Semantic Virtual Environments

Karsten A. Otto
Freie Universität Berlin
Institut für Informatik
Takustraße 9, 14195 Berlin, Germany
otto@inf.fu-berlin.de

ABSTRACT

Today's Virtual Environment (VE) systems share a number of issues with the HTML-based World Wide Web. Their content is usually designed for presentation to humans, and thus is not suitable for machine access. This is complicated by the large number of different data models and network protocols in use. Accordingly, it is difficult to develop VE software, such as agents, services, and tools.

In this paper we adopt the Semantic Web idea to the field of virtual environments. Using the Resource Description Framework (RDF) we establish a machine-understandable abstraction of existing VE systems — the Semantic Virtual Environments (SVE). On this basis it is possible to develop system-independent software, which could even operate across VE system boundaries.

Categories and Subject Descriptors: I.2.4 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods – *semantic networks*

General Terms: Design, Languages

Keywords: Semantic Web, virtual environments, integration, framework, components

1. INTRODUCTION

Virtual Environments (VE) are used today for a number of application domains, such as education, product presentation, and entertainment. Each VE has particular requirements regarding functionality and performance. To fulfill these requirements in the best possible manner, VE systems utilize a variety of data models and network protocols.

This heterogeneity makes it difficult to develop new VE software for such an environment. Each support tool, service, and agent must be specifically tailored to fit its underlying VE system. It is thus generally not possible to re-use the software for another environment, even if it belongs to the same application domain. The problem is that the software needs to work on the level of environment semantics, which is however hidden behind the VE system interface.

A similar problem exists today in the HTML-based World Wide Web. The Resource Description Framework (RDF) was developed to expose the hidden semantics of Web pages, in order to facilitate access for Semantic Web software. Considering the similarity, we apply the same idea to expose the semantics of an existing virtual environment, in the form of a *Semantic Virtual Environment (SVE)*. For this purpose, we need to describe both the world model and communication channels of its underlying VE system in a machine-understandable fashion.

WWW 2005, May 10–14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

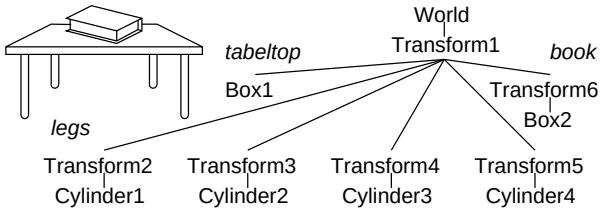

Figure 1: World Rendering and Scene Tree

2. WORLD MODEL

The world model represents the contents of a virtual environment, typically with lots of detail for realistic graphical rendering. Figure 1 shows an example. A user can easily interpret the rendering via the human senses, to form a mental model of its meaning: A book lying on a table.

In contrast, SVE software lacks the human preceptive abilities, and will likely find it difficult to derive task-relevant information from the pixels of a rendering. It can only work with the scene tree itself, which states that four cylinders and a box are spatially located relative to another box. It is not clear that the combination of `Cylinder1-4` and `Box1` is supposed to represent a table, or that `Box2` (the book) is actually a separate object. We must describe all this explicitly in the *SVE description*, which plays the same role for SVE software as the mental model does for a human user.

Within the SVE description, we model the SVE itself and its contents as RDF resources, deriving URIs from internal identifiers (names or numbers) of the world model if possible. Figure 2 shows the result. The environment resource `#myenv` serves as starting point for describing the SVE. In particular, its `sve:channel` property refers to the associated communication channel `#mychan`. This in turn specifies the address information via `sve:host` and `sve:port`, as well as the protocol via `rdf:type` and an appropriate RDFS class.

The SVE resource also provides the interesting possibility of linking environments together. A link may simply be an `rdfs:seeAlso` property, or a more complex relationship describing a detailed link role and relative spatial arrangement. System-independent SVE software could use such links to navigate a web of related environments.

Finally, the SVE resource serves as starting point for en-

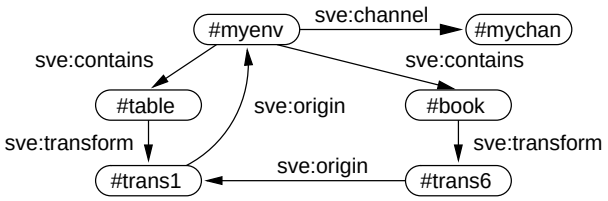

Figure 2: SVE Description (simplified)

vironment content discovery. It uses multiple `sve:contains` properties for this purpose; each refers to an RDF resource representing a content object. Other (derived) properties may also indicate relationships of a more specific nature.

The description of the content objects is an important aspect of the SVE description, since this enables SVE software to find task-relevant information. We can derive part of it from the existing world model. In particular, the transformation (position, orientation, size) of an object is of equal importance to both rendering and SVE software. We then use the extensibility of RDF to add more properties describing the object's meaning. For example, we may annotate `#mybook` with Dublin Core vocabulary terms. However, the most important aspect of an object is its *type*. Human users classify objects according to the details of their representation. We model this information explicitly using `rdf:type` with RDFS classes of an appropriate application domain.

Note that modeling details and vocabularies may differ among environments. We assume that established data integration techniques (inference rules, upper ontologies) will enable SVE software to compensate for this, as long as there is a sufficient degree of shared semantics.

3. ENVIRONMENT DYNAMICS

While the world model covers all static aspects of a virtual environment, its channels handle the dynamic aspects. This includes participation management, object movement, text chatting, and other domain specific interactions. VE client programs utilize the environment's associated channels for this purpose, exchanging corresponding protocol messages.

To truly participate in a virtual environment, SVE software must handle these dynamic aspects as well. However, processing the raw protocol messages is not an option, since they are usually specific to a particular VE system. Instead, the SVE software requires a *channel adapter*, which can extract all relevant information from the channel's messages and convert it into system-independent *SVE events*.

We model each SVE event as a small RDF graph of its own, with a special RDF resource to denote the event itself. We then describe the event in a similar manner as content objects, by populating the event resource with properties from an appropriate RDFS vocabulary. In particular, we again use `rdf:type` as the primary tool to express the event's meaning. Modelling events in this way makes it very easy to augment them with additional information during processing, or to aggregate and convert them into new events. This is important, since raw events derived from protocol messages are usually rather simple in nature, but occur very frequently. SVE software likely needs to convert these simple events into fewer and more meaningful ones.

Figure 3 shows an example conversion chain for object movement events. Initially, the channel adapter provides

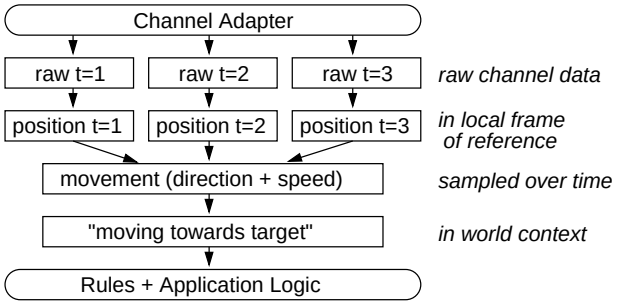

Figure 3: Conversion Chain of SVE Events

a sequence of *raw events*, converted directly from protocol messages. In this example, each specifies the position of an object at a given time, in the environment's native coordinate system. As a first processing step, SVE software may want to convert these positions to an internal frame of reference, especially if it interacts with a number of linked environments using different coordinate systems. The next processing steps may then handle these *position events* independent of their origin. Furthermore, few objects "teleport" to positions at random; most move in a more or less continuous manner. By sampling the position events over time, it is possible to discover such continuous movement, and derive additional information about direction and speed of the object. SVE software can thus convert a few position events into a new *movement event* that describes this movement. With the given direction and speed, SVE software can also predict the path of the object in the environment. It may use the prediction for collision detection with the content objects specified in the SVE description. On this basis the SVE software can convert the movement event into a *contextual event*, which expresses the fact that the moving object is headed towards a given world object. From this point on event conversion is less mathematical and more semantical in nature. In particular, SVE software may need a data integration step again (see section 2), before it can finally process the contextual event in its application logic.

4. IMPLEMENTATION

Clearly, it is much easier to develop application logic on the basis of high-level contextual events, instead of having to work with low-level raw events. Also, most conversion steps are largely independent of the application logic; for that matter, so is the channel adapter. It makes sense to keep this functionality separate, in the form of independent *SVE software components*. These components can easily be re-used to drive other application logic, speeding up SVE software development. They also facilitate the adaptation of some given application logic to a new VE system.

We are currently developing the SEVEN platform, using the Java-based OSGi framework to realize the SVE components. To date it supports adapters for a few selected VE systems (Cube and VOS), which were chosen for their different application domains and associated network protocols. In addition to these, we plan to analyze more of the existing virtual environment systems, and integrate them into SEVEN by applying our SVE techniques.

For more information, please visit our Web site:
`http://www.inf.fu-berlin.de/~otto/sve/`