A UNIFIED HUMAN INTERACTION-BASED THEORY AND FRAMEWORK FOR SIMULATION MODELING AND VISUALIZATION DESIGN

By

ZACH EZZELL

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2012

To Mom and Dad

ACKNOWLEDGMENTS

I thank my research advisor and supervisory committee chair, Dr. Paul Fishwick for teaching me how to conduct research and giving me the freedom and time to truly explore ideas during our numerous projects. I would not have pursued a Ph.D. if not for my experience working with Dr. Fishwick as an undergraduate. I thank my supervisory committee members, Dr. Douglas Dankel, Dr. Jeffrey Ho, Dr. Samsun Lampotang and Dr. Benjamin Lok for their time, ideas and support. I also thank my collaborators at the University of Central Florida, Dr. Juan Cendan and Dr. Jonathan Kibble, for their expert feedback from a medical education perspective.

I thank my Mother, Father, and two sisters for their love and support. My family instilled me with an intellectual curiosity at a young age and has always provided much-needed emotional support, especially during the ups and downs of graduate school. I would also like to thank my loving girlfriend, Molly for her endless support and kindness. Finally, I would like express further gratitude towards Molly and my Father for helping me polish this document.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF OBJECTS

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

A UNIFIED HUMAN INTERACTION-BASED THEORY AND FRAMEWORK FOR SIMULATION MODELING AND VISUALIZATION DESIGN

By

Zach Ezzell

December 2012

Chair: Paul A. Fishwick
Major: Computer Engineering

Visualizing dynamic phenomena by leveraging computer graphics and simulation is of increasing interest to society. In the classroom, such dynamic visual modules can motivate students and give them a holistic view of a given event. Industry practitioners and researchers also leverage such modules to aid in comprehending simulation results and communicating ideas. Current methods used to construct these customized views, however, are expensive. Content experts must employ engineers to code the mathematical model that defines the dynamic behavior of the scenario. Engineers must then somehow visualize the output of the model. These tasks are performed with engineering software tools or just pure computer programming. This work is towards defining an interface and interaction model that will compress this engineering overhead, and thus narrow the user-interface gap between mathematical modeling and 3D interactive visualization design and consumption.

Interaction within the proposed interface centers on a visualization of a 3D semantic network, or ontology, in which domain concepts are represented by nodes and an edge between two nodes represents a semantic relationship between two concepts in the domain space. To create custom modules, visual attributes can be added to

nodes within the semantic graph that point to graphical resources, or define dynamic behavior. In this dissertation it is demonstrated that through interactions with this semantic graph, a designer can sculpt and annotate simulation models into meaningful interactive, animated visualizations that integrate 2D and 3D information to relay the dynamics of an event to an observer. The theoretical interface and interaction techniques will be presented along with a case study visualization constructed using a prototypical implementation of the theory.

CHAPTER 1
INTRODUCTION

Content experts leverage visualizations of simulated scenarios during a variety of academic and industrial endeavors. For example, consider visualizations of plant growth (Tang et al., 2011) and human lung dynamics (Santhanam et al., 2008). Practitioners create visualizations to analyze and tune simulation models and communicate ideas to invested third parties (e.g., management, students). Visualizations of these types are often based on an underlying mathematical model that conforms to the modeling rules within a paradigm (e.g., the systems dynamics paradigm in ecology, the compartmental modeling paradigm in physiology). The creation of these customized visualization modules can be quite resource-intensive. During creation, the content expert will create an abstract simulation model using the domain-modeling practices for which they are familiar. In addition to defining dynamic behavior, the expert will often create a primitive pre-visualization or sketch of the envisioned graphics and user-interaction. Engineers and computer graphics specialists will then realize this design with a variety of tools and approaches. The engineer will need to map the abstract model into an engineering software paradigm (e.g., using Simulink or LabVIEW) to create an executable simulation. After the simulation model is made executable, a graphics specialist will link the simulation output to visualization parameters (e.g., using scripts in a 3D modeling package or a graphics programming API) to create the desired animation when the simulation executes. Someone will also be responsible for encoding the desired interaction including the ability to tweak simulation parameters in the interface and see the results in real-time.

Attempts have been made to compress the visualization construction pipeline through the creation of various simulation and visualization frameworks. These frameworks, however, are lacking in options for visual output, require engineering training, or operate across multiple user-interfaces that employ different interface metaphors. For example, in Simulink (2012), an industry leading model-based simulation platform, to link a 3D visualization with an executable simulation model, a user must (1) encode the simulation model within the Simulink graphical simulation modeling window, (2) build the virtual 3D scene in VRML editor window and (3) use a third connective interface to link visualization parameters to simulation variables. An abstract sketch of this interaction model is shown in Figure 1-1.

The motivation behind this work is to define an interaction model that shrinks the human-computer interface gap between simulation model-building and visualization construction activities while remaining general enough to be applied across different content modeling paradigms. Ideally, such an interface would afford the content expert, who is not necessarily an engineer, the ability to sculpt interactive and expressive 3D visualizations with minimal technical training. The interaction model proposed is structured around a semantic network, or ontology, which is graph-based encoding of knowledge understandable by a machine and a human (Berners-Lee et al., 2001). It will be demonstrated how the ontological structure can be leveraged to meet user interface and interaction requirements of both the design and consumption of executable simulation models and their corresponding visualizations.

This dissertation will present background in several applicable areas, the theoretical interface model and visualization design methodology, and a case-study built

using a prototypical implementation of the theory. The remainder of this chapter will explain the three foundational models that are synthesized in the proposed interaction theory, articulate the research question posed, and discuss the potential broader impact of this work.

1.1 To "Model"

The term "model" may be interpreted differently depending upon one's background. The definition of model most applicable to this research is "an economical physical surrogate so that we may employ different notations and metaphors as a way to increase our understanding of the phenomena being modeled" (Fishwick, 1995). Three core model types are encountered when constructing dynamic 3D visualizations: ontological, dynamic, and graphical. To fully understand the interface synthesis this work aims to achieve, it is important these three base model types are well understood.

1.1.1 Ontological Models

The concept of ontology is phrased by Gruber (1993) as a "specification of a conceptualization." In a given field, an underlying set of semantics is always present. For example, the field of psychology contains the sub-field cognitive psychology, which contains the paradigm information processing, which was first described by the person Donald Broadbent in the year 1958. With proper attention to detail, ontological models can be constructed and used to formalize these concepts and semantic relationships into a format which is understandable by humans and machines (Woods, 1975). The ontological spectrum, shown in Figure 1-2, was defined by Daconta et al. (2003) to highlight the relationship between the expressive power of a knowledge model and the cost of creating and maintaining it.

An ontology can be represented mathematically as a graph, with nodes representing concepts in a domain and edges representing relationships between concepts. Figure 1-3 shows a visualization of a portion of the foundational model of human anatomy ontology (FMHAO) (Rosse and Mejino, 2007). The portion shown pertains to the human heart, and the concept *Human body* serves as the root of the ontology. The Human Body *has a* Cardiovascular System, which, in turn, *has a* Heart. Notice also the relationships denoting dynamics (i.e., blood circulation), such as *has arterial supply*. An ontology is sometimes referred to as a "semantic network" or "semantic web," a term popularized by Berners-Lee et al. in 2001.

1.1.2 Dynamic Models

Dynamic models are used to recreate or further understand the dynamics of phenomena. Due to their abstract nature, they can also serve as blueprints for the design of simulation and visualization applications. The compartmental model is a specific type of a dynamic model and will be used in the exploration of this thesis. Compartmental models represent systems which can be abstracted into interconnected compartments containing some substance (e.g., fluid). The levels of substance at each compartment are typically governed by a set of differential equations. For instance, Beneken (1965) created a compartmental model for cardiovascular physiology. The model, shown in Figure 1-4, is composed of compartments (e.g., left atrium, pulmonary arteries) which are typically drawn out as circles. Directed edges, drawn as lines with arrows, connect the compartments and represent the flow of blood. This physiological model is explored further in Chapter 4.

1.1.3 Graphical Models

A graphical model is a machine understandable abstraction of an entity's visual attributes. Graphical models are used in a variety of applications, from visual effects in film to computer-aided design in engineering. Many tools exist to create and edit this model type, such as the 3D modeling package Blender (Blender3D, 2010). Graphical models consist of material information (e.g., color, reflectivity) and a geometric model composed of vertices, edges and faces. The information contained within a graphical model should be sufficient to create a render (i.e., a 2D image created using the model as input) from a given perspective. Three representations of graphical models are explored in Figure 1-5.

1.2 Thesis Statement

Thesis statement: A three-dimensional visualization of the domain-ontology can serve as a central user interface structure to connect simulation modeling and visualization construction activities and allows domain-specific semantics to guide the interactive modeling process.

I believe exploring the interface defined in this thesis will ultimately result in new interaction techniques that will allow experts to leverage their preexisting domain modeling knowledge to create interactive 3D visualizations of dynamic processes. The resulting visualizations would synthesize ontological models, dynamic models, and graphical models. These visualizations have many uses, including the analysis of simulation results and education. The "domain-specific semantics" refer to the modeling semantics used by practitioners in a given domain. Modeling semantics include domain specific terminology and the graphical properties of visual simulation models (Zeigler et al., 2000). To support this thesis statement, an ontology-based

interaction theory will be defined and this theory will be situated within a framework that defines a complete simulation-modeling and visualization construction tool.

1.3 Contributions

This dissertation provides two primary contributions to the field of modeling and simulation. Both contributions pertain to the visualization of simulation model output.

Contribution 1: The definition of a domain-ontology-based interaction model to connect simulation modeling to visualization construction activities. Current simulation modeling tools allow users to build visualizations controlled to varying degrees by simulation output. However, visualization design features in these tools are tacked-on in separate interface modalities and require custom coding for any advanced visual output (e.g., mesh deformations, particle systems). There is current work on creating advanced visualization design systems with "easy-to-use" interfaces, but this work is focused within the InfoVis community and is based on data, not simulation models, and often does not utilize 3D graphics. The ontology-based interaction model and framework presented this dissertation: (1) serves as a first step towards unifying simulation modeling and 3D visualization construction at the graphical user interface level, and (2) leverages semantics familiar to the arbitrary domain practitioner so that simulation models will not have to be translated to the general engineering modeling paradigm used in current tools and (3) in contrast to current applications of ontologies, demonstrates how the 3D visualized ontologic structure can serve as a virtual instrument to guide user interactions.

Contribution 2: The definition of the technical framework required to implement the interaction model. The interaction theory from Contribution 1 is an essential requirement for integrated simulation and visualization tools, but it is not enough on its

own to build the envisioned system. The system can only be built after the interaction model is placed within a complete framework. A theoretical framework is described in this dissertation and implementation details, surfaced through the coding of a case study, inform the requirements of the framework.

1.4 Broader Impact

If the framework outlined in this dissertation is employed, the broadest potential benefit is the creation of general user-interface that affords simulation and visualization construction and consumption activities to practitioners in various domains, and makes the construction process more efficient for engineers that use current methods (see Section 2.4). This applies to any domain in which 3D visualizations are leveraged. I believe the particular domain of education, however, could benefit in specific ways that are detailed in the remainder of this section.

1.4.1 Education

There are benefits in using 3D visualizations to help educate someone about a spatial arrangement (e.g., anatomy) or process (e.g., the pumping of oxygen through the body). For studies on how 3D visualizations can be beneficial in an educational setting, see the work by Prinz et al., (2010); Beermann et al., (2010); Quarles et al., (2008); or Silén et al., (2008). Collectively, these studies demonstrate the breadth of potential visualization applications in education. There are several educational concepts I would like to explore further, and in doing so highlight how the ontology-centered framework could provide some foundational technology to support these concepts. The educational concepts are systems thinking, constructivism, and scaffolding. A high-level description of how the proposed theoretical framework could

be leveraged to support these concepts is provided in the following sections along with an overview of the respective concept.

1.4.1.1 Systems Thinking

The abstract models that drive the visualizations may be leveraged as an educational affordance. This is in-line with systems thinking, a concept that has experienced a recent proliferation in education. Systems thinking is the practice of viewing a system as a set of components and interrelations which influence the system's behavior as a whole (Weinberg, 2001). Courses on system modeling are becoming part of curricula in fields such as ecology (Jørgensen et al., 2007) and biology (Klipp et al., 2009). In physiology, an area used in a case study for this research, there is a push to emphasize general models when teaching instead of presenting topics as "a series of unrelated phenomena" (Modell, 2000).

Different ways in which to emphasize models in the physiology class room have been explored. One example is an educational software package created by Rothe and Gersting (2002) which contains a minimal-complexity compartmental model for cardiovascular circulation. The model drives a simulation and is visualized with the simulation output for the student's edification. While there were issues with the usability (some of which have been addressed in revisions), 75% of first-year medical students who used the package reported that their understanding of circulation was improved. Another example is a physical respiration model created by Kuebler et al. (2007) for use in a physiology lab course. Of the students that participated in the course, 70% reported the interactive teaching afforded by the model increased their understanding of respiratory mechanics.

The proposed ontology-centered framework could enable new educational software that falls within the systems thinking paradigm. This could be achieved by creating 3D visualizations that are co-located with the abstract dynamic models that drive them. In the proposed framework, these dynamic models are encapsulated within the visualized ontology because they are utilized in the design phase and are, therefore, readily available to be shown in any resulting visualization module. An interactive 3D visualization with this characteristic is presented as a case study in Chapter 4.

1.4.1.2 Constructivism

Others have integrated a model-based approach when teaching by employing the constructionism learning paradigm. Constructionism is an extension of constructivism, a theory that states we learn by constructing knowledge models. A core principle of constructionism is that the construction of these knowledge models may be aptly facilitated by building tangible things in the real-world (Papert and Harel, 1991). Aldridge (2009) applied this theory when teaching a course on the physiology of congenital heart defects by having his students build physical models of defects out of materials such as Play-Dough, cardboard and Styrofoam. Students often claimed the model-building exercise increased their understanding of the pathology in question.

Because one intention of the proposed framework is to support visualization design based more on domain knowledge and less on engineering knowledge, students could potentially use the resulting tool to build animated visualizations of dynamic processes, such as the heart defects explored by Aldridge, and thus take part in a constructivist learning exercise.

1.4.1.3 Scaffolding

Scaffold learning is the process of providing educational guidance proportional to one's competence (Vygotsky, 1978). With interactive computer-based scaffold learning tools, supplemental information fades from the interface as the user's understanding of the subject grows (Jackson et al., 1998; Oliver and Herrington, 2003). Scaffolding is known by many in education and psychology to be an effective teaching strategy (Roehler and Cantlon, 1997). For quantitative study results on the efficacy of scaffolding see, for example, Toth (2000) or Chang et al. (2002).

In the proposed framework, content embedded in the visualized domain-ontology (e.g., classification information, diagrams) is the supplemental information which could fade away as the students understanding grows. This could be achieved by an educator setting the domain-ontology concepts and attributes that are to be displayed in a given module.

1.4.2 Healthcare

In healthcare, human error is one of the main causes of preventable deaths and injuries, as illuminated by the landmark report "To Err is Human" (Kohn et al., 1999). For example, a study showed that human error accounted for 82% of anesthesia-related preventable deaths (Cooper et al., 2002). Since the release of "To Err is Human," strides have been made to increase patient safety by reducing human error. However, there is still much need for improvement, especially in the area of training (Leape et al., 2009). One reason new training paradigms are being suggested is because some junior physicians feel they have had insufficient exposure to uncommon conditions and procedures before embarking on independent practice (Mason and Strike, 2003). Some suggest a new medical training paradigm should integrate competency-based

knowledge with clinical skill and expose trainees to more uncommon conditions (Rodriguez-Paz et al., 2008).

Consider a common source of human error: the administration of drugs to patients. Adverse drug effects (ADEs) are a primary reason patients die in a hospital (Bates et al., 1995). As a result of the intricate and complex nature of the cardiovascular system, cardiovascular drugs are known to cause the most ADEs (Ebbesen et al., 2001). Even more difficult for trainees to understand is the varying effects drugs and other medical interventions have on patients with abnormal circulatory physiology. For example, hypoplastic left heart syndrome (HLHS) is a congenital heart defect in which some chambers of the heart fail to develop. HLHS results in abnormalities of blood flow and oxygen delivery which have been identified by medical educators as difficult for learners to visualize and, therefore, understand. The ability to identify subtle differences between pathophysiological scenarios is critical because interventions that may be helpful in one scenario could be harmful in another (e.g., the administration of oxygen may be harmful to an infant with HLHS). New training paradigms that integrate simulation with traditional teaching methods will help trainee physicians avoid making harmful errors because of their lack of understanding of a rare and complex pathophysiology.

The ontology-centric methodology outlined in this dissertation could provide important technology to help address some of the needs of this newly proposed medical training paradigm. Starting with a visualized domain-ontology, such as one representing anatomy, medical educators should be able to construct a dynamic model, such as the Beneken (1965) model for cardiovascular physiology shown in Figure 1-4, and then create a 3D visualization of the physiological process by augmenting the visualized

ontology with attributes, such as 3D meshes. The 3D interaction within the resulting integrative visualizations could expose medical trainees to more uncommon conditions and integrate their textbook knowledge of anatomy and physiology with interactive 3D renderings, potentially increasing their spatial understanding of the anatomy and physiology in question (Beermann et al., 2010).

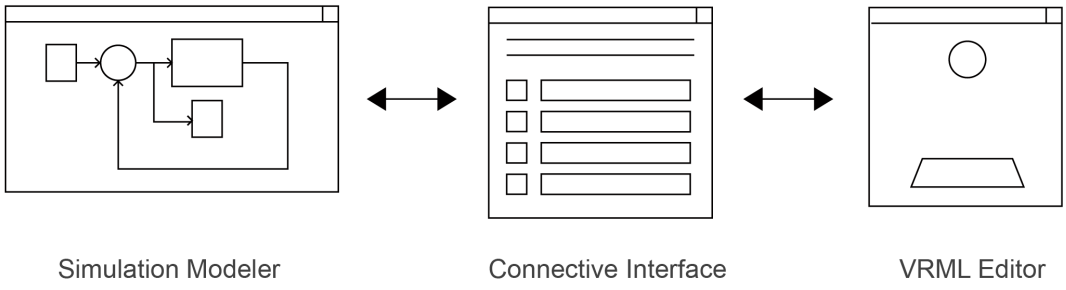


Figure 1-1. A sketch of the interface approach taken by Matlab's Simulink to link simulation variables to visualization parameters. The "connective" interface lists all possible visualization attributes and allows the user to choose which parameters can be influenced by the executing simulation model.

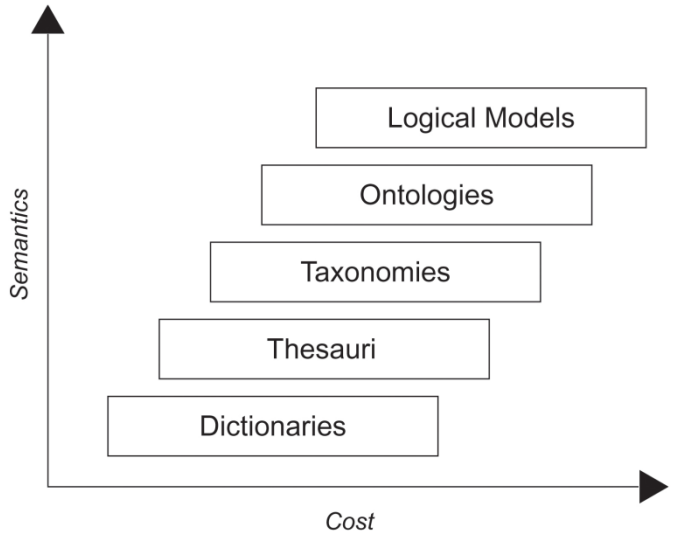


Figure 1-2. The ontological spectrum proposed by Daconta et al. (2003).

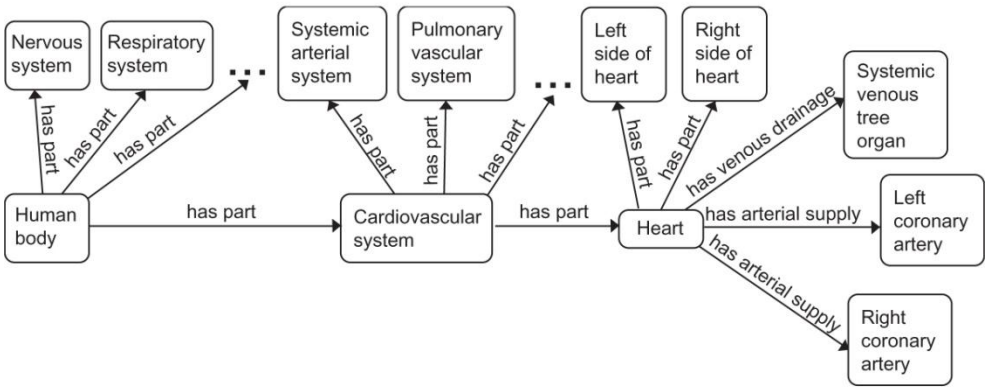


Figure 1-3. A portion of the foundational model of human anatomy ontology (Rosse and Mejino, 2007).

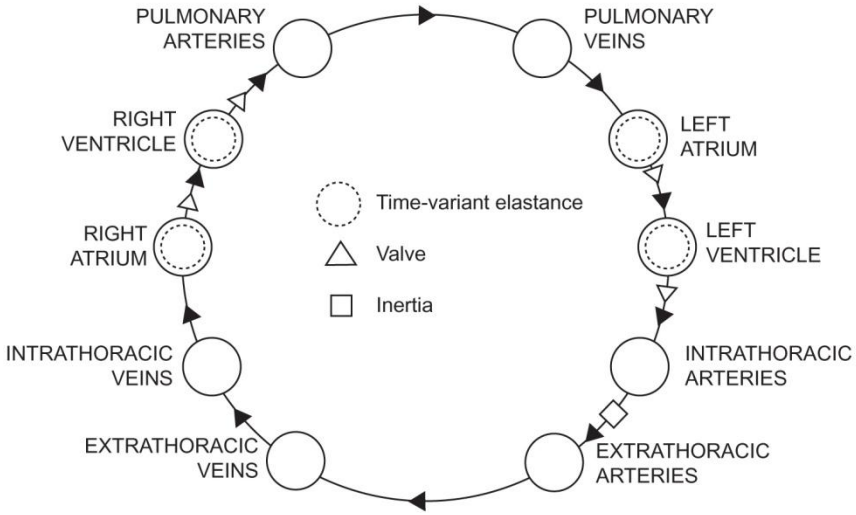
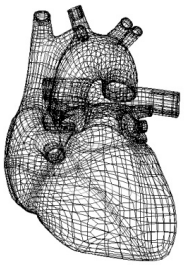


Figure 1-4. A dynamic compartmental model for cardiovascular physiology (Beneken, 1965).

```
# XSI Wavefront OBJ Export v3.0
# File Created: Wed Jan 25 14:24:37 2006
# XSI Version: 5.0.2005.1013-1
mtllib Heartv1new.mtl

o Heartv1
# Hierarchy (from self to top father)
g Heartv1

#begin 1712 vertices
v -5.264736 3.706316 -4.612714
v 2.876265 1.028328 -0.875701
v 2.312174 2.329379 -2.869420
v 1.520226 5.847047 -5.753866
v -6.192419 6.590274 -5.252864
v -5.100938 3.505112 0.157234
v 2.354231 2.223136 3.524546
v 2.358639 3.654171 3.787921
v 1.991272 7.870456 3.089629
v -6.221792 7.309062 1.223697
v -6.487953 8.245123 -5.329836
v -6.784747 8.750317 -0.077993
v 0.224080 10.987264 -0.942205
v 1.191596 11.924877 -4.302668
```



A B C

Figure 1-5. Different representations of graphical models. A) Text from an ".obj" file that lists the vertices of a mesh and connectivity information to form edges and faces. B) A visualization of the geometric model of the heart. C) A render created using the complete graphical model, with material information.

CHAPTER 2
BACKGROUND

This research builds on previous work across several computer science sub-domains. In Section 2.1, background on ontologies, including those leveraged in simulation, is presented. In Section 2.2, background is given on visualization frameworks. Integrative multimodeling can be applied to link these two areas in the user interface, and is described in Section 2.3. Finally, Section 2.4 provides an overview of the state-of-the-art tools used in industry.

2.1 Ontologies

2.1.1 Ontologies: A Wide View

"The Semantic Web," the seminal 2001 Scientific American article, depicted a future of the Internet where reasoning is employed over vast ontologies, composed of classes and relationships, in the support of a more intelligent browsing and searching experience for users (Berners-Lee et al., 2001). A variety of new knowledge-based applications would also emerge. It is over a decade later and the proliferation of the Semantic Web via such standards as the Resource Description Framework (RDF) is debatable, but ontologies have still became useful in many ways. In biological sciences, for instance, the Gene Ontology Project, created by a consortium of biological scientists to aid in the construction of a unified vocabulary across different biological domains, has been very successful and continues to grow (The Gene Ontology Consortium, 2000; The Gene Ontology Consortium, 2010). Another example of an innovative use of ontologies is the "Port" ontology (Liang and Paredis, 2003) which defines how mechanical components of systems fit together to automate engineering model composition.

The increasing adoption of ontologies may be attributed in part to the graphical tools which have been developed for creating and editing them. Perhaps the most prolific of these tools is Protégé, which was developed at Stanford and maintains an active community of thousands of users. Protégé allows users to create knowledgebases with a hierarchical class structure and supports adding relationships between classes (Noy et al., 2001). It also allows users to view their knowledgebases in several graphical layouts, including a frames-based view and a graph-based view of classes and attributes. After the substantial adoption of Protégé by practitioners in a variety of fields, the Web Ontology Language (OWL) plug-in was developed (Knublauch et al., 2004). The plug-in adds support for creating and editing ontologies encoded in the OWL and support for reasoning using description logics. Reasoning functions supported by OWL include consistency checking (i.e., determining if a class could be instanced), and classification (i.e., determining a conceptual hierarchy among classes). Figure 2-1 shows a screen shot of the Protégé loaded with an ontology representing a newspaper.

2.1.2 Ontology Visualization

The concept of the Semantic Web propagated through the computer science community, which prompted some researchers to explore the visualization potential of the Semantic Web's underlying ontological structure. Van Harmelen et al. (2001) were some of the first to demonstrate the expressive potential of a visualized ontology. They created an ontology visualization representing the Dutch job market by employing an augmented version of the spring-force graph visualization technique (Eades, 1984) in which spring-force values are mapped to semantic relationships resulting in semantically similar concepts being visualized near each other. Currently, Protégé

provides some mechanisms for 2D graph-based visualizations of the ontologic structure. TGViz Tab (Alani, 2003) is a Protégé plug-in which leverages TouchGraph technology built with Java (TouchGraph, 2010).

Due to the shortcomings of the 2D layout (e.g., clutter, overlapping), some began exploring visualizing ontologies in three dimensions. OntoSphere3D (Bosca and Bonino, 2006) allows users to browse a dynamic yet fixed-height 3D tree and click on nodes to "zoom-in" and change the level-of-detail. At the "root" of the ontology, concepts are visualized on a sphere which the user may rotate to see the big picture and deduce what the ontology is "about." Another example is the work done by Butain (2008), where semantic search results are visualized in 3D. Attributes of a semantic search "hit" are arranged in a radial fashion about the result. Conversely, the hierarchy of which the result is a member is arranged vertically. For more approaches to 2D and 3D visualization of ontologies, see the survey performed by Katifori et al. (2007).

One final visualization effort worth mentioning is the Intelligent 3D Visualization Platform (I3DVP) created by Kalogerakis et al. (2006). They defined a methodology and constructed a platform that facilitated the merger of an ontology for graphics primitives with an arbitrary domain ontology. Part of their motivation was to provide more visual results when performing Semantic Web enabled functions, such as semantic search. As a demonstration of the methodology, ontology-based graphical representations were merged with a portion of the Gene Ontology. I3DVP also supports visualization-based queries (e.g., "find all things represented by a sphere") and partial visualization automation (e.g., "all sub-classes of a given type should be represented by a sphere").

2.1.3 Ontologies in Simulation

The work of Tolk and Turnitsa (2007) provides a suitable introduction to the idea of applying ontological means to the field of modeling and simulation. They focus on the theory of applying different parts of the ontological spectrum (Daconta et al., 2003), shown in Figure 1-3, to challenges in the field. These challenges include the formal definition of agreed-upon simulation concepts and primitives, multi-resolution modeling, the creation of a universal language in support of simulation-system composability and execution, and the automation of simulation application development. Various research efforts address these issues and are described later in this section.

The Semantic Web was described by visionaries in the infancy of semantics-based application development and representations. Much in the same vein, Tolk (2006) describes the "Dynamic Web" as an extension of the Semantic Web that functions as a "web of composable services." He outlines the fundamental requirements for the Dynamic Web: (1) services must provide a self-description in a standardized format; (2) services must be searchable based on their functionality; (3) Communication with the services must be possible; (4) services must be able to be initiated with data; and (5) services must be able to exchange data with other services. The remainder of this section is dedicated to highlighting research efforts that utilize ontologies in different ways to benefit the field of modeling and simulation.

2.1.3.1 Ontologies to define simulation concepts

The Discrete-event Modeling Ontology (DeMO) was created as a prototype ontology for modeling and simulation (Miller et al., 2004). DeMO was built using the OWL plug-in for Protégé. The ontology contains three top-level classes: DiscreteEventModel, ModelComponent, and ModelMechanism. Under these three

base classes is a collection of subclasses representing the necessary concepts within discrete-event modeling. For instance, two subclasses of DiscreteEventModel (the “DeModel” shown in Figure 2-2) are State-OrientedModel and Event-OrientedModel. ModelComponent has corresponding subclasses such as state and event. The authors of DeMO concede that the classes such as state and event are fundamental concepts and will require further formal specifications which are implementation-specific for a given simulation environment. However, once users of DeMO provide these specifications for the "primitive" types, they can leverage the entire DeMO semantic network that is machine-understandable. This will permit automation of certain areas of simulation application development. The decrease in application development time through inference and reuse of ontology components afforded by the use of DeMO is touted as a core offering of ontologies to the field of modeling and simulation (Lacy and Gerber, 2004).

2.1.3.2 Ontologies for semantic web based simulation systems

Proposals have been made to utilize web components and online knowledgebases as the foundation for new component-based simulation tools (Chandrasekaran et al., 2002). This approach is appealing because it allows simulation designers to choose from state-of-the-art components when building a simulation system. An ontology could be utilized to organize the available components and allow sophisticated searching and semi-automation when piecing together a simulation system. Chandrasekaran et al. also suggest the use of simulation and modeling ontologies, such as DeMO, to create simulation components with a high level of interoperability. JSIM (Miller et al., 1999) is suggested as web-based language to code the simulation components. Figure 2-3 depicts the different simulation components needed in the Semantic Web. Some are

entirely decoupled (e.g., databases and visualization tools) and others are coupled to varying degrees (e.g., animators and optimizers). The completely decoupled modules will integrate into simulation systems well because internet throughput will have a minimal effect. With smart design, however, the coupled modules could be also be integrated.

2.1.3.3 Ontologies for environmental data representation

Authors of the Synthetic Environment Data Representation Ontology (sedOnto) project aimed to create an ontology for synthetic environments based on the Synthetic Environment Data Representation and Interchange Specification (SEDRIS) (Bhatt et al., 2004). Synthetic environments (e.g., a virtual town square) have many applications in simulation (e.g., military simulations and vehicular piloting simulations). SEDRIS (2012) provides a representation of environmental data and a loss-less, non-proprietary interchange format. The latter makes SEDRIS a natural fit for ontological formalization. The sedOnto is created after the SEDRIS UML notation is mapped to an OWL ontology. The authors of sedOnto eventually supplemented their approach with a SEDRIS to OWL transform tool (STOWL) (Bhatt et al., 2005). STOWL uses assertions to automate the transformation of a SEDRIS-defined Synthetic Environment into one defined in the OWL language using the Jena 2 Ontology API (Carroll et al., 2004). The transformation process is illustrated in Figure 2-4, along with a snippet of the resulting sedOnto OWL file.

2.1.3.4 Ontologies for model translation

The Process Interaction Modeling Ontology for Discrete Event Simulation (PIMODES) was developed to support the open interchange of simulation models in commercial simulation software packages (Lacy, 2006; Silver et al., 2006). PIMODES

is a vendor-neutral intermediate model format encoded in OWL. The simulation software packages tested in the development of PIMODES were ARENA, ProcessModel, AnyLogic and ProModel. The process interaction data structures of these software packages were studied and aided in creating “harmonized” concepts which were composed to develop the final ontology. To demonstrate the potential of PIMODES, Lacy created a translator program. The flow and function of this program is depicted in Figure 2-5. Lacy’s work demonstrates the potential of ontologies to aid in the conversion between proprietary model formats. During tests, however, translations did not always work flawlessly as some semantic information was lost due to lack of access to the inner workings of the commercial (closed-source) software packages.

2.1.3.5 Ontologies to support re-use

As a lighter alternative to distributed simulation solutions, Bell et al. (2008) developed a framework for simulation model re-use that utilizes an inter-organizational semantic network. The framework defines the ontology engineering process and the semantic search technology required to intelligently locate models. All models are based on representations found in commercial off-the-shelf simulation packages (CSPs), such as Simul8.

The ontology engineering process starts with the analysis of models created in CSPs . This analysis leads to the creation of OWL ontology classes to represent the necessary simulation components, a process known as Component Typing. These classes are then merged with DeMO. Dependencies are determined between classes in DeMO and concepts in the domain. The result is a simulation model ontology which contains domain specific knowledge of use to practitioners. This resulting ontology is termed the Discrete Event Simulation Component (DESC) ontology and is exported as

an OWL file and stored on a SEDI4G server. SEDI4G (Bell and Ludwig, 2005) is a research effort into grid-based systems for semantic discovery.

A component discovery system is also defined as a semantic search program that intelligently traverses the SEDI4G server given a text string as input. Results are returned to the searcher and the appropriate model can be selected. In the software prototype developed, the models are downloadable to an XML file which can be imported into Simul8, but this approach could theoretically be used with any CSP. A high-level view of the framework defined by Bell et al. is depicted in Figure 2-6.

2.1.3.6 Ontologies to support frameworks

Benjamin et al. (2006) identified distributed, federated simulation systems as being difficult to build and requiring a long development time. Ontologies play an integral role in a framework which was developed to address these issues. Benjamin et al. describe several ways in which an ontologically-centered design philosophy is beneficial to simulation model development. Ontologies serve to harmonize terminology, promote a common knowledge and language, and help relate organizational goals to simulation goals. They can also aid in model construction by clearly and unambiguously separating levels of abstraction and determining simulation model objects, structure, and logic. Finally, they can help identify data sources and can inform data mining techniques by extracting semantics from textual descriptions.

Ontologies can also be incorporated into the infrastructure required for distributed simulation systems. This approach has several advantages. Creating a simulation model ontology can provide a common language which can be used to share semantics between commercial simulation packages. Ontologies can also be useful when constructing simulation systems composed of federated components by ensuring

component congruency and semantic interoperability. Additionally, support for multi-resolution modeling (i.e., models representing the same phenomena but with different levels of detail) can be added to ontological representations. With this support, rules can be applied to model ontologies; and models of different resolutions can be merged, or new hybrid models can be created from multiple models representing the same phenomena but in varying capacities.

The ontology-centric design philosophy described in this section was applied to create the Framework for Adaptive Modeling and Ontology-driven Simulation (FAMOS) (Benjamin et al., 2005). Modeling in FAMOS entails selecting and editing templates and composing models together from library components including ontologies for different domains and process models. Simulation and domain ontologies can be utilized in FAMOS and it provides a Model Composition tool with a “visual programming” interface in support of the composition of distributed, federated simulations systems. Miller and Baramidze (2005) also noted the ability ontologies have for supporting multimodeling and multi-faceted reasoning to generate and interpret simulation results.

2.1.3.7 Merging simulation ontologies with domain ontologies

Of the previous applications of ontologies to simulation systems, the methodology proposed in this document is most akin to those which utilize domain-specific ontologies. Silver et al. (2007) developed the ontology driven simulation (ODS) framework. Their framework utilizes domain ontologies along with DeMO and consists of an ontology mapping tool, a model markup generator, and a simulation code generator. The ontology mapping tool allows users to map classes of the domain ontology to classes in the simulation ontology. Their design suite was demonstrated starting with the problem-oriented medical records ontology (PMRO) to build a clinical

simulation. Examples of mappings in this scenario include mapping the class Patient in PMRO to the class Resource in DeMO, and mapping the class ClinicalExamination in PMRO to the class Activity in DeMO.

After the domain ontology is aligned with the simulation ontology, an intermediate markup file can be generated that represents the simulation instance as either a Petri Net model or a process interaction model. This intermediate markup file can then be translated into executable models that run in JSIM (Miller et al., 2000) or ARENA. Silver et al. describe four main benefits of the ontology driven approach: (1) the use of a shared vocabulary facilitates good communication between modelers and end-users, (2) models created for different simulation packages using different simulation world views can be represented in a unified language, (3) models are encoded in a Semantic Web compatible language, thus supporting potential web-based model repositories and web-component-based simulation systems, and (4) model development can be sped up by allowing modelers to assemble models by using a set of predefined classes represented in the domain-specific and simulation ontologies.

Together, these projects illuminate the roles ontologies have played in the field of modeling and simulation. They are often used to formally define programmatic structure as a way to encourage simulators to speak the same language when coding systems. This supports interoperability and model composability. Further, various levels of reasoning have been applied to simulation ontologies resulting in the partial automation of simulation application development. Some also explored the use of domain ontologies for constructing simulation systems. The methodology proposed in this document will compliment these previous efforts in utilizing ontologies in modeling and

simulation by manifesting the ontologies as 3D graphs within a visualization environment for authoring of simulation models and visualizations. The 3D graphs can then be used as part of an integrative visualization or as an interface affordance.

2.2 Visualization Frameworks

Since researchers began to explore the potential of computer graphics, the visualization community has called for scientific visualization frameworks that would abstract low-level graphics manipulation and rendering functions away from the user, and would provide a more intuitive integration between data sources and visualization design features for domain practitioners (Treinish et al., 1992). Some of the first attempts at creating such a theoretical framework were by Haarslev and Moller (1988), and Upson et al. (1989). Since these seminal projects, much research has been dedicated to designing visualization frameworks that employ the increasing power of hardware and leverage our increasing understanding of what makes interaction techniques effective (Tory and Moller, 2004). This continued work has provided a set of visualization tools to those working in many scientific and engineering domains. For the remainder of this section, select work towards creating visualization design systems is presented in chronological order.

Visualization of Experimental Systems (Haarslev and Moller, 1988) – one of the first efforts in visual dataflow programming to take advantage of the mouse and keyboard. This allowed users to directly manipulate dataflow graphs. The system was originally designed to support image processing techniques but was extended to allow construction of more general algorithms, and to allow visualization of their output.

Application Visualization System (Upson et al., 1989) - a system oriented towards engineers and scientist who did not necessarily have experience in computer

programming. Within the framework, a direct manipulation interface was employed to allow the user to construct dataflow networks that ultimately fed into one of the provided visualization types (e.g., voxel or iso-surface renderings). There is a similar approach to visualization construction in many modern tools (see Section 2.4).

Object-Oriented 3D Graphics Toolkit (Lucas et al., 1992) – a framework oriented towards visualization application programmers. The framework allowed programmers to more efficiently develop visualizations by providing features such as a scene graph data structure to organize the scene's graphics for rendering, direct manipulation interface functions, and code templates for 2D widgets.

VISAGE (Schroder et al., 1992) – an object-oriented visualization design library. Objects were built in LYMB (Lorsen and Yamrom, 1989), an animation programming library coded in the C language. A collection of visualization objects exist in the library (e.g., structured grids, unstructured point sets). Each object in the library supported the notion of time, allowing a visualization designer to animation their representations.

VISTA (Senay and Ignatius, 1994) – a knowledge-based system for visualization design. Domain knowledge is mapped to visualization primitives (e.g., 3D scatter plot, shape, size). Once a visualization is automatically generated based on input data, the designer may interactively modify the results to further suit their needs.

The Visualization Toolkit (Schroeder et al., 1996) – a C++ library of visualization objects. VTK is intended to be executed in a dataflow environment and can utilize a variety of rendering APIs that were prolific at the time of development (e.g., OpenGL, Sun's XGL).

DSR Open Lab (Kolodnytsky and Kovalchuk, 2001) – an interactive visualization system with a focus on the graphical user-interface and usability. The software used OpenGL to render a multitude of 2D and 3D histograms. The user interface code was multi-threaded, permitting new levels of interactive graphical manipulation.

Davis (Huh and Song, 2002) – a statistically-oriented Java based tool for creating advanced 2D data plots. Davis was unique at its time of release because it supported sophisticated statistical visualizations techniques (e.g., flipped empirical distributions, Grand Tours) through user-interface controls.

The InfoViz Toolkit (Fekete, 2004) – a Java 3D-based code library that allows a programmer to easily encode traditional information visualization structures. Data is obtained by importing tables with named columns.

prefuse (Heer et al., 2005) – a robust java-based toolkit for creating information visualizations. The toolkit provides classes to support typical information visualization tasks and renders visualization output using the Java2D API. The creators of prefuse performed several user surveys and experiments that allowed them to identify issues with the toolkit design and vindicated perfuse as a useful tool for visualization constructions for those with Java programming experience.

VIS-STAMP (Guo et al., 2006) – a visualization system for space-time and multivariate patterns. VIS-STAMP provides geovisual analytic tools with support for cartographic-based visualizations. It also supports analysis of spatio-temporal patterns by employing self-organizing maps, parallel coordinate plots, recordable matrices, and recordable map matrices.

Hear And Now, Near At Hand (Hannah) (Einsfeld et al., 2007) – a 3D user interface to synthesize desperate data within a single view. Instead of collecting data from databases or flat files, Hannah requires the input information to be encoded in an ontological format. This allows Hannah to create more meaningful 3D visualizations that leverage additional semantics. The creators of Hannah describe interaction within the system akin to those interaction techniques found in the 3D modeling packages, in that 3D objects within visualizations can be scaled, rotated, and translated using direct manipulation.

ManyEyes (Viégas et al., 2007) – a website that allows visitors to create visualizations of uploaded data. There is also a social component to ManyEyes, where users can view and comment on other's data visualizations.

Web Data to Visualization (Gilson et al., 2008) – an experimental framework whereby webpages are mapped to domain ontologies (e.g., an ontology for music sales charts), which is in turn mapped to a ontology of visualization concepts. The OMEN (Mitra et al., 2005) ontology mapper is used to organize meta-level information and map the semantics across the different paradigms.

Protoviz (Bostock and Heer, 2009) – a toolkit written in Javascript that allows visualization designers to leverage a set of preexisting graphical primitives to create data-based visualizations. Protoviz provides support for rendering in HTML 5, SVG and Flash. Protovis serves as the core of the collaborative web-based visualization design tool, ProtoViewer.

Graphical Interface Construction Kit (GiCK) (Singh and Sivaswamy, 2010) – an interface construction tool that allowed novice users to create user-interfaces for

simulation systems. The interface is specified in an XML file and then parsed and created by GiCK.

Behaviorism (Forbes et al., 2010) – a Java/OpenGL based framework for creating 2D and 3D visualizations. Users of the framework create visualizations by designing around three graph structures: the data graphs that define data sources, the scene graph that defines graphical resources and their transformations, and the timing graph which is composed of 'behaviors' which are scheduled events that affect the visualization scene. A collection of interactive visualizations were created with behaviorism that demonstrate the versatility of the approach.

EditFlow (Benzaken et al., 2011) – One of the first information visualization frameworks built on top of a database management system (DBMS). The DBMS supports strong scalability and persistence across multiple users, features not inherently present in frameworks that operate by loading all required visualization data in-memory.

2.3 Integrative Multimodeling

Fishwick (2004) presented the challenge of integrative multimodeling, which requires using multimodels and adhering to new HCI criteria which did not traditionally apply to simulation. Most models of use are multimodels since individual model types are often only able to represent a subset of a system's behavior. Integrative multimodeling calls for a closer linkage between the abstract model and the phenomena which it represents. One way to achieve this is to graphically represent the abstract model and phenomena within one cohesive interface and allow user interactions on both the abstract and concrete graphical representations. This new modeling technique also has four main HCI requirements: usability, emotion (by means of connotation), immersion, and customization.

2.3.1 An Integrative Model-Blending Environment

An early application of the principles of integrative multimodeling is the work done by Park and Fishwick (2005). A framework was created that allowed for the integration of different model types within a 3D simulation environment. Dynamic, interaction, ontological, and geometric model types were included. The framework provides a Model Explorer for dynamic modeling and an Ontology Explorer with OWL integration so users can create OWL classes and subclasses. The framework runs inside of Blender (Blender3D, 2012) and requires the RUBE simulation library (Hopkins and Fishwick, 2001) to build dynamic models and execute simulations. Users can create geometry using Blender or import pre-existing meshes. Dynamic model components (e.g., blocks within a Functional Block Model) can be selected from RUBE and associated with geometry in the scene. Using Blender Game Engine functionality, an interaction model can be created and linked with geometry. Shim and Fishwick (2008) demonstrated the use of integrative multimodeling by co-locating a system dynamics model for ecological phenomena and behavior.

2.3.2 The Augmented Anesthesia Machine

A recent attempt to apply integrative multi-modeling to anesthesia training has been successful by employing mixed reality. Quarles et al. (2008) created a mixed reality training tool that co-locates abstract models used to teach anesthesia machine operation with the components on the physical anesthesia machine. The tool, known as the Augmented Anesthesia Machine (AAM), uses a “magic lens” interface that employs a tracked tablet PC which serves as a window into the real world. The tablet is tracked with six degrees-of-freedom using infrared cameras and small reflective objects. Users hold the tablet PC in front of the physical anesthesia machine and see a 3D rendering of

the machine on the display from the perspective of the tablet. On top of this 3D rendering, the abstract model is visualized using iconography, and color-coded particle systems are used to visualize invisible gas flow within the machine.

User studies were performed that compared the AAM to traditional training methods (e.g., using only an instruction manual and the anesthesia machine) and web-applets which used abstract visualizations (Fischler et al., 2008). The results of the study showed that the AAM was most successful at bridging the gap between abstract and concrete knowledge, thus proving the approach is a potentially viable educational technique which should be explored further. The author of the AAM points toward the use of ontologies as a way to define a theoretical framework for such mixed-reality systems (Quarles, 2009).

I view the proposed methodology as a natural evolution of the work done by Quarles et al. (2008). The methodology will provide a theoretical foundation for software tools such as the AAM. By using domain ontologies (e.g., an ontology for anesthesia equipment) as the starting point for dynamic model building and visualization construction, users will not have to hard code visualization solutions.

2.4 Simulation and Visualization Design in Practice

This section is dedicated to providing a brief overview of the state-of-the-art for simulation-based visualization design. This includes existing technology that provides some sort of interface to visualization functions that can be used create to reasonably sophisticated visualizations based on collected real-world data or simulation output. These technologies range from pure graphics programming to purely graphical user-interfaces for non-programmers.

2.4.1 Graphics Programming

Graphics programming is the most powerful and customizable way to create visualizations. Graphics programmers leverage low-level APIs such as OpenGL or DirectX, or higher-level programming environments such as OGRE. When employing graphics programming for visualization design, simulation models that drive the visualization can be coded in a separate package or along with the graphics in the same code base. An example of employing graphics programming is the detailed, segmented cardiac visualization coded in OpenGL (Wang et al., 2011). Tools that abstract away graphics programming from the end visualization designer (including the goal of this work) still must build upon some graphics API provide the visual output to the user.

A limitation of the pure graphics programming approach is that it, of course, requires programming aptitude which practitioners operating in most domains do not possess. In some cases graphics programming is even overkill for those with programming expertise and in this case the designer will choose to use one of the visualization design tools listed in the following sections.

2.4.2 Engineering Simulation Tools with Visualization Add-ons

I classify engineering simulation tools as tools that provide a graphical user interface to advanced mathematical modeling. Examples of such tools are Simulink, LabVIEW, and Ptolemy. The Simulink method of creating visualizations based on the encoded mathematical model was described in Chapter 1 and depicted in Figure 1-1. LabVIEW provides an interface similar to Simulink for mathematical modeling, but integrates 3D visualization differently. Within LabVIEW, 3D visualization objects, known as SceneObjects, are represented as blocks and can be connected to the greater mathematical model on the LabVIEW canvas (National Instruments, 2008). OpenGL is

the underlying graphics technology used in LabVIEW to render SceneObjects. Transform nodes can be added to the canvas and used to link simulation variables to graphical parameters. Visualization output (all of the SceneObjects) are rendered in a 3DPicutreControl window when the simulation model is executed. LabVIEW also provides extensions for graphics programmers to code custom visualization objects to be utilized in comprehending simulation output.

Ptolemy, a Java-based simulation platform for simulation modeling and execution, operates in a manner similar to LabVIEW. The focus of Ptolemy, however, is more on merging heterogeneous model types within a single modeling environment (Lee and Seshia, 2011). With regards to visualization, Ptolemy leverages transformation objects that may be placed on the Ptolemy canvas (this is similar to LabVIEW). These transformation objects may be chained together to create a final composite transform that can be based on simulation output to varying degrees. An example of this is shown in Figure 2-7. A basic simulation of orbital dynamics between the Sun, Moon and Earth was created in Ptolemy and visualized with 3D shapes that Ptolemy provides (spheres in this case). The portion of the model shown in the figure pertains solely to the graphical output; simulation calculations are performed at a different level of the model.

2.4.3 Game Engines and 3D Modeling Packages

Game engines are software tools that enable designers to create interactive virtual environments with a focus on customizable graphical presentation and diverse user input. Examples include The Unreal Engine, the Source Engine, and Unity3D (Petridis et al., 2012). Similar to simulation packages, each game engine requires users to learn the specific modeling paradigm defined within the engine. Often a game engine will support scripting in some pre-existing programming language. Using such tools can be

viewed as engineering practice because any content modeling performed will need to be translated through the game engines modeling paradigm and script will potentially need to be coded. Certain engine designers are aiming for ways to open their technology to practitioners in domains outside of engineering and computer science, but the current state of the art still requires substantial engineering effort (Thiriet et al., 2011).

Modern 3D modeling packages are similar to game engines in that they provide 3D rendering capabilities and often include scripting features for the creation of customized plugins. However, 3D modeling packages focus on providing a tool set for the creation of 3D content (modeling, texturing and animating 3D objects), not the synthesis of content and interaction to create a "game." Visualizations based on simulation dynamics can be encoded in 3D modeling packages, but this requires custom scripting or selection from a pre-determined set of simulation features, such as the fluid or cloth dynamics options provided in Blender.

2.5 Summary

In this chapter, relevant background was presented from the areas of ontologies, simulation, visualization frameworks, and integrative multimodeling. Ontologies have been used to approach some challenges in simulation, but have been applied mostly on the computational level. Visualization frameworks have progressed recently but have been oriented mainly towards information visualization, not simulation systems. Finally, integrative multimodeling is a new area concerned with linking abstract and real-world representation and is applied in this work towards the definition of new theoretical interface. State-of-the art tools used in practice for simulation-based visualization construction were also presented in this chapter.

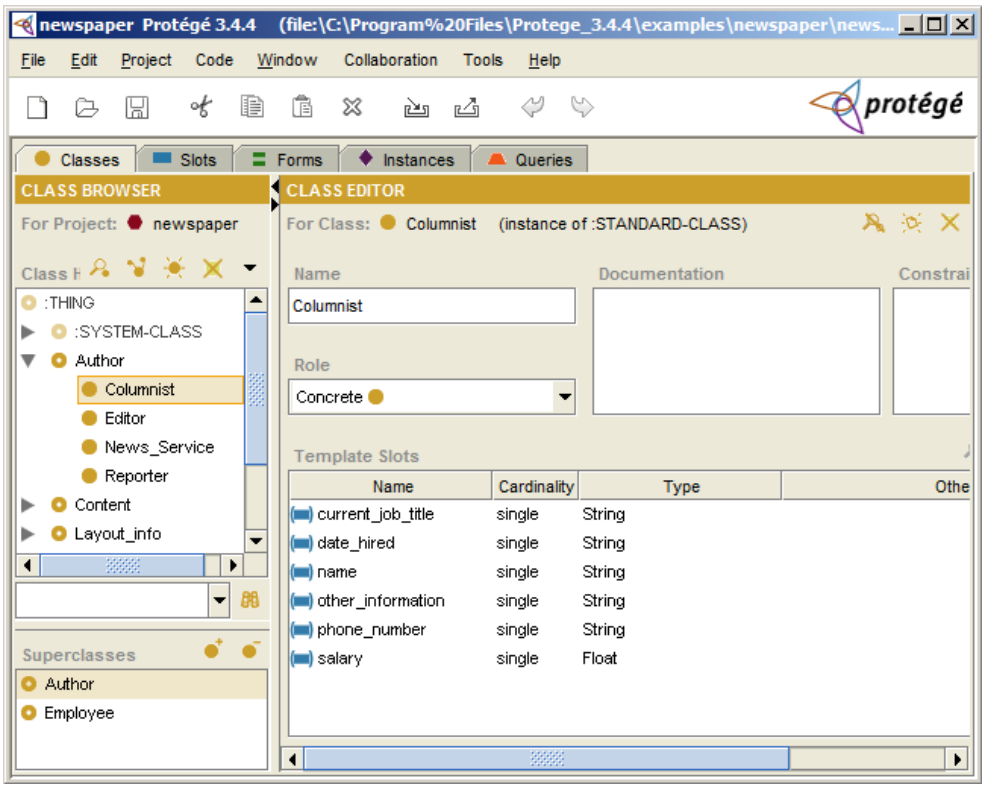


Figure 2-1. Protégé (Noy et al., 2001) displaying an ontology for a newspaper. The class that is selected is Columnist, which is a subclass of Author. "Slots" (i.e., attributes) of a Columnist are shown next to a blue icon.

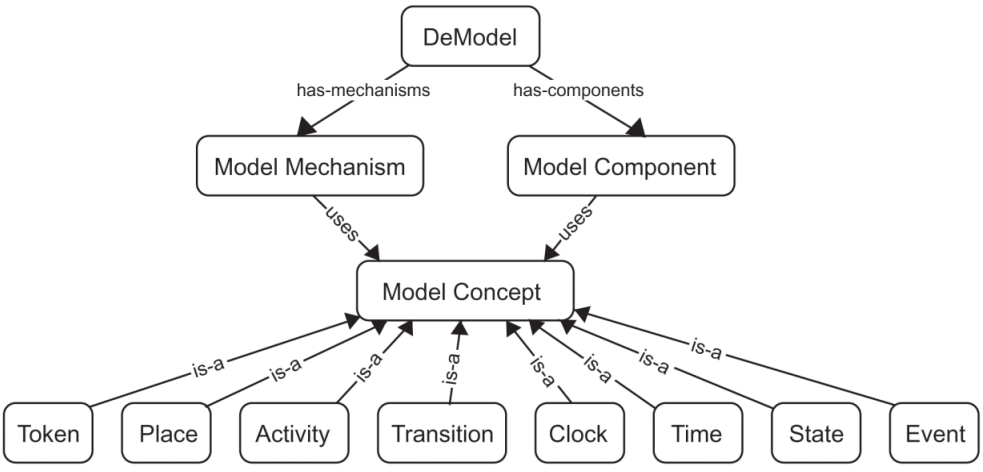


Figure 2-2. A partial view of DeMO (Miller et al., 2004). Model Concept types are enumerated.

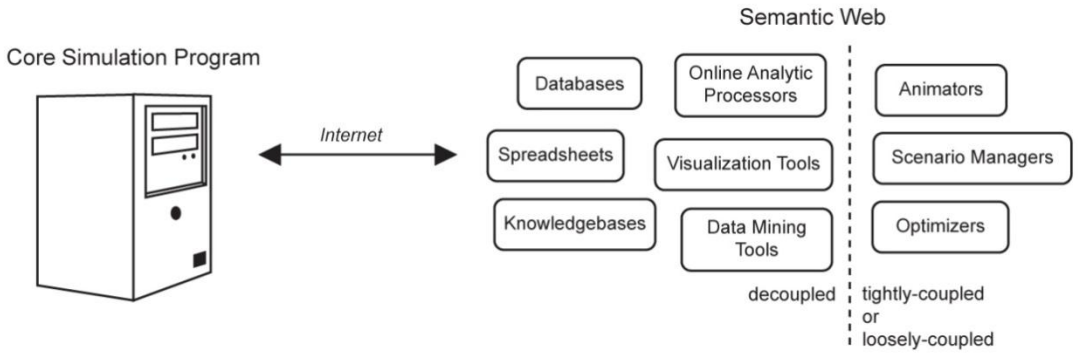


Figure 2-3. A high-level view of the Semantic Web enabled simulation design system proposed by Chandrasekaran et al. (2002).

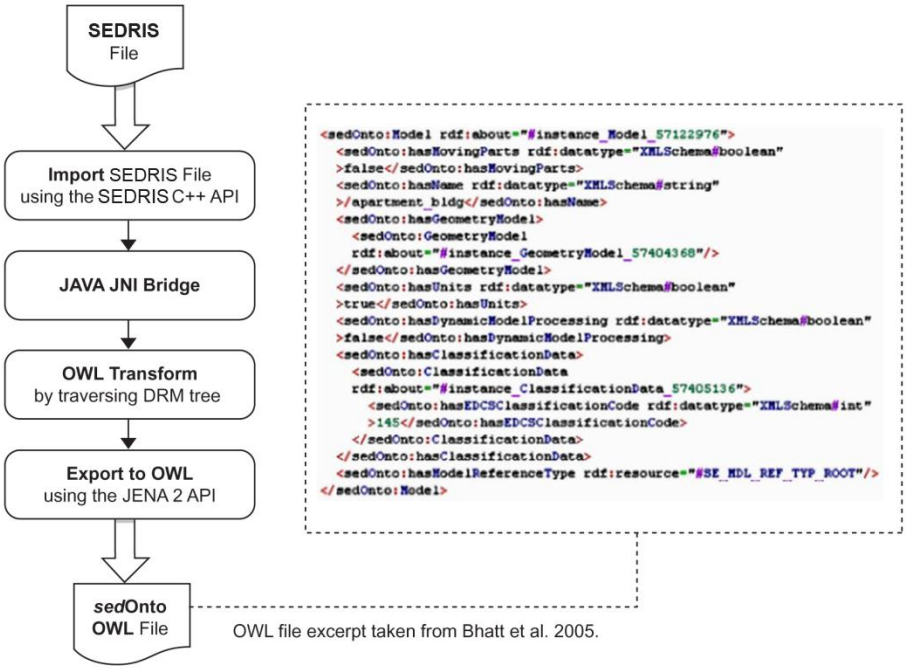


Figure 2-4. Transforming a SEDRIS file into an OWL ontology (Bhatt et al., 2005).

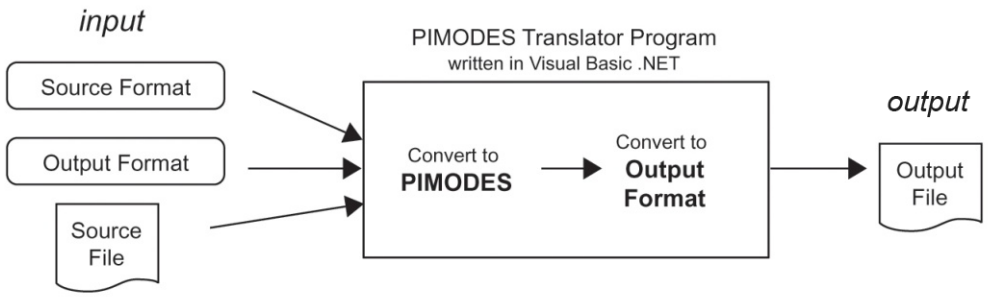


Figure 2-5. High-level depiction of the translator written by Lacy (2006) to demonstrate PIMODES.

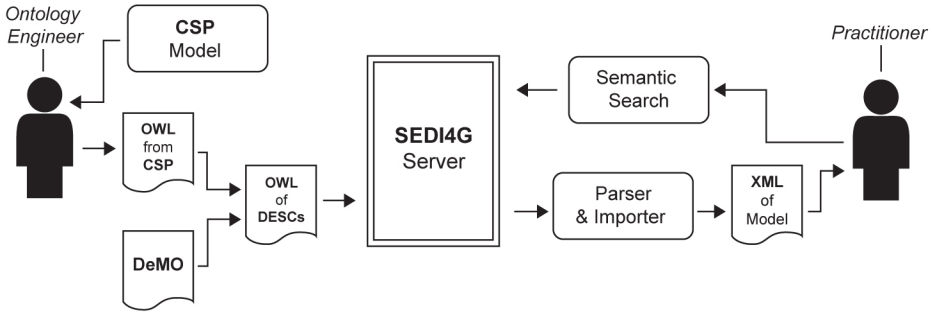


Figure 2-6. The ontology engineering, storage and search framework created by Bell et al. (2008).

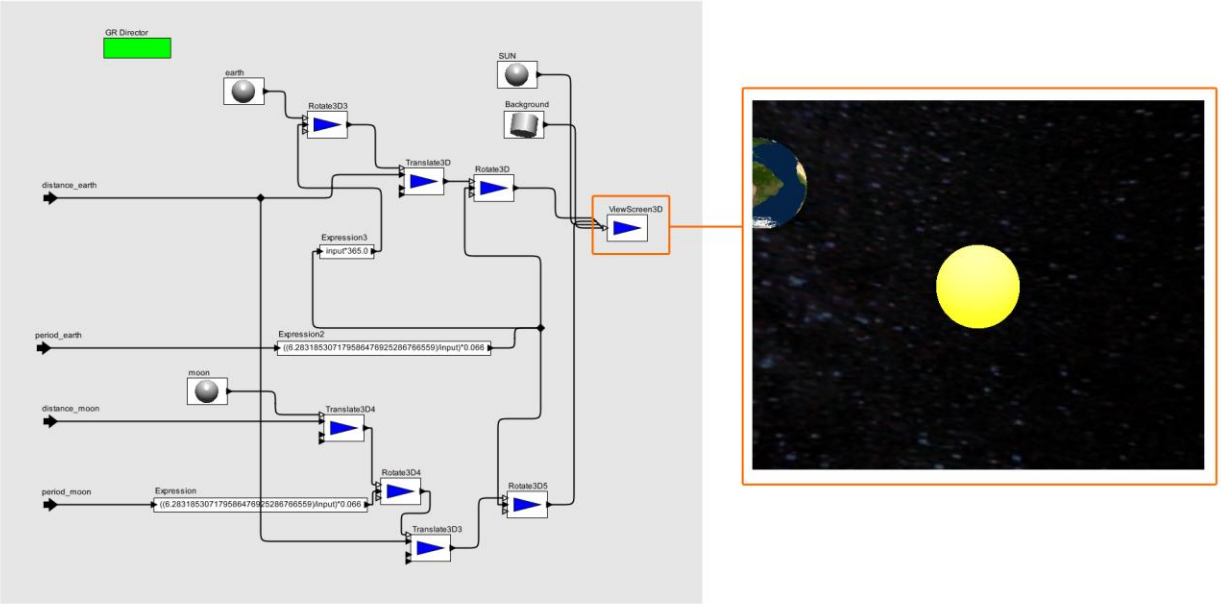


Figure 2-7. An example of transforming and rendering 3D objects based on simulation variables in Ptolemy. Textured spheres representing the Sun, Earth and Moon are transformed using a series of Ptolemy transform objects (translate rotate, scale). A ViewScreen3D object collects the transformed spheres as input and renders them in a separate Java3D view, shown to the right.

CHAPTER 3
ONTOLOGY-CENTERED INTERACTION THEORY AND FRAMEWORK

The theoretical interface requirements and design methods described in this chapter represent the core contribution of this work. A case study presented in the following chapter may help ground the high-level concepts presented here through real-world examples. The design of the proposed framework should facilitate simulation model building and integrative visualization construction within a single interface model that leverages pre-existing domain semantics when possible. The defined interaction theory should demonstrate that a visualized ontology could be the central connective interface structure through which these different activities can be afforded. The ontology-centric design can also support executing the designed simulation models and, in turn, "playing" the designed visualizations. The interface requirements described in this chapter support a simulation and visualization design environment. This is a 3D graphical environment with controls to start, pause, and reset the simulation and visualization at any time.

The three main components of an ontology -- concepts, relationships and attributes -- all play important roles in the proposed interaction theory. The role of each component will be described in Section 3.1. A detailed look at the various design affordances of the theory will be presented in Section 3.2. A technical framework that can house the proposed interaction theory is presented in Section 3.3. A methodology for using the proposed framework is presented in Section 3.4. Finally, a classification, comparisons, and limitations of the framework are discussed in Section 3.5.

3.1 The Base Structure

3.1.1 Concepts

In a domain ontology, concepts represent high-level objects in the domain space (e.g., in the medical domain anatomy could be represented by a set of interconnected organs). Likewise, when an ontology is visualized, a concept can serve as a high-level interface handle to parameters required for simulation and visualization. That is, the visualization of the ontology concept can serve as a handle to more information about the concept in the form of *attributes*; and can serve as a visualization design interface structure that allows users to perform graphical transformations. When a node is transformed, any 2D or 3D visual objects that are assigned to the concept as attributes will be transformed as well. The visualization of the concept object is the graphical parent of all visual objects assigned through attributes. A depiction of this basic idea is given in Figure 3-1. The concept is visualized as a gray sphere. When the sphere is selected, a graphical widget appears to allow for 3D translations. A 3D mesh (the bunny) and image (xray.jpg) are assigned to this particular concept. An example graphical transform of translation is depicted in Figure 3-1c, but a rotation or scale widget could be rooted at the ontology concept as well. As the concept is translated, the 3D mesh moves with the concept and the 2D image maintains the same 2D offset.

3.1.2 Attributes

A basic visualization of a concept should serve as an interface to attributes of that concept. When a concept is *selected*, not only should graphical transform options become available, a list of all attributes relevant to that concept should be displayed. Depending on the interaction modality (different modalities are described in Section 3.2.9) attributes can either be read-only or modifiable. In the proposed framework,

attributes play a critical role; they can be used to define simulation variables, visualization parameters, and any required *influence* between simulation and visualization. To have consistency in the interaction model and to massage the ontological structure into something that is well-suited for a visual interface to graph based simulation models and visualization design, attributes may be added to relationships. In the common notion of ontology, relationships do not hold attributes and, therefore, a translation will need to occur if the ontologies leveraged in this framework are to be utilized elsewhere. Attributes of relationships are discussed in the next section and the required translation is discussed in Section 3.2.11.

3.1.3 Relationships

Relationships in an ontology denote a unidirectional semantic connection between two concepts (e.g., “has a”). When visualized, relationships are often represented as an edge that connects two ontology concepts. In the proposed framework, relationship edges can be used to just relay basic semantics as in typical ontological solutions, or can be sculpted into a meaningful curved path within a 3D visualization environment. The resulting curved edge can be used to denote the path of flow within a dynamic system, or allow further expression and serve as a guide for animated objects, such as particle systems.

The physical structure of the relationship, as defined interactively by the visualization designer, can be encoded by an attribute assigned to the edge. The attribute should be a list of ordered points that define the edge and a potential extra attribute that determines if the point set should represent a “hard” edge that is piece-wise linear, or a curved edge that should be interpolated between the points . It may be beneficial to add attributes to edges as well, depending on the modeling paradigm being

leveraged and the edge's influence on visualization. Attributes for a relationship should be displayed when the edge is selected, as is the case with attributes attached to concepts.

3.2 Structural and Semantic Affordances

A collection of design affordances will be presented in this section. These affordances are diverse, ranging from the ability to store simulation equations as attributes, to using relationships to form the path of particle systems. The definition of affordance used in this context is: what the ontological structure provides, or "furnishes" (Gibson, 1979), in support of simulation modeling and visualization design. The following 11 sections each describe an affordance. Collectively, I view this set as a showcase for the use of the ontology-centric approach described in this dissertation.

3.2.1 Meta-attributes

Meta-attributes are attributes of attributes that are required in the proposed framework to define properties significant to simulation modeling and visualization design activities. These meta-attributes include values to determine if an attribute should be visible in a resulting visualization viewer module and if an attribute should be a numerical variable or text-based. If an attribute is deemed a variable, meta-attributes exist that can define the range of acceptable values for the variable, or a discrete set of acceptable values if the variable is crisp.

When adding ontology member attributes required when creating an envisioned visualization module (e.g., the 3D position of an ontology concept), a designer may want to limit visibility of such attributes in the final visualization. Design-centered attributes will not be relevant in most viewing contexts and likely distract from the information the designer intends to convey. To this end, a meta-attribute of "visible in viewer" is added

to each ontology concept attribute. This is a true or false property whose value determines if an attribute will be visible in the viewer module.

Other meta-level attributes in the framework allow a designer to attach interactive controls to numeric attributes. These controls allow viewers to tweak visualization properties in real-time. Attributes could be linked to simulation dynamics (e.g., coefficients in the underlying model) or purely graphical features (e.g., the color of a material assigned to a 3D mesh). Regarding numerical simulation and visualization parameters, there is a true or false value that determines whether an attribute is marked as a "variable." Variable attributes must be numeric in value and be of a scalar or 3D vector type. 3D vector types are well-suited for some visualization parameters (e.g., position: x, y, z and color: red, blue, green). Further, attributes that are marked as a variable can also be marked as "adjustable."

Adjustable attributes can be interactively modified. The control for adjusting should allow modifying a variable attribute to values within a defined range. This range is bounded by a "min" and "max" meta-level attribute. One way to support this is through a linear slider control. With respect to the slider behavior, a designer may not always want this control to be smooth and continuous, but allow only for discrete values from a set. To this end, the proposed model allows for "crisp" adjustable variables. When an adjustable variable is marked as crisp, a set of possible values may be assigned to the variable. These values should be sorted and the interactive control (e.g., the slider) should "snap" to values when a variable is marked as adjustable and crisp. All meta-attributes are depicted in the user-interface sketch presented in Figure 3-2.

3.2.2 Attributes as Simulation Equations

While most simulation modeling paradigms can be made to conform to the ontological graph structure, there will always be mathematical customizations that are desired in certain simulated scenarios (Krahl, 2002). Often these additions are made through custom scripts. In visual environments (e.g., Simulink, Ptolemy), equation blocks can be added to the canvas that consider any input as a variable to use within the equation, and output a calculated value. Another approach in these environments is to leverage a functional block model notation to build a graph that represents the desired formulation. In functional block model notation, operators are blocks within a flow graph with operands as input, and the resulting value of the operation as output.

Representing equations as graphical objects (e.g., blocks on a canvas in Simulink or Ptolemy) within a dynamic model representation does not mesh well with the ontological graph visualization approach. Because the purpose of an equation is to calculate a value which, ultimately, is just an attribute of some concept within a domain (e.g., blood volume is an attribute of an organ in the domain of physiology), the equation can be treated as such and its representation can be encapsulated within that of the concept's. Along with a de-cluttering of the visual design environment, defining simulation equations as attributes of concepts has the additional benefit of allowing equations to be represented in a manner closer to their native text-based form as compared to the to graph-based structures.

Any implementation of an ontology-based equation solver will need to define some base syntax to denote the concept of time, both total time and the current time slice; and to determine from which concept a variable is being referenced. For the latter, the dot notation used in object-oriented programming could be employed. Figure

3-3 explores the dot notation with two approaches for retrieving a variable value from a foreign concept. In Figure 3-3A, syntax is shown of the following structure: `concept(<concept_name>).<attribute_name>` . In Figure 3-3B, a simple function `output()` is used to find a node adjacent to an out-going edge with a given name. This syntax has the following structure: `output(<out-going relationship type>).<attribute_name>`. A function of `input()` can be used in a similar manner. The second syntax style has the advantage of being general enough so that a single equation attribute could be distributed across an entire dynamic model because no explicit concept names are required.

3.2.3 Taxonomies for Object Creation

Ontology inheritance is represented by an *is a* relationship between two concepts. A tree of concepts connected by *is a* forms a taxonomy. This taxonomic structure can be useful when creating simulation or visualization objects. If a root concept can serve as a template for other concepts then any design choice applied to the root can be automatically propagated to any concepts connected through out-going *is a* relationships. During visualization construction, certain graphical properties can be applied automatically to a collection of sub-concepts (e.g., all items of a certain type should be colored red). In simulation models, given a certain modeling paradigm, model blocks may all have the same variable attached (e.g., in compartmental modeling all compartments have simulation variables volume, pressure and flow).

3.2.4 Creating Transform Hierarchies and Vertex Groups

The *has a* relationship is a common semantic notion and can serve as a useful interface affordance. The proposed framework leverages this relationship type to create

graphical hierarchies (i.e., scene graphs) and to define vertex groups within a mesh to create vertex animations. A *has a* relationship between two ontology nodes will form a parent-child pair. This mimics common parenting or grouping functionality present in 3D modeling and animation software (Derakhshani, 2010). With *has a* relationships present, transformations applied to a source node (the parent) of the relationship are propagated down to the destination nodes (the children). This technique can be applied to transform a set of disjoint meshes or to define regions within a single mesh to be transformed. Figure 3-4 illustrates the use of the *has a* relationship in a simple scene containing a tea pot on top of a table. Because a *has a* relationship connects the two concepts, when the table is transformed, the tea pot will be transformed as well. The tea pot will be transformed about the location of the table concept node (i.e., the tea pot transform's origin point will be the point of the table concept node).

Vertex animations are animations in which a subset of vertices is deformed within a mesh over time. Different transforms may be applied to different vertex subsets, resulting in complex animations. This is in contrast to animation in which a transform is applied uniformly across an entire mesh. For example, compare an animation of a beating heart in which the heart scales uniformly to an animation in which the different chambers of the heart scale independently over time. Vertex animations are often created in 3D modeling software packages through a process of "rigging" (Vasconcelos, 2011). Some sort of skeletal structure is provided in the user interface that can be registered with a mesh and manipulated to create vertex animations. Designers can create simulation-based vertex animations in the proposed framework by leveraging the ontology visualization as the required skeletal structured.

In the proposed framework, rigging 3D meshes by leveraging the ontology structure requires *has a* relationships be present in the ontology. The *has a* relationships, which should be unidirectional, along with their adjacent ontology nodes form a hierarchy of parts and subparts. If a mesh is assigned to the root of this hierarchy, then in some cases its subparts will represent sub-regions of the assigned mesh. The ontology nodes representing subparts can be positioned within the root mesh and serve as the interface handle to creating vertex groups of the root mesh. These groups should be centered at the ontology node and can be bounded by a variety of geometry. Further, the span of this bounding geometry can be controlled by the addition of an attribute named `influence radius`.

Once vertex groups are formed by positioning ontology nodes, affine transformations may be applied to each vertex set. Transformations should treat the ontology node's position as the origin point, as opposed to inheriting the origin of the mesh's object space (i.e., the origin point by which the positions of all the mesh's vertices are defined). These transformations may be linked to simulation variables of a node through the addition of influence attributes. A vertex group for the concept "Head" is shown in Figure 3-5. The bounding geometry for this group is a transformed sphere. All vertices within the bounding geometry are added to the "Head" group and may be transformed independently from the rest of the mesh's vertices.

3.2.5 Influence for Ontology-Based Animation

The ontology can serve to connect simulation variables to visualization parameters resulting in dynamic 3D visualizations based on simulation behavior. By adding influence attributes to ontology concepts, a designer can link the value of one attribute to another by mapping between linear ranges during simulation execution. To create the

semantic link in the ontology, attributes `influence source`, `influence destination`, `influence source range`, and `influence destination range` are required. Ranges can be bounded by either vector or scalar values. A list of suggested attributes that could be used to create and tune influences is shown in Table 3-1.

Adding influences should be supported during simulation execution, resulting in the desirable feature of immediately seeing the effect of mapping a simulation variable to a visualization parameter (Shneiderman and Plaisant, 2005). Combining influences with adjustability (see Section 3.2.1) can result in a simple way for a designer to interactively fine-tune a visualization parameter to achieve a desired animation effect over time.

3.2.6 Designed Ontology Representations

In some cases, visualization of the graph-based ontological structure, with concepts as nodes and relationships as edges, may be useful on its own with few additional enhancements (Katifori et al., 2008). For example, concepts could be color coded or relationship visualizations could be set to an edge width that maps to a simulation coefficient. Special attributes can be defined to support the graphical customization of the ontology visualization. This is required to differentiate between what is an additional graphical element (to be visualized along with the ontology) and what graphical elements are to replace or augment the standard ontology components visualization. A way to allow the designer to delineate between these two options is to allow a set of attributes with a prefix to determine that it refers to the graphical properties of the ontology visualization itself. For example, the prefix `presentation`

could be used. In this case the attribute `presentation mesh` refers to the 3D mesh used to represent the concept and the attribute `mesh` would refer to the mesh assigned to the concept that is still visible even if the ontology is hidden.

3.2.7 Ontology Guided Particle Systems

A particle system is a collection of duplicated 2D or 3D graphical objects that are animated and subjected to external forces over time. Particle systems are used in real-time computer graphics to simulate a variety of effects (Hastings et al., 2007), including fluid flow and lighting effects. Because of their increasing application, tools for the creation of particle systems are included in most modern game engines and 3D modeling packages. Particle systems can be useful when visualizing simulation output and are an important consideration when designing general purpose visualization software. The graph-based ontology visualization can serve as an interface handle to connect particle appearance and dynamics to simulation variables.

The ontology concepts can serve as particle emission points. From the emission points, particles can be ejected in a certain direction or follow the path defined by the curvature of an out-going relationship edge. Particle properties (e.g., speed, size, color) can be set by adding attributes to the appropriate concepts and relationships. Particles that follow the path of a relationship can have constant properties derived from attributes assigned to the relationship, or have dynamic properties that change as the particle moves along the relationship (e.g., the color of the particle could fade from blue to red overtime). Dynamic properties can be encoded by attaching attributes to both source and destination ontology concepts. For example, if an attribute `particle size` is attached to both the source and destination concepts of a relationship, then the

size of the particle can be can linearly interpolated as it traverses the connecting relationship, thus causing the particle to shrink or grow overtime.

The interpolation of particle behavior along a relationship can be facilitated by checking attributes types on relationships and adjacent concepts. If a particle attribute is not of variable type (i.e., numerical), then it can be assigned the name of an attribute assigned to adjacent concepts. In other words, the particle property is either assigned a constant numerical value at the relationship level, or bound to adjacent concept attributes and subjected to interpolation across the relationship. This is illustrated in Figure 3-6, where the particle property `particle speed` is added as an attribute to a relationship and assigned the value `speed`. Because this attribute is not of variable type, the value for particle speed is calculated by considering the numerical values of the attribute named `speed` that should be present in the two adjacent concepts. A collection of suggested particle attributes are described in Table 3-2.

3.2.8 Attribute Expansion

Each concept has a collection of attributes. These attributes may be of varying interest to someone consuming the final visualization module. As part of the proposed framework, two styles of attribute visualization are defined: docked and expanded. Attributes that are docked are presented along with all other attributes within a list configuration. This representation of attributes should be of the lowest detail, taking up the least space. For example, if an attribute is an image, then when it is docked it should only be displayed as the image name, or the name accompanied by a small thumbnail. If the attribute is a simulation variable, then when it is docked it can be

displayed as its real-time numerical value, and an interactive 2D variable plot when it is expanded.

If presented with a collection of docked attributes after selecting a concept or relationship, a designer or consumer can make a choice as to which attributes are of interest, and the interesting attributes can be expanded. Generally, an expanded attribute should be represented at a higher fidelity than docked attributes within the visualization while maintaining a visual link to the concept. An image attribute, when expanded, could be displayed as the full image along with an edge pointing to the concept. This image could be moved about the screen space on a 2D plane parallel to the camera's view plane and centered at the concept's 3D position. This infinite plane can be thought of as the concepts "display space" where all expanded attributes are visualized. In Figure 3-7, the dashed arrow depicts the process of expanding a docked attribute. Figure 3-8 shows a schematic relating an attribute's display space and the 3D view.

3.2.9 Ontology-Based Interaction Modalities

A simulation and visualization designer will most often utilize semantics during the design phase that are irrelevant to the end consumer (e.g., co-worker, student) and, if visualized, would result in unnecessary clutter. End consumers also have no need to add or remove attributes due to their more passive relationship with the visualization and simulation that they are consuming. Further, transforming the visualization objects in a designed module is unnecessary and may confuse users presented with transform widgets, and relationship curvature controls. To this end, two interaction modalities are defined and supported by the ontological structure: a designer modality and a viewer modality.

Basic interactions should be supported across both modes. These interactions include: 3D camera controls, selection of concepts and relationships, attribute expansion, and the adjusting of attributes that are marked adjustable. Within the design modality, additional interactions should be supported, including: the ability to add and remove concepts, the ability to add and remove relationships between concepts, the ability to transform concepts (and any assigned visualization attributes), the ability to sculpt the curvature of any relationship, the ability to add and remove attributes of concepts and relationships, and the ability to view and modify all meta-attributes.

3.2.10 Semantic Styling

Most ontology visualization efforts provide a 2D or 3D node and edge based approach with minimal visual diversity or customization tailored to the domain from which the ontology was derived (Pietriga, 2006). To address this, graph style sheets were created that map the semantics embedded in an ontology into meaningful graphical representations. One example of this is STOOG (Style-sheet-based Toolkit for Graph Visualization), created by Artignan and Hascoët (2010), which provides an interface for users to define 2D icons to represent ontology concepts and to specify which attributes are rendered in the final ontology visualization. GraphViz (Ganser and North, 2000) was leveraged in STOOG for automated graph layout.

In simulation modeling, graph style sheets can be used to tailor the ontology representations to be consistent with the visual simulation modeling practices used in a given domain. Many graph-based simulation modeling paradigms exist and leverage a different set of visual syntax (Page, 1994). Being able to express dynamics in a visual paradigm familiar to a practitioner could help ease the visualization designer into the 3D virtual environment. In the given scenario, a practitioner could use a customized

version of the proposed framework with a graph style-sheet applied that was created based on the models used in that domain. Figure 3-9 demonstrates a simple transform from a graph-based ontology representation with two concepts and a relationship into that of a systems dynamic model with *source*, *flow rate*, and *level* objects.

3.2.11 Ontology Portability through RDF

The RDF (Resource Description Format) was created to support the envisioned Semantic Web. The abstract RDF structure can be understood by machines and humans. An RDF object consists of a subject, an object and a predicate (Miller, 1998). For example, in the statement 'visualization is a topic of this dissertation,' in RDF the subject is 'visualization,' the object is 'dissertation' and the predicate is 'is a topic of.' This RDF format maps cleanly to the ontological structure, which is composed of concepts connected by unidirectional relationships. Attributes in an ontology can be stored in RDF as terminal objects. Due to the fact that the ontological structure is augmented in the proposed framework, with meta-attributes and attributes attached to relationships, a simple translation phase is required to map the augmented ontology into the RDF standard.

To perform the translation from the augmented ontology, concepts and relationships can be elevated one level. This elevation will create concepts from elements that were previously attributes, and create attributes from elements that were previously meta-attributes. A technique to account for attributes of relationships is to morph the relationship into a concept that connects to the previous destination concept and has all the attributes attached to the original relationship (Gomez-Perez and Corcho, 2002). Objects of this type are referred to in RDF as "blank objects" because they have no reference to online semantic repositories but serve to link various well-

defined semantics in scenarios that demand a more complicated structure than the simple object, predicate, subject triple can provide. A depiction of the meta-attribute and relationship attribute morphing is depicted in Figure 3-10.

Depending on the desired application of the ontology, this mapping for meta-attributes and relationship attributes may obfuscate the core domain semantics. To address this, instead of performing a mapping, a supplemental file structure could be defined that houses all semantics defined in the framework that do not map cleanly to the base ontological structure. This file could be exported along with the base ontology for permanent storage and interoperability.

3.3 Technical Requirements

In this section, the high-level technical requirements will be presented. This will help summarize the proposed framework and determine the various modules required to implement the simulation model builder and visualization designer defined in this chapter. The framework requires the following components: an ontology parser, an ontology exporter, a simulation solver, an ontology solver, a graphics renderer, a designer user interface, and viewer user interface. An ontology pruning component and a domain styling component are optional.

The *ontology parse*r serves to load the ontology, encoded in a certain format (e.g., RDF), from a file. The parser should create the necessary internal ontology data structures in memory to be used by the other components of the framework. The *ontology exporter* saves the ontology for later use or use in other ontology-based frameworks.

To execute any simulation model, a *simulation solver* will need to be created. The simulation solver should: read the current values of simulation variables from the

ontology; perform the necessary calculations, perhaps domain specific calculations; and then write the new values for the simulation variable back into the ontology. There should also be an *ontology solver* whose purpose is calculating the values of any influences and solving any attributes that are equations. Any variable, simulation- or visualization-oriented, that is not listed as just purely numerical (i.e., it references another attribute by name), will have its value resolved by the ontology solver. The simulation solver can be updated at any interval but will need to know the time since the last update (i.e., the simulation time-slice). The effect of large time-slices when solving certain simulation models should be considered, because large time-slices can be problematic in models with sensitive coefficients.

For graphics rendering, a component dedicated to the rendering of 3D textured objects is required. This *graphics renderer* component can be coded in a low-level graphics language, or one utilizing a higher-level graphics library. The graphics component will need to communicate with the ontology solver to obtain any ontology concept attributes linked to graphical properties within the visualization scene. These properties range from the standard affine transforms applied to meshes, to the speed or color of particles as they move along a relationship's curved path. The graphics rendering component should be updated at an interval conducive to smooth animation and high-user interface responsiveness (e.g., 30 times a second or greater). Ideally, the simulation solver will run in a thread separate from the graphics rendering to ensure the highest precision simulation possible (i.e., have the simulation time-slice not be limited by a graphical refresh rate).

The final required components pertain to user interaction. For simulation and visualization design, there will need to be a *designer user interface*. For consumption a, *viewer user interface* will need to be created. The distinction between these interfaces is defined in Section 3.2.9. The optional *ontology pruning component* (see Section 3.4.4) will interface with the ontology importer and allow users to hand pick concepts to be placed in the 3D design environment. The optional *semantic styling component* (see Section 3.2.10) interfaces with the graphical renderer and dresses the basic ontology representation with graphical objects used in abstract simulation modeling in a given domain. The entire architecture described in this section is illustrated in Figure 3-12. In the architecture, the ontology component serves to store the current state of the ontology in memory.

3.4 Methodology

In this section, the high-level methodology is presented. The methodology is defined to provide a step-by-step account of how the theoretical tool defined earlier in this chapter could be leveraged by a designer. There are three primary steps to the methodology: ontology acquisition, modeling building, and execution.

3.4.1 Ontology Acquisition

To leverage an ontology visualization as an interface affordance, one must first obtain an ontology encoded with the desired semantics. Naturally, there are two ways to fulfill this requirement. The first is to seek out a pre-existing ontology which may be published by experts in the given field. For example, there is the Modelica ontology (Pop et al., 2004) that supports simulation of physical systems (e.g., those containing electrical, thermal, or hydraulic components). Other pre-existing ontologies include those that are intra-organizational and often not publically-accessible. These ontologies

enable knowledge sharing and reuse across departments within a single organization (Blomqvist and Öhgren, 2008). The second option is to simply construct the ontology oneself. In turn, the newly constructed ontology can be used as a starting point in future efforts.

If a pre-existing ontology is employed, then it should be analyzed and modified if the semantics required by the modeler are not already present. Examples of modifications include adding new sub-concepts (i.e., adding semantic precession) necessary for the desired simulation and visualization, adding entirely new concepts, or adding relationships between pre-existing concepts.

3.4.2 Simulation Model Building and Visualization Construction

To eventually execute a simulation and corresponding visualization, system dynamics need to be defined. To achieve this, relationships specific to simulation dynamics should be added between existing concepts in the ontology, if they are not already present. If more than one relationship type can denote dynamic flow, then the specific dynamics of interest need to be defined to avoid ambiguities when it is time to execute the model.

Concurrently with simulation model building, expressive 3D visualization can be created by adding visual parameters as attributes to ontology concepts. Examples of visualization attributes include 3D meshes and materials. When a visualization attribute is added to an ontology concept, it should be rendered in the ontology visualization and rooted at the node of the given concepts. Further, influences can be added to ontology members to link visualization attributes to simulation variables (e.g., attribute A *influences* attribute B, where A is a dynamic variable and B is a visualization parameter).

As a final construction requirement of the methodology, the graph-based ontology visualization itself can also be modified to express certain ideas. This requirement supports integrative multimodeling. For example, the visualization of a relationship denoting dynamic flow can be morphed into a curve that traces the path of the flow within the 3D visualization environment.

3.4.3 Simulation Execution and Visualization Animation

A correctly formed model can be executed once the needed ontological concepts are established, and the simulation model and visualization attributes are defined within the ontology. The method of execution will depend on the mathematical paradigm in which a given domain-ontology exists. For example, in some physiology simulation models, blood flow is represented by the hydrodynamic metaphor where blood volume in compartments is updated by calculating pressures, resistances, and valves between adjacent compartments. The execution requirement implies that the specific solver is connected to the interface and is coded depending on the desired application domain. As the defined simulation model is executed, simulation parameters should be updated at each time interval. In turn, visualization parameters that are linked to simulation attributes through influences are updated. This results in a real-time 3D animation based on simulation output.

3.5 Classification, Comparisons and Limitations

3.5.1 Classification

In Table 3-3, the proposed usage of ontologies in the interface is codified according to the classification system presented by Paulheim and Probst (2010). The ontologies that are well-suited for the proposed methodology leverage “real world” semantics, but there is no semantic complexity requirement on them. The visualized

ontology can be leveraged during design time (e.g., to build a dynamic model) or run time (e.g., to label components and denote relationships). Further, ontologies are visualized as graphs, and textual labels serve to “verbalize” ontology members. Finally, in the proposed methodology, interaction is allowed as a means to change which ontology members are visible and to modify the ontology within the visualization environment.

3.5.2 Comparisons

A core principle of integrative multimodeling is that the synthesis of multiple models may support new visualization and interaction techniques beneficial to the areas of design and education. This synthesis can be achieved in the human-interface layer, as is the case with two notable integrative multimodeling efforts: the work of Park and Fishwick (2005) (see Section 2.3.2), and the work of Quarles (2009) (see Section 2.3.3). to further articulate the proposed methodology, the remainder of this section presents a comparison of these previous works against the proposed ontology-centric approach, as well as a classification of the particular usage of ontologies the proposed methodology will employ.

Park and Fishwick (2005) developed an integrative multimodeling framework that leveraged ontologies. They created a holistic ontology composed of classes pertaining to concepts in simulation, visualization, and the domain of interest (e.g., military, chemistry). Portions of the ontology were represented in the interface as lists. This enabled users to fill-in some required information (e.g., the name of a “.obj” file or type of model: Functional Block Model or Finite State Machine). However, the ontology was not displayed within the visualization environment, leaving the integrative visualization to be composed solely of graphical and dynamic models. The proposed methodology

builds on this framework by placing the domain ontology within the integrative visualization along with the dynamic and graphical models. I believe this placement represents the most important aspect of what is presented in this dissertation: the ontological structure can serve as the foundation for a new interface model supporting simulation model building and visualization design activities. The Park and Fishwick framework provides an environment for model blending (i.e., the blending of graphical models with dynamic models), but model design and execution required the traditional engineering edifice such as Blender Game Engine logic blocks and Python scripts.

Quarles (2009) showed the Augmented Anesthesia Machine (AAM) to be effective in some areas of training. However, no formal method was provided to create similar style visualizations for different processes. The proposed methodology is a first step toward defining a conceptual framework for building tools such as the AAM. With domain ontologies manifested in the visualization environment, dynamic models can be constructed by supplementing the domain knowledge with new “dynamics” information and visualizations can be built by adding visual attributes to the members of the ontology. The 3D collocation required by the AAM will come naturally in the proposed methodology because the abstract knowledge (located in the ontology) and dynamic model are used as interface affordances and, therefore, are manifested in the 3D visualization environment from the start.

3.5.4 Limitations

It is important to consider the potential limitations of the theoretical framework described in the previous sections. Thus far, I believe there are two major limitations: one pertaining to parsing and visualizing large, pre-existing domain ontologies; and

another pertaining to simulation complexity of models outside of the realm of equation-based and continuous.

Domain ontologies can grow quite large. For example, the Foundational Model of Anatomy (FMA) ontology contains over 75,000 concepts and over 2,000,000 relationships. While there is active research in automated and partially-automated large graph layout (e.g., see the work of Diaz et al. (2002)), and there is research into what makes a graph easily comprehendible; large graph structures are still widely considered hard for humans to comprehend outside of just a "big picture" understanding of a network (von Landesberger et al., 2011). Users of the proposed framework will need to delve into deep semantic detail to build executable models and visualizations with a desirable level of precision. Due to these restraints, there will most likely need to be a pruning phase in which a subset of semantics are chosen from the larger ontology, and then this subset is imported into the framework.

The semantic pruning phase can be handled in many ways. External ontology editing tools (e.g., Protégé) could be used to edit a large ontology down to the desired concepts and relationships, or the framework could be modified to include an ontology representation that is not 3D. The latter option may be more desirable because pruning could occur at any time in the design process and within the same interface in which the design activities are taking place. The 2D text-based hierarchal list (leveraged in Protégé and the FMA online viewer) could provide a clean interface into a vast ontology, allowing users to choose which concepts to "drag" into the 3D design environment. A high-level sketch of such an interface is depicted in Figure 3-11.

Regarding simulation model complexity, there are certain domain modeling paradigms that map cleanly to the base ontological graph structure (e.g., compartmental modeling, system dynamics, finite state machines). These model types are often flow-based. There are less constrained model types, however, that do not have such a straight forward semantic map onto an ontological graph. One such hard-to-map model type is the rule-based model. Consider the simulation and visualization of a manufacturing assembly line. In manufacturing simulation, there are many factory components driven by different rule sets. Further assembly line visualization can be very intricate. For example, there may be conveyor belts, robotic arms and human workers. The dynamics between human workers and other objects are particularly difficult to animate because the human may interact with another object, changing the position and orientation of the object as the human holds it in their hands. Complex, multi-model environments such as the manufacturing assembly line may be best served by other simulation platforms that are coded with the appropriate functionally and graphical representations.

3.6 Summary

In this chapter, an interface theory was presented along with a framework to house the theory and define a software tool that could be implemented. The central structure of the proposed theory is a visualized 3D ontology. The three main structures of the ontology (concepts, attributes, and relationships) all play an important role in the theory. These roles were detailed along with 11 semantic and structural affordances of the visualized ontological structure with respect to simulation modeling and visualization design. Further, a methodology was provided to describe how a designer would use the theoretical tool. Finally, the tool was classified using the Paulheim and Probst (2010)

scheme for describing the role of ontology in the user-interface, and theoretical limitations of the tool were discussed.

Table 3-1. A list of suggested attributes for assigning the influence between simulation variables and visualization parameters

Attribute name	Attribute value
`influence source`	The name of the simulation attribute to drive the influence.
`influence source min`	The minimum source value permitted for the influence. This value can be scalar or 3D vector. Values less than the `min` value should be clamped to the `min` value. This attribute must be of the type "variable."
`influence source max`	The maximum source value permitted for the influence. This value can be scalar or 3D vector. Values greater than the `max` value should be clamped to the `max` value. This attribute must be of the type "variable."
`influence destination`	The name of the attribute which will be influenced.
`influence destination min`	The minimum destination value permitted for the influence. This value can be scalar or 3D vector. Values less than the `min` value should be clamped to the `min` value. This attribute must be of the type "variable."
`influence destination max`	The maximum destination value permitted for the influence. This value can be scalar or 3D vector. Values greater than the `max` value should be clamped to the `max` value. This attribute must be of the type "variable."
`influence bounds`	The type of bounding geometry to define the region of influence.
`influence radius`	The scale of the bounding geometry. This attribute must be of the type "variable."
`influence falloff`	The type of falloff calculation applied to influence vertices based on their distance from the central influence point (i.e., the 3D location of the parent concept).

Table 3-2. A list of suggested attributes for assigning particle systems properties to ontology concepts and relationships.

Attribute name	Attribute value
`particle emission`	The numerical value or name of the variable to determine the rate of particle emission from the source concept.
`particle emission factor`	A numerical value that can be used to scale the particle emission. This attribute must be of the type "variable."
`particle color`	The color of the particles. This attribute may be linked to other attributes' values either within the relationship, or in source and destination concepts. If the color value is to be hard coded, then this attribute must be of the "variable" type and of the "vector" type.
`particle image`	An image used to render a single particle. This image may be tinted by the value of `particle color`.
`particle speed`	The speed at which the particle is animated (e.g., travels along a relationship curve). This value can be provided in unit lengths per second, where the unit length is determined within the 3D environment.
`particle random offset`	A numerical value to serve as the upper bound of a random offset of a particle from a predefined path. A random unit vector can be calculated and multiplied by a random value from the range [0, <`particle random offset`>] to find the 3D position of the offset from a point along a relationship curve.
`particle wander interval`	A numerical value to represent the time, in seconds, that particles will take to "wander" from one random point to another in the region defined by the particle random offset.

Table 3-3. The characteristics of ontologies well-suited for use in the proposed framework. This classification scheme was defined by Paulheim and Probst (2010).

Characteristic	Classification
Domain	Real world
Complexity	Informal, low, medium and high
Usage	Design time and run time
Visualization	Graphical and verbalized
Interaction	View and edit

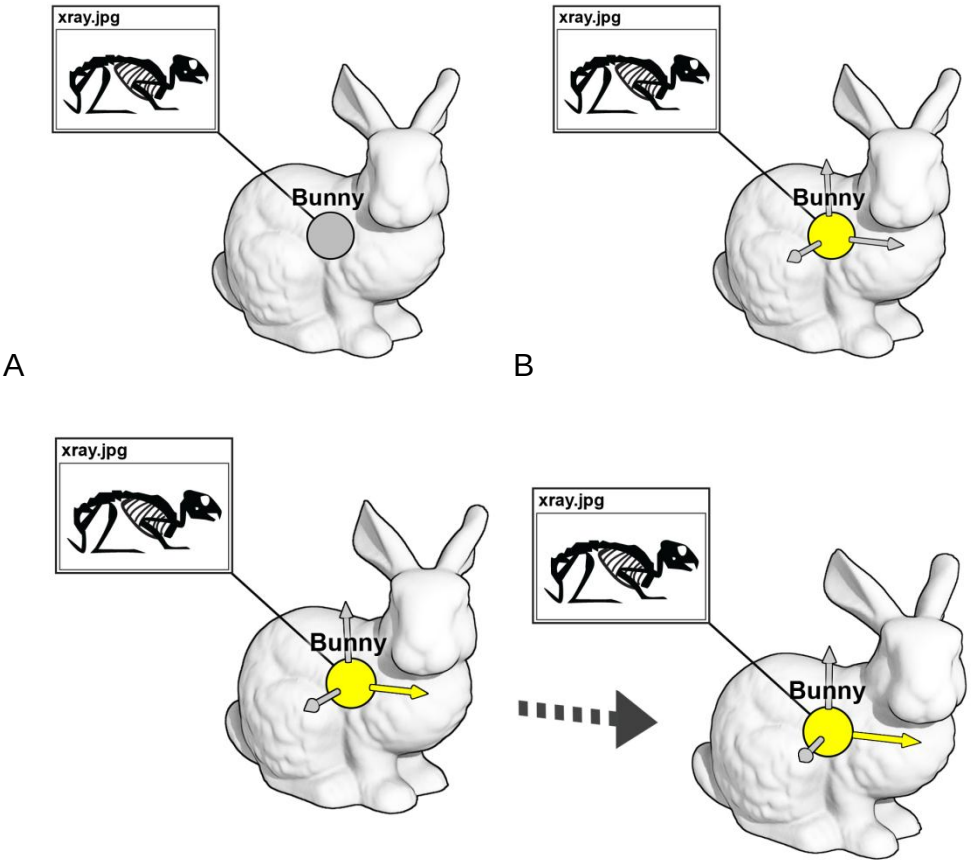


Figure 3-1. A concept Bunny with graphical attributes of a 3D mesh and a 2D image (xray.jpg). A) A sketch of a concept with a mesh and image attribute. B) A translation widget appears rooted at the concept when the mesh is selected. C) The translation of a concept when an axis of the translation widget is grabbed and dragged. The mesh is transformed along with the concept and the image maintains its 2D offset.

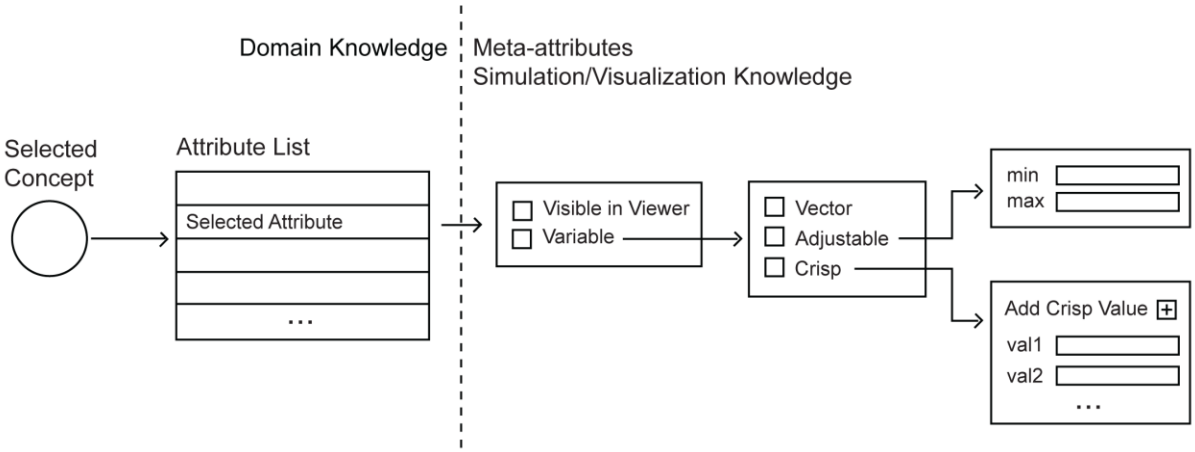
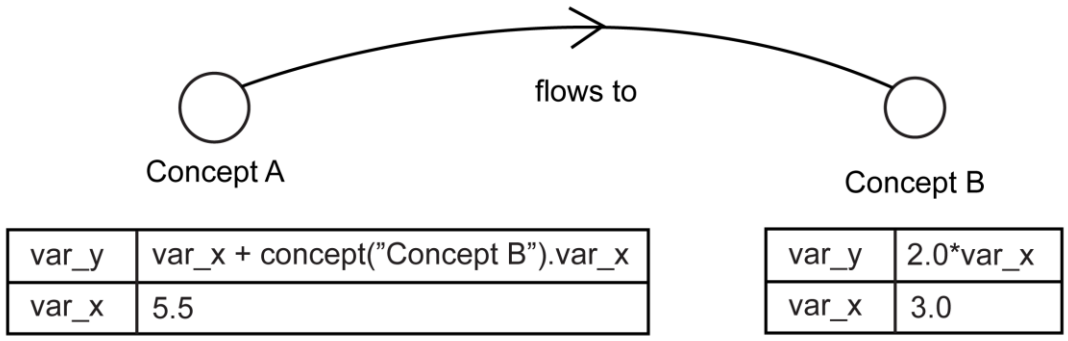
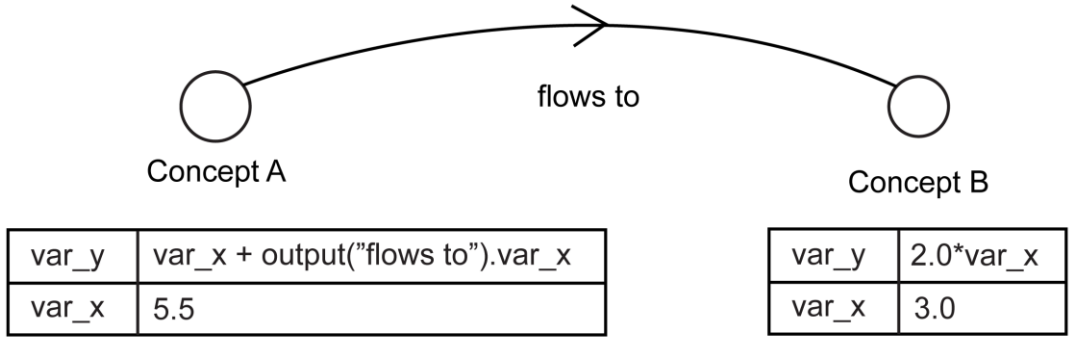


Figure 3-2. A high-level sketch of a user interface to meta-attributes.



A



B

Figure 3-3. Two different syntax approaches for accessing attributes across multiple concepts within equations.

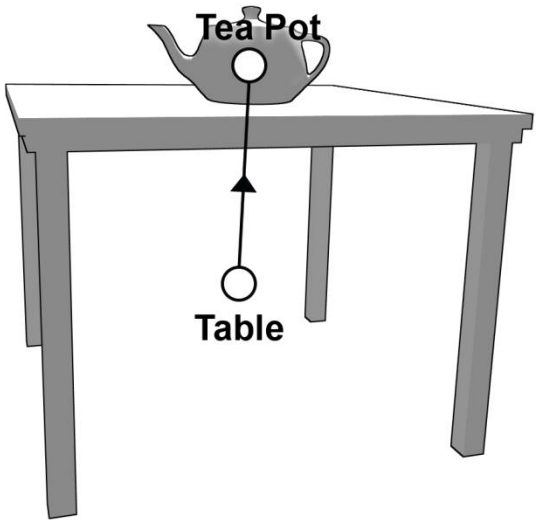


Figure 3-4. A simple 3D scene with an ontology visualization serving as an interface to mesh transformations. The relationship between “Table” and “Tea Pot” is *has a* and, therefore, when the table is transformed, the tea pot will be as well, treating the Table concept location as the origin point.

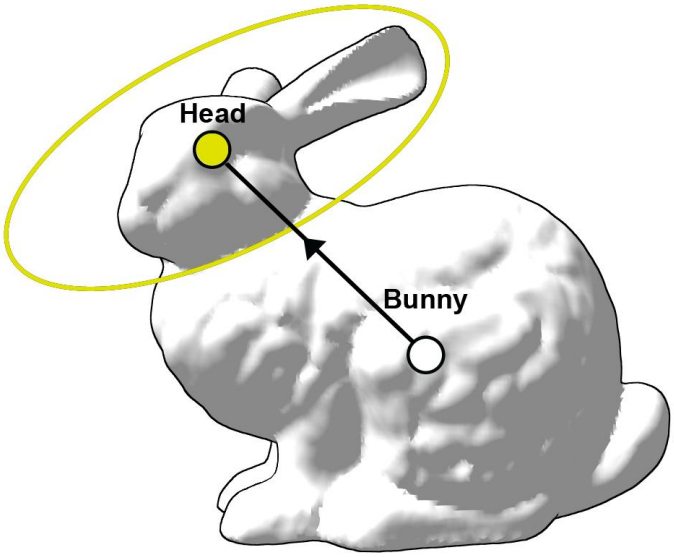


Figure 3-5. A simple illustration of using a *has a* relationship to form vertex groups. The Head concept is assigned a sphere as bounding geometry. This sphere was then transformed by a designer to encompass the desired set of vertices.

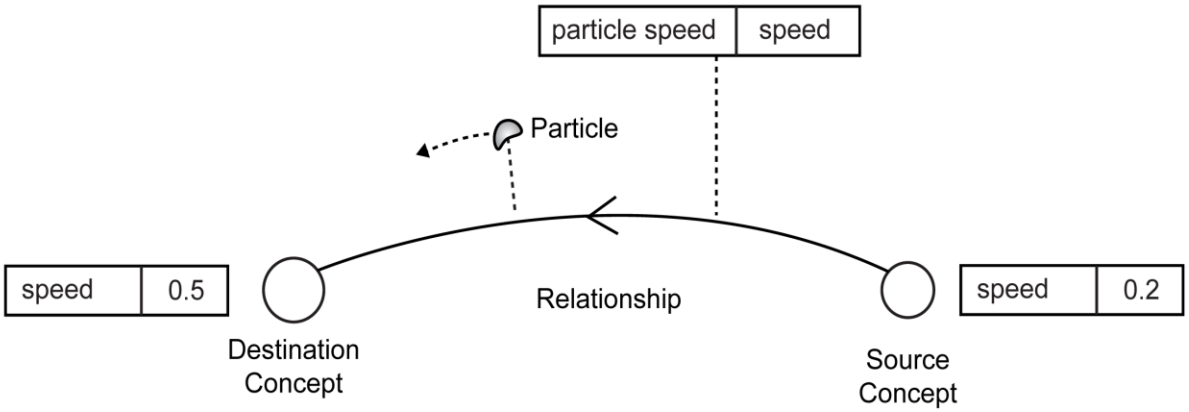


Figure 3-6. A depiction of a particle traversing a relationship curve. In this example, the particle speed is determined by interpolating between the speed attribute assigned to the source and destination concept.

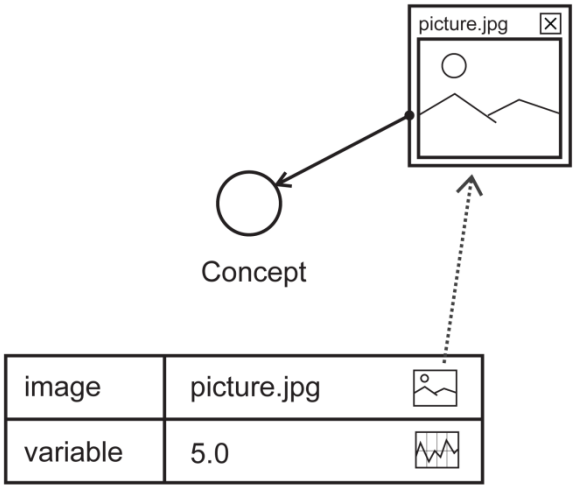


Figure 3-7. The action of expanding an attribute. The attribute `image`, represented by the image file name and a thumbnail in the attribute list, is dragged in to the 3D environment and then represented by a higher resolution version of the image that maintains its own visual link to the concept (the edge connecting the image and the concept).

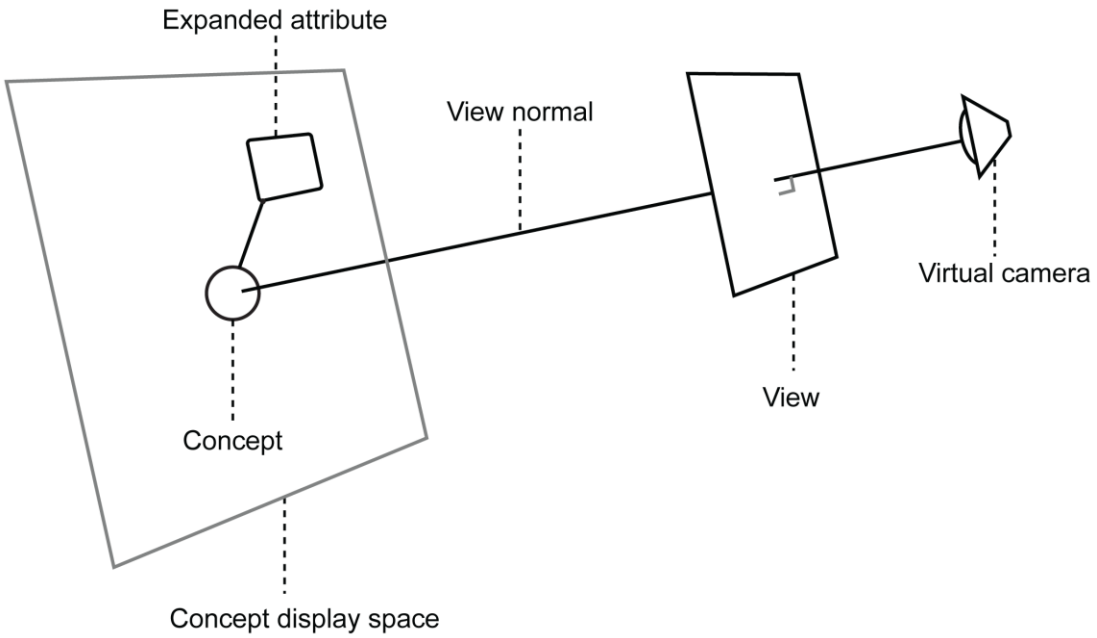


Figure 3-8. A depiction of a concept's display space. The display space is a plane parallel to the view plane that should hold all expanded attributes of a given concept.

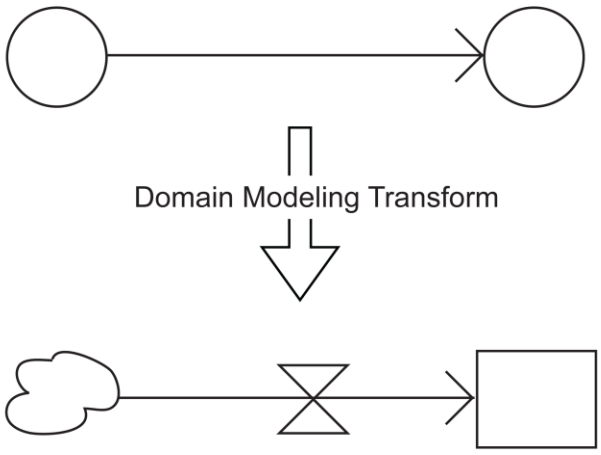


Figure 3-9. A simple example of applying a graph style-sheet to an ontology to create a graphical representation used in systems dynamics modeling. The concepts shown above should contain domain-appropriate semantics to be interpreted by the style sheet. The system dynamics model contains a source (source concept), a flow (the relationship), a rate (the hour glass icon), and a level (destination concept).

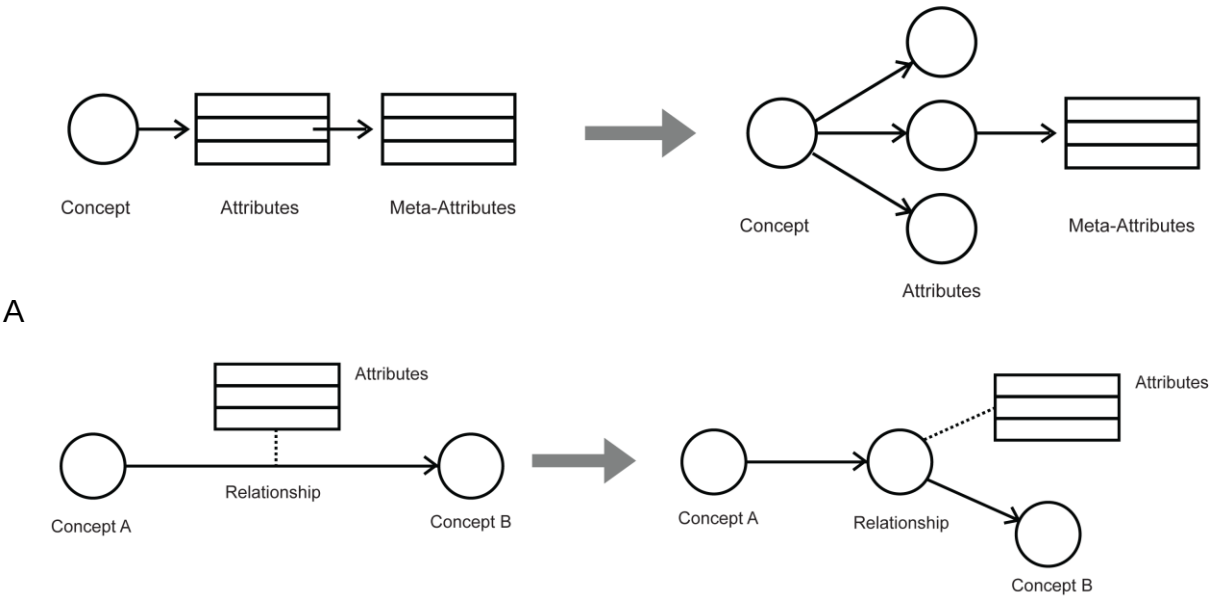


Figure 3-10. Different approaches to morphing the ontology structure used in the prototype into the standard "triple" structure. A) A sketch of the morphing of an ontology supporting meta-attributes to a standard ontology with just concepts, relationships, and attributes. B) A sketch of the morphing of an ontology supporting relationships with attributes, to a standard ontology.

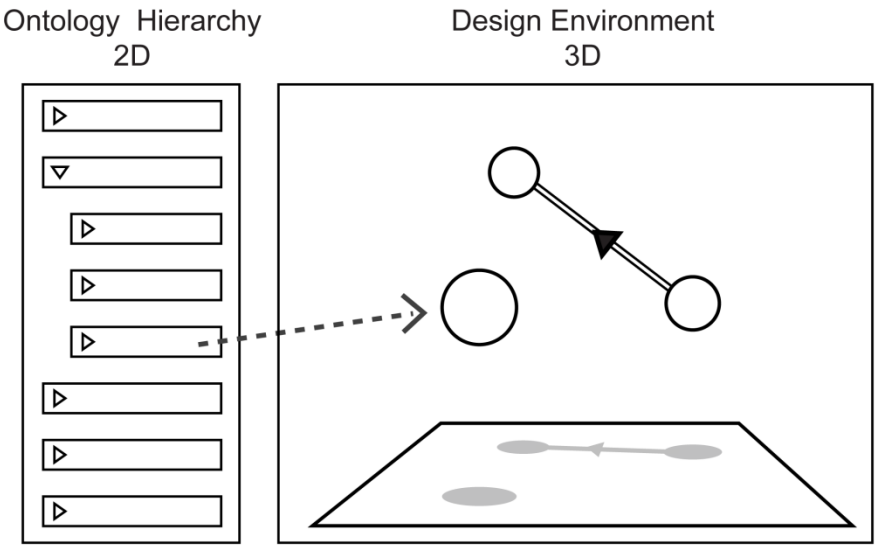


Figure 3-11. A sketch of a potential ontology pruning interface where concepts can be dragged into the 3D design environment from a 2D list-based view.

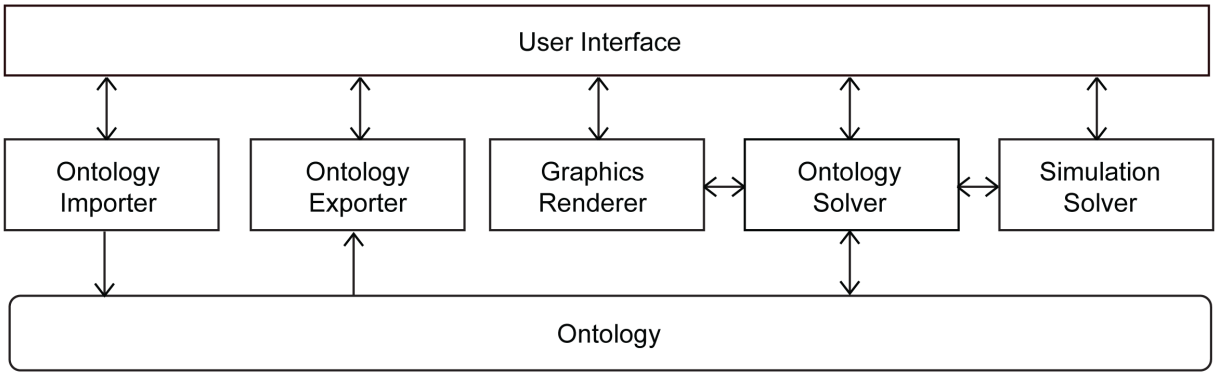


Figure 3-12. A high-level diagram depicting the interfaces between the various required components of the proposed framework.

CHAPTER 4
CASE STUDY

A software prototype of the theoretical tool defined in Chapter 3 will be presented in this chapter. To demonstrate key functionality of this prototype, a simulation and visualization was constructed that showcases the important features. To create this case study, the Beneken (1965) cardiovascular model was chosen as the base simulation model from which to build an interactive, integrative 3D visualization of the human cardiovascular system. In Section 4.1, the model will be described in detail along with some of the model's lineage and reasoning for why this model was chosen. The software prototype and the construction of the simulation and visualization are presented in Section 4.2. In Section 4.3, preliminary human factors are considered, including an example lesson based on the resulting simulation and visualization that was created by a medical educator.

4.1 Cardiovascular Modeling

Many credit Euler (1775) with the creation of the first mathematical model to simulate blood flow. Weber (1850) is often credited for creating the first circulation model to employ the hydrodynamic metaphor (i.e., using valves and pumps). However, this introductory discussion will be limited to the history of modeling efforts in cardiovascular physiology in the computing age.

One of the earliest examples of leveraging formal modeling techniques and computer technology to simulate a physiological process is the work done by Warner (1959). He employed the power of the electronic analog computer to create two novel simulations: one of the carotid sinus (an area of bifurcation in the carotid artery), and one of the circulatory system. The latter is worthy of further explanation due to the

implementation of a compartmental model and the adherence to a simulation design process that is used today on modern computers. The "block diagram" created by Warner is shown in Figure 4-1. A set of equations are used at each block to solve for blood pressure and blood volume. Because this model was implemented on an analog computer, each equation had to be built as an electric circuit and the continuous outputs of these circuits were fed into oscilloscopes for visualization.

4.1.1 The Beneken Model

Soon after Warner published his analog-computer-compatible circulation model, Beneken submitted a compartmental hydraulic-metaphor model as his PhD thesis (1965), which was also primed for analog computation. Beneken found the functions that drive the beating of the heart in the Warner model to be problematic because they transition too abruptly from systole (contraction) to diastole (relaxation). As a result, the Beneken model contains a smoother cardio controller function. The model Beneken created is shown in 4-2. The typical set of equations for a compartment in the Beneken model account for blood pressure, flow, and change in volume. The pressure, p, in a compartment at time t is defined by,

$$p(t) = E(v(t) - UV) \tag{4-1}$$

where E is the elastance of the compartment (a time-variant function for compartments representing heart chambers, a constant for the other compartments), $v(t)$ is the volume at time t of the compartment, and UV is the unstressed volume of the compartment.

The flow,f, into a compartment at time t is defined by,

$$f(t) = \frac{p_{in}(t) - p(t)}{R} \quad (4\text{-}2)$$

where $p(t)$ is the pressure of the compartment at time t, $p_{in}(t)$is the pressure of the input compartment at time t , and R is the resistance between the compartments.

The change in volume in a compartment, $\frac{dv}{dt}$, at time t is defined by,

$$\frac{dv(t)}{dt} = f(t) - f_{out}(t) \quad (4\text{-}3)$$

where $f(t)$ is the flow into the compartment, and $f_{out}(t)$ is the flow out of the compartment.

There is also influence from inertia between the concepts intrathoracic arteries and extrathoracic arteries. For a complete description of the remaining equations and coefficients, consult Beneken's thesis (1965).

4.1.2 The Goodwin et al. and Sá Couto et al. Models

The Beneken model is still used today as a foundation for new models used in simulations for training scenarios. One such model is for infant cardiovascular physiology. Goodwin et al. (2004) tuned the coefficients defined in Beneken's adult circulation model to estimate the circulation of an infant. As part of the tuning, for example, the total blood volume was decreased from 4740 milliliters to 685 milliliters and the heart rate parameter was increased from 72 beats-per-minute to 129 beats-per-minute. Other coefficients, such as arterial and venous resistances, were changed as well.

Also using the Beneken model as a mathematical foundation, Sá Couto et al. (2006) built models representing four pathologies. Many pathological models share the same compartments as the base physiological model and differ only in connectivity of

compartments and placement of resistances and valves. The four pathologies modeled by Sá Couto et al. are shown in Figure 4-3.

4.2 Case Study

4.2.1 Software Prototype

A software prototype needed to be created to demonstrate the proposed interaction theory using a real-world example. The prototype was coded in C++ and leverages the Open-source Graphics Rendering Engine (OGRE, 2012) for managing the rendering of textured 3D geometry. OGRE was chosen because it is open-source, leverages either the DirectX or OpenGL hardware accelerated graphics libraries, and supports the powerful and fast programming language C++. The prototype runs on computers using the Windows 7 operating system. A depiction of the architecture of the software prototype is shown in Figure 4-4. This architecture is one possible instantiation of the high-level architecture presented in Chapter 3. In the prototype architecture, the “simulation solver” component is implemented as a compartmental model solver because this is required to execute the Beneken model. Another implementation choice was to encode the ontology in the XML format (notice the XML importer and exporter components in Figure 4-4). XML was chosen because of its universal adoption, and because it can be easily understood by future researchers continuing work on this topic.

A screenshot of the created prototype is shown in Figure 4-5 with interface components labeled. A concept named “Concept ” is placed within the visualization environment and selected so a translation widget is displayed rooted at the concept. The *Simulation Control* allows users to play, pause, reset, and change the time-scale of the simulation. The time-scale ranges from 0.01 (one one-hundredth real-time) to 1.0

(real-time). The *Importer* imports ontologies encoded in the XML format. Conversely, the *Exporter* exports ontologies from the prototype into the XML format. Concepts are created using the *concept creator* by entering the concept name in the text field and then pressing the "+" button. There is no relationship creator because relationships are created in the prototype by the user holding the left mouse button down on a concept and then dragging to another concept and releasing. Relationship types are determined through an attribute named `type` assigned to relationships automatically upon creation.

The *Visibility Settings* control the visibility of concepts and relationships. Within the visibility settings are: particle visibility that determines if particles are rendered; knowledge visibility settings that determine if the ontology structure is visible; a "show all" option to determine if concepts are visible even if they have no visible adjacent relationships; and a list of current relationships in the ontology with the option to set a given relationship type visible or change the color coding of the relationship type. The *Attribute Editor* displays all of the attributes of a selected concept (e.g., the selected concept "Concept" in Figure 4-5) along with the meta-attributes of the selected attributes (see the right portion of the attribute editor), and allows for editing of the attributes and meta-attributes by a designer.

4.2.2 Solving the Compartmental Model

The simulation solver used in this prototype is tailored to solve the Beneken model. With more development effort, this solver could be made to solve any compartmental model. For this case study development effort was spent towards showcasing the design affordances of the visualized ontology and to create a complete interactive, integrative visualization of the human cardiovascular system, as opposed to

creating a general tool. I consider the creation of a more general tool a promising direction for future efforts (see Section 5.2). The implemented solver looks for the required semantics as attributes attached to concepts and relationships in the scene. If these semantics are present when "play" is pressed in the Simulation Control, the simulation will execute and, in turn, the visualization will animate. The required attributes are shown in Table 4-1. The solver uses these attributes as coefficients to solve the necessary differential equations of the Beneken model using the forward Euler method. The solver runs in a thread separate from the main graphics rendering and user interface thread with a time-step of one millisecond.

4.2.3. Constructing the Executable Simulation and Visualization

In this section a step-by-step account of the process of constructing an interactive, integrative visualization of the human cardiovascular system is presented. This visualization will include a heart with four chambers that "beat" according to the value of the simulation variable `volume`. Particle systems will also be attached to the relationships of the simulation model to illustrate the dynamics of blood in the arteries and veins. The construction of the visualization will demonstrate the interaction a designer can have with the ontological structure in support of design activities.

To fully relay the interaction and visual dynamics involved in the design process described in this section, a video is provided as Object 4-1.

Object 4-1. Simulation and visualization design demonstration video (.wmv file 120 MB).

4.2.3.1 Constructing the base model

The base executable Beneken model is constructed in the prototype by adding the 10 required concepts (i.e., compartments in the compartmental model) using the concept creator, and connecting them with *flows to* relationships denoting the path of blood flow. The *flows to* relationship type must be used because the solver looks for this specific relationship type to solve the model. In the prototype, relationships denoted as *flows to* are automatically styled with a peach color, but any relationship can be colored with any color by assigning an attribute `presentation color` to the relationship and marking it as a variable that is of the vector type to provide red, blue, and green values. A snap shot of constructing the base model is shown in Figure 4-6. This snap shot was taken as the designer adds a relationship between the Pulmonary Arterial Tree and the Pulmonary Venous Tree concepts.

This model was constructed through collaborations with a medical educator, with the end goal being to deploy the constructed visualization in a classroom (see Section 4.3). The educator suggested changing several concept names to be consistent with current teaching terminology. This accounts for the difference in some of the concept names when comparing the model depicted in Figure 4-6 to the model depicted in Figure 4-3. Further, abbreviations are used in the prototype to minimize visual clutter. If a concept has an attribute named `abbr`, then the value of this attribute will label the concept in the visualization environment (e.g., `abbr = RV` is assigned to the Right Ventricle concept).

Figure 4-7 shows the completed Beneken model, with all the required concepts, relationships and attributes added. The Left Ventricle concept is selected and its

attributes are rendered as a list in the Attribute Editor. The required attributes of simulation concepts to properly solve the model are enumerated in Table 4-1. Some of the attributes are shown in the Attribute Dock in Figure 4-7 (flow, volume, pressure, elastance min, elastance max). One technique for the efficient creation of these attributes is to: create a super-concept, (e.g., “Beneken Chamber”); add the required attributes to this concept; and then attach this concept to the 10 concepts of the Beneken model with an *is a* relationship. This technique employs the taxonomic approach discussed in Section 3.2.3. A distinction should be made on the *is a* relationship so just the attribute types and names are propagated down and not the attribute values, as all 10 concepts require different values for these attributes. A way to achieve this implementation detail is to leave attributes blank whose value should be over-written. One could imagine, however, additions to the theory where attributes are added to the *is a* relationship to precisely define the nature of the attribute propagation.

Once the Beneken model is made executable, dynamics can be explored. Figure 3-8 shows the result of expanding the attribute `pressure` that is attached to the Left Ventricle concept. Once the attribute is expanded it is placed in its parent concept’s presentation space (see Section 3.2.8 for a full definition of attribute expansion and the display space). In this case study, variable attributes are rendered as numerical values when docked, and interactive plots when expanded. Expansion is done by the user by pressing the left mouse button on the gray pad to the left of the attribute names, and dragging this value into the 3D scene, and then releasing. Once the expanded display (the plot in this case) is in the scene, its 2D offset from its parent concept can be

changed by dragging the display with the right mouse button. The display may be closed by pressing the 'x' button in the upper right corner of the display.

4.2.3.2 Adding dynamic visualization

Once the simulation model is defined and its execution provides output that is deemed correct, visualization elements can be added to the scene to illustrate behavior of certain simulation variables. This is achieved by adding visual elements as attributes to the appropriate concepts, and then linking these visual properties to simulation variables through influence attributes. The visualization to be created for the primary case study shall include a beating heart and particles systems attached to the relationships between simulation concepts to illustrate the dynamics of blood flow.

To begin constructing the visualization, a mesh of the heart needs to be placed in the scene. To achieve this, a Heart concept is added to the scene to hold the mesh. A `mesh` attribute is added to this concept and assigned a value of `heart.mesh`, and a `material` attribute is added to this concept and assigned a value of `heart_new_mat`. Within the prototype, meshes and material must conform to the OGRE standard of .mesh and .material files. A 3D heart mesh was purchased online and converted to the .mesh format using the free 3D modeling and animation package, Blender. The material file `heart_new_mat` contains a simple definition of material properties that account for the red shading of the heart. This simple coloring may also be achieved by adding an attribute of `color` to the concept. The result of adding the Heart concept and the two visualization attributes (`mesh` and `material`) is shown in Figure 4-9.

After the heart mesh is added to the scene, it needs to be semantically linked to the simulation model to achieve the beating effect. This will require linking the heart to

its four chambers, all present in the Beneken model. These chamber concepts are the Left Atrium, Left Ventricle, Right Atrium and Right Ventricle. For demonstration purposes, the process of rigging the right atrium of the heart will be detailed. To start the process of rigging, the Heart is linked to the Right Atrium concept through a *has a* relationship. The designer then positions the Right Atrium concept near the perceived center of the right atrium of the heart mesh. The positioning of the Right Atrium concept is depicted in Figure 4-10.

There is a collection of vertices in the heart mesh that approximate the region of the right atrium. It is this vertex group that should be deformed according to the `volume` attribute of the Right Atrium concept to create a realistic animation of a beating heart chamber. A vertex group can be created in the prototype by adding a set of influence attributes (introduced in Section 3.2.5) to the Right Atrium concept. When all required influence attributes are present, the vertices bounded by the influence are highlighted. This is depicted in Figure 4-11. A sphere is used as the default bounding volume in the prototype, but as discussed in Section 3.2.5, more complex geometry could be used to bound sets of vertices. Quadratic falloff is employed so vertices closer to the boundary of the sphere (i.e., further away from the concept node), will be influenced less. This creates smooth boundary conditions for influence regions and, therefore, no hard edges will be created when the heart mesh animates. Other types of falloff could be employed and potentially defined through an `influence falloff` attribute. Table 4-2 lists all the attributes that are added to the Right Atrium concept to achieve the influence.

4.2.3.3 Sculpting relationships

Within a cardiovascular visualization, curved relationships can trace the path of blood flow and also serve as the basis of more advanced features. One feature is the ability to create an animation path for particle systems that not only traces blood flow, but visually demonstrates the oxygenation of blood, and the pulsation of blood through the veins and arteries. Using ontological relationships to represent an abstraction of three-dimensional flow requires relationships to be renderable as curves. To satisfy this requirement, 3D spline functionality was incorporated into the prototype. Splines are smooth interpolations of piece-wise linear polynomials (Bartels et al., 1983). To create relationship splines, designers add control points to a control hull of a relationship, and then translate these control points to immediately see the resulting curvature. This process is depicted in Figure 4-12, where the curvature of the *flows to* relationship between the Right Ventricle and Pulmonary Arterial Tree is defined. To create the complete visualization, all *flows to* relationships of the Beneken model should be sculpted. In Figure 4-13, the entire model is shown co-located with the heart mesh.

When implementing such a spline-based system, careful consideration must be paid to the frame of the curve. The frame of the curve is defined by the tangent, normal, and bi-normal vectors of the curve at any point (Yamaguchi, 1988). The frame is important because it is used to define the three orthogonal axes required to calculate any offset from the curve in 3D space. A simple illustration of the frame is shown in Figure 4-14A. Offset calculated using the frame can be used to give a relationship "width" (i.e., render it as a tube shape with a particular radius) or to define the offset of a particle within a particle system. Frames must stay oriented correctly across the curve to avoid any sudden jumps in the normal or bi-normal. Sudden jumps in the frame's

orientation will result in a twisting of a curve with any width, and the instantaneous jumping of offset particles from one side of the curve to another.

In the prototype, curved relationships are given width by extruding a square along the underlying path, which consists of a set of solved spline points. To keep the frames from twisting (i.e., keep the square extrusion from abruptly twisting around the solved path), the technique of rotation minimizing frames (RMF) is employed. With this technique, the tangent vector is calculated as the normalized displacement vector between adjacent solved points (Wang et al., 2008). A depiction of the tangent vectors between a series of solved points is shown in Figure 4-14B. The minimum rotation to transform one tangent vector to the next is determined, and this rotation transform is applied to the previous normal and bi-normal vectors to calculate the next normal and bi-normal vectors. An example of the extrusion twisting and the RMF fix is shown in Figure 4-15.

4.2.3.4 Creating particle systems

Particle systems are added to sculpted relationships to visualize the 3D path and dynamics of blood flow and show the deoxygenation of blood between the extrathoracic arteries and extrathoracic veins. Section 3.2.7 introduces the idea of leveraging the ontology structure as an interface affordance for creating and editing particle systems. A particle system is added to each *flows to* relationship. Particle systems emit particles from the source concept, animate the particles along the curve of the relationship, and remove the particles when they reach the destination concept. The particles within the particle system are each a billboarded (i.e., always oriented towards the virtual camera), textured plane. The plane is textured with the value of the `particle image` attribute added to the relationship. Figure 4-16 shows the image cell.jpg that is assigned to the

`particle image` attribute to texture the particles. The image is gray-scale and may be tinted with a color by assigning a particle color attribute to the relationship. A list of all particle attributes assigned to the 10 edges of the Beneken model is provided in Table 4-3.

Particle attribute values may be bound to values of attributes assigned to both the source and destination concepts. Figure 4-17 shows the result of binding the particle color attribute of a relationship to the attribute presentation color in the relationship's source and destination concept. This results in a linear interpolation of the particle color as it traverses the edge. Figure 4-18 shows the result of adding particle systems to all 10 relationships. What is depicted in 4-18 represents a completed module envisaged by a designer.

4.2.3.5 Defining the viewer module

A designer often has a certain audience in mind when creating visualizations. Given this audience, it is up to the designer to choose what interactivity should be permitted in the final visualization module, and what attributes should be visible. Within the proposed framework, many attributes will be design-oriented (e.g., consider the influence attributes listed in Table 4-1), and not important to an end consumer. For the case study, only the simulation variables `volume`, `pressure` and `resistance` are set to "visible in viewer." With respect to interactivity, the `resistance` attributes are made adjustable and given a range of 0.01 to 2.0. A snap shot of the designed visualization rendered in the "viewer" modality is shown in Figure 4-19. The prototype viewer provides a subset of the functionality of the prototype designer. The distinction between the two modalities in described in Section 3.2.9.

4.2.4 Extensions for Hypovolemic Shock

With a base cardiovascular model defined, one can explore augmentations that can be used to simulate different pathophysiology or abnormal conditions. To this end, a simulation and visualization of hypovolemic shock was constructed in the designer prototype by starting with the base executable simulation model defined in Section 4.2.3.1. Hypovolemic shock occurs when there is substantial loss of blood volume in the veins. The shock simulation model contains the following two extensions from the base model: an electrocardiogram (ECG); and a mapping of blood volume loss to heart rate and the strength of the heart-beat. The ECG was added as an attribute to a Heart concept. The equation used to create a realistic ECG signal was defined by McSharry et al. (2003).

Table 4-4, taken from Lawrence et al. (2006), demonstrates the symptoms of hypovolemic shock. To add the effects of this type of shock to the simulation model, code was added to the compartmental model solver of the prototype. This type of shock is represented by a conditional function with the percentage ranges defining the condition. This needed to be partially “hard coded” because the equations as attributes feature (discussed in Section 3.2.2), does not support conditional formulations in its current state.

The two influences in the hypovolemic shock case study are shown in Figure 4-21. The volume of the heart (calculated by summing the volume of all the heart’s subcomponents) is linked to the scale. The volume of the heart in the range (200ml, 600ml) is mapped to a scale in the range ([1.0,1.0,1.0], [1.3,1.3,1.3]). This results in a beating effect that can visually relay the pulse frequency and strength. As also depicted in Figure 4-21, total blood volume is mapped to the color of the skin. The blood volume

influence ranges from 4240 ml (a normal amount of blood for an adult male) to 2500 ml. The skin color ranges from [red: 0.83, green: 0.82, blue: 0.81] (very pale) to [red: 0.86, green: 0.72, blue: 0.63] (a typical Caucasian skin tone). The result of these influence additions are shown in Figure 4-22.

When the shock simulation is executed, the equations in the model are solved and total blood volume is used to scale the ECG output (much like it is used to color the skin of the human body). This linkage results in a weakened ECG signal as blood is drained. Clearly, a medical domain expert would desire a more sophisticated and accurate linkage of physiological parameters to ECG output, but this example demonstrates how using equations as attributes can enable executable ontology-based simulation models that cross mathematical paradigms.

4.3 Preliminary Human Considerations

A theoretical framework was described in Chapter 3. Future work that builds on this framework should be informed by user feedback because the ultimate goal is to make visualization construction and consumption simpler for the end user compared to current techniques. Two types of feedback were sought out: student feedback on the ontology-enabled visualization consumption interface; and expert feedback on ontology-enabled visualization construction interface. In support of both surveys, an example lesson plan was created through collaborations with a medical educator, Dr. Juan Cendan of the University Of Central Florida College Of Medicine, who provided the introductory narratives and questions for the lesson. The lesson plan presents two real-world scenarios, and then guides participants through the functions of the viewer prototype loaded with the Beneken executable simulation and visualization of human cardiovascular system. The lesson plan serves as a tutorial for the prototype's function

and also as a learning module. The complete lesson plan is presented in Appendix A. The lesson plan was encoded in a webpage and rendered in a side panel of the software prototype, as depicted in Figure 4-23. For the purpose of the study, the prototype software was referred to as KOG (Knowledge, Ontology, Graph).

4.3.1 Student Survey

Second year medical students at the University of Central Florida were presented with the option to participate in a study that compares two modules for teaching cardiovascular function. Students that participated in the study were given a pre-test assessing their general second year medical knowledge, randomly given one of the two learning modules with built-in lesson plan, and then given a post-test which contains the same questions as the pre-test but in a different order. No time constraints were placed on participants and time spent with the module was not tracked. Study enrollment was low, so a full statistical comparison of the modules was not viable. Presented in this section are the responses to the open-ended survey questions of the four students that performed the study with the KOG viewer. The results are as follows:

Q1: Do you find this educational software to be different from tools you have used in the past to learn physiology?

All four students said “Yes.”

Q2: If you found this educational software to be different, please elaborate on the differences.

Student 1: It is the most interactive form of learning and encourages self-directed exploration. There aren't very many other ways to explore all of the multiple ways you can view the heart's functions (PV loops, EKG, BP...) when you affect flow or HR.

Student 2: Using this technology required a knowledge of what the components in the loop were. Higher level of understanding of the anatomy was required.

Student 3: More interactive.

Student 4: It was very interactive. I like the possibility to see how one parameter affects the other. Instead of reading the text I could easily see the relation.

Q3: Do you believe this educational software can provide any benefit to the learning experience? If so, please elaborate.

Student 1: Yes!

Student 2: Yes, these tools are good to use if given appropriate scenarios

Student 3: Yes, but I would have gotten more out of it if this was a required SLM [Self Learning Module] or information was to be used for testing purposes because I probably would have spent more time with it.

Student 4: Yes, it is helpful. I would follow up the quiz questions with quick explanation though.

Q4: Please provide any additional feedback you may have on the content or technology that was just presented to you.

Student 3: I liked the technology and content. However, I didn't think the pre and post test questions were reflective of content that would be learned with this presentation.

Student 4: The first time I used it was a little confusing because there were so many instructions. The second scenario was much easier and faster to navigate. Once you go beyond learning how to use the software it becomes very helpful.

There are several observations of the student response I think are worth further consideration. First, Student 2 noted the technology required knowledge of "what the components in the loop were" and that a "higher level of understanding of the anatomy was required." This comment suggests that an understanding of anatomy was required to effectively use the tool's interface. Such a linkage of abstract and concrete knowledge was explored by Quarles et al. (2008) with respect to anesthesia machine function. The viewer prototype could be used to run a similar study with respect to

anatomy and physiology. Such a study would explore any benefits or downfalls of the integrative visualizations constructed with the proposed approach.

Secondly, all four students reported the tool could benefit the learning experience, but three out of four mentioned that for the tool to be useful, it needs to be integrated into pedagogy by an instructor. This early feedback suggests that, at least in the context of medical education, these types of simulation-based visualizations are likely not to be leveraged in an exploratory manner by most students.

Lastly, in Student 4's additional feedback, it was claimed that they learned the interface after performing just one scenario, stating "the second scenario was much easier and faster to navigate" and the software became "very helpful." For this student, the labeled ontology served as an interface that could be learned quickly and permitted fast, intuitive access to desired information (in the form of attributes). From a perspective of visualization consumption, the 3D ontology graph could prove to be a highly effective interface. To make this claim, a new study will need to be performed. This is a challenging task, however, because there are no current standards employed by educational visualizations that could be compared to the ontology-based method. One potential approach is to collect a large sample of visualizations (possibly freely available online) that explore the same basic phenomenon and have a group of students perform a lesson with each module, including the ontology-based approach.

4.3.2 Expert Survey

Five experts took part in a survey that aimed to assess the prototype viewer and also the design theory employed by the proposed framework. Consider an "expert" to be someone working in medical education as an instructor or engineer in a medical research lab that would potentially leverage visualizations in their work. Each expert

was presented with a 12 minute video demonstrating the design features of the proposed framework. In the video, the process of constructing the Beneken beating heart visualization is illustrated. The video is provided as Object 4-1.

After watching the video, the expert was given a link to download and install the same prototype viewer provided during the student survey (see Section 4.3.1). The time participants took with the prototype viewer was not tracked. Three medical faculty members and two engineers from research labs took part in the survey. The template expert survey provided by Ravden et al. (1989) was used and the complete results are presented in Appendix B. In this section an overview of the responses will be given along with an analysis.

The overall expert response to the presentation (i.e., the video and viewer prototype) was promising. It was stated that "there is potential here," "there is a lot of versatility in the animation system and infinite uses by the look of it," and that there are "many 'experiments' that can be done." It was also mentioned that "an orientation of 5 minutes with students would be enough to navigate this." An engineer referred to the system overall as "very intuitive." These positive responses are encouraging, but the major reason for performing this survey is to seek out valuable criticism from the experts.

Much of the critical feedback from the experts pertained to their experience with the viewer prototype, and not the design concepts presented in the video. This is understandable because they were able to interact with the software prototype and had a more passive experience with the content in the video. I believe this is indicative of the challenge of trying to assess preliminary technology in non-interactive form (e.g.,

paper mock-ups, videos). There was, however, one important critique of the design theory presented to the experts. Two of the medical faculty noted that the design language in the video was not intuitive for non-programmers. While technical programming concepts were not mentioned in the video, there were computer graphics terms that may be unfamiliar to some domain experts (e.g., “mesh,” “vertices,” “control points”). This feedback specifically from the medical faculty suggests that even though all participants were able to effectively consume the interactive visualization, the process of designing may still remain predominantly an engineering practice. Findings from this survey inform a placement of the proposed framework within the current visualization construction pipeline. This placement is discussed further in Chapter 5 along with future directions for research.

4.4 Summary

In this chapter, a case study was detailed that demonstrates the theory presented in Chapter 3. Preliminary human considerations were also discussed based on the results of two surveys. The case study, a simulation and visualization of the human cardiovascular system, demonstrates a series of steps to facilitate simulation and visualization construction by leveraging domain knowledge. A compartmental model is constructed and additional visualization semantics are attached to the model. There are other sequences of interactions carried out by the domain expert that can result in the same dynamic visualization. One could start with an existing ontology and augment this with simulation semantics and graphical objects. Conversely, the graphics (3D and 2D media) could serve as a starting point and ontological structures, including dynamics, could be added. The approach taken will depend on the resources of the

domain expert and what co-interaction between the three model types (ontologic, dynamic, and graphic) is envisioned.

Table 4-1. The semantics required to recreate the 10-compartment Beneken model of the human cardiovascular system.

Attribute	Parent Structure	Description
`volume`	Concept	The blood volume in a chamber.
`pressure`	Concept	The blood pressure in a chamber.
`flow`	Concept	The out flow of blood in a chamber.
`unstressed volume`	Concept	The unstressed volume of a chamber.
`elastance`	Concept	The elastance of a chamber.
`elastance min`	Concept	The minimum elastance of a chamber of the heart.
`elastance max`	Concept	The maximum elastance of a chamber of the heart.
`inertia`	Relationship	The inertia value which should be placed between intrathoracic arteries and extrathoracic arteries.
`valve`	Relationship	The attribute that determines if a relationship has a valve.
`valve open factor`	Relationship	The factor in which to multiply flow rate if the relationship has a valve and the valve is open.
`valve close factor`	Relationship	The factor in which to multiply flow rate if the relationship has a valve and the valve is closed.

Table 4-2. Influence attributes added to the Right Atrium concept to drive the animation of the right atrium portion of the heart mesh.

Attribute	Type	Value
`influence radius`	Variable, Scalar	0.43
`influence source`	Basic	Volume
`influence destination`	Basic	Scale
`influence source min`	Variable, Scalar	0
`influence source max`	Variable, Scalar	320
`influence destination min`	Variable, Vector	0.7,0.7,0.7
`influence destination max`	Variable, Vector	2.2, 2.2, 2.2

Table 4-3. Particle system attributes added to *flows to* relationships.

Attribute	Type	Value
`particle emission`	Basic	Flow
`particle image`	Basic	Cell.jpg
`particle color`	Basic	Presentation color
`particle emission factor`	Variable, Scalar	1.0
`particle random offset`	Variable, Scalar	0.2
`particle wander interval`	Variable, Scalar	6
`particle random offset`	Variable, Scalar	0.2

Table 4-4. The effects of hypovolemic shock as illustrated by Lawrence et al. (2006).

Blood loss amount	Blood Pressure	Pulse	Skin Vasoconstriction	Urine Output
≤10 %	NL	NL	NL	Flow
20%	↓	↑	↑	↓
30%	↓↓	↑↑	↑↑	↓↓
50%	↓↓↓	↑↑	↑↑↑	↓↓↓

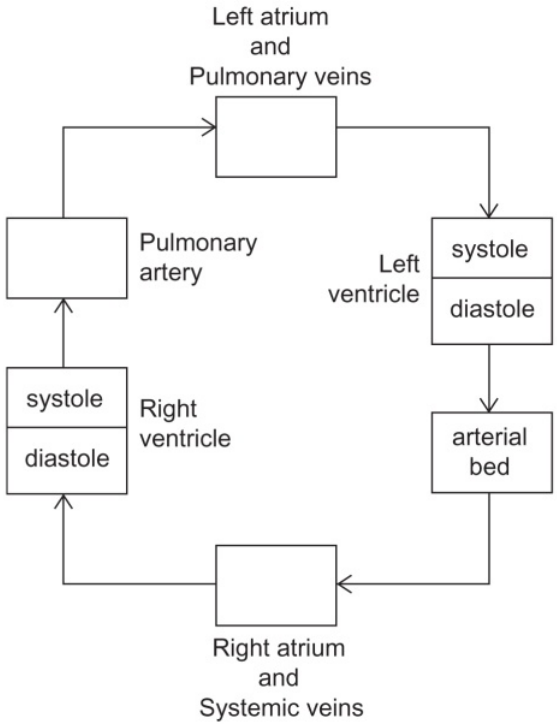


Figure 4-1. A block diagram model created by Warner (1959) to simulate blood circulation.

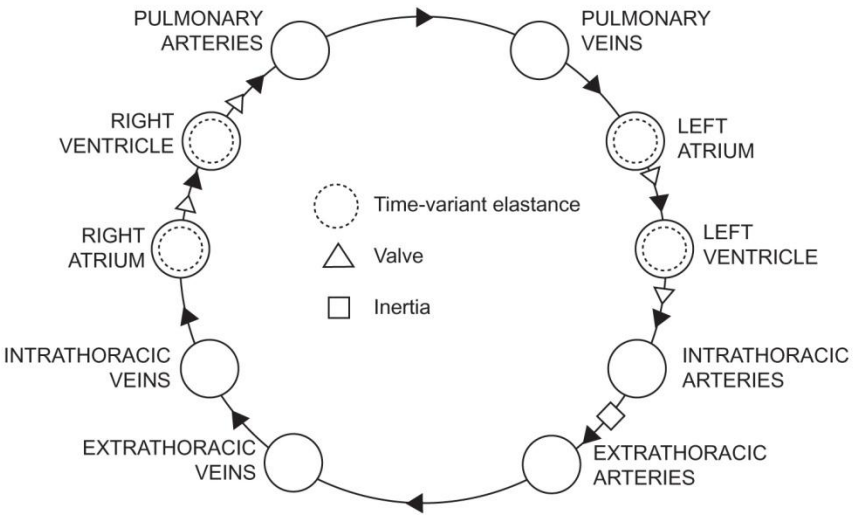


Figure 4-2. The Beneken (1965) compartmental model for blood flow.

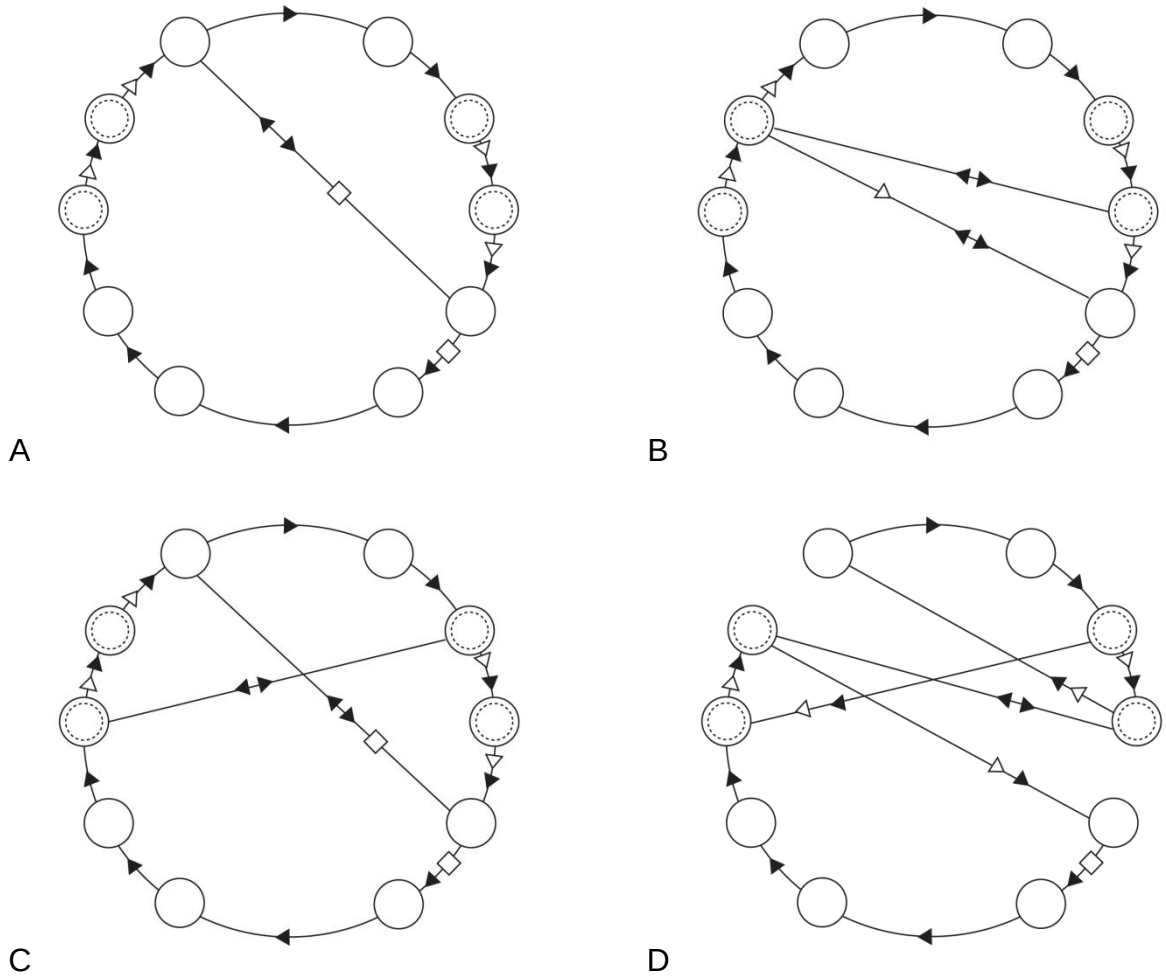


Figure 4-3**.** Pathology models by Sá Couto et al. (2006). See Figure 4-2 for a key of symbols and compartment names. A) Patent ductus arteriosus. B) Tetralogy of Fallot. C) Coarctation of the aorta with patent foramen ovale and a small patency of the ductus arteriosus. D) Transposition of the great arteries with septal defects.

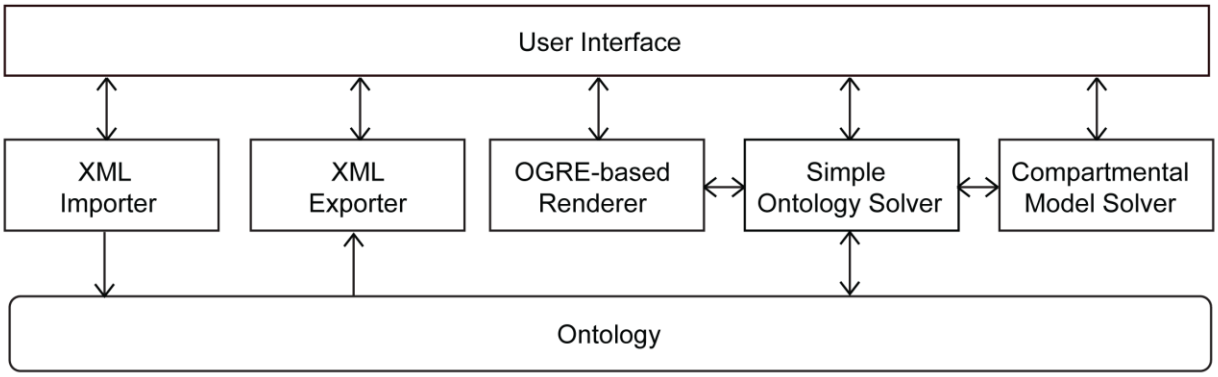


Figure 4-4. The architecture of the software prototype created to demonstrate the proposed theory. This architecture is an instantiation of the high-level architecture presented in Chapter 3.

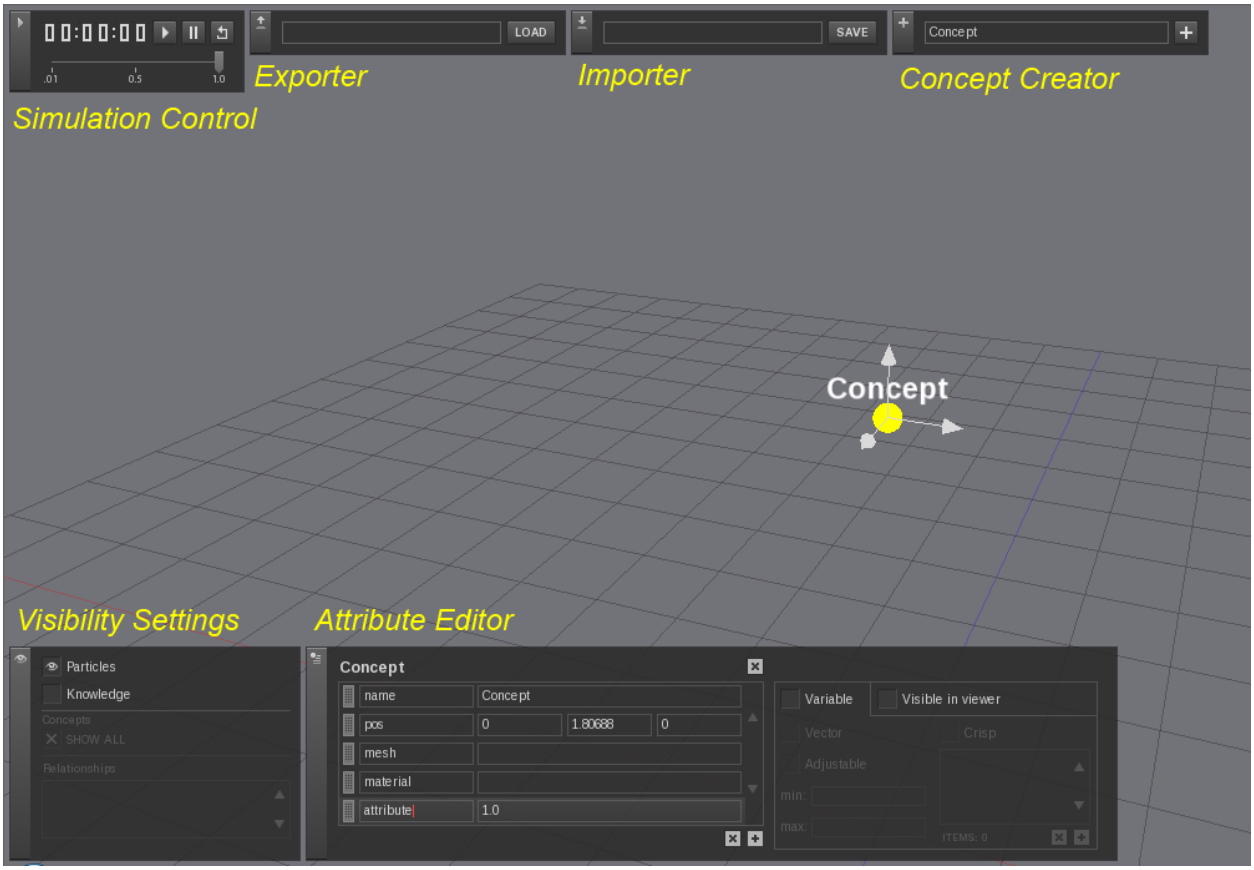


Figure 4-5. A screenshot of the prototype. The yellow text labels the different components of the user-interface. A single concept name “Concept” is selected and its attribute named “attribute” is selected in the attribute editor.

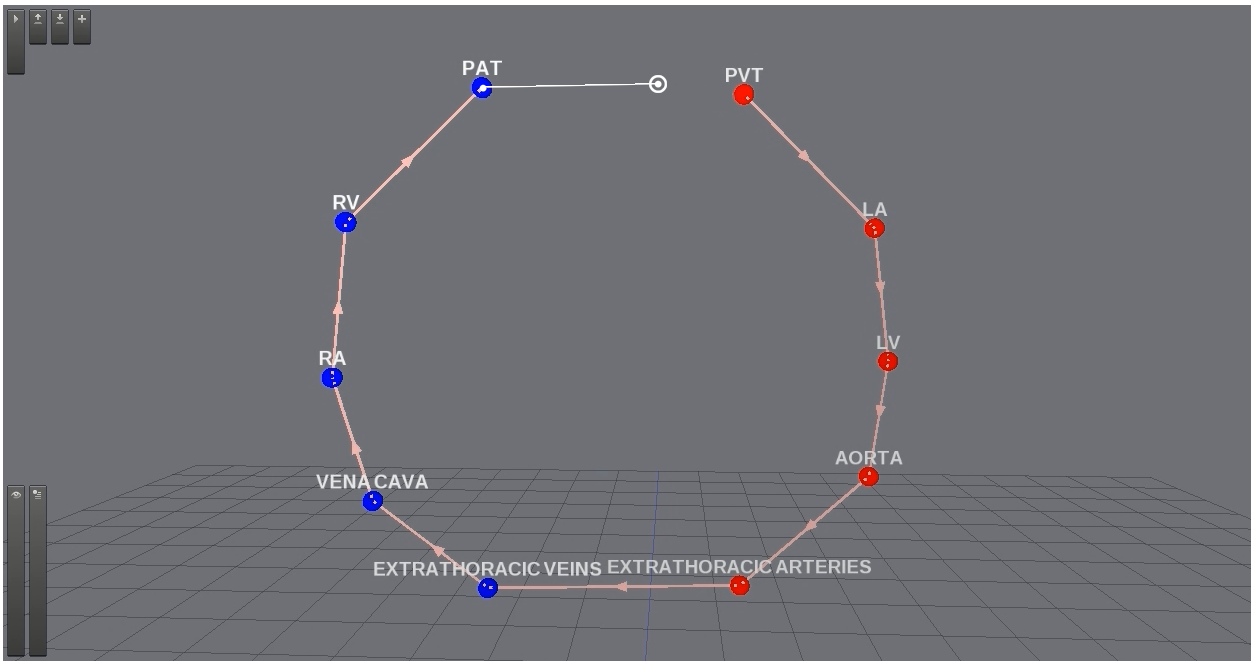


Figure 4-6. A snap shot of building the Beneken model. The designer adds a relationship between the Pulmonary Arterial Tree concept and the Pulmonary Venous Tree concept to denote blood flow.

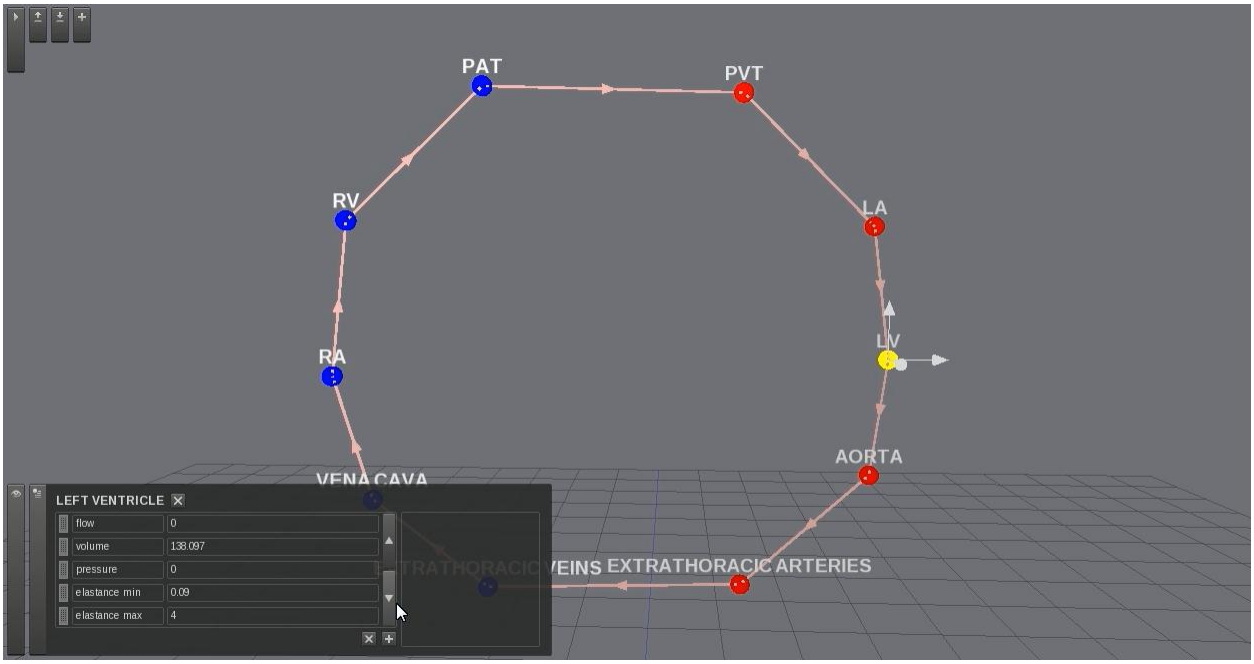


Figure 4-7. The complete Beneken model created with the software prototype. The Left Ventricle is selected so a translation widget appears rooted at this concept. All attributes of the Left Ventricle are also shown in the Attribute Editor.

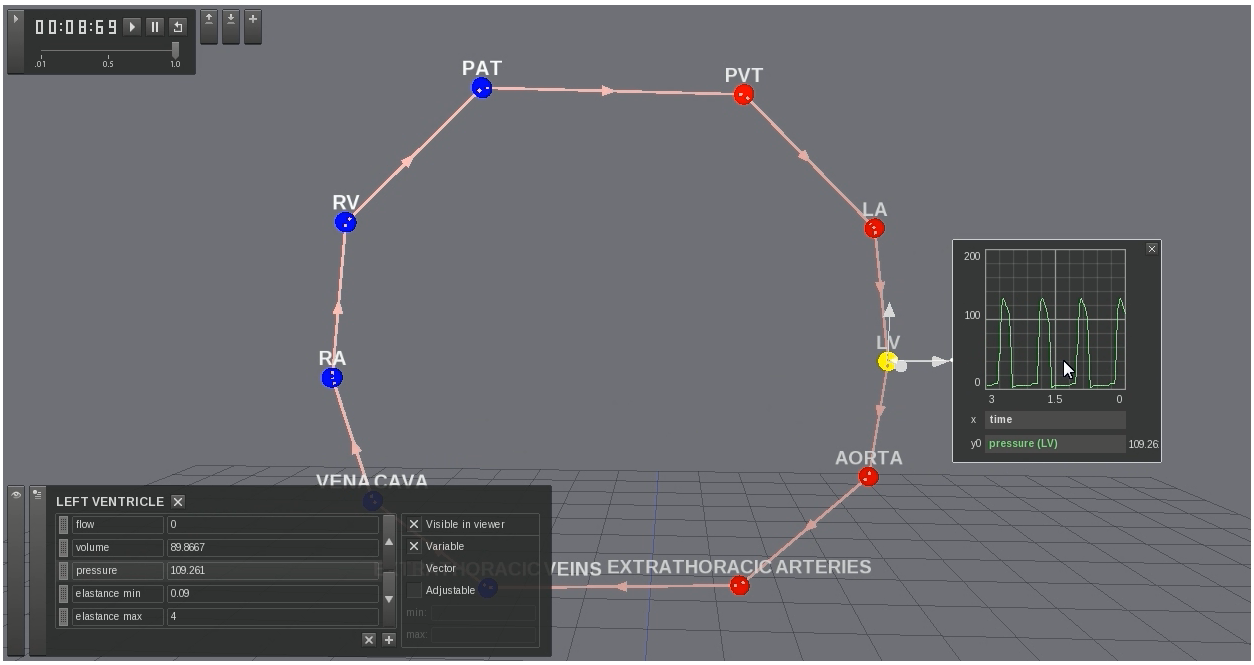


Figure 4-8. The Beneken model executing in the software prototype. The attribute pressure of the Left Ventricle concept is expanded.

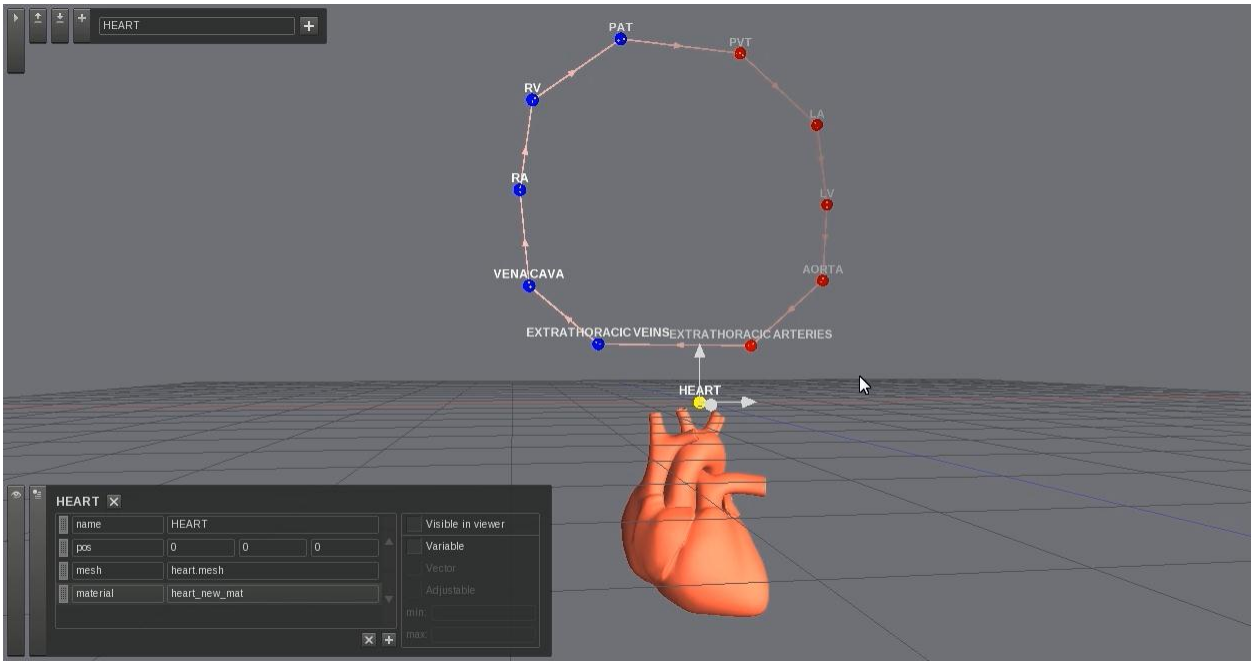


Figure 4-9. The result of adding a Heart concept and assigning it a mesh and material attribute.

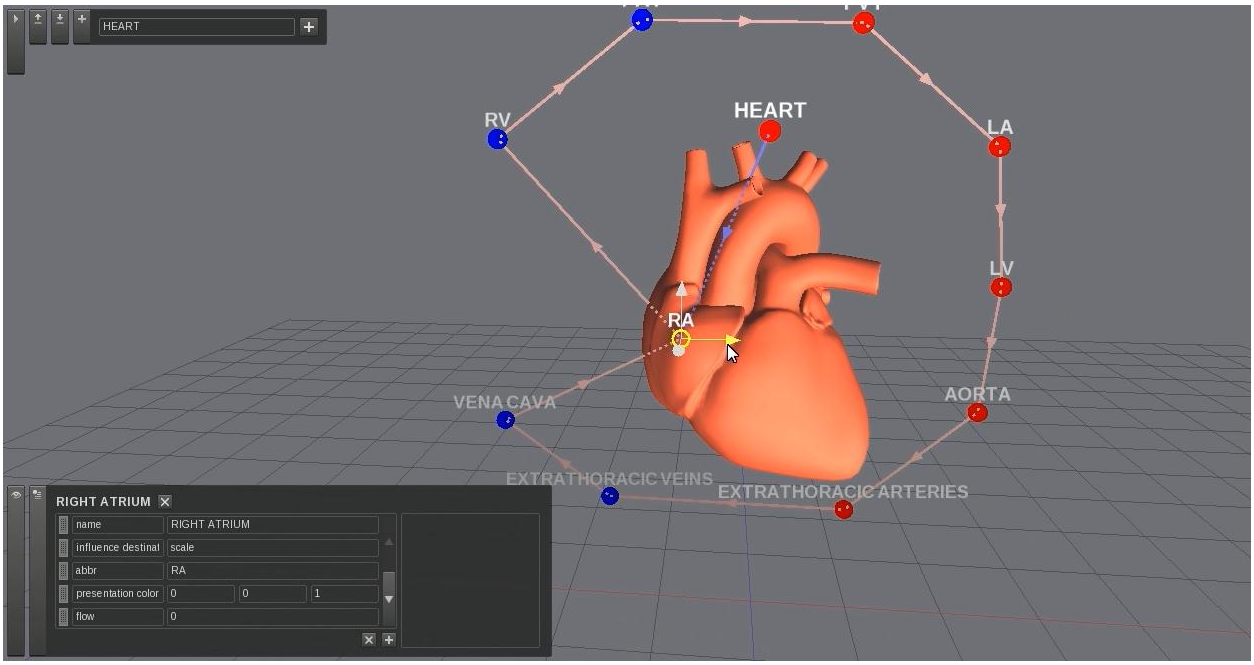


Figure 4-10. A snap shot of the prototype while a designer positions the Right Atrium concept to be within the heart mesh. The Heart is linked to the Right Atrium through a *has a* relationship, which is colored blue in the prototype.

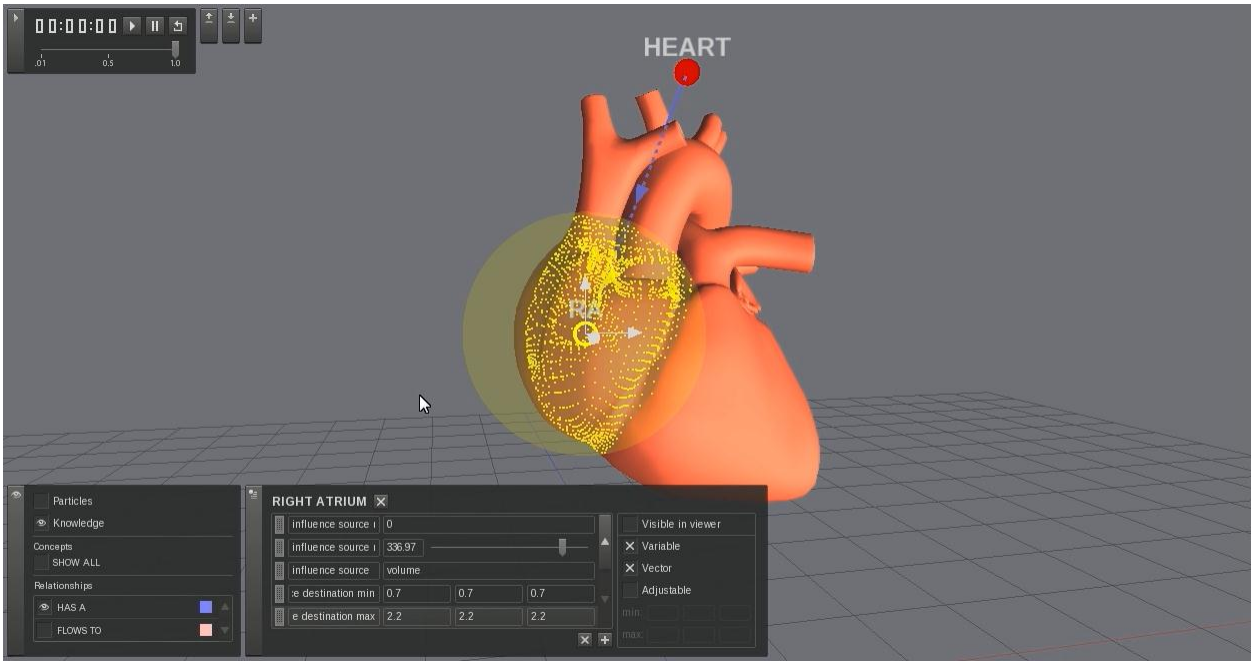


Figure 4-11. The influence of the Right Atrium concept over the heart mesh. Concepts and relationships not involving *has a* semantics are hidden in this view as determined by the settings in the View Settings interface (lower left).

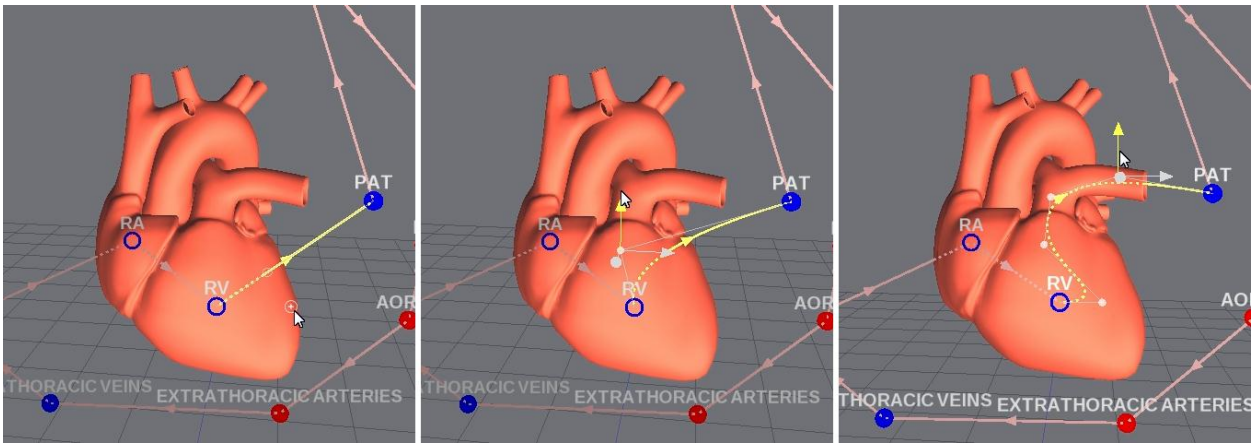


Figure 4-12. Three snap shots of a designer forming a curve from the *flows to* relationship between the concepts of Right Ventricle and Pulmonary Arterial Tree. Control points are added to the relationship to form a hull that defines curvature (left). The control points can be translated to change the relationship's curvature (center, right).

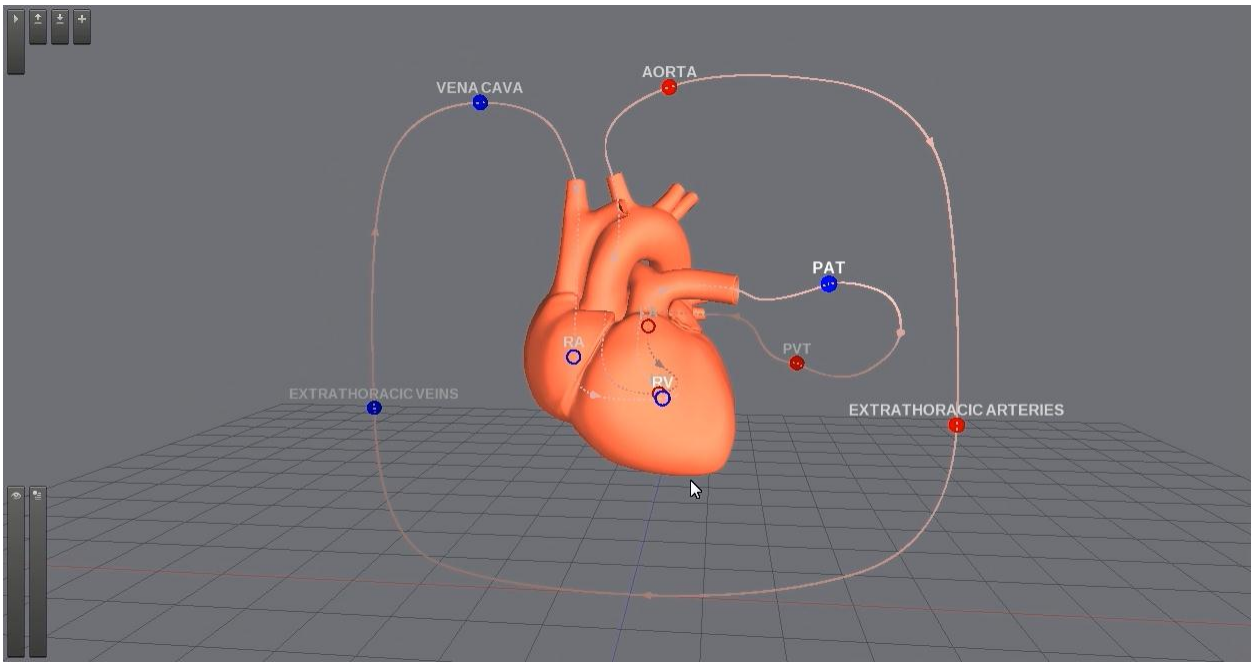


Figure 4-13. The complete Beneken model co-located with the heart mesh in 3D. Concepts are color coded based on whether blood in the corresponding chamber is oxygenated (red is oxygenated, blue is deoxygenated).

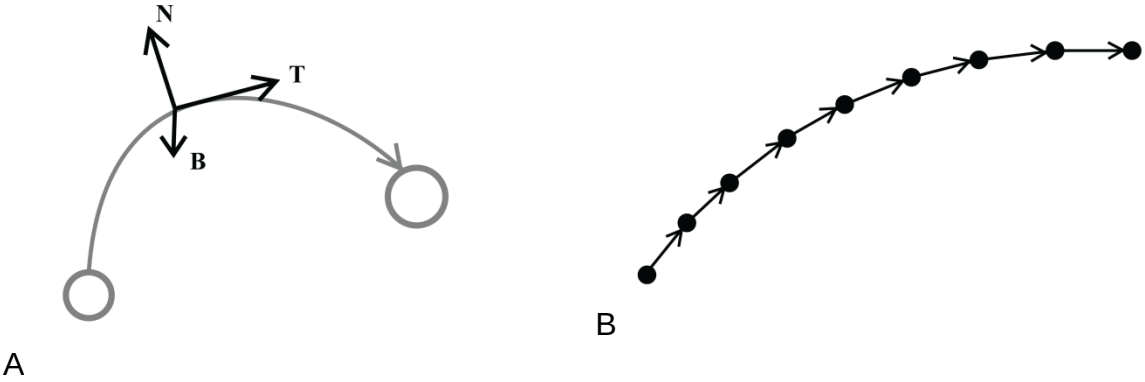


Figure 4-14. The frame of a curve. A) An illustration of the frame of a curve, including the tangent (**T**), normal (**N**), and bi-normal (**B**) vectors. B) A "zoomed-in" view of a solved spline. Solved spline points are connected by line segments. Once normalized, these segments can be used as tangent vectors when calculating the frame of the curve at a given point. Given the tangent, the rotation minimizing frame technique can be used to calculate the normal and bi-normal.

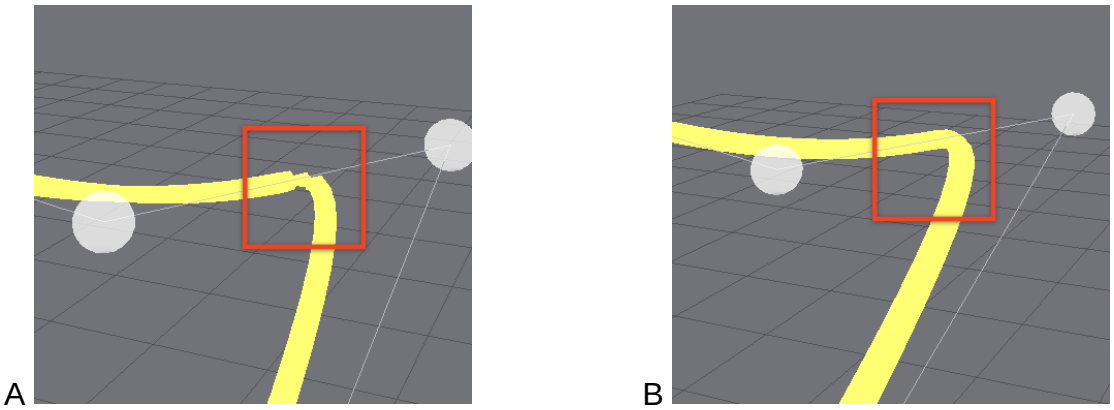


Figure 4-15. The result of using the rotation minimizing frame technique. A) An arbitrary source vector (e.g., <0,1,0>) is used in the frame calculation to extrude a square along a curve to create thickness. An undesirable twist of the extrusion is highlighted. B) The same curve, without the twist, when the rotation minimizing frame technique is employed.

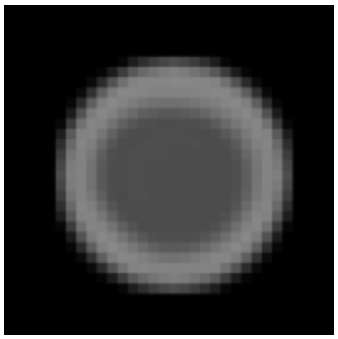


Figure 4-16. A 32x32 pixel image, cell.jpg, used in the creation of particles for the case study. The black in the image maps to transparency. An image is assigned to a particle system through the `particle image` attribute.

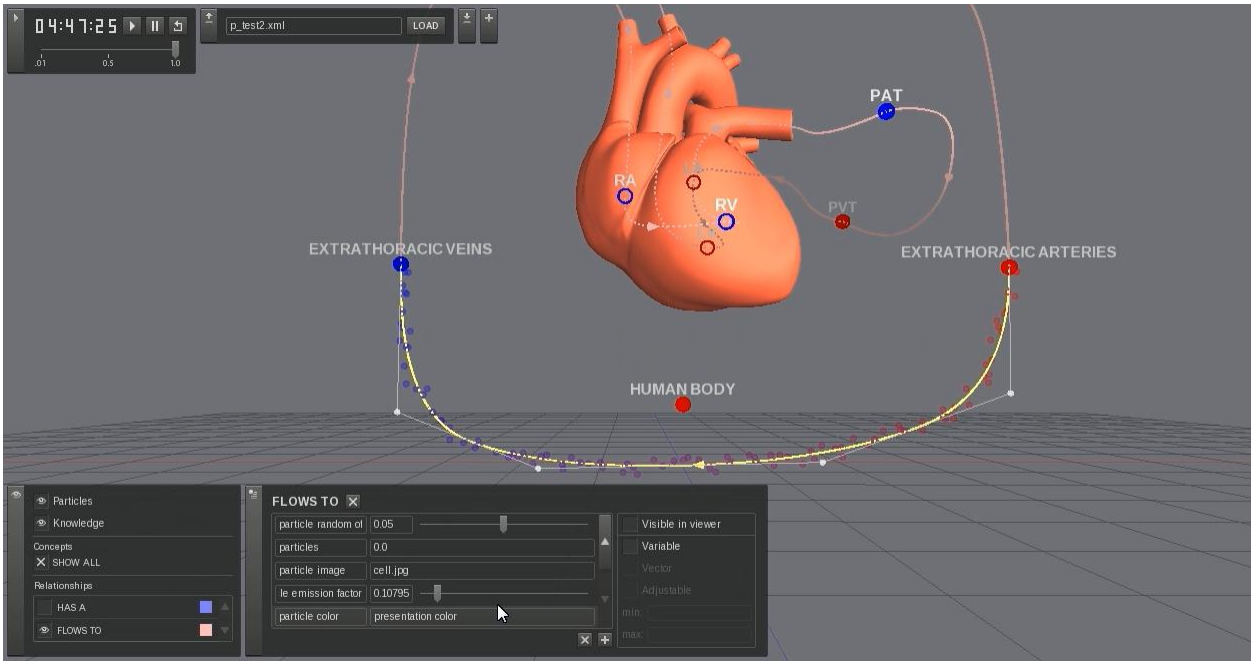


Figure 4-17. A particle system animates between the Extrathoracic Arteries and Extrathoracic Veins concepts.

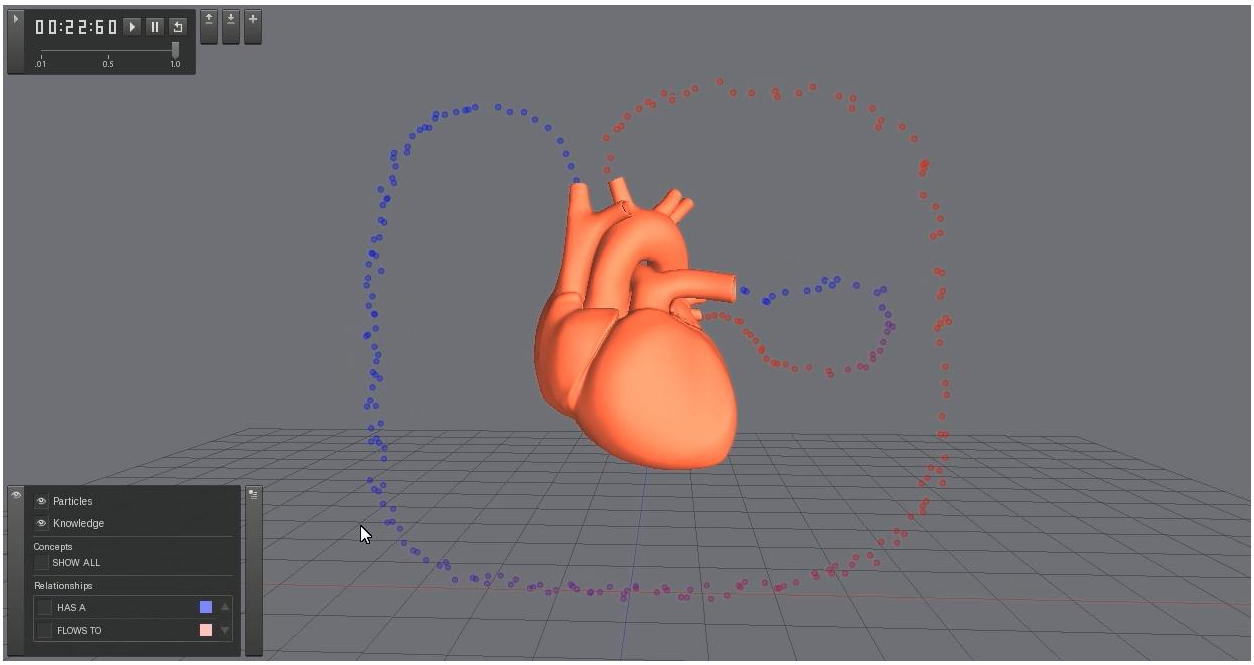


Figure 4-18. The complete Beneken model co-located with the heart mesh in 3D. Concepts are color coded based on whether blood in the corresponding chamber is oxygenated (red is oxygenated, blue is deoxygenated).

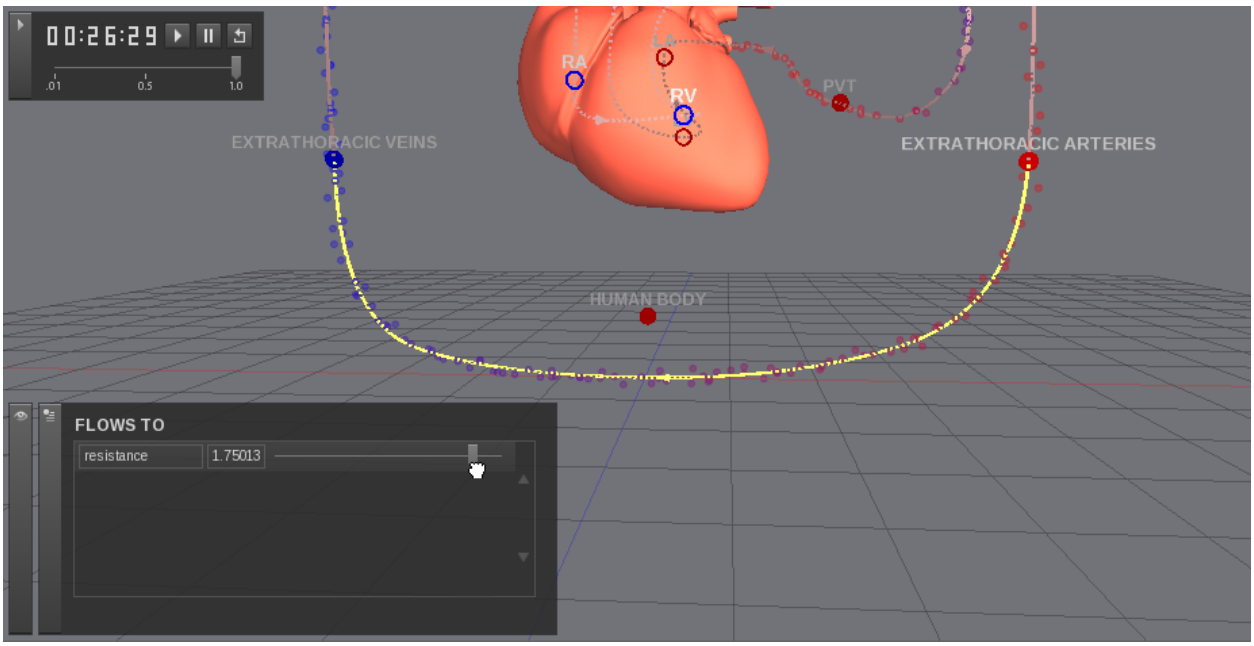


Figure 4-19. A snap shot of the designed visualization rendered in the prototype viewer. A user drags a slider to manipulate the resistance between the extrathoracic arteries and extrathoracic veins.

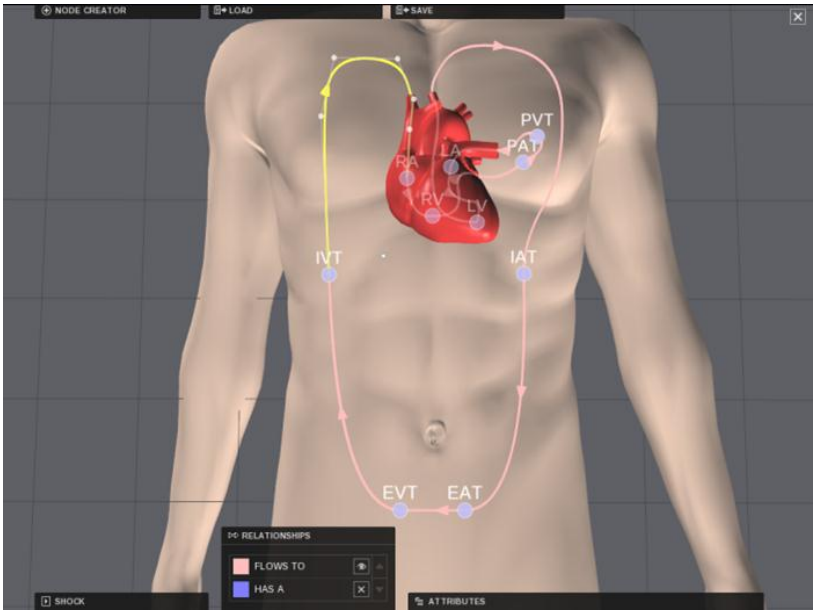


Figure 4-20. A snap shot of the Beneken model co-located with heart and human body 3D geometry. A *flows to* edge is selected and its control hull is made visible.

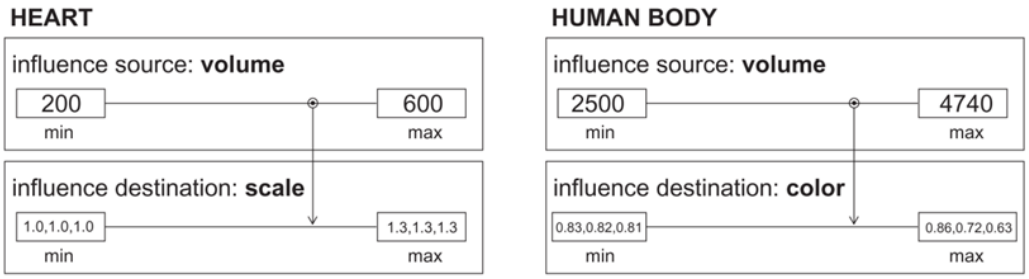


Figure 4-21. An illustration of two influences added to the concepts Heart and Human Body within the shock visualization.

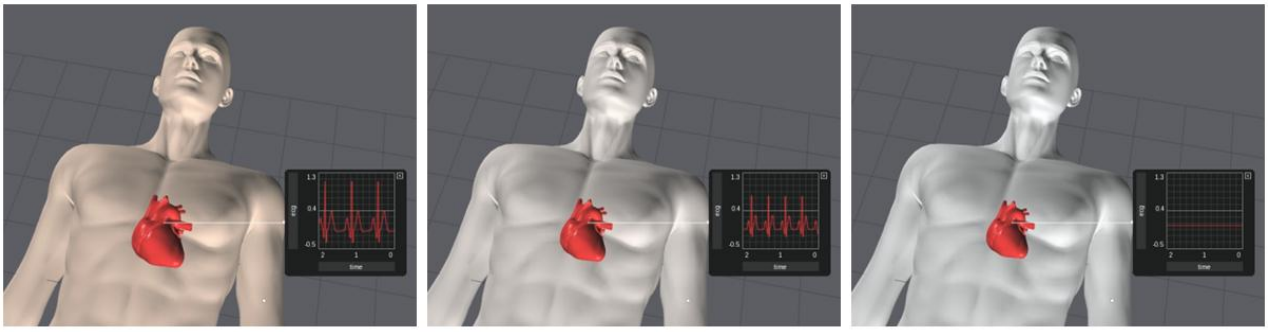


Figure 4-22. A simulation and visualization of hypovolemic shock over time. The human's skin tone becomes less saturated and the heart beat weakens.

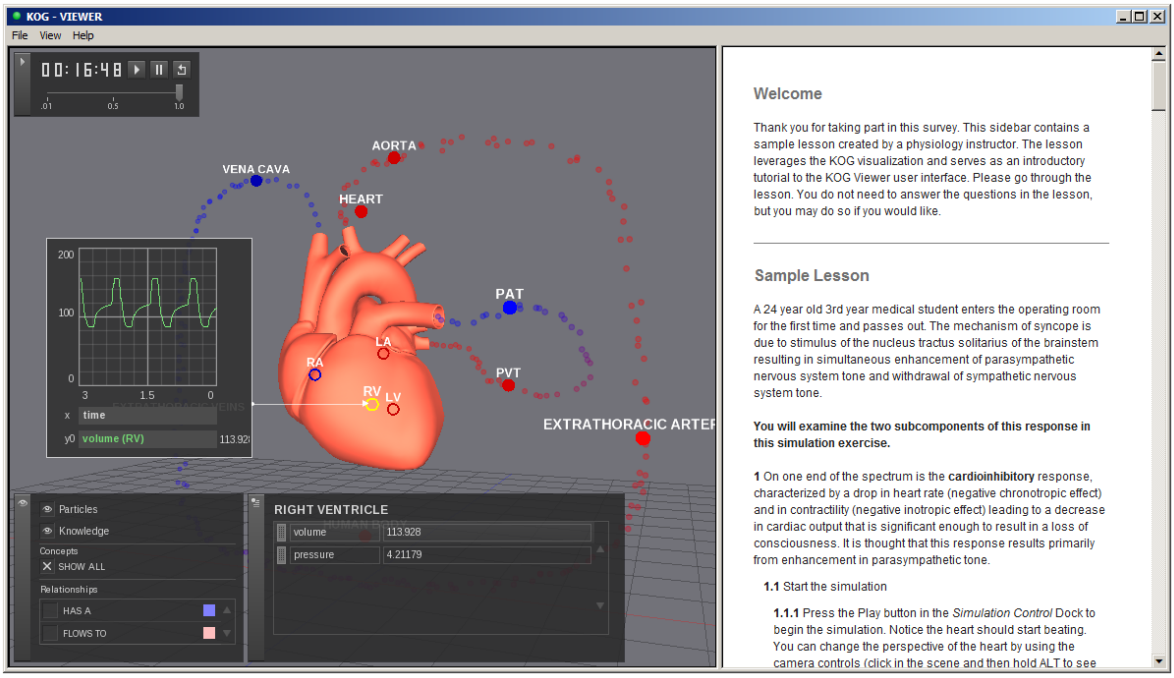


Figure 4-23. The viewer prototype with lesson plan that was distributed with the student and expert surveys.

CHAPTER 5
CONCLUSIONS

In this dissertation a new interaction theory was presented that connects simulation model building and visualization construction activities at the user-interface level. The theory is built around an interactive 3D graph structure: a visualization of an augmented ontology. The motivating claim for this work is: a three-dimensional visualization of the domain-ontology can serve as a central user interface structure to connect simulation modeling and visualization construction activities and allows domain-specific semantics to guide the interactive modeling process. Such an interface may improve the simulation and visualization construction pipeline by allowing simulations and visualizations to be constructed in a more efficient manner compared to current techniques. The interface could also open simulation and visualization design and consumption to a wider audience.

The ontology visualization, with concepts, relationships, and attributes, can encode domain specific semantics in a format that is machine-processable and also understandable by the domain experts. In the defined interface, the visualization of an ontology as a 3D graph serves to anchor other visualization elements such as meshes and variable plots. Relationship visualizations can be sculpted to trace meaningful 3D paths within the visualization environment for particle systems and other animated objects. To demonstrate the methodology, a software prototype was created. In turn, an executable simulation and visualization of the human cardiovascular system was created with the prototype. I believe this case study presented in Chapter 4, along with the structural and semantic affordances presented in Chapter 3 make a strong case for the proposed interface and justify future efforts toward expanding this work.

Research and development of the ontology-based interaction theory and preliminary feedback has led to a projection of how such a framework could be utilized in present day visualization construction efforts. Currently, if a domain expert wishes to create a visualization of a dynamic process, they must work with engineers to build the visualization and create any executable simulation models. This process is depicted in Figure 5-1A. A domain expert presents a design (represented by a paper sketch in the figure) to a visualization expert. The visualization expert builds the visualization by synthesizing a set graphical resource with executable interaction and simulation models. The visualization expert may need to employ an engineer with simulation expertise to fully implement the simulation. This additional expertise is represented under the generic label of "software engineer" in Figure 5-1 and manifests in many forms (e.g., such as through pre-coded modules found online, and through collaborations with a simulation specialist). Also worth noting is that what is being distinguished in this description is "expertise." It is possible for one person to possess one, two, or all three of the required skill sets for visualization construction.

By leveraging the framework proposed in this dissertation, the construction process could be improved by lessening the effort required by the computer engineering expert and afford the domain expert more creative control over the final visualization module. This improved process is presented in Figure 5-1B. The computer engineer needs to create a customized simulation solver for the domain of interest and pass it to the visualization expert. The visualization expert then can build the visualization with the ontology-centered tool and give this "first draft" visualization to the domain expert. The domain expert can finely tune the influences between simulation and visualization

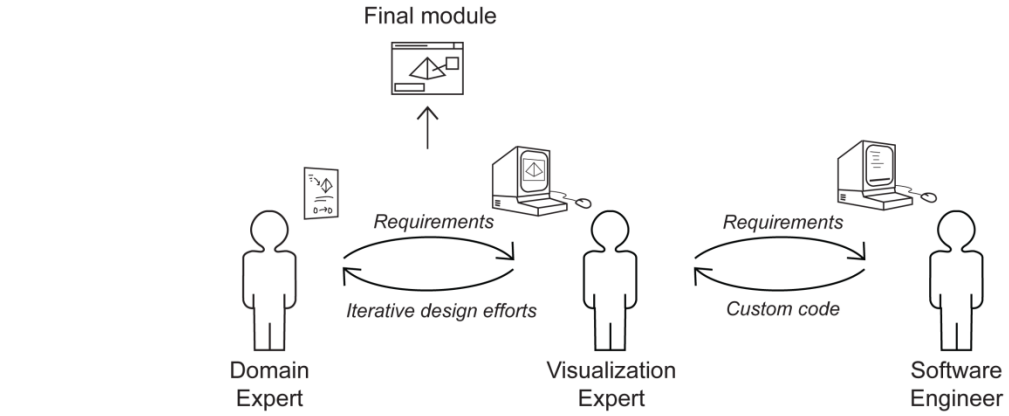
and any required simulation coefficients. The tuning by the domain expert is represented by the “refinement” loop in Figure 5-1B. The end goal of this work is depicted in Figure 5-1C, where a domain expert constructs an interactive, integrative visualization independently. In this process, the domain expert has complete expressive freedom and does not require the resources of the visualization and computer engineering experts.

Future work to expand on the theory and framework presented in this dissertation falls into two categories: human-computer interaction studies, and technical enhancements. Studies could be run that expand on the preliminary studies presented in Section 4.3. Such a study is currently underway at the University of Central Florida College Medicine. This study replicates the procedure of the student survey that was carried out during this work, but is being administered to more participants. The study also includes questions that will be used to compare the ontology-based visualization viewer to the learning modules currently used by the College of Medicine instructors. The goal of this study is to identify which characteristics of integrative visualizations are useful for various educational purposes.

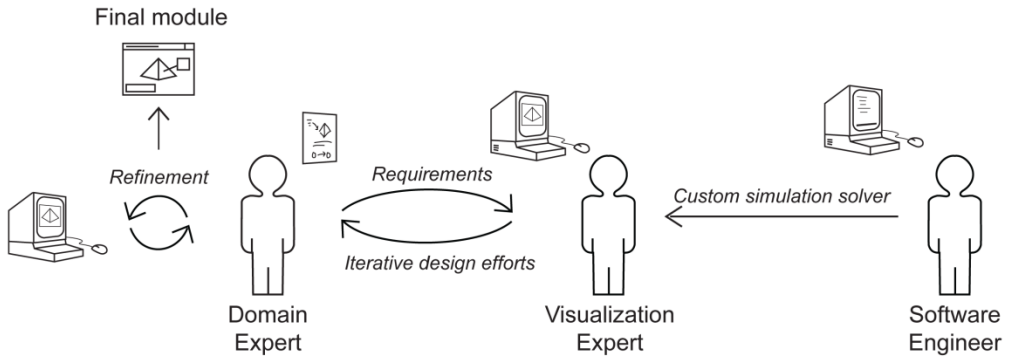
With respect to technical enhancements, the prototype can be developed further to allow for general simulation model building within a particular modeling paradigm (e.g., compartmental modeling). This was not done during this dissertation because of the development effort required to create a general purpose solver. However, with much of the other required development (e.g., coding of the rendering and interaction system) complete, this may be a viable option for future work. A general version of the ontology-based designer prototype could be deployed to experts and the expert survey from

Section 4.3.1 could then be carried out again, replacing the video demonstration that was used with a fully functional designer prototype.

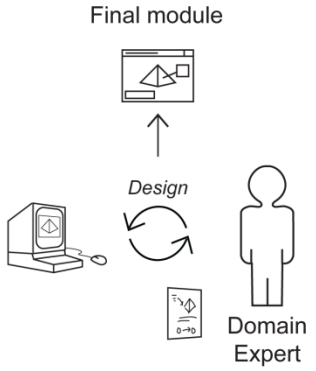
Collectively, this proposed future work could help further define the strengths and limitations of the approach. Thus far the strengths have been identified as: the domain-semantics based interface; and simulation modeling and visualization construction activities taking place in the same 3D interaction space. Limitations have been identified as: the approach may not scale well to large visualization scenes with many concepts and simulation rules; and certain required computer graphics concepts may still be prohibitively advanced to be employed by domain experts. By considering the strengths and limitations identified thus far and by future work, a designer should be able to make an informed choice as to whether or not the ontology-based visualization tool should be used to build the visualization in mind.



A



B



C

Figure 5-1. Various engineering processes for constructing simulation-based 3D interactive visualizations. A) The approach with current tools. B) A compressed approach that could be used with the current version of the ontology-based framework. C) The ultimate goal and reason to continue this work.

APPENDIX A
EXAMPLE LESSON

In this appendix, a lesson plan is presented that was created in collaboration with a medical educator. The target audience for this lesson plan is second year medical students. The lesson was displayed in a side panel of the software prototype and students were asked to complete the lesson as part of a survey. The lesson is formatted here in a manner similar to how it was rendered within the prototype, with figures in-line.

Welcome

Thank you for taking part in this survey. This sidebar contains a sample lesson created by a physiology instructor. The lesson leverages the KOG visualization and serves as an introductory tutorial to the KOG Viewer user interface. Please go through the lesson. You do not need to answer the questions in the lesson, but you may do so if you would like.

Sample Lesson

A 24 year old 3rd year medical student enters the operating room for the first time and passes out. The mechanism of syncope is due to stimulus of the nucleus tractus solitarius of the brainstem resulting in simultaneous enhancement of parasympathetic nervous system tone and withdrawal of sympathetic nervous system tone.

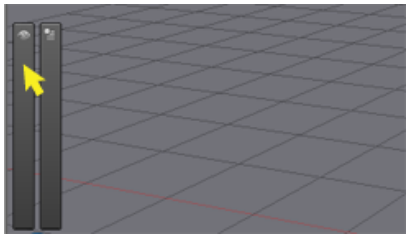
You will examine the two subcomponents of this response in this simulation exercise.

1 On one end of the spectrum is the **cardioinhibitory** response, characterized by a drop in heart rate (negative chronotropic effect) and in contractility (negative inotropic effect) leading to a decrease in cardiac output that is significant enough to result in a loss of consciousness. It is thought that this response results primarily from enhancement in parasympathetic tone.

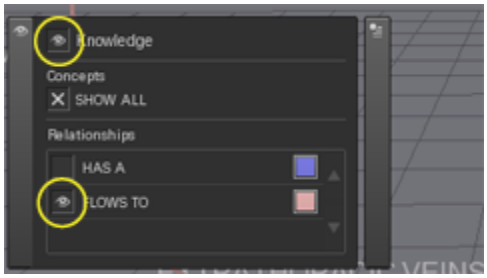
1.1 Start the simulation

1.1.1 Press the Play button in the *Simulation Control* Dock to begin the simulation. Notice the heart should start beating. You can change the perspective of the heart by using the camera controls (click in the scene and then hold ALT to see controls). Press CTRL + R at any time to reset the camera position.

1.1.2 Expand the *View Settings* Dock.

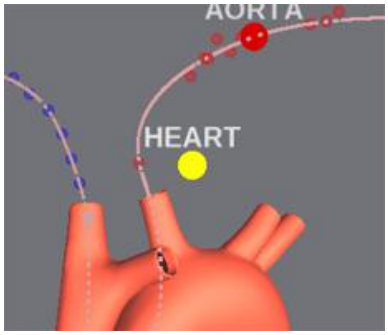


1.1.3 In the *View Settings* Dock, check "Knowledge" to show anatomical concepts and "Flows To" to show blood flow relationships.

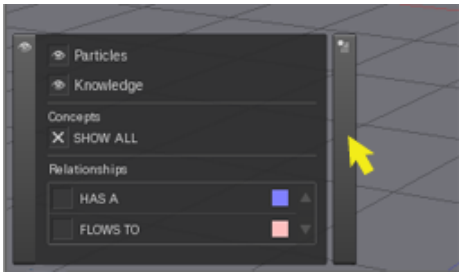


1.2 Display the baseline features

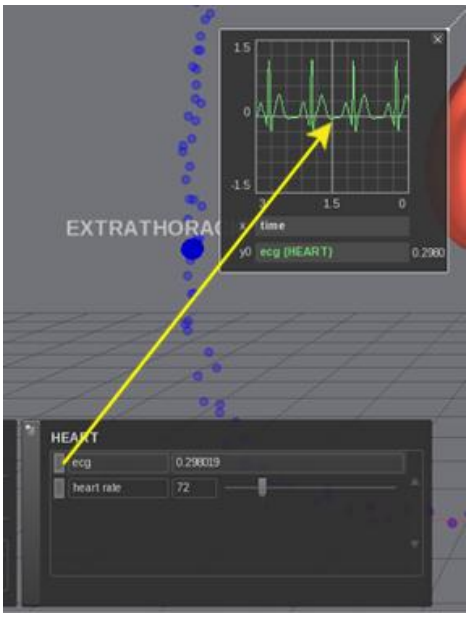
1.2.1 Select the Heart node (click it with the left mouse button and it should turn yellow).



1.2.2 Expand the Attribute Dock.



1.2.3 Drag and drop the ECG field into the scene. **NOTE:** Plots can be removed by pressing the 'x' in the upper right corner of the plot. Plots can be moved by dragging the plot with the right mouse button

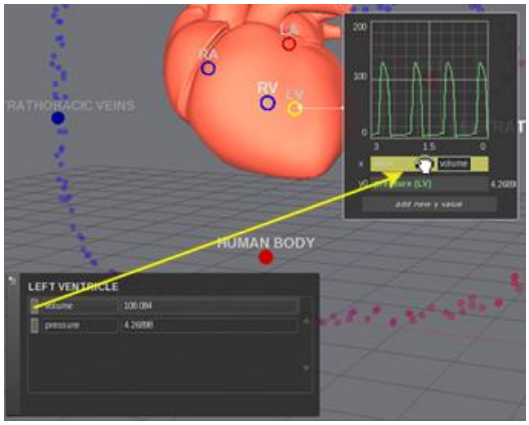



Press and hold the right mouse button to move plots.

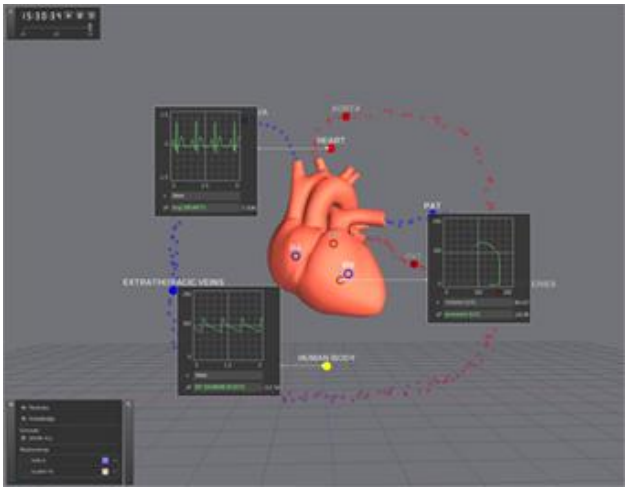
1.2.4 Select the Human Body node and drag the Blood Pressure (BP) plot into the scene.

1.2.5 Select the Left Ventricle (LV) and drag the 'pressure' plot into the scene.

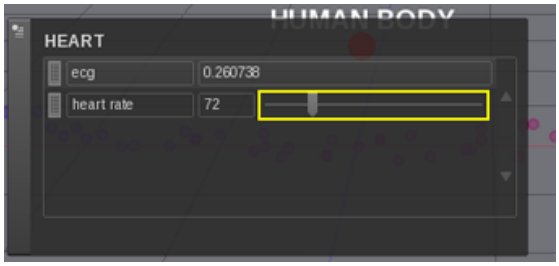
1.2.6 Now drag volume of the Left Ventricle (LV) onto the x-axis of the pressure plot. This creates the P-V loop. Remember that you can always remove the plot and start over if a mistake is made.



1.2.7 Now the simulation should be executing (the heart should be beating) and you should have three plots in your scene. Remember you can control the perspective on the heart by moving the camera (hold ALT to see the camera controls) and you can move the plots by right clicking anywhere on the plot and dragging.



1.3 You are now ready to answer Question 1. Question 1 requires you to change the heart rate. To do so you will need to use the slider for heart rate that appears in the *Attribute Editor*Dock when the Heart node is selected. Please answer Question 1 now.



Question 1

Interact with the visualization using the variable plots and sliders to answer the following questions. Be sure to "submit" your responses before moving on.

1A. What happens to the Blood Pressure when the HR=150?

1B. What happens to the BP when the HR =50?

1C. As the heart rate increases, what happens to the amount of blood pumped, per stroke, by the left ventricle?

1D. As heart rate increases, which component of the ventricular cycle is most affected: Ventricular filling or emptying?

2 On the other end of the spectrum is the vasodepressor response, caused by a drop in blood pressure (to as low as 80/20) without much change in heart rate. This phenomenon occurs due to vasodilation, probably as a result of withdrawal of sympathetic nervous system tone.

2.1 Start or Reset the simulation.

2.1.1 If you are continuing this exercise from Question 1 then you can simply reset the simulation (see image below for reset button). The variable plots will stay in the scene but simulation parameters will be reset.

2.1.2 Make sure "Knowledge" and "Flows To" are both set to visible (see 1.1.2) in the *View Settings* Dock.

2.1.3 If you are starting this part of Scenario 1 with a freshly opened copy of KOG (your plots from part 1 are no longer visible), then you will need to add the following variable plots to the scene: 'BP' from the Human node, 'ECG' from the Heart node, and create the P-V loop from the Left Ventricle (LV) node (see 1.2.5 and 1.2.6).

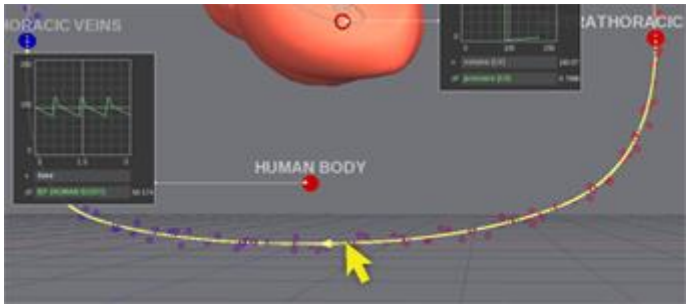
2.1.4 Press Play to begin the simulation.

2.1.5 At any time you may adjust the simulation speed using the slider in the Simulation Control dock. The range is from 0.01 (one-hundredth the speed of real-time) to 1.0 (real-time).

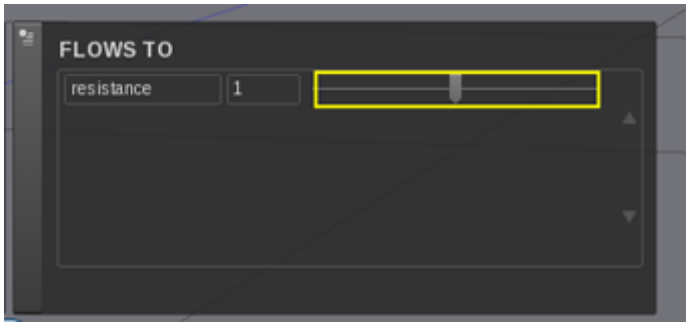
2.1.6 You should now see a beating heart and with three variable plots in the scene. Remember you can move the camera (press ALT to see controls) and move the plots (Right click and drag the plots) at any time.

2.2 You are now almost ready to begin Question 2. To answer Question 2, you will need to change peripheral resistance (the resistance between the Extrahoracic Arteries and the Extrathoracic Veins).

2.2.1 To access peripheral resistance, select the edge between Extrahoracic Arteries and the Extrathoracic Veins.



2.2.2 To modify peripheral resistance, use the slider for resistance in the *Attribute Editor* Dock.



2.3 You are now ready to answer Question 2. Please do so now.

Question 2

2A. What happens to the Blood Pressure when peripheral resistance is very low (0.5)?

2B. What happens to the BP when the resistance is elevated (2.0)?

2C. Return to the low resistance state (0.5) and compound that with a low heart rate (50). What happens to the blood pressure now?

2D. Now, increase the resistance to 2.0 but keep the heart rate at 50. What happens to the blood pressure? Which component appears to have a more substantial effect?

2E. Feel free to investigate the interplay between heart rate, peripheral resistance and blood pressure. You can report other relationships to us in the text box below.

Final Thoughts

The majority of people with vasovagal syncope have a **mixed** response somewhere between or as a combination of these two ends of the spectrum.

Exploratory

You may continue to explore the visualization. Every 'flows to' edge has a resistance variable that can be modified.

APPENDIX B
EXPERT SURVEY RESULTS

Results from an expert survey about a case study on the proposed framework are presented in this appendix. Three medical faculty and two engineers took part in the survey.

What are the best aspects of the system?

Medical Faculty 1: Best aspects are that there is potential here. It is also a relatively simple model which can be good or bad.

Medical Faculty 2: It allows you to modify a parameter and see the interplay with other factors in the loop.

Medical Faculty 3: There is a lot of power in the physiological model and many "experiments" that can be done. There is obviously a lot of versatility in the animation system and infinite uses by the look of it. The ability to visualize - to see real-time changes and make observations is very important.

Engineer 1: Good visual representation of heart. Very intuitive.

Engineer 2: Being able to manipulate variables, thinking about what I want to see.

What are the worst aspects of the system?

Medical Faculty 1: Lacks the ability to stop at certain points for students to record average pressures, or to calculate variables when things have changed.

Medical Faculty 2: Cannot display timing markers on graphs to coordinate time on one loop vs another (or with the ECG).

Medical Faculty 3: Not completely intuitive how to include charts in the animation - needs the instruction sheets.

Engineer 1: Artery to vein that we adjusted pressure in could use a visual representation like a vessel that gets bigger and smaller depending on resistance.

Engineer 2: Spending time getting numerical values from the graph. Although I could have estimated much faster.

Are there any parts of the system which you found confusing or difficult to fully understand?

Medical Faculty 1: The demo video is in "programming" language and the actual development is not useful to a basic scientist. The terms do not mean much to me.

Medical Faculty 2: Would be helpful to have numbers from graphs/loop display at side

Medical Faculty 3: Not really - some of the terminology is not intuitive for someone who is not a programmer but this was not a problem from the user perspective when doing the exercise.

Engineer 1: Visually it is easier for me to see the name of the attribute next to the axis it is plotting vs. seeing them both below the x-axis. Not that big of a deal

Engineer 2: It's been awhile since I've seen pressure-volume loops. I had to Google that just to be certain. But that is the beauty of this kind of learning, I could read about it and mess around with it at the same time.

Were there any aspects of the system which you found particularly irritating although they did not cause major problems?

Medical Faculty 1: Not being able to stop to collect number data or to overlay PV loops to compare before or after a condition/treatment.

Medical Faculty 2: Cannot start knowledge section without turning off ALT control box. Print is small. Do not see an option to enlarge.

Medical Faculty 3: It may be better to have a separate panel of icons to drag graphs into the animation.

Engineer 1: No ability for Y axis look inversion makes it very difficult for me to navigate. (I always use inverted look).

Engineer 2: I made some mistakes and couldn't go back, I had to restart. – I'd like to see default values marked. – I was changing the graph scales, and there was some funny behavior; I could not use the number pad. – I'd like to see units (mmHg? probably) what are units of stoke volume? CC? Seeing this I don't know.

What were the most common mistakes you made when using the system?

Medical Faculty 2: Remembering function of mouse and key settings

Medical Faculty 3: I had to check back on the instructions once or twice to figure out how to alter resistance if I was just working in another area like heart rate - but this was OK after using the system for a few minutes. An orientation of 5 minutes with students would be enough to navigate this.

Engineer 1: Only mistake was inverting the plot for x and y axis on the pressure/volume curve.

Engineer 2: Not enough mistakes to have a most common one. I built the PV loop the other way around once so the axes were reversed. If this were a public, live teaching tool, that should be allowed - but not without a note that appears for "common settings" Similar to a note for default settings in the user adjustable variables.

What changes would you make to the system to make it better from the user's point of view?

Medical Faculty 1: If a basic scientist were to use it, it should be in terms of how we can modulate. Not sure if it's meant for a programmer to maintain or a basic scientist.

Medical Faculty 3: Have an interface to select variables like pressure, volume, flow without having them hidden one step away inside another panel?

Engineer 1: Y axis look inversion ability. Y value units next to the Y axis. Dropping Y value next to the Y axis.

Engineer 2: Click on any point on the graph, see the (y) value with units.

Do you currently leverage 3D visualizations in your work? Why or why not?

Medical Faculty 1: Currently, no as physiology is more concept based and less dependent perhaps on 3 D. It could be more beneficial in other systems such as the kidney.

Medical Faculty 2: Yes.

Medical Faculty 3: Yes - we use them more for anatomical work at present but incorporating physiology is excellent - student need to integrate structural relations with function. Nice work

Engineer 1: Yes. Mixed reality simulation.

Engineer 2: Yes, it's the only way to fly.

LIST OF REFERENCES

Alani H (2003). TGVizTab: An ontology visualization extension for Protégé. Proceedings of the 2003 Workshop on Visualization Information in Knowledge Engineering. ACM Press: New York, NY, pp. 434-435.

Aldridge RM (2009). Using models to teach congenital heart defects: a hands-on approach. *Dimensions of Critical Care Nursing* **28**(3):116-122.

Artignan G and Hascoët M (2011). From Coding to Automatic Generation of Legends in Visual Analytics. *Lecture Notes in Business Information Processing* **73**(4): 404-417.

Bates DW, Cullen DJ, Laird N, Petersen LA, Small SD, Servi D, Laffel G, Sweitzer BJ, Shea BF, Hallisey R et al (1995). Incidence of adverse drug events and potential adverse drug events. *American Medical Association* **274**: 29-34.

Bartels R H, Beatty J C and Barsky A (1987). *An introduction to splines for use in computer graphics and geometric modeling*. Kaufmann Publishers: Los Altos, CA, USA.

Beermann J, Tetzlaff R, Bruckner T, Schöebinger M, Müller-Stich BP, Gutt CN, Meinzer HP, Kadmon M and Fischer L (2010). Three-dimensional visualization improves understanding of surgical liver anatomy. *Medical Education* **44**(9): 936-940.

Bell D and Ludwig SA (2005). Grid service discovery in the financial markets sector. *Journal of Computing and Information Technology* **13**(4): 265-270.

Bell D, Mustafee N, de Cesare S, Taylor SJE, Lycett M and Fishwick PA (2008). Ontology engineering for simulation component reuse. *International Journal of Enterprise Information Systems* **4**(4): 47-61.

Beneken JEW (1965). *A mathematical approach to cardiovascular function: the uncontrolled human system*. PhD thesis, Medisch Fysisch Institut TNO.

Benjamin P and Graul M (2006). A framework for adaptive modeling and ontology-driven simulation. Proceedings of the SPIE, Enabling Technologies for Simulation Science X, **6227**: 622705.1-622705.11.

Benjamin P, Patki M and Mayer R (2006). Using ontologies for simulation modeling. In: Perrone LF, Lawson B, Liu J and Wieland FP (eds). *Proceedings of the 2006 Winter Simulation Conference*. IEEE Computer Society: Washington, DC, USA, pp 1151-1159.

Benzaken V, Fekete J, Hemery P, Khemiri W and Manolescu I (2011). EditFlow: Data-intensive interactive workflows for visual analytics. *IEEE 27th International*

Conference on Data Engineering. IEEE Computer Society: Washington, DC, USA, pp 780-791.

Berners-Lee T, Hendler J and Lassila O (2001). The Semantic Web. *Scientific American* **284**(5): 34-43.

Bhatt M, Rahayu W and Sterling G (2004). sedOnto: A Web enabled ontology for synthetic environment representation based on the SEDRIS specification. *Fall Simulation Interoperability Workshop. Fall Simulation Interoperability Workshop.* IEEE Computer Society Press: Piscataway, NJ, USA.

Bhatt M, Rahayu W and Sterling G (2005). Synthetic environment representational semantics using the web ontology language. *Proceedings of the Sixth International Conference on Intelligent Data Engineering and Automated Learning*. Springer: Berlin, pp 9-16.

Blender3D (2012). Blender - Home. http://www.blender.org, accessed 1 September 2012.

Blomqvista E and Öhgren A (2008). Constructing an enterprise ontology for an automotive supplier. *Engineering Applications of Artificial Intelligence* **21**(3): 386-397.

Bosca A and Bonino D (2006), OntoSphere3D: A multidimensional visualization tool for ontologies. *Proceedings of the 17th International Conference on Database and Expert Systems Applications*. IEEE Computer Society: Washington, DC, USA, pp 339-343.

Buntain, C (2008). 3D ontology visualization in semantic search*. Proceedings of the 46th ACM Southeast Conference*. ACM Press: New York, NY, USA, pp 204-208.

Carroll J, Dickinson I, Dollin C, Reynolds D, Seaborne A and Wilkinson K (2004). Jena-implementing the semantic web recommendations. *Proceedings of the Thirteenth International World Wide Web Conference*. ACM Press: New York, NY, USA, pp 434-435.

Chandrasekaran S, Silver G, Miller JA, Cardoso J and Sheth AP (2002). Web service technologies and their synergy with simulation. *Proceedings of the 2002 Winter Simulation Conference*. In: Yücesan E, Chen CH, Snowdon JL and Charnes JM (eds). IEEE Computer Society: Washington, DC, USA, pp 606-615.

Chang K, Sung Y and Chen I (2002). The effect of concept mapping to enhance text comprehension and summarization. *Experimental Education* **71**: 5-23.

Cooper JB, Newbower RS, Long CD and McPeek B (2002). Preventable anesthesia mishaps: a study of human factors. *Quality and Safety in Health Care* **11**: 277-282.

da Silva P (2001). User interface declarative models and development environments: a survey. In: Palanque P and Paternò F (eds). *Proceedings of the 7th International Conference on Design, Specification, and Verification of Interactive Systems*. Springer-Verlag: Berlin, pp 207-226.

Daconta MC, Obrst LJ and Smith KT (2003). *The Semantic Web: A Guide to the Future of XML, Web Services and Knowledge Management*. Wiley Publishing: Indianapolis, IN, USA.

Derakhshani D (2010). *Introducing Maya 2012*. Wiley Publishing, Inc: Indianapolis, IN, USA.

Diaz J, Petit J and Serna M (2002). A survey of graph layout problems. *ACM Computer Surveys* **34**(3): 313-356.

Eades P (1984). A heuristic for graph drawing In: Stanton RG (ed*). Congressus numerantium*. Utilitas Mathematica: Winnipeg, pp 149-160.

Ebbesen J, Buajordet I, Erikssen J, Brørs O, Hilberg T, Svaar H and Sandvik L (2001). Drug-related deaths in a department of internal medicine. *Archives of Internal Medicine* **161**(19): 2317-2323.

Einsfeld K, Ebert A and Wölle J (2007). HANNAH: a vivid and flexible 3D information visualization framework. In: Banissi E, Burkhard RA, Grinstein G, Cvek U, Trutschl M, Stuart L, Wyled TG, Andrienko G, Dykes J, Jern M, Groth D and Ursyn A (eds). *11th International Conference on Information Visualization*. IEEE Computer Society: Washington, DC, USA, pp 720-725.

Euler L (1775). Principia pro motu sanguinis per arterias determinando. *Opera posthuma mathematica et physica* **2**:814–823.

Ezzell Z, Fishwick PA, Lok B, Pitkin A and Lampotang S (2011). An ontology-enabled user interface for simulation model construction and visualization. *Journal of Simulation*. **5**: 147-156.

Fekete JD (2004). The InfoVis Toolkit. In: Ward MO and Munzner T (eds). *Proceedings of the 10thIEEE Symposium on Information Visualization.* IEEE Computer Society: Washington, DC, USA ,167-174.

Fischler IS, Kaschub CE, Lizdas DE and Lampotang S (2008). Understanding of anesthesia machine function is enhanced with a transparent reality simulation. *Simulation in Healthcare* **3**(1): 26-32.

Fishwick P A (1995). *Simulation Model Design and Execution: Building Digital Worlds*. Prentice Hall: Upper Saddle River, NJ, USA.

Fishwick PA (2004). Toward an integrative multimodeling interface: a human-computer interface approach to interrelating model structures. *Journal of Simulation* **80**: 421-432.

Forbes AG, Höllerer T and Legrady G (2010) behaviorism : a framework for dynamic data visualization. *Transactions on Visualization and Computer Graphics* **16**(6): 1164-1171.

Ganser ER and North SC (2000) An open graph visualization system and its applications to software engineering. *Software – Practice and Experience* **30**: 1203-1233.

The Gene Ontology Consortium (2000). Gene Ontology: tool for the unification of biology. *Nature Genetics* **25**: 25-29.

The Gene Ontology Consortium (2010). The Gene Ontology in 2010: extensions and refinements. *Nucleic Acids Research* **38**(1): 331-335.

Gibson JJ (1979). *The Ecological Approach to Visual Perception.* Houghton Mifflin: Boston.

Gilson O, Silva N, Grant PQ and Chen M (2008). From web data to visualization via ontology mapping. In: Vilanova A, Telea A, Scheuermann G and Möller T (eds). *Proceedings of the 200 IEEE-VGTC Symposium on Visualization*. Blackwell Publishing, Oxford, UK, pp 959-966.

Gomez-Perez A and Corcho O (2002). Ontology languages for the semantic web. *Intelligent Systems* **17**(1): 54-60.

Gruber TR (1993). A Translation approach to portable ontology specifications. *Knowledge Acquisition* **5**: 199-220.

Guo D, Chen J, MacEachren AM and Liao K (2006). A Visualization system for space-time and multivariate patterns (VIS-STAMP). *IEEE Transactions on Visualization and Computer Graphics* **12** (6): 1461-1474.

Haarslev V and Moller R (1988). Visualization of experimental systems. *1988 IEEE Workshop on Visual Languages*, pp 175-182.

Hastings E, Guha R and Stanley KO (2007). NEAT particles: design, representation, and animation of particle system effects. *Proceedings of the 2007 IEEE*

Symposium on Computational Intelligence and Games. IEEE Computer Society: Washington, DC, USA, pp 154-160.

Heer J, Card SK and Landay JA (2005). prefuse: a toolkit for interactive information visualization. In: van der Veer GC and Gale C (eds). *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, New York, NY, USA, pp 421-430.

Hopkins JF and Fishwick PA (2003). The rube framework for software modeling and customized 3-D visualization. *Visual Language and Computing* **14**(1): 97-117.

Huh MY and Song K (2002). DAVIS: A Java-based data visualization system. *Computational Statistics* **17**(3): 411-423.

Jackson SL, Krajcik J and Soloway E(1998). The design of guided-learner-adaptable scaffolding in interactive learning environments. In: Atwood ME, Karat MC, Lund A, Coutaz J and Karat J (eds). *Proceedings of the 1998 Conference on Human Factors in Computing Systems*. IEEE Computer Society: Washington, DC, USA, pp 187-194.

Jørgensen SE, Fath B, Bastianoni S, Marques J, Muller F, Nielsen S N, Patten B, Tiezzi E and Ulanowicz R (2007). *A New Ecology: Systems Perspective*. Elsevier: Oxford, UK.

Kalogerakis E, Christodoulakis S and Moumoutzis N (2006). Coupling ontologies with graphics content for knowledge driven visualization. In: Simon A, Welch G and Bolas M (eds). *Proceedings of the 2006 IEEE conference on Virtual Reality*. IEEE Computer Society: Washington, DC, USA, pp 43-50.

Krahl D (2002). The Extend simulation environment. In: Yucesan E, Chen CH, Snowdon JL and Chames JM (eds). *Proceedings of the 2002 Winter Simulation Conference*. IEEE Computer Society: Washington, DC, USA, pp 205-213.

Katifori A, Halatsis C, Lepouras G, Vassilakis C and Giannopoulou E (2007). Ontology visualization methods – a survey. *ACM Computing Surveys* **39** (4).

Klipp E, Liebermeister W, Wierling C, Kowlad A, Lehrach H and Herwig R (2009). *Systems Biology: A Textbook*. Wiley Publishing: Weinheim, Germany.

Kohn L T, Corrigan J M and Donaldson M S (2000). *To Err Is Human: Building a Safer Health System*. National Academy Press: Washington, DC, USA.

Kolodnytsky M and Kovalchuk A (2001). Interactive software tool for data visualization. In: Loncaric S and Babic H (eds). *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis*. IEEE Computer Society Press: Piscataway, NJ,USA, pp 91-93.

Knublauch H, Fergerson RW, Noy NF and Musen MA (2004). The Protégé OWL plugin: an open development environment for semantic web applications. In: McIlraith SA, Plexousakis D and Harmelen F (eds). *Proceedings of the 2004 International Semantic Web Conference*. Springer: Berlin, pp 229-243.

Kuebler WM, Mertens M and Pries AR (2007). A two-component simulation model to teach respiratory mechanics. *Advances in Physiology Education* **31**: 218-222.

Lacy LW (2006). *Interchanging discrete-event simulation process-interaction models using the web ontology language – OWL*. PhD thesis, University of Central Florida.

Lacy L and Gerber W (2004). Potential modeling and simulation applications of the web ontology language – OWL. In: Ingalls RG, Rossetti MD, Smith JS and Peters BA (eds). *Proceedings of the 2004 Winter Simulation Conference.* IEEE Computer Society Press: Piscataway, NJ, USA, pp 265-270.

Lawrence PF, Bell RM and Dayton MT (2006). *Essentials of General Surgery*. Lippincott Williams & Wilkins: Philadelphia, PA, USA.

Leape L, Berwick D, Clancy C, Conway J, Gluck P, Guest J, Lawrence D, Morath J, O'Leary D, O'Neill P, Pinakiewicz D and Isaac T (2009). Transforming healthcare: a safety imperative. *Quality and Safety in Health Care* **18**: 424-428.

Lee EA and Seshia SA (2011). *Introduction To Embedded Systems*. LeeSeshia.org

Liang V and Paredis C J (2003). A port ontology for automated model composition. In: Chick S, Sánchez P J, Ferrin D and Morrice D J (eds). *Proceedings of the 2003 Winter Simulation Conference*. IEEE Computer Society Press: Piscataway, NJ, USA pp, 613-622.

Lin M C and Canny J F (1991). A fast algorithm for incremental distance calculation. *Proceedings of International Conference on Robotics and Automation.* IEEE Computer Society: Washington, DC, USA, pp 1008-1014.

Loresnsen WE and Yamrom B (1989). Object-oriented computer animation. *Proceedings of the IEEE 1989 National Aerospace and Electronics Conference*. IEEE Computer Society: Washington, DC, USA, pp 588-595.

Lucas B, Abram GD, Collin NS, Epstein DA, Gresh DL and McAuliffe KP (1992). An architecture for a scientific visualization. In: Kaufman A and Nielson G (eds). *Proceedings of the 3rd Conference on Visualization*. IEEE Computer Society Press: Los Alamitos, CA, USA, pp 107-114.

Mason WT and Strike PW (2003). See one, do one, teach one--is this still how it works? A comparison of the medical and nursing professions in the teaching of practical procedures. *Medical Teacher* **25**(6): 664-666.

Matkovic K, Hauser H, Sainitzer R and Groller ME (2002). Process visualization with levels of detail. *Proceedings of the 2002 IEEE Symposium on Information Visualization.* IEEE Computer Society: Washington, DC, USA, pp 67-70.

McSharry PE, Clifford GD, Tarassenko L and Smith LA (2003). *A dynamical model for generating synthetic electrocardiogram signals.* Transactions on Biomedical Engineering **50**(3): 289-294.

Mitra P, Noy N and Jaiswal A (2005). Ontology mapping discovery with uncertainty. In: Gil Y, Motta E, Richard Benjamins VR and Musen MA (eds).*Proceedings of the 4th International Semantic Web Conference*. Springer: London, pp 537-547.

Miller J A, Baramidze GT, Sheth AP and Fishwick PA (2004). Investigating ontologies for simulation modeling. *Proceedings of the 2004 Symposium on Simulation*. IEEE Computer Society: Washington, DC, USA, pp 55-71.

Miller E (1998). An introduction to the resource description framework. *Bulletin of the American Society for Information Science and Technology* **25**(1): 15–19.

Miller A, Seila A F and Xiang X (1999). The JSIM web-based simulation environment. *Future Generation Computer System* **17**: 119-133.

Modell HI (2000). How to help students understand physiology? Emphasize general models. *Advances in Physiology Education* **23**: 101-107.

National Instruments (2008). Create advanced user interfaces with OpenGL based 3D visualization [White Paper]. http://www.ni.com/white-paper/3170/en#toc2 accessed 1 September 2012.

Noy NF, Sintek M, Decker S, Crubézy M, Fergerson RW and Musen MA (2001). Creating semantic web contents with Protégé-2000. *IEEE Intelligent Systems* **16**(2): 60-71.

OGRE (2012). OGRE – Open Source 3D Graphics Engine. http://www.ogre3d.org, accessed 1 September 2012.

Oliver R and Herrington J (2003). Exploring technology-mediated learning from a pedagogical perspective. *Interactive Learning Environments* **11**(2): 111-126.

Page EH (1994). *Simulation modeling methodology: principles and etiology of decision support*. PhD thesis, Virginia Tech.

Papert S and Harel I (1991). *Constructionism*. Ablex Publishing: Norwood, NJ.

Park M (2005). *Ontology-based customizable 3D modeling for simulation*. PhD thesis, University of Florida.

Park M and Fishwick PA (2005). Integrating dynamic and geometric model components through ontology-based inference. *Journal of Simulation* **81**(12): 195-813.

Paulheim H and Probst F (2010). Ontology-enhanced user interfaces: a survey. *International Journal on Semantic Web and Information Systems* **6**(2): 39-59.

Petridis P, Dunwell I, Panzoli D, Arnab S, Protopsaltis A, Hendrix M and de Freita S (2012). Game engines selection framework for high-fidelity serious applications. *International Journal of Interactive Worlds* **2012**: 1-19.

Phanouriou C (2000). *UIML: A device-independent user interface markup language.* PhD thesis. Virginia Polytechnic Institute and State University.

Pietriga E (2006). Semantic web data visualization with graph style sheets. In: Proceedings of the 2006 ACM Symposium on Software Visualization. ACM: New York, NY, USA, pp 177-178.

Pop A, Fritzson P and Johansson O (2004). An integrated framework for model-driven design and development using Modelica. In: Elmegaard J, Sporring K, Sørensen and EK, (eds). *Proceedings of the 45th Conference on Simulation and Modeling of the Scandinavian Simulation Society*, pp 23-24.

Prinz A, Bolz M and Findl O (2010). Advantage of three dimensional animated teaching over traditional surgical videos for teaching ophthalmic surgery: a randomized study. *Ophthalmology* **89**(11): 1495–1499.

Puerta A R (1997). A model-based interface development environment. *IEEE Software* **14** (4): 40-47.

Quarles J (2009). *The design and evaluation of a mixed reality approach to interactively blend dynamic models with corresponding physical phenomena*. PhD thesis, University of Florida.

Quarles J, Lampotang S, Fischler I and Fishwick P (2008). A Mixed reality approach for merging abstract and concrete knowledge. *Proceedings of the 2008 IEEE Virtual Reality Conference.* IEEE Computer Society: Washington, DC, USA, pp 27-34.

Ravden SJ, Graham I and Johnson GI (1989). *Evaluating usability of human-computer interfaces: A practical method*. Wiley Publishing: Chichester, England.

Rodriguez-Paz JM, Kennedy M, Salas E, Wu AW, Sexton JB, Hunt EA and Pronovost PJ (2009). Beyond “see one, do one, teach one”: toward a different training paradigm. *Quality and Safety in Health Care* **18**:63-68.

Roehler L and Cantlon D (1997). Scaffolding: a powerful tool in social constructivist classrooms. In Hogan K and Pressley M (eds). *Scaffolding student learning: Instructional approaches and issues.* Brookline: Cambridge, MA.

Rothe CF and Gertsing JM (2002). Cardiovascular interactions: an interactive tutorial and mathematical model. *Advances in Physiology Education* **26**: 98-109.

Rosse C and Mejino JLV (2007). The foundational model of anatomy ontology. In: Burger A, Davidson D and Baldock R (eds). *Anatomy Ontologies for Bioinformatics: Principles and Practice*. Springer: London.

Sá Couto CD, van Meurs WL, Goodwin JA and Andriessen P (2006). A model for educational simulation of neonatal cardiovascular pathophysiology. *Simulation in Healthcare* **1**: 4-12.

Santhanam AP, Imielinska C, Davenport P, Kupelian P and Rolland JP (2008). Modeling real-time 3-D lung deformations for medical visualization. *IEEE Transactions on Information Technology in Biomedicine* **12**(2): 257-270.

Schroeder WJ, Lorensen WE, Montanaro GD and Volpe CR (1992). VISAGE: An Object-Oriented Scientific Visualization System. In: Kaufman A and Nielson G (eds). Proceedings of the 3rd Conference on Visualization. IEEE Computer Society Press: Los Alamitos, CA, USA, pp 107-114.

Schroeder WJ, Martin KM, Lorensen WE (1996). The design and implementation of an object-oriented toolkit for 3D graphics and visualization. In: Yagel R and Nielson GM (eds). *Proceedings of the 7th conference on Visualization*. IEEE Computer Society Press: Los Alamitos, CA, USA, pp 93-100.

SEDRIS (2012). SEDRIS Technologies: The Source for Environmental Data Representation & Interchange. http://www.sedris.org, accessed 15 September 2012.

Senay H and Ignatius E (1994). A knowledge-based system for visualization design. *Computer Graphics and Applications* 1**4**(6): 36-47.

Shneiderman B and Plaisant C (2005). Designing the User Interface: Strategies for Effective Human-Computer Interaction. Pearson Education, Inc: NJ, USA.

Shim H and Fishwick P A (2008). Visualization and interaction design for ecosystems. In: Jorgensen S E and Elsevier B V (eds). *Encyclopedia of Ecology*. Elsevier: Amsterdam, The Netherlands, pp 3685-3693.

Silén C, Wirell S, Kvist J, Nylander E and Smedby Ö (2008). Advanced 3D visualization in student-centered medical education. *Medical Teacher* **30**(5): 115-124.

Silver GA, Lacy LW, and Miller JA (2006). Ontology based representations of simulation models following the process interaction world view. In: Perrone LF, Lawson B, Liu J and Wieland FP (eds). *Proceedings of the 2006 Winter Simulation Conference*. IEEE Press: Washington, DC, USA, pp 1168-1176.

Silver GA, Hassan OA, and Miller JA (2007). From domain ontologies to modeling ontologies to executable simulation models. In: Henderson SG, Biller B, Hsieh MH, Shortle J, Tew JD and Barton RR. *Proceedings of the 2007 Winter Simulation Conference*. IEEE Press: Washington, DC, USA, pp 1108-1117.

Simulink (2012). Simulink: Simulation and Model-Based Design. http://http://www.mathworks.com/products/simulink/, accessed 1 September 2012.

Sing J and Sivaswamy J (2010). Creating user interface for interactive simulations. In: Biswas G, Carr D, Chee YS and Hwang WY (eds). *IEEE 3rd International Conference on Digital Game and Intelligent Toy Enhance Learning*. IEEE Computer Society: Washington, DC, USA, pp 38-45.

Tang L, Chen C , Zou J, Lin Y, Lin D and Li J (2011). OntoPlant: An integrated virtual plant software package for different scale applications. *Proceedings of the 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services*. IEEE Computer Society: Washington, DC, USA, pp 308-314.

Thiriet P, Batier C, Rastello O, Sylvestrev E, Reeth NV, Guillot A, Collet C and Hoyek NE (2011). 3D human anatomy learning - demonstration of 3D Tools used in teaching: 3D videos, podcasts, PDF. In: Verbraeck A, Helfert M, Cordeiro J and Shishkov B (eds). *Proceedings of the 3rd International Conference on Computer Supported Education*. SciTePress, pp.408-411.

Toth E (2000). Representational scaffolding during scientific inquiry: interpretive and expressive use of inscriptions in classroom learning. In: Gleitman L R, Joshi A K (eds). *Proceedings of the twenty-second annual conference of the cognitive science society*. Erlbaum: Mahvah, NJ, USA, pp 953-958.

Tolk A (2006). What comes after the Semantic Web – PADS implications for the Dynamic Web. *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation.* IEEE Computer Society: Washington, DC, USA, pp 55-62.

Tolk A and Turnitsa C D (2007). Conceptual modeling of information exchange requirements based on ontological means. In: Henderson SG, Biller B, Hsieh MH, Shortle J, Tew JD and Barton RR. *Proceedings of the 2007 Winter Simulation Conference.* IEEE Computer Society: Washington, DC, USA, pp 1100-1107.

TouchGraph (2012). TouchGraph Navigator 2. http://www.touchgraph.com, accessed 15 September 2012.

Treinish LA, Butler DM, Senay H, Grinstein GG, Bryson ST (1992). Grand challenge problems in visualization software. In: Kaufman A and Nielson G (eds). *Proceedings of the 3rd Conference on Visualization*. IEEE Computer Society Press: Los Alamitos, CA, USA, pp 366-371.

Troy M and Moller T (2004). Human Factors in Visualization Research. *IEEE Transactions on Visualization and Computer Graphics* **10** (1): 72-84

Upson C, Faulhaber TA, Kamins D, Laidlaw D, Schlegel D, Vroom J, Gurwitz R and van Dam A (1989). The application visualization system: a computational environment for scientific visualization. *IEEE Computer Graphics and Applications* **9**(4): 30-42.

Vasconcelos V (2011). *Blender 2.5 Character Animation Cookbook*. Packet Publishing: Birmingham, UK.

Viégas FB, Wattenberg M, van Ham F, Kriss J, McKeon M (2007) Many Eyes: A Site for Visualization at Internet Scale. *IEEE Transactions on Visualization and Computer Graphics* **13** (6): 1121-1128.

von Landesberger T, Kuijper A, Schreck T, Kohlhammer J, van Wijk JJ, Fekete JD and Fellner DW (2011). Visual analysis of large graphs: state-of-the-art and future research challenges. *Computer Graphics Forum* **30**(6): 1719 – 1749.

Vygotsky LS (1978). *Mind and society: the development of higher mental processes.* Harvard University Press: Cambridge, MA.

Wang K, Zhang L, Gai C and Zuo W (2011). Illustrative visualization of segmented human cardiac anatomy based on context-preserving model. *Proceedings of the 2011 Conference on Computing in Cardiology*. IEEE Press: Washington DC, USA, pp 485-489.

Warner H R (1959). The use of an analog computer for analysis of control mechanisms in the circulation. *Proceedings of the Institute of Radio Engineers*, pp 1913-1916.

Wang W, Juttler B, Zheng D and Liu Y (2008). Computation of rotation minimizing frames. *ACM Transactions on Graphics* **27**(1): 2:1-2:18.

Weber EH (1850). Uber die Anwendung der Wellenlehre auf die Lehre vomKreislaufe des Blutes und ins besondere auf die Pulslehre. Ber Verh Kgl Sachs Ges Wiss. Math Phys Kl.

Weinberg GM (2001). *An introduction to general systems thinking.* Dorset House: New York.

Woods WA (1975). What's in a link: foundations for semantic networks. In: Bobrow D and Collins A (eds.). *Representation and Understanding: Studies in Cognitive Science*. Academic Press: New York, NY, USA.

Yamaguchi F (1988). *Curves and surfaces in computer aided geometric design*. Springer: Berlin.

Yu X and Ganz A (2011). MiRTE: Mixed reality triage and evacuation game for mass casualty information systems design, testing and training. *Proceedings of the 2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp 1899-2002.

Zeigler BP, Praehofer H and Kim TG (2000). *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press: San Diego, CA.

BIOGRAPHICAL SKETCH

Zach Ezzell was born in 1983 in Crystal River, Florida to Peter and Patricia Ezzell. Zach grew up in Crystal River, graduated from Crystal River High School in 2001 and then began his studies at the University of Florida. As an undergraduate, Zach studied digital arts and was a University Scholar. This allowed him to be introduced to the world of research. Zach continued at the University of Florida to pursue his Ph.D. under the advisement of Dr. Paul Fishwick and was awarded a four-year UF Alumni Fellowship. Zach worked on a variety of projects during his time as a graduate student, including projects funded by the Department of Defense, NASA, and the National Institutes of Health. His Ph.D. work focused on defining a better user-interface for constructing 3D visualizations based on simulation models. His work has appeared in numerous journals and conferences including the Journal of Simulation and the Winter Simulation Conference. After graduation, Zach plans to co-found a company with Dr. Fishwick to create mobile technology to support public health investigators.