

A Temporal Model of Virtual Reality Objects and their Semantics

Jorge Campos, Kathleen Hornsby and Max J. Egenhofer
National Center for Geographic Information and Analysis and
Department of Spatial Information Science and Engineering
University of Maine
Orono, ME, 04469-5711, USA
{jorge,khornsby,max}@spatial.maine.edu

Abstract

This paper introduces a model to represent the evolution of VR objects over time. Based on this model a new classification of virtual reality objects is proposed and the semantics associated with each class of object are described. This paper also presents a rationale for modifying object semantics under manipulations by an observer of the object behavior. The manipulation of the spatio-temporal configurations of VR objects and their behaviors allows observers to create their own visualization of the environment, gaining insight and discovering relationships among dynamic phenomena.

Keywords: Visualization, virtual reality, semantic modeling, spatio-temporal modeling.

1. Introduction

Geographic Information Systems (GISs), like many scientific graphic-intensive applications, are extending their representational capability and presenting information in a multi-dimensional, multi-sensorial, and immersive milieu (i.e., virtual reality environments). The integration of Virtual Reality (VR) with GISs has been reported in such applications as urban planning [15], ecology [12], data visualization [8], and archaeology [3]. These applications typically combine standard GIS capabilities with a VR interface to visualize and explore geographic information.

The concept of VR technology was introduced in the early 1970s. This technology, however, has still not matured and is the subject of a number of different scientific communities (e.g., computer science, psychology, and philosophy) to explore the cognitive validity of such a technology as a medium for visualization of information. In a broad context, the sense of immersion, navigation, and interaction in a virtual environment have been recognized as a key factor of VR technology [13]. Research in human-computer interfaces [11] and an increasing exposure of users to VR technology will soon overcome current users' lack of expertise in dealing with virtual environments. The next generation of GIS users has been mastering their ability to navigate and explore VR environments, today a ubiquitous interface for computer games and some educational software. In addition to the visual outcome of such computer programs, their success also relies on the new way to present information. VR is a medium that comes close to the way humans perceive information [7], reducing the impedance between the representation of the information and people's mental conceptualizations of space and time.

This paper presents a new classification of VR objects based on their existence, visibility, and activity characteristics and describes associated semantics. The semantics play a significant role for the interaction between VR objects, as well as for the interaction between observers and the virtual environment. Our work focuses on modeling the evolution of VR objects over time and provides a rationale for modifying the semantics of these objects through the manipulation by an observer of the object behavior.

The remainder of this paper is structured as follows: Section 2 describes the structure and operations of an animation data model. Section 3 discusses the characteristics used to define the semantics of VR objects. Section 4 introduces a classification of VR objects based on an exhaustive combination of their individual semantic characteristics. Section 5 presents the set of operations used to model the semantics of VR objects. Section 6 introduces the history model of a VR object (i.e., a model to represent the evolution of the semantics of an object over time). Section 7 describes the rationale used to change the semantics of VR objects through the manipulation of their behavior. Section 8 draws conclusions and indicates future work.

2. Animation Model for VR Environments

The scientific visualization community has proposed many different data models [5,9,14] to represent time variation of information through animations. An analysis of many data models that support computer animations shows an emphasis on rapid developments of prototypes and portability between applications, but shows little support for manipulations of animations. The conceptualization of such data models has driven implementations with very low

levels of abstraction and without sufficient information and methods to combine and manipulate animations. Users are constrained to seeing animations by mimicking VCR operations in which the number of animated objects in a scene can easily exceed the users' capacity to understand the dynamic environment. In order to overcome such limitations we introduce an animation model for VR environments. The major goal of this model is to provide a high-level abstraction of the objects' behaviors and to identify a reasonable set of operations that allow users to easily combine and manipulate such behaviors.

The modules of the animation model (Figure 1) represent increasing abstractions of the animated objects' behaviors. Animations are built up constructively from the lowest to the highest level of abstraction in the model.

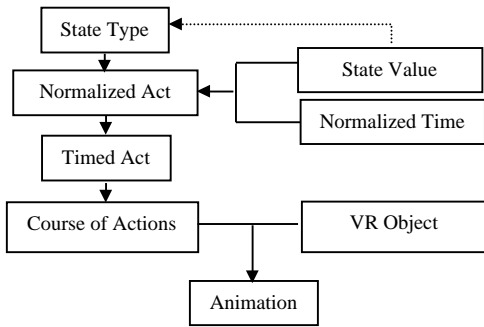


Figure 1. Modules of the animation model.

State Type represents types of attributes for which the user can sense the variation. *Normalized Act* is an abstraction composed of a *State Type*, an ordered sequence of *State Values* and *Normalized Times*, and an interpolation function that generates intermediate state values of the act. *Normalized Act*, *Timed Act*, and *Course of Actions* represent different stages of the behavior of the animated phenomena. *Animation* aggregates pairs of *Course of Actions* and *VR objects* representing the dynamic environment. *VR object* is an abstraction that represents the graphical presentation of the animated object. The unique restriction applied to these objects is that they need to be compatible with the *Course of Actions* (i.e., the object needs to have all attributes being animated in the *Course of Actions*).

2.1 Model of Time

Normalized Act, *Timed Act*, *Course of Actions*, and *Animation* have their own temporal model. All of them have a simple linear model of time, that is, they define simple intervals in their respective time spaces (Figure 2).

Normalized Acts define an interval over a normalized temporal space (t_{na}), whereas *Timed Acts* define an interval resulting from the mapping of *Normalized Acts* over the positive space. *Normalized Acts* are mapped onto the *Timed Act* space (t_{ac}) by specifying their duration. Such a mapping is equivalent to performing a scaling of the normalized

space. In this way, both intervals have their starting time coinciding with the origin of their respective coordinate system.

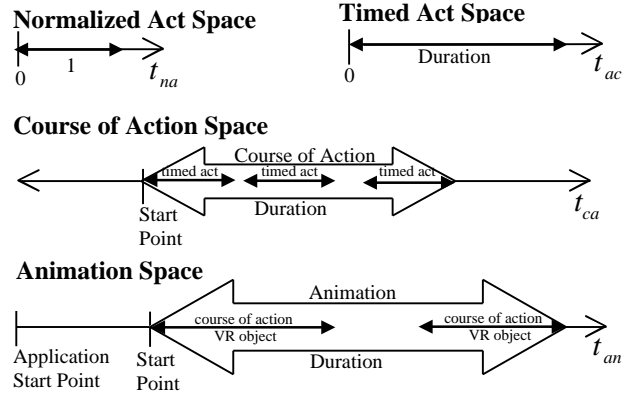


Figure 2. Temporal models of *Normalized Acts*, *Timed Acts*, *Course of Actions*, and *Animations*.

Course of Actions defines a simple interval resulting from the combination of *Timed Acts*. *Timed Acts* are positioned along the *Course of Actions* timeline (t_{ca}) by specifying their absolute start point on the *Course of Actions* space or by specifying a temporal relation between existing *Timed Acts*. Such a transformation is equivalent to performing a translation of the *Timed Act* space. *Courses of Actions* are combined together with *VR objects* to form more abstract entities, called *Animations*. *Animation* also defines an interval in a positive space, where the origin of the animation space coincides with the start time of the application. *Courses of Actions* are mapped onto the *animation* space (t_{an}) in a similar fashion, by specifying a start point in the *Animation* domain or through temporal relations between existing *Courses of Actions*. Thus, a direct mapping of the *Course of Actions* space onto the *Animation* space starts the animation as soon as the animation time reaches the start point of the first *Course of Actions* in the set.

Animations are formed by a combination of pairs of *Course of Actions* and *VR Objects*. An *Animation* can represent the behavior of a single object, a group of objects, or all objects of the environment. For instance, consider an application running an animation of all vehicles in a city (e.g., cars, buses, and trucks). One can design one *Animation* for every vehicle, different *Animations* for each kind of vehicle, or an *Animation* with all vehicles in the model. The first option is unmanageable for humans, considering the large number of objects in the environment. The second option gives rise to a reasonable number of entities (*Animations*) to be manipulated by users on the fly. The last option is the approach used by existing applications and has the disadvantage of limiting users to explore the animation as a whole.

Normalized and *Timed Acts* are coherent with the intuitive notion of intervals, a time associated with some

event occurring in the world [1]. *Course of Actions* and *Animation*, on the other hand, do not fit well with this notion. We can have a *Course of Actions* or an *Animation* composed by *Timed Acts* or *Courses of Actions*, respectively, which do not meet or overlap, implying a period of inactivity inside the interval. The advantage of reducing such entities to a simple interval is that users can use high level of abstraction of the object's behavior to perform spatial-temporal reasoning. Users seeing *course of actions* and *animations* as a simple temporal interval can easily combine and manipulate such intervals to form a new *Animation* that is more suitable for the exploration and investigation of unusual behavior.

2.2 Animation Operations

The *Animation* module provides a set of high level operations, which allows users to manipulate and combine a *Course of Actions*. It is beyond the scope of this paper to present all operations of every module of the animation model. We selected only a set of operations over entities with a high level of abstraction, whose result change the spatio-temporal configuration of the animation. This set of operations can be divided in two groups: a group that imposes a temporal relation between *Course of Actions* and a group that resembles operations over mathematical sets. The domain of all operations is existent *Courses of Actions* and their associated VR object, whereas their range is a new *Animation*.

The first group of operations is `makeMeets`, `makeStarts`, `makeFinishes`, and `makeEquals`.

The `makeMeet` of two *Courses of Actions* produces a new animation where the second *Course of Actions* follows the first (i.e., the second *Course of Actions* starts immediately after the first). The result is that temporal gaps between the *Courses of Actions* are eliminated. This operation can be used to analyze the behavior of two objects that have a large temporal separation.

The `makeStarts` and `makeFinishes` of two *Courses of Actions* produces a new animation where both *Courses of Actions* start or finish at the same time. These operations can be used to simulate a situation when the simultaneous occurrence of *Courses of Actions* is desired while preserving the original pace of the behaviors used as arguments.

The `makeEquals` of two *Courses of Actions* produces a new animation when both *Courses of Actions* start and finish at the same time. This operation changes the start point and duration of the second *Course of Actions*. The result is that the pace of the behavior of the second argument changes.

The second group of operations is `intersection`, `union`, and `difference`.

The `intersection` of two *Courses of Actions* generates an animation defined over an interval obtained from the intersection of the intervals used as the arguments

of the operation. The result is an animation defined over an interval when only the simultaneous occurrences of both *Courses of Actions* will be present. With this operation a user can verify mutual interferences between the movement of different objects.

The `difference` of two *Courses of Actions* generates an animation defined over an interval when the second *Course of Actions* is not defined (i.e., an interval when only the first *Course of Actions* occurs). This operation permits the user to isolate the behavior of the first object during the period when the second object does not occur.

The `union` of two *Courses of Actions* is a simple combination of two *Courses of Actions* while preserving the temporal configuration of individual arguments. This operation is useful for combining animations forming complex animations.

3. Characteristics of VR Objects' Semantics

The module *Course of Actions* models the quantitative evolution of the behavior of VR objects. This module, however, does not promptly inform how active is an object. Modeling the evolution of the activity of VR objects is an important abstraction, which allows an observer to perform qualitative temporal reasoning about the patterns of VR objects behaviors, identifying frequency, durations and synchronization between objects' activities [2]. *Activity* is a characteristic of an object that represents periods when the object is performing an associated action. Based on its activity an object can be considered *active* or *inactive*. An *active* object has the value of at least one of its attribute different from the value perceived in the previous instant of observation. An *inactive* object, on the other hand, refers to the case where all the values of an object's attributes equal the values of the attributes from the previous instant of observation. A VR object without an associated *Course of Actions* has always an *inactive* state and it is called an immutable object. A VR object with an associated *Course of Actions* has an *active* state during periods with an associated action (i.e., *Timed Acts*) and an *inactive* state elsewhere. This object is called a mutable object.

A model that captures only the active characteristic of the object, however, is not rich enough to model the semantics of such objects in a VR environment. In such an environment it is important to be aware of the visibility and existence of the object as well. Modeling the identities of objects over time has been explored in the context of GISs [6,10] and captures the semantics associated with the concept of an object's existence. In VR data models, the evolution of the object's existence and visibility states are not modeled. The existence of the object is almost always assumed [9] and strongly related with its visibility.

Visibility is the characteristic of an object that determines if an observer can see the object. Based on its visibility, an object can be classified as *visible*, *invisible*, or *non-visible*. A *visible* object is an object that the user can

see. An *invisible* object is an object inside the field of view of the observer and there are no obstacles between the object and the observer, which the observer cannot see. Invisible objects reflect either an intrinsic characteristic of the object or a characteristic that can be manipulated by observers for analysis purposes. A *non-visible* object, on the other hand, is an object outside the field of view of the observer, behind a visible object, or so distant such that it cannot be seen. The non-visible status of an object can only be verified at run-time considering both the observer and the object's position. In this paper we deal only with visible and invisible objects.

Existence refers to the physical presence or occurrence of an object or, for conceptual objects, the belief in or perception of an object [6]. Based on its existence state, an object can be classified as *non-existent* or *existent*. The *non-existent* state describes the case where an object does not exist at the instant of observation. The object has been destroyed, will be created, or simply does not exist at any time. The existence of an object is almost always associated with the notion of its appearance (i.e., the graphical realization of the object). However, the existence of an object does not imply a particular graphical realization and *vice-versa* [4]. Some objects do not have an associated visual representation. Alternatively, an observer can manipulate object visibility, turning the object into an invisible object, for instance for analysis purposes. Other objects do not exist at the instant of observation (e.g., a building that will be constructed in the future). The visualization of non-existing objects is likely to occur as the result of a temporal manipulation of the object existence or activity, discussed later in section 7.

4. Taxonomy of VR Objects

Existence, activity, and visibility are orthogonal characteristics of a VR object. The combination of these three characteristics gives rise to eight possible statuses for an object at a certain time. Objects with different combinations of characteristics carry different semantics in the model. We classify each object in a virtual reality environment based on the combination of these characteristics (Figure 3).

| | | Visible | Invisible |
|--------------|----------|----------------|-------------------|
| Existent | Active | <i>actor</i> | <i>spy</i> |
| | Inactive | <i>scenery</i> | <i>camouflage</i> |
| Non-existent | Active | <i>ghost</i> | <i>fable</i> |
| | Inactive | <i>mirage</i> | <i>myth</i> |

Figure 3. Classification of VR objects accordingly their existence, activity and visibility characteristics.

4.1 Existent VR Objects

Actor, *scenery*, *spy*, and *camouflage* objects are elements of the existent subset of VR object statuses. The existent status of these objects requires that the object exists and is performing the activity, if any, that it is supposed to be performing at the instant of the observation.

Actor is an object with an existent, active, and visible state. An *actor* represents a mutable and visible object performing its associated activity, for example, a car traveling between two cities in a non-stop trip. *Scenery* is an object with an existent, inactive, and visible state. Scenery represents immutable and visible objects, or mutable and visible objects during periods of inactivity. A car that stopped at a gas station, or a building that always has the same size and appearance are examples of *scenery* objects. The semantics associated with *actor* and *scenery* require their visual realizations and that both the observer and other objects be sensitive to the presence of these objects (e.g., an *actor* or a *scenery* object can block the path and the sight of the observer).

Spy is an existent, active, and invisible object. A *spy* represents a mutable and invisible object during the performance of its associated activity. *Camouflage* is an existent, inactive, and invisible object. *Camouflage* represents an immutable and invisible object or a mutable and invisible object during periods of inactivity. The semantics associated with *spy* and *camouflage* objects are similar to the semantics associated with *actor* and *scenery* objects, except that *spy* and *camouflage* objects are invisible. *Spy* and *camouflage* represent objects that do not have a graphical representation or objects that are intentionally hidden by an observer to facilitate the analysis of the environment.

Consider, for instance, a scenario where an urban traffic analyst explores a virtual environment to analyze the flow of vehicles in some critical intersections of the city. During the exploration of the dynamic environment some buildings obstruct the visualization of an interesting configuration of the traffic's flow. Thus, the analyst can hide a group of buildings to clean up his or her field of view facilitating the observation of phenomena of interest. These buildings, originally *scenery* objects, are transformed into *camouflage* objects when hidden by the analyst. A *camouflage* building permits the visualization of every object behind it, but the environment is still sensitive to the physical presence of this object (e.g., a *camouflage* building blocks the path of the analyst).

4.2 Non-Existent VR Objects

A model that has only existent objects is not rich enough to represent the semantics of all objects in the VR environment. Including additional semantics that treat non-existent versions of objects gives us a further group of VR objects.

Ghost, *mirage*, *fable*, and *myth* compose the non-existent subset of VR object statuses. *Ghost* is an object with a non-existent, active, and visible state, while *mirage* is an object with a non-existent, inactive, and visible state. *Ghost* and *mirage* share some semantic characteristics with their existent versions (i.e., *actor* and *scenery*, respectively); the only difference is that the observer and other objects are not sensitive to the physical presence of *ghost* and *mirage* objects. *Ghost* and *mirage* are useful to visually compare existent and non-existent objects while avoiding the interference of the non-existent ones in the environment. Consider, for instance, a scenario where an observer explores an archaeological site. Some buildings in this scenario still exist while others have been destroyed. The status of the existent buildings is set as a *scenery* object, while the status of non-existent ones is set as a *mirage* object. The semantics associated with *mirage* buildings requires their visual realizations, but does not impose physical constraints on the environment (e.g., the observer can walk through *mirage* buildings). If the archaeological site has active objects of the ancient time, these objects are modeled as *ghost* objects. The semantics of *ghost* objects also does not impose physical constraints on the environment, but requires the visualizations of the object while performing its associated activity.

Fable and *myth* represent the non-existent versions of *spy* and *camouflage* objects or the hidden version of *ghost* and *mirage* objects. *Fable* is a non-existent, active, and invisible object, while *myth* is a non-existent, inactive, and invisible object. The semantics associated with *fable* and *myth* is that the environment and the observer are not sensitive to the presence of these objects and the observer cannot see them. For instance, if the observer of the archaeological site decides to explore the actual configuration of the environment, he or she can hide the *mirage* and *ghost* objects, transforming them into *myth* and *fable* objects, respectively. This kind of manipulation generates an environment where only existent objects are visible.

5. Operations over VR Objects

The status of VR objects can change over time. Some objects constantly change their status, others change it only a few times, and others have a particular status during their entire lifetime. The change from one status to another is accomplished through the set of semantic operations (i.e., *appear*, *disappear*, *activate*, *deactivate*, *kill*, and *resuscitate*). These operations can be arranged in three different groups. Operations of each group act only over a specific characteristic of the object (i.e., visibility, activity, and existence). Each group has exactly two operations where one operation is the inverse of the other. All operations are bijectives, thus their domain and range are equipotent. The domain of each operation is a subset of the set of VR objects' statuses (i.e., *actor*,

scenery, *spy*, *camouflage*, *ghost*, *mirage*, *fable*, and *myth*). The range of each operation is the complimentary subset of its respective semantic domain.

Visibility-related operations includes *appear* and its inverse, *disappear*. *Appear* and *disappear* model the transition of the visibility state of an object to visible or invisible, respectively. These operations direct the application to start or to stop rendering the graphical representation of the object (Figure 4).

| | |
|-----------------------------------|--------------------------------------|
| <i>appear(spy)=actor</i> | <i>disappear(actor)=spy</i> |
| <i>appear(camouflage)=scenery</i> | <i>disappear(scenery)=camouflage</i> |
| <i>appear(fable)=ghost</i> | <i>disappear(ghost)=fable</i> |
| <i>appear(myth)=mirage</i> | <i>disappear(mirage)=myth</i> |

Figure 4. Mappings of *appear* and *disappear* operations.

Activate and *deactivate* form the group of activity-related operations. *Activate* indicates that the object starts to perform its associated *activity*. After the *activate* operation the object becomes an active object, implying that the value of at least one of its attributes for which the observer can sense the variation will change at the next instant. The *deactivate* operation indicates that the object stops performing its associated *activity*. Figure 5 shows the result of these operations over their respective semantic domains.

| | |
|---------------------------------|-----------------------------------|
| <i>activate(scenery)=actor</i> | <i>deactivate(actor)=scenery</i> |
| <i>activate(camouflage)=spy</i> | <i>deactivate(spy)=camouflage</i> |
| <i>activate(mirage)=ghost</i> | <i>deactivate(ghost)=mirage</i> |
| <i>activate(myth)=fable</i> | <i>deactivate(fable)=myth</i> |

Figure 5. Mappings of *activate* and *deactivate* operations.

The existence-related operations are *kill* and *resuscitate*. *Kill* models the case where the object ceases to exist. *Resuscitate* indicates that an object comes to an existent state. Figure 6 shows the mappings of existence-related operations.

| | |
|------------------------------|-------------------------------------|
| <i>kill(actor)=ghost</i> | <i>resuscitate(ghost)=actor</i> |
| <i>kill(scenery)=mirage</i> | <i>resuscitate(mirage)=scenery</i> |
| <i>kill(spy)=fable</i> | <i>resuscitate(fable)=spy</i> |
| <i>kill(camouflage)=myth</i> | <i>resuscitate(myth)=camouflage</i> |

Figure 6. Mappings of the *kill* and *resuscitate* operations.

The domain and range of each operation limit possible transitions between different statuses. The combinations of these operations, however, can model all possible transitions among elements of the set of VR objects statuses. For instance, a sequence of operations (denoted by \circ) that turn a *myth* into an *actor* could be:

activate \circ *appear* \circ *resuscitate(myth)=actor*

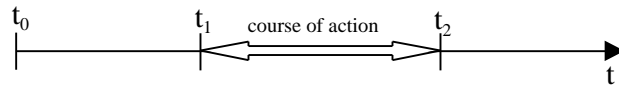
6. History of VR Objects

In order to keep track of a VR object's semantics along its lifetime we introduce *history*, an extension for the animation data model to represent an object's changing status over time. *History* models the evolution of the status of every VR object with respect to existence, visibility, and activity characteristics.

Each VR object has its own *history*. Each *history* has a set of operations that model the evolution of an object's status during its lifetime. *Create* is the first operation performed in the object's *history*. Its domain is the empty set and its range is an non-existent, invisible, and inactive object (i.e., *create* has no arguments, and always returns a *myth* object). It is a design decision to start the object's *history* with a *myth* status. If this is not the case, however, the initial status of the object can be changed through an appropriate combination of semantic operations performed immediately after the *Create* operation. Semantic operations (i.e., *activate*, *deactivate*, *appear*, *disappear*, *kill*, and *resuscitate*) are performed along the object's *history*, modeling the evolution of the object's status over time.

Figure 7 shows the graphical representation of the *course of actions* and *history* of an object. *Course of actions* models the quantitative evolution of the object's activity. If the object is a vehicle, for example, the *course of actions* indicates the variation of the car's position during the interval $[t_1, t_2]$. *Courses of actions* impose a constraint on the temporal attribute of activity-related operations (i.e., *activate* and *deactivate*). These operations occur in the object's *history* only when the object starts or finishes the performance of its associated behavior. *Activate* and *deactivate* model the qualitative evolution of the object's activity.

object's activity



object's history

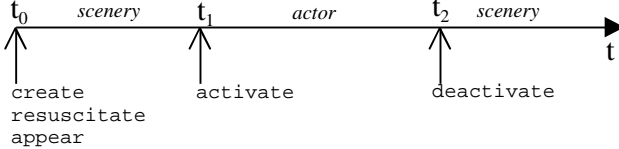


Figure 7. Graphical representation of the *history* and *course of actions* models of an object.

The object's *history* reveals the evolution of the objects' statuses. At the instant t_0 (i.e., the lower bound of the object's *history*) the operation *create* is performed generating a *myth* object. At the same time the operations *resuscitate* and *appear* change the status of the object to a *scenery* object. Semantic operations are ordered by their temporal attribute and performed in chronological order. If more than one operation occurs at the same time, however, the order of execution will be defined by the order

of inclusion of the operation in the model. At the instant t_1 , an *activate* operation changes the status of the object to an *actor*. This operation reflects the fact that the object has an associated activity (i.e., *course of action*) that starts at the instant t_1 (e.g., the car starts to move). At the instant t_2 , a *deactivate* operation transforms the object back into a *scenery* object, reflecting the end point of the *course of action* (e.g., the car stops its movement). The object remains as *scenery* indefinitely since there are no other operations to be performed in the model. For Example, from the instant t_2 on, the car can be seen parked somewhere in the virtual environment.

7. Manipulations of VR Objects

The exploration of spatio-temporal VR environments needs to be more than a mere reproduction of real dynamic environments. Indeed, observers need to have a rich set of operations to manipulate the objects and their behaviors, creating their own view of the environment, gaining insight, and discovering relationships between dynamic phenomena.

Some important manipulations of VR objects involve the redefinition of the time when the object performs its associated activity. Phenomena that have occurred at different times can be manipulated and observed at the same time, facilitating the comparison of their behavior. Allowing observers to interfere with the original flow of the objects' dynamics, however, requires the modification of an objects' *history* in an automatic and consistent way.

Consider, for instance, a scenario where an analyst explores a virtual environment involving a storm and a ship traveling from Boston, Massachusetts to Portland, Maine. The storm system developed a path somewhere between Boston and Portland. Figure 8 depicts the graphical representation of the storm and ship activities (i.e., their associated *courses of actions*) and the evolution of their statuses (i.e., their *histories*). The exact path and size of the storm and the path of the ship are modeled by their respective *courses of actions*.

The *history* of the storm reveals that the object starts as a *myth* and remains with this status until the instant t_1 , when a sequence of operations (i.e., *activate*, *appear*, and *resuscitate*) changes its status to an *actor*. The semantics associated with an *actor* object indicates that the object exists, is visible, and is performing the activity modeled by its *course of actions* (e.g., the storm is changing its position and size). The storm remains as an *actor* until the instant t_3 , when another sequence of operations (i.e., *deactivate*, *kill*, and *disappear*) transform it back into a *myth*, modeling the end of the storm.

The *history* of the ship shows that the object starts as a *myth*, implying that the environment and the observer are not sensitive to the presence of the ship and the observer cannot see it. The status of the ship as a *myth* at the start of its *history* can be interpreted as a lack of knowledge about

the ship during this period. The ship retains the *myth* status until the instant t_2 , when a sequence of operations (i.e., *resuscitate* and *appear*) transform it into a *scenery* object. The semantics associated with a *scenery* object is that the object can be seen and the environment is sensitive to the presence of the object. The position of the ship is a dock at the port of Boston. This information is available as the initial value of position's attribute of the ship. The ship remains docked at Boston until the instant t_4 , when the occurrence of an *activate* operation transforms its status into an *actor*, indicating that the ship starts its trip to Portland. A *deactivate* operation at the instant t_5 denotes that the ship finishes its trip and becomes a *scenery* object at the Portland's port (i.e., the ship no longer is performing an associated activity). The ship keeps the *myth* status indefinitely.

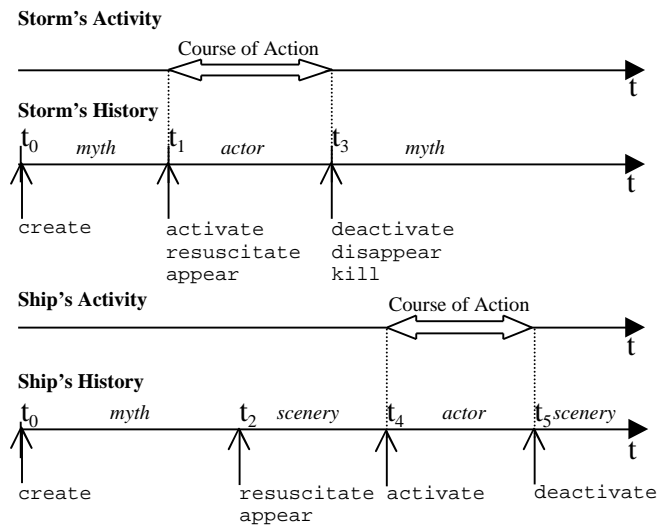


Figure 8. Graphical representation of the *course of actions* and *history* of the storm and ship.

The analysis of the storm and ship *histories* discloses that the ship was safely anchored during the occurrence of the storm. However, an observer can manipulate the original flow of the object behavior and explore the virtual environment with hypothetical configurations. Consider, for instance, a situation where the observer makes the ship start its trip at the same instant of the formation of the storm (Figure 9). This configuration can be accomplished with an operation that changes the start point of the ship's *course of actions*. In such a manipulation the history model of the storm is not affected, but the *history* of the ship needs to be updated accordingly to reflect the new spatio-temporal configuration of the environment.

The modified *history* of the ship shows that the ship starts as a *myth* and keeps its status until the instant of the formation of the storm. At this instant the ship appears at the port of Boston and immediately starts its trip to Portland as a *ghost* object. The *ghost* semantics implies that the ship can be seen performing its associated activity, but its

presence does not interfere with the environment. The ship finishes its trip at the instant t_n , becoming a *mirage* at the port of Portland. The *mirage* semantics implies that the ship still can be seen and its presence does not interfere with the environment, but there is no activity associated with the ship. The *ghost* and *mirage* semantics associated with the ship reflects the fact that the ship is not supposed to be doing its trip or to be docked at the Portland's port during these periods. The ship remains as a *mirage* until the instant t_5 when it becomes a scenery object at the Portland's port. After the instant t_5 , the ship continues with its original history.

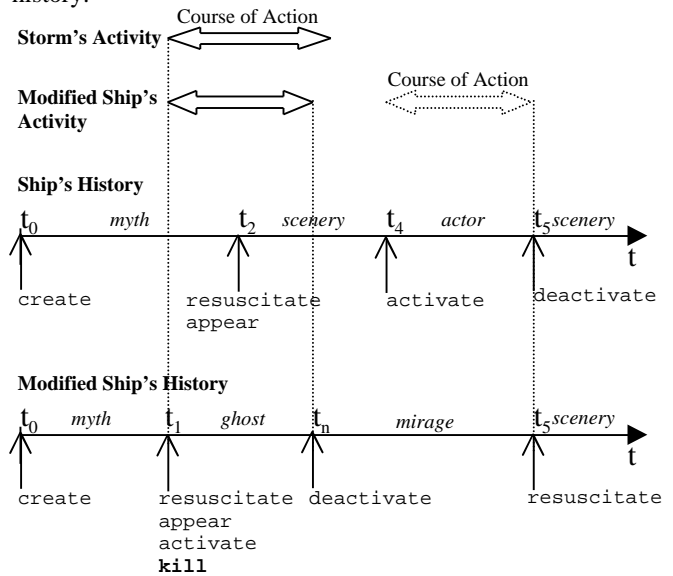


Figure 9. Manipulation of the ship *activity* and the modified version of the ship *history*.

The modified *history* of the ship illustrates the rationale used to update the evolution of VR objects' statuses when its activity is manipulated. In this case, the course of action of the ship is translated to a new position in the temporal domain. The translation of the *course of action* implies the reposition of the *activate* and *deactivate* operations in the object's *history*. The temporal displacement between these two operations is kept the same as the original configuration (i.e., $t_n - t_1 = t_5 - t_4$). This constraint guarantees the original pace of the ship's trip. All operations occurring in the ship's *history* between the original and the new position of the *activate* operation are positioned at the same time (i.e., at the new position of the *activate* operation) forming a long sequence of operations. The order of the operations in the sequence reflects the order that they occur in the original model. If the result of the sequence operation is an element of the existence set, a *kill* operation is added at the end of the sequence. The *kill* operation represents the fact that the object's activity was manipulated. Finally, a *resuscitate* operation is introduced at the original position of the *deactivate* operation, restoring the original semantics of the object.

8. Conclusions and Future Work

VR technology has been shown to be a prominent and promising interface for visualization and exploration of information in a multi-dimensional, multi-sensorial, and immersive fashion. In the context of GISs, in particular, the exploration of spatio-temporal VR environments needs to be more than a mere reproduction of real dynamic environments. Observers need a rich set of operations to manipulate the objects and their behaviors, so they can create their own view of the environment, thus gaining insight and discovering relationships between dynamic phenomena. Semantics of VR objects are valuable information in the process of the virtual exploration of an environment and play a significant role in interactions between observers and VR environments, as well as in interactions among VR objects themselves.

This paper introduces a model to represent VR objects' semantics. Based on this model a new classification of VR objects is presented, and the semantics associated with each class of object is described. A set of operations, acting upon individual characteristics of the object, models the object's semantics. In order to capture an object's changing semantics we introduce *history*, an extension for VR data models to represent the evolution of the semantics of VR objects over time. This paper outlines the rationale used to modify the evolution of an object's semantics during the manipulation of the object behavior. Some important manipulations of VR objects involve the redefinition of the time when the object performs its associated activity. In the future, we plan to consider the impact that other kinds of manipulations will have on the *history* of an object, for example, when an observer stops the activity of a single object while other objects continue to be active.

VR data models can also be extended to support causality. Causal relations are also a valuable piece of information that can be used to compose a new version of the virtual environment. It is important for an observer to isolate objects affected by a specific change in the configuration of the environment or object actions that lie in a cause-and-effect chain. This work will be further extended with new rationales to represent the modified version of the object's *history* based on causal relations.

Acknowledgments

This work was partially supported by the National Institute of Environmental Health Sciences, NIH, under grant number 1 R 01 ES09816-01; the National Imagery and Mapping Agency under grant numbers NMA202-97-1-1023 and NMA201-00-1-2009, and NMA201-01-1-2003; by the National Science Foundation under grant numbers IIS-9613646, IIS-9970123, and EPS-9983432; and by the Brazilian Research Council, CNPq, under grant number 200020/00-5.

References

- [1] J. Allen and G. Ferguson, *Actions and Events in Interval Temporal Logic*, in *Spatial Temporal Reasoning*. Kluwer Academic. pp. 205-245. 1997.
- [2] C. Blok, B. Kobben, T. Cheng, and A. Kuterama. Visualization of Relationships Between Spatial Patterns in Time by Cartographic Animation. *Cartography and Geographical Information Science*. **26**(2):139-151,1999.
- [3] A. van Dam, A. Forsberg, D. Laidlaw, J. LaViola, and R. Simpsons. Immersive VR for Scientific Visualization: A Progress Report. *IEEE Computer Graphics and Applications*. **20**(6):26-52,2000.
- [4] M. Egenhofer and K. Hornsby, *Spatio-Temporal Reasoning about Identity and Visibility*, in *Integrating Spatial and Temporal Databases*. 1998: Schloss Dagstuhl, Germany, November 1998. p. (Presentation)
- [5] M. Green and S. Halliday. A Geometric Modeling and Animation System for Virtual Reality. *Communications of the ACM*. **39**(5):46-53,1996.
- [6] K. Hornsby and M. Egenhofer. Identity-Based Change: A Foundation for Spatio-Temporal Knowledge Representation. *International Journal of Geographical Information Science*. **14**(3):207-224,2000.
- [7] R. Jacobson, *Virtual Worlds, Inside and Out*, in *Cognitive and Linguistic Aspects of Geographic Space*. Kluwer Academics, Netherlands. pp. 507-514. 1991.
- [8] M. Kraak, G. Smets, and P. Sidjanin, *Virtual Reality, The New 3-D Interface for Geographical Information Systems*, in *Spatial Multimedia and Virtual Reality*. Taylor and Francis, London. pp. 131-136. 1999.
- [9] H. Luttermann and M. Grauer. VRML History: Storing And Browsing Temporal 3D-Worlds. *ACM Fourth Symposium on VRML*. Paderborn, Germany. pp. 150-163, 1999.
- [10] D. Medak. Lifestyles – an Algebraic Approach to Change in Identity. *Spatio-temporal Database Management, International Workshop STDBM'99*. Edinburgh, Scotland. pp. 19-38, 1999.
- [11] S. Oviatt and P. Cohen. Multimodal Interfaces that Process What Comes Naturally. *Communications of the ACM*. **43**(3):45-53,2000.
- [12] J. Raper, *Multidimensional Geographic Information*. Taylor & Francis, London, 2000.
- [13] T. Sheridan. Interaction, Imagination, and Immersion: Some Research Needs. *ACM Symposium on Virtual Reality Software and Technology VRST'00*. Seoul, Korea. pp. 1-7, 2000.
- [14] P. Strauss and R. Carey. An Object-Oriented Graphics Toolkit. *Computer Graphics*. **26**(2):341-349,1992.
- [15] E. Verbree, G. Maren, R. Germs, F. Jansen, and M. Kraak. Interaction in Virtual World Views - Linking 3D GIS with VR. *International Journal of Geographical Information Science*. **13**(4):385-396,1999.