Defining an Ontology on a Virtual World

Two current major trends in internet technology are the use of virtual worlds in modeling and problem solving, and the use of ontologies to facilitate knowledge sharing. Given that we can expect the future internet to be a seamless integration of traditional websites and virtual worlds, we can assume that whatever benefits are to be gained from incorporating ontologies with a traditional website will also be gained by incorporating ontologies with a virtual world. However, the use of an ontology in a virtual world poses special problems not faced by a traditional website. Our project consists of using the Protégé ontology platform and the OpenSimulator virtual worlds server to develop a proof of concept of such an incorporation. These two platforms provide us with three advantages: Protégé includes a well-defined plug-in system, OpenSimulator is an extensible implementation of a virtual world compatible with the popular Second Life, and this should allow us to eventually incorporate previously defined medical ontologies with the hospital modeling projects currently utilizing Second Life.

The basic abstraction we will be using is the definition of an ontology over a data set. In a traditional website the data available on the site is encoded in the text of the html, or, if automatically generated from a database back end or scripts, is at least encoded in html at the time of display. Since most of this information is already text, it is relatively simple to develop a formal ontology that can be applied to that information. This follows from the fact that formal ontologies, formal logic of any sort, is defined in syntactic terms. A useful metaphor here might be that of a program written in some programming language, i.e., exists only at the syntactic level, versus a program that exists in some runnable form, i.e., exists at a semantic level. If the data is a set of syntactically correct programs for a given language, a compiler would be something like the definition of an ontology over that data. Similarly, a relational database management system could be construed as a sort of ontology defined
over a set of SQL statements. That is, the definition of an ontology over a set of data is the creation of a formalism that allows raw textual data to be manipulated by a computer in ways useful to an end user. The most frequent of these uses is to define a series of axioms, the basic terms of which correspond to objects or relations in a particular domain of knowledge, and on which automated reasoners can function so as to supply us with information not made explicit in the axioms. In a virtual world, in Second Life and OpenSimulator in particular, the situation is complicated both by the fact that not all the data is defined in a textual manner, and by the fact that user interaction with a virtual world is a far more dynamic process than it is in traditional websites. These two aspects are related in that for virtual worlds useful data is implicit in the behaviors within the world, both creator and user defined. The goal, then, of defining an ontology over the data of a virtual world involves associating the categories and relations of an ontology with the objects and interactions in the virtual world.

In OpenSimulator the basic units of interaction are going to be collections of prims and scripts associated with those prims. The passing of certain sorts of messages between these collections could correspond to relations in the ontology. We can operate on the assumption that any given virtual world has been created for the purpose of expressing knowledge of some domain, and so the definition of the ontology itself is really of concern only to the creators of the virtual world. However, this does not mean that the utilization of the ontology is only of use to the creator. This leads us to define several sub projects. Firstly, we need to define a method by which creator defined objects in OpenSimulator can be connected with elements of the ontology as instances of classes. This would not extend the functionality of the virtual world itself, but would allow observers of behavior in the virtual world to extract semantically meaningful data and draw inferences from it using the tools included in Protégé. Since both OpenSimulator and Protégé store information in a relational database, it should be possible to add appropriate columns to the table definitions used by OpenSimulator, as well as any additional
tables needed, to store the information used by Protége. This would allow us to define objects in the virtual world as instances of classes in our ontology in a way that is completely transparent to OpenSimulator itself. Alternatively, we could define scripts that hold data meaningful to our ontology and place those scripts on the appropriate objects. We could then have the Protége system read that data for its own use. Although this approach would be easier to implement, it would probably result in more difficult, if not impossible, additional functionality down the road. However, this approach would allow us a looser coupling of the systems which may be sufficient for some uses, and wouldn't introduce any potential incompatibility into the OpenSimulator architecture. The next sub project involves exposing elements of the Protége API to OpenSimulator itself so that semantically meaningful data could be generated by scripts in the virtual world and acted on accordingly. This too could be done in a more or less tightly coupled way. More important than the general Protége API, and in fact the general API might not be appropriate, would be to define an API that is meaningful in relation to a specific ontology, allowing the behaviors and relations encoded by the ontology to be more consistently scripted in the virtual world. A further sub project, although one not essential at this phase, would be a tool to translate from a Protége ontology to a library or module that could be loaded into OpenSimulator to enable this sort of semantically aware scripting. Once aspects of the ontology can be utilized from within scripts, it is a relatively trivial matter to allow end users to access the same functionality in their scripts; it is merely a matter of appropriate permissions. The last step, which we will not be concerned with at this stage, would be to allow for the creation or extension of the ontologies by users from within the domain of the virtual world.

For this semester, our initial goal is to demonstrate a modification of the database definitions that can be accessed by either OpenSimulator or Protége, and our final goal is to define a specification and begin implementing an API to call Protége functionality from within OpenSimulator scripts.