MODELLING, QUERYING AND DEVELOPING SPATIOTEMPORAL APPLICATIONS BASED ON VRML

Alex Vakaloudis, Babis Theodoulidis
Timelab, Dept. of Computation
UMIST
{alvak, babis}@sna.co.umist.ac.uk

ABSTRACT

Despite the number of proposals for processing 2D spatial data, the areas of 3D spatial information is still quite empty. Moreover, only few attempts appear towards incorporating the temporal and the 3D spatial domains. At the same time the vast growth of Internet and WWW has set a new domain for applications. One of the most promising Web browser plug-ins is VRML, the means for 3D spatial navigation. With the work presented in this paper we wish to extend this functionality by providing temporal navigation, data storage and spatiotemporal querying. The result is an architecture for 3D Spatiotemporal databases accessible over the Internet. A data model for spatiotemporal applications (3D-STDM) is analysed. It follows a technique separating spatial and non-spatial information. A bridge table is used to connect these two parts. The spatial part is described by a set of tables that cover the spatial and the positional characteristics of an object. By timestamping the object tuples and identifier, a unified spatiotemporal entity is generated. It has continuous temporal characteristics; two successive object states are connected via the trend field which records the object transitional evolution. A query language (3D-STQL) for spatiotemporal object manipulation is also provided. Apart from the operators, it contains predicates dependent or independent on user's viewpoint. It also includes temporal and spatiotemporal operators that capture specific spatiotemporal features. 3D-STQL is interfaced to VRML via WWW Browser providing remote access to spatiotemporal information. The entire framework complies with the VRML standard.

Keywords: VRML, Spatiotemporal Databases, Internet databases

1. INTRODUCTION

The domain of two-dimensional spatial databases is rich due to the large and increasing commercial and academic interest in GIS. On the other hand, the three-dimensional spatial domain was in the past neglected due to the inefficiency of computer power to support the complicated 3D geometrical calculations. Nowadays it has begun to attract the attention of researchers and developers. It offers the basis for more realistic presentations and opportunities for visualisation of more data. Possible applications for 3D data include advanced geography, architecture, astronomy and also all the areas of scientific visualisation.

At the same time, the latest advances in communications and networking established Internet and WWW as major sources of information. A number of Internet databases have already been constructed for remote use via a Web-browser interface. Technologies assisting to this effort include Java's Remote Method Invocation and Java Database Connectivity (JDBC) [JDBC97]. Even GIS applications have appeared that employ Internet access. Being in rather early stages of development they currently supply the standard GIS functions (zoom-in, zoom-out, pan, identify) implemented in Java [IGIS97] [ESRI97].

The expansion of Internet is also supported by a set of graphical plug-ins, which enable users to access graphical data through a Web browser. The most exceptional is arguably *Virtual Reality Modelling Language* (VRML) [ISO97], a tool for representing 3D graphics and Virtual Reality in WWW [VRML96]. By examining the number of VRML browsers, authoring tools and worlds we can safely assume that it has been adopted as the typical way for constructing virtual presentations.

VRML is the means for spatial navigation [SDSC97]. With the present work, we wish to extend its services by providing a mechanism for temporal navigation. Such a task is eased by inclusion of time among the VRML concepts and also by its routing and scripting utilities. Moreover we seek to enhance VRML's functionality with database storage, data querying and manipulation.

This effort results to the specification of a 3D Spatiotemporal Database interfaced to the Internet through a VRML browser. Even more it is the foundation for increasing VRML object reusability [Walcz97]. By maintaining a permanent repository of VRML objects, the Virtual Worlds developer can select the objects that fit to his needs and therefore (re)use them for dynamically constructing multiple worlds.

The remainder of this paper is structured as follows: Following we describe the data model for storing 3D Spatiotemporal Data and we show how it can be the basis for developing spatiotemporal applications. Paragraph 3 analyses the query language for the selection of objects through spatiotemporal criteria. Next we present the interface architecture that permits remote access and querying of 3D Spatiotemporal databases by using a WWW Browser. We close with references to related and future work.

2. 3D SPATIOTEMPORAL MODELLING

2.1 Basic Concepts

In this section we discuss the basic concepts for modelling 3D spatiotemporal entities within a database repository. This effort (termed 3D-STDM) relies on the relational model and aims to capture the spatiotemporal characteristics of an application. More specifically, it is constructed in order to fulfil the following requirements:

- Record the 3D spatial characteristics of an object with an open format. This includes modelling of its shape, appearance and positioning in the world.
- Adopt a format similar to VRML in order to facilitate translation from and to VRML.
- Allow grouping of spatial objects so as to describe composite, real world objects. For example permit the modelling of a building comprised by many windows, doors and walls.
- Model the temporal evolution of spatial and non-spatial objects.
- Make the querying (selection, spatiotemporal comparison) of spatiotemporal objects feasible.

First, let us consider a primitive 3D object such as a box, sphere, cylinder or cone. Its spatial characteristics may be easily modelled with VRML. Using the VRML mechanism, different attributes are used to describe the object's appearance, geometry and placement. Therefore a 3D Object VObject can be defined as:

Definition: A *VObject* is a tuple:

VObject {id, geometry, appearance, placement}

where *geometry* models the geometry of the object, *appearance* its appearance and *placement* its positioning. Placement is extended by *Orientation*, a 3D vector that points to the top side of the object, if such exists and will be used later as an argument for Viewpoint-independent operators.

Here, we extend this modelling approach by incorporating temporal features. First we give the definition of the VTime concept:

Definition: A Timestamp *VTime* marks the valid and transaction time for a recording and also connects to the previous recording.

VTime: {(valid_time, transaction_time), (mode, value)}

The first pair timestamps the spatial attribute's state. Both valid time (i.e. particular state's time) and transaction time (time of the recording) are noted. The second pair is termed *trend* and connects two successive recorded states so as the user can interpolate into the period they form and thus extract a non-explicitly recorded situation. Let us consider the example of a steadily moving car whose position is known at 1:00 and at 1:15. With the help of trend we may compute its position at 1:05, 1:08 and 1:14. The arguments to the trend interpolation procedure are mode, value and specify the way mutation occurs (e.g. linear, acceleration, cyclic mode). Obviously object variations can happen in a number of possible ways depending on the nature of a any usage. The application developer must choose and use the modes appropriate for a specific application.

The requirement for the inclusion of trend features in the temporal semantics derives from the fact that 3D Spatiotemporal objects have continuous behavioural characteristics. Because they represent real-life objects (and not their abstraction as it is the case for the 2D domain) they simply cannot disappear. They move, rotate change shape and appearance but always maintain their existence even in more than one form. Steel and glass for example are used for building a car and after twenty years, when car's life is over, they may be part of a scrap yard. The material exists forever even if it changes pattern (steel-car's framework, scrap metal). To model the ever-lasting existence of 3D objects we introduce Lifestamp:

Definition: Lifestamp connects two different formations of 3D a spatial object. It does so by pointing to the object that is the continuation of the current object. It also timestamps the moment of the change (i.e. death of object).

Lifestamp {NextID, VTime}.

where: NextID points to the object ID,
VTime is the timestamp for the change of formation

The combination of VObject and VTime leads us to the definition of VTObject, the concept describing the storable 3D Spatiotemporal primitive object:

Definition: A Spatiotemporal Object VTObject is a VObject whose attributes have the VTime timestamp attached on them.

VTObject: VObject x VTime $\Leftrightarrow$
VTObject: {id, (geometry, VTime) (appearance, VTime) (placement, VTime)}

Having defined the 3D Spatiotemporal primitive object, the next task is to define a grouping mechanism in order to represent real-life objects and moreover to link them to their non-spatial attributes. To achieve this we first introduce the notion and Index.

Definition: An Index describes the relationship of a graphical object with its non-spatial attributes. More specifically it links the VTObject.id with the table (Table.Name) and the entry (Table.KeyField.id) where the non-spatial attributes of the object are described.

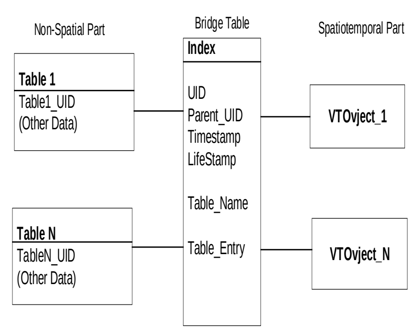
Index:{cVTObject.id, pVTObject.id, Table.Name, Table.KeyField.id, VTime, Granularity, Lifestamp}

where: cVTObject.id, pVTObject.id are VTObject ids,
Table.name is the name of the table where the object is classified,
Table.KeyField.id is the key value in that table,
VTime is the timestamp of the tuple
Granularity is the granularity of all the temporal values associated with the object
Lifestamp points to the previous formation of this object

The first VTObject.id points to a primitive object while the second to its parent, which also has a placement and maybe a specific appearance. (e.g. five cylinders comprise a table; a table and two chairs are inside a room etc.) This part-of relationship can change (somebody moved the chair outside the room) and hence is considered temporal. Table.Name and Table.KeyField specify the real-life object that the VTObject graphically describes.

2.2 Developing Spatiotemporal applications

Figure 1 demonstrates how to develop a spatiotemporal application by incorporating the concepts described above. We follow a segregated technique which separates spatiotemporal from non-spatial data and employs *Index* to link them together. The spatiotemporal part consists of VTObject entries and records the shape, appearance and positioning of



a spatial object through time. In the case of the UMIST Database for example it would contain the Main Building's dimension, address, colour etc. The

Fig.1
Modelling Spatiotemporal Applications

non-spatial part has all data without spatial context. These may refer to spatial and/or non-spatial objects. In the UMIST case it could include the name of any building, its constructor, its usage and also details about the financial status of UMIST, employee records etc.

The non-spatial part concerns general modelling and totally depends on each application thus lying outside our interests. The spatiotemporal part though is common for every application. Figure 2 goes further in detail and presents the relational modelling for this part by implementing the main concept of VTObject.

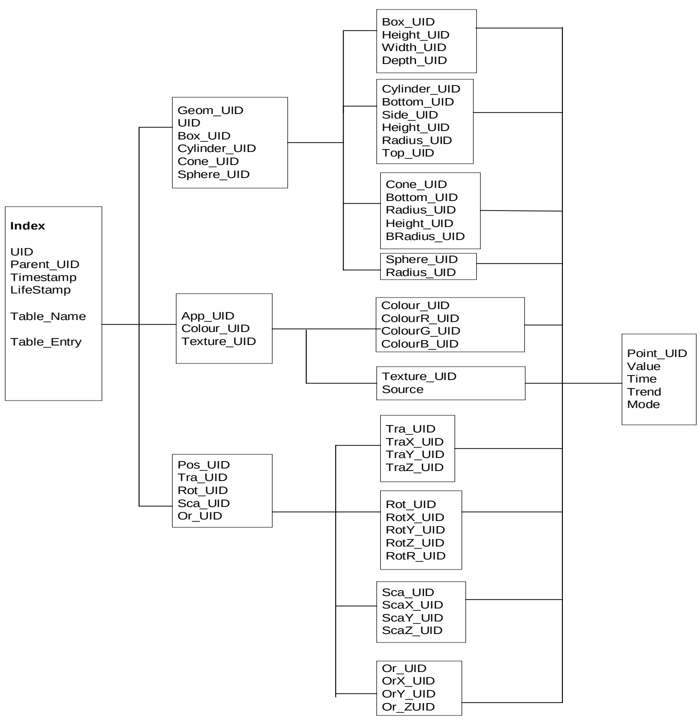


Fig 2
The Spatiotemporal Part of the application

For the population of the spatiotemporal part we use the *Reader* graphical tool. Reader (Fig. 3) gets its resources from existing VRML files because they contain customised and visually tested 3D objects. It parses the VRML file, identifies the scene graph and the spatial-visual values of the objects that belong to it. Then, it is up to the user to associate parts of the scene graph (a.k.a. VRML objects) with entities of the non-spatial part. Reader updates the Index field and prompts for association using the trend method to previously recorded values if such exist.

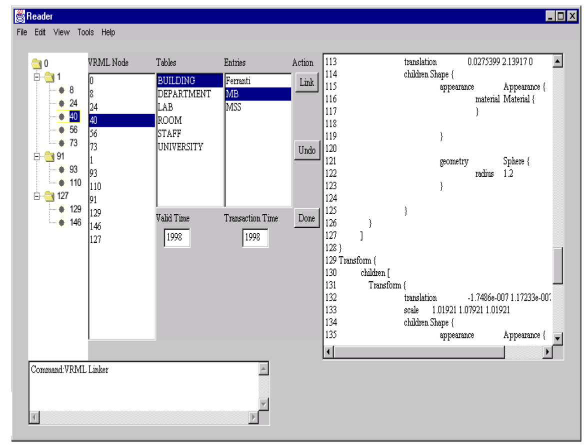


Fig. 3
The Reader tool feeds the Spatiotemporal Repository with the contents of VRML files

3. QUERYING 3D SPATIOTEMPORAL DATA

Storing objects in the spatiotemporal repository is meaningless unless we also provide a method for extracting the appropriate information. The selection of objects is mainly done according to spatiotemporal criteria. However, common attribute (i.e. non-spatial) parameters may also come into play. A query example mixing spatiotemporal and attribute criteria is “Give the thick books printed by McGraw Hill during 1998”. “Thick” refers to geometry while “McGraw Hill” is attribute information. The most preferable technique that unifies spatial and non-spatial querying is an extension to SQL (e.g. implemented using Oracle PL/SQL). This way we can ask for attribute information and at same time by using the operators below, to investigate the spatiotemporal repository. The set of operators consist of temporal, conventional spatial (i.e. ones derived from the 3D domain and defined again here), 3D specialised spatial (that use specific 3D features) and spatiotemporal predicates briefly analysed in this section.

We first introduce three temporal operators, which return a VTObject over a period, select a snapshot of a VTObject (i.e. a VObject) and examine if a VTObject exists under a certain form at a time t.

• [VObject] at (VTObject, VTime)
(Returns VTObject corresponding to VTObject at VTime)
• [VTObject] choose (VTime, VTime)
(Returns the VTObject in a period)
• [boolean] exists (VTObject,VTime,Table.Name)
(Checks if VTObject exists in a specific form at VTime)

We see that transition from the spatiotemporal to the spatial domain is smooth because a VObject is produced by applying the *at* operator to its respective VTObject.

Spatial operators have been extensively researched in the 2D domain (12). They are also employed here:
• [boolean] equal (VObject, VObject)
(The two objects consist of the same parts)
• [boolean] disjoint (VObject, VObject)
(The two objects do not have any common parts)
• [boolean] meet (VObject, VObject)
(Some parts of the objects meet while the rest disjoint)
• [boolean] in (VObject, VObject)
(One object is entirely inside the other)
• [boolean] overlap (VObject, VObject)
(There exist parts that overlap, in and disjoint)
• [float] distance (VObject, VObject)
(The distance between two objects)
• [float] volume (VObject)
(The volume of a VObject)

In short, the definition of disjoint is given:

Disjoint (VTOject1, VTObject2) ⇔
VTOject1 and VTObject2 have no common parts. If VTOject1, VTObject2 are not primitive objects, this is delegated to all their children.

Apart from these conventional predicates 3D worlds offer opportunities for full interaction. In the 3D domain, the user becomes an integral part of the overall setting. With the 2D maps he remains external to the environment whereas in 3D worlds he may navigate in all three dimensions. Moving inside the world, his positioning influences the way he conceptualises the relationships among the objects. Figure 4 exemplifies this fact. The two boxes have stable positions. Nevertheless Viewer 1 believes that Box 2 is behind Box 2. On the contrary, Viewer 2 assumes that Box2 is in front of Box 1. A third Viewer thinks that Box 1 is on the left of Box 2.

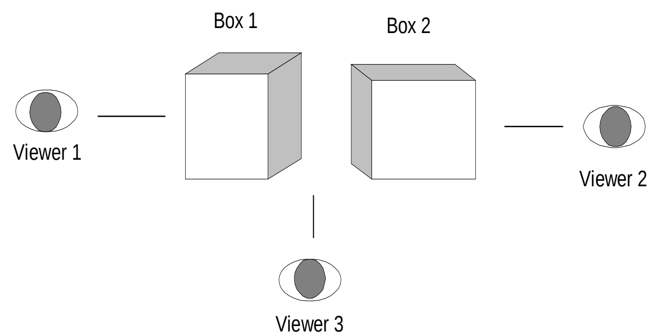


Fig. 4
Different viewpoints lead to different relationships

To capture all these occurrences we introduce the concept of viewpoint:
Definition: Viewpoint represents the user's view as

Viewpoint {position, orientation, direction}

where position is a Cartesian 3D point
orientation is a Cartesian 3D point
direction is a Cartesian 3D point

Viewpoint consists of three 3D points. The first one marks the user's position, the second his orientation and the third his viewing direction. Viewpoint is employed as an argument to the following Viewpoint dependent operators:

• [true, false, indefinite] infront (VObject, VObject, Viewpoint)
(The first object is in front of the second one as seen from the viewer in Viewpoint position)
• [true, false, indefinite] behind (VObject, VObject, Viewpoint)
(The first object is behind the second one as seen from the viewer in Viewpoint position)
• [true, false, indefinite] up (VObject, VObject, Viewpoint)
(The first object is above the second one as seen from the viewer in Viewpoint position)
• [true, false, indefinite] below (VObject, VObject, Viewpoint)
(The first object is below the second as seen from the viewer in Viewpoint position)
• [true, false, indefinite] left (VObject, VObject, Viewpoint)
(The first object is on the left of the second as seen from the viewer in Viewpoint position)
• [true, false, indefinite] right (VObject, VObject, Viewpoint)

(The first object is on the right of the second as seen from the viewer in Viewpoint position)

The indefinite result occurs when non-visible objects are examined. The assumption made here is that the user can only compare objects that are on his visible part of world which is determined by the values of the Viewpoint vectors.

Apart from the viewpoint operators there is a need to cater for relationships independent from the user's viewpoint. Some 3D objects have a certain directioning specified by the Orientation vector. For example “a glass is on the table” no matter where the viewer is.
The functions addressing this matter are:

- [true, false, indefinite] upon (VObject, VObject)

(The Orientation vector indicates that the first object is upon the second)

- [true, false, indefinite] beneath (VObject, VObject)

(The Orientation vector indicates that the first object is beneath the second)

Finally we include a set of spatiotemporal operators examining spatial mutations over a period of time.

- [true, false, indefinite] grow (VTObject, VTime, VTime)

(Volume grows within the temporal period)

- [true, false, indefinite] shrink (VTObject, VTime, VTime)

(Volume decreases within the temporal period)

- [true, false, indefinite] approach (VTObject, Viewpoint, VTime, VTime)

(The distance between the viewer and the object increases within the temporal period)

- [true, false, indefinite] leave (VObject, Viewpoint, VTime, VTime)

(The distance between the viewer and the object decreases within the temporal period)

The operators listed here describe generic functions. The potential developer may combine them to compose complicated and application specific predicates. For example:

between (VObject1, VObject2, VObject3, Viewpoint): (VObject1 is Between VObject2 and VObject3)
infront (VObject1, VObject2, Viewpoint) AND infront (VObject1, VObject2, Viewpoint)

over (VObject1, VObject2):
(VObject1 is Over VObject2 e.g. glass is Over table)
Upon (VObject1, VObject2) AND meet (VObject1, VObject2)

4. INTERFACING 3D SPATIOTEMPORAL DATABASES TO VRML

The data model and the query language specified in the last two paragraphs follow an open approach and hence can be implemented using any popular 3D format (DXF, 3DS etc.) Since however we based this effort on VRML standard, the obvious choice is to employ VRML browsers for building the user-interface to a 3D spatiotemporal database. Fig. 5 describes the architecture for remotely accessing, querying and displaying 3D spatiotemporal data. The top level illustrates what the user receives on their web browser. This includes the Spatiotemporal presentation via a *VRML browser*, a *clock*, controls for *temporal navigation* (VRML browser handles spatial navigation), a *window for displaying any temporal but non-spatial data.* All the above functionality is embedded in a WWW browser (Netscape, Microsoft IE etc.).

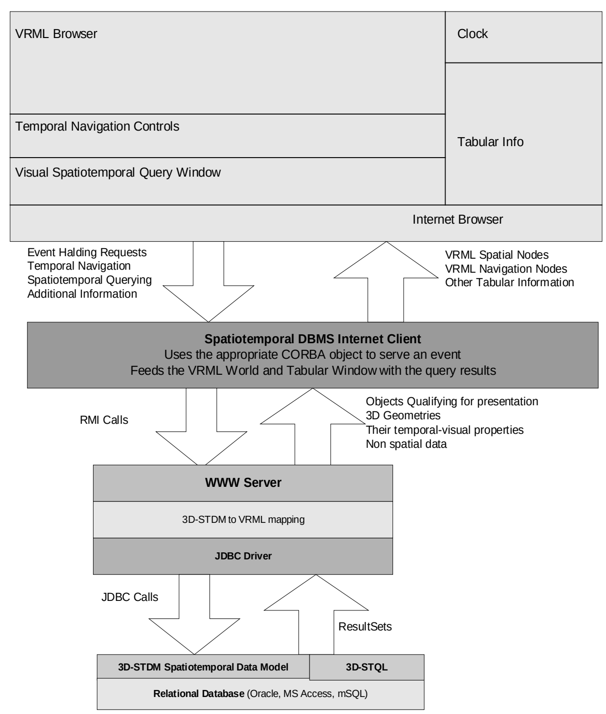


Fig. 5
A three-tier architecture for a 3D Spatiotemporal Database

The user through this WWW interface mechanism makes requests for:

- Temporal navigation
- Spatiotemporal Querying
- Additional Querying

These requests are served by RMI connections to a WWW server. The server-side accesses the underlying database through JDBC calls and executes spatiotemporal queries on 3D-STQL. Query results come back from the database in ResultSet formations. Other RMI objects carry out the mapping from 3D-STDM to VRML and either feed the VRML browser with objects using External Authoring Interface technology [VAGEA97] or the tabular window with non-spatial data.

5. RELATED WORK

The necessity for the persistent storage of VRML Objects has been acknowledged by VAG [VRML97a], the governing body for VRML, which established the Database Working Group [VRML98]. Oracle have submitted a proposal with Java mappings to persistence [Oracl97]. Another approach linking VRML code to Databases is described in [Kamiu97] where only the source code (text) for the VRML Objects is stored and further used for the construction of Virtual Worlds by non-graphical criteria.

In the spatiotemporal databases domain, researchers are integrating spatial and temporal concepts [Choro98]. The simplest technique is the snapshot model that considers time as the third (or fourth) geometrical dimension and timestamps the state of world at a particular point {Lagra92]. Worboys introduces the ST-simplex, a spatiotemporal structure that supports 3D space and 2D time [Worbo94]. Moving points and regions [Gutin98] is another spatiotemporal structure that conceptualises lines as points evolving over time. The history graph model introduces the idea of history graphs to provide historical links between versions of objects. De Cambray employs behavioural functions to link two explicitly known states of 3D objects [Yeah92] thus maintaining continuous knowledge about object status. Hamre, [Hamre94] uses a four-dimensional structure to model points in the 3D space and time field.

A notation for spatiotemporal queries has been developed by researchers in TimeCenter [Tsotr97]. It is based on previous work of the authors in the Temporal databases domain. A set of spatiotemporal operators is discussed in [Worbo96]. MOST is an effort to query spatiotemporal objects in future time where multiple states might occur [Sist98]. Finally, extensive surveys on Spatiotemporal Databases have been compiled by Armenakis Armen92] and Abraham and Roddick [Abrah98].

6. CONCLUSIONS AND FUTURE WORK

We introduced a set of concepts for the storage and the selection of 3D Spatiotemporal objects based on the VRML standard. We also showed how to use these concepts as the foundation for developing spatiotemporal applications. The data model (3D-STDM) and the query language (3D-STQL) try to capture the advanced 3D Spatiotemporal features of an application. The spatial objects and their histories are recorded in a database repository. Their selection is delivered by applying a set of spatial, temporal and spatiotemporal operators. To make full use of the special characteristics of the 3D domain we employed the user's viewpoint as a query argument. The objects retrieved are translated to VRML code, become remotely accessible through RMI and may be viewed in any VRML Browser.

The entire framework is a complete approach to real-world modelling as it offers remote access and exploitation of data varying in all 3D spatial and temporal dimensions. Furthermore it contributes to the effort of unifying VRML with databases and GIS [Roehl97] [Moore97]. Parts of virtual cities can be presented according to the user's criteria (e.g. "Give me all the high buildings of the City Centre"). By using the Viewpoint node of VRML [VRML97b], the viewer can also navigate in dynamically constructed virtual worlds.

Future work involves the specification of an event model and its links to the spatiotemporal objects that have been selected by spatiotemporal and attribute criteria from the database repository. Events generated by the user and the application will be captured and trigger further actions in the 3D environment.

7. REFERENCES

[Abrah98] T. Abraham, J.F. Roddick, Survey of Spatiotemporal Databases, Kluwer Acad. Publishers, Boston 1998

[Armen92] Armenakis C., Estimation and organisation of Spatio-temporal Data, Proc. Of Canadian Conf on GIS, Ottawa, Canada, 1992

[Choro98] Chrorochronos project Newsletter, No.2 30/04/98.

[Egenh89] M. Egenhofer, A. Frank, J. Jackson, A topological data model for spatial databases, 1989 Springer Lecture Notes in Computer Science Vol. 409.

[ESRI97] ESRI, MapObjects Internet Server, http://maps.esri.com/MapObjects/demos.htm, 1997

[Gutin98] R-H Guting, M. Bohlen, N. Lorenzos, M Schneider, C. Jensen, M. Vazirgiannis, M. Erwig, A Foundation for representing and querying Spatiotemporal Data, Chrorochronos technical meeting 30/04/98.

[Hamre94] T. Hamre, An Object-Oriented Conceptual Model for Measured and Derived Data Varying in 3D Space and Time, Int. Symposium on Spatial Data Handling 1994.

[IGIS98] InternetGIS, http://internetgis.com, 1998

[ISO97] International Standard ISO/IEC 14772-1:1997.

[JDBC97] Javasoft ,JDBC Specification, 1997, http://splash.javasoft.com/jdbc

[Kamiu97] M.Kamiura, H. Oiso, K. Tajima, K. Tanaka, Spatial Views and LOD-Based Access Control in VRML-object Databases Springer Lecture Notes in Computer Science Vol. 1274., 1997

[Lagra92] G. Lagran Time in Geographical Information Systems, ed. Taylor & Francis London 1992.

[Moore97] K. Moore, J.Dykes, J.Wood, Using Java to interact with geo-referenced VRML within a virtual field course ICA Visualisation Commission meeting, Stockholm 1997

[Nguye96] T. Nguyen, V. Shrinivasan Accessing relational databases from the World Wide Web. Proc. Of ACM SIGMOD Int. Conf. On Management of Data 1996.

[Oracl97] Oracle, VRML Data Repository API, Oracle proposal 1997 http://www.olab.com/vrml/overview.html

[Peque94] D. Peuquet, E. A. Wentz, An Approach for Time-Based Analysis of Spatio-temporal Data, Int. Symposium on Spatial Data Handling 1994.

[Roehl97] B.Roehl, Adding regions to VRML http://ece.uwaterloo.ca/~broehl/vrml/regions.html 1997

[SDSC97] San Diego SuperComputer Center ,The VRML Repository, http://www.sdsc.edu/vrml/repository.html, 1997

[Sistl98] A. P. Sistla, O. Wolfson, S. Chamberlain, S.Dao, Querying the Uncertain position of Moving Objects, Temporal Databases: Research and Practice, Lecture Notes in Computer Science Vol. 1399, 1998

[Teyli97] R. van Teylingen, W. Ribasly, L. van der Mast Virtual Data Visualiser, Vol. 3, IEEE Transactions in Visualisation and Computer Graphics No.1 1997

[Tsotr97] V. Tsotras, C.S. Jensen, R.T. Snodgrass, A Notation for Spatiotemporal Queries, TimeCenter TR-10, 1997

[VAGEA97] VAG 1997, External Authoring Interface Spec, http://vrml.sgi.com/moving-worlds/spec/

[VRML97a] VRML Architecture Group, VRML Specification, 1997 http://vrml.sgi.com/moving-worlds/spec/

[VRML97b} The VRML Architecture Group http://vag.vrml.org

[VRML98] The VRML Database Working Group http://www.vrml.org/WorkingGroups/dbwork/

[Walcz97] K. Walczak, Why Database support for VRML, Basic information track for Computational science, 1997

[Worbo94] M. Worboys, A Unified Model for Spatial and Temporal Information, The Computer Journal, No. 1 1994.

[Worbo96] M. Worboys, GIS: A computing Perspective. Taylor and Francis 1996

[Yeah92] T.S. Yeah, B. de Cambray, Time as a geometric dimension for modelling the evolution of entities: a 3D approach, 1992.