THE TIMEMAP PROJECT: INTERACTIVE MAPS OF TIME-DEPENDENT CULTURAL FEATURES USING MULTIPLE INTERNET DATA SOURCES

Ian Johnson
johnson@acl.archaeology.usyd.edu.au
Archaeological Computing Laboratory
University of Sydney

MAPS, DESKTOP MAPPING & CULTURAL ATLASES

Humans are three-dimensional beings with a well-developed grasp of space. We use pictures to navigate our way around modern GUI interfaces, and we are comfortable with information presented in atlases, newspapers and TV programs in the form of **maps**. Maps are an extremely powerful way of presenting data in an easily grasped form.

There is a figure floating around which says that over 80% of commercial databases contain some sort of spatial data, which is generally unexploited by traditional textual database applications or spreadsheets. The potential to derive new kinds of information from this data, and to display it in a more effective and engaging manner, has lead to the development of Desktop Mapping Systems (consumer-oriented GIS), as another tool in the 'office' computing environment. We now see mapping capabilities built into Microsoft Excel and other spreadsheet and database applications, and there are a number of reasonably priced and relatively user-friendly desktop mapping applications available on the broad consumer market which tap into existing databases such as Access, .dbf files or Excel spreadsheets.

The use of maps is well-established as a means of presenting cultural information. There are several printed atlases showing the extent of regional cultures in antiquity. What is interesting about the printed atlases is the fact that progress through the book is not progress through space, as in most traditional atlases, but progress through time.

In the same way, desktop mapping can be used to present a series of maps through time, perhaps as overlaid layers which can be switched on and off. But there is far greater potential here than a mere electronic simulacrum of a printed atlas. Not only can one zoom in and out seamlessly and switch layers on and off or change their symbology, but one can potentially build a database structure behind the mapping front end which allows one to store, present and eventually analyse the full richness of the historic record. This opens the way for us to interpolate maps for gaps in the historic record and eventually to build animated maps of cultural extents, invasions, communication networks and the like.

Temporal GIS

Temporal GIS (TGIS) are Geographic Information Systems (GIS) which explicitly handle temporal information. TGIS is a hot research area, one which only dates back a few years because the technical problems of standard 2D GIS had to be solved adequately to support the more complex structures required by a three or four dimensional GIS.[1]

[1] It is worth noting here that conventional GIS are really 2D - even though they might have the ability to show contours or to give you three-dimensional views, they are looking at a **surface,** not a volume. If you try to do something like record an archaeological site with excavated layers, you can't do it with a conventional GIS and

Although a TGIS system should explicitly handle time (t), which might be regarded as the fourth dimension, it does not necessarily have to handle the third dimension, elevation (z). The TimeMap project, discussed in this paper, aims to develop a 2D GIS with explicit support for the time dimension. The TimeMap GIS approach is a simple TGIS, more of a temporal Desktop Mapping System, and the two fundamental requirements are that we manage searchable spatial data and create a time-enabled mapping front-end. There are of course several subsidiary requirements which we will discuss later in this paper.

THE TIMEMAP PROJECT

The TimeMap project[2] (http://www.TimeMap.net) is a project to develop a methodology for recording and mapping cultural features[3] which can be attributed some form of spatial location and have a temporal range and/or variability. By cultural features we mean:

1. Data/fact-based entitities such as sites, events, monuments, collected objects, historical documents, images and so on (Sites and Monuments Registers, Museum catalogues, Archives and Inventories);
2. Cultural areas, boundaries and networks – synthetic entities which are the result of people analysing the data and deciding where those boundaries or features fall. Databases of this type are currently uncommon.
3.

In a nutshell, the aim of the TimeMap project is to develop "A software system, backed by an explicit spatio-temporal data recording methodology incorporating spatio-temporal fuzziness; allowing the superimposition of data layers, including base maps and satellite images, derived from different remote datasets across the Internet; querying and display of data layers with explicit support for temporal information & spatio-temporal fuzziness, display using a time-enabled map interface, temporal interpolation and in-line animation". The meaning of some of these terms will become clearer as we discuss them.

Cultural Feature databases

Let us look briefly, and simplistically, at the history of cultural databases. They started out life in the 1970s as big SPSS or SAS files. There was much talk of "data banks" which would solve all our problems. In the 1980s; they moved to relational databases, and in the 1990s they have moved further into Desktop Mapping and GIS. For the end of the 1990s and the start of the 21st century, I think the areas of growth will be **temporal**, **multimedia** and **Internet-distributed** GIS. One cannot therefore look at temporal data handling in isolation – any forward-looking application must address issues of multimedia data storage and presentation, and must be capable of operating in a distributed environment across the Internet. All of these developments result in the storage of a greater variety and volume of

---

they don't handle time other than as a series of map layers for different periods, or simply by attaching attribute values, e.g. period, to geographic objects.

[2] TimeMap is a project of the Archaeological Computing Laboratory at the University of Sydney. The author is director of the methodology component of the project.

[3] Perhaps also natural features because, as archaeologists, we are interested in phenomena such as changing coastlines and land area with siltation or postglacial sea-level rise.

information, increasing complexity, and an increase in the capabilities of what one can present and the ways one can look at it and, of course, much greater data structuring, management and application complexity.

This does not, however, mean that the front-end seen by users need be complex or difficult to use. On the contrary, with advances in technology, map-based presentation of data, particularly when combined with touch screens, can potentially provide intuitive and engaging user interfaces for public access and educational applications.

Application areas and audiences

Table I lists some examples of application areas to which TimeMap might be relevant. These can be at all sorts of scales, from long-term and continent-wide phenomena (such as climate or landscape change), down to temporally and spatially localised events (such as the progress of campaigns or specific battles in World War II).

Table I: TimeMap potential application areas

Cataloguing & retrieval • Storage & retrieval of historical documents (including archaeological or historical sites, museum collections) in a spatio-temporal context
Analysis & interpretive applications • Settlement patterning on landscape • Expansion & decline of empires • Migration & colonisation • Trade routes & exchange • Settlement layout/urban growth • Battles & campaigns • Landscape evolution e.g. coastlines, hydrology • Climate change

At the landscape scale, TimeMap will probably function mostly as a cataloguing and retrieval system, allowing the display of historical and archaeological documents of all types within a map- and time-based interface (Fig. 1). This has the potential to greatly enhance access to database information, and to present the results of searches in a much more engaging format. At the other end of the spectrum, it is particularly hard to show the processes of ebb and flow of military campaigns, and the synchronicity or otherwise of events, using traditional maps with arrows, symbols and textual information. A time-enabled GIS, such as the TimeMap front-end, will allow the user to investigate the changing configuration through time and create animations to better understand the events.

Table II: Application audiences

Domain	**Applications**
Public interest Museums Education Research	Information booths Video animations Database with map display Spatio-temporal retrieval of historical documents & specimens Analysis of change & relationship

	between GIS layers through time Predictive modelling & simulation

The sort of audiences or applications to which TimeMap may be relevant fall into two lists which go hand-in-hand, but not necessarily in a one-to-one relationship (Table II). Public interest applications will typically be oriented towards information booths (which we will soon see on every street corner, at least if you happen to be hosting an Olympic Games), video animations which you might see at the entrance of a museum, and access to databases with map display (such as collections databases accessible in museums through public terminals – allowing one to see where objects came from, where other similar objects came from, what cultures existed at the time and so on). At the other end of the list, research applications will tend to be more analytical, including measurement of change and perhaps even prediction of future conditions.

Pilot projects

We are currently working on two pilot projects:

1. The Virtual Historic Sydney project, directed by Andrew Wilson, is looking at the growth of Sydney in the 18th and 19th centuries, and assembling a wide variety of map, textual and visual historic resources for Sydney;
2. The AsiaMap project, directed by Roland Fletcher, is examining the spread and collapse of Asian empires, with emphasis so far on developing animations of culture areas rather than a substantial database of background information.

TIMEMAP METHODOLOGY

A model for time-varying features

We record time-varying features using a **snapshot/transition model** as illustrated in Fig. 2, which shows an area feature n a space-time cube (x and y with t vertically). We know the extent of the area at, in this case, four points in time (left-hand illustration). By interpolating between these we can guess the extent of the cultural area at intervening points in time (right-hand illustration).

The advantage of this model is that it mirrors the way in which we know about the past. We typically have information at particular points in time – the position of an archaeological site of known date; the date when a particular map was drawn, or when someone recorded the position of an army; the cultural affiliation of people occupying a location; or the edge of an empire. Our knowledge is therefore based on snapshots through time, but we do not know explicitly what happened in the intervening spaces. However, we often know something about the intervening period - for instance, that it was probably a linear change in between the two enclosing snapshots; that there was a change at increasing or decreasing speed; or that it was probably about the middle of the interval. This information can be used to improve the interpolation of intermediate conditions.

A big advantage of the snapshot/transition model over other spatio-temporal models (see Langren 19??) is that it is fairly easy to implement using widely available GIS tools. One can simply record the snapshots, be they points, lines or areas, by digitising into a conventional GIS or desktop mapping system. One can describe the transitions between them, as attributes

of the digitised objects, through the dates of the enclosing snapshots together with a fuzzy descriptive function.

Data structure

The data structure we use is fairly simple, consisting of two relational tables. The **feature table** describes the features we are recording, for example the Mongol, Tang and Mughal empires, historic neighbourhoods or administrative areas in Sydney or archaeological sites. It records some static description which will typically be little more than a label (e.g. The Mongol Empire) and some text describing what it is, where information about it (electronic or hard-copy) can be located, its starting and ending dates and maximum spatial extent (i.e. its bounding space-time cube). The **incarnation table** describes one or more snapshots for each feature in the feature table, including a graphical (spatial) description of the object, transition information between consecutive snapshots, dating information for the snapshot, attributes which vary through time (such as population) and pointers to other resources, such as WWW pages.

The two tables which we work from need not actually exist in the data being accessed. They can be created as SQL views on an SQL Server database or generated on local data using a "base query", which is defined within the metadata (see later).

Fuzziness

Table III: Types of fuzziness

	Spatial	**Temporal**
Uncertainty	We don' t know the exact location of something	We don' t know exactly when something happened
Diffuseness	The boundaries of an object are not sharply defined	A change or transition occurs gradually

There are four types of fuzziness relevant to modelling cultural features in time and space, as shown in Table III. Fuzziness can be either spatial or temporal, and can take the form of uncertainty or diffuseness. **Uncertainty** means we don' t know quite when a transition occurred or where the boundary of a cultural area lay. **Diffuseness** means the change is gradual or the edge of a cultural area gradually diffuses into another one - you don' t step over an invisible line and land in another culture. I believe we must take fuzziness into account explicitly when recording cultural data, and this is done all too rarely - it is far easier to record neat lines than graded, weighted boundaries, particularly when using a computer. Recording fuzziness and incorporating it into analysis and display is by no means a trivial problem and, although we have put some thought into it, it is not one for which we claim to have a solution. It would be an excellent topic for a PhD thesis.

TMView: a time-enabled map interface

Fig. 3 illustrates our time-enabled map interface, TMView, with historic and modern data layers for central Sydney superimposed. The grey rhombus is a scanned 1807 topographic survey, warped to fit the modern coordinate system. The map window in this interface is driven by a desktop mapping system component. We are currently using MapInfo as an OLE Automation server, but we will be moving to a GIS component such as MapX or MapObjects,

which can be compiled into our software, both to avoid loading unneeded functions and to reduce the cost of distributing the software. It should be possible to use one of the distribution-royalty-free components on the market as we are only asking the map window to do fairly standard desktop mapping functions such as zoom in, zoom out, display layers, switch layers on and off, change symbolism, display attribute data for an object and superimpose datasets in different projections. All the database access, time filtering and interpolation is done behind the scenes in Delphi code (a modern variant of the Pascal programming language).

At the top of the interface are simple functions such as zoom in, zoom out, selection of objects on the map, and opening and closing of datasets.[4] On the left are the map layers (queries) which are displayed in the map window. The **Attach dataset** button simply attaches the dataset to the application - it does not cause anything to be displayed on the map. Attaching a dataset involves navigating to find the dataset, either on a local directory or on an SQL server across the Internet (see next section).

The **Add layer/query to map** button allows one to specify an SQL query against any of the attached datasets (the current version provides a dialogue with pulldown selection of tables, fields, operators and functions, but will eventually be replaced with some sort of ‘ wizard’). The dialogue uses the standard TimeMap names for tables and fields shared by all (or most) TimeMap datasets, such as starting and ending time, latitude and longitude. These standard names are transparently translated into the actual field names used by the individual datasets to create a valid SQL query – the user does not need to know the terminology used by different datasets from which they might draw information.

Additional attribute fields which are not part of the standard TimeMap definition can either be displayed using the actual field names in the target dataset, or the metadata can define more informative extended field names for display. The query dialogue allows the application of selection filters to the dataset and selection of just the fields required, both of which are important for ensuring effective access to large datasets across the Internet (see later). The dialogue also allows one to specify an informative name for the query which is what will appear in the list of layers on the left.

The time slider bar at the bottom of the screen allows the user to set a time span for the map display. Clicking the Redraw button updates all the SQL queries with the new time limits, reruns the queries to recreate the visible layers and redraws the map. Although this may seem somewhat ponderous, all the processing is local and based on reduced datasets when drawn from a remote server, so the performance is quite acceptable and well above anything that can be achieved with map generation and download from a remote server. Once the queries have been processed, pan, zoom and other functions proceed at normal desktop mapping speeds.

---

[4] We use the term **dataset** to refer to a database, either local or remote, which includes a TimeMap metadata file or table (next section).

Metadata structure

The TimeMap front-end software, TMView, will be able[5] to simultaneously access several remote datasets which have different structures, different field names and run on different server platforms, so there is no need for us to impose a fixed structure on TimeMap-accessible databases. The key to this flexibility is the use of **metadata**. Our methodology defines a set of metadata, stored as a single table, which can be added to any server or local database (Fig. 4) to make it TimeMap compatible. The metadata table describes the structure of the database so that our software can interact with the data without the need for standardisation.

When we attach to a TimeMap-enabled dataset we read in the metadata and store it in memory so that we can provide a translation between TimeMap' s concepts and the terms and structures used by the database. For example, if we are interested in the minimum and maximum time field, TMView might call it MinTime and MaxTime, but the database might call it StartTime and EndTime, and these fields might be in two different tables - or the database might have a date classification system which must be translated to a relative scale, or might not have a time component at all and therefore has no equivalent fields. The metadata allows us to define the domain (the space-time limits of the dataset), the names and extended labels for tables and fields, which tables have which fields, what the fields contain or how they should be interpreted, which fields contain URLs for linking to external data, the methods used for storing spatial (GIS) data, and other bits of relevant information. The TimeMap metadata standard is extensible to allow for further conceptual translations which may be needed as the project proceeds.

When opening an existing project, the datasets to be attached and the queries used to display map layers are stored in a **project file (**similar to a MapInfo workspace or ArcView project). The queries read from this file will be updated with any time filtering specified in the project file and used to display an initial map corresponding with the state of the project when it was last saved. Whenever the time limits are changed all the queries are modified and rerun.

Internet mapping and distributed data

I sincerely hope that the kind of GIS mapping we currently see on the WWW will not survive, because it actively discourages the user from interacting with the data. The majority of applications either send the user a predetermined product (e.g. scanned city plans of the area within 1km of an address you have located through the White Pages) or subject them to a series of menu choices, with no visible cue as to what the end product will look like, after which a map is built on a remote server and sent as a screen-resolution bitmap. If the map

[5] Multiple dataset access through metadata has been prototyped locally and is currently being extended to remote datasets. Remote access should have been implemented by time of publication.

isn't exactly what you want – and it never is – then it's back to the server again for a new version.[6] Furthermore, there is no way of integrating your own data into the map produced.

By contrast, on a desktop mapping system with local data, you have almost instant response to changes in layers displayed, layer order, symbolism, labelling, zoom, pan or querying of attribute data, and in most cases it is trivial to add new datasets to the application. That is the way things should be.

To my mind the only viable approach to Internet mapping is to put the mapping application on the client (the user's desktop computer) and access the data alone from the server. An example of this type of approach can be seen in the freely-downloadable ArcExplorer program from ESRI. ArcExplorer can pull map data from ESRI's data server across the Internet, but processes the display of the data locally on the client computer, seamlessly integrated with the user's own datasets stored locally on the hard disk or local network. Desktop mapping and GIS systems are all moving to provide remote access to datasets stored in spatially-enabled SQL servers.
This client-based approach to desktop mapping has important consequences for the way we manage our data. If the Smithsonian Institution or the British Museum wished to allow people to create maps showing the provenience of items in their collections (assuming they have locational information at some level for most of their accessions), it should not be necessary to keep the entire Smithsonian and British Museum catalogues on your desktop computer, gathering dust and becoming out of date. Instead, databases should reside with their creators and maintainers, wherever possible, so that the data accessed is current to the moment. The same thing is happening in business, with corporate datasets being stored on spatially-enabled relational databases which can be accessed remotely by desktop mapping front-ends.

The key to effective access to remote datasets across the Internet is to limit the amount of data which needs to be downloaded to support your application, without downloading so little data that you are continually going back to the server whenever you change anything about your map. In TimeMap this is achieved by filtering the data first by the application of domain limits (x, y, t), then by selecting a subset for display (e.g. when you go to the British Museum catalogue for Scythian tombs, you would only download Scythian tombs, not the entire catalogue) and finally by limiting the selection to only those fields you are interested in (perhaps as little as the X,Y location). By using an SQL query to select the data for display, all of these filters can be applied and processed on the remote server before the resulting dataset is downloaded to the client and cached locally for rapid access and further queries if required.

Only when the local queries go beyond the limits of the cached dataset (e.g. through moving to a larger domain, widening the selection criteria or requiring additional fields) does the

---

[6] Some remote server-side mapping applications are starting to download more intelligence into the map images sent to the user, so that one can do some zooming or querying, for example, on database attributes, without going back to the server for a new map.

application need to go back to the server for more information. The tricky thing is managing the process of updating the locally cached data to reflect changes in the source database or queries which go beyond the limits of the cached data. The key to responsiveness is putting the GIS display and querying of cached datasets on the client, while using the capabilities and horsepower of the server to filter huge source datasets to minimise downloads.

We have also looked at a three-tier approach. This means that an intermediary server gets the filtered data from the source datasets, caches it centrally, manages updates and then provides it to the client application. There are some advantages in terms of database access, cache management and the creation of 'thinner' clients, and disadvantages in terms of requiring maintenance of a server and additional potential failure points, and we have not yet decided if we will go down that route.

The TimeMap metadata catalogue

The Internet is not a very good filing system. If there was a "My Internet" icon on the Windows desktop which worked in a tree view like the "My Computer" icon, it would be very hard to find anything useful, unless you already knew exactly where it was located. That is, of course, why we use URLs. However, the information needed to connect to a remote dataset on an SQL server is quite detailed and moderately complicated, so you cannot reasonably expect people to type the information in as you would a URL. We therefore need some way of registering TimeMap-enabled datasets with a central catalogue.

We have set up such a catalogue in an SQL server database (Interbase 4) at the Sydney University Archaeological Computing Laboratory. This catalogue is accessible both to the TMView software (Fig. 5) and to a web-based search page (Fig. 6). The catalogue stores a title (for listing in the TMView connection dialogue), connection parameters for the dataset (host address, port, database name etc.) and metadata describing the content, spatial extent and ownership of the dataset (using Dublin Core metadata elements, extended as necessary). The web-based search operates primarily on the metadata, whereas TMView primarily uses the connection parameter information.

We have also written software (TimeMap Metadata Manager, Fig. 7) to update the catalogue remotely across the Internet, allowing individuals to register their datasets with the TimeMap project and enter metadata without intervention from us, enforcing standard metadata element names, schemes and authority lists. New records are flagged, allowing us to vet and approve each entry before it appears in the catalogue. The software also allows us to update the metadata definitions to incorporate new elements or add new schemes and authority lists. The catalogue will eventually be mirrored at the Center for Advanced Spatial Technologies, University of Arkansas (http://www.cast.uark.edu) and the Archaeological Data Service in the UK (http://ads.ahds.ac.uk/ahds/).

Connection to remote datasets

The process of connection to a remote dataset is illustrated in Fig. 8. The dataset is located through the TimeMap catalogue described above, and the metadata is read from the TimeMap metadata table attached to the database. When the user defines a new map layer through an SQL query, the full query is built starting with the current domain (space-time cube) and the base query or view specified in the metadata, to which the user's SQL query is added. This query is submitted to the remote server and the result set is downloaded to the local machine, where it is stored in a standard TimeMap local dataset with its own metadata. Once available

locally, the current time query is applied and the results are displayed – the only difference between this and a local dataset is periodic updating of the data as required from the remote source. [7]

We are working at the moment on storing datasets as standard textual databases. This allows the use of any SQL server, including cheaper and lighter products such as MySQL or Interbase, rather than demanding a spatially-enabled database server such as Oracle or Inform. Since TimeMap needs to work with geographic objects[8] and since standard databases do not store geographic objects explicitly, we are working on ways of storing geographic objects as binary objects and creating geographic objects from this data on the fly once downloaded. To allow efficient retrieval of spatial data, the bounding rectangles of the geographic objects are also stored in the textual database, allowing simple spatial filtering on the server with standard (non-spatial) SQL.

Temporal Interpolation

As noted earlier, one of the strengths of the snapshot/transition model is that it allows us to interpolate the state of objects between snapshots (Fig. 2). This is essential if we wish to be able to display maps for any given point in time, whether or not the date in question corresponds with the dates at which specific information was recorded.

People are used to doing temporal interpolation by hand. A historical synthesis may use a variety of resources to discuss, for example, the layout of a town and its evolution through some range of years, with reconstruction maps at particular points in time based on an interpretation of more, or less, contemporary information. The methodology behind the interpolation is rarely laid out explicitly.

The TimeMap data structure, on the other hand, records explicit codification of the rules for temporal interpolation between snapshots (which are, however, themselves often based on syntheses), allowing automation of the temporal interpolation process and the generation of views at any arbitrary time interval. This is an essential prerequisite for the development of animations (see later). Temporal interpolations provide us with a best-guess of the intermediate state of the mapping domain – they don’ t pretend to reflect reality, merely our best ideas about it.

Rather than a simple linear interpolation between snapshots, we allow for the recording of a number of possible interpolation schemes which can be selected according to knowledge about the nature of the evolution of the entity. For example, if we have the position of the end of a railway line at two dates, we can interpolate linear growth between the two positions in the absence of other knowledge, but if we know that there was a prolonged strike near the earlier date we might specify a transition function with a slow start and late spurt.

---

[7] TMView will manage periodic updates of the local dataset from the source (on increase of domain, at startup, at specified intervals or only on specific request).

[8] Consisting of points, lines and polygons represented by sequences of latitude/longitude or UTM x/y coordinates.

We have yet to define a full set of transition functions, and it is probable that the list will be open-ended. Similarly, we have as yet little idea how we are going to cope with spatial and temporal fuzziness in building interpolations (another good project for a PhD thesis). On the other hand, we have made good progress towards developing routines to carry out interpolation between two non-fuzzy incarnations of a geographic object, and hope to be able to start incorporating them into our software within the year.

Display methods

Traditional 2D display methods (time slices, symbolism, arrows and change maps) are quite inadequate for representing complex or continuous phenomena such as growth and collapse of empires, the development of a city or the ebb and flow of a military campaign.

The TimeMap project is developing two approaches for representing complex phenomena. The TMView mapping front-end tackles the problem first by applying time-based queries on the fly and displaying a map showing features which were extant in the time interval specified. However, this can become very confusing when an object is changing its form continuously and there are many snapshots to display, and it does not show the progress of change.

The next step is to interpolate the snapshots to display our best-guess view for a specific point in time. This gives a much less confusing view, but it is still a static view – it fails to show what happened before and after the selected date, or the dynamics of the changes.

The only really satisfactory solution to displaying time-varying phenomena is to use time itself to display them in the form of animations. Within the TMView application these animations will be generated on-the-fly from the underlying database, using automated interpolation of snapshots to generate the intermediate frames. This is our target over the next year. Our aim is to display the animations in such a way that they can be moved forwards, backwards or stopped to study their pattern – they will not be simple videos which are played once or in a loop – and the animations will link to other data in the same way as static maps e.g. clicking on a map object will bring up information relating to that object at the point in time displayed in the animation (presumably pausing the animation), with links to earlier and later information.

It is worth noting that animation of maps is not uncommon, but most examples suffer from two major problems. First is the quality of the animations, which are often produced with available tools for a lowest-common-denominator platform. Second, and far more important, is the one-off nature of most animations.

One-off animations are generally based on data which is collected and prepared specifically as a base for the animation, rather than as a multi-purpose database. The data is often recorded in arbitrary pixel-based coordinates, rather than geographic coordinates, and the animation is generated from these purpose-specific resources using the best available tools and some considerable degree of technical expertise. Consequently, the production of a new animation with different data or different parameters or incorporating extra data layers, means redoing all or part of the work. It is not enough to specify a different database or a different query and press the button again.

On the other hand, the type of data-based animation we are developing for TMView should allow the naive user to generate their own animations on demand, given an appropriate

database., and also to add in, remove or change the symbolism of any type of GIS layer (time enabled or not), This allows for a what-if approach to the data rather than the sort of static pre-digested view of the data available from traditional animation products.

Having said all this, the AsiaMap component of the TimeMap project has been developing one-off animations of Asian empires, notably the Mongols (Fig. 9).using Houdini, a high-end animation package running only on Silicon Graphics workstations. The aim of these animations is to demonstrate the potential of snapshot interpolation in producing smooth animation of cultural features and the preparation of visually striking video sequences for public display. Although the animations themselves are one-off productions, the data used to generate these animations were drawn from digitised geographically-registered outlines stored in a TimeMap GIS dataset, so the data can be used later as a basis for TMView enquiry and animations.

Linking to other resources

Information relating to cultural features shown on a time-enabled map interface may be stored as attribute data within the GIS dataset, or it may be available elsewhere on the Internet as static or dynamic web pages.

Our methodology allows the recording of two types of URL:

1. **Static URLs** are those which load a specific web page whose URL is simply recorded in a field in the dataset. One or more URLs can be recorded for a single object – these might be used, for example, to link to special web pages that describe an archaeological site or monument, or to link to the home page of a museum, library or cultural institution.

2. **Dynamic URLs** are those which link to web pages generated on-the-fly from a database. Dynamic URLs will typically be used for features or objects which are inventoried in a web-enabled museum catalogue, monuments or archive inventory. TMView reads the accession number or other identifying field(s) for an object and generates the appropriate URL to retrieve the dynamically generated page for the object. The instructions for doing this are stored in the metadata attached to the dataset rather than being hard-coded into the software.

In the current TMView interface, linked web pages are displayed by selecting one or more objects on the map, using pointer or marquee selection tools, and then clicking the browse button (globe icon). This opens a browser window with a pull down list of all the URLs relating to the objects selected. Selection from the list loads the appropriate web page. We are about to change this mechanism, however, so that clicking the browse button will generate an HTML file listing the links, together with other information drawn from the database, and open this file in the user' s standard web browser.

CONCLUSION

In this paper I have attempted to rapidly cover all of the main areas of methodological research and development which form part of the TimeMap project. Current work is focussing on connection to remote datasets with the TMView front-end and the development of inline animation. We hope to have prototype software available for beta testing before the end of 1998.

Up-to-date information on the TimeMap project, as well as on AsiaMap and Virtual Historic Sydney, can be found on the following URL: http://www.timemap.net

Acknowledgments
I wish to thank Pat Caldon, Steven Hayes, Reba Kearnes and Andrew Wilson for their work on software and methodology development for the TimeMap project; Andrew Wilson and Roland Fletcher for permission to use material from the Virtual Historic Sydney and AsiaMap projects respectively; Suzanne Wayne for editorial assistance. The satellite base image for the AsiaMap animations was kindly provided by WorldSat Inc. of Toronto. The TimeMap project is funded by the University of Sydney through the Pro-Vice-Chancellor (Research). My participation in the PNC Conference was kindly sponsored by Lewis Lancaster and the Electronic Cultural Atlas Initiative.