hasMcoord = aShape.IsMeasured
    if (hasZcoord) then
      zCoordField = Field.Make("ShapeZ",#FIELD_DOUBLE,12,3)
      theNewFTab.AddFields({zCoordField})
    end
    if (hasMcoord) then
      mCoordField = Field.Make("ShapeM",#FIELD_DOUBLE,12,3)
      theNewFTab.AddFields({mCoordField})
    end
  theNewFieldList = theFieldList.DeepClone
  theNewFTab.AddFields(theNewFieldList)
  t = FTheme.Make(theNewFTab)
end

' set parameters for interpolation method
interpList = InterpolationDialog.Show(t, cellSize)
if (interpList.Count < 2) then
  if (needDelete) then
    theNewFTab.SetEditable(FALSE)
    t = NIL
    theFTab.DeActivate
    theFTab = NIL
    theNewFTab.DeActivate
    theNewFTab = NIL
    av.PurgeObjects
    File.Delete(theNewShapefile)
    theNewShapefile.SetExtension("shx")
    File.Delete(theNewShapefile)
    theNewShapefile.SetExtension("dbf")
    File.Delete(theNewShapefile)
  end
  return NIL
end


'Choose Fields to grid
FieldList = t.getFtab.getFields
zFieldList = msgBox.MultiListAsString(FieldList, "Choose fields for Grid calculation", "Field List")
if (zFieldList = NIL) then
  msgBox.Info("No fields selected...bailing.", "exiting")
  exit
end
'msgBox.listasString(zFieldList, "zFieldList", "")

'Choose Clip Cover
clip = msgBox.YesNoCancel("do you want to clip the grids?", "Clip?",TRUE)
if (clip = true) then
  tlist = List.Make
  pthms = av.Run("aanView.Sub-Setup"," ").Get(1)
  clipThm = msgBox.choice(pthms, "Which polygon theme would you like to use as the clip coverage?", "Polygon themes")
  elseif (clip = NIL) then
  exit
else
end


'calculate Grids
for each zField in zFieldList
 'zField = interpList.Get(0)
 anInterp = interpList.Get(1)

 ' export z and m shapefile to non-z or non-m shapefile
 ' export only the selected set
 if ((theClassName = "PointZ") or
    (theClassName = "MultiPointZ") or
    (theClassName = "PointM") or
    (theClassName = "MultiPointM")) then
   av.ClearMsg
   av.ClearStatus
   av.ShowStopButton
   av.ShowMsg("Exporting Shapes...")
   theIndex = 0
   theBitMap = theFTab.GetSelection
   if (theBitMap.Count = 0) then
     numFeatures = theFTab.GetNumRecords
     theBitMap.SetAll
     unsetBitmap = TRUE 'reset flag for end of loop
   else
     numFeatures = theFTab.GetNumSelRecords
 unsetBitmap = FALSE
end
done = FALSE
offset = -1
while (not done)
 rec = theBitmap.GetNextSet(offset)
 offset = rec
 if (rec <> -1) then
  theShape = theFTab.ReturnValue(theShapeField,rec)
  if ((theClassName = "PointZ") or
     (theClassName = "PointM")) then
   if (zField.GetName = "ShapeZ") then
    theZ = theShape.GetZ
    if (theZ.IsNull.Not) then
     theNewRecnum = theNewFTab.AddRecord
     theNewFTab.SetValue(zCoordField,theNewRecnum,theZ)
     theShape = theShape.AsPoint
     theNewFTab.SetValue(theNewShapeField,theNewRecnum,theShape)
    end
   elseif (zField.GetName = "ShapeM") then
    theM = theShape.GetM
    if (theM.IsNull.Not) then
     theNewRecnum = theNewFTab.AddRecord
     theNewFTab.SetValue(mCoordField,theNewRecnum,theM)
     theShape = theShape.AsPoint
     theNewFTab.SetValue(theNewShapeField,theNewRecnum,theShape)
    end
   else
    theValue = theFTab.ReturnValue(theFTab.FindField(zField.GetName),rec)
    if (theValue.IsNull.Not) then
     theNewRecnum = theNewFTab.AddRecord
     theNewFTab.SetValue(zField,theNewRecnum,theValue)
     theShape = theShape.AsPoint
     theNewFTab.SetValue(theNewShapeField,theNewRecnum,theShape)
    end
   end
  elseif ((theClassName = "MultiPointZ") or
       (theClassName = "MultiPointM")) then
   for each p in theShape.AsList
    if (zField.GetName = "ShapeZ") then
     theZ = p.GetZ
     if (theZ.IsNull.Not) then
      theNewRecnum = theNewFTab.AddRecord
      theNewFTab.SetValue(zCoordField,theNewRecnum,theZ)
      p = p.AsPoint
      theNewFTab.SetValue(theNewShapeField,theNewRecnum,p)
     end
    elseif (zField.GetName = "ShapeM") then
     theM = p.GetM
     if (theM.IsNull.Not) then
      theNewRecnum = theNewFTab.AddRecord
      theNewFTab.SetValue(mCoordField,theNewRecnum,theM)
      p = p.AsPoint
      theNewFTab.SetValue(theNewShapeField,theNewRecnum,p)
     end
    else
     theValue = theFTab.ReturnValue(theFTab.FindField(zField.GetName),rec)
     if (theValue.IsNull.Not) then
      theNewRecnum = theNewFTab.AddRecord
      theNewFTab.SetValue(zField,theNewRecnum,theValue)
      p = p.AsPoint
      theNewFTab.SetValue(theNewShapeField,theNewRecnum,p)
     end
    end
   end
  end
  theIndex = theIndex + 1
  progress = (theIndex/numFeatures) * 100
  doMore = av.SetStatus(progress)
  if (not doMore) then
```

```
        theNewFTab.SetEditable(FALSE)
        if (unsetBitmap) then
          theBitmap.ClearAll
        end
        t = NIL
        theFTab.DeActivate
        theFTab = NIL
        theNewFTab.DeActivate
        theNewFTab = NIL
        av.PurgeObjects
        File.Delete(theNewShapefile)
        theNewShapefile.SetExtension("dbf")
        File.Delete(theNewShapefile)
        theNewShapefile.SetExtension("shx")
        File.Delete(theNewShapefile)
        return NIL
      end
    else
      done = TRUE
    end
  end
  theNewFTab.Flush
  if (unsetBitmap) then
    theBitmap.ClearAll
  end
  if (theNewFTab.GetNumRecords = 0) then
    MsgBox.Error(zField.GetName++" is null for all features","Create Contours")
    theNewFTab.SetEditable(FALSE)
    t = NIL
    theFTab.DeActivate
    theFTab = NIL
    theNewFTab.DeActivate
    theNewFTab = NIL
    av.PurgeObjects
    File.Delete(theNewShapefile)
    theNewShapefile.SetExtension("dbf")
    File.Delete(theNewShapefile)
    theNewShapefile.SetExtension("shx")
    File.Delete(theNewShapefile)
    return NIL
  end
  theFTab = theNewFTab
end
' perform interpolation
r = Grid.MakeByInterpolation(theFTab,aPrj,zField,anInterp,{cellSize, box})

' clean up temp shapefile if exists
if (needDelete) then
  theNewFTab.SetEditable(FALSE)
  t = NIL
  theFTab.DeActivate
  theFTab = NIL
  theNewFTab.DeActivate
  theNewFTab = NIL
  av.PurgeObjects
  File.Delete(theNewShapefile)
  theNewShapefile.SetExtension("shx")
  File.Delete(theNewShapefile)
  theNewShapefile.SetExtension("dbf")
  File.Delete(theNewShapefile)
end

if (r.HasError) then
  return NIL
end

'add the grid to the list of grids
theRlist.add(r)

end
```

```
end



fThemeList = { }

if (clip = true) then
  av.run("Spatial.ExtractByGraphic_alt", {theRlist,clipThm,zFieldlist})
else
  fThemeList = theRlist
end

for each thm in fThemelist
  theView.AddTheme(thm)
end
```

Appendix B "Jpeg Maker" Script

```
' jpeg maker_series
' created 4/10/01 by Jon Boright

'  This script allows the user to create jpegs using 3D analyst and an existing series of surface grids.


'Select themes to add to JPEG series
srcnames = SourceDialog.Show("Themes to add for JPEG creation")
'msgBox.list(srcnames, "themes to be added", "themes")


'Select the Key Theme and get the Key Legend

theScene = av.GetActiveDoc
theThemeList = theScene.getThemes
theKeyTheme = msgBox.list(theThemeList, "Key Theme", "Choose a Key Theme")
theKeyLegend = theKeyTheme.getLegend

'select visible themes
visThemeList = { }
for each t in theThemeList
  if (t.isVisible) then
    visThemelist.add(t)
  end
end

'msgBox.list( visThemeList , "The Visible Themes" , "Vis Themes")

' set image resolution and other image variables

imageType = #TYPE_JPG
fileExt = "jpg"

theViewer = theScene.GetDisplay.GetViewers.Get(0)
widthDefault = theViewer.GetPosition.GetWidth.AsString
status = TRUE
while (status)
  resWidth = MsgBox.Input("Input the desired width for the output image (height will be scaled relative to the width):","Snapshot
Viewer",widthDefault)
  if (resWidth = NIL) then
    return NIL
  end
  if (resWidth.IsNumber and (resWidth.AsNumber > 0)) then
    status = FALSE
  else
    status = TRUE
    MsgBox.Error("The width must be a number greater than 0","Snapshot Viewer")
  end
end


   aFN = av.GetProject.GetWorkDir.MakeTmp("JPEG",fileExt)
   aFN = FileDialog.Put(aFN,"*."+fileExt,"Your JPEG folder")
   FNameDir = aFN.returnDir.AsString+"\"

   msgBox.info(FNameDIR, "FNameDir")

   'if (aFN = NIL) then
   '  return NIL
   'end

''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

'Start the loop
```

```
themeCount = srcNames.count
'msgbox.info("there are "++themeCount.asString++" themes to be added", "theme count")

for each i in 0..(themeCount-1)
   'check that the names are really there...
   'msgBox.info(srcNames.get(i).asString, "theme name")
 aTheme = NIL

''' Delete 3rd theme back...

 if (i > 1) then
   oldTheme2 = theScene.FindTheme(srcNames.get(i-2).asString)
   theScene.DeleteTheme(oldTheme2)
 end

 'make the new theme
 aTheme = Theme.Make(srcNames.get(i))
 if (i>0) then
   oldtheme = theScene.getThemes.get(0)
   oldTheme.setVisible(false)
 end


 ''''''''''''''''''''''
 'for each vt in visThemelist
  'vt.setVisible(false)
 'end

 oldCount = theScene.GetThemes.Count

 theScene.AddTheme(aTheme)

 'av.run("JPG.Scene.add", aTheme)
 ' Scene.Add



 ''''''''''''''''''''''


 'set the legend properties
 aTheme.setLegend(theKeyLegend.clone)
 theLegend = aTheme.getLegend
 theLegExt = DDDLegendExtension.Make
 theLegExt = theLegExt.setBaseSurface(aTheme)
 theLegExt.setOffset("4.1")
 theLegExt.setZfactor(0.15)
 'theLegExt = theLegExt.setBase("3")
 theLegend.SetExtension(theLegExt)


 'theLegExt.Illuminate(true)
 aTheme.setVisible(true)
 atheme.SetLegendVisible(false)

 theScene.getDisplay.Flush

 aTheme.InvalidateLegend
 theScene.InvalidateTOC(NIL)
 theScene.getDisplay.Invalidate(true)
 aTheme.updateLegend

''''''''''''''''''''''''''
'ADD THEME checks
if (theScene.GetThemes.Count > oldCount) then
 if (theScene.GetActiveThemes.Count = 0) then
   theScene.GetThemes.Get(0).SetActive(TRUE)
 end
 if (oldCount = 0) then
   theThemes = theScene.GetThemes
```

```
    r = RectZ.MakeEmpty
    for each t in theThemes
      if (t.Is3D) then
        r = r.UnionWith(t.ReturnMBB)
      else
        aRect = aTheme.ReturnExtent
        anOrigin = aRect.ReturnOrigin
        aSize = aRect.ReturnSize
        if (aSize.GetX < aSize.GetY) then
          zra = aSize.GetX
        else
          zra = aSize.GetY
        end
        zra = zra / theScene.GetDisplay.GetExaggeration
        aRectZ = RectZ.Make(anOrigin@(-zra*0.5), aSize@(zra))
        r = r.UnionWith(aRectZ)
      end
    end
    if (r.IsEmpty) then
      return NIL
    elseif ((r.ReturnSize) = (0@0)) then
      theScene.GetDisplay.PanTo(r.ReturnOrigin)
    else
      theScene.GetDisplay.SetMBB(r)
    end
  end

  av.GetProject.SetModified( TRUE )
end

'make the original Themes Visible again
  for each vt in visThemelist
    vt.setVisible(true)
    vt.invalidateLegend

    theScene.InvalidateTOC(NIL)
    theScene.getDisplay.Invalidate(true)
    vt.updateLegend

  end
  theScene.getDisplay.Flush


  ''''''''''''''''''''


  'make the snapshot

    FName = ((FNameDir+srcNames.get(i).getFileName.getBaseName)+".jpg").asFilename

    'imageDir = srcNames.get(i).getFileName.returnDir
    'imageName = srcNames.get(i).getFileName.getBaseName
    'msgBox.info(FName.asString, "FName")

   'end
    ok = theViewer.Snapshot(FName, imageType, resWidth.asNumber)
    if (not ok) then
      MsgBox.Error("Error writing image","Snapshot Viewer")
    end

    if (i > 0) then

    end

end 'end of loop
''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''
```